

# **SCIENTIFIC NOTEBOOK**

## **612-18E**

by

Dennis M. LeNeveu

Southwest Research Institute  
Center for Nuclear Waste Regulatory Analyses  
San Antonio, Texas

## INITIAL ENTRIES

Scientific Notebook: #612-18E

Issued to: R. Janetzke for use by consultant Dennis LeNeveu

Issue Date: September 17, 2003

Account Number: 20.06002.01.113

Title: TPA 5.0 Code Development

Participants: R. Janetzke  
S. Mohanty  
R. Rice  
C. Scherer  
O. Pensado  
R. Benke  
P. LaPlante  
G. Adams  
B. Winfrey  
M. Smith  
D. LeNeveu

Objective:

This scientific notebook will document the work performed in the development of the TPA code.

## ENTRIES

**Start date: Sunday, January 23, 2005, 8.00 P.M.**

**Activity:** Compile Exec.f for version 5.0.0e of the TPA code Lehey LF95 V.5.71

General comment: IMPLICIT NONE not consistently used. Defaults for integers are sometime used. As a consequence in some routines many variables are undeclared  
There are many unused variables in many routines

exec.f – call to sampleharzedcurve had a comma out of bounds – variable in call moved to next line

Condxyzt.f – warning- in call to xgauleg argument 4 does not match dummy argument

invent.f – 2 errors – two variables already declared in include files - new version made

nfenv.f- 15 errors numTime declared after it was used. A new version was made

Eight unresolved symbols:

```
exec.obj : error LNK2001: unresolved external symbol _EBSREL@76
zportpc.obj : error LNK2001: unresolved external symbol _DERF@4
zportpc.obj : error LNK2001: unresolved external symbol _NDPERR@4
zportpc.obj : error LNK2001: unresolved external symbol _NDPEXC@0
zportpc.obj : error LNK2001: unresolved external symbol _INVALOP@4
zportpc.obj : error LNK2001: unresolved external symbol _DVCHK@4
zportpc.obj : error LNK2001: unresolved external symbol _OVEFL@4
zportpc.obj : error LNK2001: unresolved external symbol _UNDFL@4
Debug/TPAExecV5.exe : fatal error LNK1120: 8 unresolved externals
Error executing link.exe.
```

TPAExecV5.exe - 9 error(s), 0 warning(s)

**End: Sunday January 23, 11.59 P.M.**

**Start: Monday, January 24, 2005, 9.10 A.M.**

**Activities:**

Call Ron Janetzke at 9:10 a.m. re project. Ron discussed a work package involving NEFTRAN  
call ended 9:45

Begin compiling nefmks.f and below at 10.00 am.

Warning: In the call to TPPRT, actual argument #2 does not match the type and kind of the corresponding dummy argument.

```
CALL TPPRT(TEND,0.0D0)
```

Note that in the subroutine tpprt the declarations are REAL rather than DOUBLE PRECISION

Successful run of nefmks.f (NEFTRAN) at 11. a.m. nefii.out produced

**End: Monday, January 24, 11 .a.m.**

**Start: Wednesday, January 26, 2005, 9.00 a.m.**

**Activities:**

Begin tracking down errors in NEFTRAN cases 1 to 3.

The following error message is obtained in nefii.out for case 1.

```
FOR ELEMENT 2, LEG 0 IS NOT FIRST LEG IN PATH, = 1
*** PROGRAM STOP FROM FLOWIN
```

Discovered that an input file compatible with older versions was sent for case 1,2 and 3. These files were missing the mass transfer coefficient and therefore the read was out of order. - 12.10 p.m.

1.00 p.m. -Reformatted old input file to be compatible with current code version. Obtained a run with array bounds error. Began tracing error with Visual Fortran Debug until 4.00 p.m.

9:30 p.m. trace error if BF array BF set in Facer used in TRNSPT  
NEF calls METHOD calls SETUP calls FACER which sets up BF  
METHOD calls TRNSPT which uses BF in line 8281

BF when used is accessed by indices in JPSC  
JPSC is initialized to 0 in FACER from 1 to MXSUB +1  
JPSC is set in line 1829 in FACER.  
End trace 11:30 p.m.

**End: Wednesday, January 26, 2005, 11:30 p.m.**

**Start: Thursday, January 27, 2005, 8:30 a.m.**

**Activites:**

Begin determination of array size of BF at 8:30 a.m.

BF initialized to zero at line 1517, loop 40 in FACER

LADD the index for BF set to zero line 1530 in FACER

LADD incremented in line 1793 in the 700 do loop of FACER

LADD reassigned in line 1817 LADD=LADD-LOFF

BF seems to be assigned values from SUM in loop 700 most of which are zero in loop 800 the first nonzero entry is found by going backwards from the end of the array to determine LOFF and BF is incremented again in loop 700 starting from the new non-zero entry

The end point for the 700 loop IS2 is set to n1-1 at line 1566  
IS2 is reset in the 800 loop as follows:

```
      DO 800 I=N1,NTX
        IF(I .GT. NCMP) THEN
C
C HAVE REACHED FIRST GRID BLOCK IN A NEW LEG. RE-INITIALIZE.
C
          IF(IP .GT. LS)CALL ADJB(VL,LEG,IP,TS,NCMP,KNTIB)
          LEG=LEG+1
          LADJ=LADJ+1
          IS2=NCMP
          NCMP=NCMP+NX(LEG)+NC(LEG)
```

IS1 the start point for the 700 loop is set in the 800 loop just before the start of the 700 loop.

End at 11:30 a.m.

Begin changing code to use allocatable arrays for BF storage at 12:30 p.m.  
End at 3:30 p.m.

10 to 10:30 p.m. respond to email from Ron Janetzke re array sizes for BF  
**End: Thursday, January 27, 2005, 11 p.m.**

**Start: Saturday, January 29, 2005 10:30 a.m.**

10:30-11:30 a.m. trace looping structure in FACER to determine number of possible iterations for BF array.

**End: Saturday, January 29, 2005, 11:30 a.m.**

**Start: Sunday, January 30, 2005, 11:30 a.m.**

11:30 a.m. Produce simplified code structure for FACER

```

DO 1000 IR=1,NOISO > NOISO IN COMMON IOSTOP
  LS=LPS(IR)
  LE=LPE(IR)
  DO 900 IP=LS,LE
    LEG=2
    LADJ=1
    IS2=N1-1
  DO 800 I=N1,NTX > NTX IN COMMON XPORTB
    IF(I .GT. NCMP) THEN
      IF(IP .GT. LS)CALL ADJB(VL,LEG,IP,TS,NCMP,KNTIB)
      LEG=LEG+1
      LADJ=LADJ+1
      IS2=NCMP
      .....
    END IF
    ...IS1=NCMP-NX(LEG)-NC(LEG)-NBAK+1
  DO 700 K=IS1,IS2
    SUM=0.
    DO 600 J=1,NVI  NVI→ IN COMMON XPORTB
      DO 590 KB=1,NSPK
        ....
        SUM=SUM+FR(KB)
        ....
      590 CONTINUE
    600 CONTINUE
    IF(SUM .EQ. 0 .AND. K .EQ. IS1+NSFB) THEN
      NSFB=NSFB+1
    ELSE
      LADD=LADD+1
      BF(LADD)=SUM
    ENDIF
  700 CONTINUE
  IF(NSFB .NE. IS2-IS1+1) THEN
    .....
750   LADD=LADD-LOFF
      JPN(KNTJ,3)=IS2-LOFF
      LDP1=LADD+1
    ENDIF
  800 CONTINUE
    ....
    DO 850 L=LDSV,LADD
      BF(L)=BF(L)*DPF
    850 CONTINUE
    LDSV=LADD+1
  900 CONTINUE
1000 CONTINUE

```

12:00 p.m. finished code analysis

Begin coding dynamic arrays 1:00 p.m.

Completed dynamic arrays in FACER. Began execution of case 1 at 2:30 .pm. Execute case 2 and case 3 to 10. p.m.

**End: Sunday, January 30, 2005, 10:00 p.m.**

**Start: Monday, January 31, 2005, 8:30 a.m.**

Begin to address overflow in JPN array from case 3. Add allocatable array for JPN.  
Debug test case 3. Start to run test case 3 10. a.m. Test case 3 finished about 12:00 p.m.

Call Ron Janetzke at 1:30 p.m. re tpa work.

Call ended at 2:30 p.m.

Call T. Andres re: response functions to replace NEFRTAN 6-7 p.m

**End: Monday, January 31, 2005, 7.00 p.m.**

**Start: Tuesday, February 01, 2005 12:30 p.m.**

Begin internet search for solvers to replace NEFTRAN

1) NAMMU- Harwell <http://www.sercoassurance.com/ea/groundwater/NMtech64.pdf> -finite element requires mesh no semi-infinite or infinite boundary conditions

2) FRAC3DVS-UofWaterloo-

<http://www.flintbox.com/technology.asp?sID=922BFDBF9A3B480FB86F69D0137332BA&tID=371FA5BA1A374A748A4F0E942FDF9938-#=3D> – chains – not semi-infinite

3) Amber-Quintessa Limited-<http://www.enviros.com/PDF/Services/ACFB0CD.pdf>

-no built in modle – compartment model simulation system – chains

4) FEM 2D –chains –finite element-

<http://www.ejge.com/1996/Ppr9602/cont2dfe.htm#CTAN/W>

4)GOLDSIM-flexible software development system –

-[http://www.goldsim.com/Downloads/Manuals/CT\\_Manual\\_Short.pdf](http://www.goldsim.com/Downloads/Manuals/CT_Manual_Short.pdf)

5) MODFLOW -[http://www.ssg-surfer.com/ssg/detailed\\_description.php?products\\_id=151](http://www.ssg-surfer.com/ssg/detailed_description.php?products_id=151)

6) ADAPTS- 1D analytical used in Canadian program

7) SWIFT- 3D –finite difference-<http://www.mpassociates.gr/software/environment/swift.html>

END: 1:30 .pm.

