

Introduction to FLUINT

SINDA/FLUINT is a comprehensive software package used by over 400 sites in the aerospace, energy, electronic, automotive, aircraft, HVAC, and petrochemical industries for design and simulation of heat transfer and fluid flow problems. It is the NASA-standard analyzer for thermal control systems.

This document introduces FLUINT, the fluid flow network capabilities. FLUINT represents only part of the complete SINDA/FLUINT package. The thermal (node/conductor) network capabilities (SINDA) and the graphical user interface (*SinapsPlus*[®]), and an additional plotting package (EZ-XY[®]) are introduced in separately available documents. Slight familiarity with SINDA is helpful.

Other supporting codes include Thermal Desktop[®], a CAD-based and FEM-compatible geometric pre- and post-processor to SINDA that includes radiation calculations (via the RadCAD[®] module). Thermal Desktop also features FloCAD[®], a geometric GUI for FLUINT.

What is FLUINT?

FLUINT is the world's most powerful and featured general-purpose thermal/hydraulic analyzer.

FLUINT is classified as a control volume or lumped parameter code. Although normally limited to internal one-dimensional (piping) networks, it can be used to solve certain classes of 2D and 3D problems. Given appropriate inputs, FLUINT can produce answers that are the same as those produced by codes that are based on finite volumes or finite differences. Most FLUINT models are much more free-form than such codes allow, enabling the creativity and experience of the analyst to be exploited without having to resort to writing specialized, single-use computer programs.

FLUINT is a network-style fluid flow simulator. Although it can be used by itself for purely hydrodynamic analyses such as pump matching, manifolding (flow distribution), or water-hammer, it can also be combined with SINDA thermal networks to simulate combined thermal/hydraulic systems. The user poses a problem by creating an arbitrary network* of thermodynamic points (*lumps*) connected by fluid flow passages (*paths*). The user may also define heat transfer routes (*ties*) between SINDA nodes and FLUINT lumps to simulate convection. In addition, the analyst defines an arbitrarily complicated solution sequence (perhaps providing auxiliary Fortran-style logic), and chooses the desired output frequencies and formats. Inputs may be defined indirectly as algebraic functions, making the code a cross between a spreadsheet and a thermal/fluid network analyzer for easy parametric analysis.

Detailed Geometry: At Your Discretion

Unlike CFD codes, *SINDA/FLUINT need not be geometry-based*. Without the use of tools such as FloCAD[®], this lack of geometry may make FLUINT more cumbersome to use than geometry-based codes for some problems with clearly defined, simple geometry. In most

* Although such networks are intentionally analogous to traditional thermal (SINDA) networks, fluid flow networks are specialized to handle the extra equations and phenomena associated with single- and two-phase flows.

cases, however, actual geometries are much more complex than need be represented for fluid flow and heat transfer solutions. Geometry-based meshes often produce unnecessarily large and inappropriately detailed models that are not only slow to solve, they can obscure results. Lacking geometry makes SINDA/FLUINT more flexible, and more appropriate for undefined or changing designs, high-level modeling (e.g. entire vehicle), what-if and sensitivity analyses, and for optimization and correlation/calibration tasks.

On the other hand, if the thermal/structural geometry is detailed and CAD drawings or structural finite element meshes are available, then FloCAD[®] enables 1D fluid networks to be generated and linked to such 3D thermal geometry without resorting to a full CFD solution. This makes FloCAD both easy to learn and rapid to use for applications such as air- or liquid-cooled electronics packaging. The resulting flow solutions are fast, which enables SINDA/FLUINT's Advanced Design modules to be used to treat analytic and systematic uncertainties, calibrate models to test data, etc.

Often, FLUINT is used in conjunction with CFD codes. For example, one might wish to calculate the frictional or heat transfer characteristics of a nonstandard component (e.g., a braided cable inside a pipe, forming an annular passage) a using 2D or 3D point analyses, and then use FLUINT to perform analyses of system-level interactions, dynamics, sizing and optimization, etc. Extrapolation and interpretation of test results (rather than CFD predictions) is also a common usage.

Working Fluids

Thermodynamic and transport properties (viscosity and conductivity) are included in the SINDA/FLUINT package for 20 common fluids, including water, ammonia, propane, and various refrigerants. In addition, the analyst may provide descriptions of alternate working fluids as one of the following categories:

1. Perfect gases, although not necessarily calorically perfect gases. Specific heat and transport properties can vary as functions of temperature. These gases can also be used as solutes within mixtures.
2. Simple liquids. Specific heat, density, compressibility, and transport properties can vary as functions of temperature.
3. Simple two-phase fluids. Enables analysts with either limited thermodynamics backgrounds or limited property data to quickly describe a two-phase fluid.
4. Arbitrarily complex single- or two-phase fluids. Enables complete descriptions of all fluid properties as Fortran instructions, including calls to third-party property databases. Many full-range descriptions of cryogenic fluids, fire retardants, the newer "green" refrigerants such as R134a, and even more accurate descriptions of water are available in this format.

Once defined, fluid descriptions can be stored separately for reuse in other models.

Working fluids can exist in one or two phases within any fluid system. Compressibility of the fluids is included.

Up to 26 real or perfect gases and simple liquids may be mixed within a single fluid system, along with one two-phase (condensible/volatile) substance. Gases can dissolve into and evolve out of any liquids. The mass of each such constituent is conserved in such cases. Otherwise, pure substances are assumed.

Lumps

Lumps represent a point at which energy and mass are conserved. Each lump has a single characteristic thermodynamic state (temperature “TL,” static* pressure “PL,” density “DL,” etc.). Lumps may represent the thermodynamic state of a finite volume within a fluid system. They may be used more abstractly to represent boundary conditions, volumeless interfaces or dead ends, etc.

There are three types of lumps, classified by their volume.

Tanks have a finite volume “VOL.” Tanks may represent a vessel or a finite cell within a subdivided volume. Their volume may grow or shrink with time according to a prescribed rate (“VDOT”), or they may stretch or contract with pressure changes according to a prescribed volumetric compliance (“COMP”). Tanks may also share *interfaces* with other tanks for modeling pistons, bellows, liquid/vapor interfaces (including those within capillary structures), and for subdividing control volumes into quasi-2D and -3D flow networks.

Plena have an infinite volume, and hence usually represent boundary conditions, sources or sinks, or huge vessels whose thermodynamic state is assumed constant.

Junctions have zero volume: both energy and mass flowing into an arithmetic node must balance the energy and mass flowing out at all times; junctions represent a local steady-state or time-independent solution. Junctions may be used to model interfaces, joining points (e.g., tees), dead ends, and negligibly small volumes.†

Analysts may apply a source or sink of power “QL” to tanks and junctions. This heat source may vary with time or temperature. As with any variation in SINDA/FLUINT, these dependencies may be defined by table look-ups or by arbitrarily complex calculations and logical manipulations. In addition to QL, users can define additional heat transfer to a lump by using a connection to a wall element represented by a SINDA thermal node. These connections between FLUINT lumps and SINDA nodes are called *ties*, as described below. Ties can be used to invoke single- and two-phase convection correlations, with the heat rate between the fluid and solid element calculated automatically.

Many other optional lump variables or *descriptors* exist.

Many lump variables are available to help describe the thermodynamic state, such as quality (“XL”), void fraction (“AL”), enthalpy (“HL”), density of the liquid phase (“DF”), partial pressure fraction of gaseous species X (“PPGX”), etc.

The user may optionally choose to define the 1, 2, or 3D coordinate location (“CX”, “CZ”, and “CZ”) of each lump, and the body force (gravity or other acceleration) vector component along those axes (“ACCELX”, “ACCELY”, and “ACCELZ”). Gravity effects, including thermosiphoning and buoyancy effects, stratification of two-phase flows, etc. can thereby be included as can time- and direction-dependent vehicle accelerations.

All such variables are available for inspection, output, and manipulation within concurrently executed user *logic blocks and spreadsheet expressions*, as will be described later. *FLUINT abounds with such variables, most of which the casual user need never know exist, since most are either not frequently used or are calculated automatically by the program.*

* Lumps may be alternatively designated as stagnant, meaning negligible velocities within the lump.

† SINDA/FLUINT divides the world into the finite, the infinite, and the negligible. Engineering judgement must be used to decide which volumes and time constants are important, and which can be neglected in order to answer the question at hand.

Nonetheless, they are available in order to make the program completely user-extensible since fluid system analyses are often full of specialized or custom equipment.

Paths

Paths describe the means by which fluid flows from one lump to another. Each path has a single characteristic *mass flowrate* “FR.” Paths may represent duct segments, valves, pumps, leaks, etc. Usually, the path flowrate is a function of the fluid state and pressure drop between the endpoint lumps: $FR_{1-2} \sim (PL_1 - PL_2)$, where FR_{1-2} is the flowrate of the fluid flowing from lump 1 to lump 2, and PL_1 is the current pressure of lump 1, and PL_2 is the current pressure of lump 2.

There are two types of paths, classified by whether or not they neglect fluid inertia.

Tubes are flow paths in which fluid inertia is taken into account. In other words, the flow-rate through a tube cannot change instantly during a transient event: the forces acting on the flow passage (friction, pressure, velocity gradients, etc.) must accelerate or decelerate the fluid over time. To model waterhammer, tubes must be used. Usually, tubes represent a duct or a segment of a duct, although more complicated components can also be simulated using tubes. Tubes are described by a hydraulic diameter “DH,” a flow area “AF,” and a length “TLEN.” As with lumps, many other optional descriptors exist from the simple (e.g., “FK” for added K-factor losses) to the sublime (e.g., “AM” for virtual or added mass coefficient that is only applicable to two-phase slip flow simulations), few of which will be needed in any single analytic case, but all of which are available nonetheless.

Connectors are flow paths in which fluid inertia can be neglected. In other words, the flow-rate through a connector is always at equilibrium with the forces on that path; connectors represents a local steady-state or time-independent solution. Connectors are further subdivided into *devices*. Devices are generic representations of common fluid system components such as pipe segments, pumps, filters, orifices, valves and control valves, etc.

The two most important and common types of connector devices are STUBEs and MFRSETs. MFRSETs represent passages with prescribed mass flowrates. They are often used as either boundary conditions or idealized pumps. STUBEs are “short* tubes,” or the time-independent (instantaneous) analog of tubes. In other words, STUBE connectors are in every way identical to tubes except that they have no inertia, just as junctions are in every way identical to tanks except they have no volume.

Ties

Ties describe the means by which heat flows between FLUINT lumps (representing the fluid) and SINDA nodes (representing the duct wall). Each tie has a single characteristic *conductance* “UA” (inverse of resistance) and heat flowrate “QTIE.”[†]

There are five types of ties, subdivided by whether the user provides the UA conductance (as with “HTU” and “HTUS” ties) or the program calculates it based on convection correlations and the current pressures, flowrates, etc. (as with “HTN”, “HTNC”, and “HTNS” ties).

Convection ties require one or more duct-like paths (i.e., tubes or STUBEs) to be named such that the program can use the shape and flowrate of the duct as part of the convection calcu-

* “Short” is actually defined as a small value of $\rho L/D$. In other words, long and/or thin and/or liquid-filled lines have more inertia than do short and/or wide and/or gas-filled lines.

† Heat transfer directly between lumps (axial conduction, back conduction) is modeled using *ties* or “fluid ties.”

lation. The thermodynamic state of the associated lump is used to determine whether standard single-phase, boiling, or condensation correlations are used.

Fties

Fties are similar to heat transfer ties (above), except that they interconnect lumps directly. In other words, they are used to simulate heat transfer *within* the fluid, which is often only important either for high-conductivity fluids such as liquid metals, and for slow moving or stagnant flows. For example, fties can be associated with ducts (tubes and STUBEs) to easily simulate axial “backconduction” in such lines.

Interfaces

Interfaces (“ifaces”) are specialized network elements enabling any two tanks (finite control volumes) to share a boundary. This boundary might represent a liquid/vapor interface or a piston, but it might also represent an imaginary subdivision of a volume.

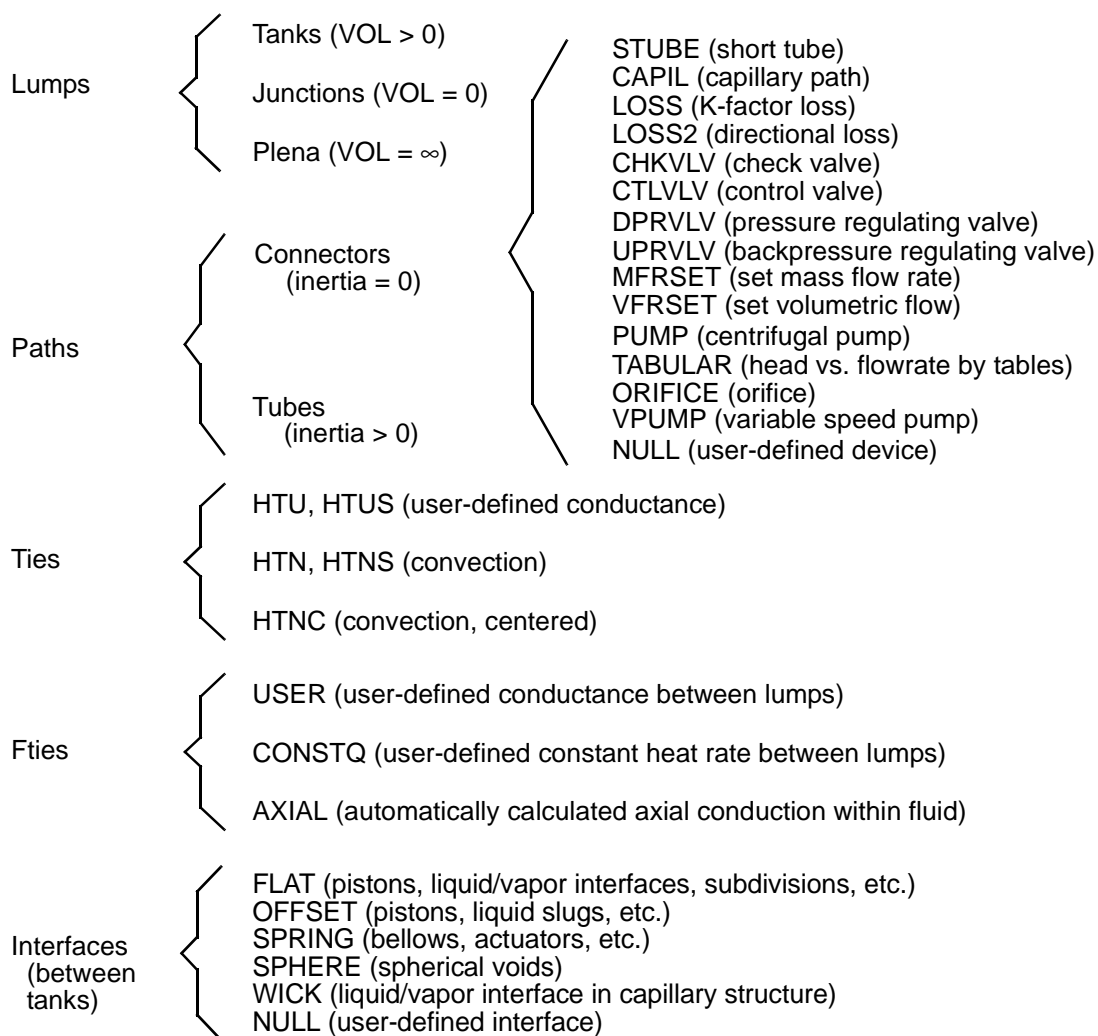


Figure 1: Hierarchy of FLUINT Network Elements

Usage Overview

SINDA/FLUINT is user-extensible, providing the analyst with complete control over inputs, outputs, and solution procedures. The program assumes very little about the problem at hand or which details are important to you as the analyst. To use SINDA/FLUINT correctly, you must have questions you want answered, and you must pose them in a way it can comprehend. There are no cook-book methods available: the experience and knowledge of the engineer is a vital ingredient in both arriving at a suitable model and an optimum solution approach. While this strategy may frustrate the casual user who is looking for an easy “joy-stick” approach, it delights the thermal/fluid engineering professional who understands that real thermal problems rarely lend themselves to such simplistic treatment or to hard-wired assumptions.

