LA-UR-97-
October 1997 - 4805

**A BRIEF REVIEW OF THE REFLOOD CLOSURE PACKAGE
OPTIMIZATION EFFORTS PERFORMED WITHIN TRAC 5.4.2SR10**

RECEIVED
APR 06 1998

David A. Pimentel
Ralph A. Nelson

OSTI

19980423 155

Los Alamos
NATIONAL LABORATORY

Photograph by Chris J. Lindberg

LA-UR-97-4805

**A BRIEF REVIEW OF THE REFLOOD CLOSURE PACKAGE OPTIMIZATION
EFFORTS PERFORMED WITHIN TRAC 5.4.25R10**

David A. Pimentel and Ralph A. Nelson, Jr.
Technology & Safety Assessment Division
Nuclear Systems Design & Analysis Group
Los Alamos National Laboratory
Los Alamos, New Mexico 87545

October 1997

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED *ph*

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

A BRIEF REVIEW OF THE REFLOOD CLOSURE PACKAGE OPTIMIZATION EFFORTS PERFORMED WITHIN TRAC 5.4.25R10

ABSTRACT

This report summarizes the implementation of tools within Version 5.4.25R10 of the Transient Reactor Analysis Code (TRAC); this implementation allows the semiautomated optimization of the reflow constitutive package. The tools included a software package external to TRAC that used a line-search method to minimize a generic function value given the function's partial derivative vector with respect to a set of closure coefficients used within TRAC's reflow model. Within TRAC, the generic function was a normalized penalty function dependent on time-averaged calculated values of vapor temperature, vapor void fraction, wall-to-fluid heat-transfer rate (or wall temperature), and the respective steady-state data. The penalty function was implemented only for a one-dimensional vessel configuration because the available reflow data were taken primarily from postcritical-heat-flux tube experiments.

1. OVERVIEW

The first objective of this report is to give a general overview of the optimization methods hand-performed on the various constitutive closure packages contained within the Transient Reactor Analysis Code (TRAC) (Section 2). The second objective is to describe in general the efforts made toward the creation of tools (Section 3), both within and external to TRAC, that would allow for the semiautomated optimization of the closure packages. These packages consist of correlations related to the new reflow model described in Ref. 1. The optimization tools that were developed include the calculation of both (1) a general function (penalty function), which is dependent on coefficients used in the closure packages, and (2) the function's partial derivative vector with respect to all of the coefficients. These calculations (Sections 3.3, 3.4, and 3.5) are performed by coding within TRAC itself. The calculated function value and its partial derivative vector are exported via files (Section 3.7) to an external software package (Section 3) that performs a line-search method to minimize the general function. Because the available experimental data specifically related to the reflow closure package is that of steady-state and quasi-steady-state tube and bundle critical heat flux (CHF) and post-CHF experiments, the penalty function and its gradient are calculated using time-averaged values of the selected system parameters. Also, the implementation of this optimization method currently is limited to a one-dimensional (1D) vessel with

a single heat structure that is specified by component number within a data file (Section 3.7).

This methodology was implemented within the 5.4.25R10 version of TRAC (Section 3.6) that includes the aforementioned new reflood model. The source code of TRAC and the optimization software package are archived on Central file system (CFS) storage space at Los Alamos National Laboratory (LANL) (Section 6).

2. METHODOLOGY—CLOSURE DEVELOPMENT USING NONLINEAR OPTIMIZATION

The optimization method implemented within TRAC 5.4.25R10 uses a variation of a penalty function,² which is often used in the nonlinear optimization of multidimensional equations.

2.1. General Optimization

The general optimization problem for a two-fluid code such as TRAC regarding the development of closure relationships becomes

$$\left\{ \begin{array}{l} P(\bar{z}, t, \{c_{ip}\}) \\ \alpha(\bar{z}, t, \{c_{ip}\}) \\ T_l(\bar{z}, t, \{c_{ip}\}) \\ T_v(\bar{z}, t, \{c_{ip}\}) \\ v_l(\bar{z}, t, \{c_{ip}\}) \\ v_v(\bar{z}, t, \{c_{ip}\}) \\ T_w(\bar{z}, t, \{c_{ip}\}) \end{array} \right\}, \quad (2-1)$$

where the equation set is that required to yield Eq. (2-1) plus a penalty function, ϕ , and $\{c_{ip}\}$ represents the set of closure coefficients. The penalty function is arbitrary, depending on the end result desired. However, for the sake of discussion, we will define it initially to be

$$\phi = \sum_{j=1}^J \sum_{k=1}^{K_j} \gamma_j (\beta - 1)_{jk}^2, \quad (2-2)$$

where β represents the ratio of a code calculated quantity to its measurement, K_j is defined as the number of data points available from an experiment that correspond to the chosen calculated quantity, and J is the number of calculated quantities to be optimized (i.e., T_w , α , q'' , ...). Also, γ_j is a weighting factor that is defined by the

analyst and that is to be used to emphasize or deemphasize a specific quantity. As an example, T_w might be chosen as the only variable to optimize. This choice would yield

$$\phi = \sum_{k=1}^K \left(\frac{T_{w,calc}}{T_{w,data}} - 1 \right)_k^2 \quad (2-3)$$

as the penalty function, where K is defined as the number of thermocouples used in an experiment. This would be the only possible comparison to be made for an old post-critical-heat-flux (CHF) tube experiment.

For a single-closure coefficient, $\{c_{tp}\} = c_1$, it can be argued that the solution might be unique because one equation has been added to the original equation set. But for $tp \geq 2$, the system must be underspecified or underdetermined. This implies the solution is not unique, and the determination of c_n would depend on $\{c_{tp}\}$ where $tp \leq n-1$.

The use of an optimizer to determine $\{c_{tp}\}$ in this general formulation requires the generation of

$$\frac{\partial \beta}{\partial c_{tp}} = \left\{ \begin{array}{c} \frac{\partial P}{\partial c_{tp}} \\ \frac{\partial \alpha}{\partial c_{tp}} \\ \frac{\partial T_l}{\partial c_{tp}} \\ \frac{\partial T_v}{\partial c_{tp}} \\ \frac{\partial v_l}{\partial c_{tp}} \\ \frac{\partial v_v}{\partial c_{tp}} \\ \frac{\partial T_w}{\partial c_{tp}} \end{array} \right\} (\vec{z}, t) \quad (2-4)$$

for all tp .

2.2. Optimization Methodology Considerations for TRAC

Although Eqs. (2-1), (2-2), and (2-4) represent the general optimization problem for closure development, we cannot solve this problem. The difficulty is that β does not exist for all of the independent parameters reflected in Eq. (2-2). Thus, typically due to the lack of hydrodynamic data, the penalty function cannot be constrained to ensure that the proper hydrodynamic state exists. For example, if an old post-CHF tube experiment were used, the pressure, mass flux, heat flux, and a set of tube wall temperatures would be known, which would allow the TRAC input model and the definition of a penalty function like Eq. (2-3) to be built. However, the system is underspecified, and hydrodynamic information is not part of the penalty function. Thus, determination of the "proper hydro state" cannot be ensured.

