Intelligent Automated Quality Control for Computational Simulation

Andrew Gelsey
gelsey@cs.rutgers.edu

CAP-TR-21
August 1994

Abstract

Computational simulation of physical systems generally requires human experts to set up a simulation, run it, evaluate the quality of the simulation output, and repeatedly invoke the simulator with modified input until a satisfactory output quality is achieved. This reliance on human experts makes use of simulators by other programs difficult and unreliable, though invocation of simulators by other programs is critical for important tasks such as automated engineering design optimization. I present a framework for constructing intelligent controllers for computational simulators which can automatically detect a wide variety of problems which lead to low-quality simulation output, using a set of evaluation methods based on knowledge of physics and numerical analysis stored in a data/knowledge base of models and simulations. I describe an experimental implementation of this framework in an intelligent automated controller for a widely used computational fluid dynamics simulator.

Computer Science Department
Rutgers University
New Brunswick, NJ 08903

Contents

1 Introduction **2**

2 Theory **3**
2.1 Evaluation Methods Using Knowledge of Relevant Physics . . 4
2.2 Evaluation Methods Using Knowledge of Relevant Numerical Analysis 9
2.3 Feedback 11

3 Experimental Results **11**
3.1 MSA1 Model Base 12
3.2 MSA1 Quality Assurance 15
3.3 MSA1 feedback 18

4 Is Quality Assurance Expensive? **19**

5 Related Work **20**

6 Future Work **21**

7 Conclusion **21**

8 Acknowledgments **22**

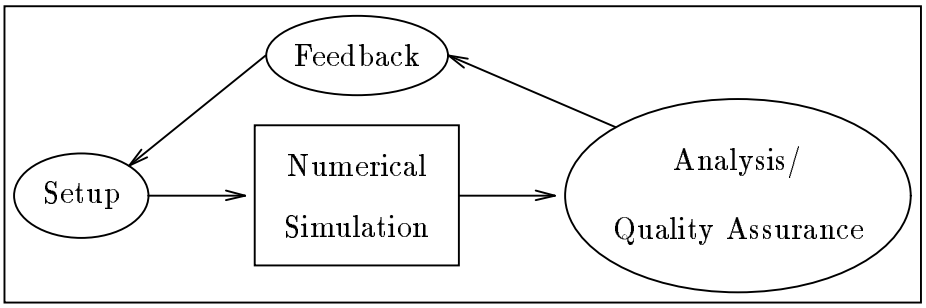

Figure 1: Typical environment for a complex numerical simulator

1 Introduction

Computational simulation of physical systems is an important scientific and engineering tool. Simulators typically exist in the environment shown in Figure 1, in which each successful simulation requires a considerable time investment by human experts. Experts are needed both to set up simulations and to analyze and understand simulation results, and achieving satisfactory output quality generally requires several iterations of the feedback loop show in Figure 1. In this loop, the expert checks whether the present simulation output is unsatisfactory, and if so changes the simulation input to try to improve the quality.

A major drawback of the simulation environment of Figure 1 is that its extensive reliance on the intervention of human experts makes use of simulators by other programs difficult and unreliable. However, invocation of simulators by other programs is critical for important tasks such as, e.g., automating the process of engineering design optimization [Vanderplaats 1984].

I present a framework for constructing intelligent controllers for computational simulators which can automatically detect a wide variety of problems which lead to low-quality simulation output. These intelligent controllers make use of a set of evaluation methods based on knowledge of physics and numerical analysis, which is stored in a data/knowledge base of models and simulations ("model base"). The use of such intelligent controllers overcomes to a large extent the drawback mentioned above: since most or all cases of low-quality simulation output are detected automatically, other programs can reliably invoke computational simulators without the need for human intervention.

I also describe an experimental implementation of this framework called MSA1 (Model/Simulation Associate 1). MSA1 is an intelligent automated controller for PMARC, a widely used computational fluid dynamics simulator developed at NASA Ames Research Center. MSA1 is automatically invoked by an automated design system for racing yachts to compute the efficiency of a yacht's keel.

2 Theory

A complex simulation can produce unacceptable output in many ways, some quite obvious and some fairly subtle. Successful automated quality assurance requires simultaneous use of an assortment of different methods for evaluating the quality of simulation output. In this section, I will discuss the following methods for automatically evaluating simulation quality:

1. Evaluation methods using knowledge of relevant physics
   (a) hierarchies of models at different levels of approximation to evaluate physical plausibility of both simulation output and simulation input[1]
   (b) checking compatibility of physical situation being simulated with assumptions and approximations on which simulator is based
   (c) simulator validation with test cases which are physical situations for which simpler models give exact results
   (d) comparison of solution features with previous simulation results for similar physical situations
   (e) estimates of modeling error due to incompletely modeled physical phenomena
2. Evaluation methods using knowledge of relevant numerical analysis
   (a) automated convergence studies for

[1]In this report, I use the term "model" to refer to a model of physical phenomena rather than, for example, a model of the geometry of a physical situation. An example of a model for certain fluid flow phenomena is the combination of the Bernoulli equation, the semantic interpretation of the variables in the equation, and the assumptions and approximations on which the equation is based.

i. values of direct interest (i.e., desired output)

ii. auxiliary intermediate values

(b) estimates of numerical error and resolution for values of direct interest and auxiliary intermediate values

(c) suitability of simulation input for numerical algorithms being used (e.g., grid quality)

(d) condition numbers for linear systems

A data/knowledge base of models and simulations ("model base"), as shown in Figure 2, is a useful way to store the information that the intelligent simulation controller needs to apply these quality assurance methods. Each problem type in the model base of Figure 2 has associated with it a set of test cases and experiment results, and a hierarchy of relevant models at differing levels of approximation. Each model in the hierarchy has an associated simulator, previous results from using this simulator, tests for the assumptions and approximations underlying the simulator, and information about the simulator's numerical algorithms. All of these elements of the model base are described in more detail in the next section.

