Using Artificial Intelligence to Control Fluid Flow Computations

Andrew Gelsey
Computer Science Department
Rutgers University
New Brunswick, NJ 08903
gelsey@cs.rutgers.edu

Abstract

Computational simulation is an essential tool for the prediction of fluid flow. Many powerful simulation programs exist today. However, using these programs to reliably analyze fluid flow and other physical situations requires considerable human effort and expertise to set up a simulation, determine whether the output makes sense, and repeatedly run the simulation with different inputs until a satisfactory result is achieved. Automating this process is not only of considerable practical importance but will also significantly advance basic AI research in reasoning about the physical world.

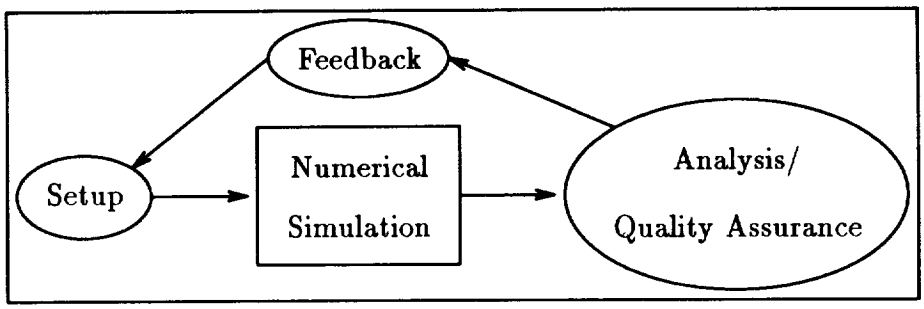


Figure 1: Analyzing a physical situation

1 Introduction

Numerical simulation is an important tool for predicting the behavior of physical systems. Many powerful numerical simulation programs exist today. However, as illustrated in Figure 1, using these programs to reliably analyze a physical situation requires considerable human effort and expertise to

- set up the simulation by transforming a description of the physical situation into a representation the numerical simulation program can successfully process,
- analyze the output of the simulation program to extract desired information and in particular to
- determine whether the output makes sense and how accurate it is likely to be, and if the output is not acceptable, to
- determine how to change the simulation program's input so that it will produce better output.

As a result, these numerical simulation programs typically can't be run successfully by inexperienced users. Perhaps more importantly, these simulation programs can't be reliably invoked by other programs. For example, an automated system for designing complex objects could not easily include a numerical simulation as part of the process it uses to evaluate new designs.

In this paper I will address these problems of setup, analysis, quality assurance, and feedback for numerical simulation.

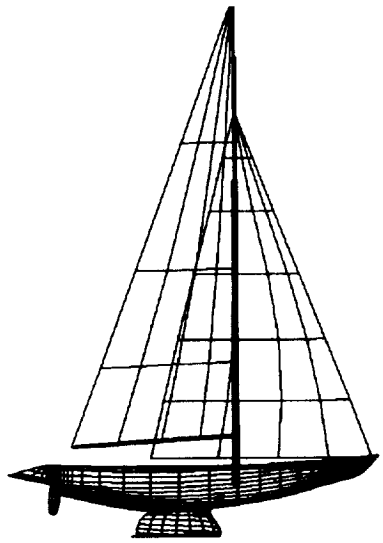
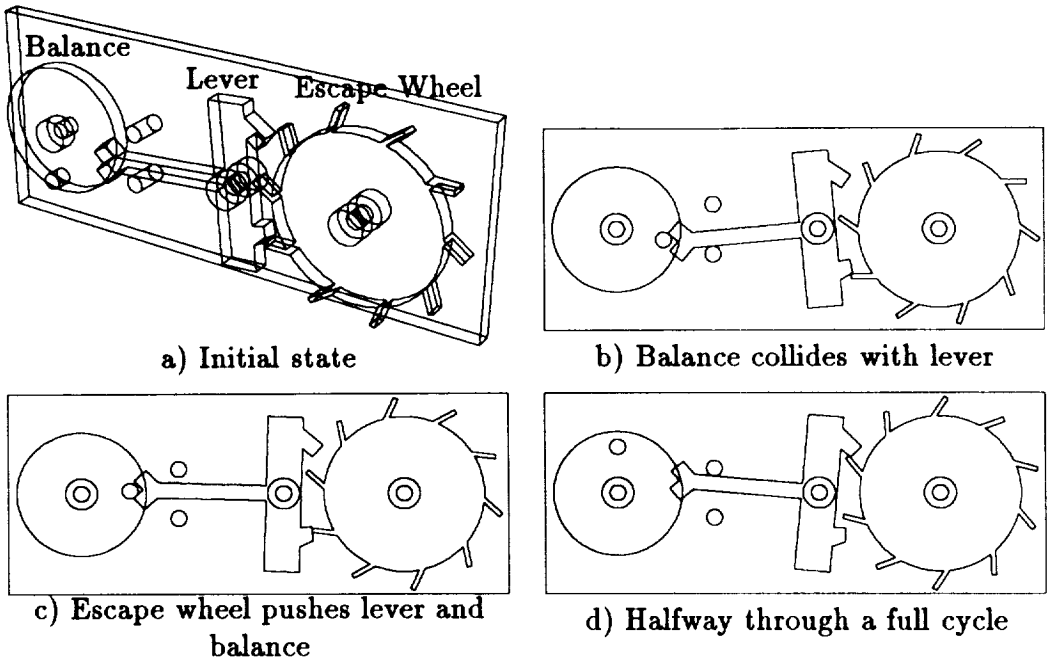


