IECEC-98-028
33rd Intersociety Engineering Conference on Energy Conversion
Colorado Springs, CO, August 2-6, 1998

OPTIMIZATION AND AUTOMATED DATA CORRELATION IN THE NASA STANDARD THERMAL/FLUID SYSTEM ANALYZER

Brent A. Cullimore
C&R Technologies, Inc.
303 971-0292 Voice
303 971-0035 FAX

ABSTRACT

SINDA/FLUINT (Ref 1-7) is the NASA-standard heat transfer and fluid flow analyzer for thermal control systems. Because of its general formulation, it is also used in other aerospace specialties such as environmental control (ECLSS) and liquid propulsion, and in terrestrial industries such as electronics packaging, refrigeration, power generation, and transportation industries.

SINDA/FLUINT is used to design and simulate thermal/fluid systems that can be represented in networks corresponding to finite difference, finite element, and/or lumped parameter equations. In addition to conduction, convection, and radiation heat transfer, the program can model steady or unsteady single- and two-phase flow networks.

C&R's Sinaps*Plus*® is a complete graphical user interface (pre- and postprocessor) and interactive model debugging environment for SINDA/FLUINT (Ref 8, 9). Sinaps*Plus* also supports the C language in addition to the traditional choice of Fortran for concurrently executed user logic.

This paper describes revolutionary advances in SINDA/FLUINT, the NASA-standard heat transfer and fluid flow analyzer, changing it from a traditional point-design simulator into a tool that can help shape preliminary designs, rapidly perform parametrics and sensitivity studies, and even correlate modeling uncertainties using available test data.

Innovations include the incorporation of a complete spreadsheet-like module that allows users to centralize and automate model changes, even while thermal/fluid solutions are in progress. This feature reduces training time by eliminating many archaic options, and encourages the performance of parametrics and other what-if analyses that help engineers develop an intuitive understanding of their designs and how they are modeled.

The more revolutionary enhancement, though, is the complete integration of a nonlinear programming module that enables users to perform formal design optimization tasks such as weight minimization or performance maximization. The user can select any number of design variables and may apply any number of arbitrarily complex constraints to the optimization. This capability also can be used to find the best fit to available test data, automating a laborious but important task: the correlation of modeling uncertainties such as optical properties, contact conductances, as-built insulation performance, natural convection coefficients, etc.

Finally, this paper presents an overview of related developments that, coupled with the optimization capabilities, further enhance the power of the whole package.

I. BACKGROUND: Thermal/Fluid Networks

A. Thermal Networks (SINDA)

SINDA uses a thermal network approach, breaking a problem down into points at which energy is conserved (*nodes*), and into the paths (*conductors*) through which these points exchange energy via radiation and conduction. While often applied as a lumped-parameter modeling tool, the program can also be used to solve the finite difference or finite element equations for conduction in appropriately meshed shells or solids.

An important improvement over ancestral versions of SINDA is the inclusion of submodels, which enabled analysts to subdivide a large network of nodes and conductors into collections of subnetworks consisting of nodes, conductors, or both. Submodels represent a convenient means of combining separately developed models, each with its own control variables, customization logic, solution method, and perhaps conflicting node and conductor numbering schemes. More often, they are simply used to improve the

organization and legibility of the model, or to perform high-level simulation manipulations such as dynamically swapping sets of boundary conditions, evaluating alternate designs or components, or simulating variable configurations.

B. Fluid Networks (FLUINT)

To answer the need to model two-phase fluid systems and to replace the cumbersome and limited "one-way conductor" methods employed by older versions of SINDA for fluid flow simulation, FLUINT was developed by NASA in the 1980's as a major expansion of SINDA. FLUINT introduced a new type of submodel composed of network elements, *lumps* and *paths*, which are analogous to traditional thermal nodes and conductors, but which are much more suited to fluid system modeling. Unlike thermal networks, fluid networks are able to simultaneously conserve mass and momentum as well as energy.

Thermal and fluid models may be used alone or together to solve conjugate heat transfer problems as typically found in thermal control, propulsion, and energy systems.

II. BACKGROUND: The Built-in Spreadsheet

A. Registers

Spreadsheets (e.g., Microsoft's Excel™) represent one of the most commonly used types of software on computers today, second only to word processors. Books have been written on adapting popular spreadsheets to engineering uses. In fact, many engineers have written their own simple SINDA-like codes inside of spreadsheets. Others have used spreadsheets as input preprocessors to SINDA/FLUINT, exploiting the ability to define inputs algebraically while preserving complex interrelationships between data. For example, defining a key dimension (e.g., a diameter or a plate thickness) in one cell allows the rest of the model to be input as a function of that cell, enabling rapid and consistent model changes.

Perhaps it is not surprising then that one of the most popular original features in Sinaps*Plus*, the graphical interface to SINDA/FLUINT, is "registers." Because of this popularity, and to enable advanced options such as the Solver described later, registers were introduced to SINDA/FLUINT in Version 3.2 in 1996. (Version 4.0 is the current release, which will be replaced by Version 4.1 this fall.)

In almost all data fields in SINDA/FLUINT, arbitrarily complex expressions may be used instead of numeric constants. In other words, "0.25*pi*(1.0/12.0)^2" can be used to specify the area of a circle, as can its numeric equivalent "5.454e-3." Common values such as π ("pi"), conversion constants, and functions ("sin()," "ln()," "max()," etc.) may be used within these expressions, which follow normal Fortran and C rules of operator precedence. Two advantages of using expressions rather than numeric inputs are improved self-documentation and reduced errors.

