EPSRC RESEARCH PROPOSAL
Scientific Computing and the Chebfun System
Case for Support

Prof. Lloyd N. Trefethen FRS
Oxford University Computing Laboratory

September 20, 2006

Summary

MATLAB has become the primary mathematical programming language for hundreds of thousands of engineers and scientists around the world. This proposal seeks funding to pursue an exciting opportunity to extend MATLAB in a fundamental way. This project is believed to be unique, with no competitors anywhere so far.

Computing systems such as Maple and Mathematica can solve some mathematical problems symbolically, delivering exact answers in special cases. However, for scientific computing and computational science, the fraction of problems that have exact solutions is very small. That is why we compute numerically and will continue to do so. Ordinarily, numerical computation is carried out with numbers. The chebfun system realizes a new vision: numerical computation with functions.

A letter of support is attached from Cleve Moler, inventor of MATLAB and Chairman and Chief Scientist of The MathWorks, Inc.

1 Previous research track record

1.1 Summary of proposer's recent relevant work

The idea for the chebfun project originated in 2001 and was developed during 2002–2006 by the proposer's DPhil student Zachary Battles (see §2.1). Mr. Battles has recently completed his work and defended his thesis [2]. An urgent opportunity is open to extend the chebfun system and the ideas underlying it closer to scientific users and the marketplace, and this proposal seeks funding to support a Research Assistant and a DPhil student to make this happen.

The crucial items of recent work relevant to this proposal are the chebfun system itself, which is freely available online [2], and the associated publication in the *SIAM Journal on Scientific Computing* [3]. More broadly, this research project stands at the interface of computer science and numerical analysis and blends the following themes:

(1) Object-oriented software design in MATLAB

(2) Numerical algorithms based on advanced methods of approximation theory

(3) Spectral methods for solution of differential equations

(4) Numerical linear algebra

(5) The linear algebra / functional analysis interface

(6) Fundamental questions of the nature and the future of numerical computation

The proposer is known for his contributions in all of these areas. Concerning (1), he was the supervisor of a former student, Prof. Toby Driscoll of the University of Delaware, who wrote the now-standard software system for Schwarz–Christoffel conformal mapping, the *SC Toolbox*, one of the first widely used examples of an object-oriented system in MATLAB [7]. Concerning (2), he has made many contributions since the early 1980s, most recently publishing an article with Berrut in *SIAM Review* that has become the standard reference on a numerical technique which is central for this project, *barycentric interpolation* [4]. A related mathematical research project just completed has shown that Gauss quadrature does not have the advantage widely attributed to it in comparison with the less famous but far simpler Clenshaw–Curtis quadrature [17]. In each of areas (3) and (4) the proposer is the author of bestselling textbooks [11, 18] as well as of numerous research articles. Concerning (5), he is known for his leadership of the field of pseudospectra of matrices and operators, a field which makes many links between discrete and continuous linear problems involving nonselfadjoint operators; a culmination of his contributions in this area is the 2005 research monograph *Spectra and Pseudospectra: the Behavior of Nonnormal Matrices and Operators* [19]. Finally, the fundamental questions of (6), some of which are discussed in §2.2.3, §2.2.4 and §2.3, are a subject on which the proposer has become known through his essays and lectures, as well as the recent SIAM Hundred-Dollar, Hundred-Digit Challenge [5, 10, 12, 13, 14, 15, 16].

1.2 Contribution to UK competitiveness

For forty years the UK has held a special position in the development of software for numerical computations, boasting three organisations with global impact: the National Physical Laboratory, the Harwell Atomic Energy Research Establishment, and the Numerical Algorithms Group (NAG). The first of these is no longer distributing a software library, but the second continues to thrive through the HSL library group now at the Rutherford Appleton Laboratory, and NAG continues to sell its famous libraries. It is hoped that the present project may grow into a system with long-term impact and thereby contribute to this valuable UK tradition.