2.2.1. Original TRAC MOD2 Reflood Model

Consider the recorrelation of the original MOD2 reflood models,^{3,4} which is a four-step process:

1. Recorrelate interfacial drag to predict the Cylindrical Core Test Facility (CCTF) Run 14 void profiles using fixed heat-transfer coefficients (both wall and interfacial).
2. Recorrelate wall post-CHF heat transfer to predict a set of tube experiments, assuming that the drag from step 1 is correct and retaining a fixed set of interfacial heat-transfer coefficients.
3. Recorrelate the interfacial heat transfer to predict vapor superheat measured in selected post-CHF tube experiments, assuming that the drag and wall heat transfer determined in steps 1 and 2 are correct.
4. Repeat step 1 using new heat-transfer coefficients. Continue repeating this process until data predictions are acceptable.

Effectively, the "individual penalty function" associated with each step was manually performed, where "brain power" served as an extremely limited optimizer. Thus (regarding only step 1, for example), the following equation was evaluated manually:

$$\phi = \sum_{j=1}^J \sum_{k=1}^{K_j} \gamma_j \left(\frac{\alpha_{calc}}{\alpha_{data}} - 1 \right)_{jk}^2 . \quad (2-5)$$

Unfortunately, recorrelating all of the interfacial drag coefficients at one time was too complex for the optimizer; thus, each step was separated further into subregions. For example, the nucleate boiling coefficients first were recorrelated by considering the end of the transient when everything was quenched. Next, post-CHF boiling

regions were added to try to limit the addition to one new region at a time by selecting other experimental time frames to analyze.

This process can be represented by defining a limited subset of the $\{c_{tp}\}$ in Eq. (2-2) to be "optimized in each subregion." For the first substep of nucleate boiling, a small number of coefficients (perhaps two or three) are valid for recorrelation, i.e., $\{c_{tp}\}$,

where

$$tp = \left\{ \begin{array}{l} \text{nucleate boiling} \\ \text{interfacial drag} \\ \text{coefficients} \end{array} \right\}$$

or

$$\{c_{tp}\}_{NB,drag}$$

After completing this task, assume the next state to be added is the highly dispersed film boiling region, and denote this set as $\{c_{tp}\}_{HD,drag}$. Realize that this second recorrelation substep depends on the assumption $\{c_{tp}\}_{NB,drag}$ is correct and remains untouched. This new limited set may be written as

$$\{c_{tp}\}_{set,drag} = \left\{ \{c_{tp}\}_{NB,drag}, \{c_{tp}\}_{HD,drag} \right\}. \quad (2-6)$$

Remember that the system is underdetermined. Effectively, this process is like building up the underdetermined states (regions) one at a time. In the end, it should lead to a valid solution, but not necessarily the only valid solution.

This process is continued until the entire set of interfacial drag coefficients is defined for step 1 and can be written as

$$\{c_{tp}\}_{drag} = \left\{ \{c_{tp}\}_{NB}, \{c_{tp}\}_{HD}, \{c_{tp}\}_{dispersed}, \{c_{tp}\}_{RW}, \dots \right\}_{drag}, \quad (2-7)$$

where the subscripts *dispersed* and *RW* represent the dispersed and rough-wavy post-CHF regions, respectively.

Once step 1 is completed for the interfacial drag as represented by Eq. (2-7), step 2 follows, continuing the subregion process as often as possible. For example, if an experiment is selected with only post-CHF highly dispersed heat transfer (HDHT), then

$$\{c_{tp}\}_{\text{set}} = \left\{ \{c_{tp}\}_{\text{drag}}, \{c_{tp}\}_{\text{HDHT}} \right\}, \quad (2-8)$$

where $\{c_{tp}\}_{\text{drag}}$ is the set represented by Eq. (2-7).

The process continues until at the end of step 3 the complete set $\{c_{tp}\}$ is produced and is represented by

$$\{c_{tp}\} = \left\{ \{c_{tp}\}_{\text{drag}}, \{c_{tp}\}_{\text{wall heat transfer}}, \{c_{tp}\}_{\text{interface heat transfer}} \right\}, \quad (2-9)$$

and the sequence of operation has been to define the various subsets of coefficients on the right-hand side of Eq. (2-9) from left to right, assuming that all other sets are constant. Having reached this point in the process, we can now return to step 1 using the new $\{c_{tp}\}_{\text{wall heat transfer}}$ and $\{c_{tp}\}_{\text{interface heat transfer}}$ to refine $\{c_{tp}\}_{\text{drag}}$ further. This iteration can be repeated as many times as needed; however, during the development of MOD2, it was found that two iterations produced reasonable results.

2.2.2. Original NRC SET and Code Development Philosophy

To represent the original NRC philosophy, we will write

$$\{c_{tp}\} = \left\{ \{c_{tp}\}_{\text{SET1}}, \{c_{tp}\}_{\text{SET2}}, \{c_{tp}\}_{\text{SET3}}, \dots \right\}, \quad (2-10)$$

where the subscript SET1 represents one Separate Effects Test (SET) defined to produce a measure of a particular closure phenomenon. The data from this experiment could be used to define $\{c_{tp}\}_{\text{SET1}}$.^a Through a process of running a number of SETs, we then could arrive at a complete set of closure relationships. Although not as clean as the original NRC code development SET philosophy, the development of the TRAC MOD2 reflood model was meant to be an approximate application of it. The general philosophy represented within Eq. (2-10) also can be more flexible than that just noted. Some of the SET experiments may be designed to produce specific coefficient sets alone,^b and these might be denoted as $\{c_{tp}\}_{\text{SET1}}$ and $\{c_{tp}\}_{\text{SET3}}$ for the sake of discussion. Others then may be built off of these initial sets. For example, $\{c_{tp}\}_{\text{SET2}}$ might be developed based on $\{c_{tp}\}_{\text{SET1}}$. An experiment where the walls are adiabatic followed by an experiment where the walls are heated would be an example of this sequence. The initial adiabatic experiment might be used to

^a One can read $\{c_{tp}\}_{\text{SET1}}$ as correlation or closure relationship in this case.

^b Independent of the other sets.

develop the drag coefficients, whereas the heated experiment might provide the wall heat-transfer coefficients, assuming that the drag coefficients are correct.

2.2.3. Closure Development Using Nonlinear Optimization

We would like to remain as true to the process implied by Eq. (2-10) as possible within the current closure development, realizing that there are significant constraints on the data available for the phenomena that we need to model.

Optimization also may allow greater flexibility in the development of $\{c_{tp}\}_{SET1}$ and $\{c_{tp}\}_{SET2}$ than that just noted above. The initial sets of quantities can be developed as just discussed. These sets then can be used as a starting point for a solution of the combined sets of coefficients and data (i.e., $\{c_{tp}\}_{SET1}$ is not constant during final development). This may produce a better set of coefficients than can be obtained otherwise. However, this point requires further investigation.

2.3. Generalized Penalty Function

The optimization process described above requires a set of automated tools for recorrelation of the TRAC closure coefficients. The method selected involves an iterative implementation of the penalty function defined by Eq. (2-2). This function is nondimensionalized and normalized to produce

$$\phi = \sum_{e=1}^E \sum_{j=1}^J \sum_{k=1}^{K_{ej}} \frac{r\gamma_j}{K_{ej}} (\hat{\beta} - 1)_{ejk}^{\xi} , \quad (2-11)$$

where E is the number of TRAC input decks to be used for each determination of a penalty function value and its gradient, r , is a dynamically determined convergence acceleration parameter that was patterned from Ref. 2, $\xi = 2$, and

$$\hat{\beta} = \frac{x_{calc}}{x_{data}} . \quad (2-12)$$

Here, x is some quantity that is either calculated or measured. It follows that the gradient of the penalty function in Eq. (2-11) is defined as

$$\frac{\partial \phi}{\partial c_{tp}} = \sum_{e=1}^E \sum_{j=1}^J \sum_{k=1}^{K_{ej}} \left\{ \frac{r\gamma_j \xi}{K_{ej}} (\hat{\beta} - 1)_{ejk}^{\xi-1} \left(\frac{\partial \hat{\beta}}{\partial c_{tp}} \right)_{ejk} \right\} , \quad (2-13)$$

where $\hat{\beta}$ is averaged over a specified region of time. These partial derivatives can be calculated within TRAC by implementing subroutines generated by ADIFOR.