A much more important advantage of expressions is that the model can be defined algebraically using *registers*. In addition to prestored constants such as "pi," the user can define up to 5000 arbitrarily named registers (e.g., "area" or "flux"). These names can then be used throughout the model instead of numeric data, or as part of expressions. For example, if a register named "diam" were created and were assigned the value "1.0/12.0", the expression in the previous paragraph could have been specified as "0.25*pi*diam^2." Figure 1 shows examples of input form for registers in Sinaps*Plus*. Since registers can be defined in terms of other registers, SINDA/FLUINT thereby ***achieves spreadsheet-like functionality:*** model changes can be made quickly and consistently. The ability to define inputs algebraically also relieves the user from having to apply such parametric variations in user logic (which may strain the programming abilities of some users). Additional features, as described later, have made the generation of parametric analyses a key feature of SINDA/FLUINT.

Other advantages of registers are harder to explain. For example, *registers enable an entire model to be built before dimensions and material selections have been finalized*. This feature enables teams of analysts to build a model concurrently with the design team in a fast turn-around project instead of waiting for final design. In fact, registers enable changes to the model to be performed relatively quickly, permitting the analysts to better support design decisions as they were made, rather than waiting to critique a final design at a point when changes are not as easily tolerated.

Another important use of registers is the ability to contain several design cases within a single model. In many ways, with a little foresight registers become a "control panel" by which both the model and its execution are controlled.

For example, consider Figure 2, which illustrates the 3 node bar problem commonly used in top-level descriptions of SINDA. In this problem, a one-dimensional bar is heated on one end and radiates to space on the other end. In older versions, the input file might appear as shown in Table 1. The definition of a nodal capacitance as "0.006" leaves little clue to another user as to the meaning

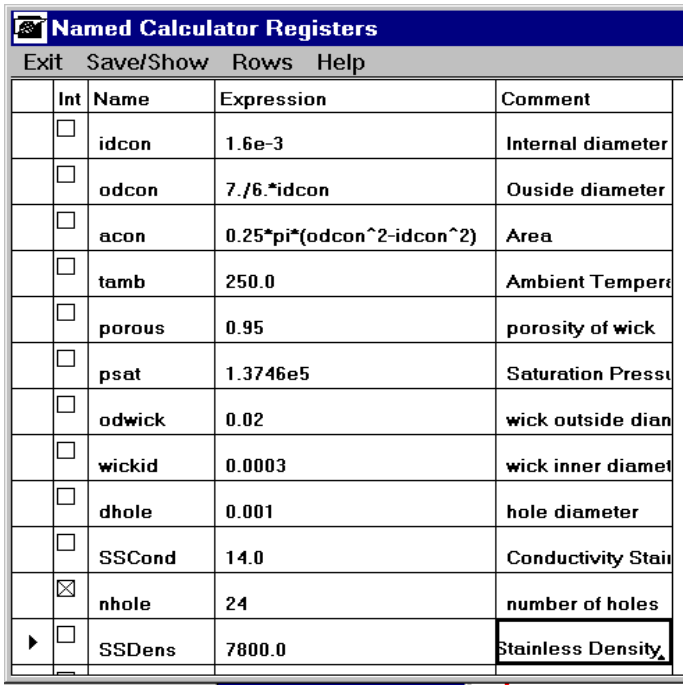

Figure 1: Sample* Sinaps*Plus Register Form

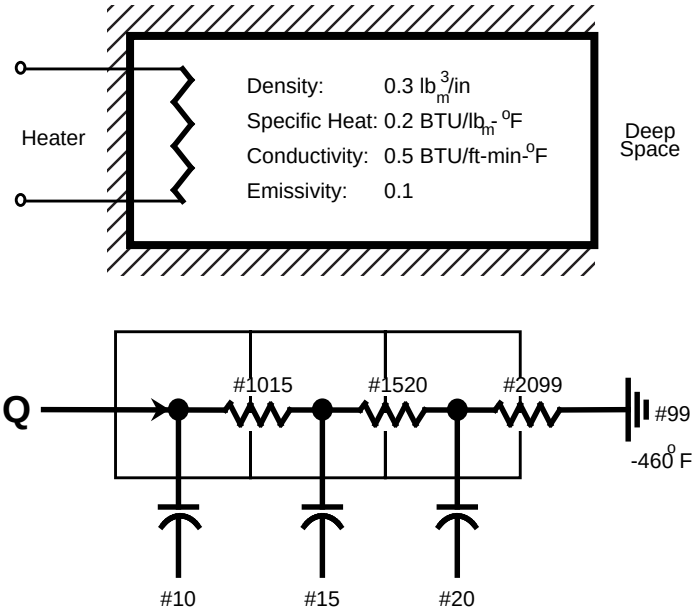

Figure 2: Simple Heated Bar and 3-node SINDA Model

of that node, much less its size, material, etc. Using the old-fashioned expressions that were available in the older versions, the "0.006" could have been replaced with "0.3*0.2*0.1*1.0/3.0," but that too yields little information about the node, and any changes to the dimensions, material, initial or boundary conditions, etc., would mean that the model must be laboriously (and perhaps erroneously) reworked.

This compares with Table 2, which presents the more modern way of handling the same problem using registers (along with built-in constants and conversion factors such as *sbcon* and *btuhrwat*). The slight size increase of the input file, which is atypical, is more than offset by the centralized control provided over the key model parameters such as dimensions and properties. Now, the definition of a node capacitance as "(Dens*Cp*Volume)/Resol" has more

Table 1: Three Node Bar Model in Prior Version

```
HEADER OPTIONS DATA
TITLE HEATED BAR SAMPLE PROBLEM
      OUTPUT  = BAR.OUT
      MODEL   = TEST
HEADER NODE DATA, SUB1
     10, 70.0, 0.006
     15, 70.0, 0.006
     20, 70.0, 0.006
     -99, -460., 0.0
HEADER CONDUCTOR DATA, SUB1
     1015, 10, 15, 0.00417
     1520, 15, 20, 0.00417
    -2099, 20, 99, 1.98E-15
HEADER CONTROL DATA, GLOBAL
     TIMEND = 1000.0, OUTPUT = 1.0
HEADER SOURCE DATA, SUB1
     10, 10.0*3.413/60.0
HEADER OPERATIONS
BUILD TEST, SUB1
      CALL FWDBCK
HEADER OUTPUT CALLS, SUB1
      IF(T15.GE.200.0) THEN
          CALL TPRINT('SUB1')
          TIMEND = TIMEN
      ENDIF
END OF DATA
```