1.3 Expertise available at host organisation

The proposer is the head of Oxford's Numerical Analysis Group, which since the 1960s has been a leading research group in this field in the UK and worldwide. The group includes seven academics, various postdoctoral researchers, and 15–20 DPhil students with wide-ranging interests in numerical linear algebra, numerical solution of partial differential equations, and optimization. All of these will be involved to varying degrees in discussions related to the chebfun project. The NA Group is situated in the Oxford Computing Laboratory, which has strong expertise in programming languages and software engineering.[1]

1.4 Previous collaborations with overseas scientists/engineers

The proposer has had extensive collaborations going back many years with colleagues in the United Kingdom, Europe, the United States and South Africa (a total of 39 coauthors so far).

[1] In 2009 or 2010 the NA Group will be transferred to Oxford's Mathematical Institute. The proposed research lies at the interface of computer science and mathematics and will be of interest to both the Computing Lab and the Maths Institute.

2 Proposed Research and Context

2.1 The chebfun system

The chebfun system computes numerically not just with numbers but with real or complex functions defined (so far) on the interval $[-1,1]$. The system is great fun to use, and it will be easier to appreciate what is being proposed here if the reviewer can download it from `http://www.comlab.ox.ac.uk/chebfun` and give it a try. For a more detailed introduction see the paper [3], also available at that web site.

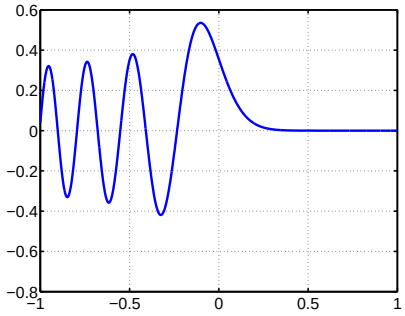
Conventional MATLAB is built on the most advanced algorithms for vectors and matrices; this is a source of its power, since so much of scientific computing in the end comes down to numerical linear algebra. The idea of chebfun computation is to create a MATLAB class that behaves syntactically like the usual MATLAB vectors. The difference is that with chebfuns, the usual vector commands in MATLAB are overloaded with analogues for continuous functions.

For example, the command

```
>> f = chebfun('real(airy(10*x))');
```

calls the chebfun constructor to produce a chebfun object `f` that will behave, up to the usual IEEE double machine precision of about 16 digits, like the Airy function $\mathrm{Ai}(10x)$. We can plot `f` with

```
>> plot(f),
```



giving the image on the right; this is an overloading of MATLAB's usual `plot` command, which draws dots connected by line segments. We can also compute various numerical quantities such as these:

```
>> f(1/3)
ans = 0.00355602812320
```
$f(\frac{1}{3})$

```
>> max(f)
ans = 0.53565665601570
```
$\max_x f(x)$

```
>> sum(f)
ans = 0.10990317364334
```
$\int_{-1}^{1} f(x)\,\mathrm{d}x$

```
>> norm(f)
ans = 0.31760629888435
```
$\left(\int_{-1}^{1} |f(x)|^2\,\mathrm{d}x\right)^{1/2}$

```
>> norm(f,1)
ans = 0.30618960369037
```
$\int_{-1}^{1} |f(x)|\,\mathrm{d}x$

All these numbers are correct to the full precision displayed, except occasionally in the last digit. About ninety MATLAB operations have been overloaded for chebfuns; other examples include `cumprod`, `cumsum`, `diff`, `imag`, `mean`, `min`, `pinv`, `qr`, `real`, `std`, `sum`, and `svd`. Thus for example the usual MATLAB command `sum(v)` adds up the entries of a vector `v`; the overloaded `sum` command for chebfuns gives the definite integral from -1 to 1, as shown above. The `introots` command returns the zeros of a chebfun in $[-1,1]$:

```
>> introots(f)
ans =
  -0.90226508533410
  -0.79441335871209
  -0.67867080900717
  -0.55205598280955
  -0.40879494441310
  -0.23381074104598
```