It is important to check that the averaged quantity $\hat{\beta}$ is stationary to reflect behavior around a limit attractor or strange attractor. If this is not the case, the characteristics of the behavior must be investigated.

3. OPTIMIZATION IMPLEMENTATION

The implementation of the preceding optimization discussion required a software package to perform the minimization of Eq. (2-11). The selected Minimizer^c is a Limited Memory Quasi-Newton Solver using a line search method that implements a direct primal method to determine a search direction.^{5,6} The Optimizer^d is a standalone program that optimizes TRAC. In addition to this optimizer package, the ADIFOR version 2.0 program⁷ was used to develop implicitly derived code that calculates partial derivative values of specific system parameters with respect to the closure coefficients. Using an automatic differentiation program such as ADIFOR has many distinct advantages over the manual coding of derivatives, as discussed in Refs. 8 and 9.

3.1. Optimization Convergence Criteria

The Optimizer has been equipped with a convergence criterion based on the tunable parameter values (i.e., independent variables of the penalty function), and the Minimizer has intrinsic convergence criteria based on both user-specified values and machine precision. First, the Optimizer considers the relative change of the tunable parameters with a stopping criterion of

$$\left| \frac{c_{tp,n} - c_{tp,n+1}}{c_{tp,n+1}} \right| \leq 10^{-14} , \quad (3-1)$$

which, when active, will preempt a restart in the steepest direction for a failed line search.⁶ Second, the Minimizer considers the relative change of the penalty function value

$$\frac{(f_n - f_{n+1})}{\max(|f_n|, |f_{n+1}|, 1)} \leq \vartheta M_p \quad \begin{cases} \vartheta = \text{user-defined factor} \\ M_p = \text{machine precision} \end{cases} \quad (3-2)$$

where the machine precision is typically on the order of 10^{-16} for double-precision accuracy. For the TRAC optimization process, $\vartheta \equiv 10^{12}$. Finally, the Minimizer considers the projected gradient (projg), which is the projection of the penalty function's gradient vector onto the space tangent to the active bounds. When the infinity norm of this projection meets the criterion

^c L-BFGS-B, version 2.3, written in FORTRAN 77 (F77).

^d F77 driver to run Minimizer and TRAC.

$$\|\text{proj } g\|_{\infty} \leq \varepsilon_{grad} , \quad (3-3)$$

the optimization process is terminated. The test of Eq. (3-3) will be very difficult to satisfy if ε_{grad} is set to anything smaller than $\sqrt{M_p}$. For the purposes of the TRAC optimization, ε_{grad} is set to 10^{-5} . The subscript n in Eqs. (3-1), (3-2), and (3-3) refers to a specific Minimizer iteration.

3.2. Optimization Algorithm

The following algorithm outlines the method of optimizing TRAC with respect to specified closure parameters:

1. Start: choose initial values for r and $\{c_{tp}\}$ (realize that $\{c_{tp}\}$ will be divided into two subsets (one set that remains constant and a much smaller set to be recorrelated). Also, the Optimizer requires knowledge of the type of independent variables' bounds and the test numbers corresponding to the desired TRAC input decks.
2. If the convergence criterion of Eq. (3-1), (3-2), or (3-3) is true, then quit the algorithm.
3. Run TRAC using the input specified within *decklist* (described below) to calculate the penalty function and its gradient with respect to $\{c_{tp}\}_{recorrelate}$.
4. If TRAC fails to generate ϕ and $\frac{\partial \phi}{\partial c_{tp}}$, then the current values of $\{c_{tp}\}_{recorrelate}$ are inappropriate for the current TRAC input. Reset $\{c_{tp}\}_{recorrelate} = \frac{\{c_{tp}\}_{recorrelate,current} + \{c_{tp}\}_{recorrelate,previous}}{2}$ and get the first input set specified within *decklist*. Return to step 3.
5. If another set of input is specified for TRAC within *decklist*, retrieve it from file and return to step 3.
6. Generate new $\{c_{tp}\}_{recorrelate}$ values^e based on the current values of ϕ and $\frac{\partial \phi}{\partial c_{tp}}$.

^e Performed by the Minimizer.

7. If the norm of the gradient, $\left\| \frac{\partial \phi}{\partial c_{tp}} \right\| \leq \zeta$, then $r \leftarrow \Omega r$, where Ω and ζ are user-defined parameters with typical values of 10 and 10^{-2} , respectively. This step was deactivated with $\Omega < 0$, and r had a constant value of one for the optimization of TRAC documented here.

8. Return to step 2.

3.3. Interfacial Heat-Transfer Optimization

Using ADIFOR, the product of the interfacial heat-transfer coefficient and the interfacial area was differentiated with respect to several closure parameters and the vapor temperature. The product and independent variables can be found in Table 1.

Within the subroutine generated by ADIFOR, the dependence of vapor temperature on the tunable parameters was not visible; therefore, the partial derivative in

Eq. (2-13), $\left. \frac{\partial \hat{\beta}}{\partial c_{tp}} \right|_{j=1} = \frac{1}{T_{vap,data}} \frac{\partial T_{vap,calc}}{\partial c_{tp}}$, was determined using

$$\frac{\partial T_{vap,calc}}{\partial c_{tp}} = \frac{\partial \dot{q}_{vap,calc}}{\partial c_{tp}} \left(\frac{\partial \dot{q}_{vap,calc}}{\partial T_{vap,calc}} \right)^{-1}, \quad (3-4)$$

where c_i is an arbitrary closure parameter, and

$$\dot{q}_{vap,calc} = (hA)_{vap,calc} (T_{vap,calc} - T_{sat}). \quad (3-5)$$

The ADIFOR generated routine was used to calculate all of the derivatives on the right-hand side of Eq. (3-4).

3.4. Interfacial Drag Optimization

Once again using ADIFOR, the interfacial drag coefficient was differentiated with respect to several closure parameters within TRAC. The said interfacial drag coefficient and closure variables are described in Table 2.

Within the subroutine generated by ADIFOR, the dependence of the drag coefficient on the void fraction was not visible; therefore, the partial derivative $\frac{\partial \alpha_{calc}}{\partial C_i}$ was

determined from second-order polynomial regressions of the time-averaged values of α_{calc} and C_i at each hydro-cell-centered elevation. These polynomials were calculated using α_{calc} and C_i at five axial positions with the current cell as a center. Using the ADIFOR-generated gradient vector in conjunction with the regressions'

derivatives allowed the partial derivative in Eq. (2-13), $\left. \frac{\partial \hat{\beta}}{\partial c_{tp}} \right|_{j=2} = \frac{1}{\alpha_{data}} \frac{\partial \alpha_{calc}}{\partial c_{tp}}$, to be determined using

$$\frac{\partial \alpha_{calc}}{\partial c_{tp}} = \frac{\partial \alpha_{calc}}{\partial C_i} \frac{\partial C_i}{\partial c_{tp}}. \quad (3-6)$$

3.5. Wall Heat-Transfer Optimization

Using ADIFOR, the total wall heat flux was differentiated with respect to several closure parameters and the heated wall temperature. The said heat flux, closure variables, and temperature are described in Table 3.

