View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Purdue E-Pubs

Purdue University

Purdue e-Pubs

Department of Computer Science Technical Reports

Department of Computer Science

1996

Computational Science as One Driving Force for All Aspects of Computing Research

John R. Rice
Purdue University, jrr@cs.purdue.edu

Report Number:

96-026

Rice, John R., "Computational Science as One Driving Force for All Aspects of Computing Research" (1996). *Department of Computer Science Technical Reports*. Paper 1282.
https://docs.lib.purdue.edu/cstech/1282

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

COMPUTATIONAL SCIENCE AS ONE DRIVING FORCE FOR ALL ASPECTS OF COMPUTING RESEARCH

John R. Rice

Purdue University
Department of Computer Sciences
West Lafayette, IN 47907

CSD TR-96-026
May 1996

Computational Science as One Driving Force for All Aspects of Computing Research

Position Paper by John R. Rice
Department of Computer Sciences
Purdue University
May, 1996

Changes are coming in the way scientific research is funded and many people are concerned that industry and government will decrease funding levels, and that the emphasis of research funding may shift. The latter fear comes from the emerging focus on funding *strategic research* — research that has the clear potential to benefit the nation's economy, health, environment, education, or other important goals. This new focus replaces the goal of simply advancing science, which prevailed for the previous 40 to 50 years. Thus the grand challenges in computational science of the late 1980s, which were to advance the frontiers of science, were later enlarged to include "national challenges" of benefit to society. Strategic research is the continuation of this shift from science for science's sake to science for society's sake. Computer science is at heart an applied field and the great majority of computing research has strategic connections. Computational science and engineering applications are one driving force that involves all aspects of computing research.

A shift toward funding "strategic" research

Strategic research is still fuzzily defined, but it refers to *areas* of research and not to types. It is orthogonal to concepts like basic, applied, long-term, or short-term. The nature of strategic research is illustrated by the six fundamental and over-reaching goals defined for all federal science and technology investments (from the National Science and Technology Council, [1], which reports to the President):

- A healthy, educated citizenry
- Enhanced national security
- World leadership in science, engineering, and mathematics
- Improved environmental quality
- Job creation and economic growth
- Harnessing information technology to support all of the other goals

Examples of important science areas that would not fit into the "strategic" scheme include studying the origins of the universe ("big bang" theory), proving Fermat's last theorem, finding whether P = NP or not, solving the four-color problem, and discovering the evolutionary origin of birds.

In my view, the current pressure to decrease — or at least realign — science research budgets is due almost entirely to the push to control budget deficits. Another intriguing but more nebulous theory also deserves passing mention: perhaps the science establishment is reaching its mature size, measured as a reasonable proportion of human activity. We may have simply reached the time when the number of scientists, after four centuries of exponential growth, must begin to stabilize. In this view the problem is not so much the lack of a reasonable and proper amount of research money, but the ever-increasing flow of bright young researchers competing for it.

Whichever theory is correct, the emphasis on strategic research appears inevitable; but as computer scientists we need not fear it. My thesis is that all subareas of computing research can prosper in an era of funding the so-called strategic areas of research. I argue for this thesis by examining science and engineering applications, though one could make equally compelling arguments based on other strategic areas. However, prosperity will not come automatically. The computing research community has been described as inward-looking, [2], and many, perhaps the bulk, of its members have avoided applications entirely. This can be understood and justified by the fact that the young field of computer science needed time to firmly establish its own foundations. Those foundations are laid. Now computer scientists must become more outward-looking, and appreciate that computational science and other applications will essentially involve and greatly challenge all subareas of computing research.

The context for the future

Growth in computing power continues to be astounding and shows no signs of abating. That this growth is unprecedented in recorded history is illustrated in Table 1, where quantitative changes in computing are compared to changes in speed of transportation, maximum power of an explosion, construction, and education. The growth of computing power over the next two decades alone — coming on top of five decades of already explosive growth — will exceed the growth in transportation speed from the time when everyone walked to the supersonic jets projected for the early 2000s.

	Area				
Year	Transportation (miles per day)	Computation (multiplies per second)	Explosive power (tons of TNT)	Construction	Education (avg. years, US)
Ancient times	40	0.005	0.0003	Great Wall	None
1890	200	0.04	0.5	Suez Canal	1
1950	6,000	40.0	1,000,000.0	Fort Peck Dam	8
1970	35,000	10,000,000.0	100,000,000.0	Aswan Dam	10
1990	35,000	5,000,000,000.0	100,000,000.0	US highways	12
2010	150,000	20,000,000,000,000.0	100,000,000.0	?	?

