NUREG/CR–6371
SEA96–3104–A:4

BLOCKAGE 2.5 Reference Manual

Manuscript Completed: November 1996
Date Published: December 1996

Prepared by
C. J. Shaffer, W. Bernahl*, J. Brideau, D. V. Rao

Science and Engineering Associates, Inc.
P. O. Box 3722
Albuquerque, NM 87190

*Software Edge, Inc.
1485 Chippewa Pathway
Riverwoods, IL 60015

M. L. Marshall, Jr., NRC Project Manager

Prepared for
Division of Engineering Technology
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555–0001
NRC Job Code W6325

DISCLAIMER

Abstract

The BLOCKAGE 2.5 code was developed by the United States Nuclear Regulatory Commission (NRC) as a tool to evaluate licensee compliance regarding the design of suction strainers for emergency core cooling system (ECCS) pumps in boiling water reactors (BWR) as required by NRC Bulletin 96-03, "Potential Plugging of Emergency Core Cooling Suction Strainers by Debris in Boiling Water Reactors." Science and Engineering Associates, Inc. (SEA) and Software Edge, Inc. (SE) developed this PC-based code. The instructions to effectively use this code to evaluate the potential of debris to sufficiently block a pump suction strainer such that a pump could lose NPSH margin was documented in a User's Manual [NRC, NUREG/CR-6370].

The Reference Manual contains additional information that supports the use of BLOCKAGE 2.5. It contains descriptions of the analytical models contained in the code, programmer guides illustrating the structure of the code, and summaries of coding verification and model validation exercises that were performed to ensure that the analytical models were correctly coded and applicable to the evaluation of BWR pump suction strainers.

The BLOCKAGE code was developed by SEA and programmed in FORTRAN as a code that can be executed from the DOS level on a PC. A graphical users interface (GUI) was then developed by SEA to make BLOCKAGE easier to use and to provide graphical output capability. The GUI was programmed in the C language. The user has the option of executing BLOCKAGE 2.5 with the GUI or from the DOS level and the Users Manual provides instruction for both methods of execution.

Table of Contents

List of Figures and Tables

Page

Table

Acknowledgements

Several individuals made significant contributions towards development of BLOCKAGE 2.5. In particular, this study considerably benefited from technical and management support of Mr. Michael Marshall (RES), who was the NRC Task Manager for this effort. He provided critical technical direction, actively participated in the model development and implementation, and performed in-depth testing of BLOCKAGE 2.5. As always, Mr. Aleck Serkiz (RES) provided significant technical insights and review of the models and their results.

At SEA, Ms. Gwen Vergera led the effort of report peer review and its production. Ms. Rose Flores was responsible for the typing and layout of the final document.

1.0 Introduction

In 1993, the NRC initiated an evaluation of the potential of loss-of-coolant-accident (LOCA) generated debris to block BWR suction strainers and prevent the ECCS from performing its long-term cooling function [NRC, NUREG/CR-6224]. The NRC had previously investigated the problem of sump screen blockage for pressurized water reactors (PWR), and sponsored the development of the computer codes PRA and TABLE to aid in that analysis [NRC, NUREG/CR-3394]. The determination of the probability of unacceptable sump blockage was based on the probability of occurrence of an initiating event and the probability of sump blockage occurring as a result of that event.

The BLOCKAGE code was developed to predict whether or not accumulation of debris on the suction strainers following a LOCA would lead to loss of ECCS pump net positive suction head (NPSH) in a BWR. BLOCKAGE 1.0 was developed and validated by reproducing the NUREG/CR-3394 results [Brideau, 1993]. BLOCKAGE 2.0 was developed to accommodate BWR features. Subsequent enhancements to BLOCKAGE resulted in BLOCKAGE 2.3, which included a transient suppression pool model to credit sedimentation used in the NUREG/CR-6224 analysis. Several additional models were included in BLOCKAGE 2.5 to increase capabilities of the code; multiple debris types could be tracked simultaneously for multiple pumps and multiple common headers; the suppression pool temperature could vary with time and the NPSH margin for each pump could be a function of the suppression pool temperature; and to provide the users with additional optional head loss correlations. A graphical user interface (GUI), originated with BLOCKAGE 2.4 and enhanced for BLOCKAGE 2.5, was created to make the code more "user friendly" and allow the user to graphically view the code results.

BLOCKAGE 2.5 allows the user to simulate debris generation and subsequent transport of multiple types of debris including fibers, particles, and metals, using either a three-zone destruction model or a user-specified quantity of debris for transport. The debris transport from the drywell to the wetwell can be location-dependent and time-dependent. The transport during the blowdown period due to depressurization flows is treated separately from the transport during the washdown period which is due to ECCS recirculation, containment spray, and steam condensate flows. Two sizes of pipe break scenarios are considered: large and medium LOCAs. The debris transport within the suppression pool including the deposition of debris on the strainers and the debris concentration within the pool is calculated separately for each discrete debris size and each debris type. The suppression pool is treated as a single volume of water, i.e., debris concentration does not vary with location within the pool. The user supplies several model parameters which are time-dependent: the calculational time step, the pump flow rates, the drywell debris transport rates, the suppression pool temperature, and the suppression pool resuspension and settling rates.

Several independent ECCS pumping systems can be modeled simultaneously, with each system consisting of multiple pumps on a common header attached to a single equivalent strainer. Each pump is considered to fail by loss of NPSH margin when the strainer head loss exceeds its temperature-dependent NPSH margin. A debris bed filtration model determines the quantity of debris that is entrained in the pump flow that is deposited on the strainer and the quantity of debris that passes through the strainer and that may be retained within the primary system.

Four user-optional head loss correlations including the NUREG/CR-6224 correlation, a BWR Owner's Group (BWROG) 1994 correlation, and two generic correlations are available to predict the strainer head losses. The failure of the ECCS to provide long-term cooling to the reactor core is flagged whenever the total ECCS flow capability drops below a user specified minimum flow rate. When selected by the user, BLOCKAGE 2.5 can also write several probabilities reports that provide information regarding the plant-wide strainer blockage probabilities correlated by pipe diameter, piping system, and break location.

BLOCKAGE 2.5 consists of two programs: 1) the BLOCKAGE GUI and the BLOCKAGE calculational engine. Figure 1-1 illustrates the two programs and the files, input screens, and results windows associated with each program.

Instructions on the effective use of BLOCKAGE 2.5 in evaluating the impact of strainer blockage on BWR ECCS recirculation reliability were provided in NUREG/CR-6370, "BLOCKAGE 2.5 User's Manual." This manual contains additional supporting information, including detailed descriptions of the analytical models contained in the code, programmer guides describing the actual coding structure, and

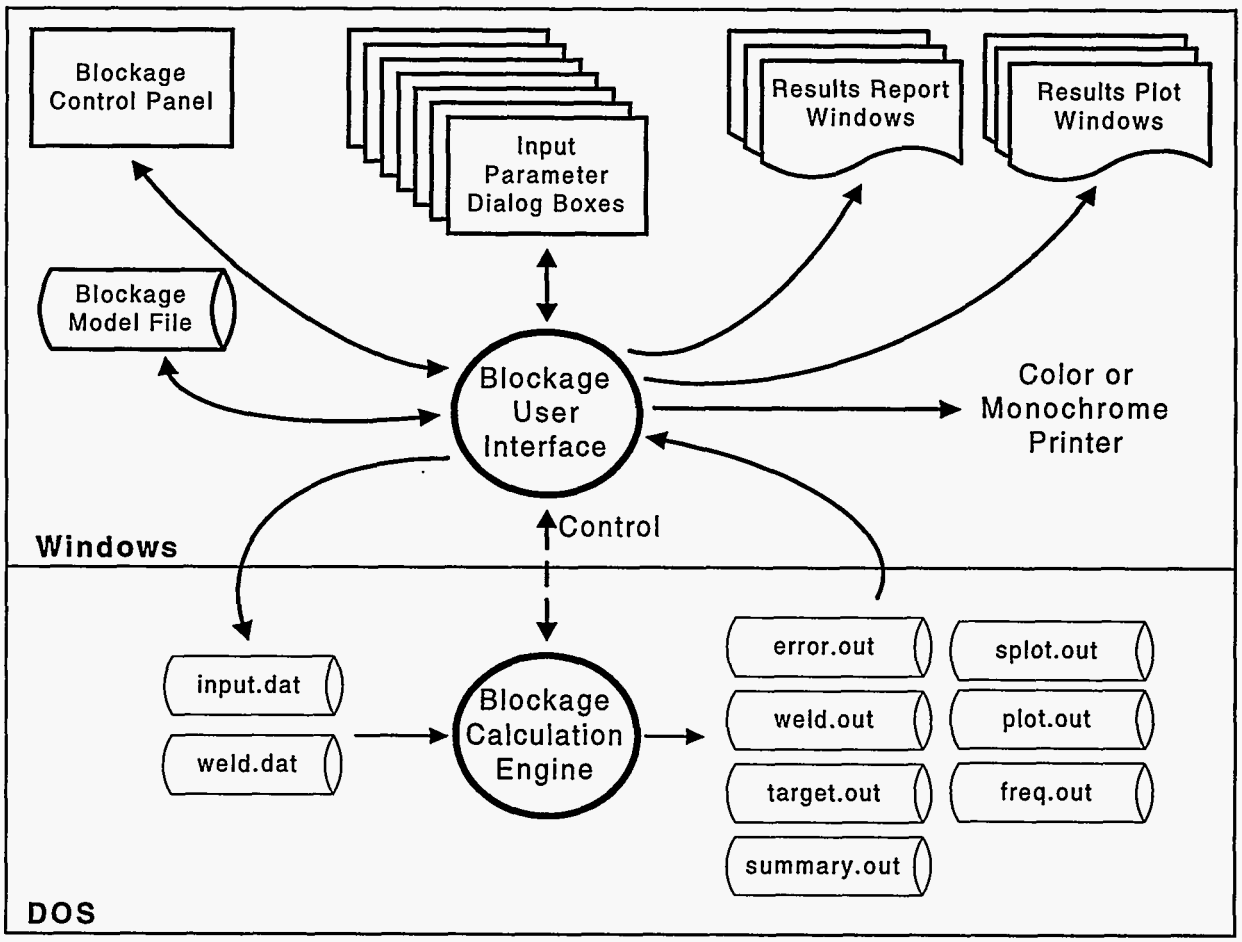


Figure 1-1: BLOCKAGE UI in Context

summaries of the coding verification and model validation exercises that were performed to ensure the correctness of the coded models. The report is organized into the following sections:

2.0 BLOCKAGE Code Model Description
3.0 BLOCKAGE Code Programmer's Guide
4.0 BLOCKAGE Code Verification and Validation
5.0 GUI Programmer's Guide
6.0 GUI Verification and Validation of Coding
7.0 References.

The appendices contain further technical and background information as follows:

A GUI Functional Specifications
B BLOCKAGE Input File Format
C Time-Base Results Variables
D Proposed Menu
E Proposed Dialogs/Windows
F Plot Layout

2.0 BLOCKAGE Code Model Descriptions

Descriptions of the models coded into BLOCKAGE 2.5 are described in this section. The methodology used in the development of these models is found NUREG/CR-6224.

2.1 Debris Attributes

The generation, transport and deposition of insulation debris depends upon a variety of attributes or properties which the user must supply to BLOCKAGE as input. These attributes includes physical properties, behavior classifications, target destruction and debris transport characteristics. These attributes are described here prior to the discussion of the individual models.

Debris Classification The prediction of strainer head loss strongly depends upon the general type of material from which the debris was formed. BLOCKAGE 2.5 was developed to estimate the strainer head loss for a cake of fibrous debris is formed that is capable of trapping particulate debris within the voids between the fibers of the cake. Metallic debris formed by destroying reflective metallic insulation (RMI) may also contribute to the strainer head loss.

The debris classification functions as an interface between the debris generation, debris transport, and head loss models. The debris generation and transport models track each specific type of debris (i.e., fiberglass, paint chips, RMI, calcium silicate, etc.) independently throughout the calculation, including the deposition of debris on the strainers. There can be multiple types of fibrous debris or multiple types of particulate debris; however, the head loss correlations available in BLOCKAGE 2.5 only allow one type of fibrous debris, one type of particulate debris, and one type of metallic debris. Therefore, all the types of fibrous debris must be combined into a single mixture of fibrous debris with the appropriate effective densities and specific surface areas to describe the mixture. Similar properties must be determined for the particulate and metallic debris mixtures. These debris classifications are used by BLOCKAGE to combine the debris types into mixtures used in calculating the strainer head losses.

The user must specify a classification for each specific type of debris that is introduced into a particular calculation. BLOCKAGE 2.5 offers the user the following four debris classifications:

- Fibrous debris (F)
- Metallic debris (M)
- Particulate debris (P)
- Ignore (I).

The Ignore classification does not represent an actual type of debris. It was included in BLOCKAGE to allow the user to track a particular debris type, but not consider its effect on strainer head loss. BLOCKAGE 2.5 will predict the quantity of this debris type deposited on each strainer, but then ignores this debris in the strainer head loss calculation. This option was needed in order to provide complete continuity between NUREG/CR-6224 strainer blockage study performed with Version 2.3 and the current Version 2.5.

Origin of Debris The origin of each type of debris introduced into a calculation must be specified by the user. The origin is used by BLOCKAGE 2.5 to determine where and how the debris is introduced into the transport calculation and to request the appropriate debris destruction and transport fractions from the user. All debris tracked by BLOCKAGE 2.5 must originate from one of three possible origins:

- Target destruction in the drywell (TG)
- Non-target drywell debris (DW)
- Wetwell debris (WW).

Debris generated by destroying insulation targets in the drywell requires the destruction fractions used by the debris generation model (unless the user specified volume option was selected by the user), and requires location-dependent transport fractions. Fiberglass or RMI piping insulation blankets are examples of target generated debris.

Debris originating in the drywell, but not introduced into the calculation as a destroyed target, does not require the specification of destruction fractions, and only requires one non-location dependent set of transport fractions. An good example of non-target drywell debris is paint chips which may originate from various locations throughout the drywell.

Debris existing in the wetwell, such as suppression pool sludge, is specified as originating in the wetwell where it is initially deposited onto the wetwell floor. This debris is resuspended into the pool by the violent primary system depressurization flows associated with a pipe break.

Note that if a particular type of debris, such as fiberglass fibers, were to originate in both the drywell (from the destruction of a target) and the wetwell (as fiber that has accumulated over a period of time), the code input requires two separate debris types as input. However, since there is no reason that the size distributions from these two sources should be the same, it makes sense that the two sources of fiberglass fibers be treated separately.

Material Density The material density is the density of an elemental component of the debris. For fibrous debris, the material density is the density of an individual fiber. For particulate debris, it is the density of an individual particle and for metallic debris, it is the density of the metal. The material densities are used in calculating the debris bed porosities on the strainers.

Fabricated Density The fabricated density is the density of the debris as it is being transported throughout the calculation. The original blockage transport models focused on the transport of fibrous insulation debris which was assumed to have the same density as it had when it was an insulation blanket on a pipe; thus, the term as-fabricated or as-manufactured. The nominal volume of debris formed from target destruction is calculated using the fabricated density, the target dimensions, and the destruction fractions. The fabricated density for metallic insulation debris should be the mass of the metallic foils in the RMI cassette divided by the volume of the metallic insulation assembly. For non-target particulate debris such as paint chips or the wetwell sludge, the fabricated density should be the material density. The debris volumes presented in the output can be multiplied by the fabricated densities to obtain debris masses. The two important considerations to remember when specifying the fabricated densities for each debris type are: 1) the fabricated density is used by the debris generation model to calculate the debris volumes, and 2) the volumes presented in the output assume that the debris is at the fabricated density.

Rubble Density The rubble density of the fibrous debris is used to determine the thickness of the fibrous debris cake on the strainer. Likewise, the metallic rubble density is used to determine the thickness of the layer of metallic debris on the strainer. The rubble density of the particulate debris is used to determine the compaction limit of the fibrous debris cake. The rubble density in some cases may be the same as the fabricated density, as has been generally used for fibrous debris, but it does not need to be the same. The rubble density of metallic debris will most likely be different than its fabricated density. The rubble density of particulate debris is usually taken to be the density of packed wetwell sludge.

Specific Surface Area The specific surface area is the total surface area of the debris divided by its volume. The specific surface area is used by the NUREG/CR-6224 head loss correlation to estimate the head loss across the strainer insulation debris. The specific surface areas, S_v, can be estimated using the following equations.

Fibers:

$$S_v = \frac{4}{d_f} \qquad (2\text{-}1)$$

where:

d_f = the diameter of an individual insulation fiber (equation assumes $L >> d_f$);

Particles:

$$S_V = 6\sum_{i=1}^{N} \frac{f_i}{d_i} \qquad (2\text{-}2)$$

where:

d_i = the diameter of individual particles in the settling velocity group i (assumed spherical)
f_i = the volume fraction of settling velocity group i
N = the number of velocity settling groups;

Metals:

$$S_v = \frac{2}{t_{sheet}} \qquad (2\text{-}3)$$

where:

t_{sheet} = the thickness of an individual metal sheet (assumes sheet area >> sheet thickness).

Terminal Settling Velocity Distributions The terminal velocity at which debris settles in a still pool of water is a function of the size of the individual debris particles and the type of debris. The BLOCKAGE models require that the terminal settling

velocity function be subdivided into discrete intervals or groups for each type of debris. The size distribution of the debris is then entered as the fraction of the debris that will settle with a characteristic velocity for that particular velocity group.

Strainer Filtration Efficiencies Debris entrained in the water being pumped through an ECCS recirculation strainer will either pass through the strainer with the water flow or be deposited onto the strainer forming a debris cake. The fraction that is deposited onto the strainer is called the strainer filtration efficiency. The filtration efficiencies are a function of the type and size of debris and the thickness of debris already deposited onto the strainer.

Primary System Retention Efficiency The debris passing through the strainer will be carried by the ECCS flow to the reactor vessel and associated piping where some debris will likely be trapped and other debris may be returned to the suppression pool. The retention efficiency determines how much of the debris passing through the strainers is not returned to the suppression pool. One minus this efficiency is immediately returned to the pool. The retention efficiency is a function of the type and size of debris.

Destruction Fractions The destruction fractions specify how much of a particular target is destroyed into a transportable form by the pipe break flows. Three destruction fractions corresponding to each of the three destruction regions or zones of the target destruction model are provided as input. These destruction fractions are debris type dependent in BLOCKAGE 2.5.

Transport Fractions The transport of debris from the drywell to the wetwell was modeled in two discrete time intervals referred to as the blowdown period and the washdown period. Debris transport during the blowdown period would be driven by the primary depressurization flows and during the washdown period by the recirculation flows from the break, by the containment spray flows if active, and by the drainage of steam condensate from the structures. These transport fractions are debris type dependent in BLOCKAGE 2.5. The transport fraction for the blowdown period, T_{blow}, is defined as the total fraction of a particular type of debris that will be transported to the wetwell during the blowdown period. The transport fraction for the washdown period, T_{wash}, is defined as the fraction of the debris remaining at the end of the blowdown period (i.e., one minus the blowdown transport fraction) that is transported to the wetwell during the washdown period.

Debris Volumes The volume of each type of debris (based on the fabricated density) generated must be specified in accordance with the origin of the debris. If the debris originates from a drywell target (TG), then its volume is computed from target data. When the debris originates from a non-target source in the drywell (DW) or in the wetwell (WW), then its volume is specified as a debris attribute.

2.2 Drywell Debris Generation Model

The BLOCKAGE code uses either a destruction zone model to determine the quantities of debris that is generated in the drywell from individual targets or allows the user to directly specify quantities of debris.

2.2.1 Destruction Zone Model

The extent of damage sustained by a target subjected to the forces of the primary system depressurization flows generally depends upon the orientation and distance between the pipe break and the insulation target. The target destruction model in BLOCKAGE approximates the destruction along lengths of pipe insulation located in the vicinity of the break by subdividing this vicinity into three discrete regions or zones of influence (methodology for this model is described in Section 3.4 of NUREG/CR-6224). A user supplied destruction fraction is applied to each of these three zones to determine the quantify of transportable insulation debris generated. The volume of debris generated (based on the fabricated insulation density) is given by the following equation.

$$V_{di} = \left[\sum_{j=1}^{N_i} A_{cj} \left(\sum_{k=1}^{3} F_{djk} \, \Delta L_{jk} \right) \right] \qquad (2\text{-}4)$$

where:

V_{di} = the volume of type i debris generated for a particular weld break

A_{cj} = the cross-sectional area of target j

N_i = the number of type i debris targets associated with the weld break
F_{djk} = the destruction fraction for target j in destruction zone k
ΔL_{jk} = the length of target j located within destruction zone k.

The boundaries of the zones are defined as multiples of the outer diameter of the broken pipe. These multiples are referred to as L/D ratios where L is the zone boundary dimension and D is the pipe diameter. The size of the destruction zones, therefore, increase with the size of the broken pipe. The zone L/D ratios used in the analyses supporting the NUREG/CR-6224 study were 3, 5, and 7.

With this model, the user must specify the dimensions of each target located within each of the three influence zones. The targets are assumed to be cylindrical in shape. These dimensions include the inner diameter of the target, the thickness of the target insulation, and the length of the target within each zone. In order to determine these lengths, the user must first specify the L/D ratios for the overall input model and a particular shape for the zones, e.g., a spherical shaped zone or 90° back-to-back cones. The BLOCKAGE code uses the target lengths to calculate debris volumes and is, therefore, impervious to the L/D ratios and the zone shapes, i.e., the L/D ratios and zone shapes become embedded in the target lengths.

2.2.2 User Defined Volumes

As an alternative method, BLOCKAGE 2.5 allows the user to simply specify the volume of debris that is generated from destroying a particular target. Using this method, discrete volumes for each particular weld break must be specified for each debris type, for each weld break location, and for each weld break pipe diameter to accommodate the differing debris attributes. The transport fractions are a function of the weld break locations. The time-dependent transport rates and the suppression pool dynamics are a function of the weld break size which is determined by the pipe diameter, i.e., a LLOCA or a MLOCA.

2.3 Drywell Debris Transport Model

The quantities of each particular type of drywell-generated debris that are transported to the wetwell suppression pool are determined from the user specified transport fractions and then transported on a time-dependent basis. The transport fractions for debris generated from the destruction of insulation targets are location dependent, whereas the transport fractions for debris generated from other non-target sources are not location dependent.

The time-dependent transport of debris from the drywell to the wetwell was modeled during two discrete time intervals referred to as the blowdown period and the washdown period. Debris transport during the blowdown period would be driven by the primary system depressurization flows, and during the washdown period by the recirculation flows from the break, by the containment spray flows if active, and by the drainage of steam condensate from the structures. A separate transport fraction is required for each of these two periods. The transport fraction for the blowdown period, T_b, is defined as the total fraction of a particular type of debris that will be transported to the wetwell during the blowdown period. The transport fraction for the washdown period, T_w, is defined as the fraction of the debris remaining at the end of the blowdown period (i.e., one minus the blowdown transport fraction) that is transported to the wetwell during the washdown period. The total debris transported to the wetwell, T_{tot}, is then:

$$T_{tot} = T_b + T_w\ (1 - T_b) \qquad (2\text{-}5)$$

The user must specify the time-dependent rates (fraction of the debris transported per second) for debris transport from the drywell to the wetwell for both the LLOCA and the MLOCA scenarios in a tabular format that is interpolated by a user optional step, linear, or exponential interpolation scheme. The user also specifies the end times for the blowdown and the washdown periods. The debris transport rate to the wetwell is provided by the following equation.

$$\frac{d}{dt} V_{ti}(t) = V_{di}\ G(t,m)\ T_i(t) \qquad (2\text{-}6)$$

where:

$$T_i(t) = \begin{bmatrix} T_{bi} & if & t \le t_b \\ T_{wi}(1 - T_{bi}) & if & t_b < t \le t_w \\ 0 & if & t > t_w \end{bmatrix} \quad (2\text{-}7)$$

and where:

$V_{ti}(t)$	=	the rate of volume transported for type i debris for a particular weld break
V_{di}	=	the volume of type i debris generated for a particular weld break
G(t,m)	=	the time-dependent transport rate for LOCA scenario m
$T_i(t)$	=	the time-interval-dependent transport fraction for type i debris
T_{bi}	=	the blowdown transport fraction for type i debris
T_{wi}	=	the washdown transport fraction for type i debris
t_b	=	the blowdown period end time
t_w	=	the washdown period end time
t	=	the calculational time

The time-dependent transport rates must be specified such that the total debris transported to the wetwell equals that required by the user specified transport fractions. This is illustrated by integrating the above time-dependent volume transport rate from zero to infinity as shown in Equation 2-8.

$$\int_0^\infty \frac{d}{dt} V_{ti}(t)\, dt = \quad (2\text{-}8)$$

$$V_{di}\left[T_{bi}\left(\int_0^{t_b} G(t,m)dt\right) + T_{wi}(1 - T_{bi})\left(\int_{t_b}^{t_w} G(t,m)dt\right)\right]$$

The integrated volume transport can only equal T_{tot} if each of the integrations of G(t,m) equals one for both LOCA scenarios. Thus, in order to get the correct total debris transport to the suppression pool, the time-dependent transport rates must integrate to one over the blowdown period and again over the washdown period. Furthermore, the rates must integrate to one using the exact time steps selected by the user.

To ensure that these integrals equal one, the BLOCKAGE code integrates the rates during the input processing using the input time steps and stops the calculation with an error message if the integrals do not equal one within a user specified input error of ± **cklim** called the check limit. The check limit is a number ranging from zero to one. If the user specifies a large check limit then the calculation of the debris transport could become inaccurate but if the number were too small, it may become difficult to satisfy the integration. Further user guidance in creating time-dependent transport rates that integrate to one is found in Section 4.3 of the User's Manual [NRC, NUREG/CR-6370].

2.4 Wetwell Debris Transport Model

The wetwell debris transport model accepts debris transported from the drywell by debris type, subdivides the debris of each type into settling velocity groups, and independently determines the transport of each velocity group of each debris type within the suppression pool. The debris entering the pool immediately becomes suspended in the pool where it can be deposited onto a strainer, pass through a strainer and be trapped within the primary system, or settle to the floor of the wetwell. Debris existing in the wetwell at the time of the pipe break is assumed to initially reside on the wetwell floor where it is subject to resuspension by the turbulent primary system depressurization flows.

The rate of debris transport from the drywell for each debris type and velocity settling group is:

$$\frac{d}{dt} V_{tij}(t) = V_{di}\, g_{ij}\, G(t,m)\, T_i(t) \quad (2\text{-}9)$$

where:

$V_{tij}(t)$	=	the volume of debris type i and velocity group j transported from the drywell
V_{di}	=	the volume of type i debris generated for a particular weld break
g_{ij}	=	the fraction of type i debris in settling velocity group j
G(t,m)	=	the time-dependent transport rate for LOCA scenario m
$T_i(t)$	=	the time-interval-dependent transport fraction defined above.

The rate of debris accumulation on the wetwell floor is the difference between the debris settling rate and the debris resuspension rate and is:

$$\frac{d}{dt}V_{fij}(t) = K_t(t,m)A_p U_{tij}\frac{V_{pij}(t)}{V_w} - K_r(t,m)V_{fij}(t) \quad (2\text{-}10)$$

where:

$V_{fij}(t)$ = the volume of debris type i and velocity group j on the wetwell floor
$V_{pij}(t)$ = the time-dependent volume of debris type i and velocity group j suspended in the pool
V_w = the volume of the suppression pool
$K_r(t,m)$ = the time-dependent resuspension factor for LOCA scenario m
$K_t(t,m)$ = the time-dependent settling turbulence factor for LOCA scenario m
A_p = the cross-sectional pool settling area
U_{tij} = the terminal settling velocity for debris type i and velocity group j.

The rate of debris deposition on the combined strainers is:

$$\frac{d}{dt}V_{sij}(t) = \left[\sum_{k=1}^{N_{str}} \varepsilon_{ij}(x_k)\left[\sum_{n=1}^{N_{pk}} \delta_{nk}M(t,m)Q_{nk}\right]\right]\frac{V_{pij}(t)}{V_w} \quad (2\text{-}11)$$

where:

$Vsij(t)$ = the volume of debris type i and velocity group j deposited on the strainers
$\epsilon_{ij}(x_k)$ = the strainer filtration efficiency for debris type i and velocity group j as a function of the fiber cake thickness on strainer k
δ_{nk} = indicates whether or not pump n on common header k is active (1 for active and 0 for not active)
$M(t,m)$ = the time-dependent flow multiplier for LOCA scenario m
Q_{nk} = the full volumetric flow capacity of pump n on common header k
N_{pk} = the number of ECCS pumps attached to common header k
N_{str} = the number of common headers.

The rate of debris retention by the primary system is:

$$\frac{d}{dt}V_{rij}(t) = r_{ij}\left[\sum_{k=1}^{N_{str}} (1-\varepsilon_{ij}(x_k))\left[\sum_{n=1}^{N_{pk}} \delta_{nk}\,M(t,m)Q_{nk}\right]\right]\frac{V_{pij}(t)}{V_w} \quad (2\text{-}12)$$

where:

$Vrij$ = the volume of debris type i and velocity group j retained by the primary system
$r_{ij}(x)$ = the primary system retention efficiency for debris type i and velocity group j

The volume of debris suspended in the suppression pool is then increased by debris transport from the drywell and decreased by debris settling onto the wetwell floor, debris deposited onto the strainers, and debris retained within the primary system, as given by:

$$\frac{d}{dt}V_{pij}(t) = \frac{d}{dt}V_{tij}(t) - \frac{d}{dt}V_{fij}(t) - \frac{d}{dt}V_{sij}(t) - \frac{d}{dt}V_{rij}(t) \quad (2\text{-}13)$$

These wetwell debris transport equations are actually mass transfer equations, even though they are implemented into the BLOCKAGE code in terms of the debris volumes based on the fabricated densities, i.e., these equations could be multiplied by the appropriate fabricated densities to obtain mass equations. Each equation in the set is solved for each velocity group of each debris type. The time-dependent parameters, K_r, K_t, G, and M are input by the user in a tabular form for both the LLOCA and the MLOCA scenarios.

2.5 ECCS Pump and Header Models

The ECCS pump and header models in BLOCKAGE 2.5 allow the user to tailor the strainer blockage calculation to the plant specific ECCS pumping systems. As many as eight independent pumping systems can be modeled in a single calculation. A single pumping system consists of multiple pumps (or a single pump) attached to a common header that draws water from a single strainer. The strainer areas of the multiple strainers attached to a common header are combined. The flow through the equivalent strainer is the combined flow of the operating pumps attached to the common header. As many as four pumps can be attached to a common header.

Multiple strainers attached to a common header will tend to behave as a single strainer. If one strainer of a series of strainers attached to a common header were to preferentially collect debris, its flow would decrease due to its increased head loss, thereby shifting its flow towards the other strainers and causing increased deposition on those strainers. The multiple strainers per common header is similar to dividing the surface area of a strainer into discrete elements and evaluating the deposition of one portion of the strainer area relative to the remaining area. Modeling each individual strainer on a common header and the shifting of the flow from one strainer to the next would have been relatively difficult to accomplish given the potential benefit of developing this capability. Therefore, multiple strainers attached to a common header are modeled as a single equivalent strainer.

Each pump attached to a common header is modeled with a separate flow capacity and a separate NPSH margin, but all pumps use a common time-dependent flow multiplier. Each pump provides ECCS flow to the reactor until strainer blockage head losses exceed its available margin of net positive suction head (NPSH) causing the pump to fail and its flow to cease. Pump performance, i.e., pump brake power, pump head, pump efficiency as a function of pump flow rate, was not modeled. The calculation of the strainer blockage head losses is based on the total flow of these pumps still operating on the common header, but the loss of NPSH of each pump is determined independently of the other pumps.

A pump is predicted to lose NPSH margin and fail when the strainer head loss due to debris deposition on the strainer exceeds its temperature-dependent NPSH margin. The temperature-dependent NPSH margin is determined using:

$$NPSH(t) = NPSH_o - \left(\frac{P_a - P_{vo}}{\gamma_{wo}} - \frac{P_a - P_v(T_w(t))}{\gamma_w(T_w(t))} \right) \quad (2\text{-}14)$$

where:

- $NPSH(t)$ = the temperature-dependent pump NPSH margin
- $NPSH_o$ = the pump NPSH margin at a user supplied reference temperature
- $T_w(t)$ = the time-dependent suppression pool temperature
- P_a = the standard atmospheric pressure (14.7 psia)
- P_{vo} = the water vapor pressure at the reference temperature
- $P_v(T)$ = the temperature-dependent water vapor pressure
- γ_{wo} = the specific weight of water at the reference temperature
- $\gamma_w(T)$ = the temperature-dependent specific weight of water

The total ECCS flow rate supplied to the reactor core is:

$$Q_{ECCS}(t) = \sum_{k=1}^{N_{str}} \sum_{n=1}^{N_{pk}} \delta_{kn} M(t,m) Q_{nk} \quad (2\text{-}15)$$

where:

- $Q_{ECCS}(t)$ = the time-dependent ECCS flow to the reactor core
- δ_{kn} = indicates whether or not pump n on header k is active (1=active and 0=not)
- $M(t,m)$ = the time-dependent flow multiplier for LOCA scenario m
- Q_{kn} = the full volumetric flow capacity of pump n on common header k
- N_{str} = the number of common headers or strainers

N_{pk} = the number of pumps attached to common header k.

The time-dependent flow multiplier which is provided in a tabular form by the user is generally intended to model the pump startup where the multiplier is initially zero and then increases to one when the pump has reached full speed. The shape of the flow multiplier curve between zero and one should represent the startup performance of the pump. The user is required to specify two pump flow multiplier curves, i.e., one curve for LLOCA weld break scenarios and one for MLOCA scenarios. The user may specify a pump flow multiplier curve that either does not reach a value of one to represent a degraded performance for the pumps or a curve that exceeds a value of one representing pumps that are driven beyond their normal rated flow capacity. However, when the flow multiplier is not used in its normal mode of zero to one, the user should refer to Section 4.4 of the Users Manual [NRC, NUREG/CR-6370] for information regarding the effect of a non-standard flow multiplier on the specification of the probabilistic criteria of failure to supply adequate ECCS flow to the reactor core.

The failure of the ECCS to provide adequate cooling to the core is used by BLOCKAGE to determine plant-wide strainer blockage probabilities. Basically the ECCS is assumed to fail when the total ECCS flow falls below a user specified minimum flow criteria once the ECCS flow has been established. Note that BLOCKAGE was designed to calculate loss of NPSH margin, therefore, ECCS flow reductions are based on loss of NPSH margin rather than actual pump trips. Also note that ECCS failure to provide adequate cooling to the core based on loss of NPSH margin is one of several methods by which the user can terminate the calculation (see Section 4.5 of User's Manual). The definition for the ECCS failure criterion that was coded into BLOCKAGE is:

$$\sum_{k=1}^{N_{str}} \sum_{n=1}^{N_{pk}} \delta_{kn} Q_{kn} < Q_{\min} \qquad (2\text{-}16)$$

where:

Q_{min} = the user specified minimum ECCS flow required cool the core.

The user can control this minimum flow criterion to tailor the probabilities report for a particular application. For example, the ECCS failure criteria can be specified as the loss of NPSH margin of any pump by setting the user input minimum ECCS flow rate to a value below the total capacity of all of the pumps but higher than the total flow of any combination of pumps where only one pump has lost NPSH margin. Thus, when any single pump is predicted to loss NPSH margin, the predicted flow to the reactor core drops below the specified minimum value and ECCS failure is signaled. A more realistic criterion would be to use a minimum flow required to cool the core that was determined by a core cooling code. Depending upon the arrangement of pumps and headers, multiple pump losses of NPSH margins may be required to actually fail to cool the reactor core. (It should be noted that when a single pump fails on a multiple pump header arrangement, the strainer blockage head loss existing at the time of loss of NPSH margin drops because the flow through the strainer is reduced.) These two schemes will likely result in different ECCS failure probabilities.

If ECCS failure occurs after the pumps have reached 100% capacity, then the failure is signaled when the total flow drops below the minimum required flow. However, in more extreme conditions, it is possible for the code to predict ECCS failure before full capacity is reached. Under these conditions, the code will determine if enough capacity exists with the remaining pumps to meet the minimum core cooling requirement once the pumps are running at full capacity. If the remaining total full capacity flow is sufficient to cool the core once the pumps reach full capacity, ECCS will not be considered to have failed. Therefore, it is assumed that the pumps will obtain full capacity in a reasonably short period of time, i.e., the core will not be damaged due to insufficient flow during this brief period immediately following the pipe break.

2.6 ECCS Strainer Screen Filtration Model

Debris entrained in the water being pumped through an ECCS recirculation strainer will either pass through the strainer with the water flow, or be deposited onto the strainer forming a debris cake. The fraction that is deposited onto the strainer is called the strainer filtration efficiency. The strainer filtration efficiency is a function of the type and size of debris and the thickness of fibrous debris cake already deposited onto the strainer.

A model was included in BLOCKAGE 2.5 that allows the strainer filtration efficiency to vary with the thickness of the fibrous debris cake already deposited onto the strainer. This model is illustrated schematically in Figure 2-1. The user specifies a strainer filtration efficiency for the bare strainer and a peak efficiency that occurs at a user specified cake thickness. Three options are available for interpolating between these two points: step, linear, and exponential. The exponential option requires that both efficiencies be non-zero. A set of efficiency values is specified for each settling velocity group of each type of debris material. This model allows fibrous materials, for example, to be filtered more efficiently than small particles and since velocity groups are related to debris size, larger particles can be filtered more efficiently than small ones. One model interpolation option and one thickness at which peak efficiency is achieved are specified for the entire calculation.

2.7 Strainer Head Loss Models

Debris accumulation on the ECCS strainer resists further flow though the strainer, thereby impeding the delivery of ECCS coolant to the reactor core. The BLOCKAGE code contains models to estimate the flow resistance referred to as strainer debris head loss. The head loss for each strainer is calculated independent of the other strainers. BLOCKAGE 2.5 contains four optional correlations that may be selected by the user to model the head loss for a debris cake consisting of fibrous and particulate debris. One correlation must be selected to be applied to all of the strainers in the calculation throughput the entire calculation. BLOCKAGE 2.5 also contains an additive term for estimating head loss due to metallic debris on the strainer that can be added to any of these four head loss correlations. The four correlations are the semi-theoretical NUREG/CR-6224 correlation [NRC, NUREG/CR-6224], the empirical BWROG correlation [BWROG, 1994], and two generic correlations which can be used to implement an alternate user correlation.

NUREG/CR-6224 Correlation The NUREG/CR-6224 head loss correlation is a semi-theoretical model developed to estimate the head loss due to a combination of fibrous and particulate debris deposited onto a strainer. This correlation has enough flexibility to handle a variety of materials such as fiberglass, mineral wool, and iron oxide particles by assigning proper values for the specific characteristics of each debris material. Its predictions have been validated for fiberglass fibrous debris and simulated suppression pool sludge [Rao and Souto, 1996].

The NUREG/CR-6224 fiber and particulate debris head loss correlation is:

$$\frac{\Delta H}{\Delta L_o} = Units\left[3.5 S_v^2 (1-\varepsilon_m)^{1.5} [1+57(1-\varepsilon_m)^3]\mu U + 0.66 S_v \frac{1-\varepsilon_m}{\varepsilon_m}\rho_w U^2\right]\left(\frac{\Delta L_m}{\Delta L_o}\right) \qquad (2-17)$$

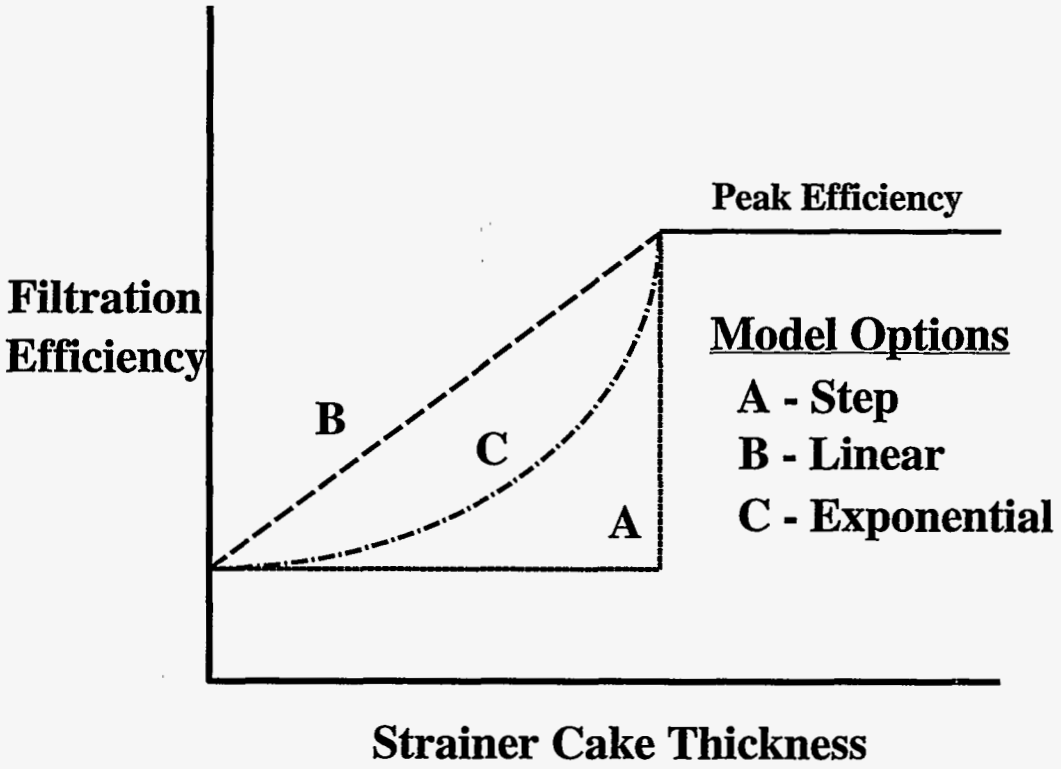


Figure 2-1: Strainer Cake Filtration Model

where:

ΔH = the strainer head loss (ft-water)
ΔL_o = the theoretical thickness of fiber bed (inch)
ΔL_m = the actual thickness of the fiber bed (inch)
S_v = the specific surface area of the fiber-particulate mixture (ft^2/ft^3)
ϵ_m = the porosity of the fiber-particulate mixture
U = the velocity of the water flow (ft/s)
μ = the viscosity of water (lbm/s-ft)
ρ_w = the density of water (lbm/ft^3)
Units = 4.1528×10^{-5} (ft-water/in/($lbm/ft^2/s^2$)).

The solution of this correlation requires some additional supporting equations. The ratio of the actual to theoretical thickness for the bed fibrous debris is a measure of the compression of the fibers due to the forces applied by the water flow passing through the bed. This compression ratio, $\Delta L_m/\Delta L_o$, was related to the head loss using the following correlation with values for empirical constants, α and γ, of 1.3 and 0.38, respectively. These values were validated for fiberglass fibrous debris and simulated suppression pool sludge.

$$\frac{\Delta H}{\Delta L_o} = \left[\left(\frac{1}{\alpha}\right)\left(\frac{\Delta L_o}{\Delta L_m}\right)\right]^{\frac{1}{\gamma}} \qquad (2\text{-}18)$$

The compression of the fibrous bed is limited by the particulate debris embedded between the fibers, i.e., the density of the bed can not exceed the rubble density of the particulate. This limiting bed thickness is:

$$\Delta L_{min} = \Delta L_o \frac{\rho_{rf}}{\rho_{rp}}(\eta + 1) \qquad (2\text{-}19)$$

where:

ΔL_{min} = the minimum bed thickness of the fiber bed (inch)
ρ_{rf} = the rubble density of the fibrous debris (lbm/ft^3)
ρ_{rp} = the rubble density of the particulate debris (lbm/ft^3)
η = the ratio of the particulate mass to the fiber mass.

The porosity of the fiber-particulate mixture on the strainer is:

$$\varepsilon_m = 1 - \left(1 + \frac{\rho_f}{\rho_p}\eta\right)(1-\varepsilon_o)\left(\frac{\Delta L_o}{\Delta L_m}\right) \qquad (2\text{-}20)$$

where:

ρ_f = the material density of fibers (lbm/ft^3)
ρ_p = the material density of particles (lbm/ft^3)
ϵ_o = the theoretical fiber bed porosity

The theoretical fiber bed porosity is:

$$\varepsilon_o = 1 - \frac{\rho_{rub}}{\rho_{mat}} \qquad (2\text{-}21)$$

where:

ρ_{rub} = the rubble density of the fibrous debris on the strainer (lbm/ft^3)
ρ_{mat} = the material density of the fibrous debris on the strainer (lbm/ft^3)

The specific surface area for the fiber-particular mixture, S_v, is determined from the specific surface area of fibers, S_{vf}, and the specific surface area of the particles, S_{vp}, is:

$$S_v = S_{vf}\left[\frac{1 + \frac{\rho_f}{\rho_p}\eta\frac{S_{vp}}{S_{vf}}}{1 + \frac{\rho_f}{\rho_p}\eta}\right] \qquad (2\text{-}22)$$

BLOCKAGE 2.5 solves these equations simultaneously using an iterative numerical solution technique which is described in Section 3.2. The head loss correlation and the fiber compressibility function constitute a two-equation and two-unknown set of equations that can not be solved explicitly for the head loss as a function of other variables. Including the minimum fiber bed thickness, ΔL_{min}, into the solution essentially creates a set of three equations which must be solved simultaneously.

<u>BWROG 1994 Correlation</u> The BWROG 1994 head loss correlation is an empirical correlation developed by the BWR Owners' Group for fiberglass fibers combined with oxide corrosion products [BWROG, 1994]. The BWROG 1994 correlation is

$$\Delta H = \left[17.71\, M_L^{0.85}\left[1 + 0.23\left(\frac{M_C}{M_L}\right)^{1.37}\right]\right]U + \left[17.0\, M_L^{0.93}\left[1 + 0.27\left(\frac{M_C}{M_L}\right)^{1.77}\right]\right]U^2 \qquad (2\text{-}23)$$

where:

ΔH = the strainer head loss (ft-water)
U = the strainer approach velocity (ft/s)
M_L = the mass of fiber on the strainer per unit area (lbm/ft^2)
M_c = the mass of corrosion product on the strainer which has passed onto or through the fiber bed per unit area (lbm/ft^2).

As implemented in BLOCKAGE 2.5, all particulate debris deposited onto the strainers is included in the corrosion products variable, M_C, not just the iron oxide particles, e.g., paint chips and calcium silicate.

<u>First Generic Correlation</u> This generic correlation was designed to allow the user to devise and implement a customized correlation based upon the theoretical fiber bed thickness and the fiber to particulate mass ratio. Further, it allows two distinct velocity terms. If this optional correlation is selected, then the user must provide input values for each of the 12 constants of the correlation. This first generic correlation is

$$\Delta H = A_1 \Delta L_o^{B_1} (C_1 + D_1 \eta)^{E_1} U^{F_1} + A_2 + \Delta L_o^{B_2} (C_2 + D_2 \eta)^{E_2} U^{F_2} \quad (2\text{-}24)$$

where:

ΔH = the strainer head loss (ft-water)
U = the strainer velocity (ft/s)
ΔL_o = the theoretical fiber bed thickness (ft)
η = the ratio of the particulate mass to the fiber mass
$A_1, B_1, C_1, D_1, E_1, F_1$ = user defined constants
$A_2, B_2, C_2, D_2, E_2, F_2$ = user defined constants.

An example of the use of this generic correlation is as follows. If the user specifies that the constants C_1, A_2 as zero; D_1 as one; and recognizes that η is proportional to $\Delta L_s/\Delta L_o$ (nominal thickness of sludge divided by the theoretical fibrous insulation thickness), the following simple head loss correlation can be formed:

$$\Delta H = a \Delta L_o^{b} \Delta L_s^{c} U^{d} \quad (2\text{-}25)$$

where:

ΔH = the strainer head loss
U = the strainer water flow velocity
ΔL_s = the nominal thickness of particulate sludge
ΔL_o = the theoretical fiber bed thickness
a = $A_1(\rho_p/\rho_s)/E_1$
ρs = the material density of a sludge particle
ρf = the material density of a fiber
b = $B_1 - E_1$
c = E_1
d = F_1.

Thus, the 12 constants required by the generic correlation can be deduced from the four constants of the simple correlation which would be known to the user.

Second Generic Correlation The second generic correlation is similar to the first generic correlation. The theoretical fiber bed thickness was replaced by the mass of fibrous debris per unit area of the strainer. If this optional correlation is selected, then the user must provide input values for each of the 12 constants of the correlation. This second generic correlation is

$$\Delta H = A_1 \left(\frac{M_f}{A_{str}} \right)^{B_1} (C_1 + D_1 \eta)^{E_1} U^{F_1} + A_2 \left(\frac{M_f}{A_{str}} \right)^{B_2} (C_2 + D_2 \eta)^{E_2} U^{F_2} \quad (2\text{-}26)$$

where:

ΔH = the strainer head loss (ft-water)
U = the strainer water flow velocity (ft/s)
M_f = the mass of fibrous debris on the strainer (lbm)
A_{str} = the strainer surface area (ft_2)
η = the ratio of the particulate mass to the fiber mass
$A_1, B_1, C_1, D_1, E_1, F_1$ = user defined constants
$A_2, B_2, C_2, D_2, E_2, F_2$ = user defined constants.

Metallic Head Loss Term The primary contribution to strainer head loss is expected to come from a debris bed consisting of fiber and particles; however, the potential exists for shredded metallic debris from destroyed RMI to contribute to the head loss. Correlations for debris beds consisting of RMI debris with fiber and particles or even RMI debris alone were not available for inclusion into BLOCKAGE 2.5. The four correlations described above are all based on fiber and particulate debris beds. Therefore, a term was incorporated into BLOCKAGE 2.5 which was added to each of the above four fiber-particulate correlations that provides the user with a method whereby the metallic head loss can be estimated and included in a calculation. This metallic head loss term is a function of the thickness, porosity, and specific surface area of the metal debris bed and the term has a user specified leading coefficient. The metallic head loss term has not been validated and it is the users responsibility to determine an appropriate and valid coefficient when incorporating this term into a calculation. This term can be easily deactivated by simply specifying zero for the leading coefficient. The metallic head loss term is:

$$\Delta H_{met} = Units\ K_{met} S_{met} \frac{1 - \varepsilon_{met}}{\varepsilon_{met}} \rho_w U^2 \Delta L_{met} \quad (2\text{-}27)$$

where:

ΔH_{met} = the strainer head loss due to metallic debris (ft-water)
U = the strainer water flow velocity (ft/s)
K_{met} = the user supplied leading coefficient
S_{met} = the specific surface area of the metallic debris (ft^2/ft^3)
ϵ_{met} = the porosity of the metallic debris
ρ_w = the density of water (lbm/ft^3)
ΔL_{met} = the thickness of the metallic debris bed (inch)
Units = 4.1528×10^{-5} (ft-water/in/($lbm/ft^2/s^2$)).

The porosity of the metallic debris is:

$$\varepsilon_{met} = 1 - \frac{\rho_{rub}}{\rho_{mat}} \qquad (2\text{-}28)$$

where:

ρ_{rub} = the rubble density of the metallic debris on the strainer (lbm/ft^3)
ρ_{mat} = the material density of the metallic debris on the strainer (lbm/ft^3).

Debris Bed Thickness Wetwell debris transport in BLOCKAGE is performed in terms of volume of debris at the fabricated density of the debris, whereas debris deposited onto a strainer will have a density described by the rubble density. Previously, it was generally assumed that fibrous debris would have the same density as when the material was an insulation target; however, this assumption may not always be correct. The equation that BLOCKAGE 2.5 uses to calculate the fibrous debris bed and again for the metallic debris bed is:

$$t_{bed} = \frac{\rho_{fab}}{\rho_{rub}} \frac{V_{fab}}{A_{str}} \qquad (2\text{-}29)$$

where:

t_{bed} = the material density of the metallic debris on the strainer (lbm/ft^3)
ρ_{fab} = the fabricated density of the fibrous or metallic debris on the strainer
ρ_{rub} = the rubble density of the fibrous or metallic debris on the strainer
V_{fab} = the volume of debris deposited onto the strainer at the fabricated density
A_{str} = the screen area of the strainer.

Mixture Properties Since debris deposited on a strainer can consist of multiple types of fibers or multiple types of particles or multiple types of metals, and the debris head loss correlations can only handle one type of fiber and one type of particles or one type of metal shreds, BLOCKAGE calculates appropriate averaged mixture properties which are used by the head loss correlations. Fabricated, rubble, and material densities and specific surface areas are calculated for each classification of debris (i.e., fibrous, particulate, or metals). The generalized equation for calculating mixture densities is:

$$m\rho x_{ck} = \frac{\sum_{i=1}^{N_{types}} \delta c_i \, \rho fab_i \, Vfab_{ik}}{\sum_{i=}^{N_{types}} \delta c_i \frac{\rho fab_i}{\rho x_i} Vfab_{ik}} \qquad (2\text{-}30)$$

where:

$m\rho x_{ck}$ = the mixture density, of debris structure x, of debris class c, on strainer k
x = indicates either fabricated, rubble, or material density
c = indicates the debris classification, i.e., fiber, particulate, or metallic
k = indicates the strainer
δc_i = indicates whether or not type i debris is classification c debris (1=yes, 0=no)
ρfab_i = the fabricated density of debris type i
ρx_i = either the fabricated, rubble, or material density of debris type i
$Vfab_{ik}$ = the volume of debris type i deposited on strainer k based on the fabricated densities
N_{type} = the total number of debris types of all classes in the calculation.

