MELCOR Data and Control Utilities

PRESENTED BY

Larry Humphries

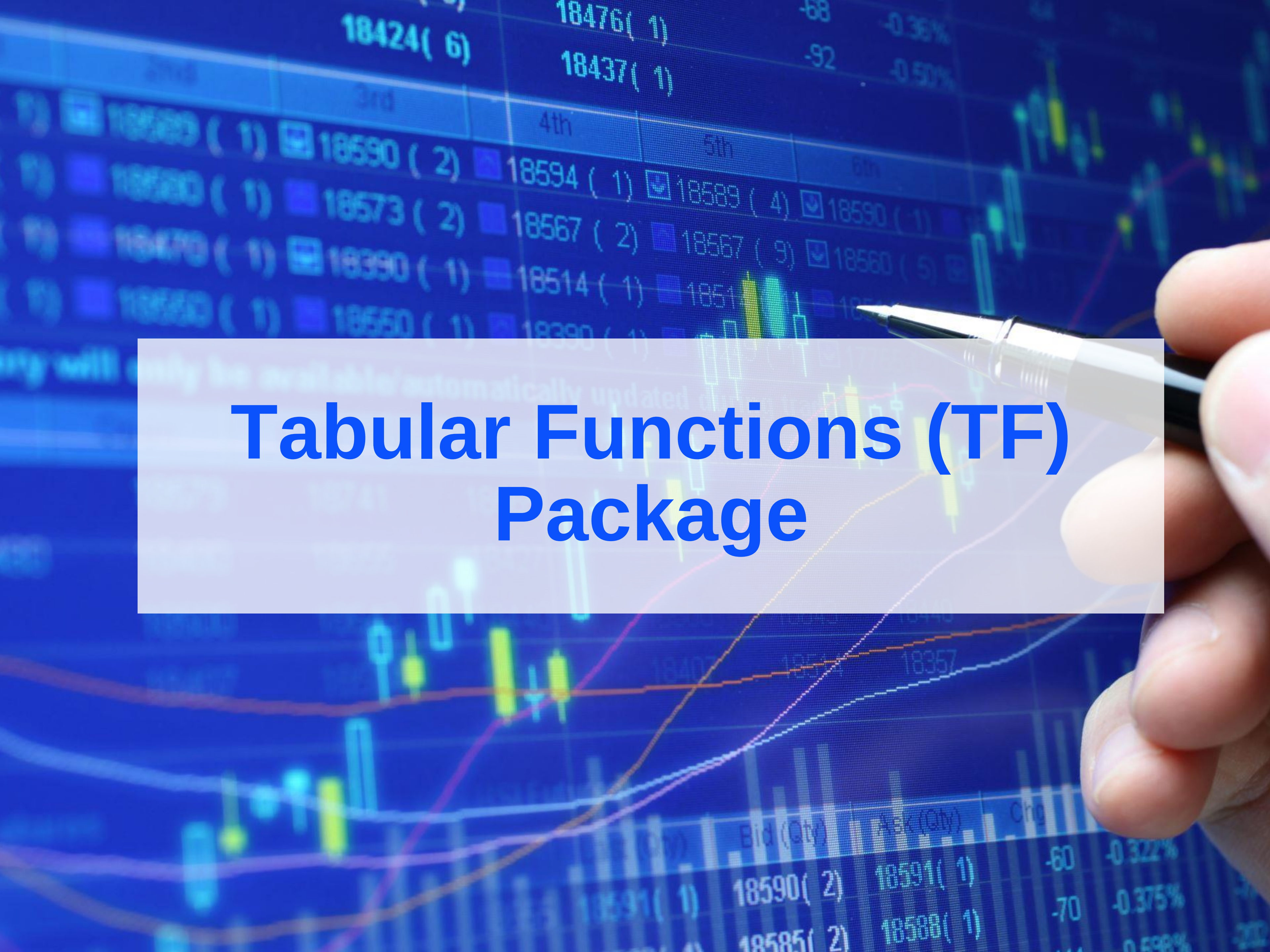
SAND2022-4172TR

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

MELCOR Data and Control Presentation Overview

- MELCOR provides data utility packages for performing commonly required functions
  - Handling of data (e.g., tabular input or output)
  - Evaluation of functions for variables and/or control logic
  - Materials properties
- This presentation covers MELCOR data and control packages
  - Tabular Functions (TF) Package: General interface to tabular data
  - External Data Files (EDF) Package: General interface to data files as input or output
  - Control Functions (CF) Package: General interface to control logic and user-defined functions
    - Includes recent improvement

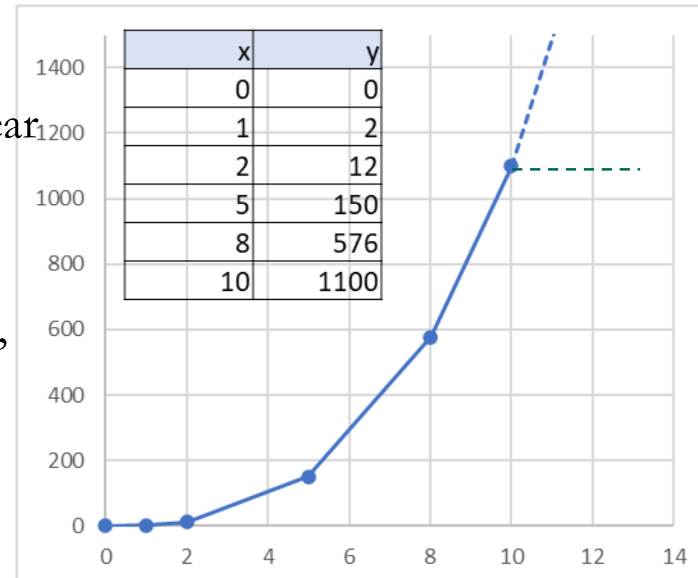
Tabular Functions (TF) Package



MELCOR Tabular Functions (TF) Package Overview

Tabular Function (TF) utility provides unified treatment

- Define 1-dimensional tables of data pairs for arbitrary independent and dependent variables
- Specify extrapolation conditions at the boundary
- Value between the specified data pairs generated by linear interpolation



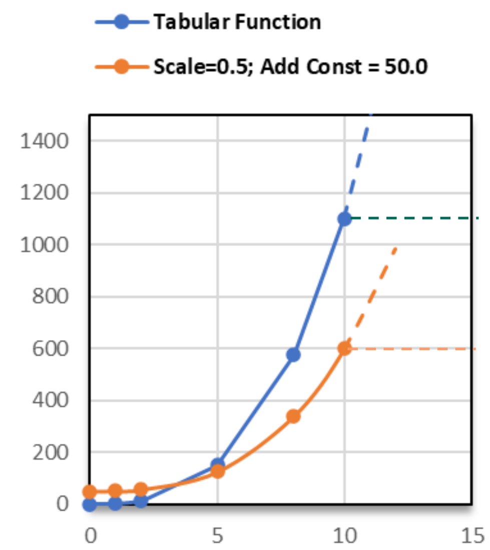
x	y
0	0
1	2
2	12
5	150
8	576
10	1100

Multiple MELCOR packages use tabular data

- Mass and/or energy sources for hydrodynamics (CVH), heat structures (HS), or aerosol/vapor fields (RN1), examples as:
  - Gas source/sink for fire and explosion in CVH
  - Aerosol sources for fire and explosion in RN
- Imposed time-dependent flow velocities in hydrodynamics (FL)
- Definition of time-specified volume conditions (CVH)
- Materials properties (MP)
- Definition of control functions (CF)

MELCOR Tabular Functions (TF) User Input

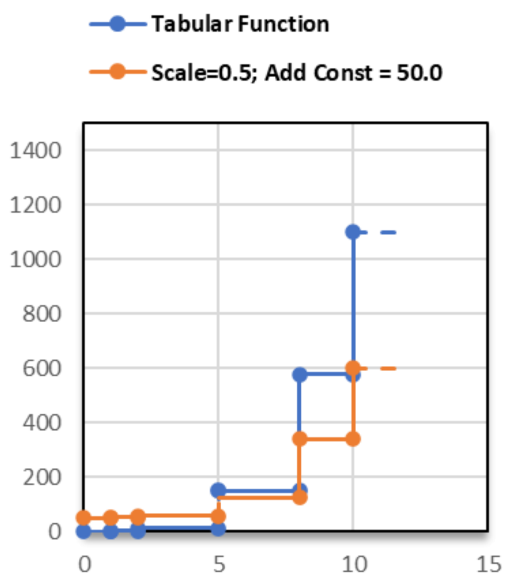
- **REQUIRED for each tabular function**
  - User-defined tabular function name
  - Number of data pairs (x,y) to define y=f(x)
  - Multiplicative scale factor
- **Optional for each tabular function**
  - Additive constant (default = 0)
  - Boundary condition for evaluation of x outside the range
    - Default is to extend the table with constant boundary value
    - Can also linearly extrapolate or treat as an error
    - Upper and lower limits independently specified
- **Calling package specifies tabular function type (e.g., velocity vs. time) and tabular function name**
- **Value returned:**
  - TF_n = Scale x $Table_n(x)$ + Additive Const.



MELCOR Tabular Functions (TF)
TF Input: Data Pairs

- **Function defined by (x,y) data pairs**
  - —Can be as few as one pair for constant value
- **Discontinuous (step) functions allowed, with the same x value in two (or more) pairs**
- **Generally, data pairs are entered in order of non-decreasing x**
  - —If there are no discontinuities, pairs can be input in any order and will be internally sorted

x	y
0	0
1	0
1	2
2	2
2	12
5	12
5	150
8	150
8	576
10	576
10	1100



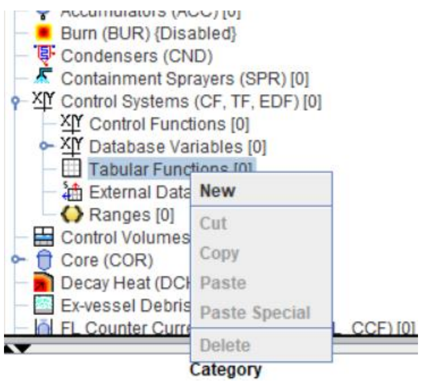
MELCOR Tabular Functions (TF)
Example TF Input

- **Input block for energy source in CVH volume**

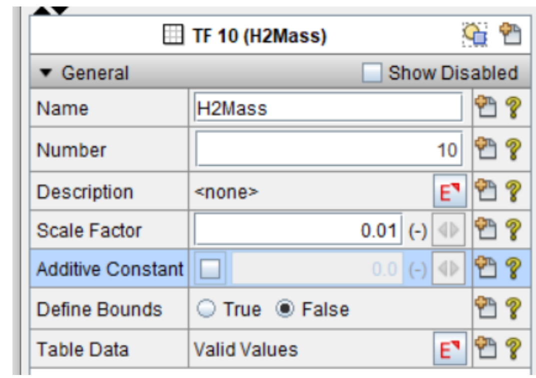
```
CV_ID CV456
!   CV_SOU table for source data
!    |         Energy to pool or atmosphere or mass of material
!    |         |   Rate or integral
!    |         |   |    Source of data (function of time)
!    |         |   |    |  TF name
!    v         vv  vvvv vv vvvv
CV_SOU 1 ! N  SourceInfo
           1  MASS  RATE TF ‘H2MASS’ ...
TF_INPUT
! TF_ID - tabular function definition
!  |  Name         Multiplier
!  |  |            |       Additive const. (optional)
!vvvv vvvvvvvvvvv  vvvvv   vvv
TF_ID ‘H2MASS’  0.01   0.0 ! Multiplier is desired MASS RATE
TF_TAB   1 ! NTFPAR   X    Y
                1    0.0  1.0 ! Constant value of 1.0
! (Value Returned = 0.01 x 1.0 + 0.0 = 0.01)
```

