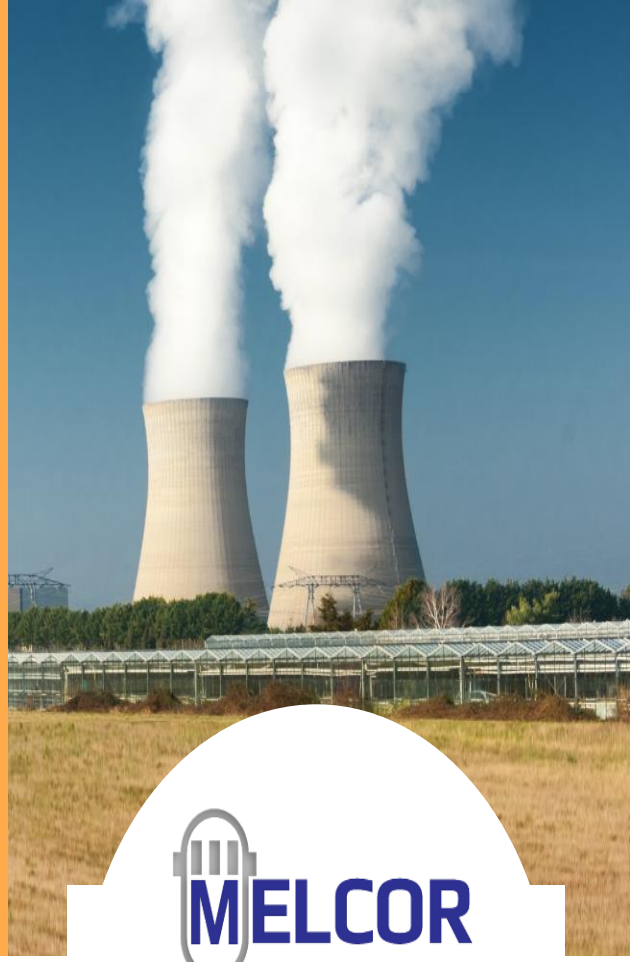
Sandia National Laboratories

Securing the future of Nuclear Energy



MELCOR

Advanced MELCOR Control Function Topics

MELCOR Development Team

U.S.NRC

SAND2024-04106O

U.S. DEPARTMENT OF ENERGY NNSA

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.

Objective of Presentation

- Provide a very cursory overview of the CF package
  - Various CF types
  - Required and optional input
  - CF evaluation
  - Likely insufficient discussion for a new user
    - Refer to the UG manual for more description
    - In-depth discussions in week-long workshop
- Main objective is to focus on more advanced topics
  - Often there is insufficient time in a week-long workshop to discuss advanced topics.
  - Many new features have been added that experienced users may be unaware of.
  - Examples drawn from applications

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
- 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
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 CFn = scale*fn[X(t)] + add
  - Upper and lower bounds
    - Results bounded within limits
  - Units (used for plotting purposes only)
- 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

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

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
- Proportional-Integral-Differential Control Function (PID)

$$f^n = R_1\, a_1^n + R_2 \int_{t^n}^{t^n} a_1(t)\, dt + R_3 \left.\frac{da_1}{dt}\right|_{t^n}$$

- 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.
- 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

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.

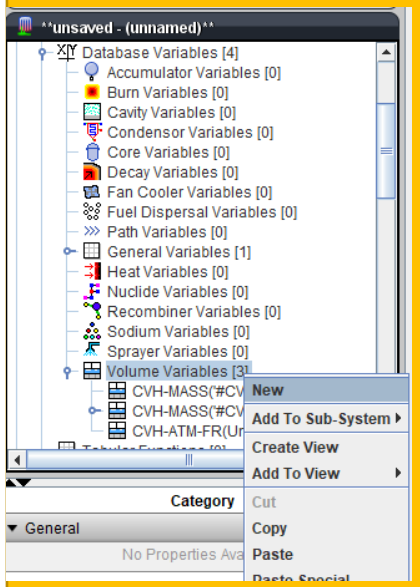
The choices permitted for NameMat always include 'POOL', 'FOG', 'H2O-VAP', or those other materials identified by input to the NonCondensible Gas (NCG) package. In certain cases (see below) the keywords 'TOTAL' (or 'ALL'), 'WATER' are also interpreted to mean, respectively, the total contribution from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

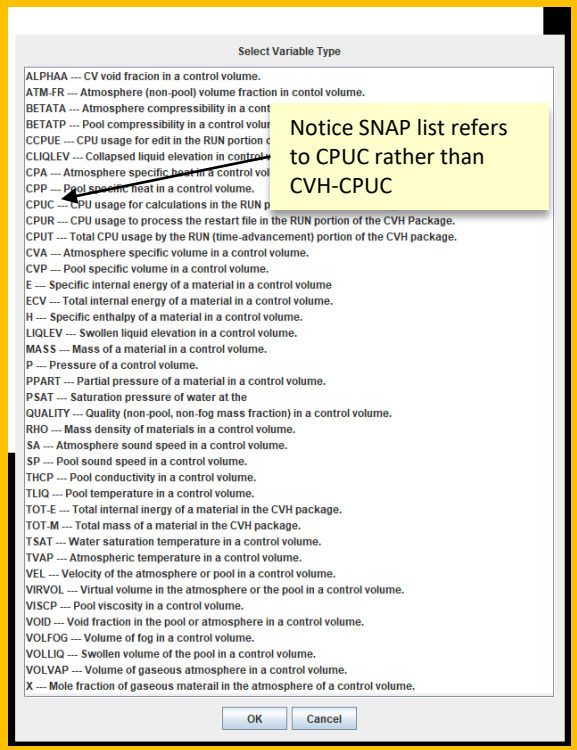
CVH-UG-65

UG list refers to CVH-CPUC

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



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



Notice SNAP list refers to CPUC rather than CVH-CPUC

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'

This is different than Vectorized control functions and Ranges. This will be discussed later

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

MELCOR Control Functions
Use of CF-CONST is best practice

- 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

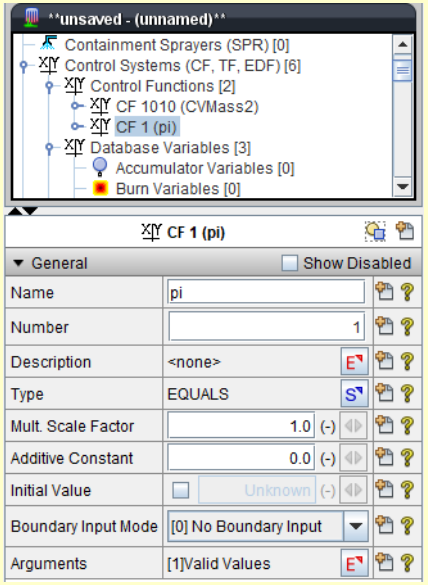
- Best Practice

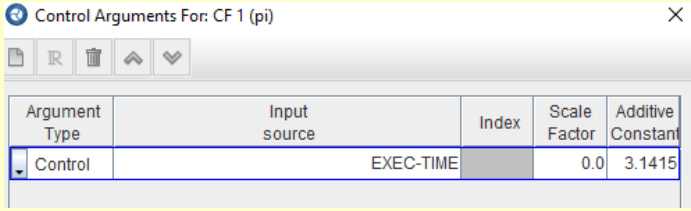
```
CF_ID 'Pi' 10 EQUALS
CF_SAI 1.0  ! Mult 1; Add 0.0 (default)
CF_ARG 1 ! NARG CHARG     ARSCAL   ARADCN
            1  CF-CONST   3.1415
```

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

Consider referencing this CF in other CFs for consistency in constants

SNAP Implementation of alternate form 2





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

Flexibility in Constructing CFs
Alternate Ways for Calculating Pipe Stress

- Equation for Simple Pipe Stress

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

- Using PIPE-STR type CF

```
CF_ID Stress PIPE-STR
CF_SAI 1.0 0.0
CF_MSC 0.37 0.45 !Inner & Outer radii
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)
```

Flexibility in Constructing CFs
Alternate Ways for Calculating Pipe Stress

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

This method not recommended!
Harder to read and more prone to mistakes!
Note: 8 Control Functions Used

- Using MELCOR Classic Control Functions (MELCOR 1.8.5)

```
CF_ID STRESS DIVIDE 17
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 16
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 MULTIPLY 15
CF_SAI 1.0 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
       1 CF-VALU(SumR2) 1. 0.
       2 CVH-P(CV500) 1. 0.

CF_ID SumR2 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 2.0 0.0
CF_ARG 2 ! NARG CHARG ARSCAL ARADCN
       1 CF-VALU(RO2) 1. 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.
```

MELCOR Control Functions
Using CF to Control MELCOR Output and Time Step Control

- When 'failure' occurs
  - Generate restart and plot at time of failure
  - Reduce time step for next 100.0 seconds

```
EXEC_INPUT
EXEC_RESTARTCF 'E+R Flag'
EXEC_PLOTCF    'E+R Flag'
EXEC_DTMAXCF    'dt-control'
...
```

'Failure' becomes true and stays true after condition met.

'E+R Flag' is true only on one cycle when Failure initially occurs (ONE-SHOT).