2.1 Evaluation Methods Using Knowledge of Relevant Physics

In the "traditional" simulation environment of Figure 1, simulation quality is evaluated by human experts. An engineer or computational scientist using a computer simulation, especially an unfamiliar one, will often compare the simulation results to some sort of "back of the envelope" calculation. The human user may also compare the simulation results to the output of some other less complex computational simulation for the same physical situation. Formalizing and automating this process provides one important method for automated simulation quality evaluation.

The back-of-the-envelope calculation an expert uses to check a simulation is based on a model of the same physical situation, but the back-of-the-envelope model is much more approximate than the model on which the computational simulation is based. A less complex computational simulation for the physical situation, if available, will also be based on a model that is

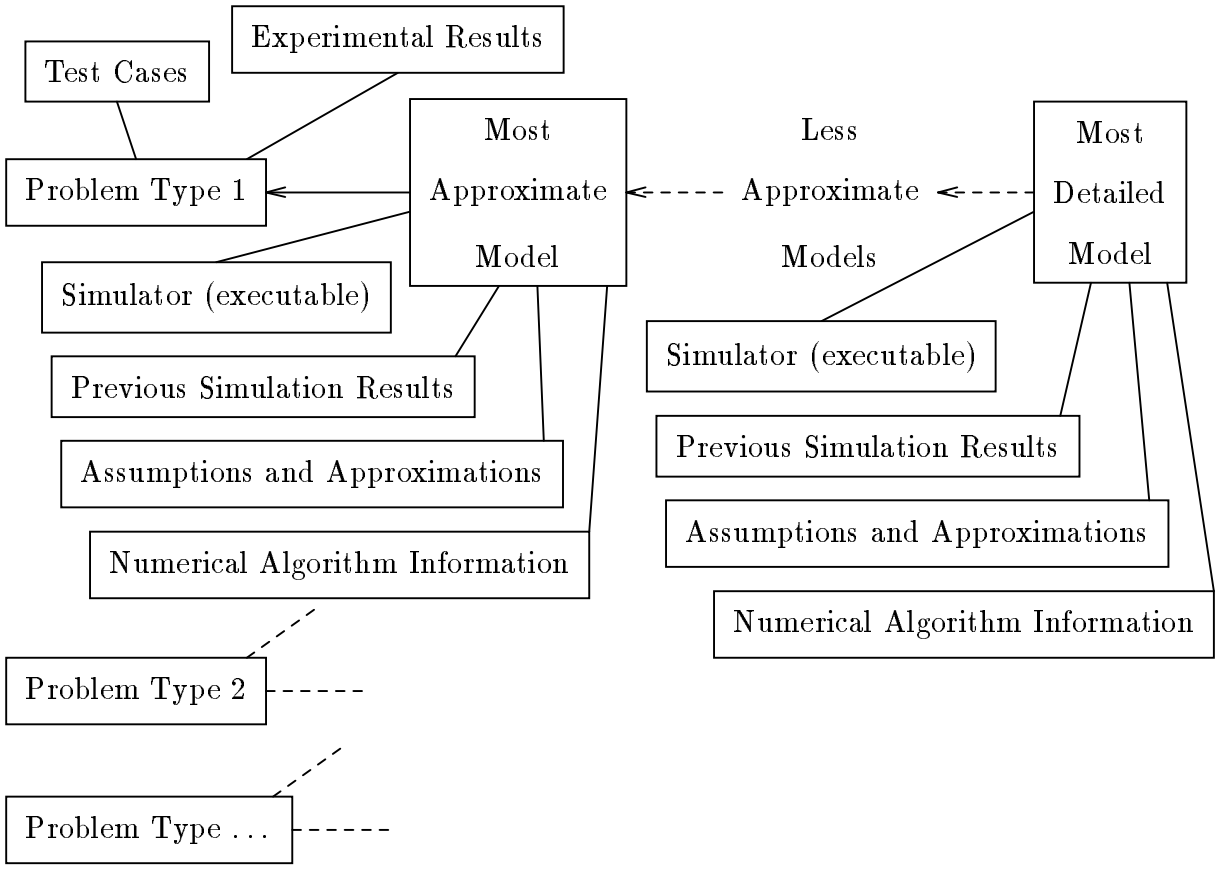

Figure 2: Data/knowledge base of models and simulations ("model base")

more approximate than the most detailed simulation, though less approximate than the back-of-the-envelope calculation. These various models can be organized into a hierarchy of models ordered by their level of approximation, as shown in the model base of Figure 2. Each hierarchy of models is associated with a problem type, a class of similar physical situations for which the models in the hierarchy are appropriate. An example of a problem type is that used later in this report in the experimental results section: physical situations involving steady fluid flow past a streamlined body possibly including lifting surfaces, with a Mach number near zero and a high Reynolds number.