The numerator of the above equation represents the mass of debris class c deposited on strainer k, whereas the denominator represents the volume of debris class c deposited on strainer k based on density type x (i.e., fabricated, rubble, or material).

The generalized equation for calculating mixture specific surface areas is:

$$mSv_{ck} = \frac{\sum_{i=1}^{N_{types}} \delta c_i \; Sv_i \frac{\rho fab_i}{\rho mat_i} Vfab_{ik}}{\sum_{i=1}^{N_{types}} \delta c_i \frac{\rho fab_i}{\rho mat_i} Vfab_{ik}} \qquad (2\text{-}31)$$

where:

mSv_{ck} = the mixture specific surface area of debris class c on strainer k
δc_i = indicates whether or not type i debris is classification c debris (1=yes, 0=no)
Sv_i = the fabricated density of debris type i
ρmat_i = the material density of debris type i.

These mixture properties are homogeneous properties applied uniformly to the entire debris bed. In reality, the ratio of particulate to fiber debris may not be uniform on the strainer since debris of differing transport and pool settling characteristics will tend to deposit onto a strainer non-uniformly. In addition, the strainer head loss will likely continue to build as additional debris is deposited onto the strainer assuming that once debris is deposited it can not leave the strainer bed or relocate in the bed. In some cases, the NUREG/CR-6224 correlation has predicted a non-realistic head loss peaking followed by a significant decrease in the head loss as the bed continues to build with fibers but not particles, resulting in a lower particulate to fiber mass ratio for the overall debris bed.

2.8 Temperature Dependent Water Properties

Three physical properties of water are used by BLOCKAGE 2.5 models: the density of water, the viscosity of water, and the vapor pressure of water. The water density and viscosity are used by the head loss correlations, and the vapor pressure is used to calculate the NPSH margin. The temperature of the suppression pool is time-dependent (specified by user input). These water properties were implemented into BLOCKAGE 2.5 using curve fits of applicable data, therefore the user does not need to supply any of these properties.

2.9 Probabilistic Models

The BLOCKAGE code has two models for calculating the pipe weld break frequency for a particular weld. These models are referred to as the Weld Method and the Plant Method. The Weld Method can be accessed using the Graphical User Interface as well as running BLOCKAGE from the DOS level but the Plant Method can only be accessed from the DOS level.

Weld Method The weld method specifies the weld break frequencies pipe diameter and weld type. The pipe weld break frequencies are input to the calculation by the user as a table of break frequencies that are a function of the pipe diameter and the type of weld. The diameters are grouped into classes to reduce the size of the table. The number of diameter classes (1 through 4) and the minimum diameter of each class are user specified. There may be as many as 20 weld types. The weld break frequency for each individual pipe weld break of the calculation is then simply extracted from this table based on the pipe diameter and weld type of that particular pipe weld.

Plant Method The plant method specifies the break frequencies for all of the weld breaks in a diameter class (same diameter class scheme as the Weld Method), i.e., the total break frequency for all of the pipe welds with pipe diameters that fit within a diameter class. Then the break frequencies for an individual pipe weld break is determined using weighting factors that are specified by the diameter class and the weld type. The weighting factor equation is

$$f_i = \frac{wf_i}{\sum_{j=1}^{N_{welds}} \delta dc_{ij} \; wf_j} fdc_i \qquad (2\text{-}32)$$

where:

f_i = the pipe weld break frequency for pipe weld break i
wf_i = the weighting factor associated with pipe weld break i
wf_j = the weighting factor associated with pipe weld break j

δdc_{ij} = indicate whether or not the diameter class for weld break j is the same as for break i

fdc_i = the break frequency for all the weld breaks of the same diameter class as weld break i.

N_{welds} = the total number of welds in the calculations

The summation in the denominator of this equation sums the weighting factors for each diameter class over all the weld breaks of that class. The individual pipe weld weighting factors, wf_i, are a function of both diameter class and weld type. The equation simply redistributes the diameter class break frequency among all the weld breaks of that diameter class.

Probabilities Reports BLOCKAGE 2.5 determines the pipe weld break frequency for each pipe weld and then determines whether or not the ECCS fails to provide adequate flow to the reactor core per the user input minimum ECCS flow requirement. The probabilities for the weld break frequencies and the frequencies for the failure to cool the reactor core are correlated by pipe diameters, diameter classes, weld break location, and piping systems.

3.0 BLOCKAGE Code Programmer's Guide

The structure of the BLOCKAGE 2.5 code is described in this Reference Manual to provide detailed code information to the user of the code. The architecture of the code and several internal aspects of the code such as numerical solutions, the control of the calculational time steps, and the integration and interpolation of time-dependent rates are discussed. Note that this section applies solely to the stand-alone FORTRAN code and not the GUI which is discussed separately in Section 5.

3.1 Code Architecture

The code architecture demonstrates the organization of the analytical models and the general flow of the calculational solution. The layout and the content of the individual subroutines, the organization of the source code, the method and location of data storage, the calculational units, and the organization of the input and output files are also.

3.1.1 FORTRAN File Description

The FORTRAN source code for the BLOCKAGE 2.5 code is contained in five separate source files. When compiled, the files are appended together into a single unit using appropriately placed INCLUDE statements. The five source files are:

- BLKAGE.FOR
- POOLIN.FOR
- POOL.FOR
- REPORT.FOR
- DATASTOR.FOR

The main program is at the beginning of source file BLKAGE.FOR. Source file, DATASTOR.FOR, contains the programming needed to store data for the code and is discussed in the following section.

The total number of lines of source coding for the entire code, including comment statements, is 6,850 lines. Abundant comments were placed throughout the code to increase the readability of the coding. The organization and location of the code subroutines is provided in Table 3-1. The functions of each of these subroutines is provided in Section 3.1.5.

3.1.2 Data Storage

Data in BLOCKAGE 2.5 is stored primarily in a series of COMMON blocks located in the source file DATASTOR.FOR. Many fixed integer parameters used to set dimensions in arrays, such as the maximum number of pipe weld breaks allowed, are specified with PARAMETER statements. The COMMON blocks are arranged by their type of data, i.e., integer, real, or character. The appropriate declarations of data types, i.e., INTEGER, REAL, CHARACTER*n, were grouped with their corresponding COMMON blocks. The variables contained in the data storage were defined in comments statements placed prior to the COMMON blocks containing the variables. The contents of this file are inserted into each subroutine requiring access to this data using INCLUDE statements placed immediately after each subroutine statement.

3.1.3 Calculational Units

BLOCKAGE 2.5 was coded entirely in English units (i.e., inch, feet, lb-mass, seconds) corresponding to methodology documented in NUREG/CR-6224 and the units employed by the bulk of the nuclear power industry in the United States. Pressure heads such as the strainer head losses and NPSH margins were coded in units of ft-water at standard temperature.

Table 3-1: Location of Subroutines Within Source Files

BLKAGE.FOR	POOLIN.FOR	POOL.FOR	REPORT.FOR
Program Blockage	POOLIN	POOL	DIALOC
BEGIN	LABEL	HLOSS	DIASYS
PINPUT		SELECT	SYSLOC
WINPUT		WATER	
SETUP		INTEG	

3.1.4 Input/Output Files

BLOCKAGE 2.5 requires that the user provide two separate input files either directly or through the GUI. The input data is arranged in a free-format style and read into the code using list-directed READ statements. Data must be entered for each of the required input parameters on a line-by-line basis as indicated in the user input descriptions and rules found in Appendix B of the User's Manual [NRC, NUREG/CR-6370].

The input files are described below:

WELD.DAT This input file contains plant-specific data regarding the pipe welds and their insulation targets, including pipe diameters, piping system identifications, and weld types. This is the type of information that would generally be collected from analyzing plant drawings and would usually only be done one time. It was intended that this input file, after quality assurance procedures were completed, could be protected from further change.

INPUT.DAT This input file generally contains input parameters specific to a particular calculation. A user may wish to change data between calculations, such as debris attributes, pumping system parameters, and time-dependent parameters when performing parameter sensitivity analyses.

BLOCKAGE 2.5 creates several output files to display the calculational results, depending upon the options selected. The subroutines that write output files are listed in Table 3-2. These output files are:

Table 3-2: List of Subroutine that Write Output Files

Output File	Originating Subroutine
TARGET.OUT	SETUP
WELD.OUT	POOL
PLOT.OUT	POOL
SUMMARY.OUT	POOL
SPLOT.OUT	POOL
FREQ.OUT	DIALOC, DIASYS, SYSLOC
PWELD1.OUT	Program Blockage
PWELD2.OUT	Program Blockage
ERROR.OUT	Several

TARGET.OUT This output file reflects the weld and target input and contains the calculated insulation volumes for each target and pipe weld break in each destruction zone. It also contains the total debris generated for each target and for each weld break. This information is written after the debris volumes are calculated and prior to beginning the time-dependent debris transport solution.

WELD.OUT This output file provides detailed output data for user selected pipe weld breaks at a user specified print frequency, at a pump loss of NPSH margin event, and at the end of the calculation. The output details include debris transport data, debris deposition on strainer data, and pump performance and loss of NPSH margin data. A summary of the debris volumes is written at the beginning of the file. The detailed output is written at the end of a particular time cycle whenever the programming determines that it is time to write another cycle. This determination is based on whether or not a pump lost its NPSH margin during the current cycle or whether or not the current time exceeds the user specified print frequency. A summary of loss of NPSH margin events is written at the end of the file.

PLOT.OUT This output file contains the time-dependent results for user selected pipe weld breaks. The file contains a total of 84 columns of data with the first column containing the time associated with that line of data. The data is written in a conversion mode format with 10 characters per column with a single space between the columns, i.e., 84(1X,1PE10.3). The data is written at the end of a particular time cycle per the user specified print frequency. When an output variable is not applicable to the current calculation, such as data for strainers 2 through 8 in a calculation that has only a single strainer, the column is filled with zeros.

SUMMARY.OUT This plant-wide summary contains results that were written to this output file for each pipe weld break at the end of the time-dependent solution. This file shows whether or not a pump lost its NPSH margin and whether or not the ECCS failed to provide long-term cooling. The times at which failures occurred are provided. It contains the pipe break frequency and it contains the total volumes of the debris that were transported to the wetwell including the debris that originated in the wetwell for each debris classification.

SPLOT.OUT This summary also shows plant-wide results for ECCS performance. However, the results have been collated by pipe weld break size, i.e., large or medium LOCA, and the results have been sorted by the total fibrous debris the was transported to the wetwell, including fibrous debris that originated in the wetwell. These data are saved at the end of the time-dependent solution for each weld break and sorted after all the weld breaks solutions have been completed. It is then written to the output file after sorting is completed.

FREQ.OUT This output file is written when the user selects the option of producing probabilistic results. It contains the pipe weld break frequencies and the ECCS failure to provide long-term cooling frequencies. The frequencies are collated by weld break diameter, piping system, and break location. These results are written near the end of the calculation.

PWELD1.OUT This output file is written only when the user selects the parameter sensitivity option (ISENS=1). Note that this option is not available when running the code with the GUI. Selected output parameters are written in a single line format for the first weld selected by the user for detailed output (idweld). The general idea is that a single line of data can be produced for each calculation of the parameter study, then these lines can all be appended into a single spreadsheet for further analysis. Guidance in performing a parameter sensitivity study is provided in Appendix B of the User's Manual.

PWELD2.OUT This output file is identical to PWELD1.OUT except that data for the second user selected weld is written to this file.

ERROR.OUT This output file will contain any error messages written during the course of the calculation. The primary source of errors written to this file pertain to invalid input parameters. A message will appear here should the iterative numerical solution of the NUREG/CR-6224 head loss correlation fail to converge. If there were no errors encountered, this file will be empty.

3.1.5 Subroutine Description

The general organization of the BLOCKAGE 2.5 subroutine structure is illustrated in Figure 3-1, which shows the locations of the subroutine call statements and the subroutines which access the data storage. These subroutines are shown organized into the following three separate groups:

- Input Processing and Setup Calculations
- Time-dependent Solution
- Probabilities Report

The input is read first by the code and processed prior to performing calculations. Initial calculations and manipulation of the input is perform before beginning the time-dependent solutions which consume the majority of the computational time. Then , if selected, the code finishes by correlating the probabilistic results.

The role of each subroutine in BLOCKAGE 2.5 is described here:

Program BLOCKAGE The main executive control function is performed in this program portion of the code. It calls the main subroutines, opens and closes the input and output files, and writes a welcome and a termination message directly to the screen.

Input Processing and Setup Calculations

BEGIN This subroutine reads and processes the initial input parameters of input file, INPUT.DAT. Some of these parameters are needed to control subsequent input processing in input file, WELD.DAT.

WINPUT This subroutine reads and processes the input file, WELD.DAT, and it calculates the volumes of the intact insulation targets for each of the three destruction zones from the input dimensions.

PINPUT This subroutine reads and processes the probability input data from the input file, INPUT.DAT.

POOLIN This subroutine reads and processes the remaining input data from the input file, INPUT.DAT, which includes the

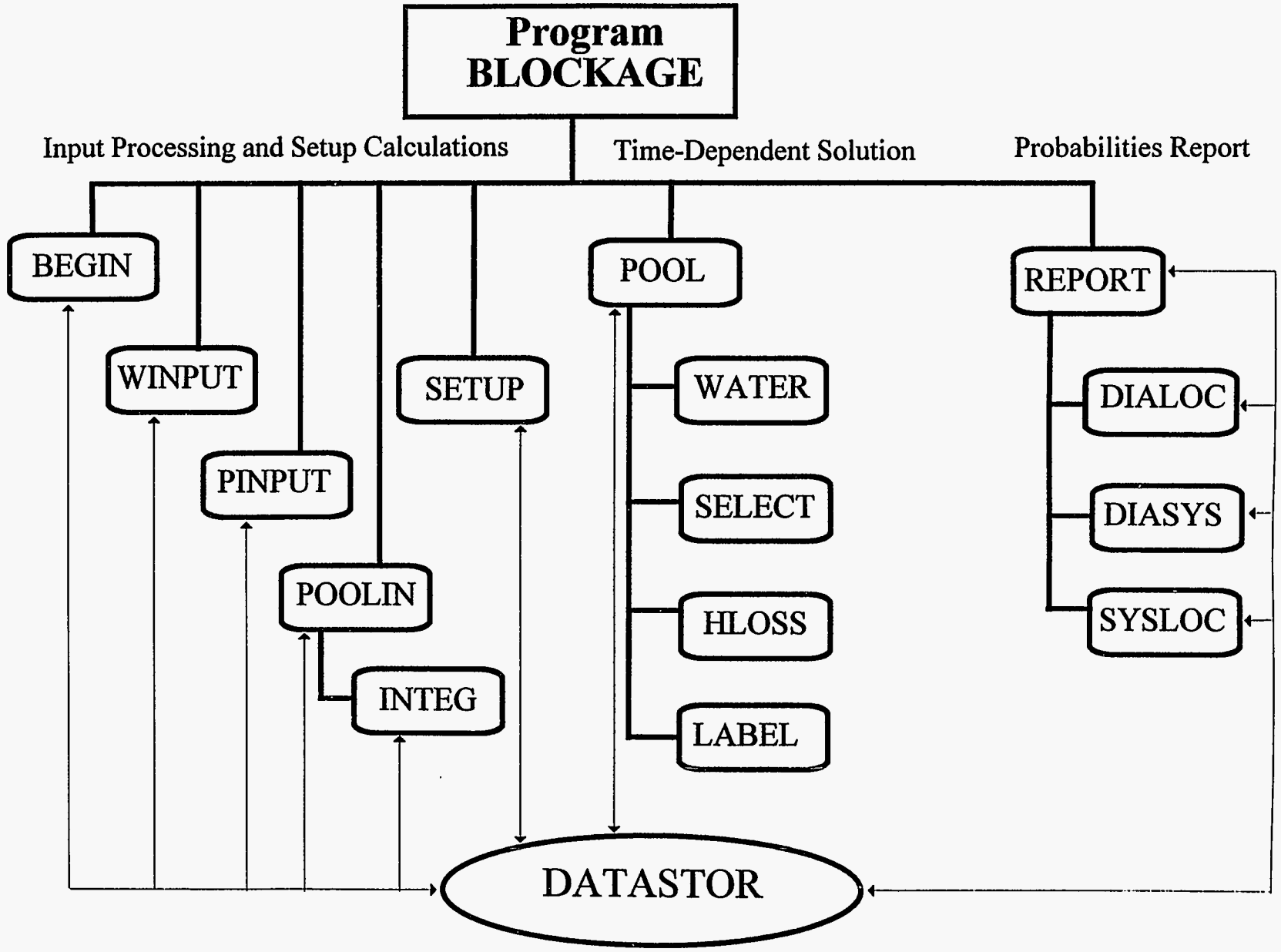


Figure 3-1: Subroutine Structure of BLOCKAGE 2.5

debris attributes, ECCS pump and header data, and the time-dependent input parameters.

INTEG This subroutine, called by subroutine POOLIN, integrates the time-dependent drywell debris transport rates as part of its input processing function using the user input time-dependent time steps.

SETUP This subroutine performs a number of calculational setup functions prior to entering the time-dependent solution. These functions include calculating the weld break frequency for each postulated pipe break using either the Plant Method or the Weld Method; calculating the insulation debris volumes using the destruction fractions; writing the TARGET.OUT output file; and determining control indices associated with pipe diameters, diameter classes, and weld types for each postulated pipe weld break.

Time-Dependent Solution

POOL This subroutine performs the time-dependent solution of the debris transport from the drywell to the wetwell and within the suppression pool. It calculates the build up of debris on the strainers and predicts the strainer head loss and trips the ECCS pumps predicted to lose NPSH margin.

WATER This subroutine contains mathematical curve fits for the temperature-dependent water properties of density, viscosity, and vapor pressure. It is called by subroutine POOL whenever one of these properties is needed.

SELECT This subroutine performs the data interpolations between entries in the tabular time-dependent input parameters and in the filtration efficiencies that are a function of the fibrous debris bed thickness. It contains algorithms for step, linear, and exponential interpolation.

HLOSS This subroutine performs the iterative numerical solution of the NUREG/CR-6224 head loss correlation. It is called by subroutine POOL which supplies the current theoretical thickness of fibrous debris deposited onto the strainer and the particulate-to-fiber mass ratio, and the flow velocity, along with the appropriate debris characteristics data. Subroutine HLOSS then returns the actual fiber bed thickness and the predicted head loss.

LABEL This subroutine contains character data in the BLOCK DATA form which is used to print headings in the output files.

Probabilities Report

REPORT This subroutine generates a report of the plant-wide pipe weld break frequencies and the corresponding ECCS failure frequencies. It also provides the fractions of the breaks that were predicted to prevent the ECCS from performing its long-term cooling function.

DIALOC This subroutine, called by subroutine REPORT, collates the probability results by pipe diameter and break location.

DIASYS This subroutine, called by subroutine REPORT, collates the probability results by pipe diameter and piping system associated with the pipe break.

SYSLOC This subroutine, called by subroutine REPORT, collates the probability results by piping system associated with the break and break location.

3.1.6 Calculational Flow Chart

Three schematic flow diagrams are presented here to facilitate understanding of the calculations performed by BLOCKAGE 2.5. These diagrams include

- a diagram illustrating the entire code
- a schematic of the input processing procedure
- a schematic of time-dependent solution.

The calculational flow of the overall code is illustrated in Figure 3-2. Following the program initiation in Program Blockage, a series of subroutines are executed which read the input files, INPUT.DAT and WELD.DAT. then process and save the data in data storage. Then a number of calculations and initializations are performed prior to solving the time-dependent debris transport equations. These setup calculations include:

- the calculation of the weld break frequencies for each user specified pipe weld break per the probabilistic input data and the method selected by the user, i.e., Plant or Weld Method.

- the calculation of the intact and the target generated debris volumes from the user specified targets by destruction zone, by target, and by weld. Volumes are also summed by debris classification and the ratios of debris to intact volumes are computed for the output. A number of indices are established for later calculational control, such as an index that indicates LOCA size, pipe diameter class, or break location. These indices are also used to place and locate data in storage arrays.

- if the debris volumes were calculated from target destruction rather than from user specified volumes, the output file, TARGET.OUT, is written after the intact and debris volumes have been calculated for each target of each weld break.

Once the setup calculations are completed, the code performs the time-dependent debris transport portion of the calculational run. Four output files are written at the end of the time-dependent solution, i.e., WELD.OUT, PLOT.OUT, SUMMARY.OUT, and SPLOT.OUT. If the user selected the probabilistic report and target generated input options, the break and ECCS failure probabilities are then collated and written to output file, FREQ.OUT. The calculation transfers back to Program BLOCKAGE where it writes the sensitivity output files, PWELD1.OUT and PWELD2.OUT, when the sensitivity option is active, and then closes all of the input/output files before finally stopping.

The input processing procedure is illustrated in the schematic shown in Figure 3-3. Generally, each line of input is read individually and processed prior to moving onto the next line of input. When a read error is encountered, the code writes an error message to ERROR.OUT, prints the message "Input Error" to the computer screen to alert the user, and then stops. Following a successful read of an input line, the input data is validated per a number of input requirements which are discussed further in Section 3.6. If an input data (integer, real, or character data) does not pass its validation tests, the code writes an error message to ERROR.OUT and to the screen. The process is repeated until each line of input is successfully read, processed, and placed in data storage before moving on to the setup calculations.

The time-dependent solution of the debris transport equations is illustrated in the schematic shown in Figure 3-4. This schematic shows two major calculational loops. The outer most loop executes the time-dependent solution for each of the weld breaks, one at a time. The next loop advances the time-dependent solution for a particular weld break through time until the requirements for the user selected termination option are meet, such as the calculational end time. Many other calculational loops are imbedded in the flow which cycle the calculation through a particular section of coding. Examples of these minor loops include cycling a portion of the calculation through each settling velocity group of each debris type, or through each of debris classifications, or through each of the individual strainers if more than one was input, or though each of the ECCS pumps in the calculation.

A section of coding at the beginning of the time loop determines the time step size for the current time cycle. The time step is determined by interpolating the user input time step table but then may be modified to force the calculation to match a user input event time, such as the end-of-blowdown time or the end-of-calculation time.

Each of the time-dependent user input parameters is determined for the current time cycle prior to executing the transport equations. This is done by interpolating the time-dependent input tables per the user selected interpolation option.

Next the debris transport equations are executed. First the debris transport rates are calculated as the finite changes in a volume per the current time step, i.e., $\Delta V/\Delta t$. The various volume inventories are updated. The volumes and rates are then summed and collated as needed for the strainer head loss predictions and for the output files.

The debris volumes deposited onto the strainers are grouped by debris classification (i.e., fibrous, particulate, metallic, or ignore) prior to performing the head loss calculation. Then the debris masses, mass ratios, mixture densities and specific surface areas, are calculated based on the debris classifications. The volumes for the fibrous and metallic debris, based on their mixture rubble

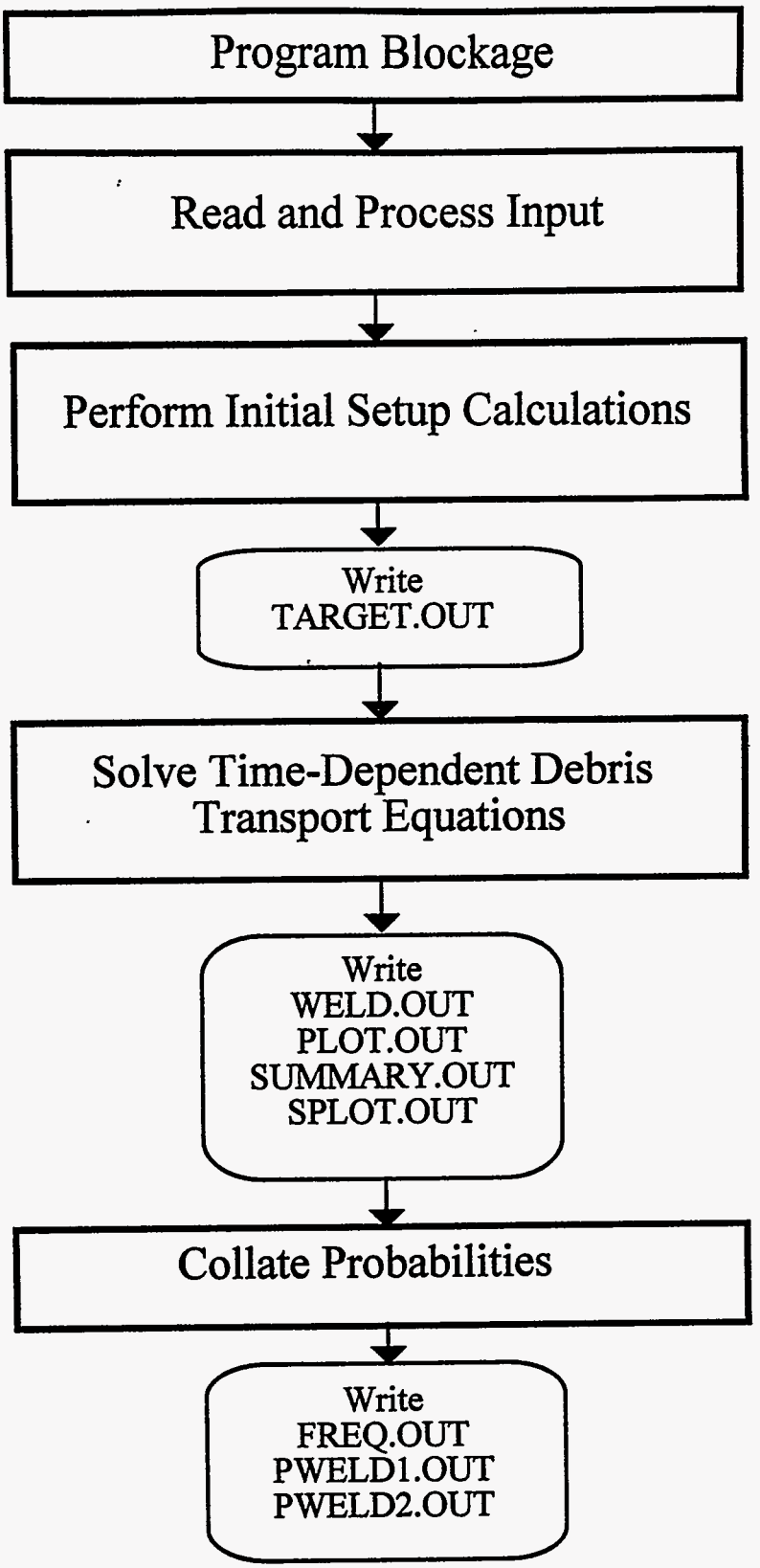


Figure 3-2: Overall Flow of a BLOCKAGE 2.5 Calculation

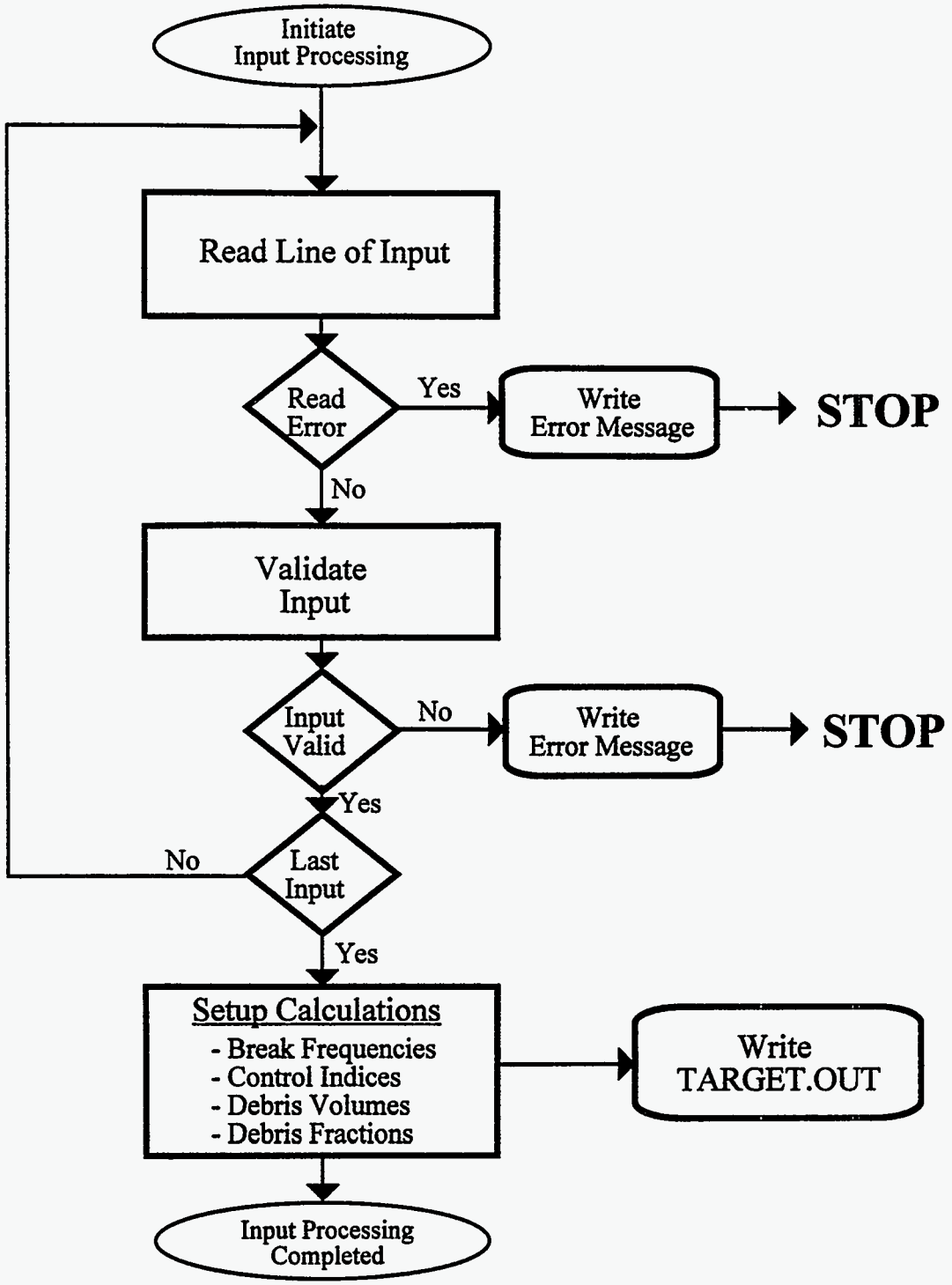


Figure 3-3: Input Processing Procedure

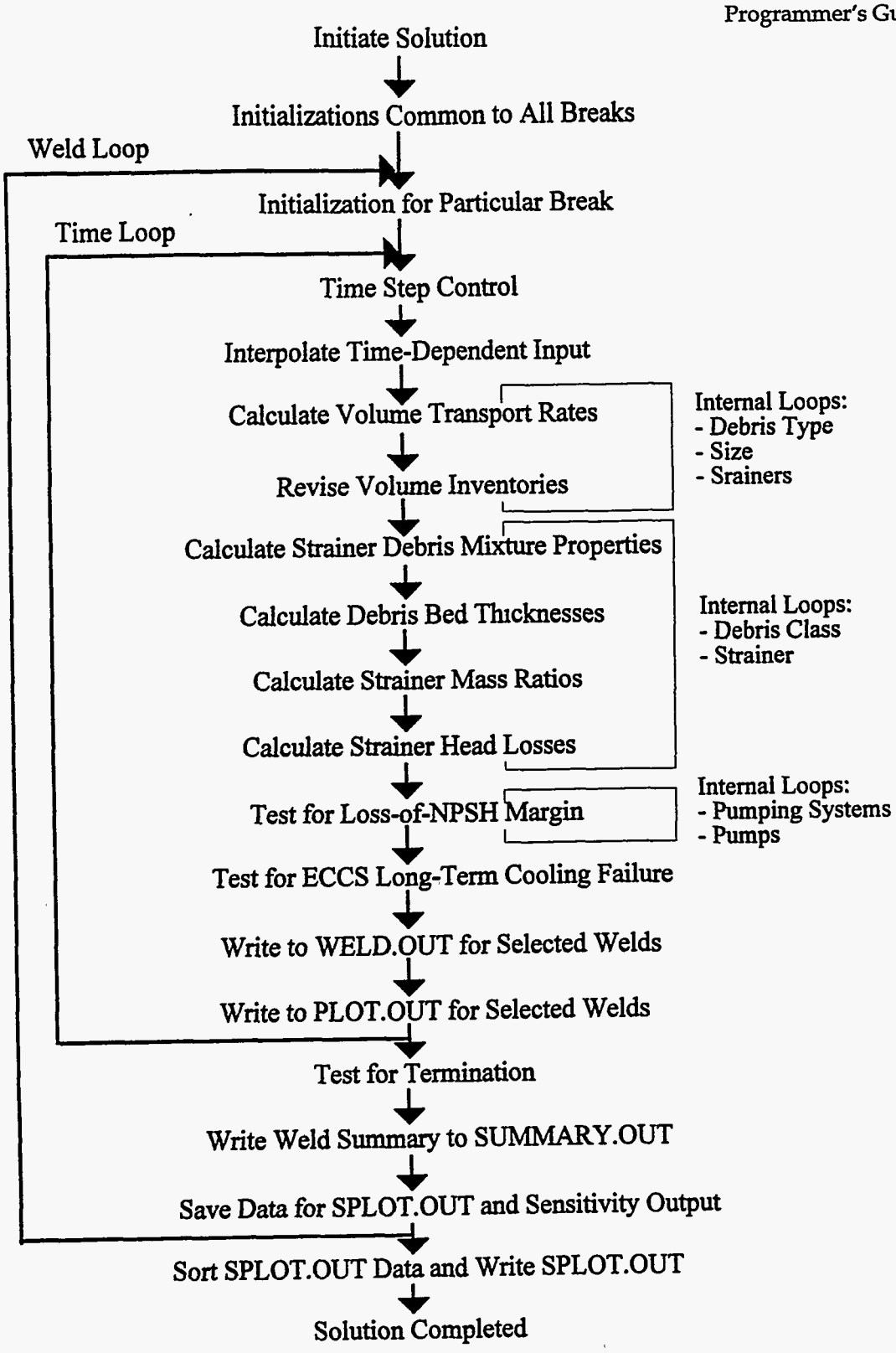


Figure 3-4: The Time-dependent Solution of the Debris Transport Equations

densities, are used to calculate the theoretical fibrous and metallic debris beds on each strainer.

The head loss correlation selected by the user is then used to calculate the debris bed head loss for each strainer. The temperature-dependent (thus time-dependent) NPSH margin for each pump is compared to the predicted head loss for its associated strainer to determine if the pump has lost NPSH margin. The pump loss of NPSH margin information is maintained in an array in a manner that ensures that once tripped, the pump remains tripped for the remainder of the calculation. Finally, the flow rates of all the non-tripped pumps are summed to determine if the ECCS has failed in its mission to provide long-term cooling.

Output information is written to the output files, WELD.OUT and PLOT.OUT, whenever indicators indicate that it is time to write to these files. The indicators are first set according to the users selection of welds for which information was to be saved in these files. These indicators are also controlled by the user specified print frequencies, and, for WELD.OUT, whether or not a pump tripped in this time cycle.

At the end of the time loop, a line of summary information is written to output file, SUMMARY.OUT, for each of the pipe weld breaks and some of this information is also saved in arrays at the end of the time loop for the SPLOT.OUT file. The SPLOT.OUT file, however, cannot be written until all of the weld breaks are completed because this information is sorted according to the volume of fibrous debris. Data is also saved for the parameter sensitivity output files, PWELD1.OUT and PWELD2.OUT, whenever that option is selected by the user. These two files are actually written from Program Blockage just prior to concluding the calculation.

3.1.7 Coding Comments

Comments were placed throughout the coding of BLOCKAGE 2.5 which describe the purpose of the coding that follows and to provide guidance to the programmer. Definitions were provided for the major code variables. In particular, all of the variables stored in the data storage were defined just prior to their use in COMMON statements.

3.2 Numerical Solutions

BLOCKAGE 2.5 numerically solves the time-dependent debris transport equations which model the transport of debris from the drywell to the wetwell and within the suppression pool. It also solves the equation set of the NUREG/CR-6224 head loss correlation using an iterative numerical solution technique.

3.2.1 Time-Dependent Debris Transport

The numerical solution of the debris transport equations employ a relatively straightforward incremental advancement of the equations through time by finite increments of time called time steps. The time is initialized to zero at the beginning the numerical solution for each of the weld breaks.

The time, t, is advanced at the beginning of the first time cycle by the time-dependent time step, Δt, as shown:

$$t = t_{old} + \Delta t(t) \qquad (3\text{-}1)$$

Each of the time-dependent input parameters, represented by $p_i(t)$, is determined for the new time cycle by interpolating the input tables. Then each of the differential transport equations are advanced one increment in time using the new parameters and the last time step values for the various volumes as illustrated by the following generic equations:

$$\Delta V = f(p_i, V_{old})\,\Delta t \qquad (3\text{-}2)$$

$$V_{new} = V_{old} + \Delta V \qquad (3\text{-}3)$$

This procedure continues until one of the termination options, such as the calculational end time, stops the solution.

3.2.2 Iterative Numerical Solution of NUREG/CR-6224 Head Loss Correlation

The iterative numerical solution of the NUREG/CR-6224 head loss correlation is self-contained in

subroutine HLOSS which is called from only one location in subroutine POOL. All parameters required by the solution are passed through the arguments of the call statement, i.e., subroutine HLOSS does not have access to the data storage. These arguments are listed in Table 3-3.

A rather simple and straightforward convergence technique, illustrated in Figure 3-5, was used to solve the numerical solution. Basically, there are two equations which must be solved simultaneously with a bed compaction limit imposed on the solution (see Section 2.7 for a description of the model and Section 4.3.3 for the verification of this solution). Both equations are set up as head loss gradients that are a function of other variables, i.e., the incremental change in the head loss per incremental change in theoretical fiber bed thickness. Both equations contain a term referred to here as the compaction ratio, which is defined as the ratio of the actual fiber bed thickness to the theoretical fiber bed thickness. The solution technique provides a guess for the compaction ratio and then calculates the head gradients. When the correct compaction ratio is guessed, both equations will return the same head gradients. In HLOSS, the solution has converged when the two head gradients agree within an absolute relative error of less than the convergence criteria of 5×10^{-5}. Except for the first cycle, the guessed values for the compaction ratio are simply the midpoint between the two bounding values which are adjusted at the end each cycle.

The initial lower bound is set at the compaction ratio corresponding to a solidly packed debris bed, i.e., the bed can not be realistically compressed tighter than this limit. The minimum compaction ratio is a function of rubble densities of the fibrous and particulate debris, the particulate-to-fiber mass ratio, and the thickness of theoretical fibrous debris on the strainer.

The initial guess for the first cycle in the iterative solution is the initial lower bound. This first cycle is used to determine whether or not the bed will be pack limited or not. If the head gradient for the NUREG/CR-6224 is larger than the compressibility function on the first cycle, then the bed is pack limited and the solution is simply this limiting compaction ratio. If this condition is not met, then the solution is determined by the convergence of the two equations.

Table 3-3: Input/Output Arguments for Subroutine HLOSS

No.	Variable Description
Input	
1	Pipe Weld Break Number
2	Strainer Number
3	Unit Number of ERROR.OUT
4	Strainer Flow Velocity
5	Water Density
6	Water Viscosity
7	Theoretical Fiber Bed Thickness
8	Particulate-to-Fiber Mass Ratio
9	Rubble Density of Fibrous Debris Mixture
10	Rubble Density of Particulate Debris Mixture
11	Material Density of Fibrous Debris Mixture
12	Material Density of Particulate Debris Mixture
13	Specific Surface Area of Fibrous Debris Mixture
14	Specific Surface Area of Particulate Debris Mixture
15	User Specified Head Loss Multiplier
Output	
16	Actual Fiber Bed Thickness
17	Head Loss

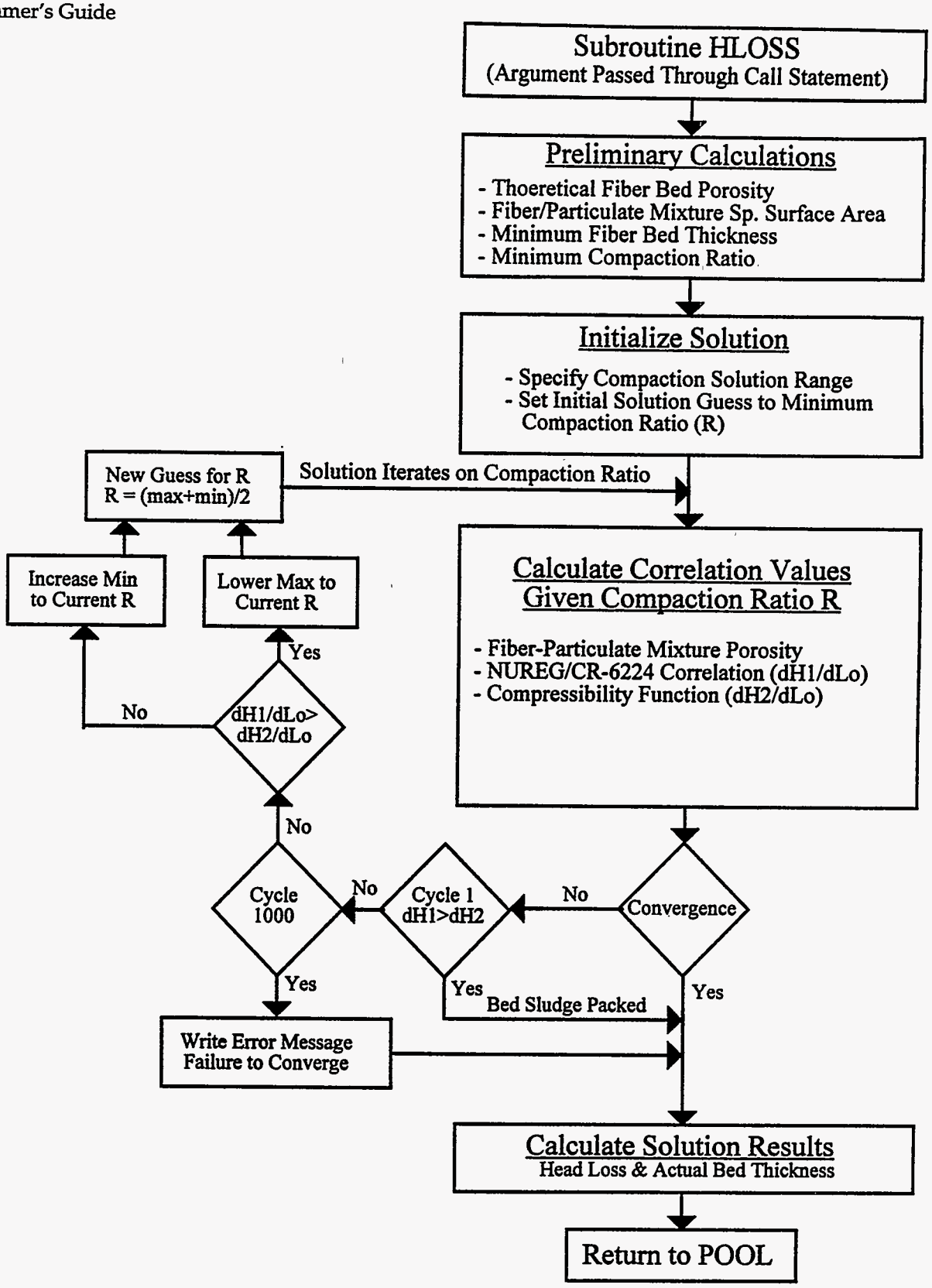


Figure 3-5: Solution Technique for NUREG/CR-6224 Head Loss Correlation

The maximum solution range value is set at 50 in HLOSS. Normally the fibrous debris compaction should be a number between 0 and 1 which corresponds to the theoretical bed thickness. However, during certain extreme conditions at the beginning of the calculation when only a small quantity of fibrous debris has been deposited, but the particulate-to-fiber mass ratio is large, compaction ratios larger than one can be calculated. This situation was not found to be a problem because the head losses are still small and the solution quickly becomes much more realistic as more fibrous debris is deposited. The value of 50 is a number that has been found to converge the solution under these extreme conditions without significantly increasing the computer time required for the solution to change.

If a calculation is attempted where the solution exists at a compaction ratio larger than 50, the solution will fail to converge resulting in an error message in ERROR.OUT. The solution fails to converge when the solution loop exceeds 1000 cycles, whereas normally, this solution is found to converge in under about 8 cycles. The input conditions creating a convergence failure would likely include a rather large strainer area and a relatively slow flow velocity and fail only in the first couple of calculational time steps. Convergence failure is not anticipated to be a common problem in the use of the code.

At the end of each solution cycle, one of the range values is adjusted prior to proceeding onto the next cycle. If the head gradient of the NUREG/CR-6224 correlation were greater than the gradient for the compressibility function, then the maximum bound is lowered to the compaction ratio of the last cycle. Conversely, if the gradient for the compressibility function is greater than that for the NUREG/CR-6224 correlation, the minimum bound is raised to the value of the compaction ratio of the last cycle. Since the distance between the lower and upper bound is reduced by one half after each cycle, convergence is achieved rather quickly.

The subroutine HLOSS is bypassed whenever the theoretical fiber bed or the flow velocity are zero. In this situation, the predicted head loss is zero and actual fiber bed thickness is the same as the theoretical bed thickness.

3.3 Time Step Control

The time step size is controlled by a time-dependent table that is provided as input by the user. At the beginning of each time cycle, a value for the time step is obtained from this table using the interpolation option also specified by the user. The time step controller may then impose additional constraints on the time step. Specifically, the controller checks to determine if the normal time step causes the next time cycle to pass over one of the input parameter event time. When the controller makes this determination, the controller calculates a new time step that will make the next time cycle exactly match the event time that would have been passed over. Event times are the times provided by the user in the various time-dependent input tables including the time step table. This logic causes the code to execute a time cycle at the exact times when these input parameters might make a major change. The event times include the end-of-blowdown time, the end-of-washdown time, and the calculational end time.

The coding for the time step controller is located in a block of coding immediately following the initiation of the time loop. User guidance in specifying the time step size is found in Section 4.2 of the User's Manual [NRC, NUREG/CR-6370] and a time step sensitivity study is documented in Section 4.3.1.5 of this Reference Manual.

3.4 Integration of Time-Dependent Transport Rates

The drywell transport model requires the user to specify a time-dependent drywell transport rate that must integrate to one over the blowdown time period and again over the washdown time period. This rate is provided as input in the tabular form, which is then interpolated using the user specified interpolation option. A further constraint is that the integration must use the same exact time steps that will be used during the debris transport calculation to ensure that the correct quantity of debris is actually transported. The input processor tests this input for validity prior to performing the transport calculation and if the input is found invalid, the calculation immediately terminates with an error message that indicates the problem.

The input processor in subroutine POOLIN calls subroutine INTEG, to perform the integration. Subroutine INTEG is called twice, once for large LOCAs and once for medium LOCAs. The call statements pass an integer to INTEG that indicates whether the integration is for a large or a medium LOCA and INTEG passes back to POOLIN the integration results for the blowdown and the washdown periods. INTEG has access to data storage. The integration results are then validated by POOLIN to determine if the integrals are within 1± **cklim** (user supplied allowed error). Subroutine INTEG contains a replica of the time step controller that performs the debris transport calculations in subroutine POOL. The integration process is shown by the following equation:

$$I_{period} = \sum_{i=1}^{t_{period}} G_i(t,m)\,\Delta t_i(t,m) \qquad (3\text{-}4)$$

where:

I_{period} = either the blowdown or the washdown period integration result

t_{period} = indicates summation over blowdown or washdown period

$G_i(t,m)$ = the drywell transport rate which is a function of the time and LOCA scenario

$\Delta t_i(t,m)$ = the time step size which is a function of the time and the LOCA scenario.

Additional information regarding this integration is found in the model description discussed in Section 2.3 of this Reference Manual and in the user guidance provided in Section 4.3 of the User's Manual.

3.5 Time and Temperature-Dependent Interpolation

BLOCKAGE 2.5 has a subroutine called SELECT that can interpolate tabular data using either a step, linear, or an exponential interpolation option. All data passed to and from subroutine SELECT is passed through the call statement arguments, i.e., the subroutine does not have access to the data storage. The arguments are:

- an index indicating the LOCA scenario
- an index indicating the user specified interpolation option
- the time (x)
- the number of entries in the data table
- the tabular event times (X)
- the tabular parameter values (Y)
- the interpolation result (y)

The interpolation equations are shown below:

Step Interpolation

$$y(x) = Y_i \qquad (3\text{-}5)$$

Linear Interpolation

$$y(x) = Y_i + \left(\frac{x - X_i}{X_{i+1} - X_i}\right)(Y_{i+1} - Y_i) \qquad (3\text{-}6)$$

Exponential Interpolation

$$y(x) = \exp\left[\ln Y_i + \left(\frac{x - X_i}{X_{i+1} - X_i}\right)(\ln Y_{i+1} - \ln Y_i)\right] \qquad (3\text{-}7)$$

where:

i indicates the position in the table such that $x \geq X_i$ but $x < X_{i+1}$

The tables are not interpolated beyond their last entry, i.e., the value returned by SELECT is simply the last entry in the table. Since it is mathematically impossible to take the logarithm of zero, coding was implemented in BLOCKAGE to prevent a zero entry in the exponential interpolation equation.

3.6 Input Processing

The input processor performs several types of input checks to validate input. The type of checking performed depends upon the type of data being validated. These input checks include the following:

- Numerical real numbers are tested to determine if their value falls within a range of values. This test was designed to prevent the user from inputting an obviously invalid number, such as negative number when only

a positive number is possible. These ranges of allowed values are in some cases quite large to allow the user a lot of latitude.

- Integer values are usually tested to determine if the value belongs to a set of valid values, such as the indicator for the probabilistic reports where the only allowed values are zero and one.
- Settling velocity groups are summed to determine if the distributions are valid. These distributions must sum to one.
- The time-dependent drywell debris rates must sum to 1 within the allow error limit.
- Certain names are required to be unique. For example, two welds are not allowed to have the same name, thereby preventing the user from entering the same weld twice.
- Certain input parameters must be compatible with other parameters.

The BLOCKAGE 2.5 processor is not sophisticated enough to detect when a line of input is missing or when the input file contains an extra line of input. If a line of data were missing, the next line will be read by the wrong READ statement, thus causing a mismatch between the input data and the code variables. If this happens, the calculation will almost certainly abort. However, the resulting error message will likely not indicate the true nature of the problem and some detective work by the user will be required to determine the root cause of the problem.

4.0 BLOCKAGE Code Verification and Validation

The BLOCKAGE code has been subjected to rigorous coding verification to ensure that the code performs as it was designed to perform. In general, extensive quality assurance (QA) was integrated into the development of the BLOCKAGE code.

4.1 Overview of Code Quality Assurance

A high level of QA was applied to the development of the BLOCKAGE code from the beginning of the development. The NRC software quality assurance guidelines [NRC, NUREG/BR-0167] were followed where applicable and where practical. Configuration management procedures were implemented in which each level of code development resulted in a code version designated as Version X.X. Each version has been archived and is retrievable.

Methods of insuring quality have included line-by-line reviews of coding, calculating results by hand, and verifying the results with analytical solutions. When code changes were not intended to alter calculational results, the before and after results were compared with the aid of computer file comparisons to locate numbers that might have changed.

The probabilistic results consisting of tables of weld break and strainer blockage frequencies correlated by pipe diameter, piping system, and piping location were verified by hand computing each number in the tables. Other models such as the calculation of insulation debris volumes were also calculated by hand. These hand calculations were done again whenever the applicable coding was altered.

The numerical time-dependent solution in BLOCKAGE has been verified by reproducing the results of a relatively complex analytical test problem using the BLOCKAGE code. The analytical test problem solved the system of differential equations inherent in the numerical solution of the BLOCKAGE code using a debris source term to the suppression pool that was mathematically continuous and applying initial conditions that rendered the differential equations linear. The BLOCKAGE code input has sufficient flexibility to run this test problem without any coding alterations. The analytical solution was accomplished using the symbolic mathematical software marketed as MAPLE V [MAPLE, 1992]. The BLOCKAGE code solutions were compared with the MAPLE V solutions by plotting the solutions together and were found to be virtually identical. Thus, the same relatively complex solution was arrived at from two totally independent means.

The NUREG/CR-6224 head loss correlation in the BLOCKAGE code involves a set of non-linear equations which can not be solved explicitly. The equations describe the strainer head loss, the fibrous debris cake compressibility, and the debris cake particulate packing limitation. These equations in BLOCKAGE were solved using a numerical approach and the solutions were verified using a graphical technique. The equations were programmed in a graphical worksheet created using the MathCad software [MathCad, 1995]. Using MathCad, the head loss and compressibility equations and the packing limitation were plotted together. The solution was then viewed graphically and compared to the BLOCKAGE code result. The convergence of the numerical solution within the BLOCKAGE code has been refined sufficiently that it is unlikely that a failure to converge will occur in future use for any realistic problem; however, if the numerical iterative solution fails to converge, an error message is printed to alert the user.