Begin to look for magic bullet in code – limit size of NTX in subroutine Chain to minimize iterations

End -4:30 p.m.

**Start: Wednesday, February 02, 2005, 8:30 a.m.**

Continue review of codes

8) MOC- 2D characteristics of flow model – source code available –USGS-

<http://water.usgs.gov/software/moc.html>

General review page: <http://www.ehsfreeware.com/gwqclean.htm>

9) MT3D-By US Environmental Protection Agency, Office of Research and Development, National Risk Management Research Laboratory, Center for Subsurface Modeling Support. MT3D is a 3D solute transport model for simulation of advection, dispersion, and chemical reactions of dissolved constituents in ground-water systems.

Finish internet search 9:00 a.m.

Begin documentation of method to employ AECL response function methodology-9:00 a.m.-  
finish preliminary draft 10:30 a.m.

Reply to email from T. Andres re: using AECL response functions 10:30-11.00 a.m.

Email Ron re tpa work 1:30 -2 pm

**End: Wednesday, February 02, 2005, 2:00 p.m.**

**Start: Thursday, February 03, 2005, 9:30 p.m.**

Add write statements to NEFTRAN for array sizes . Start case 2 – 10:30 p.m.

**End: Thursday, February 03, 2005, 10:30 p.m.**

**Start: Friday, February 04, 2005, 2:00 a.m.**

Add calls for CPU time and write out CPU time. Start case 3:00 – 2:30 a.m.

Record and email results to R. Janetzke 8:00-8:30 a.m. Conference call with Ron Janetzke 9:30-10:30 a.m. – decided to work on speeding up NEFTRAN. The general approach discussed was to attempt to divide large chains and multiple legs into separate problems each with its own input.

1:00-3:00 p.m. read NEFTRAN manual and examine code to determine best way to split chains and legs into separate problems.

**End: Friday, February 04, 2005, 3:00 p.m.**

**Start: Monday, February 07, 2005, 10. a.m.**

Examine output and input files for secular equilibrium or no mobility extreme cases

Examine manual to determine reason for large NTX (number of receiver blocks) to 12:00 a.m.

12:30 p.m. create small input file for case3 with only Cm246 and U238 input file and follow in debugger. The critical parameter in determining CPU time seems to be the number of receive blocks NTX set in SETUP and DXDT. NXT depends on VELISO and DT.

Start slow case 3 with only 1 chain Cm246-U238 running at 4:30 p.m. CPU time for run – 36 minutes.

7:30 pm. Trace the calculation of NTX

$$NX(K) = INT(PATH(MPATH(K))/DX(K) + 0.01)$$

$$NXX = NXX + NX(K) \text{ in DXDT}$$

```

DO 190 K = 1,NPATH
  IF(KFDX(K) .EQ. 0) THEN
    DXK(K) = VELISO(K,IR)*DT/CNM ***** VELISO from CHAIN from FLOWIN
    NXK = INT(PATH(MPATH(K))/DXK(K) + 1.)
    DXK(K) = PATH(MPATH(K))/FLOAT(NXK)
    IF(DXK(K) .LT. DX(K)) THEN
      DX(K) = DXK(K)
      NX(K) = NXK ***** determines NX(K)
    ENDIF DO 210 K = 1, NPT
  IF(VDT(K) .NE. 0) THEN
    NEXK = INT(VDT(K)/DX(K)) + 1
    NEX = NEX + NEXK
  ENDIF
  NXX = NXX + NX(K) *****NX(K) determines NXX
210 CONTINUE
  NTX = NXX + NEX in DXDT *****NXX determines NTX
  VELISO(1,IR) = Y/TRAVT(II) line 6023 in subroutine chain
  VELISO(K,IR) = ABS(PORE(K))/RKI(K,IR) in chain
  VELISO(K,IR) = ABS(TDVEL(M,K))*TDSAT(M,K)/RKI(K,IR) in chain
  VELISO(K,IR) = ABS(TDVEL(M,K))*TDSAT(M,K)/RKI(K,IR) in CHAIN
  READ(11,*,END=1070) TDTM(I), LG, TDVEL(I,1), TDSAT(I,1) in FLOWIN
  OPEN(11,FILE='NEFII.VEL',STATUS='OLD')

```

The large NTX seems to be due to a large mobile RD in leg 2. 10: 00 p.m.

**End: Monday, February 07, 2005, 10. p.m.**

**Start: Tuesday, February 08, 2005, 9 a.m.**

Examine setting of NTX for standard case with small run times. VELISO are relatively large, the smallest being 0.757

Examine source for determining running with only 1 leg. Source release rate read in SOURCE:

```

IF (iopt(14) .ne. 0) THEN
  OPEN (UNIT=14,FILE='sotnef.dat',STATUS='OLD')

```

The input file data is printed out to TAP24 file – an unformatted file used by the transport module TAP24 is opened in FLOWIN:

```

OPEN(24,FILE='TAPE24',FORM='UNFORMATTED',STATUS='UNKNOWN')

```

TAP24 is created in SOURCE

```

WRITE (24) itot
DO 20 i=1,num
  WRITE (24) text(i),(srcext(i,j),j=1,itot)

```

20 CONTINUE

Data is read from TAP24 in SUBROUTINE TRNSPT(T,NTP,NPT,KALL) as follows:

```

REWIND(24)
READ(24) NTOTX

```

```
    READ(24) TS1, (S24(I),I=1,NTOTX)
    DO 20 I = N1,N2
      SR1(I - N1 + 1) = S24(I)
20  CONTINUE
    TS1 = TS1 - TRLSE
    READ(24) TS2, (S24(I),I=1,NTOTX)
    DO 30 I = N1,N2
      SR2(I - N1 + 1) = S24(I)
30  CONTINUE
```

File TAP24 is also read in SUBROUTINE SRCIN as follows:

```
    READ(24,END=50)TS2,(S24(I),I=1,NTOTX)
```

Discovered that the code requires a minimum of two legs or the program stops in FLOWIN  
Try changing the properties of the second leg to be a short leg with large velocity so that the  
nuclides are passed through unchanged. 12:30 p.m.

1 p.m. Make leg 1 very short. Found an infinite loop in METHOD due to TSUM being single  
precision. Corrected by declaring TSUM double precision at 3.00 p.m.

**End: Tuesday, February 08, 2005, 3 p.m.**

**Start: Wednesday, February 09, 2005, 8:30 a.m.**

Investigate mixing cell to determine if it is 1<sup>st</sup> leg. Changing the length of the first leg affects the  
results in a case where the mixing cell is not used therefore one can assume that the first leg is  
used for transport. Try running some shortened chains assuming that radionuclides with large  
retardation factors are immobile to 2:00 p.m.

**End: Wednesday, February 09, 2005, 2:00 p.m.**

**Start: Thursday, February 10, 2005, 8:30 a.m.**

Investigate FACER to limit array sizes. Eliminated adding zero sums to end of BF array then  
removing them later. This change caused an array bounds overflow and was removed at 12:30  
p.m. Try to get a one leg case running by bypassing error trap in code. Didn't work got array  
bounds overflow. Special code is devoted to leg 2. Try making leg 2 have very high velocities  
and unit retardation. Array bounds exceeded. Changed MAXTIM from 50000 to 500000 in  
sizes2.inc. Stop at 3:30 p.m. Review manual for optimization strategies 7:30-8:30 p.m.

**End: Thursday, February 10, 2005, 8:30 p.m.**

**Start: Friday, February 11, 2005, 9:00 a.m.**

Run Case 3 with only Cm246 and U238 with Courant number changed from 1.5 to 0.15 Run time decreased to 38 seconds. Email Ron Janetzke re results to 10:30 a.m. Call Ron Janetzke re work status 1 to 1:30 p.m. Change maxtim back to 50000 and run standard case with Courant number lowered to 0.15 and compare to standard value of 1.5. The case with 0.15 gave substantially different results than the standard case with 1.5. The most evident was CL36 as shown in the table below

YEAR	CL36 C=1.5	CL36 C=0.15
7.54E+03	0.00E+00	0.00E+00
7.56E+03	3.58E-20	0.00E+00
7.58E+03	5.67E-18	0.00E+00
7.60E+03	1.38E-16	0.00E+00
7.62E+03	1.34E-15	0.00E+00
7.64E+03	7.36E-15	0.00E+00
7.66E+03	2.69E-14	0.00E+00
7.68E+03	7.26E-14	0.00E+00
7.70E+03	1.56E-13	0.00E+00
7.72E+03	2.83E-13	0.00E+00
7.75E+03	4.72E-13	0.00E+00
7.77E+03	1.34E-12	0.00E+00
7.79E+03	9.85E-12	0.00E+00
7.81E+03	6.37E-11	0.00E+00
7.83E+03	2.75E-10	0.00E+00
7.85E+03	8.49E-10	0.00E+00
7.87E+03	2.02E-09	0.00E+00
7.89E+03	3.93E-09	0.00E+00
7.91E+03	6.61E-09	0.00E+00
7.93E+03	1.03E-08	0.00E+00
7.95E+03	2.32E-08	0.00E+00
7.97E+03	1.15E-07	0.00E+00
7.99E+03	5.79E-07	0.00E+00
8.01E+03	2.14E-06	0.00E+00
8.04E+03	5.93E-06	0.00E+00
8.06E+03	1.30E-05	0.00E+00
8.08E+03	2.40E-05	0.00E+00
8.10E+03	3.87E-05	0.00E+00
8.12E+03	5.62E-05	0.00E+00
8.14E+03	7.58E-05	0.00E+00
8.16E+03	9.65E-05	0.00E+00
8.18E+03	1.17E-04	0.00E+00
8.20E+03	1.37E-04	0.00E+00
8.22E+03	1.54E-04	0.00E+00
8.24E+03	1.68E-04	0.00E+00
8.26E+03	1.77E-04	0.00E+00
8.28E+03	1.82E-04	0.00E+00
8.30E+03	1.84E-04	0.00E+00
8.32E+03	1.83E-04	0.00E+00
8.35E+03	1.81E-04	0.00E+00
8.37E+03	1.79E-04	0.00E+00

