From: AAAI Technical Report FS-92-01. Copyright © 1992, AAAI (www.aaai.org). All rights reserved.

Intelligent Scientific Computing

Professor Stanly Steinberg
Department of Mathematics and Statistics
University of New Mexico
Albuquerque NM 87131-1141 USA
stanly@math.unm.edu

Abstract

I will describe what the scientific computing process is (should be?) when it is carried out by humans and indicate where symbolic and artificial intelligence methods can be used to significantly improve the process.

Introduction

In this paper, I will describe my notion of intelligent scientific computation, as done by humans. Immediately I expect the reader to wonder what right I have to do this, so I will describe a little of my background in scientific computing. My academic department has about ten faculty who have a serious interest in scientific computing. Typically, we have at least one ongoing seminar on scientific computing and I follow my colleague's work quite closely. I have been consulting for the company Ecodynamics Research Associates for about ten years. This company provides scientific computing software for government laboratories and their contractors. I attend, present papers at, and help organize from three to six workshops or conferences each year. The areas that I have worked on are static electric-field, ocean, compressible flow, laser-switch, porous-media flow, and contaminant transport in porous media modeling. Also, I have done an optimal design for electrodes using a numerical optimization algorithm coupled with an electric field modeling code.

My interest in scientific computing began about fifteen years ago when I used a computer algebra system (CAS) to compute analytic solutions to partial differential equations (PDEs). Later I used a CAS to obtain perturbation expansions to PDEs. My serious numerical work started when I began consulting for Ecodynamics and used a CAS to write some rather large numerical subroutines. Since that time, I have done research on numerical grid generation using variational methods and finite difference methods using grids generated by such algorithms. One of my Ph.D. student wrote a thesis on a mixed finite-element method. My main work at Ecodynamics is to develop numerical algorithms and provide FORTRAN code for those algorithms. The numerical algorithms are typically analyzed using a CAS system and the code is generated using a CAS system.

I have broad interests in applied mathematics, scientific computing, and the science that underlies the models that I work with. A consequence of this is that, at Ecodynamics, I am frequently called on to trouble shoot problems in numerical codes. Many of my comments will be related to these bug-chasing experiences.

Scientific computing is carried out in a number of ways: I will describe my preferred method. The first thing to do is to break the process up into a number of steps:

- choose an analytic model
- analyze the model
- discretize the model
- analyze the model
- choose a computer
- do the simulations

In a typical solution of a modeling problem, these steps are intermixed.

Choosing an Analytic Model

The choice of a model for a given situation is in what I think of as the domain of the scientist or engineer, but the person doing the scientific computing related to the model must have some familiarity with the underlying science. The first thing to note is that the choice of an analytic model step is frequently skipped by modelers. However, if a modeler goes directly to a discrete model, then an analytic model can be "backed out" of the discrete model. What I have in mind when I say an analytic model, is that the problem has been formulated mathematically and that it is possible to determine if the model is *properly posed* (see below). I have only worked with PDE models, so my comments are restricted to these and closely related types of models. Analytic models come in sizes that range from a

From: AAAI Technical Report FS-92-01. Copyright © 1992, AAAI (www.aaai.org). All rights reserved.
few algebraic equations and, perhaps, an ordinary differential equation, to those that involve a system of five to ten PDEs in three space dimensions and time.

An important point here is that analytic models are more accurate than the numerical models. That is, for complicated problems, analytic results provide more accurate descriptions of the system being modeled than can be provided by discrete models using the largest computers. Moreover, the mathematical properties of the analytic model more accurately represent important properties of the system being modeled than do the mathematical properties of the discrete model. For example, numerical effects, such as loss of conservation, diffusion, and dispersion quickly swamp the related physical properties, while in analytic models these effects can be set to zero, which for the relevant systems is an accurate approximation. Thus a typical goal of scientific computing is to accurately approximate an analytic model.

Analyzing the Analytic Model

An analytic model is properly posed if conditions such as the following are satisfied:

- there are the same number of PDEs as unknown functions
- there are as many initial conditions for an unknown function as the highest number of time derivatives of that functions appearing in the PDEs
- there are a "proper" number of boundary conditions
- the dependencies and units of all variable are known
- etc.

The point here is to "formally" guarantee that the model problem has a unique solution (or at least one solution and at most a finite number of solutions). If this is not the case then converting the analytic model to a discrete model produces a disaster.