4.2 Code Development QA History

QA procedures were applied to the development of the BLOCKAGE code to verify that the coding performed as designed and intended. The QA procedures applied along with the features added to the code at each enhancement of the code is summarized in Table 4-1.

The original PC-based version of the code, BLOCKAGE 1.0, was designed to exactly reproduce the combined functions of two older main-frame computer codes, PRA and TABLE, which were used to perform strainer blockage frequency calculations for PWR plants for the resolution of USI A-43, "Containment Emergency Sump performance" [NRC, NUREG-0869]. BLOCKAGE 1.0 was able to reproduce the results published for the reference PWR plant [NRC, NUREG/CR-3394], thereby establishing a consistency between the past and present studies. The code was then further modified to accommodate special characteristics of a representative BWR resulting in BLOCKAGE 2.0 and strainer blockage probabilities were calculated with the BLOCKAGE 2.0 code [Zigler, 1994].

The BLOCKAGE code was then further modified to incorporate a suppression pool model capable of

providing time dependent behavior for settling and resuspension of insulation debris within the pool and the subsequent deposition of the debris on a strainer. This initial suppression pool model and several other model enhancements and coding improvements and verifications resulted in BLOCKAGE 2.4, which was used to provide analytical support to the BWR strainer blockage study reported in NUREG/CR-6224 [NRC, NUREG/CR-6224].

The latest code development resulting in BLOCKAGE 2.5 represents a major revision of the code. The enhancements added to BLOCKAGE 2.5 include the following:

- The capability to track multiple types of debris (up to 20 types) throughout each calculation (whereas previous versions only tracked 1 type of fibrous debris, 1 type of non-target generated particulate debris originating in the drywell, and 1 type of debris originating in the wetwell).
- The capability to model up to 8 independent pumping systems where each pumping system has a single strainer but up to 4 separate pumps on a common header (previous versions were all limited to one strainer and one pump).
- The capability of varying the suppression pool temperature as a function of time and calculating the NPSH margin for each pump as a function of the pool temperature.
- The NUREG/CR-6224 head loss correlation was supplemented by three other optional correlations, i.e., the BWROG correlation and two generic correlations that allow the users to devise and implement their own correlations.

4.3 Verification of Coding

The coding of the BLOCKAGE 2.5 code has been subjected to a variety of coding verification tests to provide extensive assurance that the coding will perform as it was designed to perform. The coding has passed each of these tests and the developers of this code have a high degree of confidence that the calculational results produced by this code are those described by the code models. Independent assessments of the coding were provided by reviewers other than the code developer.

Two coding verification tests were performed to provide verification of the overall code. These tests included:

- A benchmark of the BLOCKAGE code against a complex analytical test problem.
- The reproduction of the NUREG/CR-6224 results calculated with Version 2.4.

Four coding verification tests were performed to verify specific portions of the code. These tests included:

- A mathematical simulation of the NUREG/CR-6224 head loss correlation.
- A spreadsheet reproduction of the target generated debris volume calculations.
- A hand calculation of each number in the NUREG/CR-6224 probabilities report.
- Tests of alternate problem configurations.

The appropriateness of the limitations imposed on the input by the BLOCKAGE code input processor were reviewed. Verification that the limitations have been correctly coded has been done for most of the input variables by attempting to input incorrect values to determine if the code will prevent the incorrect values from being used. In addition, the output format was independently reviewed for appropriateness.

Many other small scale tests have been performed throughout the development of the code, such as a hand calculation for a specific equation or a specific block of coding. These tests were not be formally documented.

4.3.1 Verification of Numerical Solution Using Analytical Test Problem

An analytical test problem was developed to verify the time-dependent solution of the BLOCKAGE code. This test problem involved the time-dependent transport of several types of debris including fibrous, particulate, and metallic debris; a temperature-dependent NPSH margin for a single pump; and the head loss calculated using a generic correlation. This problem was designed to test as much of the coding in BLOCKAGE as reasonably possible and was not intended to represent realistic conditions.

Table 4-1: Code Development and Quality Assurance Historical Overview

Code Version Date Completed and Developers	Development Features	QA Summary
BLOCKAGE 1.0 December 1993 R. Walsh, N. Ruiz, and C. Shaffer	• Original coding designed to replicate functions of PRA/TABLE (PWR) codes.	• PRA/TABLE codes reproduced with one difference attributed to an error in PRA/TABLE. • Line-by-line review of code. • Hand calculation of results.
BLOCKAGE 2.0 January 1994 C. Shaffer	• Implemented a weld-type diameter-class weld break frequency model. • Destruction factor input by L/D region. • Transport fractions input by weld location. • Head loss correlation coefficients converted to user input.	• Line-by-line review of modified code. • Hand calculation of results.
BLOCKAGE 2.0a August 1994 D. V. Rao	• Initial BWR suppression pool model incorporated.	• Programmer verification of suppression pool model. • Peer verification of blockage results.
BLOCKAGE 2.1 February 1995 C. Shaffer	• Removal of obsolete coding designed to exactly reproduce results of PRA/TABLE codes. • General clean-up of code. • I/O enhancements. • Extensively comment coding. • Convert developmental hardwired input to code input.	• Line-by-line review of complete code. • Reproduced Version 2.0a results.
BLOCKAGE 2.2 March 1995 C. Shaffer	• Incorporated new strainer blockage correlations. • Added strainer filtration and system retention efficiencies. • Treating drywell particulate separate from wetwell sludge. • Temperature-dependent water properties. • Other supporting modification.	• Ensured BLOCKAGE coded with single set of units, English units. • Analytical test problem verification of time-dependent solution using MAPLE software. • New head loss correlations verified by graphic solution using MathCad software. • Intermediate results were hand calculated. • Line-by-line review of modified code.
BLOCKAGE 2.3 April 1995 C. Shaffer	• Implementation of screen welcome message and leading comment block to code. acknowledging NRC ownership and SEA's development efforts. • Added liability disclaimer. • Deleted obsolete debug coding.	• Calculational engine not altered. • Calculational results did not change as verified by performing before and after file comparisons. • Previous analytical tests applies to this version. • Previous hand calculated verifications still apply to this version. • Line-by-line review of modified code.

Table 4-1: Code Development and Quality Assurance Historical Overview (continued)

Code Version Date Completed and Developers	Development Features	QA Summary
BLOCKAGE 2.4 June 1995 C. Shaffer	• Several I/O coding modifications were requested prior to the development of the BLOCKAGE code User Interface. • A modification allowing user specification of debris volumes was implemented.	• Calculational engine not altered. • Changes to calculational results were traced to I/O modifications by performing before and after file comparisons. • Previous analytical tests applies to this version. • Previous hand calculated verifications still apply to this version. • Line-by-line review of modified code
BLOCKAGE 2.4a November 1995 C. Shaffer	• The numerical iterative blockage correlation solution was modified to ensure convergence under conditions involving minute quantities of fibrous debris deposited on the strainer at the beginning of the calculation.	• The calculation showing the failure of the solution to converge was successfully rerun. The failure had not affected the calculational results. • It was verified that this failure to converge had not affected any previous calculational results, in particular those results reported in NUREG/CR-6224.
BLOCKAGE 2.5x (Pre-Release Beta Version) May 1996 C. Shaffer	• Add the capability to track multiple types of debris. • Add the capability to model multiple independent pumping systems with multiple pumps arranged on a common header. • Add capability to vary suppression pool temperature with time. • Add capability to calculate pump NPSH margin as a function of suppression pool temperature. • Add BWROG and two generic head loss correlations as user options. • Extended strainer filtration efficiency model to accommodate differing filtration efficiencies for different materials and debris sizes. • Additional options for terminating calculations. • Input/Output revised to accommodate model revisions. • Code architecture restructured to accommodate these new and much more generalized models.	• Analytical test problem verification of time-dependent solution using MAPLE software. • Additional head loss correlations verified with hand calculated results. • Debris volume calculations verified with hand calculated results. • The results reported in NUREG/CR-6224 were reproduced. • The probability models were not changed, therefore previous hand calculated verification of model still valid. • The input processing was independently reviewed. • Test problems were devised and run to help ensure that the code would run problems with alternate configurations from the sample test problems used to debug the code, i.e., 1) a problem with only 1 debris type, 1 debris size, 1 pump, etc. 2) a problem using the maximums of 20 debris types, 20 debris sizes, 32 pumps, etc.

Table 4-1: Code Development and Quality Assurance Historical Overview (continued)

Code Version Date Completed and Developers	Development Features	QA Summary
BLOCKAGE 2.5 Final Version October 1996 C. Shaffer	• Code finalized for public release. • Addressed NRC comments on Beta version. • Fixed minor errors. • Clarified I/O terminology. • Added more input processing capability. • Removed unnecessary double precision coding. • Added new plot variables for NPSH margin and strainer approach velocities. • Upgraded code to Lahey FORTRAN 90 compiler.	• Code output files for several input models created both before and after a code modification were compared electronically and differences evaluated to ensure that errors were not introduced into the code. • For code modifications that affected results, the altered results were individually evaluated to ensure that the modification was implemented correctly and to determine the potential impact on previous analyses.

The analytical test problem solved the system of differential equations inherent in the numerical solution of the BLOCKAGE 2.5 code using a debris source term to the suppression pool that was mathematically continuous, with initial conditions applied that rendered the differential equations linear. The BLOCKAGE code input has sufficient flexibility to run this test problem without any coding alterations. The analytical solution was accomplished using the symbolic mathematical software marketed as MAPLE V [MAPLE, 1992]. The BLOCKAGE code solutions were compared with the MAPLE V solutions by plotting the solutions together and were found to be virtually identical. Thus, the same relatively complex solution was arrived at using two totally independent means.

The independence of this verification test was accomplished by an independent review of the test results by an engineer that was not directly associated with the development of the code because the analytical test was actually performed by the principal developer of the code as a final step in the code debugging process. The independent reviewer agreed that the comparison of the test problem results as calculated with BLOCKAGE and with MAPLE were in excellent agreement and that the applicable coding of BLOCKAGE was verified.

4.3.1.1 Description of Test Problem

The analytical test problem was designed with as much complexity as was reasonable possible to solve analytically. The solution of the analytical test problem required that all the differential equations be both linear and mathematically continuous. Specifically, this required 1) the suppression pool resuspension and the settling turbulence factors; the strainer filtration and system retention efficiencies, and the pump flow be constant values rather than variable as allowed by BLOCKAGE; and 2) the time-dependent pool temperature and drywell transport rate be specified as a mathematically continuous function rather than the tabular data allowed by BLOCKAGE.

The values of constant parameters not dependent upon the type of debris are provided in Table 4-2, and the values used to specify debris attributes are presented in Table 4-3. The analytical test problem included 7 types of debris with varied attributes. In addition, the target generated fibrous debris type had three velocity groups and the wetwell sludge groups had two velocity groups which provided verification to velocity group dependent coding. The same filtration efficiency was used for both the initial and the peak input values so that the filtration efficiency remained constant throughout the calculation.

Table 4-2: Values of Constant Parameters

Parameter	Value
Suppression Pool Resuspension Factor	0.0005 sec^{-1}
Suppression Pool Settling Turbulence Factor	0.25
Pump Flow	10000 GPM
Initial Pool Temperature	80 °F
Initial NPSH Margin	38 ft-water
Screen Area	40 ft^2
Volume of Pool	58900 ft^3
Cross-Sectional Area of Pool	5000 ft^2

The pool temperature was specified as a linear temperature increase with time from an initial temperature of 80 °F to a temperature of 180°F at 20,000 seconds. The calculation was terminated at 20,000 seconds. Since the pump NPSH margin in BLOCKAGE was modeled as function of the pool temperature, the NPSH margin in the analytical test problem also varied with time.

The drywell transport rate was specified using an exponential function that decreases with time t, such that the function integrated to nearly one between 0 and 20,000 seconds. The function was

$$G(t) = 0.001\, e^{-t/1000} \tag{4-1}$$

and the integrated value was:

$$\int_0^{20000} 0.001\, e^{-t/1000}\, dt = 0.9999999979 \tag{4-2}$$

The first generic correlation implemented in BLOCKAGE was used to calculate a strainer head loss in the analytical test problem. The NUREG/CR-6224 correlation was not used because that correlation must be solved using a numerical technique. The generic correlation used in this test problem was devised for the purpose of testing coding and does not represent any experimental data or other correlation. The metallic head loss term was included to test the metallic head loss coding. The correlation as used in this problem is given in Equation 4-3:

$$\Delta H = 3.1\left[0.9\,\Delta L_o^{1.8}\,(0.2+1.5\,\eta)^{0.8}\,U + 1.1\,\Delta L_o^{1.9}\,(0.2+0.9\,\eta)^{0.77}\,U^2 + 5\,S_v\,\frac{1-\varepsilon_m}{\varepsilon_m}\,\rho_w\,U^2\right] \tag{4-3}$$

where:

- ΔH = the strainer head loss, ft-water
- U = the strainer flow velocity, ft/sec
- ΔL_o = the uncompressed fibrous cake thickness on the strainer, ft
- η = the ratio of particulate mass to fibrous mass on the strainer
- S_v = the specific surface area of the metallic debris, ft^2/ft^3
- ϵ_m = the porosity of the metallic debris (= 1 - ρ_{rub}/ρ_{mat}),
- ρ_{rub} = the rubble density of the metallic debris, lbm/ft^3
- ρ_{mat} = = the material density of the metallic debris, lbm/ft^3
- ρ_w = the density of the water, lbm/ft^3.

Table 4-3: Attributes of Analytical Test Problem Debris

Debris Type	#	Origin of Debris[1]	Class[2]	Volume (ft^3)	Transport Fraction	Settling Velocity (fps)	Filt. Eff.[3]	Retent. Eff.[4]	Fabricated Density (lbm/ft^3)	Rubble Density (lbm/ft^3)	Material Density (lbm/ft^3)	Surface Area (ft^2/ft^3)
Fiberglass	1	TG	F	12.5	0.4	0.002	0.5	0.9	2.4	2.4	175.	170,000
	2	TG	F	62.5	0.4	0.010	0.8	0.3	2.4	2.4	175.	170,000
	3	TG	F	50.	0.4	0.040	0.9	0.6	2.4	2.4	175.	170,000
Cal. Silicate	1	TG	P	2.	0.7	0.040	0.6	0.5	90.	20.	110.	20,000
RMI	1	DW	M	26.	0.2	0.08	0.9	1.0	55.	105.	491.	2400
Paint	1	DW	P	3.	0.6	0.005	0.5	0.7	180.	45.	180.	50,000
WW-Fiber	1	WW	F	5.	-	0.001	0.8	0.7	5.1	7.0	275.	90,000
Sludge	1	WW	P	1.56	-	0.005	0.4	0.6	324.	65.	324.	120,000
	2	WW	P	1.04	-	0.03	0.8	0.9	324.	65.	324.	120,000
Junk	1	WW	I	5.	-	0.5	1.	1.	300.	95.	491.	900

[1] Origin of debris, i.e., TG for drywell targets, DW for non-target debris in drywell, and WW for debris originating in wetwell (see Section 2.1).
[2] Classification of debris, i.e., F for fibrous, M for metallic, P for particulate, and I for ignore (see Section 2.1).
[3] Strainer filtration efficiency.
[4] Primary system retention efficiency.

where:

NPSH	=	the time-dependent NPSH Margin, ft-water
$NPSH_o$	=	the NPSH at the initial pool temperature of 80 °F, ft-water
P_{vap}	=	the vapor pressure of water in psia at the time-dependent water temperature, T_w
P_{vapo}	=	the vapor pressure of water in psia at the initial pool temperature of 80°F
P_a	=	the atmospheric pressure of 14.7 psia
ρ_w	=	the density of water in lbm/ft^3 at the time-dependent water temperature, T_w
ρ_{wo}	=	the density of water in lbm/ft^3 at the initial pool temperature of 80°F.

The strainer head loss was analytically calculated using the generic correlation devised for this problem and finally the head loss was subtracted from the NPSH margin to show the potential of strainer blockage. This problem was terminated just prior to strainer blockage.

The graphical results of the analytical solution are shown in Section 4.3.1.4 where they are compared with the results from the BLOCKAGE solution. Many components of the analytical solution were algebraically complex, in particular the solution for the head loss. The solution for the fibrous debris cake thickness on the strainer, a moderately complex analytical solution, is shown in Equation 4-10 as an example:

4.3.1.3 BLOCKAGE Code Solution of Test Problem

The analytical test problem was solved by BLOCKAGE 2.5 without any coding modifications. All of the problem specifications were designed to match the code input capabilities. The time-dependent drywell transport function was implemented into the BLOCKAGE input model using the code's exponential interpolation option by simply specifying the value of the function at time zero and again at the end time of 20,000 seconds. Similarly the time-dependent water temperature was implemented using the linear interpolation option and specifying 80 °F at time zero and 180 °F at 20,000 seconds. The user specified volume option was used to enter the target volumes for fiberglass and calcium silicate. Once the input was entered, a simple execution of BLOCKAGE provided the results.

The primary results for the coding verification study were the time-dependent plots which could be directly compared with the time-dependent results of the analytical solution. For a calculation involving a single strainer, the BLOCKAGE code provides the user with plot data for 29 output variables. However, several additional internal variables were also plotted to make the verification study more complete. The data for these other variables were obtained by implementing coding which created another plot output file containing data for the additional variables. This additional output file did not affect the calculational engine of the code or affect the calculation in any way. (This debug level coding is not supported or available for the general user.)

$$
\begin{aligned}
THKf = {} & 14.32130905 + .5104279062e^{(-.0010000000001799t)} \\
& -.4038008850e^{(-.00060462353979972t)} \\
& -.8961007054e^{(-.00029715453202035t)} \\
& +9.068438724e^{(-.00099999999880125t)} \\
& -9.164033190e^{(-.00084504569903761t)} \\
& -6.881149722e^{(-.00019246956316114t)} \\
& +1.579627950e^{(-.0015984267986812t)} \\
& -1.925260946e^{(-.0010000000005656t)} \\
& -5.279367004e^{(-.00011358534075324t)} \\
& -3.396954682e^{(-.0004383866028t)} \sinh^{(-.0001200308963t)} \\
& -.9300911856e^{(-.0004383866028t)} \cosh^{(-.0001200308963t)}.
\end{aligned}
\qquad (4\text{-}10)
$$

A sampling of the results of the analytical test problem are illustrated in the Figures 4-1 through 4-4. Figure 4-1 shows the fibrous debris from all sources that was suspended in the suppression pool, settled onto the wetwell floor, deposited onto the strainer, retained within the primary system, and the total debris that was transported from the drywell. A total of 50 ft^3 was transported from the drywell and 5 ft^3 was initially located on the floor of the suppression pool. The volume of fiber suspended in the pool and settled onto the wetwell floor increased as fibrous debris was transported from the drywell and then decreased as the transport process completed. Since the problem assumed a constant resuspension factor, the debris on the wetwell floor was slowly resuspended into the pool where it ultimately was either deposited onto the strainer or retained within the primary system. This assumption, of course, was not realistic. Figure 4-2 shows similar transport results for the particulate debris where a total of 3.2 ft^3 of calcium silicate and paint chips particulate were transported from the drywell and 2.6 ft^3 of sludge originated in the wetwell.

The particulate-to-fiber and the metallic-to-fiber mass ratios for debris deposited onto the strainer are shown in Figure 4-3. Head loss correlations for a strainer cake consisting of fibers and particulate are generally dependent upon the ratio of the two masses. These mass ratios illustrate that difference types of debris deposit onto the strainer at different rates. Further, since each type of debris may have a different density, the effective density of the mixture for each class of debris can vary with time.

The time-dependent NPSH margin and strainer head loss are shown in Figure 4-4. The NPSH margin decreases with time due to the increasing temperature of the pool. The head loss increases with time due to debris being deposited onto the strainer. Pump loss of NPSH margin would be predicted to occur when the head loss exceeds the NPSH margin. The difference between the NPSH margin and the head loss is also shown. This difference for the analytical test problem was 0.31 ft-water at 20000 seconds.

This sampling of results provides an overview of the analytical test problem. The next step in the coding verification process was to compare the analytical solution and the BLOCKAGE code solution, to ensure that the solutions were the same.

4.3.1.4 Comparison of BLOCKAGE Code and Analytical Solutions

Thirty BLOCKAGE code variables were independently verified by plotting the analytical solution of that variable with the BLOCKAGE code solution. The following output variables were checked:

No.	Variable
1	Volume of Fibrous Debris Suspended in the Pool
2	Volume of Fibrous Debris Settled onto the Wetwell Floor
3	Volume of Fibrous Debris Deposited onto the Strainer
4	Volume of Fibrous Debris Retained by the Primary System
5	Volume of Fibrous Debris Transported from the Drywell
6	Volume of Particulate Debris Suspended in the Pool
7	Volume of Particulate Debris Settled onto Wetwell Floor
8	Volume of Particulate Debris Deposited onto the Strainer
9	Volume of Particulate Debris Retained by the Primary System
10	Volume of Particulate Debris Transported from the Drywell
11	Volume of Metallic Debris Suspended in the Pool
12	Volume of Metallic Debris Settled onto the Wetwell Floor
13	Volume of Metallic Debris Deposited onto the Strainer
14	Volume of Metallic Debris Retained by the Primary System
15	Volume of Metallic Debris Transported from the Drywell
16	Mass of Fibrous Debris Deposited onto the Strainer
17	Mass of Particulate Debris Deposited onto the Strainer
18	Mass of Metallic Debris Deposited onto the Strainer
19	Particulate to Fibrous Debris Mass Ratio on the Strainer
20	Metallic to Fibrous Debris Mass Ratio on the Strainer
21	Effective Material Density of Fibrous Mixture on the Strainer

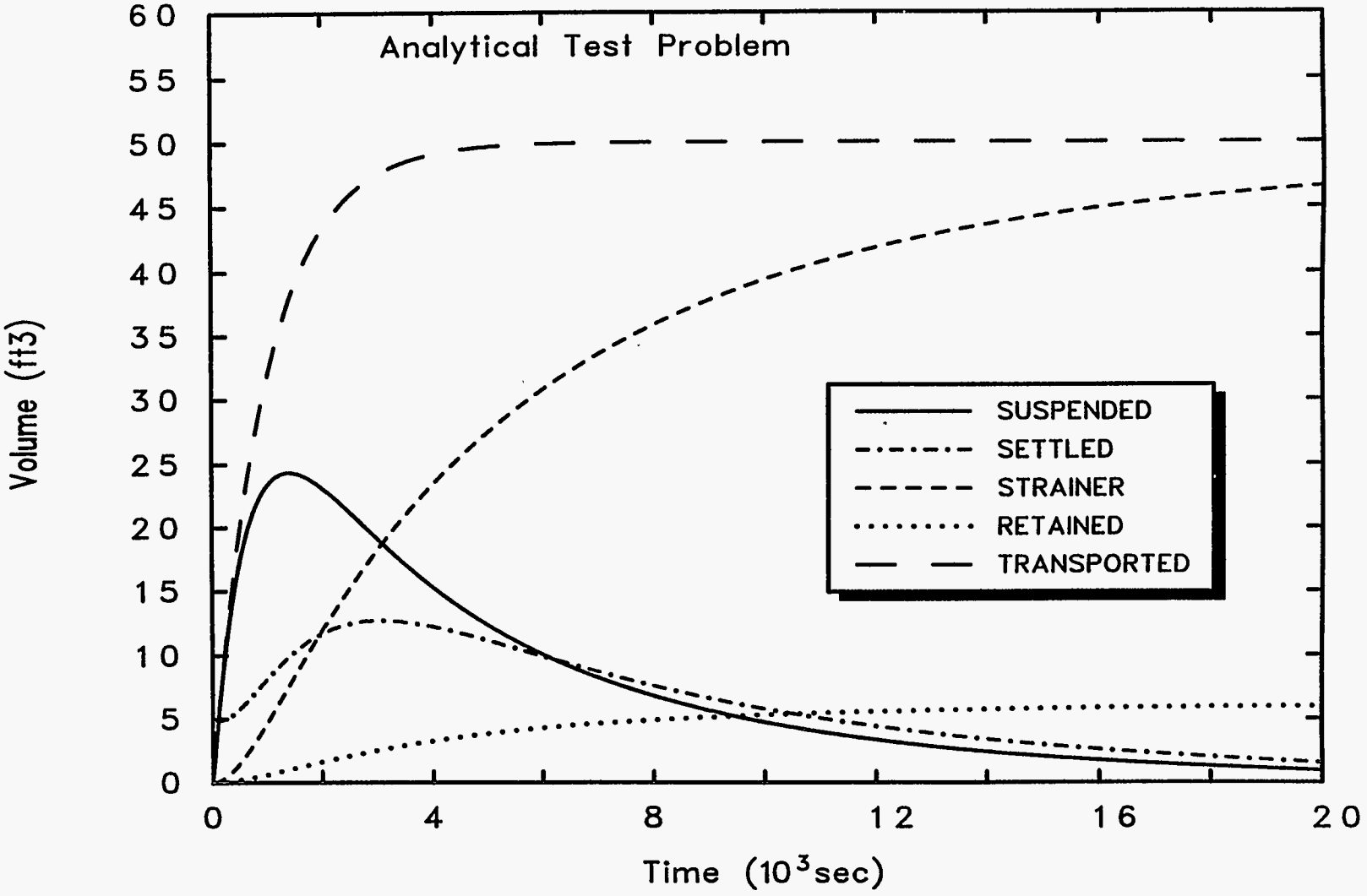


Figure 4-1: Fibrous Debris Transport

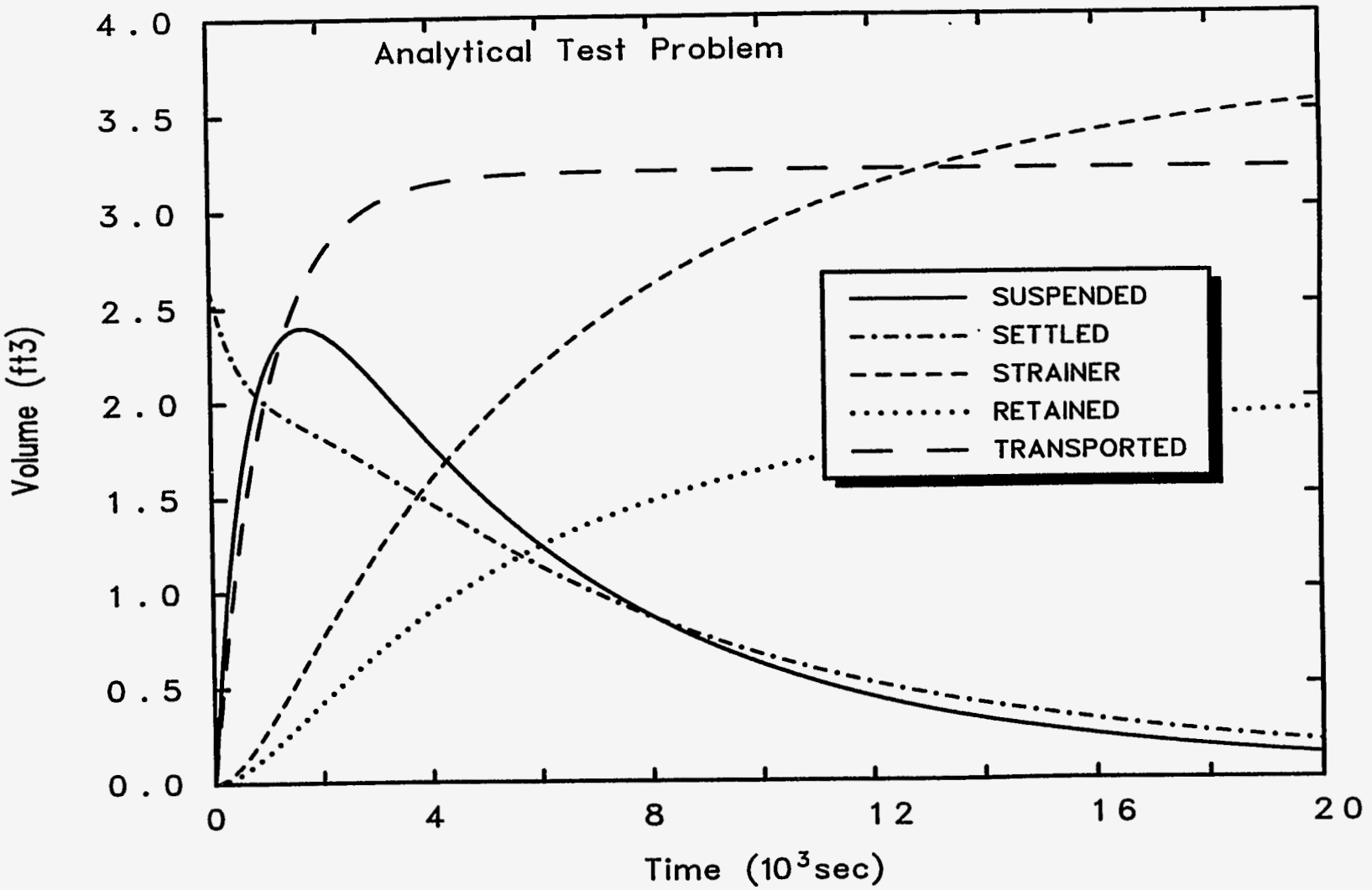


Figure 4-2: Particulate Debris Transport

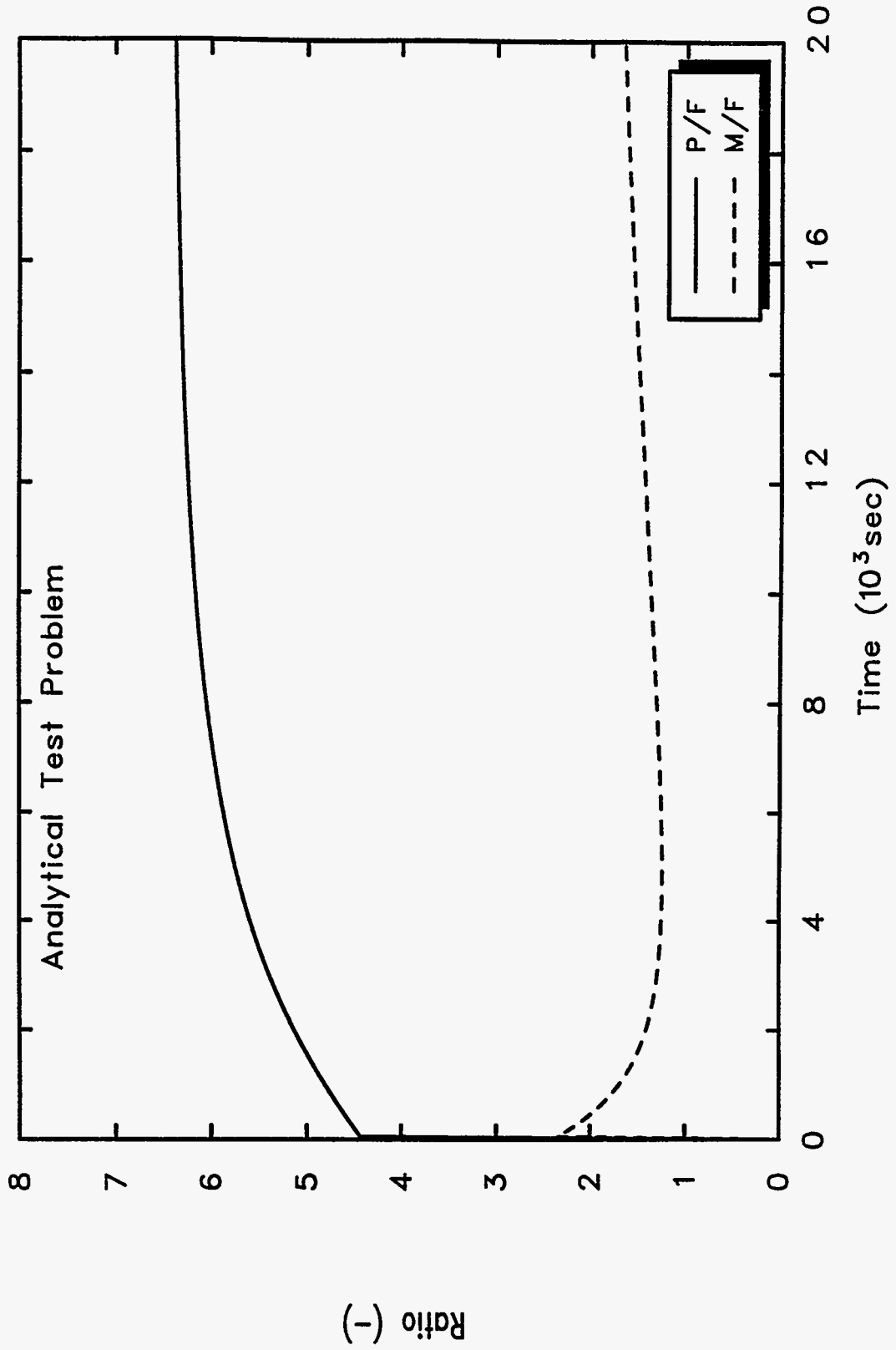


Figure 4-3: Strainer Mass Ratios

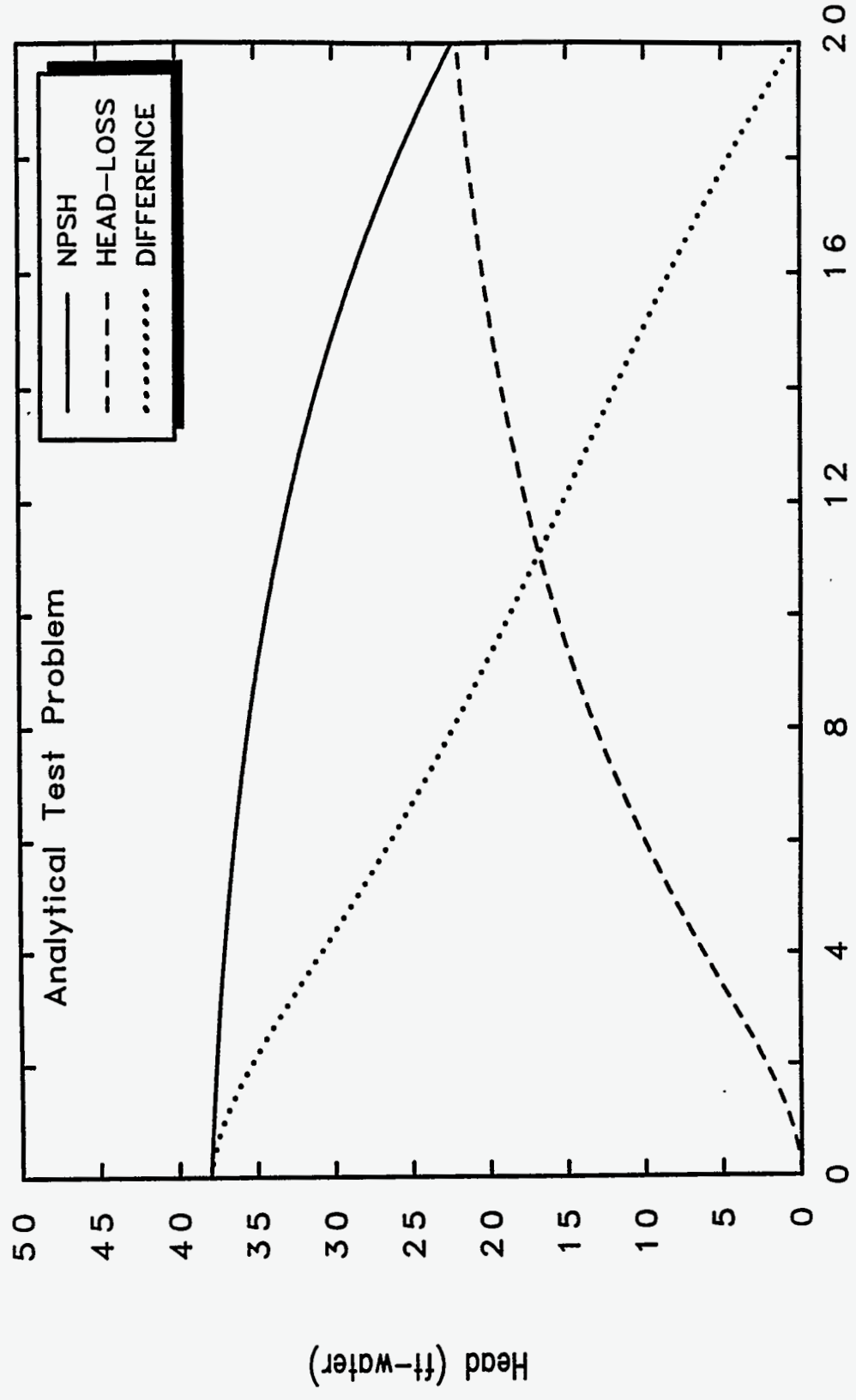


Figure 4-4: NPSH and Strainer Head Loss

No.	Variable
22	Effective Material Density of Particulate Mixture on the Strainer
23	Effective Rubble Density of Particulate Mixture on the Strainer
24	Effective Specific Surface Area of Fibrous Mixture on the Strainer
25	Effective Specific Surface Area of Particulate Mixture on the Strainer
26	Strainer Cake Thickness of Fibrous Debris
27	Strainer Cake Thickness of Metallic Debris
28	Net Pump Suction Head Margin
29	Strainer Head Loss
30	Difference Between NPSH Margin and Strainer Head Loss.

The analytical and BLOCKAGE solutions for each of these variables are compared in Figures 4-5 through 4-34, respectively. The two solutions can not actually be identical since the BLOCKAGE code must use a finite time step size as opposed to the infinitely small time increment that is inherent in any analytical solution of differential equations. The BLOCKAGE code solution was obtained using a time step of 0.5 seconds. There is no discernible differences between the two solutions for the 30 variables examined, as is clearly shown in these figures. The sensitivity of the time step size is examined in Section 4.3.1.5.

The time-dependent solution of the BLOCKAGE code has been verified to a very high level. Areas of coding not verified by this analytical test problem were verified independently. These independent coding verifications are discussed in the following sections.

4.3.1.5 Time Step Sensitivity

The sensitivity of the analytical test problem results to the size of the time step was determined as another verification of the BLOCKAGE coding and to provide guidance to the users of the code in the selection of a valid time step. Only the time step size was varied in this study and the time step sizes selected were: 0.5, 2.5, 10, 30, 60, and 120 seconds. All of the preceding base case results used a time step size of 0.5 seconds. The results of these six BLOCKAGE calculations and the analytical solution were plotted for eight output variables. The output variables were:

No.	Variable
1	Volume of Fibrous Debris Transported from the Drywell
2	Volume of Fibrous Debris Suspended in the Pool
3	Volume of Particulate Debris Suspended in the Pool
4	Volume of Metallic Debris Suspended in the Pool
5	Particulate to Fibrous Debris Mass Ratio on the Strainer
6	Metallic to Fibrous Debris Mass Ratio on the Strainer
7	Strainer Cake Thickness of Fibrous Debris
8	Strainer Head Loss.

These comparisons are shown in Figures 4-35 through 4-42 for the eight variables lists above, respectively. Sections of these figures were enlarged to allow a detailed examination of the study comparisons. These enlarged sections are shown in Figures 4-43 through 4-50, respectively. These figures clearly show that the BLOCKAGE code results converged towards the exact analytical solution as the time step size was reduced. This is another strong indication of the quality of the coding in BLOCKAGE, i.e., if the results had diverged rather than converged as the time step was decreased, it would have been a strong indication that something was wrong in the coding.

The results of the study indicated that a time step size of 1 second is more than adequate for most aspects of BLOCKAGE calculations. Past blockage analytical studies have used 10 seconds for the later portion of the calculation, after about 2500 seconds, where the debris transport is not very transient or strainer blockage has already occurred. Some previous studies have used a small time step of 0.1 seconds through the period of strainer blockage in order to better capture the predicted time for pump loss of NPSH margin, i.e., at one numerical time the pump has not lost its NPSH margin, but at the next numerical time it has lost its NPSH margin, therefore the smaller that time interval, the more closely the user will know the predicted time at which NPSH margin was lost.

