

# Flowchart for Creating PWR Cooldown P-T Curves

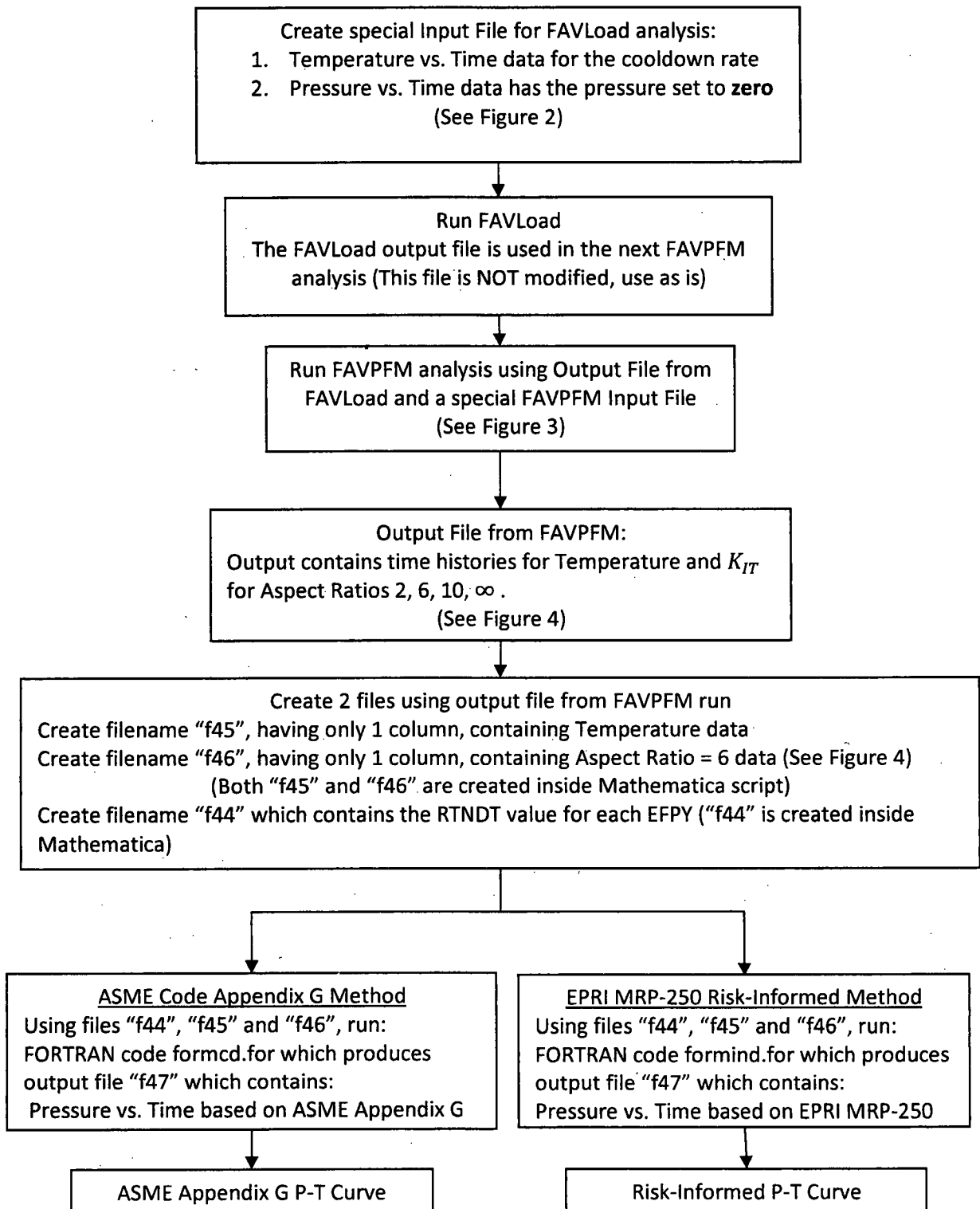


Figure 1: General Cooldown Flowchart

B/27

```

*****
*   Palisades Cooldown of 50 F / hr; 32 EFPY, RTNDT(t/4) = 208.4
*****
*-----*
*   IRAD = INTERNAL RADIUS OF PRESSURE VESSEL [IN] *
*   W    = THICKNESS OF PRESSURE VESSEL WALL (INCLUDING CLADDING) [IN] *
*   CLTH = CLADDING THICKNESS [IN] *
*-----*
GEOM  IRAD=86.0    W=8.75    CLTH=0.25 ← Palisades RPV geometry
.
.
.
*****
*   T    = BASE AND CLADDING STRESS-FREE TEMPERATURE [F] *
*   CFP  = crack-face pressure loading flag
*   CFP = 0 ==> no crack-face pressure loading
*   CFP = 1 ==> crack-face pressure loading applied
*****
SFRE  T=532.0    CFP=0    ← Set stress-free temperature (matches temperature at time zero)
.
.
.
*   SET FLAGS FOR RESIDUAL STRESSES IN WELDS
*-----*
*   NRAX = 0      AXIAL      WELD RESIDUAL STRESSES OFF
*   NRAX = 101    AXIAL      WELD RESIDUAL STRESSES ON
*   NRCR = 0      CIRCUMFERENTIAL WELD RESIDUAL STRESSES OFF
*   NRCR = 101    CIRCUMFERENTIAL WELD RESIDUAL STRESSES ON
*-----*
RESA  NRAX=0
RESC  NRCR=0    ←No residual stresses
.
. (Specify only 1 transient below)
.
*****
*   THERMAL TRANSIENT: COOLANT TEMPERATURE TIME HISTORY
*   NT = NUMBER OF (TIME,TEMPERATURE) DATA PAIRS
*****
NTTH  NT=3
*   =====
*   TIME[MIN]  T(t) [F]
*   =====
*   0.00      532.00
*   554.40    70.00
*   600.00    70.00 } Define temperature-time history
*****
*   PRESSURE TRANSIENT: PRESSURE vs TIME HISTORY
*   NP = NUMBER OF (TIME,PRESSURE) DATA PAIRS
*****
NPTH  NP=2
*   =====
*   TIME[MIN]  P(t) [ksi]
*   =====
*   0.000    0.0
*   600.0    0.0 } Zero pressure-time history

```

Figure 2: FAVLoad Input File

```

*                               EXAMPLE INPUT DATASET FOR FAVPFM
*-----
*  NSIM    = NUMBER OF RPV SIMULATIONS
*  IGATR    = NUMBER OF INITIATION-GROWTH-ARREST (IGA) TRIALS PER FLAW
*  WPS_OPTION = 0 DO NOT INCLUDE WARM-PRESTRESSING IN ANALYSIS
*  WPS_OPTION = 1          INCLUDE WARM-PRESTRESSING IN ANALYSIS