Such an approximation hierarchy of models can be a useful tool for simu-

lation quality assurance, because each model in the hierarchy can be used to check that the predictions of the less approximate models in the hierarchy are within reasonable bounds. More complex models have more ways to fail and thus tend to be less robust than simpler models, so a reasonable heuristic is to be suspicious of a complex model whose output falls outside the error range for a simpler model of the same problem. To use this heuristic as part of an automated system for simulation quality assurance requires computational implementations of the models (as shown in Figure 2) so that the intelligent simulation controller can simultaneously apply all of the relevant models to each physical situation.

Using a more approximate model to check the output of a less approximate model may at first seem counterintuitive: by definition, shouldn't a "less approximate" model produce superior output? This seeming paradox is explained by recognizing that the approximation hierarchy is an ordering of the **nominal** models, not of their actual software implementations. If an implementation of a nominally "less approximate" model is producing lower quality output than it should, its output may actually be much more approximate than that of a nominally more approximate model — in fact, its output may be completely wrong.

There are a number of ways in which a model A can be "more approximate than" another model M:

1. A may include additional simplifying assumptions. If all of the assumption are satisfied by a particular situation, then the predictions of A have no more error than the predictions of M.

2. A may include additional simplifying assumptions which will never be perfectly satisfied. This is quite a common form of approximation, and A may be quite useful even though its assumptions are violated, as long as the errors that result are small enough. For example, by introducing the assumption of an incompressible, inviscid, irrotational fluid, the Navier-Stokes equations of fluid flow may be simplified to the more tractable equations of potential flow [Newman 1977]. However, no real fluid will ever satisfy these assumptions perfectly. Nevertheless, effects of deviations from these assumptions are frequently small enough to be neglected, so potential flow is often a very useful model.

3. A and M may be based on the same set of assumptions, but A may

involve a simpler set of calculations and return results with a larger expected error.

Any of these forms of approximation is acceptable for quality assurance. The key requirement for a model hierarchy used for quality assurance is that more approximate models should also be more robust, so that they are likely to be useful checks on the quality of the less approximate models.

The "model base" of Figure 2 also includes a representation of the assumptions and approximations on which each model is based. A common cause of poor or meaningless simulation results is the use of a simulator for analyzing a physical situation which is not compatible with the simulator's underlying assumptions. Though these assumptions are known to the simulation developer, they are almost never explicitly represented in the resulting simulator. This lack may be remedied by associating simulation codes with model representations including information about underlying assumptions and approximations. To be useful as part of an automated system for simulation quality assurance, this information must be operational: procedures must exist in the system which can use the information to actually check that the physical situation to be analyzed is compatible with the assumptions and approximations underlying the simulator.

Complex simulators should be periodically validated by applying them to test cases for which simpler models in the same model hierarchy give exact results. For example, a simulator which predicts forces on lifting bodies in fluid flow might be tested on examples like a sphere, which should have zero lift, or a monoplane wing with elliptic planform, which has a simple exact relationship between lift and drag. As show in Figure 2, the test cases stored in the model base are associated with a problem type, so that all models in the hierarchy for that problem type can be applied to each test case. Automated revalidation is desirable in a variety of situations, for example on installing a new software release of a simulator or when an operating system kernel or run-time libraries change.

The model base of Figure 2 also includes data about results of previous simulations. A expert uses past experience when checking for possible simulation quality problems, evaluating features of the simulation output in the context of previous similar simulations. An intelligent automated simulation controller can, to some extent, imitate this capability of the human expert by comparing solution features to simulation results for similar physical sit-

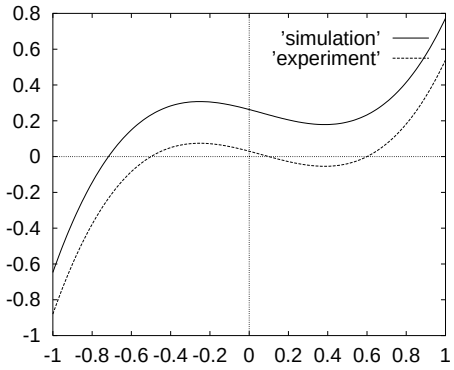

Figure 3: Recalibration example

uations as recorded in the model base. Automation of this process requires implementation of feature extractors, which are procedures for extracting features from simulation output, and feature evaluators, which are procedures for evaluating plausibility of extracted solution features by comparing them to the model base.

A useful capability for an intelligent simulation controller is the estimation of the accuracy of simulation results. The next section of this report discusses estimating error for the numerical algorithms used by the simulator. However, even if numerical computations are done quite accurately, simulation results may have significant inaccuracies due to the approximations of the model which the simulation executes. To assist the intelligent controller in estimating this modeling error, the model base of Figure 2 includes experimental results for each problem type. The modeling error for the various models in the hierarchy associated with a particular problem type may be estimated by applying each model to the same physical situations for which experimental results are stored in the model base. Note that relative errors may be more important than absolute errors: if the predictions of a simulator differ significantly from experimental results, but if both results change by similar amounts in response to changes in the input situation, then the experimental data may be used to "recalibrate" the predictions of the simulator and greatly reduce the modeling error. For example, in Figure 3 the simulation results differ significantly from the experimental results, but both change similarly as their input varies, so if the simulation output is recalibrated by adding the difference between the two results at one point,

then the recalibrated simulation output may serve as a useful predictor of the experimental results over some range of input.

