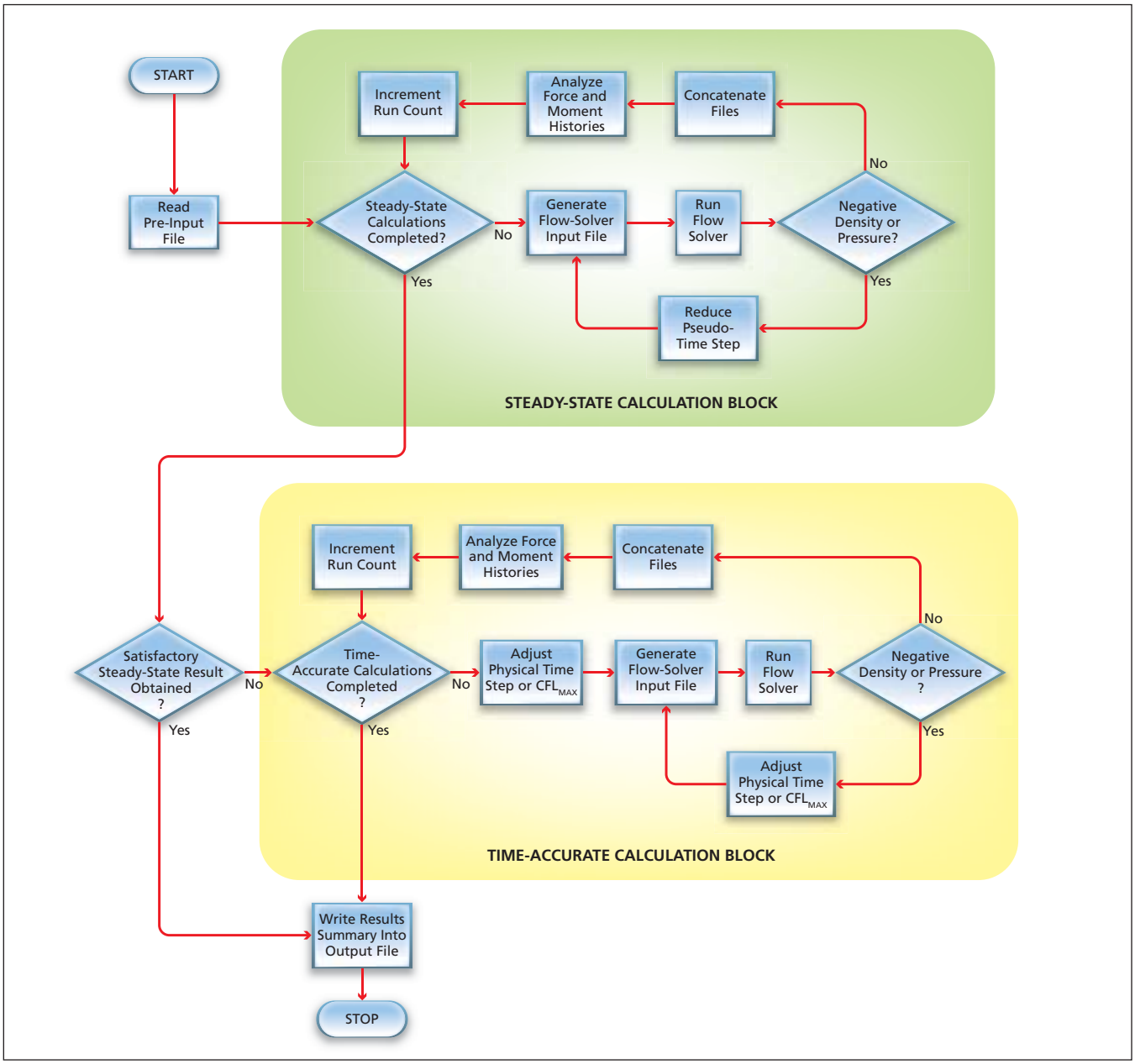
Automated CFD for Generation of Airfoil Performance Tables

Data for all flow conditions of interest are generated efficiently.

Ames Research Center, Moffett Field, California

A method of automated computational fluid dynamics (CFD) has been invented for the generation of performance tables for an object subject to fluid flow. The method is applicable to the generation of tables that summarize the effects of two-dimensional flows about airfoils and that are in a format known in the art as "C81." (A C81 airfoil performance table is a text file that lists coefficients of lift, drag, and pitching moment of an airfoil as functions of angle of attack for a range of Mach numbers.) The method makes it possible to efficiently generate and tabulate data from simulations of flows for parameter values spanning all operational ranges of actual or potential interest. In so doing, the method also enables filling of gaps and resolution of inconsistencies in C81 tables generated previously from incom-



This **Flow Chart** represents the sequences of operations in automated CFD according to the method described in the text.

plete experimental data or from theoretical calculations that involved questionable assumptions.