Figure 2: *Stars & Stripes*, winner of the 1987 America's Cup competition

2 Example 1: Racing Yachts

Much of the work on automated interfaces to numerical simulators that I will discuss in this paper has been done in the context of a larger system for automated design of racing yachts like the one in Figure 2. In order to evaluate a possible yacht design, the design system must determine how much time the yacht will need to traverse a specified race course, which will depend on the various forces acting on the yacht. Some of these forces can be computed quite accurately with simple formulas: for example, much of the drag on the yacht is due to an effect known as skin friction, which is directly proportional to the surface area of the yacht. Other forces, however, can only be computed with sufficient accuracy by using powerful numerical simulators. One force in this category is *lift-induced drag*, the main example I will discuss in this paper.

The force the wind exerts on a yacht can be decomposed into two components, one oriented in the direction the yacht is moving and the other perpendicular to the direction of motion. The perpendicular force must be balanced by a force from the yacht's keel, which acts as a lifting surface just like an airplane's wing does, except that for the yacht the lift force is horizontal instead of vertical. The physical effect which generates this lift force also necessarily generates a corresponding drag force, lift-induced drag, which can significantly



a) Initial state

b) Balance collides with lever

c) Escape wheel pushes lever and balance

d) Halfway through a full cycle

Figure 3: Clock or watch escapement mechanism

affect a yacht's performance.

In order to tractably compute this force, a number of modeling assumptions must be made. The key assumption is that viscosity may be neglected when computing lift-induced drag, because viscous contributions to drag will be modeled in other ways, for example as skin friction. If viscosity is neglected, the general nonlinear Navier-Stokes equations of fluid flow may be reduced to a linear partial differential equation, Laplace's equation for velocity potential

$$\nabla^2 \phi = 0$$

The velocity vector of the fluid at any point is given by the gradient of the scalar velocity potential $\phi(x, y, z)$.

3 Example 2: Clockwork Mechanisms

The escapement mechanism in Figure 3 keeps the average speed of a clock or watch constant by allowing the escape wheel, which is pushed clockwise by a strong spring, to advance by only one tooth for each oscillation of the balance.

I've discussed this mechanism in prior publications [Gelsey 1987, Gelsey 1989, Gelsey 1990, Gelsey 1991]. Modeling the behavior of this mechanism requires a series of assumptions. The first assumption is that Newton's laws of motion are adequate for this domain. Newton's laws assume forces are computed using other methods, and the remaining modeling assumptions are models for the various forces. In particular, I've explored two alternative ways of modeling collision forces: a microscopic model in which solid objects are allowed to overlap slightly in space, generating a repulsive force proportional to their depth of overlap (the depth of overlap approximates the slight distortion of the part that is the actual source of the contact force), and a macroscopic model in which moving parts are assumed to collide inelastically and move under geometric constraints from then on. Both models predict similar behaviors for the escapement mechanism.