The objective of this component of the overall penalty function was to allow the user to select the system variable to optimize (q''_{wall} or T_{wall}), depending on the available experimental data. For the q''_{wall} option, the ADIFOR routine was capable of

calculating the partial derivative in Eq. (2-13), $\left. \frac{\partial \hat{\beta}}{\partial c_{tp}} \right|_{j=3} = \frac{1}{q''_{wall,data}} \frac{\partial q''_{wall,calc}}{\partial c_{tp}}$, directly;

however, the chain rule had to be implemented for the T_{wall} option,

$$\left. \frac{\partial \hat{\beta}}{\partial c_{tp}} \right|_{j=3} = \frac{1}{T_{wall,data}} \frac{\partial T_{wall,calc}}{\partial c_{tp}} = \frac{1}{T_{wall,data}} \frac{\partial T_{wall,calc}}{\partial q''_{wall,calc}} \frac{\partial q''_{wall,calc}}{\partial c_{tp}}. \quad (3-7)$$

3.6. General Descriptions of TRAC Subroutines Used for Optimization

Table 4 includes brief descriptions of the subroutines and named common blocks used for the optimization of the reflood closure packages. Within the subroutine `htifn`, the following logic exists:

```

if (ad4code) then
  call g_corifht(alp,alpo,rov,rol,visv,visl,p,arv,arl,cthi,alv,dr,
1      wa,tln,tn,dx,vol,jcell,hla,cl,rova,cv,tssn,
2      sidx,favol,sigma,vvvol,vlvol,dalva,hgam,hfg,
3      gravol,alpmn,alpmx,vlvc,fasmlt,vlalp,finan,
4      bitn,bit,vrv,wfsv,gavv,hdv,hlatw,hdva,chtia,
5      alve,tsat,spifz,xfunh,
6      xzchf,xzth,xzsm,xzag,xalpsm,xalpag,
7      icrl,icru,icrr,nsgrid,zsgrid,irefld,njflip,
      lcore,lld,lcond,flip)
else
  call corifht(alp,alpo,rov,rol,visv,visl,p,arv,arl,cthi,alv,dr,
1      wa,tln,tn,dx,vol,jcell,hla,cl,rova,cv,tssn,
2      sidx,favol,sigma,vvvol,vlvol,dalva,hgam,hfg,
3      gravol,alpmn,alpmx,vlvc,fasmlt,vlalp,finan,
4      bitn,bit,vrv,wfsv,gavv,hdv,hlatw,hdva,chtia,
5      alve,tsat,spifz,xfunh,

```

```

5          xzchf,xztb,xzsm,xzag,xalpsm,xalpag,
6          icrl,icru,icrr,nsgrid,zsgrid,irefld,njflip,
7          lcore,lld,lcond,flip)
endif

```

Within the subroutine `cif3n`, the following logic exists:

```

if (ad4code) then
  call g_rbshrz
else
  call rbshrz
endif

```

Within the subroutine `htc`, the following logic exists:

```

if (ad4code) then
  call g_htvssln(j,tave, roave, cpave, cnave,
&      emave, a(lhdr+k5), a(lpr+k5), a(ltsatr+k5), a(lhfgr+k5),
&      a(lclr+k5), a(lcvr+k5), a(lcplr+k5), a(lcpvr+k5),
&      a(lrolr+k5),
&      a(lrovr+k5), a(lromr+k5), a(lvislr+k5), a(lvisvr+k5),
&      alpha, a(lsigr+k5), odum, vlin, vlzp, zdum1,
&      vvin, a(lvvcr+kx), zdum2, a(lvmzr+kx),
&      zdum3, zdum4,
&      a(ltlr+k5), a(ltvr+k5), zslab, pdratr, idum1,
&      a(lelr+k5), a(levr+k5), a(ldrvdt+k5), a(ldrldt+k5),
&      a(lear+k5), a(lroar+k5), a(lpar+k5), a(ltssnr+k5),
&      a(lhlslr+k5), a(lhvsr+k5), hrg,
&      hl,          hv,          qchf,          tchf,
&      a(lihtf+k7), a(lstnu+k7), a(ltld+k7), dtstepr,frsttime)
else
  call htvssln(j,tave, roave, cpave, cnave,
&      emave, a(lhdr+k5), a(lpr+k5), a(ltsatr+k5), a(lhfgr+k5),
&      a(lclr+k5), a(lcvr+k5), a(lcplr+k5), a(lcpvr+k5),
&      a(lrolr+k5),
&      a(lrovr+k5), a(lromr+k5), a(lvislr+k5), a(lvisvr+k5),
&      alpha, a(lsigr+k5), odum, vlin, vlzp, zdum1,
&      vvin, a(lvvcr+kx), zdum2, a(lvmzr+kx),
&      zdum3, zdum4,
&      a(ltlr+k5), a(ltvr+k5), zslab, pdratr, idum1,
&      a(lelr+k5), a(levr+k5), a(ldrvdt+k5), a(ldrldt+k5),
&      a(lear+k5), a(lroar+k5), a(lpar+k5), a(ltssnr+k5),
&      a(lhlslr+k5), a(lhvsr+k5), hrg,
&      hl,          hv,          qchf,          tchf,
&      a(lihtf+k7), a(lstnu+k7), a(ltld+k7), dtstepr,frsttime)
endif

```

3.7. Optimization Input/Output File Descriptions

The Optimizer and TRAC use several files to communicate with one another and the analyst. First, the TRAC input decks must be stored in a subdirectory of the

current working directory called *DECKS*. Within *DECKS*, the files are named *###.dmp*,^f *tracin.###.rst*, and *data.###*. The files are the TRAC binary dump file, the TRAC restart input deck, and the experimental data file, respectively. An example of the TRAC restart input deck follows some comments. For the details of the restart input deck's format see the TRAC Code Manual.⁴ In addition to the referenced information, a few input variables of specific interest are discussed as follows:

1. *inopts*—defined as 2 to activate the *rafopts* namelist variables.
2. *ntune*—number of parameter values to read from *rafin* by TRAC.
3. *irafset*—indicates closure coefficient set to use; 1 = Conditional (default), 2 = Absolute.
4. *raftime*—zero or greater value indicates TRAC to read *rafin* values; negative value indicates TRAC to ignore values in *rafin*.
5. *rafopt*—value of 1 indicates TRAC to activate closure parameter logic.

```

free format
*
*****
* main data *
*****
*
*          numtcr          ieos          inopt          nmat
*              16              0              2              1
*
*****
*****
*****
*****
*
*          winfrith-upflow post-chf heat transfer experiment
*          test 405 : xeo = subcooled (xeo=-0.0167 at dryout point)
*                  p   = 2.2200E+05 Pa
*                  g   = 1.9960E+03 kg/m**2-s
*                  P   = 2.2180E+03 watts
*
*****
*****
*****
*
*
*****
* namelist data *