Maximum time step reduction for next 100 seconds after failure occurs (1stReduction)

```
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.
CF_ID 'Failed'  106  L-EQUALS
CF_LIV FALSE    ! Initial value is .false.
CF_CLS 'LATCH' ! Once true, always true
CF_ARG 1
       1 CF-VALU('Failure')   0.    0.
```

```
CF_ID  dt-control 1000 max
CF_SAI 1.0 0.0 -1.0
cf_arg 2
       1 CF-VALU(1stReduction) 1.0 0.0
       2 CF-VALU(2ndReduction) 1.0 0.0
!
CF_ID tfail 1002 trip !Time since failure
CF_SAI 1.0 0.0 0.0
CF_ARG 1
       1 CF-VALU('Failed')
!
CF_ID 't<100' 1003 l-gt !<100 s after failure
CF_LIV false
CF_ARG 2
       1 CF-CONST 100.0
       2 CF-VALU(tfail) 1.0 0.0
!
CF_ID 'Fail&t<100' 1004 l-and
CF_LIV false
CF_ARG 2
       1 CF-VALU(Failed)  ! failed (latch)
       2 CF-VALU('t<100')  ! timer
!
CF_ID 1stReduction 1005 l-a-ifte
CF_SAI 1.0 0.0 -1.0
CF_ARG 3
       1 CF-VALU('Fail&t<100')
       2 CF-CONST        0.01
       3 CF-CONST        -1.0
```

when tfail >100.0 s, return to MELCOR maximum time step

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)

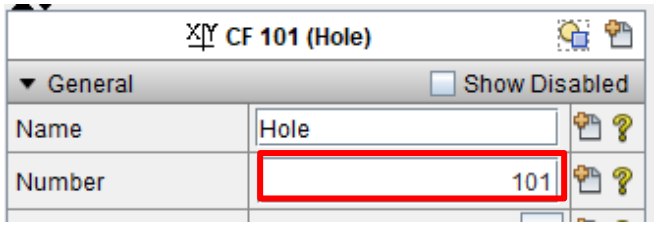
Example: T-O-R

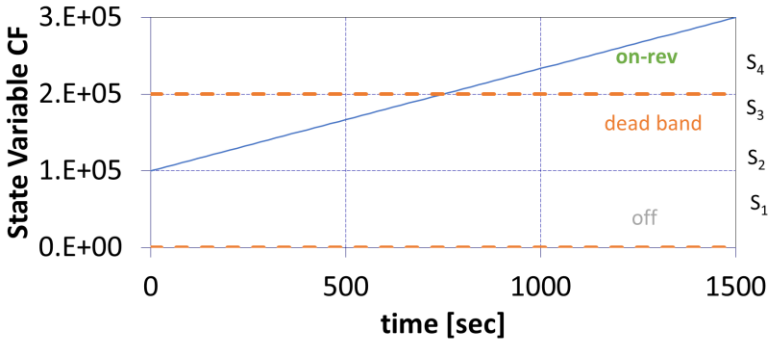
```
CF_ID     'Hole'    T-O-F
CF_SAI     0.5      0.0     0.0
CF_MSC    -1.0      2.E5
CF_ARG   1   ! Pressure calculated by CF
         1   CFVALU('pressure')   1.0    0.0
```

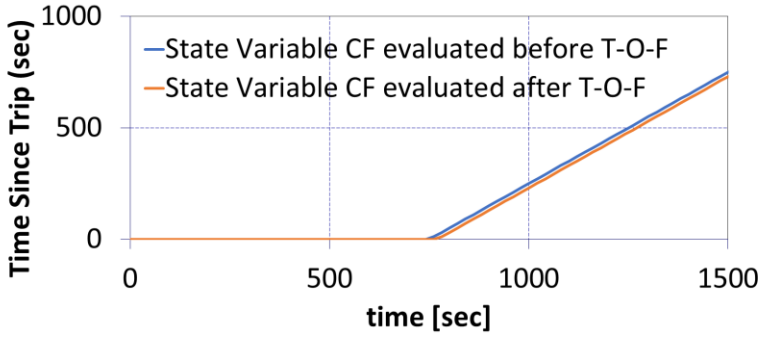
Value of trip is different whether state variable CF ('pressure') is evaluated before or after CFVALU('Hole').

Difference is time-step dependent.

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







Example of CF Intentionally Using a Value From Previous Time Step

- 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.

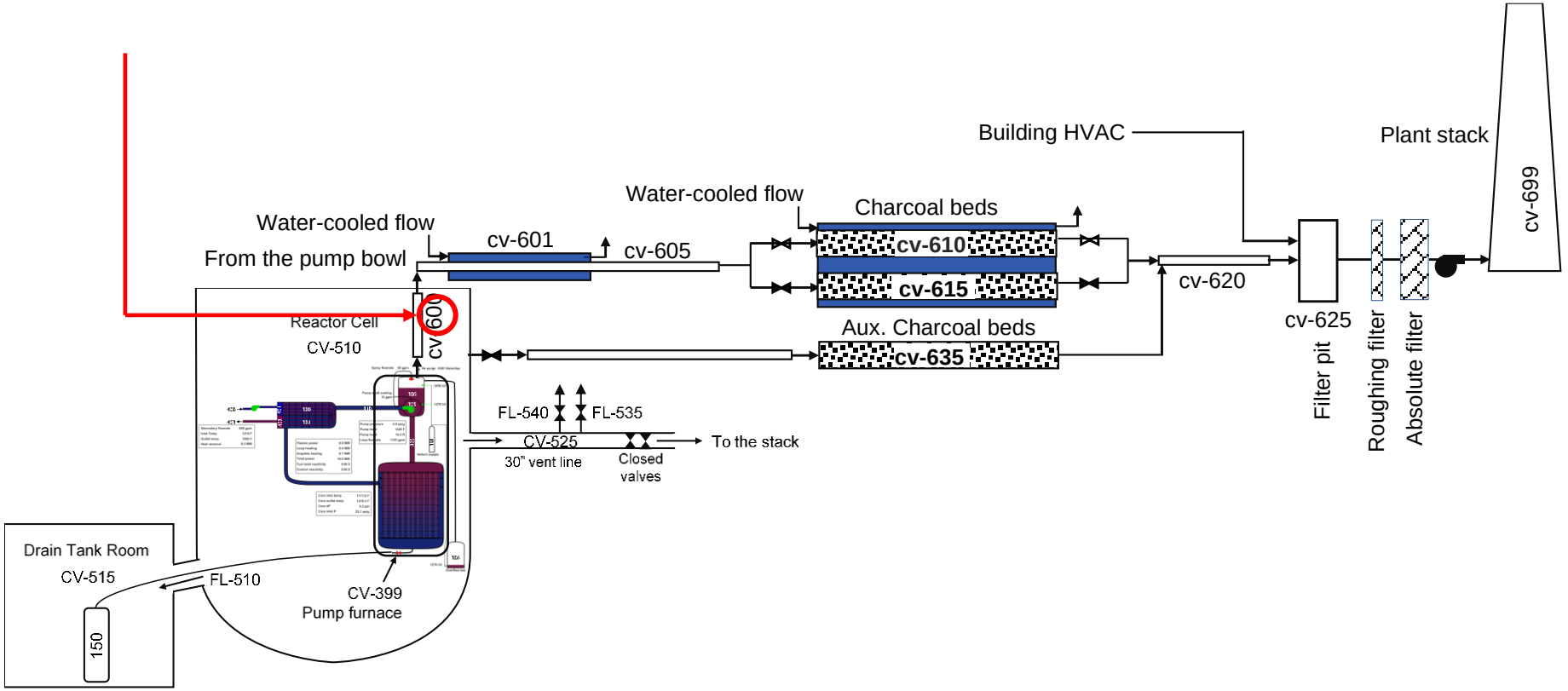
- Change any CF and TF parameters from the restart
  - Allow addition of new CFs and TFs
  - Easy to run variations of a failure criterion
  - Run multiple scenarios that branch late in a sequence
    - Define input to include several failure paths
    - Run alternate sequences by restarting from a point 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

- 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.

Application of LAG Control Function
MSR- offgas system

Lag example for off-gas filter

- Helium purge flow introduced via the pump bowl, overflow tank, and the pump shaft
- Off-gas flow control valve adjusted to maintain +5 psig over-pressure by regulating the helium exit flow
- Entire primary loop pressure response to changes in the valve position



Lag control function used to smooth pressure signal for valve controller

- MELCOR variables are updated at the timestep frequency and may be too rapidly changing
- Prior to lag variable, old-time weighting was used to smooth input signals

$$P^{t} = c * P^{t} + (1 - c) * P^{t-1}$$

Where P is a signal (e.g., pressure) and “t” is current time and “t-1” is the previous timestep value and c varies the amount of old-time weighting

This is a bad practice because it is timestep dependent!

- The lag is not timestep dependent and can be used to smooth input or output signals for plant controllers.
- Example on the RHS is a Proportional-Integral controller for the off-gas valve to maintain a 5 psig over-pressure

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Pump-bowl Pressure stabilization
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
cf_id           'P_lag'      1098                    lag
cf_sai          1.0          0.0                     0.0000E+00
cf_msc                       0.25                    1.0
cf_arg          1
                1            cvh-p('Pump_Bowl')      1.0

cf_id           'Perr'       1099                    Formula
cf_sai          1.0          0.0                     -2.1399E+00
cf_formula      3            active*(P-Pset)
                1            active                  cf-valu('OffG_Vlv')
                2            P                       cf-valu('P_lag')
                3            Pset                    135798.8
!
cf_id           'OG_I'       1100                    integ
cf_sai          1.0          0.0                     9.8925E+05
cf_ulb          1            0.0
cf_arg          2
                1            cf-valu('Perr')         1.0
                2            exec-time               1.0

