Applied Numerical Mathematics 7 (1991) 287–296
North-Holland

Sum-accelerated pseudospectral methods: the Euler-accelerated sinc algorithm

John P. Boyd
Department of Atmospheric, Oceanic and Space Science and Laboratory for Scientific Computation, University of Michigan, 2455 Hayward Avenue, Ann Arbor, MI 48109, USA

Abstract

Boyd, J.P., Sum-accelerated pseudospectral methods: the Euler-accelerated sinc algorithm, Applied Numerical Mathematics 7 (1991) 287–296

Pseudospectral discretizations of differential equations are much more accurate than finite differences for the same number of grid points N. The reason is that derivatives are approximated by a weighted sum of all N values of $u(x_i)$, rather than just three as in a second-order finite difference. The price is that the $N \times N$ pseudospectral matrix is *dense* with N nonzero elements (rather than three) in each row.

Truncating the pseudospectral sums to create a *sparse* discretization fails because the derivative series are alternating and very slowly convergent. However, these series are perfect candidates for sum-acceleration methods. We show that the Euler summation can be applied to a standard pseudospectral scheme to produce an algorithm which is both *exponentially accurate* (like any other spectral method) and yet generates *sparse* matrices (like a finite difference method). For illustration, we use the sinc basis with an evenly spaced grid on $x \in [-\infty, \infty]$. However, the same techniques apply equally well to Chebyshev and Fourier polynomials.

1. Introduction

Finite difference methods and pseudospectral schemes both approximate a derivative by a weighted sum of the values of $u(x)$, i.e.,

$$\left.\frac{du}{dx}\right|_{x=x_i} \approx \Delta_0 u(x_1) + \sum_{k=1} \{\Delta_k u(x_{i+k}) + \Delta_{-k} u(x_{i-k})\}. \tag{1.1}$$

For a second-order centered finite difference, for example, the sum is truncated at $k = 1$ and the weights are $\Delta_0 = 0$ and $\Delta_{\pm 1} = \pm 1/(2h)$. For pseudospectral methods, however, the sum extends over *all* points on the grid. For the particular case of the sinc basis, the grid is evenly spaced and extends over the whole interval $x \in [-\infty, \infty]$; equation (1.1) is an *infinite* series.

The reward for using more terms is more accuracy: a pseudospectral method with N points has an error that decreases *exponentially fast* as N increases, in contrast to the $O(1/N^2)$ error of a second-order difference. The rub is that when such approximations are used to discretize a differential equation and thus convert it into a matrix problem, the pseudospectral matrix is *dense* whereas the corresponding finite difference matrix is *sparse*. An obvious question is: Can one somehow combine the good features of both spectral and finite difference methods by inventing an algorithm that is both *sparse* and *exponentially accurate*?

Our answer is to *weight* the terms in (1.1) by a standard sum-acceleration method. In this paper, we use only the classic method of Euler described by Morse and Feshbach [3], Hamming [3], Wimp [9], and Boyd and Moore [2], but a variety of other schemes are possible.

An even simpler strategy is to simply *truncate* by dropping all terms with $k > K$. In this next section, we show that truncation gives a terrible approximation to the derivative because the sum (1.1) converges very poorly. The Euler summation, however, transforms the slowly converging sum into a rapidly converging series which can be successfully truncated without serious loss of accuracy.

2. The sinc pseudospectral method

The "sinc" or "Whittaker cardinal function" is defined by

$$\operatorname{sinc}(x) \equiv \sin(\pi x)/(\pi x). \tag{2.1}$$

The interpolation grid is

$$x_i = hi, \quad i = 0, \pm 1, \pm 2, \ldots, \tag{2.2}$$

while the basis functions are shifted-and-translated copies of the sinc function:

$$C_j(x;\, h) = \operatorname{sinc}\big([x - x_j]/h\big), \quad j = 0, \pm 1, \pm 2, \ldots . \tag{2.3}$$

Since $\operatorname{sinc}(0) = 1$ while $\operatorname{sinc}(x)$ vanishes at all other integers,

$$C_j(x_i;\, h) = \delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{2.4}$$

The numerical solution is

$$u(x) \approx \sum_{j=-\infty}^{\infty} u(x_j)\, \operatorname{sinc}\left(\frac{x - x_j}{h}\right). \tag{2.5}$$