4 Setting up a Numerical Simulation

A model of a physical situation suitable for numerical simulation must include a set of state variables, quantities whose values represent the state of the physical system. The model must also include a set of differential equations describing the relations between the values of the various state variables. The particular equations for a specific situation must be instantiated from general template equations like Newton's laws or Laplace's equation. The instantiated equations will refer to a specific set of state variables, for example the positions and velocities of the three moving parts of the escapement mechanism in Figure 3, and therefore may be solved by a numerical method which will then compute the values of the state variables for the specific situation being modeled.

I've discussed the problem of identifying state variables and differential equations for clockwork mechanisms extensively in prior publications [Gelsey 1987, Gelsey 1989, Gelsey 1990], so I'll focus here on the fluid flow domain. The state of the fluid is represented by the value of the velocity potential $\phi(x, y, z)$ at every point. For a computer simulation, however, ϕ must be represented by its value at a finite set of discrete points. Picking these points is the principal problem of setting up a numerical simulation of fluid flow. PMARC[1], the numerical simulation program we use to compute lift-induced drag, reformulates Laplace's equation as an integral equation, so instead of

[1]Panel Method Ames Research Center

having to choose points throughout the volume of fluid surrounding the racing yacht, it is only necessary to chose points on the surface of the yacht, by partitioning the surface into a set of pieces called panels.

It is important to emphasize that the initial description of a physical situation to be analyzed typically does not identify state variables. In the case both of mechanical devices and fluid flow, the input consists primarily of a purely geometrical description supplemented by information about various other physical properties like masses of parts in a mechanical device or density of fluid in a flow problem. Currently we represent mechanical device geometry using CAD/CAM Constructive Solid Geometry techniques [Requicha 1980], while ship geometry is represented using B-spline surfaces [Rogers and Adams 1990]. These geometrical descriptions must then be partitioned into pieces corresponding to different state variables.

We have received considerable advice about how to create "good" panelizations of a surface from the domain experts with whom we are working, Drs. Fritts, Salvesen, and Letcher, all members of the design team for the *Stars & Stripes* yacht shown in Figure 2. The principle guidelines we have been given include

- Resolve stagnation points and other areas of rapidly changing potential. The fluid flow hitting a solid body must change its direction in order to pass around one side of the body or the other side. As a result, there are *stagnation points* on the surface of the body where fluid velocity is zero. Since the velocity potential changes rapidly near points like these, small panels must be used in those areas to properly represent the state of the fluid flow.

- Attempt to maintain a nearly orthogonal mesh of panels. Panels should resemble rectangles rather than long, thin diamonds.

- Attempt to keep panel aspect ratio low. Panels should not be far longer than they are wide.

- Control panel expansion ratio. Bound the ratio of the areas of adjacent panels so that very small panels won't have very large neighbors.

- Attempt to orient panel boundaries along streamlines of fluid flow.

Managing the tradeoffs between these goals can be a complex reasoning problem. For example, if velocity potential near stagnation points changes rapidly

in one direction but remains fairly constant in the other, resolution of the stagnation point may best be achieved with long, thin panels which tend to violate aspect ratio constraints. In some cases constraints need to be applied more and less stringently in different areas of the body's surface. For example, a rounded body can't have an extremely orthogonal mesh everywhere, but orthogonality should be maintained as widely as possible.

5 Quality Assurance

A major difficulty in using numerical simulation programs, especially in an automated manner, is that it is generally difficult to tell whether or not to trust the output of the program. A human expert building a model of physics on which to base a numerical simulator makes use of a body of knowledge and assumptions which does not end up in the simulation program itself. For example, PMARC is based on the assumption that viscosity safely can be neglected. This assumption tends to hold much better for long, thin shapes than for short, fat ones. However, PMARC can be run to predict flow around a short, fat shape, and to an inexperienced user the results will look perfectly reasonable, until compared with experimental data, if any is available.