cf_id           'OG_PI'      1101                    Formula
cf_sai          1.0          0.0                     9.8903E-01
cf_ulb          1            0.0
cf_formula      6            l-a-ifte(active,freeze,gain1*Perr+gain2*Pi)
                1            active                  cf-valu('Transient')
                2            freeze                  cf-valu('OG_PI')
                3            gain1                   1.0E-04
                4            Perr                    cf-valu('Perr')
                5            gain2                   1.0E-06
                6            Pi                      cf-valu('OG_I')
```

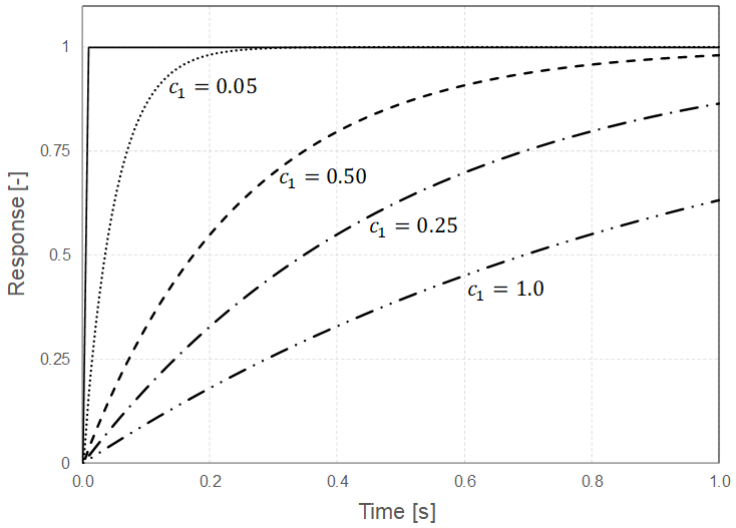
Lag Control Function (Implementation)

- Use of Lag CF to process thermal and fluid data for system control processes.

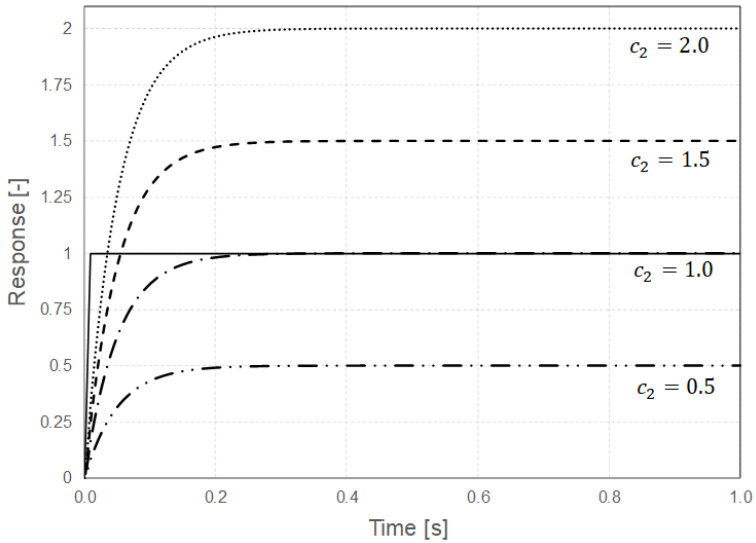
```
CF_ID 'LAG'          5028    LAG
CF_SAI 1.0    0.0     0.0
CF_ARG 1
       1 CF-VALU('cSteps')  1.0   0.0   !function, a1
CF_MSC 0.5           1.0
```

first MSC parameter is the time lag (c1) while the second is a multiplier (c2)

$$f(t) = \int_{t_0}^{t} \frac{c_2 a_1(t') - f(t')}{c_1} dt'$$



unit step varying c_1 with c_2 = constant = 1



unit step function varying c_2 with c_1 = constant = 0.01.

Lag Control Function (Solution)

Integral equation for lag:

$$f(t) = \int_{t_0}^{t} \frac{c_2 a_1(t') - f(t')}{c_1} dt'$$

Trapezoidal approximation to integral equation

$$f(t^{n+1}) \approx f(t^n) \cdot e^{-\frac{dt}{c_1}} + \frac{c_2}{c_1} \cdot \frac{dt}{2} \cdot \left[a(t^n) \cdot e^{-\frac{dt}{c_1}} + a(t^{n+1}) \right]$$

Third-order Pade (1,1) approximate for exponent

$$\exp\left(-\frac{dt}{c_1}\right) \approx \frac{1 - \frac{dt}{2c_1}}{1 + \frac{dt}{2c_1}}$$

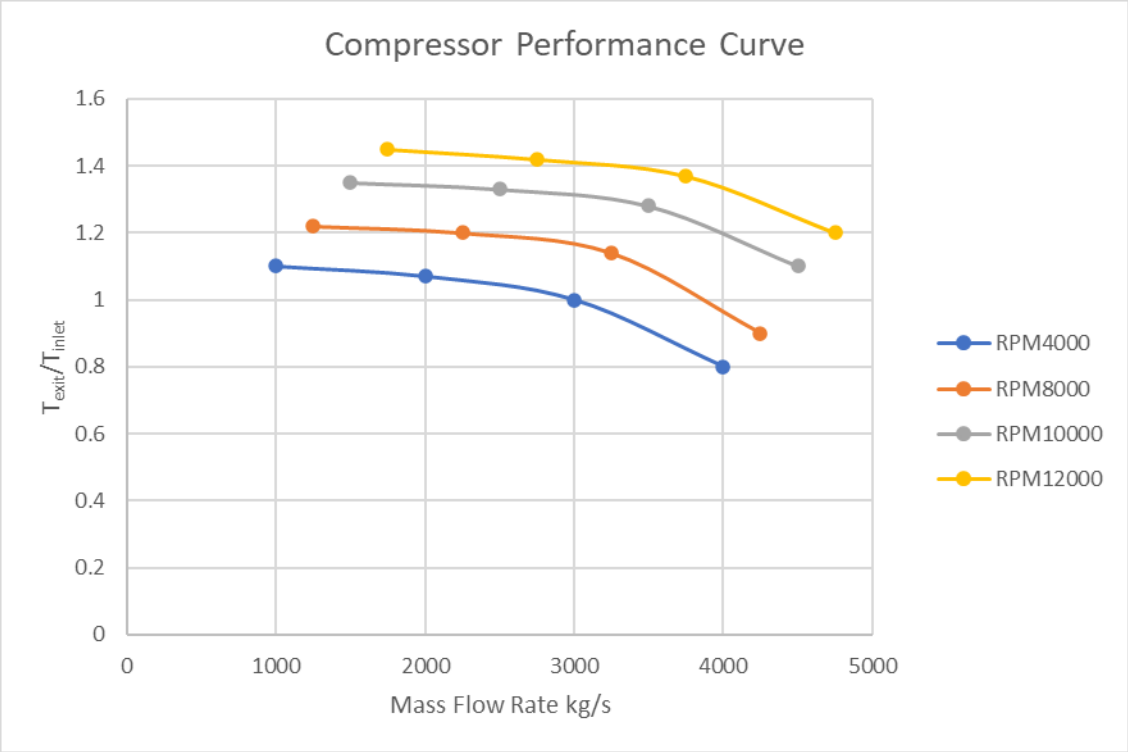
Substitution of exponent approximation into trapezoidal approximation to integral equation

$$f(t^{n+1}) \approx \frac{f(t^n) \cdot \left(1 - \frac{dt}{2c_1}\right) + \frac{c_2 \cdot dt}{2c_1} \left[a(t^n) + a(t^{n+1}) \right]}{\left(1 + \frac{dt}{2c_1}\right)} + dt^2 \cdot \left\{ \frac{\frac{c_2}{4\, c_1^2} \left[a(t^{n+1}) - a(t^n) \right]}{\left(1 + \frac{dt}{2c_1}\right)} \right\}$$

Solution, neglecting higher order terms

$$f(t^{n+1}) = \frac{f(t^n) \cdot \left(1 - \frac{dt}{2c_1}\right) + \frac{c_2 \cdot dt}{2c_1} \left[a(t^n) + a(t^{n+1}) \right]}{\left(1 + \frac{dt}{2c_1}\right)}$$

CF for Interpolation of 2-Dimensional Table



Numbers are not representative of any known component

RPM4000		RPM8000		RPM10000		RPM12000	
1000	1.1	1250	1.22	1500	1.35	1750	1.45
2000	1.07	2250	1.2	2500	1.33	2750	1.42
3000	1	3250	1.14	3500	1.28	3750	1.37
4000	0.8	4250	0.9	4500	1.1	4750	1.2

CF for Interpolation of 2-Dimensional Table

$$\frac{T_{exit}}{T_{inlet}} = f(\omega(rpm), \dot{m}(\frac{kg}{s}))$$

```
cf_input
cf_id ‘Texit’ 8021 tab-fun
cf_sai 1.0 0.0 0.0
cf_msc ‘RPM'
cf_arg 1
     1 cf-valu('rpm') 1.0 0.0
```

```
tf_input
tf_id ‘RPM' 1.0 0.0
tf_tab 4 ! rpm
     1 4000.00  ‘rpm4000'
     2 8000.00  ‘rpm8000'
     3 10000.00  ‘rpm10000'
     4 12000.00  ‘rpm12000'
```

```
cf_id ‘rpm4000' 8011 tab-fun
cf_sai 1.0 0.0 0.0
cf_msc ‘rpm4000'
cf_arg 1
     1 cf-valu('flow') 1.0 0.0
```

```
tf_id ‘rpm4000' 1.0 0.0
tf_tab 4 ! flow
     1 1000.00 1.1
     2 2000.00 1.07
     3 3000.00 1.0
     4 4000.00 0.8
```

```
cf_id ‘rpm8000' 8012 tab-fun
cf_sai 1.0 0.0 0.0
cf_msc ‘rpm8000'
cf_arg 1
     1 cf-valu('flow') 1.0 0.0
```

```
tf_id ‘rpm8000' 1.0 0.0
tf_tab 4 ! flow
     1 1250.00 1.22
     2 2250.00 1.2
     3 3250.00 1.14
     4 4250.00 0.9
```

```
cf_id ‘rpm10000' 8013 tab-fun
cf_sai 1.0 0.0 0.0
cf_msc ‘rpm10000'
cf_arg 1
     1 cf-valu('flow') 1.0 0.0
```

```
tf_id ‘rpm10000' 1.0 0.0
tf_tab 4 ! flow
     1 1500.00 1.35
     2 2500.00 1.33
     3 3500.00 1.28
     4 4500.00 1.1
```