*  PC3_OPTION = 0 DO NOT PERFORM FRACTURE ANALYSIS OF CATEGORY 3
*              FLAWS IN PLATES
*  PC3_OPTION = 1 PERFORM FRACTURE ANALYSIS OF CATEGORY 3 FLAWS IN
*              PLATES
*  CHILD_OPTION = 0 DO NOT INCLUDE CHILD SUBREGION REPORTS
*  CHILD_OPTION = 1          INCLUDE CHILD SUBREGION REPORTS
*  RESTART_OPTION = 0 THIS IS NOT A RESTART CASE
*  RESTART_OPTION = 1 THIS IS A RESTART CASE
*  IRTNDT    = FOR ESTIMATING RADIATION-INDUCED SHIFT IN RTNDT
*  TC        = INITIAL RPV COOLANT TEMPERATURE
*              (applicable only when IRTNDT=993) [F]
*  EFPY      = EFFECTIVE FULL-POWER YEARS OF OPERATION [YEARS] *
*-----
CNT2 IRTNDT=2006 TC=556 EFPY=32 IDT_OPTION=1 IDT_INI=0 ILONG_OUT=0
*****
* THE LDQA RECORD PROVIDES THE OPPORTUNITY TO CHECK LOAD-RELATED
* SOLUTIONS SUCH AS TEMPERATURE, STRESSES, AND KI.
*   IQA    = 0 ==> THIS EXECUTION IS NOT FOR LOAD QA
*   IQA    = 1 ==> THIS EXECUTION IS FOR LOAD QA
*   IOPT   = 1 ==> GENERATE TIME HISTORY AT SPECIFIC THROUGH WALL
*               LOCATION
*   IOPT   = 2 ==> GENERATE THROUGH WALL DISTRIBUTION AT SPECIFIC TIME
*   IFLOR  = 1 ==> FLAW ORIENTATION IS AXIAL
*   IFLOR  = 2 ==> FLAW ORIENTATION IS CIRCUMFERENTIAL
*   IWELD  = 0 ==> DOES NOT INCLUDE THRU-WALL WELD RESIDUAL STRESS
*   IWELD  = 1 ==> DOES INCLUDE THRU-WALL WELD RESIDUAL STRESS
*   IKIND  = 1 ==> INNER-SURFACE BREAKING FLAW
*   IKIND  = 2 ==> EMBEDDED FLAW
*   IKIND  = 3 ==> OUTER-SURFACE BREAKING FLAW
*   XIN IS ONLY USED IF IKIND=2 (EMBEDDED FLAWS)
*   XIN    = IF IOPT=1; LOCATION OF INNER CRACK TIP FROM INNER SURF.[IN]
*   XIN    = IF IOPT=2; FLAW DEPTH [IN]
*   XVAR:   IF IOPT=1; XVAR=FLAW DEPTH[IN]
*           IF IOPT=2; XVAR=TIME [MIN]
*   ASPECT = ASPECT RATIO; FOR SURFACE BREAKING FLAWS: 2,6,10,999(infin)
*           FOR EMBEDDED FLAWS: ANY VALUE > 0
*-----
LDQA IQA=1 IOPT=1 IFLOR=1 IWELD=0 IKIND=1 XIN=0.65 XVAR=2.1875 ASPECT=6

(End of file)                                For cooldown, XVAR = ¼ × thickness (internal flaw)
                                              (Value is for Palisades)