Table 2: Three Node Bar Model in Modern Version

```
HEADER OPTIONS DATA
TITLE HEATED BAR SAMPLE PROBLEM, WITH REGISTERS
      OUTPUT  = BAR.OUT
      MODEL   = TEST
HEADER REGISTER DATA
    DENS    = 0.3
    CP      = 0.2
    CON     = 0.5
    EMIS    = 0.1
    THICK   = 0.1
    WIDE    = 1.0
    LONG    = 3.0
    AREA    = THICK*WIDE
    VOLUME  = AREA*LONG
     HR2MIN  = 60.0
     RESOL   = 3.0
HEADER NODE DATA, SUB1
  GEN 10,3,5,70.0,(DENS*CP*VOLUME)/RESOL
  -99, -460., 0.0
HEADER CONDUCTOR DATA, SUB1
  GEN 1015,2,505,10,5,15,5 \
             (CON/12.0)*AREA/(LONG/RESOL)
  -2099, 20, 99, EMIS*AREA*sbcon/(HR2MIN*144.0)
HEADER CONTROL DATA, GLOBAL
     TIMEND = 1000.0, OUTPUT = 1.0
HEADER SOURCE DATA, SUB1
     10, 10.0/(btuhrwat*HR2MIN)
HEADER OPERATIONS
BUILD TEST, SUB1
      CALL FWDBCK
HEADER OUTPUT CALLS, SUB1
      IF(T15.GE.200.0) THEN
          CALL TPRINT('SUB1')
          TIMEND = TIMEN
      ENDIF
END OF DATA
```

intuitive meaning. More importantly, not only will the original author find model changes to be painless, but so will the engineer who inherits the model.

B. Dynamic Registers

Sizing and sensitivity studies are very important, yet are often neglected because they are time-consuming. Sensitivity studies are especially needed since they help the analyst better understand not only the model, but more importantly the design itself.

Extensive use of registers, while a tremendous improvement in its own right, enables the use of even more powerful parametric and design/correlation features. SINDA/FLUINT "remembers" every place that a register was used in the definition of a model, and knows how to propagate changes to registers throughout the model *while the solution is proceeding*.

For example, in the bar problem presented earlier (Figure 2, Table 2), the purpose of the model is to find the time (to the nearest minute) at which the middle node exceeds 200 degrees. Assume that the user wanted to get a plot of this result as a function of the bar specific heat ("Cp"), holding other properties constant. In the older codes, the user could assemble such a plot by making repeated runs, each returning a single point on the plot. Or the user could write a little logic (perhaps a Fortran DO loop or a C while block) inside of OPERATIONS to run through a series of values. However, in the latter case the user of older codes would then assume the burden of updating the model consistently, in this case perhaps by

adding a nested loop to change the capacitance values for the 3 nodes.

To perform a parametric variation of the specific heat (Cp) of the bar sample problem within one run in SINDA/FLUINT, updating the network to reflect a change in design parameters is trivial:

```
Cp = new_value
call upreg
```

The call to UPREG *updates the entire model on the basis of the current value of the registers*. In this problem, only Cp was changed, and that value was only used in the definition of the capacitance of three nodes.

While such logic may be trivial in this sample problem, it can become very complex in the more typical 1000 to 10,000 node problems, especially considering that the values of density and conductivity would also have to change to reflect realistically available materials. Updating a large and complex model "dynamically" (during the production of the solution) is laborious and even dangerous without registers.

For example, perhaps an uncertainty such as a bond joint conductance is being evaluated, and many such joints exist in the system. The conductance of such a bonded joint could be defined as a register named "Hbond" and that register would be used throughout the model wherever that type of joint was found. Running a parametric sensitivity analysis of performance versus as-built bond joint conductance becomes an easy task using dynamic registers.

This example illustrates the importance of having automated changes made in a manner consistent with how the user originally defined the problem: in terms of the basic dimensions and properties. Often seemingly complex models with tens of thousands of nodes can be centrally controlled via the key underlying properties and dimensions of the system, which typically number perhaps 50 to 100.

Registers can be used in the definition of nodes, conductors, source terms, fluid system variables, and even array data and control parameters. ***Almost every data value in SINDA/FLUINT can be defined as an algebraic expression, and sweeping model changes can be made with just a few lines.*** As long as users are consistent in the definition of the model, then they need not remember where in the model those definitions were used, nor how to access and change them in logic.

The examples presented in this section are extremely simplified. Actually, the user has extensive control not only over the update process, but also over the registers and the places in which they were used. For example, the user can request that only thermal data be updated, or only node data in a certain submodel, or only the data within a specific array, etc. The user can also disconnect and reconnect all or part of the model from being automatically updated if desired. Users can even check to see if a particular value is defined via a register-containing expression (vs. a hard-wired value). Registers themselves can be changed by providing a new expression instead of a fixed value, and registers can be updated independently from the model if desired.

There are many traditional features in SINDA/FLUINT and older versions of SINDA that attempted to provide such functionality, such as numbered user constants (e.g., "XK33" and "K402") in SIV nodes/conductors and source data. However, they are clumsy and confusing compared to the new register-based methods. Therefore, these outdated methods are discouraged and they will become undocumented in the future.

III. THE SOLVER: OPTIMIZATION

Both the ability to define a model algebraically and to make dynamic variations to it while it is being solved are by themselves very powerful features that are already quite popular with users even though they are comparatively recent additions to SINDA/FLUINT.

However, those features were just the first steps towards an even more revolutionary change in SINDA/FLUINT: the inclusion of automated design optimization and test data correlation solutions: the "Solver."