Steps for Adding TF in SNAP

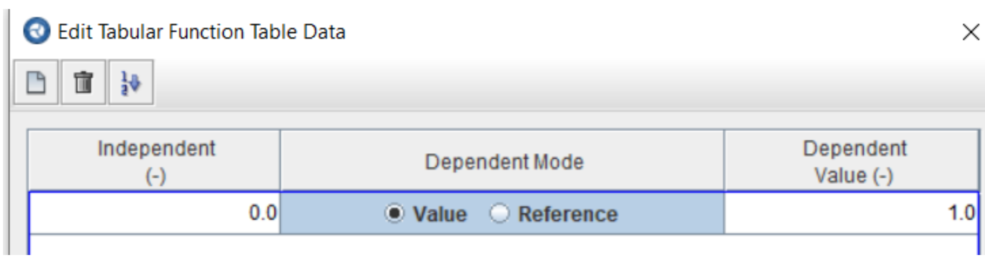
1) Create New Tabular Function



2) Provide Name, Number, Factors, Constant, Bounds



3) Add Data Pair(s)




```
TF_INPUT
! TF_ID – tabular function definition
!   |  Name           Multiplier
!   |  |              |        Additive const. (optional)
!vvvv vvvvvvvvvvv    vvvvv    vvv
TF_ID 'H2MASS'  0.01   0.0 ! Multiplier is desired MASS RATE
TF_TAB   1 ! NTFPAR   X    Y
                1    0.0  1.0 ! Constant value of 1.0
! (Value Returned = 0.01 x 1.0 + 0.0 = 0.01)
```

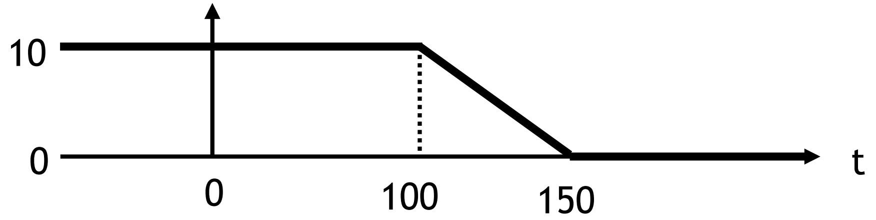
MELCOR Tabular Functions (TF) Example TF Input (2)

Input block for forced jet pump velocity

```
FL_VTM     1 !NFLT      FLNAME        NTFLAG  NFUN
              1      FlowPath151       TF     'Jet-V'
TF_INPUT
! TF_ID – tabular function definition
!   |  Name
!   |   |          Multiplier for table data
!vvvv vvvvvvv    vvvv
TF_ID 'Jet-V'    10.0 ! Multiplier is rated flow velocity
! Three points in table
          v
TF_TAB     3 ! NTFPAR   X        Y
               1     0.0      1.0
               2     100.0    1.0
               3     150.0    0.0
```

Time-dependent velocity for flowpath 'FlowPath151' described by a Tabular Function (TF) named 'Jet-V'

Default extrapolation option is to extend the table with constant value at lower and upper boundaries



Variable Input and Named Comment Blocks

ASCII

Define Variables in Global Input

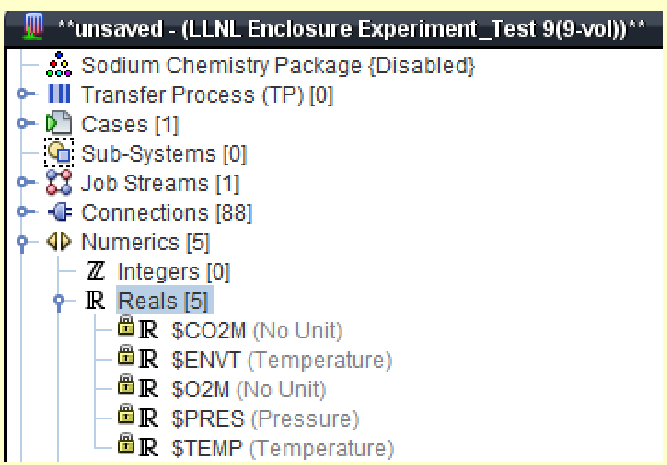
```
CommentBlock t9expfl

(((t9expfl
! t9 test
VariableValue {{{pres=100927.0}}} {{{temp=302.15}}} {{{o2m=0.208}}} {{{co2m=0.0005}}}
)))

(((t11expfl
! t11 test
VariableValue {{{pres=101250.0}}} {{{temp=292.15}}} {{{o2m=0.2081}} {{{co2m=0.0004}}}
)))
```

Use Variables in CVH input

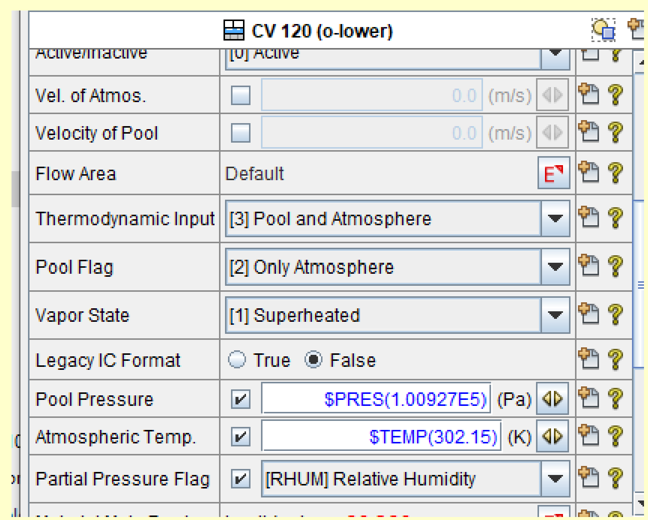
```
CV_ID      'i-lower'    100
CV_THR      NONEQUIL    FOG     ACTIVE
CV_PAS     SEPARATE    ONLYATM    SUPERHEATED
!        ptdit      pvol
CV_PTD       PVOL     {{{pres=}}}
!        atmid      tatm
CV_AAD       TATM     {{{temp=}}}
!        nmmat
CV_NCG        3     RHUM   0.5
!          n     namgas     mass
           1      'N2'     0.7915
           2      'O2'     {{{o2m=}}}
           3      'CO2'     {{{co2m=}}}
```

SNAP

Define Variables in Numerics Input



Use Variables in CVH input



External Data Files (EDF) Package

MELCOR External Data Files (EDF) Overview

A general means of communication (read or write) with external data files containing time history data

- Facilitate input of data (e.g., source definition and/or boundary condition) too large for TF
- Output data histories for use with another code or special purpose plot program

External Data File (EDF) utility provides unified treatment

- Defines file types, data format, and time control of data read and write
- Handles connection, opening, positioning, input or output, and closing of named file
- Any package can request interpolation to any time within current time step in any READ file

MELCOR External Data Files (EDF) File Types and Structures

Three types of external data files

- READ: data read in for use by MELCOR packages
- WRITE: user-selected data written to specified file
- PUSH: collection of data written at request of another MELCOR package

Each file contains values of time and one or more dependent variables, referred to as "data channels"

Each record in the file contains a value of time and the value(s) of the dependent variable(s) at that time

MELCOR External Data File (EDF) Input

Required input for each external data file

- User-defined name or ID (EDF_ID)
- Direction and mode of data transfer (READ, WRITE, PUSH)
- Name of file on computer system

Required input for WRITE and PUSH data files

- Control information for time interval between records (start time and time increment)

Required input for WRITE data files only

- List of dependent variables to be written, chosen from available control function arguments

MELCOR External Data File (EDF) Input (2)

Optional input for each external data file

- External data file format (default is unformatted)
- Format specification uses FORTRAN syntax
- Time offset between data and MELCOR calculation
  - Intended to handle data with different time reference
  - Useful for experimental data or in interfacing with another simulation code
  - tFile = tMELCOR + tOffset

MELCOR External Data Files
Example Input using EDF - WRITE

Input fragment to write to an external data file containing user-selected variables of interest for post-processing

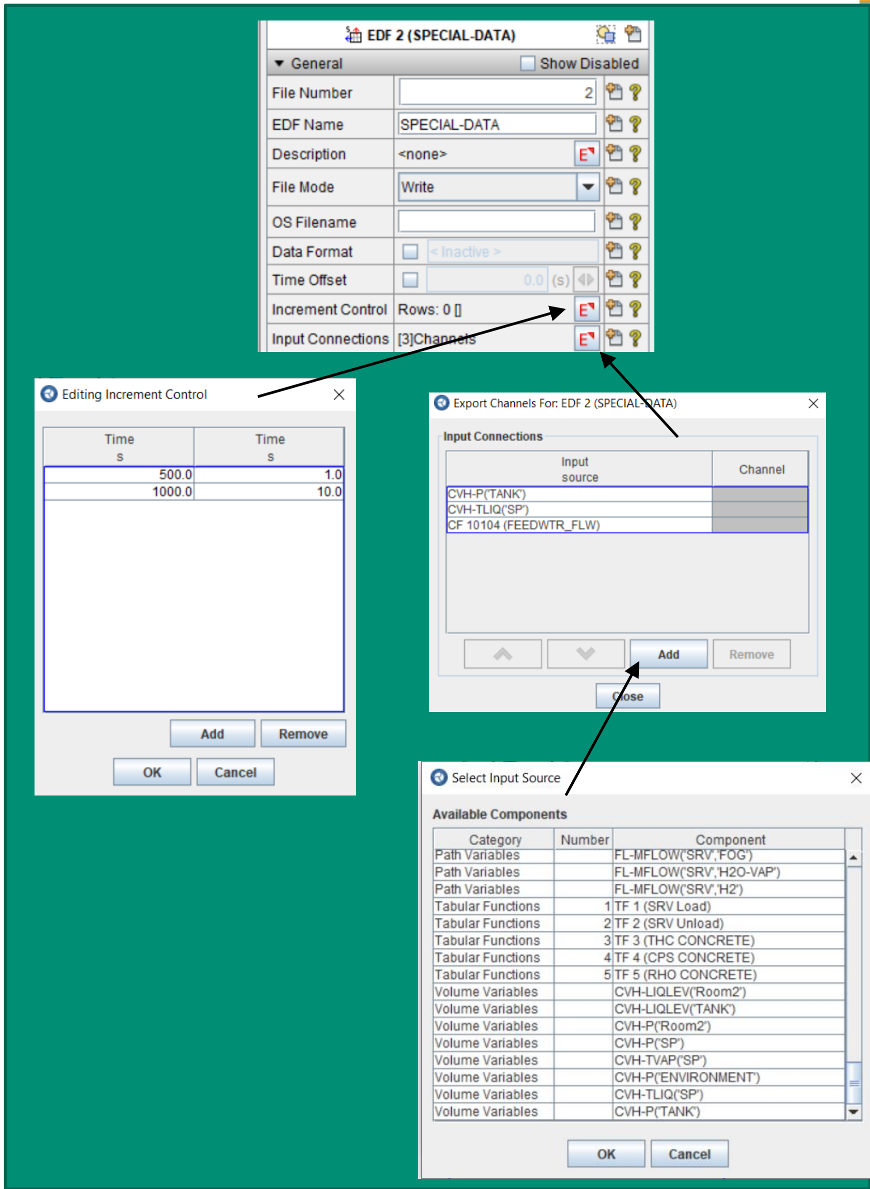
```
EDF_INPUT
!  User identification
!   |    Name
!   |    |              Direction and mode of transfer
!vvvvv  vvvv            vvvv
EDF_ID  SPECIAL-DATA  WRITE 'specdat.dat' ! Name of file on system
EDF_CHN    3     ! Number of data channels (3) to be written
           1  CVH-P(TANK)         ! pressure
           2  CVH-TLIQ(SP)           ! Pool temperature
           3  CF-VALU(FEEDWTR_FLOW) ! control function, feedwtr_flow
! EDF_DTW for write increment control
!     |            Starting at time TWEDF
!     |            |      Write a record every DTWEDF seconds
!vvvvvv            vvvvv  vvvvvv
EDF_DTW    2 !NT     TWEDF  DTWEDF
              1      500.0   1.0
              2      1000.0  10.0
! Note dependent variables (data channels) must be 'control function'
! arguments
```

