

1/195

SOFTWARE RELEASE NOTICE

01. SRN Number: EBS-SRN-142		
02. Project Title: EBSPAC Development		Project No. 20-5708-572
03. SRN Title: EBSPAC Version 1.0		
04. Originator/Requestor: QA		Date: 1/9/97
05. Summary of Actions		
<input checked="" type="checkbox"/> Release of new software <input type="checkbox"/> Release of modified software: <input type="checkbox"/> Enhancements made <input type="checkbox"/> Corrections made <input type="checkbox"/> Change of access software <input checked="" type="checkbox"/> Software Retirement <i>VJ 12/10/01</i>		
06. Persons Authorized Access		
Name	RO/RW	A/C/D
Sitakanta Mohanty Dennis Vinson	RW RW	
07. Element Manager Approval: <i>N. Sridhar</i>		Date: 1/9/98 <i>7/8</i>
08. Remarks: Future versions of EBSPAC will be managed in accordance with TOP-018 by NRC Staff		

9/19/95

SOFTWARE RELEASE NOTICE

01. SRN Number: EBS-SRN-142		
02. Project Title: EBSPAC Development		Project No. 20-5708-572
03. SRN Title: EBSPAC Version 1.0		
04. Originator/Requestor: QA		Date: 1/9/97
05. Summary of Actions		
<input checked="" type="checkbox"/> Release of new software		
<input type="checkbox"/> Release of modified software:		
<input type="checkbox"/> Enhancements made		
<input type="checkbox"/> Corrections made		
<input type="checkbox"/> Change of access software		
<input type="checkbox"/> Software Retirement		
06. Persons Authorized Access		
Name	RO/RW	A/C/D
Sitakanta Mohanty Dennis Vinson	RW RW	
07. Element Manager Approval: <i>N. Sridhar</i>		Date: 1/9/97 <i>7/8</i>
08. Remarks:		
Future versions of EBSPAC will be managed in accordance with TOP-018 by NRC Staff		

10/195

SOFTWARE SUMMARY FORM

01. Summary Date: 1/8/97		02. Summary prepared by (Name and phone) Cathy Garcia (210) 522-5471		03. Summary Action: None	
04. Software Date: 1/6/97		05. Short Title: EBSPAC Version 1.0			
6. Software Title: Engineering Barrier System Assessment Code				07. Internal Software ID: None	
08. Software Type: <input type="checkbox"/> Automated Data System <input checked="" type="checkbox"/> Computer Program <input type="checkbox"/> Subroutine/Module		09. Processing Mode: <input type="checkbox"/> Interactive <input type="checkbox"/> Batch <input checked="" type="checkbox"/> Combination		10. APPLICATION AREA a. General: <input checked="" type="checkbox"/> Scientific/Engineering <input type="checkbox"/> Auxiliary Analyses <input checked="" type="checkbox"/> Total System PA <input checked="" type="checkbox"/> Subsystem PA <input type="checkbox"/> Other b. Specific:	
11. Submitting Organization and Address: Southwest Research Institute Center for Nuclear Waste Regulatory Analyses 6220 Culebra Road San Antonio, TX 782238-5166			12. Technical Contact(s) and Phone: Sitakanta Mohanty (210) 522-5185		
13. Narrative: EBSPAC Predicts Container Life and Radionuclide Releases from the EBS.					
14. Computer Platform SUN SPARC		15. Computer Operating System: SUN OS		16. Programming Language(s): FORTRAN 77	
17. Number of Source Program Statements: 5,947		18. Computer Memory Requirements: Varies		19. Tape Drives: None	
20. Disk/Drum Units: Disk		21. Graphics: None		22. Other Operational Requirements None	
23. Software Availability: <input checked="" type="checkbox"/> Available <input type="checkbox"/> Limited <input type="checkbox"/> In-House ONLY			24. Documentation Availability: <input checked="" type="checkbox"/> Available <input type="checkbox"/> Inadequate <input type="checkbox"/> In-House ONLY		
Software Custodian: <i>Sitakanta Mohanty</i>				Date: <i>1/14/97</i>	

4 / 145

CENTER FOR NUCLEAR WASTE REGULATORY ANALYSES

DESIGN VERIFICATION REPORT FOR CNWRA SOFTWARE: EBSPAC V. 1.0

May 15, 1997

EBSPAC VERSION 1.0

1. Scientific Notebook Documentation Development: CNWRA Electronic Scientific Notebook number 170 was verified; it is in the QA Records Room and it documents the EBSPAC Version 1.0 software development.
2. Programming Language: ANSI Standard FORTRAN 77 confirmed by the Software Custodian.
3. Internal Documentation: On 1/7/97, B. Mabrito reviewed portions of the EBSPAC Version 1.0 software on one of the CNWRA platforms. EBSPAC_fail and other files were reviewed and there were clear and numerous internal documentation comments meeting the requirements of TOP-018 Section 5.4.4. Printouts of screens were made and are part of the EBSPAC Version 1.0 folder documentation.
4. Software Labels and Data
 - a. Header Data and Format: EBSPAC Version 1.0 header data and the format were compared against TOP-018 (Revision 5 dated 4/3/97) Section 5.4.6 and found acceptable. Printout sheets of the header data and format are in the EBSPAC Version 1.0 scientific and engineering software folder in the QA Records Room.
 - b. NRC Data: EBSPAC Version 1.0 header NRC data and the format were compared against TOP-018, Section 5.4.6, third bullet and found generally acceptable.
 - c. Source Code Header: EBSPAC Version 1.0 header data was compared to TOP-018 Section 5.4.6, fourth bullet, and found generally acceptable.
5. Unique Run Identification: At the top of output files a unique identifier is created on the print out. For instance, a page of the "Calculation of Waste Package Failure Time" file printed and kept in the EBSPAC Version 1.0 QA Records Folder showed the following: "Version 1.0 Mon Jan 6 17:36:57 1997". This file was created on that date/time and it fully meets the unique run identification requirements of TOP-018, Section 5.4.5.
6. Software Analysis and Results
 - a. Analysis: The plusFORT programming tool was utilized as the software analysis tool on EBSPAC Version 1.0 during its development. Files of SPAG.fig were used during the development of the code. SPAG is a multi-purpose tool for analyzing and improving FORTRAN programs. It combines restructuring and re-formatting with both static and dynamic analysis in a single package. SPAG has the ability to: (i) restructure FORTRAN 66 to FORTRAN 77, (ii) identify and remove unused variables and code fragments, (iii) insert explicit type declarations for certain typed variables, (iv) convert back and forth between FORTRAN 77 and code with FORTRAN 90 extensions, (v) make re-formatted code clearer by using upper and lower case to distinguish

different types of symbols, (vi) can insert probes to perform a dynamic analysis of programs as they run, (vii) insert timing probes to identify computational hot-spots, and (ix) write out symbol table files for reference. The plusFORTH reference manual, Revision B, is in Ron Janetzke's office as of 5/15/97.

b. Analysis Report: Since the software analysis of EBSPAC Version 1.0 was conducted incrementally during development and few copies of documentation were made, there is a minimum of objective evidence available. However, several selected pages have been printed out and are in the EBSPAC Version 1.0 scientific and engineering software QA Folder and they frequently reference "End of declarations inserted by SPAG" throughout the document (and have been highlighted with a blue highlighter in that folder).

c. Resolution of Comments: The developer(s) reviewed all of the warning and error messages provided by SPAG and addressed the most critical problem statements. According to developer S. Mohanty, the code analyzing software are very different from each other. Therefore, using a different analyzer, such as FOR_STUDY, may give warnings not found in SPAG.



CNWRA Software Developer Date
(Sitakanta Mohanty) 5/15/97



CNWRA Software Custodian Date
(BRUCE MASRIGO) 5/15/97

original to: Software Folder
cc: CNWRA Software Developer

```
-rwxrwxrwx 1 root daemon 1624 20 14:52 ebspac.nuc  
-rwxrwxrwx 1 root daemon 129 Jan 20 14:52 ebspacru  
    UNIX name: ebspacrun  
-rwxrwxrwx 1 root daemon 3124 Jan 20 14:59 ex_fail.inp  
    UNIX name: example_fail.inp  
-rwxrwxrwx 1 root daemon 2330 Jan 20 14:52 ex_relea.inp  
    UNIX name: example_release.inp  
-rwxrwxrwx 1 root daemon 68425 Jan 20 14:53 fail.f  
-rwxrwxrwx 1 root daemon 149 Jan 20 14:53 floobs.dat  
-rwxrwxrwx 1 root daemon 139251 Jan 20 14:53 release.f  
-rwxrwxrwx 1 root daemon 754061 Jan 20 15:17 tefkti.inp
```

EBSPEC VERIFICATION

EBSPEC code is one of its kind and therefore no other code can be used for an exhaustive verification of this code. Although EBSPEC has borrowed models from previously established codes, such as, SOTEC or/and SCCEX, these two codes can not be used for such verification testings because of many modifications to the models. Therefore, various intuitive verifications are conducted to demonstrate that the code behaves in some limiting cases as one would expect.

TEST 1

In the first test, the flow rate into the WP was changed. The following is the result from ebspac_release.f when all parameters are kept at the default values. In this data set summary results are presented for this standard run. Note that xno loss is the radioactivity that would have been in the WP had there not been any release. Amwp represents the amount of radionuclide in the wp water at the end of 10000 years. Xmass represents the amount of radionuclides that have come out of the WP. Fracleft is the fraction of the radionuclides originally present in the wp is still in the wp. In other words, fracleft = 0.98 implies that 98% of that particular radionuclide has already come out. A value of 0.02 indicates that the total amount of that particular radionuclide present in the wp has come out and a maximum error is nearly 2%.

$$Q_{in} = 2.086E-3 * 1.0 \text{ m}^3/\text{yr}$$

cumulative release [ci/cell] by 10000.0 years					
namall	halflife(yr)	xno loss(ci)	amwp(ci)	xmass(ci)	fracleft
CM246	0.473E+04	0.195E+02	0.191E+02	0.746E+00	0.96
PU242	0.386E+06	0.514E+04	0.523E+04	0.645E+00	1.00
U238	0.447E+10	0.104E+04	0.106E+04	0.671E-01	1.00
CM245	0.850E+04	0.182E+03	0.179E+03	0.696E+01	0.96
AM241	0.432E+03	0.193E+03	0.190E+03	0.639E+01	0.97
NP237	0.214E+07	0.202E+04	0.202E+04	0.372E+02	0.98
AM243	0.738E+04	0.198E+05	0.202E+05	0.842E+01	1.00
PU239	0.241E+05	0.763E+06	0.777E+06	0.962E+02	1.00
PU240	0.654E+04	0.575E+06	0.586E+06	0.727E+02	1.00
U236	0.234E+08	0.109E+04	0.111E+04	0.739E-01	1.00
U234	0.245E+06	0.600E+04	0.610E+04	0.387E+00	1.00
TH230	0.770E+05	0.524E+03	0.533E+03	0.116E-01	1.00
RA226	0.160E+04	0.409E+03	0.410E+03	0.543E+01	0.99
PB210	0.223E+02	0.407E+03	0.342E+03	0.718E+02	0.82
CS135	0.230E+07	0.114E+04	0.907E+00	0.116E+04	-0.02
I129	0.157E+08	0.969E+02	0.625E-02	0.986E+02	-0.02
TC99	0.213E+06	0.389E+05	0.230E+01	0.395E+05	-0.02
NI59	0.800E+05	0.107E+05	0.166E+04	0.921E+04	0.14
C14	0.573E+04	0.150E+04	0.365E-02	0.152E+04	-0.02
SE79	0.650E+05	0.112E+04	0.114E+04	0.112E-02	1.00
NB94	0.203E+05	0.185E+04	0.301E+00	0.188E+04	-0.02
zeroth yr inventory for liquid releases:				10040.5	
zeroth yr inventory for gaseous releases:				8107.1	

127/155

1000-yr c14 inv. for liq. rel [ci/cell]:	10036.7
1000-yr c14 inv. for gaseous rel. [ci/cell]:	8104.1

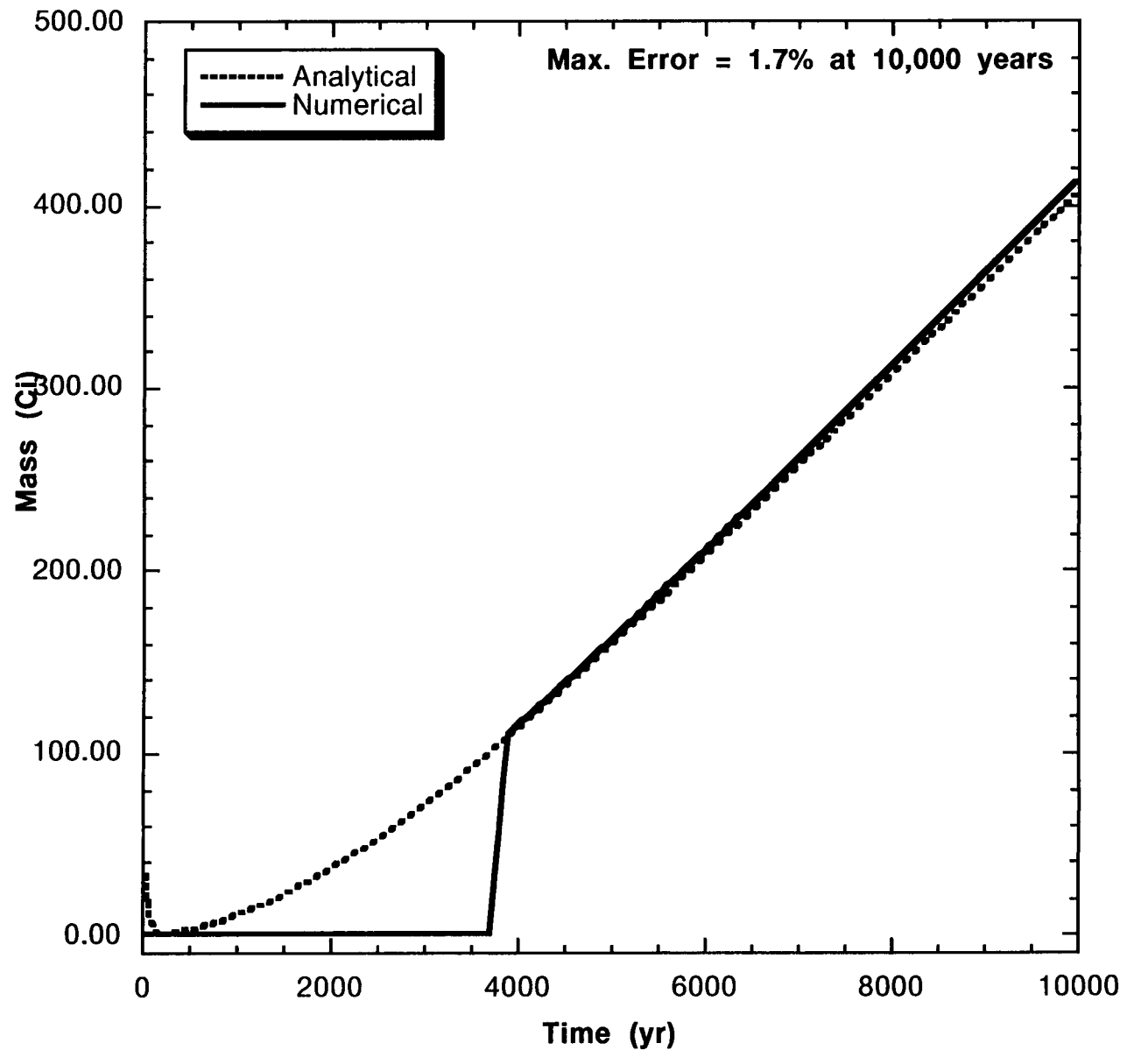
The results presented below are obtained by drastically increasing the flow rate of water into the wp so that if the SF leaching is fast, then all material in the WP must come out in 10,000 years. AS seen in the last column of the output table, all radionuclides have come out of the WP and a maximum error of 1% is seen.