```

Figure 3: FAVPFM Input File

\*\*\*\*\*  
 TIME HISTORY RESULTS FOR INTERNAL SURFACE BREAKING FLAW  
 2.188 inches IN DEPTH FROM INNER SURFACE  
 INTERNAL-SURFACE BREAKING FLAW  
 AXIAL FLAW WITHOUT RESIDUAL STRESSES  
 \*\*\*\*\*

TRANSIENT NUMBER 1

NSTEP	TIME	HOOP		KX-> X = ASPECT RATIO				
		TEMP	PRESS	STRESS	K2	K6	K10	KINF
1	0.00	532.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1.00	531.98	0.00	-0.01	0.02	0.08	0.09	0.14
3	2.00	531.87	0.00	0.00	0.09	0.24	0.28	0.41
4	3.00	531.67	0.00	0.02	0.19	0.45	0.51	0.74
5	4.00	531.41	0.00	0.06	0.30	0.68	0.77	1.10
6	5.00	531.10	0.00	0.10	0.43	0.92	1.05	1.48
7	6.00	530.75	0.00	0.14	0.56	1.18	1.33	1.86
8	7.00	530.37	0.00	0.19	0.70	1.43	1.62	2.24
9	8.00	529.97	0.00	0.25	0.84	1.68	1.90	2.63
10	9.00	529.54	0.00	0.30	0.97	1.94	2.19	3.01
11	10.00	529.09	0.00	0.35	1.11	2.19	2.47	3.39
12	11.00	528.63	0.00	0.41	1.25	2.44	2.75	3.76
13	12.00	528.15	0.00	0.46	1.38	2.68	3.02	4.12

Copy this column to file "f45"  
 (Performed internally by  
 Mathematica script)

Copy this column to file "f46"  
 (Performed internally by  
 Mathematica script)

(Output continues to time = 600 as defined in FAVLoad Input File)