The nature of this growth is illustrated by a look at the recent history of a simple application: compute where the cooling water pipes should go in an automobile engine block (see Figure 1).

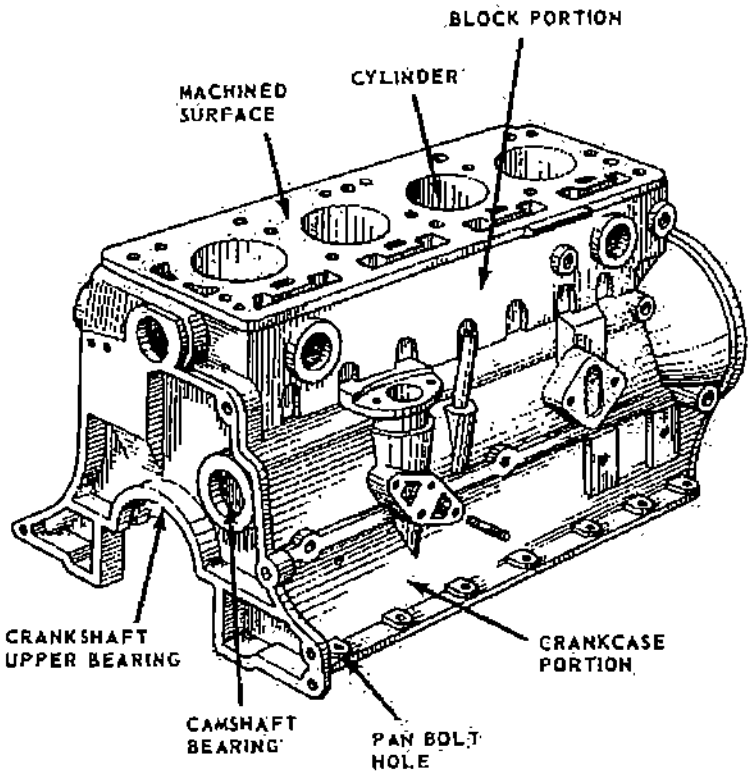

Figure 1: A typical automotive engine block. Three decades ago it was economically infeasible to compute the best locations for the cooling pipes. Today it is quick and cheap.

This real-world problem has been "solved" for many decades by building engine prototypes, making experiments, and using noncomputational analog methods — expensive and time-consuming approaches that do not optimize the cooling system design. A computer solution should not be fundamentally difficult. It involves one of the best-understood physical phenomena, heat flow. One just has to solve the Poisson problem for a complicated three-dimensional object. Methods and machines were available in 1940 that could, in principle, solve such a problem. Yet as a practical matter (or rather, *im*practical), I estimate that this computation for just one engine block would have cost the entire wealth of the United States in 1940. When I first encountered the problem in 1963 there had been enormous progress in both computing hardware and algorithms since 1940. Nevertheless, the computation was still not economically feasible. Today the computer time to solve it costs a few tens of dollars. Very significantly, algorithmic progress has been a larger factor in decreasing the cost than progress in the speed of computing hardware.

Though we can quibble about whether computers will become 1,000 or 5,000 times faster over the next 20years, as an order-of magnitude estimate we can expect with confidence that 10 megaflops of power with 10 megabytes of memory will cost $5 in 2015. Moreover, for every computer we are using now, we can expect that by 2015 there can be, for the same cost, 999 other machines working in tandem with it to provide better service.

How computational science and engineering will drive computing research

Three examples will illustrate the nature of future computational science applications, and show how these "strategic" projects will invigorate all aspects of computer science.

Designing physical objects and mechanisms

The first application is the use of computer simulation for designing simple and complex physical mechanisms, an area also known as electronic prototyping. It is a near term application that will be pervasive in industry and which involves essentially all of computer science. Figure 2 shows a collection of images from a current system devised to facilitate such work [3]. This is a prototype *problem-solving environment* [4], PDELab, for applications based on partial differential equations. A typical problem is illustrated: optimizing the shape of the end of a piston rod. We want to reduce the size of the end piece while maintaining adequate strength.