2.2 Evaluation Methods Using Knowledge of Relevant Numerical Analysis

Computational simulations of physical systems often involve numerical solution of partial differential equations and thus require as input a "grid", which is a discretization of the independent variables. It is clearly desirable that simulators produce "grid-independent" results, since the particular grid employed is an artifact of the numerical computation method, while the goal of the simulation is to predict the actual behavior of the physical situation being modeled. Numerical algorithms have a "truncation error" associated with discretization, and a grid-independent simulation is one in which truncation error is sufficiently small to ensure the desired accuracy.

Truncation error for stable numerical algorithms decreases as grids are refined, becoming zero in the limit of an infinitely fine grid. If the intelligent simulation controller runs a series of simulations with grids that are identical except that the spacing of grid cells is halved on each successive run, then simulation results should form a convergent sequence. These automated "convergence studies" are a valuable tool for simulation quality assurance. Of course, computational resource limitations will generally allow only a few grid refinements, but often the number will be sufficient to identify a pattern of convergence and bound the numerical truncation error. A single convergence study including a series of grid refinements may produce several (hopefully) converging sequences, since both values of direct interest (i.e., desired output) and auxiliary intermediate values should be converging.

The most basic use for an automated convergence study is to verify that simulation output values are in fact converging and therefore at least somewhat trustworthy. Assuming that this basic test has been passed, the convergence study results can also be used by the automated simulation controller to estimate numerical error and resolution both for values of direct interest and for auxiliary intermediate values. The rate at which truncation error for a numerical algorithm converges to zero is generally at least $O(h)$, where h is the grid spacing. Therefore, if h is halved in each successive simulation run in the convergence study, the truncation error for each simulation run

should be no more than half of the truncation error for the previous run, so the difference in output values between the two simulation runs with smallest h will be an upper bound for the truncation error of the simulation run with smallest h (the finest grid). If the convergence rate of the numerical algorithm is known or can be estimated more precisely, a tighter bound for the error can be found [Dahlquist and Bjorck 1974]. Available information about truncation error convergence rates for numerical algorithms should be stored in the model base (Figure 2).

The model base should in addition hold information about input requirements for numerical algorithms. For example, numerical algorithms are often sensitive to geometrical features of grids such as grid cell orthogonality, aspect ratio, etc. Information about such sensitivities should be stored in the model base, and in order for the simulation controller to automatically use this information for quality assurance, appropriate grid feature extractors and feature evaluators must be available.

Many numerical algorithms, particularly those for solving the partial differential equations that arise in modeling physical systems, are formulated as a three stage process:

1. set up a system of linear algebraic equations
2. solve the linear system
3. postprocess the solution

For such an algorithm, the condition of the linear system is an important consideration in evaluating quality of simulation output. The most direct way for the intelligent simulation controller to use the linear system condition number is by checking that the roundoff error associated with the condition of the linear system is significantly less than the bound on the simulation's truncation error. Of course, the precision of the floating point arithmetic must be taken into account: with quadruple precision, a condition number as high as 10^{20} would still allow 10 digits of accuracy in the result. [Dahlquist and Bjorck 1974]

An additional use for the condition number is to compare it to condition numbers from similar simulations stored in the model base. A condition number significantly higher than condition numbers from comparable simulations may indicate that the simulation has been set up or executed incorrectly, suggesting a possible simulation quality problem. If simulations that have been

correctly set up and executed nevertheless yield excessive condition numbers, typically the only remedy would be to change the numerical algorithm or to use higher precision floating point arithmetic.

2.3 Feedback

An important of element of the problem solving environment of Figure 1 is the feedback loop by which low quality simulations are improved. Clearly, not every problem detected by automated quality assurance can be corrected automatically. For example, if a low quality simulation is the result of a programming bug in the simulator, automated quality assurance can serve the very important role of detecting symptoms of the problem, but a human will be needed to actually correct the bug in the simulator program. The following methods for automated quality improvement are feasible, however:

1. using a more detailed model in the model base, if available
2. using a finer grid
3. rearranging a grid to improve resolution in lower quality areas

3 Experimental Results

I have experimentally investigated quality assurance for computational simulation output by implementing MSA1 (Model/Simulation Associate 1), an intelligent automated controller for PMARC, a widely used panel method computational fluid dynamics simulator developed by NASA Ames Research Center. MSA1 is currently used as a design evaluation module by DA (Design Associate), a system developed at Rutgers for automated design of racing yachts [Ellman *et al.* 1992]. DA applies various optimization algorithms [Vanderplaats 1984] to improve yacht designs, and at each step of the optimization DA automatically invokes MSA1 if necessary to compute the efficiency of the racing yacht's keel.

A yacht's keel is a lifting surface which generates a horizontal "lift" force perpendicular to the yacht's direction of motion. This lift force L has an

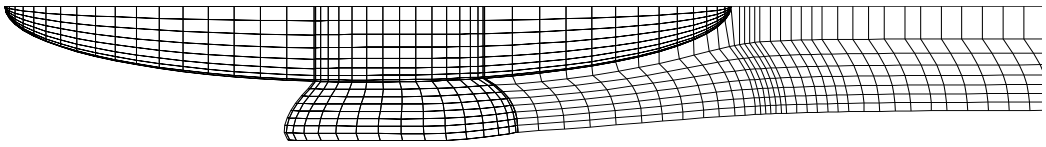

Figure 4: MSA1 panelization of yacht hull below waterline, keel, and trailing vortex wake for input to PMARC

associated lift-induced drag force D_i which has the value

$$D_i/q = \frac{(L/q)^2}{2\pi T_{\text{eff}}^2} \tag{1}$$

where $q = \rho v^2/2$ is the dynamic pressure, ρ is fluid density, v is fluid velocity relative to the yacht, and T_{eff} is the yacht's "effective draft", which remains fairly constant for a particular keel over a wide range of values of L and q. Higher values of effective draft T_{eff} result in less drag for a given lift, so T_{eff} serves as a measure of the efficiency of a yacht's keel.