Figure 4: FAVPFM Output File

```

implicit real*8(a-h,o-z)
dimension temp(1000),akic(1000),akit(1000)

open (unit=44,file='f44') ← Input file generated inside Mathematica script
open (unit=45,file='f45') } ← Input files (described in Figure 4)
open (unit=46,file='f46') }
open (unit=47,file='f47') ← Output file containing ASME Appendix G Pressure vs Time
                           Curve

read (44,*) ntimes
read (44,*) rtndt

do 20 itstep = 1,ntimes
  read (45,*) temp(itstep)
20 continue

do 30 itstep = 1,ntimes
  read (46,*) akit(itstep)
30 continue

beta   = 0.0 } ← ASME Appendix G code values
alpha  = 2.0 }
psteady = 2.101

rtndt   = rtndt + beta

thick   = 8.75 } RPV geometry (Palisades)
ri       = 86.0 }
iflor   = 1
itran   = 1

  if (itran.eq.1) then
C*****
C itran = 1 ==> cooldown; iflor = 1 ==> axial; iflor = 2 ==> circum
C*****
    if (iflor.eq.1) then
      amm = 0.926 * sqrt(thick)
    else
      amm = 0.443 * sqrt(thick)
    endif
  else
C*****
C itran = 2 ==> heatup
C*****
    if (iflor.eq.1) then
      amm = 0.893 * sqrt(thick)
    else
      amm = 0.443 * sqrt(thick)
    endif
  endif

  do 10 itstep = 1,ntimes
    astep = itstep - 1
    akic(itstep) = 33.2 + 20.734 * exp(0.02 * (temp(itstep)-rtndt))
    pall = (akic(itstep) - akit(itstep)) * (thick / ri)
    *      * (1 / alpha) * (1 / amm)
    if (pall.ge.psteady) pall = psteady
    akim = amm * (pall * ri / thick)
    aktot = akim + akit(itstep)
    write (47,101) astep,pall
101  format (2f10.2)
10  continue

```

Figure 5: Formcd.for FORTRAN Code, for ASME Appendix G (Cooldown)

```

implicit real*8(a-h,o-z)
dimension temp(1000),akic(1000),akit(1000)

open (unit=44,file='f44')
open (unit=45,file='f45')
open (unit=46,file='f46')
open (unit=47,file='f47') } ← Same file descriptions as in Figure 5
                           ← Output file containing MRP-250 Pressure vs Time
                           Curve

read(44,*) ntimes
read(44,*) rtndt

do 20 itstep = 1,ntimes
  read (45,*) temp(itstep)
20 continue

do 30 itstep = 1,ntimes
  read (46,*) akit(itstep)
30 continue

beta   = 110.0
alpha  = 1.0 } ← MRP-250 values
psteady = 2.101

rtndt   = rtndt + beta

thick   = 8.750
ri       = 86.0
iflor    = 1
itran    = 1

if (itran.eq.1) then
C*****
C itran = 1 ==> cooldown; iflor = 1 ==> axial; iflor = 2 ==> circum
C*****
  if (iflor.eq.1) then
    amm = 0.926 * sqrt(thick)
  else
    amm = 0.443 * sqrt(thick)
  endif
else
C*****
C itran = 2 ==> heatup
C*****
  if (iflor.eq.1) then
    amm = 0.893 * sqrt (thick)
  else
    amm = 0.443 * sqrt (thick)
  endif
endif

do 10 itstep =1,ntimes
  astep = itstep - 1
  akic(itstep) = 33.2 + 20.734 * exp(0.02 * (temp(itstep)-rtndt))
  pall = (akic(itstep) - akit(itstep)) * (thick / ri)
  *      * (1 / alpha) * (1 / amm)
  if (pall.ge.psteady) pall = psteady
  akim = amm * (pall * ri / thick)
  aktot = akim + akit(itstep)
  write (47,101) astep,pall
101  format (2f10.2)
10  continue