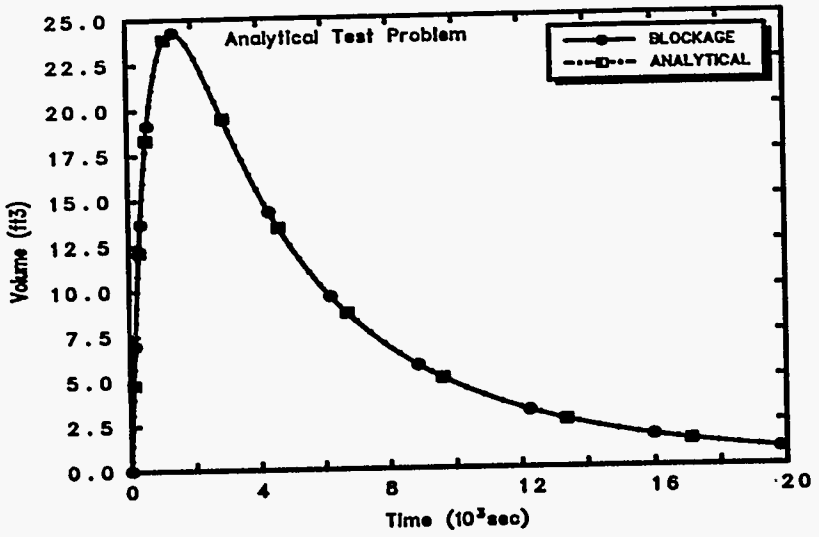


Figure 4–5: Fiber Suspended in Pool

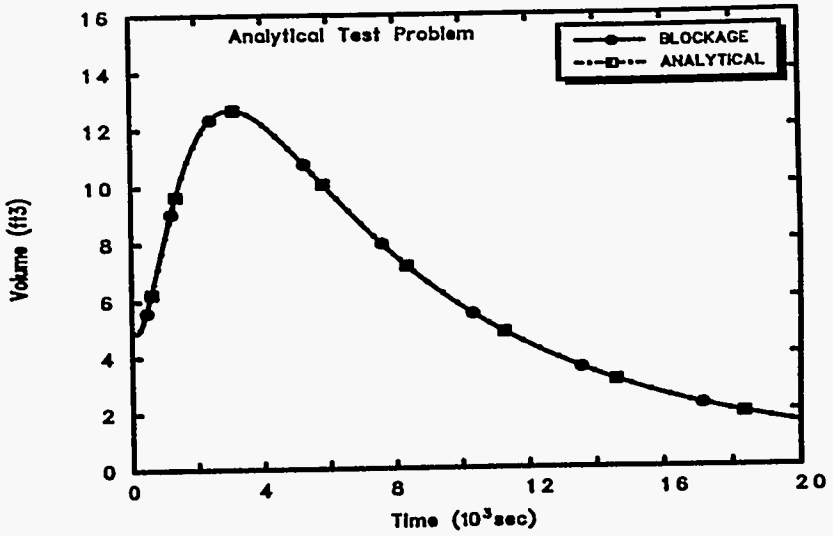


Figure 4–6: Fiber Deposited onto WW Floor

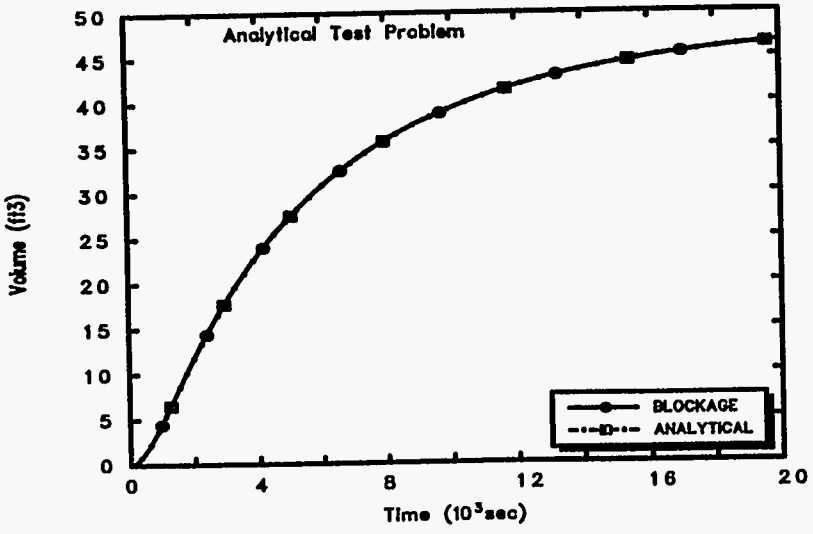


Figure 4–7: Fiber Deposited onto Srainer

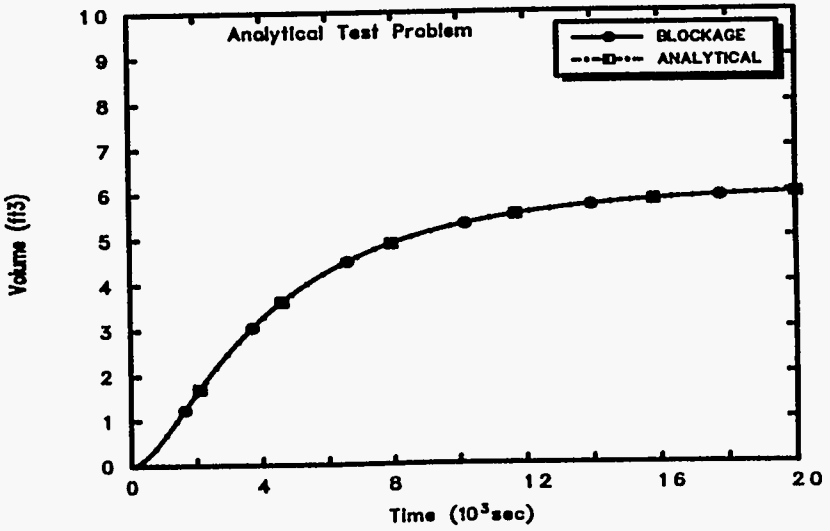


Figure 4–8: Fiber Retained by Primary System

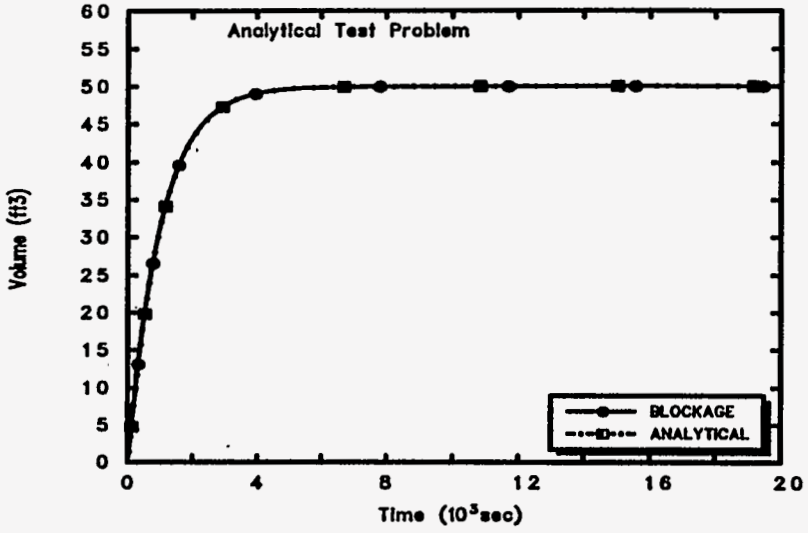


Figure 4–9: Total Fiber Transported from Drywell

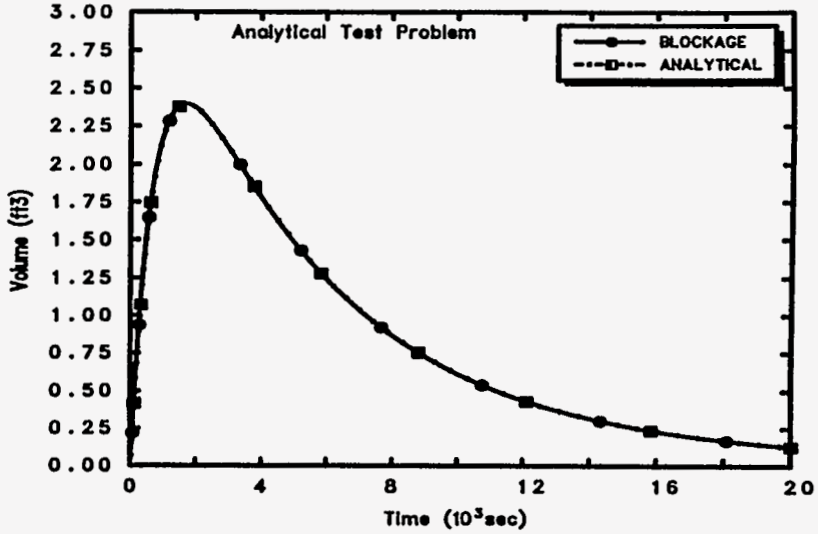


Figure 4–10: Particulate Suspended in Pool

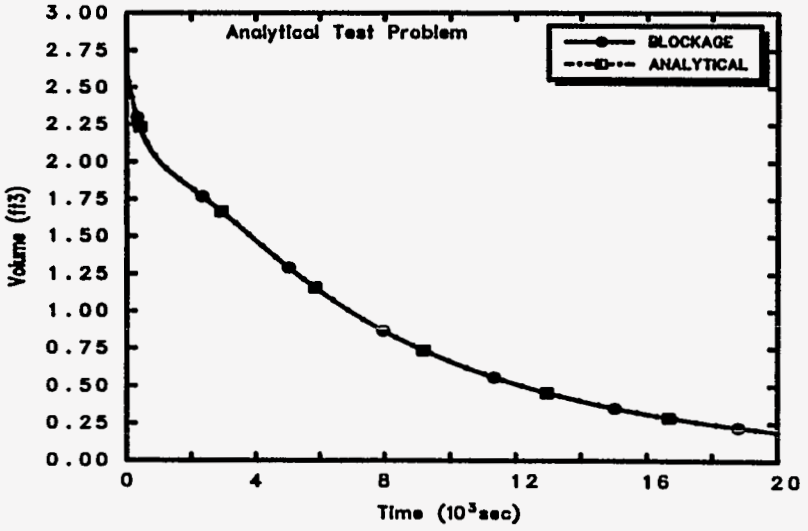


Figure 4–11: Particulate Deposited onto WW Floor

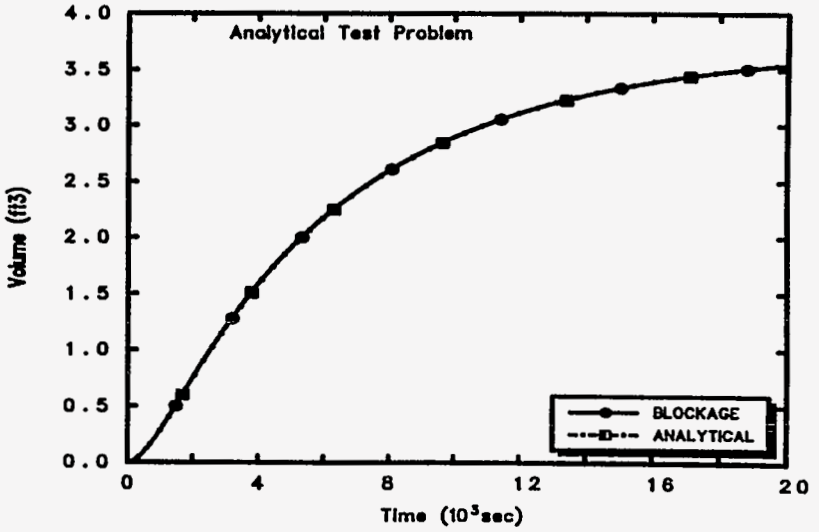


Figure 4–12: Particulate Deposited onto Strainer

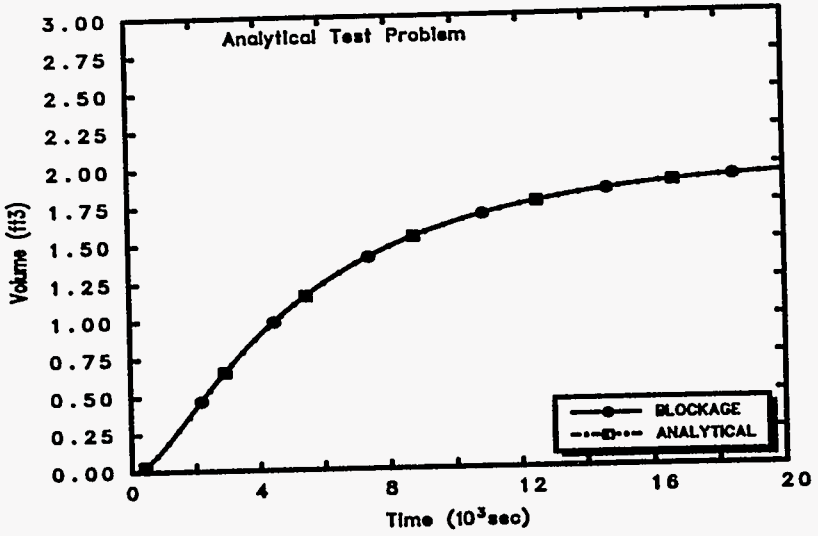


Figure 4–13: Particulate Retained by System

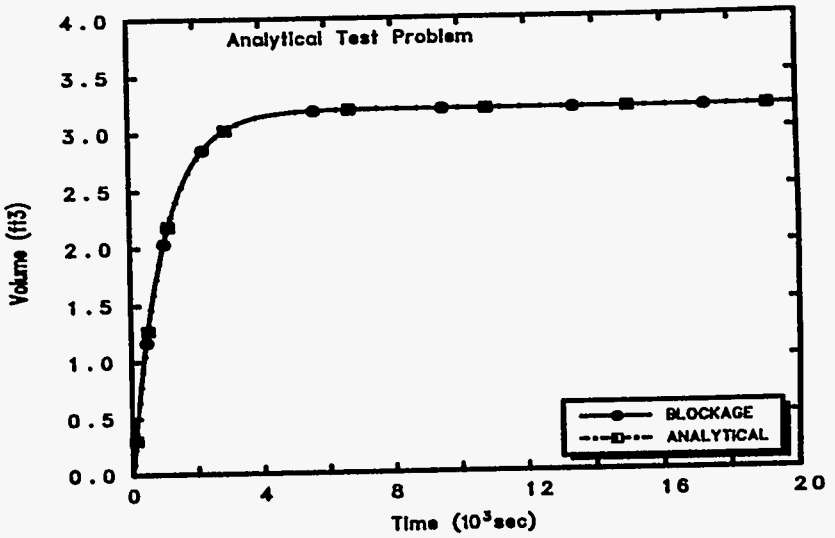


Figure 4–14: Total Particulate Transported from DW

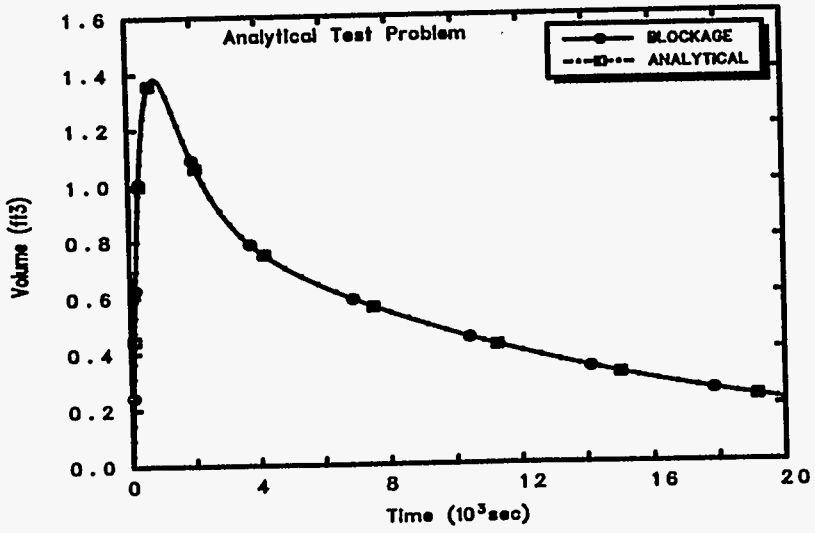


Figure 4–15: Metals Suspended in Pool

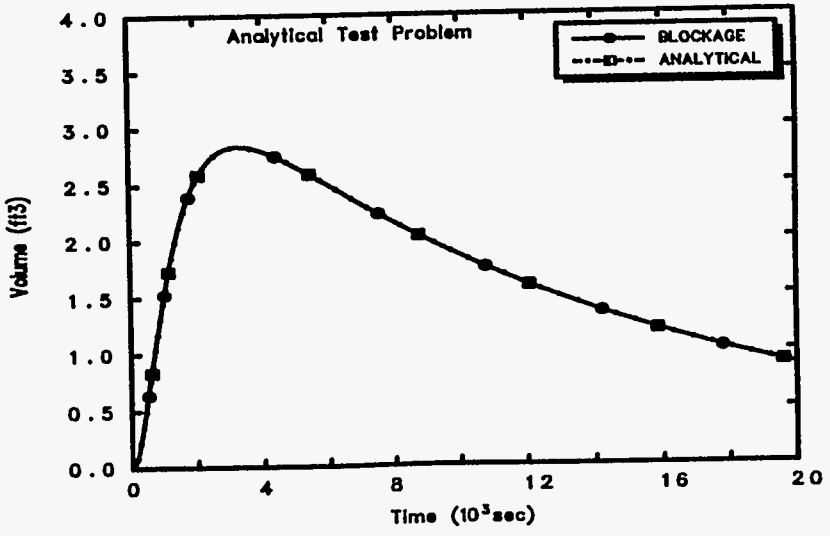


Figure 4–16: Metals Deposited onto WW Floor

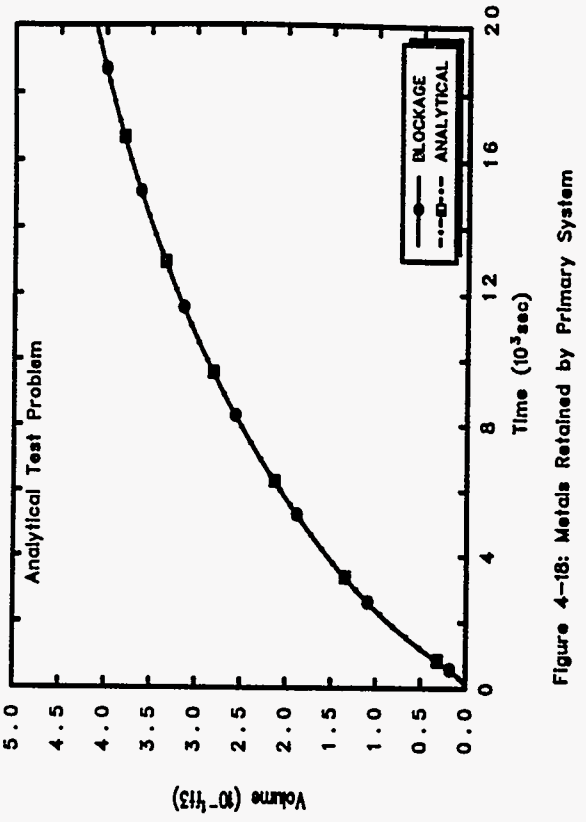


Figure 4-18: Metals Retained by Primary System

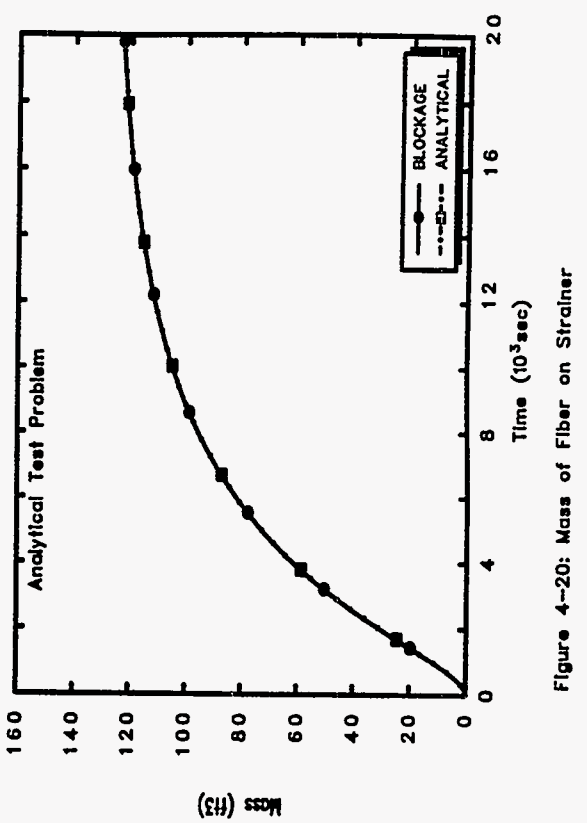


Figure 4-20: Mass of Fiber on Strainer

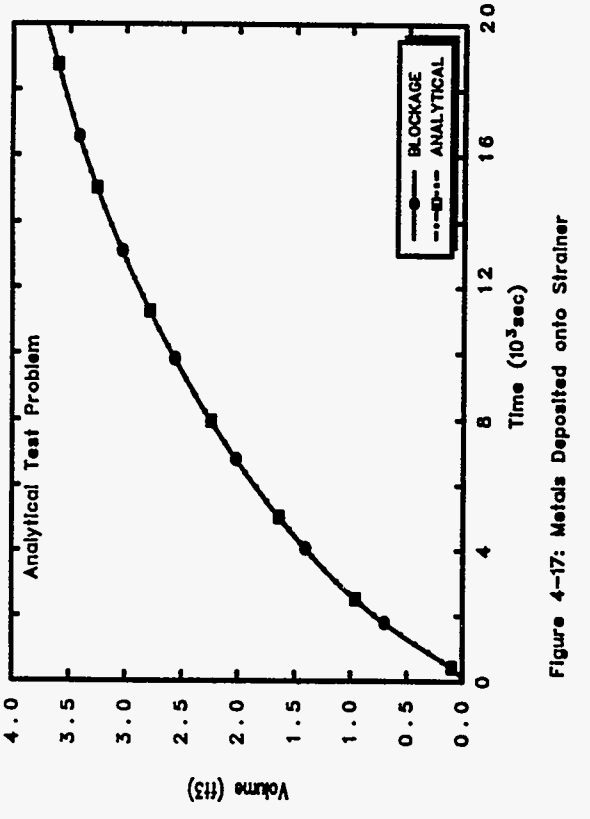


Figure 4-17: Metals Deposited onto Strainer

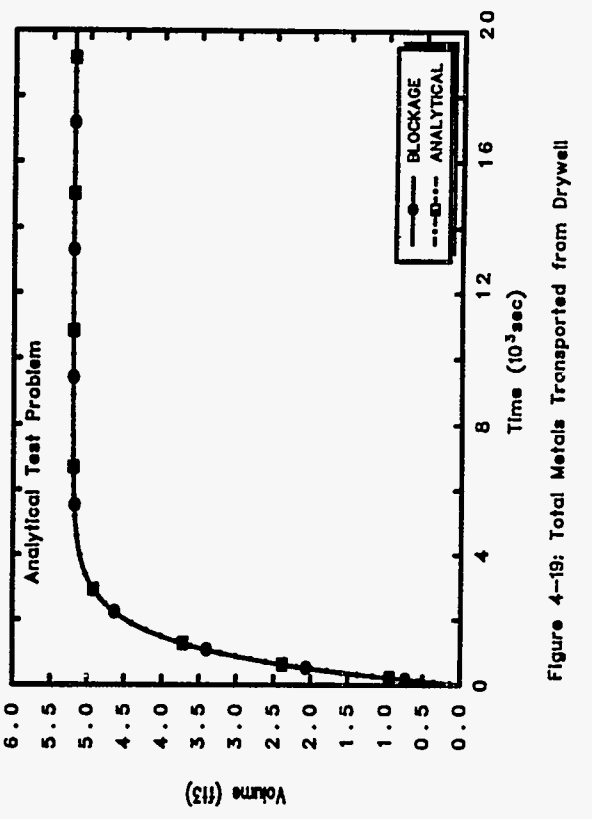


Figure 4-19: Total Metals Transported from Drywell

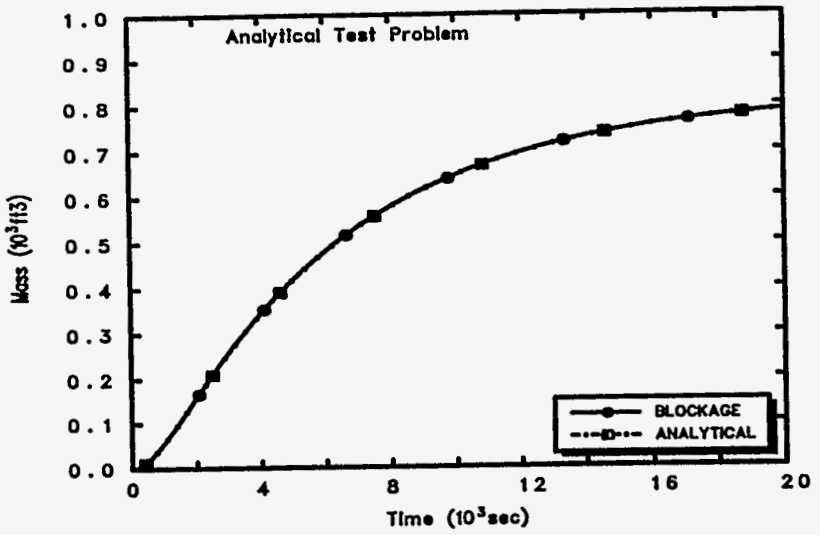


Figure 4–21: Mass of Particulate on Strainer

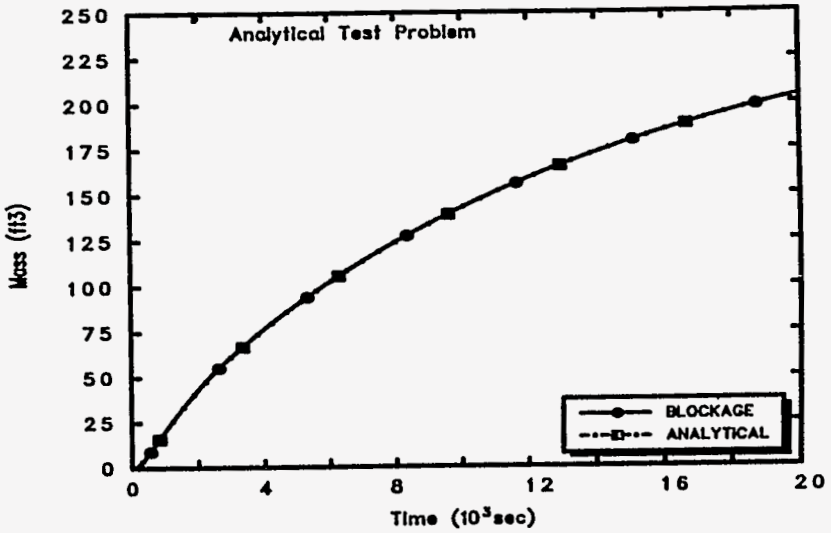


Figure 4–22: Mass of Metals on Strainer

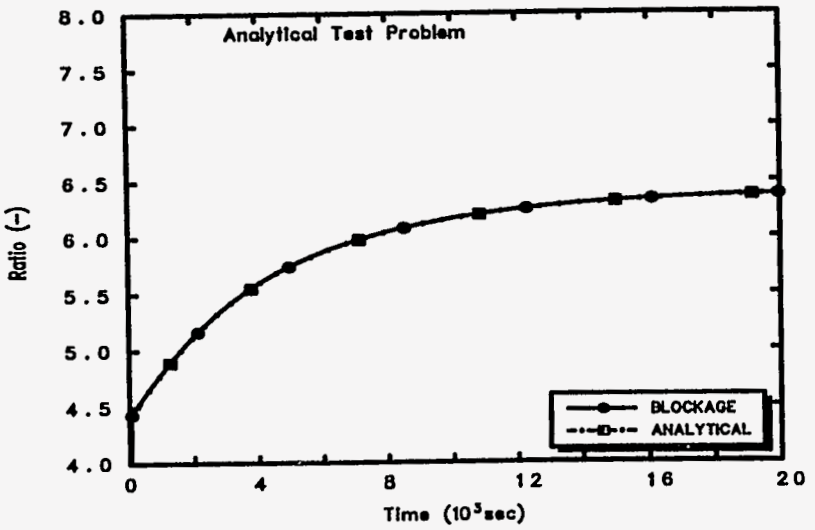


Figure 4–23: Particulate to Fiber Mass Ratio

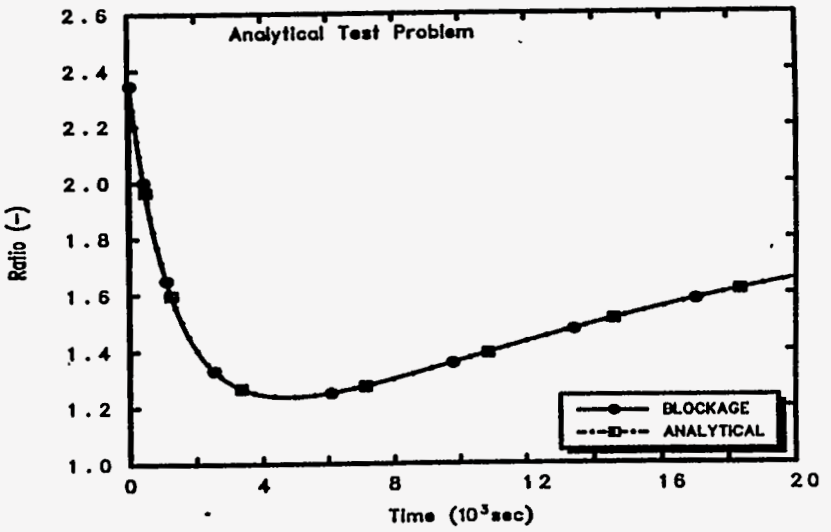


Figure 4–24: Metals to Fiber Mass Ratio

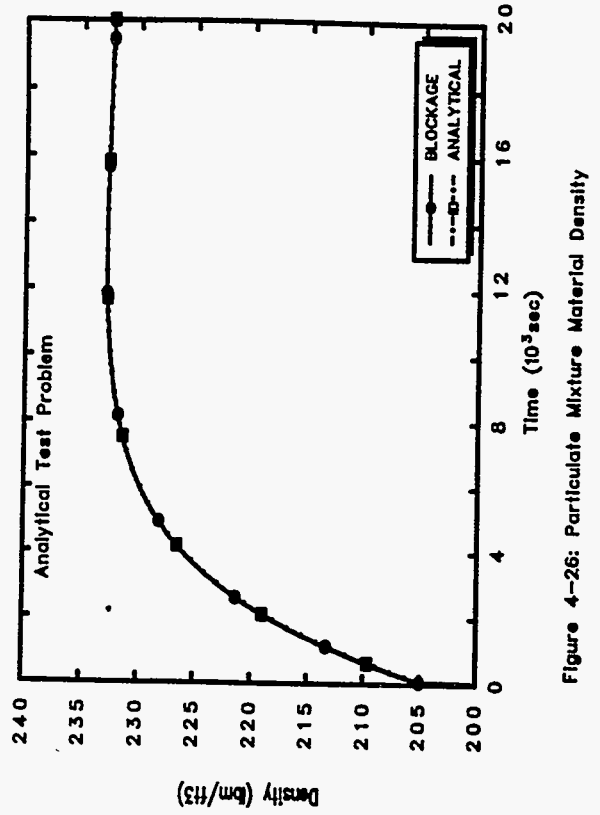


Figure 4-26: Particulate Mixture Material Density

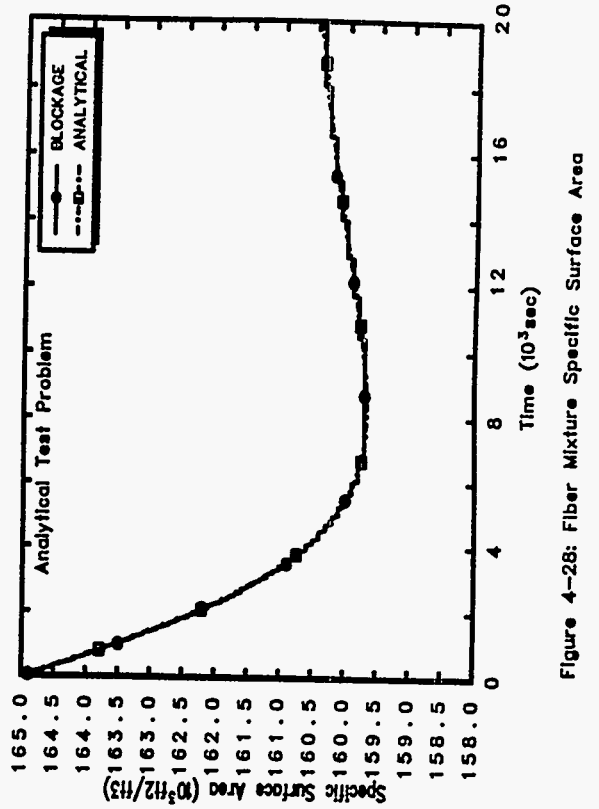


Figure 4-28: Fiber Mixture Specific Surface Area

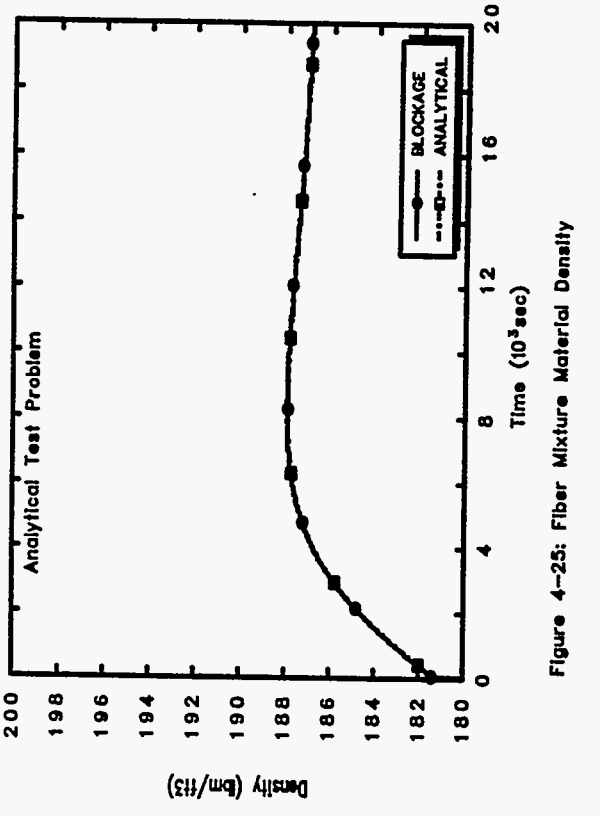


Figure 4-25: Fiber Mixture Material Density

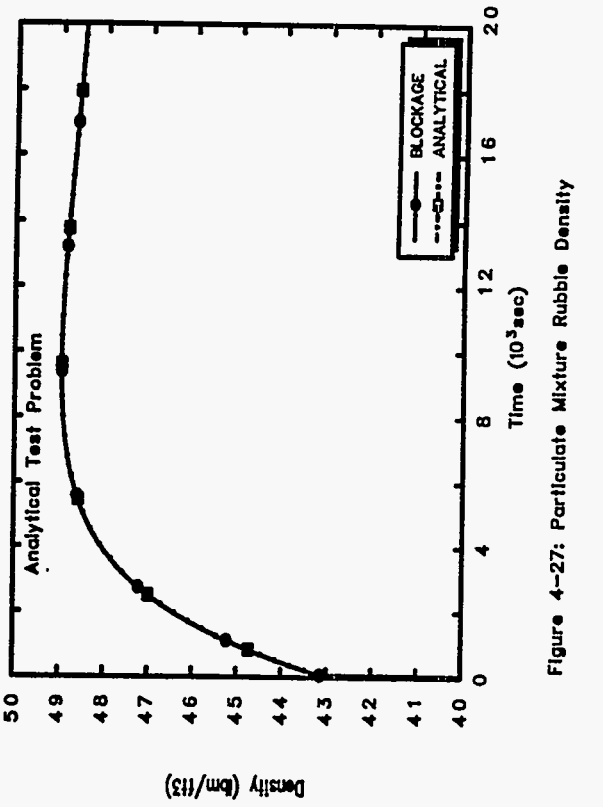


Figure 4-27: Particulate Mixture Rubble Density

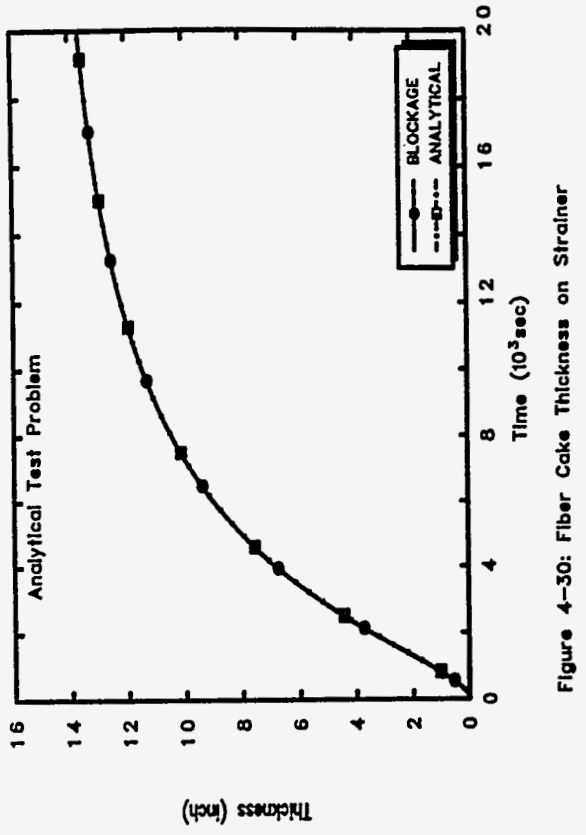


Figure 4-30: Fiber Cake Thickness on Strainer

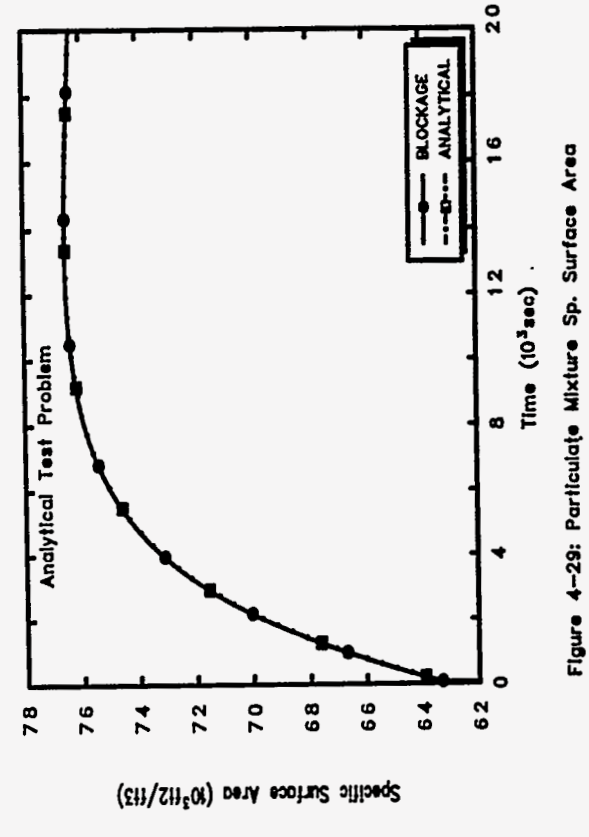


Figure 4-29: Particulate Mixture Sp. Surface Area

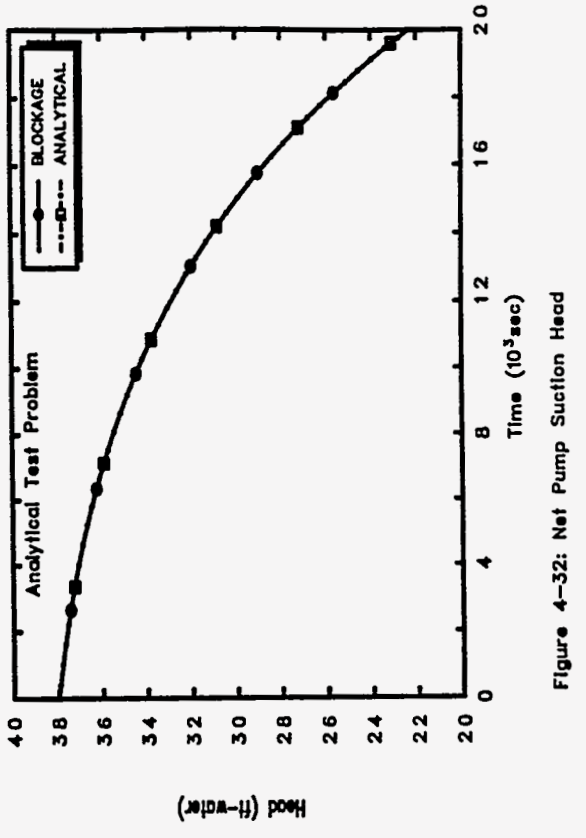


Figure 4-32: Net Pump Suction Head

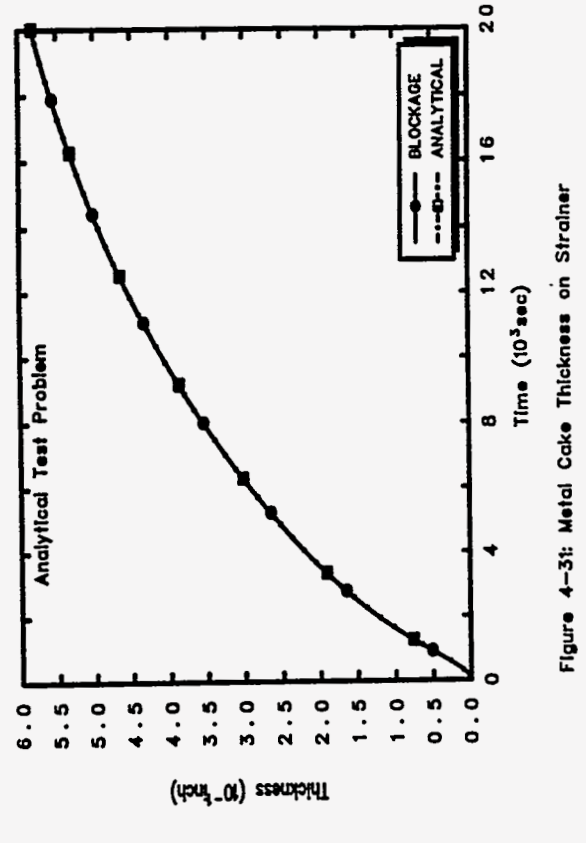


Figure 4-31: Metal Cake Thickness on Strainer

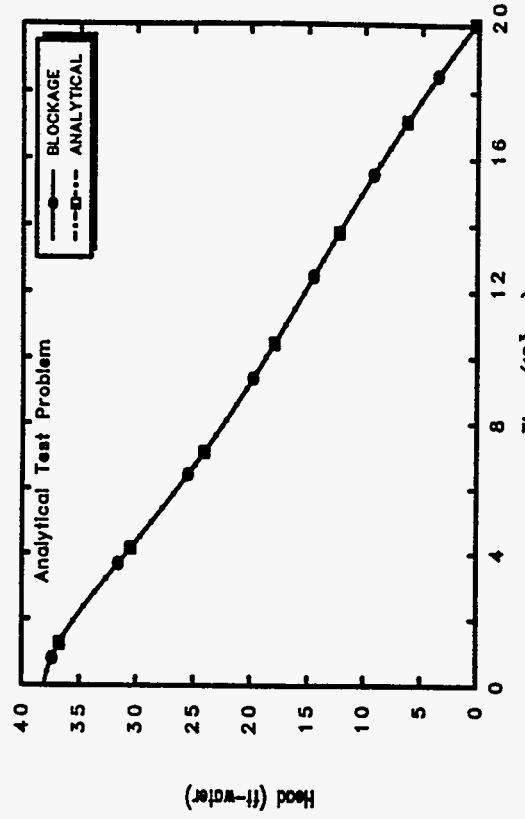


Figure 4-34: NPSH Margin Minus Head Loss

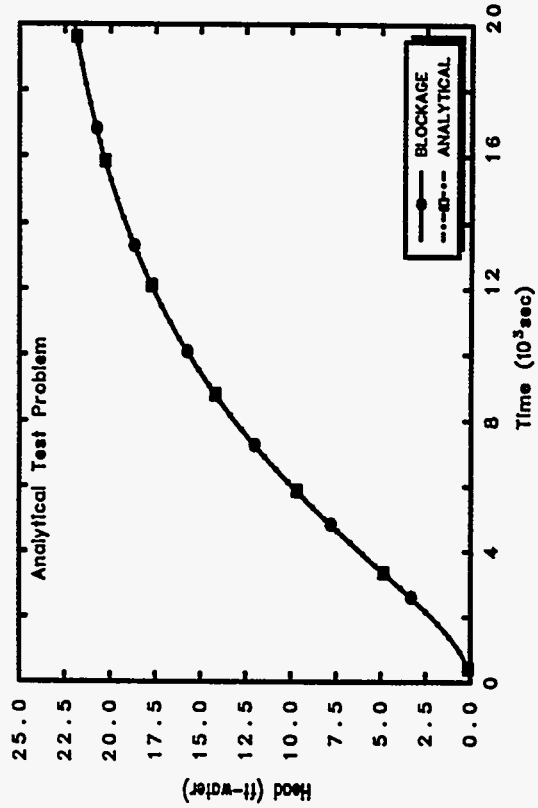


Figure 4-33: Strainer Head Loss

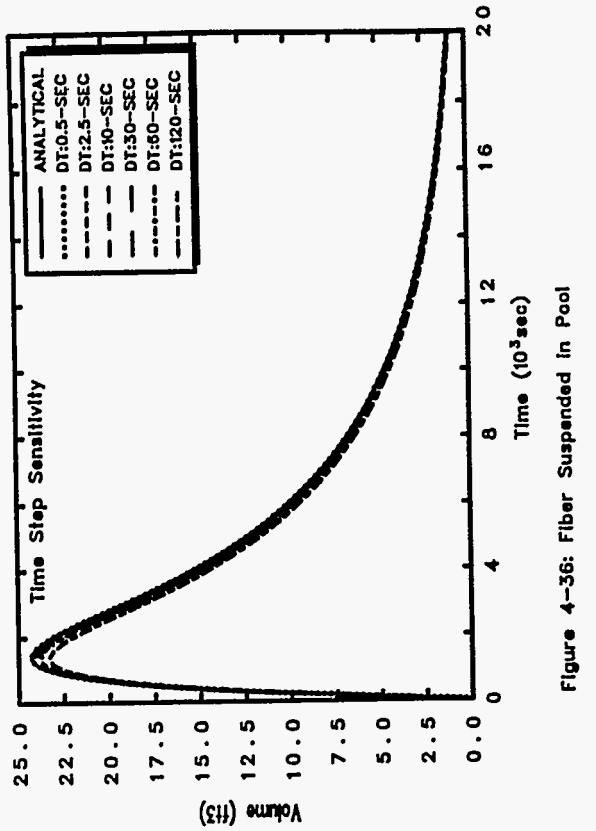


Figure 4-36: Fiber Suspended in Pool

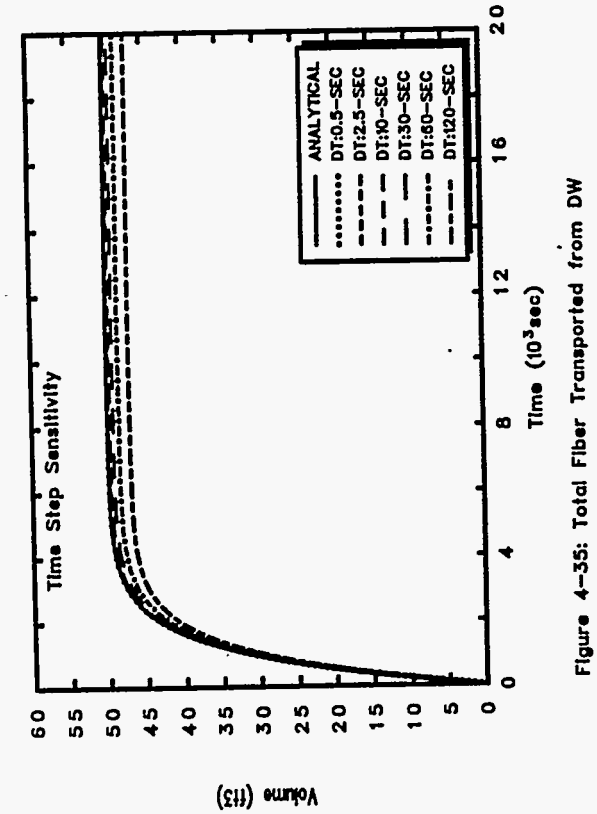


Figure 4-35: Total Fiber Transported from DW

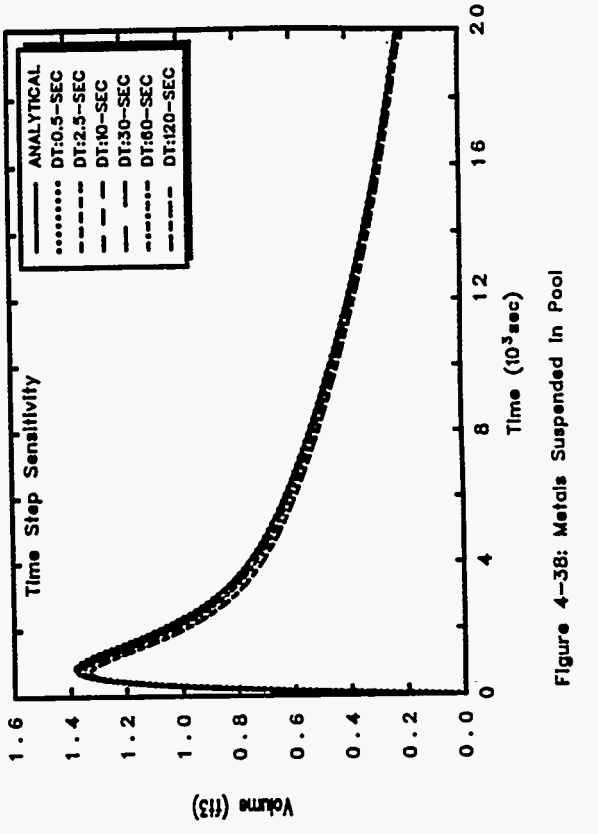


Figure 4-38: Metals Suspended in Pool

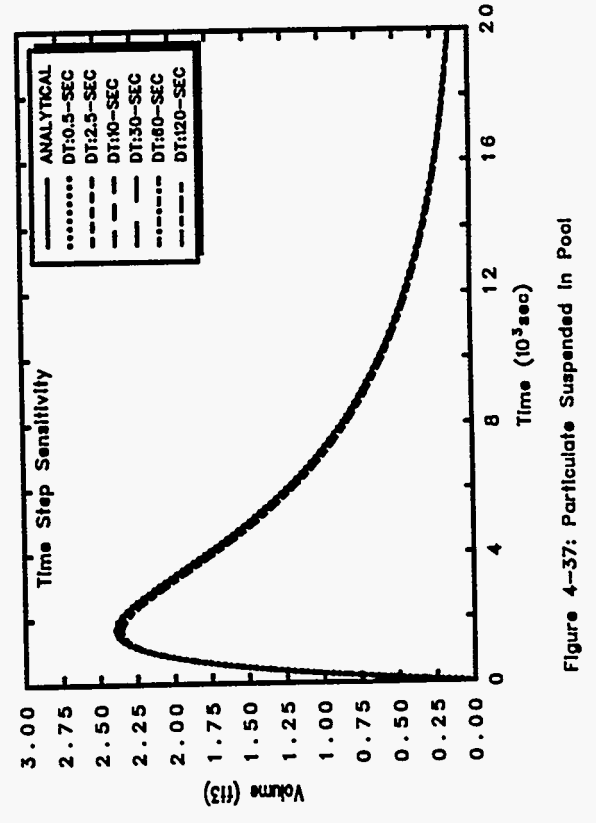


Figure 4-37: Particulate Suspended in Pool

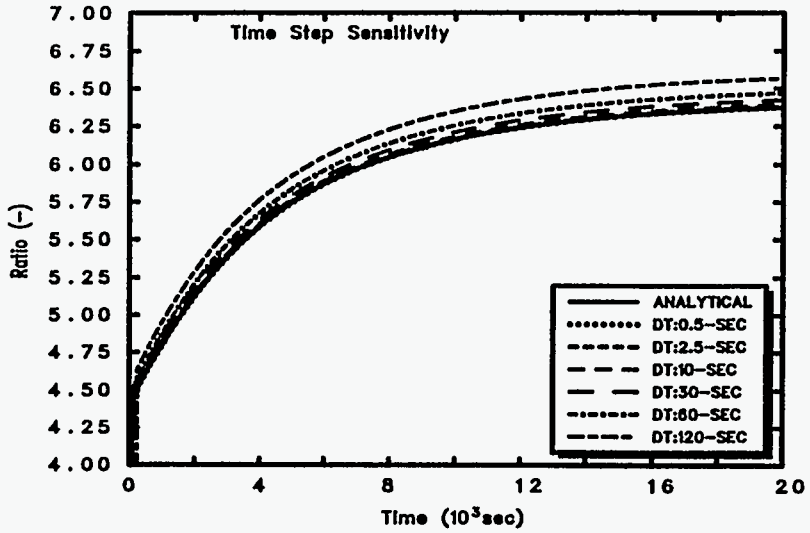


Figure 4-39: Particulate to Fiber Mass Ratio

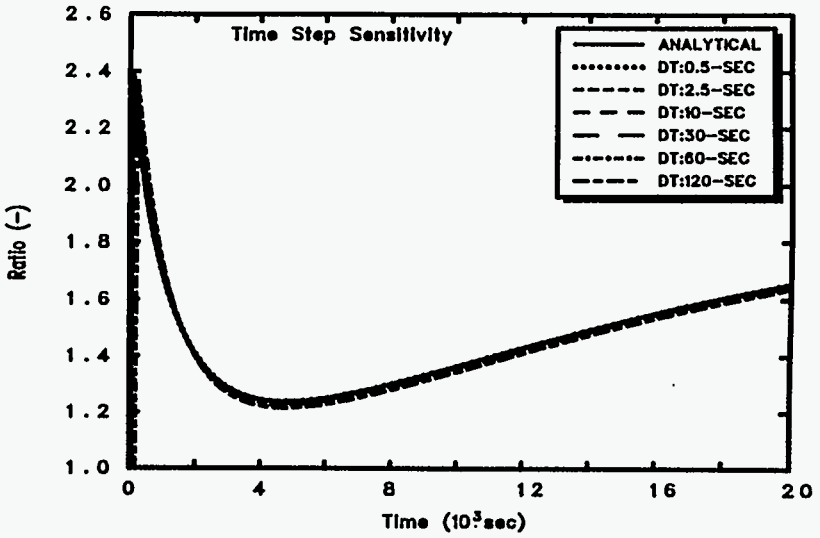


Figure 4-40: Metals to Fiber Mass Ratio

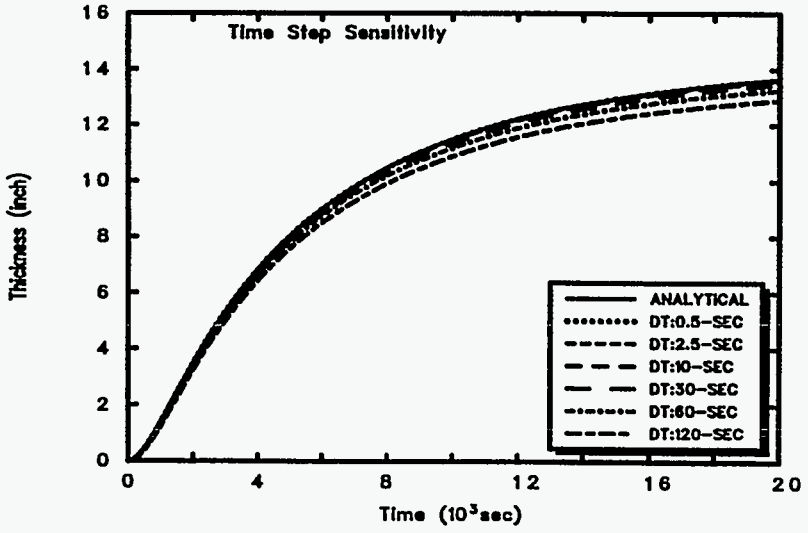


Figure 4-41: Fiber Cake Thickness on Strainer

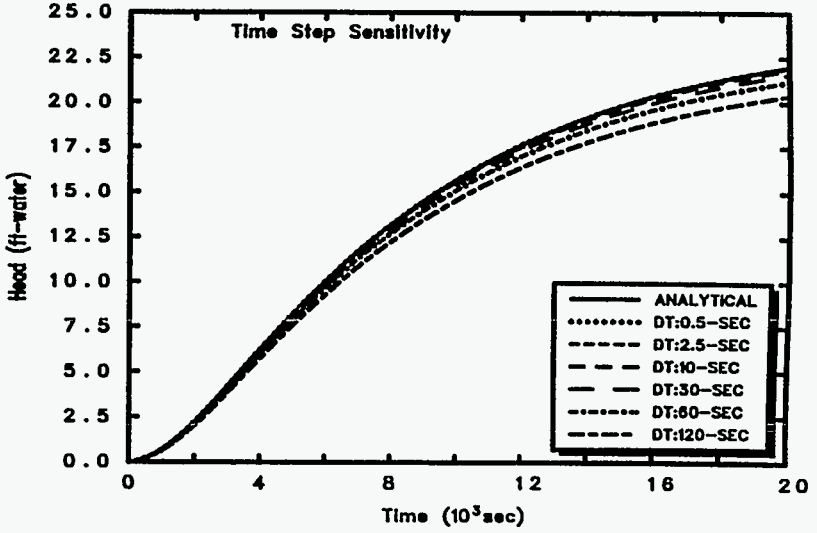


Figure 4-42: Strainer Head Loss

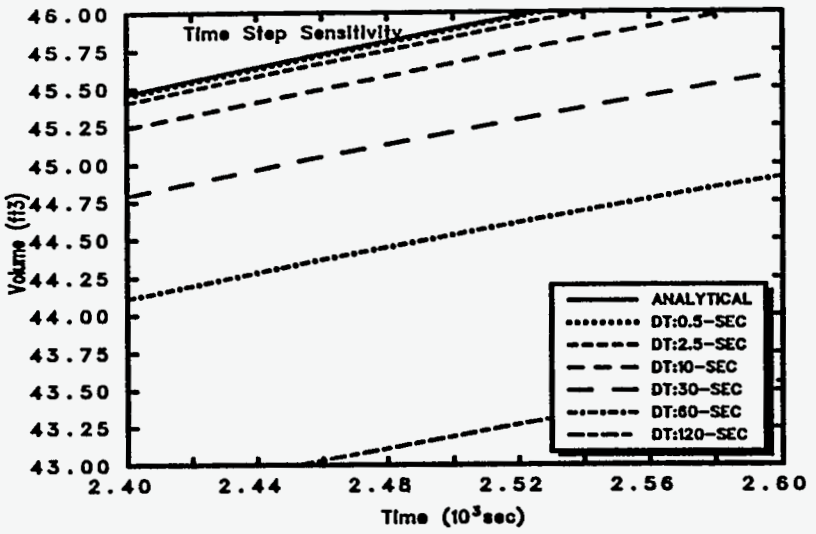


Figure 4-43: Total Fiber Transported from DW

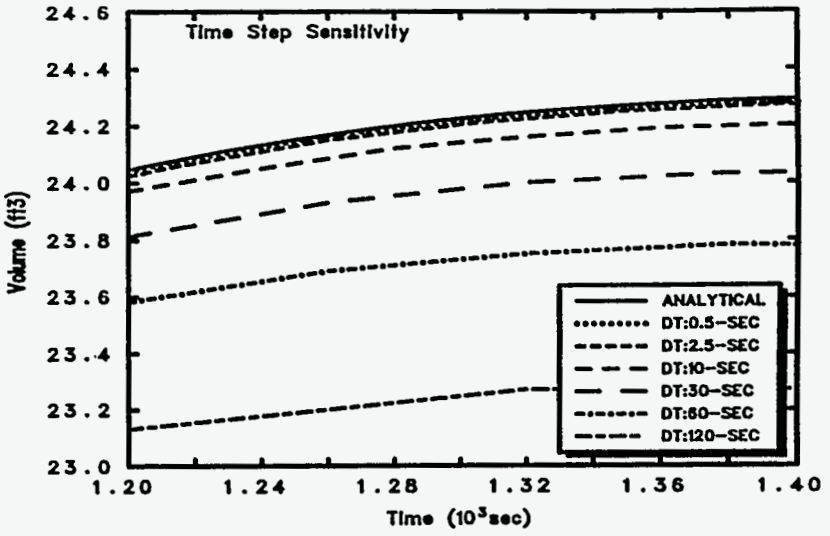


Figure 4-44: Fiber Suspended in Pool

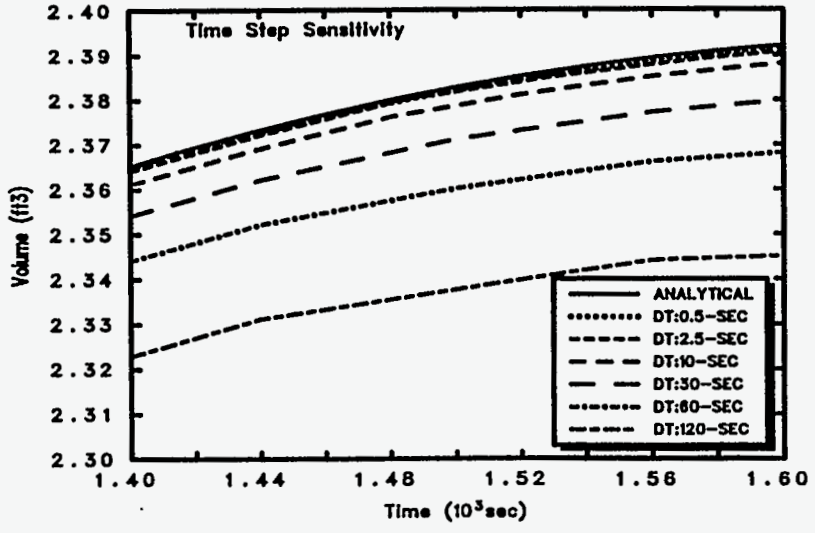


Figure 4-45: Particulate Suspended in Pool

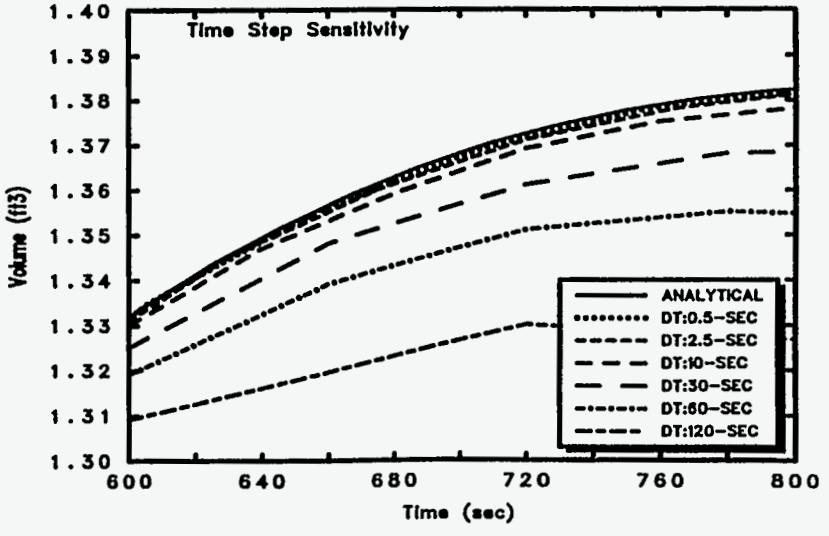


Figure 4-46: Metals Suspended in Pool

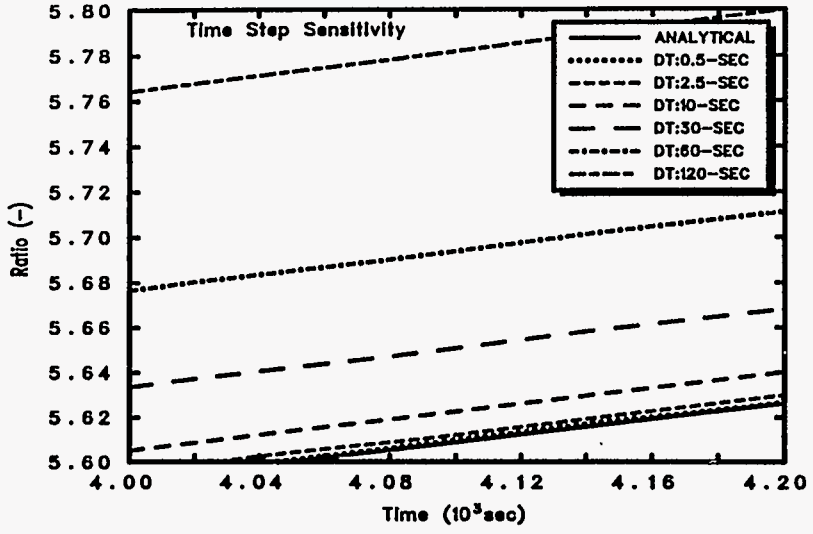


Figure 4-47: Particulate to Fiber Mass Ratio

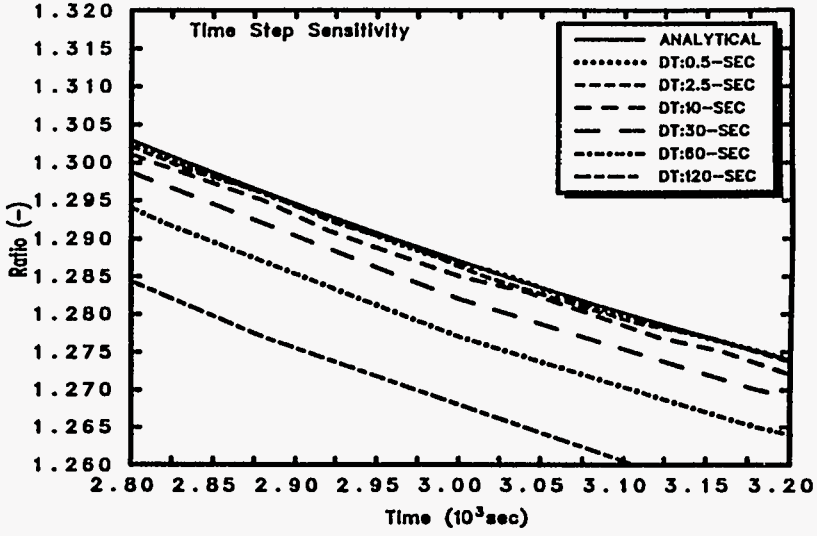


Figure 4-48: Metals to Fiber Mass Ratio

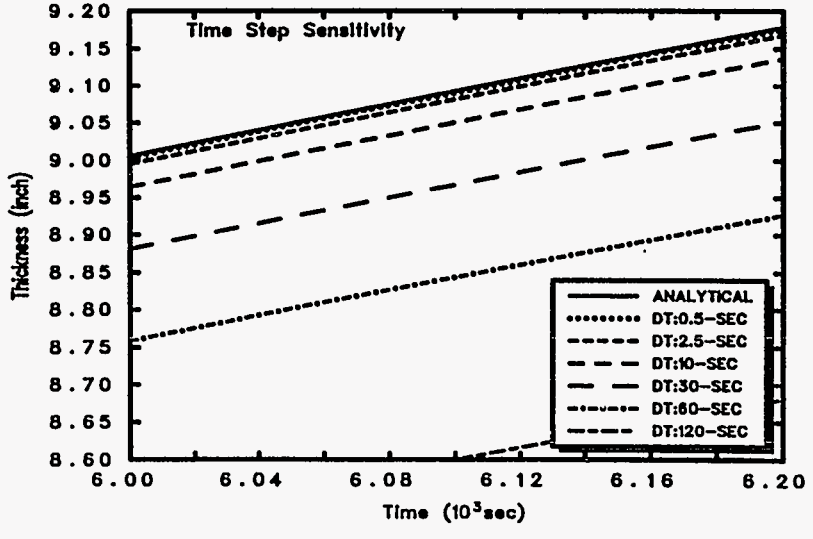


Figure 4-49: Fiber Cake Thickness on Strainer

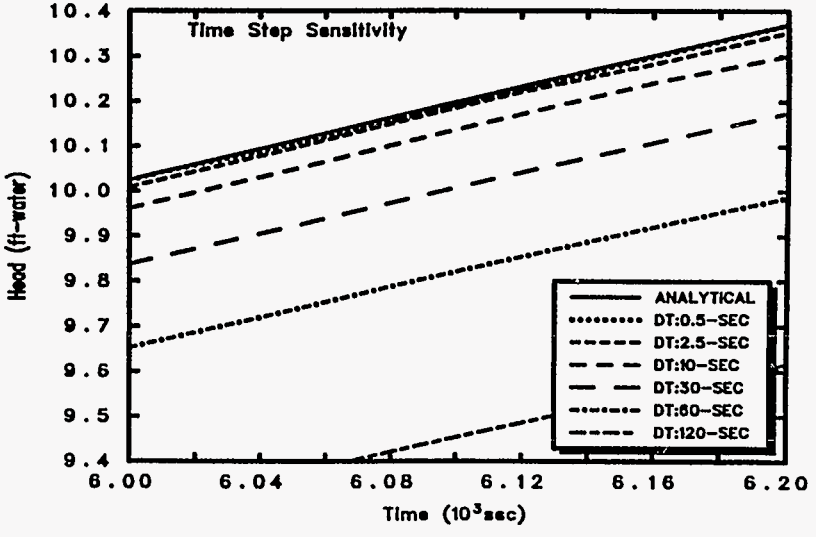


Figure 4-50: Strainer Head Loss

4.3.2 Verification of Code Capability to Reproduce NUREG/CR-6224 Results

The BLOCKAGE code analyses supporting the strainer blockage study documented in NUREG/CR-6224 were produced using Version 2.3 of the BLOCKAGE code. The subsequent code development required to achieve the objectives inherent in Version 2.5 were not intended to alter the basic methods implemented into the code models. Therefore, a design requirement was applied to the development of Version 2.5 specifying that Version 2.5 be able to reproduce the NUREG/CR-6224 results. In addition to providing a needed quality assurance step, this design requirement also provided continuity to the overall program.