MSA1 automates all parts of the problem solving environment shown in Figure 1, though in this report I focus on simulation quality assurance. DA represents yachts as B-spline surfaces, so in order for DA to automatically invoke MSA1 when it generates a new candidate yacht design, MSA1 must be able to automatically form input grids for PMARC based on these B-spline surfaces. Figure 4 shows a grid automatically generated by MSA1. [Yao and Gelsey 1994] describes our methods for automated grid generation. Section 3.3 discusses feedback in MSA1.

3.1 MSA1 Model Base

The model base in MSA1 currently has only one problem type: physical situations involving steady fluid flow past a streamlined body possibly including lifting surfaces, with a Mach number near zero and a high Reynolds number. The associated hierarchy of models at different levels of approximation presently contains four models.

The most approximate model is the qualitative model

$$D \geq 0 \tag{2}$$

indicating that drag cannot be negative. This fact may seem too obvious to be worth stating, but in fact PMARC often predicts negative drag when run on improper input data. No previous simulation results and numerical algorithm information are associated with this model. A (trivial) simulator is associated with the model. At present the list of associated assumptions and approximations is empty: we are not aware of a physical situation that violates this model.

The next model is

$$T_{\text{eff}} \leq T \tag{3}$$

(effective draft is no greater than physical draft T). This model has two associated assumptions. The first assumption is that the situation is consistent with the previous model (Equation 2) which is necessary in order that T_{eff} exist, since if drag is negative, Equation 1 which defines T_{eff} will have no physically meaningful solution. The second assumption is that the physical situation involves only one lifting surface, since additional lifting surfaces such as winglets may yield effective draft greater than physical draft. If either assumption is violated the model is not applied.

The next model is

$$T_{\text{eff}} = \sqrt{T^2 - \frac{2A_{\max}}{\pi}} \tag{4}$$

and approximates effective draft as a function of actual draft T and cross-sectional area $A_{\max}$ [Newman and Wu 1973, Letcher 1975]. This model has the same assumptions as the previous one, a (trivial) associated simulator, and data on previous simulation results which is needed to estimate how large a deviation between this model and the more detailed PMARC model is acceptable.

The most detailed model in the current MSA1 model base is the one executed by PMARC. PMARC is a potential-flow panel method, so its key underlying assumption is that the potential-flow approximation of fluid flow is applicable to the physical situation being analyzed. As mentioned earlier, "applicable" does not mean that the assumptions of potential flow are perfectly satisfied, but rather that effects of deviations from these assumptions are small enough to be neglected. Deviations from the potential flow assumption of an incompressible fluid will be small if the Mach number is low. If the Reynolds number is high and the physical situation involves only streamlined bodies, then viscous effects will be confined to a thin boundary layer, and the

potential flow model may usefully be applied outside that boundary layer.

Verifying the Mach and Reynolds number requirements is straightforward, but determining whether a particular body is actually streamlined is a difficult problem, so in the present version of MSA1 the verification of this assumption is implemented as a check that flow velocities predicted by PMARC are not significantly greater than those in the simulation results for similar situations stored in the model base. The situations stored in the model base are assumed to have been checked for plausibility by human experts, so the flow velocities should provide a reasonable scale for comparison. Violations of the streamlined-body assumption typically cause a potential flow simulation to predict very high flow velocities as streamlines are forced to follow the body instead of separating as they would in the actual physical situation.

Besides the assumptions and approximations, previous simulation results, and executable simulator, the PMARC model in the MSA1 model base also has information about the PMARC numerical algorithm associated with it. PMARC solves for a perturbation potential which is treated as constant on each panel (grid element). This constant-coefficient approximation suggests a convergence rate of $O(h)$ where h is the spacing between panels. The numerical algorithm information for PMARC also includes grid feature extractors and evaluators for the following grid features which appear to influence PMARC solution quality:

1. maximum panel aspect ratio (ratio of length of the panel's longest side to its shortest side) over all panels

2. minimum orthogonality (smallest angle between line segments meeting at a panel corner)

3. maximum noncoplanarity (angle between normal vectors of neighboring panels)

4. maximum expansion ratio (ratio of areas of neighboring panels)

In addition to the model hierarchy, the single problem type in the current MSA1 model base also has associated test cases and experimental results. The following test cases are included:

1. sphere

2. monoplane wing with elliptic planform

The experimental results in the current MSA1 model base currently consist only of a small amount of summary data comparing simulation results to experiments: detailed experimental results are not presently included. The summary data is primarily from [Letcher *et al.* 1987], a paper by the hydrodynamics experts with whom we are collaborating. Though experimental results are only needed for some of the quality assurance methods I have described, clearly a full MSA implementation would benefit from a larger collection of experimental data.