In older versions of SINDA/FLUINT and in older SINDA-like codes, there are really only two types of solutions: steady-state and transient, although these can be arranged by the user into arbitrarily complex solution sequences. Furthermore, older SINDAs are strictly point design simulation tools: given a mathematical model of a fixed design, they predict the performance of that design. Given an input value of Y (perhaps size/shape, flowrate, etc.), they return a value of X (perhaps temperatures, pressures, etc.): X = X(Y). But what if the situation is reversed, and the design of the system needs to be determined to meet a performance specification? Or more simply, what if the code needs to return Y=Y(X)? Or what if the point of the analysis were to find the mathematical model of a certain design that best fits test data (in other words, the design stays fixed but the uncertainties in the model are varied)?

Such design or correlation problems were approached in older versions of SINDA/FLUINT by making parametric runs, or by adding somewhat complicated user logic to permute the model. Both of these operations have been greatly facilitated by the advent of dynamic registers, but a more complete feature is available in Version 4.0 and later: a Solver module to perform the model variations *automatically* according to simple user instructions.

The types of applications addressed by the Solver include:

1. *One Dimensional Sizing or Goal Seeking.* Example: SINDA/FLUINT can easily predict the temperature at the outlet of a heat exchanger as a function of the flowrate: T=T(F). If instead, the user wanted to find the value of flowrate that yielded a specific temperature at the outlet, F=F(T), the Solver can be used to invert the problem without complicated logic or controls. The user would simply assign the goal of the analysis to be the desired set-point temperature (T), indicate the variable to be permuted (F, the flowrate), and SINDA/FLUINT would internally iterate a user-specified solution procedure (perhaps as simple as a single steady-state run) until the required flowrate was found.

2. *Optimization.* The design and sizing example described above can be generalized into a more complex question with more than one design variable. For example: What are the values of X, Y, Z ... et cetera that yield the maximum (or minimum value) of A, subject to the constraints that X is between C_1 and C_2, and that D is less than E, etc.?

   For example, a fin cross section could be optimized by finding the thickness along its length (at N evenly spaced cross sections) that maximizes fin efficiency while weighing less than 40 grams, or that minimizes mass while preserving a fin efficiency of at least 0.85. Or, the analyst might wish to know what number of parallel pipes of which diameter maximize heat exchange without exceeding a pressure drop budget of 10 kPa.

3. *Automated Test Data Correlation.* Instead of an optimum design, the user can equivalently seek to find the values of uncertain performance parameters (e.g., contact conductances, surface optical properties, insulation performance, natural convection coefficients) that produce the best fit to test data (i.e., the best mathematical model), whether steady state or transient.

   In this case, the goal of the analysis is a minimum difference between test data and predictions, perhaps as measured by the sum of squared differences (yielding a least squares curve fit). Perhaps the most uncertain parameter needs to be adjusted first, followed by the next most uncertain parameter, etc., or perhaps the best fit varying all uncertainties simultaneously should be sought.

The advent of dynamic registers together with the integration of a generalized nonlinear programming system or "optimizer" have enabled SINDA/FLUINT to achieve this ideal.

The Education Gap - Perhaps because of a prior lack of adequate tools, few thermal engineers have had training in formal optimization techniques and most are unfamiliar with the terminology and underlying mathematical basis. To many, "optimization" simply means the process of manually adjusting design parameters in an uncontrolled fashion until a more satisfactory result has been achieved.

Furthermore, many engineers have become so accustomed to the limitations of previously existing codes that the limited "point design simulation" approach seems natural and even inevitable. They have perhaps forgotten the true purpose of such codes: to assist in the generation and refinement of a design. They also tend to be suspicious that a computer program could really produce an intelligent design or a useful correlation to test data, perhaps due to an overselling of "artificial intelligence" software in the 1980s. The Solver thus seems to many to be unnatural or "too good to be true."

Optimization is not "artificial intelligence." And it won't produce useful results unless the analyst has adequately defined the problem. Optimization requires the user to provide a means by which a design can be analytically evaluated (i.e., quantitatively measured to compare with other design alternatives), and criteria by which useless designs can be discarded and useful designs retained.

Formal optimization requires the user to define:

1. *An objective.* This can be defined as a value to be maximized (e.g., a performance metric or figure of merit) or minimized (e.g., weight or life cycle cost). *The objective allows a quantitative comparison between any two design options.*
2. *Design variables.* These are the model parameters whose values can be changed as needed to achieve the objective. A given set of values for these design variables completely describes a single point design. *Design variables are the parameters to be sized or selected: their final values are the end-product of the optimization.*
3. *Constraints.* Often, these are simply the upper or lower limits on the design variables. Constraints can be much more complicated, however, involving not just the design variables but also the model predictions (design performance). *Constraints distinguish a viable design from a useless design.*
4. *An evaluation procedure.* The user must define a procedure by which the objective value can be calculated for a given point design. This procedure will typically involve traditional SINDA/FLUINT solutions routines, and may be as simple as a single steady state simulation. *The procedure is the method by which the objective and constraints can be calculated given a specific set of design variables (i.e., a point design).*

The Solver allows the user to designate these variables, constraints, and procedures in a method that is natural for SINDA/FLUINT users. The Solver can be viewed by the experienced SINDA/FLUINT user as a means of setting up a series of traditional SINDA/FLUINT runs that are launched automatically and tasked with a specific analytic goal. Of course, no separate runs are actually performed. The Solver is executed internal to a single SINDA/FLUINT run: *the Solver is essentially a high-level solution routine, running steady states and transients as needed to achieve its purpose.*

Analysts familiar with optimization techniques have reacted enthusiastically to the introduction of the Solver, and their successes will hopefully teach other engineers to phrase their design requirements in a mathematical fashion to exploit computers to refine designs or even to suggest nonintuitive design alternatives.

In the meantime, however, the Solver *is* being eagerly exploited to assist in a previously laborious and informal task: the correlation of models to test data. While many analysts are not accustomed to approaching the design task analytically, they *are* accustomed to the task of having to back out as-built performance from test data, and it appears to be a universally despised task.

