Comments on Modeling, Simulation and Information Systems Technology Roadmap– perspectives from heliospheric physics, space plasma physics and turbulence theory

W H Matthaeus

Bartol Research Institute and Department of Physics and Astronomy University of Delaware

A spectrum of research and scientific computing problems

- Grand challenge problems
- State of the art computations
- Difficult production problems/ planned expansion
- Routine computing with maximum resources
- Data handling and archive problems

Interaction between the science and the technology:

- Learn how to use the technology and methodologies
- Develop ways to use the technologies
- Push the envelope
- Adaptation for changing computing paradigms
- Maintaining contact with applied math and science

Issue: evolving computing paradigms

- Core→Fast drum cache (IBM)
- RAM (IBM/Intel); fast Disks
- Virtual memory (IBM370, VAX…)
- Stacklib (CDC)
- Pipelining (CDC Star/Cyber 200)
- Vector (Cray)
- Parallel architectures (MPP, TM…)
- RISC /Large cache (Workstations)
- NOW → Beowulf (distributed memory)
- Fast interconnect (Dolphin, Miranet, Gbit, Infiniband…)
- Shared memory multiprocessors ($$$)
- Multicore CPUs/ local shared memory/ clusters of these…

Question: algorithms must evolve along with these changes. How much of this adaptation could have been done by software? To what degree is it necessary or desirable for the computational physicist (end scientific user) to retool for such changes in the future? Can compilers do some of this for us? (Open-MP, etc)

Issue: role of computational physics and applied mathematics

- Evolution –
  - From home-grown applications with use of system and library routines (Linpak, system routines, FFT drivers…etc)
  - Towards use of packaged codes (Zeus, Bats-R-Us…), established frameworks, community codes (CCMC), commercial packages (IDL, Mathematica…)

- Education and Scholarship –
  - Traditionally PhD students studied applied math background of numerical methodologies. Needed to understand derivation of methods, convergence properties, etc.
  - With growth in use of higher level languages, frameworks, group and community developed codes, and use of architecture-motivated language constructs (like MPI) this has become much more the exception than the rule. The scientific user has become typically more detached or insulated from the details of the codes that relate to the science.

Question: Is this detachment and lack of study of computational physics and mathematics a problem, or is it just a new division of labor? If it's a problem, how can we get the required and relevant study and scholarly activity back into out curricula and training? (It is not in the interest of the technology-developers to promote this…)

Comments on languages and compilers

- At various times, computer scientists have told us to use:
  - Forth, Pascal, C, C++, Python…
- Physicists have continued to use Fortran (even Fortran 90) , but it is almost impossible to get a computer science department to teach a course in computational methods in Fortran
- If you would like to hear them, I have horror stories about use of Python/Fortran framework, and use of Fortran/MPI/C++ framework…
- I tell my PhD students that they MUST learn Fortran
- Packages/languages such as Matlab, IDL, Mathematica are extremely useful for data analysis, graphics, and special purposes
- Choice of language is a major factor in teaching/learning computational physics & mathematics
- In recent years there has been a great increase in compilers (and perhaps language constructs, e.g., in MPI) that produced unstable code, code that is not transportable, or code that produces different results on different computers. (This includes Fortran, too.) It is a serious problem !
- Is the open-source model adequate to control/correct such problems?

Question: How much work is being done/will be done on improvement of compilers? How much of the work of adjusting to changing computing paradigm can be accomplished at the compiler level? Can stability and portability be improved ? Is it in NASA's interest to promote these as goals?

Some particular computational challenges for space physics /heliophysics

- The field includes "individual" grand challenge problems such as 3D MHD turbulence
  - Large Reynolds number → large problems
  - Best methods involve major interprocessor communication
  - Like hydro but worse, especially with Hall effect or coupling to inhomogeneous system of small scale kinetic physics
- The coupled heliospheric system (e.g., Space Weather), involves difficult cross-scale and multiphysics couplings such as

  chromosphere → corona → solar wind

  or

  large system-wide scales → inertial range turbulence → kinetic physics response

These problems are of comparable (or greater) difficulty as "flow around a 747" or terrestrial global circulation models for climate, and direct solution through modeling and simulation will represent a major challenge for decades to come.

We need a well trained students and effective workforce to address these problems, as well as capable technology and appropriate tools to interface between computational science and computational implementations.