$Q_{in} = 2.086E-3 * 1000000.0 \text{ m}^3/\text{yr}$

cumulative release [ci/cell] by 10000.0 years					
namall	halflife(yr)	xnoloss(ci)	amwp(ci)	xmass(ci)	fracleft
CM246	0.473E+04	0.195E+02	0.587E-06	0.197E+02	-0.01
PU242	0.386E+06	0.514E+04	0.636E-02	0.519E+04	-0.01
U238	0.447E+10	0.104E+04	0.113E-03	0.105E+04	-0.01
CM245	0.850E+04	0.182E+03	0.291E-04	0.184E+03	-0.01
AM241	0.432E+03	0.193E+03	0.259E-06	0.195E+03	-0.01
NP237	0.214E+07	0.202E+04	0.240E-02	0.204E+04	-0.01
AM243	0.738E+04	0.198E+05	0.171E-02	0.200E+05	-0.01
PU239	0.241E+05	0.763E+06	0.470E+00	0.771E+06	-0.01
PU240	0.654E+04	0.575E+06	0.491E-01	0.581E+06	-0.01
U236	0.234E+08	0.109E+04	0.102E-03	0.110E+04	-0.01
U234	0.245E+06	0.600E+04	0.608E-03	0.606E+04	-0.01
TH230	0.770E+05	0.524E+03	0.201E-03	0.529E+03	-0.01
RA226	0.160E+04	0.409E+03	0.485E-06	0.413E+03	-0.01
PB210	0.223E+02	0.407E+03	0.716E-07	0.411E+03	-0.01
CS135	0.230E+07	0.114E+04	0.109E-02	0.115E+04	-0.01
I129	0.157E+08	0.969E+02	0.653E-05	0.979E+02	-0.01
TC99	0.213E+06	0.389E+05	0.241E-02	0.393E+05	-0.01
NI59	0.800E+05	0.107E+05	0.931E-03	0.108E+05	-0.01
C14	0.573E+04	0.150E+04	0.453E-05	0.151E+04	-0.01
SE79	0.650E+05	0.112E+04	0.574E-04	0.113E+04	-0.01
NB94	0.203E+05	0.185E+04	0.316E-03	0.187E+04	-0.01
zeroth yr inventory for liquid releases:				10040.5	
zeroth yr inventory for gaseous releases:				8107.1	
1000-yr c14 inv. for liq. rel [ci/cell]:				10036.7	
1000-yr c14 inv. for gaseous rel. [ci/cell]:				8104.1	

TEST 2

In the second test, the numerical calculations used in the code were verified. This test involves comparing the numerical computation of the amount of radionuclides at the end of 10,000 years and the results from the analytical solution by using Bateman equation for radionuclide chains. The numerical method, and the Bateman equation, both are used in the code. The code is faked so that no release takes place from the WP. In that case, if SF matrix leaching is fast, then all materials in the wet region in the solid portion will come to the water in the WP. Therefore the amount of radionuclide in the solid if there were no water (calculated by using analytical Bateman equation) will be same as the amount in the liquid if that material has dissolved and has come to the liquid where the amount is calculated by using the numerical method. Comparison of the mass of PB210 is presented in the figure below. In this figure, the mass of PB210 computed by Bateman equation, and the numerical method are compared in which the dotted line represents the result from the Bateman equation. The solid line shows that there is no dissolution of the solid mass until after nearly 3600 years. Then note that the solid line rises very fast to match up with the dotted curve, suggesting that the leaching time is very short (of the order of 2000 years). After this leaching period, one would expect the two curves until 10,000 years to be identical. This figure show an excellent match between these two curves and a maximum error of 1.7% is seen by the end of 10,000 years.



MAIN INPUT DATA FILES CORRESPONDING TO THE CODE VERIFICATION

data file 1: EXAMPLE_FAIL.INP

```

\example input file for ebspac_fail
|
\simulation time
10000.          ! tend: simulation time length [yr]
\              ! when iflag=1 (defined later)
\
5.682, 1.802   ! wplen,wpdia: wp length and diameter [m]
0.1, 0.02     ! cthick1,cthick2: wp layers 1&2 thicknesses [m]
|
\choose source of temperature data
2              ! iflag 1: emp. equation, 2: tab.data
1              ! nset (temp.-rel hum. relationship to use
49.9999999    ! timintv (used when iflag=2)
|
\other temperature parameters
0.            ! age of fuel (not used in this version)
|
\Dry oxidation of wp outer overpack
5.            ! grainr: metal grain radius [micrometer]
25            ! nseries (terms in the infinite series)
0.7e-3       ! gbthick [micrometer]
1.e-2        ! constant: used in the dry oxidation equn.
|
\evaporation-condensation
0.65         ! humdc: critical relative humidity
2.e-3        ! filmthk: thickness of water film [m]
97.          ! ctemp: boiling point of water [C]
|
\Corrosion Parameters(Ep: pitting potential [mV]; Erp: repassivation potential
[mV])
-584.8       ! xipto: outer overpack Ep intercept
3.92        ! pttemo: temp. coef. of outer overpack Ep intercept
-24.5       ! slpto: outer overpack Ep slope
-1.1        ! slpttemo: temp. coef. of outer overpack Ep slope
-620.3      ! xirpo: outer overpack Erp intercept
0.47        ! rptemo: temp. coef. of outer overpack Erp intercept
-95.2       ! slrpo: outer overpack Erp slope
0.88        ! slrptemo: temp. coef. of outer overpack Erp slope
200.        ! xipti: inner overpack Ep intercept
0.          ! pttemi: temp. coef. of inner overpack Ep intercept
-240.       ! slpti: inner overpack Ep slope
0.          ! slpttemi: temp. coef. of inner overpack Erp slope
422.8       ! xirpi: inner overpack Erp intercept
-4.1        ! rptemi: temp. coef. of inner overpack Erp intercept
-64.        ! slrpi: inner overpack Erp slope

```

```

-0.80          ! slrptemi: temp. coef. of inner overpack Erp slope
0.75, 0.5      ! betaox1, betahy1: beta kinetics parameters
\             for oxygen and water for WP outer overpack
0.75, 0.5      ! betaox2, betahy2: beta kinetics parameters for
\             oxygen and water for WP inner overpack
3.80e12, 1.6e-1 ! rkox1 [c*m/y/mol], rkhy1 [c/m2/yr]
37300., 25000. ! gox1 [J/mol], ghy1 [J/mol]
3.0e10, 3.2     ! rkox2 [c*m/y/m], rkhy2 [c/m2/yr]
40000., 25000. ! gox2 [J/mol], ghy2 [J/mol]
3.15e5, 0.0, 0.0 ! aa(1,1) [C/m2/yr], aa(1,2), aa(1,3)
6.30e4, 0.0, 0.0 ! aa(2,1) [C/m2/yr], aa(2,2), aa(2,3)
-0.46         ! eexpt: in Vshe
8.66e-3       ! rcoef:
0.45         ! rexpont:
2.05e-4       ! crate2: m/yr corrosion rate
0.0          ! xcouple, a fractional coupling strength
0.0          ! xread
3.e-1        ! clconc: chloride concentration [mol/L]
3.e-4        ! clcrit1: crit. chloride conc. for 1st layer [mol/L]
2.e-3        ! clcrit2: crit. chloride conc. for 2nd layer [mol/L]
100.         ! cfactor: chloride dilution factor
9.0          ! refph: reference pH
1.0, 1.0     ! taus:deposit tortuosity,spor:deposit porosity
|
\Runge-kutta control parameters
1.e-3, 1.e0   ! dtini, dtmax
1.e-2, 1.e-30 ! errrel (same as eps), errabs (same as tiny)
|
\end
////2000     ! nhista (used when iflag=1)

```

data file 2: RELEASE.INP

```

\input data file for ebspac release code: ebspac_release.f
|
\Cell information
2335.          ! xcon: number of WP
1000          ! iscon: # of WP in scenario failure
4500.         ! sftime: time of scenario failure [y]
0.00         ! deffrac: defective fraction of WPs/cell
|
\WP information
1.421, 4.572          ! dint1: wp ID, xint1: internal length [m]
4.83                ! xvol: wp internal vol[m3]
|
\Thermal data
'temphumd.dat'      ! temfil: temp. file (output from ebspac_fail.f)
97.                ! ctemp: BP of water at atm. condition [C]
|
\Flow parameters
'floebes.dat'      ! hydfil: flow parameters file
7.                ! funnel: funnel factor
|
\SF materials
\
2800.            ! amassc: SF mass [kg]
250., 10000., 1.0E-08, 0.5 ! fuewt,fueden,fuedif,fuepor
\
\Fuel leaching
2, 9.0, 0.2, 2.e-3, 5.e-1 ! imodel,phvalue,oxgnovpr [atm],cco3 [mol/L],wetfrac
|
\ Radionuclide inventory
'ebspac.nuc'      ! elefil: nuclide names, halflife,inventory
|
\C-14 generation
0.5e-3          ! r0z: initial radius of SF particle [m]
37.            ! rhou: density of SF [kg mole/m^3]
1.e-5          ! rlz: radius of the SF grain [m]
1.e-6          ! r2z: subgrain fragment radius after trans. frac. [m]
6.1e-4         ! thclad: thickness of cladding [m]
7.2e-4         ! cfuel: c-14 [ci] /kg SF
4.89e-4        ! czmetal: c-14 [ci] /kg SF in Zry clad & other
metals
2.48e-5        ! czoxide: c-14/kg SF in initial Zry oxide & crud
6.2e-6         ! cgap: c-14/kg SF in grain and gap
|
\NUMERICAL
\
\Grids
10, 10          ! imax,jmax: # of grid nodes in i,j directions
\X-COOR of grid nodes

```

```

.1, .5, 1.0, 2.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0
\Y-COOR of grid nodes
.1, .5, 1.0, 2.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0
\ Note: zones apply to the above grid, zones are the same for all cell
\ZONES
4          ! nzones: no. of zones for material types
1   1   1   1   1   ! iz,ib,jb,ie,je: for zone 1
2   2   1   2   1   ! iz,ib,jb,ie,je: for zone 2
3   3   1   3   1   ! iz,ib,jb,ie,je: for zone 3
4   4   1   10  1   ! iz,ib,jb,ie,je: for zone 4
|
\Rock parameters
0.14, 0.34, 0.34, 0.34      ! rpor(1..nzones): rock porosity
|
\ Radionuclide transport
5.6e-5, 5.6e-5, 5.6e-5, 5.6e-5      ! rdifff(1..nzones):diffusion coef. [m2/yr]
5.0          ! driftdia [m]
|
\ Solution algorithm control parameters (Runge-Kutta)
25., 0.0, 10.          ! dtinit [yr], dtmin [yr], dtmax [yr]
1.0e-2, 1.0e-10      ! eps, tiny
|
\output parameters
200          ! nbt: number of time intervals for output
|
\END

```

data file 3: FLOEBS.DAT

```

scratchy1:/u01/mohanty> more floebs.dat
1.          ! flowfactr: flow factor
0.          2.086E-03      ! time(yr) and infiltration(m^3/yr)
10000.      2.086E-03      ! time(yr) and infiltration(m^3/yr)

```

REQUIREMENT DESCRIPTION

169/195

**SOFTWARE REQUIREMENTS DESCRIPTION
FOR THE ENGINEERED BARRIER SYSTEM
PERFORMANCE ASSESSMENT CODE (EBSPAC)**

(Revised June 14, 1996)

by
**Sitakanta Mohanty
Tae Ahn
Peter C. Lichtner
Ronald W. Janetzke
Gustavo A. Cragolino**

**Center for Nuclear Waste Regulatory Analyses
San Antonio, Texas**

Version 1.0

Approved by:



**Narasi Sridhar, Manager, EBS Element
CNWRA**

1 INTRODUCTION

This Software Requirements Description (SRD) document is the first stage in the development of the Engineered Barrier System Performance Assessment Code (EBSPAC), a numerical model for the evaluation of container life and source term. EBSPAC will be used as the source term module in the Nuclear Regulatory Commission (NRC) Total System Performance Assessment (TPA) code.

2 SOFTWARE REQUIREMENTS DESCRIPTION: EBSPAC

This SRD briefly outlines the software function, technical bases, computational approach, program flow and user interface, hardware and software requirements, graphics, and pre- and post-processors that are relevant to the development and use of the code EBSPAC.

Because EBSPAC will be used as a module in the TPA code that will execute hundreds of Monte Carlo runs, the EBSPAC code must be based on simplified models that will allow fast computation. As a rule of thumb, the design will be such that the average execution time for a single EBSPAC run will be in the neighborhood of 22 seconds on CRAY-YMP with eight processors. With an execution time scale factor of approximately 10, this 22 cray-seconds will be equivalent to 220 seconds on a UNIX IPX machine. EBSPAC will be developed and will be placed under Center for Nuclear Waste Regulatory Analyses (CNWRA) configuration management as required by the CNWRA Technical Operating Procedure (TOP)-018.

2.1 SOFTWARE FUNCTION

The code EBSPAC will be used to provide the source term to the TPA code. The code will provide waste package failure time and the release rate of radionuclides from the engineered barrier system (EBS). The code may eventually be used to evaluate the issue of criticality in conjunction with the U.S. Department of Energy (DOE) reference waste package design proposed in the TSPA-95 report (U.S. Department of Energy, 1994, 1995). However, criticality calculations are not considered in the first version of EBSPAC. Additionally, the multipurpose canister (MPC) performance is ignored in this version in correspondence with TSPA-95 calculations.

The first iteration of EBSPAC will be based on the SCCEX code modified to be applicable to the reference design. EBSPAC will not perform calculations to address "substantially complete containment" because of a possible rule change which may eliminate subsystem requirements.

2.2 TECHNICAL BASIS: PHYSICAL AND MATHEMATICAL MODELS

The code EBSPAC will consist of a number of modules based on a simplified physical description of the underlying processes which govern the failure of the waste package and ultimate release of radionuclides. Figures 1 and 2 show a simplified conceptualization around which the EBSPAC code will be built. This includes the horizontal emplacement of the waste package instead of the vertical emplacement implemented in SCCEX and SOTEC (Cragnolino et al., 1994; Sagar et al., 1992) codes. The main modules to be included in the EBSPAC code are briefly described below. The flow diagram in Figure 3 shows various components of the EBSPAC code.

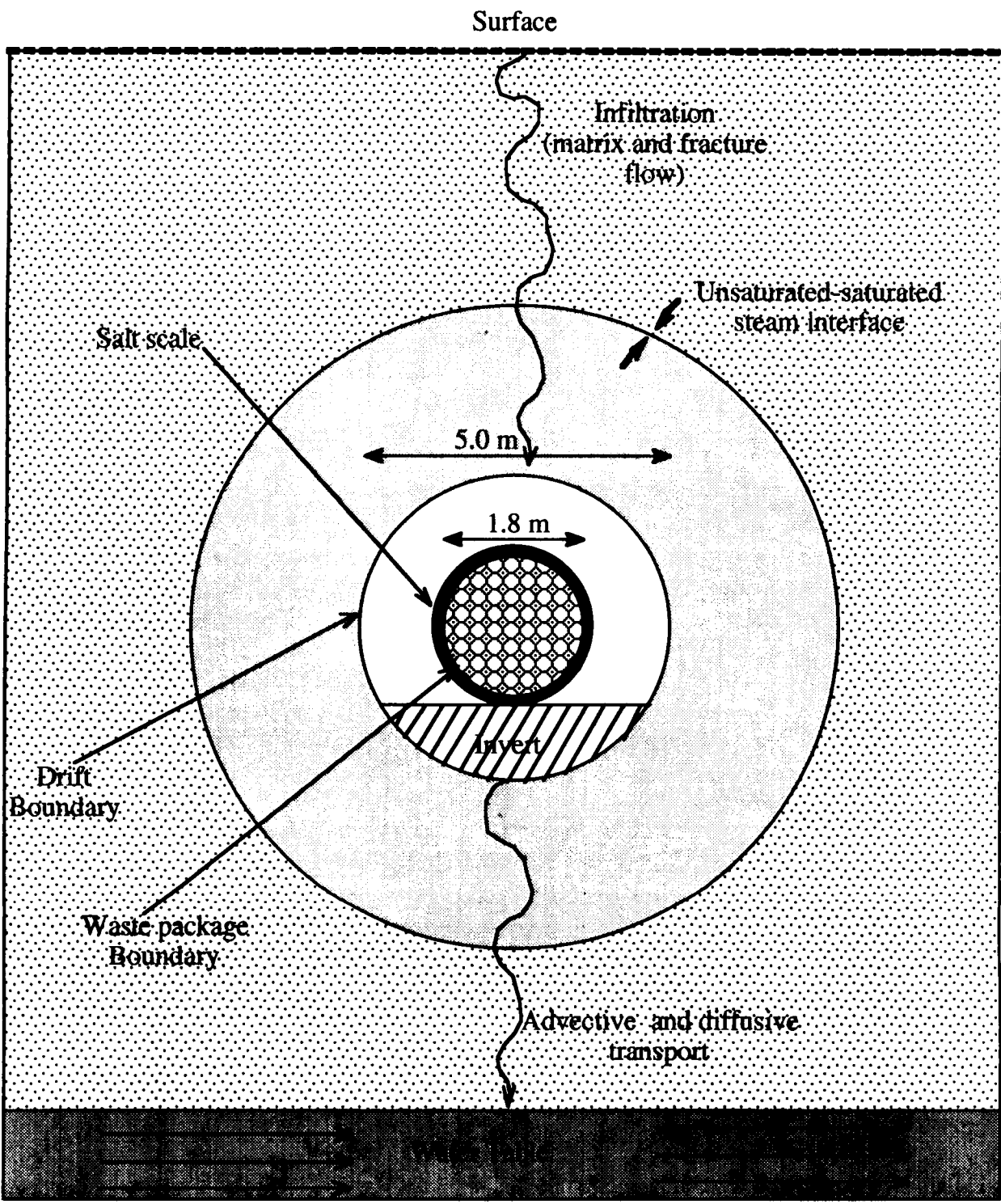


Figure 1. Conceptualization of the radionuclide release from waste packages emplaced in a horizontal drift

