Predictions for Scientific Computing Fifty Years From Now

Lloyd N. Trefethen
Oxford University Computing Laboratory
LNT@comlab.ox.ac.uk

This essay is adapted from a talk given June 17, 1998 at the conference "Numerical Analysis and Computers—50 Years of Progress" held at the University of Manchester, England in commemoration of the 50th anniversary of the Mark 1 computer.

Fifty years is a long, long time in any technological field. In our own field of numerical analysis or scientific computing, think back to 1948. Around the world, numerical problems in 1948 were solved on paper, or with mechanical calculators that had little in common with today's computers. Some of the algorithms we use today were already in existence then, but on the whole, the last fifty years have changed numerical computing beyond recognition. The next fifty will do it again.

Scientific computing is a part of computing, and the changes ahead in computing are so dramatic, and so tempting to predict, that it has been hard to confine myself to predictions specifically numerical. I have done my best, but occasionally the temptation has proved irresistible.

My remarks will consist of twelve predictions. In putting these together, I strived to make statements that seemed both interesting and likely to be true. I did not aim for them to orbit around a unifying theme; but that is nevertheless what happened, as I shall discuss at the end.

1. We may not be here.

In the 20th century, everything technological seems to be changing exponentially. Transportation, communication, medicine, materials—in every area, one generation looks unlike the last and laughably unlike the one before. This raises a problem. Exponentials do not go on for ever; something happens to them. It may turn out, for example, that the exponential was only the first half of an S-curve. Now in my opinion, many of the exponentials we are sitting on have not yet started turning into S's. Here at the beginning of the third millennium, biology is just beginning its great explosion, and although electronics got a head start of a few decades, it is hardly slowing down yet. In a decade or two the slowdown may be in sight, or it may not; but in any case, technology here in 1998 is booming.

The presence of exponentials all around us overshadows any attempt to predict the future. I want to dwell for a moment on just one of the shadows, one that has nothing specifically to do with computing. In my opinion, our position on an exponential trajectory is evidence that technological civilizations do not last very long. I do not claim that our civilization must end within fifty years, or five hundred, but I do believe there is reason to doubt that it can survive for, say, a million years. I cannot think about the future without this view pressing upon me.

My reasoning has nothing to do with any particular cataclysm that may befall us, such as environmental poisoning or exhaustion of resources or asteroid impact or biological or nuclear war. The argument is more abstract, and it goes like this. The industrial explosion on earth began just two or three hundred years ago. Now if technological civilizations can last a million years, how do you explain the extraordinary coincidence that you were born in the first few generations of this one? — in the very first century of radio, television, light bulbs, telephones, phonographs, lasers, refrigerators, automobiles, airplanes, spacecraft, computers, nuclear power, nuclear weapons, plastics, antibiotics, and genetic engineering?

I believe that the explanation of our special position in history may be that it is not so special after all, because history tends not to last very long.

This is an odd kind of argument, and many people don't like it. They say: one must be born somewhere, and early in history is no more special than at any other point. *Someone* must be born early; why not we? But I do not think the Copernican Principle—for that is what this argument has been called by J. R. Gott of Princeton University—can be brushed aside so easily. In fact, we have long been comfortable with similar reasoning in the field of statistical mechanics. It is accepted that the air you are breathing right now behaves like a gas solely for reasons of probability. Other behaviors are possible, including the proverbial disappearance of all the molecules into the corners of the room, but you never see them because they are vastly less likely.

There is a second line of evidence, sometimes known as Fermi's paradox, that also suggests that technological civilizations are short-lived. Most of us agree that even if the first microbes came to earth from space, the development of advanced terrestrial life has occurred locally. The human race is not an outpost of a galactic society; it is a domestic product. How can we explain this if technological civilizations last millions of years? An ages-old technological civilization will expand across its galaxy, simply because it can. (Don't ask why, for expanding is what life does. If one species doesn't, another will replace it.) Yet in a million years of expanding at a hundredth the speed of light, a civilization can spread ten thousand light years, a distance encompassing billions of stars. Is it plausible that technological civilizations are so rare as to arise on only one star among billions?

I believe that the explanation of the emptiness out there may be that technological civilizations perish before they start to spread across their galaxy—or that they start spreading, then perish in a cataclysm so great as to take the galaxy with them.

Suddenly the problem of predicting fifty years of scientific computing begins to look easy! Let's get down to it.