These numbers are the smallest six roots of $\mathrm{Ai}(10x)$, as can be verified from p. 478 of [1]. For the derivative we use `diff`, an overloading of MATLAB's discrete difference operator for vectors, and thus for example we can find the maxima and minima of $\mathrm{Ai}(10x)$ as follows:

```
>> x = introots(diff(f)); disp([x f(x)])

  -0.95354490524336     0.32102228819471
  -0.84884867340197    -0.33047622914798
  -0.73721772550478     0.34230124441162
  -0.61633073556395    -0.35790794371229
  -0.48200992111787     0.38040646862815
  -0.32481975821798    -0.41901547803256
  -0.10187929716475     0.53565665601570
```

The indefinite integral from -1, another chebfun, is returned by the command `cumsum`. The following test gives reassuring evidence that differentiation and integration are inverses in the chebfun system, up to the usual IEEE precision of about 16 digits:

```
>> norm(f-diff(cumsum(f)))
ans = 1.017709325872272e-15
```

Each of the computations above required less than 0.04 secs. on a workstation.

How is all this done? The mathematical basis of the chebfun system is a pair of closely related ideas:

- *Polynomial interpolation in Chebyshev points implemented by Salzer's barycentric formula* [4, 8]
- *Chebyshev expansions implemented by the Fast Fourier Transform* [9, 11]

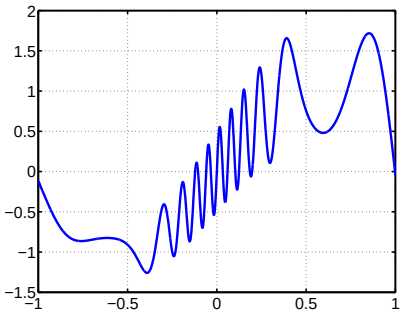
These are combined together with state-of-the-art numerical algorithms for integration, zerofinding and other operations. In particular, two features that make the system fast and accurate are

- *Adaptive determination of the number of points needed to represent each function to about* 16 *digits*
- *Zerofinding via eigenvalues of Chebyshev companion matrices with divide-and-conquer recursion*

The result is a system with, we have found, rather astonishing capabilities. For example, the Airy function f above may seem complicated, but its chebfun representation to 16-digit precision is a polynomial of degree just 59. The more irregular function

```
>> f = chebfun('sin(3*x)+.5*exp(x).*sin(100*x./(1+3.*x.^2))')
```

plotted on the right still only needs a polynomial of degree 237. Even a function represented by a polynomial of degree 100,000 is integrated by `sum` to full accuracy on our workstation in 0.2 secs. Some of the mathematics underpinning the extraordinary speed, accuracy, and stability of high order polynomial interpolants is old, and some of it is new, but it is safe to say that until the arrival of the chebfun system few people were aware of these possibilities. The investigations proposed here, despite their classical foundations, will explore poorly charted territory even from an algorithmic point of view. From a software point of view they are completely new.

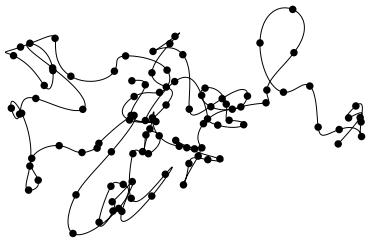


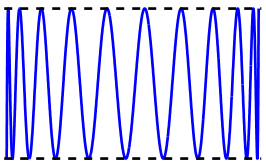
Like MATLAB itself, the chebfun system is not just an interactive calculator and problem-solving environment but most importantly a programming language. Suppose, for example, we want to determine the location of the third maximum of $\mathrm{Ai}(x)$ in $(-\infty, 0]$. Here is a program that will solve this problem.

```
% program third.m
  f = chebfun('real(airy(10*x))');          % Airy function of scaled argument
  fd = diff(f);                             % derivative
  fdd = diff(f,2);                          % second derivative
  extrema = introots(fd);                   % extrema
  indices_of_maxima = find(fdd(extrema)<0); % indices of maxima
  maxima = extrema(indices_of_maxima);      % maxima
  maxima = sort(maxima);                    % make sure they are sorted
  third_maximum = maxima(end-2);            % pick out the 3rd
  third_maximum = third_maximum*10          % rescale to standard Airy function
```