MELCOR External Data Files
Example Input using EDF - WRITE

ASCII

```
EDF_INPUT
!  User identification
!    |    Name
!    |    |              Direction and mode of
transfer
!vvvvv   vvvv           vvvv
EDF_ID  SPECIAL-DATA  WRITE ‘specdat.dat’ !
Name of file on system
EDF_CHN    3     ! Number of data channels
(3) to be written
          1  CVH-P(TANK)          ! pressure
          2  CVH-TLIQ(SP)            ! Pool
temperature
          3  CF-VALU(FEEDWTR_FLOW) !
control function, feedwtr_flow
! EDF_DTW for write increment control
!      |             Starting at time TWEDF
!      |             |      Write a record
every DTWEDF seconds
!vvvvvv              vvvvv  vvvvvv
EDF_DTW    2 !NT      TWEDF  DTWEDF
           1      500.0    1.0
           2      1000.0   10.0
! Note dependent variables (data channels)
must be ‘control function’
! arguments
```

SNAP



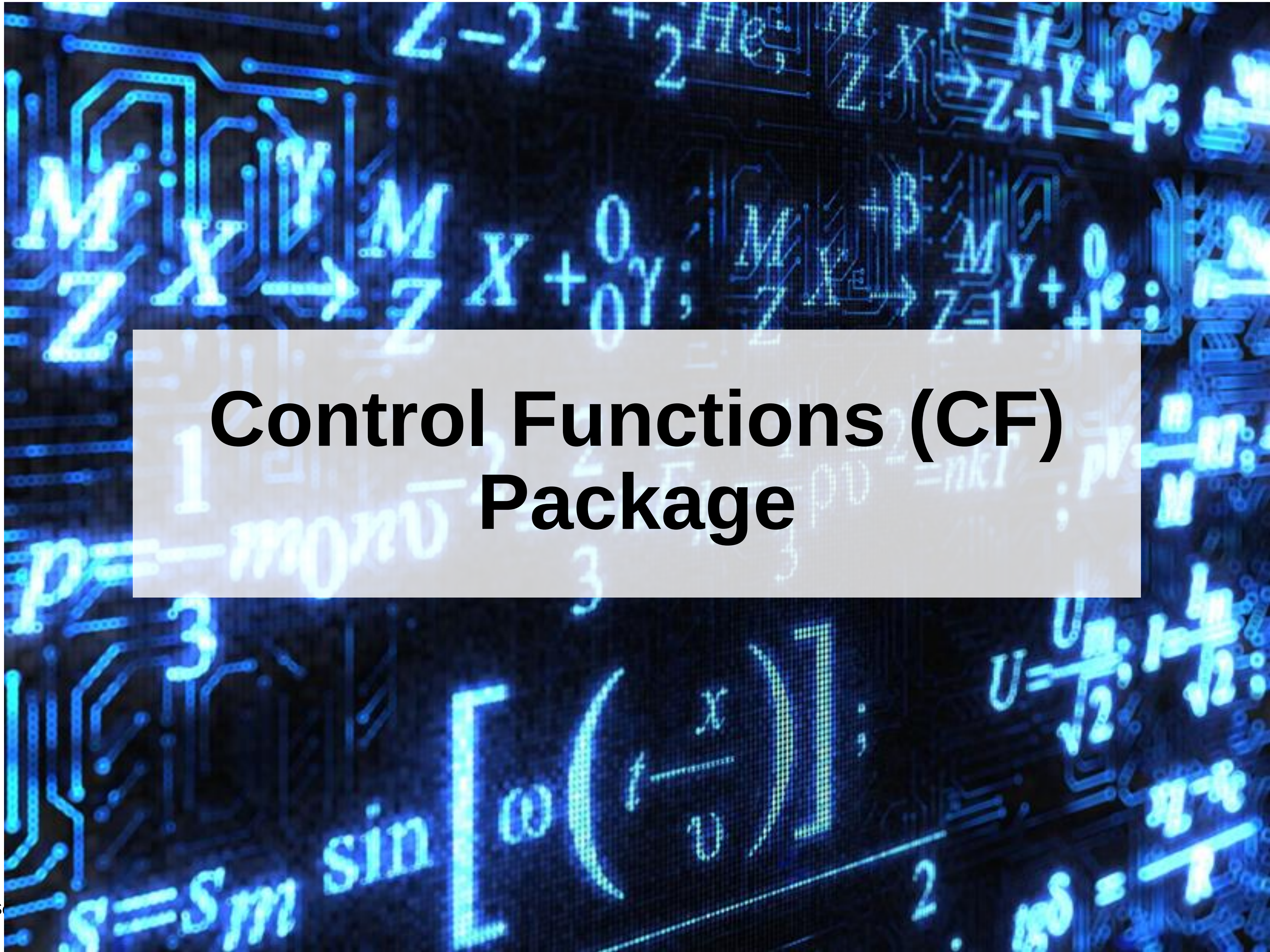
MELCOR External Data Files
Example Input using EDF - READ

Input fragment for steam source read from an unformatted file

```
CV_ID CV123
!  Integral steam mass and enthalpy from EDF 7 (British units)
CV_SOU 2 !N, SourceInfo
   1 MASS INTEGRAL EDF EDF7 1  H2O-VAP 0.4535924 ! pound to kg
   2 AE   INTEGRAL EDF EDF7 2  1055.06           ! BTU to J
...
EDF_INPUT
!  User identification
!    |    Name
!    |    |     Direction and mode of transfer
!vvvvv  vvvv  vvvv
EDF_ID  EDF7  READ '../data/steam.dat' ! Name of file on system
EDF_CHN   2     ! Number of data channels
EDF_TIM 7200.0 ! t=0 in MELCOR is 7200s on file
```

- **Each record in file '../data/steam.dat' contains values of ($t, \int^t \dot{M}dt', \int^t \dot{H}dt'$) in British units.**

Control Functions (CF) Package



MELCOR Control Functions (CF) Overview

"Control Functions" are simply user-defined functions of MELCOR-calculated variables

- May be LOGICAL- or REAL-valued
- All functions are evaluated at the start of every time step
- All control-function-based models are numerically explicit
- Recent improvement

Many uses, not just control

- Define door behavior, failure conditions, chemical reactions.
- Define internally-calculated sources and boundary conditions

Many variables in MELCOR database are available as arguments for control functions

- Any CF variable can be written to an external data file
- Any CF variable can be added to the plot file

MELCOR Control Functions Control Function Arguments

Many variables in MELCOR time-dependent database are available as function arguments

- Not all variables, due to coding required to access them
- Most are REAL-valued, but a few are LOGICAL
- Listed, by package, in the various User's Guides

Most packages use names of form xyz-name

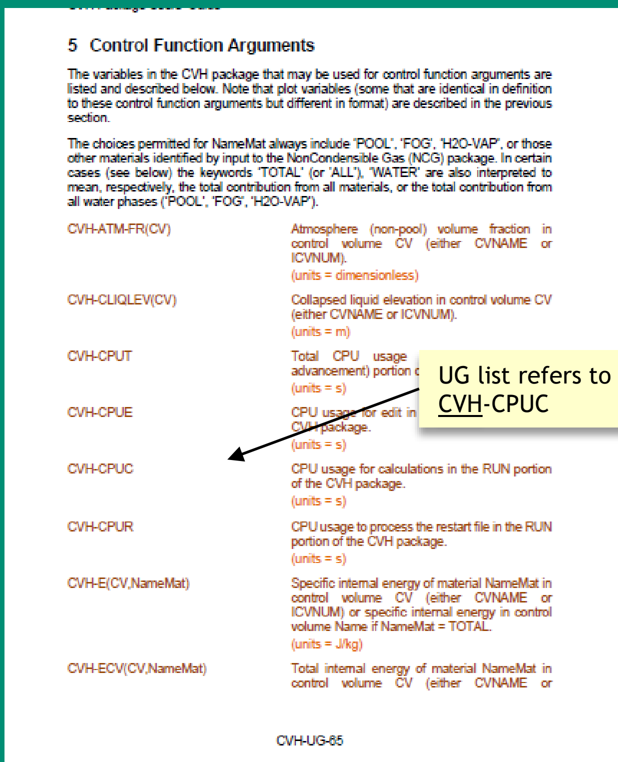
- "xyz" identifies the package and "name" the variable
- e.g.) CVH-TOT-M(O2) is total O2 mass in CVH package

Simple names for those defined by Executive Package

- EXEC-TIME is problem time
- EXEC-DT is (system) time step
- EXEC-CPU is (total) computer time

Where To Find CF Arguments

Listed & Described in package UG (i.e., CVH)



5 Control Function Arguments

The variables in the CVH package that may be used for control function arguments are listed and described below. Note that plot variables (some that are identical in definition to these control function arguments but different in format) are described in the previous section.

The choices permitted for NameMat always include 'POOL', 'FOG', 'H2O-VAP', or those other materials identified by input to the NonCondensible Gas (NCG) package. In certain cases (see below) the keywords 'TOTAL' (or 'ALL'), 'WATER' are also interpreted to mean, respectively, the total contribution from all materials, or the total contribution from all water phases ('POOL', 'FOG', 'H2O-VAP').

CVH-ATM-FR(CV) — Atmosphere (non-pool) volume fraction in control volume CV (either CVNAME or ICVNUM). (units = dimensionless)

CVH-CLIQLEV(CV) — Collapsed liquid elevation in control volume CV (either CVNAME or ICVNUM). (units = m)

CVH-CPUT — Total CPU usage ... advancement) portion ... (units = s)

CVH-CPUE — CPU usage for edit in ... CVH package. (units = s)

CVH-CPUC — CPU usage for calculations in the RUN portion of the CVH package. (units = s)

CVH-CPUR — CPU usage to process the restart file in the RUN portion of the CVH package. (units = s)

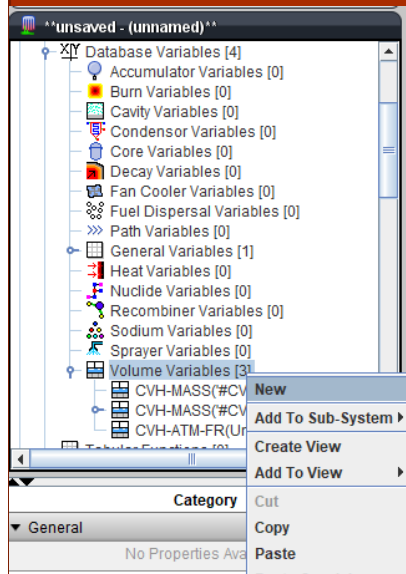
CVH-E(CV,NameMat) — Specific internal energy of material NameMat in control volume CV (either CVNAME or ICVNUM) or specific internal energy in control volume Name if NameMat = TOTAL. (units = J/kg)

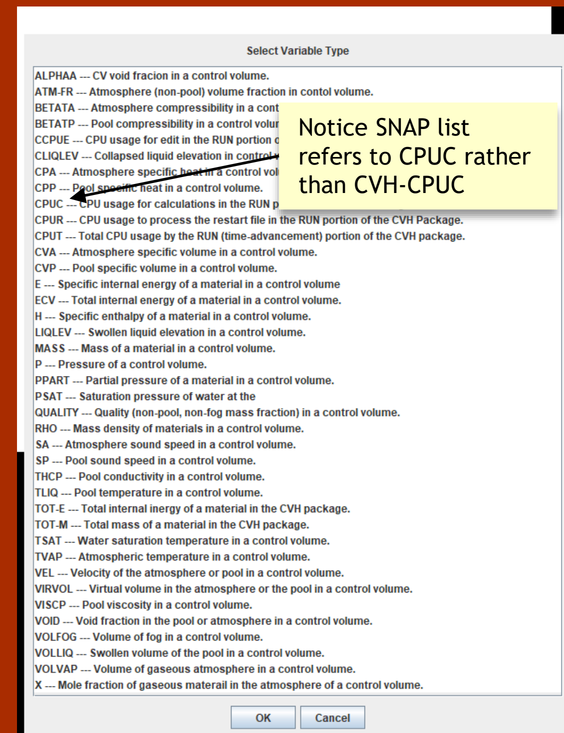
CVH-ECV(CV,NameMat) — Total internal energy of material NameMat in control volume CV (either CVNAME or

CVH-UG-65

UG list refers to CVH-CPUC

Drop-down list of SNAP supported CF arguments in Database Variables





Notice SNAP list refers to CPUC rather than CVH-CPUC

Notice SNAP refers to 'CVH variables' as 'Volume Variables'

Database Variables (CF arguments available to model)

Control Function arguments must be added to Database Variables before they can be used for input.