Therefore, the focus of the examples in this paper will be on test data correlation tasks, rather than on design optimization. It is hoped that the users flocking to the Solver to automate their cor-

relation work will, in the process, learn about formal optimization techniques such that they can be exploited in the early stages of their next design project.

IV. THE SOLVER: TEST DATA CORRELATION

One of the most time-consuming tasks of a thermal/fluid analyst is the correlation against test data, which is often a necessary step at the end of the design cycle. Thermal/fluid models can be very accurate once certain familiar factors are known. These factors (e.g., as-built insulation performance, optical properties, contact and bond-line conductances, natural convection coefficients, head loss factors, etc.) can rarely be predicted with any precision before hardware is built, but they can be extracted from test data and used to improve the analytic predictions. In essence, *thermal/fluid models are often used simply to intelligently extrapolate test data to untestable conditions.*

Correlation is often referred to as "solving the reverse problem," "reverse engineering," "model adjustment," "model refinement," or "model calibration."

An optimization solution such as the Solver provides automated correlation tools intrinsically. When performing a correlation, the terms defined in the previous section are still relevant, but they take on a new meaning.

1. The "objective" becomes a best fit with test data, which *might** mathematically be defined as minimizing the squared error summed over all points of comparison:

   $$\text{minimize: } \Sigma(T_{test} - T_{predict})^2$$

2. The "design variables" become the parameters whose values are uncertain, such as bolted joint conductances, component head loss factors, etc.
3. The "constraints" become the reasonable limits on the design variables. For example, the contact conductance of a certain bond type might be limited to within the range of 500 to 1000 W/m^2-K.
4. The "procedure" becomes whatever SINDA/FLUINT solutions or series of solutions are required to generate a comparison to one or more sets of test data. This can range from a single steady state to simultaneous comparison of multiple steady or transient tests.

A. Correlation Example One

As a first simple example, consider one last time the three node bar problem. Perhaps the surface emissivity is uncertain. If the bar is bare aluminum, the best guess might be 0.1, but the value could range from 0.08 to 0.2. The "emis" variable would be defined as the design variable with the constraint that 0.08<emis<0.2.

Being a transient, the objective function might be a best match against a time history. In other words, the objective value would be a comparison against test data accumulated over the duration of the test run. Or, if only the end time were known from test data (i.e., the time at which the middle of the bar reached 200 degrees), then the absolute value of the difference between model predictions at the end of the transient and the test data would become the value to be minimized.

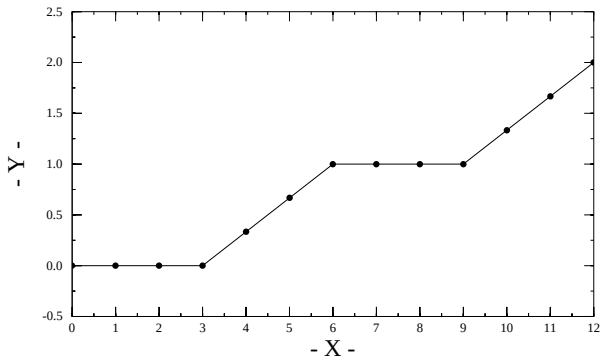

Figure 3: Fake Data to Compare Against

B. Correlation Example Two

To demonstrate data correlation methods without delving into the details of a specific model, consider a dummy curve of "test data" to be correlated at 13 points as shown in Figure 3. This data is to be correlated using a third order polynomial "prediction" of the form $y = A + Bx + Cx^2 + Dx^3$. In other words, the task of the Solver is to find the values of A, B, C, and D that best fit the fake test data at the 13 points shown in Figure 3.

A complete input file for this case is presented in Table 3. A fake thermal submodel is used to overcome the fact that this contrived example actually uses no traditional SINDA/FLUINT networks nor solution routines.

Four registers (able, baker, charlie, delta) are used to represent the coefficients A, B, C, and D in the polynomial "prediction." These registers are also listed as design variables, and hence are changed by the Solver (called from within OPERATIONS) according to the evaluations performed in PROCEDURE.

Most of the input file is devoted to generating an array of elements to represent the test data (which in a real case would be read into or included into the input file), and to generating the polynomial prediction and placing it too in an array. This preparation allows the auxiliary routine COMPARE to be used to generate the data needed by the Solver. COMPARE, which can also perform reporting tasks, is part of a general purpose data correlation tool suite available to help the user deal with the copious amounts of data used in real correlations.

Figure 4 presents the data for the least squares fit ("RMSERR") prepared in Table 3, as well as a alternative MINIMAX method not described in this paper. Also shown are variations in the underlying numerical methods used by the Solver ("METHO") of which three are available in addition to various control and customization constants.

* There are many ways of defining "best fit," as described in the User's Manual (Ref 7).

Table 3: Comparison of Polynomial "Prediction:" with Fake Test Data

```
header options data
title fake fitting: least squares via RMS
       output  = fakefit.ls
header array data, fake
 1 =    0.,     0.0
        3.,     0.0
        6.,     1.0
        9.,     1.0
        12.,    2.0
C "TEST DATA"
       10 = SPACE,13
C "PREDICTIONS"
       20 = SPACE,13
header node data, fake
header register data
       able    = 0.01
       baker   = 0.01
       charlie = 0.01
       delta   = 0.01
header design data
       able
       baker
       charlie
       delta
header operations
build ff, fake
defmod fake
       do 1 itest=0,12
              xtest   = float(itest)
              call d1deg1(xtest,fake.a1
     .                   ,fake.a(10+itest+1))
1      continue
       call solver
       call destab
       call compare(a20,a10,0,0,'report',-1)
header procedure
defmod fake
       do 1 itest=0,12
              xtest   = float(itest)
              a(20+itest+1) = able
     .                + baker*xtest
     .                + charlie*xtest**2
     .                + delta*xtest**3
1      continue
       call compare(a20,a10,0,0,'rmserr',object)
end of data
```