At the top level, this application involves the following subareas of computing:

- *Simulation of physics.* Many kinds of phenomena must be modeled, and modeled more accurately than needed just to control the mechanism. Further, everything must be simulated — not just one or two key parts of the mechanism.

- *High-performance computing.* A useful rule of thumb is that it takes from 100 to 1,000 times as much computing power to optimize a design as it does to simulate one instance. Thus computers that deliver tens, hundreds, and thousands of gigaflops are necessary.

- *Artificial intelligence and expert systems.* Expertise is needed throughout such an application. Many components of good design are not yet codified in a way that they can be used routinely. Thus a large array of heuristics about design must be incorporated. Furthermore, the point is not just to design the mechanism but to manufacture it, which introduces many further constraints. These constraints are even less well understood than those of design and they must be represented primarily by heuristics. Shape optimization involves many variables interrelated in complex, nonlinear ways. Such optimization can consume enormous computing resources. It is plausible that the emerging idea of application-specific optimization algorithms will be needed. Here one applies optimization algorithms to a set of similar problems (for example, optimizing the shape of a piston rod or a crank handle) and "learns" those tactics that are effective for this particular set of problems. This is a promising, simply stated idea, but a stiff challenge to carry through! Finally, even the management of the computing resources for such a computation-intensive application requires sophisticated optimization and heuristics.

- *Geometry and graphics.* It is obvious that physical design requires extensive geometry and graphics facilities. It is less well recognized that "geometric computing" is grossly underdeveloped compared to numerical, symbolic, and logical computing. Even simple shapes cannot yet be manipulated in hardware, or at hardware-like speeds; yet design applications sorely need this capability.