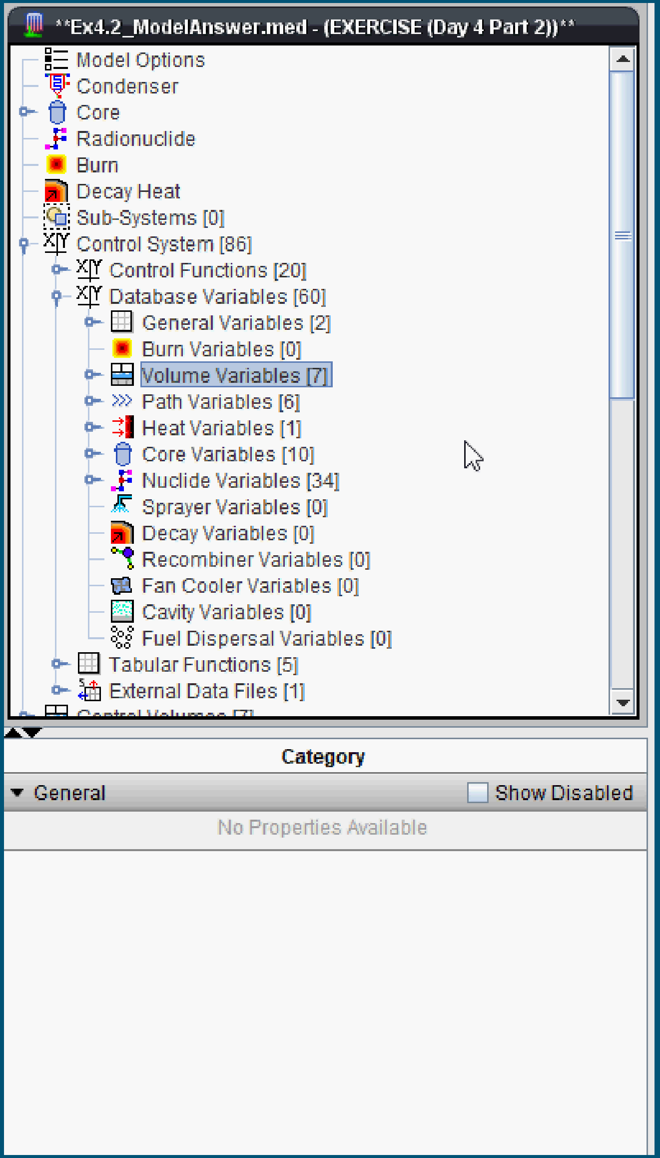
Used as input to control functions

Control Function arguments are organized by package

- General Variables (EXEC)
- Burn Variables (BUR)
- Path Variables (FL)
- Heat Variables (HS)
- Core Variables (COR)
- Nuclide Variable (RN)
- Sprayer Variables (SPR)
- Decay Variables (DCH)
- Recombiner Variables (PAR)
- Fan Cooler Variables (FCL)
- Cavity Variables (CAV)
- Fuel Dispersal Variables (FDI)

Adding a CF argument to the database

- Right Click Package category and select 'New'
- New variable appears in list
- Make selection to MELCOR CF arguments



Example: Add swollen liquid level for wetwell to database.

MELCOR Control Functions
Control Function Argument Arrays

Many control function arguments are essentially elements of arrays

- Index is user-defined name of volume, flowpath, etc.
- Index is added to name in a parenthesis
  - CVH-P(ROOM1) is pressure in 'ROOM1' volume
  - CVH-TVAP(ROOM1) is atmosphere temperature in 'ROOM1' volume
- Arrays may have more than one index
  - FL-MFLOW(vent,all) is total mass flow in flowpath 'vent'
  - EDF(out-10, 2) is data channel 2 in EDF 'out-10'
  - RN1-ADEP(HS1, LHS, CE, TOT) is total deposited mass of CE class on the left hand side (LHS) of heat structure 'HS1'

MELCOR Control Functions Direct Use of CF Arguments

Any CF argument can be written to an external data file (EDF package)

```
EDF_INPUT
EDF_ID ‘Misc Data’ WRITE ‘Misc.dat’ ! File name on system
EDF_CHN   3 !N New Name Value
             1 CVH-P(CV150)          ! Pressure in volume CV150
             2 FL-MFLOW(FL199,ALL)   ! Mass flow in path FL199
             3 CVH-TVAP(CV150)       ! Atmosphere temperature in
                                     ! Volume CV150
! EDF_DTW for write increment control
!     |           Starting at time TWEDF
!     |           |      Write a record every DTWEDF seconds
!VVVVVV           VVVVV  VVVVVV
EDF_DTW  1 !NT     TWEDF  DTWEDF
           1      1000.  10.       ! Output frequency
EDF_FMT  4E12.5 ! Format: time + 3 variables
```

MELCOR Control Functions Direct Use of CF Arguments (2)

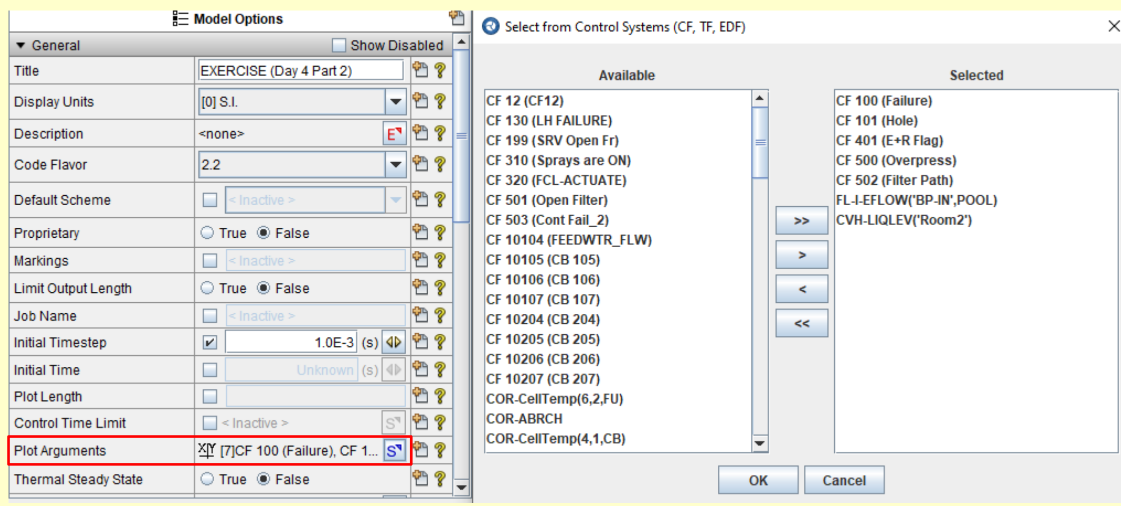
Any CF argument can be added to the plot file (EXEC_PLOT)

- Add any number in MELGEN input: written for entire run
- Add any number on MELCOR restart: included in the plot file for the duration of current execution

ASCII

```
EXEC_PLOT    7
  1 CF-VALU('Failure')
  2 CF-VALU('Hole')
  3 CF-VALU('E+R Flag')
  4 CF-VALU('Overpress')
  5 CF-VALU('Filter Path')
  6 FL-I-EFLOW('BP-IN',POOL)
  7 CVH-LIQLEV('Room2')
```

SNAP



Note that a CF argument must be added to Control System Database before it can be assigned to a plot variable

MELCOR Control Functions Composite Functions

Values of control functions are available for use as arguments of other control functions

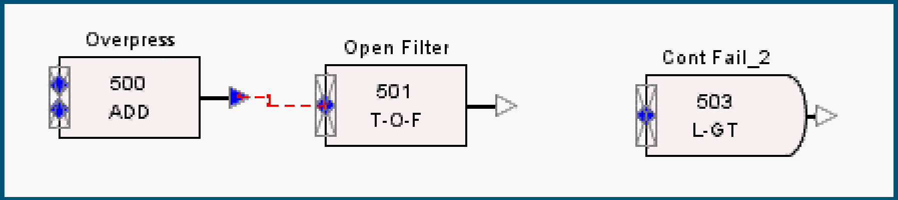
- Can construct composite functions such as $\sin\left(\sqrt{\sum M_I}\right)$

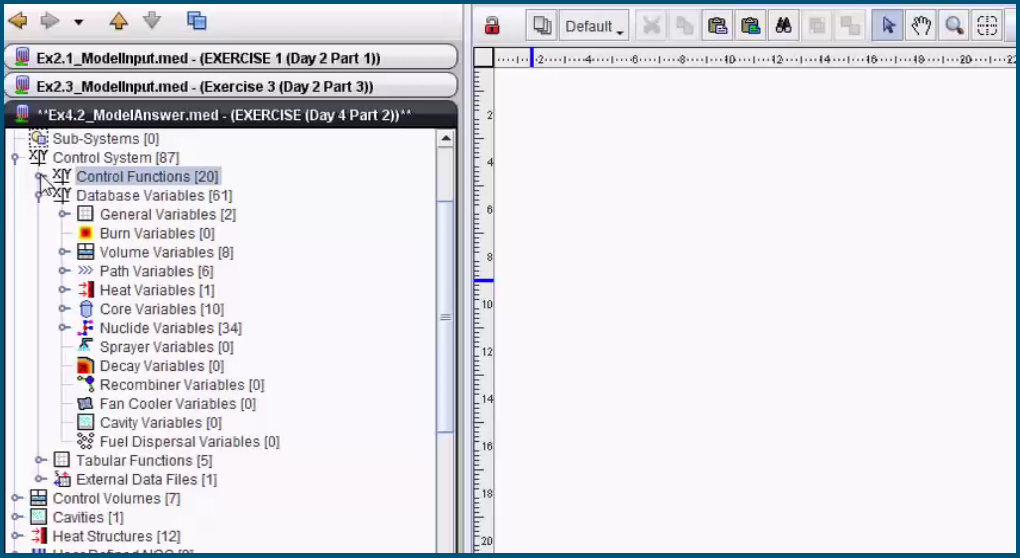
Functions are evaluated in the numerical order of the CF number (not on order read)

- A function should ordinarily use only previously-defined functions as arguments
- There are exceptions, where the value from the previous time step is desired
  - Evaluating out of order will use the previous time step value

Control Function connections

- Connecting output from one CF to input of another CF
  - Graphically
    - Drag both CF objects to the view and use connection tool
    - Cannot make connection from property window
- Connecting control function arguments to the input of a control function
  - Drag control function object and all Database variables to view
  - Make adjustments to multipliers later from properties window
  - Cannot make connection from property window





Example: Activate Sprays when containment pressure exceeds 1.2E5 Pa.

MELCOR Control Functions
CF Input: Required Input

Required input for each control function

- User-defined name
- Function type (Add, EXP, SIN, L-AND, L-OR, etc.)
  - Type determines whether value is REAL or LOGICAL
- Number of arguments
- List of arguments

Required input for REAL-valued control function

- Multiplicative scale factor

MELCOR Control Functions
CF Input: Optional Input

Optional Input for each control function

- Initial value (real, true or false)
  - Only needed if value will be needed early

Optional Input for REAL control function

- Additive constant for function (default = 0.0)
  - Evaluated as $CF_n = scale*f_n[X(t)] + add$
- Upper and lower bounds
  - Results bounded within limits
- Units (used for plotting purposes only)

CF_Units is the ASCII record for specifying units for a control function. Currently, the SNAP MELCOR plugin does not support this feature.