Running it gives an answer in 0.04 secs. on our workstation that is correct in all but the final digit (cf. a_5' in Table 10.13 of [1]):

```
>> third
third_maximum = -7.37217725504781
```





The first plot on the right shows a smooth interpolant through 100 data points in the x-y plane computed in 0.08 secs. by a two-line chebfun program. The second shows the result of a more advanced chebfun computation. Here, by the Carathéodory–Fejér method, the best minimax polynomial approximation $p_{20}(x)$ of $f(x) = \exp(\exp(x))$ on $[-1, 1]$ has been computed; the plot shows the error curve $(f - p_{20})(x)$, with amplitude 7.6×10^{-11}. This computation took 0.4 secs. and the program is 20 lines long.

Of course, the above brief introduction to the chebfun system gives only a glimpse, with hardly any discussion of its algorithms or program design, not to mention crucial issues of complexity and accuracy. More details can be found in [2] and [3]; see also §2.2.4.

2.2 Programme, methodology and management

To date, the chebfun system has been exhibited in a few classrooms but not yet applied to substantial problems of scientific computing. Our plan is to move in that direction. Three of the problems we will tackle can be addressed within the existing chebfun system (Projects 1, 3, 4), and one will require a significant extension (Project 2). In every case, success will require overcoming numerical analysis research challenges with wider implications.

The investigations proposed here will be carried out by a team consisting of the proposer, a DPhil student and a postdoctoral Research Assistant. These three will have offices on the same hallway and will talk daily, with scheduled meetings each week. The DPhil student will take the lead on Projects 1 and 2 (under close supervision) and the Research Assistant on Projects 3 and 4, as indicated in the Diagrammatic Workplan. The DPhil student will own the official version of the code. Longer term code management and distribution issues can be optimized with the help of advice from Oxford's Open Source Software Advisory Service (OSS Watch).

2.2.1 Project 1: Time-dependent PDE (partial differential equations)

[18 months, with the DPhil student taking the lead]

One of the fundamental problems of scientific computing is the solution of time-dependent PDEs. Linear examples include the wave equation of acoustics and the Schrödinger equation of quantum mechanics. Nonlinear examples include the KdV equation of nonlinear optics, the Hodgkin–Huxley equations of neural transmission and the Gray–Scott reaction-diffusion system of chemical and biological pattern formation. One of the powerful techniques for such computations is *spectral methods*, based on global Fourier and Chebyshev expansions of the solution with respect to the space variable [11]. The chebfun system has a spectral flavour, since it is implemented by Chebyshev expansions. The crucial difference is that the number of grid points is adaptively determined to achieve close to 16-digit precision rather than being fixed in advance. The question is, how can one perform numerical time-stepping when the data to be stepped are continuous functions of x rather than grid values?

This may sound straightforward, but it is not: the obvious methods fail due to wiggly chebfuns appearing with exploding amplitudes. Such instabilities are partly classical, going back to von Neumann and others in the 1950s, but the essence of the matter here is different because of the continuous treatment of the x variable. The proposer is an expert at investigating numerical instabilities and will be closely involved with the DPhil student in carrying out the necessary experiments and analysis to identify the nature of the problem. Once it is found, we will apply the chebfun system to a wide range of PDEs. Just as the chebfun system is the most powerful environment to date for manipulating smooth functions on an interval, it is realistic to hope that when this instability problem is solved we may have a software tool for solving PDEs in one space dimension which will take its place as the best "automatic" code for such problems. We believe that success on this problem alone, even if no other parts of this research plan were carried out, would constitute a significant contribution with long-term impact in scientific computing.