2. We'll talk to computers more often than type to them, and they'll respond with pictures more often than numbers.

A dramatic change in the last twenty years has been the arrival of graphical interfaces. When I was a graduate student at Stanford around 1980, we played with some Alto machines donated by Xerox, early workstations featuring windows, icons, mice and pointers, but I thought these were party tricks, too silly to catch on. Today the descendants of the Altos have driven other machines to extinction. It takes no special insight to predict that soon, an equally great change will occur as we take to interacting with computers by speech. It has been a long time coming, but this transformation is now around the corner.

It is good fun to imagine what computer graphics will be like in fifty years. I hardly dare, except to note that three-dimensional images will be as ordinary as Velcro.

Curiously, though the development of speech and graphics will make our machines ever more human in feel, less apparently numerical, the underlying computations will continue to be based on numbers represented digitally to many digits of precision. The digital idea is what makes everything possible, and it is not going to go away. This is one sense in which the scientists and engineers of the future will be further removed from the details of computing than we are, just as we are further removed than were our parents.

3. Numerical computing will be adaptive, iterative, exploratory, intelligent—and the computational power will be beyond your wildest dreams.

Adaptive computing is one of the glories of the computer age. Gauss quadrature was invented two centuries ago, but adaptive quadrature didn't arrive until the 1960s. Adaptive ODE solvers came soon after, and have turned the solution of most ordinary differential equations into the use of a black box. Partial differential equations are not yet boxed in black, but the trend is in that direction. As time goes by, adaptivity managed by the computer's intelligence becomes more and more widespread. Computers are not as wise as people, but they can explore a forest of possibilities faster than we can reason them through. In fifty years, this is how most numerical problems will be solved. We will tell the machine what we want, and the machine, an intelligent control system sitting atop an encyclopedia of numerical methods, will juggle computational possibilities at incomprehensible speed until it has solved the problem to the accuracy required. Then it will give us the answer; and if we insist, it may even tell us something of how it got there.

The power unleashed by this kind of computing will be vast. Large parts of physical reality will be simulated in real time before our eyes, with effects so far beyond what the pioneers of computing could envision that to call it computation almost seems to miss the point.

When computations are all intelligent, when everything is embedded in a control loop, the mathematical landscape will change. One distinction that means a great deal to us today is the distinction between linear and nonlinear problems. Broadly speaking, linear problems can be solved in one pass, but nonlinear ones require iteration. In fifty years, when everything is embedded in an iterative loop anyway, this difference will have faded. For the same reason, today's big distinction between forward and inverse problems will have faded too.

My next prediction is a corollary.

4. Determinism in numerical computing will be gone.

Recently our family rented a car for a holiday. One evening we wanted to look at the stars, which meant turning off the dome light. We couldn't figure out how to do it! A decade ago, closing the doors and flipping a switch would have sufficed, but nowadays, cars are more intelligent. In some, the light stays on for a fixed period after you close the door, but in ours, the situation was more complicated. There was an interlock with the engine, and some additional intelligence that we never got to the bottom of. Eventually we got the light off, but we were not quite sure how we had done it, or if we could do it the same way again.

Have you noticed how many of our machines behave this way? Photocopiers used to be deterministic: you put in your document, pressed a button, and out came a copy. Nowadays they are animated by microprocessors and have complicated arrays of internal states. The first copy may come out in landscape orientation, but the second in portrait, if the machine decides in-between that it ought to change modes. The copier may have strong opinions about all kinds of things, and resist your attempts to impose your will on it. Yesterday I wanted to put in new paper when the machine thought this was unnecessary, and to achieve my end I was forced to outwit it by tricking its paper sensor.

Further examples are all around us. Typewriters used to be deterministic, if clunky: you knew what would happen when you pressed a key. Nowadays, in Word or LaTeX, changing one character of input may alter the whole document in startling ways. Why, at highway rest stops, even toilets are intelligent devices now whose states of mind we don't fully understand. When you've used the toilet, washing your hands becomes a battle to outsmart a sensor that seems sullen about detecting your presence, and if you manage to get your hands wet, you have an intelligent hot air blower to make friends with if you want to dry them.