```

^f ### refers to a test number.


```

*****
*
&rafopts ntune=48,   irafset=1,   raftime=0.0,   rafopt=1.0 &
&inopts icflow=0, nhtstr=1, nrslv=1, ipatch=999, newrfd=1,
ipowr=-1, tpowr=0.0, idiag=1, iadded=10 &
*
*      dstep      timet
*      -1      -1.0000e+00
*      stdyst      transi      ncomp      njun      ipak
*      0      1      6      4      1
*      epso      epss
*      1.0000e-03      0.0000e-00
*      oitmax      sitmax      isolut      ncontr
*      10      0      0      0
*      ntsv      ntcb      ntcf      ntrp      ntcp
*      1      0      0      1      1
*
*****
* component-number data *
*****
*
* iorder*      1      2      3      4      5
* iorder*      999e
*
*****
* material-properties data *
*****
*
* matb *      123e
* ptbln *      2e
*
* prptb(1,i)      prptb(2,i)      prptb(3,i)      prptb(4,i)      prptb(5,i)
* 2.0000e+02      8.3700e+03      4.6100e+02      1.0006e+01      0.0000e+00
* 5.0000e+03      8.3700e+03      4.6100e+02      9.7435e+01      0.0000e+00
e
*
*****
* control-parameter data *
*****
*
*
* signal variables
*      idsv      isvn      ilcn      icn1      icn2
*      1      0      0      0      0
*
* trips
*      ntse      ntct      ntsf      ntdp      ntsd
*      0      0      0      0      0
*      idtp      isrt      iset      itst      idsg
*      -1      2      0      1      1
*      setp(1)      setp(2)
*      0.0000e+00      0.0000e+00
*      dtsp(1)      dtsp(2)

```

```

0.0000e+00      0.0000e+00
*      ifsp(1)      ifsp(2)
*      0            0
*
*****
* component data *
*****
*
end
*
*****
*****
*****
*****      time step control      *****
*****
*****
*****
*****
*****
*
*      dtmin      dtmax      tend      rtwfp      powerc
*      1.0e-05      0.005      40.0      -10.0      1.0
*      edint      gfint      dmpint      sedint
*      1.0      0.1      5.0      10.0
*****
*
-1.0

```

An example of the *data.###* file, which will be named DATA by the Optimizer so that TRAC can recognize it, is shown in Fig. 3-1.

The current value of the convergence parameter, r , is stored in a file called *rvalue* within the current working directory, which is read by TRAC and is both read and written by the Optimizer. If *rvalue* does not exist, then r has a default value of 1. The following is an example of the file called *rvalue* that is read by TRAC and read/written by the Optimizer. The format of the file is limited to only the following:

1. The first 20 columns are reserved for the value of r in double-precision scientific notation, as shown below.

```
1.0000000000000000E+00
```

TRAC writes the current tunable parameter values to a file called *tune.out*. The following is an example of the file called *tune.out* that is written by TRAC and read by the Optimizer. The format of the file is limited to the following three items:

1. The first row is reserved for the value of the variable storing the number of tunable parameters (ntune).

2. The first 13 columns are reserved for the variable label.
3. Columns 14 through 33 contain the number of tunable parameters (ntune) in integer format or the current tunable parameter value in double-precision scientific notation, as shown.

ntune	25
fgam2	3.00000000000000E+01
frw	1.25000000000000E+00
rel1	1.87500000000000E+01
rel2	1.00000000000000E-04
rel3	-3.00000000000000E-06
ret1	1.04250000000000E+01
ret2	-1.92500000000000E-03
ret3	1.25000000000000E-07
dqcmax	2.00000000000000E+01
dqcmim	2.00000000000000E+00
dqdzmx	2.00000000000000E+03
adqdz	8.00000000000000E-01
xdqdz	1.50000000000000E+00
fac013	9.42800000000000E-01
fac014	3.00000000000000E-03
fac015	1.30000000000000E-01
fac016	5.90000000000000E-01
fac017	1.00000000000000E-01
fac018	1.50000000000000E+01

1	3	
0.945	703.15	
0.141	0.6863	
0.452	0.9038	
0.797	0.9580	
999		
qwall		
0.009	623.40	1.150E+05
0.015	617.20	1.147E+05
0.020	614.10	1.143E+05
0.040	613.70	1.121E+05
0.080	632.50	1.082E+05
0.160	655.30	1.049E+05
0.310	674.10	1.024E+05
0.460	697.40	1.007E+05
0.607	719.70	1.013E+05
0.758	727.30	1.044E+05
0.840	731.90	1.041E+05
0.880	738.20	9.835E+04
0.900	718.50	9.699E+04
0.910	692.70	9.752E+04

← T_{vapor} data cards

← α_{vapor} data cards

← component desired for time-averaging (mandatory)

← qwflg value for wall HT penalty function (mandatory)

← z from top of lower HP (m), wall TC (°C), and HF (W/°Cm²) data (mandatory)

Fig. 3-1. Example of the *data###* file.

fwebb	1.2000000000000E+00
facthv	1.0000000000000E-02
fac019	2.3000000000000E-02
facth11	5.0000000000000E-01
facth12	9.0000000000000E-01
fdham	1.0000000000000E+00

The Optimizer reads from *tune.out* and writes new values to a file called *rafin*, which must exist in the current working directory before running the Optimizer because it is required by TRAC during the optimization process. The following is an example of *rafin* that is read by TRAC and written by the Optimizer. The format of the file is limited to the following three items:

1. Columns 1 through 5 are ignored.
2. Columns 6 through 13 are reserved for the variable label.
3. Columns 14 through 33 are reserved for the tunable parameter value in double-precision scientific notation, as shown.

fgam2	3.0000000000000E+01	used in CORIFHT (RAFHTIF)
frw	1.2500000000000E+00	used in RBSHRZ (IFCRS)
rel1	1.8750000000000E+01	used in HTVSSLN (RAFHTIF)
rel2	1.0000000000000E-04	used in HTVSSLN (RAFHTIF)
rel3	-3.0000000000000E-06	used in HTVSSLN (RAFHTIF)
ret1	1.0425000000000E+01	used in HTVSSLN (RAFHTIF)
ret2	-1.9250000000000E-03	used in HTVSSLN (RAFHTIF)
ret3	1.2500000000000E-07	used in HTVSSLN (RAFHTIF)
dqcmx	2.0000000000000E+01	used in HTVSSLN (RAFHTIF)
dqcmim	2.0000000000000E+00	used in HTVSSLN (RAFHTIF)
dqdzmx	2.0000000000000E+03	used in HTVSSLN (RAFHTIF)
adqdz	8.0000000000000E-01	used in HTVSSLN (RAFHTIF)
xdqdz	1.5000000000000E+00	used in HTVSSLN (RAFHTIF)
fac013	9.4280000000000E-01	used in HTVSSLN (RAFHTIF)
fac014	3.0000000000000E-03	used in HTVSSLN (RAFHTIF)
fac015	1.3000000000000E-01	used in HTVSSLN (RAFHTIF)
fac016	5.9000000000000E-01	used in HTVSSLN (RAFHTIF)
fac017	1.0000000000000E-01	used in HTVSSLN (RAFHTIF)
fwebb	1.2000000000000E+00	used in HTVSSLN (RAFHTIF)
facthv	1.0000000000000E-02	used in HTVSSLN (RAFHTIF)
fac019	2.3000000000000E-02	used in HTVSSLN (RAFHTIF)
facth11	5.0000000000000E-01	used in HTVSSLN (RAFHTIF)
facth12	9.0000000000000E-01	used in HTVSSLN (RAFHTIF)
fdham	1.0000000000000E+00	used in HTVSSLN (RAFHTIF)

A file called *gamma* within the current working directory will contain the values of and a string of characters corresponding to each parameter γ_j . The strings of characters are IFHT, VOID, and QWAL (all caps), corresponding to the interfacial heat transfer, void fraction, and wall heat-transfer components of the penalty

function, respectively. If the file *gamma* does not exist or does not contain a value for a specific penalty function component, the default value is zero. The following is an example of the file called *gamma* that is read by TRAC and the Optimizer. The format of the file is limited to the following three items:

1. The first 20 columns are reserved for the value of γ_j in double-precision scientific notation, as shown below.
2. Columns 21 and 22 must be blank.
3. One of the labels shown below then must be placed, starting in column 33.