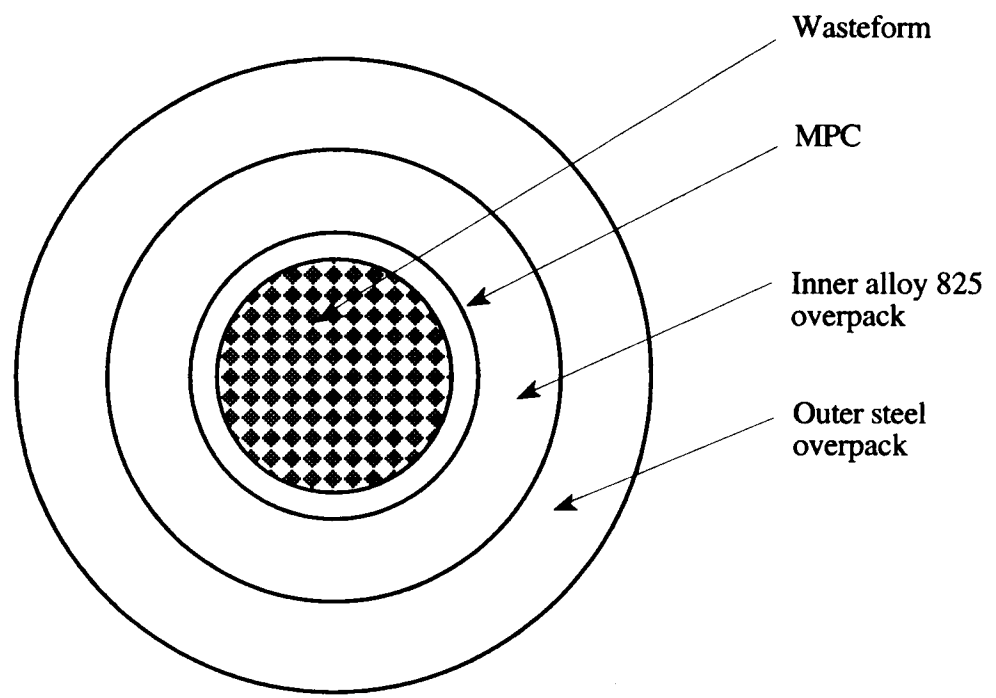


Figure 2. Schematic showing the magnified view of the cross-section of a waste package

2.2.1 Thermal Module

The thermal module will provide the temperature distribution as a function of time and position within the EBS as well as the surrounding geosphere. The thermal model will initially be based on conduction as the only heat-transfer mechanism. This module will reflect the geometry adopted in the reference design. Temperature distribution data will be stored in a file at various times, and, therefore, this module will not be called from within EBSPAC. External temperature calculation is justifiable because conduction is considered to be the dominant heat transfer mechanism and also the amount of heat to be generated by the source in the future is known deterministically. Other heat-transfer modes (convection and radiation) may be considered in future versions of EBSPAC. For example, in the absence of backfill, radiation may become an important heat transfer mechanism.

2.2.2 Environmental Module

The environmental module processes input from external data sources. These data are to include ground water composition, such as concentrations of species Na^+ , Cl^- , HCO_3^- , and O_2 and pH among others, provided as functions of time. In addition, relative humidity and liquid saturation will be provided as functions of time. MULTIFLO will be used for generating fluid composition and

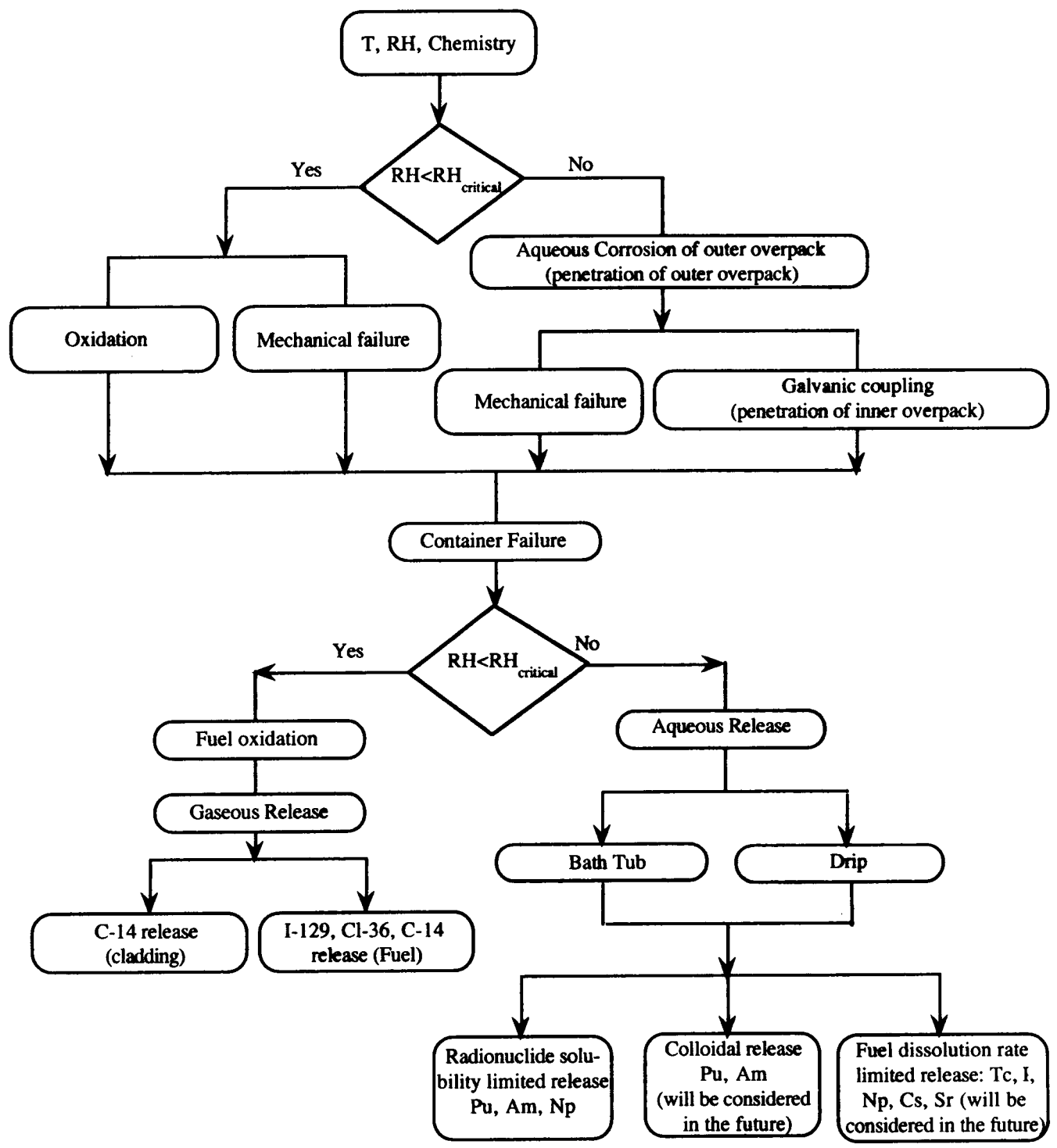


Figure 3. Flow chart showing the logic sequence of Engineered Barrier System Performance Assessment Code

Revised 6/14/96

species concentration data. The environmental conditions are expected to be affected significantly only by a few scenarios, including different mechanism or process of water movement (e.g., fracture flow). Calculating the environmental conditions outside the EBSPAC leaves the user with the option for using MULTIFLO only selectively. The MULTIFLO code meets the quality assurance requirements of TOP-018.

Because the onset of aqueous corrosion is dictated by the waste package surface conditions, especially the presence of a water film, the time at which the surface will be covered by a water film (i.e., the surface wetting time and area wetted) will be determined by using either the vapor-pressure gradient model (Walton et al., 1995) or the critical relative humidity and the temperature approach adopted by the DOE.

The environmental module will provide the amount of water reaching the waste package surface which will be used in determining the wetted area of this surface.

2.2.3 Corrosion Module

The corrosion module will be changed to address the DOE reference design. The basic equations (Sridhar et al., 1993; Cragolino et al., 1994) will follow the SCCEX code with appropriate modifications to represent the DOE reference waste package materials and design. The SCCEX corrosion model includes:

- Oxidation
- General corrosion
- Localized corrosion (pitting and crevice corrosion)
- Stress corrosion cracking
- Galvanic corrosion

The corrosion process at any given time is dictated by the corrosion potential and the corresponding critical potential associated with the process. Therefore, solution chemistry and relative humidity data read in by the environmental module will also be used in this module. Galvanic corrosion becomes important only after the outer overpack is penetrated. Hence, the corrosion module will consist of two stages—that for the outer overpack and that for the inner overpack.

2.2.4 Mechanical Failure Module

The applicability of the mechanical failure models used in SCCEX, namely, yielding, buckling, and fracture models, will be evaluated from the point of view of horizontal emplacement. For the new design, fracture failure as a result of the thermal embrittlement of the steel overpack is an important failure mode to be considered. The mechanical failure model should take into account material instability due to long-term thermal exposure. A simplified calculation of waste package temperature (thermal model) and changes in mechanical properties of overpack material will be considered in EBSPAC. For the first version, priority will be given to the fracture failure.

2.2.5 Release Rate Module

Gaseous Release

For the sake of simplicity, the gaseous release will be treated separately from the liquid release. The gaseous release will include three release modes:

- (1) Prompt release: This will account for the loosely-held C-14 and I-129 on the cladding and cladding/fuel gap and grain boundaries that will be released immediately following the container failure (within the first 100 years after the failure).
- (2) Oxidation of cladding: The zirconium cladding, which is a strong oxygen getter, will make this oxygen available for carbon oxidation and release of C-14. Other buried irradiated structural metal will also undergo the same process.
- (3) Oxidation of UO₂: This will account for the largest amount of the inventory of the C-14 which is contained in the form of solid solutions, elemental carbon, carbides, and oxycarbides. Dry oxidation of UO₂ will also change its crystal structure, resulting in volume expansion and fracture.

These three processes have already been implemented in SOTEC (Sagar et al., 1992). These models will be implemented in EBSPAC considering more recent discussions (Ahn, 1994) where appropriate. Improved calculation efficiency will be considered because of the possibility of extending calculations to beyond 10,000 years.

Liquid Release

The liquid release will account for the radionuclides released from the fuel, production from radioactive decay of another radionuclide, losses from radioactive decay, advective flow, and diffusion. The liquid release of radionuclide from the fuel will include only congruent release option with the assumption that the radionuclides are contained in the matrix of the UO₂. The advective mass transfer will be a simple mass balance (i.e., amount out = concentration × water flow rate out of the container). The diffusive mass transfer will be assumed to be taking place from a spherical source as implemented in SOTEC. The diffusion model will consider radioactive decay. Radioactive decay chains are not considered, however. Attention will be given to improving calculation efficiency because of the possibility of extending these calculations to above 10,000 years.

During a later phase of EBSPAC code development activities, an attempt will be made to include the effects of uranyl silicates, potential secondary alteration products of the spent fuel in the expected high-silica concentration groundwater likely to encounter the waste package. Thermodynamic data is only poorly known at this time for these minerals. However, because of their large molar volumes, resulting in large positive volume changes in replacement reactions of spent fuel, formation of these alteration products could play an important role in dissolution of spent fuel.

A simplified model for formation and transport of colloids may be considered at a later phase of code development.

2.2.6 Inventory Model

The inventory in the failed waste package will be monitored by performing mass balance calculation at specified time steps. The mass balance calculation will include depletion due to decay, generation of daughter products, and mass depletion due to diffusive and advective releases.

Although SCCEX conducted performance calculations up to 1,000 years, EBSPAC will calculate source term for periods extending to 10,000 years. In concurrence with the National Academy of Sciences (NAS) recommendation, the subsystem performance requirement periods of 300 to 1,000 years will not be considered.

2.3 COMPUTATIONAL APPROACH

EBSPAC will closely follow the existing approaches in SOTEC and SCCEX in order to minimize code development time. Specific attempts will be made to address numerical oscillations by incorporating methods to minimize them. The use of faster solvers for solving a system of ordinary differential equations will be investigated and will be compared against the fourth order Runge-Kutta method currently used.

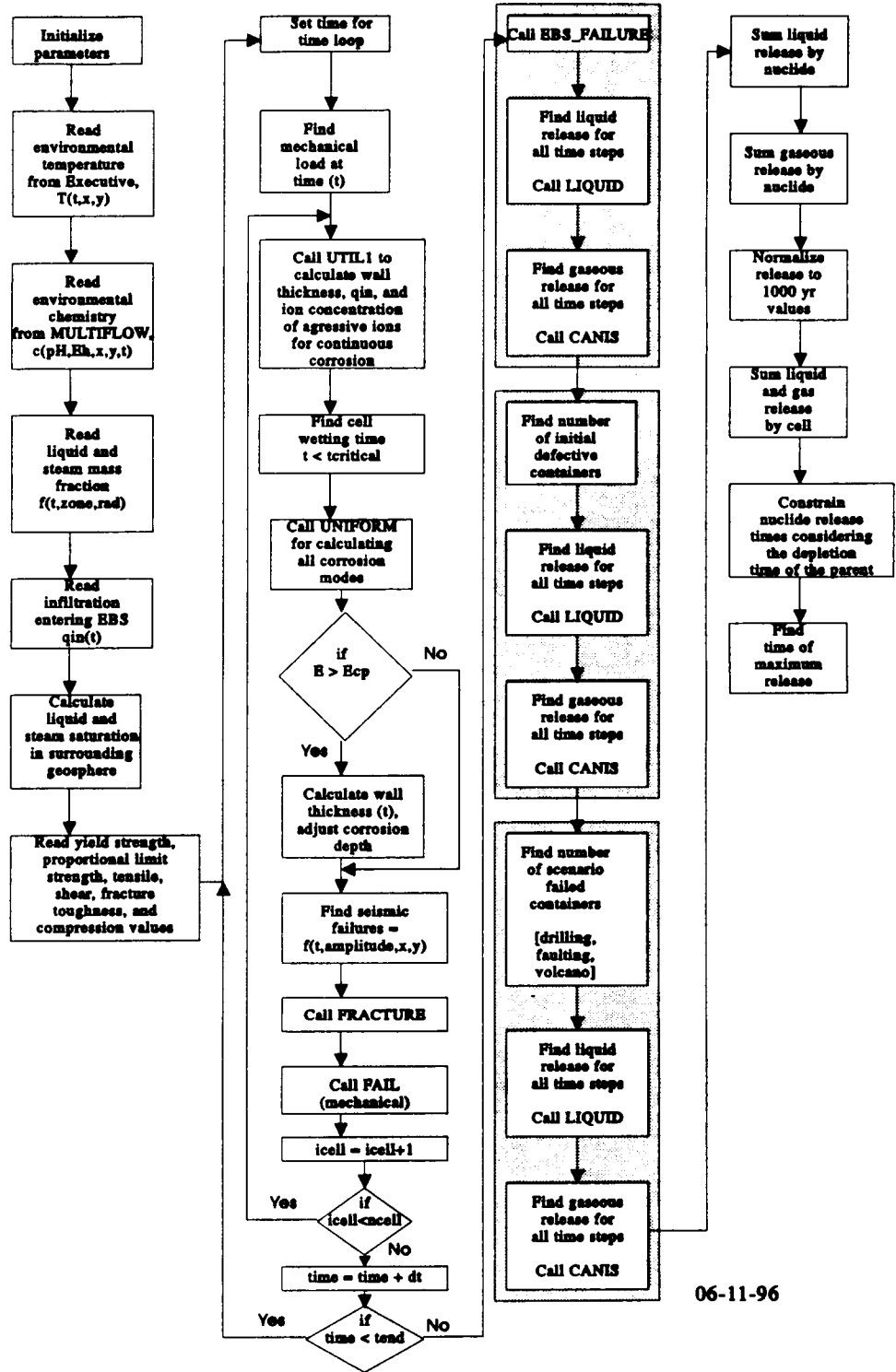
2.4 DATA FLOW AND USER INTERFACE

An initial version of the comprehensive computation flow chart has been developed for EBSPAC and is presented in Figure 4. The flow chart presents three segments of the EBSPAC code. The first column shows mainly the input data that will be obtained from the output of other codes such as MULTIFLO and ABAQUS. The second column presents the portion of the code that will calculate corrosion and mechanical failures. The third column represents release calculations for three release scenarios: (i) releases from initially defective canisters, (ii) releases subsequent to corrosion and mechanical failure, and (iii) releases from containers failed due to drilling, faulting, and volcanic activities. The fourth column represents the modules that will be used for record keeping (e.g., cumulative gas release, cumulative liquid release, inventory, etc.) and preparing data for output that can be used in TPA code.

The code is run on a UNIX system by entering the command:

ebspac [-i] < input < database . . . > output

where the square brackets designate that *-i* need not be entered in the command line, *input* represents the input file, *output* the output file, and . . . stands for possible additional options to be specified at a later date.