What's true of toilets will be true of numerical computations. In fifty years, though the answers you get will be accurate without fail to the prescribed precision, you will not be sure of duplicating them exactly if you solve the the problem a second time. I don't see how this loss of determinism can be stopped. Of course, from a technical point of view, it would be trivial to make our machines deterministic by simply leaving out all that intelligence. However, we will not do this, for intelligence is too powerful. We'll put it in, ever more of it. In the last fifty years, the great message to be communicated to scientists and engineers was that it is not reasonable to ask for exactness in numerical computation. In the next fifty, they will learn not to ask for repeatability either.

Let me move now to more concrete algorithmic matters.

5. The importance of floating point arithmetic will be undiminished.

So much will change in fifty years that it is refreshing to be able to predict some continuity. One thing that I believe will last is floating point arithmetic. Of course, the details will change, and in particular, word lengths will continue their progression from 16 to 32 to 64 to 128 bits and beyond, as sequences of computations become ever longer and require, if only at a logarithmic pace, ever more accuracy to contain accumulation of errors. But I believe the two defining features of floating point arithmetic will persist: relative rather than absolute magnitudes, and rounding of all intermediate quantities.

Outside the numerical community, some people feel that floating point arithmetic is an anachronism, a 1950s kludge that is destined to be cast aside as machines become more sophisticated. Computers may have been born as number crunchers, the feeling goes, but now that they are fast enough to do arbitrary symbolic manipulations, we must move to a higher plane. In truth, however, no amount of computer power will change the fact that most numerical problems cannot be solved symbolically. You have to make approximations, and floating point arithmetic is the best general-purpose approximation idea ever devised.

Human scientific ingenuity seems almost unlimited, when given enough time to operate. Certainly my own habit is to be optimistic about the crossing of merely technical hurdles. This brings me to my next prediction.

6. Linear systems of equations will be solved in $O(N^{2+\epsilon})$ flops.

Matrix computations as performed on machines around the world typically require $O(N^3)$ floating point operations—"flops"—where N is the dimension of the problem. This statement applies exactly for computing inverses, determinants, and solutions of systems of equations, and it applies approximately for eigenvalues and singular values. But all of these problems involve only $O(N^2)$ inputs, and as machines get faster, it is increasingly aggravating that $O(N^3)$ operations should be needed to solve them.

Strassen showed in 1968 that the $O(N^3)$ barrier could be breached. He devised a recursive algorithm whose running time was $O(N^{\log_2 7})$, approximately $O(N^{2.81})$, and subsequent improvements by Coppersmith, Winograd and others have brought the exponent down to 2.376. However, the algorithms in question involve constants so large that they are impractical, and they have had little effect on scientific computing. As a result, the problem of speeding up matrix computations is viewed by many numerical analysts as a theoretical distraction of no real importance. This is a strange attitude to take to the most conspicuous unsolved problem in the field! Of course, it may be that there is some reason why no practical algorithm can ever be found, but we certainly do not know that at this point. A "fast matrix inverse" may be possible, perhaps one with complexity $O(N^2 \log N)$ or $O(N^2 \log^2 N)$, and discovering it would change everything.

In 1985 I made a bet with Peter Alfeld of the University of Utah that a matrix algorithm with complexity $0(N^{2+\epsilon})$ for any $\epsilon > 0$ would be found within ten years. None was, and I gave Alfeld a check for $100. We renewed our bet, however, to 2005, and in that year I will renew it again if necessary. One morning, with luck, the headlines will appear. I think fifty years should be long enough.

Prediction 7 has a special historical context.

7. Multipole methods and their descendants will be ubiquitous.

The conjugate gradient and Lanczos algorithms were invented around 1950, and their story is a curious one. Nowadays we have no doubt as to what these methods are good for: they are matrix iterations, which for certain matrices bring those $O(N^3)$ operation counts down to $O(N^2)$ or even better. Though there are constants hidden in the "O", these methods are sometimes much faster than Gaussian elimination and its relatives when N is large.

What is curious is that Hestenes, Stiefel, Lanczos and the rest didn't see this coming. In the 1950s, N was too small for these iterations yet to be competitive, but all the mathematical pieces were in place. These men knew something of the convergence properties of their iterations, enough to have been able to predict that eventually, as machines grew faster, they should beat the competition. Yet they seem not to have made this prediction. A numerical analyst writing an essay like this one in 1960 might not have mentioned conjugate gradients or Lanczos at all.