8.39E+03	1.76E-04	0.00E+00
8.41E+03	1.72E-04	0.00E+00
8.43E+03	1.66E-04	0.00E+00
8.45E+03	1.60E-04	0.00E+00
8.47E+03	1.52E-04	0.00E+00
8.49E+03	1.44E-04	0.00E+00
8.51E+03	1.35E-04	0.00E+00
8.53E+03	1.26E-04	0.00E+00
8.55E+03	1.16E-04	0.00E+00
8.57E+03	1.07E-04	0.00E+00
8.59E+03	9.79E-05	0.00E+00
8.61E+03	8.97E-05	0.00E+00
8.64E+03	8.24E-05	0.00E+00
8.66E+03	7.59E-05	0.00E+00
8.68E+03	7.02E-05	0.00E+00
8.70E+03	6.51E-05	0.00E+00
8.72E+03	6.02E-05	0.00E+00
8.74E+03	5.56E-05	0.00E+00
8.76E+03	5.11E-05	0.00E+00
8.78E+03	4.68E-05	0.00E+00
8.80E+03	4.28E-05	0.00E+00
8.82E+03	3.93E-05	0.00E+00
8.84E+03	3.61E-05	0.00E+00
8.86E+03	3.34E-05	0.00E+00
8.88E+03	3.09E-05	0.00E+00
8.90E+03	2.86E-05	0.00E+00
8.92E+03	2.65E-05	0.00E+00
8.95E+03	2.44E-05	0.00E+00
8.97E+03	2.24E-05	0.00E+00
8.99E+03	2.05E-05	0.00E+00
9.01E+03	1.88E-05	0.00E+00
9.03E+03	1.72E-05	0.00E+00
9.05E+03	1.58E-05	0.00E+00
9.07E+03	1.46E-05	0.00E+00
9.09E+03	1.36E-05	0.00E+00
9.11E+03	1.26E-05	0.00E+00
9.13E+03	1.16E-05	0.00E+00
9.15E+03	1.07E-05	0.00E+00
9.17E+03	9.84E-06	0.00E+00
9.19E+03	9.00E-06	0.00E+00
9.21E+03	8.23E-06	0.00E+00
9.23E+03	7.54E-06	0.00E+00
9.26E+03	6.95E-06	0.00E+00
9.28E+03	6.43E-06	0.00E+00
9.30E+03	5.96E-06	0.00E+00
9.32E+03	5.53E-06	0.00E+00
9.34E+03	5.12E-06	0.00E+00
9.36E+03	4.72E-06	0.00E+00
9.38E+03	4.33E-06	0.00E+00
9.40E+03	3.96E-06	0.00E+00

9.42E+03	3.61E-06	0.00E+00
9.44E+03	3.31E-06	0.00E+00
9.46E+03	3.05E-06	0.00E+00
9.48E+03	2.82E-06	0.00E+00
9.50E+03	2.61E-06	0.00E+00
9.52E+03	2.42E-06	0.00E+00
9.55E+03	2.25E-06	0.00E+00
9.57E+03	2.07E-06	0.00E+00
9.59E+03	1.91E-06	0.00E+00
9.61E+03	1.74E-06	0.00E+00
9.63E+03	1.59E-06	0.00E+00
9.65E+03	1.46E-06	0.00E+00
9.67E+03	1.34E-06	0.00E+00
9.69E+03	1.23E-06	0.00E+00
9.71E+03	1.14E-06	0.00E+00
9.73E+03	1.06E-06	0.00E+00
9.75E+03	9.86E-07	0.00E+00
9.77E+03	9.13E-07	0.00E+00
9.79E+03	8.41E-07	0.00E+00
9.81E+03	7.72E-07	0.00E+00
9.83E+03	7.05E-07	0.00E+00
9.86E+03	6.44E-07	0.00E+00
9.88E+03	5.89E-07	0.00E+00
9.90E+03	5.42E-07	0.00E+00
9.92E+03	5.01E-07	0.00E+00
9.94E+03	4.65E-07	0.00E+00
9.96E+03	4.32E-07	0.00E+00
9.98E+03	4.01E-07	0.00E+00
1.00E+04	3.71E-07	0.00E+00

Note that the values for Cl36, I129, Tc99, Ni59, Se79 for a Courant number of 0.15 are all zero whereas with a Courant number of 1.5 non-zero values were obtained which is not at all conservative or acceptable. All the chains in the standard case had zero release except for the Cm245 chain that had a very small release for the last four time steps for a Courant number of 1.5. For a Courant number of 0.15 the last four time points gave zero release rate as shown below

Time	Cm245 C1.5	Am241 C1.5	Np237 C1.5
9.8086E+03	0.0000E+00	0.0000E+00	3.2235E-25
9.9043E+03	0.0000E+00	0.0000E+00	1.2167E-24
1.0000E+04	0.0000E+00	0.0000E+00	4.0759E-24
1.0096E+04	0.0000E+00	0.0000E+00	1.2368E-23

Time	Cm245 C0.15	Am241 C0.15	Np237 C0.15
9.8086E+03	0.0000E+00	0.0000E+00	0.0000E+00
9.9043E+03	0.0000E+00	0.0000E+00	0.0000E+00
1.0000E+04	0.0000E+00	0.0000E+00	0.0000E+00
1.0096E+04	0.0000E+00	0.0000E+00	0.0000E+00

Runs of case 1, case 2 and case 3 were made with Courant numbers of 0.15. Case 1 ran in 1 minute 19 seconds with a Courant number of 0.15

The results for the Cm246-U238 cases were almost identical for the two Courant numbers. The results for Cm 245-Am241-Np237-U233-Th229 were also almost identical for the two Courant numbers.

**End: Friday, February 11, 2005, 3:00 p.m.**

**Start: Monday, February 14, 2005, 1:00 p.m.**

For case 1 with the standard and smaller Courant numbers, the Am243-Pu239 chain gave identical results of zero release. The chain of U234-Th230-Ra226-Pb210 were nearly identical for the two Peclet numbers as were the results for radionuclides Cs135, I129, Se79, Nb94 and Cl36. Tc99 release was different in only the last two entries in the tail. To 4:00 p.m.

Time	Tc99 C=1.5
2.6554E+05	1.1957E-02
2.7130E+05	1.1370E-02
2.7706E+05	1.0811E-02
2.8282E+05	4.3175E-03
2.8858E+05	5.9999E-04
2.9434E+05	1.5643E-05
3.0010E+05	1.8601E-11
3.0586E+05	8.3432E-18

Time	Tc99 C=0.15
2.6554E+05	1.1958E-02
2.7130E+05	1.1370E-02
2.7706E+05	1.0811E-02
2.8282E+05	4.3177E-03
2.8858E+05	6.0073E-04
2.9434E+05	1.5840E-05
3.0010E+05	8.9034E-09
3.0586E+05	1.0574E-14
3.1162E+05	4.7389E-21

Case 2 was executed with a Courant number of 0.15 and ran in 1 minute 38 seconds of CPU. Graphs were made of the release rate for the first two chains. Some differences were noted in the Cm246 chain where some values were somewhat displaced. All other data between the two Courant numbers were very close. From 8:00 to 11 p.m.

**End: Monday, February 14, 2005, 11:00 p.m.**

**Start: Tuesday, February 15, 2005, 9:00 a.m.**

Case 3 with Courant number of 0.15 ran in 2 minutes 31 seconds of CPU. The results for all radionuclides were almost identical for the two Courant numbers. Spreadsheets were prepared to document the results and an email was sent to R. Janetzke containing the results to 12:30 p.m.

**End: Tuesday, February 15, 2005, 12:30 p.m.**

**Start: Wednesday, February 16, 2005, 8:30 a.m.**

Resend emails to Ron Janetzke as his server does not accept zip files. The email was rejected for being too big. Files were FTPed. Cases with I129 standard input were prepared with the velocity file changed to create jumps in the velocity value. Results were recorded and emailed to Ron Janetzke until 4:00 p.m.

**End: Wednesday, February 16, 2005, 4.00 p.m.**

**Start: Thursday, February 17, 2005, 8:30 a.m.**

Begin to test chains with disparate velocities. For standard case increased the simulation time from 1e4 years to 1e6 and added an order of magnitude change in the velocity at 1e5 years. Obtained an array bounds exceeded in the following code lines in the routine TPPRT.

```
mksqdi = qdis*2.831685e-02  
tcount = tcount + 1  
tarr (tcount)= t  
totvol(tcount) = mksqdi*dtfp
```

tarr is dimensioned by maxtim and exceeds array bounds. The simple solution is to increase maxtim.. Finished examination at 11:30 a.m.

Received email from Ron Janetzke with a more typical nefii.vel file. The file was successfully executed with the standard case in 7 minutes and 43 seconds. The resulting nefii.out file was emailed to Ron Janetzke. Time elapsed was from 3:30 pm. to 4:30 p.m.

**End: Thursday, February 17, 2005, 4:30 p.m.**

**Start: Friday, February 25, 2005, 9:00 a.m.**

Begin task defined by Ron Janetzke in an email as follows: "I think there is a point where a large slope of the velocity curve will send NEFTRAN arrays out of bounds. Could you make a couple of runs by modifying one or more slopes in the sawtooth pattern of the NEFII.VEL and see if the run times go up as the slope becomes larger."

The velocity profile was plotted in EXCEL. Results were rerun with original code compiled in double precision. Results were somewhat different than those of altered code with allocatable arrays compiled in single precision. The altered code was compiled in double precision. Results were different from single precision and different than original code but similar in values. Most of the radionuclides had zero release. Graphs were made of the results.

To simplify the process an input file was made with only I129 with a simulation time of 1 million years. This was run with sawtooth velocities with jumps of two orders of magnitude. Array bounds were exceeded in totvol and tar with MAXTIM set at 500000. Using two steps of one

order of magnitude each to achieve two order of magnitude jumps also resulted in array bounds exceeded. Runs terminated at 4:00 p.m.