The differentiation weights are:

$$\text{first derivative} \quad \Delta_k^{(1)} = \begin{cases} 0, & k = 0, \\ \dfrac{(-1)^{k+1}}{hk}, & k \neq 0; \end{cases}$$

$$\text{second derivative} \quad \Delta_k^{(2)} = \begin{cases} \dfrac{-\pi^2}{3h^2}, & k = 0, \\ \dfrac{2(-1)^{k+1}}{h^2 k^2}, & k \neq 0. \end{cases} \tag{2.6}$$

The pseudospectral method converts a typical differential equation such as

$$\begin{aligned} &a_2(x)u_{xx} + a_1(x)u_x + a_0(x)u = f(x), \quad x \in [-\infty, \infty], \\ &u(x) \to 0 \quad \text{as } |x| \to \infty \end{aligned} \tag{2.7}$$

into the matrix equation

$$\boldsymbol{Hu} = \boldsymbol{f} \tag{2.8}$$

where the matrix elements are

$$\begin{aligned} &\boldsymbol{u}_i = u(x_i), \qquad \boldsymbol{f}_i = f(x_i), \\ &\boldsymbol{H}_{ij} = a_2(x_i)\Delta^{(2)}_{j-i} + a_1(x_i)\Delta^{(1)}_{j-i} + a_0(x_i)\delta_{ij}. \end{aligned} \tag{2.9}$$

The numerical solution is represented for all x by the series (2.5).

One can show that as long as $u(x)$ is smooth and decays exponentially for large $|x|$, the error decreases exponentially fast with the grid spacing h and the span of the (truncated) computational grid. We deliberately omit a detailed discussion of the numerics of the sinc method because this is an article about sum-acceleration, but refer the reader to [1,8].

The derivative weights in (2.6) have two striking characteristics: (i) the weights *alternate in sign* with k, and (ii) decrease *slowly* in amplitude. The alternating signs of the weights imply that, unless $u(x)$ is rapidly varying (in which case we should use a smaller grid spacing h!), the terms in the derivative sum will alternate in sign also. Thus, the partial sums of the derivative series will alternately overshoot and undershoot the true value of the derivative. The slow decrease of the amplitude of the weights—as $1/k$ for the first derivative and $1/k^2$ for the second derivative—imply that the amplitudes of these partial sum oscillations will decrease very slowly.

For example, consider the second derivative of the function $u(x) \equiv 1$; this is exactly differentiated by the sinc series for all grid spacings so that the errors are solely due to the truncation of the derivative. The exact derivative is zero so that S_n is its own error. The series is

$$\begin{aligned} u_{xx} &\approx -\tfrac{1}{3}\pi^2 u(0) + \sum_{k=1}^{\infty} 2\frac{(-1)^{k+1}}{k^2}\{u(x+k)+u(x-k)\} \quad \text{for general } u(x);\ h=1, \\ &= -\tfrac{1}{3}\pi^2 + 4\sum_{k=1}^{\infty}\frac{(-1)^{k+1}}{k^2} \quad \text{for } u(x)\equiv 1. \end{aligned} \tag{2.10}$$

The partial sums are defined (for any $u(x)$) to be the sum of all terms in the series up to and including $k = n$, i.e. (for the second derivative)

$$S_n \equiv -\tfrac{1}{3}\pi^2 u(0) + \sum_{k=1}^{n} 2\frac{(-1)^{k+1}}{k^2}\{u(x+k)+u(x-k)\}. \tag{2.11}$$

The first column of Table 1 lists the partial sums for the second derivative of $u(x) \equiv 1$. As advertised, the sums oscillate in degree n and decrease slowly. An obvious strategy for accelerating the convergence of the series is to *average* each pair of partial sums by defining a new sequence

$$T_n = \tfrac{1}{2}(S_n + S_{n-1}), \quad \text{once-averaged partial sums} \tag{2.12}$$

As shown in the second column of Table 1, these "once-averaged" partial sums converge much faster than the original sequence. However, the T_n are themselves alternating with n. This suggests defining a new sequence $U_n \equiv \frac{1}{2}(T_n + T_{n-1})$. This converges even faster, but it, too, is alternating. Repeating this process until all partial sums of a given degree has been exhausted