```
cf_id ‘rpm12000' 8014 tab-fun
cf_sai 1.0 0.0 0.0
cf_msc ‘rpm12000'
cf_arg 1
     1 cf-valu('flow') 1.0 0.0
```

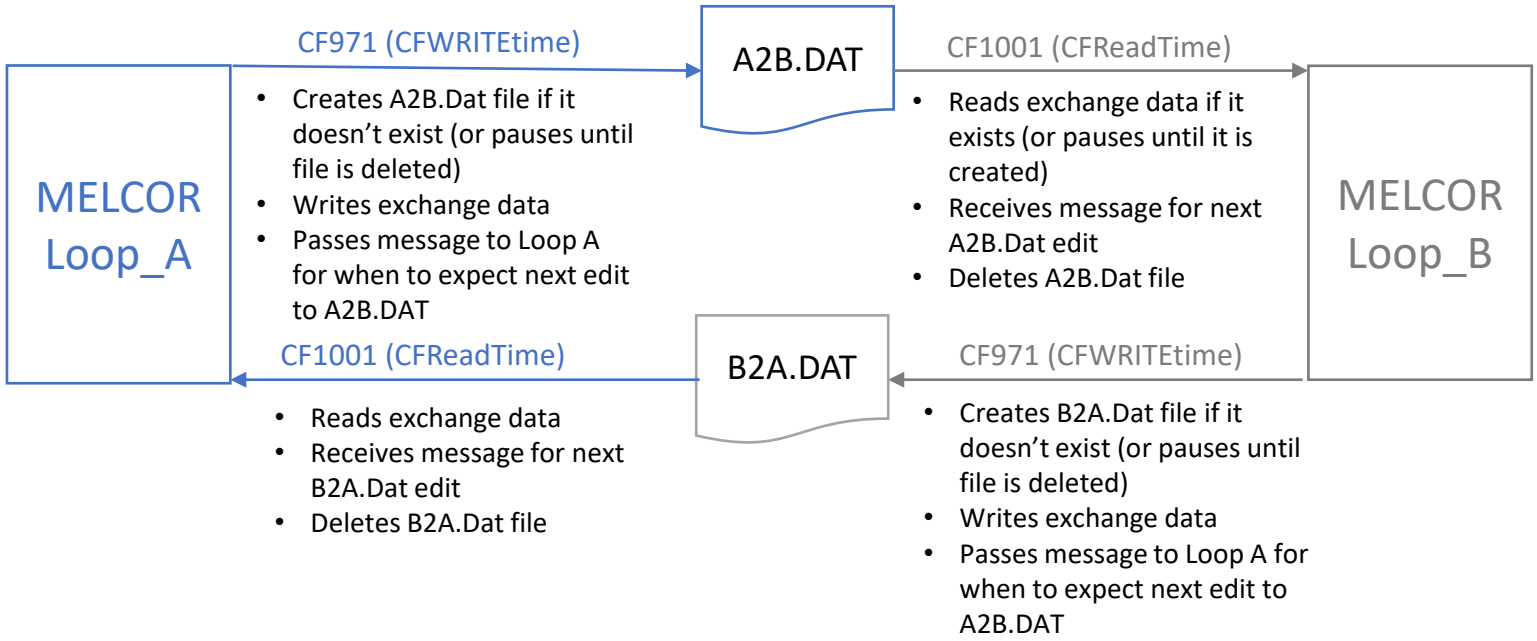
```
tf_id ‘rpm12000' 1.0 0.0
tf_tab 4 ! flow
     1 1750.00 1.45
     2 2750.00 1.42
     3 3750.00 1.37
     4 4750.00 1.2
```

(

MELCOR ‘READ’ and L-READ’ Control Functions

- 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
  - Data file is immediately deleted after it is read by the CF
- Similarly, a WRITE type CF was developed to write to a changedata file.
  - Writes the time channel and a number of output variables to an exchange file
  - Does not delete this output file
  - Skips writing to the file until the file has been deleted externally.

Simple Explicit Coupling with Read/Write Control Functions



Loop_A

```
EXEC_CFEXFILE B2A.DAT
…

CF_ID   'CFreadTime'    1001   READ

CF_ID      'CFWRITEtime'        971       WRITE
CF_MSC     'CFreadTime'
CF_ARG 1    !                NARG                    CHARG
 1   CF-VALU('CFreadTime')            1.00          0.0


EXEC_CFEXFILE 'B2A.DAT'  -  'CFreadTime'
EXEC_CFEXWRITE '..\LOOPB\A2B.DAT'
```

Loop_B

```
EXEC_CFEXFILE A2B.DAT
…

CF_ID   'CFreadTime'   1001   READ

CF_ID      'CFWRITEtime'        971       WRITE
CF_MSC     'CFreadTime'
CF_ARG 1
1   CF-VALU('CFreadTime')              1.0           1.0

EXEC_CFEXFILE A2B.DAT  -  'CFreadTime'
EXEC_CFEXWRITE '..\LOOPA\B2A.DAT'
```

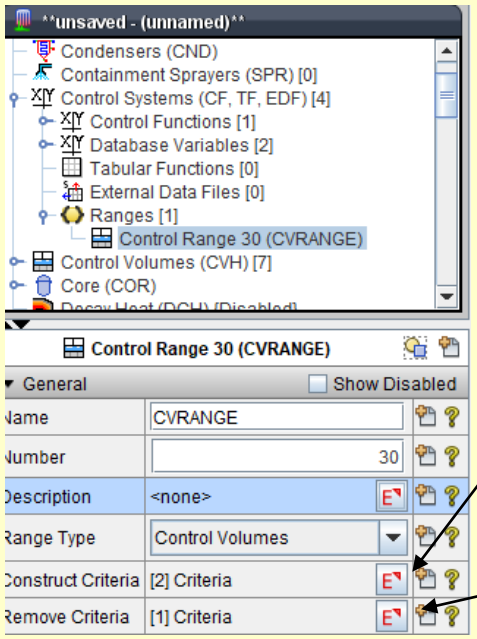
Control Function Ranges

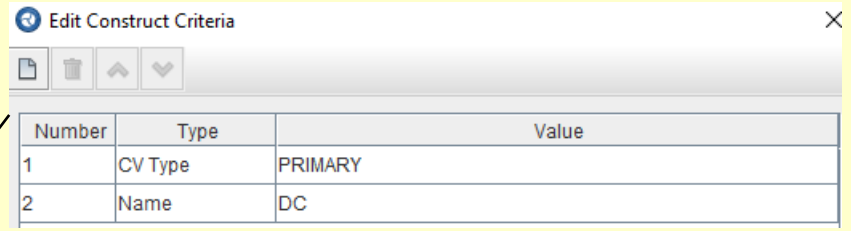
The range is an object that is defined once in the database and then can be referenced by one or more control function arguments. The range specifies an ordered list of objects such a control volumes, COR cells, materials, or components

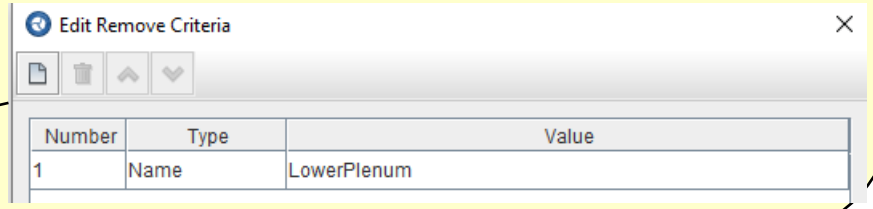
Define a Range (ASCII):

```
                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):







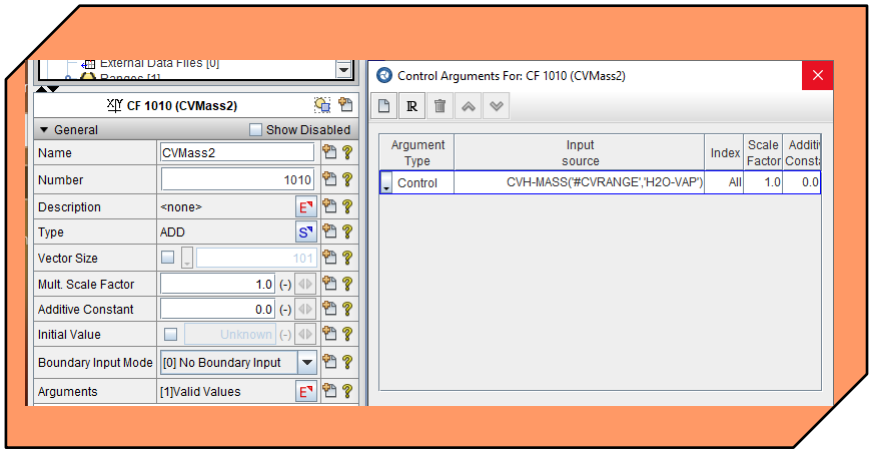
Control Function Ranges

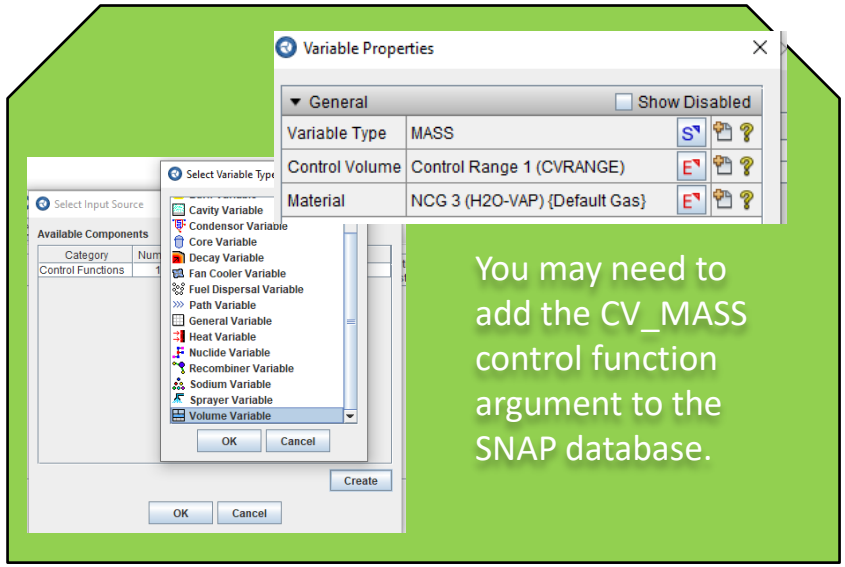
A range can be referenced by control functions and control function arguments and may also be used in specifying input. The hashtag (#) that precedes range specified for the volume in the CF argument indicates a range of control volumes rather than a single volume.