**End: Friday, February 25, 2005, 4:00 p.m.**

**Start: Saturday, February 26, 2005, 8:30 a.m.**

Begin to trace the references to the totvol and tarr arrays that are going out of bounds. Tarr and totvol are dimensioned by maxtim in routines updcurl, upddr2, inidr2, nefdit, wrtdr2, method and tprt and passed in named common block curcom. In all totvol and tarr are dimensioned in seven routines which is contrary to coding standards. Totvol is used to form array dummyarray which is passed to routine timval in updcurl and upddr2. The array dummyarray is local to both updcurl and upddr2. Totvol and tarr are not used in inidr2, nefdit, wrtdr2 and method even though it is passed in common. tarr and totvol are set in routine tprt in the following lines.

```
mksqdi = qdis*2.831685e-02
tcount = tcount + 1
tarr (tcount)= t
totvol(tcount) = mksqdi*dtfp
```

Totvol and tarr are used only in updcurl and upddr2 where they are passed to the routine timval. The routine updcurl and upddr2 are not used in the program NEF. This is verified by the visual Fortran browser. Also the arrays tarr and totvol are never used. This has also been verified by the visual Fortran browser.

From this I conclude that in the standalone program the arrays totval and tarr that are going out of bounds can be commented out. The value for maxtim was reset to 50000 in include file SIZES2.inc

The code was compiled in double precision with compiler switch `\real_size:64`. This resulted in several warning messages as follows:

Because of COMMON, the alignment of object is inconsistent with its type.

The `\real_size:64` switch was removed and recompiled. Misalignment messages disappeared. Warning messages:

Warning: In the call to TPPRT, actual argument #2 does not match the type and kind of the corresponding dummy argument.

```
CALL TPPRT(TEND,0.0D0)
```

Remained because the dummy argument is real and 0.0D0 is double. This was left alone for now.

Obtained array bounds exceeded in the following line:

```
curies(tcount,is+mfndx) = curout(is)*dftp
```

in routine TPPRT.

The code was searched and it was discovered the array curies is not used so this line was commented out.

The code ran successfully with the sawtooth velocities with maxtim at 50000 at 10:30 a.m. in 31 seconds CPU for I129 only.

The code was run again with two order of magnitude jumps in the velocity file. Thos code ran successfully in 5 minutes and 10 seconds of CPU for I129 only. Runs terminated at 11:00 a.m.

**End: Saturday, February 26, 2005, 11 a.m.**

**Start: Monday, February 28, 2005, 9:00 a.m.**

Begin documentation of results of code changes to eliminate out of bounds error for velocity files and email and FTP results to Ron Janetzke to 11 a.m.

Run a case with an order of magnitude lower time steps. Results graphed and sent to R. Janetzke. With order of magnitude smaller time steps run was completed in 31 seconds CPU. From 1.p.m to 3:30 p.m.

**End: Monday, February 28, 2005, 3:30 p.m.**

**Start: Friday, March 04, 2005, 8:30 a.m.**

Make code changes of commenting out curies, totvol and tarr arrays to avoid out of bounds error Changing Real declarations to Real\*8 according to QA instructions given by Ron Janetzke was investigated and found to be very time consuming and error prone and was not implemented – to 10:30 a.m.

**End: Friday, March 04, 2005, 10:30 a.m.**

**Start: Wednesday, April 13, 2005, 9:30 a.m.**

The code release.f was examined for process level validation testing . The subroutines in releaset5e are:

subroutine liqrel(dtt, tcool, vtmp, itype, tend)- liquid source term model  
subroutine derivs(t, y, dydx) - Main code module for analyzing seismic induced mechanical failure.

subroutine decay(t, xni0, xnij, alam) - chain decay by bateman equation to calculate contribution of the ith chain member to the jth chain member where j is a daughter member  
subroutine odeint(ystart, nvar, x1, x2, eps, h1, hmin, hmax, nok, nbad, tiny, dydx) - This routine, odeint, is based on the Runge-Kutta method for the integration of ordinary differential equations (ODE) from the book "Numerical Recipes in FORTRAN"  
subroutine leachrt(temp, srate,ctotal) - spent fuel matrix dissolution  
subroutine opnfil(unit, filnam, status, lplain, errnum, errmsg) - This module opens a formatted file on the unit specified with the attributes provided in the argument list.  
subroutine init - ?  
subroutine input - ?  
subroutine rdelem(unit, elefil) - Module to input the radionuclide inventory and properties  
subroutine lint(xa, ya, n1, n, x, y) - Module for linear interpolation between two specified limits  
SUBROUTINE rkqs(y,dydx,n,x,htry,eps,yscal,hdid,hnext,derivs) - This routine, rkqs, is based on the Runge-Kutta method of integration with adaptive stepwise control  
SUBROUTINE rkck(y,dydx,n,x,h,yout,yerr,derivs) - ?

The calling sequence in main is:

```
call init
call input
call liqrel(dt, tcool, vmax1, itype, simtimex )
call decay(ttemp(itemp), atall, atoms, alam)
call opnfil(20, 'ebsnef.dat', 'replace', .true., errnum, errmsg)
call decay(tempvar1, atall, atoms, alam)
call decay(tempvar1, atall, atoms, alam)
```

The calling sequence in liqrel is:

```
call decay(t - dt, atall, atoms, alam)
call odeint(ystart, nvar, tstart, tstop, eps, dtinit, dtmin,dtmax, nok, nbad, tiny, dydx)
```

The calling sequence in derivs is

```
call lint(tflo, qin, maxtim, ntflo, t, qinwp)
call decay(t, atall, atoms, alam)
call leachrt(tavg, srate,utotal)
```

the calling sequence in odeint is:

```
call derivs(x, y, dydx)
call rkqs(y,dydx,nvar,x,h,eps,yscal,hdid,hnext,derivs)
```

Functions are:

```
FUNCTION linearInterpolation(xValue, yValue, numberOfElements, & xInterpolate)
FUNCTION getDSFactor(idFailure, timeValue)
FUNCTION getWeldWPFactor(idFailure, timeValue, fractionTilt)
function sawetfunc(t)
FUNCTION linearInterpolation(xValue, yValue, numberOfElements, & xInterpolate)
FUNCTION searchPosition(searchVector, searchElement, & numberOfElements)
```

Finish 11:00 a.m.

Start 1.00 p.m.

The output files are: relcum.out, rel\_flow.out, diagnose.out, trelease.out, frac\_rel.out, cumrelse.out, inv1000.out, relfrac.out, maxrel.dat and ebsnef.dat. the main input file is ebsrel.inp In this input file the names of three other input files are found. These names appear typically to be ebstrh.dat, ebsflo.dat and ebspac.nuc. Phone call with Ron Janetzke re validation from 3:30 to 4:30 p.m.

**End: Wednesday, April 13, 2005, 4:30 p.m.**

**Start: Thursday, April 14, 2005, 9:00 a.m.**

Begin compiling TPA500q.

EBSREL.f:

Warning: In the call to RELEASECOLLOIDS, actual argument #4 does not match the type and kind of the corresponding dummy argument.

call releasecolloids ( iebnefdat, nnucl, numclnuc, cldnames)

This line of code was added to EBSREL.f

C\*\*\*\*\*Code added by D.LeNeveu April 14, 2005

character\*6 cldnames(maxclnuc)

C\*\*\*\*\*end of code added

In subroutine newinventdb() in file invent.f the variables yearofburnupdata and yearofemplacment are declared twice once in the common block include 'inventp.i' and once in the subroutine. The declaration in the subroutine was commented out.

The following code change was made twice in the file nfenv.f to avoid compilation error in Visual Fortran

cc dml 04-14-10 This declaration has been moved to avoid compilation error in VISUAL FORTRAN

c ZW SCR478 05-25-04:

c INTEGER inFile, numBeforeRows, numAfterRows, numTime

INTEGER numTime

cc end of change by D. LeNeveu

The file linintrap.f had to be added from the codes directory to the build. The build did not work because eight objects were undefined. An email was sent to Ron Janetzke requesting the missing code.

Relaset.f was compiled using zportfundate made especially for the NEFTRAN code compiled under Visula Fortran. A successful compilation was obtained but the input files obtained from tpa500e were of the wrong format and the code would not run. In particular ebsflo.dat was

missing a required fifth column. A good set of input files was requested from Ron Janetzke by email.

**End: Thursday, April 14, 2005 11:30 a.m.**

**Start: Friday, April 15, 2005 at 9:00 a.m.**

Attention has been diverted for one day to review Final Environmental Impact Statement for a Geologic Repository for the Disposal of Spent Nuclear Fuel and High-Level Radioactive Waste at Yucca Mountain, Nye County, Nevada, Volume 1, Chapters 5 and 2. High level review completed and emailed to Sitkanta Mohanty at 3:30 p.m.

**Start: Monday, April 18, 2005, 12:30 p.m.**

Begin tracing computations in Releasetf for validation exercise.

The governing differential equation for Releasetf appears to be equation 8.1 of the TPA version 4.0 code manual. The critical parameter for driving this equation is  $w_{li}$  – rate of transfer from the SF into the resident water in the WP because of through leaching of the spent fuel. This is determined from the spent fuel dissolution rate  $r$  described in section 8.34. The difference between  $w_{li}$  and  $r$  appears to be the units and the radionuclide dependence.  $w_{li}$  has units of mol/a while  $r$  has units of  $\text{mg}/\text{m}^2/\text{d}$ . The spent fuel dissolution rate  $r$  has five modes depending on the value of imodel. Only four of these are documented in section 8.3.4. The documentation for the fifth option appears to be missing.