Table 1
Partial sums of the second derivative of $u(x) \equiv 1$

n	S_n (partial sum)	$T_n = \frac{1}{2}(S_n + S_{n-1})$	$U_n = \frac{1}{2}(T_n + T_{n-1})$	$V_n = \frac{1}{2}(U_n + U_{n-1})$
0	−3.2899	—	—	—
1	0.7101	−1.28987	—	—
2	−0.2899	0.21013	−0.539868	—
3	0.1546	−0.06765	0.071243	−0.2343126
4	−0.0954	0.02958	−0.019035	0.0261041
5	0.0646	−0.01542	0.007076	−0.0059792
6	−0.0465	0.00902	−0.003201	0.0019374
7	0.0351	−0.00572	0.001651	−0.0007752
8	−0.0274	0.00385	−0.000935	0.0003579
9	0.0220	−0.00271	0.000569	−0.0001834
10	−0.0180	0.00198	−0.000365	0.0001017
11	0.0150	−0.00149	0.000245	−0.0000600
12	−0.0127	0.00115	−0.000170	0.0000373
13	0.0109	−0.00091	0.000122	−0.0000241
14	−0.0095	0.00073	−0.000090	0.0000162
15	0.0083	−0.00059	0.000068	−0.0000112
16	−0.0073	0.00049	−0.000052	0.0000079
17	0.0065	−0.00041	0.000040	−0.0000057
18	−0.0058	0.00034	−0.000032	0.0000042
19	0.0052	−0.00029	0.000026	−0.0000032
20	−0.0048	0.00025	−0.000021	0.0000024

gives the nth degree Euler sum. In Table 1, the first nonzero number in each column is the corresponding Euler sum.

As explained in [2], each averaging accelerates the rate of convergence of an alternating series by one power of n. Thus, since the error in the second derivative sum is proportional to $1/n^2$, the error in T_n is $O(1/n^3)$, that of U_n is $O(1/n^4)$, and so on. The error in the Euler sum, however, decreases *exponentially* fast with n.

Figure 1 employs a different order of derivative (first) and a different function (proportional to $\operatorname{sech} \frac{1}{10}x$), but with similar results. The nth and $(n+1)$st partial sums bracket the true derivative, but the convergence is poor. Even S_{19} and S_{20} are crude approximations to u_x. For all derivatives and all functions, truncation—approximating the derivative by S_n for small to moderate n—is unacceptably inaccurate. It is only when the derivative sum is accelerated that one can obtain a sparse approximation which is reasonably accurate.

3. The modified Euler summation

In the Euler summation, the infinite series

$$S \equiv \sum_{k=0}^{\infty} a_k \tag{3.1}$$

is approximated by the nth order Euler sum E_n defined by either of the two equivalent series

$$E_n \equiv \sum_{k=0}^{n} \mu_{nk} S_k \equiv \sum_{k=0}^{n} w_{nk} a_k \tag{3.2}$$

where

$$\mu_{nk} \equiv \frac{n!}{2^n k!(n-k)!}, \quad k = 0, 1, \ldots, n, \tag{3.3}$$

$$w_{nk} = \sum_{j=k}^{n} \mu_{nj}. \tag{3.4}$$

The classical Euler summation has several desirable properties [9]. First, the method is "triangular", i.e., $\mu_{nk} = 0$ for $k > n$, which implies that E_n depends only on the first $(n+1)$ terms of the series. Second, all the weights are positive: "Because of their numerical stability, positive methods are the most frequently used." [9]. Third, Euler summation is "Toeplitz", which in turn implies that the acceleration is "regular". This means that the accelerated sum converges whenever the original series converges, and to the same sum. "Regularity" is exceedingly important because it guarantees that the Euler accelerated sinc method will always give the correct answer provided that the grid spacing is sufficiently small and the degree of the Euler sum is sufficiently high: the Euler–sinc algorithm is as reliable as finite differences.