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):

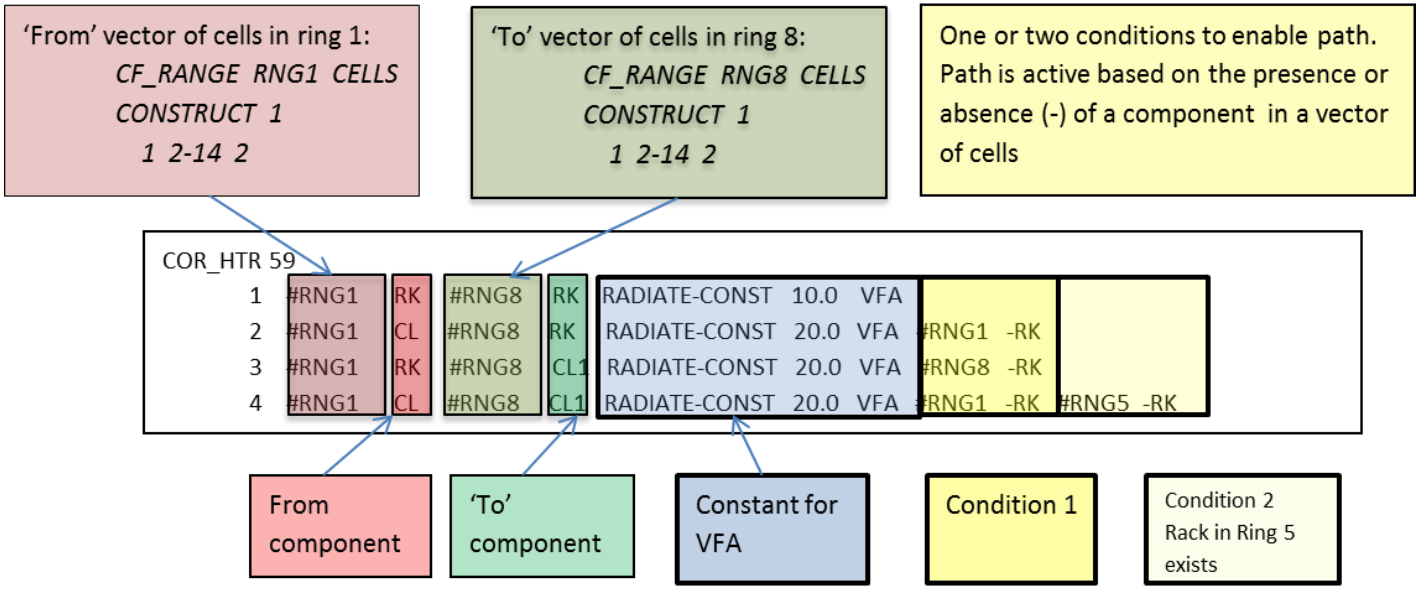




You may need to add the CV_MASS control function argument to the SNAP database.

Ranges Used in Input: Vectorized COR_HTR Input

- Reduces number of input records significantly.
  - Otherwise input is required cell by cell.
  - Unnecessary CF logic required to determine existence of components.
  - Difficult to read (QA)
    - Input for a cell is scattered among COR_HTR records and multiple CF records
  - One example reduced number of records from over 7000 records to under 100



Vector Control Functions

- A control function may be defined as a vector of a given size by applying the CF_VCF record.
  - CF_VCF 10 !This control function will have 10 elements
  - CF_VCF #MyRange !This control function will have the same dimension as MyRange
- A vectorized control function performs its specified operation, such as ADD, returning a unique result for each vector element.
  - Each element defined from the list supplied by a control function ranges or the similarly indexed value from vector control functions.
- A vectorized control function returns a range of values that can be passed to other control functions, either
  - As a range, i,e., CF-VALU(MyVectorCF)
  - By individual elements,i,e., CF-VALU(MyVectorCF[2])

Type	Available	Not Planned for Addition
Real Value	FORMULA, ADD, MULTIPLY, DIVIDE, ABS, SQRT, SIN, COS, TAN, EQUALS, Power-I, Power-R, Power-V, EXP, LN, LOG, MAX, MIN, L-A-IFTE, ARCSIN, ARCCOS, ARCTAN, SINH, COSH, TANH, SIGNI, SIGN, UNIT-NRM, DIM, LM-CREEP	DER-F, DER-C, INTEG, PID, TAB-FUN, LAG, HYST, All TRIPs, All User Defined Functions
Logical Value	L-GT, L-GE, L-EQ, L-NE, L-EQUALS, L-NOT, L-AND, L-OR, L-EQV, L-L-IFTE	

Vector & Scalar Functions with Vector Arguments

- All real vectorized control functions are 'overloaded' and the calculated results depends on whether the control function is declared as a VCF or not.

MELCOR Control Function Descriptor	Vector CF VCF #MyRange	Scalar CF No VCF record

`cf_id 'Sum' 100 ADD`
`cf_sai 1.0 0.0 0.0000E+00`
`cf_vcf #MyRange !Absent for Scalar CF`
`cf_arg 2`
`1 CF-VALU("MyCF") 1.0`
`2 CF-CONST 2.0`
	$\vec{CF} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + k = \begin{pmatrix} k + x_1 \\ k + x_2 \\ k + x_3 \end{pmatrix}$	$CF = \sum_i^n (k + x_i)$

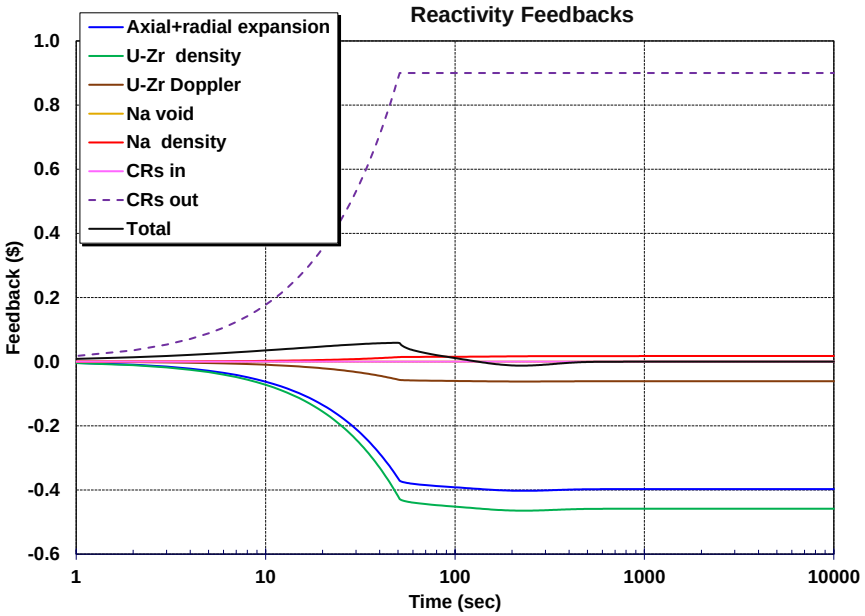
`cf_id 'Product' 100 Multiply`
`cf_sai 1.0 0.0 0.0000E+00`
`cf_vcf #MyRange !Absent for Scalar CF`
`cf_arg 2`
`1 CF-VALU("MyCF") 1.0`
`2 CF-CONST 2.0`
	$\vec{CF} = x \bullet k = \begin{pmatrix} k \bullet x_1 \\ k \bullet x_2 \\ k \bullet x_3 \end{pmatrix}$	$CF = \sum_i^n (k \bullet x_i)$

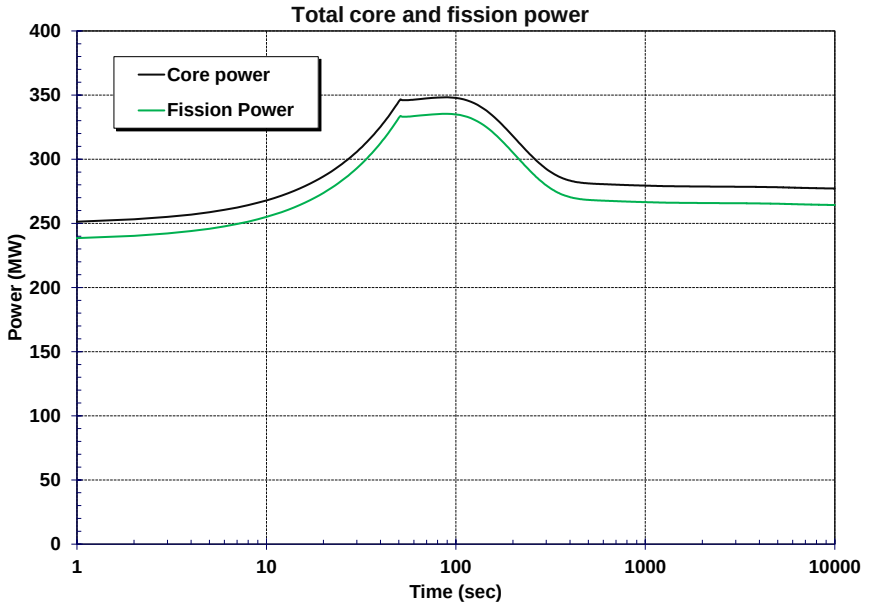
`cf_id 'Product' 100 Multiply`
`cf_sai 1.0 0.0 0.0000E+00`
`cf_vcf #MyRange !Absent for Scalar CF`
`cf_arg 2`
`1 CF-VALU("MyCF") 1.0`
`2 CORcelltemp(#MyRange,FU)`
	$\vec{CF} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \bullet \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} y_1 \bullet x_1 \\ y_2 \bullet x_2 \\ y_3 \bullet x_3 \end{pmatrix}$	$CF = \sum_i^n (y_i \bullet x_i)$

