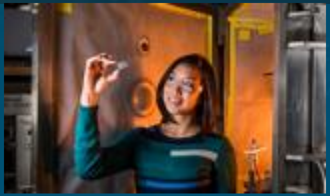
MELCOR Code Coupling



PRESENTED BY

Larry Humphries and Brad Beeny

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.

Why code coupling with MELCOR?

MELCOR is a fully-integrated, system-level computer code

- Prior to the development of MELCOR, separate effects codes within the Source Term Code Package (STCP) were run independently
  - Results were manually transferred between codes leading to a number of challenges
    - transferring data
    - ensuring consistency in data and properties
    - capturing the coupling of physics

Advantages of using a fully-integrated tool for source term analysis

- Integrated accident analysis is necessary to capture the complex coupling between a myriad of interactive phenomenon involving movement of fission products, core materials, and safety systems.
- A calculation performed with a single, integrated code as opposed to a distributed system of codes reduces errors associated with transferring data downstream from one calculational tool to the next.
- Performing an analysis with a single integrated code assures that the results are repeatable.
- Methods for performing uncertainty analysis with an integrated tool such as MELCOR are well established.
- Time step issues are internally resolved within the integral code

However, the rare need for coupling to MELCOR may still exist

- Development of new models for possible future integration into the code
- Internal requirement for using a specific code to model a particular aspect of the source term calculation.

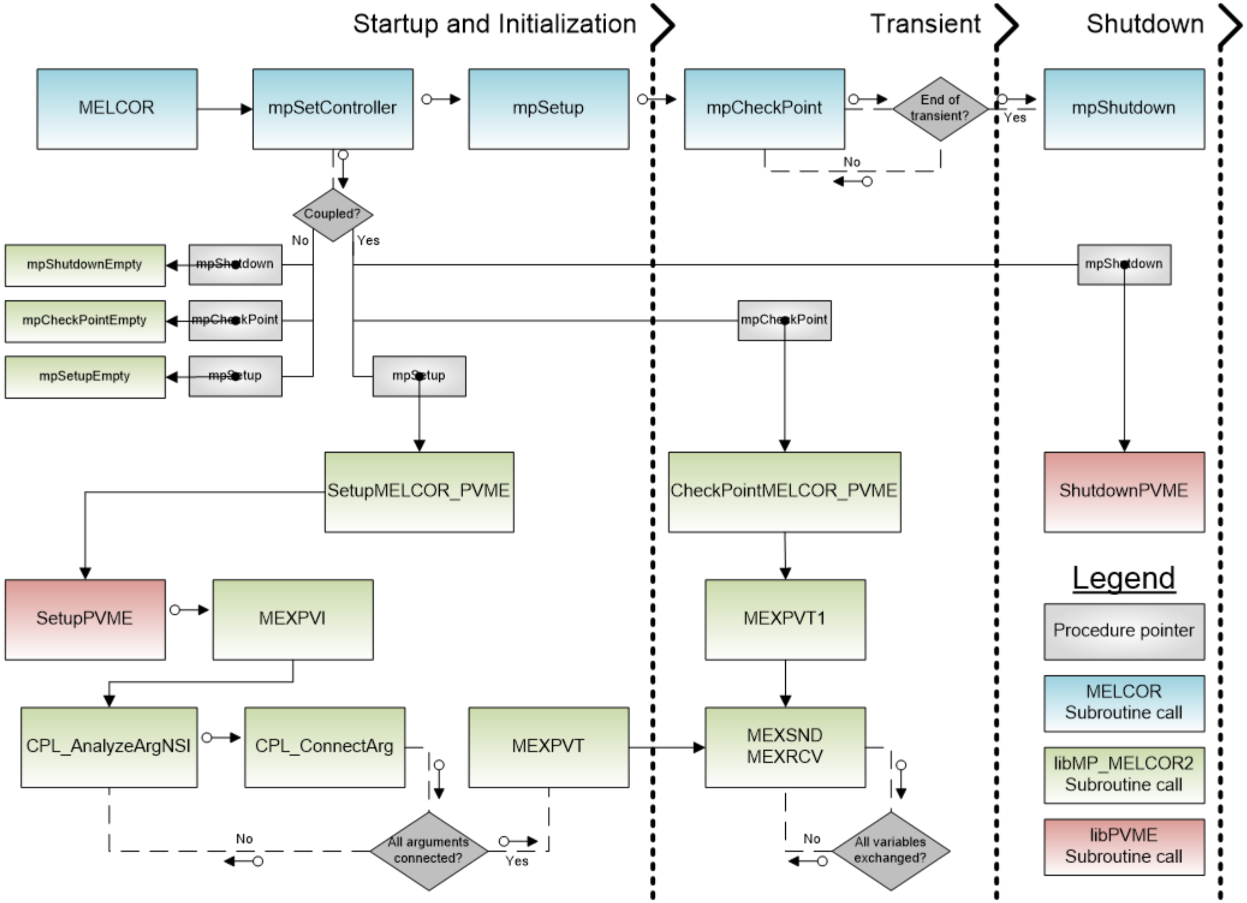
Explicit Coupling with Control Functions - PVM