Fourth, the sinc derivative of a *polynomial* is a sum of terms proportional to

$$\sigma_\alpha \equiv \sum_{k=0}^{\infty} (-1)^k (k+1)^\alpha. \tag{3.5}$$

The fact that E_n is *exact* for sums σ_α of degree $0 \leqslant \alpha < n$ implies that the nth order Euler–sinc algorithm computes the *exact* first derivative of all polynomials of degree n and the exact second

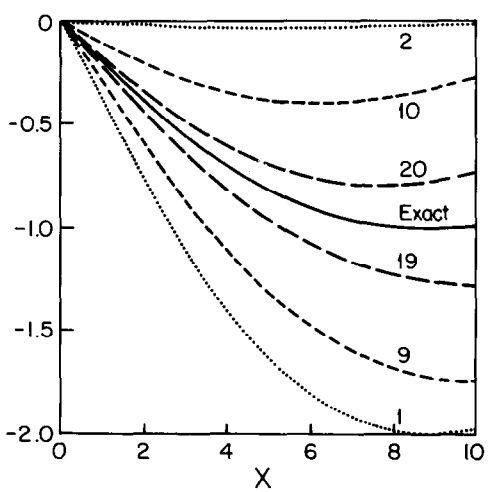

Fig. 1. Solid curve: the first derivative of $u(x) = 20\ \mathrm{sech}(x/10)$. Dotted: $k = 1$ (lower) and $k = 2$ (upper) partial sums of the sinc derivative series. Short dashes: $k = 9$ (lower) and $k = 10$ (upper) partial sums. Long dashes: $k = 19$ and $k = 20$ (unaccelerated) partial sums.

derivative of all polynomials in x of degree $(n+1)$ or less *except* for the *constant*. Table 1 illustrates that E_n computes the second derivative of $u(x) \equiv 1$ only approximately. The analytical reason for this failure of exact differentiation is that the second derivative of a constant is proportional to (3.5) for $\alpha = -2$ and the Euler sum is exact only for alternating sums of *positive* powers of k. Because of symmetry, the sinc approximations to the first derivative of all even polynomials and the second derivative of all odd polynomials are zero, so the first derivative of the constant is computed exactly: only the second derivative is approximate.

Fortunately, we can *modify* the Euler weights w_{n0} so that the second derivative of $u(x) \equiv 1$ is exact without spoiling the exactness of differentiation for the higher powers of $(x - x_i)$. The modified weight is

$$w_{n0}^{(\mathrm{mod})} \equiv 2 \sum_{k=1}^{n} \frac{(-1)^k w_{nk}}{k^2} \tag{3.6}$$

where the right-hand side of (3.6) is simply the negative of the contribution of all the other weights to the second derivative of the constant. The exact differentiability of higher powers of $(x - x_i)$ is not spoiled by (3.6) because these powers vanish at $x = x_i$, so the weight w_{n0} is irrelevant to the non-constant terms in an x-polynomial.

Table 2 illustrates what happens when the classical and modified Euler weights are applied to the same differential equation. The classical Euler method (with $w_{n0} = 1$) is clearly converging and is a huge improvement over truncation of the (unweighted) sinc derivative sum, which is also listed. However, the modified Euler summation is much more accurate.

In the remainder this paper, we shall use *only* the modified Euler method whose $k = 0$ weight is given by (3.6) for all n while the weights for $k = 1, \ldots, n$ are still defined by (3.3) and (3.4).

Table 2
A comparison of the truncated and Euler-accelerated sinc methods for various truncation orders n for a typical problem. The ODE is $u_{xx} - x^2 u = -\mathrm{e}^{-(1/2)x^2}$ with the exact solution $u = \mathrm{e}^{-(1/2)x^2}$. The grid spacing $h = 1/4$ with a total of 65 grid points

n	Truncated sinc	Classical Euler–sinc	Modified Euler–sinc
1	1.1	0.95	0.0060
2	0.79	0.90	0.0027
3	1.83	0.79	0.00022
4	0.47	0.62	0.000034
5	1.91	0.43	0.000010
6	0.22	0.26	0.00000042
7	0.29	0.14	0.00000041
8	0.089	0.069	0.000000099
9	0.083	0.033	0.0000000089
10	0.032	0.016	0.0000000079
11	0.026	0.0076	0.0000000022
12	0.012	0.0036	0.00000000040
13	0.0090	0.0017	0.00000000021
14	0.0064	0.00083	0.000000000086

The modified Euler method is exact for second derivatives of polynomials of degree $(n+1)$ and for first derivatives of degree n. (Because of symmetry which makes the appropriate sinc sum vanish independently of the summation weights, these degrees of exactness are increased by 1 for both derivatives when n is odd.) Finite difference formulas using $(2n+1)$ points are exact for polynomials of degree $2n$, which is usually higher than for the Euler sum method because exact differentiation of polynomials is the sole criterion that determines the finite difference weights. However, for $n=1$, i.e., a three-point approximation, the Euler-accelerated sinc method is *identical* with second-order finite differences for both the first and second derivatives.