It is also important to know if the analytic model is *well posed*, that is, it possesses a unique solution that depends continuously on the data in the model. Both well-posed and *ill-posed* models are useful. However, the numerical methods used to study each type of model are drastically different. Well-posed models are certainly properly posed. The easiest way to check if an analytic model is well posed is to determine if it is a special case of some class of initial boundary-value problems for which the result is known. Otherwise, obtaining the answer is a serious research question.

During the analysis it is possible to identify parts of the equations in the model as being diffusion terms, transport terms, forcing terms, insulted boundaries, boundaries on which the solution is given, etc. Each of the classifications correspond to important physical processes and mathematical properties. This classification seriously impacts the type of discretization used for such terms. It is also important to classify the model using mathematical properties such as linearity and translation invariance. Again, such properties correspond to important physical properties of the underlying system and seriously impact the choice of numerical algorithms.

It is important to realize that the choice of an analytic model is heavily influenced by how difficult it will be to work with its discretization.

Numerical Models

There are two important types of choices in creating a discrete model. Which model is to be discretized and by what method is it to be discretized?

The choice of a discretization of a continuum model relies on the properties of the process being modeled, or what is the same thing, the mathematical properties of the model. The choice of discretization also depends on the architecture of the machine to be used for the computing. For example, on a serial or vector machine, the time evolution of diffusion terms is handled implicitly. Such decisions are in the domain of a scientific computing expert and it takes years of experience to learn to make good choices.

I think of analytic models as coming in three typical sizes: simple, modest, and large. The simple models typically involve a few algebraic equations and, perhaps an ordinary differential equation. Modest models involve PDEs in two space dimensions or in time and one space dimension. Large models are, of course, anything larger. Large models commonly involve five to ten PDEs in time and three space dimensions. Numerical problems coming from simple models can be solved by hand or require at most a small PC, while the solution of numerical problems in modest models require a large PC or workstation class machine. The solution of numerical problems in large models can easily require computing resources that exceed those provided by the largest machines available.

Let's take a look at an example. To accurately model the transport of a contaminant in a complex aquifer requires a full three dimensional model. However, there is not sufficient data to fully characterize the aquifer, so the missing data will be simulated statistically. To assess how far the contaminant will be disbursed, the statistical estimates of the missing parameters are used to generate from a hundred to a thousand simulations, which should provide sufficient data to estimate a useful worst-case scenario. Such simulations are well beyond the capabilities of all but the largest computers and are very expensive. However, it can be determined that most of the aquifer flow is between two impermeable layers and mostly in horizontal direction. This allows the three dimensional model to be replaced by a two dimensional model. Now it is more reasonable to do the statistical analysis and estimate the worst-case parameters. Finally, a modest set of three-dimensional simulations can be run to check the two-dimensional

From: AAAI Technical Report FS-92-01. Copyright © 1992, AAAI (www.aaai.org). All rights reserved.
simulations. If a particularly poor numerical method is chosen for the two-dimensional models then, in fact, they can be replaced by a simple algebraic model, which is a lot cheaper to work with than any PDE model.

What we have here is a complex problem in trade-offs between accuracy and computational expense. Experienced modelers know a lot about this problem, but I believe that serious mistakes are made in assessing such trade-offs in most complex modeling problems.

Analyzing the Numerical Model

The analysis of discrete models is well advanced for simple linear systems and manageable with the help of a CAS for complex systems. The linear analysis of nonlinear systems is equivalent to the analysis for linear systems; a full nonlinear analysis of all but the simplest systems is still an area of intense research.

Here is a critical connection between the discrete and the continuum (The Lax-Richtmyer Theorem). In the linear case, under the assumption that the continuum model is well posed, it is possible to check if the discrete model is well posed. Without the continuum model, this is a far more difficult problem. The typical items checked in the analysis are: What is the order of accuracy of the numerical scheme; and is the scheme stable. Checking the stability is algebraically complex in all but the simplest cases. It is also important to know the diffusion coefficients, the dispersion coefficients, and the group velocities for the discretization. In transport problems it is important to know if a scheme is TVD (total variation diminishing) or ENO (essentially non-oscillatory).

No scheme does everything well, so in a given problem trade-offs are mandatory. For example, implicit schemes are very stable but highly diffusive. For some problems this works well, but for others the diffusion destroys the usefulness of the method.