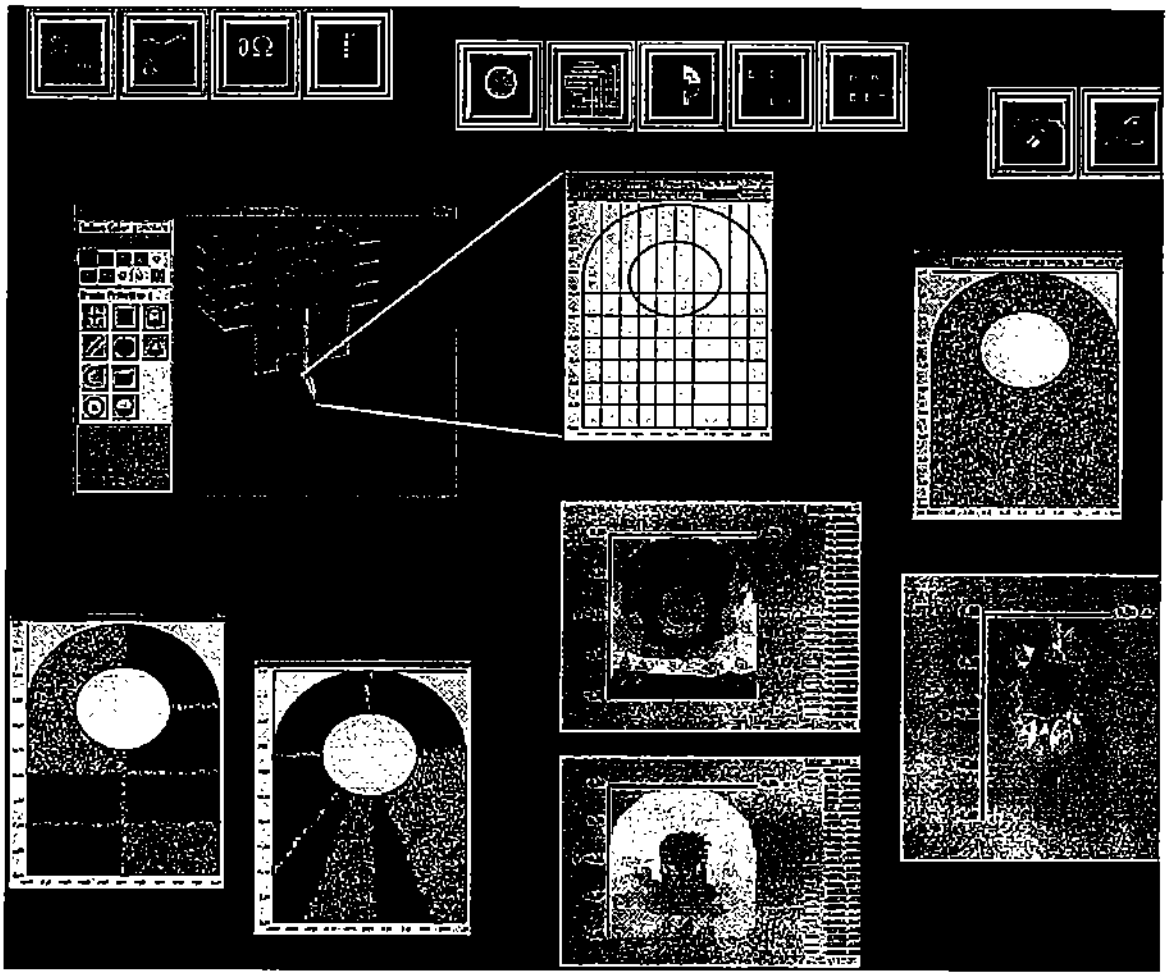

Figure 2: Screen images from the PDELab system, showing steps in optimizing the shape of a simple engine part. Improving this type of design-and-optimization system demands advances in many areas of computer science.

- *Databases.* The data items relevant to mechanical systems are not like simple banking records. Further, an enormous number of such items must be available as part of the knowledge about how things are designed and how things are manufactured. Database technology needs substantial changes to accommodate the needs of this type of application.

- *Human interfaces.* A physical design system will involve millions of lines of code, and thousands of software modules and subsystems. Yet a designer should have easy, natural, and responsive control of the design process. This presents a great challenge to the designers of human interfaces.

The substructure of this design application involves an even broader range of the subareas of computing.

- *Algorithms and data structures.* Hundreds of these are needed for all the specialized representations and manipulations performed on complex objects. The objects involved have many attributes and mountains of associated data. The detailed nature of these structures is not known in advance so the definition must be dynamic and the large sets of them must be self-organizing so that access paths to the structures are dynamically determined.

- *Parallel algorithms.* Parallel computing is the only hope to provide the computing power needed. It will not be easy to supply the power needed in a responsive way.

- *Knowledge bases, smart algorithms, adaptivity, and learning.* The substructure of the "smart systems" involved in this application will include most techniques and methodologies of artificial intelligence applications. These methodologies have been under development for several decades with mixed results. It is time to produce effective systems for adaption and learning. The increased computing power now available, plus developing application-area-specific techniques, may make this happen.

- *Symbolic systems.* Mathematics is the basis of most modeling of physical systems and is heavily used in some approaches to geometry computations. Thus manipulation and analysis of mathematical formulas is necessary in this application.

- *Languages.* Language of many forms is involved: jargon from application areas, cues, natural language, and, above all, visualization.

As if the above were not enough to keep computing researchers busy, this application is harder yet because it involves:

- *Distributed design.* Even modestly complex mechanisms are designed by teams, so networks of collaborating people and computers must be supported. Each application will have its own operating system, one with more demands and as much complexity as the generic operating systems of the 1980s. There will be more heterogeneous resources to manage, more deadlines, and more synchronization constraints.

- *Performance analysis.* The performance of such systems will be awful at first. There are hundreds of places to lose efficiency.

- *Security.* There is no information more valuable to a company than the designs and specifications of the products it plans to introduce.

Finally, this application involves the central, unsolved problem of computer science: *how to engineer software effectively.* Millions of lines of code will be involved, from multitudes of sources; still, the system must be reliable and efficient. We have a long way to go in understanding how to build such systems well.

Reality, simulation, and virtual reality

Reality is the starting point of our first application. Simulation should compute what would happen in reality. The design/optimization computation is simulation; the second contact with reality comes when the object is manufactured and used.

Virtual reality is a form of simulation that is a direct extension of the design system. Whatever is involved in the virtual reality environment is simulated well enough that the human sensory inputs are (nearly) the same as for reality. As this methodology advances, it will involve all the human senses, not just vision. And, as more senses are involved, the simulation must be more complete. For example, walls in a building now are simulated by idealized planes with color and texture superimposed. When sound and touch are included, then the walls must simulate much of the physical structure of real walls. Virtual environments are similar to virtual reality in that everything must seem "real" to the humans in the environment. However, they may combine many actual objects with simulation. For example, in pilot-training simulators the cockpit is real but the motion, the rest of the airplane, and the views out the windows are simulated. A virtual environment may also be completely or partly artificial. For example, one could be that of a "boat" navigating through the bloodstream of a person or through the molten materials inside a blast furnace. A completely artificial virtual environment could be based on a pseudophysical representation of the flow of money and other financial instruments in the economy of a city. Then a person in the environment could directly "observe" these flows as the economy changes.