Application of Vector CFs
UTOP – Withdraw of highest-worth CR

- The highest-worth CR withdraws over 51 sec to insert 0.9$.
- The net reactivity initially increases but is subsequently balanced by the negative feedbacks after the CR is withdrawn

- The core power rises to 346 MW in response to the reactivity insertion but subsequently drops in response due to the strong negative fuel feedback.
- The long-term power stabilizes at 280 MW
  - The maximum intermediate loop heat removal was assumed to be limited to (280 MW) ~112% of rated





MELCOR Point Kinetics

- Required inputs (cor_pkm0x)
  - All relevant feedbacks in dollars [$] – example uses vector control functions
  - Control rod worth for SCRAM [$]
  - Any neutron sources [neutron/s]
- 6 delayed-neutron group decay constants in sensitivity coefficient 1405
  - Default developed for a high-temperature gas reactor (HTGR) (thermal neutron reactor)
- Other reactor-specific point kinetics data in sensitivity coefficient 1406
  - For example, sc-1406(2) is the total effective delayed neutron fraction, **β**
- Disable built-in feedbacks (sensitivity coefficient 1404)
  - Default feedbacks originally formulated for high-temperature gas reactor (HTGR)

```
cor_sc      6
            1     1404     0.0      1
            2     1404     0.0      2
            3     1404     0.0      3
            4     1404     0.0      4
            5     1404     0.0      5
            6     1404     0.0      6
            7     1404     0.0      7
```

Feedback Effect	SCALE Value
Axial fuel expansion coefficient (cents/K)	-0.1347 ± 0.0033
Radial grid plate expansion coefficient (cents/K)	-0.3376 ± 0.0067
Fuel density coefficient (cents/K)	-0.2444 ± 0.0044
Structure density coefficient (cents/K)	-0.0125 ± 0.0021
Sodium void worth ($)	-0.4623 ± 0.0165
Sodium density coefficient (cents/K)	-0.1252 ± 0.0389
Doppler coefficient ($ with T in K)	-1.004 ln(T) + 15.67
Sodium voided Doppler coefficient ($ with T in K)	-0.776 ln(T) + 13.68
Primary control assemblies ($)	-22.07
Secondary control assemblies ($)	-15.77

SFR fuel Doppler feedback example

First, define fuel temperatures vector range

```
cf_range    RANGEFU cells 1
construct   1   ! Axial Radial
            1      4-13    1-6
```

Second, get fuel temperatures

```
cf_id       'Tfu'   4001   formula
cf_sai      1.0     0.0    0.0000E+00
cf_vcf      #RANGEFU
cf_formula  1   T
            1   T   cor-celltemp(#RANGEFU,fu)
```

Third, calculate feedback

```
cf_id       'fb-Dopp0'    4014    formula
cf_sai      1.0           0.0
0.0000E+00
cf_vcf      #RANGEFU
cf_formula  3     a*ln(T)+b
            1     a    -1.004
            2     b    15.67
            3     T    cf-valu('Tfu')
```

Fourth, apply weighting factors (e.g., volume, power, power^2)

```
cf_id       'fb-Dopp1'   4015     add
cf_sai      1.0          0.0      0.0000E+00
cf_arg      60
            1      cf-valu('fb-Dopp0')[1]     1.7647E-03
            2      cf-valu('fb-Dopp0')[2]     8.8236E-03
            3      cf-valu('fb-Dopp0')[3]     9.7116E-03
            4      cf-valu('fb-Dopp0')[4]     3.0481E-02
            5      cf-valu('fb-Dopp0')[5]     1.5723E-02
…
            58     cf-valu('fb-Dopp0')[58]    2.4429E-02
            59     cf-valu('fb-Dopp0')[59]    1.2601E-02
            60     cf-valu('fb-Dopp0')[60]    1.3301E-02
!                                             1.0000E+00
```

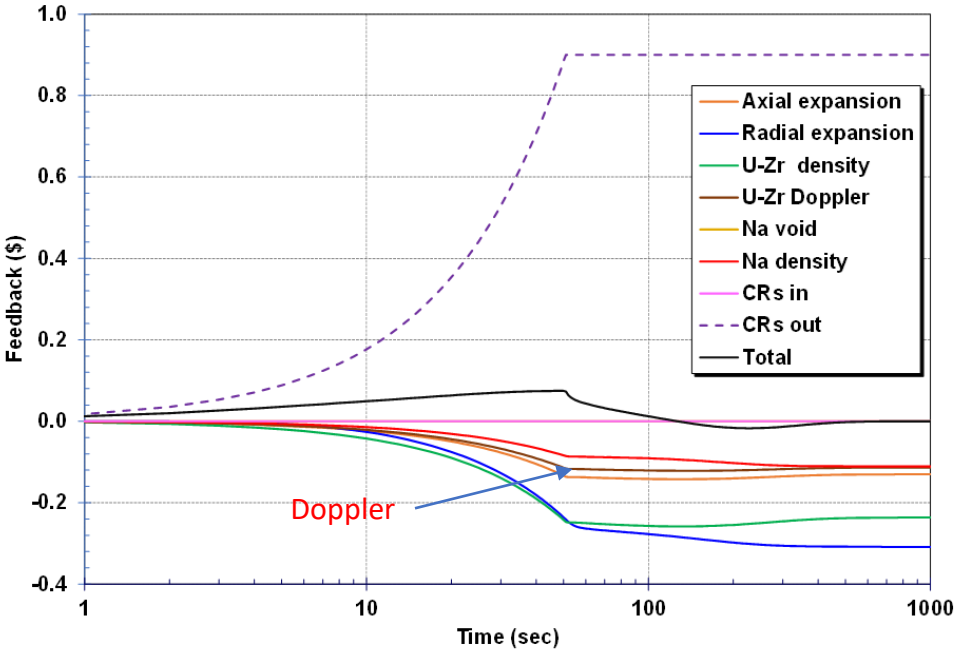
Fifth, freeze steady state values

```
cf_id       'fb_Dopp-ss'     4016       formula
cf_sai      1.0    0.0       0.0000E+00
cf_formula  4      l-a-ifte(t>t0,self,fb)
            1      t          exec-time
            2      t0         -10.0
            3      self       cf-valu('fb_Dopp-ss')
            4      fb         cf-valu('fb-Dopp1')
```

SFR fuel Doppler feedback example

Sixth, calculate the Doppler change from full-power steady state conditions

```
cf_id        'del_Dopp'    4017    formula
cf_sai       1.0           0.0     0.0000E+00
cf_formula   2   fb-fbss
             1   fb          cf-valu('fb-Dopp1')
             2   fbss        cf-valu('fb_Dopp-ss')
```

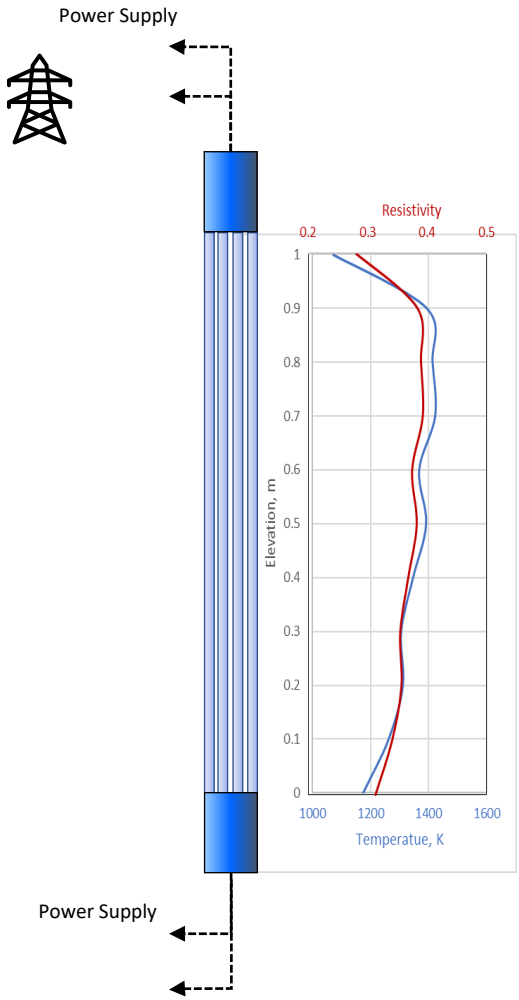


Seventh, sum feedbacks

```
cf_id        'React'       4029    formula
cf_sai       1.0           0.0     0.0000E+00
cf_formula   8      Axial+Radial+FuRho+Doppler+NaVoid+NaRho+CRout+CRin
             1      Axial      cf-valu('fb-FuExp')
             2      Radial     cf-valu('fb-RadExp')
             3      FuRho      cf-valu('fb-FuRho')
             4      Doppler    cf-valu('del_Dopp')
             5      NaVoid     cf-valu('del_void')
             6      NaRho      cf-valu('del_NaRho')
             7      CRout      cf-valu('CR-out')
             8      CRin       cf-valu('CRs-in')
```

Example of High Level Vector Control Function Electric Heater Element Modeling

- Electrical Heater Elements are frequently used to simulate nuclear heat generation in fuel rods
  - Voltage across bundle and resistance of fuel rods determines power generation
  - Local resistivity dependent on local temperature.
  - Temperature feedback determines the power distribution in the bundle
- MELCOR modeling
  - Uses a high level control function that leverages ranges and vector control functions.
    - User specifies a range of COR cells and power functions for each ring in that range.
    - Allows multiple materials for elements (W, Cu, Mo)
    - Allows specification of electric losses external to elements
    - HTML expanded to include electrical heating element
      - resistivity as a function of temperature.
      - Video of temperature evolution.



$$R_{Ring\ j} = \sum_{\substack{i \\ Ring\ j\ cells}}^{NAXL} R_{i,j} = \sum_{\substack{i \\ Ring\ j\ cells}}^{NAXL} \frac{\rho_{i,j}}{Area_{i,j}} dz_i = \sum_{\substack{i \\ Ring\ j\ cells}}^{NAXL} \rho_{i,j} XLBYA_{i,j}$$

X-sectional Area, $Area_{i,j}$ is calculated implicitly from material masses and mass densities.