The method can be implemented by use of any of a variety of digital processors comprising hardware and software subsystems capable of simulating flows. The hardware subsystem could be, for example, a microprocessor, a mainframe computer, a digital signal processor, or a portable computer. The software subsystem can include any of a number of flow solvers — that is, computer programs that solve the governing equations of flow. One such program that is particularly suitable for use in this method is ARC2D, which utilizes finite-difference techniques to numerically solve the Reynolds-averaged Navier-Stokes equations of two-dimensional compressible flow.

At the beginning of a process using this method, the processor receives a description of the airfoil and a pre-input file, which contains parameters representative of the ranges of flow conditions in which the airfoil is to be tested via computational simulations. The processor can perform steady-state and/or time-accurate calculations for simulating flows. Steady-state calculations are typically applicable to such conventional flow conditions as small angles of attack with fully attached flows for which the solutions are independent of time. Time-accurate calculations model the temporal behaviors of time-varying flows.

The upper part of the figure illustrates steady-state calculations according to this method. After reading the pre-input file, the processor determines whether the steady-state calculations specified by that file have been completed. If the calculations have not been completed, the processor generates a flow-solver input file, then the processor executes the flow solver using this input file. If the output of the flow solver includes a negative density or pressure, which is physically impossible, then the pseudo-time step used in the flow solver is reduced and the flow solver is run again using the same inputs. This sub-process is repeated, if necessary, until neither the pressure nor the density in the output of the flow solver is negative, at which point the output of the flow solver is concatenated into an output file. Next, the processor analyzes the residual history of forces and pitching moments and increments the run count. The processor then returns to the step in which it determines whether the steady-state calculations have been completed. If the calculations are found to have been completed, the processor determines whether satisfactory results were obtained. If satisfactory results were not obtained, the processor switches to time-accurate mode.

The lower part of the figure depicts time-accurate calculations according to this method. First, the processor determines whether the time-accurate calculations have been completed. If not, the processor adjusts the physical time step or the maximum allowable value, CFLMAX, of the Courant-Friedrichs-Levy number. [The Courant-Friedrichs-Levy number (CFL) is the product of a time step and a speed characteristic of the flow.] Next, the processor generates a flow-solver input file using the adjusted physical time step or adjusted CFLMAX. If negative density or pressure is found in the output of the flow solver, then the physical time step or CFLMAX is further adjusted, a corresponding new flow-solver input file is generated, and the flow solver is run again. This subprocess is repeated, if necessary, until neither the pressure nor the density is negative. Next, the processor analyzes the force and moment histories and increments the run count. The processor then returns to the step in which it determines whether the time-accurate or the steady-state calculations have been completed. If the time-accurate calculations are found to have been completed, or if the steady-state calculations have been completed with satisfactory results, then the processor writes the results into an output file.

This work was done by Roger Strawn of the U.S. Army and E. A. Mayda and C. P. van Dam of the University of California for Ames Research Center. Further information is contained in a TSP (see page 1).
ARC-15649-1

Progressive Classification Using Support Vector Machines

An approximate classification is generated rapidly, then iteratively refined over time.

NASA's Jet Propulsion Laboratory, Pasadena, California

An algorithm for progressive classification of data, analogous to progressive rendering of images, makes it possible to compromise between speed and accuracy. This algorithm uses support vector machines (SVMs) to classify data. An SVM is a machine learning algorithm that builds a mathematical model of the desired classification concept by identifying the critical data points, called support vectors. Coarse approximations to the concept require only a few support vectors, while precise, highly accurate models require far more support vectors. Once the model has been constructed, the SVM can be applied to new observations. The cost of classifying a new observation is proportional to the number of support vectors in the model. When computational resources are limited, an SVM of the appropriate complexity can be produced. However, if the constraints are not known when the model is constructed, or if they can change over time, a method for adaptively responding to the current resource constraints is required. This capability is particularly relevant for spacecraft (or any other real-time systems) that perform onboard data analysis.

The new algorithm enables the fast, interactive application of an SVM classifier to a new set of data. The classification process achieved by this algorithm is characterized as progressive because a coarse approximation to the true classification is generated rapidly and thereafter iteratively refined. The algorithm uses two SVMs: (1) a fast, approximate one and (2) slow, highly accurate one. New data are initially classified by the fast SVM, producing a baseline approximate classification. For each classified data point, the algorithm calculates a confidence index that indicates the likelihood that it was classified correctly in the first pass. Next, the data points are sorted by their confidence indices and progressively reclassified by the slower, more accurate SVM, starting with the items most likely to be incorrectly classified. The user can halt this reclassification