0.000000000000000E+00	IFHT
0.000000000000000E+00	VOID
1.000000000000000E+00	QWAL

A file called *bounds* within the current working directory may contain the word BOUNDED,^g LOBOUNDED,^h or HIBOUNDEDⁱ (all caps). If one of these options is selected, then the second line of *bounds* will contain a number (η) by which the initial independent variable values^j ($\{c_{ip}\}_{initial}$) will be multiplied to obtain upper bounds. For all bounded options, the lower bounds are preset to 10^{-10} . If *bounds* does not exist in the current working directory, then no bounds are placed on the independent variables and the Optimizer considers the problem to be unconstrained. The following is an example of the file called *bounds* that is read/written by the Optimizer. The format of the file is limited to the following two items:

1. The first line is reserved for the label indicating desired bounds.
2. The second line is reserved for the factor by which bounds are calculated based on the initial tunable parameter values found in *rafin*.

LOBOUNDED
1.0E+02

A file called *decklist* within the current working directory must contain a list of the test numbers the analyst wishes to be used in the optimization process. The Optimizer will not operate without this file in the current working directory. The following is an example of the file called *decklist* that is read by the Optimizer. The format of the file is limited to the following:

1. Each test number must exist on a line by itself.

g Indicates upper and lower bounds put onto the independent variables.
h Indicates only lower bounds put onto the independent variables.
i Indicates only upper bounds put onto the independent variables.
j Found in *rafin*.

363
364
365
366
367
373
375
378
379
380
398
399
400
401
402
403
404
405
412
413
414
434
441
446
448
451
456

A file called *inclvect* within the current working directory may contain the integer values to activate or deactivate the tunable parameters. The following is an example of the file called *inclvect* that is read by TRAC and read/written by the Optimizer. The format of the file is limited to the following three items:

1. The first five columns are reserved for the integer value (0 or 1) that activates or deactivates a desired tunable parameter.
2. Columns 6 through 10 are ignored.
3. Columns 11 through 18 are reserved for the tunable parameter label.

0	fgam2	corifht
0	frw	rbshrz
0	adqdz	htvssl
0	alpctr	htvssl
0	dqcmx	htvssl
0	dqcmim	htvssl
0	dqdzmx	htvssl
0	fac013	htvssl
0	fac014	htvssl
0	fac015	htvssl

0	fac016	htvssl
0	fac017	htvssl
0	fac018	htvssl
0	fac019	htvssl
0	facthl1	htvssl
0	facthl2	htvssl
0	facthv	htvssl
1	fdham	htvssl
0	fwebb	htvssl
0	rel1	htvssl
0	rel2	htvssl
0	rel3	htvssl
0	ret1	htvssl
0	ret2	htvssl
0	ret3	htvssl
0	xdqdz	htvssl

4. ADDITIONAL TOOLS RELATED TO OPTIMIZATION

There are some additional features that were added to the code as a result of the optimization effort. It was necessary to view where on the interfacial drag model selection logic map¹ that a particular coarse node existed. The logic map defines how the various flow-regime drag coefficients are calculated at an axial position relative to the CHF location and within specified void fraction boundaries. The various locations on the map were assigned labels, as shown in Fig. 5-1. This tool was implemented using time-averaged values of the post-CHF regime axial boundaries (i.e., z_{CHF} , $z_{\text{Transition-Boiling}}$, z_{Smooth} , $z_{\text{Rough/Wavy}}$, z_{Agitated} , and $z_{\text{Dispersed}}$). This could be extended to include time-dependent output at the specified information dump intervals.

5. IMPLEMENTATION PROBLEMS AND DEFICIENCIES

In the process of learning how to use the ADIFOR software, it was found that ADIFOR may not generate coding to calculate all of the desired partial derivatives within the selected TRAC subroutines. This was found to be caused by the way a particular TRAC routine originally was written. As a result, the user cannot assume that ADIFOR will generate an array to hold an independent variable's partial derivative vector. For example, if the user assumes that the Penalty function is dependent on variables $\vec{z} \equiv (x, y, z)$, which are stored in a named common block, ADIFOR may generate storage space only for the vectors $\frac{\partial x}{\partial \vec{z}}$ and $\frac{\partial y}{\partial \vec{z}}$ because there actually exists no dependence on z within the Penalty function. Because the vector initializations $\frac{\partial x}{\partial \vec{z}} = (1, 0, 0)$ and $\frac{\partial y}{\partial \vec{z}} = (0, 1, 0)$ are necessary to obtain nonzero partial

Axial Position	Highly Dispersed	Q	R	S
	Dispersed	N	O	P
	Agitated	K	L	M
	Rough/Wavy	H	I	J
	Smooth	E	F	G
	Transition Boiling	B	C	D
	Nucleate Boiling	A	A	A
		Low Void	Intermediate Void	High Void

Fig. 5-1. Logic identification key for the interfacial-drag-coefficient model selection.

derivatives from the ADIFOR routines, if code developers assume functional dependence on z , they would initialize $\frac{\partial z}{\partial \bar{z}} = (0,0,1)$ and receive compilation errors. Therefore, the code developer using ADIFOR-generated routines must verify that the Penalty function is in fact dependent on all of the chosen independent variables.

Another feature of ADIFOR-generated routines that was problematic is related to INCLUDE statements within the original source code to be differentiated. If the program to be differentiated references external files using the INCLUDE statement, ADIFOR will paste the contents of the INCLUDED file within the generated derivative code. Thus, if the derivative code is to be included within the latest release of the program, all of the pasted files' text must be replaced with the original INCLUDE statements to maintain a consistent format. This proves to be tedious within a code such as the modernized version of TRAC.

The additional tools discussed in Section 4 should be extended to include the heat-transfer model selection logic for both the interfacial heat transfer and the wall-to-fluid heat transfer.¹

6. ARCHIVE DESCRIPTIONS AND STORAGE LOCATIONS

The source code of the HP version of TRAC 5.4.25R10 and the L-BFGS-B optimization package is located on the CFS node named `/n9mod2/davidhp/`, and the archives are described as follows:

ad4_files.tar.gz	Files (including directory structure) required to generate the ADIFOR routines used to calculate the partial derivatives of the Penalty function using the command: Adifor2.0 AD_SCRIPT=name.adf , where name refers to a specific .adf file.
adifor2.tar.gz	ADIFOR 2.0 software package for the Sun operating system (executable and configuration files) used to generate desired FORTRAN 77 code—directory structure preserved.
hpux_lib.tar.gz	ADIFOR 2.0 function libraries compiled for the HP-UX operating system—directory structure preserved.
lbfgs2.3.tar.gz	Original source and documentation of the limited memory nonlinear optimization software. ⁶
optmzr.tar.gz	Optimization software package source that uses a modified version of L-BFGS-B, version 2.3, and calls a TRAC executable as an external process.
trac.tar.gz	HP version of TRAC 5.4.25R10 source code (.f and some .h files) and HP Makefile.
w_input.tar.gz	<ol style="list-style-type: none"> 1. Winfrith¹⁰ transient input decks (tracn.###, where ### corresponds to an experiment number) that are programmed for tend = 30-s runs. 2. Winfrith restart decks (tracn.###.rst) corresponding to the same experiments. 3. Winfrith quasi-steady-state experimental data files (data.###). 4. some examples of the files described in Section 3.6.