Choosing a Computer

If one only has a workstation or a PC then the choice of a computer is not difficult (but choosing a model may be). Many people who do scientific computing have the choice of a wide range of serial, vector, and parallel computers. What machine should be chosen? Unfortunately, the choice is not based on just how many cycles can be obtained for a dollar. For example, explicit algorithms are embarassingly parallel, while implicit algorithms are strikingly fast. Unfortunately, implicit algorithms require global communication and thus are poorly suited for parallel machines. There is a trade-off between algorithms and architecture that is not yet well understood.

Do the Simulations

Again, this is not as simple as it may seem. To set up some simulations, thousands of parameters must be entered into a computer. These involve such things as describing the physics and geometry of the problem. Generating grids for problems with complex geometry can take several person months. The output from the simulation can be enormous and incomprehensible, requiring analysis and special displays such as three-dimensional graphics. Simulations often involve running a single simulation, analyzing the results, altering the parameters and then running the simulation. This can be done hundreds of times in the analysis or design of devices or systems.

Scientific Computing

A great problem in scientific computing is that typically there is neither enough human nor monetary resources to do it well. It is not possible to try a large number of variants of three-dimensional finite-difference algorithms using human programmers because the expense is just too great even if the skilled personnel can be found. There is a similar problem with the analysis of algorithms. To complete a modeling job in a reasonable time, one can only do a cursory analysis and rely on experience to produce reasonable results. Each step of the modeling process involves short cuts and trade-offs whose impact on the process is difficult to judge.

It appears to me that the intelligent way to do such complex tasks as scientific computing is to break the process into small steps and build tools that help humans deal more efficiently with each step. For example, in choosing models it would be a great help to have a data base that contains all of the usual models along with a CAS system that can be used to combine, simplify and manipulate the models. The analysis of all but the simplest analytic models requires the aid of CAS tools. The discretization of the model is aided by the use of CAS tools, but this technology is not up to the task. The logic of what discretization to use is complex enough and requires enough expertise that a combined CAS and AI system is needed. Analyzing the discrete model is a fairly direct task that requires substantial algebraic and, typically, some numerical computing. At some point, all this information must be put together and decisions made about the various aspects of the model process; it is clear that AI techniques can be used here.

Bugs

How are bugs found? In the usual modeling process all parts of the process are suspect. If the computed results don't agree with experiment or intuition, then the experiment, intuition, model, algorithm or code could be wrong. If the code has error, then all parts of the code are suspect. In such a situation the debugging process is costly and time consuming (if one likes detective work, it can also be great fun). A way to reduce the problems in the debugging process is to have as many parts of the model as possible certified correct. For example using

From: AAAI Technical Report FS-92-01. Copyright © 1992, AAAI (www.aaai.org). All rights reserved.
a well-tested linear equation solver or generating the solver using a system that is well tested and known to produce correct solver greatly reduces the probability that there is a problem in the solver. It is practical for many parts of the scientific computing process to set the probability of a part having an error to zero.

Then, if none of the pieces contain an error, but there is still a bug, it has to be in the interfaces. For example, the linear system being solved may not be in the class for which the solver is guaranteed to produce accurate results. This can be the result of an easily detected error such as the lack of symmetry or it can be more subtle and detectable only from proper analysis of the discrete problem. In large scientific computing problems, the interfaces are a real nightmare; AI technology can certainly help.

Summary

I like to view the scientific computing process as a sequence of problem transformations[1]. The process begins with a description of a problem using technical English, some equations, and drawings or other geometric specifications. This is transformed into a mathematical model. In fact, there are always many models available. We can think of there being a most general model and then various simplifications. A systematic method of changing from one mathematical model to another is important. Moreover, if one has a general model, then it is possible to estimate the errors made in going to a simplified model. Next the mathematical model is transformed into a discrete model and then the discrete model is transformed into a computer program. Of course, such a process can be broken up into many smaller steps.

As long as the transformation of the model is determined, then for all but the simplest models, a CAS system is the best tool for doing the transformation. A CAS system is also the tool of choice for analyzing the various models. However, when it comes to a decision of what transformation to use, there are many influential factors, and AI techniques are the tools of choice for helping with such decisions.

I believe that the choice of how simplified an analytic model to use is one of the most difficult "local" problems in scientific computing. Many times I have seen scientists or engineers choose an overly simplified model because they have an out-dated idea of the power of numerical algorithms and modern computers.

The most difficult global problem is: How can such a complex task as scientific computing be made more scientific? This is now my favorite area of research[2].

[1] Thanks to Richard Zippel.

[2] Joint work with Elaine Kant.