PVM coupling is routinely used by at least one MELCOR licensee

- Coupling between RELAP and MELCOR v2 (containment and primary system simulated by different codes)
- Interface was updated, formalized, and documented in 2013.

PVM Coupling Requirements

- Parallel Virtual Machine (PVM) software
  - PVMEXEC Program – Developed by Idaho National Laboratory (INL).
  - PVM Library – The Parallel Virtual Machine (PVM) software library –maintained by Oak Ridge National Laboratory
- FORTRAN 2003 compliant compiler



MELCOR ‘READ’ and L-READ’ Control Functions

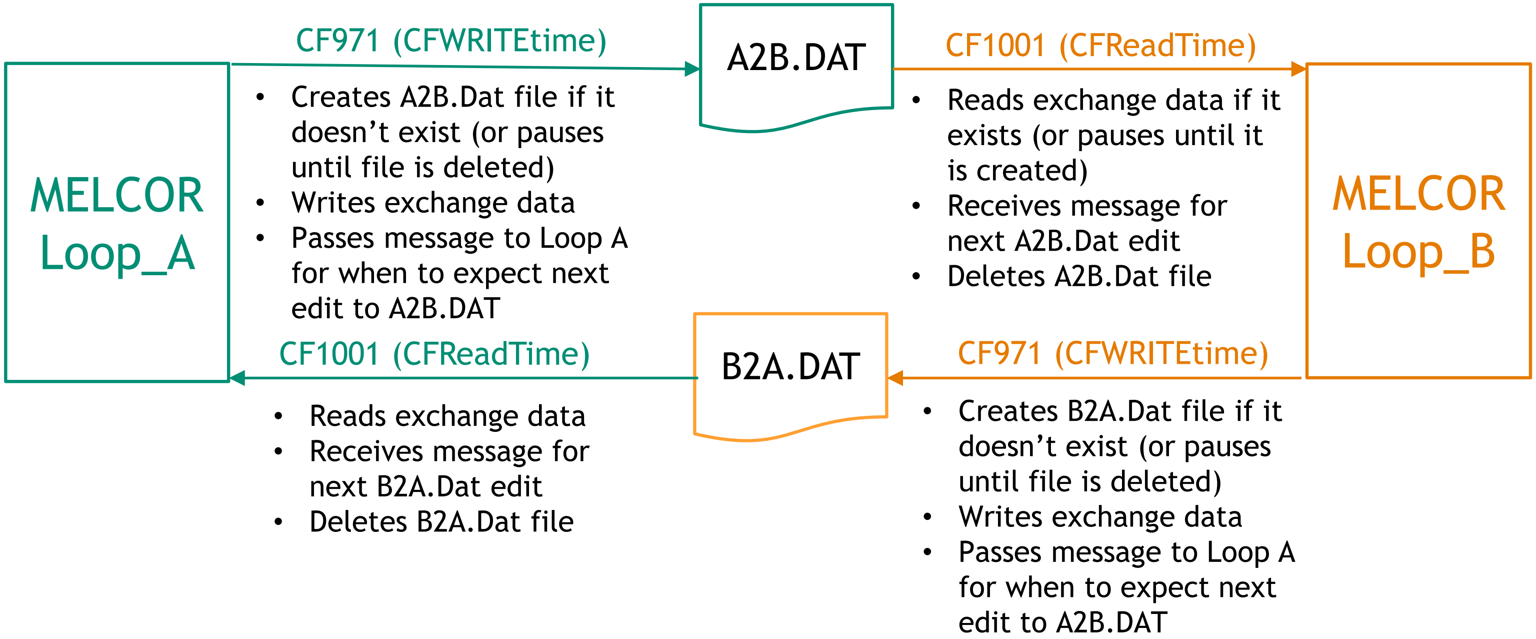
Change actual value of control function thru READ (for REAL-valued) and L-READ (for LOGICAL-valued) option during a MELCOR run