4. Numerical examples: solutions to differential equations

Table 2 shows an example which is slanted in favor of the unaccelerated sinc method because (i) the exact solution, $u=\exp(-0.5x^2)$, decays very fast so that the grid can be restricted to a relatively narrow range of x and (ii) the solution and forcing are entire functions, free of singularities throughout the entire complex plane except at infinity, which implies very rapid error reduction as the grid size h is decreased. Consequently, the sinc method, with just 65 points, has an error limited by machine precision to about fourteen decimal places. Even so, the modified Euler acceleration of the sinc method is much more efficient.

If we assume that the dense matrix of the standard sinc algorithm and the banded matrix of the Euler–sinc method are solved by Gaussian elimination, then, denoting the total number of grid points by N, the matrix algebra costs are [1]

$$W_{\text{sinc}} \sim \mathrm{O}(\tfrac{2}{3}N^3) \quad \text{and} \quad W_{\text{Euler}}(n) \sim \mathrm{O}(2Nn^2), \quad n \ll N. \tag{4.1}$$

If we use E_6 so that the Euler–sinc matrix has thirteen nonzero elements in each row (versus 65 for the standard sinc algorithm), the maximum pointwise error is only 4.2×10^{-7} and yet the matrix cost is reduced by a factor of 27!

In reality, Gaussian elimination is so expensive that it is usually replaced by iterative schemes, especially for multi-dimensional problems. In this paper, we used the Richardson iteration-with-finite-difference preconditioning suggested by Orszag [6]. For the differential equation $u_{xx}+q(x)u=f(x)$, this iteration is

$$\boldsymbol{H}_{\text{fd}}\boldsymbol{u}^{j+1} = \boldsymbol{u}^j + \tau\{\boldsymbol{f} - \boldsymbol{u}^j_{xx} - \boldsymbol{q}\boldsymbol{u}^j\}, \tag{4.2}$$

where $\boldsymbol{H}_{\text{fd}}$ is the matrix of the second-order finite difference approximation to the differential operator, $\boldsymbol{u}_{xx}$ is the sinc or Euler–sinc approximation to the second derivative, and $\boldsymbol{qu}$ is the vector whose components are $q(x_i)u(x_i)$. Orszag shows the reward for solving a second-order finite difference problem for each j is that the error decreases by approximately a factor of 2.5 at each iteration when using the optimum pseudotime step of $\tau=0.6$.

If we neglect the setup costs of factoring the finite difference matrix and so on, the costs of the preconditioned Richardson iteration are

$$W_{\text{sinc}} \sim \mathrm{O}(2N^2)/\text{iteration} \quad \text{and} \quad W_{\text{Euler}}(n) \sim \mathrm{O}([7+3n]N)/\text{iteration}, \tag{4.3}$$

where additions and multiplications are both counted with equal weight. The ratio of work now grows as $\mathrm{O}(N/n)$ instead of $\mathrm{O}([N/n]^2)$, which obviously reduces the advantages of Euler acceleration. However, if we examine a second example which is more realistic than the first in

that (i) the solution has poles for finite complex x and (ii) decays as an exponential rather than Gaussian, then the Euler-accelerated algorithm is still vastly superior to the standard sinc method.

Figure 2 compares the maximum pointwise (L_∞) errors for the Euler-accelerated sinc algorithm and the truncated sinc method after a sufficient number of Richardson iterations so that the matrix-solving error is negligible. (In the "truncated sinc" method, we approximate the dense sinc matrix by a sparse matrix by setting all elements more than n columns away from the matrix diagonal equal to 0.) The problem is

$$u_{xx} - u = -2\,\mathrm{sech}^3 x, \qquad \text{Exact solution: } u(x) = \mathrm{sech}\ x. \tag{4.4}$$

We used 201 grid points with a grid spacing $h = 0.3$ in Fig. 2.