The development of code Versions 2.4 and 2.4a did not alter the calculational engine and the continuity of the NUREG/CR-6224 results was accomplished by simply performing file comparisons of the before and after results and then ensuring any differences could be attributed to changes in the output formats rather than the numbers. However, the development of Version 2.5 included a nearly complete rewrite of the calculational engine to accommodate the multiple debris types and multiple pumping system design requirements; therefore, it was necessary to perform a much more detailed comparison of the Version 2.5 results to the NUREG/CR-6224 results to determine that the basic methods implemented into the models were unchanged and that Version 2.5 was capable of reproducing the NUREG/CR-6224 results.

The NUREG/CR-6224 reference plant problem which included a probabilities report associated with its 345 weld break possibilities and detailed analyses of four representative weld breaks was run with BLOCKAGE 2.5 and directly compared to the previously verified results from Version 2.4. There were two significant differences between the two versions which had to be accounted for when comparing the numbers: 1) the treatment of the pump following its predicted loss of NPSH margin and 2) the output accounting used to correlate the debris transport results.

In the older versions of the code, the pumps were assumed not to trip upon loss of NPSH margin, thereby allowing the strainer to continue to build up debris in order to ascertain the potential severity of the blockage potential, i.e., although the pump was modeled to trip at a head loss of 14 ft-water for the reference plant, the maximum potential head losses frequently reached much larger values, even thousands of ft-water for some conditions. Obviously, the head loss would not continue to increase if the pump had tripped upon loss of NPSH margin. The multiple strainer and multiple pump options implemented into the Version 2.5 required that each pump be tripped after pump loss of NPSH margin was predicted so it could not further affect the performance of the other pumps that have not lost their NPSH margin. Therefore, the performance of the single NUREG/CR-6224 reference plant pump following loss of NPSH margin could not be directly reproduced by BLOCKAGE 2.5. Therefore, a simple debug-level coding option (not available to the user) was implemented whereby the pump loss of NPSH margin could be prevented so that the NUREG/CR-6224 maximum head losses could be reproduced.

The debris accounting differs considerably in Version 2.5 from that of Version 2.4. In Version 2.4, the user was only allowed one type of fibrous debris which was generated in the drywell, only one type of particulate debris generated in the drywell, and only one type of particulate debris originating in the wetwell. Further, both the drywell and wetwell particulate results were combined when presented in NUREG/CR-6224. Each debris type in Version 2.5 was tracked separately to accommodate the general nature of the modifications associated with the multiple debris transport. BLOCKAGE 2.5 no longer assumes that all fibrous debris must originate in the drywell or that only one type of particulate debris can be formed in the drywell and one type in the wetwell. Thus, comparing the results required a little accounting.

The comparison of the actual numbers produced by Versions 2.4 and 2.5 showed that the numbers were identical except for some numerical last-digit round-off associated with different output formats. Even the detailed results such as the time-dependent distributions of debris by velocity group and the debris transport rates were found to be identical. Time-dependent plots were shown to compare well also. The probabilistic results were identical.

In conclusion, the BLOCKAGE 2.5 code is capable of reproducing the NUREG/CR-6224 results with a high degree of accuracy. The continuity of the blockage analysis program was maintained.

4.3.3 Verification of the Iterative Solution of the NUREG/CR-6224 Head Loss Correlation

The NUREG/CR-6224 head loss correlation in the BLOCKAGE code involves a set of non-linear equations which can not be solved explicitly. The equations describe the strainer head loss, the fibrous debris cake compressibility, and the debris cake particulate packing limitation. These equations in BLOCKAGE were solved using a numerical approach and the solutions were verified using a graphical technique. These equations are presented in the descriptions of the strainer head loss models in Section 2.7 and the numerical solution of the equations is discussed in the BLOCKAGE code programmer's guide, Section 3.2. The verification of the iterative numerical solution is the subject of this section.

The verification of the NUREG/CR-6224 correlation solution was accomplished by:

- plotting the equation set together and graphically determining the correct solution,
- plotting the solution to the equations in a 3-D surface plot, and
- substituting the numerical solution from BLOCKAGE back into the equations to ensure that the solution fit the equations.

The graphical solution was accomplished by programming the equations into a graphical worksheet created using the MathCad software and simultaneously plotting the head loss and the compressibility equations and the packing limitation. Several BLOCKAGE code solutions have been checked using the graphical technique to verify the correctness of the coding in BLOCKAGE. One example of a graphical solution is shown in Figure 4-51. The two equations were solved for the head loss per unit of fiber cake thickness (without compression). Then the head loss per unit of thickness was plotted as a function of the compaction fraction defined as the ratio of the compressed fiber cake thickness to the fiber cake thickness without compression. All other parameters in the equation were fixed at the appropriate values for a particular moment in time for a particular calculation. Note that the shape and position of these plotted equations change as a BLOCKAGE calculation progresses through time. The point where the two equations cross one another is the solution in units of ft-water/in. Thus, multiplying this solution by the fiber cake thickness yield the head loss. The vertical line, designated as $_{Rmin}$ in Figure 4-51 indicates the compaction fraction where the particulate debris density in the cake exceeds the particulate rubble density (usually taken to be 65 lbm/ft^3 for wetwell sludge) and further compaction would not be physically realistic. For the particular solution shown in Figure 4-51, the compaction or packing limit did not impact the solution.

Increasing the particulate to fiber ratio of the solution shown in Figure 4-51 from 1.0 to 3.84 yields the solution shown in Figure 4-52 where the compaction limit does limit the solution. Here, the solution is determined as the intersection between the first head loss per unit thickness equation and the vertical minimum packing line. The solution shown in Figure 4-52 was the NUREG/CR-6224 solution for the weld designated as RCA-J006 at the time of predicted blockage (head loss of 14 ft-water).

The failure of an iterative numerical solution technique to converge is a potential problem for any numerical technique. The convergence of the numerical solution within the BLOCKAGE code has been refined sufficiently that it is deemed unlikely that a failure to converge will occur in the future use of BLOCKAGE 2.5 for any problem involving realistic parameters. If a failure to converge were to occur, it would likely occur at the onset of the debris deposition on the strainer. At these onset conditions, the cake thickness and head loss are trivial and the failure to converge would not likely impact the overall calculational results. However, if the numerical iterative solution fails to converge, an error message is printed to alert the user.

The two most influential parameters of the NUREG/CR-6224 correlation, for a given strainer approach velocity and a given strainer area, were the fiber cake thickness and the particulate to fiber mass ratio. (The remaining parameters were the densities, specific surface areas, and the water temperature.) A 3-D surface plot of the solution to the head loss correlation is shown in Figure 4-53 using the same parameters that were used in Figures 4-51 and 4-52. The 3-D solution shows the head loss as a function of fiber cake thickness (uncompressed) and the particulate to fiber mass ratio.

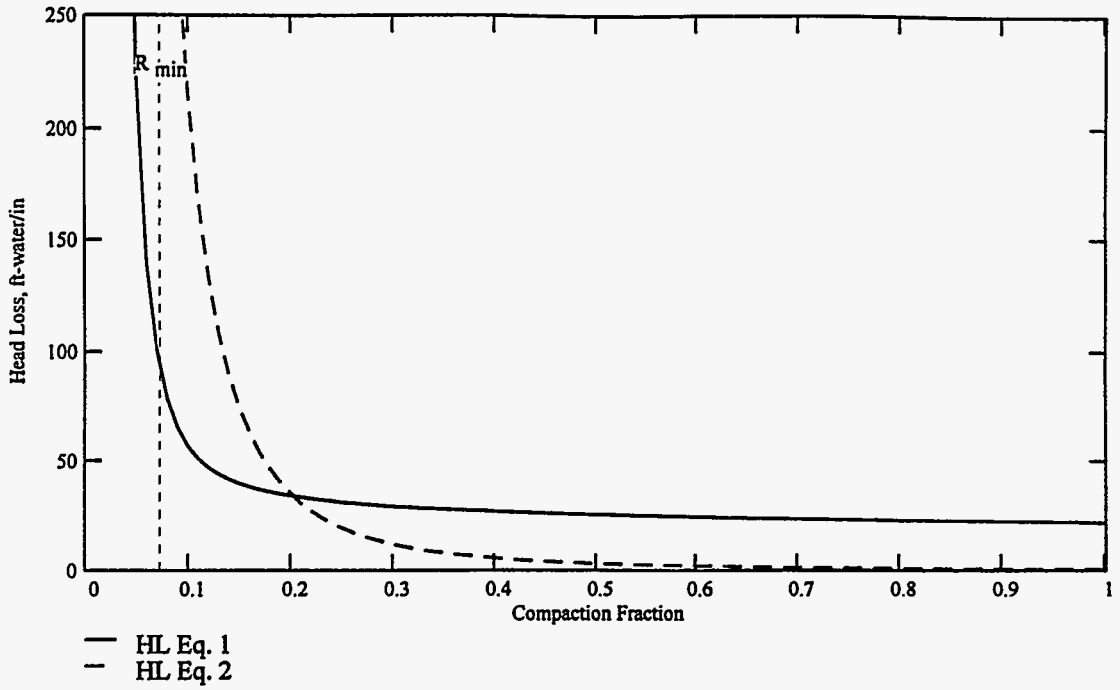


Figure 4-51: NUREG/CR-6224 Equations

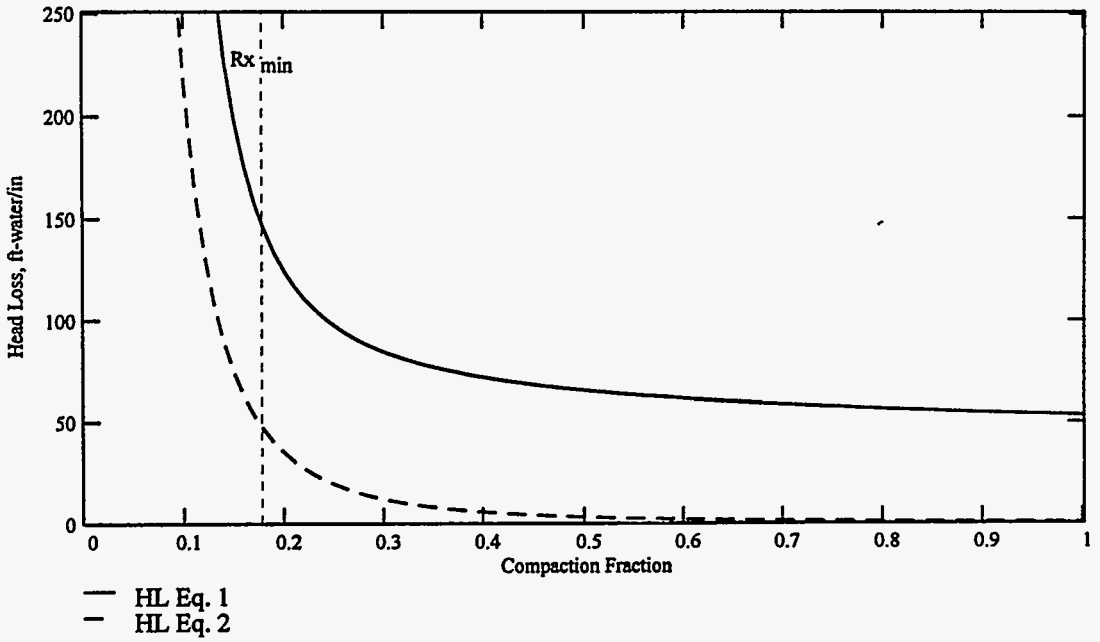


Figure 4-52: NUREG/CR-6224 Equations

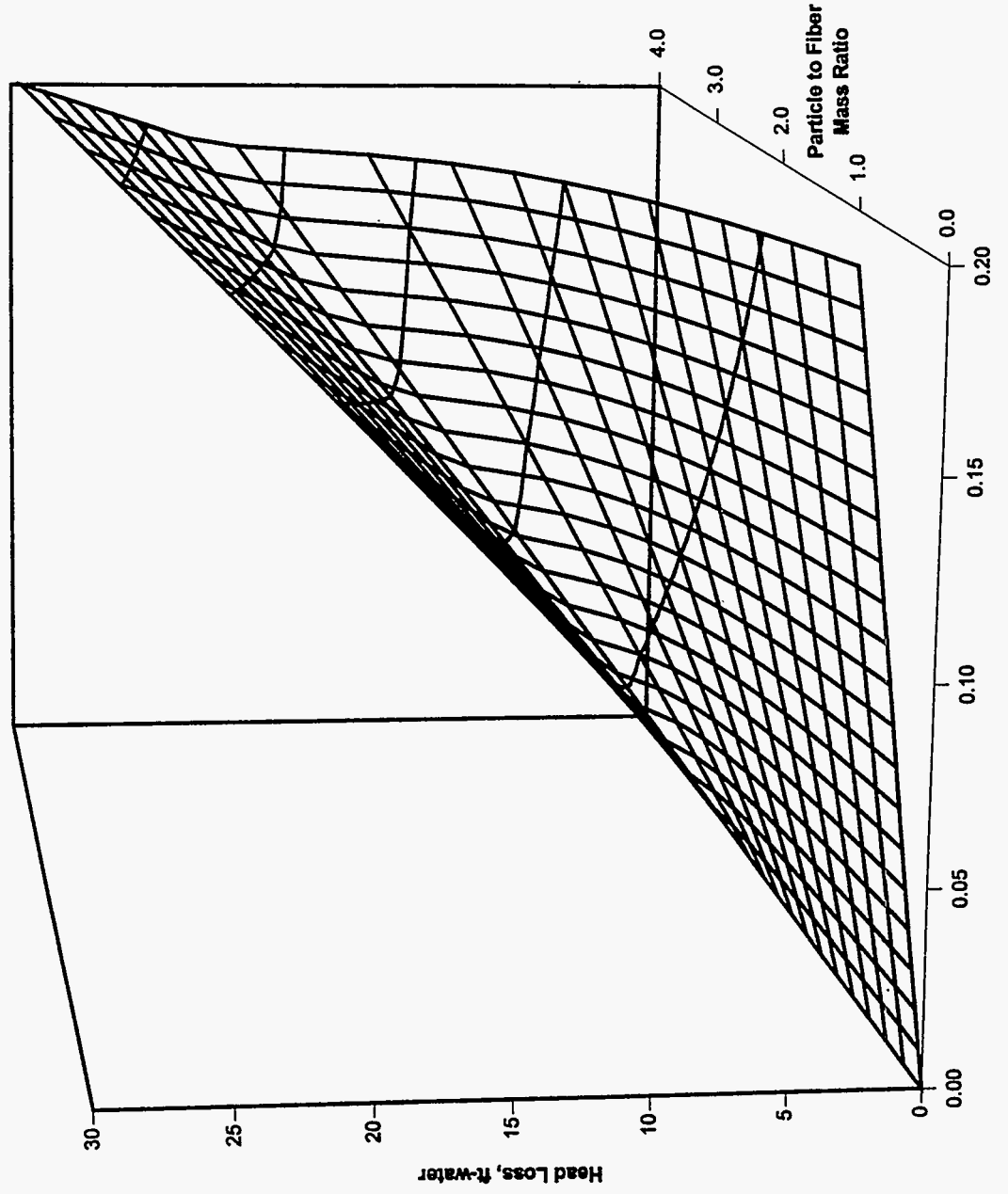


Figure 4-53: Head Loss Correlation

The shape of the 3-D surface area illustrates some of the features of the NUREG/CR-6224 correlation. First, it illustrates that the strainer head loss does not build rapidly when particulates are not present (shown as the line where the particulate to fiber mass ratio is zero). The addition of particulate debris to the fiber cake rapidly increases the strainer head loss. The particulate compaction limit, as determined by the particulate rubble density, is also illustrated in the 3-D surface area plot as the dramatic change in the slope of the surface towards the rear of the figure, i.e., the head loss was increasing very rapidly with further increase of particulate to the cake until the compaction limit was reached, then further increases in the head loss became much less pronounced. The solution shown in Figure 4-51 is found in the 3-D plot as a point located the foreground at the coordinates of 0.096 inches of fiber cake and a particulate to fiber mass ratio of 1. The solution shown in Figure 4-52 is found in the rear of the 3-D plot at the coordinates of 0.096 inches of fiber cake and a particulate to fiber mass ratio of 3.84.

In summary, the iterative numerical solution to the NUREG/CR-6224 head loss correlation was subjected to extensive testing and code verification to ensure that the BLOCKAGE code correctly solves the correlation. Thus, this area of the coding has been thoroughly verified.

4.3.4 Verification of Volume Calculations

Target and debris volumes are calculated and collated by debris type and class by the BLOCKAGE code prior to performing the time-dependent transport calculations. When target input is selected by the user, the debris volume of each individual target is calculated using the user specified destruction fractions. The transportable drywell debris volumes of the target generated debris (or user specified debris volumes when this input option is selected) and the non-target drywell debris volumes are determined using the user specified transport fractions. Then the transportable debris volumes were collated by debris type for the time-dependent transport calculation. The debris volumes are also collated by debris classification as output information.

Since these volume calculations are performed by the code prior to performing the time-dependent solution, these calculations were not verified by the analytical test problem. The coding associated with the volume calculations was initially verified by the code developer by performing a variety of hand calculations; however, an independent and more rigorous verification was deemed necessary. Therefore, an independent test of the volume calculations was devised where the volumes were determined using a spreadsheet type of calculation and compared with the BLOCKAGE code volumes. The spreadsheet calculation was performed by an engineer not directly associated with the development of the code.

Two test problems were devised to verify the volume calculations, i.e., one problem involved calculating the volumes from target dimensions and the second problem employed the user specified volume option. Each problem contained multiple target data for three separate weld breaks which were at different locations. Both problems considered 11 debris types with debris originating in the drywell from target destruction and from non-target debris and with debris originating in the wetwell. The input numbers used for the target dimensions, the destruction fractions, and the transport fractions were specified with values that avoided rounded numbers like 0.5, 1.0, or 2.0. Thus, the test problems were complex and varied enough to ensure that coincidental duplication of the BLOCKAGE computed volumes by hand unlikely.

The hand verification calculations were performed using an electronic spreadsheet. First, the BLOCKAGE code input data were entered into the spreadsheet. Then volumes were calculated for each intact target, the debris generated from each target using the destruction fractions, and the transportable debris using both the blowdown and washdown transport fractions. All the spreadsheet calculated results were displayed with five decimal place accuracy.

All volumes calculated with the BLOCKAGE code were reproduced and verified by the spreadsheet calculations. The BLOCKAGE and spreadsheet results were exactly the same with the exception of a few last digits which could be attributed to round-off differences between the two methods. The test problems were run once again using the final release of the code to ensure that the coding that performed the volumes was not altered after the original tests were performed. Thus, the complexity, diversity, and independence of the coding verification tests precluded any accidental duplication of the BLOCKAGE results, thereby certifying that the

volume calculations in BLOCKAGE were correctly coded.

4.3.5 Verification of Probabilistic Reports

The BLOCKAGE code prepares a probabilities report (also referred to as a frequency report) upon user request that tabulates the plant-wide weld break and the strainer blockage frequencies by the pipe diameter, the diameter class, the piping system, and the weld break location. Further, the report provides ratios of the strainer blockage frequencies to the weld break frequencies. These reports are obviously of little use when the user is only studying a single weld break or even a few weld breaks. However, when a plant-wide data set such as that of the NUREG/CR-6224 reference plant which consisted of 345 separate possible weld breaks is used, the probabilities report provides the overall view of the performance of the plant. It allows the user to easily determine which systems or which pipe diameters tend to drive the overall blockage probability.

The BLOCKAGE coding that produces the probabilities reports was verified by reproducing the probabilities with hand calculations at several advancements in the development of the code. Every number in the probabilities reports produced for the NUREG/CR-6224 study (reference plant with a plant-wide data base of 345 welds) using BLOCKAGE 2.3 were hand calculated from the code summary report. The code summary report included the weld break frequency for each of the 345 potential weld breaks and the code determination of whether or not the pump suction strainer blocked for each of those welds. The weld break frequencies in the summary report were verified by comparing the frequencies to the weld break frequency table included in the code input. These probabilities were sorted and summed by pipe diameter, piping system, weld break location, and by whether or not strainer blockage occurred. The probabilities reports for code Version 2.3 were completely verified.

Since the probabilities report coding was not altered in any way that would affect the calculation of the break and blockage frequencies between Version 2.3 and the current Version 2.5, the coding remains verified for BLOCKAGE 2.5, as well. Electronic file comparisons were made to determine the coding which changed between the two versions and it was determined that only a couple of output format statements were altered such as increasing the length of the code version name. Additionally, when the NUREG/CR-6224 results were reproduced (see Section 4.3.2), it was verified that the calculated probabilities had not changed. Thus, the coding that produces the probabilities reports in BLOCKAGE 2.5 has been thoroughly verified.

4.3.6 Verification of Input Processing

The input processor of BLOCKAGE 2.5 tests the code input against predetermined limitations in an attempt to prevent the user from entering invalid or meaningless data. The input is also checked by the GUI using the same limitations. These input data limitations were established to:

- Limit an integer value to a specific set of allowed numbers when required
- Ensure that a number is positive when a negative number or a zero is not allowed
- Ensure that input values are not excessively large
- Ensure that names and identifiers are unique when required
- Ensure that distributions sum to one when required
- Ensure that time-dependent transport rates integrate to one when required

The BLOCKAGE input processor does not have the capability of determining when a line of input data is missing, if an extra line of data exists in an input file or if a line of data does not contain the required data. However, this problem is circumvented by using the GUI. If the user executes the code from the DOS level, it is the users responsibility to make sure that the input files contain the required data in the correct format. If the data is not in the correct format, the code will likely abort the calculation; however, the error message indicating the reason for the abort will likely not correctly indicate the actual root cause of the aborted run. In this case, the user will have to perform a little detective work to find the true reason for the aborted run.

The input limitations were independently reviewed to determine that the chosen limitation values were realistically reasonable and valid. In addition, numerous combinations of illegal input values were attempted to determine if BLOCKAGE 2.5 actually detected these illegal values, wrote a meaningful message to the output error message file, and terminated the calculation. Although this verification

effort was not totally exhaustive, a reasonably high level of confidence has been achieved in the input processor of BLOCKAGE 2.5.

4.3.7 Testing of the BLOCKAGE Code Using Alternative Configurations

One of the most difficult aspects of completely verifying any code is ensuring that the code executes properly for all possible input configurations. Typically, test problems only use a fraction of the capability of the code. For example, BLOCKAGE 2.5 allows a user to include as many as 32 individual pumps in a particular calculation; however, it is unlikely that this many pumps will ever be used in a single calculation. The coding included many calculational loops that were controlled by variable input parameters such as the number of debris types or the number of welds. One type of error that can enter a code and not be found during coding verification activities is an incorrect index on a DO LOOP or storage array or the use of a constant number when a variable parameter should have been used.

A minimum dimensions test and a maximum dimensions test were devised to help ensure that a full range of problem configurations could be run.

- In the minimum dimensions test, one was selected for each parameter input, such as one pump on one pumping system, one debris type with one settling velocity group, one target on one weld, etc.

- In the maximum dimension test, the maximum allowed input value was selected for each input parameter such as 4 pumps for each of 8 pumping systems, 20 debris type each with 20 settling velocity groups, 40 targets per weld, 10 piping systems, 5 weld break locations, etc.

These two problems both executed successfully providing a strong indication that the parameters which controlled the coding looping and data storage logic were correctly coded.

4.4 Validation of BLOCKAGE 2.5 Predictions

4.4.1 Selection of Experiments

A review of the existing literature revealed that several experiments were conducted to develop head loss correlations that relate head loss across a strainer to the quantity and type of debris deposited on the strainer, flow through the strainer and the fluid temperature. The majority of these tests, however, employed 'closed loop' facilities to obtain the head loss data. Validation of BLOCKAGE using experimental data from such tests is very difficult and is not appropriate. On the other hand, very few experiments were conducted using an experimental set-up that is ideal for BLOCKAGE validation. These experiments, commonly referred to as open-loop experiments, use a relatively large tank into which debris are introduced in a manner similar to an actual BWR LOCA. Typically these tests also employee small-scale strainers and realistic debris. Examples of such experiments include:

1. Pennsylvania Power and Light (PP&L) Company experiments
2. Performance Contracting, Inc. experiments, and
3. Boiling Water Reactor Owner's Group (BWROG) experiments.

The validation effort used data from PP&L [NRC, Brinkman and Brady, 1994] tests for the following reasons: 1) the data has been made available to the NRC for public usage prior to start of this study, 2) the data was widely referenced in the past studies, and 3) these are the only tests that measured concentration data which is vital to determining the quantity of sludge filtered by the debris bed. As a result, the PP&L tests were selected for validation.

4.4.2 Description of Experiments

Experiments conducted by PP&L at the Alden Research Laboratory, Inc. (ARL) were used to validate

BLOCKAGE 2.5. These experiments employed an open loop, equipped with a small scale semi-conical strainer and a mixing tank to obtain the head loss data as a function of time and concentration of debris and debris-sludge mixtures. The tank is 5ftx8ftx9ft with a water volume of 240 ft^3 and floor area of 40 ft^2. The strainer is 2.7 ft^2 in area with 1/8 inch holes typical of ECCS strainers used in the operating BWRs. A total of 14 tests were conducted at a temperature of about 70 °F using pure insulation debris (i.e., debris without any sludge loading). Of these, eight tests (Tests 01-08) used debris that were described as 'fibers', and the remaining tests were conducted using larger size 'shreds'. In each test, a known quantity of insulation was initially added and the tank water was thoroughly mixed to obtain a homogeneous debris water mixture. After a short period of mixing, the debris water mixture was allowed to flow through the strainer, where the debris was filtered. To maintain a constant concentration throughout the test, insulation debris were discretely at pre-selected levels. The quantity of insulation debris to be added to the tank to maintain such constant concentration was calculated from a mass balance equation assuming a filtration coefficient of 1.0. The concentration of insulation in the tank varied from 0.00005 wt% to 0.003% wt%. The flow rates varied between 0.67 ft/s and 0.96 ft/s. Each test was run until head loss exceeded 22 ft-water, after which flow through the loop could not be maintained at a steady rate (apparently the pump used had low head and became unstable when the head loss across the strainer increased beyond 22 ft-water). The flow velocity and head loss across the strainer were measured directly in the experiment, whereas the concentration of the insulation debris was estimated from simple mass balance equations (mass added to the tank/volume of water in the tank). The concentration and the flow information was used to estimate the debris layer buildup on the strainer as a function of time.

The procedure was very similar in the tests that used sludge and insulation mixtures. Also, in these tests known quantities of sludge and insulation debris were added and mixed with the tank water to obtain a uniform initial concentration of debris and sludge. Later on, additional quantities of insulation debris were added to maintain the initial debris concentration, but no sludge was added. This resulted in steady decrease in sludge concentration as a function of time while the insulation debris concentration remained steady. The sludge concentration, measured as a function of time, was used to estimate quantity of sludge filtered by the debris cake formed on the strainer surface. Once again the head loss and the flow through the strainer were measured directly, where as the insulation debris layer thickness was estimated using a simplified mass balance equation.

For each test, insulation debris concentration, debris theoretical thickness, flow rate and head loss measured/calculated by ARL were listed as a function of time [NRC, Brinkman and Brady, 1994] (Table 2-4 of Appendix-C).

4.4.3 Validation Procedure

A total of six tests were selected for validation. In these tests (Test Nos. 1, 3, 4, 5, 26 and 29) strainer cake thickness varied from 0.1 inch to 5 inches, while the sludge-to-fiber thickness on the strainer varied between 3 and 8. BLOCKAGE was used to recreate each of these tests as closely as possible [1]. To ensure that BLOCKAGE runs accurately simulated experiments, BLOCKAGE results corresponding to strainer flow and debris loading were compared to the measured/calculated data. The head loss transient response predicted by BLOCKAGE was then compared to the experimental data.

4.4.4 Results of Validation

With the results presented in Figure 4-54 through 4-58, it can be clearly stated that good agreement was observed between BLOCKAGE 2.5 and experimental data. In all cases, the measured head loss was within 50% of BLOCKAGE predictions; with BLOCKAGE predictions always higher than the experimental data. Considering that the experimental data is associated with large uncertainties ($>\pm 35$) such a comparison is exceptional. Also note that largest discrepancy was observed when the head loss across the strainer exceeded 22 ft-water and the pump flow became unsteady. At such conditions, the BLOCKAGE solution algorithm, which does not include pump-performance models, is not expected to provide very good comparison. Two observations can be made form this validation effort:

[1] The flexibility offered by BLOCKAGE was found to be of great use for this purpose.

1. The BLOCKAGE solution algorithm and the head loss correlations are accurately implemented and they provide reasonably accurate prediction of head loss across the strainer.

2. BLOCKAGE predicted head losses tend to be slightly higher than the actual head losses, which is expected considering that the head loss equation was developed to bound the experimental data in most cases.

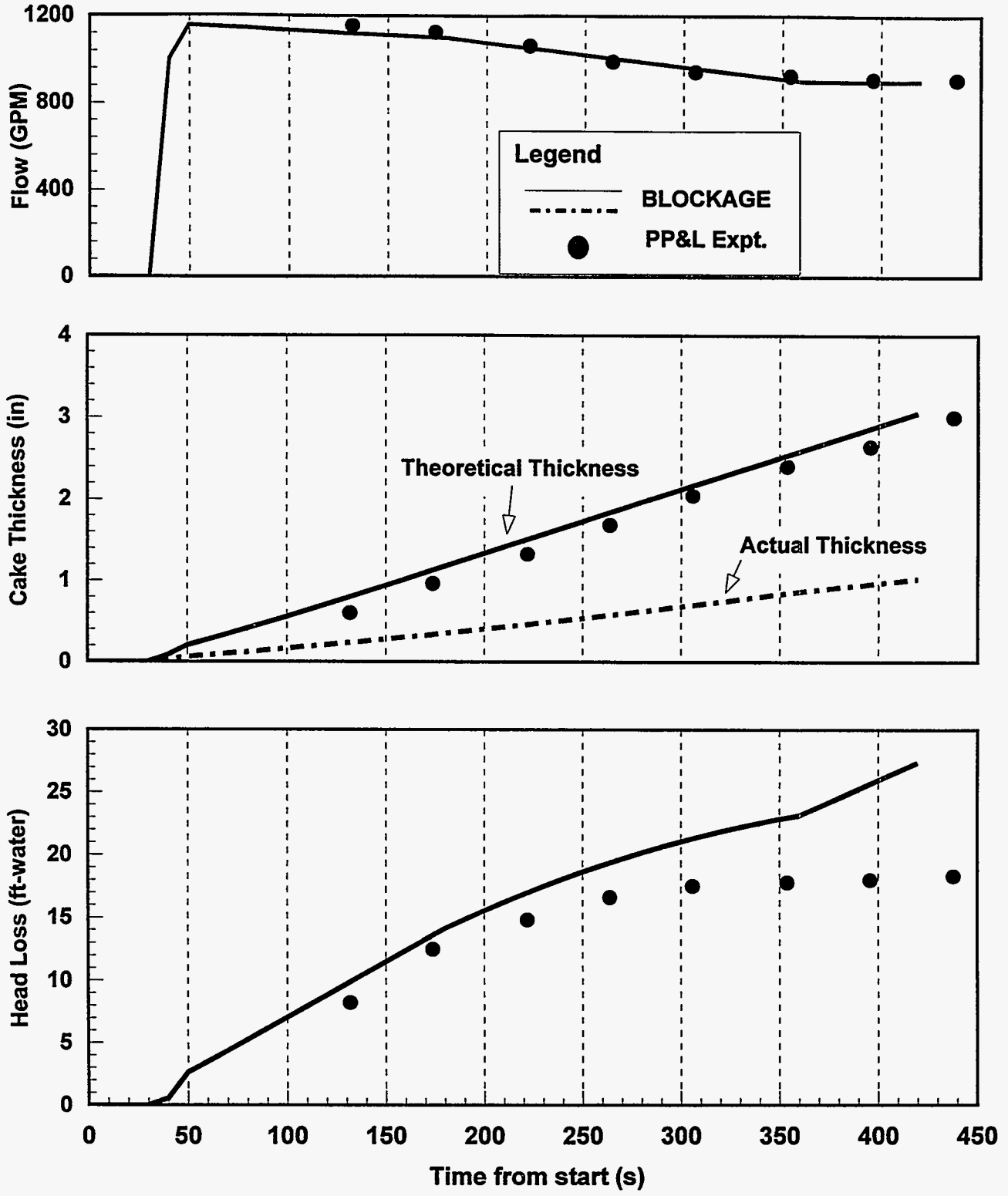


Figure 4-54: Comparison of BLOCKAGE Predictions with PP&L Test #1

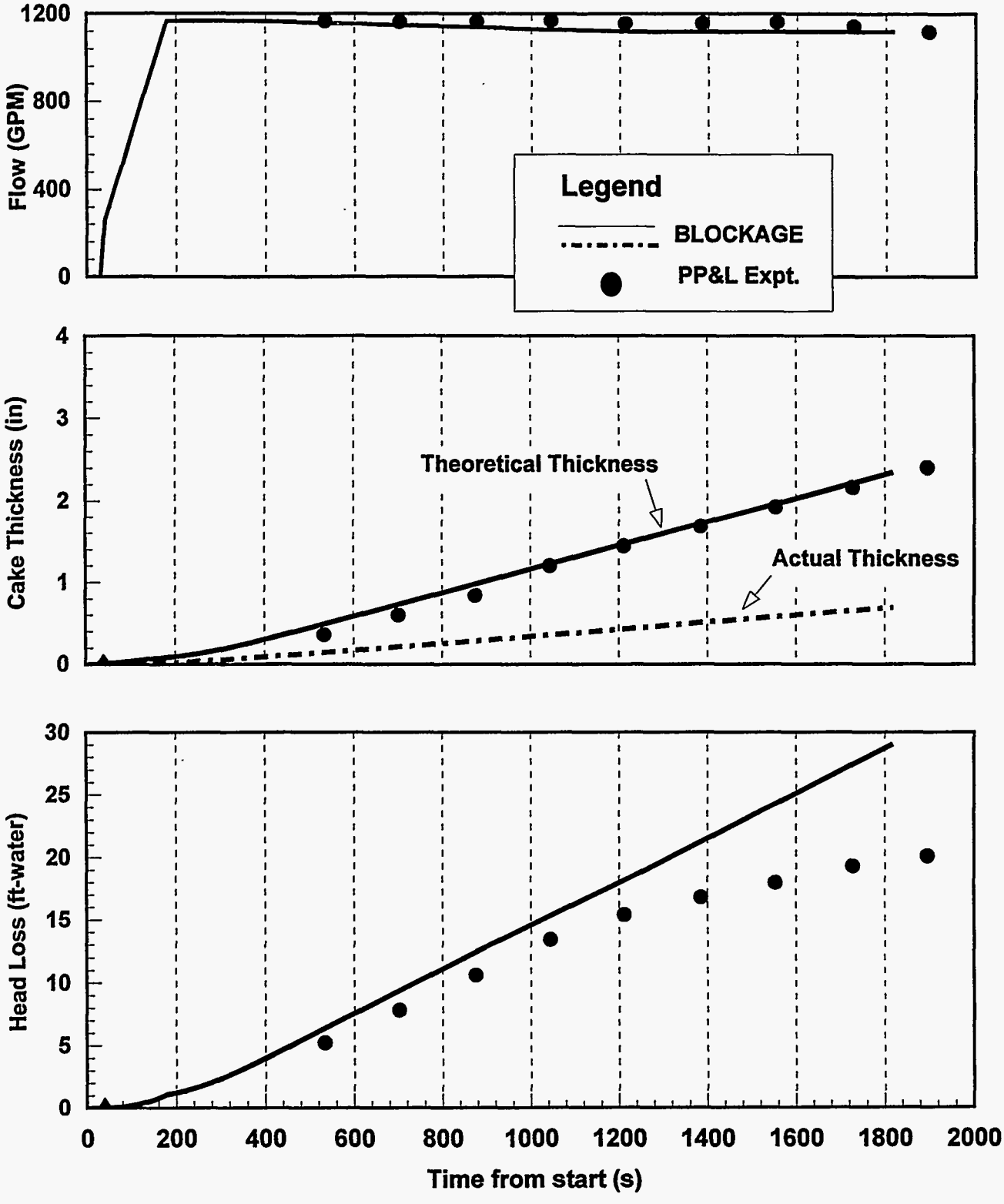


Figure 4-55: Comparison of BLOCKAGE Predictions with PP&L Test #3

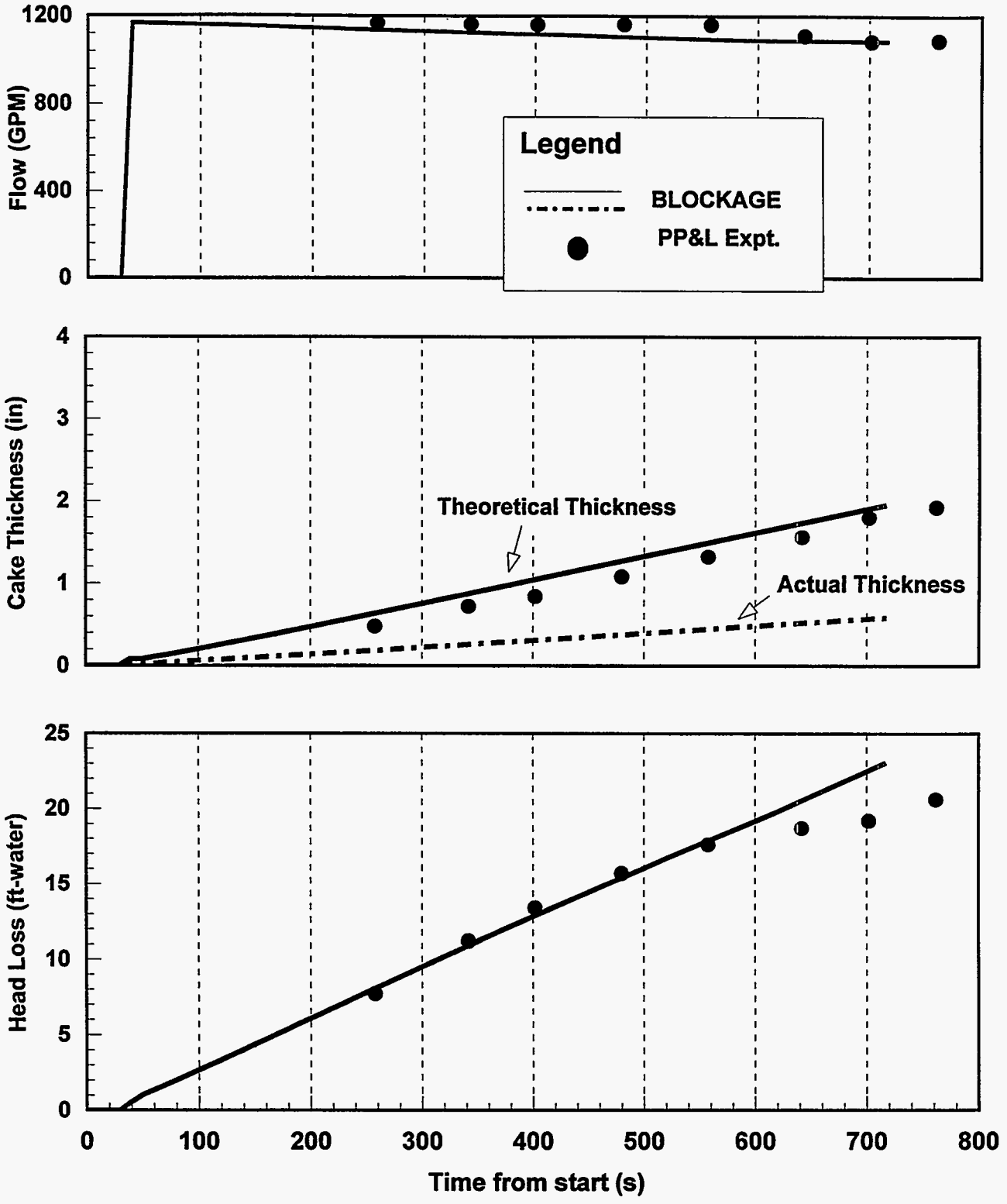


Figure 4-56: Comparison of BLOCKAGE Predictions with PP&L Test #4

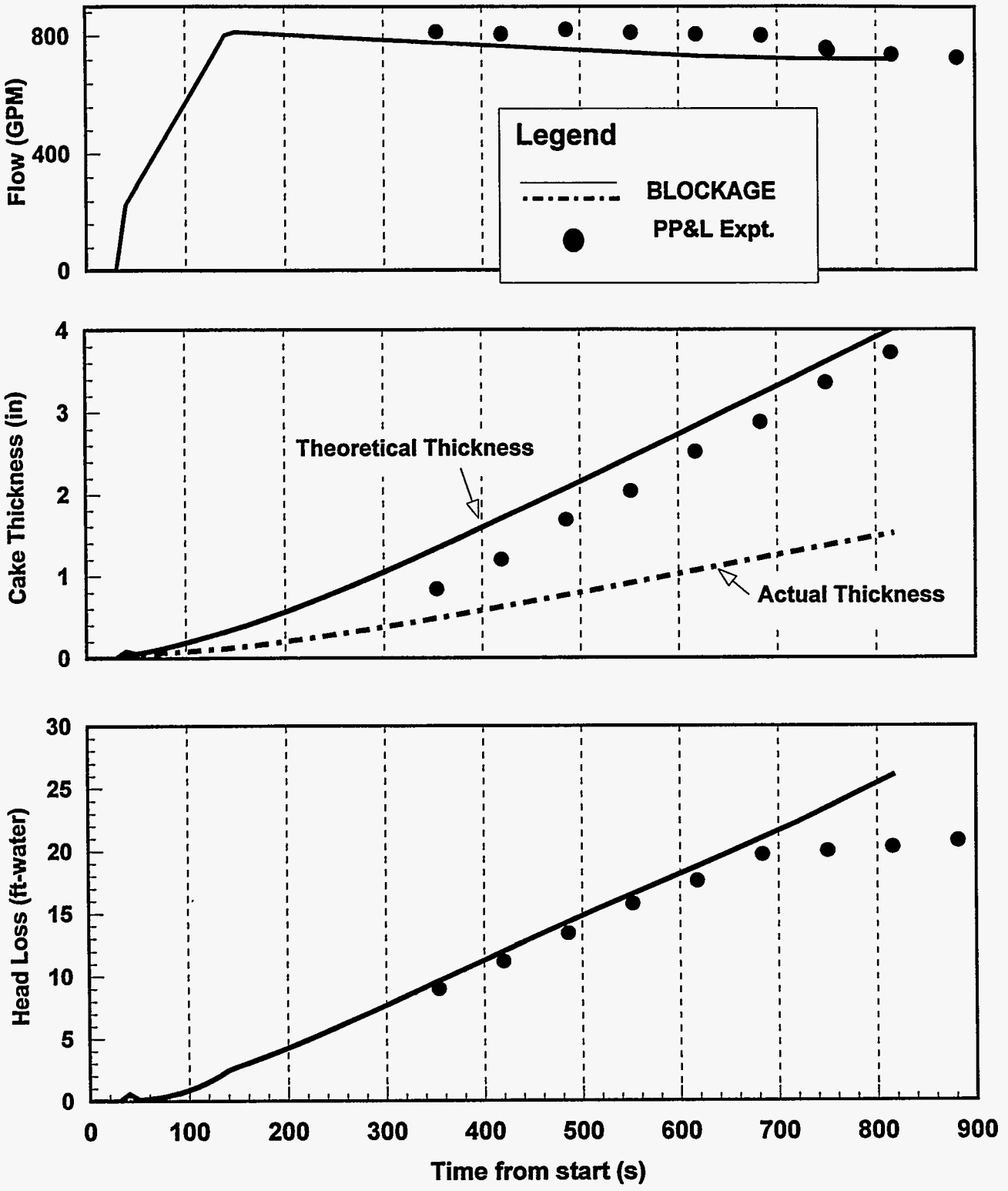


Figure 4-57: Comparison of BLOCKAGE Predictions with PP&L Test #5

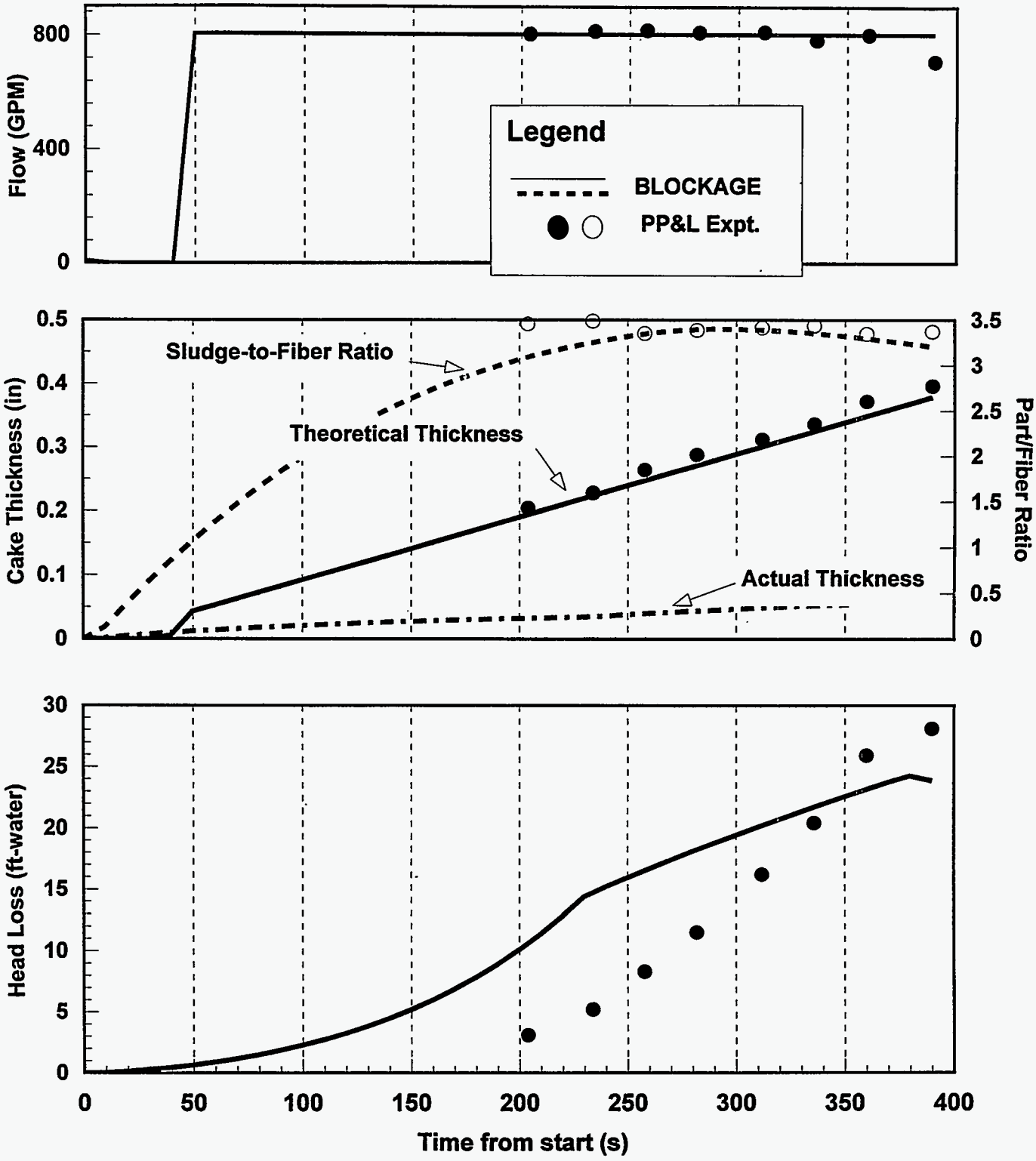


Figure 4-58: Comparison of BLOCKAGE Predictions with PP&L Test #26

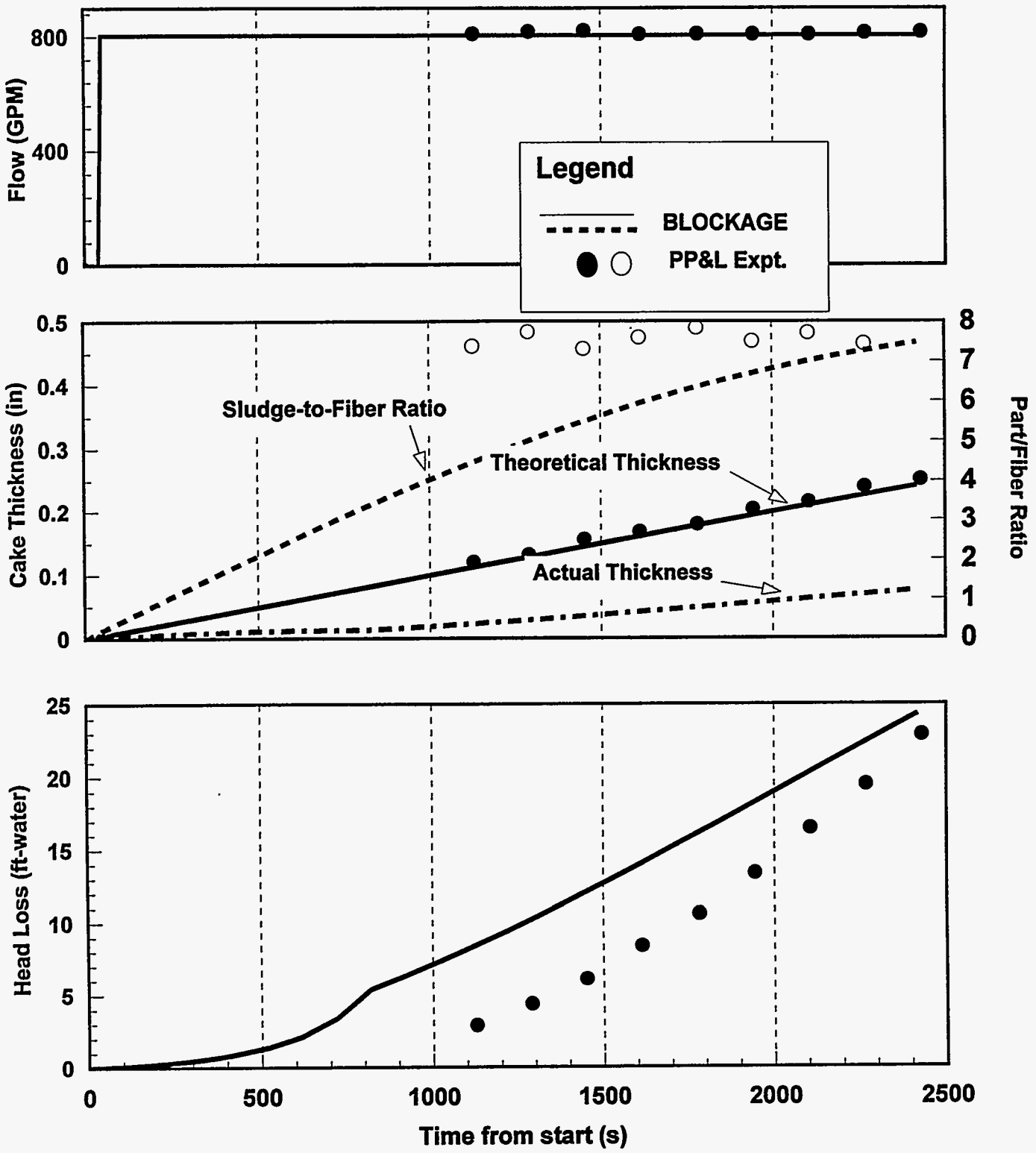


Figure 4-59: Comparison of BLOCKAGE Predictions with PP&L Test #29

5.0 GUI Programmer's Guide

5.1 General Architecture

5.1.1 Overview

The BLOCKAGE application consists of two programs: 1) the BLOCKAGE Graphics User Interface (GUI) and 2) the BLOCKAGE Calculation Engine. Figure 5-1 illustrates the two programs and the files, input screens, and results windows associated with each program. This section of the Programmer's Guide covers the GUI program.