06-11-96

Figure 4. Proposed Engineered Barrier System Performance Assessment Code flow diagram

2.4.1 I/O

The input and output for EBSPAC will be presented in free format. Emphasis will be given to make the input file more readable and compatible to Monte Carlo sampling options to be invoked by the TPA executive code.

At the completion of program execution, the EBSPAC code will provide the following:

- Waste package failure time
- List of daughter products created before the waste package fails
- Mass of parent nuclide and mass of daughter products
- Radionuclide release rates as a function of time
- Cumulative release at the end of program execution

A partial list of input data includes:

General parameters

- Length of simulation time
- Length of waste package
- Waste package radius
- Packing radius
- External fixed boundary radius for the geosphere
- Container thickness (two types)
- Metric ton of waste per container
- Age of fuel
- Initial temperature
- Vertical extent of repository
- Spent-fuel alteration/dissolution rate
- Decay constants for radionuclides
- Initial time steps in years
- Minimum time step in years
- Ratio between evaporation and rate of water encroachment

Evaporation parameters

- Vapor diffusion coefficient
- Reference temperature for diffusion coefficient
- Geometric factor for rock
- Geometric factor for packing
- Porosity of packing for vapor diffusion
- Porosity of rock for vapor diffusion
- Water film thickness
- Dripping rate
- Concentration of NaCl in dripping water
- Initial salt around the container
- Initial scale on the container

Revised 6/14/96

- Funnel area for capture of darcy flow onto container
- Edge width of the circular umbrella
- Temperature at the waste package where the edge of umbrella hits
- Atmospheric pressure

Corrosion parameters

- Transfer coefficient (alpha) for oxygen reduction
- Transfer coefficient (beta) for water reduction
- Rate constant for oxygen reduction
- Rate constant for water reduction
- Activation energy for oxygen reduction
- Activation energy for water reduction
- Current density for air/steam environment
- Active current density
- Passive current density
- Nitrate/chloride ratio
- Reference pH
- Tortuosity in scale
- Porosity in scale
- Critical potentials for localized corrosion and stress corrosion cracking (E_{rp} , E_p)

Mechanical properties

- Young's modulus
- Poisson's ratio
- Yield strength
- Safety factor for uncertainties in mechanical properties
- Fracture toughness (K_{Ic} , J_{Ic})

This partial input list is by no means exhaustive. New parameters will be added and some of these parameters presented will be deleted during the course of EBSPAC code development.

2.5 HARDWARE AND SOFTWARE REQUIREMENTS

EBSPAC code will be a mixture of ANSI FORTRAN 77 and Fortran 90 routines with the main program being written with Fortran 90 language. Fortran 90 provides the option for compiling routines written in FORTRAN 77 language. Fortran 90 will provide significant improvement in coding (i.e., will use fewer code lines) and simplify input by making it more readable.

The FORTRAN 77 portion of the code will use *do-endo* and *include* statements as well as system-dependent calls for timing routines. The code will run on a SUN work station with the Fortran 90 compiler.

The use of Fortran 90 is justifiable for large codes because these codes, which will be audited for accuracy or will run on multiple platforms, must be developed in a controlled and standardized manner. Unlike FORTRAN 77, Fortran 90 lends itself to a fully structured approach.

Two of the undesirable features from the EBSPAC code development point of view are the structure and content of cryptic input files and the non-standard INCLUDE statement of Fortran 77. The cryptic input in FORTRAN 77 is reduced by writing a library of routines which permit free-format input using a system of keywords and free-format numerical data. Free-format means that a specific column location for data is not required. Also, both integer and floating point values could be supplied with or without decimal points or exponents. A large percentage of the previous source term code (i.e., SOTEC) was dedicated to the implementation of this free-format mechanism. The functions available in Fortran 90 eliminate the need for both of these procedures. The use of the NAMELIST statement for input provides a standardized free-format input mechanism. The MODULE concept in Fortran 90 virtually eliminates the need for INCLUDE statements altogether.

Fortran 90 will reduce the cost associated with the implementation of these functions which are necessary for the controlled development of software.

2.6 GRAPHICS

No special graphics output devices are required at this time. Output will be in the form of ascii files written in a spreadsheet-like format that can be read in by the user-preferred graphics software.

2.7 PRE- AND POST-PROCESSORS

No pre- or post-processors will be designed at this stage. The pre- and post- processors will be made a part of TPA executive code.

3 REFERENCES

Ahn, T.M. 1994. Long-term C-14 source term for a high-level waste repository, *Waste Management* 14(5): 393-408.

Cragolino, G., N. Sridhar, J. Walton, R. Janetzke, T. Torng, J. Wu, and P. Nair. 1994. "Substantially Complete Containment"—Example Analysis of a Reference Container. CNWRA 94-003. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

Lichtner, P.C. 1994. *Multiphase Reactive Transport Theory*. CNWRA 94-018. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

Sagar, B., R.B. Codell, J. Walton, and R.W. Janetzke. 1992. *SOTEC: A Source Term Code for High-Level Nuclear Waste Geologic Repositories—User's Manual: Version 1.0*. CNWRA 92-009. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

Sridhar, N., J.C. Walton, G.A. Cragolino, and P.K. Nair. 1993. *Engineered Barrier Performance Assessment Codes (EBSPAC) Progress Report—October 1, 1992 through September 25, 1993*. CNWRA 93-021. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

181/156

Revised 6/14/96

U.S. Department of Energy. 1994. *Initial Summary Report for Repository/Waste Package Advanced Conceptual Design. Revision 00*. CRWMS M&O Document No. B00000000-01717-5705-00015. Las Vegas, NV: TRW Environmental Safety Systems, Inc.

U.S. Department of Energy. 1995. *Total System Performance Assessment—1995: An Evaluation of the Potential Yucca Mountain Repository. Revision 01*. CRWMS M&O Document No. B00000000-01717-2200-00136. Las Vegas, NV: TRW Environmental Safety Systems, Inc.

Walton, J.C., and P.C. Lichtner. 1995. *Quasi-Steady State Model for Coupled Liquid, Vapor, and Heat Transport: Application to the Proposed Yucca Mountain High-Level Waste Repository*. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

183/195

**SOFTWARE REQUIREMENTS DESCRIPTION
FOR THE ENGINEERED BARRIER SYSTEM
PERFORMANCE ASSESSMENT CODE (EBSPAC)**

by
**Sitakanta Mohanty
Peter C. Lichtner
Ronald W. Janetzke
Gustavo A. Cragolino**

**CNWRA
San Antonio, Texas**

Version 1.0

Approved by:



**Narasi Sridhar, Manager EBS Element
CNWRA**

1 INTRODUCTION

This Software Requirements Description (SRD) document is the first stage in the development of the Engineered Barrier System Performance Assessment Code (EBSPAC), a numerical model for the evaluation of container life and source term. EBSPAC will be used as the source term module in the NRC Total System Performance Assessment (TPA) code.

2 SOFTWARE REQUIREMENTS DESCRIPTION (SRD): EBSPAC

This SRD briefly outlines the software function, technical bases, computational approach, program flow and user interface, hardware and software requirements, graphics, and pre- and post-processors that are relevant to the development and use of the code EBSPAC.

Because EBSPAC will be used as a module in the TPA code that will execute hundreds of Monte Carlo runs, the EBSPAC code must be based on simplified models that will allow fast computation. As a rule of thumb, the design will be such that the average execution time for a single EBSPAC run will be in the neighborhood of 22 seconds on CRAY-YMP with eight processors. With an execution time scale factor of approximately 10, this 22 cray-seconds will be equivalent to 220 seconds on a UNIX IPX machine.

2.1 SOFTWARE FUNCTION

The code EBSPAC will be used to provide the source term to the TPA code. The code will provide waste package failure time and the release rate of radionuclides from the engineered barrier system. The code may eventually be used to evaluate the issue of criticality in conjunction with the DOE reference waste package design proposed in the TSPA-95 report (U.S. DOE, 1994, 1995). However, criticality calculations are not considered in the first version of EBSPAC. Additionally, the multipurpose canister (MPC) performance is ignored in this version in correspondence with TSPA-95 calculations.

The first iteration of EBSPAC will be based on the SCCEX code modified to be applicable to the reference design.

2.2 TECHNICAL BASIS: PHYSICAL AND MATHEMATICAL MODELS

The code EBSPAC will consist of a number of modules based on a simplified physical description of the underlying processes which govern the failure of the waste package and ultimate release of radionuclides. Figures 1 and 2 show a simplified conceptualization around which EBSPAC code will be built. This includes the horizontal emplacement of the waste package instead of the vertical emplacement implemented in SCCEX and SOTEC (Cragolino et al., 1994; Sagar et al., 1992) codes. The main modules to be included in the EBSPAC code are briefly described below. The flow diagram shown in Figure 3 shows the stages at which these major modules are called.

2.2.1 Thermal Module

The thermal module will provide the temperature distribution as a function of time and position within the EBS as well as the surrounding geosphere. The thermal model will initially be based on conduction as the only heat-transfer mechanism. This module will reflect the geometry adopted in the reference design. Temperature distribution data will be stored in a file at various times and therefore this module will not be called from within EBSPAC. External temperature calculation is justifiable because conduction is considered to be the dominant heat transfer mechanism and also the amount of heat to be generated by the source in the future is known deterministically. Other heat-transfer modes (convection and radiation) may be considered in future versions of EBSPAC. For example, in the absence of backfill, radiation may become an important heat transfer mechanism.

2.2.2 Environmental Module

The environmental module processes input from external data sources. These data are to include ground water composition, such as concentrations of species Na^+ , Cl^- , HCO_3^- , and O_2 and pH among others, provided as functions of time. In addition, relative humidity and liquid saturation will be provided as functions of time. MULTIFLO will be used for generating fluid composition and species concentration data. The environmental conditions are expected to be affected significantly only by a few scenarios, including different mechanism or process of water movement (e.g. fracture flow). Calculating the environmental conditions outside the EBSPAC leaves the user with the option for using MULTIFLO only selectively.

Because the onset of aqueous corrosion is dictated by the waste package surface conditions, especially the presence of a water film, the time at which the surface will be covered by a water film (i.e., the surface wetting time and area wetted) will be determined by using either the vapor-pressure gradient model (Walton et al., 1995) or the critical relative humidity and the temperature approach adopted by the DOE.

The environmental module will provide the amount of water reaching the waste package surface which will be used in determining the wetted area of this surface.

2.2.3 Corrosion Module

The corrosion module will be changed to address the DOE reference design. The basic equations (Sridhar et al., 1993; Cragolino et al., 1994) will follow the SCCEX code with appropriate modifications to represent the DOE reference waste package materials and design. The SCCEX corrosion model includes

- general corrosion
- localized corrosion (pitting and crevice corrosion)
- stress corrosion cracking
- galvanic corrosion

The corrosion process at any given time is dictated by the corrosion potential and the corresponding critical potential associated with the process. Therefore, solution chemistry and relative humidity data read in by the environmental module, will also be used in this module. Galvanic

corrosion becomes important only after the outer overpack is penetrated. Hence, the corrosion module will consist of two stages—that for the outer overpack and that for the inner overpack.

2.2.4 Mechanical Failure Module

The applicability of the mechanical failure models used in SCCEX, namely, yielding, buckling, and fracture models, will be evaluated from the point of view of horizontal emplacement. For the new design, fracture failure as a result of the thermal embrittlement of the steel overpack is an important failure mode to be considered. The mechanical failure model should take into account material instability due to long-term thermal exposure. A simplified calculation of waste package temperature (thermal model) and changes in mechanical properties of overpack material will be considered in EBSPAC. For the first version, priority will be given to the fracture failure.

2.2.5 Release Rate Module

Gaseous Release:

For the sake of simplicity, the gaseous release will be treated separately from the liquid release. The gaseous release will include three release modes:

- (1) Prompt release: This will account for the loosely-held C-14 and I-129 on the cladding and cladding/fuel gap and grain boundaries, that will be released immediately following the canister failure. (within the first 100 years after the failure).
- (2) Oxidation of cladding: The zirconium cladding, which is a strong oxygen getter, will make this oxygen available to carbon, thus forming $^{14}\text{CO}_2$ upon oxidation. Other buried irradiated structural metal will also undergo the same process.
- (3) Oxidation of UO_2 : This will account for the largest amount of the inventory of the C-14 which is contained in the form of solid solutions, elemental carbon, carbides, and oxycarbides. Dry oxidation of UO_2 will also change its crystal structure, resulting in volume expansion and fracture.

These three processes have already been implemented in SOTEC (Sagar et al., 1992). These models will be implemented in EBSPAC considering more recent discussions (Ahn, 1994) where appropriate.

Liquid Release:

The liquid release will account for the radionuclides released from the fuel, production from radioactive decay of another radionuclide, losses from radioactive decay, advective flow, and diffusion. The liquid release of radionuclide from the fuel will include only congruent release option with the assumption that the radionuclides are contained in the matrix of the UO_2 . The advective mass transfer will be a simple mass balance (i.e., amount out = concentration x water flow rate out of the container). The diffusive mass transfer will be assumed to be taking place from a spherical source as implemented in SOTEC. The diffusion model will consider radioactive decay. Radioactive decay chains are not considered, however.

During a later phase of ESBSPAC code development activities, an attempt will be made to include the effects of uranyl silicates, potential secondary alteration products of the spent fuel in the expected high-silica concentration groundwater likely to encounter the waste package. Thermodynamic data is only poorly known at this time for these minerals. However, because of their large molar volumes, resulting in large positive volume changes in replacement reactions of spent fuel, formation of these alteration products could play an important role in dissolution of spent fuel.

A simplified model for colloids transport may be considered at a later phase of code development.

2.2.6 Inventory Model

The inventory in the failed waste package will be monitored by performing material balance calculation at specified time steps. The mass balance calculation will include depletion due to decay, generation of daughter products, and mass depletion due to diffusive and advective releases.

Although SCCEX conducted performance calculations up to 1000 years, ESBSPAC will calculate source term for periods extending to 10,000 years. In concurrence with the National Academy of Sciences (NAS) recommendation, the subsystem performance requirement periods of 300 to 1,000 yr will not be considered.

2.3 COMPUTATIONAL APPROACH

ESBPAC will closely follow the existing approaches in SOTEC and SCCEX in order to minimize code development time. Specific attempts will be made to address numerical oscillations by incorporating methods to minimize them. The use of faster solvers for solving a system of ordinary differential equations will be investigated and will be compared against the fourth order Runge-Kutta method currently used.

2.4 DATA FLOW AND USER INTERFACE

A first-cut comprehensive computation flow chart has been developed for ESBSPAC and is presented in Figure 4. The flow chart presents three segments of the ESBSPAC code. The first column shows mainly the input data that will be obtained from the output of other codes such as MULTIFLO and temperature code such as ABAQUS. The second column presents the portion of the code that will calculate corrosion and mechanical failures. The third column represents release calculations for three release scenarios: i) releases from initially defective canisters, ii) releases subsequent to corrosion and mechanical failure, and iii) releases from containers failed due to drilling, faulting, and volcanic activities. The fourth column represents the modules that will be used for record keeping (e.g. cumulative gas release, cumulative liquid release, inventory, etc.) and preparing data for output that can be used in TPA code.

The code is run on a UNIX system by entering the command:

```
ebspac < input < database . . . > output
```

where the square brackets designate that *-i* need not be entered in the command line, input represents the input file, output the output file, and stands for possible additional options to be specified at a later date.

2.4.1 I/O

The input and output for EBSPAC will be presented in free format. Emphasis will be given to make the input file more readable and compatible to Monte Carlo sampling options to be invoked by the Total Performance Assessment (TPA) executive code.

At the completion of program execution, the EBSPAC code will provide the following:

- Waste package failure time
- List of daughter products created before the waste package fails
- Mass of parent nuclide and mass of daughter products
- Radionuclide release rates as a function of time
- Cumulative release at the end of program execution

A partial list of input data includes:

General parameters

- Length of simulation time
- Number of cells
- Length of waste package
- Waste package radius
- Packing radius
- External fixed boundary radius for the geosphere
- Container thickness (two types)
- Initial thermal power density
- Metric ton of waste per container
- Age of fuel
- Initial temperature
- Vertical extent of repository
- Spent-fuel alteration rate
- Decay constants for radionuclides
- Initial time steps in years
- Minimum time step in years
- Ratio between evaporation and rate of water encroachment

Evaporation parameters

- Vapor diffusion coefficient
- Reference temperature for diffusion coefficient
- Geometric factor for rock
- Geometric factor for packing
- Porosity of packing for vapor diffusion

- Porosity of rock for vapor diffusion
- Water film thickness
- Dripping rate
- Effective concentration of NaCl in dripping water
- Effective concentration of scale forming minerals in dripping water
- Scale factor for vapor pressure lowering relative to NaCl solution
- Initial salt around the container
- Initial scale on the container
- Funnel area for capture of darcy flow onto container
- Edge width of the circular umbrella
- Temperature at the waste package where the edge of umbrella hits
- Atmospheric pressure

Corrosion parameters

- Transfer coefficient (alpha) for oxygen reduction
- Transfer coefficient (beta) for water reduction
- Rate constant for oxygen reduction
- Rate constant for water reduction
- Activation energy for oxygen reduction
- Activation energy for water reduction
- Current density for air/steam environment
- Active current density
- Passive current density
- Nitrate/chloride ratio
- Reference pH
- Tortuosity in scale
- Porosity in scale
- Critical potentials for localized corrosion and stress corrosion cracking (E_{rp} , E_p)

Mechanical properties

- Young's modulus
- Poisson's ratio
- Yield strength
- Safety factor for uncertainties in mechanical properties
- Fracture toughness (K_{Ic} , J_{Ic})

This partial input list is by no means exhaustive. New parameters will be added and some of these parameters presented will be deleted during the course of EBSPAC code development.

