

FUTURE DEVELOPMENT OF SOME AREAS OF APPLIED MATHEMATICS

SULTAN SIAL

Department of Mathematics
Lahore University of Management Sciences
Opposite Sector U, DHA 54792, Lahore
P a k i s t a n.
E-mail: sultans@lums.edu.pk

Introduction

Applied mathematics has greatly benefited in the recent past from the availability of inexpensive desktop computing systems with large memory. Going beyond the desktop the development of supercomputers of a few extremely fast CPUs or clusters of many off-the-shelf CPUs have allowed extremely large systems to be studied. Computers (typically clusters) on the current Top 500 list are used by universities, oil companies (Saudi Aramco is 45th on the list), banks and militaries.

This development has made it possible to realize the modeling of detailed real-world systems analogous to the way that chemists or biologists can view small versions of larger systems and note the response to carefully monitored conditions.

What is the role of the applied mathematician in an age of computation? I believe that it is advancing ideas about how to develop algorithms that are appropriate for the problems being tackled as well as for the non-mathematicians: engineers, physicists, chemists, biologists, who will use the algorithms.

In this article, I discuss approaches to the computational modeling of mesoscale systems as well as numerical solutions of ordinary differential equations. I will outline a philosophy of approaching such problems as they are discussed and close with a discussion of future possibilities.

Disclaimer

All views expressed are merely my own and not necessarily shared by anyone else at LUMS (Lahore University of Management Sciences).

An Applied Mathematician is a Mathematician

First, one should emphasize that an applied mathematician is a mathematician first. It is not enough to simply attack a problem with whatever software and hardware one might find available within one's budget.

The theory of Sobolev gradients is rooted in the search for a central theory for partial differential equations in a variational setting [1]. This illustrates my point about an applied mathematician being a mathematician first. The development of the numerical techniques is guided by a worthy analytic goal.

Prototypical Problem

Consider the prototypical problem $u' = u$ on the interval $[0,1]$. Suppose that one didn't know the answer and desired a numerical solution. One could define an operator D_0 , which approximates the identity and another operator D_1 , which is an approximation of derivative. We do not care in principle whether this is done in a finite-element, finite-difference or other numerical setting. What is important is finding the right mathematical setting to do the problem in as will be demonstrated presently.

The equation could be considered to be satisfied when

$$F(u) = \langle D_0u - D_1u, D_0u - D_1u \rangle$$

is zero or some rather small number [1,2].

Recalling that the gradient $\nabla F(u)$ of a functional F is defined by

$$F(u+h) = F(u) + \langle \nabla F(u), h \rangle + O(h^2)$$

we find that

$$\nabla F(u) = (D_0 - D_1)^t (D_0 - D_1)u$$

where the t stands for adjoint. If the operators are represented by matrices the adjoint would be represented by the transpose.

The interpretation is that the gradient points in the direction (in function space) of the greatest increase of the functional. It makes intuitive sense then to move in the opposite direction in order to decrease the error the fastest.

Since one does not know how far one should move this would have to be an iterative process: find the gradient, move a small step in the direction opposite to the gradient, find the gradient again etc We can represent the iteration symbolically by

$$u \rightarrow u - \lambda \nabla F(u)$$

We have now rediscovered the reasoning behind the technique known as steepest descent. The reasoning is quite intuitive and programming this algorithm for a uniform grid using finite differences or with a finite element software would be simple.

The Wetware

Year after year one sees faster computers with greater memory opening up greater possibilities for numerical analysis and scientific computation. Algorithms are written that exploit new possibilities such as multiple CPUs on a motherboard. However, the applied mathematician should keep in mind that the engineer, chemist, or physicist using the algorithm has a different sort of training and different priorities than those of a mathematician. For this reason I propose the following:

A Least Action Principle for Applied Mathematicians

Algorithms should be as simple and elegant as possible, so as to be understandable by the widest range of possible users, while being mathematically and intellectually honest.

The steepest descent algorithm is certainly simple and elegant and could be explained to the likely users.

Numerical Results

The prototypical problem was attacked using equally spaced nodes and first-order finite-difference operators. The results are terrible. As the mesh becomes finer the step size λ has to be decreased otherwise the scheme becomes unstable. So one has to either settle for very coarse grids or wait a very long time to get the answer.

We have run into the CFL (Courant-Friedrichs-Lewy) condition. This condition becomes even more severe as the order of the derivatives increase or if we go from one to two or three dimensions.

The Least Action Principle Again

We could plot the gradient and try to see what is going on. In the finite-difference case we see that the gradient is extremely spikey and at the beginning (if we start with a constant value of u) is quite large at the end points.

So one remedy which suggests itself is to use an explicit fourth-order Runge-Kutta scheme. This would allow us to use a larger step size for any given mesh than the steepest descent scheme we started with. However, this is still a descent scheme and will still suffer from the CFL condition. We would in effect be left with the same difficulty but the numerical scheme would be more complicated.

We could use a Newton's method. But the numerator involved in the method is the gradient and we would also have to instruct the person using our algorithm about how the Newton's method will happily take one to infinity if one does not start close enough to the required solution.

Another possibility is conjugate gradient. However, one is again dealing with a gradient and in fact exploring directions other than ones that decrease the error the fastest.

We could refine the mesh spacing and use even higher order operators. This one seems to make the situation worse.

The worst possible "solution" is to artificially smooth the gradient without any rhyme or reason.

Finally, we could appeal to the suggested least action principle for applied mathematicians. To move in the direction that reduces the energy the fastest seems quite sensible. Let's keep that idea if we possibly can. If not for our own sake for the sake of the non-mathematicians who need to use the algorithms that we develop. Recall that applied mathematicians are mathematicians first. We need to look at the definition of gradient.

