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

process at any point, thereby obtaining the best possible result for a given amount of computation time. Alternatively, the results can be displayed as they are generated, providing the user with real-time feedback about the current accuracy of classification.

Computational savings are realized through the guided application of resources only to those items that are estimated to be misclassified. The coarse approximation may suffice for items that can be classified easily, and more computation can be devoted to ambiguous or difficult cases. Thus, the algorithm enables the user to exert direct, dynamic control over the balance between classification speed and accuracy. When constraints on computation time and other resources preclude a totally accurate classification of all the data, this algorithm provides the best possible approximation to the classification of each item, rather than fully classifying only a fraction of the data set and leaving the rest marked "unknown."

This work was done by Kiri Wagstaff and Michael Kocurek of Caltech for NASA's Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

The software used in this innovation is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-44089.

Active Learning With Irrelevant Examples

Classification algorithms can be trained to recognize and reject irrelevant data.

NASA's Jet Propulsion Laboratory, Pasadena, California

An improved active learning method has been devised for training data classifiers. One example of a data classifier is the algorithm used by the United States Postal Service since the 1960s to recognize scans of handwritten digits for processing zip codes. Active learning algorithms enable rapid training with minimal investment of time on the part of human experts to provide training examples consisting of correctly classified (labeled) input data. They function by identifying which examples would be most profitable for a human expert to label. The goal is to maximize classifier accuracy while minimizing the number of examples the expert must label.

Although there are several well-established methods for active learning, they may not operate well when irrelevant examples are present in the data set. That is, they may select an item for labeling that the expert simply cannot assign to any of the valid classes. In the context of classifying handwritten digits, the irrelevant items may include stray marks, smudges, and mis-scans. Querying the expert about these items results in wasted time or erroneous labels, if the expert is forced to assign the item to one of the valid classes.

In contrast, the new algorithm provides a specific mechanism for avoiding querying the irrelevant items. This algorithm has two components: an active learner (which could be a conventional active learning algorithm) and a relevance classifier. The combination of these components yields a method, denoted Relevance Bias, that enables the active learner to avoid querying irrelevant data so as to increase its learning rate and efficiency when irrelevant items are present.

The algorithm collects irrelevant data in a set of rejected examples, then trains the relevance classifier to distinguish between labeled (relevant) training examples and the rejected ones. The active learner combines its ranking of the items with the probability that they are relevant to yield a final decision about which item to present to the expert for labeling. Experiments on several data sets have demonstrated that the Relevance Bias approach significantly decreases the number of irrelevant items queried and also accelerates learning speed.

This work was done by Kiri Wagstaff of Caltech and Dominic Mazzoni of Google, Inc. for NASA's Jet Propulsion Laboratory. For more information, contact iaoffice@jpl.nasa.gov. NPO-44094

A Data Matrix Method for Improving the Quantification of Element Percentages of SEM/EDX Analysis

John F. Kennedy Space Center, Florida

A simple 2D $M \times N$ matrix involving sample preparation enables the microanalyst to peer below the noise floor of element percentages reported by the SEM/EDX (scanning electron microscopy/energy dispersive x-ray) analysis, thus yielding more meaningful data.

Using the example of a 2×3 sample set, there are $M = 2$ concentration levels of the original mix under test: 10 percent ilmenite (90 percent silica) and 20 percent ilmenite (80 percent silica). For each of these M samples, $N = 3$ separate SEM/EDX samples were drawn. In this test, ilmenite is the element of interest. By plotting the linear trend of the M sample's known concentration versus the average of the N samples, a much higher resolution of elemental analysis can be performed. The resulting trend also shows how the noise is affecting the data, and at what point (of smaller concentrations) is it impractical to try to extract any further useful data.

This work was done by John Lane of Kennedy Space Center. For further information, contact the Kennedy Innovative Partnerships Program Office at (321) 861-7158. KSC-13303