The inputs to the program may include:

1. network description: a set of lumps, paths, ties, interfaces, fties, and perhaps SINDA nodes and conductors describing the device or system
2. associated support data if needed: fluid properties, event profiles such as fluxes versus time, etc. SINDA/FLUINT offers a spreadsheet-like feature for defining key parameters (e.g., dimensions, loss coefficients) in a central “control panel” such that the remainder of the inputs can be defined indirectly on the basis of these parameters. This spreadsheet feature not only facilitates model building and upkeep, it also enables parametric analysis, optimization, data correlation, goal seeking, etc.
3. solution sequence: operations to be applied such as finding a steady-state solution for initial conditions, followed by a parametric series of transient integrations all starting with those initial conditions, etc.
4. output operations: including the amount, type, and frequency of outputs
5. control parameters to define or customize units, physical constants, solution methods or accuracy, etc.
6. supplementary logic, such as convection correlations, or simulation of electronic controllers, user-defined devices, etc.

Outputs may be user-defined, but generally include temperatures, pressures, and flowrates.

In its traditional form, SINDA/FLUINT is a batch-style code that accepts an ASCII input file and returns binary and ASCII results files. A Fortran compiler is required since each SINDA/FLUINT run builds and executes a custom program as shown in Figure 2.

While SINDA/FLUINT may still be employed in the traditional manner, two graphical user interfaces (GUIs) are also available. *SinapsPlus*[®] retains the geometry-free nature of SINDA/FLUINT: users sketch thermal and/or fluid networks on the screen, launch a run and can even monitor and control it interactively, and can postprocess (color, plot, etc.) results using such sketches. *Thermal Desktop*[®] is a geometric (CAD-based) tool that eliminates cumbersome generation of geometric factors by hand, and enables interfaces to CAD and FEM software. The *FloCAD*[®] module of *Thermal Desktop* allows 1D FLUINT circuits to be built in 3D and to interconnect with 2D or 3D thermal models.

Users new to SINDA/FLUINT are strongly encouraged to use *SinapsPlus* or *Thermal Desktop* to avoid having to learn the traditional ASCII input file formats. An introduction to *SinapsPlus* is available separately.

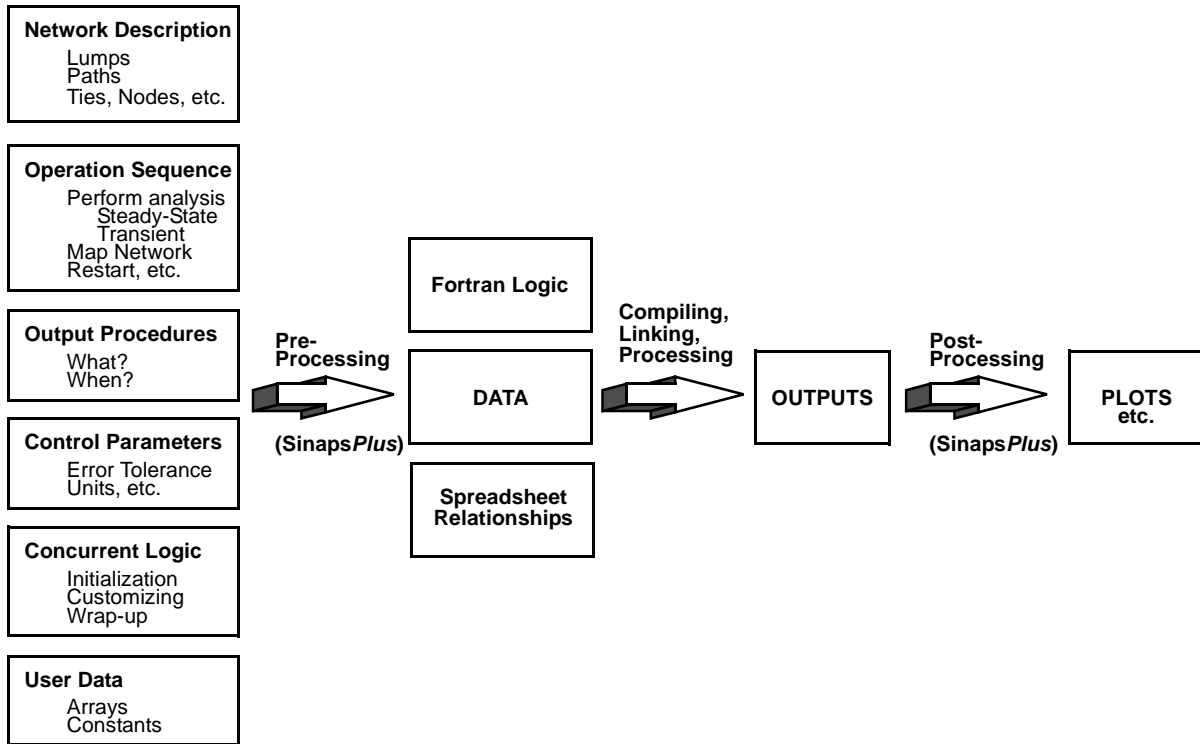


Figure 2: Basic Data Flow

User Logic

In addition to its geometry-independence, the feature that sets SINDA/FLUINT apart from other analyzers is its extensive use of user logic and spreadsheet-like interrelationships to both define and customize the solution approach. In essence, SINDA/FLUINT uses Fortran as its command language, although spreadsheet-like interrelationships can alternately be used.

Take the simple example of defining the end of a transient event. Assume that the duration of an event is unknown and may itself be the goal of the analysis. For example, the analyst might wish to know how long a hot water valve can be opened until a vessel somewhere downstream achieves a certain temperature. With SINDA/FLUINT, a simple line of Fortran-like logic might be used to detect such an event and terminate the solution:

```
IF (TL100 .GT. 212.0) TIMEND = TIMEN
```

where the problem end time (“TIMEND”) is set to the current time (“TIMEN”) if the temperature of lump #100 (“TL100”) ever exceeds 212 degrees. Alternatively, the user could simply supply the following definition of “TIMEND” using spreadsheet expressions:

```
TIMEND = (TL100 > 212)? TIMEN : 1.0E30
```

Logical instructions are also the method by which the user defines the solution sequence and the output operations. Additional logic may be inserted before, during, or after each network solution (i.e., steady state or transient analysis) as needed to tailor the execution. Entire libraries of reusable auxiliary routines are often generated by experienced users.

You do not need to know much Fortran in order to use SINDA/FLUINT. You can perform straight-forward analyses using a few simple commands such as:

```
CALL STEADY
```

to request that a new steady-state solution be performed.

However, if you already know some Fortran or are willing to learn a few simple manipulations, you will find few limits to your ability to pose new problems to SINDA/FLUINT.

Advanced Features

This section lists some of the more advanced features that are contained in SINDA/FLUINT, many of which will not be described in this document.

Macrocommands--Input commands exist that allow analysts to generate multiple network elements (i.e., lumps, paths, ties) as well as strings of such elements that represent discretized continuous flow passages ("duct macros"). Duct macros not only represent convenient means of quickly generating discretized models of lines for heat exchanger segments, manifolds, etc., they also automatically invoke more physics (e.g., spatial accelerations or momentum flux terms) than would individually input elements since the program knows they represent a continuous passage.

Exploitation of Symmetry--Parallel and identical flow passages and subcircuits frequently occur in fluid systems. Examples include ideally manifolded parallel lines. FLUINT features convenient means of modeling one such passage and then specifying the number of such passages in the system. This "duplication" feature allows symmetries in the system to be exploited in order to reduce model size.

Phase-specific Suction--Any FLUINT path may be directed to extract only liquid or only vapor from upstream lumps when both phases are present. This feature may be used to simulate the blockage of vapor in filters and capillary restrictions, liquid/vapor separators, or stratification of two-phase control volumes.

Capillary Devices--Options exist for modeling capillary passages (filters, wicks, and other bubble barriers) as well as capillary evaporator pumps, or devices that utilize capillary action to yield a pumping action when heat is applied.

Two-phase Flow Regimes and Slip Flow--If two-phases are present, flow regime mapping options may be invoked to improve the pressure drop and heat transfer calculations. These options automatically choose between one of four regimes (bubbly, slug, annular, and stratified) based on local flow conditions, orientation with respect to body forces or accelerations, etc. If desired, slip flow (differences in velocity between liquid and vapor) can be modeled. Otherwise, the default two-phase flow assumption is homogeneous.

Nonequilibrium Two-phase Control Volumes--By default, liquid and vapor phases are assumed to be in thermal equilibrium within control volumes. Modeling options exist that remove this assumption, enabling finite rate mass and heat transfer rates between phases.

Choking and Fast Hydrodynamic Transients--FLUINT can be directed to watch for choking (sonic limited or critical flows) locally or globally through out a fluid network, and to automatically simulate choking if it is detected. Also, options exist to monitor fast transient analyses (waterhammer and acoustic wave propagation events) if desired. Otherwise, FLUINT's implicit methods normally enable short time-scale events to be skipped when the time-con-

stants of interest are longer (e.g., the focus of the analysis is on thermal rather than hydrodynamic events).

Fluid Mixtures--Up to 26 real or perfect gases and/or simple liquids may be mixed within a single fluid network along with a condensible/volatile substance, with conservation of mass equations applied to each fluid species. Phenomena such as reduced heat transfer coefficients due to diffusion-limited condensation are modeled. Within mixtures, any number of gases may dissolve into any number of liquids, with phenomena such as homogeneous nucleation modeled. Liquids can be assumed to be miscible or immiscible, perhaps as a function of temperature. Solubilities can be defined in a wide variety of ways to exploit available data, or even to estimate solubilities in the absence of data.

Goal Seeking--The value of any input variable can be found given a desired response, reversing the traditional solution sequence.

Design Optimization--SINDA/FLUINT can perform multiple-variable design optimization (sizing, design synthesis) with arbitrarily complex constraints.

Automated Data Correlation--SINDA/FLUINT can be used to automatically adjust the uncertainties in a model as needed to correlate to test data (steady state, transient, or a complex mixture of the two).

Worst-case Design Scenarios--Given a list of uncertainties and variations, SINDA/FLUINT can be tasked with seeking the worst-case scenario to be used as a design case.

Reliability Engineering (Statistical Design)--Inputs may be specified not just as deterministic points but rather as ranges or probabilistic distributions. The code can then predict the chances that failure criteria will be exceeded. In fact, it can be combined with design optimization to synthesize a design based on reliability constraints, including perhaps defining what tolerancing is required.

FLUINT Sample Problem: Basic

This section develops a simple FLUINT model using the traditional ASCII input file approach. Modern graphical methods are available using *SinapsPlus®* or Thermal Desktop, which are described in separate documents. A template input file is available in the installation set, but this sample will start from a blank slate. Variations of this basic model will be made in later sections. (This model is developed in English units; SI units may be used alternatively in the code.)

Problem Description

Consider two adiabatic 1 ft³ vessels connected by a 10 ft long by 1 inch (inner) diameter adiabatic line. The vessels are filled with steam at 11 psia. The first vessel, initially at 200°F (slightly superheated), is fitted with a piston that begins to compress that vessel at time zero at a volumetric rate of 1.0 ft³/minute. The second (fixed volume) vessel is initially at 400°F.

A partially closed valve is located in the line near the second vessel. The valve is partially closed (throat area of 0.1 in²), at which position it exhibits a K-factor head loss of 20 (i.e., $\Delta P = K\rho V^2/2$, where $K=20$). See Figure 3.

How long (after the piston begins moving) does it take to either exceed 30 psia or 450°F anywhere within the system?

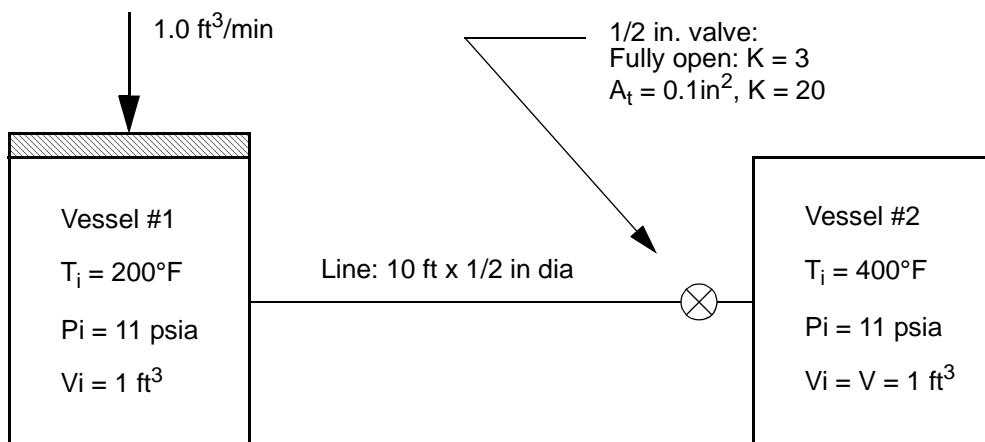


Figure 3: Sample Problem Schematic

This problem will be first worked at a very simple level, and will then be gradually expanded to include more details and design questions in later sections.

Model and Input Development

Registers--It is useful to start by defining a set of *registers* containing major problem variables (properties, dimensions, etc.). Changes to these registers in subsequent runs will propagate through the input file automatically, facilitating updates. These changes can also be propagated during a run, facilitating parametric investigations. Registers are the basis for many powerful options in SINDA/FLUINT, so their extensive use is strongly encouraged.

Using arbitrary user-defined names, the basic parameters of the model are defined as:

```
length = 10.0
diam = 0.5/12.0
area = 0.25*pi*diam**2
throat = 0.1/144.0
pinit = 11.0
tinit1 = 200.0
tinit2 = 400.0
```

where “area” is the flow area of the pipe, which might be useful for calculating other inputs. One register may be defined per line. Note that registers may either be set to constants, or to expressions which perhaps use built-in functions (such as $\sin(x)$, $\ln(x)$, $\max(x,y)$ etc.) or built-in constants (such as “pi” in the above expression for “area”), and which perhaps contain references to other registers (such as “diam” in the above expression for “area”).

SINDA/FLUINT input files are subdivided into blocks called *Header blocks*. The above declarations of registers are placed into a section titled **HEADER REGISTER DATA**. A complete subsection of the input file is therefore:

```
HEADER REGISTER DATA
length = 10.0
diam = 0.5/12.0
area = 0.25*pi*diam**2
```

```
throat = 0.1/144.0
pinit = 11.0
tinit1 = 200.0
tinit2 = 400.0
```

Notice that the “H” in HEADER must be placed in column 1. This is one of the few column restrictions in SINDA/FLUINT: column 1 is reserved for certain top-level commands.