The following code corresponds to the spent fuel dissolution options described in section 8.3.4

```
tempk = temp + 273.
```

C--> model 1: in the absence of Ca and Si

```
if ( imodel.eq.1 ) then
  xxx = 9.310 - 0.142*pco3 - 16.7*log10(oxgnovpr)
&    - 0.14*phvalue - 2130./tempk + 6.811*log10(tempk)
&    *log10(oxgnovpr)
  rate1 = 10**xxx
  srate1 = rate1*1.E-6*365.
  srate = srate1
end if
```

C--> model 2: dissolution in the presence of Ca and Si

```
if ( imodel.eq.2 ) then
c   a rate equation to introduce experimental data
c   obtained at 25 C and 85 C
```

```

aaa = preexpo
c   aaa = 3.45*1.e4
    rate2 = aaa*exp( - 34.3/8.314E-3/tempk)
    srate2 = rate2*1.E-6*365.
    srate = srate2
end if

```

C--> model 3: user-specified dissolution

```

if ( imodel.eq.3 ) then
  srate = usrlrate
end if

```

C--> model 4: model for schoepite solubility

```

if (imodel.eq.4 ) then
c   calculate uranium species in equilibrium with schoepite
c   uranyl ion K from Vant Hoff eqn
    data ksc0,hsc,tref,rgas/64818.,-50.39,298.15,.0083144/
c   hydrogen ion conc
    h=10**(-phvalue)
    ksc=ksc0*exp (hsc*(1./tref-1./tempk)/rgas)
    usc=ksc*h**2
c   concentration uo2co3
    data k10,h1 /4.434,-19.7/
    k1=k10*exp (h1*(1./tref-1./tempk)/rgas)
    cco3=10**(-pco3)
c   calculate bicarbonate from total carbonate and pH, igoring U species
c   assumes that pco3 includes all carbonate species
    data hbc0,hbc2/9.51,-14.708/
    data kbc00,kbc20/4.5217e-7,2.1321e10/
    kbc0=kbc00*exp (hbc0*(1./tref-1./tempk)/rgas)
    kbc2=kbc20*exp (hbc2*(1./tref-1./tempk)/rgas)
    hco3=cco3/(1.+1./(kbc2*h)+h/kbc0)
c   hco3=kbc2*cco3*h
    c1=usc*hco3/(k1*h)
c   concentration of uo2(co3)2
    data k20,h2/5.1713e3,-47.9/
    k2=k20*exp (h2*(1./tref-1./tempk)/rgas)
    c2=usc*hco3**2/(k2*h**2)
c   concentration of uo2(co3)3
    data k30,h3/2.4010e9,-4.9/
    k3=k30*exp (h3*(1./tref-1./tempk)/rgas)
    c3=usc*hco3**3/(k3*h**3)
c   concentration of UO2(OH)2

```

```

data k00,h0/1.9962e10,0.0/
k0=k00*exp (h0*(1./tref-1./tempk)/rgas)
c0=usc/(k0*h**2)
c  solubility of uranium species in water
ctotal=usc+c1+c2+c3+c0
end if

if (imodel.eq.5) then
c  Glass release model from TSPA-SR depends on pH
if(phglass.le.7.0) then
c  low pH case
sr=kefflow*10**(etalow*phglass)*exp(-ealow/(rgas*tempk))
else
sr=keffhi*10**(etahi*phglass)*exp(-eahi/(rgas*tempk))
end if
c  convert gm/m2/day to kg/m2/yr
srate=sr*365./1000.
end if

```

A major job is to determine the time dependence of the main drivers in the governing differential equation. Tempk is a function of time. Tempk is determined from tavg when leachrt is called. Tavg is determined in liqrel as follows:

```

tavg = tcan(it - 1)
tcan and other temperature data is read in as follows:

```

```

do 100 it = 1, ntemp
if(imodel.ne.5) then
c  read temperatures for fuel waste packages
read (nunit(3), *) inputit, ttemp(it), tcan(it), tave(it)
else
c  read temperatures for glass canisters by switching
c  tcan and tave
read (nunit(3), *) inputit, ttemp(it), tave(it), tcan(it)
end if

```

in subroutine input

The parameter qout – water leaving the waste package at time t is passed in the common block intvars. qout is set in subroutine derives:

```

if(bathflow.eq.0) then
c  bathtub model

```

```

      if(.not.fullw) then
        if (vwnow.le.vmax ) then
          qout = 0.0
        else
c       set derivitive to zero
          qout=qinwp
        end if
      else
        qout=qinwp
      end if
    else
c     flow through model - force immediate outflow
      qout=qinwp
      vwnow=vfullflow
    end if

    flowrate_out(it) = qout
    flowrate_in(it) = qinwp

```

qinwp is determined by interpolation from a cal to lint using the variable qin(maxtim) passed in the common block flowdata. Qin is set in the following code

```

      do i = 1, ntflo
        ds_factor = getDSFactor(itype, tflo(i))
        weld_wp_factor = getWeldWPFactor(itype, tflo(i), ftilt)
        qin(i) = (flore(i)*flowfactr) * ctfmult(i) * ctflow(i) *
&       ds_factor * weld_wp_factor

```

In main program releaset5e

Ctflow is the time dependent flow factor initialized to zero in this code:

```

      do 200 i = 1, ntflo
        floref(i) = 0.
        ctflow(i)=0.0
        ctfmult(i)=0.0
c
      200 continue
In subroutine init.

```

Ctflow is read in subroutine input as follows:

```

          read (nunit(5), *) tflo(i), floref(i),ctfmult (i), ctflow(i),
1    sawettab(i)
      200 continue

```

from ebsflo.dat

The UO<sub>2</sub> dissolution rate, uo2rate, leaching rate of sf matrix radionuclide [kg/yr]  
c is determined from:

```
if(imodel.eq.4) then
  uo2rate=utotal*xfrac*qout*238.

else if(imodel.eq.5) then
  sareat=y(3)*xfrac*simglass

else
  sareap = 4. * pi* r0z**2
  sareat = sareap*(y(3)*xfrac)/(4./3.*pi*r0z**3*fueden)

endif

if(imodel.eq.5) then
  saream = sareat
else

if(y(2).gt.0.0) then
  saream = sareat * cladfactor * (1.0-y(2)/ltube0)
else
  saream = sareat * cladfactor
end if
end if

if (y(3).gt.0.0.and.imodel.ne.4) then
  uo2rate = srate*saream
elseif (y(3).le.0.0. and. imodel.ne.4) then
  uo2rate = 0.0
end if
```

End analysis 3:30 p.m.

Phone call Ron Janetzke re validation planning and compiler 4:30-5.00 p.m.

Investigate solution methods to governing differential equation 7:00-8:30 p.m.

**End: Monday, April 18, 2005 8:30 p.m.**

**Start: Tuesday, April 19, 2005, 9:00 a.m.**

Begin compiling TPA500q code with ZPORTCunx from Ron Janetzke. Code changes were made to special zportunx.f so make it work with Visual Fortran also calls to an unused Sun function zportiee\_flags in exec.f was commented out. A dummy SUN function called SH was added to zportunx.f. The executable formed would not run because of the function indexperiso(name). In this routine there is the following statement:

```
if (ikey .ne. 929318) then
  call newinventdb()
endif
```

unfortunately the value for key is not set - then the code goes looking for a file called burnup.dat, then it tries to open drifts.dat in subroutine fillsubareas which I could not find.

Phone call to R. Janetzke 3:30 to 4:00 re validation for releaset.f

Changed path.i include file to set path to codes and data and ran tpa code. It produced many output files and then an error message about as follows:

snllhs.exe is not recognized as an internal or external command, operable or batch file

file not found .....lhs.out

**End: Tuesday, April 19, 2005 4:30 p.m.**

**Start: Wednesday, April 20, 2005, 9:00 a.m.**

Review SCR changes sent my Ron Janetzke to plan validation for releaset.f. Changes are relatively minor.

Begin to trace in code nuclide inventory and decay and use of solubility. Not itype is failure type. Radionuclide inventory is read in from elefil ebspac.nuc

The first entry is the number of elements. Then five parameters are read for each element

```
read (unit, 9001) elem(ielem), ncon(ielem), sol(ielem),
&          rde(ielem), dele(ielem)
```

Elem(ielem)-2 character Element name

```
c ncon(ielem): number of chains in which isotopes of element ielem
c sol(ielem):  solubility limit of the element ielem
C rd:         retardation factor for each isotope
c rde(ielem): retardation factor for elements
c dele:       Diffusion correction factor for elements
```

Then two more are read:

```
read (unit, *) kcon(ielem, l), icon(ielem, l)
c kcon(ielem,l): chain identifier among all chains in which element ielem
c                occurs; l is the index for all chains with ielem
```

c icon(ielem,l): position of ielem in the chain kcon(ielem,l)

Then

```

c  read number and size of chains
  read (unit, *) nchns, (ni(i), i=1, nchns)
  inuc = 0
c  read atomic wt, name, halflife [yr], inventory [ci], gap fraction
  do 200 k = 1, nchns
    nn = ni(k)
    do 150 i = 1, nn
      read (unit, 9002) amall(k, i), namall(k, i), halflife(k, i),
&          curall(k, i), fggap(k, i)
      inuc = inuc + 1
    150 continue
  200continue

```

Curies are converted to atoms in subroutine liqrel as follows:

```

C-->  convert curies/wp -> atoms/wp
      atall(k, i) = curall(k, i)*(halflife(k,i)*1.6834E18)

```

Bateman equations are used to calculate inventory in subroutine decay

```

      subroutine decay(t, xni0, xnij, alam)
t -time
c xni0:  initial inventory of the jath elemnt of the kchn chain -input
C xnij:  nventory of the jth element of kchn chain at time t - output
c alam:  decay constant for each radionuclide - input

```

alam calculated main in as follows:

$$\text{alam}(k, i) = \log(2.0)/\text{halflife}(k, i)$$

Here is the congruent release implementation in subroutine derivs

```

do 300 k = 1, nchns
  do 250 i = 1, ni(k)

    if(imodel.eq.5) then
      amass(k, i) = (atoms(k,i)/6.02E23)*amall(k, i)*0.001
    else
      amass(k, i) = (1. - fggap(k,i))*(atoms(k,i)/6.02E23)
&          *amall(k, i)*0.001
    end if