An automated solution to this "quality assurance" problem requires representing and using this body of knowledge and assumptions, which can be broadly classified into the following categories

- Modeling assumptions, e.g. PMARC assumes zero viscosity — these assumptions may or may not be directly testable. An example of a testable assumption from my microscopic model of collision forces in mechanisms is the assumption that the volumes of moving parts will overlap only slightly. If a large overlap occurred during a simulation that would indicate that the simulation output was not trustworthy.

- Expectations about input, e.g. long, thin body — more directly testable properties of the input typically derived from the simplifying assumptions, often by a fairly complex chain of reasoning.

- Expectations about output

  - ranges of plausible output values, e.g. velocities within fluid flowing around a body shouldn't be greater than several times the body's velocity with respect to the surrounding fluid.

  - consistency requirements for output values, e.g. the sum of the kinetic and potential energy in a clockwork mechanism should not increase as time passes
  - qualitative relationships among output values, e.g. fluid flow around a solid body should have a stagnation point where the flow hits the body, then should speed up as the fluid passes around the body again and then slow down again as the fluid merges together and leaves the body.
  - convergence behavior — as space and time discretizations are refined, differential equation solutions should settle down to asymptotic values
  - sensitivity to minor input perturbations — typically, small perturbations should generate small responses
  - response to extreme input values — may be computable from overall trends
  - trusted output results from similar physical configurations
- Test cases with known solutions
- Associated simpler models useful for coarse checking of output
- Associated deeper models to use if satisfactory output cannot be achieved
- Alternative numerical methods that might be appropriate

Some of these criteria could be better applied given a database of past experience. For example, values of lift and drag forces for similar sailing yachts could be used to reject implausible values resulting from a bad simulation.

6 Feedback

Successful analysis of a physical situation will typically require running a numerical simulator several times and intelligently modifying the simulator's input between each run. In the case of fluid flow, the heuristic nature of the guidelines for producing a good panelization make it likely that the first attempt will yield an unacceptable solution. Quality assurance knowledge must then be applied to recognize problems in the solution and attempt to diagnose and correct flaws in the initial panelization that led to these problems. In the

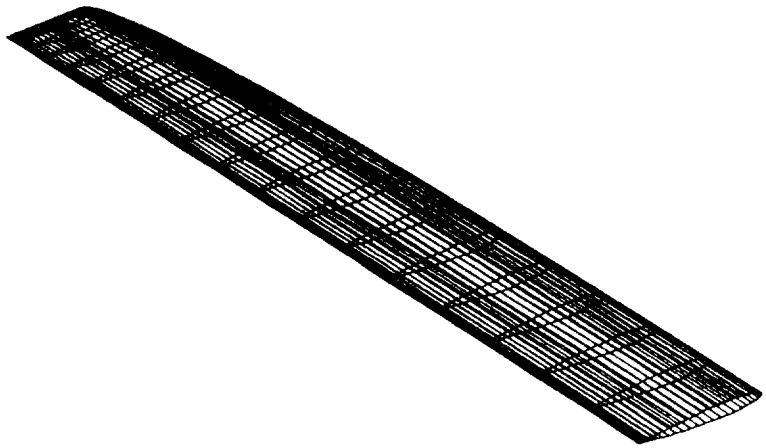


Figure 4: NACA 0012 wing

mechanics domain, feedback between simulation runs is needed not so much to change state variables as to change initial simulation conditions in order to more fully understand the physical situation (see [Gelsey 1991]).

7 Experimental Results

In the fluid flow domain, my colleagues and I have implemented initial versions of automated setup, quality assurance, and feedback for the PMARC numerical simulator mentioned earlier. The resulting system will accept a description of a solid body's geometry and automatically panelize the body's surface and run PMARC on it. However, this initial system does not directly apply the panelization guidelines discussed earlier; instead, it generates panelizations based on a distribution scheme called cosine spacing which tends to create panelizations following the guidelines, at least for fairly simple shapes. At present we have implemented quality assurance checks for most of the panelization guidelines and are in the process of investigating where on a body's surface the various guidelines tend to be most important and how tradeoffs between the guidelines should be handled.