Lumps and Paths--The volumes of the vessels are important to the system response. Therefore, each will be represented by “tanks” (i.e., control volumes) that are labeled 1 and 2, respectively. The volume within the line and valve, however, is assumed negligible (less than 0.013 ft³), and hence any lumps required there will be volumeless junctions.

Does the line need to be discretized (i.e., subdivided into strings of lumps and paths), or can it simply be represented by a single path? The answer to this question depends on whether or not any gradients in significant properties (temperature, density, etc.) are expected within that line. Since there is no heating of the line, the question becomes whether or not a significant pressure difference is expected to exist within the line. An assumption is made (and later verified) that the pressure difference between the tanks will be small at all times. Therefore, a single path is used to represent the line.

Although the line is long and thin and therefore might be represented by a tube (the inertial path in FLUINT), it contains low density gas and therefore little mass. Also, the time scale of inertial effects is on the order of hundredths of seconds, whereas the duration of the imposed event is on the order of tens of seconds or minutes. Therefore, the inertia of the line will be assumed to be negligible. An inertialess STUBE connector will therefore be used to represent the line rather than a tube.

The losses in the partially closed valve, together with entrance losses (say, $K=0.5$) can be added directly to this STUBE connector.* (Duct frictional losses are by default calculated automatically according to an internal curve fit to the Moody chart.) However, it is desired to distinguish the valve losses from the line losses, so a separate path is used to represent the valve. A simple LOSS connector is chosen to represent the path. A junction (number 10) must now be added in order to connect the paths in series. This junction represents the valve inlet. Since the pressures in the tanks has been assumed to roughly equal, choking detection and simulation can be disabled, otherwise it will be applied by default.

Units of ft, °F, hr, and psia will be used. Units of m, °C, s, and Pa could have been used alternatively, as could absolute temperature units (R or K). (FloCAD[®] permits many more unit choices and conversions.)

In a traditional ASCII input file, it is convenient to state the default conditions for lumps and paths. These defaults will be used by all subsequent lump and path definitions, unless specifically overridden by those definitions or replaced by new default values. The way in which these defaults are stated might appear as:

```
C SET LUMP DEFAULTS
LU DEF,PL = pinit          $ 11 PSIA INITIAL PRESSURE
      TL   = tinit1, XL = 1.0  $ 200 DEG F INIT TEMP., ALL GAS/VAPOR
      VOL  = 1.0              $ 1 CU FT VOLUME IN TANKS
```

* Since pressures are static by default in FLUINT, a K-factor of 1.0 need *not* be added to the exit. Instead, the acceleration loss from zero flow *at the inlet* is automatically included by designating the inlet tank as “stagnant.”

C PATH DEFAULTS

```
PA DEF, FR= 0.0          $ ZERO INITIAL FLOWRATE
    DH = diam            $ 1/2 INCH DIA. (CIRC AREA DEFAULTED)
```

Notice the use of “C” in column 1 for comments. The dollar sign (“\$”) may also be used for in-line comments. The “PA” and “LU” must occur in column 1, since they designate *subblocks*, or multi-line zones in which path or lump data may be defined. Within each subblock, inputs are column-independent and are usually keyword-driven (e.g., “DH” for hydraulic diameter, etc.). Notice that register names have been substituted for various inputs; expressions containing references to registers could have been used as well.

Notice that the default temperature defined above is incorrect for vessel #2. That overgeneralization can be corrected when Tank #2 is input. Alternatively, or a new “LU DEF” input can be issued prior to the input of that lump.

To define the lumps (in any order) using lump subblocks:

```
LU TANK,1, VDOT = -1.0*60.0 $ VESSEL #1, SHRINKING WITH TIME
    LSTAT = STAG             $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
LU JUNC,10                $ LINE OUTLET / VALVE INLET
LU TANK,2, TL = tinit2     $ VESSEL #2, INITIALLY AT 400 DEG F
    LSTAT = STAG             $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
```

Notice the use of a numerical expression for the definition of VDOT for Tank #2, converting ft³/min to ft³/hr. SINDA/FLUINT allows such expressions within blocks in order to make the models more self-documenting. These expressions can become quite complex, perhaps including spreadsheet-like interrelationships between inputs and outputs.

The designation of “LSTAT=STAG” means that the two end tanks terminate the line and have negligible velocities ... that their pressure will be interpreted as total or stagnation, and that an accelerational loss will be applied to any path that extracts fluid from such lumps. This designation, which is normally rarely needed except on plena (the infinite-volume lumps), overrides the default assumption that the pressures are to be treated as static.

The paths (connectors of varying device models) might be defined as follows:

```
PA CONN,10,1,10          $ LINE: PATH 10 FROM 1 TO 10
    DEV = STUBE           $ "SHORT TUBE" DEVICE
    TLEN = length         $ LENGTH
    FK = 0.5              $ INLET LOSS (K FACTOR)
PA CONN,20,10,2          $ VALVE MODEL: PATH 20
    DEV = LOSS            $ GENERIC LOSS DEVICE
    FK = 20.0             $ VALVE LOSS (K FACTOR)
    AFTH = throat         $ THROAT AREA
```

Actually, it is usually more convenient to arrange the inputs in “schematic order,” mixing lump and path definitions together:

```
LU TANK,1, VDOT = -1.0*60.0 $ VESSEL #1, SHRINKING WITH TIME
    LSTAT = STAG           $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
PA CONN,10,1,10          $ LINE: PATH 10 FROM 1 TO 10
    DEV = STUBE           $ "SHORT TUBE" DEVICE
    TLEN = length         $ LENGTH
    FK = 0.5              $ INLET LOSS (K FACTOR)
```

```

LU JUNC,10          $ LINE OUTLET / VALVE INLET
PA CONN,20,10,2    $ VALVE MODEL: PATH 20
    DEV = LOSS      $ GENERIC LOSS DEVICE
    FK = 20.0       $ VALVE LOSS (K FACTOR)
    APTH = throat   $ THROAT AREA
LU TANK,2, TL = tinit2 $ VESSEL #2, INITIALLY AT 400 DEG F
    LSTAT = STAG    $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL

```

All the data describing lumps, paths, and ties is placed in a header block called **HEADER FLOW DATA**. Actually, there may be more than one network or *submodel* in each model, for purposes that are described later. Therefore, even if there is only one submodel in this sample problem, it must be given a unique alphanumeric name (just as each node must be given a numeric identifier that is unique *within* its submodel). Using “CHAMBR5” as the submodel name yields the complete input block:

```

HEADER FLOW DATA, CHAMBR5
    FID = 718          $ WATER IS THE WORKING FLUID
C SET LUMP DEFAULTS
LU DEF,PL = pinit     $ 11 PSIA INITIAL PRESSURE
    TL = tinit1, XL = 1.0 $ 200 DEG F INIT TEMP., ALL GAS/VAPOR
    VOL = 1.0         $ 1 CU FT VOLUME IN TANKS
C PATH DEFAULTS
PA DEF, FR= 0.0       $ ZERO INITIAL FLOWRATE
    DH = diam         $ 1/2 INCH DIA. (CIRC AREA UNLESS SPECIFIED)
C DEFINE MODEL
LU TANK,1, VDOT = -1.0*60.0 $ VESSEL #1, SHRINKING WITH TIME
    LSTAT = STAG      $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
PA CONN,10,1,10      $ LINE: PATH 10 FROM 1 TO 10
    DEV = STUBE       $ "SHORT TUBE" DEVICE
    TLEN = length     $ LENGTH
    FK = 0.5          $ INLET LOSS (K FACTOR)
LU JUNC,10           $ LINE OUTLET / VALVE INLET
PA CONN,20,10,2     $ VALVE MODEL: PATH 20
    DEV = LOSS        $ GENERIC LOSS DEVICE
    FK = 20.0         $ VALVE LOSS (K FACTOR)
    APTH = throat     $ THROAT AREA
LU TANK,2, TL = tinit2 $ VESSEL #2, INITIALLY AT 400 DEG F
    LSTAT = STAG      $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL

```

Notice that the “FID = 718” that is found within the subblock formed by the **HEADER** command. This statement names water (by its ASHRAE number) as the working fluid for this submodel. User-defined fluids and mixtures of fluids may also be used. For example, a more complete and accurate description of water is available and is normally recommended as an alternative to the built-in, simplified properties.

Units and Top-Level Control--Normally, as long as the user provides inputs that obey a consistent unit system, SINDA does not enforce a unit convention. When fluid submodels are used, however, the analyst must choose between US Customary (old English) or SI (metric) units. Such model-level decisions are placed in a data block called **HEADER CONTROL**

DATA, which applies to all submodels if the special name “GLOBAL” is used. For the current problem:

```
HEADER CONTROL DATA, GLOBAL
C ENG UNITS (ft-F-lb/hr-psia) will be used:
  UID = ENG
```

Within this unit system, the user may optionally choose degrees Rankine and psig. (SI users may choose degrees C or K, etc.)

Other top-level data that is appropriate to include in this model-level block is the problem end time (“TIMEND”) for transients. This is chosen as 59 seconds (since Vessel #1 would shrink to an illegal zero volume in 60 seconds). In the default units of hours:

```
TIMEND = 59.0/3600.0 $ RUN FOR 59 SECONDS
```

Output Specifications--Despite their occasional Fortran-like appearance, all of the previously described inputs are *data blocks* since they specify network or control data rather than execution instructions. Unlike previously described inputs, the desired outputs are specified by a *logic block*. Logic blocks are pseudo-Fortran listings that are converted into real Fortran by SINDA/FLUINT, and then compiled and executed.

To specify the desired outputs, the user supplies the output operations to be performed by the code at predefined intervals (1% of the problem end time, by default), using canned routines and/or user-supplied output instructions:

```
HEADER OUTPUT CALLS, CHAMBRS      $ DESIGNATE OUTPUT OPERATIONS
  CALL LMPTAB('ALL')              $ LUMP TABULATION
  CALL PTHTAB('ALL')              $ PATH TABULATION
  IF(PL1 .GE. 30.0D0 .OR.         $ LOGIC TO STOP TRANSIENT INTEGRATION
    .   TL1 .GE. 450.0 .OR.
    .   TL2 .GE. 450.0) TIMEND = TIMEN
```

The routine “LMPTAB” tabulates lump properties such as temperature, density, and pressures. The use of ALL in single quotes means all fluid submodels (which in this case is one) will be listed. Otherwise, individual submodels may be specified by their name. “PTHTAB” functions analogously for paths. These operations will be performed every OUTPTF hours, as designated in HEADER CONTROL DATA, GLOBAL.

In the above block, “PL1” means the pressure in lump 1, “TL2” the temperature in lump #2, etc. Such variables are *translated* by SINDA/FLUINT into a reference to a cell in a Fortran array. While the details of the translation are usually not important, it *is* necessary to know that such translations occur, and that they can be customized and controlled.

The last line of Fortran logic checks to see if the temperature or pressure limits are ever exceeded, and halts the transient if so by setting the end time (TIMEND) equal to the current time (TIMEN). Notice that “PL” or lump pressures are double precision values, but that almost all other SINDA/FLUINT names represent single precision variables.

By default, OUTPUT CALLS is executed before and after each steady-state solution, and as requested in transients. The user can customize the calling frequency if desired. In fact, the value of the output frequency and other control constants may be changed within logic blocks, and therefore during a solution procedure. In the above logic, the answer will as accurate as the output interval (about 1/2 second in this case).

More accuracy could have been achieved by defining TIMEND directly in CONTROL DATA, using a spreadsheet relationship and avoiding Fortran logic:

```
TIMEND = (max(chambrs.TL1, chambrs.TL2) >= 450)? timen : 59/3600
```

(The “chambrs.PL1 >= 30” clause has been omitted in the above relationship for clarity.) The above logic would be automatically invoked at least every time step as part of the self-resolving spreadsheet that operates concurrently with SINDA/FLUINT execution.

Solution Sequence--Like OUTPUT CALLS, the entire solution sequence is specified as a logic block, meaning that the user has complete control over program execution from start to finish.

The solution sequence is specified in a header block called HEADER OPERATIONS. Instructions placed in this block will become the main driver for the run to be made. It will be turned into a once-through subroutine, meaning that once the operations contained in that block have been executed, the program will stop.

In this particular sample problem, a single transient run is needed.* Such solutions are requested by calling single routines such as:

```
CALL TRANSIENT
```

Before starting a transient, the user must select the problem end time TIMEND (input as a large value if unknown):

```
TIMEND = 59.0/3600.0 $ RUN FOR 59 SECONDS
```

This statement may be placed either in the *logic block* HEADER OPERATIONS *prior* to the call to TRANSIENT, or it may be placed in the *data block* HEADER CONTROL DATA, GLOBAL, where initializations may be made. This data block just *happens* to have a format that looks like a logic block.

If instead a steady-state solution had been requested, the user must first provide the maximum number of iterations that each steady-state call may attempt before either convergence is achieved or the program gives up. Each iteration represents a single pass through the solution equations, with each nodal temperature being updated once. The maximum number of iterations is a global control constant named NLOOPS. Generally, NLOOPS should be set to the maximum size the user can afford: a number that is normally estimated based on prior experience and knowledge of each model. Generally, the larger the model, the more iterations will be required to solve it.

Since SINDA/FLUINT can apply multiple submodels to any problem, the user must define the list of submodels that will participate in the next solution, even if only one such submodel exists. This list of fluid submodels is declared by a BUILDF statement of the format:

```
BUILDF config, sm1, sm2, ...smN
```

where “config” is the arbitrary user name for the current *configuration* or active subset of the master model, and “sm1” through “smN” are the names of fluid submodels comprising this list. Note that the “B” in “BUILDF” must be placed in column 1, whereas other instructions in this and other logic blocks follow the Fortran column conventions.[†]

* This is atypical. Most fluid transients must start from a valid initial condition, which is usually provided by a prior steady-state analysis.

† Namely, columns 1 through 5 are reserved for numeric labels, column 6 for continuation characters, and columns 7 through 72 for the statement itself.

Using a short-cut to build all fluid submodels, the complete input block becomes:

```
HEADER OPERATIONS
BUILDF ALL
      CALL TRANSIENT          $ START A TRANSIENT ANALYSIS
```

The control constants* OUTPTF and TIMEND were already initialized in CONTROL DATA, GLOBAL.

Other Input Sections -- Other data blocks are used to control and customize program execution, such as naming the files to be used by SINDA for outputs and other purposes. One such block, which must always occur first within the input file, is the OPTIONS DATA block:

```
HEADER OPTIONS DATA
TITLE TWO CHAMBERS WITH VALVED CONNECTING LINE
      OUTPUT = chambers.out
```

In the above block, the file to use for program output is specified along with a title to appear at the top of each output page.