Optional Input for LOGICAL control function

- Message to be output when function switches state
  - Report user-defined 'events' in the output files
- Logical function classification as 'LATCH' or 'ONE-SHOT'
- If initially FALSE, 'ONE-SHOT' can be TRUE for one step only; if initially TRUE, 'LATCH' can only be .FALSE. once

MELCOR Control Functions Built-in Functional Forms

- **Most FORTRAN and simple math functions**
  - **Arithmetic, trigonometric, hyperbolic, and LOGICAL**
- **Tabular function (using table in TF package)**
- **IF-THEN-ELSE structures**
- **Numerical integrals and derivatives**
  - **Includes a proportional-integral-differential (PID) controller**
- **Hysteresis function**
  - **References TF package to defined loading/unloading curves**
- **A variety of “trips”**
  - **Trips are REAL-valued; value returned is time since trips**
  - **Simplifies logic involving delays**
  - **Usable as timer or ramp-generator**

MELCOR Control Functions Built-in Functional Forms (2)

- **Larson-Miller creep rupture Control Function (LM-CREEP)**
  - **Evaluates cumulative damage based on the Larson-Miller creep rupture failure model and gives time to rupture in seconds**
- **Pipe stress control function (PIPE-STR)**
  - **Evaluates maximum stress in a thick-walled cylindrical pipe under internal pressure**
- **User-Defined function (FORMULA)**
  - **Allows definition of a complicated function on a single record instead of series of records**
- **Lag function**
  - **Evaluated as a scaled change in the function value by scaling the change in the argument (Time Lag) as well providing a multiplication scale for the argument.**

Exercise 2.5a

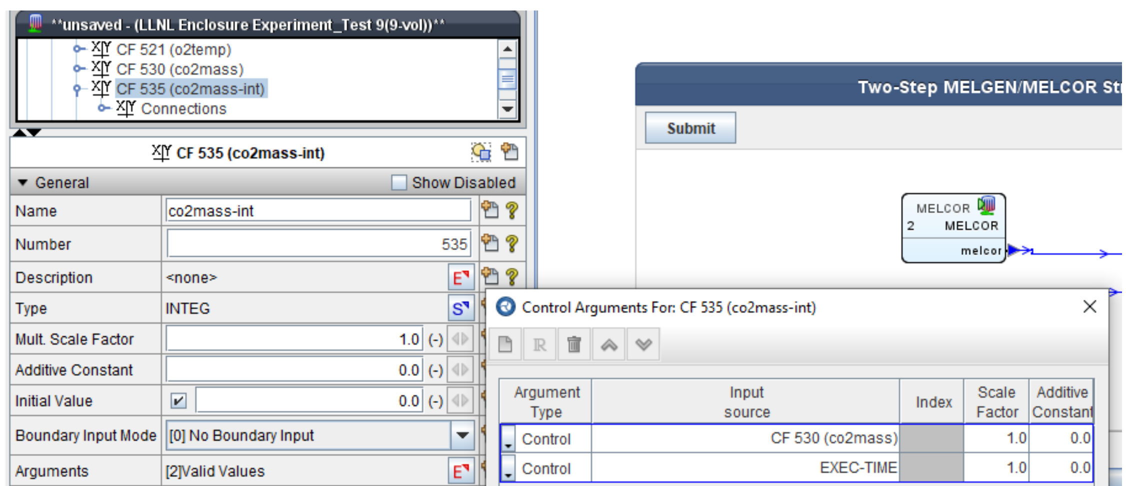
Create an Integration TYPE CF

- **Import 2.5a_start.inp into SNAP or work with the text file.**
- **Create a CF to integrate the rate of CO_2 generation (sourced into the problem) to calculate the cumulative mass of CO_2 generated.**
  - Name the CF 'co2mass-int'
  - Number the CF #535
  - Make it an INTEG type
  - Use the CF 'co2mass' as the integrand
  - Integrate over time.
  - Plot results or check values in output file

ASCII Solution

```
cf_id  'co2mass-int'    535  INTEG
CF_SAI        1.0          0.0      0.0
cf_arg  2 !n
        1  cf-valu('co2mass')   1.0   0.0
        2  exec-time     1.0
```

SNAP Solution



Order of Operations in Evaluating MELCOR Control Function

Demonstrate the order of operations MELCOR uses to evaluate the following control function

```
...
CF_INPUT
!  User identification
!  |   Name
!  |   |       Type of function (add argument)
!vvvv  vvvvvv  vvvvvv
CF_ID  'CF12'  ADD
!      Multiplier for function
!      |   Added constant
CF_SAI 0.0 70.E6
!       Bounds used
!       |      LowerBound
!       vvvv   vvv   UpperBound
CF_UBL  BOTH   2.0   7.0
CF_ARG     1 ! NARG CHARG      ARSCAL  ARADCN(optional)
                 1  EXEC-TIME  1.0     0.0
                 2  CF-CONST   1.0
```

Order of Operations in Evaluating MELCOR Control Function

1ST Step

- **Evaluate the individual arguments**
  - Arg(n) = Package_Arg_Value(n) * ARSCAL + ARADCN

```
...
CF_INPUT
!  User identification
!  |   Name
!  |   |       Type of function (add argument)
!vvvv  vvvvvv  vvvvvv
CF_ID  'CF12'  ADD
!      Multiplier for function CFSCAL
!      |     Added constant CFADCN
CF_SAI 1.0   0.0                    2.0 ! Initial value
!            LowerBound
!            vvv   UpperBound
CF_UBL Both 2.0    7.0
CF_ARG     2 ! NARG CHARG      ARSCAL   ARADCN(optional)
                1  EXEC-TIME   1.0      0.0
                2  CF-CONST    1.0
```


Sandia National Laboratories

Order of Operations in Evaluating MELCOR Control Function

2nd Step

- **Perform function on the scaled argument(s)**
  - **For this case: Func_Arg = Arg(1) + Arg(2)**

```
...
CF_INPUT
!  User identification
!  |   Name
!  |   |       Type of function (add arguments)
!vvvv  vvvvvv  vvvvvv
CF_ID  'CF12'  ADD     ! Func_Arg = Arg(1) + Arg(2)
!      Multiplier for function CFSCAL
!      |     Added constant CFADCN
CF_SAI 1.0   0.0                     2.0 ! Initial value
!           LowerBound
!           vvv   UpperBound
CF_UBL Both 2.0   7.0
CF_ARG    2 ! NARG CHARG      ARSCAL   ARADCN(optional)
                1  EXEC-TIME  1.0      0.0 ! Arg(1) = Problem_Time*1.0+0.0
                2  CF-CONST   1.0          ! Arg(2) = 1.0
```


Sandia National Laboratories

Order of Operations in Evaluating MELCOR Control Function

3rd Step

- **Apply function scaling and additive values**
  - **InterFunc = Func_Arg * CFSCAL + CFADCN**

```
...
CF_INPUT
!  User identification
!  |   Name
!  |   |       Type of function (add argument)
!vvvv  vvvvvv  vvvvvv
CF_ID  'CF12'  ADD
!      Multiplier for function CFSCAL
!      |     Added constant CFADCN   InterFunc=Func_Arg*CFSCAL+CFADCN
CF_SAI 1.0   0.0                     2.0 ! Initial value
!           LowerBound
!           vvv   UpperBound
CF_UBL Both 2.0   7.0
CF_ARG    2 ! NARG CHARG      ARSCAL  ARADCN(optional)
                1  EXEC-TIME  1.0     0.0
                2  CF-CONST   1.0
```


Sandia National Laboratories

Order of Operations in Evaluating MELCOR Control Function

4th Step

- **Impose upper and lower boundaries**
  - **Func = max(LowerBound,min(InterFunc,UpperBound))**

```
...
CF_INPUT
!  User identification
!  |   Name
!  |   |       Type of function (add argument)
!vvvv  vvvvvv  vvvvvv
CF_ID  'CF12'  ADD
!      Multiplier for function CFSCAL
!      |     Added constant CFADCN
CF_SAI 1.0   0.0                    2.0 ! Initial value
!           LowerBound
!           vvv   UpperBound
CF_UBL Both 2.0   7.0
CF_ARG    2 ! NARG CHARG      ARSCAL   ARADCN(optional)
                1  EXEC-TIME  1.0      0.0
                2  CF-CONST   1.0
```


Sandia National Laboratories

MELCOR Control Functions Simple Examples

Simple examples first to demonstrate CF format and usage

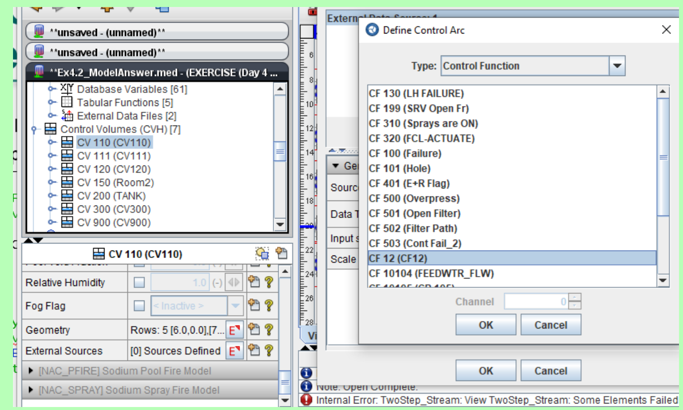
- More examples and complete list of built in function types given in CF package user's guide
- There are often several ways to build a function

MELCOR Control Functions Example Input Using CF

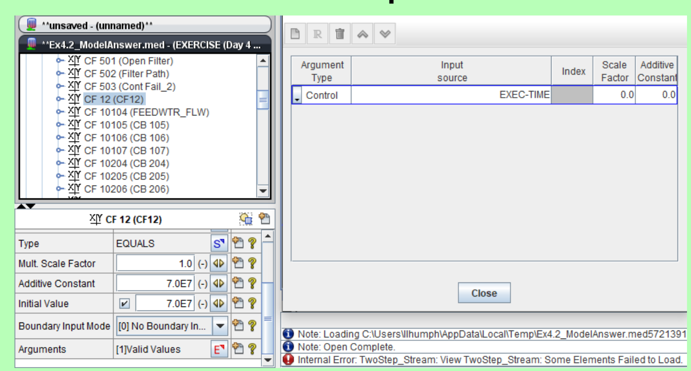
- **Input block for energy source in core**
  - **(same as TF example input shown earlier, but uses CF)**

```
CV_ID CV110
!  Results equivalent to TF example earlier,
but use CF
CV_SOU 1 ! N  SourceInfo
           1  PE  RATE CF CF12
...
CF_INPUT
CF_ID  'CF12'   001       EQUALS
CF_SAI 0.0 70.E6
CF_ARG    1 ! NARG CHARG      ARSCAL
          1  EXEC-TIME  0.0
! Must specify one argument
! Value of ‘CF12’ = 0.0 x [(EXEC-TIME x 0.0) +
0.0] + 70.E6 = 70.E6
```

CV Input



CF Input



MELCOR Control Functions Example Input Using CF (2)

- **Alternate form 1 for constant control function**

```
CF_ID 'Pi' 10 EQUALS
!       Multiplier for function
!       vvvvv
CF_SAI 3.1415 ! Add 0.0 (default)
CF_ARG 1 ! NARG CHARG     ARSCAL   ARADCN
             1   EXEC-TIME    0.0      1.0
```

Value returned is
3.1415 x [(EXEC-TIME x 0.0) + 1.0] + 0.0 = 3.1415

- **Alternate form 2 for constant control function**

```
CF_ID 'Pi' 10 EQUALS
CF_SAI 1.0  ! Mult 1; Add 0.0 (default)
CF_ARG 1 ! NARG CHARG     ARSCAL   ARADCN
             1   EXEC-TIME   0.0      3.1415
```

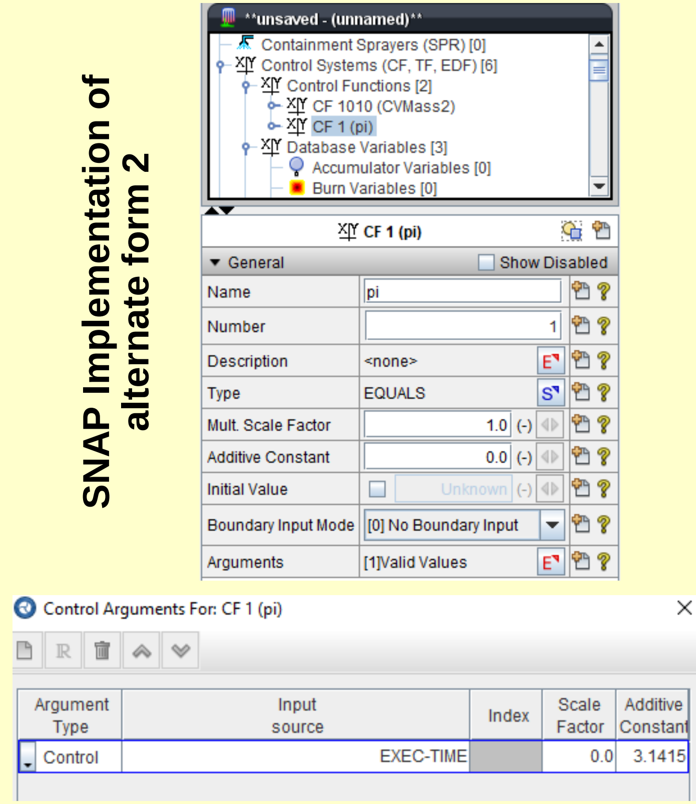
Value returned is
1.0 x [(EXEC-TIME x 0.0)+3.14156] + 0.0 = 3.14156

- **Best Practice (not implemented in SNAP)**

```
CF_ID 'Pi' 10 EQUALS
CF_SAI 1.0  ! Mult 1; Add 0.0 (default)
CF_ARG 1 ! NARG CHARG     ARSCAL   ARADCN
             1   CF-CONST   3.1415
```

Value returned is 1.0 x [3.14156] + 0.0 = 3.14156

SNAP Implementation of alternate form 2



Argument EXEC-TIME may need to be added to model database.