3.2 MSA1 Quality Assurance

The current MSA1 implementation includes the following quality assurance capabilities:

1. Before running simulations, the physical situation to be analyzed is checked as much as possible for compatibility with the modeling assumptions of the four models in the model base:
   (a) the automated gridding system [Yao and Gelsey 1994] is queried to determine how many lifting surfaces are present.
   (b) the Mach number is computed and checked for small magnitude (the model base specifies how small is adequate).
   (c) the Reynolds number is computed and checked for large magnitude (the model base specifies how large is adequate).
2. All models compatible with the situation to be analyzed are executed.
3. Simulation output from the various models is checked for consistency:
   (a) if PMARC predicts a negative drag, this result is inconsistent with the most approximate (and most robust) model in the model base, so the PMARC results are classified as highly suspicious
   (b) otherwise, if PMARC predicts nonnegative drag, and if only one lifting surface is present, the PMARC results are compared to the results of the other two models, with major deviations classified as suspicious

If the prediction from a more complex model fails to match that of a simpler model to within the model error range of the simpler model, MSA1 will do one or both of the following, depending on how it is invoked:

(a) print a message about the discrepancy for a human user to read

(b) return information to the program calling MSA1 indicating the problem

4. the "streamlined body" assumption for PMARC is checked for internal consistency by comparing velocity values in the PMARC solution to those stored in the model base.

5. When there is a change in a simulator or its setup or postprocessing procedures, the simulator can be reapplied to the test cases in the model base.

6. MSA1 evaluates certain solution features relative to simulation results for similar physical situations. Feature extractors and feature evaluators are presently available for the following PMARC model solution features:

   (a) minimum and maximum pressure coefficient

   (b) minimum and maximum doublet value[2]

   (c) minimum and maximum doublet jumps between neighboring panels in each direction on each surface patch

   Significant deviations of these solution features from values for past successful simulations of comparable situations indicates a faulty solution or poor resolution.

7. When the other quality assurance measures are satisfactory, MSA1 estimates the residual modeling error based on the summary data comparing experimental results to corresponding past simulation results.

[2]Doublets are singularity elements from which a PMARC potential-flow solution is constructed. [Katz and Plotkin 1991, Ashby *et al.* 1992]

panels	T_{eff}	lift	drag	min doublet	max doublet
64	2.26637	0.732586	0.0166294	-0.605920	0.522966
256	2.15254	0.670246	0.0154307	-0.628584	0.544243
1024	2.12155	0.664054	0.0155927	-0.696013	0.549740
4096	2.11605	0.668038	0.0158624	-0.702774	0.551128

Figure 5: MSA1 automated convergence study

8. MSA1 automatically runs convergence studies for the PMARC model. (The other models in the current model base do not involve discretization.) Each time PMARC grid spacing is halved, execution time is multiplied by approximately 64. Thus there is no significant overhead from running a convergence study: the entire convergence study takes only 1% or 2% longer than running PMARC only on the finest grid. The table in Figure 5 shows data from a particular convergence study run automatically by MSA1, and Figure 6 graphically illustrates the convergence of T_{eff}, the desired output, as grid spacing h is reduced by refining the grid.

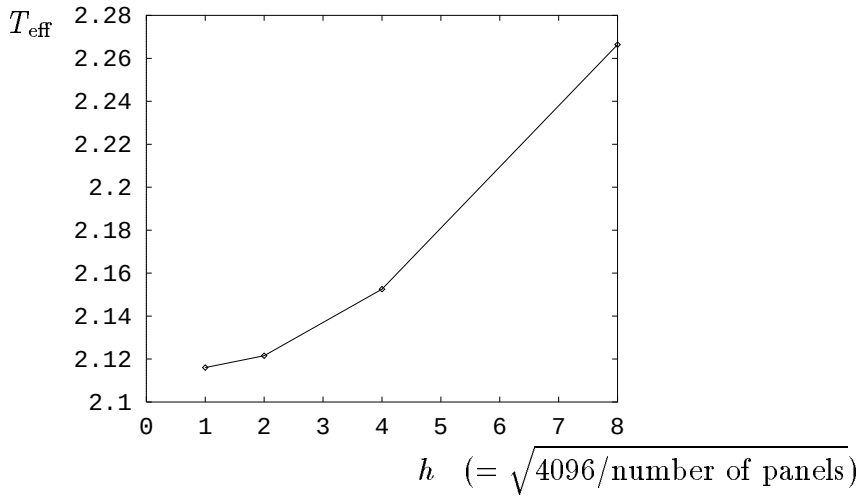

Figure 6: MSA1 automated convergence study: effective draft as a function of normalized panel spacing ($h = 1$ for the finest grid used).

hc	hs	kc	ks	j_{hc}	j_{hs}	j_{kc}	j_{ks}	T_{eff}
72	18	36	18	0.081207	0.0310633	0.0531022	0.0413898	2.11688
120	11	40	15	0.0406243	0.0411513	0.0472425	0.0461382	2.11322
116	10	46	16	0.0426034	0.0422480	0.0412613	0.0442939	2.11035
120	10	46	17	0.0400858	0.0426122	0.0413007	0.0430784	2.11020

Figure 7: MSA1 automated feedback

9. If the convergence study indicates convergence, MSA1 treats the difference in output values between the two PMARC runs with finest grids as an upper bound for the numerical error in the solution, as the convergence rate of $O(h)$ for PMARC stored in the model base does not justify a tighter error bound.

10. The PMARC input grid quality is estimated using the grid feature extractors and feature evaluators stored in the model base.

11. The condition number for the PMARC linear system is compared to condition numbers in previous simulations of comparable situations. MSA also checks that the roundoff error to be expected based on the linear system condition is significantly less than the truncation error estimated in the automated convergence study.