Other Gradients

Reexamining the notion of gradient we recall that the gradient was defined in terms of the inner product. There are of course infinitely many possible inner products and there is no need to consider the "usual" inner product to be somehow natural or favored.

One such inner product would give us a different way of defining a new gradient $P\nabla F(u)$ by

$$F(u+h) = F(u) + \langle P\nabla F(u), h \rangle + \langle P\nabla F(u)_x, h_x \rangle + O(h^2)$$

The benefit of this new gradient is that it takes into account the spatial derivatives that appear in the problem.

Numerical Results

The following table compares results for the usual gradient and the Sobolev gradient in terms of the step size that could be used, the number of steps required, and the CPUs to reduce the error to a required tolerance.

Nodes	Step size	Steps	CPUs
51	0.00020/0.9	4645/44	0.06/0.05
101	0.000050/0.9	19335/44	0.34/0.11
201	0.000012	81541/38	2.03/0.15

It is seen that we have retained an explicit steepest-descent scheme but the CFL condition no longer applies. The same step size can be used no matter how fine the mesh is and it takes about the same number of steps to reduce the error to the required tolerance.

When I have mentioned such results to quite knowledgeable applied mathematicians I have often been told that they are impossible. This is because of a deeply ingrained aversion to steepest descent.

Poisson-Boltzmann Equation

We have used the prototypical equation (a more detailed analysis is in [1]) to illustrate the point that even simple ODEs can give rise to unexpected numerical difficulties and that focusing on the mathematical setting allows one to get around these difficulties. The hard work should be on the mathematical side before the coding takes place. Now we consider an equation of importance in biochemistry.

If we think of a molecule in a saline solution with equal numbers of positive and negative ions distributed according to Boltzmann weighting then one model for such a system is that the electric potential is found by minimizing

$$F(u) = \int \kappa \cosh(u) + \frac{\epsilon}{2} |\nabla u|^2 + \rho u$$

Alternatively, people try and solve

$$\kappa \sinh(u) - \nabla \cdot (\epsilon \nabla u) = 0$$

subject to various boundary conditions [3]. Consider the problem where $\kappa = 10$ in the solution and 0 in the charge $\epsilon = 2$ distribution, in the solution and 80 in the charge distribution. The system is one dimensional on the interval $[0,10]$ and the charge distribution is $\rho = 0$ except for the interval $[0.30,0.35]$ where it is 100 and $[0.70,0.75]$ where it is -100 .

We can predict two sources of numerical difficulties. These are the exponential nonlinearities as well as the sharp discontinuities in the coefficients.

We consider the "usual" gradient, the Sobolev gradient discussed previously and a new Sobolev gradient defined by

$$F(u+h) = F(u) + \langle \max(\kappa, 1) P \nabla F(u), h \rangle + \langle \epsilon P \nabla F(u)_x, h_x \rangle + O(h^2)$$

This is a combination of the techniques discussed in [2] and [3]. The Sobolev gradient has been modified so as not to take only the derivatives into account but also the relative importance of the various terms and the changing coefficients in different regions of the system.

Numerical Results

The comparison of the results for the various gradients is seen below. Steepest descent with Sobolev gradients is seen to be efficient and weighting for coefficients is also seen to offer an advantage.

Nodes	Step size	Steps	CPUs
101	0.000066 / 0.02 / 0.2	16 587 / 366 / 92	4.10 / 1.83 / 0.81
201	0.000015 / 0.02 / 0.2	74 866 / 434 / 94	28.98 / 4.92 / 1.52
401	0.0000039 / 0.02 / 0.2	289 224 / 488 / 120	206.13 / 12.72 / 4.69

Least Action Principle for Applied Mathematicians

The least action principle for applied mathematicians can now be considered to be:

Algorithms should be as simple and elegant as possible, so as to be understandable by the widest range of possible users, while being mathematically and intellectually honest, and taking all important features of the problem into account.

Other Application of Sobolev Gradients

To end this article let me mention that Sobolev gradients have been applied for a wide variety of problems in addition to the ones discussed such as phase transitions [4], superconductivity [5,6], shock waves and airflow [1], and shape-memory systems to name just a few.

Future Directions

An important application of numerical modeling using Sobolev gradients will likely be in materials sciences. In particular, now that the various possible symmetries of all known shape-memory materials have been worked out it should be possible to tackle this problem from the point of view of energy minimization [7]. It should also be possible to make progress in the theory and numerics of integro-differential equations. Another important point is to develop more systematically convergence results and settle the question of what is the “best” gradient for a problem and how to define “best”.

REFERENCES

- [1] J. W. Neuberger, Sobolev, Gradients and Differential equations, *Springer Lecture Notes in Mathematics*, 1670, 1997.
- [2] W. T. Mahavier, A Numerical Method Utilizing Weighted Sobolev Descent to Solve Singular Differential Equations, *Nonlinear World*, **4**(4), (1997).
- [3] W. B. Richardson, Sobolev preconditioning for the poisson-boltzmann equation, *Computer Methods in Applied Mechanics and Engineering*, **181**(4), (2000).
- [4] S. Sial, J. W. Neuberger, T. Lookman, A. Saxena, Energy minimization using Sobolev gradients, *Journal of Computational Physics*, **189**, (2003).
- [5] J. W. Nueberger, R. J. Renka, Numerical determination of vortices: simulation of cooling, *Superconducting Science and Technology*, **16**, (2003).
- [6] S. Sial, A Sobolev gradient algorithm for minimum energy states of s-wave superconductors - finite element setting, to appear in *Superconducting Science and Technology*.
- [7] S. R. Shenoy, T. Lookman, A. Saxena, A. R. Bishop, Martensitic Textures: Multiscale Consequences of Elastic Compatibility, *Physical Review B*, **60**(18), (1999).