MELCOR Control Functions Example Input Using CF (3)

- **Example CF Input: Confinement failure with message**

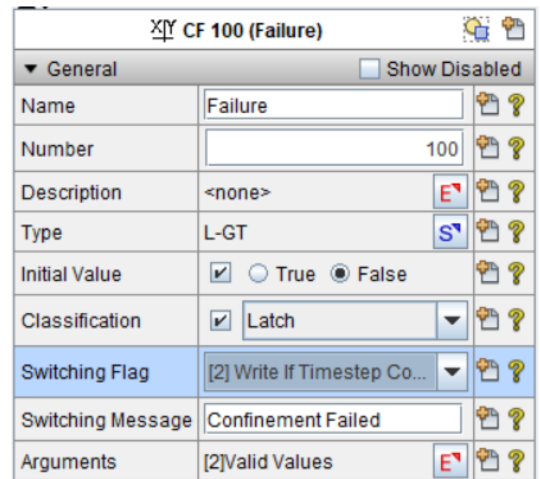
```
! LOGICAL function, .true. if arg1 >
arg2
CF_ID 'Failure' 100 L-GT
CF_LIV FALSE ! Initial value is .false.
CF_CLS LATCH ! Once .true., stays .true.
```

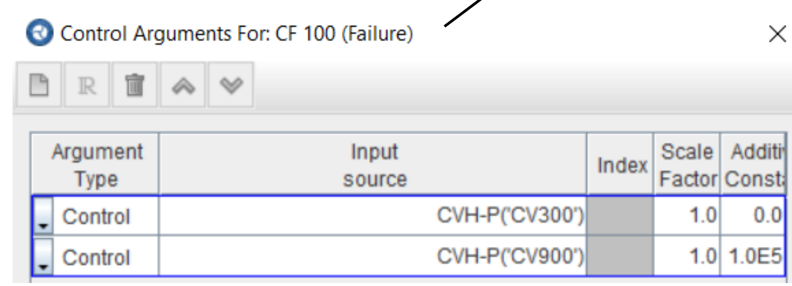
Writes to all files at completion of time step

Message to be written

```
CF_MSG FULL-OUTPUT ‘Confinement Failed’
CF_ARG 2  !  Argument       Scale  Add
          1  CVH-P(‘CV300’) 1.0    0.0
!Pressure in volume CV300
          2  CVH-P(‘CV900’) 1.0    1.E5
!Pressure in volume CV900 + 1.E5
```

CF becomes true if CV300 pressure exceeds CV900 by 1 bar





Sandia National Laboratories

MELCOR Control Functions Example Input Using CF (3)

- **Example CF Input: Confinement failure with message**

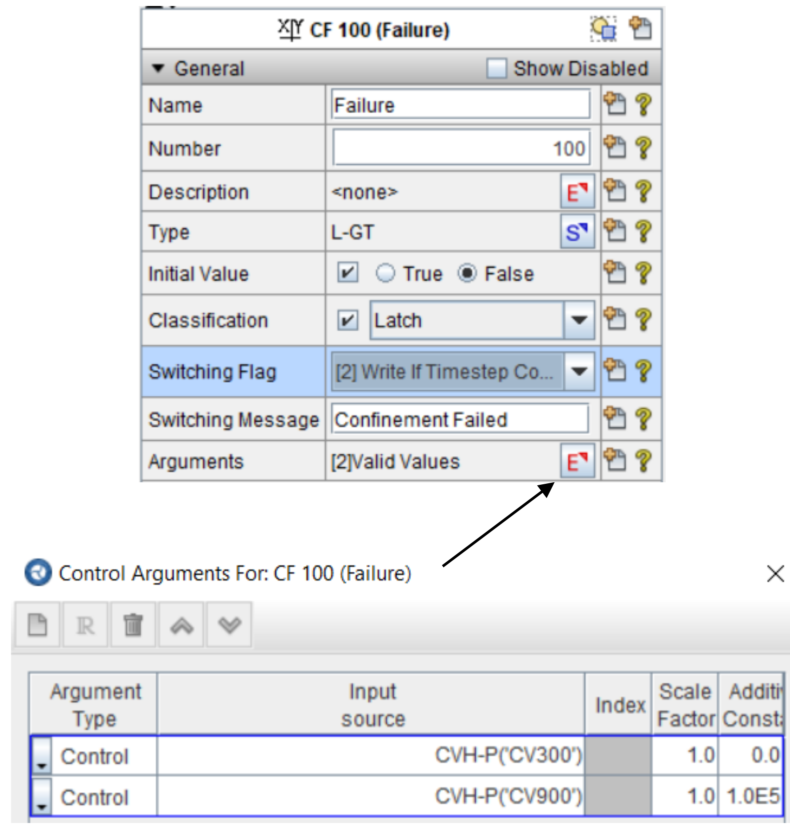
```
! LOGICAL function, .true. if arg1 >
arg2
CF_ID 'Failure' 100 L-GT
CF_LIV FALSE ! Initial value is .false.
CF_CLS LATCH ! Once .true., stays .true.
```

Writes to all files at completion of time step

Message to be written

```
CF_MSG FULL-OUTPUT 'Confinement Failed'
CF_ARG 2  !  Argument        Scale   Add
          1  CVH-P('CV300') 1.0    0.0
!Pressure in volume CV300
          2  CVH-P('CV900') 1.0    1.E5
!Pressure in volume CV900 + 1.E5
```

CF becomes true if CV300 pressure exceeds CV900 by 1 bar



MELCOR Control Functions Example Input Using CF (4)

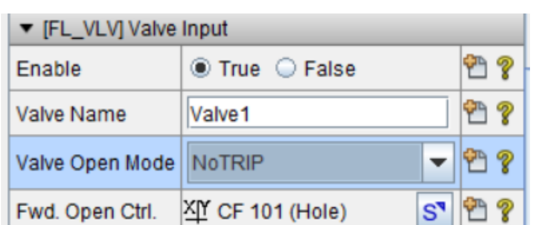
Example CF Input: Opening a valve (or door) in a flowpath

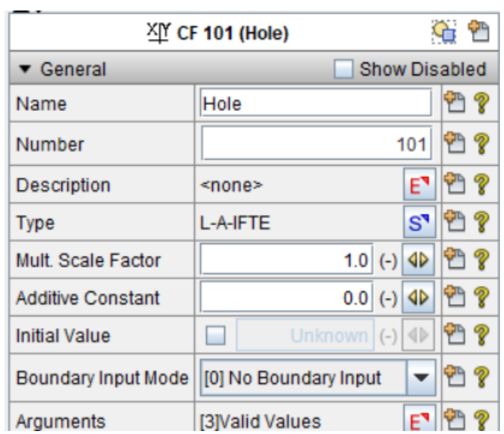
```
FL_VLV   1 !   NV VLVNAME     FLNAME    KEYTRIP NVFONF
           1   ‘Valve1’     ‘FL399’    NoTRIP  ‘Hole’
…
CF_INPUT
! REAL function, equivalent to IF-THEN-ELSE
                          vvvvvvvv
CF_ID   'Hole‘ 101 L-A-IFTE
CF_SAI   1.0
!              Argument         Scale    Add
CF_ARG   3 ! NARG CHARG         ARSCAL   ARADCN
             1 CF-VALU(‘Failure’) 0.0    0.0 ! LOGICAL, true after failure
             2 EXEC-TIME          0.0    1.0 ! Hole = 1.0 if arg 1 true
             3 EXEC-TIME          0.0    0.0 ! Hole = 0.0 if arg 1 false
```

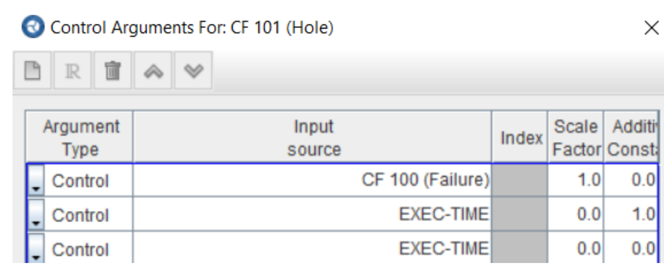
CF ‘Hole’ gives open fraction

Options: NoTrip, Trip, NoTripCV

In the SNAP implementation the valve is a property (optional) for a flowpath and is not a separate table input.







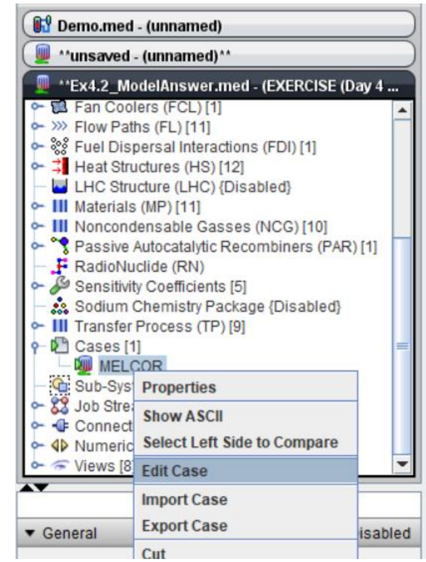
MELCOR Control Functions Example Input Using CF (5)

- **Generate restart and plot at time of failure**

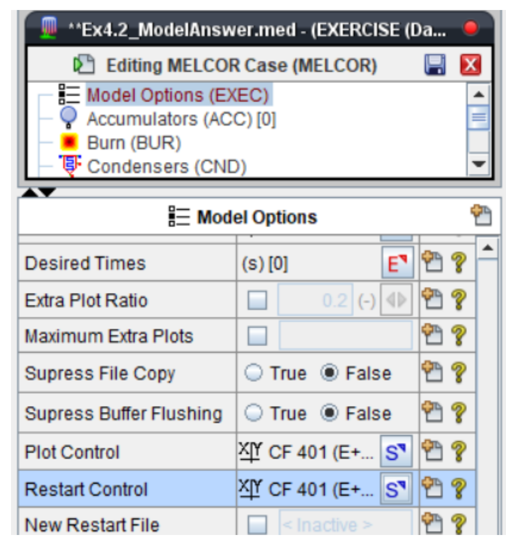
ASCII

```
EXEC_INPUT                     (MELCOR input)
EXEC_RESTARTCF ‘E+R Flag’      restart or
EXEC_PLOTCF    ‘E+R Flag’      plot dump if
...                            CF ‘E+R Flag’
                               is .true.

LOGICAL function ‘E+R Flag’, set equal to
argument (L-EQUALS) determines edit.
(Start of step on which CF becomes true)

CF_INPUT
CF_ID ‘E+R Flag’  105  L-EQUALS
CF_LIV FALSE     ! Initial value is
.false.
CF_CLS ONE-SHOT ! .true. only once
CF_ARG 1
       1 CF-VALU(‘Failure’)   0.    0.
```

(1) Edit MELCOR Input



(2) Edit EXEC input under Model Options



MELCOR Control Functions Example Input Using CF (6)

- **Calculate maximum pressure in volume 200**

```
! REAL function, 2 or more arguments
!                           vvv
CF_ID   ‘Peak P.200’  110  MAX
CF_SAI  1.0  0.0  0.0 ! Initialize to zero
!                   Argument          Scale     Add
CF_ARG 2 ! NARG       CHARG          ARSCAL    ARADCN
            1       CVH-P(‘CV200’)     1.0       0.0 ! *CURRENT*
                                        ! pressure in volume CV200
            2 CF-VALU(‘Peak P.200’)  1.0       0.0 ! *PREVIOUS*
                                        ! value of maximum
```

This is an example of a CF that references itself. In this case, it uses the value from the previous timestep.