It is with this history in mind that I mention multipole methods, by which I mean methods related to the recent algorithms of Rokhlin and Greengard for N-body problems and integral equations. Times have changed, and now we all think asymptotically. When multipole methods were being invented in the 1980s, they were competitive in 2D but not in 3D. But Rokhlin and Greengard saw immediately that these techniques reduced counts from $O(N^2)$ to $O(N)$, give or take a logarithmic factor, so how could they not win in the long run? And so they will.

The success of multipole methods will exemplify a general trend. As time goes by, large-scale numerical computations rely more on approximate algorithms, even for problems that in principle might be solved exactly in a finite number of steps. Approximate algorithms are more robust than exact ones, and oddly, they are also often faster.

8. Breakthroughs will have occurred in (i) *matrix preconditioners,* (ii) *spectral methods, and* (iii) *time stepping for partial differential equations.*

As I said, I am an optimist about merely technical hurdles; most of them eventually get crossed. The three predictions here are examples. The business of matrix preconditioners is vitally important, but it is a jungle these days—surely improvements are in store! Spectral methods for PDEs are in a similar state—remarkably powerful, but varying awkwardly from one application to the next. Order is needed here, and it will come. As for time-stepping, this is the old problems of stiffness, reasonably well in hand for ODEs but still unsolved in a general way for PDEs. To this day, the CFL restriction constrains our computations all across the range of science and engineering. To get around these constraints, time steps are taken smaller than we would wish, huge matrix problems are solved at great cost, and physically important terms are thrown away just because they are too hard to implement. The CFL condition will not disappear, but new weapons will be devised to help us in the day-to-day struggle against it.

Old dreams abound, some of them painfully slow to be fulfilled. In fifty years, however, many will come to fruition.

9. The dream of seamless interoperability will have been achieved.

Users and onlookers complain year after year, why is so much human intervention needed to get from the whiteboard to the solution? Why does one computer program have to be written for the grid generator, another for the discretization, and still another for the linear algebra, requiring interfaces all along the way with repeated opportunities for human error? Why are symbolic and numerical calculations separate? Why can't our ideas and tools blend together into a seamless interoperable system? Well, of course, they can, and getting there is merely an engineering problem. Fifty years from now, humans will have vanished from most of these intermediate steps. The grids and the solvers will be coupled, and humans will more and more rarely catch sight of actual numbers in the course of doing science.

It's easy to predict a revolution—say, linear algebra in $O(N^{2+\epsilon})$ operations—without providing a hint as to how it will be achieved. The next two predictions try to go further and give the hint too.

10. The problem of massively parallel computing will have been blown open by ideas related to the human brain.

The information revolution is underway; we do not know where it will take us, but the journey has begun. By contrast, the revolution in understanding how the brains of humans and other higher animals work has not arrived yet. Some key idea is missing.

Another fact about our scientific landscape is that the problem of massively parallel computing is stalled. For decades it has seemed plain that eventually, serial computers must run up against the constraints of the speed of light and the size of atoms, at which point further increases in power will come about through massive parallelism. Yet parallel computing nowadays is a clumsy business, bogged down in communication problems, nowhere near as far advanced as everyone expected a decade ago.

I believe that the dream of parallel computing will be fulfilled. And it is hard to avoid the thought that if parallel computing and the human brain are both on the agenda, then the two revolutions in store will somehow be linked. Brain researchers will make discoveries that will transform our ideas of how to compute in parallel; or researchers in parallel computing will make discoveries that will unlock the secrets of the brain; or, just as likely, the two fields will change in tandem, perhaps during an astonishing five or ten years of upheaval. The upheaval could begin tomorrow, or it might not begin for another generation. But it is destined to occur, and by 2048, I expect that parallel computing and the understanding of the brain will be incomparably further advanced than they are now, and much more strongly linked to one another.

Understanding the brain is a revolution in biology that must be coming. Meanwhile, there is another revolution in biology that is already happening: the working out of DNA/RNA genomes and their implications. Every organism from virus to man is specified by a program written in the alphabet of the nucleotides. Since Watson and Crick, we have known this must be true, and in 1995, the first genome of a free-standing organism—the bacterium *Haemophilus influenzae*—was sequenced. Since then, dozens more have followed, and everything in biology, from development to drug design, is being reinvented as we watch. If I give you the sequence `KPSGCGEQNMINFYPNVL` in the standard code for the amino acids, this is enough for you to determine in a few seconds that I am speaking of an α-macroglobulin proteinase inhibitor of *Octopus vulgaris*, and to locate related enzymes in ten other species. Just point your browser to `http://www.ncbi.nlm.nih.gov` and run *blastp*.