Complete Input File--The complete input file defining the above problem is as follows:

```
HEADER OPTIONS DATA
TITLE TWO CHAMBERS WITH VALVED CONNECTING LINE
      OUTPUT = chambers.out
C
HEADER REGISTER DATA
      length = 10.0
      diam = 0.5/12.0
      area = 0.25*pi*diam**2
      throat = 0.1/144.0
      pinit = 11.0
      tinit1 = 200.0
      tinit2 = 400.0
C
HEADER FLOW DATA, CHAMBERS
      FID = 718          $ WATER IS THE WORKING FLUID
C SET LUMP DEFAULTS
LU DEF,PL = pinit      $ 11 PSIA INITIAL PRESSURE
      TL = tinit1, XL = 1.0 $ 200 DEG F INIT TEMP., ALL GAS/VAPOR
      VOL = 1.0          $ 1 CU FT VOLUME IN TANKS
C PATH DEFAULTS
PA DEF, FR= 0.0        $ ZERO INITIAL FLOWRATE
      DH = diam          $ 1/2 INCH DIA. (CIRC AREA UNLESS SPECIFIED)
C DEFINE MODEL
LU TANK,1, VDOT = -1.0*60.0 $ VESSEL #1, SHRINKING WITH TIME
      LSTAT = STAG      $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
PA CONN,10,1,10       $ LINE: PATH 10 FROM 1 TO 10
      DEV = STUBE      $ "SHORT TUBE" DEVICE
      TLEN = length    $ LENGTH
```

* Again, an historical misnomer. "Constants" are actually Fortran variables that whose value may be changed during the course of processor execution.

```

        FK = 0.5                $ INLET LOSS (K-FACTOR)
LU JUNC,10                    $ LINE OUTLET / VALVE INLET
PA CONN,20,10,2              $ VALVE MODEL: PATH 20
        DEV = LOSS             $ GENERIC LOSS DEVICE
        FK = 20.0              $ VALVE LOSS (K-FACTOR)
        AFTH = throat          $ THROAT AREA
LU TANK,2, TL = tinit2       $ VESSEL #2, INITIALLY AT 400 DEG F
        LSTAT = STAG          $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
C
HEADER OPERATIONS
BUILDF ALL
        CALL TRANSIENT        $ START A TRANSIENT ANALYSIS
C
HEADER CONTROL DATA, GLOBAL
C ENG UNITS (ft-F-lb/hr-psia) will be used:
        UID = ENG
        TIMEND = 59.0/3600.0  $ RUN FOR 59 SECONDS
C
HEADER OUTPUT CALLS, CHAMBR5 $ DESIGNATE OUTPUT OPERATIONS
        CALL LMPTAB('ALL')    $ LUMP TABULATION
        CALL PTHTAB('ALL')    $ PATH TABULATION
        IF(PL1 .GE. 30.0D0 .OR. $ STOP PROBLEM LOGIC
.          TL1 .GE. 450.0 .OR.
.          TL2 .GE. 450.0) TIMEND = TIMEN
END OF DATA

```

The final optional line enables even more comments to be appended to the file, since SINDA/FLUINT will not read past this command.

Other than the first mandatory OPTIONS DATA block, with few exceptions the remaining HEADER blocks may be input in any order.

Execution

Internally, SINDA/FLUINT follows a two-step process, as was shown in Figure 2. In the first step, the *preprocessor*, the data file is scanned and analyzed for consistency. Any format errors or missing data will be flagged and will cause the run to terminate. If no such errors are found, the preprocessor will write out a Fortran file created from the user's inputs. The Fortran compiler will then be invoked, and the SINDA/FLUINT library will be linked with the resulting object code to create the *processor*, which will be unique for each problem run. The processor is then executed, with the instructions defined in OPERATIONS completely defining its scope.

On Unix machines (as an example), the above sequence may be invoked as:

```
sinda cham.inp >pp.out
```

where cham.inp is the name of the file containing the model, and "pp.out" is the name of the file to contain preprocessor messages (which are normally discarded for successful runs). Fortran compiler errors, if any, will be either displayed on the screen or written to a file (depending on the operating system and the compiler). Processor output will be directed to the file named within cham.inp, which is named "chambers.out" in this case.

On PCs, a Windows-based utility called SINDAWIN is used to launch a run if neither SinapsPlus[®] nor Thermal Desktop[®] is used.

Results

Without SinapsPlus^{*} or Thermal Desktop, the presentation of the output must be made in tabular form, as shown in the rotated sample output page. This output contains the final page of LMPTAB and PHTTAB information. As can be seen, the limiting factor is the temperature of the second vessel, which exceeds 450°F in 53 seconds.[†] The competing effects of compressed gas and injected cold gas result in only a modest temperature rise in that volume.

As assumed, the pressure differences between the two tanks are small. Losses in the valve exceed line losses, keeping the Reynold's number in the line mostly in the laminar regime throughout the event.

This model requires much less than a minute of CPU time on most machines.

FLUINT Sample Problem: More Details

In this section, the previously defined sample problem will be reworked in various details, illustrating key FLUINT features.

Variation 1: User-defined Working Fluids and INSERT Directives

In addition to water, 19 other common fluids can be referenced as working fluids in the manner demonstrated above. These “standard library” descriptions cover the liquid, vapor, and dome regions up to the critical pressure.

Analysts often require a fluid that is not contained within the standard library, or require a more complete (perhaps supercritical) description of a library fluid, or require a simpler[‡] description than is provided by the library. All these purposes may be addressed by specifying user-defined fluids.

To illustrate the use of user-defined working fluids, assume that the working fluid in the above sample problem is air instead of steam. Air, while not contained within the standard library, is easily described as a perfect gas. Perfect gases are signaled by a fake four digit ASHRAE identifier starting with 8, such as “8000” or “8765.”

Fluid descriptions are meant to be developed once and reused many times, promoting the development of user libraries. For this reason, fluid descriptions are made in separate HEADER blocks. A simple description of air is as follows:

```
HEADER FPROP DATA,8729,SI,0.0
C SIMPLIFIED AIR (1 ATM, NEAR 300K (540R)) IN SI UNITS, DEGREES K
C
      MOLW = 28.96                $ MOLECULAR WEIGHT
```

* Even if SinapsPlus is not used, C&R provides a free plotter on the PC called EZ-XY[®].

† Results, including number of iterations, can vary slightly from version to version and even from host machine to host machine because of minor changes in internal numerical approaches, round-off errors, etc.

‡ A certain computational overhead exists for fluid descriptions: the wider the range of valid properties, the more expensive the analyses that use that description, even if the full range is not used within a particular model. Therefore, an all-liquid or all-gas analysis of a standard library fluid would execute considerably faster if the fluid description were simplified. A specialized external program named RAPPR (“rapid properties”) exists for just this purpose.

TWO CHAMBERS CONNECTED BY VALVED LINE

 MODEL = SINDA85
 FWDBCK

FLUID SUBMODEL NAME = CHAMBERS ; FLUID NO. = 718

 MAX TIME STEP = 5.902208E-05 ; LIMITING TANK = 1 REASON = VOLUME CHANGE LIMIT
 LAST TIME STEP = 5.073615E-05 VS. DTMAXF/DTMINF = 1.000000E+30 / 0.00000 ; AVERAGE TIME STEP = 6.050532E-05
 PROBLEM TIME
 TIMEN = 1.474999E-02 VS. TIMEND = 1.638889E-02

LUMP PARAMETER TABULATION FOR SUBMODEL CHAMBERS

| LUMP TYPE | TEMP | PRESSURE | QUALITY | VOID FRACT. | DENSITY | ENTHALPY | HEAT RATE | MASS RATE | ENERGY RATE |
|-----------|-------|----------|---------|-------------|------------|----------|-----------|-----------|-------------|
| 1 TANK | 335.6 | 23.76 | 1.000 | 1.000 | 5.0820E-02 | 1207. | 0.000 | -2.728 | -3029. |
| 2 TANK | 450.8 | 23.74 | 1.000 | 1.000 | 4.4108E-02 | 1262. | 0.000 | 2.728 | 3293. |
| 10 JUNC | 335.6 | 23.75 | 1.000 | 1.000 | 5.0808E-02 | 1207. | 0.000 | 0.000 | -1.1844E-04 |

TWO CHAMBERS CONNECTED BY VALVED LINE

 MODEL = SINDA85
 FWDBCK

FLUID SUBMODEL NAME = CHAMBERS ; FLUID NO. = 718

 MAX TIME STEP = 5.902208E-05 ; LIMITING TANK = 1 REASON = VOLUME CHANGE LIMIT
 LAST TIME STEP = 5.073615E-05 VS. DTMAXF/DTMINF = 1.000000E+30 / 0.00000 ; AVERAGE TIME STEP = 6.050532E-05
 PROBLEM TIME
 TIMEN = 1.474999E-02 VS. TIMEND = 1.638889E-02

PATH PARAMETER TABULATION FOR SUBMODEL CHAMBERS

| PATH TYPE | LMP 1 | LMP 2 | DUP I | DUP J | STAT | XL UPSTRM | FLOWRATE | DELTA PRES | REYNOLDS | MACH | REGIME |
|-----------|-------|-------|-------|-------|------|-----------|----------|------------|----------|--------|---------|
| 10 STUBE | 1 | 10 | 1.0 | 1.0 | NORM | 1.000 | 2.728 | 5.8939E-03 | 2360. | 0.007x | 1 PHASE |
| 20 LOSS | 10 | 2 | 1.0 | 1.0 | NORM | 1.000 | 2.728 | 1.3181E-02 | | 0.013* | |

```

CP      = 1.005E3          $ SPECIFIC HEAT
K       = 26.1E-3         $ CONDUCTIVITY
V       = 18.5E-6         $ VISCOSITY

```

In the subblock formed by the HEADER FPROP DATA, the user ID of the desired fluid is named, along with the local unit system. In this case, the fluid description is provided using standard SI units with degrees Kelvin. The zero “0.0” at the end of the first line is the value of absolute zero (“ABSZRO” in SINDA/FLUINT) in the local unit system. If degrees centigrade had been desired, the last number would have been input as -273.15. SINDA/FLUINT will then convert the fluid properties into the master model unit set if required. In other words, the unit system within a fluid properties block need not match that of the model in which it is used.

“MOLW” is the average molecular weight of the gas. “CP” is the specific heat, “V” is the dynamic viscosity, and “K” is the thermal conductivity. Since they are input as singular values in the above block, these properties will remain constant for all temperatures and pressures.

In order to make the properties vary linearly with temperature, a reference temperature “TREF” can be specified, along with a slope or temperature derivative for each property at that temperature:

```

HEADER FPROP DATA,8729,SI,0.0
C SIMPLIFIED AIR (1 ATM, NEAR 300K (540R))
C
MOLW = 28.96          $ MOLECULAR WEIGHT
TREF = 300.0         $ REFERENCE TEMPERATURE
CP   = 1.005E3, CPTC = 0.03    $ SPEC HEAT AT TREF, CP TEMP COEF
K    = 26.1E-3, KTC  = 8.0E-5  $ COND. AT TREF, PLUS TEMP COEF
V    = 18.5E-6, VTC  = 0.051E-6 $ VISC. AT TREF, PLUS TEMP COEF

```

More complete variations may be specified for any or all properties in the form of *bivariate* arrays (e.g. temperature/property pairs such as T_1 , k_1 , T_2 , k_2 , etc.) as follows:

```

HEADER FPROP DATA,8729,SI,0.0
MOLW = 28.96
TMIN = 90.0, TMAX = 1000.0
AT,V, 90.0,6.35E-6, 100.0,7.06E-6, 110.0,7.75E-6, 120.0,8.43E-6,
      130.0,9.09E-6, 140.0,9.74E-6, 160.0,11.0E-6, 200.0,13.4E-6,
      220.0,14.5E-6, 320.0,19.5E-6, 340.0,20.4E-6, 380.0,22.1E-6,
      400.0,22.9E-6, 500.0,26.8E-6, 600.0,30.3E-6
AT,K, 90.0,8.3E-3, 100.0,9.2E-3, 110.0,10.2E-3, 150.0,13.8E-3,
      160.0,14.6E-3, 180.0,16.4E-3, 200.0,18.1E-3, 220.0,19.8E-3,
      240.0,21.5E-3, 300.0,26.1E-3, 400.0,33.1E-3, 500.0,39.5E-3,
      600.0,45.6E-3
AT,CP,90.0,1.002E3, 200.0,1.002E3, 280.0,1.004E3, 340.0,1.007E3
      400.0,1.013E3, 500.0,1.029E3, 600.0,1.051E3

```

The above block might be placed into a separate file named air.inc. It can then be added to any input file using the INSERT directive within that input file:

```
INSERT air.inc
```

Where “I” must be placed in column 1. If the file `air.inc` is not contained within the same sub-directory or folder as the input file, the full path name (Unix, DOS, etc.) can be specified. INSERT files help promote the creation of user libraries of fluid and material property libraries, although they have many other uses as well.

Once a fluid has been described and verified, it may be used in one or more fluid submodels. To invoke the above fluid description in the sample problem:

```
INSERT air.inc
HEADER FLOW DATA, CHAMBERS, FID = 8729
C SET LUMP DEFAULTS
LU DEF,PL = pinit          $ 11 PSIA INITIAL PRESSURE
... {et cetera} ...
```

Variation 4 (below) explores working fluid mixtures.

Variation 2: Convective Heat Transfer, Duct Macros, and More

(For this sample problem extension, readers are assumed to be familiar with basic SINDA thermal networks, and/or to have read the corresponding SINDA introduction.)

What if the fluid in the line connecting the two vessels were not adiabatic? Suppose the line consisted of an insulated aluminum ($r = 170 \text{ lb}_m/\text{ft}^3$, $k = 95 \text{ BTU/hr-ft-F}$, $C_p = 0.21 \text{ BTU/lb}_m$) pipe with an outer diameter of 0.6 in. The initial temperature is 300°F.

Registers--The following registers are added to the model in preparation for later inputs:

```
dens = 170.0
cond = 95.0
cp   = 0.21
tinitw = 300.0
outerd = 0.6/12.0
apipe = 0.25*pi*(outerd^2 - diam^2)
```

where *apipe* is the cross-sectional area of the pipe for purposes of calculating axial conduction (if applicable) and nodal capacitance. Other registers are more obvious: *dens* for the density of the pipe material, *cond* for its conductivity, *cp* for its specific heat, *tinitw* for the initial temperature of the pipe, and *outerd* for the outer diameter (OD) of the pipe. This list is in addition to the previously defined registers:

```
length = 10.0
diam = 0.5/12.0
area = 0.25*pi*diam**2
throat = 0.1/144.0
pinit = 11.0
tinit1 = 200.0
tinit2 = 400.0
```

Thermal Submodels and Submodel Usage--Even though the pipe is externally insulated, thermal interactions will exist between the wall mass and the fluid. To model the mass of the pipe requires a different type of network: a traditional SINDA thermal (node and conductor) submodel. Finite masses may be represented by *diffusion nodes*, and conductive heat paths by *linear conductors*.

Models may be composed of collections of thermal and/or fluid submodels. Common uses of submodels include:

1. Combined models. Submodels enable SINDA/FLUINT models to be combined without internal numbering or control conflicts.
2. Organization. Even if a single analyst were building the entire model, it is convenient to use submodels for improved organization and better self-documentation. This is perhaps the most frequent use of submodels.
3. Dynamic model variations. Submodels may be dynamically added or deleted from the solution as needed to model changing geometries, materials, boundary conditions, assumptions, etc. To change the current configuration, a new BUILD and/or BUILDF statement is issued defining the new set of active submodels. Any submodels not currently defined as active are ignored by subsequent analyses, and any conductors or ties that extend to nodes or lumps in inactive submodels are also ignored.

Single Node--If temperature gradients along the pipe can be neglected, then a single diffusion node (submodel "WALL", node number 1) can be used to represent the isothermal pipe wall mass. Nodes are defined in their own unique HEADER block as follows:

```
HEADER NODE DATA, WALL
C   N#, Ti, density * Cp * Length * X-sectional Area
    1, tinitw, dens*cp*length*apipe
```

where the above information represents the node number, the initial temperature, and the nodal capacitance (product of density, volume, and specific heat).