2.2.2 Project 2: Subintervals and transformations: toward a general-use package

[18 months, with the DPhil student taking the lead]

For smooth functions on $[-1,1]$, the chebfun system is amazingly effective. However, even if we restrict attention to one dimension, not all problems are posed on this interval. We have already seen this with the example on p. 3 of the Airy function, where we had to introduce a scale factor of 10 to get the desired results. Another serious matter is the assumption of smoothness. The Chebyshev interpolants on which the chebfun system is built converge exponentially for analytic functions, but if there is a singularity in a derivative, this property is lost. For example, the function $|x|$ cannot be represented to full accuracy in the chebfun system since that would require on the order of 10^{16} points. (In practice the system stops at 2^{16} points by default and issues a warning message.)

For applications it is important to extend the chebfun system to work around these limitations. One can devise various ways to do this on an ad hoc basis, for example involving breakpoints between concatenated intervals with arbitrary endpoints. How to do this in a manner that preserves the elegance, speed, and reliability of the chebfun system, however, is a challenging problem. We shall explore various possibilities and develop the one that best handles functions with a finite number of singularities or near-singularities. (Wavelets is one intriguing idea to be considered, but we expect piecewise polynomials to be a more likely choice.) The performance of this extension will then be tested and honed on a variety of problems with a special focus on the DPhil student's topic of solution of time-dependent PDEs.

2.2.3 Project 3: Krylov subspace iterations for operators

[18 months, with the RA taking the lead]

One can easily write a MATLAB program that takes one chebfun as input and produces another as output, representing a linear or nonlinear operator. We have done this for many examples; it is an elegant process in the chebfun system.

The question is, how can one use such formulations to solve problems involving such operators? Scientific computing is full of problems of the form $Lu = f$, where L is a linear differential or integral operator, and their nonlinear analogues. The chebfun system seems a natural framework for solving such problems to high precision with the aid of familiar matrix iterations such as conjugate gradients or GMRES (or Lanczos or Arnoldi, for related eigenvalue problems). Amazingly, though the idea of applying matrix iterations to operators goes back to the 1950s, it seems there has never been a numerical implementation before the chebfun system. The experiments reported in [2] are successful for integral operators. For differential operators, there is a problem: the spectrum of the operator is unbounded, so Krylov subspace iterations fail. In ordinary scientific computing one gets around this problem by finite matrices and preconditioning. We will aim to find a solution for chebfuns that is both natural mathematically and powerful in practice, making use where appropriate of continuous analogues of preconditioning. Progress here will shed light on an old issue in scientific computing: the relationship between the "discretize then solve" and "solve then discretize" paradigms. In the chebfun system, discretization and solution go hand in hand.

2.2.4 Project 4: Accuracy analysis

[18 months, with the RA taking the lead]

It was mentioned at the beginning that although computer systems such as Maple and Mathematica can solve some problems exactly in rational arithmetic, the fraction of problems that have exact solutions is small. The basis of practical numerical computations is floating-point arithmetic, which is usually viewed as an engineering compromise but can also be regarded as an algorithmic idea, as follows. In rational arithmetic, expressions grow exponentially in length as the computation proceeds. Thus it would be quite impossible to solve a PDE, say, by computing in rational arithmetic to the end and then "evaluating the result to 16 digits". The numerators and denominators would have more digits than there are electrons in the universe! Floating-point arithmetic prunes this combinatorial explosion by rounding to 16 digits at every step rather than once at the end. Each floating point operation delivers the exactly correct result, modulo a perturbation of relative order 10^{-16}. Upon this idea has been built the vast edifice of scientific computing.

One level up from numbers to functions, it is the same idea that drives the chebfun system: *at each step, to obtain the exact result modulo a small perturbation.* For example, in the chebfun system $\sin(x)$ and $\cos(x)$ are represented by polynomials of degrees 13 and 14, respectively, but $\sin(x)\cos(x)$ is represented by a polynomial of degree 17, not 27. Thus like floating-point arithmetic but in a richer context, chebfun arithmetic can be viewed as an algorithmic idea to tame a combinatorial explosion, and in this respect the

chebfun system is fundamentally different from systems like Maple and Mathematica and potentially far more powerful for some applications.