- 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



MELCOR Loop_A

CF971 (CFWRITEtime)

- Creates A2B.Dat file if it doesn’t exist (or pauses until file is deleted)
- Writes exchange data
- Passes message to Loop A for when to expect next edit to A2B.DAT

A2B.DAT

CF1001 (CFReadTime)

- Reads exchange data if it exists (or pauses until it is created)
- Receives message for next A2B.Dat edit
- Deletes A2B.Dat file

MELCOR Loop_B

CF971 (CFWRITEtime)

- Creates B2A.Dat file if it doesn’t exist (or pauses until file is deleted)
- Writes exchange data
- Passes message to Loop A for when to expect next edit to A2B.DAT

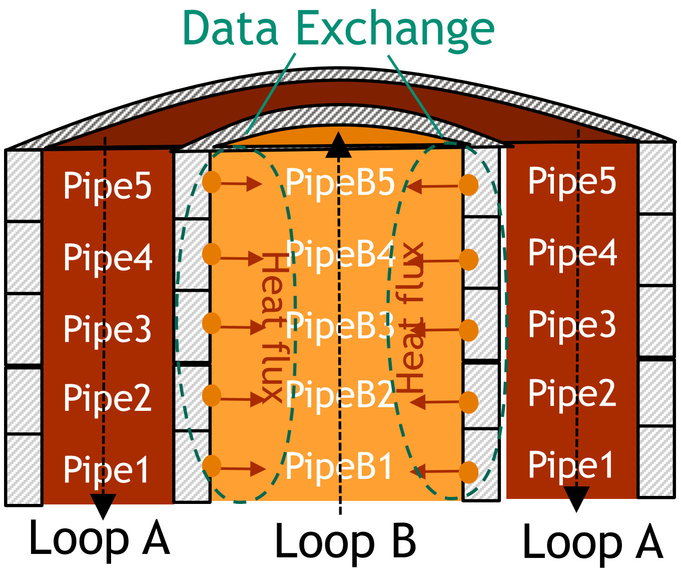
B2A.DAT

CF1001 (CFReadTime)

- Reads exchange data
- Receives message for next B2A.Dat edit
- Deletes B2A.Dat file