BLOCKAGE GUI is a Microsoft Windows™ program developed using Microsoft Visual C++ Version 1.51 and the Microsoft Foundation Class Library Version 2.5. Working with both of these products requires a fairly deep understanding of Visual C++, the Microsoft Windows™ Software Development Kit, the Microsoft Foundation Class (MFC) library and the Microsoft MFC Application Framework. This document assumes that the reader is familiar with these topics.

5.1.2 Class Descriptions

Figure 5-2 provides an overview of the classes used in BLOCKAGE GUI. BLOCKAGE GUI was generated using the Microsoft Application Wizard and uses the Microsoft Foundation Class Library. Most of the BLOCKAGE GUI classes were produced by the Application Wizard and derived from an MFC class. Below is a brief description of each class.

CblockuiApp Encapsulates the code to initialize, run, and terminate BLOCKAGE GUI. This class was generated automatically by the Application Wizard and customized for BLOCKAGE GUI. CBlockuiApp is derived from CwinApp.

CmainFrame Provides the functionality of a Windows™ multiple document interface (MDI) frame window along with members for managing the window. This class was generated automatically by the Application

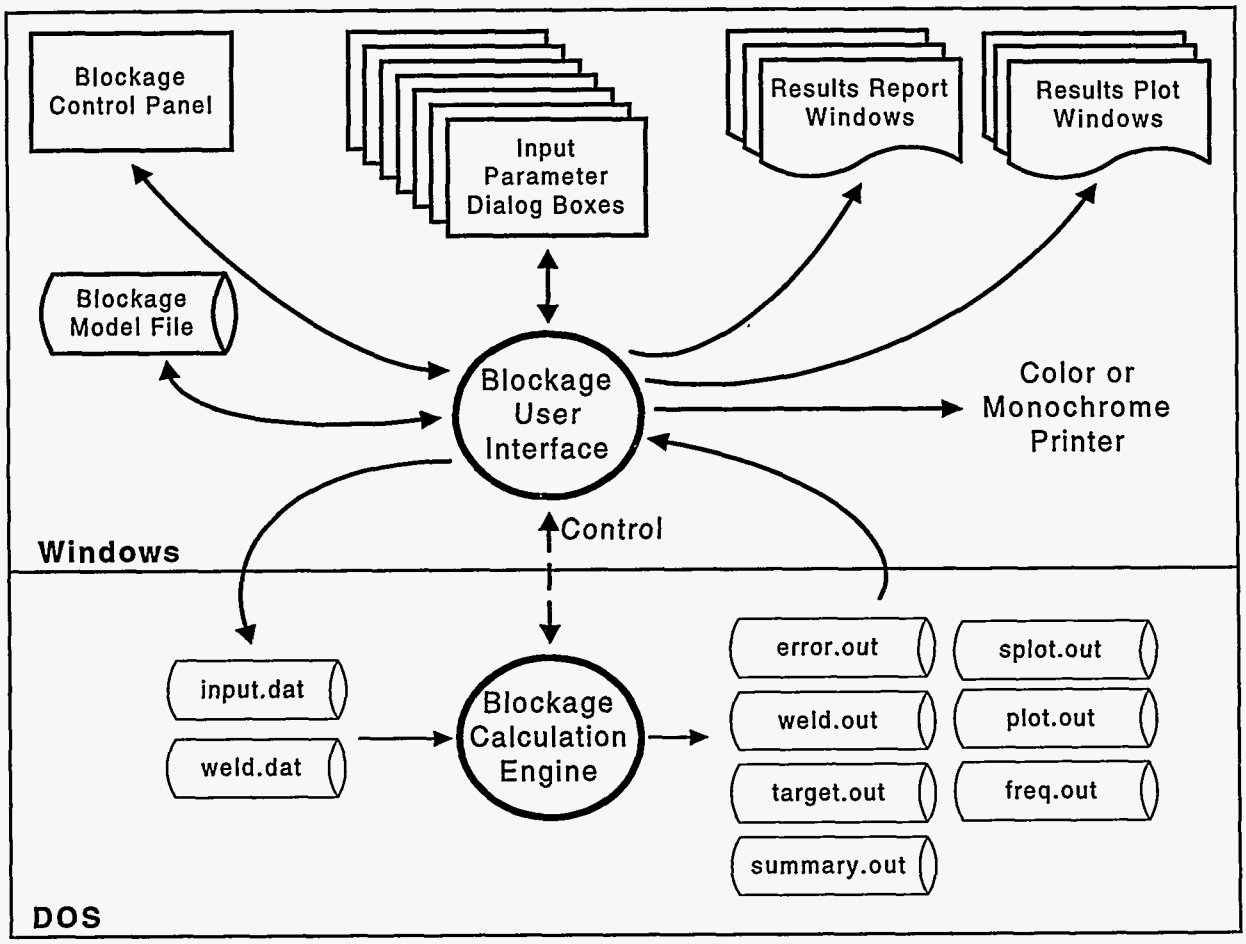


Figure 5-1: BLOCKAGE UI in Context

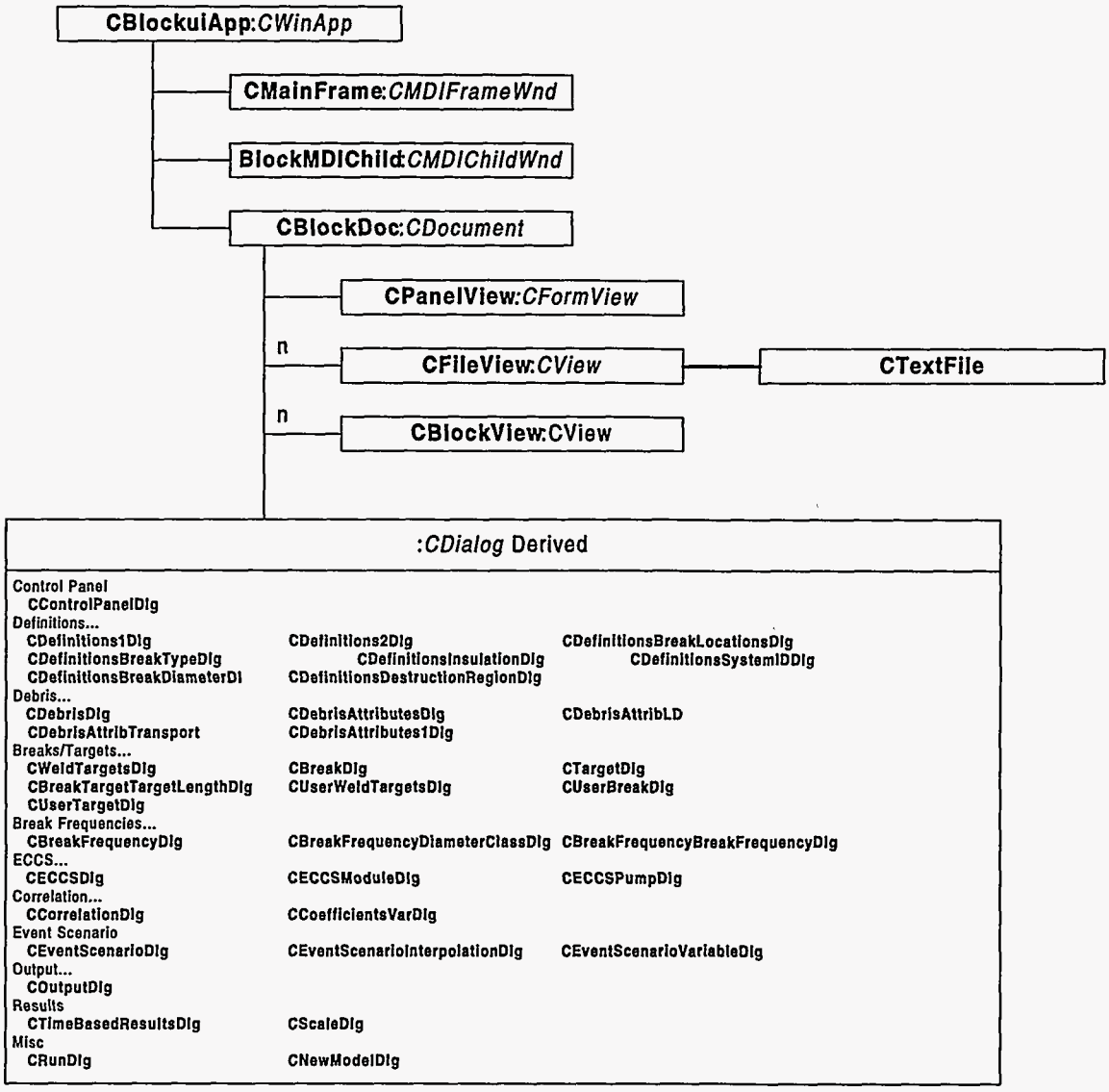


Figure 5-2: Classes

Wizard and was customized for BLOCKAGE GUI. CMainFrame is derived from CMDIFrameWnd.

BlockMDIChild Provides the functionality of a Windows™ multiple document interface (MDI) child window. This class was generated automatically by the Application Wizard and was customized for BLOCKAGE GUI. BlockMDIChild is derived from CMDIChildWnd.

CblockDoc Provides the functionality for the MFC document, which in the case of BLOCKAGE GUI corresponds to a model. This class supports the standard operations for creating a document, loading a document, saving a document, and so forth. CBlockDoc was generated automatically by the Application Wizard and was customized for BLOCKAGE GUI. CBlockDoc is derived from Cdocument.

CpanelView Provides the functionality for the BLOCKAGE GUI control panel view. A view is attached to a document and acts as an intermediary between the document and the user. The panel view displays the control panel and supports control panel operations. Each BLOCKAGE GUI document has one and only one CPanelView. The CPanelView is derived from Cview.

CfileView Provides the functionality for a BLOCKAGE GUI report file view. A view is attached to a document and acts as an intermediary between the document and the user. The file view displays a report in a scrolling text window and supports printing the view. The CFileView is derived from Cview.

CtextFile Supports reading and buffering data from a text file for using in a file view. Each CTextFile class is associated with one and only one CfileView.

CblockView Provides the functionality of a BLOCKAGE GUI plot view. A view is attached to a document and acts as an intermediary between the document and the user. The plot view displays an X,Y plot with time along the X axis and the variables of interest along the Y axis. The plot view displays a plot in a window and supports printing and managing the view. The CBlockView is derived from Cview.

Cdialog BLOCKAGE GUI includes 37 dialogs that are derived from CDialog. Most of these dialog boxes are used to handle input parameters display and entry. All of these dialog boxes have an extremely similar format due to the inherent structure of CDialog and the common support routines provided by the BLOCKAGE GUI Dialog Utilities.

5.1.3 Components

Figure 5-3 provides an overview of the BLOCKAGE GUI components. A component is a set of related routines contained in a single source file. BLOCKAGE GUI components are divided into three classes, described below, and a class of Microsoft supplied components.

Application Architecture Classes

These components support the MFC application architecture for BLOCKAGE GUI.

Application Contains the CBlockuiApp class. This class is the main application level class for BLOCKAGE GUI.

Frame Contains the CMainFrame class. This class handles the MDI frame.

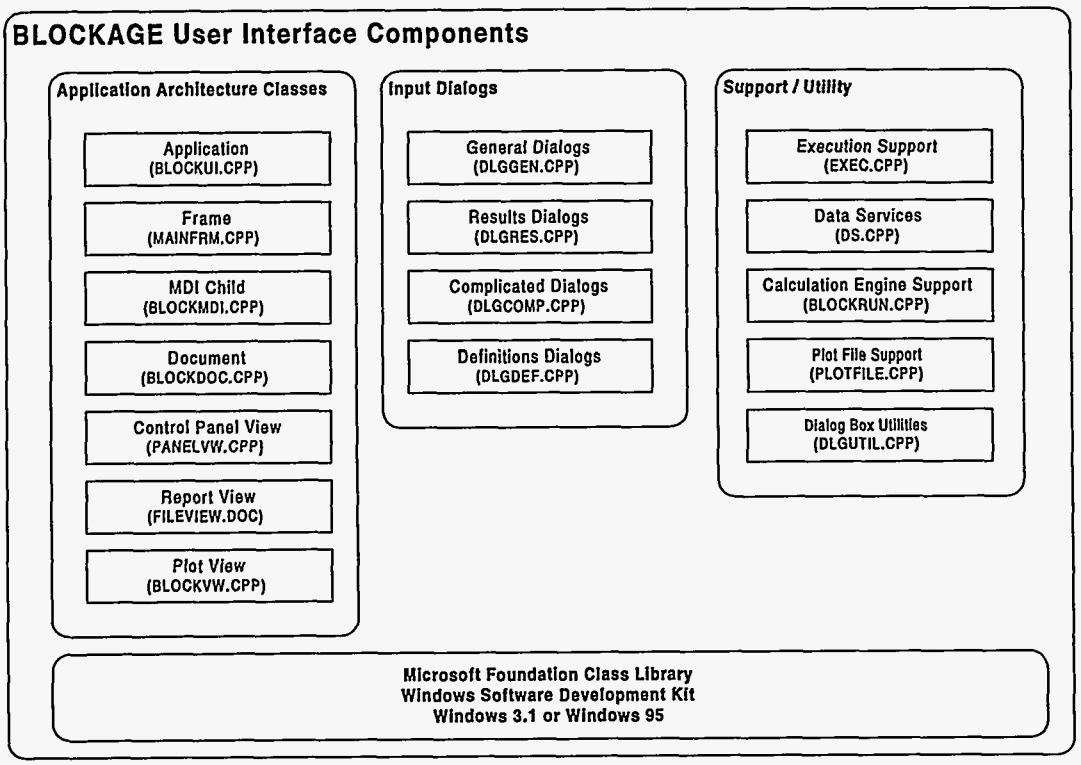


Figure 5-3: Components

MDI Child Contains the BlockMDIChild class. This class handles MDI children.

Document Contains the CBlockDoc class. This class handles the BLOCKAGE GUI document and most of the document related commands. The dialog boxes for selecting the model types and running the calculation engine are also in this component.

Control Panel View Contains the CPanelView class. This class handles the control panel view.

Report View Contains the CFileView and CTextFile classes. These classes handle report views.

Plot View Contains the CBlockView class. This class handles plot views.

Input Dialog Boxes

The major portion of the BLOCKAGE GUI code space is devoted to the 35 dialogs required to display and edit the input parameters and generate results. All of these dialog boxes have an extremely similar format due to the inherent structure of CDialog and the common support routines provided by the BLOCKAGE GUI.

Dialog Box Utilities Due to the number of dialog boxes, they are divided into four groups as listed below.

Definition Dialogs Support the Input->Definitions dialogs.

Complicated Dialogs Supports the two most complicated BLOCKAGE GUI dialogs, the Input->Event Scenario and Input->Debris dialogs.

Results Dialogs Supports the Results->Time-Based and Scale dialogs

General Dialogs Supports the input parameter dialogs not handled in the previous dialog groups.

Support/Utility

The support and utility components provide general purposes services that are used by other BLOCKAGE GUI components.

Execution Support Supports executing external applications and tracking their execution state. These routines are used to launch the calculation engine and wait for it to complete.

Data Services Supports the allocation and deallocation of the main BLOCKAGE GUI dynamic data structures.

Calculation Engine Support Supports running the calculation engine. This includes checking for sufficient disk space for output files, building the event scenario master table, and generating the BLK files.

Plot File Support Supports reading data from the PLOT.OUT file.

Dialog Utilities Supports the BLOCKAGE GUI dialogs. This component consists of a great many small routines to center dialogs, load fields, validate fields, format fields, and so forth

5.2 Component Descriptions

This section summarizes the routines that make up each of the components. This material is not designed to replace the in-line documentation in the source files of each component. Rather, the intention is to provide a high-level overview that will allow a programmer to see the general structure of BLOCKAGE GUI. Many of the routines listed below where generated by the Microsoft Foundation Class Application or Class Wizard and minor modifications made to their existing behavior. Since these routines are well documented in the Microsoft documentation, they are not described here. Instead, the abbreviation "MFC" appears in the description field.

5.2.1 Application Architecture Classes

5.2.1.1 Application

The application class component consists of the BLOCKUI.CPP and BLOCKUI.H files. This component is the main application level class for BLOCKAGE GUI. These routines match the required MFC organization.

Routine	Description
CBlockuiApp::CBlockuiApp	BlockuiApp constructor
CBlockuiApp::~CBlockuiApp	BlockuiApp destructor
CBlockuiApp::InitApplication	MFC
CBlockuiApp::InitInstance	MFC
CBlockuiApp::FirstInstance	MFC
CBlockuiApp::OnFileNew	File New command handler
CBlockuiApp::OnFileOpen	File Open command handler
CBlockuiApp::OpenDocumentFile	Open Document handler
CBlockuiApp::OnBlockageCloseDocument	Close Document handler
CBlockuiApp::OnBlockageUpdateAllViews	Updates all file views based on file existence, removes any error view, and resets results available buttons.
CBlockuiApp::OnHelpSearch	Handles Help->Search command.
GetTBVField	Retrieves a field from a Time Based Variable string.
HourGlassOn	Turns the hourglass on and increments the hourglass count.
HourGlassOff	Decrements the hourglass count and turns off the hourglass if the count is zero.
HourGlassOffNOW	Turns the hourglass off regardless of the hourglass count and sets the count to zero.

5.2.1.2 Frame

The main MDI frame class consists of the files MAINFRM.CPP and MAINFRM.H. This component handles the MDI frame.

Routine	Description
CMainFrame::CMainFrameMainframe	CMainFrame constructor
CMainFrame::~CMainFrameMainframe	CMainFrame destructor
CMainFrame::PreCreateWindowMainframe	MFC
CMainFrame::GetStatusBarHWndMainframe	MFC
CMainFrame::AssertValidMainframe	MFC
CMainFrame::DumpMainframe	MFC
CMainFrame::OnCreate	MFC
CMainFrame::OnMouseMove	MFC routine to handle mouse movement. Minor override to handle status bar.

5.2.1.3 MDI Child

The MDI child frame class consists of the files BLOCKMDI.CPP and BLOCKMDI.H. This component handles MDI children. These are the unchanged AppWizard generated routines except for the addition of comments and the extension of OnClose.

Routine	Description
BlockMDIChild::BlockMDIChild	BlockMDIChild constructor
BlockMDIChild::~BlockMDIChild	BlockMDIChild destructor
BlockMDIChild::PreCreateWindow	MFC
BlockMDIChild::OnClose	MFC routine to handle close of the MDI window. Standard MFC except if the window is the panel view, check for saving and send message to close the document.

5.2.1.4 Document

This BLOCKAGE GUI document class consists of the files BLOCKDOC.CPP and BLOCKDOC.H. This component handles the BLOCKAGE GUI document and most of the document related commands. The dialogs for selecting the model types and running the calculation engine are also in this component.

Core

Routine	Description
CBlockDoc::CBlockDoc	CBlockDoc constructor
CBlockDoc::~CBlockDoc	CBlockDoc destructor
CBlockDoc::AssertValid	MFC
CBlockDoc::Dump	MFC

Update Menus

The following routines support updating the menus to indicate command availability.

Routine	Description
CBlockDoc::OnUpdateInputBreakFrequency	Command availability for the Input->Break Frequencies command.
CBlockDoc::OnUpdateInputWeldstargets	Command availability for the Input->Breaks/Targets command.
CBlockDoc::OnUpdateResultsErrors	Command availability for the Results->Errors command.
CBlockDoc::OnUpdateResultsFrequency	Command availability for the Results->Frequency command.
CBlockDoc::OnUpdateResultsTargetdetail	Command availability for the Results->Target Detail command.
CBlockDoc::OnUpdateResultsVolumesorted	Command availability for the Results->Volume Sorted command.
CBlockDoc::OnUpdateResultsWelddetail	Command availability for the Results->Break Detail command.
CBlockDoc::OnUpdateResultsWeldsummary	Command availability for the Results->Break Summary command.
CBlockDoc::OnUpdateResultsInputparameters	Command availability for the Results->Input Parameters command.
CBlockDoc::OnUpdateResultsTimebased	Command availability for the Results->Time-Based command.

General/Utility

Routine	Description
CBlockDoc::BlockSetModifiedFlag	Sets the flag indicating that the BLOCKAGE GUI model has been modified and needs to be saved.

New, Open, Save Document

Routine	Description
CBlockDoc::OnNewDocument	Handles creating a new document.
BlockDataInit	Initializes all BLOCKAGE GUI data variables with the default model.
BadFileMessage	Displays a bad file message and terminates the application.
CBlockDoc::FixPath	Changes the extension for the save file to ".BLK" This routine is used after a ".CDT" file has been imported.
ImportErrorMessage	Displays a CDT read error message and terminates the application.
CBlockDoc::OnOpenDocument	Handles MFC OnOpenDocument.
CBlockDoc::OnFileSave	Subclasses the OnFileSave class to perform a "save as" operation if the file was opened from a CDT file.
IsSep	Determines if the specified character is a separator.
CBlockDoc::ProtectedRead	Reads a block of data.
ProtectedReadChar	Reads a data character.
ReadToken	Reads a token from the input line.
ReadString	Reads a string from the input file, terminating at the EOL or a separator character not in single quotes.
ReadStringToEOL	Reads a string from the input file, terminating at the EOL.
ReadLong	Reads a long from the input file.
ReadWord	Reads a WORD from the input file.
ReadBool	Reads a BOOL from the input file.
ReadFloat	Reads a float from the input file.
ReadStringTableEnum	Reads an enumerated string value from the input file based on a table of enumerated values.
ReadStringArrayTableEnum	Reads an enumerated string value from the input file based on a table of pointers to enumerated values.
ReadFloatTableEnum	Reads an enumerated float value from the input file based on a table of enumerated float values.
ImportDataInit	Initializes all BLOCKAGE GUI data variables before reading a CDT file.
CBlockDoc::Serialize	MFC routine to serialize (open/save) document data structures.
CBlockDoc::OnSaveDocument	Routine to handle command to save document.
CBlockDoc::Import	Reads a model in Combined DaT (CDT) file format.

Input Menu

Routine	Description
CBlockDoc::OnInputECCS	Handles the Input->ECCS command.
CBlockDoc::OnInputBreakfrequencies	Handles the Input->Break Frequencies command.
CBlockDoc::OnInputConfiguration	Handles the Input->Configuration command.
CBlockDoc::OnInputEventScenarios	Handles the Input->Event Scenarios command.
CBlockDoc::OnInputOutput	Handles the Input->Output command.
CBlockDoc::OnInputWeldsTargets	Handles the Input->Break Targets command.
CBlockDoc::OnInputCorrelation	Handles the Input->Correlation command.
CBlockDoc::OnInputDebris	Handles the Input->Debris command.

Run Menu

CBlockDoc::OnRunAnalysis	Handles the Run->Analysis command

Results Menu

Routine	Description
CBlockDoc::OnResultsInputparameters	Handles the Results->Input Parameters command.
CBlockDoc::OnResultsErrors	Handles the Results->Errors command.
CBlockDoc::OnResultsWelddetail	Handles the Results->Break Detail command.
CBlockDoc::OnResultsTargetdetail	Handles the Results->Target Detail command.
CBlockDoc::OnResultsFrequency	Handles the Results->Frequency command.
CBlockDoc::OnResultsWeldsummary	Handles the Results->Break Summary command.
CBlockDoc::OnResultsVolumesorted	Handles the Results->Volume Sorted command.
CBlockDoc::OnResultsTimeBased	Handles the Results->Time-Based command.

Report and Plot Support

The following routines are used in generating report and plot output.

Routine	Description
strcatfloat	Append the string representing a floating point number to another string.
strcatcenter	Appends one string to another, but centers the new string within a certain length field.
PrintHeader	Print the header for a time based variable report page.
CBlockDoc::GenerateReport	Generate the time based variable report.
CBlockDoc::GeneratePlot	Generate a time based variable plot.
CBlockDoc::CreateOrActivateFrame	MFC routine to create or activate a frame.
CBlockDoc::UpdateFileViews	Updates all displayed file views. This routine is called after an analysis has been run or a new model file opened.
CBlockDoc::RemoveErrorViews	Locates and removes error views.
CBlockDoc::RemoveResultsViews	Closes all displayed views.
CBlockDoc::CreateFileView	Creates a new file view. If the view already exists, activate it.
CBlockDoc::RefreshInputView	If an input view is displayed, recreates and redisplays it.
CBlockDoc::SetResultsAvail	Sets variables indicating results file availability and update control panel buttons.
CBlockDoc::GetPrinterPageLength	Determines the printer page length.

Dialog New Model

Routine	Description
CNewModelDlg::CNewModelDlg	Routines to handle the New Model Dialog.
CNewModelDlg::DoDataExchange	MFC
CNewModelDlg::OnInitDialog	MFC
CNewModelDlg::OnOK	MFC

Dialog Run Analysis

Routine	Description
CDlgRun::CDlgRun	Routines to handle the Run Dialog
CDlgRun::DoDataExchange	MFC
CDlgRun::OnInitDialog	MFC
CDlgRun::OnTimer	Timer routine to check periodically to see if the analysis task has completed.
CDlgRun::OnOK	MFC

5.2.1.5 Control Panel View

The control panel view class consists of the files PANELVW.CPP and PANELVW.H. This component handles the control panel view. The CPanelView is derived from CForm. The form is used to display a dialog containing the control buttons.

Routine	Description
CPanelView::CPanelView	CPanelView constructor
CPanelView::~CPanelView	CPanelView destructor
CPanelView::DoDataExchange	MFC
CPanelView::MyGetDocument	Cover for GetDocument optimized for using in CPanelView..
CPanelView::OnSize	Handles WM_ONSIZE message.
CPanelView::OnInitDialog	MFC
CPanelView::OnInitialUpdate	Routine to resize window based on form size. This routine causes only the portion of the form with buttons to be visible.
CPanelView::SetButton	Enables or disables the indicated button.
CPanelView::OnButtonxx	Routines to receive and redirect button presses to commands.

5.2.1.6 Report View

The report view class consists of the files FILEVIEW.CPP and FILEVIEW.H. In addition to the CFileView itself, this component defines a CTextFile class used in reading report data from text files.

Framework

Routine	Description
CFileView::CFileView	CFileView constructor
CFileView::~CFileView	CFileView destructor
CFileView::AssertValid	MFC
CBlockDoc* CFileView::GetDocument	MFC
CFileView::Dump	MFC
CFileView::OnInitialUpdate	MFC

General

Routine	Description
CFileView::ReAssociate	Updates an existing view by re-reading the underlying text file and refreshing the view.
CFileView::OnSize	Handles the WM_SIZE message.
CFileView::HandleSize	Determines the client area attributes and configures scroll bars.
CFileView::OnDraw	Handling drawing file view.

Scrolling

Routine	Description
CFileView::OnHScroll	Handles the horizontal scroll bar messages. For large movements, the entire window is redrawn. For small movements, the window contents are physically scrolled, the new area is cleared, and the new characters are drawn.
CFileView::OnVScroll	Handles the vertical scroll bar messages. For large movements, the entire window is redrawn. For small movements, the window contents are physically scrolled, the new area is cleared, and the new characters are drawn.

CFileView::OnKeyDown	Handles scrolling the view using keyboard characters. The keyboard characters used in scrolling are detected and the appropriate scroll action is taken in either the horizontal or vertical scroll bar.
CFileView::DrawColumn	Draws the indicated column for each line being displayed. This routine is used during scrolling.
CFileView::DrawLine	Draws the indicated line. This routine is used during scrolling.
CFileView::ClearEdge	Utility routine to draw a filled rectangle at the end of the screen during scrolling.

Printing

Routine	Description
BOOL CFileView::OnPreparePrinting	MFC
CFileView::OnBeginPrinting	MFC
CFileView::OnPrepareDC	MFC
CFileView::OnEndPrinting	MFC
CFileView::OnPrint	MFC
BOOL CFileView::InitFont	Creates a 10 point font for the window and initialize sizing variables based on the font.
CFileView::DeInitFont	Releases the font object.

Save Results

Routine	Description
CFileView::OnUpdateResultsSaveResults	Command availability for Results->Save Results.
CFileView::OnResultsSaveResults	Handles the Results->Save Results command.

Text File

Routine	Description
CTextFile::CTextFile TextFile	CTextFile constructor
CTextFile::~CTextFile TextFile	CTextFile destructor
CTextFile::Associate	Associates a file with the CTextFile object. This involves creating a buffer, analyzing the file, and loading the first buffer.
CTextFile::LoadBuff	Loads the buffer so that the n'th line is the first line in the buffer.
CTextFile::GetLine	Get the n'th line from the file and copies it to the buffer.
CTextFile::GetState	Returns the current state of the CTextFile object. The most frequent use is to get error state if an error is encountered.

5.2.1.7 Plot View

The plot view component consists of the files BLOCKVW.CPP and BLOCKVW.H. This class handles drawing plot views to the screen and printer.

Routine	Description
CBlockView::CBlockView	CBlockView constructor
CBlockView::~CBlockView	CBlockView destructor
CBlockView::AssertValid	MFC
CBlockView::Dump	MFC
CBlockView::GetDocument	MFC

CBlockView::OnInitialUpdate	MFC
CBlockView::CalcExtents	This routine takes the window size and determines which plot elements should be drawn and where they should be drawn.
CBlockView::DrawLegendItem	Draws an item in the legend. An item consists of the leading line and the related text description.
CBlockView::DrawGraph	Main routine to draw the graph and all related items to the display or printer.
SelectLogTableEntry	Given a range, looks through the table of possible log tick graduations and selects the best.
CBlockView::DrawTickMarks	Draws the tick marks for the X and Y axes.
CBlockView::DrawTick	Draws an individual tick mark.
CBlockView::OnDraw	Standard OnDraw handling for graphs.
CBlockView::OnPreparePrinting	MFC
CBlockView::OnBeginPrinting	MFC
CBlockView::OnEndPrinting	MFC
CBlockView::OnUpdateEditScale	Command availability for Edit->Scale
CBlockView::OnEditScale	Handles the Edit->Scale command.
CBlockView::OnLButtonDblClk	Translate a left click in the client area of the graph to an implied Edit->Scale command.
CBlockView::ScalePoint	Scales a dataset point into logical coordinates.
CBlockView::ScaleMouse	Scales a mouse (client area) coordinate into a dataset coordinate.
CBlockView::StartDraw	The StartDraw and Draw functions are used to assemble a polyline for output. StartDraw sets an index into an array used to assemble the polyline.
CBlockView::Draw	The StartDraw and Draw functions are used to assemble a polyline for output. Draw takes a line segment and adds it to the ploy line.
UpdateVars	Utility routine used to optimize TryDraw calculations.
CBlockView::TryDraw	Takes a line segment, clips it to a bounding rectangle, and draws the line segment, if appropriate.
CBlockView::OnMouseMove	This routine accepts mouse coordinates and, if the coordinates are in the plot area, updates the status bar to contain the (x,y) coordinates in dataset space of the cursor.
Signdiff	Routine to determine if two floating point numbers differ in sign.
swap	Routine to swap a pair of floating point numbers.

5.2.2 Input Dialog Boxes

The major portion of the BLOCKAGE GUI code space is devoted to the 35 dialogs required to display and edit the input parameters and generate results. All of these dialog boxes have an extremely similar structure due to the inherent organization of CDialog and the common support routines provided by the BLOCKAGE GUI Dialog Utilities. The dialog box routines will be very familiar to any Windows™ C++ programmer. The general structure of the routines for a dialog box is described below as an introduction for a new developer. Please consult the Microsoft documentation for the details surrounding dialog box code structure.

Each dialog box code section starts with the dialog constructor and the DoDataExchange function. In BLOCKAGE GUI, neither of these routines performs any special processing - they are just the standard MFC routines. Following these routines is the message map data structure which MFC uses to map dialog box events into message handlers, see below.

The first code specific to a BLOCKAGE GUI dialog is the OnInitDialog routine. This routine loads the

controls in the dialog box. This consists of loading edit fields, setting edit field limits, setting radio buttons, initializing drop down lists, and so forth. For list boxes, the OnInitDialog code calls a BLOCKAGE GUI routine named LoadLB. This routine requires a callback function, usually named "GetLB_<the list box name>." This routine returns each line in the list box and NULL if the last line has been reached.

As the user works with the dialog box, he enters data and changes controls. This causes control messages to be generated. The messages set up in the aforementioned message map data structure are captured and directed to the message handling routines. All message handling routines are named after the controls in the dialog and have the form "::On<control/action>". For example, ::OnButton1 or ::OnSelchangeCombo1. The code for these handlers has the standard MFC format for the message involved.

One especially important message handling routine is the ::OnOK routine that processes the OK button press. When this routine is called, it checks all of the parameters in the dialog box for validity and generates an error message if there is a problem. If all fields are consistent, the ::OnOK routine unloads the dialog box contents into the proper data structures and terminates so as to close the dialog box.
Due to the number of dialog boxes, they are divided into four groups as listed below. Almost all of the top level dialog boxes use one or more sub or child dialog boxes to support editing fields in the main dialog.

5.2.2.1 Definition Dialog Boxes

The definitions related dialog boxes are contained in the DLGDEF.CPP file.

Dialog	Description
CDefinitions1Dlg	Definitions dialog for break database defined volumes
CDefinitions2Dlg	Definitions dialog for user defined volumes
CDefinitionsBreakLocationDlg	Definitions Break Locations subdialog
CDefinitionsBreakTypeDlg	Definitions Break Type subdialog
CDefinitionsSystemIDDlg	Definitions System Identifier subdialog
CDefinitionsBreakDiameterDlg	Definitions Break Diameter subdialog
CDefinitionsDestructionRegionDlg	Definitions Destruction Region subdialog

5.2.2.2 Complicated Dialog Boxes

The complicated dialog boxes are contained in the DLGCOMP.CPP file. The "complicated" dialog boxes are the Event Scenario and Debris dialogs.

Dialog	Description
CEventScenarioDlg	Event Scenario dialog
CEventScenarioInterpolationDlg	Event Scenario Interpolation option subdialog
CEventScenarioVariableDlg	Event Scenario Variable subdialog
CDebrisDlg	Debris dialog
CDebrisAttributesDlg	Debris attributes subdialog
CDebrisAttribLD	Debris Destruction region subdialog
CDebrisAttribTransport	Debris Transport fraction subdialog

5.2.2.3 Results Dialog Boxes

The results related dialog boxes are contained in the DLGRES.CPP file. The results dialog boxes are the Time-Based Results and Scale dialog boxes.

Routine	Description
CScaleDlg	Scale dialog
CTimeBasedResultsDlg	Time-Based results dialog
GetUnits	Retrieves the units field out of a string.
GetFirstUnits	Retrieves the units for the first variable selected in a list box.
LoadPredefined	Load the settings for the indicated predefined report.

5.2.2.4 General Dialog Boxes

The remainder of the input dialog boxes are contained in the DLGGEN.CPP file.

Dialog	Description
CCorrelationDlg	Correlation dialog
CCoefficientsVarDlg	Correlation variable subdialog
CAboutDlg	About box dialog
CECCSDlg	ECCS dialog
CECCSModuleDlg	ECCS Module subdialog
CECCSPumpDlg	ECCS Pump subdialog
CControlPanelDlg	Control Panel dialog
COutputDlg	Output dialog
CBreakFrequencyDlg	Break Frequency dialog
CBreakFrequencyDiameterClassDlg	Break Frequency diameter class dialog
CBreakFrequencyBreakFrequencyDlg	Break Frequency break frequency subdialog
CWeldTargetsDlg	Break database, Breaks/Targets dialog
CBreakDlg	Break database, Breaks/Targets Break subdialog
CTargetDlg	Break database, Breaks/Targets Target subdialog
CBreakTargetTargetLengthDlg	Break database, Breaks/Target Target Length subdialog
CUserWeldTargetsDlg	User volume, Breaks/Targets dialog
CUserBreakDlg	User volume, Breaks/Targets Breaks subdialog
CUserTargetDlg	User volume, Breaks/Targets Target dialog

5.2.3. Support/Utility

The support and utility components provide general purposes services that are used by other BLOCKAGE GUI components.

5.2.3.1 Execution Support

Supports executing external applications and tracking their execution state. These routines are used to launch the calculation engine and wait for it to complete.

Routine	Description
ExecGetInfo	Finds the window and task associated with the indicated task instance
EnumWndProc	Procedure called by EnumWindows. Used to look through all windows for the one with the running task.
ExecIsAppRunning	Determines if a task is still running.

5.2.3.2 Data Services

Supports the allocation and deallocation of the main BLOCKAGE GUI dynamic data structures.

Routine	Description
DSInit	Allocate and lock main BLOCKAGE GUI data structures.
DSDeInit	Release storage for main BLOCKAGE GUI data structures.

5.2.3.3 Calculational Engine Support

The calculational engine support component consists of the BLOCKRUN.CPP and BLOCKRUN.H files. These routines support writing the BLKAGE files and running BLKAGE.

Routine	Description
_lwritestr	Utility routine to write a string to a file.
GetDiskSpace	Returns the amount of disk space available on the current drive.
HeaderAdjust	Adjusts the estimated number of lines in a file to account for a periodic page header.
EstimateSpaceRequired	Estimates the disk space required to store the output files for the model.
MTInsert	Inserts an entry in the master table. This routine only sets the times and step sizes. The interpolation routine sets the variables after the MT time steps have been built.
MasterTable	Builds the master table given the values specified in the Event Scenario dialog. This requires normalizing the time values across all varariables and interpolating the associated variable values.
FSZx	Converts a float to a string and returns the address of the string. Designed for use in wsprintf calls.
MakeszFloat	Converts a float to a string and standardizes the output format.
WriteModel	Creates the INPUT.DAT and WELD.DAT files for the model.
WriteInput	Creates the INPUT.OUT file holding the report describing the current model.

5.2.3.4 Plot File Support

The plot file support component consists of the PLOTFILE.CPP and PLOTFILE.H files. These routines assist in reading the PLOT.OUT file and locating variable fields therein.

Routine	Description
GetLineIndex	Returns the index of an indicated field in a PLOT.OUT file line.
PlotFileOpen	Opens the PLOT.OUT file for use by PlotFile routines.
PlotFileLocate	Looks through the file to find the section containing the indicated break id.
PlotFileGetNextID	Finds the next break ID in the PLOT.OUT file, if it exists.
PlotFileAdvance	Attempts to read the next line in the plot file.
PlotFileField	Gets the value from a particular field in a plot data line.
PlotFileClose	Closes the plot file.

5.2.3.5 Dialog Utilities

The dialog services component consists of the DLGUTIL.CPP and DLGUTIL.H files. These routines provide a wide variety of utility routines for use by dialog boxes.

General Dialog Support

Routine	Description
CenterDialog	Centers the dialog within the main window
SetStatus	Sets the status bar contents.
IsStatusReady	Returns an indication of whether "Ready" is displayed in the status bar.

Dialog Field Support

Routine	Description
LoadEditWithStr	Loads an edit control with a string.
LoadEditWithInt	Loads an edit control with an integer.
LoadEditWithFloat	Loads an edit control with a float.
ValidateIntWithEdit	Validates the contents of an edit control against two integer limits.
ValidateFloatWithEdit	Validates the contents of an edit control against two float limits.
ValidateStrWithEdit	Validates the contents of an edit control against a minimum string length.
LoadStrWithEdit	Loads a string with the contents of an edit field.
LoadIntWithEdit	Loads an integer with the contents of an edit field.
LoadFloatWithEdit	Loads a float with the contents of an edit field.
IsEditEmpty	Checks to see if an edit field is empty.
LoadEditWithNullStr	Initializes an edit field to a null string and sets limit string length.
LoadEditNullInt	Initializes an edit field to a null string and sets the limit string length for a number.
LoadEditWithNullFloat	Initializes an edit field to a null string and sets the limit string for a number.
SetTabs	Sets the tabs for a list box or edit field.
TrimString	Removes the white space from the beginning and ending of a string.
TrimNumber	Removes extraneous noise from a number and standardizes the format. This involves removing trailing zeros right of the decimal point, standardizing the scientific notation representation, and so forth.

Dialog List Box Support

Routine	Description
LoadLB	Loads a listbox with entries using a callback function.
LoadLBString	Loads a resource string into a list box.
LoadCBString	Loads a resource string into a combo box.
LBGetSel	Returns the selection in the indicated list box.
GetColumn	Modifies a string to only contain the nth column. Used during list box validation.
ValidateLBUnique	Checks a list box to make sure that the entries in the nth column are unique.
wcompare	Comparison functions for use as callbacks in qsort routine.
fcompare	Comparison functions for use as callbacks in qsort routine.
CopyRecord	Copies a list entry for the indicated list box from one position to another.
ShiftRecords	Shifts records associated with entries in the indicated list box.

HandleInsertButton	Handles the insert button associated with a list box.
RequireOneEntry	Displays an error message if the indicated list box doesn't have at least one entry.
HandleDeleteButton	Handles the insert button associated with a list box.
DeleteMessage	Generates a warning message indicating that the data in question can't be deleted since it is in use.

Interdialog Checks/Remapping

Routine	Description
CheckBreakDiameterInUse	Checks to see if a break diameter is in use.
CheckSystemIDInUse	Checks to see if a system id is in use.
CheckLocationInUse	Checks to see if a location is in use.
CheckWeldTypeInUse	Checks to see if a weld type is in use.
RemapBreakDiameters	This routine is called after break diameters have been modified, to find all "old" break diameters and change them for the new set.
RemapDebris	This routine is called after insulations have been modified to find all "old" insulations and change them for the new set.
RemapSystemIDs	This routine is called after system ids have been modified to find all "old" system ids and change them for the new set.
RemapLocations	This routine is called after locations have been modified to find all "old" location references and change them for the new set.
RemapWeldTypes	This routine is called after weld types have been modified to find all "old" weld type references and change them for the new set.
CheckUniqueAbbr	The function checks to make sure that the edit field abbreviation is a legitimate, unique abbreviation for the specified abbreviation list. If inserting, this means that the abbreviation must be unique. If modifying, the abbreviation must be unique OR match exactly the item we are modifying.

File Support

Routine	Description
CheckResultExistance	Checks to see if the specified file exists and, optionally, contains data.
BuildResultsFileName	Builds a result file name by catenating the path with the file name.
GetTempFileName	Returns an unused temp file name.
GetCurrentDir	Returns the current document path or, if none, the help file path.
StripFilename	Strips the filename off of a full pathname.

Debris Translations

Routine	Description
BuildDebrisTG	Fills in the debris management structure used when modifying debris types or displaying debris types in the target dialog.
DebrisEnumToTGEnum	Converts a debris enum to a origin TG enum.
TGEnumToDebrisEnum	Converts an origin TG enum to a straight debris enum.
DebrisIsTG	Checks a debris enum value to see if it is defined as source = origin drywell, target.

DebrisInUseAsTG	Checks a debris enum value to see if it is in use by a target as origin drywell, target.

5.2.4 Help

The following routines provide the integration of help.

BLOCKUI.CPP

Routine	Description
OnHelpSearch()	Displays the help search dialog.

DLGGEN.CPP

Routine	Description
OnHelpDisclaimer()	Displays the help disclaimer dialog.

All files containing dialog boxes

Routine	Description
<Dialog Class>::OnHelp()	Displays the help dialog for the appropriate dialog class.

5.2.5 Installation

BLOCKAGE GUI uses an installation utility package called InstallShield to support software installation. InstallShield consists of a number of utilities for interpreting installation scripts, compressing files, breaking compressed files into installation diskettes, and so forth. InstallShield is well documented in an extensive manual.

The BLOCKAGE GUI specific components of the installation consist of a batch file, BUILD.BAT, and an installation script, SETUP.RUL. The batch file is similar to a C++ make file. It contains all of the steps for building the set of installation disks. One of these steps it to submit the SETUP.RUL file to the InstallShield "compiler". InstallShield translates the instructions in SETUP.RUL into compiled commands that are executed by InstallShield during installation. The SETUP.RUL file contains instructions for displaying the "Welcome" screen, selecting the install directory, selecting the component to install, building the program groups, modifying the AUTOEXEC.BAT file and displaying the README file. The SETUP.RUL file is fully commented.

A second batch file, QBUILD.BAT, only runs the InstallShield compiler and does not compress the BLOCKAGE GUI files for distribution. QBUILD.BAT is only used for debugging SETUP.RUL.

5.3 Core Data Structures/File Formats

BLOCKAGE GUI uses several approaches to memory allocation to store its data structures. The core application classes, documents, views, text files, and dialog boxes, are allocated off of the global heap. This is usually done transparently to the BLOCKAGE GUI code itself by the application framework. The remaining data structures are allocated explicitly by BLOCKAGE GUI and are illustrated in Figure 5-4.

Permanent Data Structures

The bulk of BLOCKAGE GUI variables are allocated from the global heap during initialization and not deallocated until the application terminates. These structures are referred to as the permanent data structures and consist of the breaks, targets, and the BLOCKAGEVARS structure.

Temporary Data Structures

The remaining explicitly allocated BLOCKAGE GUI variables are allocated for a short time for a particular purpose. These structures are referred to as the temporary data structures and consist of dialog box "shadow" variables used during many dialogs to hold the working data and file view text buffers used to store report file information.

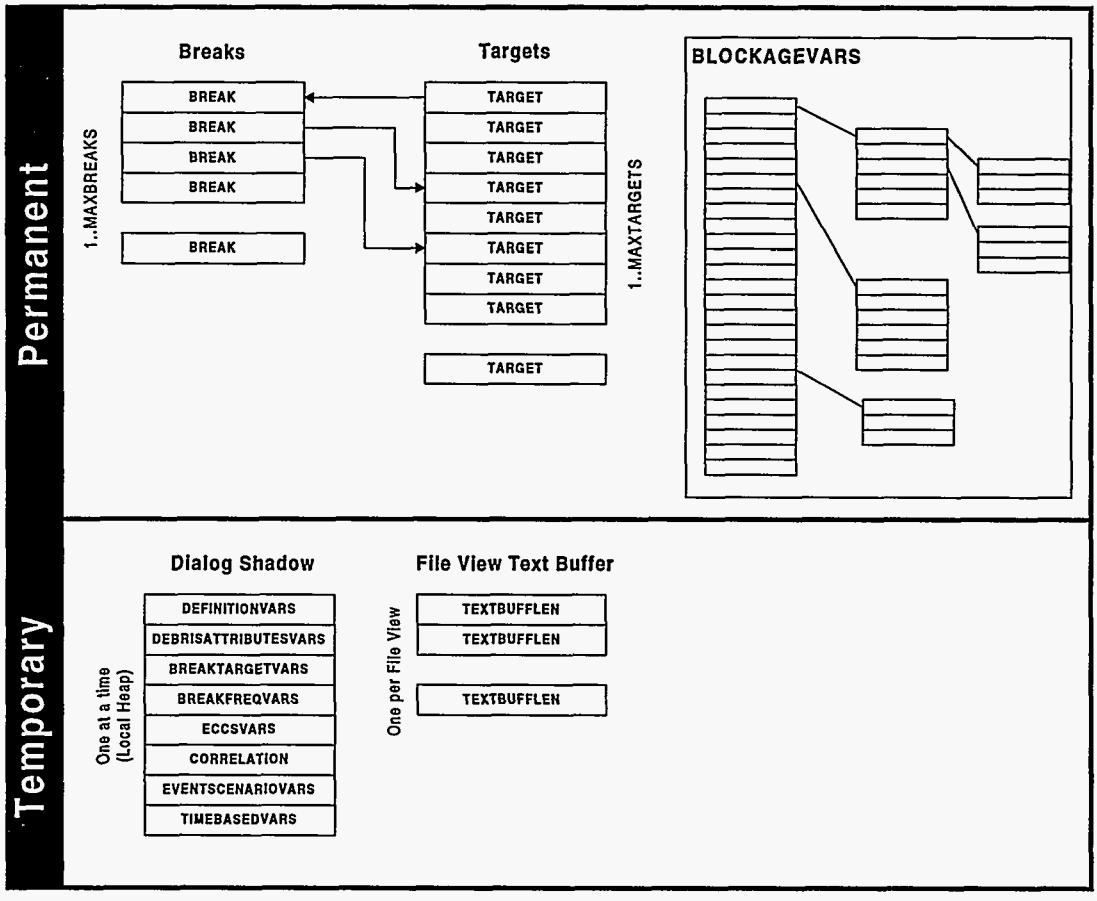


Figure 5-4: Data Structure Overview

Sections 3.1 through 3.3 describe memory usage for the key structures in more detail. Sections 3.4 and 3.5 describe the format of the BLOCKAGE GUI model and combined data files.

5.3.1 Break/Targets Structures

The breaks and targets are stored in two large dynamically allocated structures. During application initialization the DSInit routine is called. This data services routine allocates a structure for the breaks and one for the targets. Each is a large array of structures of either type BREAK or TARGET. These arrays are allocated for the maximum number of allowed breaks and targets. Global variables are used to store the address and handle of the memory allocated to the breaks and the targets.

As breaks are allocated, a counter keeps track of the next available break. When a break is removed, the elements in the break array are shifted to fill the deleted element. Similarly, when a break is added the elements of the array are shifted to make room for the new break. Each break has an index into the target array pointing to the first target associated with the break. The targets for the break are maintained sequentially in the target array. As targets are added and deleted, the targets must frequently be moved and the corresponding break indexes updated. Although this may seem like a lot of work, it is always tied to a user action so there is no noticeable delay. This approach has the strong advantage that the resulting data structure is very simple.

When the application terminates, DSDeInit is called. This routine deallocates the storage for the breaks and targets.

5.3.2 Main BLOCKAGE GUI Variables Structure

All of the data for BLOCKAGE GUI model, aside from the breaks and targets, is stored in a single

structure of type BLOCKAGEVARS. During application initialization the DSInit routine is called. This data services routine allocates global memory to hold the BLOCKAGEVARS structure. The BLOCKAGEVARS structure uses a number of substructures to store related or repeating elements. When allocated, all of these structures are dimensioned to their maximum size, thus eliminating the need for reallocation during execution. When the application terminates, DSDeInit is called. This routine releases the storage for BLOCKAGEVARS.

