INTRODUCTION

FORTRAN CALCULUS is a very high order extension of the FORTRAN 77 language for calculus-based programming. This extension includes a set of calculus-level macro statements and a calculus run-time environment. Together they provide an automatic programming platform for inverse problems of higher mathematical systems -- problems involving multidimensional systems of equations in which the unknowns are independent variables or parameters, rather than dependent variables. Examples include optimization problems, nonlinear systems of equations, boundary-value problems of differential equations, implicit differential equations, model fitting and process identification. Such problems may be defined as hierarchies of dynamic problem units called *calculus processes*.

A calculus process consists of

- a *problem statement*,
- a *model* containing its equations, and
- a *solver* containing its solution algorithm.

The problem statement and the model are the visible parts of the problem, defined by the user. The solver is an invisible tool that is part of the run-time environment. The problem statement is a calling sentence in an upper level subprogram which invokes the lower level of the hierarchy consisting of the model and solver subprograms.

The form of the problem statement of the calculus process is provided by the calculus-level macro statements, FIND, INVOKE, INITIATE, and INTEGRATE. These extensions to FORTRAN syntax are translated into a sequence of FORTRAN subroutine calls which define and control the calculus process.

The model is a new kind of FORTRAN subroutine whose real arithmetic operators are overloaded for differentiation. Thus the equations of the model are differentiated during execution with respect to independent variables defined the problem statement. This differentiation process is the central mechanism of FORTRAN CALCULUS. It is an exact numerical process using analytic differentiation formulas in an extended arithmetic, but no symbolic derivation is involved. It is the mathematical context of this differentiation which characterizes a calculus process. This context is a coordinate system, created dynamically by the problem statement. This coordinate system is *inherited* by each model subroutine as it is called during process execution. When the calculus process is concluded, the coordinate system is destroyed.

The solver is a generic algorithm subsystem used as a solution control system. It performs its function solely on the basis of an exchange of variable and partial derivative

values from the problem statement and one or more calls to the model subroutine. The FORTRAN CALCULUS library contains a "pantheon" of built-in solvers for general classes of problems.

The combination of macro statements, calculus environment, models, and solvers is a platform for addressing applications in a higher order "shorthand" of *non-algorithmic* programming, reducing them to application problem statements virtually devoid of mathematical mechanics. The primary effect is to *segregate* engineering modeling from solution mechanics, reducing or even eliminating the need for user skill in numerical analysis. A by-product of this is the *purification* of engineering software of the entanglements that have inhibited software reusability. The purified models become reusable elements that are easily recast into new applications, and in some cases may become standardized engineering utilities, employable in a broad base of applications.

A further effect is the *elevation* of existing engineering software from the realm of simulation to the realm of optimization. This is an automatic run-time process of the calculus environment, performed via differential arithmetic. Thus it is completely invisible to the user and independent of the source code. In effect, differential arithmetic turns a simulation model into an optimization model for any arbitrary set of unknown independent parameters, when called upon during execution. Potentially any engineering subroutine that generates continuous or piecewise continuous function values can be elevated in this manner. The net benefit is that such models can be used as inverse vehicles for computing the input parameters that drive the engineering to meet specific goals. Moreover, optimization methods are multidimensional one-to-many solution methods, that are capable of simultaneously solving many of these driving parameters.

The combined benefits of non-algorithmic programming, segregation of engineering from solution mathematics, reusable engineering software, and elevation from simulation to optimization, offer a software renaissance to scientific software developers and to engineering application developers. The calculus platform submerges the mathematics into the role of engines rather than programs, and programs become merely problem statements. This paves the way for a similar evolution of standardized reusable engineering utilities that can be submerged into platform libraries in the manner of solvers, becoming the model-bases of a new order of *knowledge integration*, with expert systems to guide application software synthesis. FORTRAN CALCULUS and its other scientific language manifestations will provide the foundation for this new order of scientific computing.

This manual is meant to serve as a reference handbook for calculus-based programming. The following is a synopsis of the manual:

Chapter 1: THE CALCULUS ENVIRONMENT describes the software engineering properties of the software system including the run-time structures of variables and

programs, the syntax and purpose of the macro statements, and usage limitations. This discussion is an essential overview of the foundation of calculus-based programming.

Chapter 2: PROBLEM ANALYSIS AND CALCULUS PROCESSES is a general treatment of system-level problem structures and the dynamic roles played by program elements. It discusses a unified approach to problem analysis, the different kinds of processes and the principles common to them. It also discusses the combining of processes to address multi-process problems.

Chapter 3: ELEMENTARY OPERATIONS AND UTILITIES is a compendium of mathematical language extensions providing tools for building mathematical programs. Included are mechanisms for computing derivatives and definite integrals; various utilities for tabular data fitting; dynamic array allocation and output report generation; and linear algebra operations.

Chapter 4: GENERAL ALGEBRAIC EQUATIONS describes the process of solving for roots of simultaneous equation systems, including determined, overdetermined, and underdetermined systems. It provides detailed information about the built-in solvers for this kind of process.

Chapter 5: ORDINARY DIFFERENTIAL EQUATIONS is a full treatment of the subject of simultaneous ODE's covering higher-order equations, implicit equations and boundary-value problems. It provides detailed information about the use of differential equation solvers.

Chapter 6: UNCONSTRAINED OPTIMIZATION is a detailed treatment of nonlinear optimization and the hierarchic combination of processes for process identification and optimal control.

Chapter 7: CONSTRAINED OPTIMIZATION is a treatment of linear, mixed integer and nonlinear programming problems. It provides detailed information about the use of the mathematical programming solvers.

Chapter 8: GRAPHICS is a description of the graphics subsystem. It describes the graphics database, the generation of graph objects, and the production (display, printing and plotting) of graphs, including contour graphs. This chapter also contains several complete examples employing features explained in previous chapters combined with graphics output.

APPENDIX A: GUIDELINES FOR DATA APPROXIMATION is a review of the various techniques of chapters 3,4,6 & 7 for applications of data fitting to mathematical functions. It provides a requirements vs. capabilities matrix and notes discussing the use of the various methods to different kinds of data approximation.