Within HEADER FLOW DATA, a tie or heat transfer connection can then be made to the wall node. The valve inlet junction (#10) is the logical place for such heat energy ingress or egress to take place. An "HTN" convection tie can be made that will automatically invoke convection heat transfer calculations. Such a tie requires the specification of a tie identifier, a lump and a node to which it should attach, and a tube or STUBE connector which should be used to provide flowrate, shape, and size information for the heat transfer passage:

```
C Tie #99 connects lump #10 to thermal node #1 in submodel wall, and
C path #10 (an STUBE) defines the shape, size, and flowrate:
T HTN, 99, 10, WALL.1, 10
```

Alternatively, an "HTNS" convection tie can be made. This type of tie effects an LMTD (log mean temperature difference) solution which, when used properly, can yield better results in single-phase models with coarser spatial resolution (i.e., fewer nodes and lumps):

```
C Tie #99 connects PATH #10 to thermal node #1 in submodel wall
T HTNS, 99, 10, WALL.1
```

The difference between these two types of ties is subtle. HTN ties represent a more true finite-difference lumped parameter assumption: in order to capture an important property gradient, more elements must be used. When too few elements are chosen, however, the total heat transfer rate is typically underestimated. In contrast, HTNS ties are segment-oriented: they assume an exponential temperature profile within each fluid section. With HTN ties, lumps are treated as subsections of a line. With HTNS ties, lumps are treated merely as end points: as an inlet state and an outlet state. The node is assumed to represent the average wall temperature along the entire length of the segment. Therefore, in the above statement, the tie is made to path #10, and it will therefore add/extract heat to the lump that is

downstream of that path (e.g., junction #10). If the flowrate in the line were ever to reverse (which never happens in this model), the HTNS tie would automatically “jump” to the current downstream lump: Tank #1. HTN ties and other finite difference style ties remain attached to the designated lump.

In addition to forced convection ties, user-defined ties may be made. These allow the analyst to specify (and perhaps calculate) the heat transfer conductance themselves. Once again, either finite difference style (HTU) or segment-oriented (HTUS) ties are available.

Independent of the tie method chosen, a few more inputs would be required to complete this model. First, a TIETAB (tie tabulation) call is made in the existing block, which will print wall node information along with tie conductances and heat rates:

```

HEADER OUTPUT CALLS, CHAMBRs      $ DESIGNATE OUTPUT OPERATIONS
CALL LMPTAB('ALL')                $ LUMP TABULATION
CALL TIETAB('ALL')                $ TIE TABULATION
CALL PHTAB('ALL')                 $ PATH TABULATION
IF(PL1 .GE. 30.0D0 .OR.           $ LOGIC TO TERMINATE TRANSIENT
.   TL1 .GE. 450.0 .OR.
.   TL2 .GE. 450.0) TIMEND = TIMEN

```

Next, the existence of the new thermal submodel must be declared by a new build statement in OPERATIONS (“BUILD” for thermal submodels and “BUILDF” for fluid submodels):

```

HEADER OPERATIONS
BUILDF ALL
BUILD ALL
CALL TRANSIENT          $ START A TRANSIENT ANALYSIS

```

If this new statement were missing and the wall model were not built into the current configuration, any input ties would be ignored, and the previous (adiabatic) case would result. Similarly, if the wall model were built but the fluid model were not built, the wall node would simply progress through the transient event without any change in temperature. BUILD and BUILDF statements can be made repeated times within a single OPERATIONS as needed to analyze various combinations of configurations.

Axially Discretized Line--Unfortunately, because of the large temperature ranges in the model, and the long thin aspect ratio of the line, the assumption of an isothermal line wall may not be valid. The line will therefore be subdivided into 5 smaller nodes (numbered 1 through 5), each connected axially by linear conductors representing axial conduction along the line. Even if this term is small, SINDA nodes must be have at least one conductor, so adding the axial conduction satisfies this requirement.

SINDA, like FLUENT, provides for GEN commands to generate multiple elements. The newly updated WALL network blocks therefore become:

```

HEADER NODE DATA, WALL
C   GEN N#, #N, NINC, Ti, density*Cp * Length * X-sectional Area/number
    GEN 1, 5, 1, tinitw, dens*cp*length*apipe/5.0
HEADER CONDUCTOR DATA, WALL
C GENERATE 4 CONDUCTORS (1,2,3,4) BETWEEN 5 NODES (1,2,3,4,5)
C GEN G#, #G, GINC, NA#, NAINC, NB#, NBINC, cond.*X-sectional Area/length
    GEN 1,4,1, 1,1,      2,1,      cond*apipe/(length/5.0)

```

Analogous to the GEN commands, FLUINT features a single command subblock that generates lumps, paths, and ties representing a continuous duct segment. Replacing STUBE #10 with a string of lumps and paths that corresponds to the nodes in WALL is achieved by the following command:

```
C CENTERED 5 SEGMENT HX MACRO #1 REPRESENTING LINE.
C START WITH LUMP (JUNCTION) 100, PATH (STUBE) 100, TIE #1, ATTACHED TO
C STARTING NODE WALL.1, GENERATE MACRO THAT CONNECTS LUMP #1 TO #10:
C
M HX,1,C,100,100,1,WALL.1, 1,10, NSEG = 5
    DHS = diam          $ HYDRAULIC DIAMETER
    TLENT = length      $ TOTAL LENGTH
    LU = JUNC, PA = STUBE $ USE JUNCTIONS AND STUBES THROUGHOUT
```

This macrocommand generates junctions 100 through 104, STUBE connectors 100 through 105, and ties 1 through 5 (default increments are unity).

Unfortunately, this macrocommand is not completely equivalent to the replaced STUBE #10 since that path also had a K-factor of 0.5 applied to represent the inlet losses. Since adding “FK=0.5” in the above macrocommand subblock would then add this factor to all six macro paths, a single line of logic will instead be added to OPERATIONS before the transient routine is initiated. (This is not necessary when using *SinapsPlus*[®] or *FloCAD*[®], since those codes enable individual paths and lumps in a macro to be edited.) The new block becomes:

```
HEADER OPERATIONS
BUILDF ALL
BUILD ALL
    CHAMBR.S.FK100 = 0.5    $ INITIALIZE INLET LOSS ON FIRST PATH
    CALL TRANSIENT        $ START A TRANSIENT ANALYSIS
```

Since OPERATIONS is executed sequentially, the value of FK for path 100 in fluid submodel CHAMBR.S (“CHAMBR.S.FK100”) is initialized to 0.5 before the transient solution begins. “CHAMBR.S.FK100” could be simplified to “FK100” if SINDA/FLUINT knew which submodel name applied. The default submodel name can be set using the DEFMOD command as follows:

```
HEADER OPERATIONS
BUILDF ALL
BUILD ALL
DEFMOD CHAMBR.S
    FK100 = 0.5          $ INITIALIZE INLET LOSS ON FIRST PATH
    CALL TRANSIENT      $ START A TRANSIENT ANALYSIS
```

Notice that submodel prefixes and DEFMOD commands were not necessary in the logic that was listed at the end of the OUTPUT CALLS block (e.g., “PL1” etc.). The reason is because that block was already specific to the submodel CHAMBR.S. If a wall node temperature had been referenced within that block, or some variable from another fluid submodel, then submodel prefix would have been required (e.g., “WALL.T3”).

Complete Input File and Results--The complete input file for the expansion of the original problem to include the mass of the tube wall is as follows:

```

HEADER OPTIONS DATA
TITLE TWO CHAMBERS WITH VALVED CONNECTING LINE
      OUTPUT = chambers.out
C
HEADER REGISTER DATA
      length = 10.0
      diam = 0.5/12.0
      area = 0.25*pi*diam**2
      throat = 0.1/144.0
      pinit = 11.0
      tinit1 = 200.0
      tinit2 = 400.0
      dens = 170.0
      cond = 95.0
      cp = 0.21
      tinitw = 300.0
      outerd = 0.6/12.0
      apipe = 0.25*pi*(outerd^2 - diam^2)
C
HEADER FLOW DATA, CHAMBERS
      FID = 718          $ WATER IS THE WORKING FLUID
C SET LUMP DEFAULTS
LU DEF,PL = pinit      $ 11 PSIA INITIAL PRESSURE
      TL = tinit1, XL = 1.0 $ 200 DEG F INIT TEMP., ALL GAS/VAPOR
      VOL = 1.0          $ 1 CU FT VOLUME IN TANKS
C PATH DEFAULTS
PA DEF, FR= 0.0       $ ZERO INITIAL FLOWRATE
      DH = diam          $ 1/2 INCH DIA. (CIRC AREA UNLESS SPECIFIED)
C DEFINE MODEL
LU TANK,1, VDOT = -1.0*60.0 $ VESSEL #1, SHRINKING WITH TIME
      LSTAT = STAG      $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
C
C CENTERED 5 SEGMENT HX MACRO #1 REPRESENTING LINE.
C START WITH LUMP (JUNCTION) 100, PATH (STUBE) 100, TIE #1, ATTACHED TO
C STARTING NODE WALL.1, GENERATE MACRO THAT CONNECTS LUMP #1 TO #10:
M HX,1,C,100,100,1,WALL.1, 1,10, NSEG = 5
      DHS = diam        $ HYDRAULIC DIAMETER
      TLENT = length    $ TOTAL LENGTH
      LU = JUNC, PA = STUBE $ USE JUNCTIONS AND STUBES THROUGHOUT
C
LU JUNC,10            $ LINE OUTLET / VALVE INLET
PA CONN,20,10,2      $ VALVE MODEL: PATH 20
      DEV = LOSS        $ GENERIC LOSS
      FK = 20.0         $ VALVE LOSS
      APTH = throat     $ THROAT AREA
LU TANK,2, TL = tinit2 $ VESSEL #2, INITIALLY AT 400 DEG F

```

```

        LSTAT = STAG           $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
C
HEADER CONTROL DATA, GLOBAL
C ENG UNITS (ft-F-lb/hr-psia) will be used:
    UID = ENG
    TIMEND = 59.0/3600.0  $ RUN FOR 59 SECONDS
C
HEADER OPERATIONS
BUILD F ALL
BUILD ALL
    CHAMBR5.FK100 = 0.5    $ INITIALIZE INLET LOSS ON FIRST PATH
    CALL TRANSIENT        $ START A TRANSIENT ANALYSIS
C
HEADER NODE DATA, WALL
C    GEN N#, #N, NINC, Ti, density*Cp* Length * X-sectional Area/number
    GEN 1, 5, 1, tinitw, dens*cp*length*apipe/5.0
HEADER CONDUCTOR DATA, WALL
C GENERATE 4 CONDUCTORS (1,2,3,4) BETWEEN 5 NODES (1,2,3,4,5)
C GEN G#, #G, GINC, NA#, NAINC, NB#, NBINC, cond.*X-sectional Area/length
    GEN 1,4,1, 1,1,      2,1,      cond*apipe/(length/5.0)
C
HEADER OUTPUT CALLS, CHAMBR5    $ DESIGNATE OUTPUT OPERATIONS
    CALL LMPTAB('ALL')          $ LUMP TABULATION
    CALL TIETAB('ALL')          $ TIE TABULATION
    CALL PHTAB('ALL')           $ PATH TABULATION
    IF(PL1 .GE. 30.0D0 .OR.     $ STOP PROBLEM LOGIC
      . TL1 .GE. 450.0 .OR.
      . TL2 .GE. 450.0) TIMEND = TIMEN
END OF DATA

```

Figure 4 displays the results of this analysis in terms of temperature histories for several important points in the model. Since the tube mass was initially warmer than the fluid in vessel #1, it acts to heat the fluid in this line during the transient event. Because warmer gas is injected into Vessel #2, it takes even less time (39 seconds) to reach the criterion of 450°F than when the pipe wall was neglected. By the end of the event, tank #1 has almost warmed (by compression) up to the wall temperature, which has stayed relatively constant due to the low gas flowrates and low film coefficients.

In retrospect, because of the brevity of the event compared to the thermal mass of the pipe, no significant temperature gradient is ever developed in the wall: a single node would have sufficed in this case. Nevertheless, resolution *was* required in the fluid line since large temperature gradients did occur there. In this particular case, the use of an HTNS tie may have eliminated even this resolution requirement.

Variation 3: Two-Phase Flow (Pure Substance)

If the initial pipe wall temperature (*tinitw*) in the above problem (Variation 2) were 150°F instead of 300°F, then condensation would occur in the connecting line, and two-phase flow would result. For this variation, the valve in the line remains shut at all times, and the line is initially full of vapor at the same temperature as Tank #1 (e.g., considerably warmer than the wall around it). Thus, rapid cooling and then condensation will occur in the connecting

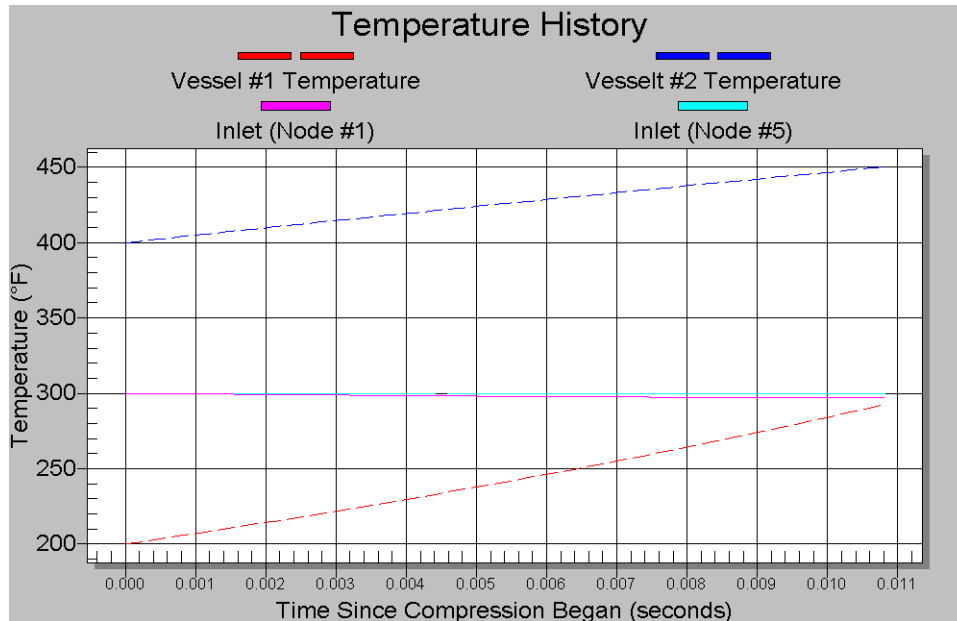


Figure 4: Temperature Histories for Model Adding Tube Wall Mass

line, gradually filling the connecting line volume with liquid. This effect will help to reduce the pressure rise that occurs as Tank #1 is compressed.

Two-phase flow will be modeled automatically by the code as the fluid temperatures drop below the local saturation condition. Nonetheless, two-phase flow not only introduces many more modeling options (and therefore decisions), its existence may render invalid some of the previous decisions. The purpose of this section is to discuss these options and assumptions, rather than to describe the details of the resulting model.

By default, a Rohsenow-Traviss condensation correlation is invoked to calculate film coefficients within the line. Since these coefficients are much larger than the previous single-phase gas case, the wall temperatures change more rapidly and larger gradients develop.