The truncated sinc algorithm is obviously terrible; the only viable way to convert the dense sinc matrix into a sparse, banded matrix is by applying a sum-acceleration method. The Euler–sinc error is the same for all E_n with $n > 17$ because all these higher-order Euler sums are limited by the 3×10^{-8} error of the standard sinc algorithm.

Remarkably, the Euler sums do not gradually asymptote to this limiting sinc accuracy, but rather equal the sinc accuracy (to within the thickness of the graph curves) for *finite n*. It is silly and wasteful to sum over all 201 grid points to approximate the derivative; the Euler–sinc method using a 37-point ($n = 18$) approximation gives the *identical* result.

The convergence rate of the Richardson matrix iteration is independent of the discretization method, so the Euler–sinc and standard sinc algorithms require the same number of iterations: the relative work is determined entirely by the cost per iteration. For this example, the E_{18}–sinc method is cheaper than the classical sinc algorithm—without loss of accuracy—by a factor of 5.4! If we are willing to tolerate some loss of accuracy from the limiting sinc accuracy, then the savings can be greatly increased: E_6 has a maximum pointwise error of less than 0.00001 but is cheaper than the preconditioned sinc iteration by a factor of 15.

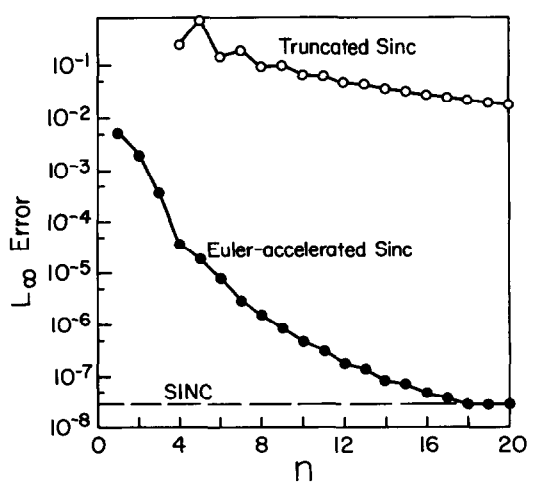

Fig. 2. The maximum pointwise error for the ordinary differential equation (4.4) as a function of n, the order of the truncation. Top (dashed): the unweighted, truncated sinc algorithm. Bottom (solid): the Euler-accelerated sinc method. The nth order approximates the derivative as a weighted sum of $(2n+1)$ points and thus is equivalent in complexity to a $(2n)$th order finite difference method.

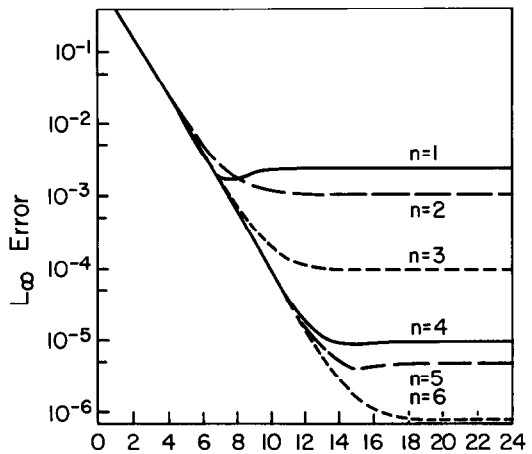

Fig. 3. The maximum pointwise error for problem (4.4) is graphed as a function of the Richardson iteration number j for the Euler–sinc method of orders 1 (top) to 6 (bottom). The initial guess is $u^0 = 0$. The grid employed 181 points with $h = 0.2$ (both different from Fig. 2).

It is possible to reduce the work of the Euler–sinc method even more by using variable order during the matrix iteration. Figure 3 shows how the error decreases with iteration number j for $n = 1, \ldots, 6$. For the first few iterations, all six curves are *superimposed*. There is no advantage in using a high order approximation for $j = 1, \ldots, 4$ because the major source of error lies not in the approximation of derivatives, but rather in the solution of the matrix problem, in the crudeness of the first guess. It is most efficient to begin with an $n = 1$ (3-point) Euler approximation for the first few iterates, then switch to $n = 2$, then to $n = 3$ and so on.