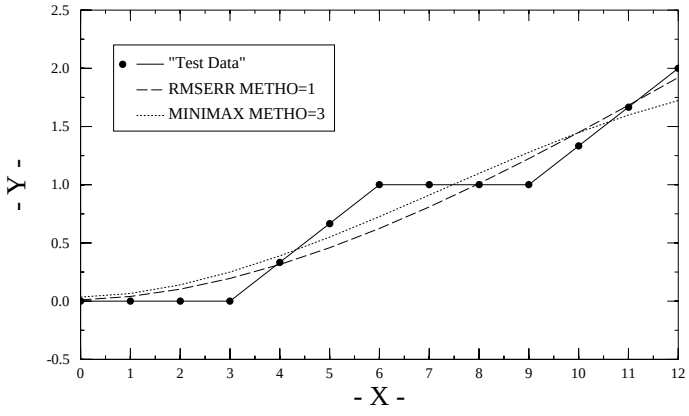

Figure 4: Results of Comparisons with Fake Test Data using Various Solver Methods

C. Correlation Example Three

In 1985 the first flight experiment of a capillary pumped loop was flown. The start-up transient of this device has been a sample problem in the SINDA/FLUINT User's Manual (Ref 7) from its inception, and in fact this model was used by many to help validate SINDA/FLUINT itself. However, this comparison was made using vendor data for a conductive pad, and using a boundary condition (the reservoir temperature) which was known to be incorrect since its measurement conflicted with other test data. Also uncertain was the heat transfer coefficient in the axially grooved aluminum condenser tubing. Furthermore, the actual heat dissipated into the evaporator was uncertain (taking into account uncertainties in the measurements and heat leaks to the environment). It has always been a source of frustration to this author that this model was used to "validate" SINDA/FLUINT when so many conditions were uncertain in the predictions.

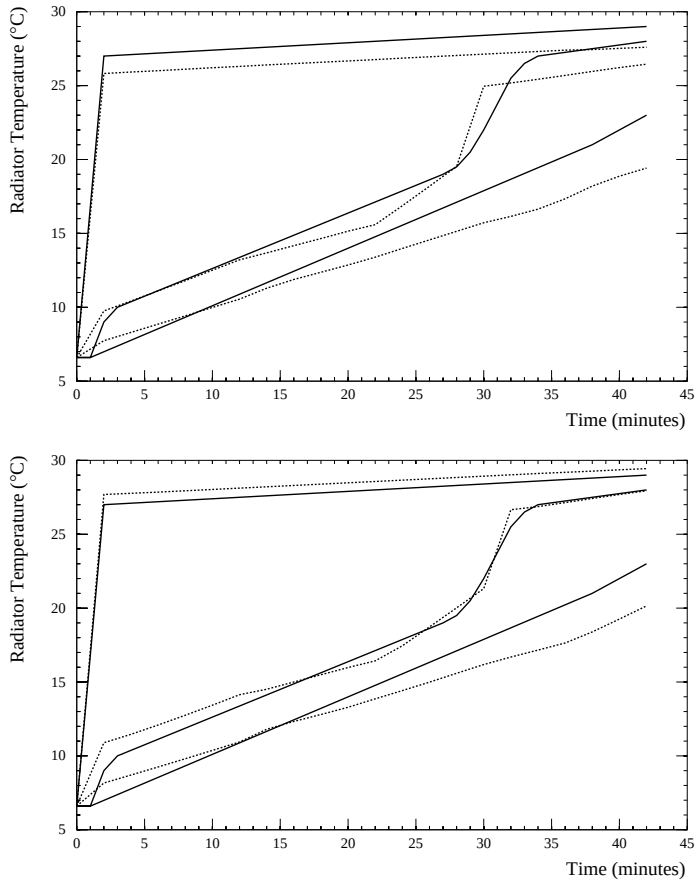

Figure 5: Pre- and Post-correlation Predictions of a CPL Condenser/Radiator Transient Event

Thus, these four uncertainties were treated as correlation parameters, and the model was refined using the Solver as documented in the freely available User's Manual (Ref 7). Figure 5 shows the previous predictions (upper graph) along with the improved predictions (lower graph) which resulted from the automated correlation of the model.

V. SPREADSHEET/SOLVER SUMMARY: THE CONTROL PANEL PARADIGM

Due to the continued success of SINDA/FLUINT, more person-years of development are going into the code now than at any time

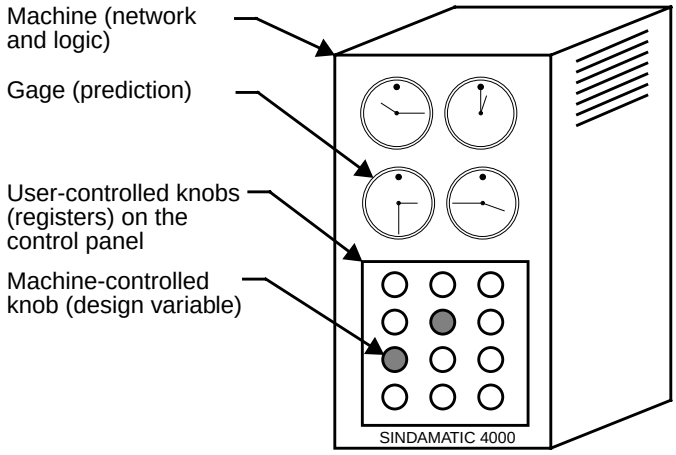

Figure 6: The Control Panel Paradigm

in its history. The new features are widespread and revolutionary, changing not only how the code is used but on which problems.

Unfortunately, these changes have happened so quickly that most users are experiencing a training deficit. Many analysts were just being introduced to the power of registers when both dynamic registers and the Solver were introduced. In addition to freely available training notes, this section is intended to provide a simple overview of these new features by introducing a conceptual paradigm: the control panel (Figure 6).