Also by default, the two-phase flow is assumed homogenous and flow regime mapping is applied. Other two-phase pressure drop calculations and correlations may be easily invoked via the "IPDC" descriptor for tubes and STUBE connectors, which could be added to the HX subblock:

```

IPDC = 1           $ McAdam's Homogenous
IPDC = 2           $ Lockhart Martinelli
IPDC = 3           $ Baroczy's
IPDC = 4           $ Friedel's
IDPC = 5           $ Whalley recommended combination of above
IPDC = 6           $ Default: Flow regime map based estimates

```

The last option, pressure drop estimates based on flow regimes, deserves a little attention. By default, one of four regimes (bubbly, slug, annular, and stratified) is predicted based on current flow conditions (void fractions, flowrates, etc.) and orientations with respect to a body force, if any. Since no body force vector nor lump coordinates (which provide orientation data for flow regime mapping) have been provided, a "zero gravity" flow regime map will

result. By applying a body force acceleration vector without providing lump coordinates, all tubes and STUBE connectors are assumed to be perpendicular to the vector (i.e., horizontal). To specify that the acceleration along the Z axis is due to gravity, the following line is used:

```
ACCELZ= grav          $ OR ACCELY OR ACCELX; Units: ft/hr2
```

(The above line may be placed in HEADER CONTROL DATA, GLOBAL or in OPERATIONS before calling the transient solution routine TRANSIENT. Use “gravsi” when using metric units.) Otherwise, vertical and slanted lines are modeled by providing the 1, 2, or 3D coordinate locations for the endpoint lumps (see Variation 6) along with the direction and magnitude of the acceleration.

The user may optionally override the mapping processing and select a regime directly, perhaps using test results.

Flow regime mapping is often a prerequisite for even higher fidelity two-phase modeling: automatic calculation of heat and mass transfer coefficients in nonequilibrium (temperatures and volumes of each phase differ) and dissolution/evolution, and slip flow (velocities of each phase differ).

In the original problem statement, the comparatively small volume of the line led quickly to the use of junctions instead of tanks to model that line. However, since the line will now be filled initially with vapor and will eventually fill with condensate, that previous assumption is much less valid. Because of the orders of magnitude difference in liquid versus vapor density, the changes in *mass* associated with the fluid in the line is no longer negligible compared to the rest of the system, and tanks must now be used. When tanks are used in two-phase systems, tubes become preferable to STUBEs because of enhanced stability (as discussed in the full User’s Manual).

When the steam from vessel #1 enters the cold line, it rapidly condenses. This deceleration of fast-moving vapor to slow-moving liquid counteracts friction, and can easily result in a net pumping action or *pressure recovery*. Also, the flow resistance of liquid is lower than vapor at the same mass flowrate. During the relatively violent event that follows, the flowrate in the line quickly grows to very high values, despite having a dead end in the form of a closed valve. In some respects, the presence of the piston serves only to initiate this action.

In fact, if the wall temperatures were much colder, flowrates can become so high that sonic limits are reached within the inlet of the connecting line. While FLUINT does not normally monitor for choked flow to avoid unnecessary computational expense, additional routines and options exist to watch for such occurrences, and/or to model choked conditions when they occur. (Refer to the last subsection for an example of choking simulation.)

The subtleties associated with a simple change in initial wall temperature should serve to illustrate the extra care that is sometimes needed in fluid system modeling, especially when two phases are present. Nonetheless, the reader should not arrive at the conclusion that two-phase analyses are intrinsically difficult, since they are performed routinely using SINDA/FLUINT. Rather, it should be noted that the violence of the resulting transient in this particular example is due in part to the artificially contrived initial conditions.

The above problem takes less than 10 CPU seconds to solve on most workstations. Interestingly, the liquid volume fraction in the line never grows beyond about two percent, so annular and stratified flows predominate. The rapid condensation in the line is partially offset by the compression heating of the fluid in Tank #1, which is the source of fluid entering the connecting line. The model runs for about 39 seconds, at which point the upper temperature

and/or pressure limits are exceeded and the logic stops the analysis. Figure 5 presents the results. Notice that the valve warms up faster than the inlet, despite the fact that the inlet fluid is substantially warmer than the fluid near the valve. The reason is that two-phase heat transfer coefficients near the valve are orders-of-magnitude larger than the single-phase film coefficients near the inlet.

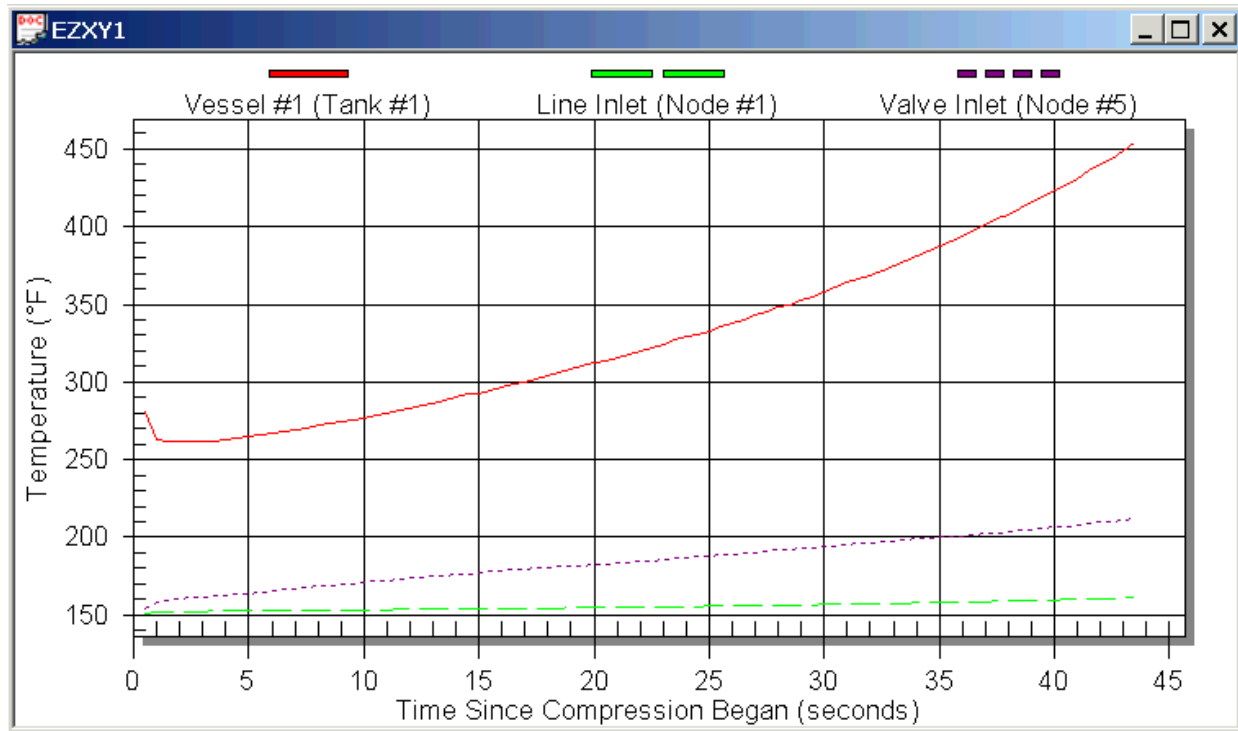


Figure 5: Temperature Histories for Two-Phase Model With Cold Wall, Closed Valve

Variation 4: Mixtures

In all previous analyses, the working fluids were all pure substances (or, in the case of air, could be treated as such) in one or two phases. Multiple constituents can also be tracked, with associated gradients in concentrations.

Assume that the gas in vessel #1 (in the original problem) is initially air (23% by mass oxygen, 77% nitrogen) while the gas in the second tank is initially 100% hydrogen. Properties for these gases are:

```

HEADER FPROP DATA,8728,SI,0.0
C N2 GAS (NEAR 1 ATM.)
C VALUES BELOW 77.36K ARE FOR VAPOR
  MOLW = 28.01,TMIN= 65.0,TMAX= 1000.0
AT,V, 65.0,4.40E-6
  77.36,5.44E-6,   80.0,5.59E-6,   85.0,5.9E-6,   90.0,6.22E-6
  95.0,6.54E-6,   100.0,6.87E-6,   105.0,7.19E-6,  110.0,7.52E-6
  115.0,7.83E-6,  120.0,8.15E-6,   125.0,8.0E-6,  126.2,8.65E-6
  130.0,8.78E-6,  140.0,9.4E-6,    180.0,11.8E-6,  200.0,12.9E-6,
  220.0,13.9E-6,  240.0,15.0E-6,   260.0,16.0E-6,  280.0,16.9E-6,

```

```

300.0,17.9E-6, 340.0,19.7E-6, 440.0,23.7E-6, 460.0,24.4E-6,
480.0,25.2E-6, 500.0,25.9E-6
AT,K, 65.0,6.1E-3, 75.0,7.1E-3, 77.36,7.4E-3, 80.0,7.6E-3,
85.0,8.0E-3, 90.0,8.5E-3, 95.0,8.9E-3, 100.0,9.4E-3,
105.0,9.8E-3, 110.0,10.3E-3, 115.0,10.7E-3, 125.0,11.7E-3,
130.0,12.1E-3, 150.0,13.9E-3, 160.0,14.7E-3, 180.0,16.5E-3,
200.0,18.3E-3, 220.0,19.9E-3, 240.0,21.5E-3, 300.0,26.0E-3
320.0,27.4E-3, 340.0,28.7E-3, 380.0,31.3E-3, 400.0,32.5E-3,
480.0,37.5E-3, 500.0,38.6E-3
AT,CP,65.0,1.039E3, 320.0,1.039E3, 380.0,1.042E3, 460.0,1.050E3
500.0,1.056E3

```

C

HEADER FPROP DATA,8732,SI,0.0

C OXYGEN AT 1 ATM

MOLW = 31.9988,TMIN= 80.0, TMAX= 700.0

```

AT,V, 80.0,6.27E-6, 100.0,7.68E-6, 120.0,9.12E-6, 130.0,9.85E-6
140.0,10.6E-6, 155.0,11.6E-6, 170.0,12.7E-6, 200.0,14.7E-6
300.0,20.7E-6, 400.0,25.9E-6, 500.0,30.5E-6, 600.0,34.7E-6
700.0,38.5E-6
AT,K, 100.0,9.1E-3, 120.0,0.0109, 130.0,0.0119, 140.0,0.0128
155.0,0.0142, 170.0,0.0156, 200.0,0.0182, 300.0,0.0267
400.0,0.0342, 500.0,0.0412, 600.0,0.0480, 700.0,0.0544
AT,CP,80.0,909.8, 170.0,909.8, 200.0,910.2, 300.0,918.4
400.0,941.5, 500.0,970.9, 600.0,1002., 700.0,1031.

```

C

HEADER FPROP DATA,8702,SI,0.0

C PARA H2 GAS (NEAR 1 ATM)

MOLW = 2.0159

TMIN = 14.0,TMAX= 1000.0

```

AT,V, 14.0,0.748E-6, 18.0,0.988E-6, 20.27,1.128E-6, 22.0,1.219E-6
26.0,1.424E-6, 30.0,1.621E-6, 35.0,1.856E-6, 40.0,2.02E-6
50.0,2.498E-6, 60.0,2.884E-6, 80.0,3.585E-6, 100.0,4.574E-6
120.0,5.408E-6, 160.0,6.3E-6, 200.0,6.901E-6, 240.0,7.419E-6
300.0,8.141E-6, 350.0,8.723E-6, 400.0,9.297E-6, 500.0,10.426E-6
AT,K, 14.0,0.01254, 18.0,0.01497, 20.27,0.01694, 24.0,0.0195
28.0,0.02247, 32.0,0.02534, 35.0,0.02741, 40.0,0.03088
50.0,0.03781, 60.0,0.04481, 80.0,0.06027, 100.0,0.08954
140.0,0.13613, 180.0,0.15565, 220.0,0.16341, 260.0,0.16914,
300.0,0.17591, 400.0,0.19745, 500.0,0.22128
AT,CP,14.0,10.54E3, 20.0,10.43E3, 24.0,10.39E3, 32.0,10.35E3
40.0,10.36E3, 50.0,10.49E3, 60.0,10.62E3, 80.0,11.72E3
100.0,13.4E3, 160.0,16.34E3, 200.0,16.07E3, 280.0,15.0E3
350.0,14.63E3, 400.0,14.55E3, 500.0,14.52E3

```

These fluids may all be combined into the submodel CHAMBERS. All are named within the subblock formed by the HEADER FLOW DATA card, using FIDx, where “x” is a single letter designating a fluid constituent:

```
HEADER FLOW DATA, CHAMBERS
      FIDO = 8732      $ OXYGEN IS "O"
      FIDN = 8728      $ NITROGEN IS "N"
      FIDH = 8702      $ HYDROGEN IS "H"
```

Initial concentration in the form of a mass fraction “XGx” are added to the lumps:

```
C SET LUMP DEFAULTS
LU DEF,PL = pinit      $ 11 PSIA INITIAL PRESSURE
      TL = tinit1, XL = 1.0 $ 200 DEG F INIT TEMP., ALL GAS/VAPOR
      VOL = 1.0        $ 1 CU FT VOLUME IN TANKS
C DEFAULT CONCENTRATIONS OF HYDROGEN, OXYGEN, AND NITROGEN
      XGH = 0.0, XGO = 0.23, XGN = 0.77
C PATH DEFAULTS
PA DEF, FR= 0.0      $ ZERO INITIAL FLOWRATE
      DH = diam        $ 1/2 INCH DIA. (CIRC AREA UNLESS SPECIFIED)
C DEFINE MODEL
LU TANK,1, VDOT = -1.0*60.0 $ VESSEL #1, SHRINKING WITH TIME
      LSTAT = STAG     $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
PA CONN,10,1,10     $ LINE: PATH 10 FROM 1 TO 10
      DEV = STUBE     $ "SHORT TUBE"
      TLEN = length   $ LENGTH
      FK = 0.5        $ INLET LOSS
LU JUNC,10          $ LINE OUTLET / VALVE INLET
PA CONN,20,10,2     $ VALVE MODEL: PATH 20
      DEV = LOSS      $ GENERIC LOSS
      FK = 20.0       $ VALVE LOSS
      APTH = throat   $ THROAT AREA
LU TANK,2, TL = tinit2 $ VESSEL #2, INITIALLY AT 400 DEG F
      LSTAT = STAG     $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
      XGH = 1.0, XGN = 0.0, XGO = 0.0$ ALL HYDROGEN INITIALLY
```

In addition, two new output routines are added to the OUTPUT CALLS block, one to tabulate constituents by lump (LMCTAB) and another to tabulate data by constituent (CNSTAB):

```
CALL LMCTAB('ALL')
CALL CNSTAB('ALL')
```

When these changes are made to the original sample problem, vessel #2 reaches 450°F in only 20 seconds, at which time it contains (by mass) 18% oxygen, 61% nitrogen, and 20% hydrogen (which is responsible for 78% of the partial pressure). The system pressure has risen to only 14.2 psia.

Variation 5: Condensible Mixtures

(This variation starts from the model that included the wall thermal submodel: Variation 2.)

In the previous variation, the original working fluid (water, fluid 718) was replaced by a non-condensable mixture. In this variation, the effects of adding a noncondensable gas (namely, air) to the original condensable water (used as the working fluid in Variation #2) are explored. The initial concentration of air in the gas is 0.1 (meaning 10% by mass).

Many of the variations are similar to those chosen in Variation #3: (1) the initial wall temperature is lowered to 150°F to attempt to force condensation to occur, (2) the internal line will be modeled with tanks and tubes (instead of junctions and STUBE connectors) to help track density changes, and (3) ACCELZ is again set to gravity (perpendicular to the line) to enable better flow regime predictions. Unlike Variation #3, however, the valve is not closed.