Electrical Heater Element Modeling – ELHEAT CF

- Electrical power generation calculated from high level CF, Qheater.
- Define a range of cells, RODS, which encompasses 22 axial levels and 3 rings.
- For each ring in Range RODS, define a CF that specifies that the ring power is obtained from the vector CF '**RingPow**' (which has 3 elements for 3 rings and pulls from previously defined EDF files)
- Specify a contact resistance vector control function (3 elements for 3 rings) **ConRes**
  - In this case, specifies a constant 0.0 contact resistance.
  - Could be temperature dependent

- COR_ELPOW **'Qheater'**
- ....
- *CF_ID **'Qheater'** 4016 ELHEAT*
  - *CF_SAI 1.0 0.0*
  - *CF_VCF **#RODS***
  - *CF_ARG 2*
    - *1 CF-VALU(**RingPOW**) 1.0 0.0*
    - *2 CF-VALU(**ConRes**) 1.0 0.0*
- *CF_ID **'RingPOW'** 4014 EQUALS*
  - *CF_SAI 1.0 0.0*
  - *CF_VCF 3*
  - *CF_ARG 3*
  - *!Power applied to each ring with electrical heaters (defined for 3 rings in Range RODS)*
    - *1 EDF('EDF20',3) 1.0 0.0*
    - *2 EDF('EDF20',1) 1.0 0.0*
    - *3 EDF('EDF20',2) 1.0 0.0*
- *CF_ID **'ConRes'** 4015 EQUALS*
  - *CF_SAI 1.0 0.0*
  - *CF_VCF 3*
  - *CF_ARG 3*
  - *!Resistance for each ring with electrical heaters (defined for 3 rings in Range RODS)*
    - *1 CF-CONST 0.000*
    - *2 CF-CONST 0.000*
    - *3 CF-CONST 0.000*
    -
- *! Range covering RODS*
- *CF_RANGE **RODS** CELLS 1000*
- *CONSTRUCT 3*
  - *1 1-22 1*
  - *2 1-22 2*
  - *3 1-22 3*

Analytical Function Type

- Analytical Functions allow the user to write the operations to be performed on the input arguments of the Control Function in Fortran code, which is then built to produce a Dynamic Link Library (DLL) which can be accessed by the main MELCOR code.
- Uses of analytical CFs
  - Analytical function CFs can be used either as input to other CFs or as output to the text output or plotfile
  - Alternatively, CFs can be used within MELCOR to replace existing modelling.
    - Control functions are evaluated once within the MELCOR calculation cycle after all packages are evaluated. Consequently, such values are said to be explicit. For most cases, explicit evaluations are sufficient. However, when states are changing rapidly, it may be desirable to have an implicit evaluation.
    - For the case of COR heat transfer coefficients, coding has been added to allow updates to the CF values within the package evaluation, i.e., implicit calculations.

      CF_MSC – Miscellaneous Numbers

      Optional

      (2) UPDATEFLAG

      This parameter can take the value 'OLD' or 'NEW'. If the value is 'OLD' then the Analytical Function will NOT be updated within the Package where it is used as a specific application, only outside of that Package with all the other Control Functions. If its value is 'NEW' then the Analytical Function WILL be updated within that Package, if the Package has been coded to allow it. If the Analytical Function does not have a specific application then any UPDATEFLAG setting is not used.

      *(type = characters *3, default = OLD)*

    - The following example demonstrates the use of CFs in calculating heat transfer coefficients.

Analytical Function Type – MELGEN Input

```
COR_CNV 4
1   'NU-FORCED-TURB-POOL'       'HT-SHROUD' 'HT-FCD-TURB-SPL'
2   'NU-FORCED-TURB-ATMS'       'HT-FCD-TURB-SPV'
3   'NU-NATURAL-TURB-ATMS'      'HT-TURB-PLATES'
4   'NU-NATURAL-LAM-ATMS'       'HT-LAM-PLATES'
```

COR input defines control functions to calculate heat transfer coefficients

```
!          CF name                ICFNum     CFTYPE
CF_ID      'HT-FRC-TURB-SPL'      5          ACF
!          Multiply  Add          Initial
CF_SAI     1.0       0.0          0.0
!          The range associated with the output
CF_VCF     #CORSURFACES
!          ANALYTICKEY                       AFAPP
CF_MSC     'DITTUS-BOELTER'       'NU-FORCED-TURB'
CF_ARG     2          ! NARG      CHARG      ARSCAL     ARADCN
  1        COR-RE-POOL(#CORSURFACES)         1.0        0.0
  2        COR-PR-POOL(#CORSURFACES)         1.0        0.0
```

This is an analytical CF type

Determines the dimenstion of the vector of values returned

Arguments passed to the FORTRAN DLL

CF Input to define interface with FORTRAN DLL

```
! Range covering all present surfaces in all fuelled COR cells
CF_RANGE CORSURFACES CELL-SURFACES 1002
CONSTRUCT 4
     1 7-10 1-2 "FU"
     2 7-10 1-2 "CL"
     3 7-10 1-2 "NS"
     4 7-10 2 "FM"
```

Surfaces for FU, CL, NS, and FM

CF Input to define ranges

Analytical Control Function Type – External User DLL

External User DLL

```
function user_evaluate_af(IDIM, AnalyticKey, UdfArguments, IERR, &
   ErrorMessage)

! Arguments
type(TUdfArguments), intent(inout) :: UdfArguments
integer, intent(inout) :: IERR
integer, intent(in) :: IDIM
character(len=*), intent(in) :: AnalyticKey
character(len=*), intent(out) :: ErrorMessage
real(real_kind) :: user_evaluate_af(IDIM)

!Internal variables
real(real_kind) :: ResultArray(IDIM), ReyNum, PranNum

select case (AnalyticKey)

case ('DITTUS-BOELTER')
!  NU = 0.023 * Re**0.8 * Pr**0.4

…Perform some error checking…

  Argument(1)=UDFarguments%Arguments(1)%Description
  Argument(2)=UDFarguments%Arguments(2)%Description

  if (Argument(1)(1:6)=='COR-RE' .AND. Argument(2)(1:6)=='COR-PR') then
    Order(1)=1
    Order(2)=2
  elseif (Argument(1)(1:6)=='COR-PR' .AND. Argument(2)(1:6)=='COR-RE') then
    Order(1)=2
    Order(2)=1
  else
    IERR = 1
    ErrorMessage = 'ACF ERROR: Dittus-Boelter arguments must be RE and PR'
    goto 100
  endif

!Now do the calculation
  do i=1,IDIM
    ReyNum = UDFarguments%Arguments(Order(1))%RealArray(i)
    PranNum = UDFarguments%Arguments(Order(2))%RealArray(i)
    if (ReyNum>0.0 .AND. PranNum>0.0) then
        ResultArray(i) = 0.023 * (ReyNum)**0.8 * (PranNum)**0.4
    else
        ResultArray(i) = -1.0
    endif

  enddo

case ('EXAMPLE VAP')

  …
```

Use of AnalyticKey allows different code to be processed for different CFs

- Function user_evaluate_af is the ‘evaluator’ function for the ACF type control function.
- Other processing of control functions is part of the generatl MELCOR release.
- User compiled DLL module.
  - uses interface modules (resources)

```
use M_ArgCF, only: TargUDF, TUdfArguments
use m_shared_constant_definitions, only: len_name16
use M_kind, only: real_kind
```

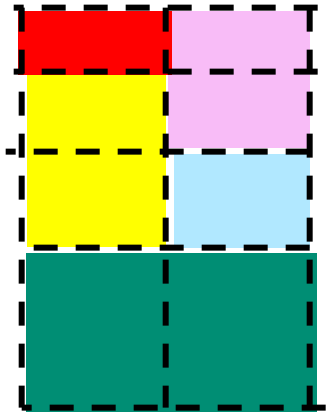
  - Examples to be provided with MELCOR release

MELGEN INPUT

```
CF_ID          'HT-FRC-TURB-SPL'              5              ACF
...
CF_VCF         #CORSURFACES  !Determines IDIM, dimension of ACF
CF_MSC         'DITTUS-BOELTER'     'NEW'
CF_ARG         2               ! NARG          CHARG
               ARSCAL          ARADCN
  1            COR-RE-POOL(#CORSURFACES)      1.0            0.0
  2            COR-PR-POOL(#CORSURFACES)      1.0            0.0
```

Dependent Ranges for Vector CFs

- A new keyword for constructing a range was added to facilitate the use of vectors in analytic functions. This new keyword references another range (i.e., #Range1) since it is entirely dependent on the other range for definition. As an example,
  - A range was constructed for all COR cells of interest in the calculation
  - it is required that the CV volume that is associated with a COR cell also be provided for each COR cell associated with that COR cell range and in the appropriate order.
  - This new construction keyword generates that range automatically and guarantees a one-to-one correspondence.



