Software

AutoChem

AutoChem is a suite of Fortran 90 computer programs for the modeling of kinetic reaction systems. AutoChem performs automatic code generation, symbolic differentiation, analysis, and documentation. It produces a documented stand-alone system for the modeling and assimilation of atmospheric chemistry. Given databases of chemical reactions and a list of constituents defined by the user, AutoChem automatically does the following:

1. Selects the subset of reactions that involve a user-defined list of constituents and automatically prepares a document listing the reactions;
2. Constructs the ordinary differential equations (ODEs) that describe the reactions as functions of time and prepares a document containing the ODEs;
3. Symbolically differentiates the time derivatives to obtain the Jacobian and prepares a document containing the Jacobian;
4. Symbolically differentiates the Jacobian to obtain the Hessian and prepares a document containing the Hessian; and
5. Writes all the required Fortran 90 code and datafiles for a stand-alone chemical modeling and assimilation system (implementation of steps 1 through 5).

Typically, the time taken for steps 1 through 5 is about 3 seconds. The modeling system includes diagnostic components that automatically analyze each ODE at run time, the relative importance of each term, time scales, and other attributes of the model.

This program was written by David John Lary of **Goddard Space Flight Center**. *Further information is contained in a TSP (see page 1). GSC-14862-1*

Virtual Machine Language

Virtual Machine Language (VML) is a mission-independent, reusable software system for programming for spacecraft operations. Features of VML include a rich set of data types, named functions, parameters, IF and WHILE control structures, polymorphism, and on-the-fly creation of spacecraft commands from calculated values. Spacecraft functions can be abstracted into named blocks that reside in files aboard the spacecraft. These named blocks accept parameters and execute in a repeatable fashion. The sizes of uplink products are minimized by the ability to call blocks that implement most of the command steps. This block approach also enables some autonomous operations aboard the spacecraft, such as aerobraking, telemetry conditional monitoring, and anomaly response, without developing autonomous flight software. Operators on the ground write blocks and command sequences in a concise, high-level, human-readable programming language (also called "VML"). A compiler translates the human-readable blocks and command sequences into binary files (the operations products). The flight portion of VML interprets the uplinked binary files. The ground subsystem of VML also includes an interactive sequence-execution tool hosted on workstations, which runs sequences at several thousand times real-time speed, affords debugging, and generates reports. This tool enables iterative development of blocks and sequences within times of the order of seconds.

This program was written by Christopher Grasso, Dennis Page, and Taifun O'Reilly with support from Ralph Fteichert, Patricia Lock, Imin Lin, Keith Naviaux, and John Sisino of Caltech for **NASA's Jet Propulsion Laboratory**. *Further information is contained in a TSP (see page 1).*

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (818) 393-2827. Refer to NPO-40365.

Two-Dimensional Ffowcs Williams/Hawkings Equation Solver

FWH2D is a Fortran 90 computer program that solves a two-dimensional (2D) version of the equation, derived by J. E. Ffowcs Williams and D. L. Hawkings, for sound generated by turbulent flow. FWH2D was developed especially for estimating noise generated by airflows around such approximately 2D airframe components as slats. The user provides input data on fluctuations of pressure, density, and velocity on some surface. These data are combined with information about the geometry of the surface to calculate histories of thickness and loading terms. These histories are fast-Fourier-transformed into the frequency domain. For each frequency of interest and each observer position specified by the user, kernel functions are integrated over the surface by use of the trapezoidal rule to calculate a pressure signal. The resulting frequency-domain signals are inverse-fast-Fourier-transformed back into the time domain. The output of the code consists of the time- and frequency-domain representations of the pressure signals at the observer positions. Because of its approximate nature, FWH2D overpredicts the noise from a finite-length (3D) component. The advantage of FWH2D is that it requires a fraction of the computation time of a 3D Ffowcs Williams/Hawkings solver.

This program was written by David P. Lockard of **Langley Research Center**. *Further information is contained in a TSP (see page 1).*
LAR-16338-1

Full Multigrid Flow Solver

FMG3D (full multigrid 3 dimensions) is a pilot computer program that solves equations of fluid flow using a finite difference representation on a structured grid. Infrastructure exists for three dimensions but the current implementation treats only two dimensions. Written in Fortran 90, FMG3D takes advantage of the recursive-subroutine feature, dynamic memory allocation, and structured-programming constructs of that language. FMG3D supports multi-block grids with three types of block-to-block interfaces: periodic, C-zero, and C-infinity. For all three types, grid points must match at interfaces. For periodic and C-infinity types, derivatives of grid metrics must be continuous at interfaces. The available equation sets are as follows: scalar elliptic equations, scalar convection equations, and the pressure-Poisson formulation of the Navier-Stokes equations for an incompressible fluid. All the equation sets are implemented with nonzero forcing functions to enable the use of user-specified solutions to assist in verification and validation. The equations are solved with a full multigrid scheme using a full approximation scheme to converge the solution on each succeeding grid level. Restriction to the next coarser mesh uses direct injection for variables and full weighting for residual quantities; prolongation of the coarse grid correction from the coarse