For the air description, we will use the simple one presented earlier, except that a new data value, "DIFV" (the diffusion volume) has been added as will be explained later:

```

HEADER FPROP DATA,8729,SI,0.0
C SIMPLIFIED AIR (1 ATM, NEAR 300K (540R))
C
      DIFV = 19.7                $ DIFFUSION VOLUME
      MOLW = 28.96              $ MOLECULAR WEIGHT
      TREF = 300.0              $ REFERENCE TEMPERATURE
      CP   = 1.005E3, CPTC = 0.03 $ SPEC HEAT AT TREF, CP TEMP COEF
      K    = 26.1E-3, KTC  = 8.0E-5 $ COND. AT TREF, PLUS TEMP COEF
      V    = 18.5E-6, VTC  = 0.051E-6 $ VISC. AT TREF, PLUS TEMP COEF

```

When a substance condenses in the presence of noncondensable gases, as is expected in this problem, noncondensable gas collects near the liquid/vapor interface forming a diffusion barrier that slows the condensation rate (and therefore results in larger temperature drops and lower effective film coefficients). This effect is automatically included by the code, providing that information necessary to calculate the diffusion constants for user-defined fluids has been provided in the form of DIFV. While DIFV is relatively easy to find or calculate for various fluids (as per guidance provided in the User's Manual), further discussion of that parameter is beyond the scope of this introduction. Otherwise, the diffusion effects are usually small compared to the "shifting" of the saturation condition due to the partial pressure of the noncondensable gas.

Results-- The addition of the gas eliminates the severe condensation rates that would otherwise occur in a pure substance, yet the existence of condensation keeps the tanks from hitting a pressure or temperature limit throughout the 1 minute transient event. By the end of the compression event, the quality of Vessel #2 is about 80%, meaning it is 20% liquid by mass although the volumetrically Vessel #2 is almost dry, and it contains about 12.5% gas by mass. Junctions could have been used instead of tanks in the connecting line since the density changes turned out to be small.

Temperature histories are shown in Figure 6. Vessel #2 quickly cools to the temperature of the incoming fluid (namely, Vessel #1) and then as condensation takes over, it tracks the temperature of the connecting line while Vessel #1 begins to heat up due to compression.

If data on the solubility of air in water were available (or could be estimated by the program using Raoult's Law), then the program would have included the effects of air dissolving into liquid water based on built-in correlations for dissolution mass transfer that rely on flow regime information. Extensive capabilities are available for dissolution/evolution modeling that are beyond the scope of this introductory document.

The multiple constituent modeling feature is currently limited to mixtures of up to 26 perfect gases and/or simple incompressible liquids with up to one real gas or condensable/volatile species.

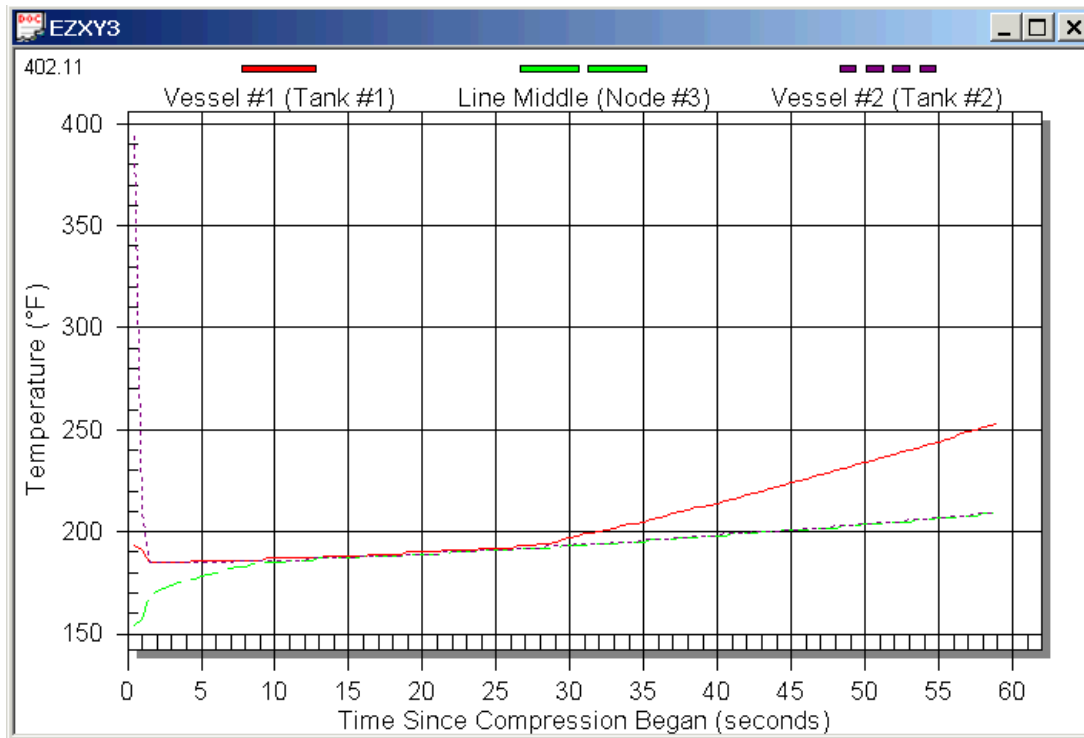


Figure 6: Temperature Histories for Model with Cold Wall and Gas, Open Value

Variation 6: Fast Transients, Choking, and User Logic

In order to illustrate the application of FLUINT to fast (hydrodynamic) transients such as acoustic waves, the original problem set-up must be changed slightly. In this variation, Vessel #2 is pressurized to 30 psia, but the valve is initially closed. The piston is in the closed position, meaning that the connecting line (which is assumed to be at 11 psia and 200°F) leads to a dead-end. The line is assumed to be adiabatic during this event.

The valve is opened rapidly (within 10ms) at time zero. The K-factor for the nearly closed valve is given to be 1000 (and 20 + 0.5 inlet loss when open). Both the K-factor and the throat area may be assumed to vary linearly as the valve opens. The transient pressure response of the system is desired.

Since the piston is in the closed position, the volume of vessel #1 is zero. This is affected simply by changing Tank #1 into Junction #1:

```
LU JUNC,1 $ VDOT = -1.0*60.0 $ VESSEL #1, (CLOSED END)
```

The connecting line is now dead-ended.

Since some resolution is needed within the line, it is convenient to start from the model used to illustrate heat transfer (i.e., the model with a wall network presented as Variation 2). However, for the purposes of this model, the WALL network is no longer needed. This can be affected simply by moving the BUILD statement in OPERATIONS after the call to the tran-

sient. (Alternatively, the thermal submodel can be removed, and the HX macro in FLOW DATA changed into the equivalent LINE macro, which generates no ties.) Also, the TIETAB call in OUTPUT CALLS can be removed or commented:

```

HEADER OPERATIONS
BUILDF ALL
      CALL TRANSIENT          $ START A TRANSIENT ANALYSIS
BUILD ALL
HEADER OUTPUT CALLS, CHAMBR5      $ DESIGNATE OUTPUT OPERATIONS
      CALL LMPTAB('ALL')      $ LUMP TABULATION
C      CALL TIETAB('ALL')     $ LUMP TABULATION
      CALL PTHTAB('ALL')     $ PATH TABULATION

```

If the line were modeled with junctions instead of tanks, the line would instantaneously pressurize to 30 psia when the valve were opened. Therefore, to capture the pressurization of the line and any associated acoustic waves, tanks will be used instead. Tubes are required instead of STUBE connectors in order to resolve any acoustic waves. The HX macro becomes:

```

C CENTERED 5 SEGMENT HX MACRO #1 REPRESENTING LINE.
C START WITH LUMP (TANK) 100, PATH (TUBE) 100, TIE #1, ATTACHED TO
C STARTING NODE WALL.1, GO FROM LUMP #1 TO #10:
M HX,1,C,100,100,1,WALL.1, 1,10, NSEG = 5
      DHS = diam          $ HYDRAULIC DIAMETER
      TLENT = length     $ TOTAL LENGTH
C      LU = JUNC, PA = STUBE $ USE TANKS AND TUBES THROUGHOUT (DEFAULT)
      TL = tinit1

```

FLUINT uses an implicit solution that is specifically designed to avoid resolution of fast transients, enabling long time steps. Nonetheless, if the time step is appropriately constrained then such events can be resolved. A routine named WAVLIM exists that helps calculate the maximum time step that can be taken in order to resolve such events:

```
CALL WAVLIM('CHAMBR5',DTEST)
```

where DTEST is a temporary Fortran variable containing the returned time step limit. Usually, a value smaller than this should be used for conservatism, say:

```
DTMAXF= 0.4*DTEST
```

where “DTMAXF” is a *control constant*, or user-specified maximum allowable time step for fluid submodels. Where should these statements be placed? If they are placed in OPERATIONS, they will only be executed once, and the data upon which WAVLIM is based may vary during the solution. As an aid in such cases, SINDA/FLUINT provides additional logic blocks which are called at prescribed times during the solution. The most common such block for fluid submodel manipulations is “FLOGIC 0,” which is called before each steady-state iteration and before each transient time step:

```

HEADER FLOGIC 0, CHAMBR5
      CALL WAVLIM('CHAMBR5',DTEST)
      DTMAXF= 0.4*DTEST

```

The frequency of which such blocks are executed assures that the calculations they perform are relevant, and that the data they manipulate is “fresh.”

A special connector path called a CTLVLV exists, which is like a LOSS connector except that it can be opened or closed by the user. Since the problem begins with the slight opening of the valve, a fully closed valve is therefore never really necessary, and the current LOSS connector is adequate. The opening of the valve (LOSS connector) may be simulated via logical instructions that are placed in the same logic block (FLOGIC 0) as above, or that can be specified in the input blocks as part of the internal spreadsheet interrelationships. The following logic manipulates the K-factor (FK) and throat area (AFTH) of the valve as a function of the current time, TIMEN:

```
AFTH20 = MAX( 1.0E-8, MIN(AOPEN, TIMEN*AOPEN/TOPEN) )
FK20 = MAX( OPENK, MIN(CLOSK, (TOPEN-TIMEN)*CLOSK/TOPEN) )
```

This logic uses several custom (user-supplied) variables named AOPEN (the valve throat area when opened), OPENK (the K-factor when open), CLOSK (the corresponding K-factor when nearly closed), and TOPEN, the time it takes to open the valve. These auxiliary variables may be declared and initialized as additional registers, which can then be used in input spreadsheet expressions or as Fortran variables in logic, or both:

HEADER REGISTER DATA

```
TOPEN = 10.0E-3/3600.0      $ TIME REQ'D TO OPEN VALVE
AOPEN = throat             $ OPEN THROAT AREA
OPENK = 20.5               $ OPENED K
CLOSK = 1.0E3              $ CLOSED K
```

Registers are available for inspection or modification in all logic blocks, and any changes to them will be reflected in any input expressions referring to them automatically.

One reason for declaring these constants as registers is because of the SINDA/FLUINT spell checker or *debug* feature. The spell checker makes sure that all variables and identifiers are declared and valid, since Fortran otherwise does not enforce such a rule and will simply use zero as values for misspelled variables, without warning. While the spell checker can be turned off, using it can help prevent errors that are difficult to trace. One side effect of placing variables in REGISTER DATA is that they then become valid names in the eyes of the spell checker. Otherwise, their presence would trigger an error message if the spell checker were enabled. Users can turn off the spell checker via a single command in OPTIONS DATA:

```
NODEBUG
```

It can also be turned on and off locally within a logic block.

When the valve first begins to open, the pressure difference across it is large enough such that choking occurs. In other words, Mach number reaches unity at the throat, and the flow through the valve cannot exceed the corresponding "critical" flowrate. As the valve opens more, and as the line pressure rises, the flowrate will eventually become limited by frictional losses rather than sonic constraints.

FLUINT choking calculations offer two different means of calculating critical flowrates when two-phases are present, although this choice is irrelevant in the current single-phase analysis. The user can also assume nonequilibrium expansion. (Nonequilibrium expansion assumes that no phase change or evolution/dissolution occurs--that the expansion event is simply too fast for such effects, and that the throat state is one that doesn't exist on equilibrium thermodynamic charts as a consequence.)

Finally, the end time needs to be adjusted to account for the small time scale of the event of interest:

`TIMEND = 50.0E-3/3600.0` `$ RUN FOR 50 MILLISECONDS`

Results--Figure 7 presents the flowrate and pressure profiles for the resulting event. Although the valve is fully opened by 10 ms, it appears to have been choked for most of that time. It takes about 30 ms for the pressure to reach 30 psia and for the flow through to valve to significantly diminish.

During the event, the steam column oscillates with a period of about 5 to 10 ms, visible in the plots as a sinusoidal waveform superimposed upon the pressure and flowrate histories. Actually, the exact period of this oscillation varies over the course of the event because the sound speed changes as the steam in the line warms up and becomes pressurized.

Variation 7: More Fluid Submodels: Heat Exchangers and Buoyancy Forces

(This variation starts from the model that included the wall thermal submodel: Variation 2.)

Suppose that the connecting line had been actively cooled by a second working fluid (perhaps pressurized liquid water). Assume that the previous tube wall represented the inner tube (annulus) of a tube-and-shell counterflow heat exchanger. The inner diameter of the new coolant annulus is 0.75 inches. Furthermore, assume that this heat exchanger were driven by natural recirculation: a large tank 10 ft above the rest of the system, and filled with water at 200°F, is plumbed to the annular section by 10 ft long by 0.5 inch lines (Figure 8). When the problem starts, the fluid in this second loop has achieved equilibrium circulation, with the wall initially at 300°F. The flow has established itself such that the coolant water flows from the valve toward vessel #1 (i.e., counterflow to the steam).