Our automated simulation controller is capable of running a series of nu-

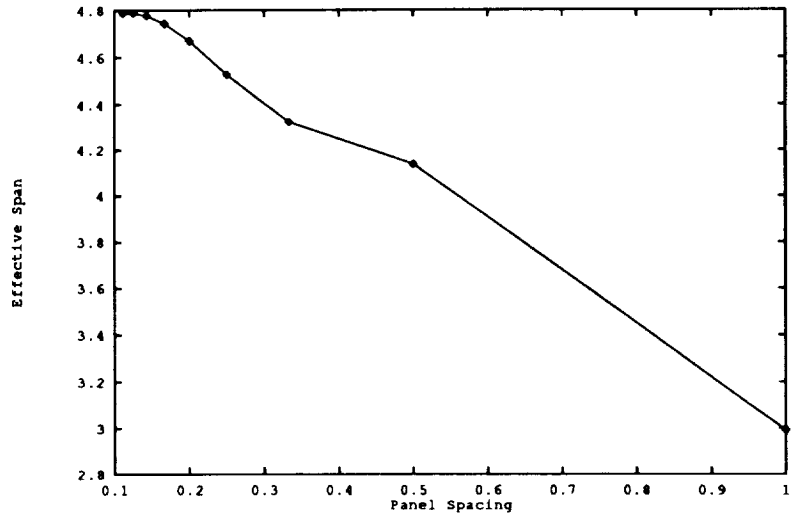


Figure 5: PMARC output for various panelizations

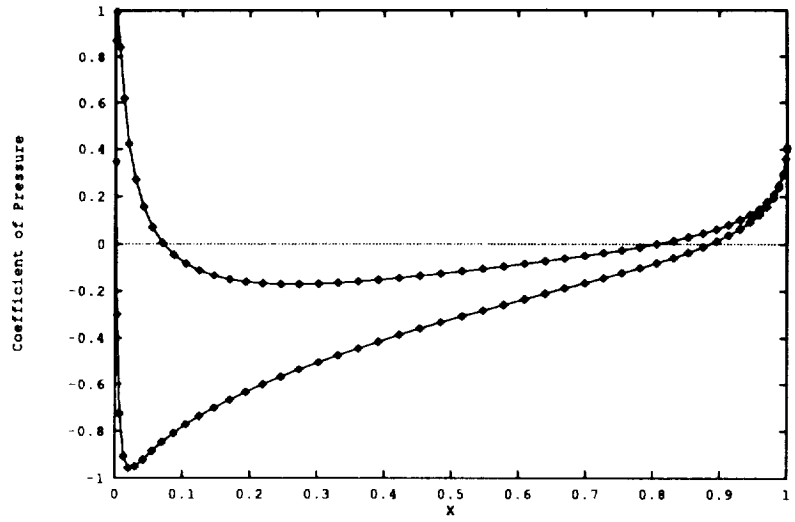


Figure 6: Pressure distribution for fluid flowing in the positive x direction. (Upper curve shows pressure on bottom of wing, lower curve shows pressure on top of wing.)

merical simulations with different panelizations. For example, Figure 5 plots PMARC output as a function of panel spacing for the simple test body in Figure 4. The largest spacing, normalized to 1, was an experiment with only 30 panels on the entire wing, while the smallest spacing, 0.1, covered the wing with 3000 panels. Note that as panel spacing decreases, the output converges to a particular value, as one would expect a reasonable solution to do. The output value here is effective span, the span of an idealized wing with similar performance. The wing in Figure 4 has a span of 5 meters, and the effective span in Figure 5 converges to a slightly lower value, as one would expect because the losses due to spillover between the upper and lower wing surfaces at the wing tip would not be present in an idealized wing. Figure 6 shows the pressure distribution along near the center of the wing as computed by the simulation with finest spacing. After Laplace's equation is solved to compute velocity potential, the coefficient of pressure plotted in Figure 6 is computed using Bernoulli's equation

$$c_p = 1 - \left(\frac{v}{v_{\text{free stream}}}\right)^2$$

Note that the pressure coefficient rises to 1, its maximum possible value, at the leading edge, indicating a stagnation point, and that the qualitative shape of the pressure distribution follows the quality assurance guidelines.

We are experimenting with various feedback schemes. We have tried simple schemes for redistributing panels to refine solution values, but have found that these don't work well. We are in the process of implementing a more sophisticated scheme which will make a more active use of panelization guidelines and quality assurance knowledge.