3.3 MSA1 feedback

MSA1 includes an automated feedback mechanism for rearranging a PMARC grid to improve resolution of the PMARC solution. PMARC solves for doublet values which are treated as constant over each panel. The difference in solution values between neighboring panels limits the resolution of the PMARC solution, so reducing the maximum over all panels of these "jumps" should improve the solution. Figure 7 shows data from the MSA1 automated feedback process, in which the gridding inputs are modified to improve the solution. The variables are as follows:

hc number of panel columns on the hull (columns are along the flow direction)

hs number of panel rows on the hull (rows are across the flow direction)

kc number of panel columns on the keel

ks number of panel rows on the keel

j_{hc} maximum jump in in doublet value between a panel on the hull and a neighboring panel in the same column

j_{hs} maximum jump in in doublet value between a panel on the hull and a neighboring panel in the same row

j_{kc} maximum jump in in doublet value between a panel on the keel and a neighboring panel in the same column

j_{ks} maximum jump in in doublet value between a panel on the keel and a neighboring panel in the same row

The feedback algorithm attempts to find new values for the gridding inputs hc, hs, kc, and ks which will maintain approximately the same total number of panels and which will reduce the largest jump by making all four jumps equal, using the approximation that each of the four new maximum jumps will equal the corresponding previous jump times the ratio of the new and old values of the corresponding gridding input variable. The largest jump on each line of the table in Figure 7 is smaller than the largest jump on the line above, as desired, and the algorithm also appears quite stable: the changes grow smaller and in fact no change is needed after the fourth line. The effect of this regridding on T_{eff}, the MSA1 output, is fairly insignificant, however.

4 Is Quality Assurance Expensive?

The following point is worth emphasizing: quality assurance is very inexpensive, *as long as* the entire simulation control and quality assurance process is fully automated. The most detailed model that is executed will generally require far more computational resources than executing the combination of all of the more approximate models in the same hierarchy. Therefore, if model setup, execution, etc. are fully automated, executing a combination of a detailed model with all of the more approximate models in its hierarchy will not be significantly more expensive than just executing the detailed

model by itself. Similarly, with an automated simulation controller, running a convergence study with several grids is not significantly more expensive than just using the finest grid by itself. However, note that if the simulation control and quality assurance process is not fully automated, the story is quite different: if human experts need to spend time setting up each additional simulation with a different model and/or a different grid, then quality assurance can become very expensive.

Various researchers have addressed the problem of choosing between models at different levels of approximation (see Section 5). The issue of quality assurance, however, suggests that the model selection problem should be rephrased. Selecting which model to use in analyzing a physical situation should not be viewed as a choice between applying a more approximate model and applying a less approximate model: if the less approximate model is used, then the more approximate model should always be used as well, because it will provide valuable quality assurance information very cheaply. The model selection problem then becomes that of choosing which is the most detailed model that should be used. (Note that MSA1 doesn't presently address the issue of model selection: as described earlier, all applicable models are always invoked in the current implementation.)

5 Related Work

Automated intelligent controllers for numerical simulators are described in [Gelsey 1991, Gelsey 1994, Sacks 1991, Yip 1991, Zhao 1991], but these only control numerical simulators for ordinary differential equations and do not address the issue of quality assurance. "Intelligent front ends" for computational fluid dynamics simulator are described in [Jambunathan *et al.* 1991, Andrews 1988]. The goal of these front ends is typically to reduce the burden on human users rather than to allow simulators to be run with complete autonomy, so automated quality assurance has not received much attention.

Forbus and Falkenhainer[1990] discuss 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. Intelligent monitoring for complex systems has received considerable attention (e.g., [Dvorak and Kuipers 1991]), but this work has focused on diagnosis of problems in dynamically changing physical systems

as opposed to problems in the execution of computational algorithms which are attempting to simulate the behavior of physical systems.

Selection from models having different assumptions and levels of approximation is addressed in [Addanki *et al.* 1991, Weld 1992, Ellman *et al.* 1993]. There is a large literature about automatic error control for ordinary differential equations, e.g., [Gear 1971, Shampine *et al.* 1976, Fatunla 1988]. Adaptive gridding for partial differential equations is discussed in [Thompson *et al.* 1985].

6 Future Work

There are a number of interesting directions for future work in quality assurance for computational simulation. The model base in MSA1 could be extended in several ways. The addition of another problem type would clearly be a valuable test for the framework. A natural candidate for a second problem type is fluid flow with Mach number near or greater than one. I have already begun investigating such compressible-flow problems in connection with a Rutgers project on design of exhaust nozzles for jet aircraft engines. Some interesting extensions are also possible in the model hierarchy for the current model type, in particular the addition of a more detailed model not relying on the potential-flow approximation of the Navier-Stokes equations of fluid flow. With this extension the fairly detailed PMARC model would itself be used as an approximate model for quality assurance of the Navier-Stokes simulation.

7 Conclusion

Computational simulation of physical systems generally requires human experts to set up the simulation, run it, evaluate the quality of the simulation output, and repeatedly invoke the simulator with modified input until a satisfactory output quality is achieved. Though invocation of simulators by other programs is critical for important tasks such as automated engineering design optimization, this reliance on human experts makes use of simulators by other programs difficult and unreliable. For example, if an optimization algorithm changes a design sufficiently that the assumptions underlying a simulator are

violated, then the simulation may give unreliable results without being detected, since the simulation results are never reviewed by a human but only used directly by the optimization algorithm.