```

**Figure 6: Formind.for FORTRAN Code, for EPRI MRP-250 (Cooldown)**

## Mathematica Script

```
SetDirectory["D:/FAVOR Background/FAVOR-ORNL/New FAVOR QA Project/Terry' Files - Working  
Directory/QA_SwRI_files_tables/PWRs/Palisades/Osvaldo Script/Test Dir"];
```

```
ClearSystemCache[];
```

```
(* Perform independent P-T curve generation *)
```

```
Directory[];
```

```
(* Copy FAVLoad and FAVPFM executables to current directory *)
```

```
CopyFile[Directory[] <> "/Executables/favloadm.exe", Directory[] <> "/favloadm.exe"];
```

```
CopyFile[Directory[] <> "/Executables/pfm510m.exe", Directory[] <> "/pfm510m.exe"];
```

```
(* Copy formcd and formind executables to current directory *)
```

```
CopyFile[
```

```
Directory[] <> "/P-T Curve Input Data/Cooldown-Palisades/FORTRAN/formcd-palisades.exe",
```

```
Directory[] <> "/formcd.exe"];
```

```
CopyFile[Directory[] <>
```

```
" /P-T Curve Input Data/Cooldown-Palisades/FORTRAN/formind-palisades.exe",
```

```
Directory[] <> "/formind.exe"];
```

```
(* User defined input *)
```

```
PWRname = "Palisades";
```

```
(* add to list all EFPY cases to be analyzed *)
```

```
EFPYlist = {"EFPY32", "EFPY60", "EFPY200", "EFPY500"};
```

```
(* for each EFPY, specify the names of each input file *)
```

```
PTlistEFPY32 = {"lp1316.in", "lp1720.in", "lp2124.in"};
```

```
PTlistEFPY60 = {"lp3740.in", "lp4144.in", "lp4548.in"};
```

```
PTlistEFPY200 = {"lp6164.in", "lp6568.in", "lp6971.in"};
```

```
PTlistEFPY500 = {"lp8688.in", "lp8992.in", "lp9396.in"};
```

```
EFPY = "3 3 3 3"; (* Number of P-T curves for each EFPY *)
```

```
RTNDT = "208.4 232.8 296.3 374.6"; (* Values of RTNDT for each EFPY *)
```

```
(* End user defined input *)
```

```
(* Calculations start here *)
```

This input is for  
the Palisades  
PWR cooldown  
data provided by  
ORNL.

User must change  
input for other  
PWR cooldown  
data.

## Mathematica Script - Continued

```
(* NOTE:
you need to modify these 2 strings to include all EFPY cases you specified above *)
list = Join[PTlistEFPY32, PTlistEFPY60, PTlistEFPY200, PTlistEFPY500];
PTlimit = Length[PTlistEFPY32] +
          Length[PTlistEFPY60] + Length[PTlistEFPY200] + Length[PTlistEFPY500];
```

```
(*Print[{"list ", list}];
Print[{"PTlimit ", PTlimit}];
Print[{"efpylist ", Length[EFPYlist]}];*)
```

```
(* Internal bookkeeping array *)
```

```
Clear[A];
Array[A, {Length[EFPYlist], 4}];
total = 0;
str = StringToStream[EFPY];
str2 = StringToStream[EFPY];
str3 = StringToStream[RTNDT];
k = 1;
Do[
  total = total + Read[str, Number]; (*Print[{"total ", total}];*)
  A[j1, 1] = k; A[j1, 2] = total;
  A[j1, 3] = Read[str2, Number]; A[j1, 4] = Read[str3, Number];
  k = k + A[j1, 3];
  (*Print[{A[j1, 1], A[j1, 2], A[j1, 3], A[j1, 4]}];*)
  {j1, 1, Length[EFPYlist]};
```

This input is based on data entered above  
for the Palisades PWR.

User must modify input for other PWR  
cooldown data.

```
(* Top of Loops *)
(* 1st loop is total number of EFPYs *)
(* 2nd loop is total number of P-T curves for each EFPY *)
```

Beginning of data  
loops for this PWR

```
Do[
  Print[{"EFPY number = ", jefpy}];
  Print[Style[EFPYlist[[jefpy]], FontFamily -> "Arial",
    FontSize -> 18, FontWeight -> "Bold", FontColor -> Hue[0]]];

  k = A[jefpy, 1];
  Do[
    Print[{"Efp curve number ", efpynum}];
    Print[{"Efp file ", list[[k]]}];

    (*Print[{"list files ", FileNames[]}];*)
```



## Mathematica Script - Continued

(\* Copy special FAVLoad and FAVPFM files to current directory

See Figs. 2 and 3  
for format

```
CopyFile[Directory[] <> "/P-T Curve Input Data/Cooldown-Palisades/FAVLoad/" <>
  EFPYlist[[jefpy]] <> "/" <> "PT-" <> list[[k]], Directory[] <> "/favloadrun.in"];
CopyFile[Directory[] <> "/P-T Curve Input Data/Cooldown-Palisades/FAVPFM/" <>
  EFPYlist[[jefpy]] <> "/pfm510mrun-" <> EFPYlist[[jefpy]] <>
  ".in", Directory[] <> "/pfm510mrun.in"];
```

```
(*Print[{"list files ", FileNames[]}];*)
```