All of the above-mentioned tar files are compressed using the GNU version gzip and thus, the gz file name extension.

TABLE 1
INTERFACIAL HEAT-TRANSFER TUNABLE PARAMETER DESCRIPTIONS¹

Variable Name	Description
fgam1	multiplier on the heat-transfer coefficient (hA) calculated using the Unal-Chen model for vapor to interface heat transfer
fgam2	multiplier on the heat-transfer coefficient (hA) calculated using the Unal-Chen model for vapor to interface heat transfer
fgam3	exponent on the heat-transfer coefficient (hA) calculated using the Unal-Chen model for vapor to interface heat transfer
fgamw	multiplier on the heat-transfer coefficient (hA) calculated using the Unal-Chen model for vapor to interface heat transfer
qvapor	heat-transfer rate from vapor to the phase interface, $\dot{q}_{vap,calc}$
spacer	multiplier on the heat-transfer coefficient (hA) for grid spacer effects
tvn	vapor temperature, $T_{vap,calc}$

TABLE 2
INTERFACIAL DRAG TUNABLE PARAMETER DESCRIPTIONS¹

Variable Name	Description
cfzi	length-weighted drag coefficient, C_i
fac008	multiplier on the subcooled nucleate boiling regime drag coefficient
fac009	multiplier on the subcooled nucleate boiling regime void fraction
fac010	multiplier on the momentum-weighted drag coefficient in the annular mist flow regime
fac011	multiplier on the regime drag coefficient upper limit in the annular mist flow regime
fac012	multiplier on the regime drag coefficient lower limit in the annular mist flow regime
fcdrop	multiplier on the regime drag coefficient in the lightly-dispersed annular mist flow regime
fc1	multiplier on the subcooled nucleate boiling regime interfacial shear
fc2	multiplier on the subcooled nucleate boiling regime void fraction
fc3	multiplier on the subcooled nucleate boiling regime drag coefficient
fc4	multiplier on the roughness coefficient in the rough/wavy flow regime
fdis	multiplier on the highly dispersed flow regime drag coefficient
ffd	multiplier on the regime drag coefficient in the highly-dispersed annular mist flow regime
ffs	multiplier on the free-stream nucleate boiling for subcooled or superheated liquid drag coefficient
fmdis	multiplier on the moderately dispersed flow regime drag coefficient
frw	multiplier on the rough/wavy flow regime drag coefficient
fsb	multiplier on the subcooled boiling flow regime drag coefficient
fsm	multiplier on the smooth flow regime drag coefficient
xhvdis	exponent on smooth and rough/wavy flow regime drag coefficient multipliers for high void condition
xmdis	exponent on post-agitated and large drop flow regime drag coefficient multipliers for high void condition
xnbs	exponent on churn-turbulent transition flow regime drag coefficient

TABLE 3
WALL HEAT-TRANSFER TUNABLE PARAMETER DESCRIPTIONS¹

Variable Name	Description
adqdz	void fraction lower bound (α_{low}) for the CHF transition boiling heat flux parameter, B, correlation
alpctr	void fraction upper bound (α_{high}) for the CHF transition boiling heat flux parameter, B, correlation
dqcmx	maximum value of Cons. used in B
dqcmim	minimum value of Cons. used in B
dqdzmx	maximum value of the CHF transition boiling heat flux parameter, B
fac013	multiplier on the condensation heat-transfer coefficient
fac014	multiplier on the condensation heat-transfer coefficient
fac015	multiplier on the natural convection heat-transfer coefficient
fac016	multiplier on the laminar flow heat-transfer coefficient
fac017	multiplier on the turbulent flow heat-transfer coefficient
fac019	multiplier on the natural convection heat-transfer coefficient in turbulent flow
facthl1	exponent on the Denham liquid heat-transfer coefficient
facthl2	exponent on the Bromley liquid heat-transfer coefficient
facthv	exponent on the vapor heat-transfer coefficient weighting factor within the film boiling regime
fdham	multiplier on the Denham inverted annular liquid film boiling
fwebb	multiplier on the vapor heat-transfer coefficient with in the film boiling regime
qtot	total heat flux from the heated surface(s) to the fluid, $q''_{wall,calc}$
rel1	zero-order coefficient (C_0) for the quadratic model of Cons. where $Re_v \leq 2000$
rel2	first-order coefficient (C_1) for the quadratic model of Cons. where $Re_v \leq 2000$
rel3	second-order coefficient (C_2) for the quadratic model of Cons. where $Re_v \leq 2000$
ret1	zero-order coefficient (D_0) for the quadratic model of Cons. where $Re_v > 2000$
ret2	first-order coefficient (D_1) for the quadratic model of Cons. where $Re_v > 2000$
ret3	second-order coefficient (D_2) for the quadratic model of Cons. where $Re_v > 2000$
tw	wall temperature, $T_{wall,calc}$
xdqdz	exponent (ψ) on the void ratio within the B parameter correlation where $\alpha_{low} < \alpha < \alpha_{high}$

TABLE 4
FILES CONTAINING SUBROUTINES AND NAMED COMMON BLOCKS USED IN
THE OPTIMIZATION PROCESS

Subroutine/Common Name (File)	Parent Routine (File)	Description
g_chen (g_chen.f)	g_htvssln (g_htvssln.f)	converted from chen (chen.f) by ADIFOR 2.0
g_corifht (g_corifht.f)	htifn (htifn.f)	performs interfacial heat-transfer calculations; converted from corifht (corifht.f) by ADIFOR 2.0
g_cpvv1 (g_cpvv1.f)	g_htvssln (g_htvssln.f)	converted from cpvv1 (cpvv1.f) by ADIFOR 2.0
g_cpvv1d (g_cpvv1d.f)	g_htvssln (g_htvssln.f)	converted from cpvv1d (cpvv1d.f) by ADIFOR 2.0
g_cpvv1h (g_cpvv1h.f)	g_htvssln (g_htvssln.f)	converted from cpvv1h (cpvv1h.f) by ADIFOR 2.0
g_hlfmr (g_hlfmr.f)	g_htvssln (g_htvssln.f)	converted from hlfmr (hlfmr.f) by ADIFOR 2.0
g_htvssln (g_htvssln.f)	htc (htc.f)	performs wall-to-fluid heat-transfer calculations; converted from rbshrz (rbshrz.f) by ADIFOR 2.0
g_hvnb (g_hvnb.f)	g_htvssln (g_htvssln.f)	converted from hvnb (hvnb.f) by ADIFOR 2.0
g_hvwebb (g_hvwebb.f)	g_htvssln (g_htvssln.f)	converted from hvwebb (hvwebb.f) by ADIFOR 2.0
g_rbshrz (g_rbshrz.f)	cif3n (cif3n.f)	performs interfacial drag calculations; converted from htvssln (htvssln.f) by ADIFOR 2.0
g_satprd (g_satprd.f)	g_htvssln (g_htvssln.f)	converted from satprd (satprd.f) by ADIFOR 2.0
g_satprh (g_satprh.f)	g_htvssln (g_htvssln.f)	converted from satprh (satprh.f) by ADIFOR 2.0
g_satprs (g_satprs.f)	g_htvssln (g_htvssln.f)	converted from satprs (satprs.f) by ADIFOR 2.0
g_tbzfcn (g_tbzfcn.f)	g_htvssln (g_htvssln.f)	converted from tbzfcn (tbzfcn.f) by ADIFOR 2.0
g_thcv (g_thcv.f)	g_htvssln (g_htvssln.f)	converted from thcv (thcv.f) by ADIFOR 2.0
g_viscv (g_viscv.f)	g_htvssln (g_htvssln.f)	converted from viscv (viscv.f) by ADIFOR 2.0
g_viscvd (g_viscvd.f)	g_htvssln (g_htvssln.f)	converted from viscvd (viscvd.f) by ADIFOR 2.0
g_viscvh (g_viscvh.f)	g_htvssln (g_htvssln.f)	converted from viscvh (viscvh.f) by ADIFOR 2.0