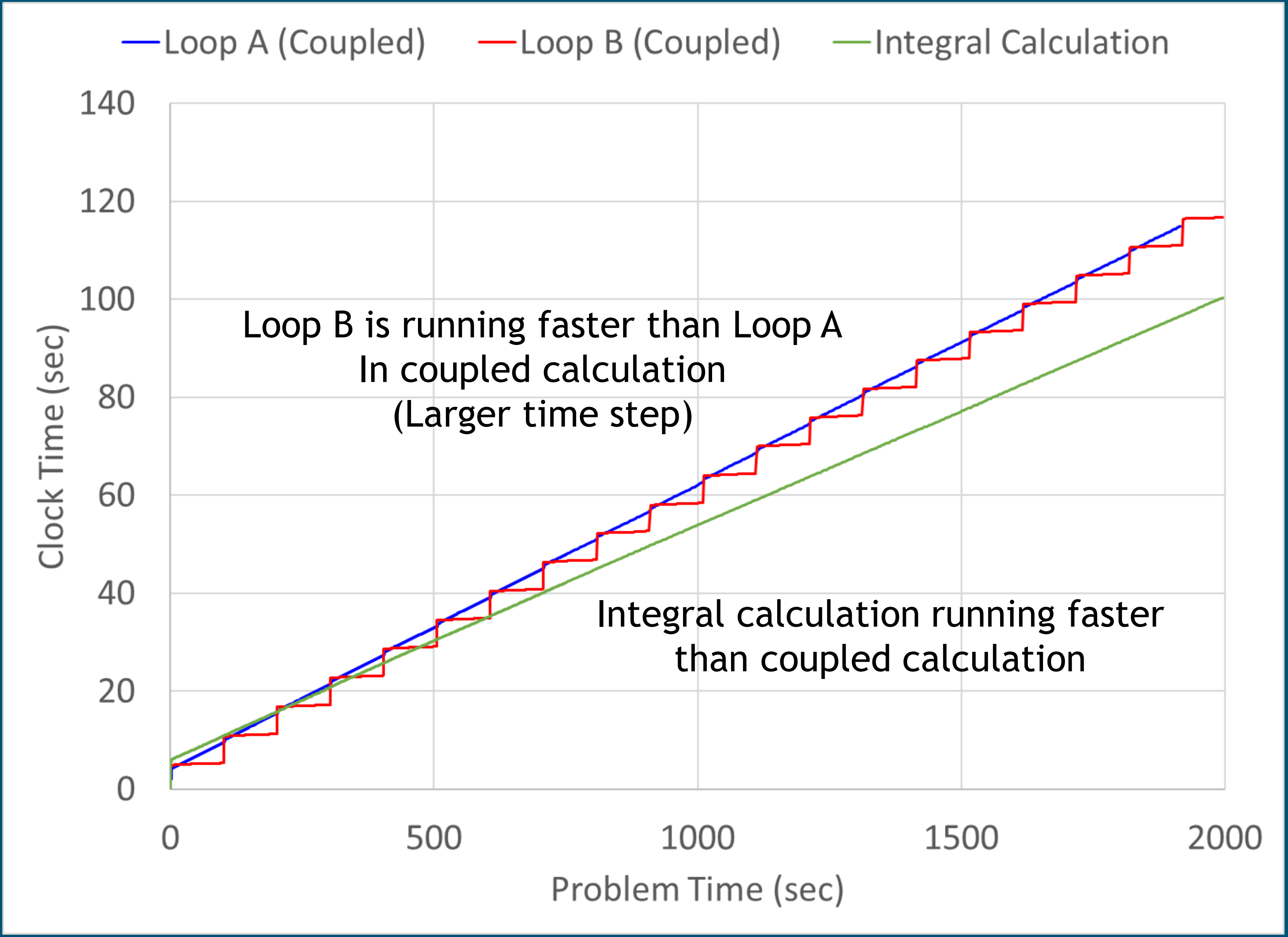
Loop_A	Loop_B
EXEC_CFEXFILE B2A.DAT … CF_ID 'CFreadTime' 1001 READ CF_ID 'CFWRITEtime' 971 WRITE CF_MSC 'CFreadTime' CF_ARG 1 ! NARG CHARG 1 CF-VALU('CFreadTime') 1.00 0.0 EXEC_CFEXFILE 'B2A.DAT' - 'CFreadTime' EXEC_CFEXWRITE '..\LOOPB\A2B.DAT'	EXEC_CFEXFILE A2B.DAT … CF_ID 'CFreadTime' 1001 READ CF_ID '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'

Simple Coupling Test Problem



	Loop A	Loop B
Flow direction	Down	Up
Output to other loop	Heat Fluxes	Temperature
Phase Inlet	Atmosphere	Pool
Heat Direction	Heat Out	Heat In
Tinlet	560 K	300+20 *sin(t*2*p/50))