(\* Execute FAVLoad \*)

```
If[FileExistsQ["favloadrun.in"],
  Run["favloadm.exe"], Print["File favloadrun.in is missing"] && Abort[]];
```

```
DeleteFile["favloadrun.echo"];
```

```
(*Print[{"list files ", FileNames[]}];*)
```

(\* Execute FAVPFM \*)

```
If[FileExistsQ["pfm510mrun.in"], If[FileExistsQ["favloadrun.out"],
  Run["pfm510m.exe"], Print["favloadrun.out missing"] && Abort[]],
  Print["pfm510mrun.in missing"] && Abort[]];
```

```
DeleteFile[{"ARREST.OUT", "FAILURE.DAT", "FLAWNO.OUT",
  "FLAWSIZE.OUT", "INITIATE.DAT", "RTNDT.OUT", "SCATTER.OUT", "TRACE.OUT",
  "pfm510mrun.echo", "favloadrun.out", "favloadrun.in", "pfm510mrun.in"}];
```

(\* scan FAVPFM output for data to create files f45 and f46 \*)

See figure 4 for  
FAVPFM outout

```
fileID = OpenRead["pfm510mrun.out"];
test = "0"; mat = {};
str = Read[fileID, Record, RecordSeparators -> {"\n"}];
While[test != "NSTEP" && str != EndOfFile,
  str = Read[fileID, Record, RecordSeparators -> {"\n"}];
  If[str != " ", test = StringTake[str, 5] (*; Print[test];*)];

piv = Read[fileID, Table[Word, {9}]];
AppendTo[mat, piv];
While[piv != EndOfFile,
  piv = Read[fileID, Table[Word, {9}]];
  If[piv != EndOfFile, AppendTo[mat, piv]]];
```

## Mathematica Script - Continued

```
Close[fileID];
mat = Drop[mat, -1];

DeleteFile[{"pfm510mrun.out"}];

(* Scan ORNL files for total number of time steps in their P-T curve data *)
```

```
fileID = OpenRead[
  Directory[] <> "/Reference P-T Curve Files/PAVLOAD_Cooldown_Input/" <> list[[k]];
mylim = Find[fileID, {"NP="}];
mylin2 = StringToStream[mylim];
SetStreamPosition[mylin2, 9];
npsave = Read[mylin2, Number];

fortout = {};
AppendTo[fortout, npsave];
Close[fileID];

AppendTo[fortout, A[jefpy, 4]];
Export["f44", fortout, "Text"];
Print[fortout];
CopyFile["f44", StringDrop[list[[k]], -3] <> ".f44data"];
```

Creates file "f44" used in  
formcd and formind FORTRAN  
codes (Figs. 5 and 6)

```
(* Run formcd and formind codes to produce SwRI P-T curves for comparison to ORNL *)
```

```
matT = Transpose[mat];
Export["f45", matT[[3]], "Table"];
Export["f46", matT[[7]], "Table"];
Run["formcd.exe"];
swriasmadat1 = Import["f47", "Table"];
Run["formind.exe"];
swriridat2 = Import["f47", "Table"];
```

Running formcd and formind  
FORTRAN codes (Figs. 5 and 6) to  
produce ASME and MRP-250 P-T  
curves

```
DeleteFile[{"f44", "f45", "f46", "f47"}];
```

```
(* Finish independent P-T curve generation *)
```

```
(* Extract P-T curve data from original ORNL files *)
```

```
(* Scan current file for first "NP=", Transient #1 (ASME) *)
```

Extract P-T curves from  
original ORNL files

```
fileID = OpenRead[
  Directory[] <> "/Reference P-T Curve Files/PAVLOAD_Cooldown_Input/" <> list[[k]];
mylim = Find[fileID, {"NP="}];
```