```

- C leach rate of daughter i of chain k [kg/y]  
 C in this version, wu is only a 1 and 0 switch.  
 c congruent release implementation

$$wleach(k, i) = (amass(k,i)/amassc)*uo2rate*wu*fut$$

250 continue

300 continue

- C amassc: initial mass of uranium oxide fuel, kg  
 read (4, \*) amassc  
 read (4, \*) fueden

This is what happens to wleach:

```
do 700 k = 1, nchns
  do 650 i = 1, ni(k)

    if ( i.gt.1 ) then
      xxx1 = amt1(k,i-1)*6.02E23/amall(k,i-1)/0.001
      rdp=xxx1*alam(k, i - 1)
    else
      rdp = 0.0
    end if

    xxx2 = amt1(k,i)*6.02E23/amall(k,i)/0.001
    dissip = xxx2*alam(k, i)

    adflux(k, i) = ccf(k, i)*qout
    difflux(k, i) = cdiff*cd(k,i)*ccf(k,i)
    rf(k, i) = adflux(k, i) + difflux(k,i)
```

C==> rate of change of mass in wp water

```
xxx3 = (rdp-dissip)
xxx3 = (xxx3/6.02E23)*amall(k,i)*0.001
dam(k,i) = wleach(k,i) + xxx3 - rf(k,i)

jj = jj + 1
dydx(jj) = dam(k, i)
if ( k.eq.1 .and. i.eq.1 ) dissipxx = dissip
if ( k.eq.1 .and. i.eq.1 ) rdpxx = rdp
```

650 continue

700 continue

Note wleach gets converted in subroutine derives into dam and then to dx dy for use in odeint the ode solver.

Note this code that affects UO2 dissolution rate:

```
call leachrt(tavg, srate, utotal)

srate = srate * ratefactor

if(imodel.eq.4) then
  uo2rate=utotal*xfrac*qout*238.

else if(imodel.eq.5) then
  sareat=y(3)*xfrac*simglass

else
  sareap = 4. * pi* r0z**2
  sareat = sareap*(y(3)*xfrac)/(4./3.*pi*r0z**3*fueden)

endif

if(imodel.eq.5) then
  saream = sareat
else

if(y(2).gt.0.0) then
  saream = sareat * cladfactor * (1.0-y(2)/ltube0)
else
  saream = sareat * cladfactor
end if
end if

if (y(3).gt.0.0.and.imodel.ne.4) then
  uo2rate = srate*saream
elseif (y(3).le.0.0. and. imodel.ne.4) then
  uo2rate = 0.0
end if
```

Cladfactor and ratefactor are read in.

Solubility sol(ielem) is used in the following code to limit concentration:

```

do 500 ielem = 1, nelem

  affected_flag = .FALSE.

  if ( vwnow.ge.1.E-20 ) then
    concn = amassl(ielem)/vwnow

    if ( concn.gt.sol(ielem) ) THEN

      IF(qout .GT. 0.0) affected_flag = .TRUE.
      concn = sol(ielem)
    endif

  else
    concn = 0.0
  end if

```

This code determines the derivatives of mass change dxdy for use in odeint taking into account solubility leach rate and diffusive flux:

```

jj = 3

do 700 k = 1, nchns
  do 650 i = 1, ni(k)

    if ( i.gt.1 ) then
      xxx1 = amt1(k,i-1)*6.02E23/amall(k,i-1)/0.001
      rdp=xxx1*alam(k, i - 1)
    else
      rdp = 0.0
    end if

    xxx2 = amt1(k,i)*6.02E23/amall(k,i)/0.001
    dissip = xxx2*alam(k, i)

    adflux(k, i) = c CFR(k, i)*qout
    diffflux(k, i) = cdiff*cd(k,i)*ccfr(k,i)
    rf(k, i) = adflux(k, i) + diffflux(k,i)

C==>   rate of change of mass in wp water

    xxx3 = (rdp-dissip)
    xxx3 = (xxx3/6.02E23)*amall(k,i)*0.001

```

```

dam(k,i) = wleach(k,i) + xxx3 - rf(k,i)

jj = jj + 1
dydx(jj) = dam(k, i)
if ( k.eq.1 .and. i.eq.1 ) dissipxx = dissip
if ( k.eq.1 .and. i.eq.1 ) rdpxx = rdp

650  continue
700  continue
do 800 k = 1, nchns
do 750 i = 1, ni(k)
jj = jj + 1
if ( i.gt.1 ) then
xxx1 = amt2(k,i-1)*6.02E23/amall(k,i-1)/0.001
rdp2=xxx1*alam(k, i - 1)
else
rdp2 = 0.0
endif
xxx2 = amt2(k,i)*6.02E23/amall(k,i)/0.001
dissip2 = xxx2*alam(k, i)
xxx3 = (rdp2-dissip2)
c    from atoms --> kg
xxx3 = (xxx3/6.02E23)*amall(k,i)*0.001
dydx(jj) = xxx3 + adflux(k, i)+diffflux(k,i)
750  continue
800  continue

do 900 k = 1, nchns
do 850 i = 1, ni(k)
jj = jj + 1
c    this is the derivative for advective and diffusive flux out only
dydx(jj) = adflux(k, i) + diffflux(k, i)
850  continue
900  continue

```

It appears that xx3 is the rate of production of radionuclides from the parent.

Note that rdp is defined as mass rate of production of r.n. from parent in the wp water

Question to sort out: what is the significance of the indexes for dxdy?

Note also that the mass balance for UO<sub>2</sub> has not yet been found so that the leachrate is terminated when all the UO<sub>2</sub> is exhausted. It appears that there is no provision in the code for mass balance in terms of ensuring that the leach rate terminates when all the UO<sub>2</sub> is exhausted

Next task is to trace how the void fraction in the fuel and the fuel surface area are determined as there is no documentation for this in the tpa code manual.

The void fraction is determined in this line of code:

```
voidfrac = xvol/(pi*rintl**2*xlintl)
```

found in main.

Xvol is defined as :

c xvol: interior volume of the wp [m<sup>3</sup>] in subroutine input.

Xvol is read in in this statement:

```
read (4, *) xvol
```

We have two surface areas defined in subroutine liqrel:

c sareap: total surface area from one SF particle  
c sareat total internal surface area in the wetted portion of sf

sareat is calculated in derivs. It is transformed into saream as defined as:

c saream: modified total internal surface area in the wetted portion of sf

saream depends on imodel and cladfactor and y(2)

y(3)=inventory? Defined where?

vwnow: volume of water in the wp at time t (same as y(1))

**End: Tuesday April 20, 4:30 p.m.**

**Start: Thursday, April 21, 2005, 9:00 a.m.**

Find in code the implementation of the time to fill to hole that should control time of leakage.

It appears to be done in subroutine derives in this code:

```
vwnow = y(1) + 0.01
if(bathflow.eq.0) then
c bathtub model
if(.not.fullw) then
if (vwnow.le.vmax ) then
qout = 0.0
```

```

      else
c      set derivitive to zero
        qout=qinwp
      end if
      else
        qout=qinwp
      end if
    else
c    flow through model - force immediate outflow
      qout=qinwp
      vwnow=vfullflow
    end if

```

```

flowrate_out(it) = qout
flowrate_in(it) = qinwp

```

```

if(.not.fullw) then
  dvdt = qinwp - qout
else
  dvdt=0.0
endif
dydx(1) = dvdt

```

The vnow is yet to be found

The flow into the canister appears to be calculated here:

```

C+++++ calculate liquid release ++++++
  do i = 1, ntflo
    ds_factor = getDSFactor(itype, tflo(i))
    weld_wp_factor = getWeldWPFactor(itype, tflo(i), ftilt)
    qin(i) = (floref(i)*flowfactr) * ctfmult(i) * ctfow(i) *
&    ds_factor * weld_wp_factor

```

In main

This is the implementation of equation 8-15 in the TPA code manual

The water flow in is read into the variable ctfow in the following line:

```

      read (nunit(5), *) tflo(i), floref(i),ctfmult (i), ctfow(i),
1      sawettab(i)

```

Qin is used in derivs as follows:

```

      call lint(tflo, qin, maxtim, ntflo, t, qinwp)

```

This appears to be the only place it is used. The flow rate in the wp at input time t is determined here from linear interpolation.

What is sawettab(i) What is it used for?

What is wetfrac(itype)?

It is defined here:

C wetfrac(i): the fractional liquid level in the wp below which the waste matrix is wet

It is read in here:

```
read (4, *) wetfrac(1), bathflowt(1)
```

c bathflowt Tell whether wp failure is bathtub or flowthru type

What is done with the retardation factor rde? Nothing???

Waste package fill time is defined here:

```
C+++++ calculate liquid release ++++++
```

```
...
```

```
DO itemp = 2, ntemp
c   The waste package is filling if fill_flag is set
   IF(flowrate_in(itemp) .GT. 0.0D0 .AND.
   &  (.NOT. fill_flag_start)) THEN
     waste_package_fill_start = ttemp(itemp)
     fill_flag_start = .TRUE.
   ENDIF
   IF(flowrate_out(itemp) .GT. 0.0D0 .AND.
   &  (.NOT. fill_flag_stop)) THEN
     waste_package_fill_stop = ttemp(itemp)
     fill_flag_stop = .TRUE.
   ENDIF

ENDDO
```

The fill time appears to be first determined in derives as follows:

```
vwnow = y(1) + 0.01
if(bathflow.eq.0) then
c   bathtub model
   if(.not.fullw) then
```

```

      if (vwnow.le.vmax ) then
        qout = 0.0
      else
c      set derivative to zero
        qout=qinwp
      end if
      else
        qout=qinwp
      end if
      else
c      flow through model - force immediate outflow
        qout=qinwp
        vwnow=vfullflow
      end if

      flowrate_out(it) = qout
      flowrate_in(it) = qinwp

```

This means that y(1) must be a cumulative variable for the volume of water in the waste package???

Y(i) seems to be modified in subroutine rkqs as follows:

```

      do 12 i=1,n
        y(i)=ytemp(i)
      12 continue

```

This very hard to follow!