I believe that this drama has implications for computing.

11. Our methods of programming will have been blown open by ideas related to genomes and natural selection.

Genetic programs and computer programs are strangely analogous. Both are absolutely precise digital codes, and no other codes that we know of have anything like the complexity of these two, with the size of a genome being of roughly the same order of magnitude (3×10^9 nucleotides for *Homo sapiens*) as the size of an operating system (2×10^9 bits for Windows 98). As a generation of engineers grows up with genomics, thinking digitally about the evolution of life on earth, our methods of computer programming will change. Traditionally, computer programs are written in a different way from biological ones. There's a programmer in the loop, an intelligence, which gives computer programs a logical structure that biological programs lack (not to mention comment cards!). Yet it is notable that nowadays, large-scale software systems are too big to be understood in detail by any individual, let alone to be mechanically analyzed or verified, and indeed, the process of industrial software design already seems as close to evolution by natural selection as to mathematical logic. Software at a place like Microsoft is generated by an endless process of experiment and test, code and correct, a process in which individual human intelligences seem less important than they used to. Operating systems evolve from one generation to the next, and they are never perfect, but they work. The process may seem repugnant to computer scientists, but it is scalable and unstoppable.

Finally, a prediction that is not really a prediction, just a pious wish.

12. If we start thinking now, maybe we can cook up a good name for our field!

* * *

Table 1 lists some highlights from the history of numerical analysis. Its attempt to extrapolate to the future summarizes some of the thoughts I have expressed in this hour.

When I looked at this collection of predictions, I was startled to see that a theme emerges from them. Some are what one might call purely technical. The others, however, those marked in the table by asterisks, have a pronounced sociological flavor. They have a theme in common:

Humans will be removed from the loop. (∗)

Think about interacting with computers by speech and graphics. In this development, as we discussed, numbers will become less and less visible to humans. The computers will still work with numbers, but they will rarely take the trouble to show them to us, for we are notoriously bad at dealing with them.

Think about adaptive computations managed by intelligent control systems. The point here is precisely to remove human beings from the loop. Humans may have provided the ingredients, the fast Fourier transforms and quasi-minimal residual iterations, but we won't be asked to meddle with the details. Describe your problem to the machine, and then—well, you might as well go have a cup of coffee.

What about seamless interoperability? This development will be all to the good, or so we imagine, and yet again, it is a process of removing humans from involvement in the details. By 2048, you and I will not be needed to connect the gridder to the solver.

What about the radical new programming methods that may develop by analogy with natural selection? These will achieve their success by depending less on human programmers, more on automatic explorations and selections. For just as humans are not reliable at mathematics, they are not very good at programming either. You need a computer program? Sketch your needs to the machine, and then—well, you might as well go have another cup of coffee.

I find I have envisioned a frightening future, a future in which humans, though still the taskmasters of computers, are no longer much involved in the details of getting the tasks done. Fifty years from now, it is hard to imagine that our machines will still be dim enough to benefit much from our assistance.

That's my report from 1998, down here on the exponential.

Table 1. Some past and future developments in scientific computing. The asterisks mark developments summarized by $(*)$.

Before 1940

- Newton's method
- Gaussian elimination
- Gauss quadrature
- least-squares fitting
- Adams and Runge–Kutta formulas
- Richardson extrapolation

1940–1970

- floating point arithmetic
- Fortran
- finite differences
- finite elements
- simplex algorithm
- Monte Carlo
- orthogonal linear algebra
- FFT

1970–1998

- quasi-Newton iterations
- adaptivity
- stiff ODE solvers
- software libraries
- Matlab
- multigrid
- sparse and iterative linear algebra
- spectral methods
- interior point methods

1998–2048

- linear algebra in $O(N^{2+\epsilon})$ flops
- multipole methods
- breakthroughs in preconditioners, spectral methods, time stepping for PDE
- * speech and graphics everywhere
- * fully intelligent, adaptive numerics
- * loss of determinism
- * seamless interoperability
- * massively parallel computing made possible by ideas related to the human brain
- * new programming methods made possible by ideas related to natural selection