Timing of coupled calculation

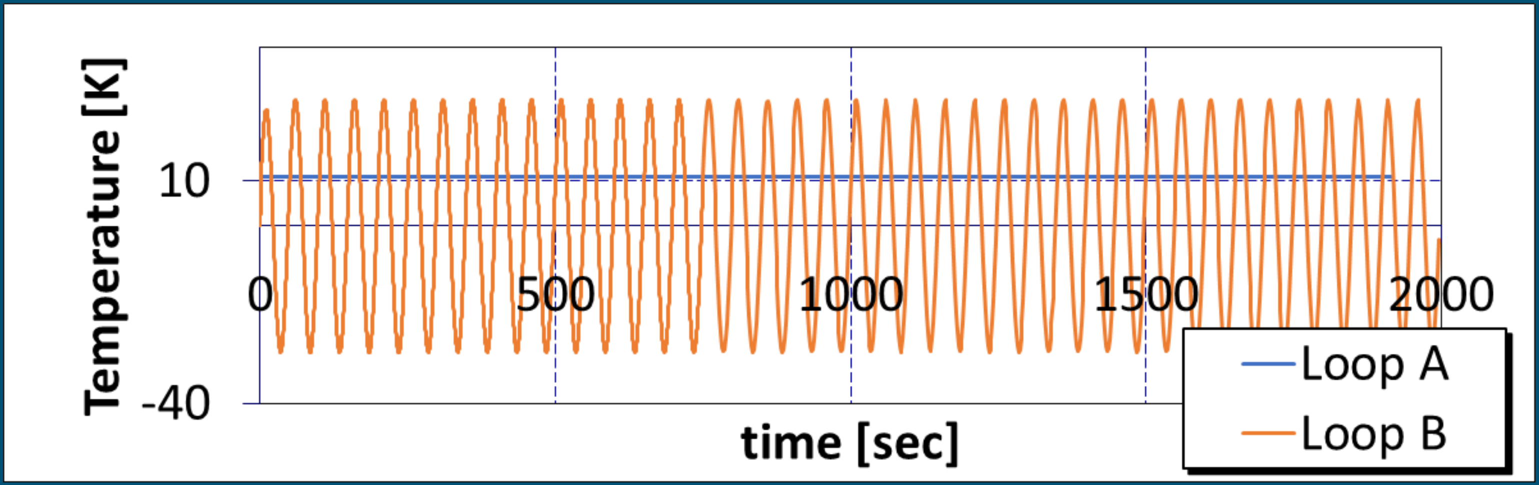


Loop B is running faster than Loop A
In coupled calculation
(Larger time step)

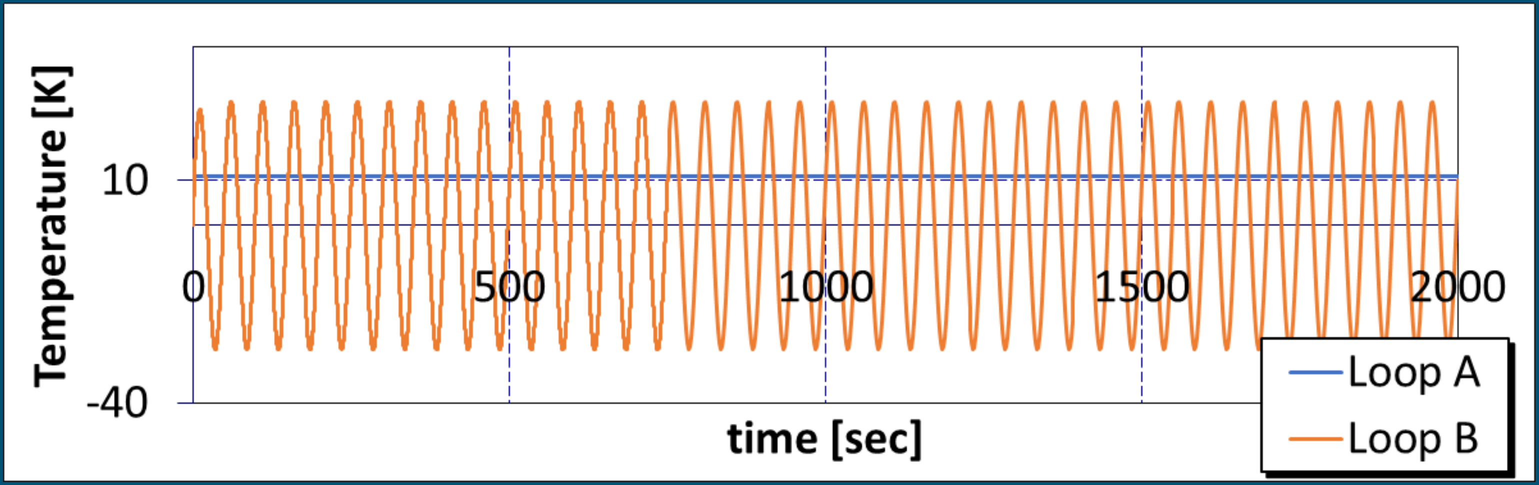
Integral calculation running faster
than coupled calculation