TABLE 5
FILES CONTAINING SUBROUTINES AND NAMED COMMON BLOCKS USED IN
OPTIMIZATION PROCESS

Subroutine/Common Name (File)	Parent Routine (File)	Description
wtmavg (wtmavg.f)	pstepq (pstepq.f)	time-averaging routine which also calculates Penalty Function and its gradient
ad4_q (ad4_q.h)	g_htvssln, htvssln and wtmavg	qtot is made global for ADIFOR's needs
ad4flag (ad4flag.h)	blkdat, cif3n, htc, htfn, main000, steady, timstp, trans, wclosure and wtmavg	global variable indicating activation of ADIFOR routines
g_ad4_q (g_ad4_q.h)	g_htvssln, wtmavg and zdrive_d	partial derivatives of qtot
ifhtglbl (ifhtglbl.h)	corifht, g_corifht and wtmavg	qvapor is made global for ADIFOR's needs
initgrad1 (initgrad1.h)	g_corifht	initializes partial derivatives
initgrad2 (initgrad2.h)	g_rbshrz	initializes partial derivatives
initgrad3 (initgrad3.h)	g_htvssln	initializes partial derivatives
iran (iran.h)		iran flag common
name1 (name1.h)	g_corifht, optdrv and wtmavg	contains closure parameter labels; used in TRAC and Optimizer
name2 (name2.h)	g_rbshrz, optdrv and wtmavg	contains closure parameter labels; used in TRAC and Optimizer
name3 (name3.h)	g_htvssln, optdrv and wtmavg	contains closure parameter labels; used in TRAC and Optimizer
optdata1 (optdata1.h)	g_corifht and wtmavg	temporary and time-averaged variable storage
optdata2 (optdata2.h)	g_rbshrz and wtmavg	temporary and time-averaged variable storage
optdata3 (optdata3.h)	g_htvssln and wtmavg	temporary and time-averaged variable storage
optmiz (optmiz.h)	optdrv and wtmavg	contains optimization parameter statements; used in TRAC and Optimizer
pimentel (pimentel.h)	input and wtmavg	contains information from DATA file
pimentel3 (pimentel3.h)	input and wtmavg	contains information from DATA file
zbdntmp (zbdntmp.h)	cif3n and wtmavg	contains instantaneous drag regime boundary information
g_out1 (g_out1.h)	g_corifht	coding which INCLUDED at end of g_corifht routine

TABLE 5 (cont)
FILES CONTAINING SUBROUTINES AND NAMED COMMON BLOCKS USED IN
OPTIMIZATION PROCESS

Subroutine/Common Name (File)	Parent Routine (File)	Description
g_out2 (g_out2.h)	g_rbshrz	coding which INCLUDED at end of g_rbshrz routine
g_out3 (g_out3.h)	g_htvssln	coding which INCLUDED at end of g_htvssln routine
calcpoly (calcpoly.f)	wtmavg	calculates a polynomial value and derivative given coefficients, order, and x value
polyfit (polyfit.f)	wtmavg	fits a polynomial of order n to given data
normal (normal.f)	polyfit	used by polyfit algorithm
gauss (gauss.f)	polyfit	used by polyfit algorithm
ordr (ordr.f)	polyfit	used by polyfit algorithm
elim (elim.f)	polyfit	used by polyfit algorithm
subst (subst.f)	polyfit	used by polyfit algorithm
pivot (pivot.f)	polyfit	used by polyfit algorithm

7. REFERENCES

1. R. A. Nelson and D. A. Pimentel. "Report on a Phenomenological Model of the Thermal Hydraulics of Hot Rod Bundles during Quenching Produced by Either Bottom or Top Flow," Technology & Safety Assessment Division, Nuclear Systems Design & Analysis Group, Los Alamos National Laboratory, report in progress, 1997.
2. L. Fox. *Optimization Methods for Engineering Design*, Addison-Wesley Publishing Company, Massachusetts, 1971.
3. R. A. Nelson and C. Unal. "A Phenomenological Model of the Thermal Hydraulics of Convective Boiling during the Quenching of Hot Rod Bundles Part I: Thermal Hydraulic Model," *Nuclear Engineering and Design* 136, pp. 277-298, 1992.
4. J. Spore, S. J. Jolly-Woodruff, T. K. Knight, J. Lin, R. A. Nelson, K. O. Pasamehmetoglu, R. G. Steinke, and C. Unal. "TRAC-PF1/MOD2: Code Manual," US Nuclear Regulatory Commission report NUREG/CR-5673, 1993.
5. R. H. Byrd, J. Nocedal, and R. B. Schnabel. "A Limited Memory Algorithm for Bound Constrained Optimization," *SIAM Journal on Scientific Computing* 16:5, pp. 1190-1208, May 1994.

6. R. H. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. "L-BFGS-B: FORTRAN Subroutines for Large Scale Bound Constrained Optimization," Northwestern University technical report NAM-11, March 1997.
7. C. Bischof, A. Carle, P. Khademi, A. Mauer, and P. Hovland. "ADIFOR 2.0 User's Guide (Revision C)," Argonne National Laboratory report ANL/MCS-TM-192, August 1995.
8. C. Bischof, A. Carle, Corliss, A. Griewank, and P. Hovland. *ADIFOR—Generating Derivative Codes from FORTRAN Programs. Scientific Programming* 1(1), pp. 1-29, 1992.
9. C. Bischof, A. Carle, P. Khademi, and A. Mauer. *The ADIFOR 2.0 System for the Automatic Differentiation of FORTRAN 77 Programs*. Center for Research on Parallel Computation, Rice University, Houston, Texas, report CRPC-TR94491, revised May 1995.
10. R. A. Savage, D. Swinnerton, and K. G. Pearson. "Heat Transfer and Voidage Data in Steady-State Post-Dryout at Low Quality and Medium Pressure: Data Report," Safety and Engineering Science Division, Winfrith Technology Centre report AEEW-R2486, June 1989.

M98004317

Report Number (14) LA-UR--97-4805

Publ. Date (11) 199710

Sponsor Code (18) NRC , XF

UC Category (19) UC-000 , DOE/ER

DOE