Within two decades vitual reality and environments will provide very high levels of realism using accurate and rather complete simulations of the physical (or pseudophysical) environment. This will draw on all the subareas of computing that the design of physical objects involved, and with more demanding performance for most of them.

Robots

Finally, consider robots. Robots with reasonable speech and vision capabilities will appear within 20 years. Their movement and touch capabilities will be useful. Their capabilities to access information and do computations will be enormous. None of these capabilities will be anywhere close to those of humans, but that is not necessary for them to be very useful. Recall that a frog sees only in black and white, and sees only things that are moving. In spite of this primitive vision system, frogs get along quite well. Robots will also.

The computational problems for robots are much more difficult than for virtual reality. Compare the requirements for walking down a hall, by a robot and by a human within a virtual reality system. The principal activities are as follows:

- *Control.* In virtual reality, a person uses natural (existing human) control mechanisms for balance, path determination, moving, and associated activities, while walking. A robot must compute its path after recognizing the environment and then control its motion in order to follow this path.

- *Vision.* In virtual reality, a person sees a scene created from a known data structure which is designed to make scene display efficient. A robot must observe an arbitrary scene and identify the major components (walls, doors, obstacles, stairs). Such scene-analysis computations have proven to be one of the more difficult challenges for computing research.

- *Sound.* In virtual reality, a person hears sounds created from a known data structure or simply recorded previously. A robot must analyze the sounds and extract the important components, such as speech, footsteps, objects colliding, and direction.

- *Touch.* In virtual reality, a person senses objects from forces created from a known data structure. To compute and deliver these forces accurately is a major computational and mechanical challenge. A robot must have tactile sensory devices and must be able to interpret their input in terms of its environment and objectives. This is an even greater challenge.

These four areas of robot capability are currently in different states of development. The mechanical control and motion problems have been studied for a long time and much progress has been made. While there are still unsolved problems, this does not appear to be a major hurdle. The vision problems also have been studied for a long time, but with less progress. It is plausible that vision requires more computational power than previously expected, and that more rapid progress can be made with continued increases in this power. The auditory problems have also been studied for some time, prinarily in the context of speech recognition. This effort has been much less than for vision but it appears that speech recognition, at least, is now feasible. It is no doubt a major challenge to extend this technology to general sound analysis. The problem of touch seems to be much less studied. We can, however, hope that useful robots can be made with primitive touch capabilities.

I have focused on the high-level computational problems of creating robots but, just as in the previous applications, there is a large, complex substructure based on the subareas of computing. A robot will be controlled by a network of powerful processors with a specialized operating system, databases, distributed control, knowledgebases, and semi-autonomous processes.

The focus on "strategic" research is not likely to be a passing fad, even though the final definition of the term is still unclear. Fortunately for computer scientists, almost all areas of computing research are applicable to strategic applications, often in the context of the extraordinarily varied universe of computational science and engineering. Many computing researchers may have to put

some effort into establishing the connection of their work to the larger world, but this task is just part of living in an ever-changing, dynamic society. This position paper is based on [5].

References

[1] National Science and Technology Council, *Fact Booklet*, The White House, Washington, D.C., 1994.

[2] J.R. Rice, "Is Computing Research an Isolated Science?", *Computing Research News*, Vol. 2, No. 2, 1990, p. 1.

[3] C.M. Hoffmann et al., "Softlab—A Virtual Laboratory for Computational Science," *Mathematics and Computers in Simulation*, Vol. 36, 1994, pp. 479–491.

[4] E. Gallopoulos, E. Houstis, and J.R. Rice, "Computer as Thinker/Doer: Problem-Solving Environments for Computational Science," *IEEE Computational Science & Engineering*, Vol. 1, No. 2, Summer 1994, pp. 11–23.

[5] J.R. Rice, "Computational Science and the Future of Computing Research," *IEEE Computational Science & Engineering*, Vol. 2, No. 4, Winter 1995, pp. 35–41.