An important task for QA is to trace all input parameters to the theory manual and a calculation scheme that can be used in QA.

The input files are:

Ebsrel.inp, ebsflo.dat, ebspac.nuc, and ebstrh.dat

Parameters in Ebsrel.inp

Name	definition	Used in
xcon	# of WP	nremain = mtot - ideofect
sawetfrac	wetted subarea	xnloss, xmass
defectt	initially defective time [yr]	if ( itype.eq.1 ) then tfail=max(tfirstflow,defectt) if (defectt.gt. min(cftime, weld_failure_time) & .and.ideofect.ne.0) then ideofect = 0 endif
ideofect	# WPs affected initial defect	if ( itype.eq.1 ) then tfail = max(tfirstflow,defectt) numwp = ideofect

```

                                end if
                                nremain = mtot - ndefect
sftimef      faulting fail time [yr]
                                if(sftimef.gt.min(cftime,weld_failure_time)
                                & .and.isconf.ne.0) then
                                    isconf = 0
                                endif

                                if (sftimev.gt.min(cftime, weld_failure_time)
                                & .and.isconv.ne.0) then
                                    isconv = 0
                                endif
                                if ( itype.eq.2 ) then
                                    tfail = max(tfirstflow,sftimef)
                                    numwp = isconf
                                end if
isconf      # WPs affected from faulting
                                iscon = isconf + isconv + iseismp1 + iseismp2
                                    + iseismp3 + iseismp4
                                if ( itype.eq.2 ) then
                                    tfail = max(tfirstflow,sftimef)
                                    numwp = isconf
                                end if
sftimev      volcano fail time
isconv      #wp affected from volcano
                                if (sftimev.gt.min(cftime, weld_failure_time)
                                & .and.isconv.ne.0) then
                                    isconv = 0
                                endif
seismt1-4    seismic failure time [yr]
seismp1-4    # WPs affected

                                if (seismt1.gt.min(cftime, weld_failure_time)
                                & .and.iseismp1.ne.0) then
                                    iseismp1 = 0
                                endif
                                if ( itype.eq.4 ) then
                                    tfail = max(tfirstflow,seismt1)
                                    numwp = iseismp1
                                end if
                                iscon = isconf + isconv +
                                & iseismp1 + iseismp2 + iseismp3 + iseismp4

dintl      wp Internal Diameter [m]
xlintl     wp internal length [m]
xvol      wp internal vol[m3]

                                if ( numwp.gt.0 ) then
                                    waterht = wetfrac(itype)*dintl
                                    rintl = dintl/2.
                                    voidfrac = xvol/(pi*rintl**2*xlintl)
                                    vabs = xlintl*(rintl**2*theta - (xxx * rintl)**2*tan(theta))
                                    xfrac = vabs/(pi*rintl**2*xlintl)

ctemp      BP of water at atm. condition [C]

C          time for onset of condensation

do 400 i = 2, ntemp
    temp1 = tcan(i - 1)
    temp2 = tcan(i)

```

```

    if ( temp2.le.temp1 .and. temp1.le.ctemp ) go to 500
400   continue
401
amassc          SF mass per WP [kg]

    in derivs
    wleach(k, i) = (amass(k,i)/amassc)*uo2rate*wu*fut
C+++++ inside the waste package ++++++

    amass0 = amassc
    in liqrel
    ystart(3) = amassc

fueden          SF density [kg/m3]

    in derivs
    if(imodel.eq.4) then
        uo2rate=utotal*xfrac*gout*238.

    else if(imodel.eq.5) then
        sareat=y(3)*xfrac*simglass

    else
        sareap = 4. * pi* r0z**2
        sareat = sareap*(y(3)*xfrac)/(4./3.*pi*r0z**3*fueden)

    endif

    velfuel=rrod*srate*3/(r0z*fueden)

wetfrac(1-8)    the fractional liquid level in the wp below which the waste
                matrix is wet

C+++++ inside the waste package ++++++
                waterht = wetfrac(itype)*dintl

bathflowt(1-8)  water contact 0-bathtub 1-flow thr

    bathflow=bathflowt(itype)
    if(bathflow.eq.0) then
c      bathtub model
        if(.not.fullw) then
            if (vwnow.le.vmax ) then
                gout = 0.0
            else
c              set derivitive to zero
                gout=qinwp
            end if
        else
            gout=qinwp
        end if
    else
c      flow through model - force immediate outflow
        gout=qinwp
        vwnow=vfullflow
    end if
    IF ( bathflow .eq. 0 ) THEN

        tot_waste_packages_filled = tot_waste_packages_filled +
&         numwp
        tot_waste_package_fill_time = tot_waste_package_fill_time +

```

```

&      numwp * (waste_package_fill_stop - waste_package_fill_start)
ELSE
  tot_waste_packages_flothru =
&      tot_waste_packages_flothru + numwp
ENDIF

imodel          leaching model

in subroutine derivs
if(imodel.eq.4) then
  uo2rate=utotal*xfrac*qout*238.

else if(imodel.eq.5) then
  sareat=y(3)*xfrac*simglass

else ... etc.

also in subroutine leachrt, input and in main

phvalue          pH                in leachrt Eq:8-9 and Table 8-1
oxgnovpr         [atm]              in leachrt Eq: 8-9
pco3             used if imodel=1    in leachrt Eq: 8-9 and Table 8-1
usrllrate        [kg/yr/m2]: used if imodel=3  in leachrt and page 8-9
preexpo          preexponential term for imodel=2 Eq:8-10
ratefactor       waste dissolution enhancement factor srate = srate*
                                                         ratefactor in derivs
ebspac.nuc'      file name for nuclide data

C-14 generation  12 parameters

r0z              radius sf particle    velfuel=rrod*srate*3/(r0z*fueden)
radu             radius of uo2 fragment  not used
radsg           subgrain radius after transgranular fracture  not used
cladfactor       protection factor for cladding. 1.0 -no protection - saream
cladve          cladding velocity enhancement factor
                                                         velfuel*cladve
ltube0          initial ltube          ystart(2) = ltube0*0.99

if(y(2).gt.0.0) then
  saream = sareat * cladfactor * (1.0-y(2)/ltube0)
else
  saream = sareat * cladfactor
end if

rrod            thickness of cladding    velfuel=rrod*srate*3/(r0z*fueden)
thclad         thickness of cladding    not used
cfuel          ci of cl4 / kg of fuel in fuel  not used
czmetal        ci of cl4 / kg of fuel in zirconium cladding  not used
czoxide        ci of cl4/kg of fuel in initial zirc oxide and crud-not used
cgap           ci of cl4 / kg of fuel in grain boundary and gap  not used

Glass          undocumented

Code fragments for glass:
c              time-dependent diffusivity term grouping
              cdiff=ftilt*d_water*diff*365*86400/(cr_length/
1 (cr_area_a+cr_area_b*t)+
1 l_internal/a_internal)

diffflux(k, i) = cdiff*cd(k,i)*ccfr(k,i)
keffhi = 10**logkeffhi

```

```

kefflow = 10**logkefflow

      else if(imodel.eq.5) then
        sareat=y(3)*xfrac*simglass

      if (imodel.eq.5) then
c      Glass release model from TSPA-SR depends on pH
        if(phglass.le.7.0) then
c      low pH case
          sr=kefflow*10**(etalow*phglass)*exp(-ealow/(rgas*tempk))
        else
          sr=keffhi*10**(etahi*phglass)*exp(-eahi/(rgas*tempk))
        end if
c      convert gm/m2/day to kg/m2/yr
          srate=sr*365./1000.
        end if

simglass      m^2/kg      used to calculate sareat
phglass      used to calculate sr
logkeffhi
etahi
eahi      kj/mol deg K
logkefflow
etalow
ealow kj/mol deg K
cr_length    length of crack, meters
cr_area_a    area of crack at t=0, m^2
cr_area_b    slope of crack growth, m^2/yr
l_internal   length of interior film pathway,
a_internal   cross section of interior film, m^2
d_water      diffusivity of water at 20C, m^2/sec
ftilt        fraction of WPs in diffusion orientation

Grid parameters

Rock parameters
rpor(1..nzones)    rock porosity      not used in releset.f

Radionuclide transport
diff(1..nzones)    diffusion coef. [m2/yr]      not used in releset.f
driftdia [m]      not used in code releset.f

Solution algorithm control parameters (Runge-Kutta)
dtinit [yr]
dtmin [yr]
dtmax [yr]
eps
tiny

output parameters
nbt      number of time intervals for output

parameters in ebsflo.dat

flowfactr    flow loading in a subarea      Eq 8-15 - qin
ntflo        number of times (data rows)    Eq 8-15 - qin

tflo(1-ntflo)    time of floref [yr]      Eq 8-15 - qin
floref(1-ntflo)    reference flow (m^3/yr)  Eq 8-15 - qin
ctfmult(1-ntflo)    time-dependent fmult factor  Eq 8-15 - qin

```



time using the Runga Kutta solver odeint y(2) appears to be this length y(1) appears to be the fill volume of the waste package and y(3) seems to be the mass of the UO2 (or glass)

**End: Friday, April 22, 2005, 4:30 p.m.**

**Start: Sunday, April 24, 6:00 p.m.**

The Visual Fortran code WinDiff was used to determine the code differences between the older version 5.0s and the current version of releaset.f. The differences were documented in a separate report.

**End: Sunday, April 24, 7:30 p.m.**

**Start: Monday, April 25, 2005, 6:00 p.m.**

Check accuracy of documentation of WinDiff results for the two versions of releaset.f compared. Begin writing report on validation testing.

**End: Monday, April 25, 2005, 8:00 p.m.**

**Start: Wednesday, April 27, 2005, 9:00 a.m.**

Continue writing report on validation testing of releaset.f.  
Break for lunch from 12:30 p.m. to 1.00 p.m.  
Break for supper at 4:30 p.m.  
Resume writing report of validation at 9:30 p.m.