2.5 **HARDWARE AND SOFTWARE REQUIREMENTS**

EBSPAC code will be a mixture of ANSI FORTRAN 77 and Fortran 90 routines with the main program being written with Fortran 90 language. Fortran 90 provides the option for compiling routines written in FORTRAN 77 language. Fortran 90 will provide significant improvement in coding (i.e., will use fewer code lines) and simplify input by making it more readable.

The FORTRAN 77 portion of the code will use *do-ends* and *include* statements as well as system-dependent calls for timing routines. The code will run on a SUN work station with the Fortran 90 compiler.

The use of Fortran 90 is justifiable for large codes because these codes which will be audited for accuracy, or codes which will run on multiple platforms must be developed in a controlled and standardized manner is essential for large codes, Unlike FORTRAN 77, Fortran 90 lends itself to a fully structured approach.

Two of the undesirable features from the EBSPAC code development point of view are the structure and content of cryptic input files and the non-standard INCLUDE statement of Fortran 77. The cryptic input in FORTRAN 77 is reduced by writing a library of routines which permit free-format input using a system of keywords and free-format numerical data. Free-format means that a specific column location for data is not required. Also, both integer and floating point values could be supplied with or without decimal points or exponents. A large percentage of the previous source term code (i.e., SOTEC) was dedicated to the implementation of this free-format mechanism. The functions available in Fortran 90 eliminate the need for both of these procedures. The use of the NAMELIST statement for input provides a standardized free-format input mechanism. The MODULE concept in Fortran 90 which virtually eliminates the need for INCLUDE statements altogether.

Fortran 90 will reduce the cost associated with the implementation of these functions which are necessary for the controlled development of software.

2.6 GRAPHICS

No special graphics output devices are required at this time. Output will be in the form of ascii files written in a spreadsheet-like format that can be read in by the user-preferred graphics software.

2.7 PRE- AND POST-PROCESSORS

No pre- or post-processors will be designed at this stage. The pre- and post- processors will be made a part of TPA executive code.

3 REFERENCES

- Ahn, T.M. 1994. Long-Term C-14 Source Term for a High-Level Waste Repository, *Waste Management*. 14(5): 393-408.
- Cragolino, G., N. Sridhar, J. Walton, R. Janetzke, T. Trong, J. Wu, and P. Nair. 1994. *Substantially Complete Containment—Example Analysis of a Reference Container*. CNWRA 94-003. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.
- Lichtner, P.C. 1994. *Multiphase Reactive Transport Theory*. CNWRA 94-018. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

Sagar, B., R.B. Codell, J. Walton, R.W. Janetzke. 1992. SOTEC: A Source Term Code for High-Level Nuclear Waste Geologic Repositories User's Manual: Version 1.0. CNWRA 92-009. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

Sridhar, N., J.C. Walton, G.A. Cragnolino, P.K. Nair. 1993. Engineered Barrier Performance Assessment Codes (EBSPAC) Progress Report— October 1, 1992 through September 25, 1993. CNWRA 93-021. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

U.S Department of Energy. 1994. Initial Summary Report for Repository/Waste Package Advanced Conceptual Design. Revision 00. CRWMS M&O Document No. B00000000-01717-5705-00015. Las Vegas, NV: TRW Environmental Safety Systems, Inc.

U.S. Department of Energy. 1995. Total System Performance Assessment—1995: An Evaluation of the Potential Yucca Mountain Repository. Revision 01. CRWMS M&O Document No. B00000000-01717-2200-00136. Las Vegas, NV: TRW Environmental Safety Systems, Inc.

Walton, J.C., and Lichtner, P.C. 1995. Quasi-Steady State Model for Coupled Liquid, Vapor, and Heat Transport: Application to the Proposed Yucca Mountain High-Level Waste Repository. San Antonio, TX: Center for Nuclear Waste Regulatory Analyses.

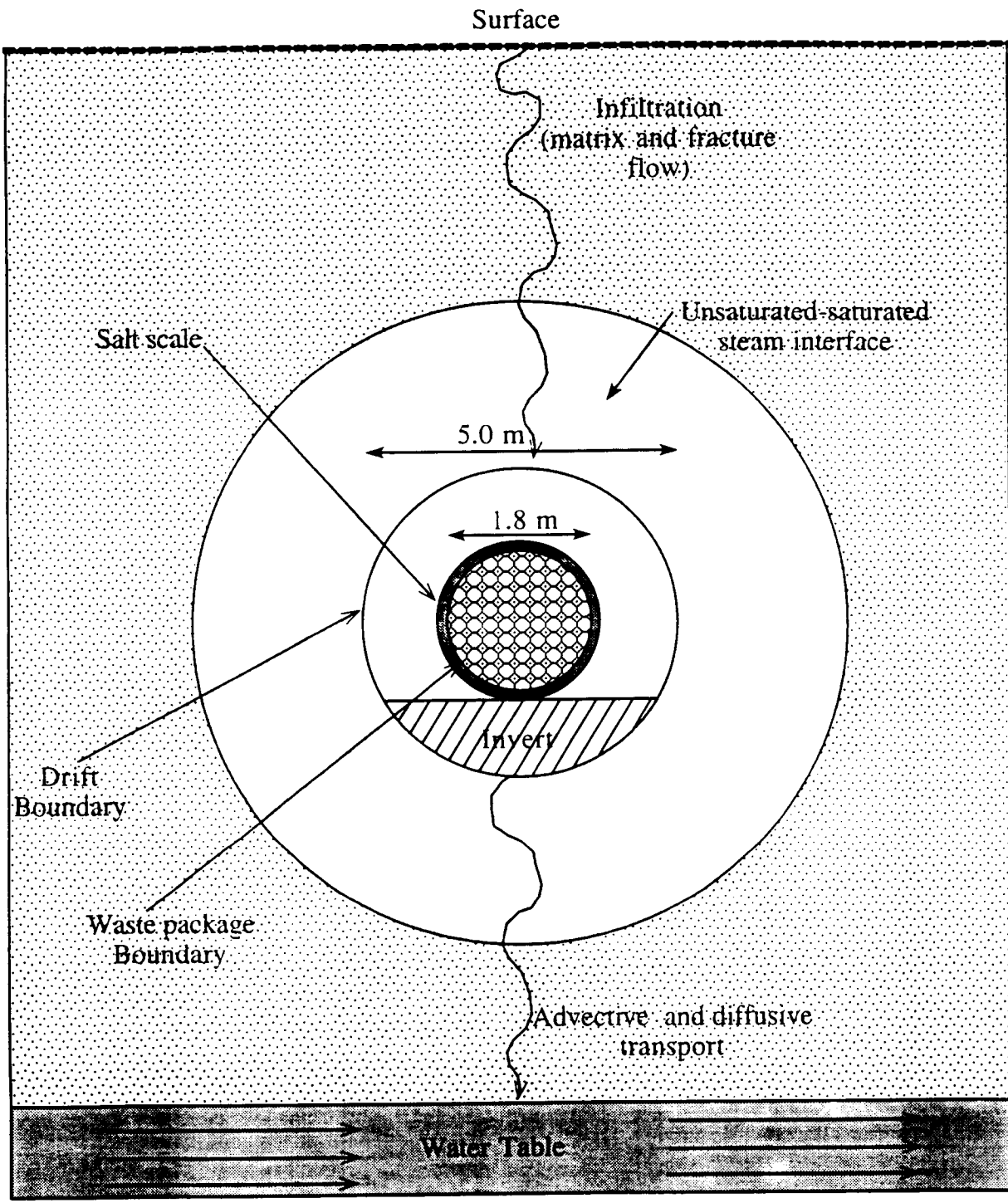


Figure 1. Conceptualization of the radionuclide release from waste packages emplaced in a horizontal drift

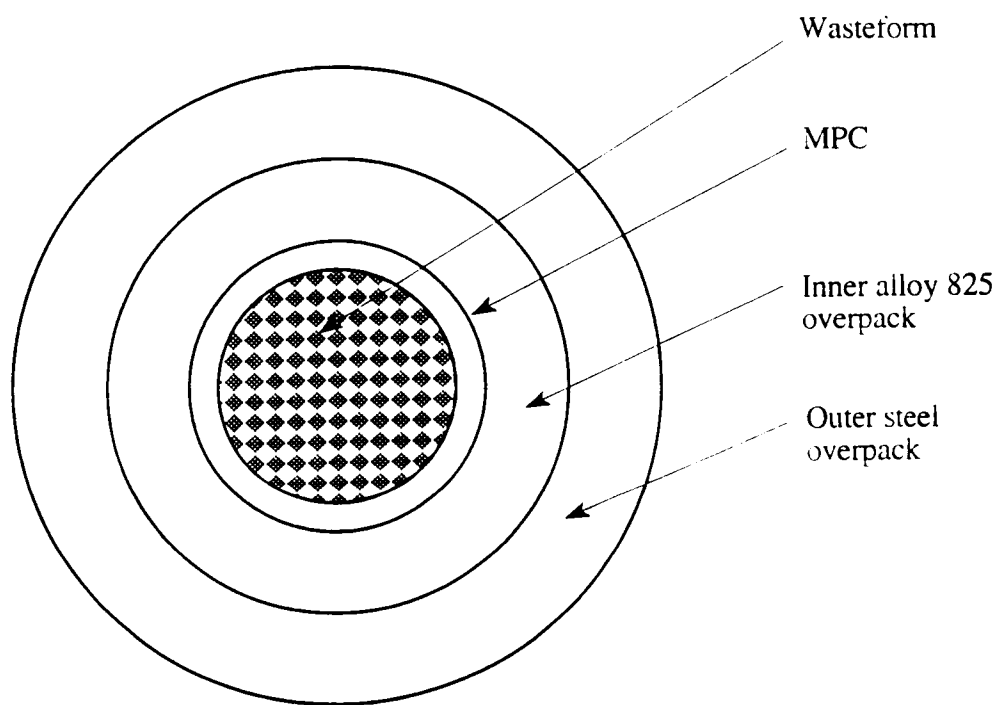


Figure 2. Schematic showing the magnified view of the cross-section of a waste package

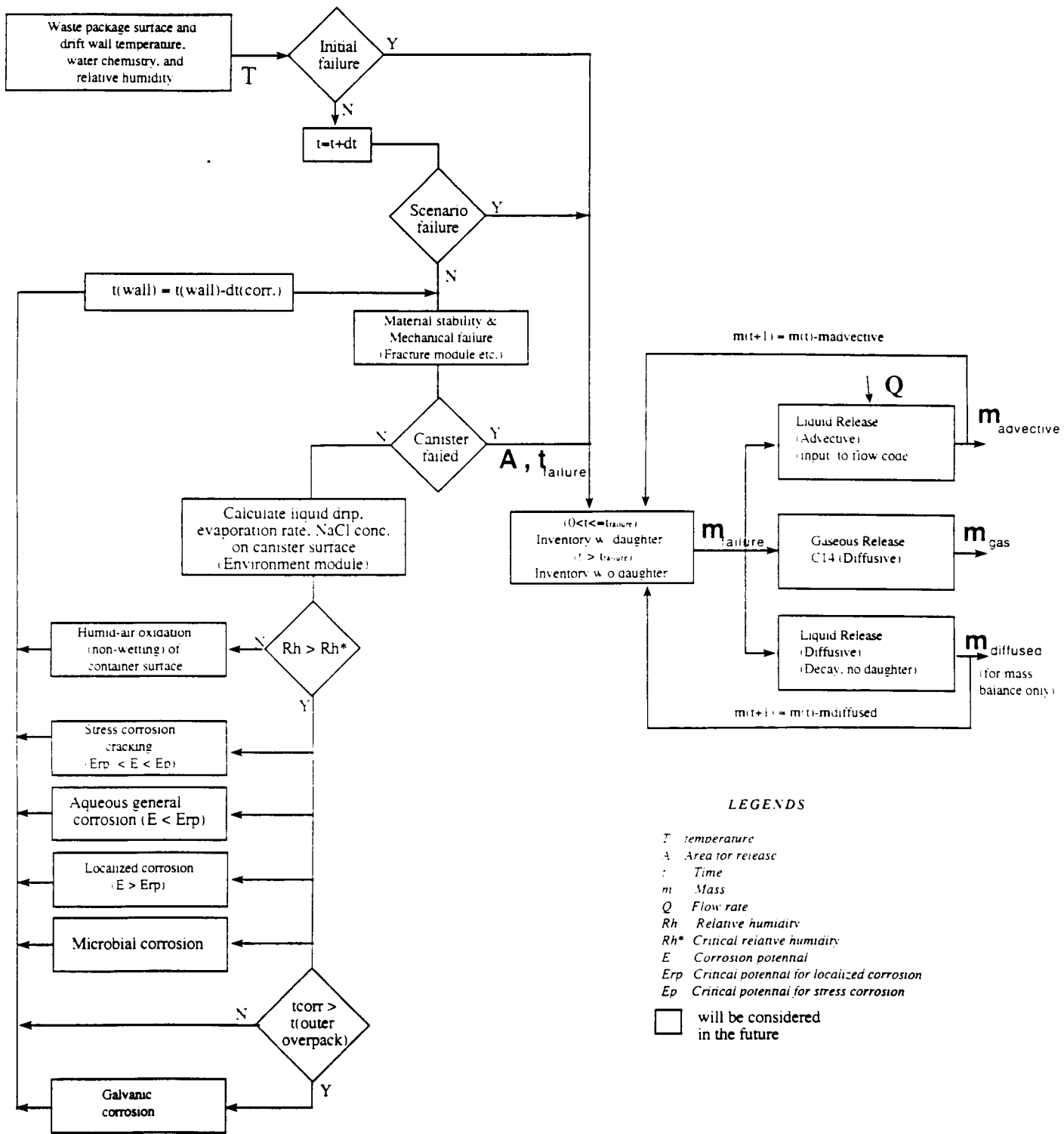
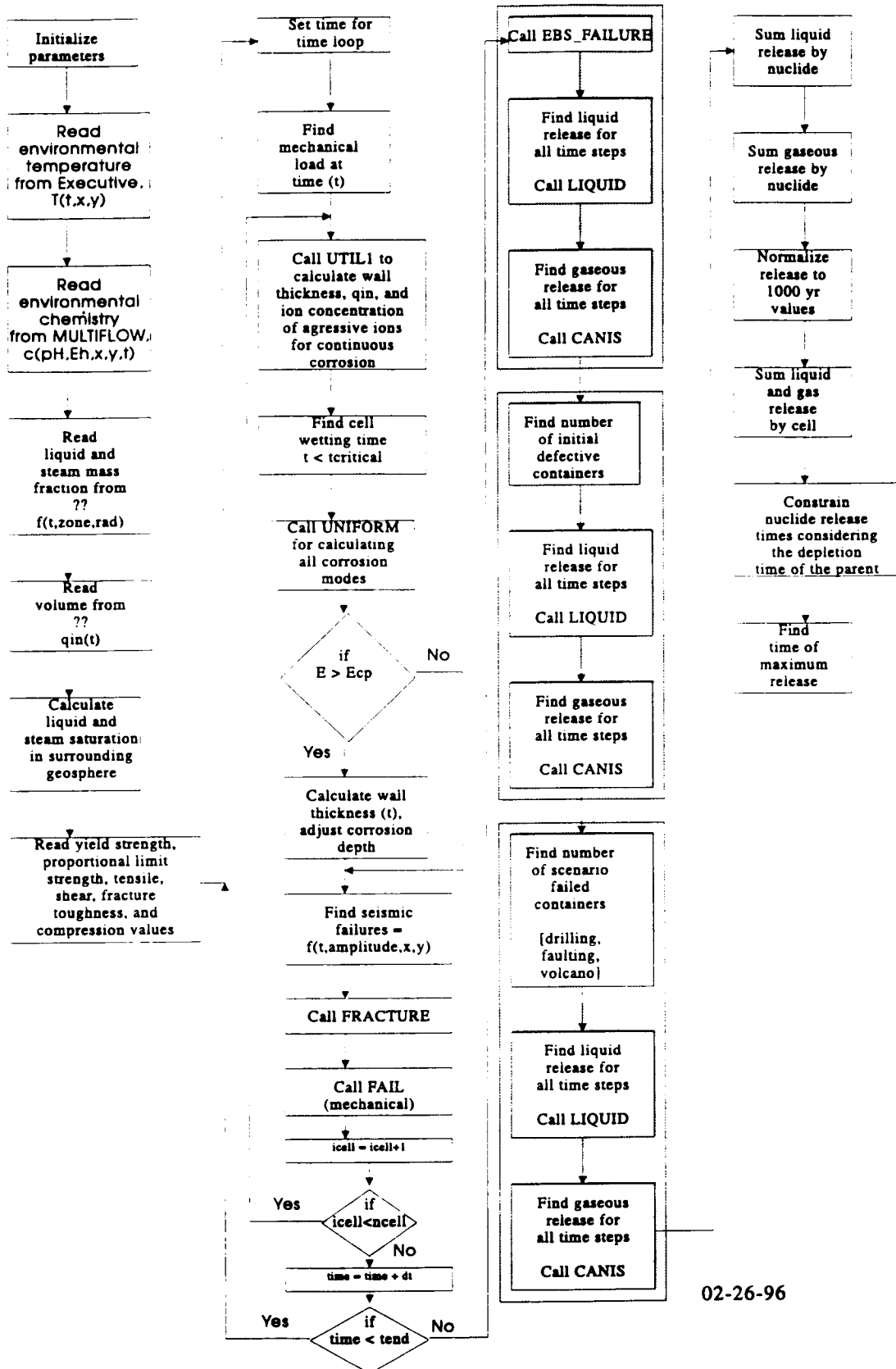