Sandia National Laboratories

MELCOR Control Functions Example Input Using CF (6)

Calculate maximum pressure in volume 200

```
! REAL function, 2 or more arguments
!                               vvv
CF_ID    ‘Peak P.200’  110  MAX
CF_SAI  1.0  0.0  0.0 ! Initialize to zero
!                  Argument         Scale     Add
CF_ARG 2 ! NARG      CHARG           ARSCAL   ARADCN
            1      CVH-P(‘CV200’)    1.0       0.0 ! *CURRENT*
                                              ! pressure in volume CV200
            2 CF-VALU(‘Peak P.200’)  1.0       0.0 ! *PREVIOUS*
                                              ! value of maximum
```

This is an example of a CF that references itself. In this case, it uses the value from the previous timestep.

MELCOR Control Functions Example Input Using CF (7)

Built in function 'PIPE-STR' expressed with 'FORMULA' in a single control function

```
! Maximum stress in a thick-walled pipe under internal pressure
! given as, PIPE-STR(t)=[(Ro^2+Ri^2)*Pi-2Ro^2Po]/(Ro^2-Ri^2)
CF_ID     'Stress'   120   FORMULA
CF_SAI   1.0   0.0
CF_FORMULA 5  ((Ro^two+Ri^two)*Pi-two*Ro^two*Po)/(Ro^two-Ri^two)
            1   Pi   CVH-P(CV500) ! Inner pressure
            2   Po   CVH-P(CV8)   ! Outer pressure
            3   Ri   0.37         ! Inner radius (constant value)
            4   Ro   0.45         ! Outer radius (constant value)
            5   two  2.0          ! (constant value)
```

Calculate unburned gasoline remaining using 'FORMULA'

```
! Liquid fuel remained calculation
CF_ID     'RemainFuel'   1001   FORMULA
CF_SAI   1.0   0.0    3.2933
CF_FORMULA 3 fuel-brate*dt
            1   fuel   cf-valu(RemainFuel)   ! Old value of RemainFuel
            2   brate  CF-VALU(gasburnrate)  ! CF for burn rate
            3   dt     exec-dt
```

Warning: There are two restrictions: (1) a logical FORMULA CF that is equal to its single logical argument is not permitted, (2) the single-character 'E' or 'e' is not permitted as a SHORTNAME

Alternate Ways for Calculating Pipe Stress

Previously Shown 2 Ways

$$\sigma_{max}(t) = \frac{\pi\left(R_o^2 + R_i^2\right) - 2R_o^2 \cdot P_0}{R_o^2 - R_i^2}$$

- **Using PIPE-STR type CF**

```
CF_ID Stress PIPE-STR
CF_SAI 1.0 0.0
CF_MSC 0.37 0.45
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
    1 CVH-P(CV500) 1. 0.   ! Inner pressure (hot leg)
    2 CVH-P(CV8) 1. 0.     ! Outer pressure (containment)
```

- **Using FORMULA type CF**

```
CF_ID   'Stress'  120  FORMULA
CF_SAI  1.0  0.0
CF_FORMULA 5 ((Ro^two+Ri^two)*Pi-two*Ro^two*Po)/(Ro^two-Ri^two)
           1  Pi   CVH-P(CV500) ! Inner pressure
           2  Po   CVH-P(CV8)   ! Outer pressure
           3  Ri   0.37         ! Inner radius (constant value)
           4  Ro   0.45         ! Outer radius (constant value)
           5  two  2.0          ! (constant value)
```

Alternate Ways for Calculating Pipe Stress

$$\sigma_{max}(t) = \frac{\pi\left(R_o^2 + R_i^2\right) - 2R_o^2 \cdot P_0}{R_o^2 - R_i^2}$$

Using MELCOR Classic Control Functions (MELCOR 1.8.5)

```
CF_ID STRESS DIVIDE 16
CF_SAI 1.0 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
     1 CF-VALU(NUMERATOR) 1. 0.
     2 CF-VALU(DENOMINATOR) 1. 0.

CF_ID Numerator ADD 15
CF_SAI 1.0 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
     1 CF-VALU(TERM1) 1.0 0.
     2 CF-VALU(TERM2) -1. 0.

CF_ID TERM1 ADD 14
CF_SAI 3.1415 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
     1 CF-VALU(RO2) 1. 0.
     2 CF-VALU(RI2) 1. 0.

CF_ID TERM2 MULTIPLY 13
CF_SAI 1.0 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
     1 CF-VALU(RO2) 2. 0.
     2 CVH-P(CV8) 1. 0.
```

```
CF_ID DENOMINATOR ADD 12
CF_SAI 1.0 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
     1 CF-VALU(RO2) 1. 0.
     2 CF-VALU(RI2) -1. 0.

CF_ID RO2 POWER-I 11
CF_MSC 2.0
CF_SAI 1.0 0.0
CF_ARG 1 ! NARG CHARG ARSCAL ARADCN
     1 CF-CONST   0.45
     2 CF-VALU(RO1) -1. 0.

CF_ID RI2 POWER-I 10
CF_MSC 2.0
CF_SAI 1.0 0.0
CF_ARG 1 ! NARG CHARG ARSCAL ARADCN
     1 CF-CONST   0.37
     2 CF-VALU(RO1) 1. 0.
```

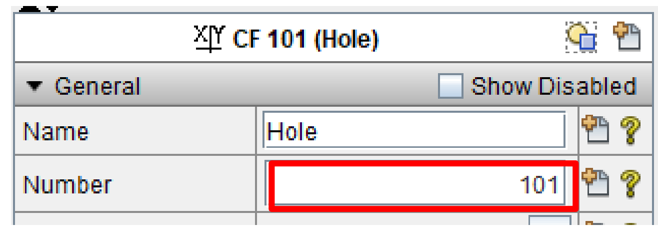
This method not recommended! Harder to read and more prone to mistakes!

Numbering of CF Determines Order of Evaluation

- **User assigns a number to a Control Function**

User assigned number

```
CF_ID  'Hole' 101 L-A-IFTE
```



- **CFs are evaluated in order of increasing number (be aware of various states of CFs)**

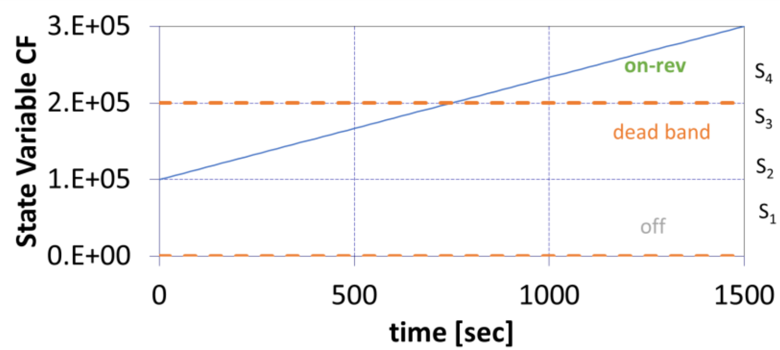
Example: T-O-R

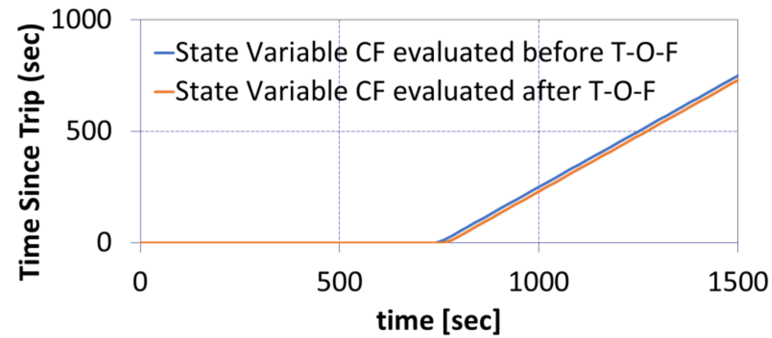
```
CF_ID     'Hole'    T-O-F
CF_SAI     0.5      0.0     0.0
CF_MSC    -1.0      2.E5
CF_ARG  1  ! Pressure calculated by CF
        1  CFVALU('pressure')  1.0   0.0
```

Value of trip is different whether state variable CF ('pressure') is evaluated before or after CFVALU('Hole').

Difference is time-step dependent.

Using CVH-P(CV300) as we did in our previous example does not have this dependency





MELCOR Control Functions Input Changes on Restart

- **Change any CF and TF parameters from the restart**
  - Allow addition of new CFs and TFs
  - Easy to run variations of a failure criterion
  - Run multiple scenarios that branch late in a sequence
    - Define input to include several failure paths
    - Run alternate sequences by restarting from a point before failure, changing break sizes, leak paths, or bounds/limits to allow a different path
    - No need to re-run a long pre-failure calculation
- **Continue calculation from last restart dump**
  - Need to set 'MEL_RESTARTFILE' record in environmental data appropriately
    - e.g., MEL_RESTARTFILE 'RUN1.RST' NCYCLE -1

Sandia National Laboratories

MELCOR Control Functions
Input Changes During a MELCOR Run(2)

- **Change actual value of control function thru READ (for REAL-valued) and L-READ (for LOGICAL-valued) option during a MELCOR run**
  - **Requires a new file containing name of CF and new value**
    - **New value type must match type of CF (REAL or LOGICAL)**
    - **New file name specified on “EXEC_CFEXFILE” record**
  - **Can be used to simply turn-on or –off a valve without stopping and restarting a calculation**
  - **Both L-READ and READ control functions could be used with SNAP on-the-fly simulations.**

Sandia National Laboratories

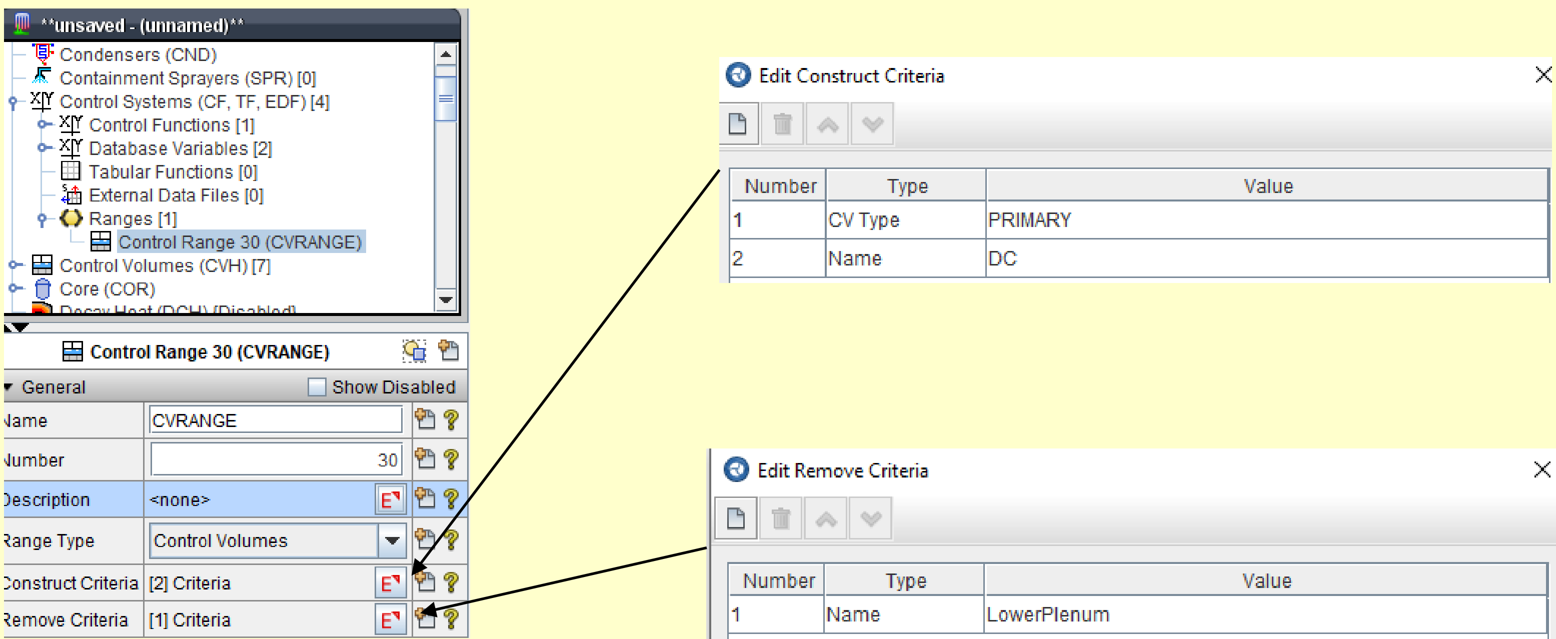
Control Function Ranges

The range is an object that is defined once in the database and then can be referenced by other control function arguments. The range specifies an ordered list of objects such a control volumes, COR cells, materials, or components