Due to the complex nature of the BLOCKAGE GUI variables, many fields of the BLOCKAGEVARS structure are related. For example, there is an array containing the permissible break diameters. Throughout the BLOCKAGEVARS structure, break diameters are referred to using the offset into the list of permissible break diameters. This type of relationship requires a number of routines that check for and maintain the consistency within the BLOCKAGEVARS structure.

5.3.3 Dialog Box Shadow Variables

Many dialog boxes in BLOCKAGE GUI use "shadow" variables during execution to facility an easy implementation of the cancel operation. When the dialog is created, BLOCKAGE GUI allocates space on the local heap for all of the variables that the dialog can affect and copies the data from the "live" variables in the BLOCKAGEVARS structure into the shadow variables structure. If space is not available for these variables, BLOCKAGE GUI displays a warning message and does not let the user enter the dialog. If the user CANCELs, BLOCKAGE GUI simply deallocates the shadow variable memory and terminates the dialog. If the user exits the dialog with OK, BLOCKAGE GUI copies the shadow variables back into the "live" variables and then terminates the dialog.

5.3.4 BLK File Format

The BLK file is a binary file and is read and written using the standard CArchive operators. For details on field lengths, please consult the CBlockDoc::Serialize routine. The format of the file is as follows:

(1) BLOCKAGE GUI File ID ("BLOCKAGE")
(2) BLOCKAGE GUI Version
(3) Size of BLOCKAGEVARS structure
(4) Size of BREAK structure
(5) Size of TARGET structure
(6) BLOCKAGEVARS Structure
(7) Number of Breaks
(8) BREAK Structures [NRC, 1..Number of Breaks]
(9) Number of Targets
(10) TARGET Structures [NRC, 1..Number of Targets]

5.3.5 CDT File Format

The CDT file format permits a model to be read into BLOCKAGE GUI in ASCII. CDT files are used primarily for debugging and testing. The format of the CDT file is as follows:

Line	Description
1:	CDT <# of break locations> <# of weld types>
2..n:	A valid INPUT.DAT file.
n+1..m:	A valid WELD.DAT file based on the previous INPUT.DAT file.
m+1:	ENDOFBREAKS

5.4. File Descriptions

5.4.1 Directory Structure

BLOCKAGE GUI source files are located in the \BLOCKUI directory and its subdirectories. Most files are located in the \BLOCKUI directory itself. The icons and version resource data is located in the RES subdirectory. Help files are located in the HELP subdirectory. The installation scripts and support files are in the INSTALL subdirectory. Finally, documentation files are located in the DOC subdirectory.
Many of the files described in this section are temporary or working files produced by the Visual C++ compiler. Although these files are not technically source files, an understanding of these files is valuable so they are described.

5.4.2 Visual C++ Source File Descriptions

Project files

File	Description
BLOCKUI.DEF	Module definition file. This file sets the program name, data and code segment attributes, HEAPSIZE and so forth.
BLOCKUI.MAK	BLOCKAGE GUI makefile. This file defines the relationship between sources files and sets compilation options.

C++ Language Source Files

".CPP" files contain C++ source code. ".H" files contain header information and are usually associated with a corresponding .CPP file.

File	Description
BLOCKUI.CPP,.H	BLOCKAGE GUI Application class
MAINFRM.CPP,.H	MDI frame class
BLOCKMDI.CPP,.H	MDI child class
BLOCKDOC.CPP,.H	BLOCKAGE GUI Document class
PANELVW.CPP,.H	Control panel View class
FILEVIEW.CPP,.H	Report file View and text file classes
BLOCKVW.CPP,.H	Plot file View class
DLGGEN.CPP	General dialog box classes
DLGRES.CPP	Results dialog box classes
DLGCOMP.CPP	Complicated dialog box classes (Event Scenario and Debris).
DLGDEF.CPP	Definitions dialog box classes
EXEC.CPP,.H	Routines to support external process execution and tracking.
DS.CPP,.H	Data Services routines
BLOCKRUN.CPP,.H	Calculation engine execution support routines.
PLOTFILE.CPP,.H	Routines to support reading variables from PLOT.OUT.
DLGUTIL.CPP,.H	Dialog box utility routines
STDAFX.CPP,.H	Precompiled header support files. Precompiled headers speed compilation.
DSTRUCT.H	Defines the BLOCKAGE GUI core data structures.

Resources

File	Description
RESOURCE.H	Defines the BLOCKAGE GUI resources.
BLOCKUI.RC	Resource file containing BLOCKAGE GUI menus, dialogs, icon definitions, strings, and so forth.
RES\BLOCKUI.RC2	Resource information for version resource.
RES\TOOLBAR.BMP	Toolbar bitmap. Provided for future use.
RES\BLOCKUI.ICO	BLOCKAGE GUI application icon
RES\BLOCKDOC.ICO	Control panel MDI child icon
RES\FILEVIEW.ICO	File view MDI child icon
RES\PLOTVIEW.ICO	Plot view MDI child icon

5.4.3 Visual C++ Generated Files

File	Description
*.OBJ	Object files corresponding to .CPP files.
BLOCKUI.RES	Compiled resource file
BLOCKUI.EXE	BLOCKAGE GUI executable
CURRENT.STS	Visual WorkBench current settings
BLOCKUI.APS	Binary version of resource file
BLOCKUI.CLW	Class Wizard status file
STDAFX.PCH	Precompiled header file
BLOCKUI.PDB	Program database file
BLOCKUI.MAP	Map file
BLOCKUI.SYM	Symbol definition file
BLOCKUI.VCW	Visual WorkBench status file

5.4.4 Installation Files

The following files are used in building the installation diskettes. All installation files reside in the INSTALL subdirectory.

File	Description
INSTALL\BUILD.BAT	Batch file to build the installation disk set.
INSTALL\QBUILD.BAT	"Quick" batch file to test installation rules.
INSTALL\SETUP.RUL	Installation rule file used by InstallShield to build installation files.
INSTALL\SETUP.INI	Initialization file to identify application.
INSTALL\README.TXT	Text file with last minute instructions for the user.
INSTALL\DISK1.ID	ID file to identify disk1
INSTALL\DISK2.ID	ID file to identify disk2
INSTALL\DISK3.ID	ID file to identify disk3
INSTALL\SETUP.PKG	Temporary file produced by InstallShield.
INSTALL\SETUP.LST	Temporary file produced by InstallShield

5.4.5 Help Files

File	Description
HELP\BLOCKUI.DBA	A ForeHelp file containing a portion of the help database.
HELP\BLOCKUI.DBB	A ForeHelp file containing a portion of the help database.
HELP\BLOCKUI.DBC	A ForeHelp file containing a portion of the help database.
HELP\BLOCKUI.DBD	A ForeHelp file containing a portion of the help database.
HELP\BLOCKUI.DBE	A ForeHelp file containing a portion of the help database.
HELP\BLOCKUI.DBF	A ForeHelp file containing a portion of the help database.
HELP\BLOCKUI.DBG	A ForeHelp file containing a portion of the help database.
HELP*.WMF	Windows metafiles embedded in the DB? files.
HELP\DOT_SM.BMP	A bitmap embedded in the DB? files.
HELP\MYDICT.SPL	Dictionary containing custom spellings used by BLOCKAGE UI help.
HELP\BLOCKUI.ERR	Error report from the help compilation.
HELP\BLOCKUI.HPJ	Intermediate help compilation file.
HELP\BLOCKUI.RTF	Intermediate help compilation file.
HELP\BLOCKUI.HLP	Final BLOCKAGE UI help file.
HELP*.LOG	Help compilation log file.

5.4.6 Documentation Files

File	Description
DOC\COVER.WP6	User's Manual Cover
DOC\TOC.WP6	User's Manual Table of Contents
DOC\SECT1.WP6	User's Manual Section 1
DOC\SECT2.WP6	User's Manual Section 2
DOC\SECT2PLT.WP6	User's Manual Section 2 Plot
DOC\SECT3.WP6	User's Manual Section 3
DOC\SECT4.WP6	User's Manual Section 4
DOC\SECT5.WP6	User's Manual Section 5
DOC\SECT6.WP6	User's Manual Section 6
DOC\APNDXA.WP6	User's Manual Appendix A
DOC\APNDXB.WP6	User's Manual Appendix B

6.0 Verification and Validation of Graphical User Interface

The BLOCKAGE GUI code has undergone an extensive and complete verification and validation process to ensure that the code performs as it was designed to. As much as possible, quality was built into the process of developing BLOCKAGE GUI. Care was taken to ensure that each stage of development was fully verified before proceeding to the next.

6.1 Overview of Code Quality Assurance

During the design and development of BLOCKAGE GUI, an extensive design review process was followed in developing the Requirements and the Functional Specification. The Functional Specification was kept continually up-to-date throughout the project, distributed and reviewed frequently, and provided as part of the final documentation. Paper prototypes, a Visual Basic prototype, and then a demonstration release were used to worth through design issues and confirm that all parties were coordinated in their concept of the application. The relatively straightforward nature of the BLOCKAGE GUI code allowed the operation of all input fields, and their translation into DAT files, to be 100% verified before release. The report generation and plot code was thoroughly tested with special emphasis on exceptional and boundary conditions. An alpha and beta release program was used to provide continuous UI testing during BLOCKAGE development and testing.

6.2 Code Development Code History

Table 6-1 outlines the major steps in the process of developing BLOCKAGE GUI.

Table 6-1: Major Steps in the Process of Developing BLOCKAGE GUI

Date	Item	QA Summary
May 22-24, 1995 Albuquerque, NM	Kickoff Meeting	• Discussed requirements in detail with NRC. • Reviewed paper prototypes and discussed design options.
June 28-30, 1995 Rockville, MD	Design Review	• Reviewed Functional Specification in depth. • Demonstrated Visual Basic prototype. • Discussed a complete set of prototyped main screens.
July 28, 1995	Telephone Design Review	• Resolved minor issues regarding Functional Specification.
August 8, 1995	Functional Specification Released	• Agreed upon Functional Specification released.
September 14, 1995	Demonstration Release	• A demonstration release of the control panel and major input dialogs was released to NRC for review.
October 8-10, 1995 Albuquerque, NM	Design Review	• Discussed details of demonstration release. • Made minor chantges to the Functional Specification.
March 15, 1996	Internal Alpha Release	• An alpha release of the software, with all major functions operational, released internally for review and testing.
March 28-29, 1996 Albuquerque, NM	Design Review	• Reviewed alpha release. • Final check of all field default values and range limits. • Discussed adding tick marks to plots and importing DAT format files.
April 16, 1996	Beta Software Release	• A beta software release was distributed to NRC for final review.
May 6, 1996	Internal Beta Software Release	• A beta software release was distributed for internal review and testing.
November 10, 1996	Final Software Release	• The final version of the software was distributed to NRC for release.

6.3 Verification of Coding

During the development of the Requirements and Functional Specification, extensive review was given to the documents by SEA, Software Edge, and the NRC. Between SEA and Software Edge, particular attention was paid to the Appendix defining the input fields and the format of the DAT input files to BLOCKAGE.

As each module was developed, notes were made regarding the special or exceptional cases that would need to be tested. During unit testing, these cases were tested. During the integration testing phase, a more formal methodology was followed to ensure that all input fields were being handled and translated properly. Test sheets were designed to list all aspects of dialog operation including:

- Navigation between dialogs,
- Tab order,
- Proper appearance on low and high resolution displays,
- Field format, alignment and width,
- Field Text operational width,
- Field validation and range checks and
- Proper operation of each control.

Each of the dialog boxes was checked and a test sheet completed. In addition to the completion of the test sheets, each input field was modified, a DAT file generated, and the DAT file checked to ensure that the field had been properly represented. Several minor problems were detected, corrected, and retested.

As an overall check, the BLOCKAGE base case models developed by SEA were entered and the corresponding DAT files checked, by a comparison program, to ensure they matched the SEA base cases. At many points during development, these bases cases were run through the BLOCKAGE engine and the results compared, using a comparison program, to ensure they matched the original DAT file base case results.

6.4 Validation of Coding

After the code was completed, the base cases described in Section 1.3 were reentered, BLOCKAGE was run, and the results compared against the results of running the original base case DAT files with BLOCKAGE. This regression testing was valuable as a final verification step and a basic validation test.

As detailed in Section 1.2, an extensive alpha and beta release program was undertaken. In the initial stages, the focus of the program was ensuring that all parties agreed on the presentation format and general workings of the code. During the latter stages, the beta program allowed SEA to use BLOCKAGE GUI during their final testing of BLOCKAGE. SEA used BLOCKAGE GUI to enter validation test cases and check them against other methods, as described in the BLOCKAGE verification and validation section. The alpha and beta program resulted in several errors being detected, corrected, and retested.

7.0 References

NUREG-0869, "USI A-43 Regulatory Analysis," A. W. Serkiz, US Nuclear Regulatory Commission, Rev. 1, October 1985.

NUREG/BR-0167, "Software Quality Assurance Program and Guidelines," February 1993.

NUREG/CR-3394, "Probabilistic Assessment of Recirculation Sump Blockage Due to Loss of Coolant Accidents: Containment Emergency Sump Performance USI A-43," Vols. 1 and 2, by J. J. Wysocki, Burns and Roe, Inc., published as Sandia National Laboratories Report No. SAND83-7116, July 1983.

NUREG/CR-6224, "Parametric Study of the Potential for BWR ECCS Strainer Blockage Due to LOCA Generated Debris," G. Zigler, J. Brideau, D.V. Rao, C. Shaffer, F. Souto, and W. Thomas, Science and Engineering Associates, Inc., SEA93-554-06-A:1, October 1995.

NUREG/CR-6367, "Experimental Study of Head Loss and Filtration for LOCA Debris," by D. V. Rao and F. J. Souto, Science and Engineering Associates, Inc., SEA 95-554-06-A:8, February 1996.

NUREG/CR-6370, "BLOCKAGE 2.5 User's Manual," by D. V. Rao, W. Bernahl, J. Brideau, C. Shaffer, and F. Souto, Science and Engineering Associates, Inc., SEA96-3104-A:3, December 1996.

Brideau, J., "BLOCKAGE 1.0 Qualification Test Report," Science and Engineering Associates, Inc., SEA93-554-06-A:3, December 1993.

G. Zigler, J. Brideau, D.V. Rao, N. Ruiz, C. Shaffer, W. Thomas, and R. Walsh, "Parametric Study of the Potential for BWR ECCS Strainer Blockage Due to LOCA Generated Debris," Preliminary DRAFT Report , SEA 93-554-06-A:1, January 1994.

MAPLE , Waterloo Maple Software, Waterloo Ontario, Canada, 1992.

MathCad, MathSoft, Inc, Cambridge, MA, 1995.

BWROG, Interim Report of the BWROG, ECCS Suction Strainer Committee, December 1994.

Brinkman, K. W. and P.W. Brady, "Results of Hydraulic Tests on ECCS Strainer Blockage and Material Transport in a BWR Suppression Pool," Pennsylvania Power & Light Company, EC-059-1006, Rev. 0, May 1994.

Appendix A

Functional Specification

Table of Contents

List of Figures

List of Tables

Preface

Attached is the BLOCKAGE User Interface (UI) Functional Specification. This document describes the operation of BLOCKAGE UI and was used as a guide to develop BLOCKAGE UI. As changes to operation were agreed upon, this document was modified to reflect the changes.

This document describes the functional characteristics of the BLOCKAGE UI software package. It is intended for use by the NRC, SEA, and Software Edge to define required system operation.

The BLOCKAGE UI functional specifications assume that the reader is familiar with the following:

- NUREG/CR-6224, "Parametric Study of the Potential for BWR ECCS Strainer Blockage Due to LOCA Generated Debris," Final Report, October 1994
- Use of the existing BLOCKAGE 2.3 code
- Windows™ 3.1 Interface Terminology

A.1 Overview

The BLOCKAGE UI provides front-end and back-end processing around the existing BLOCKAGE code. BLOCKAGE is a PC-DOS application to analyze BWR ECCS strainer blockage from LOCA-generated debris transported to the suppression pool.

BLOCKAGE reads two ASCII input files, weld.dat and input.dat, and produces numerous ASCII output files (see Figure A-1). Before the availability of BLOCKAGE UI, the user had to modify and view the input and output files using text editors or word processors. The BLOCKAGE input files have a complex format requiring that the user edit them with support documentation in hand. In short, BLOCKAGE is excellent at performing debris generation and transport modeling, but has a very basic user interface that is difficult for new users to learn.

BLOCKAGE UI is a Microsoft® Windows™ 3.1 application that provides an up-to-date user interface for BLOCKAGE, while using the existing BLOCKAGE code to perform the debris generation and transport analysis. BLOCKAGE UI uses drop down menus, forms based data entry and scrolling output windows to shield the user from the underlying BLOCKAGE input and output files. BLOCKAGE UI also uses BLOCKAGE generated output files to produce custom reports and plots. Finally, since BLOCKAGE UI is a Windows™ application, it supports a wide

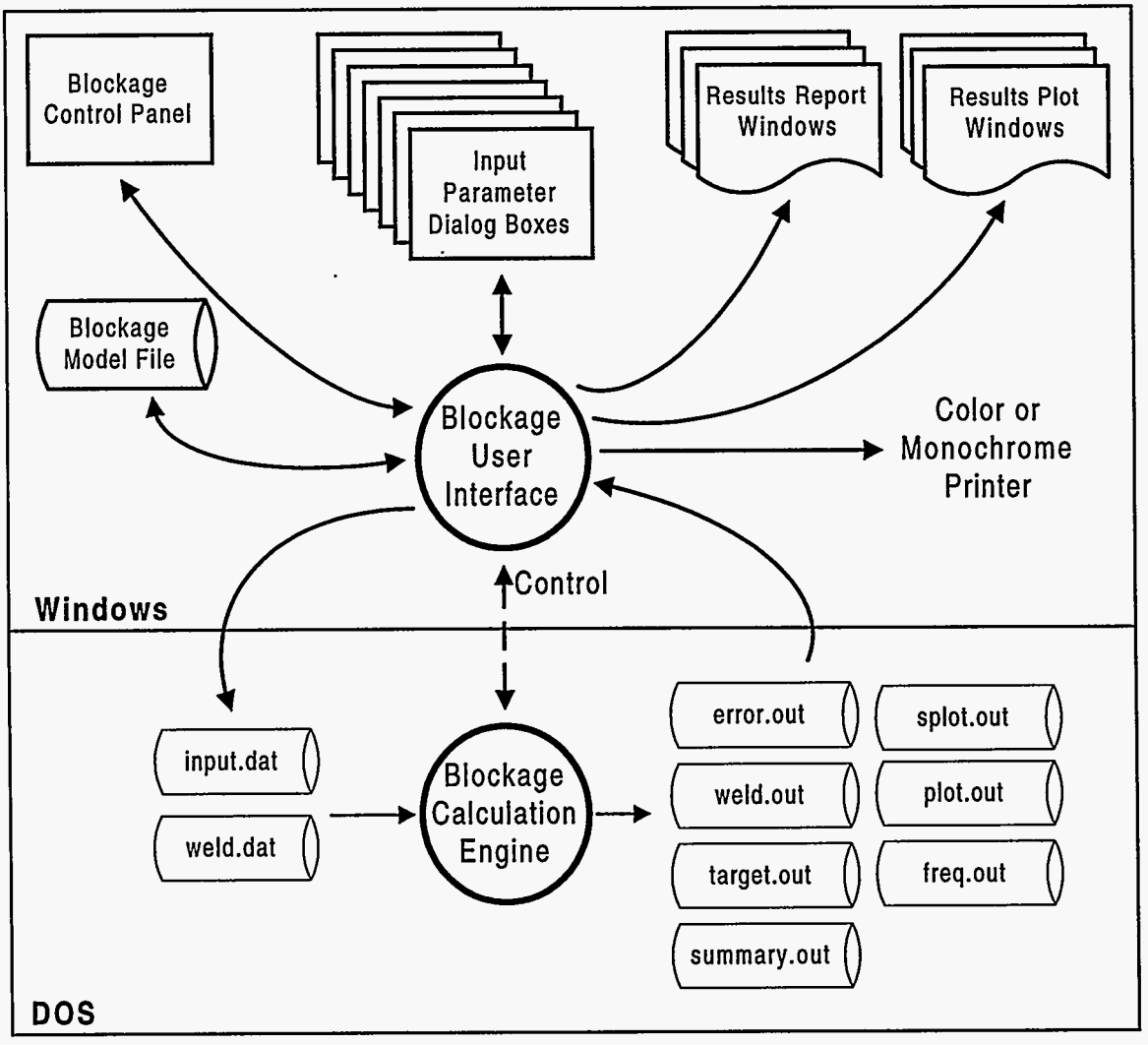


Figure A-1: BLOCKAGE UI/BLOCKAGE Overview

variety of display devices and printers. Overall, BLOCKAGE UI greatly simplifies the task of editing parameters and viewing results. This should lead to increased productivity and a reduced error rate.

A.1.1 System Requirements

BLOCKAGE UI runs on standard IBM-PC compatible computers. BLOCKAGE UI requires the following minimum configuration:

- 80386-based PC with at least 4 MB of RAM memory
- Hard disk with at least 2 Mb free
- Windows™ 3.1 supported color video card and associated monitor capable of displaying at least 16 colors at a resolution of 640x480 pixels
- Mouse or other pointing device
- 1.4 MB 3.5 inch or 1.2 MB 5.25 inch floppy drive
- Printer supported by Windows™ (optional)
- MS-DOS version 3.3 or later
- Microsoft Windows™ version 3.1 or later, running in enhanced mode

A.1.2 Window Types

BLOCKAGE UI uses a standard Windows™ application window. Within this main window, BLOCKAGE UI displays three types of child windows to represent BLOCKAGE data and interact with the user as indicated below.

BLOCKAGE UI Control Panel Window When the user opens a blockage model file, BLOCKAGE UI displays the BLOCKAGE UI control panel window. This window provides buttons that the user can select to view input parameters, run BLOCKAGE and open results windows.

Input Parameter Windows BLOCKAGE UI displays Windows™ dialog boxes to allow the user to view and edit BLOCKAGE input parameters. A variety of standard Windows™ controls are used to manage the data in the dialog boxes including: edit fields, radio buttons, check boxes, list boxes, and drop down lists. Once an input parameter dialog window is open, the user must finish working with the window and close it before opening other windows.

Results Windows BLOCKAGE UI displays results in report windows and plot windows. If a report is too wide or long to completely fit in a report window, BLOCKAGE UI provides scroll bars to allow the user to view the entire report. For plots, on the other hand, BLOCKAGE UI scales the image in a plot window so that the entire plot is visible.

The user may open several results windows simultaneously. BLOCKAGE UI uses a standard Windows™ interface style called the "Multiple Document Interface,"(MDI), to manage these windows. MDI provides window elements, such as close boxes, along with menu entries to allow the user to close, maximize, minimize, restore, resize, and arrange windows.
The BLOCKAGE UI control panel is the window associated with a blockage model. Closing the control panel automatically closes all other child windows. The user may have as many windows opens as he chooses, however a practical limit is four of five. Viewing multiple windows is especially valuable for comparing plots.

A.1.3 Menu Structure and Operation

BLOCKAGE UI commands are accessible through a standard Windows™ drop down menu. This menu follows Windows™ standards for organization and has the usual sections for managing files, editing objects, arranging windows and requesting help. BLOCKAGE UI provides additional menu commands to allow the user to edit input parameters, run BLOCKAGE and view results. All functions available though the BLOCKAGE UI control panel are also available using menu selections.

A.1.4 Status Bar

BLOCKAGE UI displays a status bar at the bottom of the main window. Normally, the status bar is blank. When the user is traversing the menu, BLOCKAGE UI displays a short description of the currently selected menu item in the status bar. When the user has a plot selected and moves the cursor across the plot area, BLOCKAGE UI displays the (x,y) coordinates of the cursor as an aid to examining interesting plot regions.

A.2 Model Files and Entering Parameters

Before running a BLOCKAGE analysis, the user must specify a fairly large number of parameters to describe the physical plant configuration and to control the BLOCKAGE code. These parameters are described in NUREG CR-6224. BLOCKAGE UI provides input forms to enter these parameters using Windows™ dialog boxes. BLOCKAGE UI organizes the input parameters into eight dialog boxes as described in Table A-1.

Table A-2 (located at the end of this appendix) lists the data fields associated with each of the input parameter dialogs and indicates their mapping to variable names in the BLOCKAGE weld.dat and input.dat files. Appendix B provides the exact format of the input.dat and weld.dat files.

A.2.1 General

To begin a new model, the user selects the New command from the menu and indicates whether BLOCKAGE should derive insulation volumes from the break database or directly from user input. BLOCKAGE UI then initializes a new set of model parameters which the user can then edit as described in the remainder of this section. This initial set of parameters is a generic BLOCKAGE model which could, if the user desired, be processed by the BLOCKAGE code. When the user has customized the model parameters to suit his requirements, he can save the parameters in a BLOCKAGE model file by selecting the Save command from the menu. To restore parameters, the user can read them from a BLOCKAGE model file by selecting the Open command from the menu.

The input parameter dialog boxes consist of standard Windows™ controls including edit fields, list boxes, drop down lists, check boxes and radio buttons. All controls operate in the standard Windows™ fashion. The user can access dialog box controls using either the keyboard or the mouse. Most dialog boxes have three buttons along the bottom: OK, Cancel, and Help. Choosing OK saves changes and exits the dialog. Choosing Cancel discards any changes and exits the dialog. Choosing Help displays a help window with text specific to the dialog box.

A.2.2 Dialog/Control Availability

Not all dialog boxes or controls are appropriate at all times. For example, if the user has decided to skip the probabilistic part of the analysis, the parameters in the Break Frequencies dialog are no longer needed. In this case, BLOCKAGE UI disables the "Break Frequencies" button on the BLOCKAGE UI control panel so that the user can not select it. A disabled button appears in a lighter shade of gray than a normal button. Input parameter fields are handled similarly. Most dialog boxes are available in all situations. Table A-3 indicates which dialog boxes are only available in certain states and the insulation volume source and probability options that determine their availability. Appendix E contains prototype screen shots for the dialog boxes listed below as well as the other major BLOCKAGE UI dialog boxes.

Table A-1: Dialog Boxes

Name	Description
Definitions	Overall plant and BLOCKAGE model configuration parameters (e.g., permissible break diameters, weld types, break locations)
Debris	Physical debris related attributes
Break/Targets	Breaks and their associated targets
Break Frequencies	Break frequency parameters
ECCS	Suppression pool parameters
Correlation	BLOCKAGE correlation selection and parameters
Event Scenario	Time-dependent BLOCKAGE function specifications
Output	BLOCKAGE output format parameters

Table A-3: Insulation Volume Source / Probability Analysis Option

Dialog	Break/Target Database Include Probability	Break/Target Database Skip Probability	User Specified Probability Not Avail
Definitions	Yes - Break Database Version	Yes - Break Database Version	Yes - User Volume Version
Break Frequencies	Yes	No	No

A.2.3 Parameter Validation

When the user selects the OK button in a dialog box, BLOCKAGE UI checks the input parameters in the dialog box for validity. If errors are detected, BLOCKAGE UI displays a message and requires the user to correct the problem before saving any changes and exiting the dialog box.

A.2.4 Numeric Values

The input parameter dialog boxes have edit fields for both integer and real numbers. Integers must be entered without commas or decimal points (e.g., "1" or "3124"). Real numbers may either be entered in standard notation or scientific notation (e.g., "12.345" or "1.2345E1"). In either case, BLOCKAGE UI supports 6 significant digits in the mantissa and 2 in the ordinate. Both mantissa and ordinate may be signed.

A.2.5 Time Dependent Variable Table

The mapping from the input parameter dialog boxes to the BLOCKAGE input files weld.dat and input.dat, for the most part, is quite straightforward. One exception to this is the handling of the time-dependent variable data. The input.dat file requires a single, master time step/variable table that combines all the time dependent variables and time steps into one table. Constructing this table manually requires the user to consider all of the time variables simultaneously and calculate intermediate data points for variables based on the time steps. This can be a lengthy and error-prone procedure. BLOCKAGE UI relieves the user of this burden by accepting the data for each of the variables and the time steps independently and combining them into a master table automatically.

A.2.5 Combined Dat File Format

BLOCKAGE UI supports importing ASCII model files called "Combined DaT Files" or CDT files. These files are based on the WELD.DAT and INPUT.DAT files which are input to BLKAGE. The CDT file capability is intended for use primarily in testing and debuging.

A.3 Running BLOCKAGE

After configuring the input parameters, the user runs the BLOCKAGE code by either clicking on the appropriate button in the BLOCKAGE UI control panel or selecting the appropriate menu item. In response, BLOCKAGE UI builds the necessary input files, checks to make sure than enough disk space exists for the output files, and executes the BLOCKAGE code. When processing is complete, BLOCKAGE UI either displays a message indicating that no errors were encountered or displays a list of the errors.

A.4 Viewing Results

After BLOCKAGE has run successfully, BLOCKAGE UI allows the user to view the analysis results in windows on the screen. BLOCKAGE UI can display both text based reports and graphical plots. If a text report exceeds the size of a window, the user can scroll the window horizontally and vertically. BLOCKAGE UI automatically scales plots to the size of their windows.

Until BLOCKAGE has been run the first time for a model file, there are no results to display. In this situation, BLOCKAGE UI disables the buttons and menus controlling these results windows to indicate this. The one exception is the input parameters report which is always available.

A.4.1 General Reports

The majority of the output from BLOCKAGE is in the form of text reports. The following reports are available:

Name	Description
Input Parameters	All input parameters for the model
Errors	Errors, if any, encountered during the BLOCKAGE analysis
Break Detail	Detailed break reports for selected breaks
Target Detail	Break/target details for selected breaks
Frequency	Break and strainer blockage summary
Break Summary	Break report summary by break
Volume Sorted	Summary report sorted by volume transported

To view a report, the user can either click on the appropriate button on the BLOCKAGE UI control panel or choose the appropriate command from the menu. If a report is already open, selecting the report again restores the window and brings it to the foreground. The user can close a report by either clicking on the close button for the window or choosing the appropriate command from the menu. The user can print reports by selecting the appropriate command from the menu. The user can save reports by selecting the appropriate command from the menu.

A.4.2 Time-Based Plots and Reports

The BLOCKAGE code produces a large file, named plot.out, which contains the values for more than 70 variables at each time step during the analysis (see Appendix C for a list of these variables). The format of plot.out was designed to be imported into spreadsheets. The extreme width of plot.out makes it difficult to view directly and the large number of time steps makes trends difficult to discern.

BLOCKAGE UI allows the user to view subsets of this time-based data and to plot up to six variables on a common axis against time. To open a new window with a time-based report or plot, the user chooses the appropriate button from the BLOCKAGE UI control panel or selects the appropriate menu item. BLOCKAGE UI then displays a dialog box where the user can describe the result window contents he desires. BLOCKAGE UI provides controls to select the following:

Format The user must select whether to view the output in report or plot format.

Title The title to appear at the top of the plot or report.

Break ID The break identifier of the break data to display. The user selects one break from a list of all breaks available.

Variables The user selects the variables to be included in the output from a list of the more than 50 available variables (see Appendix C). The user may select up to ten variables for a report and up to six variables for a plot. All plot variables will share the same axis and therefore must be of the same units.

Scale For plots, the user can select the initial scaling to use. Scaling is described in Section A.2.2.

Predefined BLOCKAGE UI stores 10 predefined report/plot formats which specify values for all of the controls listed above, except Break ID. If the user desires to view one of the predefined reports or plots, he selects it from a list and the other controls in the dialog box change to reflect the new settings. The user may also select a predefined format as a starting point and then modify the selection.

A.4.2.1 Plot Format

The plots generated by BLOCKAGE UI are intended for quick, exploratory data analysis -- not for presentation quality output. BLOCKAGE UI plots have the following characteristics:

- A title appearing at the top of the plot.
- A y-axis labeled in units of the variable(s) of interest with value labels and tick marks.
- An x-axis labeled in units of time with value labels and tick marks.

- Each variable displayed as a line connecting all data points. Each line has an automatically assigned, distinct, color and line pattern.
- A legend, below the x-axis, indicating the color and line pattern associated with each variable.

The plot format may change based on the size of the window. In particular, for very small windows BLOCKAGE UI may drop the legend.

A.4.2.2 Scaling

Plot scaling may either be automatic or manual for the x and y axes independently. When the user selects automatic scaling, BLOCKAGE UI selects appropriate limits automatically. When the user selects manual scaling, he must specify a start value and a stop value. The user can also specify whether each axis should be plotted in a linear or logarithmic fashion. This selection is available for both automatic and manual scaled axes. Note: Changing the log/linear setting for an automatically scaled axis may change the scale values selected for the axis.

A.4.2.3 Cursor Usage During Plot Viewing

As the user moves the cursor over the plot area of a selected plot, BLOCKAGE UI displays the (x,y) coordinates of the cursor in the status bar. This allows the user to easily determine variable values at interesting locations.

A.4.2.4 Printing

BLOCKAGE UI supports printing plots to any Windows™ supported printer. If the printer is color capable, the output will be in color. If the printer is a monochrome printer, BLOCKAGE UI draws all variables in black and the variables can only be identified by their line patterns. Only one plot is printed per page.

A.5. Miscellaneous Functions/Topics

A.5.1 Online Help

BLOCKAGE UI provides the user access to online help. Help is available using the standard menu items for help or through the Help buttons in each input parameter dialog box.

General Help The general help provides access to the BLOCKAGE UI User's Manual. Diagrams are not available in the online help.

Dialog Box Help When viewing an input parameter dialog box, the user may choose the Help button to receive help on the function of the dialog box and the particular controls and fields in the dialog box.

A.5.2 About Box

BLOCKAGE UI provides a standard Windows™ "About..." dialog box for the display of the BLOCKAGE UI version, the BLOCKAGE version, and contact points to receive more information regarding the software.

A.5.3 Spreadsheet Compatibility

The BLOCKAGE plot.out file was designed to be spreadsheet compatible. BLOCKAGE UI provides no additional support for formatting data into spreadsheet compatible formats. It is anticipated that if sensitivity analysis support is added to BLOCKAGE UI, additional spreadsheet compatible files would be produced.

A.5.4 Sensitivity Analysis

BLOCKAGE UI does not provide access to the sensitivity analysis functions of BLOCKAGE. Access to these functions may be provided in a future version.

Table A-2: BLOCKAGE UI Fields Summary

Definitions (Volume Method= Break Database)	Type	Limits	Variable	Default
Case Name	a40		runid	Default
Break/Target Database Name	a40		probid	Default
Skip Probabilistic Analysis	enum of Skip, Don't Skip	enum	iprob	Don't Skip
Destruction Regions				
Array of L/D	real	0<x<20; Increasing	ld[1..maxld]	3, 5, 7
Break Diameters				
Array of Break Diameters	real	0<x<100; strictly increasing 1<=npwd<=maxpwd	pwds[1..npwd]	1,2,4,10,16,18,22
System Identifiers				
Array of IDs	a6	Distinct, not null	Reference Only	RECIRC, MSTEAM, FEEDW, HPCI, RHR
Array of Description	a26	not null, 1<=nsys<=maxsys	nsys, systbl[1..nsys]	Recirculation Loop, Main Steam, Feedwater, HPCI Steam Line, RHR Injection Lines
Break Locations				
Array of Abbreviation	a2	Distinct, not null 1<=npwl<=maxpwl	pwls[1..npwl]	H,M,L
Array of Description	a40	not null, 1<=npwl<=maxpwl	lcdesc[1..npwl]	Above 776 ft Grating, Between 757/776 Gratings, Below 757 ft Grating
Weld Types				
Array of Abbreviation	a2	Distinct 1<=npwt<=maxpwt	pwts[1..npwt]	S1,S2,S3 - Stainless Steel Type x C1,C2,C3 - Carbon Steel Type x
Array of Description	a40		Reference Only	None

Table A-2: BLOCKAGE UI Fields Summary - continued

Definitions	Type	Limits	Variable	Default
(Volume Method= User Input)				
Case Name	a40		runid	Default
User Volume Dataset	a40		probid	Default
Break Diameters				
Array of Break Diameters	real	0<x<100; strictly increasing 1<=npwd<=maxpwd	pwds[1..npwd]	1,2,4,10,16,18,22
Break Locations				
Array of Abbreviation	a2	Distinct, not null 1<=npwl<=maxpwl	pwls[1..npwl]	H,M,L
Array of Description	a40	not null	lcdesc[1..npwl]	Above 776 ft Grating, Between 757/776 Gratings, Below 757 ft Grating

Table A-2: BLOCKAGE UI Fields Summary - continued

Debris - Main Dialog	Type	Limits	Variable
Filtration			
Cake Thickness for Peak Filtration (ft)	real	0<=x<=3	tfilt
Interpolation	enum of Step Linear Exponential	enum	ifilt
Debris			
Array of Abbr	a2	Distinct, not null 1<=ntypes<=maxdeb	dname[1..ntypes]
Array of Name	a20	not null	dname[1..ntypes]
Array of Class	enum of Fibrous, Metalic, Particulate, Ignore	enum	dclass[1..ntypes]
Array of Origin	enum of Drywell (targets), Drywell (Non-targets), Wetwell	enum	dloc[1..ntypes]
Array of Volume (cu ft)	real	0<=x<=250	volume[1..ntypes] not available if dloc[i]=Drywell (targets)
Array of Number of Settling Velocity Groups	int	1<=x<=maxvel	ngroup[1..ntypes]

1) For defaults, see report following Debris dialogs.

Table A-2: BLOCKAGE UI Fields Summary - continued

Debris - Details (for ith entry)	Type	Limits	Variable
Abbr	a2	Distinct, not null	dname[i]
Name	a20	not null	dname[i]
Class	enum of Fibrous, Metalic, Particulate, Ignore	enum	dclass[i]
Origin	enum of Drywell (targets), Drywell (Non-targets), Wetwell	enum	dloc[i]
Volume (cu ft)	real	0<=x<=250	volume[i] not available if dloc[i]=Drywell (targ)
Fabricated Density (lb /cu ft)	real	.5<=x<=1250	dfab[i]
Packing Density (lb/ cu ft)	real	.5<=x<=1250	drub[i]
Material Density (lb/ cu ft)	real	.5<=x<=1250	dmat[i]
Specific Surface Area (sq ft/cu ft)	real	10<=x<=1e7	ssa[i]
Transport Fractions, iff origin= Drywell (targets)			
Array of Location	-	Ref Only	All
Array of Blowdown	real	0<=x<=1	tfract[i,1]
Array of Washdown	real	0<=x<=1	tlate[i,1..npwl]
Transport Fractions, iff origin= Drywell (non-targets)			
Array of Location	a2	Ref Only	Ref pwls[1..npwl]
Array of Blowdown	real	0<=x<=1	tfract[i,1..npwl]
Array of Washdown	real	0<=x<=1	tlate[i,1..npwl]
Destruction Fractions			
Array of Region	real	Ref Only	Ref ld[1..maxld]
Array of Fraction	real	0<=x<=1	dfract[1..maxld]
Velocity Groups			
Array of Velocity Group	a6	Distinct within a debris type	vname[i,1.. ngroup[1..ntypes]]
Array of Filtration Strainer Eff Initial	real	0<=x<=1	efilt[i,1.. ngroup[1..ntypes],1]
Array of Filtration Strainer Eff Peak	real	0<=x<=1	efilt[i,1.. ngroup[1..ntypes],2]
Array of Retention Factor	real	0<=x<=1	eheld[i,1.. ngroup[1..ntypes]]
Array of Settling Velocity (ft/sec)	real	0<=x<=10	vterm[i,1.. ngroup[1..ntypes]]
Array of Distribution Fraction	real	0<=x<=1; for all i (abs(sum(fdisk[i, all j]-1))<=.00001)	vname[1.. ngroup[1..ntypes]]

1) For defaults, see report following Debris dialogs.

Table A-2: BLOCKAGE UI Fields Summary - continued

```
==========================================================================================
DEBRIS DEFAULTS
==========================================================================================

Cake Thickness for Peak Filtration: 2.08E-2 ft
Interpolation: Linear

------------------------------------------
Abbreviation: NK
Name        : Nukon
Class       : Fibrous
Origin:     : Drywell (Target)
```

Volume cu ft	Fabricated Density lb/cu ft	Rubble Density lb/cu ft	Material Density lb/cu ft	Specific Surf Area sq ft/cu ft
n.a.	2.4	2.4	175.	171000

Transport Fractions

Location	Blowdown	Washdown
H	0.15	0.1176
M	0.35	0.2308
L	0.45	0.5455

Destruction Fractions

Region	Fraction
3.	0.75
5.	0.6
7.	0.4

Velocity Group	Filtration Strainer Eff Initial	Filtration Strainer Eff Peak	Retention Factor	Settling Velocity ft/sec	Distribution Fraction
NK-1	1.	1.	1.	6.5306E-4	0.43067
NK-2	1.	1.	1.	1.6443E-3	0.14922
NK-3	1.	1.	1.	2.6061E-3	0.11011
NK-4	1.	1,	1.	4.1303E-3	8.125E-2
NK-5	1.	1.	1.	6.5461E-3	5.995E-2
NK-6	1.	1.	1.	1.0375E-2	4.424E-2
NK-7	1.	1.	1.	1.6443E-2	3.265E-2
NK-8	1.	1.	1.	2.6061E-2	2.409E-2
NK-9	1.	1.	1.	4.1303E-2	1.778E-2
NK-10	1.	1.	1.	6.5461E-2	1.312E-2
NK-11	1.	1.	1.	0.10375	9.68E-3
NK-12	1.	1.	1.	0.24606	2.725E-2

Table A-2: BLOCKAGE UI Fields Summary - continued

```
-------------------------------------------
Abbreviation: PT
Name        : Paint
Class       : Particulate
Origin:     : Drywell (Non-Target)
```

Volume cu ft	Fabricated Density lb/cu ft	Rubble Density lb/cu ft	Material Density lb/cu ft	Specific Surf Area sq ft/cu ft
0.8	142.	65.	142.	20000.

Transport Fractions

Location	Blowdown	Washdown
All	0.5	0.75

Velocity Group	Filtration Strainer Eff Initial	Filtration Strainer Eff Peak	Retention Factor	Settling Velocity ft/sec	Distribution Fraction
PT-1	0.	0.5	0.5	1.E-2	1.

```
-------------------------------------------
Abbreviation: SD
Name        : Sludge
Class       : Particulate
Origin:     : Wetwell
```

Volume cu ft	Fabricated Density lb/cu ft	Rubble Density lb/cu ft	Material Density lb/cu ft	Specific Surf Area sq ft/cu ft
2.6	324.	65.	324.	150000

Velocity Group	Filtration Strainer Eff Initial	Filtration Strainer Eff Peak	Retention Factor	Settling Velocity ft/sec	Distribution Fraction
SD-1	0.	0.5	0.5	6.5306E-4	0.20897
SD-2	0.	0.5	0.5	1.6443E-3	4.658E-2
SD-3	0.	0.5	0.5	2.6061E-3	5.477E-2
SD-4	0.	0.5	0.5	4.1303E-3	6.325E-2
SD-5	0.	0.5	0.5	6.5461E-3	7.145E-2
SD-6	0.	0.5	0.5	1.0375E-2	7.848E-2
SD-7	0.	0.5	0.5	1.6443E-2	8.322E-2
SD-8	0.	0.5	0.5	2.6061E-2	8.442E-2
SD-9	0.	0.5	0.5	4.1303E-2	8.101E-2
SD-10	0.	0.5	0.5	6.5461E-2	7.249E-2
SD-11	0.	0.5	0.5	0.10375	5.943E-2
SD-12	0.	0.5	0.5	0.24606	9.593E-2

Table A-2: BLOCKAGE UI Fields Summary - continued

Breaks/Targets (Volume Method= Break Database)	Type	Limits	Variable	Default
Array of Breaks				
Break ID	a8	Distinct	weldid[1..maxwls]	Example
System ID	enum of Sys IDs	enum	sysid[1..maxwls]	Recirc
Outside Diameter (in)	enum of pwds[]	enum	wdiam[1..maxwls]	22.0
Weld Type	enum of pwts[]	enum	wtype[1..maxwls]	S1
Location	enum of pwls[]	enum	wloc[1..maxwls]	H
Array of Targets (per Break)				
Number	AUTO		1..ntgts[i] where i=associated break	
Inside Diameter (in)	real	0<x<200	tgtdia[i,1..ntgts[i]]	22.0
System Code	a3		tgtsys[i,1..ntgts[i]]	RCA
Insulation	enum of dname[] where dloc[]=Drywell (target)	enum	tgttyp[i,1..ntgts[i]]	NK
Insulation Thickness (in)	real	0<x<100	tgtthk[i,1..ntgts[i]]	3.0
Array[1..maxld] of Insulation Length (ft)	real	Increasing,x >0	tgtlen[i,1..ntgts[i]]	11,18,25

Breaks/Targets (Volume Method= User Input)	Type	Limits	Variable	Default
Array of Breaks				
Break ID	a8	Distinct	weldid[1..maxwls]	Example
Outside Diameter (in)	enum of pwds[]	enum	wdiam[1..maxwls]	22.0
Location	enum of pwls[]	enum	wloc[1..maxwls]	H
Array of Targets (per Break)				
Number	AUTO		1..ntgts[i] where i=associated break	
Debris Type	enum of dname[] where dloc[]=Drywell (target)	enum	tgttyp[i,1..ntgts[i]]	NK
Volume (cu ft)	real	0<x<250	vuser[i,1..ntgts[i]]	1.0

Table A-2: BLOCKAGE UI Fields Summary - continued

Break Frequencies	Type	Limits	Variable	Default
Break Diameter Classes				
Array of Smallest Diameter	enum of pwds[]	enum excluding largest entry, strictly increasing	ndc,wdctbl[]	1,4,16,18
Array of Label	a5	distinct, not null	wdclbl[1..ndc]	1-2.4-10,16,18-22
Break Frequency				
Array of Frequency	real[][]	0<x<1	wwffwf[1..npwt, 1..ndc]	S1 1e-6,1e-6,2e-6,2e-7 S2 1e-6,1e-6,2e-6,2e-7 S3 1e-6,1e-6,2e-6,2e-7 C1 2e-7...2e-7 C3 2e-7...2e-7 C4 2e-7...2e-7

Table A-2: BLOCKAGE UI Fields Summary - continued

ECCS	Type	Limits	Variable	Default
Suppression Pool Volume (cu ft)	real	10<x<1e6	vpool	58900
Pool Settling Area (sq ft)	real	x>0; height=volume/area; .001<height<100	apool	5000
Pool Reference Temperature (deg F)	real	40<x<200	tref	125
Min Flow to Cool Core (GPM)	real	0<x<100,000	eccmin	25000
Pump Modules				
Name	a6	Distinct, not null	mname[1..nmods]	Mod1
Area (sq ft)	real	0<x<1000	area[1..nmods]	
Pumps				
Name	a6	Distinct within module, not null	pname[1..nmods,1..npumps[i]]	Pump1
Flow (GPM)	real	0<x<100,000	pflow[1..nmods, 1..npumps[i]]	
Reference NPSH (ft-water)	real	0<x<250	npsho[1..nmods, 1..npumps[i]]	

Table A-2: BLOCKAGE UI Fields Summary - continued

Correlation	Type	Limits	Variable	Default
Correlation	enum of NUREG/CR6224, BWROG Generic 1 Generic 2	enum	iblk	NUREG/CR6224
If Correlation=Generic1, Generic2				Generic1, Generic2
A1	real	0<=x<=1e11	cblk[1]	12,12
B1	real	0<=x<=1e11	cblk[2]	3,3
C1	real	0<=x<=1e11	cblk[3]	0,0
D1	real	0<=x<=1e11	cblk[4]	1,1
E1	real	0<=x<=1e11	cblk[5]	1.5,1.5
F1	real	0<=x<=1e11	cblk[6]	2,2
A2	real	0<=x<=1e11	cblk[7]	0,0
B2	real	0<=x<=1e11	cblk[8]	1,1
C2	real	0<=x<=1e11	cblk[9]	1,1
D2	real	0<=x<=1e11	cblk[10]	1,1
E2	real	0<=x<=1e11	cblk[11]	1,1
F2	real	0<=x<=1e11	cblk[12]	1,1
For all Correlations				NUREG/CR6224,BWROG,Generic1,Generic2
K	real	0<=x<=1e11	if iblk=SEA or BWROG, cblk[1] else cblk[12]	0,0,0.1,0.1

Table A-2: BLOCKAGE UI Fields Summary - continued

Event Scenario	Type	Limits	Variable	Default
End of Calculation Time (sec)	real	x>0	tend	21600
Large LOCA Break Size (in)	real	0<x<100	dlloca	6
Interpolation	enum of Step, Linear, Exponential	enum	interp[1..MAXTDV] 1=Time Step 2= Pump Flow 3= Pool Temperature 4= Drywell Transport 5= Resuspension 6= Turbulence	Step Linear Linear Step Step Exponential
Check Limit	real	0<=x<=1	chklim	1e-3
Medium LOCA				
End of Blowdown (sec)	real	x>0	tblowdn[2]	600
End of Washdown (sec)	real	x>0, washdown> blowdown	twashdn[2]	2400
Time Steps				
Array of Start Time (sec)	real	First entry =0, strictly increasing	->timtbl[2,1..]	Time Time Step 0, 1 320, .1 850, 1
Array of Time Step Size (sec)	real	x>0	->dttbl[2,1..]	2500, 10
Pump Flow Multiplier				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[2,1..], ->dttbl[2,1..], ->numdt[2]	Time, Multiplier 0, 0 49, 0 50, .1
Array of Multiplier	real	x>=0	->qflow[2][1..]	500, 1
Pool Temperature				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..],->dttbl[1,1..], ->numdt[1]	Time, Rate 0, 80 49, 95 50, 95 320, 150 500, 170 600, 170 720, 160 850, 150 2400, 100
Array of Temperature (deg F)	real	40<=x<=212	->temp[1,1..]	2500, 195

Table A-2: BLOCKAGE UI Fields Summary - continued

	Type	Limits	Variable	Default
Drywell Transport Rate (per second)				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[2,1..], ->dttbl[2,1..], ->numdt[2]	Time, Rate 0, .0016667 600, .00055556 240, 0
Array of Rate (per sec)	real	0<=x<=1	->g[2,1..]	
Resuspension Factor				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[2,1..], ->dttbl[2,1..], ->numdt[2]	Time, Factor 0, 1 600, 0
Array of Factor	real	0<=x<=1	->resk[2,1..]	
Turbulence Factor				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[2,1..], ->dttbl[2,1..], ->numdt[2]	Time, Factor 0, .0001 600, .0001
Array of Factor	real	0<=x<=1	->turbk[2,1..]	720, .5
Large LOCA				
End of Blowdown (sec)	real	x>0	tblowdn[1]	120
End of Washdown (sec)	real	x>0, washdown> blowdown	twashdn[1]	1920
Time Steps				
Array of Start Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..]	Time Timestep 0, .1 420, 1
Array of Time Step Size (sec)	real	x>0	->dttbl[1,1..]	2500, 10
Pump Flow Multiplier				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..], ->dttbl[1,1..], ->numdt[1]	Time, Multiplier 0, 0 4, 0 5, .1
Array of Multiplier	real	x>=0	Generates ->qflow[1,1..]	49, 1 50, 1
Pool Temperature				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..], ->dttbl[1,1..], ->numdt[1]	Time, Rate 0, 80 4, 81 5, 81 49, 100 50, 130 100, 180 120, 180 420, 170 1920, 150
Array of Temperature (deg F)	real	40<=x<=212	->temp[1,1..]	2500, 120

Table A-2: BLOCKAGE UI Fields Summary - continued

Drywell Transport Rate	Type	Limits	Variable	Default
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..], ->dttbl[1,1..], ->numdt[1]	Time, Rate 0, .00833333 120, .00055556 1920, 0
Array of Rate (per sec)	real	0<=x<=1	Generates ->g[1,1..]	
Resuspension Factor				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..], ->dttbl[1,1..], ->numdt[1]	Time, Factor 0, 1 120, 0
Array of Factor	real	0<=x<=1	Generates ->resk[1,1..]	
Turbulence Factor				
Array of Time (sec)	real	First entry =0, strictly increasing	->timtbl[1,1..], ->dttbl[1,1..], ->numdt[1]	Time, Factor 0, .0001 120, .0001 420, .5
Array of Factor	real	0<=x<=1	->turbk[1,1..]	