ΔT Inlet Temperature – Outlet temperature

Coupled Calculation

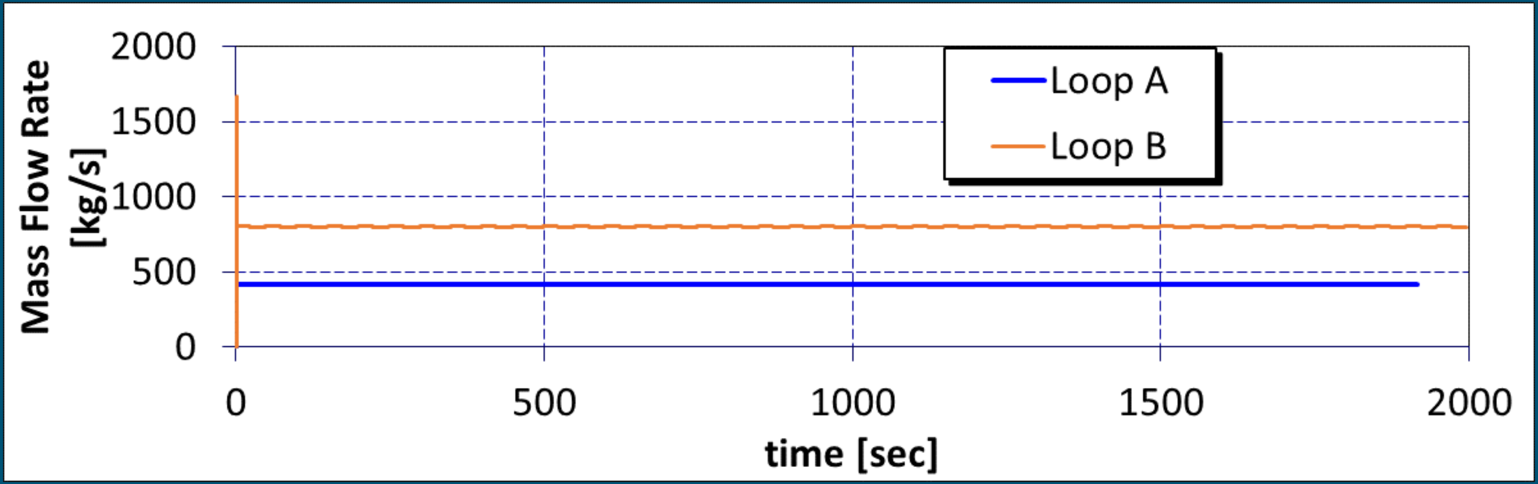


Integral Calculation

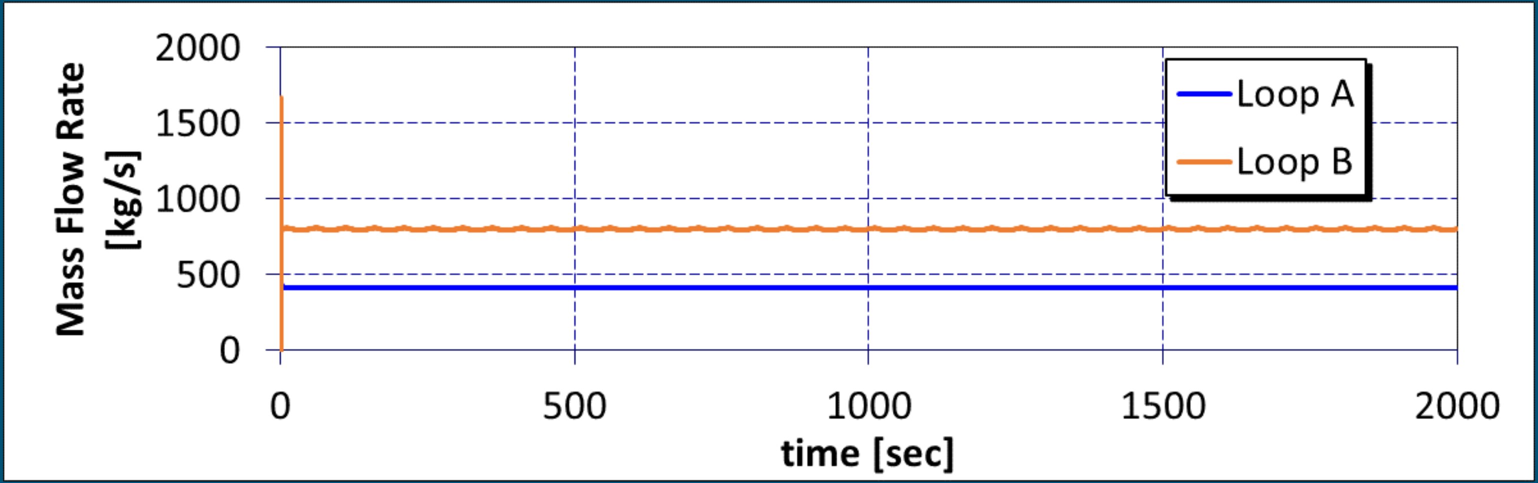


Mass Flow Loop A & Loop B

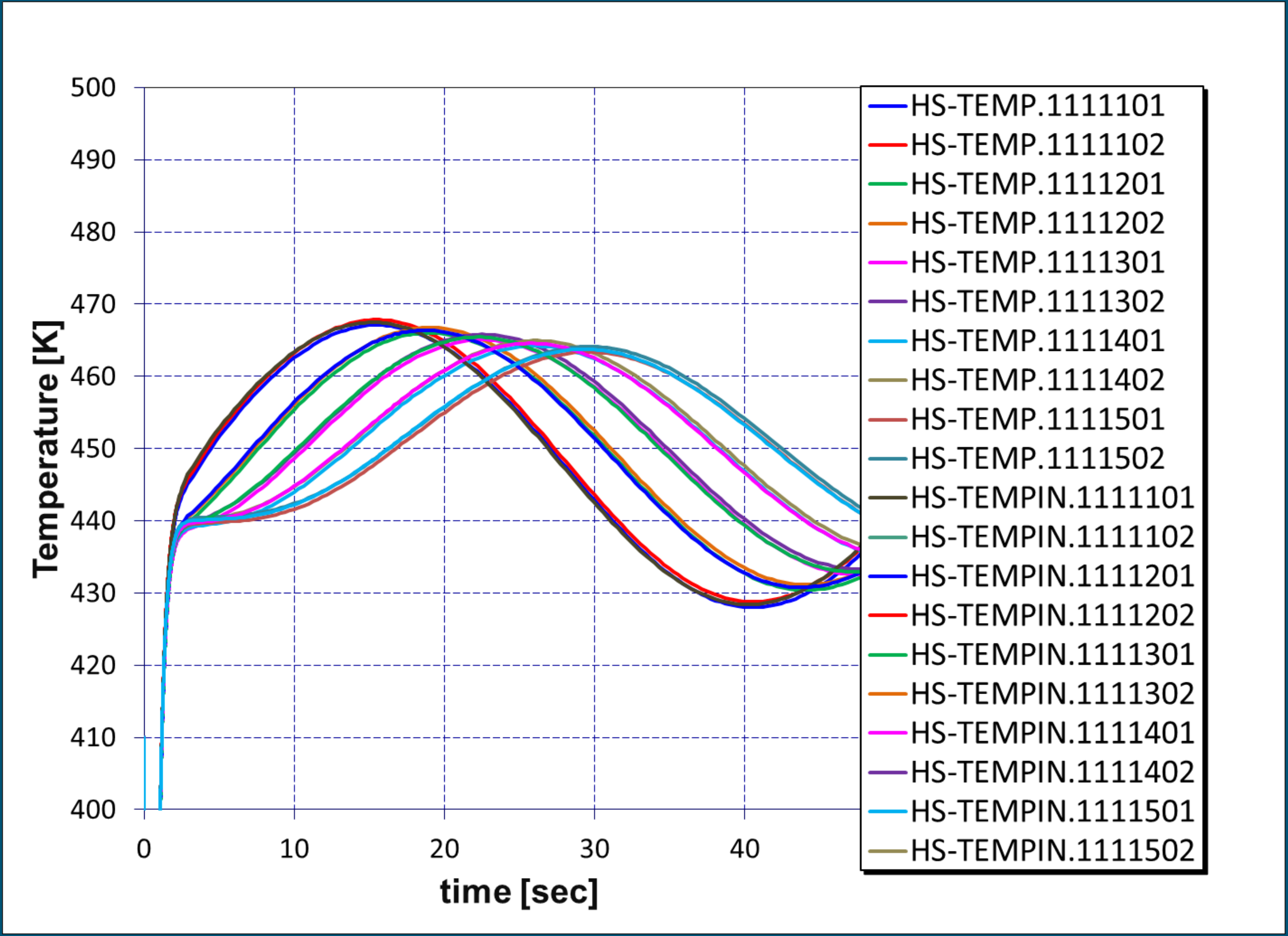
Coupled Calculation



Integral Calculation



Loop B HS Response



Data Exchange Files

B2A.DAT

```
CF_ID CFREADTIME      303.5000000000
CF_ID TOUTERS      443.8691619685      438.0188212212      435.2802719149      435.7085004724      438.7645643772
```

A2B.DAT

```
CF_ID CFREADTIME      303.5000000000
CF_ID FLUXES   -136466.4513476432   -137075.4226302063   -137642.2671272269   -138141.1221339761   -138557.0256977761
```

Driver Program Routine

```
 T=0.0; READtime=0.5
   DO While(T<= 2000.0)
     T=T+0.5

!  Run time advancement in driver code
! …
! Interface with MELCOR

      IF(T>=Readtime) THEN  !CFWRITE