Define a Range (ASCII):

```
           name     type      ndim  Number
CF_RANGE   CVRANGE  CVOLUMES  2     30
CONSTRUCT  2
 1 CVTYPE='PRIMARY'
 2 DC
REMOVE  1
 1  LowerPlenum
```

Define a Range (SNAP):



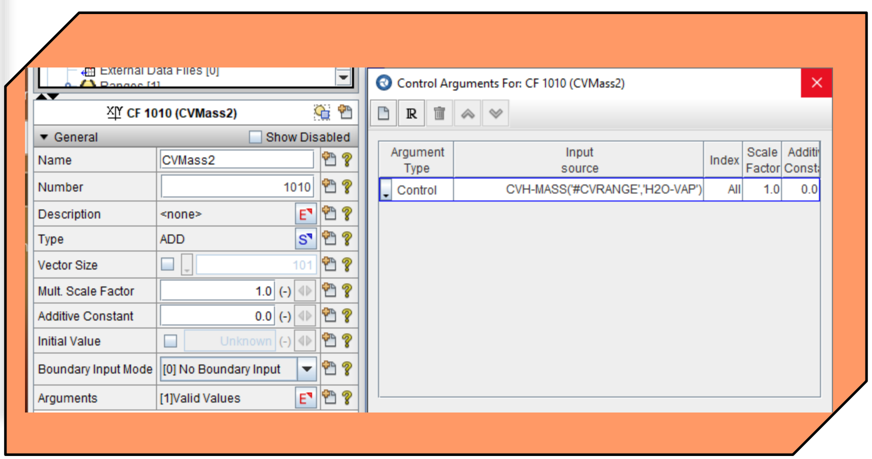
Control Function Ranges

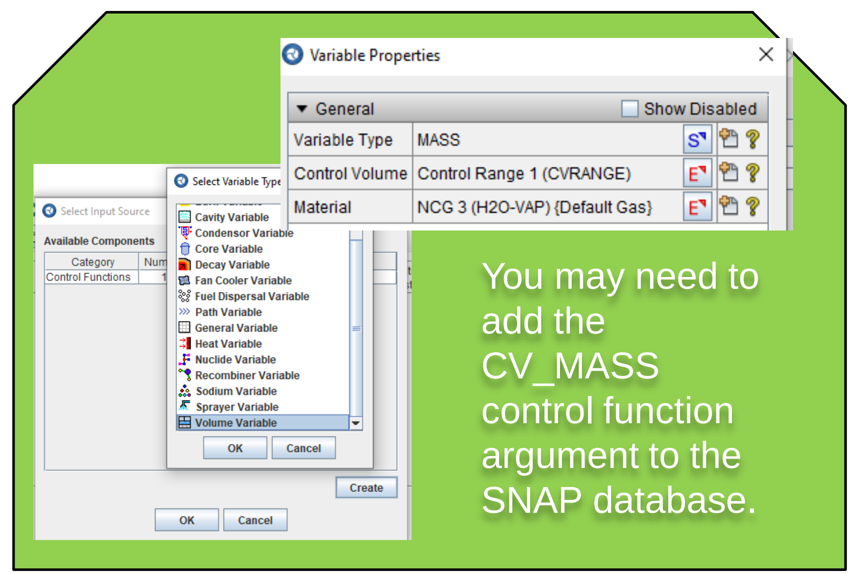
- A range can be referenced by control functions and control function arguments. The hashtag (#) that precedes range specified for the volume in the CF argument indicates a range of control volumes rather than a single volume.

Reference a Range (ASCII):

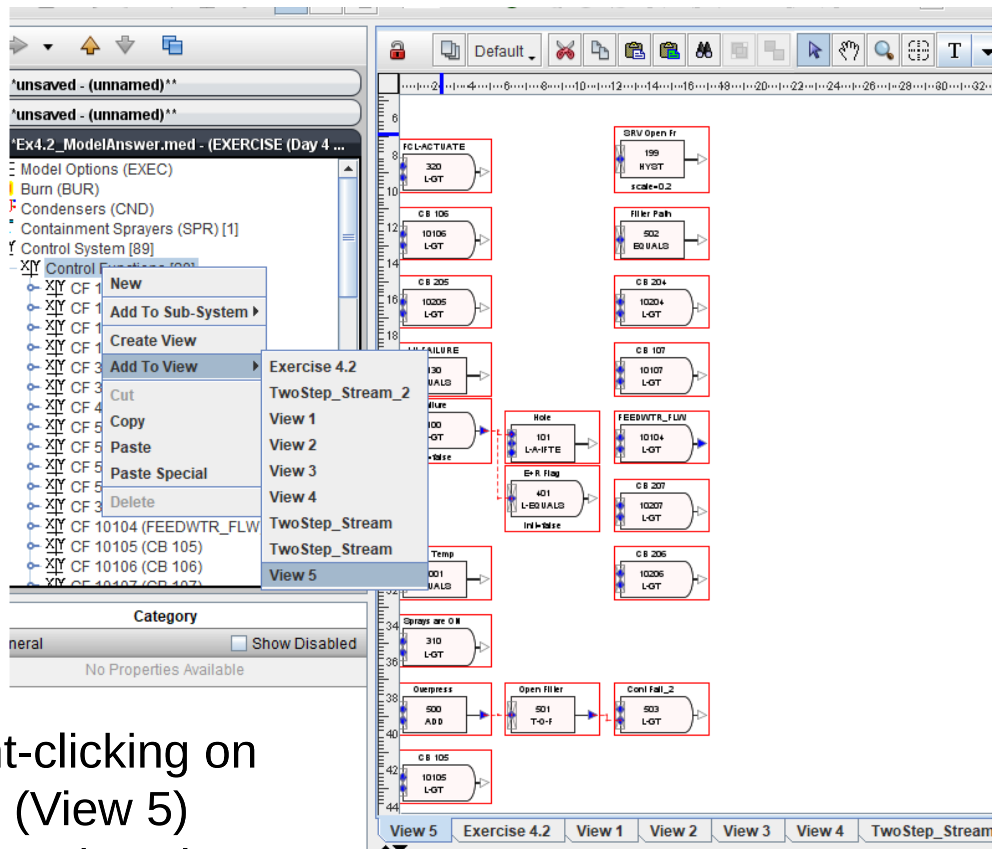
```
CF_ID       'CVMass2'  1010  ADD
CF_SAI 1.0 0.00
CFVALR (INITIAL VALUE)
CF_ARG 1
  1   CVH-MASS(#CVRANGE,'H2O-VAP')
1.0    0.0
```

Reference a Range (SNAP):





Viewing a Control Function Network in SNAP



- Create new view by right-clicking on Views in navigator pane (View 5)
- Right click on Control Functions in navigator pane
- Add to view previously created (View 5)

Sandia National Laboratories

Exercise 2.5c
Add a Formula TYPE CF

Methane Gas Reaction: $\mathbf{CH_4 + 2\,O_2 \rightarrow CO_2 + 2H_2O}$

1. **Use CF_FORMULA to model CO2 mass generation rate (kg/s)**

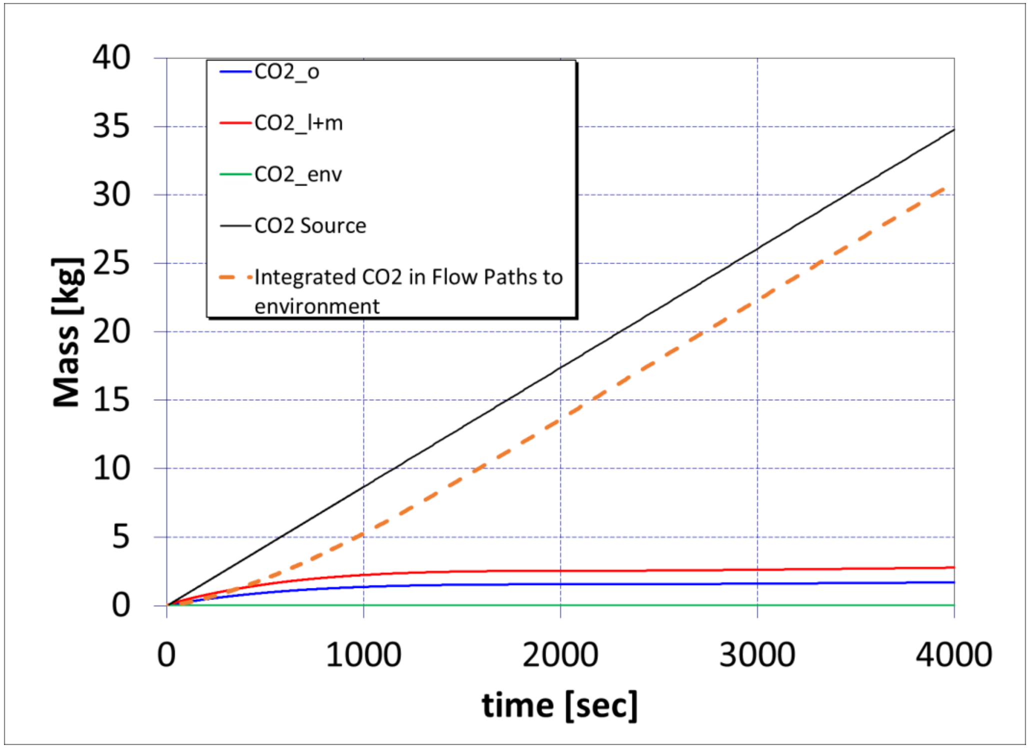
Control function 'o2mdotc' gives the rate at which oxygen is consumed in the fire

- If we know the O_2 mass consumption rate [kg/sec], then the CO_2 mass generate rate [kg/sec], 'co2mdotc' is related since 2 moles of O_2 yields 1 mole of CO_2
  - Thus, the formula would be 'o2mdotc'*0.5*mwco2/mwo2
- Molecular weights of CO_2 and O_2 are provided in the input as CFs
  - MW_{co2}=cf-valu('mw_co2'),
  - MW_{o2}=cf-valu('mw_o2')

2. Create a CF Network view from SNAP
   1. Create a new view
   2. Right-click on the CF navigator and add to this new view.

Exercise 2.6
Adding a Range and Vector CF

- Add a CVOLUME range, 'o-vol' to include all of the outer CVs (o-upper, o-middle, o-lower)
- Create an 'ADD' type CF that sums all CO_2 mass for that range of CVs. Subtract the initial mass of CO2 in the range of volumes to show changes in mass.
  - May be easiest to run MELGEN to get the initial mass in the range
- Run MELGEN and plot the mass over time and compare with the integral CO_2 source
- Create a Range 'i+m vol' to include all i- and m- CVs and a Range 'environment' to include all Environment CVs.
  - Subtract initial mass
- Create 'ADD' type CFs that sums all CO_2 mass for those range.
- Perform a mass balance on CO_2 mass
  - Show CO_2 mass in Range 'o-vol', Range 'i+m vol', Range 'Environment' and the integrated CO2 mass, 'co2mass-int' CF#535.
  - What's missing



Sandia National Laboratories

End of Data and Control