If a model (a set of thermal and fluid networks and their associated logic) can be thought of as a complex machine, then registers are knobs on the machine's control panel. Registers allow a large, complex model to be constructed using a few basic dimensions, properties, boundary conditions, performance metrics, etc. in a natural, self-documenting algebraic style. Thus, the set of registers becomes a centralized panel for affecting sweeping model changes. A few turns of any knob (register) can completely alter the model (machine), either between runs (statically) or within a run (dynamically, using the UPREG family of routines).

When using the Solver, the user turns over the control of a few designated knobs (the darkened ones in Figure 6) to the program itself, and defines instead the rules by which these knobs should be adjusted, and to what purpose.

VI. IMPROVEMENTS IN PROGRESS

A. Advanced Expressions (Version 4.1)

The built-in spreadsheet in SINDA/FLUINT has proven to be such a popular feature that significant expansions are now in progress. These expansions include:

1. References to response (output) variables such as "smn.T14" (the temperature of node 14 in submodel "smn") are possible in expressions. This will allow the model to adjust itself based on any SINDA/FLUINT parameter including control parameters and a model's own responses.
2. Conditional operations can be used in expressions, allowing single or nested IF/THEN/ELSE type logical switching within an expression. For example, the user can make the diameter of a line be either 4mm or 8mm or 9mm, depending on which of three working fluids has been selected. This feature will also allow the user to easily "store" multiple cases within one model, switching back and forth, for example, between beginning and end-of-life properties, duty cycles, environments, etc.

In essence, each "data value" in old SINDA can now be as complex as a logic block, allowing tremendous variability to be programmed into a model by the user in a fashion that is often more straightforward than are traditional logic blocks.

B. Simultaneous Thermal Solutions (Version 4.1)

An alternative to the iterative solution methods has been introduced for thermal submodels for both steady and transient problems: a simultaneous sparse matrix solution. While iterative methods are robust and forgiving, they are not as repeatable as is a simultaneous solution, are more sensitive to initial conditions, and are often slower for conduction-dominated problems such as those translated from structural FEM meshes. The repeatability of the matrix solution is important when using the Solver to discern minor trends between nearly similar point designs.

Using SINDA/FLUINT, the user can create hybrid solution approaches by customizing the methods used for each submodel. For example, some submodels (perhaps representing highly conductive components such as aluminum base-plates or heat sinks) may use simultaneous methods while others continue to use the traditional iterative methods.

C. Advanced Flow Solutions (Version 4.1 and 4.2)

With its ability to handle steady-state and transient two-phase flows, conjugate heat transfer, slip flow and homogeneous flow, diverse flow regimes, capillary device and phenomena modeling, arbitrary fluids, mixtures of working fluids, and nonequilibrium control volumes, etc., SINDA/FLUINT is one of the most powerful thermal/fluid analyzers available. It has been used in nuclear reactor design, in vapor compression cycle transients, in compressor design, in liquid propulsion system simulations, and of course in the single- and two-phase spacecraft thermal control systems for which it was originally intended.

Because it is user-extensible, it can be applied to systems involving physical phenomena which are not available in prepackaged form within the code. Nonetheless, there are certain phenomena that are either common enough to warrant a prepackaged simulation capability, or that are too cumbersome for the average user to model easily.

Therefore, a multi-year effort is being completed this year that is extending the current two-phase thermal-hydraulic capability to include nonequilibrium duct flow and the complete handling of equilibrium and nonequilibrium dissolution and evolution of gases. Most of these features will be available in Version 4.1 later this

year, while a few advanced nonequilibrium options won't be available until Version 4.2 next year.

SPECIES-SPECIFIC SUCTION - In earlier versions, once substances were mixed they could not be extracted from one another except to the extent that they existed in separate phases (in which case the phase-specific suction options could be used). The user now has the ability to extract any species and/or any phase from an upstream control volume. This option is useful for modeling chemical reactions and certain diffusion phenomena.

NCG DISSOLUTION AND EVOLUTION - At high pressures, a gas will dissolve slowly into a liquid with which it is in contact. At lower pressures, the gas will slowly evolve back out of solution. Both processes are normally diffusion limited.

One of the primary motivations for adding nonequilibrium capabilities is the increased need to model systems in which the transport and influences of noncondensible gases are important. These gases include generated hydrogen in ammonia thermal control systems, helium and nitrogen pressurants in liquid propulsion systems, and nitrogen pressurants in fire retardant delivery systems.

For example, combined dissolution modeling and nonequilibrium methods are needed to estimate the void fraction and gas content of any bubbles that exit a condenser when trace amounts of NCG are present at the inlet. As the vapor condenses along the length of the condenser, some amount of gas will be dissolved into the condensate, but if the residence time is short enough most will remain in the bubble. The difference can be critical in designing devices to trap NCG, since most rely on capillary action to separate a gas-containing bubble from the condensate, and such methods cannot stop the dissolved gases from passing on to the rest of the system. Transient shifting of gas from one place to another in a system (in this case, a capillary pumped loop) has also been evidenced, and attempting to understand the ways in which NCG is transported and evolved can be important for the design and operation of such devices.

A very general and flexible user interface has been created to assist in dissolution/evolution modeling. Dissolution data is difficult to find for many chemical systems, and can come in many forms (e.g., Henry's law coefficients, Raoult's Law coefficients) and perhaps needs to be estimated (e.g., Ostwald coefficients). Also, the presence of other dissolved species or liquids can greatly influence the saturated mole fraction, although fortunately in most applications of interest such complicating factors rarely exist.

Other complexities modeling in the new system include homogenous nucleation of gases, which can be a rather explosive phenomenon.

Gas dissolution/evolution modeling features will be available in Version 4.1 this fall.

CONTROL VOLUME INTERFACES - A new FLUINT network element has been added to Version 4.1 which allows the user to specify a moveable boundary between any two control volumes ("tanks" in FLUINT). These *interfaces* simplify the modeling of liquid/vapor interfaces including those found in bubbles, nonequilibrium (two-fluid) models, and porous structures such as capillary wicks. Interfaces may also be used to model pistons, bladders, bellows, servo-actuators, valve stems, etc.