5. Conclusions and open problems

The slowly convergent derivative sums of a standard pseudospectral method can be accelerated by the modified Euler summation. When n is sufficiently large—but still small in comparison to the total number of grid points N—the Euler–sinc algorithm reaches the limit of the pseudospectral error. Thus, one can truncate the sum *without loss of accuracy* and save an *order of magnitude* in computing time.

For simplicity, we have discussed the sinc pseudospectral algorithm on an unbounded interval. This eliminates the complexities of boundary effects, unevenly spaced grids, and complicated interpolation functions. However, as explained in [1], there is a very close relationship between the sinc method and the much more familiar Fourier and Chebyshev polynomial algorithms. We are presently extending this sum-acceleration strategy to these other basis sets.

A naive conclusion would be that unaccelerated pseudospectral methods are finished: only an idiot would solve a large, dense, poorly-conditioned matrix problem when sum-acceleration methods allow it to be replaced by a sparse, better-conditioned matrix with savings of an order of magnitude at no loss in accuracy. For sinc applications such as in [4,8], this is almost certainly true. There is no Fast Fourier Transform (FFT) for Whittaker cardinal functions, so minimizing bandwidth via sum-acceleration is the only strategy for escaping manipulations of large dense matrices. The unaccelerated sinc method is dead.

For Chebyshev and Fourier methods, however, the FFT can be used to *bypass* the direct calculation of the collocation matrix so the the residual on the right-hand side of the Richardson iteration can be evaluated in $O(N \log N)$ operations (instead of $O(N^2)$ as for unaccelerated sinc methods). The correct conclusion is that *naive*, non-FFT pseudospectral algorithms are hopelessly non-competitive.

If acceleration strategies are as efficient for Chebyshev and Fourier series as for sinc sums, then we can safely conclude that acceleration will be at least *competitive* with FFT-based tactics. One open problem is to perform direct comparisons between these strategies to verify this expectation.

An intriguing problem, which is the theme of work already in progress, is the relationship between sum-accelerated pseudospectral methods and finite difference schemes. As noted above, the $n = 1$ modified Euler–sinc algorithm is *identical* with second-order finite differences. For larger n, the methods are different. However, finite differences of all n can be obtained from the sinc pseudospectral method via a sum-acceleration scheme that is a regular, Toeplitz triangle. Low-order finite differences are more accurate than the Euler–sinc method of the same order for differentiating functions whose Fourier transforms are strongly peaked about zero wavenumber. However, the sinc–Euler method is more uniformly accurate in wavenumber and seems to always be superior for *large* n.

Another unresolved issue is the choice of sum-acceleration methods. The Euler acceleration is the Old Reliable: it is effective on almost all alternating sums. However, as pointed out by Wimp [9], special purpose accelerations are almost always more effective for restricted classes of series than general purpose algorithms like Euler's. It must be emphasized that the Euler summation is almost certainly not the last word on accelerated pseudospectral methods, but merely the first word.

Acknowledgement

This work was supported by the National Science Foundation through grants DMS8716766 and OCE8800123.

References

[1] J.P. Boyd, *Chebyshev & Fourier Spectral Methods*, Lecture Notes in Engineering 49 (Springer, New York, 1989).
[2] J.P. Boyd and D.W. Moore, Summability methods for Hermite functions, *Dyn. Atmos. Oceans* 10 (1986) 51–62.
[3] R.W. Hamming, *Numerical Methods for Scientists and Engineers* (Dover, New York, 2d ed., 1986).
[4] J.R. Lund and B.B. Riley, A sinc-collocation method for the computation of the eigenvalues of the radial Schroedinger equation, *IMA J. Numer. Anal.* 4 (1984) 83–98.
[5] P.M. Morse and H. Feshbach, *Methods of Theoretical Physics*, Vol. 1 (Wiley, New York, 1953).
[6] S.A. Orszag, Spectral methods for problems in complex geometries, *J. Comput. Phys.* 37 (1980) 70–92.
[7] D.A. Smith and W.F. Ford, Acceleration of linear and logarithmic convergence, *SIAM J. Numer. Anal.* 16 (1979) 223–240.
[8] F. Stenger, Numerical methods based on Whittaker cardinal, or sinc functions, *SIAM Rev.* 23 (1981) 165–224.
[9] J. Wimp, *Sequence Transformations and Their Applications* (Academic Press, New York, 1981).