Table A-2: BLOCKAGE UI Fields Summary - continued

Output	Type	Limits	Variable	Default
Break/Target Report				
Include Velocity Group Distributions	enum of Include, Don't Include	enum	iprint	Don't Include
Output Frequency (sec)	real	1<=x<=1e6	pfreq	
Time-based Output				
Output Interval	int	x>=0	iplot	100
Selected Breaks				
Array of Break IDs	multiple select of weldid[]	at least one	iweld, idweld[]	Example
Early Termination Option	enum of None, First Pump Cavitation First Strainer Blocked, All Strainers Blockec, ECCS<Minimum	enum	iterm	None

Appendix B

BLOCKAGE Input File Format

Table of Contents

List of Tables

B.1 INPUT.DAT Format

Table B-1: General Parameters

Rec Type /Count	Description	Units	Variable	Type	Limits and Conditions
1/1	Case name	-	runid	a40	
2/1	Input file name	-	compid	a12	
3/1	Probabalistic option	-	iprob	int	1=Perform prob anal 0=Don't perform prob anal If ivolm=1, iprob is treated as 0 regardless of value
	Volume method	-	ivolm	int	0=Volume from targets 1=User input volumes
	Termination Option	-	iterm	int	1= at first pump cavitation 2= at first strainer blocked 3= at all strainers blocked 4= when ECCS flow < minimum 5= at user end time
4/1					
	Lines per page	-	ipage	int	>=10
	Plot frequency	-	iplot	int	>=0
	Sensitivity study flag	-	isens	int	0=No Study 1=Study
	Number of welds for detailed output	-	iweld	int	1<=x<=maxprt
	Velocity group option	-	iprint	int	0= do not output velocity group distribution 1= output velocity group distribution
	Detailed output frequency	sec	pfreq	real	1<=x<=1e6
IF isens=1					
5/1	Sensitivity study tag	-	psens	real	0<=x<=1e7
ENDIF isnes					
6/ i=1,iweld	Weld Ids for detailed output	-	idweld[i]	a8	Must be elements of weldid[] from WELD.DAT

Table B-2: Probabalistic Input

Rec Type/ Count	Description	Units	Variable	Type	Limits and Conditions
IF iprob=1					
7 // 1	Break frequency calculation method	-	break	a1	P=Plant W=Weld
8/1	Number of pipe diameter classes	-	ndc	int	1<=x<=maxdc
9/1	Smallest pipe diameters in class	in	wdctbl[i]	real	i=1,ndc; 0<=x<=100; strictly increasing
10/1	Pipe diameter class labels	-	wdclbl[i]	a5	i=1,ndc; distinct
IF break=P					
11/1	Break frequency by diameter class	1/Rx-yr	wdcffr[i]	real	i=1,ndc;0<=x<=1
12 / j=1,npwt	Weighing factor by diameter class byweld type	-	wwffwf[i,j]	real	i=1,ndc;x>=0
ELSEIF break=W					
12 / j=1,npwt	Break frequency by diameter class by weld type	1/Rx-yr	wwffwf[i,j]	real	i=1,ndc;x>=0
ENDIF break					
ENDIF iprob					

Table B-3: Debris Types

Rec Type/ Count	Description	Units	Variable	Type	Limits and Conditions
13/ 1	Number of debris types		ntypes	int	1<=x<=maxdeb
	Number of debris types in drywell from target destruction	-	ntg	int	0<=x<=maxdeb; ntg+ndw+nww=ntypes<= maxdeb
	Number of debris types in drywell not from target destruction	-	ndw	int	0<=x<=maxdeb; ntg+ndw+nww=ntypes
	Number of debris types in wetwell	-	nww	int	0<=x<=maxdeb; ntg+ndw+nww=ntypes
14/1	Debris type identifiers	-	debris[i]	A2	i=1,ntypes; distinct; Values for ntg must match pims values in weld.dat
15/1	Interpolation option for strainer filtration efficiency model	-	ifilt	int	1=step 2=linear 3=exponential
	Cake thickness corresponding to peak filtration efficiency	ft	tfilt	real	0<=x<=1

Table B-3: Debris Types - Continued

Rec Type / Count	Description	Units	Variable	Type	Limits and Conditions
FOR i=1,ntypes					
16/1	Debris type name	-	dname[i]	A20	Distinct
	Debris type identifier	-	--	A2	Must match debris(i)
	Debris type classification	-	dclass[i]	A1	F= Fibrous M= Metallic P= Particulate N= Ignore NOTE: ntg values must match those in weld.dat
	Debris type origin	-	dloc[i]	A2	TG= Drywell from target destruction or user input volume DW = Drywell not from target destruction WW= Wetwell
	Number of debris type settling velocity groups	int	ngroup[i]	int	1<=x<=maxvel
	Debris type volume	ft3	volume[i]	real	0<=x<=100 IGNORED IF ORIGIN=TG
17/1	Fabricated density	lbm/ft3	dfab[i]	real	0.5<=x<=1250
	Packing density	lbm/ft3	drub[i]	real	0.5<=x<=1250
	Material density	lbm/ft3	dmat[i]	real	0.5<=x<=1250
	Specific surface area	ft2/ft3	ssa[i]	real	100<=x<=2e6
IF dloc[i]=TG and ivolm=0					
18/1	Destruction fractions	-	dfract[i,j]	real	j=1,maxld; 0<=x<=1
19/1	Blowdown fractions by location	-	tfract[i,j]	real	j=1,npwl; 0<=x<=1
20/1	Washdown fractions by location	-	tlate[i,j]	real	j=1,npwl; 0<=x<=1
ELSEIF dloc[i]=DW					
19/1	Blowdown fraction	-	tfract[i,j]	real	j=1,1; 0<=x<=1
20/1	Washdown fraction	-	tlate[i,j]	real	j=1,1; 0<=x<=1
ENDIF					
21/j=1, ngroup[i]	Settling velocity group name	-	vname[i,j]	a6	Distinct within a debris type
	Initial strainer efficiency	-	efilt[i,j,1]	real	0<=x<=1
	Peak strainer efficiency	-	efilt[i,j,2]	real	0<=x<=1
	Retention factor	-	eheld[i,j]	real	0<=x<=1
	Settling velocity	ft/sec	vterm[i,j]	real	0<=x<=10
	Distribution fraction	-	fdist[i,j]	real	0<=x<=1; for all i (abs(sum(fdisk[i,all j]-1))<=.00001)
ENDFOR i					

Table B-4: Pump/Pool Parameters

Rec Type/ Count	Description	Units	Variable	Type	Limits and Conditions
22/1	Supression pool area	sq ft	apool	real	height=vpool/apool; .001<height<100
	Supression pool volume	cu ft	vpool	real	10<x<1e6
23/1	Number of pumping modules	-	nmods	int	1<=x<=maxecc
	Min flow to cool core	GPM	eccmin	real	0<=x<=1000000
	Reference NPSH temperature	deg F	tref	real	40<=x<=200
FOR i=1,nmods					
24	Pump module name	-	mname[i]	A6	Distinct
	Number of pumps	-	npumps[i]	int	1<=x<=maxpmp
	Area	sq ft	area[i]	real	0<x<1000
25/j=1,npumps[i]	Pump name	-	pname[i,j]	A6	Distinct within a pump module
	Flow capacity	GPM	pflow[i,j]	real	0<x<100000
	Reference NPSH	ft-water	npsho[i,j]	real	0<x<250
ENDFOR i					

Table B-5: Strainer Blockage Correlation

Rec Type/ Count	Description	Units	Variable	Type	Limits and Conditions
26/1	BLOCKAGE correlation	-	iblk	int	1= NUREG/CR6224 compressible fiber 2= BWROG 3= Generic 1 4= Generic 2
	Number of user defined coefficients	-	ncoef	int	If iblk=3 or 4 then x=13 else x=1
	BLOCKAGE correlation multiplier	-	hsens	real	1<=x<=10
27/1	BLOCKAGE correlation constants	-	cblk[i], i=1,ncoeff	real	0<=x<=1e8

Table B-6: Time Dependent Functions

Rec Type/ Count	Description	Units	Variable	Type	Limits and Conditions
28/1	Minimum LLOCA diameter	in	dlloca	real	0<=x<=100
	End time of calculation	sec	tend	real	tend>0
	Check limit for flagging incomplete fiber transfer to ww	-	cklim	real	0<=x<=1
29/1	Time dependent input interpolation method	-	interp[i]; i=1,MAXTDV	int	1=step 2=linear 3=exponential
30/1	LLOCA Number of input time table entries	-	numdt[1]	int	0<x<=maxdt
	LLOCA Blowdown end times	sec	tblowdn[1]	real	0<x<twashdn[1]
	LLOCA Washdown end times	sec	twashdn[1]	real	tblowdn[1]<x
31 / i=1,numdt[1]	LLOCA Time of time table input entry	sec	timtbl[1,i]	real	timtbl[1,1]=0, strictly increasing, must include tblowdn[1] & twashdn[1]
	LLOCA Time step sizes	sec	dttbl[1,i]	real	x>0
	LLOCA Pump flow multiplier	-	qflow[1,i]	real	x>=0
	LLOCA Pool temperature	deg F	temp[1,i]	real	40<=x<=212
	LLOCA Drywell transport rate	frac/sec	g[1,i]	real	x>=0
	LLOCA Resuspension factor	-	resk[1,i]	real	0<=x<=1
	LLOCA Turbulence factor	-	turbk[1,i]	real	0<=x<=1
32/1	MLOCA Number of input time table entries	-	numdt[2]	int	0<x<=maxdt
	MLOCA Blowdown end time	sec	tblowdn[2]	real	0<x<twashdn[2]
	MLOCA Washdown end times	sec	twashdn[2]	real	tblowdn[2]<x<tend
33/i=1,numdt[2]	MLOCA Time of time table input entry	sec	timtbl[2,i]	real	timtbl[2,1]=0, strictly increasing, must include tblowdn[1] & twashdn[1]
	MLOCA Time step sizes	sec	dttbl[2,i]	real	x>=0
	MLOCA Pump flow multiplier	-	qflow[2,i]	real	x>=0
	MLOCA Pool temperature	deg F	temp[2,i]	real	40<=x<=212
	MLOCA Drywell transport rate	/sec	g[2,i]	real	x>=0
	MLOCA Resuspension factor	-	resk[2,i]	real	0<=x<=1
	MLOCA Turbulence factor	-	turbk[2,i]	real	0<=x<=1

B.2 WELD.DAT Format for Volume from Targets (ivolm=0)

Table B-7: WELD.DAT Format for Volume

Rec Type / Count	Description	Units	Variable	Type	Limits and Conditions
1/1	Weld dataset name	-	probid	a40	
2/1	Number of weld diameters	-	npwd	int	1<=x<=maxpwd
3/12+i,i=npwd/5	Weld diameters	in	pwds[i]	real	i=1,npwd; 0<x<100; strictly increasing; 5 diams per line except last which has 1-5
4/1	Number of systems	-	nsys	int	1<=x<=maxsys
5 /, i=1,nsys	System descriptions	-	systbl[nsys]	a26	
6/1	Number of weld locations	-	npwl	int	1<=x<=maxpwl
7/1	Weld locations	-	pwls[1..npwl]	a2	Distinct
8 / i=1,npwl	Weld Location Description	-	lcdesc[1..npwl]	a40	
9/1	Number of weld types	-	npwt	int	1<=x<=maxpwt
10/1	Weld types	-	pwts[i]	a2	i=1,npwts
11/1	Number of insulation materials	-	npim	int	1<=x<=maxpim
12/1	Insulation material identifier	-	pims[i]	a2	i=1,npims Distinct
13/1	Insulation material type	-	fibflg[i]	a1	i=1,npims F= Fibrous M= Metallic P= Particulate N= Neglect in head loss calc
14/1	Destruction region L/Ds	-	ld[i]	real	i=1,maxld; 0<x<20
15/1+past welds and targets i=1,maxwls or EOF	Weld ID	-	weldid[i]	a8	Distinct
	System ID	-	sysid[i]	int	1<=x<=nsys
	Outside diameter of pipe	in	wdiam[i]	real	Element of pwds
	Weld type	-	wtype[i]	a2	Element of pwts
	Weld location	-	wloc[i]	a2	Elemens of pwls
	Number of targets (including self)	-	ntgts[i]	int	x<=maxtgts
	Target number	-	tgtno(1)	int	x=1
	Target outside diameter	in	tgtdia(i,1)	real	0<x<1000
	Target reference information	-	tgtsys(i,1)	a3	Ref only
	Target insulation type	-	tgttyp(i,1)	a2	Element of pims
	Target insulation thickness	in	tgtthk(i,1)	real	0<x<100
	Target lengths	ft	tgtlen(i,1,k)	real	k=1,maxld 0<x<=1000

Table B-7: WELD.DAT Format for Volume - Continued

Rec Type / Count	Description	Units	Variable	Type	Limits and Conditions
16 //14+past welds and targets, j=2,ntgts[i]	Target number	-	tgtno(1)	int	x<=maxtgts
	Target inner diameters	in	tgtdia(i,j)	real	0<x<1000
	Target reference information	-	tgtsys(i,j)	a3	x=j
	Target insulation material	-	tgttyp(i,j)	a2	Element of pims
	Target insulation thickness	in	tgtthk(i,j)	real	0<x<100
	Target lengths	ft	tgtlen(i,j,k)	real	k=1,maxld 0<=x<=1000; tgtlen(i,j,z)<=tgtlen(i,j,z+1)

B.3 WLED.DAT Format for User Volumes (ivolm=1)

Table B-8: WELD.DAT format for User Volumes

Rec type / Count	Description	Units	Variable	Type	Limits and Conditions
1/1	Weld dataset name	-	probid	a40	
2/1	Number of weld locations	-	npwl	int	1<=x<=maxpwl
7/1	Weld locations	-	pwls[1..npwl]	a2	Distinct
8/3+i,i=1,npwl	Weld Location Description	-	lcdesc[1..npwl]	a50	
11/1	Number of insulation materials	-	npim	int	1<=x<=maxpim
12/1	Insulation material identifier	-	pims[i]	a2	i=1,npims
13/1	Insulation material type	-	fibflg[i]	a1	i=1,npims F= Fibrous M= Metallic P= Particulate N= Neglect in head loss calc
14/1	Destruction region L/Ds	-	ld[i]	real	i=1,maxld; 0<x<20
FOR i=1,maxvol or EOF					
17/1	Volume Break ID		weldid[i]	a8	Distinct
	Location		wloc[i]	a2	Elements of pwls
	Outside Diameter	in	wdiam[i]	real	0<=x<=100
	Number of volumes for this break		ntgts[i[	int	1<=x<=maxtgt
	Break volume number		tgtno[i,1]	int	x=1
	Insulation Type		tgttype[i,1]	a2	Element of pims
	Volume	cu ft	vuser[i,1]	real	0<x<1000
18/ j=2,ntgts	Break volume number		tgtno[i,j]	int	x=j
	Insulation Type		tgttype[i,j]	a2	Element of pims
	Volume	cu ft	vuser[i,j]	real	0<x<1000

Table B-9: Maximum Values

Identifier	Description	Value
maxwls	number of welds	800
maxdt	time parameters per break size	100
maxtgt	targets per weld or user volume	40
maxpwd	weld diameters	30
maxdeb	fiber and particulate groups	20
maxpwt	weld types	20
maxvel	settling velocity groups	20
MAXCOE	correlation coefficients	13
maxpim	insulation materials	10
maxvol	user volumes	10
maxsys	system identifiers	10
maxpmp	pumps per module	8
MAXTDV	time dependent variables	6
maxprt	selected welds	5
maxpwl	weld locations	5
maxecc	pump modules	4
maxdc	diameter classes	4
maxld	destruction regions	3

Appendix C

Time-Based Results Variables

Line	Report Header	Plot Legend	Units
1	Time\Step	Time Step	sec
2	Flow Multiplier	Flow Mult	-
3	Pool Temp	Pool Temp	deg F
4	DW Trans\Rate	DW Rate	1/sec
5	Resuspend\Rate	Rsus Rate	1/sec
6	Turbulence\Factor	Turb Fact	-
7	Fiber Vol\From Drywell	F Drywell	ft^3
8	Fiber Vol\Suspended	F Suspend	ft^3
9	Fiber Vol\Settled	F Settled	ft^3
10	Fiber Vol\Strainers	F Strain	ft^3
11	Fiber Vol\Retained	F Retain	ft^3
12	Part Vol\From Drywell	P Drywell	ft^3
13	Part Vol\Suspended	P Suspend	ft^3
14	Part Vol\Settled	P Settled	ft^3
15	Part Vol\Strainers	P Strain	ft^3
16	Part Vol\Retained	P Retain	ft^3
17	Metal Vol\From Drywell	M Drywell	ft^3
18	Metal Vol\Suspended	M Suspend	ft^3
19	Metal Vol\Settled	M Settled	ft^3
20	Metal Vol\Strainers	M Strain	ft^3
21	Metal Vol\Retained	M Retain	ft^3
22	Total ECCS Flow	Tot Flow	GPM
23	NPSH\Change	del-NPSH	ft-water
24	Clean NPSH\Pump 1	cNPSH-1	ft-water
25	Clean NPSH\Pump 2	cNPSH-2	ft-water
26	Clean NPSH\Pump 3	cNPSH-3	ft-water
27	Clean NPSH\Pump 4	cNPSH-4	ft-water
28	Clean NPSH\Pump 5	cNPSH-5	ft-water
29	Clean NPSH\Pump 6	cNPSH-6	ft-water
30	Foul NPSH\Pump 1	fNPSH-1	ft-water
31	Foul NPSH\Pump 2	fNPSH-2	ft-water
32	Foul NPSH\Pump 3	fNPSH-3	ft-water
33	Foul NPSH\Pump 4	fNPSH-4	ft-water
34	Foul NPSH\Pump 5	fNPSH-5	ft-water
35	Foul NPSH\Pump 6	fNPSH-6	ft-water
36	Strainer 1\Velocity	Str1 Vel	ft/sec
37	Strainer 1\Theo Thick	Str1 Tthk	inch
38	Strainer 1\Act Thick	Str1 Athk	inch
39	Strainer 1\M/F Masses	Str1 M/F	-
40	Strainer 1\P/F Masses	Str1 P/F	-
41	Strainer 1\Head Loss	Str1 Hlos	ft-water
42	Strainer 2\Velocity	Str2 Vel	ft/sec
43	Strainer 2\Theo Thick	Str2 Tthk	inch
44	Strainer 2\Act Thick	Str2 Athk	inch
45	Strainer 2\M/F Masses	Str2 M/F	-
46	Strainer 2\P/F Masses	Str2 P/F	-
47	Strainer 2\Head Loss	Str2 Hlos	ft-water

Line	Report Header	Plot Legend	Units
48	Strainer 3\Velocity	Str3 Vel	ft/sec
49	Strainer 3\Theo Thick	Str3 Tthk	inch
50	Strainer 3\Act Thick	Str3 Athk	inch
51	Strainer 3\M/F Masses	Str3 M/F	-
52	Strainer 3\P/F Masses	Str3 P/F	-
53	Strainer 3\Head Loss	Str3 Hlos	ft-water
54	Strainer 4\Velocity	Str4 Vel	ft/sec
55	Strainer 4\Theo Thick	Str4 Tthk	inch
56	Strainer 4\Act Thick	Str4 Athk	inch
57	Strainer 4\M/F Masses	Str4 M/F	-
58	Strainer 4\P/F Masses	Str4 P/F	-
59	Strainer 4\Head Loss	Str4 Hlos	ft-water
60	Strainer 5\Velocity	Str5 Vel	ft/sec
61	Strainer 5\Theo Thick	Str5 Tthk	inch
62	Strainer 5\Act Thick	Str5 Athk	inch
63	Strainer 5\M/F Masses	Str5 M/F	-
64	Strainer 5\P/F Masses	Str5 P/F	-
65	Strainer 5\Head Loss	Str5 Hlos	ft-water
66	Strainer 6\Velocity	Str6 Vel	ft/sec
67	Strainer 6\Theo Thick	Str6 Tthk	inch
68	Strainer 6\Act Thick	Str6 Athk	inch
69	Strainer 6\M/F Masses	Str6 M/F	-
70	Strainer 6\P/F Masses	Str6 P/F	-
71	Strainer 6\Head Loss	Str6 Hlos	ft-water
72	Strainer 7\Velocity	Str7 Vel	ft/sec
73	Strainer 7\Theo Thick	Str7 Tthk	inch
74	Strainer 7\Act Thick	Str7 Athk	inch
75	Strainer 7\M/F Masses	Str7 M/F	-
76	Strainer 7\P/F Masses	Str7 P/F	-
77	Strainer 7\Head Loss	Str7 Hlos	ft-water
78	Strainer 8\Velocity	Str8 Vel	ft/sec
79	Strainer 8\Theo Thick	Str8 Tthk	inch
80	Strainer 8\Act Thick	Str8 Athk	inch
81	Strainer 8\M/F Masses	Str8 M/F	-
82	Strainer 8\P/F Masses	Str8 P/F	-
83	Strainer 8\Head Loss	Str8 Hlos	ft-water

Appendix D

Proposed Menus

Proposed Menus

File
- New
- Open...
- Close
- Save
- Save As...
- ---
- Print...
- Print Setup...
- ---
- <Recent Files>
- ---
- Exit

Edit
- Scale...

Input
- Definitions...
- Debris...
- Breaks/Targets...
- Break Frequencies...
- ECCS...
- Correlation...
- Event Scenarios...
- Output...

Run
- Analysis

Results
- Input Parameters
- ---
- Errors
- Break Detail
- Target Detail
- Frequency
- Break Summary
- Volume Sorted
- ---
- Time-Based...
- ---
- Save Results...

Window
- Cascade
- Tile
- Arrange Icons
- ---
- <Named Windows>

Help
- Contents
- Search...
- ---
- About BLOCKAGE ...
- Disclaimer...

Appendix E

Proposed Dialogs/Windows

Table of Contents

Figures

Tables

Proposed Dialogs/Windows

The following proposed screen shots depict the control panel window and the major dialog boxes. These images are intended to convey the general contents of the dialogs and imply their operation. Minor changes in field naming, arrangement, or space may occur in the implementation of the dialogs.

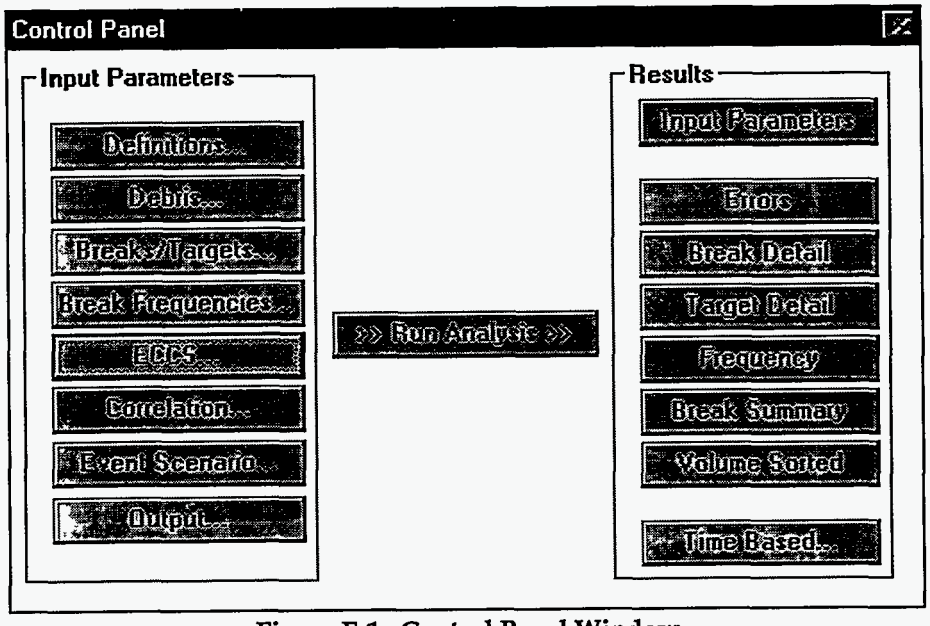


Figure E-1: Control Panel Window

Note: The Break Frequencies... button is disabled for User Volume models.

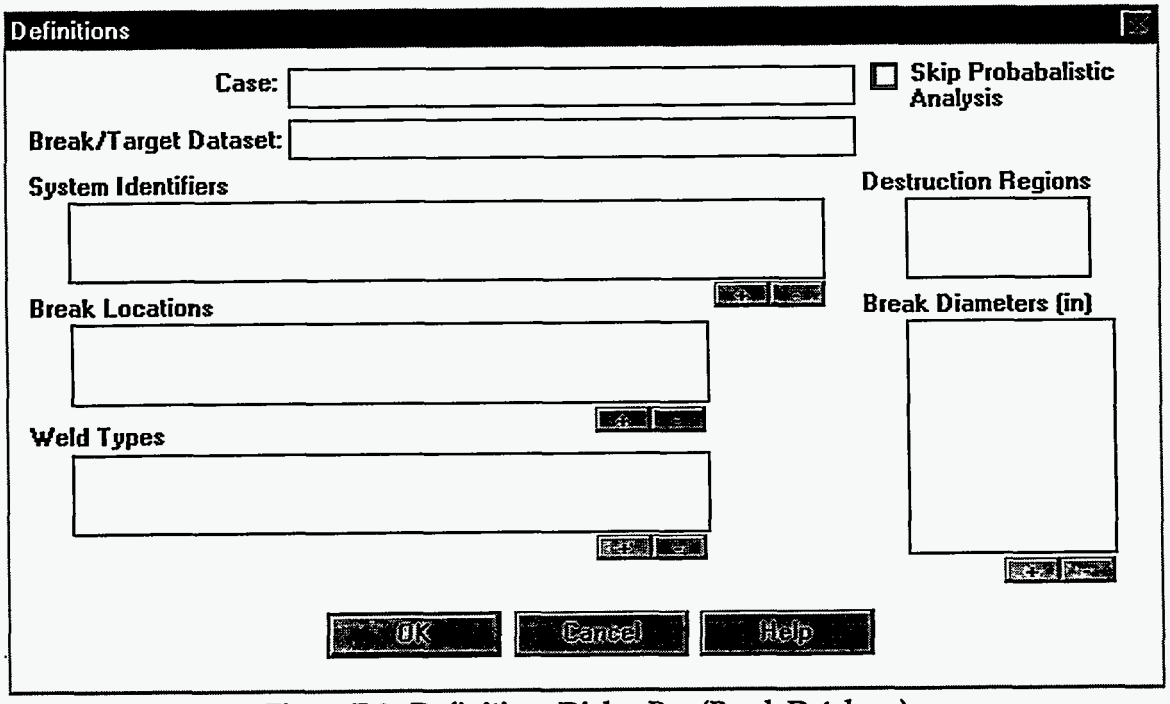


Figure E-2: Definitions Dialog Box (Break Database)

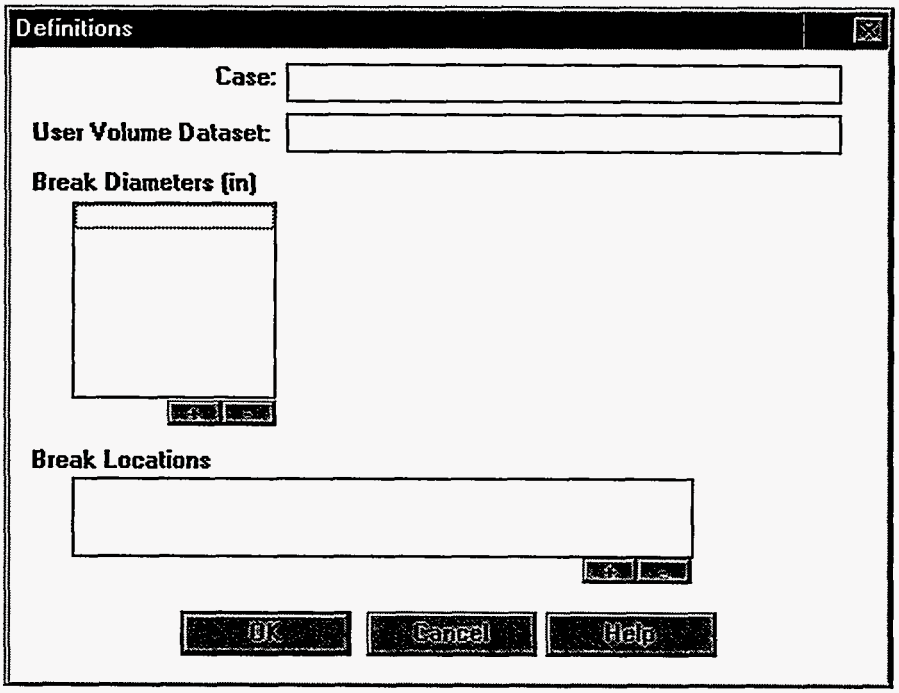


Figure E-3: Definitions Dialog (User Volumes)

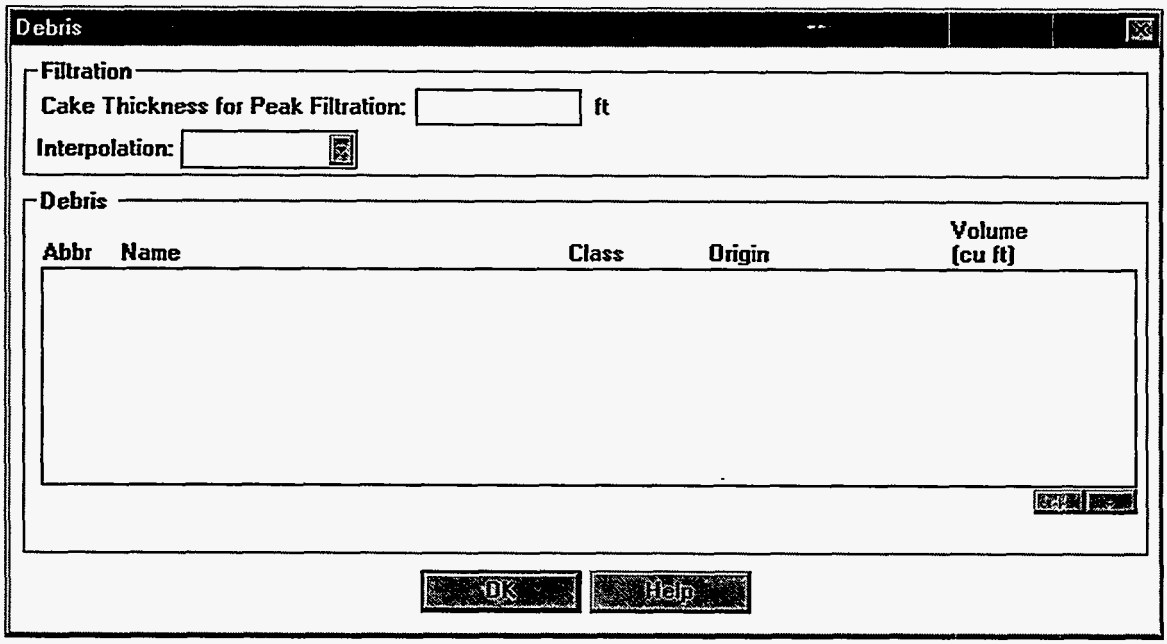


Figure E-4: Debris Dialog Box

Notes:

1) Interpolation combination box entries: Step, Linear, Exponential
2) Debris entries with Origin=Drywell (Target) have "-" displayed for their volumes.
3) Double clicking on an entry in the Debris list box or selecting an entry and clicking "Edit..." displays the dialog below.

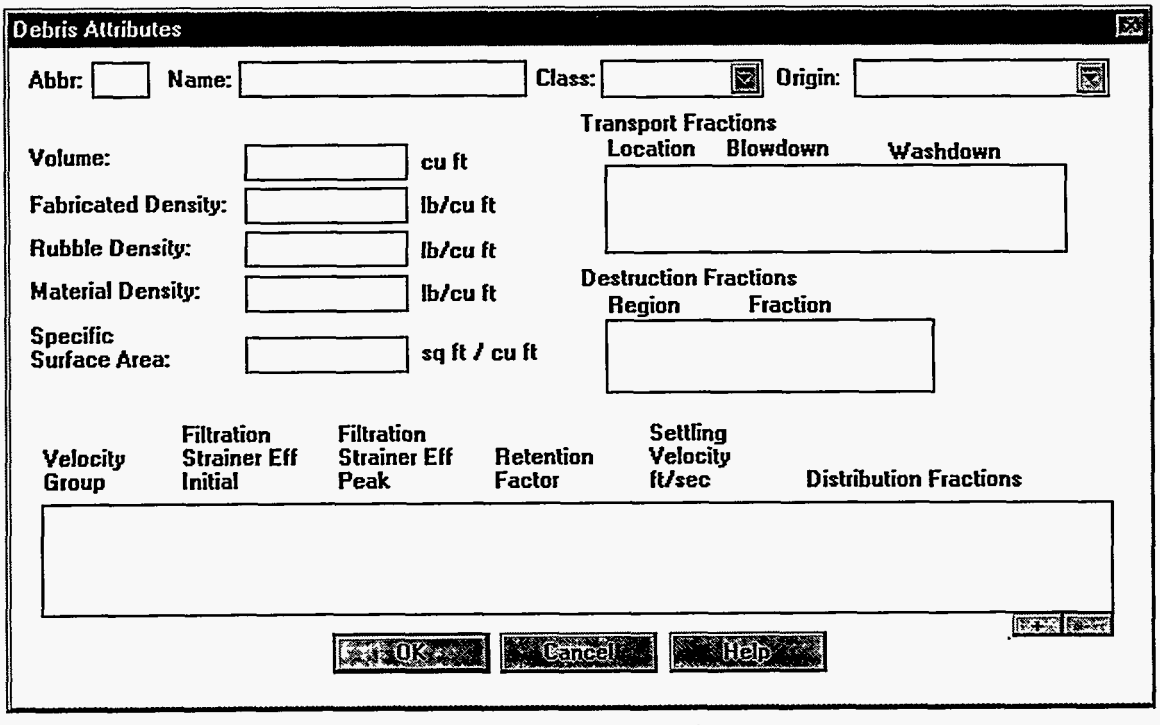


Figure E-5: Debris Attributes

Notes:

1) Class combination box entries:
   - Fibrous
   - Metallic
   - Particulate
   - Ignore

2) Origin combination box entries:
   - Drywell (Targets)
   - Drywell (Non-targets)
   - Wetwell

3) The Volume field is disabled in Origin = Drywell (Targets).
4) The Destruction Fraction section is not displayed for User Volume models.
5) The Transport Fractions list box is disabled if Origin=Wetwell, contains one line with location "ALL" if Origin="Drywell (Non-targets)", and contains one line per break location if Origin="Drywell (Targets)".

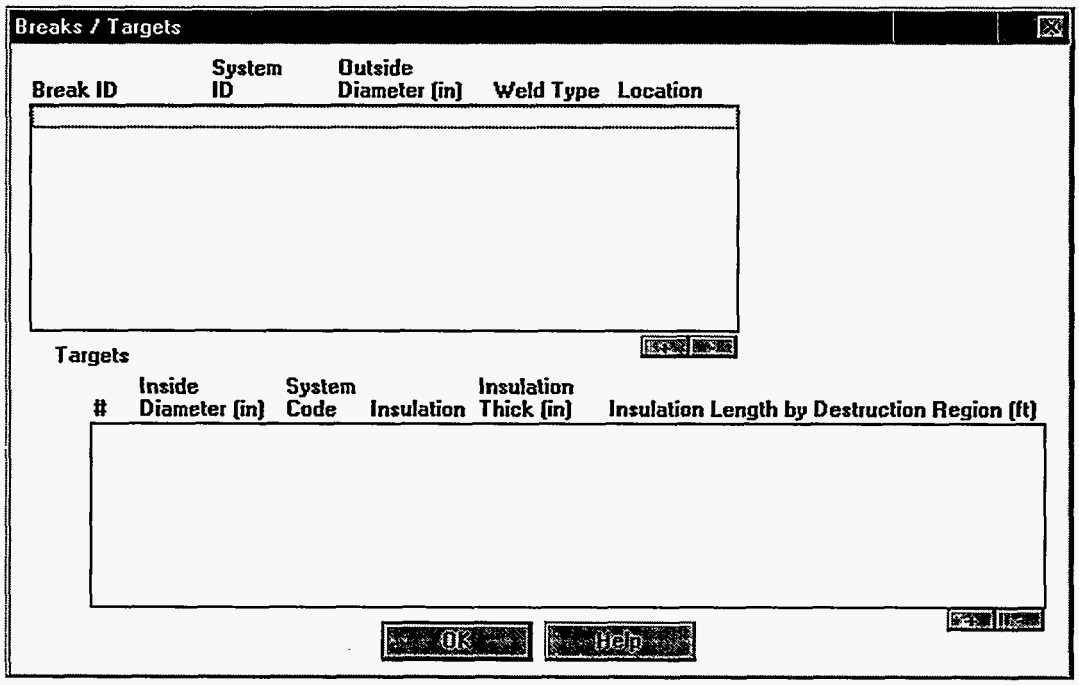


Figure E-6: Breaks/Targets (Break Database)

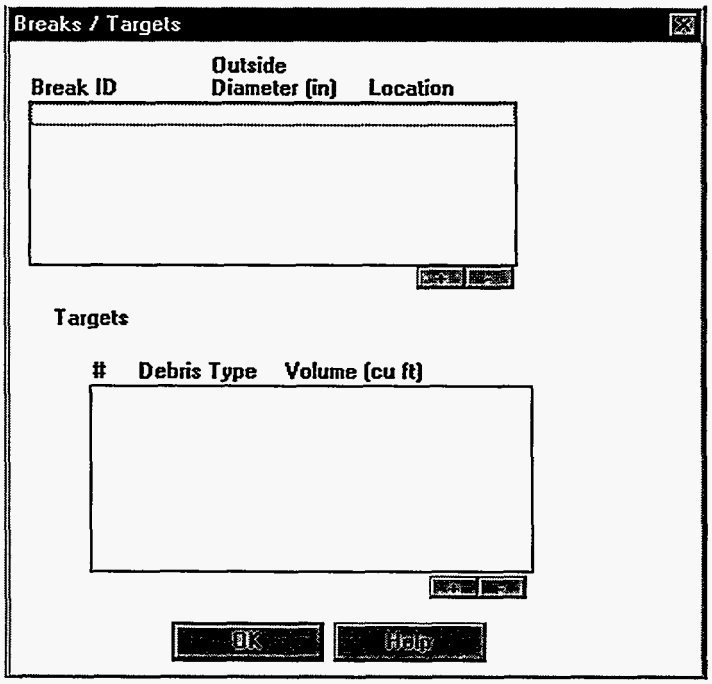


Figure E-7: Breaks/Targets (User Volume)

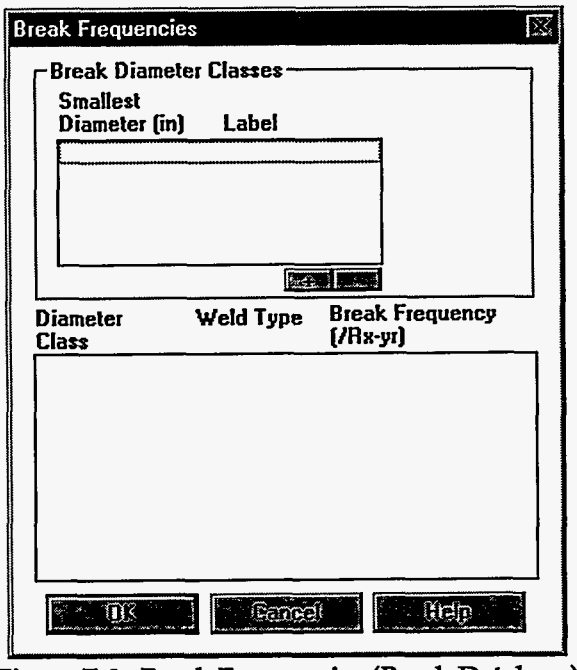
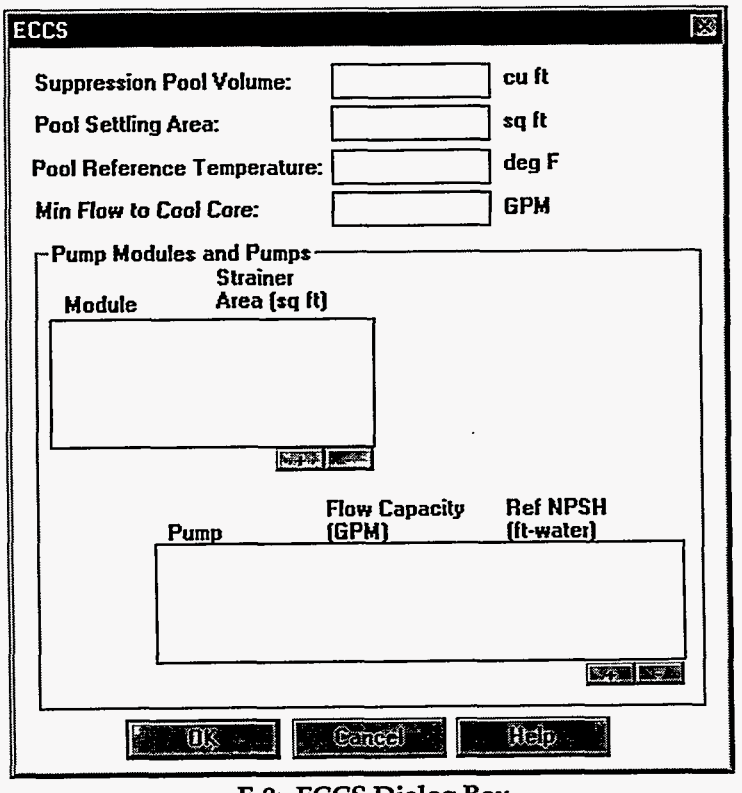


Figure E-8: Break Frequencies (Break Database)



E-9: ECCS Dialog Box

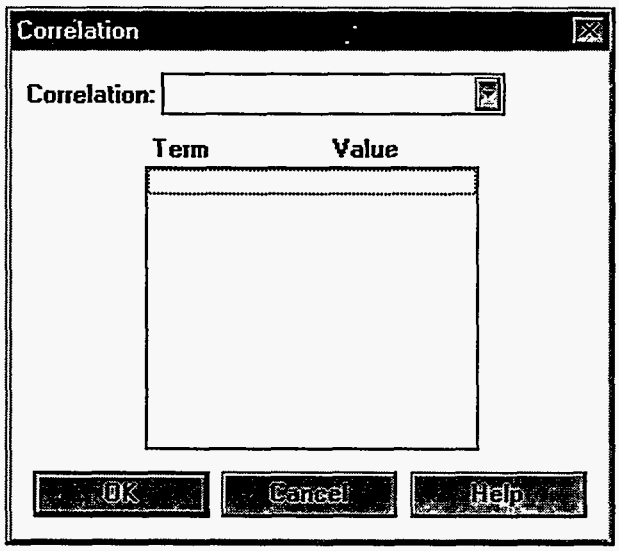


Figure E-10: Correlation Dialog Box

The following table describes the combination box entries and the associated terms in the term/value table.

Table E-1: Box Entries and Terms

Correlation	Terms
NUREG/CR6224	K
BWROG	K
Generic-1	A1, B1, C1, D1, E1, F1, A2, B2, C2, D2, E2, F2, K
Generic-1	A1, B1, C1, D1, E1, F1, A2, B2, C2, D2, E2, F2, K

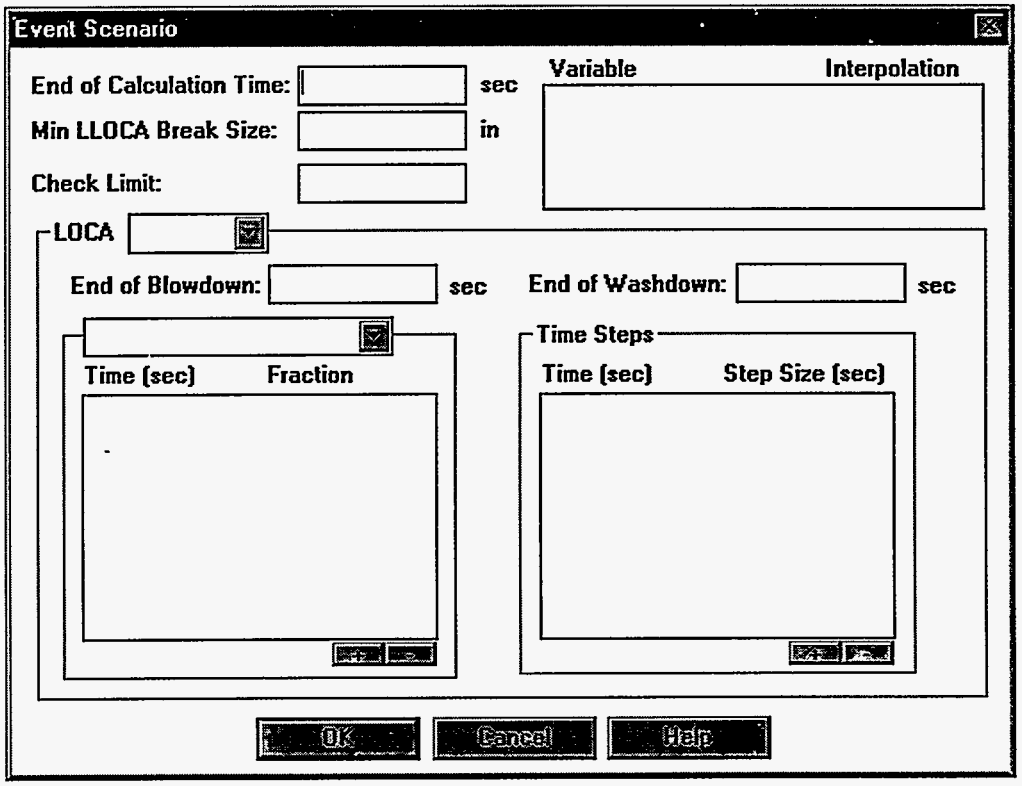


Figure E-11: Event Scenario Dialog Box

Notes:

1) For each variable, the following interpolation values are allowed:
   - Step
   - Linear
   - Exponential
2) Variable interpolation list box entries:
   - Time Step
   - Pump Flow Multiplier
   - Pool Temperature
   - Drywell Transport Rate
   - Resuspension Factor
   - Turbulence Factor

3) LOCA combination box entries:
   - Medium
   - Large
4) Variable time table combination box entries:
   - Pump Flow Multiplier
   - Pool Temperature
   - Drywell Transport Rate
   - Resuspension Factor
   - Turbulence Factor

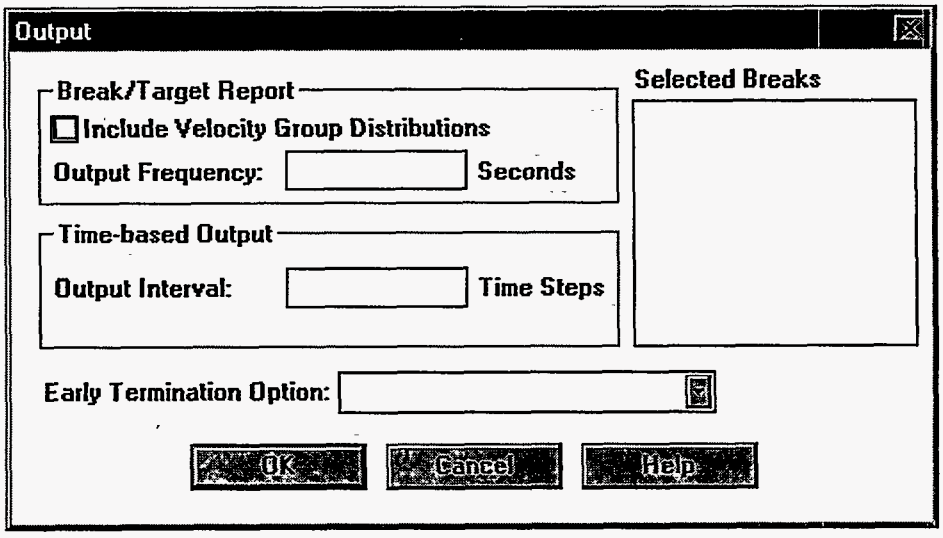


Figure E-12: Output Dialog Box

Notes:

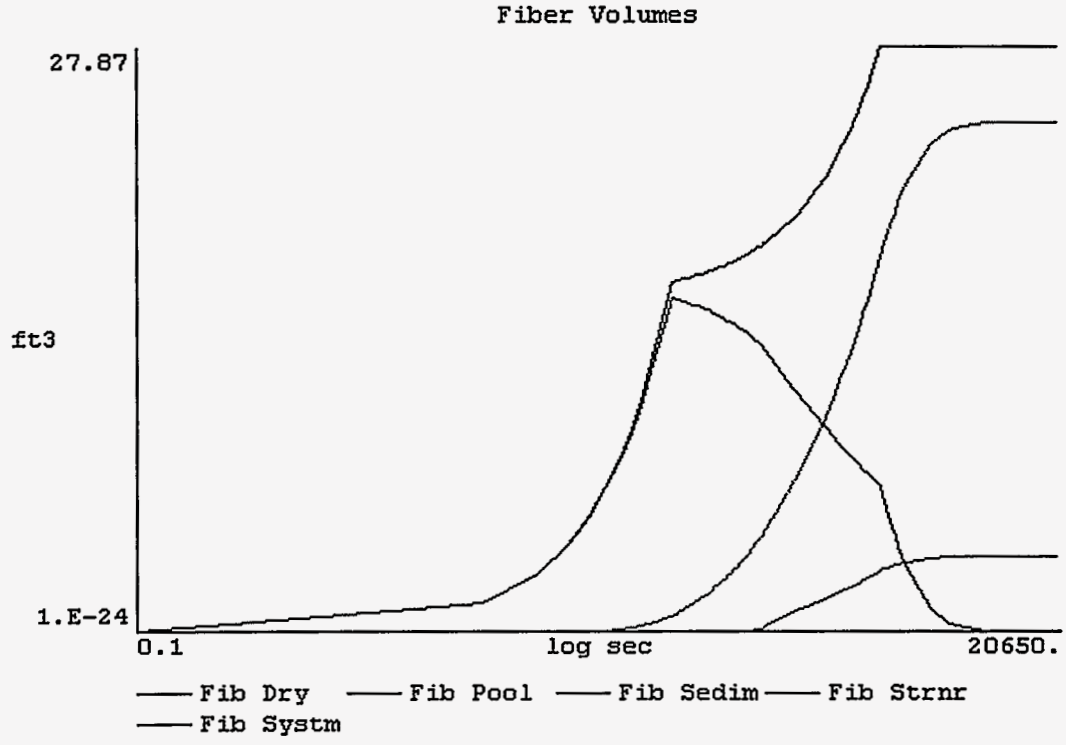
1) The Selected Breaks list box is a multiple select list box.

2) Early Termination Option combination box entries:
   - None
   - First Pump Loss of NPSH Margin
   - First Strainer Blocked
   - All Strainers Blocked
   - ECCS < Minimum

Appendix F

Plot Layout

Plot Layout


Fiber Volumes
27.87
ft3
1.E-24
0.1
log sec
20650.
Fib Dry
Fib Pool
Fib Sedim
Fib Strnr
Fib Systm

NRC FORM 335
(2-89)
NRCM 1102,
3201, 3202

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

(See instructions on the reverse)

1. REPORT NUMBER
(Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)

NUREG/CR-6371
SEA 96-3104-A:4

2. TITLE AND SUBTITLE

BLOCKAGE 2.5 Reference Manual

3. DATE REPORT PUBLISHED

MONTH	YEAR
December	1996

4. FIN OR GRANT NUMBER
W6325

5. AUTHOR(S)

C. J. Shaffer, W. Bernahl*, J. Brideau, D. V. Rao

6. TYPE OF REPORT

7. PERIOD COVERED (Inclusive Dates)

8. PERFORMING ORGANIZATION - NAME AND ADDRESS (If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)

Science and Engineering Associates, Inc.
6100 Uptown Blvd. NE
Albuquerque, NM 87110

*Software Edge, Inc.
1485 Chippewa Pathway
Riverwoods, IL 60015

9. SPONSORING ORGANIZATION - NAME AND ADDRESS (If NRC, type "Same as above"; if contractor, provide NRC Division, Office or Region, U.S. Nuclear Commission and mailing address.)

Division of Engineering Technology
Office of Nuclear Regulatory Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

10. SUPPLEMENTARY NOTES

M.L. Marshall,Jr.,NRC Project Manager

11. ABSTRACT (200 words or less)

The BLOCKAGE 2.5 code was developed by the United States Nuclear Regulatory Commission (NRC) as a tool to evaluate licensee compliance regarding the design of suction strainers for emergency core cooling system (ECCS) pumps in boiling water reactors (BWR) as required by NRC Bulletin 96-03, "Potential Plugging of Emergency Core Cooling Suction Strainers by Debris in Boiling Water Reactors." Science and Engineering Associates, Inc. (SEA) and Software Edge, Inc. (SE) developed this PC-based code. The instructions to effectively use this code to evaluate the potential of debris to sufficiently block a pump suction strainer such that a pump could lose NPSH margin was documented in a User's Manual [NRC, NUREG/CR-6370].

The Reference Manual contains additional information that supports the use of BLOCKAGE 2.5. It contains descriptions of the analytical models contained in the code, programmer guides illustrating the structure of the code, and summaries of coding verification and model validation exercises that were performed to ensure that the analytical models were correctly coded and applicable to the evaluation of BWR pump suction strainers.

The BLOCKAGE code was developed by SEA and programmed in FORTRAN as a code that can be executed from the DOS level on a PC. A graphical users interface (GUI) was then developed by SEA to make BLOCKAGE easier to use and to provide graphical output capability. The GUI was programmed in the C language. The user has the option of executing BLOCKAGE 2.5 with the GUI or from the DOS level and the Users Manual provides instruction for both methods of execution.

12. KEY WORDS/DESCRIPTORS (List word or phrases that will assist researchers in locating the report.)

BWR Suction Strainers
Loss of NPSH
Debris Generation
Debris Transport
Debris Interceptors
Head Loss Correlations

13. AVAILABILITY STATEMENT
Unlimited

14. SECURITY CLASSIFICATION
(This Page)
Unclassified
(This Report)
Unclassified

15. NUMBER OF PAGES

16. PRICE