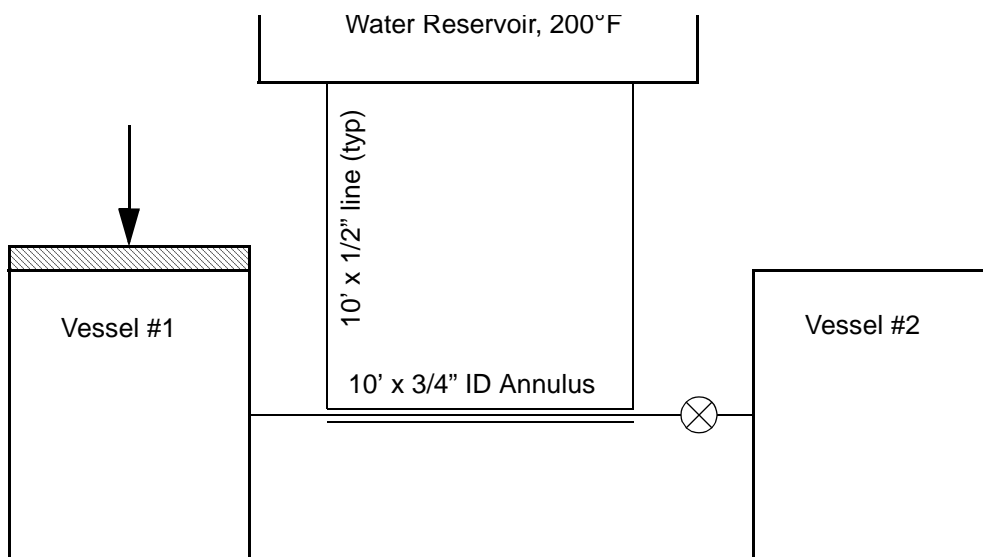


Figure 8: Schematic for Passive Circulation Shell and Tube Heat Exchanger

Although both fluid systems use the same working fluid and can therefore be contained within the same fluid submodel, the best approach is to define a new fluid submodel, which will be called CIRC. As long as pressure in this new loop remains high enough for the liquid

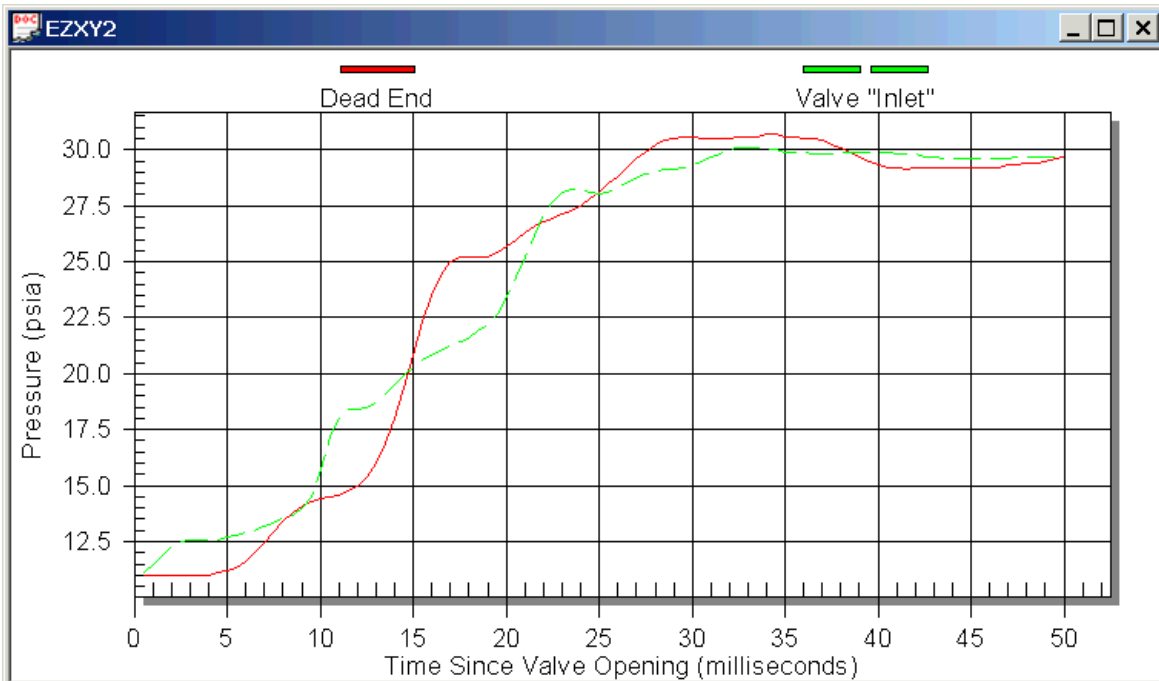
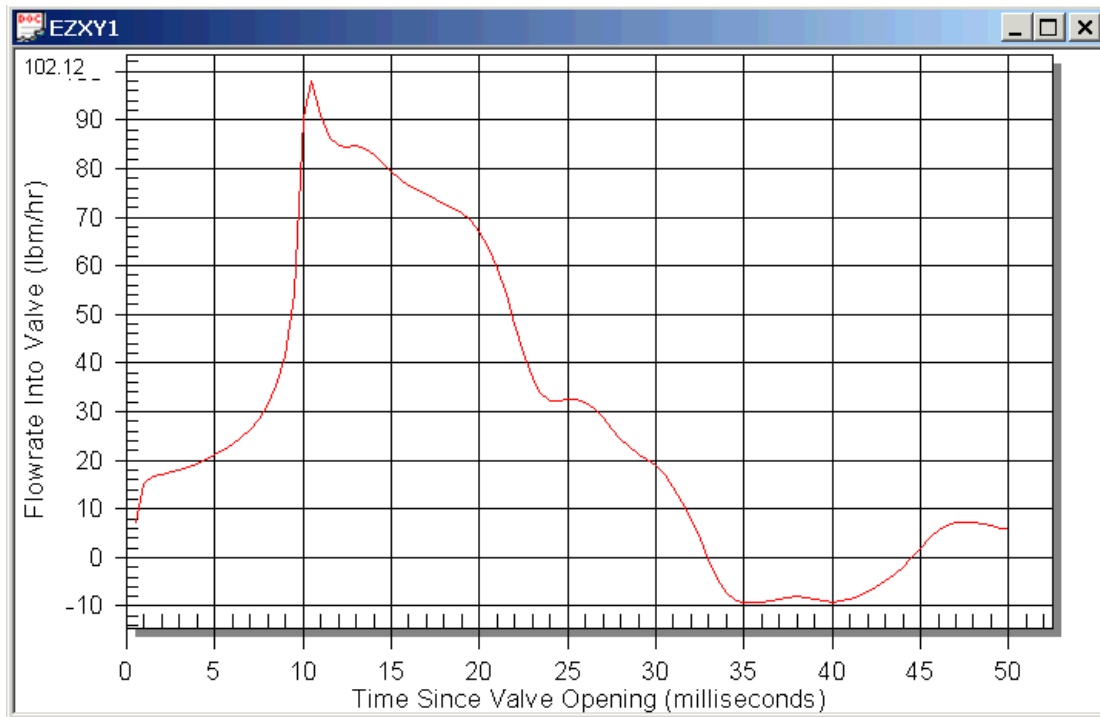


Figure 7: Flowrate and Pressure Histories for Valve Opening Event

to stay liquid, then a simple user-defined liquid (four digit fake ASHRAE number beginning with “9”) can be used to represent the fluid.

The following simplified liquid water description is output by the external program RAPPR (with minor editing):

```

HEADER FPROP DATA, 9718, ENG, 0.0
C THIS FLUID DESCRIPTION WAS CREATED BY RAPPR FOR FLUID 718
C PROPERTIES ARE ACCURATE FOR SATURATED LIQUID OVER FULL RANGE
C
      TMIN = 491.70001 , TMAX = 1150.0000
AT,V,      491.70001 , 4.3486681
           505.67493 , 3.3405128
           522.67493 , 2.5328307
           542.67493 , 1.9282620
           566.67493 , 1.4802542
           598.67493 , 1.1333355
           646.67493 , 0.87072533
           1150.0000 , 1.1891712
AT,K,      491.70001 , 0.33202705
           1150.0000 , 0.61937493
AT,D,      491.70001 , 62.451759
           971.67493 , 48.035072
           1150.0000 , 30.815065
AT,ST,     491.70001 , 4.81158635E-03
           708.67493 , 3.69433896E-03
           819.67493 , 2.83849891E-03
           1150.0000 , 5.88794137E-05
AT,CP,     491.70001 , 1.1114392
           531.67493 , 1.0746689
           712.67493 , 1.0225220
           1063.6749 , 1.4633179
           1144.2500 , 2.1658936
AT,COMP,   491.70001 , 2.81523830E-06
           708.67493 , 4.03973172E-06
           850.67493 , 5.77156106E-06

```

where arrays of properties versus temperature include V (dynamic viscosity), K (thermal conductivity, D (density), ST (surface tension), CP (specific heat), and COMP (compliance). Surface tension is only needed when used in two-phase mixtures, and the compressibility is only needed for waterhammer, choking, or speed of sound calculations. This information is placed in a file named water.inc, to be addressed by an INSERT directive. The fluid submodel CIRC will therefore use FID = 9718 within its HEADER FLOW DATA subblock.

The new submodel, CIRC, connects to the WALL nodes from the outside. Neglecting the gradient through the pipe wall, the new fluid submodel can attach directly to the existing wall nodes. Assuming that this water-cooled jacket is itself adiabatic on the outside, how should such an annular passage be modeled?

The flow area (“AF”) of the passage is $\pi(D_o^2 - D_i^2)/4$, where D_o is 0.75 inches and D_i is 0.6 inches (the OD of the original connecting line). The wetted perimeter is $\pi(D_o + D_i)$. By the definition of the hydraulic diameter (“DH”), $DH = 4A/P = (D_o - D_i)$. (An effective diameter,

DEFF, may also be specified for highly noncircular cross sections, but this detail is neglected for simplicity. DEFF defaults to the hydraulic diameter, DH.)

When convective heat transfer calculations are invoked, FLUINT assumes by default that the entire wetted perimeter is available for heat exchange. Since this is not true in this case, a *tie area fraction* is applied as a correction, reducing the heat transfer area by $D_i/(D_o + D_i)$.

Since the wall will cool down once the transient begins, the flowrate through the recirculation loop will gradually change. How fast does the liquid water move through this loop compared to the event duration? Based on preliminary analyses, the flowrate of the recirculation loop stays above about 60 lb_m/hour, meaning that the entire loop will be swept clear at least every 1.5 minutes or so. Since this time scale is on the order of the entire transient event, tanks should be used to represent the volumes within the recirculation line, so as not to neglect fluid lag effects. Similarly, the ascending line will be subdivided to better monitor these effects, since temperatures in that line have the most effect on the buoyancy driving force. Actually, since the line descending from the reservoir never changes temperature, its volume can be neglected and it can be represented by a single path.

Since hydrodynamic transients are unimportant, STUBE connectors may be used in place of tubes. A plenum is naturally used to model the reservoir.

Using the following new registers for convenience,

```
length2 = 10.0
diam2   = 0.5/12.0
od      = 0.75/12.0
id      = douter
```

the FLOW DATA portion of the new submodel becomes:

```
INSERT water.inc
HEADER FLOW DATA, CIRC, FID = 9718
C
LU DEF, PL = 100.0          $ DEFAULT PRESSURE
    TL = 200.0             $ DEFAULT TEMP
PA DEF, FR = 100.0        $ DEFAULT FLOWRATE
    TLEN = length2        $ DEFAULT LENGTH
    DH = diam2            $ DEFAULT DIAM
C
LU PLEN,1000, CZ = length2 $ 10 FT HIGH (ELEVATION)
    LSTAT = STAG          $ NEGLIGIBLE VELOCITIES: TREAT PL AS TOTAL
PA CONN,10,1000,10       $ DESCENDING LINE
    DEV = STUBE
    FK = 0.5              $ INLET LOSS
C
LU DEF, CZ = 0.0          $ LOWER LEVEL: ZERO ELEVATION
C
LU JUNC,10                $ INLET TO ANNULUS
C ANNULAR PASSAGE ADIABATIC OUTSIDE, CONVECTION INSIDE
M HX,1,C,104,105,5,WALL.5, 10,1, NSEG = 5          $ MATCH CHAMBERS
    LUINC = -1, PAINC = -1, TIINC = -1, NINC = -1   $ NEGATIVE INCR
    TL = 200.0, TLINC = 10.0                       $ SPEFICY INIT TEMP GRADIENT
    DHS = od-id                                     $ HYD DIAMETER
```

```

TLENT = length                $ TOTAL LENGTH
AFS = 0.25*pi*(od^2-id^2)    $ FLOW AREA
AFRACT = id/(od+id)          $ ONLY ID IS ACTIVE (AREA FRACT)
PA = STUBE                    $ USE STUBES (TANKS DEFAULT)
LU DEF, TL = 250.0            $ CHANGE DEFAULT TEMP TO 250
LU JUNC,1                     $ OUTLET OF ANNULUS
M LINE,2,C,21,21, 1,1000, NSEG = 5 $ ASCENDING LINE
    DHS = diam2                $ HYD DIA
    TLENT = length2            $ TOTAL LENGTH
    CZ = length2/5., CZINC = length2/5. $ VARY ELEVATION ALONG IT
    PA = STUBE                 $ USE STUBES (TANKS DEFAULT)

```

Notice the repeated use of “DEF” subblocks to define and then redefine default parameters for subsequent inputs. “LINE” is an HX-like duct macro that generates no ties, and hence is by default adiabatic (although ties can be added manually if desired).

Note also that “CZ” is the coordinate location or elevation of each lump on the Z axis. The acceleration along that axis is due to gravity, which in units of ft/hr² yields:

```

ACCELZ= 32.2*3600.0**2        $ or "accelz = grav"

```

The above line may be placed in HEADER CONTROL DATA, GLOBAL or in OPERATIONS before calling the transient solution routine TRANSIENT.

How can we calculate the initial flow through the recirculation loop, and initialize that loop without disturbing the other submodels, whose initial conditions are prescribed?

Using submodels, we can build a configuration inside OPERATIONS in which CHAMBR8 doesn't exist. Then, the wall submodel can be placed in a boundary state by calling the auxiliary routine DRPMOD or HTRMOD routines. Using DRPMOD results in the following:

```

HEADER OPERATIONS
BUILDF CON1, CIRC
BUILD CON1, WALL
    CALL DRPMOD('WALL')    $ SUSPEND WALL SUBMODEL AS BOUNDARY

```

Next, a steady state solution, which will act on CIRC using WALL as a boundary condition, is invoked by a call to the steady-state solution routine STEADY:

```

DEFMOD CIRC
    FK26 = 1.0              $ EXIT LOSS INTO RESERVOIR
    CALL STEADY              $ CALL FOR A STEADY STATE

```

Submodel WALL is now “awakened” using the routine ADDMOD, and a new BUILDF is issued to add CHAMBR8 to the list of active models before a transient is initiated:

```

BUILDF CONF, CHAMBR8, CIRC
    CALL ADDMOD('WALL')    $ AWAKEN WALL SUBMODEL
    CHAMBR8.FK100 = 0.5    $ INITIALIZE INLET LOSS ON FIRST PATH
    CALL TRANSIENT          $ START A TRANSIENT ANALYSIS

```

For completeness, the new OPERATIONS block is provided below:

```

HEADER OPERATIONS
BUILDF CON1, CIRC
BUILD CON1, WALL

```

```
CALL DRPMOD('WALL')    $ SUSPEND WALL SUBMODEL AS BOUNDARY
DEFMOD CIRC
FK26 = 1.0              $ EXIT LOSS INTO RESERVOIR
CALL STEADY             $ CALL FOR A STEADY STATE
BUILDF CONF, CHAMBR5, CIRC
CALL ADDMOD('WALL')    $ AWAKEN WALL SUBMODEL
CHAMBR5.FK100 = 0.5    $ INITIALIZE INLET LOSS ON FIRST PATH
CALL TRANSIENT         $ START A TRANSIENT ANALYSIS
```

Results Summary--Because of the cooling provided by the recirculation loop, the pressure and temperature limits are never exceeded: the run continues to an end time of 59 seconds. To continue modeling the system response, the volume of Tank #1 could have been allowed to shrink to an arbitrarily small size, at which point the VDOT (volume rate) could be set to zero to simulate the stopping of the piston as it reaches the bottom of vessel #1.

The initial flowrate in the recirculation line is about 230 lb_m/hr, dropping quickly to about 65 lb_m/hr as the sensible heat in the pipe wall, which is the driving force for the buoyancy-driven flow, is extracted. The pipe wall cools to about 200°F in 30 seconds, then slowly heats back up due to the warm gas flow.

More Information

If you have questions about the use or availability of SINDA/FLUINT, Sinaps*Plus*[®], EZ-XY[®], Thermal Desktop[®], RadCAD[®], or FloCAD[®] please contact:

C&R Technologies, Inc.
E-mail: info@crtech.com
Web site: www.crtech.com

The web site contains demonstration versions, on-line hypertext user's manuals, training materials, and other announcements. Additional working fluid descriptions are also available to users.