**End: Wednesday, April 27, 2005, 10:30 p.m.**

**Start: Thursday, April 28, 2005, 9:00 a.m.**

Move code and data directories for TPA500q so that they have a 10 or less character path. After 1 hour of fiddling with the path for the tpa code a successful run was obtained. The trick was to put the tpa code in a directory with a path less than 18 characters and to include a / at the end of the path specification. The final path that worked was: D:\tpa500q\ Note the final \ is essential.

Complete the first draft of the Validation report and email to Ron Janetzke for comments to 12:30 p.m.

**End: Thursday, April 28, 2005, 12:30 p.m.**

**Start: Thursday, May 5, 2005**

Begin working on Failt lack of convergence. Compile and run Failt base case. The error messages about corrosion potential not converging are received. Begin debugging.

The error was traced to a precision error in the Newton-Raphson convergence loop in the subroutine corrode. The variable zc is calculated by the difference between two the variables. As the two other variables become nearly equal, the precision in the difference is of the order of  $10^{-x}$  of the value of the variables that are almost equal. The exponent x is machine dependent. The following code fixes the problem.

```
C*****Code added by D. LeNeveu May 5,2005
      real*8 relercpass
C*****End of code added by D.LeNeveu
```

C\*\*\*\*\*Code added by D. LeNeveu May 5,2005

relercpass=precision(cpass)

C\*\*\*\*\*End of code added by D.LeNeveu

do 100 i = 1, 10

yko2 = rkhy \* exp(ghy \* yyy1)

curhy = -yko2 \* exp( - xxx1 \* eec)

curmax= -4. \* cfarad \* do2w \* taus \* spor \* cbulko2 /

& max(scalthk, filmthk)

xko2 = rkox \* exp(gox \* yyy1)

curox = -xko2 \* cbulko2 \* exp( - xxx3 \* eec)

& /(1.0 + xxx2\*exp(-xxx3\*eec))

curox = max(curmax, curox)

cathod = -(curox + curhy)

zc = cpass - cathod

C\*\*\*\*\*Code added by D. LeNeveu May 5,2005

c if zc is of the order of the fractional precision in cpass

c then zc cannot get any smaller and is determined by floating point

c truncation error

if (abs(zc) .lt. 10.0\*cpass\*10.0\*\*(-relercpass)) then

zc=0.0

end if

C\*\*\*\*\*End of code added by D. LeNeveu

All three error cases in the file: failt\_all\_cases\_all\_files.zip were successfully run with no lack of convergence messages appearing. – to 1.00 p.m

Since the compiler used at CNWRA uses only Fortran 77 intrinsic functions the intrinsic function precision has been removed. The variable relercpass has been hard coded at 1.e-15 in accordance with iee standards for double precision.

The code now is:

C\*\*\*\*\*Code added by D. LeNeveu May 5,2005

c Set relative precision to one part in ten to the fifteen for iee standards

c for double precision

relercpass=1.D-15

C\*\*\*\*\*End of code added by D.LeNeveu

do 100 i = 1, 10

yko2 = rkhy \* exp(ghy \* yyy1)

curhy = -yko2 \* exp( - xxx1 \* eec)

curmax= -4. \* cfarad \* do2w \* taus \* spor \* cbulko2 /

& max(scalthk, filmthk)

xko2 = rkox \* exp(gox \* yyy1)

curox = -xko2 \* cbulko2 \* exp( - xxx3 \* eec)

& /(1.0 + xxx2\*exp(-xxx3\*eec))

curox = max(curmax, curox)

```

      cathod = -(curox + curhy)
      zc = cpass - cathod
C*****Code added by D. LeNeveu May 5,2005
c   if zc is of the order of the fractional precision in cpass
c   then zc cannot get any smaller and is determined by floating point
c   truncation error
      if (abs(zc) .lt. 10.0D0*relercpass) then
          zc=0.0d0
      end if
C*****End of code added by D. LeNeveu
      dzc = -xxx1 * curhy - xxx3 * curox /
      &      (1 + xxx2 * exp(-xxx3 * eec))
c   if ( abs(zc).le.1.0e-8 ) go to 150
C*****Code added by D. LeNeveu May 5,2005
      if ( abs(zc).le. abs(10.D0*relercpass*eec*dzc) ) go to 150
C*****End of code added by D. LeNeveu
      eec = eec - zc/dzc

100  continue

```

The following code line change was made to ensure that convergence occurs no matter the value of ecc

```

      if ( abs(zc).le. abs(10.D0*relercpass*eec*dzc) ) go to 150

```

to 3:30 p.m.

**End: Thursday, May 5, 2005, 3:30 p.m**

**Start: Sunday, June 5, 2005, 4:00 p.m.**

Begin to examine errors in failt for files sent by R. Janetzke June 3,2005.  
The failt.f code is:

```

File Date:          05/13/05
Release Version:    5.0

```

It was verified that previous modifications to the subroutine corode involving precision are incorporated into version 5.0 above.

The code was compiled with the new version and run with the input files sent by R. Janetzke on June 03/05.

The new version of failt.f had many routines missing. In order to get a successful compilation the following files from tpa500q were added:

```

corrosn.f, integrt.f, linintrp.f, srchpos.f, weldfail.f, zportfdate.f

```

The error case was run successfully and a lack of convergence error was noted.

The lack of convergence was for a special case where the variable rkhy is zero.

rkhy is read determined from rkhy1 and rkhy2 which are data read in.

rkhy1 [Coul/(m<sup>2</sup> yr)] is rate constant for water reduction outer overpack  
rkhy2 [Coul/(m<sup>2</sup> yr)] is rate constant for water reduction inner overpack

When ilayer is 1 rkhy1 is used. Whern ilayer is 2 rkhy2 is used for rkhy.

The convergence loop that is failing is used to ensure the sum of the currents is zero and solves a transcendental equation of the form:

$$C = a \exp(-ef) / (1 + b \exp(-ef)) + y \exp(-ex)$$

for e called eec in the code which is the total current. If y is zero then the above equation can be solved analytically for e as follows:

$$E = 1 / f \{ \ln[(a - cb) / c] \}$$

When rkhy is zero then the variable y above is also zero.

**End: Sunday, June 5, 2005, 8:00 p.m.**

**Start: Monday, June 6, 2005, 8:30 a.m.**

An if block was added to the code to implement the case derived above for rkhy = 0.

This code was tested by determining the value of eec with the new code fragment and increasing the number of iterations to 50 in the existing code. With the number of iterations set to 50 the previous code converged to value of eec of -5.18442934523809. With the code addition for the special case of rkhy = 0 the new code fragment gave an identical value for eec.

The number of iterations was left at 50 to handle other potential slowly converging cases.

An email with the changed code was sent to R. Janetzke to 11.00 a.m.

**End: Monday, June 6, 2005, 11.00 a.m.**

**Start: Friday, June 17, 2005, 10.00 a.m.**

Begin investigating the out of bounds error in module setdis in the code nefmks.f

The archived version of nefmks.f on my disk was run with the input files sent June 16/05 and the array bounds exceeded error was duplicated. The Courant number was changed from 1.5 to 0.15 as recommended previously. The array bounds error did not occur. The results up to the point of the array bounds error were compared for Courant numbers of 0.15 and 1.5

**End: Friday, June 17, 2005, 4.00 p.m.**

**Start: Monday, June 20, 2005, 9:00 a.m.**

A graph on a Excel spreadsheet for the two courant numbers was prepared and sent to Ron Janetzke. The archived code for allocatable arrays for FP was run

with a Courant number of 1.5. The F array overflowed its bounds. Work was begun to fix this.

**End: Monday, June 20, 2005, 6:00 p.m.**

**Start: Tuesday, June 21, 2005, 1:00 p.m.**

Discovered array bounds exceeded for the array F in SETDIS. F is passed in the common block XPORTA to other routines. It appears not to be used anywhere except in SETDIS if IOPT(24) not equal to zero to write out discharge factors and in subroutine trnspt at line 8590

```
SUM = SUM + F(I)*RHO(NTX-K+1,IRP)
```

**End: Tuesday, June 21, 2005, 6:00 p.m.**

**Start: Wednesday, June 22, 2005, 8:00 a.m.**

The variable F was removed from the common block XPORTA and passed through the module SVBF. F was made allocatable and allocated in the subroutine SETDIS. A trail run was done to completion. The execution time was 49 minutes 44 seconds on a Pentium two 2.8Mhz processor. Efforts commenced to decrease CPU run time.

**End: Wednesday, June 22, 2005, 6:00 p.m.**

**Start: Thursday, June 23, 2005, 8:30 a.m.**

Efforts continue to decrease CPU. The large value of NTX in subroutine DXDT is investigated. Changing the value of CNM to a lower value of 0.15 in the following code in the subroutine DXDT speeds execution

```
DO 200 IR = 1,NOISO
  TM = TRAVT(IR+MFNDX)
  IF(TM .LE. C1*THALF(IR)) THEN
    IF(TM .LE. TUB - TRLSE) THEN
      DO 190 K = 1,NPATH
        IF(KFDX(K) .EQ. 0) THEN
          DXK(K) = VELISO(K,IR)*DT/CNM
          NXK = INT(PATH(MPATH(K))/DXK(K) + 1.)
          DXK(K) = PATH(MPATH(K))/FLOAT(NXK)
          IF(DXK(K) .LT. DX(K)) THEN
            DX(K) = DXK(K)
            NX(K) = NXK
          ENDIF
        ENDIF
      CONTINUE
    ENDIF
  ENDIF
200 CONTINUE
```

This was not implemented because other inadvertent consequences of this change might result.

**End: Thursday, June 23, 2005, 3:00 p.m.**

Entries into Scientific Notebook #612-18E for pages 1-44 have been made by Dennis M. LeNeveu.

No original text entered into this Scientific Notebook has been removed.

AR White, manager February 12, 2010.

I have reviewed this scientific notebook and find it in compliance with QAP-001. There is sufficient information regarding methods used for conducting tests, acquiring and analyzing data so that another qualified individual could repeat the activity.

AR White, Manager 2/12/2010