The fourth aim of this research proposal will be to make this vision precise. Our goal will be to establish theorems that show that the chebfun system is guaranteed to perform with a certain accuracy in this step-by-step sense. (Of course, the search for such theorems may suggest improvements in the system.) Thus we aim to lay the foundations of a rounding error analysis for functions rather than numbers. Such work appears to be entirely new.

2.2.5 Other topics

We have many plans in mind in addition to those identified above, which will be pursued if time permits. Here are three of them.

Continuous analogues of Householder and LU factorizations. The chebfun system includes algorithms for computing continuous or "quasimatrix" analogues of certain matrix factorizations and related objects such as the QR factorization, the SVD (singular value decomposition) and the pseudoinverse. (Such ideas have never been implemented before computationally.) These factorizations potentially have applications to many problems, beginning with least-squares fitting. It would be desirable to pursue some of these applications and to investigate further factorizations whose continuous analogues pose more challenges. One problem is to establish a continuous analogue of the Householder (as opposed to Gram–Schmidt) method for computing QR factorizations and to investigate its advantageous stability properties, if any. Another is to investigate the continuous analogue of LU factorization, which we expect will involve "peeling off" finite-rank pieces from an operator of infinite rank. All of this would be novel work both conceptually and computationally and might lead in unexpected directions. It is possible that the continuous LU factorization problem may be relevant to the instabilities of §2.2.1.

Global optimization. The chebfun system finds all the zeros or extrema of a function in $[-1, 1]$ to high precision in a fraction of a second, even if there are many of these as in the example plotted on p. 4. What makes this possible is the numerically stable global Chebyshev representation coupled with fast zerofinding algorithms developed by Battles in his thesis [2], which are based upon methods investigated by Boyd (see [6] and other recent papers). This global aspect of chebfun computations is a remarkable capability, unusual in the field of numerical optimization (see, e.g., Chapter 4 of [5]). It would certainly be fruitful to investigate scientific applications of this aspect of the chebfun system.

Extension to two or three space dimensions. Much of scientific computing concerns problems in two and three dimensions, and if the chebfun system is to have the widest impact, it will need to be extended eventually to a multidimensional setting. Nevertheless we do not make this one of the initial goals of the present research project, for we consider that it is important first to make the one-dimensional foundations as strong as possible. One may note that analogously, MATLAB developed for twelve years before sparse matrix capabilities were added, which opened the door to all kinds of multidimensional PDE computations.

2.3 Relevance to beneficiaries, and a historical remark

We regard the potential beneficiaries of the chebfun project as all those engaged in mathematical computation or scientific computing. This vision is long-term. We expect there will be immediate impact, with users able to solve problems easily that might otherwise be hard. At the same time the larger purpose is to introduce new ideas and software into the field into numerical computation that may develop in fruitful directions.

The history of MATLAB itself is noteworthy. Cleve Moler invented MATLAB in the late 1970s as an interface to the LINPACK and EISPACK libraries, and in those days most people assumed that its use was mainly educational. But if that was true then, it is certainly not true now. Moler and Little founded the MathWorks, Inc.; the system expanded in all kinds of directions while also being made ever faster; and now MATLAB is a development tool relied upon around the world by banks, automobile manufacturers and engineers and scientists in all fields. A crucial condition for MATLAB's success was that it is built on a core idea: fast access to the best algorithms for vectors and matrices.

The chebfun system too is built on a core idea: fast access to the best algorithms for manipulating smooth functions. If further development of this system is supported by the EPSRC, perhaps its future too may hold a trajectory from educational to practical.

2.4 Dissemination and exploitation

The results of this project will be disseminated through talks at universities and conferences, journal publications (at least six are planned: see the Diagrammatic Workplan) and the web. The proposer has an exceptional record of communication of research results, with numerous national and international speaking invitations every year, and this topic has an especially appealing educational and interactive side.