I have presented a framework for constructing intelligent automated simulation controllers which can determine the quality of computational simulation output using evaluation methods based on knowledge of relevant physics and numerical analysis. The automated simulation controller makes use of a data/knowledge base of models and simulations ("model base"). The use of such intelligent controllers overcomes to a large extent the drawback mentioned above: since most or all cases of low-quality simulation output are detected automatically, other programs can reliably invoke computational simulators without the need for human intervention.

I have also described MSA1, an experimental implementation of an intelligent automated controller for PMARC, a widely used computational fluid dynamics simulator. MSA1 is automatically invoked by DA, an automated design system for racing yachts, to compute the efficiency of a yacht's keel.

8 Acknowledgments

This research depended critically on the collaboration of hydrodynamics experts Dr. Martin Fritts and Dr. Nils Salvesen of Science Applications International Corp., and Dr. John Letcher of AeroHydro Inc. This work was done as part of the Rutgers CAP (Artificial Intelligence and Design) project, and benefited significantly from interaction with fellow faculty member Gerard Richter, graduate student Ke-Thia Yao, and the other members of the CAP project. This research was partially supported by NSF grant CCR-9209793, ARPA/NASA grant NAG2-645, and ARPA contract ARPA-DAST 63-93-C-0064.

References

[Addanki *et al.* 1991] Sanjaya Addanki, Roberto Cremonini, and J. Scott Penberthy. Graphs of models. *Artificial Intelligence*, 51:145–177, 1991.

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

[Ashby *et al.* 1992] Dale L. Ashby, Michael R. Dudley, Steve K. Iguchi, Lindsey Browne, and Joseph Katz. *Potential Flow Theory and Operation Guide for the Panel Code PMARC_12.* NASA Ames Research Center, December 1992.

[Dahlquist and Bjorck 1974] Germund Dahlquist and Ake Bjorck. *Numerical methods.* Prentice-Hall, Englewood Cliffs, N.J., 1974.

[Dvorak and Kuipers 1991] Daniel Dvorak and Benjamin Kuipers. Process monitoring and diagnosis. *IEEE Expert*, pages 67–74, June 1991.

[Ellman *et al.* 1992] T. Ellman, J. Keane, and M. Schwabacher. The Rutgers CAP Project Design Associate. Technical Report CAP-TR-7, Department of Computer Science, Rutgers University, August 1992.

[Ellman *et al.* 1993] T. Ellman, J. Keane, and M. Schwabacher. Intelligent model selection for hillclimbing search in computer-aided design. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., 1993.

[Fatunla 1988] Simeon Ola Fatunla. *Numerical methods for initial value problems in ordinary differential equations.* Academic Press, Boston, 1988.

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

[Gear 1971] C. William Gear. *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice-Hall, Englewood Cliffs, New Jersey, 1971.

[Gelsey 1991] Andrew Gelsey. Using intelligently controlled simulation to predict a machine's long-term behavior. In *Proceedings, Ninth National Conference on Artificial Intelligence*, pages 880–887, Cambridge, MA, July 1991. AAAI Press/The MIT Press.

[Gelsey 1994] Andrew Gelsey. Automated reasoning about machines. *Artificial Intelligence*, ?(?):?, ? 1994. To appear.

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

[Katz and Plotkin 1991] Joseph Katz and Allen Plotkin. *Low-speed aerodynamics: from wing theory to panel methods.* McGraw-Hill, 1991.

[Letcher *et al.* 1987] John S. Letcher, Christopher P. Cressy, James C. Olivier III, and Martin J. Fritts. Hydro-numeric design of winglet keels for *Stars & Stripes*. *Marine Technology*, 24(4):265–285, October 1987.

[Letcher 1975] John S. Letcher, Jr. Sailing hull hydrodynamics, with reanalysis of the *Antiope* data. *Transactions of the Society of Naval Architects and Marine Engineers*, 83:22–40, 1975.

[Newman and Wu 1973] J. N. Newman and T. Y. Wu. A generalized slender-body theory for fish-like forms. *Journal of Fluid Mechanics*, 57(4):673–693, 1973.

[Newman 1977] John Nicholas Newman. *Marine hydrodynamics.* MIT Press, Cambridge, Mass., 1977.

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

[Shampine *et al.* 1976] L. F. Shampine, H. A. Watts, and S. Davenport. Solving non-stiff ordinary differential equations—the state of the art. *Siam Review*, 18:376–411, July 1976.

[Thompson *et al.* 1985] Joe F. Thompson, Z. U. A. Warsi, and C. Wayne Mastin. *Numerical grid generation : foundations and applications.* North-Holland, Amsterdam, 1985.

[Vanderplaats 1984] Garret N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design : With Applications.* McGraw-Hill, New York, 1984.

[Weld 1992] Daniel S. Weld. Reasoning about model accuracy. *Artificial Intelligence*, 56:255–300, 1992.

[Yao and Gelsey 1994] Ke-Thia Yao and Andrew Gelsey. Intelligent automated grid generation for numerical simulations. In *Proceedings, 12th National Conference on Artificial Intelligence*, pages 1224–1230, Seattle, Washington, August 1994.

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

[Zhao 1991] Feng Zhao. Extracting and representing qualitative behaviors of complex systems in phase space. In *Proceedings, 12th International Joint Conference on Artificial Intelligence*, 1991.