          CALL CFWRITE(IERR)
      ENDIF  !CFWRITE
     IF(T>=ReadTime)THEN  !READ from File
          CALL CFREAD(IERR)
     ENDIF
   ENDDO
```

Writing Routine

```
Subroutine CFWRITE(IERR)
      integer(4) ::IERR
50     INQUIRE ( FILE=CFEXWRITE, OPENED=LOPEN, IOSTAT=ISTAT, EXIST=LEXIST )
       IF(LEXIST) GOTO 50  !Potential for infinite loop as written
       OPEN (unitWRITE,FILE=CFEXWRITE,STATUS='NEW',FORM='FORMATTED',IOSTAT=ie)
       WRITE (unitWrite,'("CF_ID "A, X,100(X,F20.10))')  'CFREADTIME', T+1.0
       WRITE (unitWrite,'("CF_ID "A, X,100(X,F20.10))')  'MASSIN', MASSIN
       CLOSE(UnitWrite)
END Subroutine CFWRITE
```

Reading Routine

```
   Subroutine CFREAD(IERR)
   integer(4) ::IERR
20  INQUIRE (FILE=CFEXFILE, IOSTAT=ie, EXIST=LEXIST )
       IF(T>ReadTime.and. ie/=0)then
         Goto 20
       ENDIF


 IF(LEXIST.and. (T>=READtime .or. OldReadTime==-999999.0) ) then
OPEN(unitREAD,FILE = CFEXFILE,STATUS='OLD',FORM='FORMATTED',IOSTAT=ie)
 !Read/parse Records in data exchange file
1         READ (unitREAD,'(A)',ERR=9999,END=9999) RECORD
        IF(RECORD == '') GOTO 1
        call exec_analyzecard (RECORD,NUMFLD)
        READ_CFNAME = characters(2)
        IF(trim(ucase(READ_CFNAME))=='CFREADTIME') then
             OldReadTime=ReadTime
             ReadTime=REALS(3)
           ENDIF
!          Parse other variables here
        GOTO 1 !go back and read next line
        ENDIF
        RETURN
9999  IERR=200
        CLOSE (unitREAD,STATUS='DELETE',IOSTAT=ie)
!If  the time read from the com file < the expected read time, revert
      If(readTime<OldReadtime)then
        ierr=200
        readTime=OldReadTime
      endif
      return
END Subroutine CFREAD
```