Version 1 of the chebfun system is already available online. Updates and related items will be made available on the web as they are developed, and in particular, one of our plans is to introduce and publicize an online collection of chebfun templates for examples and applications that we intend to build up as a significant resource. We emphasize that although there is every possibility that before long the chebfun project may develop a commercial side, the research funded by this grant will be in the public domain.

Concerning written publications related to the chebfun system, we have two particular plans in addition to more standard research papers. The first will be an expository article for the Education section of *SIAM Review* concerning the use of the chebfun system to illustrate fundamental results of approximation theory. For a starting point to illustrate what this article may be like, see the examples on p. 1751 of [3]. The second will be a broader expository article targeted at the *American Mathematical Monthly*. The proposer has experience publishing with both of these journals.

References

[1] M. Abramowitz and I. A. Stegun, eds., *Handbook of Mathematical Functions*, Dover, 1964.

[2] Z. Battles, *Numerical Linear Algebra for Continuous Functions*, DPhil thesis, Oxford University, 2006; and chebfun software available at `http://www.comlab.ox.ac.uk/chebfun`.

[3] Z. Battles and L. N. Trefethen, An extension of MATLAB to continuous functions and operators, *SIAM J. Sci. Comp.* 25 (2004), 1743–1770, available at `http://www.comlab.ox.ac.uk/chebfun`.

[4] J.-P. Berrut and L. N. Trefethen, Barycentric Lagrange interpolation, *SIAM Review* 46 (2004), 501–517.

[5] F. Bornemann, D. Laurie, S. Wagon, and J. Waldvogel, *The SIAM* 100*-Digit Challenge*, SIAM, 2004; appeared also in German translation as *Vom Lösen numerischer Probleme: Ein Streifzug entlang der "SIAM* 10×10 *Digit Challenge"*, Springer, 2006.

[6] J. P. Boyd, Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding, *SIAM J. Numer. Anal.* 40 (2002), 1666–1682.

[7] T. A. Driscoll, Schwarz–Christoffel Toolbox for MATLAB, `http://www.math.udel.edu/~driscoll/software/SC/index.html`.

[8] N. J. Higham, The numerical stability of barycentric Lagrange interpolation, *IMA J. Numer. Anal.* (2004) 24, 547–556.

[9] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*, Chapman & Hall, 2003.

[10] L. N. Trefethen, The definition of numerical analysis, *SIAM News* v. 25, no. 6, November 1992; also *IMA Bulletin* v. 29, March/April 1993.

[11] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, 2000.

[12] L. N. Trefethen, Predictions for scientific computing 50 years from now, *Mathematics Today*, 2000. (This essay won the Catherine Richards Prize of the Inst. for Maths. and Applics., 2000.)

[13] L. N. Trefethen, A hundred-dollar, hundred-digit challenge, *SIAM News*, Jan./Feb. 2002, with follow-up in *SIAM News*, July/Aug. 2002.

[14] L. N. Trefethen, Who invented the great numerical algorithms?, historical talk given in a number of venues during 2004–2006.

[15] L. N. Trefethen, Ten digit algorithms, Report NA-05/13, Numerical Analysis Group, Oxford U. Computing Lab., 2005. (Text of A.R. Mitchell Lecture at the 2005 Dundee Biennial Conference on Numerical analysis.)

[16] L. N. Trefethen, Numerical analysis, chapter to appear in W. T. Gowers, ed., *Princeton Companion to Mathematics,* Princeton U. Press.

[17] L. N. Trefethen, Is Gauss quadrature better than Clenshaw–Curtis?, *SIAM Review,* submitted.

[18] L. N. Trefethen and D. Bau, III, *Numerical Linear Algebra*, SIAM, 1997. (SIAM's bestselling book of the past decade.)

[19] L. N. Trefethen and M. Embree, *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*, Princeton U. Press, 2005. (Winner of Honorable Mention, 2005 Awards for Excellence in Professional/Scholarly Publishing in Mathematics and Statistics, Association of American Publishers.)