Figure 3. Flow chart showing the major components of the source term



02-26-96

Figure 4. Proposed engineered barrier system performance assessment code flow diagram

15 | 196

Software
Analysis Tools (SPAS)

pcl4

JOB 165

fail.f

For: root
Date: Wed Jan 8 15:30:58 CST 1997
Submit queue: Ethernet
Submitted: 116:47:36
Started: 116:47:36



QMS 1725 Print System

QMS 1725 (1st floor)

16/195

**==EBSPAC_FAIL.spg processed by SPAG 4.00 at 14:01 on 7 Jan 1997

program ebspac_fail
implicit none

C*** Start of declarations inserted by SPAG

real*8 aa, age, betahy1, betahy2, betaox1, betaox2, cfactor,
& clconc, clconca, clcrit1, clcrit2, constant1, cpass,
& ctemp, cthick1, cthick2, curact, d, dtini, dtmax
real*8 dtmin, e, ecor, ecrit, eps, errabs, errrel, filmthk,
& garb1, gbthick, ghy1, ghy2, gox1, gox2, grainr, humd,
& humdc, odev, pnt, pntd
real*8 pntf, rate, refph, rkhy1, rkhy2, rkox1, rkox2, rkqc,
& rthick, sfactor, sollay, spor, taus, tavg, tcan, temp2,
& temp3, tend, tfail, thicktot
real*8 timintv, timr, tiny, tstart, tstop, twet, twetf, tyears,
& wpdia, wplen, x, xcouple, xread, y, z
integer i, icorr, ifail, iflag, iflag1, iflag3, iflagcl, ilayer,
& nbad, nhist, nhista, nintv, nok, nseries, nset, nv

C*** End of declarations inserted by SPAG

c-----
c Code: EBSPAC (Engineering Barrier System Performance Assessment Code)

c Part: ebspac_fail.f. This is one of the two parts of EBSPAC

c Version: 1.0

c Date: January 6, 1997

c Purpose: The EBSPAC computer program has been developed jointly by the
c Center for Nuclear Waste Regulatory Analyses (CNWRA) and the
c U. S. Nuclear Regulatory Commission (NRC) for use in the
c review of DOE's licence application for the HLW Geologic
c Repository by the Office of NMSS, DWM.

c ebspac_fail.f part of EBSPAC calculates waste package failure
c time & ebspac_release.f part of EBSPAC calculates the rate of
c release of nuclides from the repository in the near field.

c The repository is assumed to be made up of cells and EBSPAC
c provides results for one cell. Input to EBSPAC is read from
c files which may either be created manually by the user or
c generated by use of codes other than EBSPAC.

c Developers: ebspac_fail.f was developed by Sitakanta Mohanty
c at the CNWRA with assistance from Tae Ahn (NRC, 301-415-5812),
c G. Cragnolino, P. Lichtner, & N. Sridhar

c User: Developed for use in NRC Iterative Performance Assessment
c III. This computer code is managed under CNWRA's CODE
c CONFIGURATION PROCEDURE. Any modifications to the source
c code must be reported to code custodian (see below).
c OTHER versions of this code will be released in the future.

c Disclaimer: This computer code has not been formally or informally
c verified, is known to have numerous bugs, and the model
c embodied in it are not validated.

c Contact: Sitakanta Mohanty (210)522-5185
c Center for Nuclear Waste Regulatory Analyses
c Southwest Research Institute, San Antonio, Tx 78250
c-----

```

parameter (nv=1, nintv=20000)
external rate, bsstep, rkqc
character*80 title1
character*7 , mode
character*24 fdate

dimension odev(nv)

common /bulk / age, tend, x, y, z, d, filmthk, e, cthick1,
&          cthick2, sfactor
common /history/ nhist
common /thist / timr(nintv), tavg(nintv), tcan(nintv),
&          humd(nintv)
common /new1 / clconca, clcrit1, clcrit2, clconc(nintv),
&          cfactor, iflagcl
common /corrpar1/ ecor, ecrit, cpass, sollay, curact, aa(2, 3),
&          xread, refph, betaox1, betahy1, betaox2,
&          betahy2, rkox1, rkhy1, rkox2, rkhy2, gox1,
&          ghy1, gox2, ghy2, spor, taus, xcouple
c common /corrpar2/ xipto, slpto, xirpo, slrpo, xipti, slpti,
c &          xirpi, slrpi
common /solve / dtini, dtmax, errrel, errabs
common /temp1 / iflag, nset, timintv, humdc, ctemp, nhista
common /dryoxdc/ grainr, gbthick, constant1, nseries
common /wpgmt/ wplen, wpdia
common /wetting/ twet
common /flags / icorr
crrrr common /cripotout/ xipto,pttemo,slpto,slpttemo,
crrrr &          xirpo,rptemo,slrpo,slrptemo
crrrr common /cripotin/ xipti,pttemi,slpti,slpttemi,
crrrr &          xirpi,rptemi,slrpi,slrptemi

open (2, file='tefkki.inp', status='old')
c open(2,file='../83mtu.doe',status='old')
open (3, file='temphumd.dat', status='unknown')
c open(9,file='fail1.inp',status='old')
open (9, file='example_fail.inp', status='old')
open (8, file='corrode.out', status='unknown')
open (19, file='transition.out', status='unknown')
open (22, file='echo_fail.dat', status='unknown')

-----
c
c nomenclature:
c
c age:      age of the waste [yr] (not used)
c tend:    end time for simulation
c filmthk: thickness of water film [m] on the wp
c cthick1: wp outer overpack initial thickness [m]
c cthick2: wp inner overpack initial thickness [m]
c sfactor: safety factor for uncertainties in mechanical properties
c betaox:  beta kinetics parameter for oxygen reduction
c betahy:  beta kinetics parameter for water reduction
c rkox:    rate constant for oxygen reduction [coulomb m/mole/yr]
c rkhy:    rate constant for water reduction in [coulomb/m^2/yr]
c gox:     activation energy for oxygen rate constant [J/mole]
c ghy:     activation energy for water reduction [j/mole]
c aa(1,3): cpass = aa(1,1)+aa(1,2)*t+aa(1,3)*t**2
c aa(2,3): cpass = aa(2,1)+aa(2,2)*t+aa(2,3)*t**2
c ecrit:   critical potential for localized corrosion [volts SHE]
c taus:    tortuosity/geometry correction for liquid diffusion

```

18 / 195

```

c      of oxygen through boiler scale
c      refph: reference ph
c      spor: porosity of scale deposited on the wp [=1.0 for liq. film]
c      dtini: initial time step in years
c      dtmax: maximum time step in years
c      errrel: relative error
c      errabs: absolute error
c      timr(i): representative time [yr]
c      tavg(i): average repository temperature [Celsius]
c      tcan(i): waste package surface temperature [Celsius]
c      ecor: corrosion potential [volts SHE]
c      cpass: passive current density [Coulomb/m^2/yr]
c      sollay: thickness of scale layer [m]
c      curact: active current density [coulomb/m^2/yr]
c      grainr: grain radius [micrometer]
c      nseries: number of terms in infinite series for wp oxidation
c      gbthick: grain boundary thickness [micrometer]
c      constant1:a constant relating matrix and grain boundary
c              oxygen diffusion in metal
c      humdc: critical relative humidity
c      clconca:chloride ion concentration near the WP [mol/liter]

```

c -----begin reading input data-----

c top18 requirements:

```

      write (*, *)
& 'EBSPAC (Engineering Barrier System Performance Assessment Code)'
      write (*, *)
& 'This is the part of the code that computes WP Failure Time'
      write (*, *) 'Version= 1.0'
      write (*, *) fdate()

```

c prepare and read input data including temperature rel. hum. history

```

      call input
      nhista = int(tend/timintv)
      call tempstry(iflag, nset, timr, tcan, tavg, humd, tend, nhist,
&                nhista, timintv)

      open (21, file='chloride.dat', status='unknown')

```

c--> prepare chloride concentration data. Remove this feature when data is available from multiflo directly

```

      garb1 = 99999.
      title1 = 'This is replacement for peters data'

      write (21, *) title1
      write (21, *) nhist, garb1, nhist, garb1, nhist, garb1
      do 100 i = 1, nhist
        if ( tcan(i).gt.ctemp ) then
          write (21, 9001) timr(i), clconca, timr(i), clconca, timr(i)
&                , clconca
        else
          write (21, 9001) timr(i), clconca/cfactor, timr(i),
&                clconca/cfactor, timr(i), clconca/cfactor
        end if

```

100 continue

19/195

close (21)

call chloride(iflag, nset, timr, clconc, nhist)

c write to the data file to be read by the subsequent program(s)

write (*, '(a58)') 'CALCULATION OF WASTE PACKAGE FAILURE TIME'
write (*, '(80("="))')
write (3, *) tend
write (3, *) nhist
write (*, '(a50,f12.0)') 'end of simulation time [yr]:', tend
write (*, '(a50,i12)')

& 'No. of rows of data to pass to release.f:', nhist

write (8, '(80("-"))')
write (8, 9004)
write (*, '(80("-"))')
write (*, 9004)
write (8, 9005)
write (8, '(80("="))')
write (*, 9005)
write (*, '(80("="))')

write (3, *) 'i,timr,tcan,tavg,humd'

c write(*,*) 'i,timr,tcan,tavg,humd'

do 200 i = 1, nhist

write (3, '(i5,10f12.4)') i, timr(i), tcan(i), tavg(i),

& humd(i)

c write(*, '(i5,10f12.4)') i,timr(i),tcan(i),tavg(i),humd(i)

200 continue

csm initialize v

do 300 i = 1, nv
odev(i) = 0.0

300 continue

ilayer = 1

pnt = 0.

twetf = 99999.

do 400 i = 2, nhist

dtmin = 0.0

eps = errrel

tiny = errabs

crrrr time = timr(i)

if (i.gt.1) then

tstart = timr(i - 1)

tstop = timr(i)

else

tstart = 0.

tstop = errabs

end if

c calculate depth of penetration of corrosion pit.....

temp2 = tcan(i)

if (humd(i).lt.humdc) then

tyears = tstop - tstart

call dryoxdwp(i, tyears, temp2, grainr, nseries, gbthick,

& constant1, pntd)

```

    pnt = pnt + pntd
    twet = timr(i)
    iflag3 = 0
    odev(1) = pnt
csss    print*, 'timr, penetration by dry oxidation =', timr(i), pnt
else
    if ( iflag3.eq.0 ) then
        twetf = twet
        pntf = pnt
cdbg    print*, 'wetting time [yr] =', twetf
cdbg    print*, 'time, penetration by dry oxidation', tstart, pntf
        iflag3 = 1
    end if
    call odeint(odev, nv, tstart, tstop, eps, dtini, dtmin,
&             dtmax, nok, nbad, tiny, rate, rkqc, ilayer,
&             cthick1)
    pnt = odev(1)
end if

```

c test if fracture failure would occur.....

```

temp3 = tcan(i)

if ( pnt.le.cthick1 .and. icorr.eq.1 )
&    call mech(i, tstart, tstop, temp3, pnt, ifail)
thicktot = cthick1 + cthick2
rthick = thicktot - pnt

mode = 'dry oxd'
if ( tstop.gt.twetf ) then
    if ( ecor.gt.ecrit .and. iflagcl.eq.1 ) then
        mode = 'local '
    else
        mode = 'general'
    end if
end if

write (8, 9003) ilayer, tstop, tcan(i), ecrit, ecor, iflagcl,
&         rthick
write (*, 9003) ilayer, tstop, tcan(i), ecrit, ecor, iflagcl,
&         rthick, mode
if ( odev(1).gt.thicktot .or. ifail.eq.1 ) then
    iflag1 = 1
    go to 500
else
    iflag1 = 0
end if

400 continue

500 continue
write (*, '(80(=""))')
write (*, '(a42,f12.0)') 'wp wetting time [yr]:', twetf
if ( iflag1.eq.1 ) then
    tfail = tstart
    write (3, *) tfail
    write (3, *) wplen
    write (3, *) wpdia
    write (*, '(a42,f12.0)') 'wp failure time [yr]:', tfail
else

```



```

tfail = tend
write (3, *) tfail
write (3, *) wplen
write (3, *) wpdia
write (*, '(a42,f12.0,a)') , '$$$ no wp failure below [yr]:',
&   tfail, '$$$'
end if

write (*, '(a42,1pe15.3)') 'penetration by dry oxidation [m]:',
&   pntf
write (*, '(a42,6x,a)') 'echoed input data in:', 'echo_fail.dat'
write (*, '(a42,6x,a)') 'output data are in files:',
&   'temphumd.dat and corrode.out'
write (*, '(80(=""))')

close (2)
close (3)
close (9)
close (8)
close (19)
close (21)

stop

9001 format (6(1x,1P1e10.4))
cdbg   print*, '**before odeint**',(odev(in),in=1,nv),tstart,tstop
c      write(*,121) tstart,tstop,eps,dtini,dtmin,dtmax
9002 format ('tstart,tstop,eps,dtini,dtmin,dtmax', 20E13.3)

9003 format (i2, 2x, 2(f10.2,1x), 2(f10.4,1x), 4x, i1, 2x, 1pe15.7,
&          2x, a7)
9004 format (1x, 'ilayer', 2x, 'tstop', 6x, 'tcan', 6x, 'ecrit', 6x,
&          'ecorr', 3x, 'chloride', 3x, 'rthick',6x, 'mode')
9005 format (1x, ' ', 2x, '[yr]', 6x, ' [C]', 6x, '[Vshe]', 5x,
&          '[Vshe]', 5x, 'flag', 6x, ' [m] ')
end

**==TEMPHSTRY.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

```

subroutine tempistry(iflag, nset, timr, tcan, tavg, humd, tend,
&          nhist, nhist3, timintv)
implicit none
C*** Start of declarations inserted by SPAG
real*8 ambient, col, dcan, dt, eps1, humd, tavg, tcan, tend,
&          timintv, timr
integer i, iflag, ii, iset, nhist, nhist3, nintv, nset
C*** End of declarations inserted by SPAG
c-----
c      Purpose: read temperature, relative humidity from a table to be
c      provided by the TPA code. Also, DOE data can be read if
c      proper data file is provided
c-----
parameter (nintv=20000)
dimension timr(nintv), tcan(nintv), tavg(nintv), humd(nintv)
c      local arrays:
dimension col(4, 12)
c-----

```

Nomenclature:

```

c maxrow: maximum number of rows provided in the temperature data file
c timr:   time at which temperature and relative humidity
c         data are provided [y]
c tcan:   surface temperature of the wp outer overpack [c]
c tavg:   temperature of the drift wall surrounding the wp [c]
c humd:   relative humidity near the wp surface [dimensionless]
c nset:   corresponds only to iflag = 2; it is the data set
c         that corresponds to a particular set of
c         temperature, humidity data

```

```

c iflag  :1      use functional temp, humidity representation
c         of DOE data
c         :2      read from a file temp,humidity
c -----

```

```

c note: the reason we have this doe model is that the temperature is
c calculated to 100,000 years. But as drift surface temperature
c is not available, therefore, a constant temperature difference
c between wp surface and drift wall (dcan) is assumed

```

```

if ( iflag.eq.1 ) then
  ambient = 25.
  dt = timintv
  dcan = 30.
  timr(1) = 0.0
  tavg(1) = 185.96
  tcan(1) = tavg(1) + dcan
  humd(1) = 0.16535
  do 50 i = 2, nhist3
    timr(i) = i*(dt - 1)
    tavg(i) = 185.96 - 30.694*log10(timr(i))
    tcan(i) = tavg(i) + dcan
    humd(i) = 0.16535 + 0.19391*log10(timr(i))
    if ( tavg(i).lt.ambient ) tavg(i) = ambient
    if ( humd(i).gt.1.0 ) humd(i) = 1.0
50  continue
  nhist = nhist3
end if

```