In fact, interfaces allow a control volume to be arbitrarily subdivided into 2D or even 3D grids, at least for quasi-stagnant flows dominated by thermal effects. This new tool thereby greatly simplifies modeling of thermal stratification in propulsion tanks, for example, or oscillating flows such as those found in pulse tube cryocoolers (a type of thermoacoustic engine). Interfaces also facilitate modeling of more complex transient phenomena in capillary systems such as capillary pumped loops and loop heat pipes.

NONEQUILIBRIUM TWO-PHASE DUCT FLOWS - It is possible for liquid and vapor phases within a control volume to transiently coexist at different temperatures and, perhaps due to surface tension or gravitational effects, at slightly different pressures. This often happens in quasi-stagnant portions of a system where mixing effects are weak, such as reservoirs and accumulators.

Although nonequilibrium modeling tools have been available in SINDA/FLUINT for many years, these tools were explicit "adjunct solutions" and therefore less stable than the remainder of the implicit fluid solution. Interfaces, described above, have eliminated this problem in Version 4.1.

However, the previously existing tools were also limited to quasi-stagnant areas in the model, and could not be easily used to model nonequilibrium within duct flow. When a pressure wave passes through a two-phase duct, the wave speed is underestimated if perfect mixing is assumed. Otherwise, equilibrium methods are perfectly adequate for most analyses of pure constituents because of the strong mixing effects and rapid phase change. However, when noncondensible phases are added to the mixture, then the rates of mass and heat transfer between the phases can be decreased significantly, making the time scales associated with nonequilibrium processes large enough to influence the simulation results if they are not correctly accounted for.

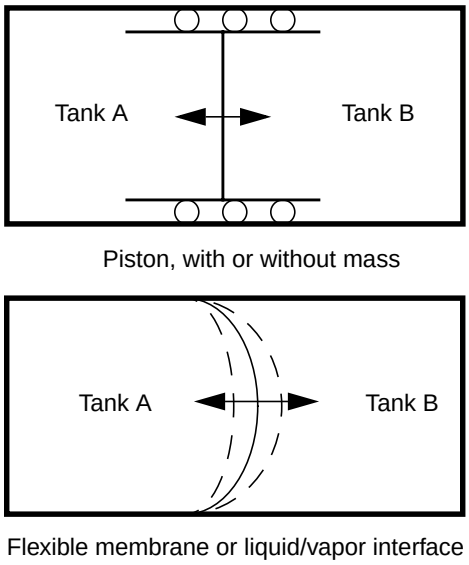

Figure 7: Example Modeling Applications of New Interface Network Elements

Version 4.2 completes the simultaneous implicit solution of full nonequilibrium two-phase ("two fluid") duct flow as a natural option in the code--as an extension of existing two-phase analysis capabilities. The option of including such effects can be turned on or off by the user in a manner analogous to the current handling of slip vs. homogeneous flow. Once completed, the previously existing nonequilibrium tools will be rendered obsolete.

VII. RELATED PROGRESS

Thermal engineers are often unable to take advantage of FEM tools or CAD databases and model building methods without generating unacceptably complicated conduction/radiation models. Available tools were simply not designed with thermal analysis in mind. C&R is therefore introducing the CAD-based Thermal Desktop™. This tool enables thermal engineers to share design data with CAD designers and to exchange information with FEM-based structural engineers, while building sensible and fast-executing system-level thermal models. The first module, a modern radiation analyzer called RadCAD® (Ref 10), has been previously released.

Design optimization and test data correlation, as described in this paper, are important features in the Thermal Desktop. Intrinsic model variability, including SINDA/FLUINT-like registers, are being designed into this tool suite given the success of their implementation in SINDA/FLUINT. For example, RadCAD® is fast enough to be called iteratively from within SINDA/FLUINT solutions, which will enable thermal designers to size or position spacecraft radiators, or to use optical properties as uncertain parameters for correlation to test data.

ACKNOWLEDGMENTS

Neither SINDA/FLUINT nor its companion program Sinaps*Plus* would exist were it not for the continuing support of the Crew and Thermal Systems Division of the NASA Johnson Space Center. Dr. Eugene Ungar and Cynthia Cross are the technical advisors for several recent enhancement projects.

REFERENCES

1. Cullimore, B.A.; and Lin, C.H.: FLUINT: Generalized Fluid System Analysis with SINDA '85. AIAA-87-1466; AIAA 22nd Thermophysics Conference, 1987.
2. Cullimore, B.A; Applications of a Generalized Thermal/Hydraulic Simulation Tool. AIAA-89-1754; AIAA 24th Thermophysics Conference, 1989.
3. Cullimore, B.A; Ring, S.G.; Ungar, E.K.; Development Status of SINDA/FLUINT and SINAPS; Thermal Fluid Analysis Workshop, 1993.
4. Cullimore, B.A.; Graphics-Based Thermal/Fluid Analysis: Recent Developments in SINDA/FLUINT and SINAPS; 25th International Conference on Environmental Systems, 1995.
5. Cullimore, B.A.; Advances in SINDA/FLUINT and SINAPS; 26th International Conference on Environmental Systems, 1996.
6. Cullimore, B.A.; SINDA/FLUINT: Recent and On-going Expansions of the Industry-Standard Thermal/Fluid Analyzer ; 27th International Conference on Environmental Systems, 1997.
7. SINDA/FLUINT User's Manual, Version 4.0; October, 1997; http://www.webcom.com/crtech
8. Sinaps*Plus* User's Manual, Version 4.0; December, 1997.
9. Ring, S.G. and Cullimore, B.A.; A New Interactive Environment for Running and Debugging SINDA/FLUINT Models, Paper 972533; 27th International Conference on Environmental Systems; 1997.
10. Panczak, T. D. and Ring, S.G.; RadCAD: Next Generation Thermal Radiation Analyzer. Paper 972441; 27th International Conference on Environmental Systems, 1997.