In the mechanical device domain I've implemented programs to automatically set up and intelligently control numerical simulations of clockwork mechanisms. [Gelsey 1987, Gelsey 1989, Gelsey 1990, Gelsey 1991]. I am currently focusing on the problem of quality assurance for this domain.

8 Related Work

Jambunathan *et al.*[1991] and Andrews[1988] discuss the use of expert systems technology to augment more traditional computational fluid dynamics programs. Most other artificial intelligence research concerning reasoning about physical systems has focused on qualitative rather than numerical simulation. [Weld and de Kleer 1990] Exceptions are the work of Sacks[1991] and Yip[1991];

however, they have focused on numerical simulators for ordinary differential equations and have not addressed the issue of quality assurance. Forbus and Falkenhainer[1990] discusses the use of qualitative simulation to check the quality of numerical simulation results; however, the approach described appears limited to physical situations modeled by ordinary differential equations.

9 Conclusion

While the problems of setup, analysis, quality assurance, and feedback for numerical simulation are clearly of considerable practical importance, their solution should also lead to significant insights into some basic AI problems. Successful use of numerical simulators appears to require a combination of spatial reasoning and reasoning about physics which has so far received little attention from AI researchers. The first step in classifying and understanding these reasoning methods is to collect a set of concrete examples of them, and the only way to be sure the examples fully capture the reasoning used in to incorporate them in programs which can successfully apply the reasoning to the control of numerical simulations. This is the goal of the research discussed in this paper.

10 Acknowledgments

The research on automated use of PMARC was done with fellow Rutgers Computer Science Dept. faculty member Gerard Richter and graduate student Ke-Thia Yao. We worked with hydrodynamicists Martin Fritts and Nils Salvesen of Science Applications International Corp., and John Letcher of Aero-Hydro Inc. Our research was supported by the Defense Advanced Research Projects Agency and the National Aeronautics and Space Administration under NASA grant NAG2-645.

References

[Andrews 1988] Andrews, Alison E. 1988. Progress and challenges in the application of artificial intelligence to computational fluid dynamics. *AIAA Jounal* 26(1):40–46.

[Forbus and Falkenhainer 1990] Forbus, Kenneth D. and Falkenhainer, Brian 1990. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *Proceedings, Eighth National Conference on Artificial Intelligence*, Boston, MA. AAAI-90.

[Gelsey 1987] Gelsey, Andrew 1987. Automated reasoning about machine geometry and kinematics. In *Proceedings of the Third IEEE Conference on Artificial Intelligence Applications*, Orlando, Florida. Also appears in Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, Morgan Kaufmann, 1990.

[Gelsey 1989] Gelsey, Andrew 1989. Automated physical modeling. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, Detroit, Michigan USA.

[Gelsey 1990] Gelsey, Andrew 1990. *Automated Reasoning about Machines.* Ph.D. Dissertation, Yale University. YALEU/CSD/RR#785.

[Gelsey 1991] Gelsey, Andrew 1991. Using intelligently controlled simulation to predict a machine's long-term behavior. In *Proceedings, Ninth National Conference on Artificial Intelligence.*

[Jambunathan *et al.* 1991] Jambunathan, K.; Lai, E.; Hartle, S. L.; and Button, B. L. 1991. Development of an intelligent front-end for a computational fluid dynamics package. *Artificial Intelligence in Engineering* 6(1):27–35.

[Requicha 1980] Requicha, Aristides A. G. 1980. Representations for rigid solids: Theory, methods, and systems. *ACM Computing Surveys* 12:437–464.

[Rogers and Adams 1990] Rogers, David F. and Adams, J. Alan 1990. *Mathematical elements for computer graphics.* McGraw-Hill, 2nd edition.

[Sacks 1991] Sacks, Elisha P. 1991. Automatic analysis of one-parameter ordinary differential equations by intelligent numeric simulation. *Artificial Intelligence* 48(1).

[Weld and de Kleer 1990] Weld, Daniel S. and Kleer, Johande, editors 1990. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann, San Mateo, California.

[Yip 1991] Yip, Kenneth 1991. Understanding complex dynamics by visual and symbolic reasoning. *Artificial Intelligence* 51(1–3).