```

c note: the original data file could contain such data that the
c temperature data at two consecutive time are the same
c (i.e., for the lack of adequate resolution). therefore,
c screening is done to ensure that temperatures at two consecutive
c times are different by at least eps1. Also, data recorded at
c time intervals smaller than mintime are discarded.

```

```

if ( iflag.eq.2 ) then
  read (2, *)
  read (2, *) ((col(ii,iset),ii=1,4), iset=1, nset)
  nhist3 = int(col(1,nset))
  write (1, *) , 'nhist3=', nhist3
  eps1 = 1.E-6
  read (2, *) ((col(ii,iset),ii=1,4), iset=1, nset)
  timr(1) = col(1, nset)
  tcan(1) = col(2, nset)
  tavg(1) = col(3, nset)
  humd(1) = col(4, nset)

```

23/195

```

nhist = 1
do 100 i = 2, nhist3
  read (2, *) ((col(ii,iset),ii=1,4), iset=1, nset)
  if ( abs(col(2,nset)-tcan(nhist)).gt.eps1 .and. col(1,nset)
&      - timr(nhist).gt.timintv ) then
    nhist = nhist + 1
    timr(nhist) = col(1, nset)
    tcan(nhist) = col(2, nset)
    tavg(nhist) = col(3, nset)
    humd(nhist) = col(4, nset)
  end if
  if ( i.eq.nhist3 .and. timr(nhist).lt.col(1,nset) ) then
    nhist = nhist + 1
    timr(nhist) = col(1, nset)
    tcan(nhist) = col(2, nset)
    tavg(nhist) = col(3, nset)
    humd(nhist) = col(4, nset)
  end if
100  continue
end if

return
end
**==CHLORIDE.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

```

subroutine chloride(iflag, nset, timr, clconc, nhist)
implicit none
C*** Start of declarations inserted by SPAG
  real*8 clconc, clconc1, col, time, timr, timr1, tmax, yprevious,
&      yyyy
  integer i, iflag, ii, iset, isetp, nhist, nhist4, nintv, nset,
&      nsetp
C*** End of declarations inserted by SPAG

```

c purpose: obtain data from the chloride data file. chloride
c data are presented for three thermal loadings only. chloride
c data are based on the assumption that there is no difference
c between backfill-no backfill, ventilation-no ventilation etc.
c-----

```

parameter (nintv=20000)
dimension timr(nintv), clconc(nintv)
dimension timr1(nintv), clconc1(nintv)
c local arrays:
dimension col(4, 12)

```

c Notes:
c time and temperature data in the data file coming from the
c near-field environment kti which contains chloride data do
c not have to match with the one coming from the tef kti.
c however, the max. simulation time (tmax) must match.

c If the chloride concentration history is not provided until
c tmax, then the last value provided in the chloride concentration
c file is repeated until tmax.

c Nomenclature:

24/195

```
c maxrow: maximum number of rows provided in the temperature data file
c timr:   time at which temperature and relative humidity
c         data are provided [y]
c nset:   corresponds only to iflag = 2; it is the data set
c         that corresponds to a particular set of chloride data
```

```
open (21, file='chloride.dat', status='old')

if ( iflag.eq.1 ) then
  read (21, *)
  read (21, *) nhist
  write (*, *) 'nhist=', nhist
  do 50 i = 1, nhist
    read (21, *) timr(i), clconc(i)
50  continue
  go to 300
end if

if ( nset.le.4 ) nsetp = 1
if ( nset.gt.4 .and. nset.le.8 ) nsetp = 2
if ( nset.gt.8 .and. nset.le.12 ) nsetp = 3

read (21, *)
read (21, *) ((col(ii,iset),ii=1,2), iset=1, nsetp)
nhist4 = int(col(1,nsetp))
do 100 i = 1, nhist4
  read (21, *) ((col(ii,isetp),ii=1,2), isetp=1, nsetp)
  write(*,*) '***',i,((col(ii,isetp),ii=1,2),isetp=1,nsetp)
  timr1(i) = col(1, nsetp)
  clconc1(i) = col(2, nsetp)
100 continue
tmax = timr1(nhist4)
c print*, 'nhist4,tmax=',nhist4,tmax

do 200 i = 1, nhist
  time = timr(i)
  if ( time.lt.tmax ) then
    call lint(timr1, clconc1, nintv, nhist4, time, yyyy)
    clconc(i) = yyyy
    yprevious = yyyy
  else
    yyyy = yprevious
    clconc(i) = yyyy
  end if
c print*, 'from chloride timr,clconc=',timr(i),clconc(i)
200 continue

300 continue
return
end
```

**==ODEINT.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

subroutine odeint(ystart, nvar, x1, x2, eps, h1, hmin, hmax,

```

&      nok, nbad, tiny, derivs, rkqc, ilayer,
&      cthick1)

implicit none

c*** Start of declarations inserted by SPAG
      real*8 cthick1, dydx, eps, errc, h, h1, hdid, hmax, hmin, hnext,
&      tiny, x, x1, x2, y, yscal, ystart, zero
      integer i, ieq, ilayer, maxstp, nbad, nmax, nok, nstp, nvar
c*** End of declarations inserted by SPAG

```

```

c-----
c      Purpose: subroutine implementing Runge-Kutta Integration algorithm
c-----
crrrr  parameter (maxstp=50000,nmax=50,two=2.0,zero=0.0)
crrrr  parameter (maxstp=50000, nmax=50, zero=0.0)
crrrr  dimension ystart(nvar), yscal(nmax), y(nmax), dydx(nmax)
crrrr  common /wetting/twet
crrrr  external rkqc, derivs
c-----

```

```

c      nomenclature:

c      nmax:  maximum number of functions
c      nvar:  a subset of nmax, i.e., nvar<nmax (# of starting values)
c      h:     step size that can be positive or negative
c      yscal: a vector against which the error is scaled
c      hdid:  step size which was actually accomplished
c      hnext: estimated next step size
c      eps:   accuracy to which integration to be done
c      h1:    guessed first step size
c      hmin:  minimum allowed step size (can be zero)
c      nok:   number of good steps taken
c      nbad:  number of bad steps taken (but retried and fixed)
c      ystart: this is replaced by values at the end of the
c              integration interval

c      ieq:   equation controlling step size,
c      errc:  critical error

c      Note: the user must provide a value of the derivative at the
c      starting point.
c-----

```

```

      h = sign(h1, x2 - x1)
      x = x1
      nok = 0
      nbad = 0
crrrr      ifudge = 0
      do 100 i = 1, nvar
         y(i) = ystart(i)
100      continue

200      continue
      do 300 nstp = 1, maxstp
c      print*,y(1),ilayer
         if ( y(1).ge.cthick1 ) ilayer = 2
cdbg      print*, 'printing from odeint : before derivs'
         call derivs(x, y, dydx, ilayer)
cdbg      print*, 'printing from odeint : after derivs'
         do 250 i = 1, nvar
            yscal(i) = abs(y(i)) + abs(h*dydx(i)) + tiny
c            yscal(i)=abs(y(i)) + tiny

```

26/195

```

250  continue
      if ( (x+h-x2)*(x+h-x1).gt.zero ) h = x2 - x
      if ( h.gt.hmax ) h = hmax
cdbg  print*, 'printing from odeint : before rkqc'
      call rkqc(y, dydx, nvar, x, h, eps, yscal, hdid, hnext, ieq,
&      errc, derivs, ilayer)
cdbg  print*, 'printing from odeint : after rkqc'
      if ( hdid.eq.h ) then
        nok = nok + 1
      else
        nbad = nbad + 1
      end if
      if ( (x-x2)*(x2-x1).ge.zero ) then
        do 260 i = 1, nvar
          ystart(i) = y(i)
260   continue
csss  eps = eps / (10.**ifudge)
csss  tiny = tiny / (10.**ifudge)
      return
      end if
      if ( abs(hnext).lt.hmin ) then
        write (*, *)
&      'runge-kutta error messages,                step size too small'
        write (*, *) 'time = ', x
        write (*, *) 'i, y(i), dydx(i), yscal(i)'
        do 280 i = 1, nvar
          write (*, *) i, y(i), dydx(i), yscal(i)
280   continue
        write (*, *) 'equation controlling step size is: ', ieq
        write (*, *) 'controlling equation error is: ', errc
        write (*, *) 'step size is: ', hdid

csss  ifudge = ifudge+1
csss  eps = eps * 10.
csss  tiny = tiny * 10.
      end if

      h = hnext
300  continue
      write (*, *) 'runge-kutta error messages'
      write (*, *) 'i, y(i), dydx(i), yscal(i)'
      write (*, *) 'time = ', x
      do 400 i = 1, nvar
        write (*, *) 'from odeint**', y(i), dydx(i), yscal(i)
400  continue
        write (*, *) 'equation controlling step size is: ', ieq
        write (*, *) 'controlling equation error is: ', errc
        write (*, *) 'step size is: ', hdid

csss  ifudge = ifudge+1
csss  eps = eps * 10.
csss  tiny = tiny * 10.
      go to 200

c     pause 'too many steps.'

c     return
      end

```

```

subroutine rkqc(y, dydx, n, x, htry, eps, yscal, hdid, hnext,
&            ieq, errc, derivs, ilayer)
  implicit none
C*** Start of declarations inserted by SPAG
  real*8 dydx, dysav, eps, errc, errcon, errmax, fcor, h, hdid,
&      hh, hnext, htry, one, pgrow, pshrnk, safety, x, xsav, y,
&      ysav
  real*8 yscal, ytemp
  integer i, ieq, ilayer, n, nmax
C*** End of declarations inserted by SPAG

```

```

C-----
C      quality controlled runge kutta integration routine
C      from numerical recipes
C-----
  parameter (nmax=50, fcor=1./15., one=1., safety=0.9,
&           errcon=6.E-4)
  dimension y(n), dydx(n), yscal(n), ytemp(nmax), ysav(nmax),
&           dysav(nmax)
crrrr common /wetting/twet
  external derivs
C-----

```

```

c      Nomenclature:

c      x:      independent variable
c      htry:   step size to be attempted
c      n:      length of the dependent variable vector y
c      ytemp:  an array containing error estimates
c      derivs: name of the subroutine with the derivative of the eq.
c              to be integrated
c      y(i):   dependent variable vector
c      dydx:   derivative of the dependent variable
c      eps:    integration to be done with required overall tolerance eps
c      hdid:   step size which was actually accomplished
c      hnext:  estimated next stepsize
C-----

```

```

  pgrow = -0.20
  pshrnk = -0.25
  xsav = x
  do 100 i = 1, n
    ysav(i) = y(i)
    dysav(i) = dydx(i)
100  continue
  h = htry
200  continue
  hh = 0.5*h
  call rk4(ysav, dysav, n, xsav, hh, ytemp, derivs, ilayer)
  x = xsav + hh
  call derivs(x, ytemp, dydx, ilayer)
  call rk4(ytemp, dydx, n, x, hh, y, derivs, ilayer)
  x = xsav + h
  if ( x.eq.xsav ) then
c      pause 'stepsize not significant in rkqc.'
    write (*, *) 'x and xsav=', x, xsav
    write (*, *) 'step size not significant'

```

28/195

```

write (*, *) 'the code is forcibly stopped'
stop
end if
call rk4(ysav, dysav, n, xsav, h, ytemp, derivs, ilayer)
errmax = 0.
errc = 0.
ieq = 0

```

c make this routine return the equation controlling
c the step size, put in variable "ieq", the error is returned
c in "errc"

```

do 300 i = 1, n
  ytemp(i) = y(i) - ytemp(i)
  if ( abs(ytemp(i)/yscal(i)).gt.errmax ) then
    ieq = i
    errc = ytemp(i)
  end if
  errmax = max(errmax, abs(ytemp(i)/yscal(i)))
300 continue

```

```

errmax = errmax/eps
if ( errmax.gt.one ) then
  h = safety*h*(errmax**pshrnk)
  go to 200
else
  hdid = h
  if ( errmax.gt.errcon ) then
    hnext = safety*h*(errmax**pgrow)
  else
    hnext = 4.*h
  end if
end if

```

```

do 400 i = 1, n
  y(i) = y(i) + ytemp(i)*fcor
400 continue

```

```

return
end

```

**=RK4.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

subroutine rk4(y, dydx, n, x, h, yout, derivs, ilayer)
implicit none

```

C*** Start of declarations inserted by SPAG
real*8 dydx, dym, dyt, h, h6, hh, twet, x, xh, y, yout, yt
integer i, ilayer, n, nmax
C*** End of declarations inserted by SPAG
common /wetting/ twet

c-----
c Basic Runge-Kutta Method for integration
c-----

```

parameter (nmax=50)
external derivs
dimension y(n), dydx(n), yout(n), yt(nmax), dyt(nmax), dym(nmax)
hh = h*0.5
h6 = h/6.
xh = x + hh

```


29/AS

```

do 100 i = 1, n
  yt(i) = y(i) + hh*dydx(i)
100 continue
  call derivs(xh, yt, dyt, ilayer)
  do 200 i = 1, n
    yt(i) = y(i) + hh*dym(i)
200 continue
    call derivs(xh, yt, dym, ilayer)
    do 300 i = 1, n
      yt(i) = y(i) + h*dym(i)
      dym(i) = dyt(i) + dym(i)
300 continue
      call derivs(x + h, yt, dyt, ilayer)
      do 400 i = 1, n
        yout(i) = y(i) + h6*(dydx(i) + dyt(i) + 2.*dym(i))
400 continue

      return
      end

```

**==BSSTEP.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

subroutine bsstep(y, dydx, nv, x, htry, eps, yscal, hdid, hnext,
& ieq, errc, derivs)
  implicit none
C*** Start of declarations inserted by SPAG
  real*8 derivs, dydx, dysav, eps, errc, errmax, grow, h, hdid,
& hnext, htry, one, shrink, x, xest, xsav, y, yerr, ysav,
& yscal
  real*8 yseq
  integer i, ieq, imax, j, nmax, nseq, nuse, nv
C*** End of declarations inserted by SPAG

```

```

c-----
  external derivs
  parameter (nmax=50, imax=11, nuse=7, one=1.E0, shrink=.95E0,
& grow=1.2E0)
& dimension y(nv), dydx(nv), yscal(nv), yerr(nmax), ysav(nmax),
& dysav(nmax), yseq(nmax), nseq(imax)
csss common /wetting/twet
  data nseq/2, 4, 6, 8, 12, 16, 24, 32, 48, 64, 96/

```

```

c-----
c   n:          number of variables
c   h:          r-k to advance the solution over an interval h
c   xh:
c   y(i):       dependent variables
c   dydx(i):    derivatives of dependent variables
c   yout(i):    an array containing the incremented y values
c-----

```

```

c
  h = htry
  xsav = x
  do 100 i = 1, nv
    ysav(i) = y(i)
    dysav(i) = dydx(i)
100 continue
200 continue
  do 300 i = 1, imax
    call mmid(ysav, dysav, nv, xsav, h, nseq(i), yseq, derivs)
    xest = (h/nseq(i))**2

```

30/195

```

call rzextr(i, xest, yseq, y, yel, nv, nuse)
errmax = 0.
do 250 j = 1, nv
  errmax = max(errmax, abs(yerr(j)/yscal(j)))
250 continue
errmax = errmax/eps
if ( errmax.lt.one ) then
  x = x + h
  hdid = h
  if ( i.eq.nuse ) then
    hnext = h*shrink
  else if ( i.eq.nuse - 1 ) then
    hnext = h*grow
  else
    hnext = (h*nseq(nuse-1))/nseq(i)
  end if
  return
end if
300 continue
h = 0.25*h/2**((imax-nuse)/2)
if ( x + h.eq.x ) pause 'step size underflow.'
go to 200

```

end

**==MMID.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

subroutine mmid(y, dydx, nvar, xs, htot, nstep, yout, derivs)
implicit none
C*** Start of declarations inserted by SPAG
real*8 dydx, h, h2, htot, swap, x, xs, y, ym, yn, yout
integer i, ilayer, n, nmax, nstep, nvar
C*** End of declarations inserted by SPAG
crrrr common /wetting/twet
c-----
external derivs
parameter (nmax=50)
dimension y(nvar), dydx(nvar), yout(nvar), ym(nmax), yn(nmax)
h = htot/nstep
do 100 i = 1, nvar
  ym(i) = y(i)
  yn(i) = y(i) + h*dydx(i)
100 continue
x = xs + h
call derivs(x, yn, yout, ilayer)
h2 = 2.*h
do 200 n = 2, nstep
  do 150 i = 1, nvar
    swap = ym(i) + h2*yout(i)
    ym(i) = yn(i)
    yn(i) = swap
150 continue
x = x + h
call derivs(x, yn, yout, ilayer)
200 continue
do 300 i = 1, nvar
  yout(i) = 0.5*(ym(i) + yn(i) + h*yout(i))
300 continue

```

31/145

```
return  
end
```

**==RZEXTR.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```
subroutine rzextr(iest, xest, yest, yz, dy, nv, nuse)  
implicit none
```

C*** Start of declarations inserted by SPAG

```
real*8 b, b1, c, d, ddy, dy, fx, v, x, xest, yest, yy, yz  
integer iest, imax, j, k, m1, ncol, nmax, nuse, nv
```