## Mathematica Script - Continued

```

mylin2 = StringToStream[mylin];
SetStreamPosition[mylin2, 9];
Hold[Read[mylin2, Number] + 3];

ornlmat1 = {};
For[j = 1, j < 4, j++, Read[fileID, Record, RecordSeparators -> {"\n"}]];
Do[
  piv = Read[fileID, Table[Number, {2}]];
  AppendTo[ornlmat1, piv],
  {1, 1, Read[mylin2, Number]};

(* Continue scanning current file for second "NP=", Transient m2 (Risk-Informed) *)

mylin = Find[fileID, {"NP="}];
mylin2 = StringToStream[mylin];
SetStreamPosition[mylin2, 9];
Hold[Read[mylin2, Number] + 3];

ornlmat2 = {};
For[j = 1, j < 4, j++, Read[fileID, Record, RecordSeparators -> {"\n"}]];
Do[
  piv = Read[fileID, Table[Number, {2}]];
  AppendTo[ornlmat2, piv],
  {1, 1, Read[mylin2, Number]};

Close[fileID];

(* Create an Excel file containing SwRI P-
T curve and ORNL P-T curve data to individual worksheets *)
Export["P-T-Curves.xls", {"SwRI-ASME" -> swriasmat1,
  "SwRI-RI" -> swriridat2, "ORNL-ASME" -> ornlmat1, "ORNL-RI" -> ornlmat2}];

(* Create archive directory *)
suffix = "/Comparison-Curves/" <> PWRname <> "/" <> EFPYlist[[jefpy]];
archivedir = Directory[] <> suffix;
DirectoryQ[archivedir];
If[jefpynum == 1, CreateDirectory[archivedir]];

If[DirectoryQ[archivedir],
  CopyFile["P-T-Curves.xls", archivedir <> "/" <> StringDrop[list[[k]], -3] <>

```

Create Excel file with  
SwRI and ORNL P-T  
curve data

Create directory to store all P-T comparison data output

## Mathematica Script - Continued

```
"-P-T-Curves.xls"] && CopyFile[StringDrop[list[[k]], -3] <> ".f44data",
archivedir <> "/" <> StringDrop[list[[k]], -3] <> ".f44data"]];
```

```
DeleteFile[{"P-T-Curves.xls", StringDrop[list[[k]], -3] <> ".f44data"}];
```

```
(* Create plots of each P-T curve for SwRI and
ORNL which are stored as JPEGs and copied to archive directory *)
```

Also create  
JPEGs of SwRI  
and ORNL P-T  
curve plots

```
gra1 = ListLinePlot[{swriridat2, ornlmat2}, Frame -> True,
PlotStyle -> {{AbsoluteThickness[6]}, {AbsoluteThickness[2], Hue[0]}},
FrameLabel -> {"Time [min]", "Pressure [PSI]", "RI", ""},
PlotLegend -> {Style["SwRI", FontFamily -> "Arial", FontSize -> 14],
Style["ORNL", FontFamily -> "Arial", FontSize -> 14]}, LegendPosition -> {0.3, 0.1},
BaseStyle -> {FontFamily -> "Arial", FontSize -> 12}, LegendSize -> {0.4, 0.3},
LegendShadow -> None, LegendBorder -> None, ImageSize -> 600};
Print[gra1];
Export[archivedir <> "/" <> StringDrop[list[[k]], -3] <> "_RI_P-T_Comparison.jpg",
gra1, "JPG", ImageResolution -> 200];
```

```
gra2 = ListLinePlot[{swriasmedat1, ornlmat1}, Frame -> True,
PlotStyle -> {AbsoluteThickness[6], {AbsoluteThickness[2], Hue[0]}},
FrameLabel -> {"Time [min]", "Pressure [PSI]", "ASME", ""},
PlotLegend -> {Style["SwRI", FontFamily -> "Arial", FontSize -> 14],
Style["ORNL", FontFamily -> "Arial", FontSize -> 14]}, LegendPosition -> {0.3, 0.1},
BaseStyle -> {FontFamily -> "Arial", FontSize -> 12}, LegendSize -> {0.4, 0.3},
LegendShadow -> None, LegendBorder -> None, ImageSize -> 600};
Print[gra2];
Export[archivedir <> "/" <> StringDrop[list[[k]], -3] <> "_ASME_P-T_Comparison.jpg",
gra2, "JPG", ImageResolution -> 200];
```

```
k = k + 1;
Print[{"Finished curve number ", efpynum, " for ", EFPYlist[[jefpy]]},
{efpynum, 1, A[jefpy, 3]}];

Print[{"Finished with EFPY ", EFPYlist[[jefpy]]},
{jefpy, 1, Length[EFPYlist]}];
```

End of data  
loops for  
this PWR

```
DeleteFile[{"favloadm.exe", "pfm510m.exe", "formcd.exe", "formind.exe"}];
```

```
Print[{"Total Time Used ", TimeUsed[]}];
```