- COR cells indicated by dashed lines
- CV volumes indicated by colors

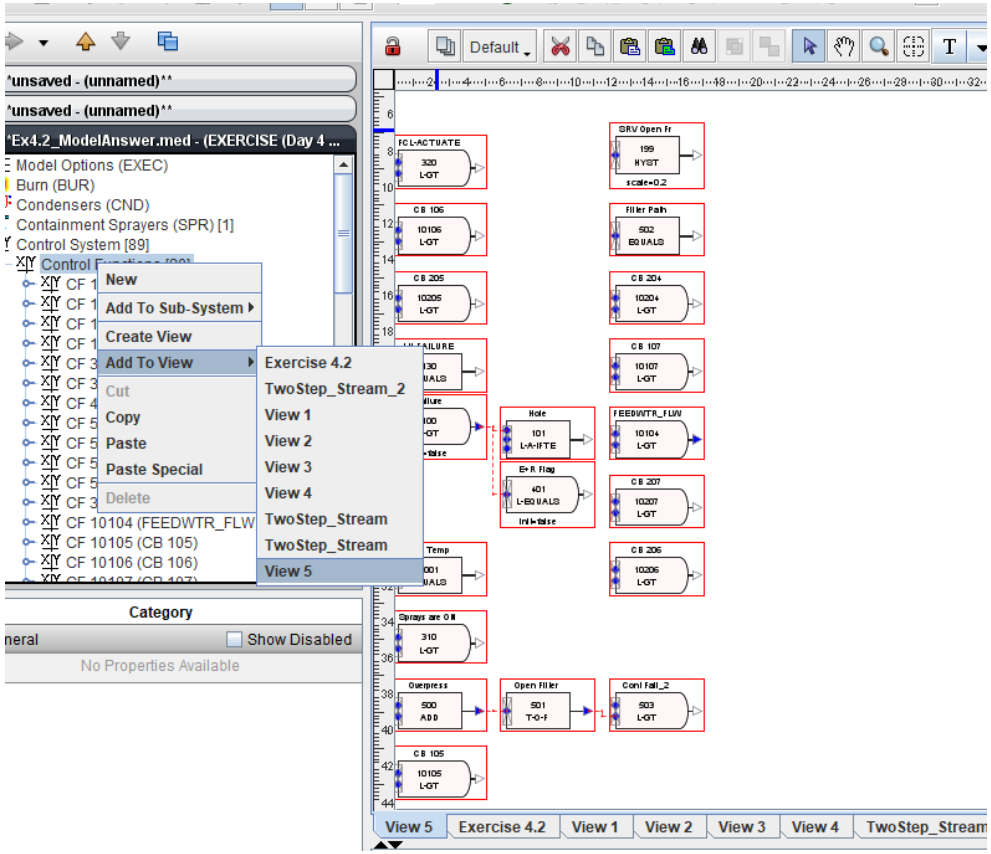
```
CF_RANGE CORCELLS2 CELLS 10
CONSTRUCT 1
  1 ALL
CF_RANGE CVCELLS CVOLUMES 20
CONSTRUCT 1
  1 #CORCELLS2
```

CORCELLS2	CVCELLS
Cell101	CV_GREEN
Cell102	CV_Yellow
Cell 103	CV_Yellow
Cell 104	CV_Red
Cell 201	CV_GREEN
Cell 202	CV_Blue
Cell 203	CV_Pink
Cell 204	CV_Pink

Viewing a Control Function Network in SNAP

- Create new view by right-clicking on Views in the Navigator
- Right click on a Control Function in Navigator pane
- Select add to view previously created

- Recently, a feature was added that allows the user to right click on a drawn control function and view an ASCII representation of its equation.
  - Version 2.7.1 SNAP plugin



Adding a CF to the HTML Output

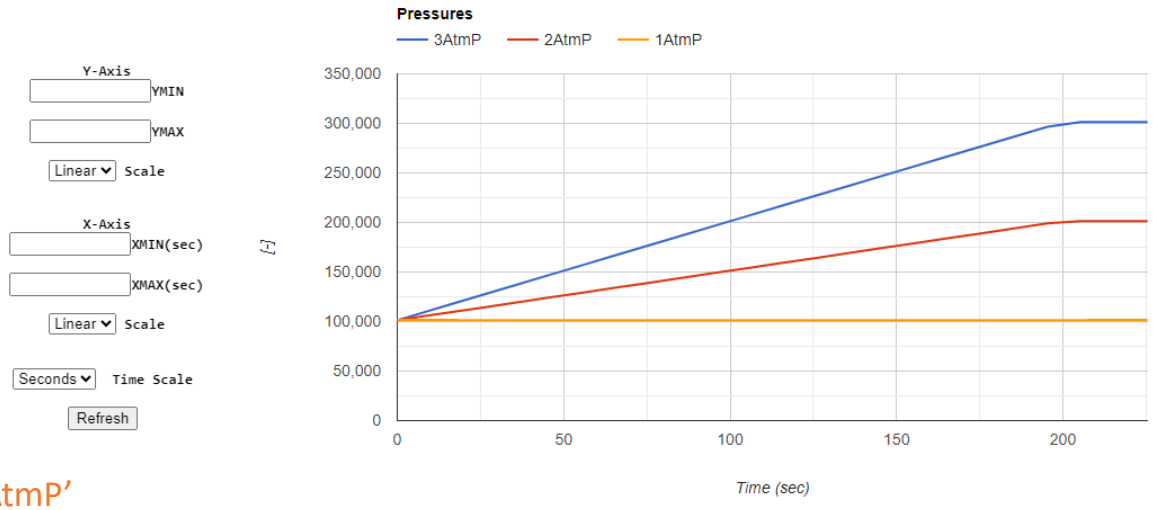
```
Program MELGEN
CF_INPUT !(
CF_ID '3AtmP' 1 EQUALS
CF_SAI 1.0 0.0
CF_ARG 1
                1 CVH-P('3Atm') 1.0

CF_ID '2AtmP’ 2 EQUALS
CF_SAI 1.0 0.0
CF_ARG 1
                1 CVH-P('2Atm') 1.0

CF_ID '1AtmP’ 3 EQUALS
CF_SAI 1.0 0.0
CF_ARG 1
                1 CVH-P('1Atm') 1.0

Program MELCOR
CF_INPUT !(
CF_HTML  4
1     'Pressures'   '3AtmP'  '2AtmP'  '1AtmP’
2     '3AtmP'   '3AtmP'  PresVec[1] ConstVec[1]  ConstVec2[1]
3     '2AtmP'  '2AtmP'  PresVec[2] ConstVec[2]  ConstVec2[2]
4     '1AtmP'   '1AtmP'  PresVec[3]  ConstVec[3]  ConstVec2[3]
```

- Control Functions Defined in MELGEN
- CF_HTML file defined in MELCOR
- Multiple charts specified in table
- Chart title first field
- Subsequent fields provide names of CFs or plot variables to include on chart



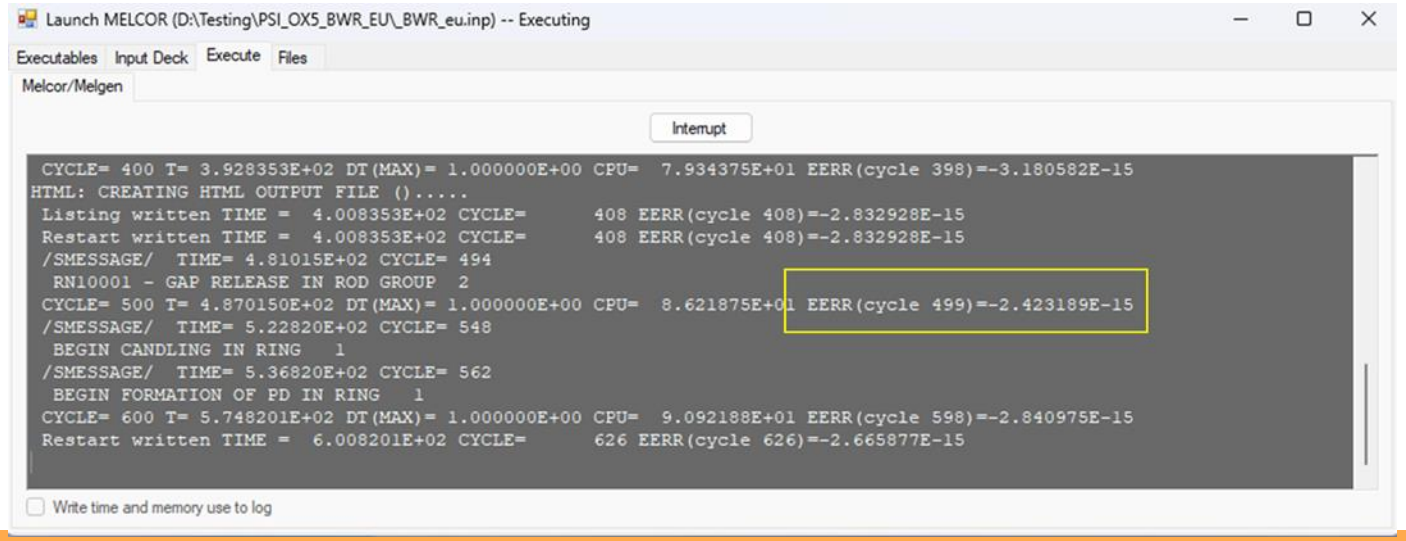
Adding a CF to the Console Output

MELCOR Program Input

User Label Data Source

- EXEC_DISPCF 'EERR' 'COR-REL-ENGY-ERR'
  - User Label: Short (<=6 character) label for data to be displayed
  - Data Source: Plot Variable or Control Function value to be displayed

If a plot variable is chosen, cycle at which variable was last evaluated is displayed




End of Advanced Control Function Topics