C*** End of declarations inserted by SPAG

```
c-----  
parameter (imax=11, nmax=50, ncol=7)  
dimension x(imax), yest(nv), yz(nv), dy(nv), d(nmax, ncol),  
&          fx(ncol)  
crrrr common /wetting/twet  
c-----
```

```
x(iest) = xest  
if ( iest.eq.1 ) then  
  do 50 j = 1, nv  
    yz(j) = yest(j)  
    d(j, 1) = yest(j)  
    dy(j) = yest(j)  
50  continue  
else  
  m1 = min(iest, nuse)  
  do 100 k = 1, m1 - 1  
    fx(k + 1) = x(iest - k)/xest  
100 continue  
  do 150 j = 1, nv  
    yy = yest(j)  
    v = d(j, 1)  
    c = yy  
    d(j, 1) = yy  
    do 120 k = 2, m1  
      b1 = fx(k)*v  
      b = b1 - c  
      if ( b.ne.0. ) then  
        b = (c - v)/b  
        ddy = c*b  
        c = b1*b  
      else  
        ddy = v  
      end if  
      v = d(j, k)  
      d(j, k) = ddy  
      yy = yy + ddy  
120  continue  
      dy(j) = ddy  
      yz(j) = yy  
150  continue  
end if  
return  
end
```

**==RATE.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

32/195

```

subroutine rate(time, odev, dvdt, ilayer)
  implicit none
C*** Start of declarations inserted by SPAG
  real*8 cfactor, clconc, clconca, clconcb, clcrit1, clcrit2,
  & crate, dvdt, humd, odev, tavg, tcan, time, timr, tin, xin
  integer i, iflagcl, ilayer, nhist, nintv, nv
C*** End of declarations inserted by SPAG
c-----
c   this returns derivatives for the main calculations
c-----
  parameter (nintv=20000)
  parameter (nv=1)
crrrr   common /bulk/ age,tend,x,y,z,d,filmthk,e,cthick1,cthick2,sfactor
  dimension odev(nv), dvdt(nv)
  common /thist / timr(nintv), tavg(nintv), tcan(nintv),
  & humd(nintv)
crrrr   common /corrpar1/ecor,ecrit,cpass,sollay,curact,
crrrr   & aa(2,3),xread,refph,betaox1,betahy1,
crrrr   & betaox2,betahy2,rkox1,rkhy1,rkox2,rkhy2,
crrrr   & gox1,ghy1,gox2,ghy2,spor,taus,xcouple
c   common /corrpar2/ xipto, slpto, xirpo, slrpo, xipti, slpti,
c   & xirpi, slrpi
crrrr   common /solve/dtini,dtmax,errrel,errabs
  common /history/ nhist
  common /new1 / clconca, clcrit1, clcrit2, clconc(nintv),
  & cfactor, iflagcl
crrrr   common /wetting/twet
c-----

c   nomenclature:

c   timr(i):representative time in years
c   tcan(i):waste package surface temperature in Celsius
c   nintv:  this is used in stead of nhist when simulation is to be done
c           for large time, especially when empirical temp, rh are used
c   nhist:  number of time intervals at which temperature, rh are stored
c   time:   time at which linear interpolation is done for the temp.
c   tin:    interpolated temperature
c   nv:     size of the vector odev
c   odev:   dependent variable
c   clconc: chloride ion concentration near the WP [mol/liter]
c   crate:  corrosion rate in m/yr
c   dvdt:   derivative of the dependent variable
c-----

  xin = 0.8
  call lint(timr, tcan, nintv, nhist, time, tin)
  call lint(timr, clconc, nintv, nhist, time, clconcb)
  do 100 i = 1, nv
    odev(i) = max(odev(i), 0.)
100 continue

  call corrode(tin, time, xin, clconcb, crate, ilayer, clcrit1,
  & clcrit2, iflagcl)

  dvdt(1) = crate

c   print*, 'from rate: time,tin,dvdt(1)=', time,tin,dvdt(1)

```

```
return
end
```

```
**==CORRODE.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997
```

```
subroutine corrode(temp, time, xin, clconcb, crate, ilayer,
&                clcrit1, clcrit2, iflagcl)
implicit none
C*** Start of declarations inserted by SPAG
real*8 aa, age, bb, berp, berp0, berp1, betahy, betahy1,
&      betahy2, betaox, betaox1, betaox2, cathod, cbulko2,
&      cfarad, clconcb, clcrit1, clcrit2, conv, cpass
real*8 crate, crate2, cthick1, cthick2, curact, curhy, curmax,
&      curox, d, ddw, delt, dense1, dense2, do2w, dzc, e, ecor,
&      ecor1, ecrit, ecthdr
real*8 eec, eexpt, erp0, erp00, erp01, filmthk, ghy, ghy1, ghy2,
&      gox, gox1, gox2, ph, pttemi, pttemo, rcoef, refph,
&      rexpont, rgas, rkhy
real*8 rkhy1, rkhy2, rkox, rkox1, rkox2, rptemi, rptemo,
&      sfactor, slpti, slpto, slpttemi, slpttemo, slrpi, slrpo,
&      slrptemi, slrptemo, sollay, spor, taus, tdiff
real*8 temp, tend, time, tk, tkref, twet, wtmol1, wtmol2, x,
&      xcouple, xgas, xin, xipti, xipto, xirpi, xirpo, xkh,
&      xko2, xread, xxx1
real*8 xxx2, xxx3, y, yko2, yyy1, z, zc, zo2
integer i, icorr, iflagcl, ilayer
```

```
C*** End of declarations inserted by SPAG
```

```
c-----
c corrosion rate & corrosion potential in the water film on the wp
c-----
```

```
common /bulk / age, tend, x, y, z, d, filmthk, e, cthick1,
&      cthick2, sfactor
common /corrpar1/ ecor, ecrit, cpass, sollay, curact, aa(2, 3),
&      xread, refph, betaox1, betahy1, betaox2,
&      betahy2, rkox1, rkhy1, rkox2, rkhy2, gox1,
&      ghy1, gox2, ghy2, spor, taus, xcouple
c common /corrpar2/ xipto, slpto, xirpo, slrpo, xipti, slpti,
c &      xirpi, slrpi
dimension bb(6)
common /wetting/ twet
common /cripotout/ xipto, pttemo, slpto, slpttemo, xirpo,
&      rptemo, slrpo, slrptemo
common /cripotin/ xipti, pttemi, slpti, slpttemi, xirpi, rptemi,
&      slrpi, slrptemi
common /prate / rcoef, rexpont, crate2, eexpt
common /flags / icorr
```

```
c-----
c ecrit: critical potential for localized corrosion (volts SHE)
c refph: reference ph
c aa(1,3): cpass = aa(1,1)+aa(1,2)*T+aa(1,3)*T**2
c aa(2,3): cpass = aa(2,1)+aa(2,2)*T+aa(2,3)*T**2
c bb(6): ecrit = bb(1)+bb(2)*T+bb(3)*T**2+
c          bb(4)* Log10(molar chloride)+bb(5)*no3/Cl+
c          bb(6)*(ph-phref)
c xipto: outer overpack Ep intercept
c pttemo: temp. coef. of outer overpack Ep intercept
```

```

c slpto: outer overpack Ep slope
c slpttemo:temp. coef. of outer overpack Ep slope
c xirpo: outer overpack Erp intercept
c rptemo: temp. coef. of outer overpack Erp intercept
c slrpo: outer overpack Erp slope
c slrptemo:temp. coef. of outer overpack Erp slope
c xipti: inner overpack Ep intercept
c pttemi: temp. coef. of inner overpack Ep intercept
c slpti: inner overpack Ep slope
c slpttemi:temp. coef. of inner overpack Ep slope
c xirpi: inner overpack Erp intercept
c rptemi: temp. coef. of inner overpack Erp intercept
c slrpi: inner overpack Erp slope
c slrptemi:temp. coef. of inner overpack Erp slope
c betaox1, betahy1: beta kinetics parameters for oxygen and water
                    for WP outer overpack
c betaox2, betahy2: beta kinetics parameters for oxygen and water
                    for WP inner overpack
c tkref: reference temperature [k]; kinetic data is for 25 celsius,
        therefore use only that temperature here
c rgas: universal gas constant [j/mole*k]
c xgas: partial pressure of oxygen
c xkh: Henry's law constant for oxygen solubility
c eec: previous ee
c cfarad: faraday constant [coulomb/equivalent]
c rkox: rate constant for oxygen reduction (coulomb m/mole/yr)
c gox: activation energy for oxygen rate constant (J/mole)
c sollay: thickness of scale layer (m)
c filmthk:thickness of water film in m prior to effluent creation
c taus: tortuosity/geometry correction for liquid diffusion
        of oxygen through boiler scale
c spor: porosity of scale deposited on the wp [=1.0 fir liq. film]
c betaox: beta kinetics parameter for oxygen reduction
c ghy: activation energy for water reduction (J/mole)
c rkhy: rate constant for water redution in (coulomb/m^2/yr)
c curox: current density for oxygen reduction
c curhy: current density for hydrogen ion reduction
c cathod: cathodic current density
c do2w: diffusion coefficient of oxygen in aqueous sol.[m^2/yr]
c zo2: # of electrons involved in the process/mole
c ddw: diffusivity of molecular oxygen in water at 25 C
c wtmol1: molecular weight of outer layer [kg/mole]
c wtmol2: molecular weight of inner layer [kg/mole]
c dense1: density of outer layer [kg/m^3]
c dense2: density of inner layer [kg/m^3]
c crate: corrosion rate of the outer overpack [m/yr]
c crate2: corrosion rate of the inner overpack[m/yr]
c xread: previous bb(1) is now xread so that bb(1) etc. are properly
        assigned to obtain xread
c cpass: passive current density in Coulomb/m^2/yr
c ecor: corrosion potential (volts SHE)
c xcouple:a fraction representing degree of coupling bet. two layers
c cbulko2:outer oxygen concentration in moles per m^3 of water
c tk: temperature in kelvin
c temp: temperature in celcius
c time: time in years
c xxx1,xxx2,xxx3,yyy1: local intermediate variables

```

```

c note: xin-> mole fraction water vapor on waste package surface
c-----

```

```

cfarad = 96485.
rgas = 8.3144
ddw = 0.063
dense1 = 7860.
wtmol1 = 0.05585
dense2 = 8140.
wtmol2 = 0.05747

```

```

sollay = 0.0
ph = refph
tkref = 273.15 + 25.
tk = 273.15 + temp

```

```

if ( ilayer.eq.1 ) then
  betaox = betaox1
  betahy = betahy1
  rkox = rkox1
  rkhy = rkhy1
  gox = gox1
  ghy = ghy1

```

```

c--> critical potentials data for outer layer (A516 steel)

```

```

c      xipto = -584.8
c      pttemo = 3.92
c      slpto = -24.5
c      slpttemo = -1.1
c      xirpo = -620.3
c      rptemo = 0.47
c      slrpo = -95.2
c      slrptemo = 0.88

```

```

erp00 = xipto + pttemo*temp
berp0 = slpto + slpttemo*temp
erp01 = xirpo + rptemo*temp
berp1 = slrpo + slrptemo*temp
erp0 = erp01 + xread*(erp00 - erp01)
bb(1) = erp0
bb(2) = 0.0
bb(3) = 0.0
berp = berp1 + xread*(berp0 - berp1)
bb(4) = berp
ecrit = bb(1) + bb(4)*log10(clconcb)

```

```

c      divide by 1000. to convert from mV to V

```

```

ecrit = ecrit/1000.
cpass = aa(1, 1) + aa(1, 2)*tk + aa(1, 3)*tk*tk

```

```

else

```

```

  betaox = betaox2
  betahy = betahy2
  rkox = rkox2
  rkhy = rkhy2
  gox = gox2
  ghy = ghy2

```

```

c--> critical potentials data for inner layer (Alloy 825)

```

```

c      xipti = 200.
c      pttemi = 0.
c      slpti = -240.
c      slpttemi = 0.
c      xirpi = 422.8

```

36/155

```

c      rptemi = -4.1
c      slrpi = -64.
c      slrptemi = -0.80

      erp00 = xipti + pttemi*temp
      berp0 = slpti + slpttemi*temp
      erp01 = xirpi + rptemi*temp
      berp1 = slrpi + slrptemi*temp
      erp0 = erp01 + xread*(erp00 - erp01)
      bb(1) = erp0
      bb(2) = 0.0
      bb(3) = 0.0
      berp = berp1 + xread*(berp0 - berp1)
      bb(4) = berp
      ecrit = bb(1) + bb(4)*log10(clconcb)
c      divide by 1000. to convert from mV to V
      ecrit = ecrit/1000.
      cpass = aa(2, 1) + aa(2, 2)*tk + aa(2, 3)*tk*tk

end if

c      print*,'erp0,berp,clconcb=',erp0,berp,clconcb

      if ( cpass.lt.0. ) then
        write (*, *) '-ve passive current density =', cpass
        stop 'bad passive current density'
      end if

c--> calculate oxygen concentration against metal using temperature,
c--> henry's law, amount of air present, passive current density, and
c--> thickness of water film

      yyy1 = (1./tkref - 1./tk)/rgas

c      partial pressure of oxygen over the solution; this is a constant
c      here because we are assuming that the o2 in the atmosphere is
c      in equilibrium with the liquid film on the waste package.
c      xgas=0.21*(1.0-xin)
c      cbulko2=xgas*henry*1000.

      xgas = 0.21
      ecthdr = 14612.
      do2w = ddw*exp(ecthdr*yyy1)
      xkh = exp(0.2984 - (5.59617*10.**3)/tk + (1.04967*10.**6)/tk/tk)
      cbulko2 = xgas*xkh*1000.

c--> estimate corrosion potential by equating oxygen reduction plus
c--> hydrogen evolution with passive current density

      deltt = 2.0
      xxx1 = betahy*cfarad/(rgas*tk)
      xko2 = rkox*exp(gox*yyy1)
c      is k298 assumed to be 1.0 ?
      xxx2 = xko2*max(sollay, filmthk)/(4.0*cfarad*do2w*taus*spor)
      zo2 = 4.0
      xxx3 = betaox*zo2*cfarad/(rgas*tk)

      if ( ilayer.ne.2 ) then
        eec = 0.
        do 50 i = 1, 15

```



```

yko2 = rkhy*exp(ghy*yyy1)
curhy = -yko2*exp( - xxx1*eec)

```

c note: oxygen current must be constrained to < diffusion
c limited current curmax. since cathodic current is negative,
c take the maximum value to get the lowest absolute value

```

curmax = -4.*cfarad*do2w*taus*spor*cbulko2/max(sollay,
& 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)
if ( cathod.gt.cpass ) then
  eec = eec + deltt
else
  eec = eec - deltt
end if
delt = deltt/2.0
50 continue

```

c--> ensure that the summation of all currents equal to zero,
c--> newton-raphson method is used as an additional step

```

do 100 i = 1, 10
  yko2 = rkhy*exp(ghy*yyy1)
  curhy = -yko2*exp( - xxx1*eec)
  curmax = -4.*cfarad*do2w*taus*spor*cbulko2/max(sollay,
& 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
  dzc = -xxx1*curhy - xxx3*curox/(1 + xxx2*exp(-xxx3*eec))
  if ( abs(zc).le.1.0E-8 ) go to 150
  eec = eec - zc/dzc
100 continue

write (*, *) 'corrosion potential not converging in 10 ',
& 'iterations.'

```

c--> now compare potentials with critical potentials

```

150 continue
ecor1 = eec
end if
if ( ilayer.ne.1 ) then
  eec = ecor1
cssss eexpt = -0.46
  eec = eec*(1. - xcouple) + xcouple*eexpt
c print*, 'xcouple,eec,ecrit=',xcouple,eec,ecrit
end if
ecor = eec
iflagcl = 0
c print*,clconcb,clcrit1,iflagcl
if ( ilayer.eq.1 .and. clconcb.gt.clcrit1 ) iflagcl = 1
if ( ilayer.eq.2 .and. clconcb.gt.clcrit2 ) iflagcl = 1

```

30/195

```

if ( eec.ge.ecrit .and. iflagcl.eq.1 ) then
  icorr = 1
  if ( ilayer.eq.1 ) then
    rexpont=0.45
    rcoef=8.66e-3
    tdiff = time - twet
    if ( tdiff.le.1E-8 ) tdiff = 1.E-8
    crate = rcoef*rexpont*tdiff**(rexpont - 1)
    ! m/yr corrosion rate
    print*,'time,twet=',time,twet
  else
    crate = crate2
  end if

cdbg   if(time.gt.2700) write(9,*) time,eec,ecrit,crate
else
  icorr = 0
  if ( ilayer.eq.1 ) then
    conv = wtmol1/dense1/cfarad/2.
  else
    conv = wtmol2/dense2/cfarad/2.25
  end if
  crate = cpass*conv

cdbg   if(time.gt.2700) write(9,*) time,eec,ecrit,crate
end if

c      penetration (mm)= 8.66 t**0.45 with t in yr.
c      print*,'time,temp,ecrit,eec,crate=', time,temp,ecrit,eec,crate

return
end
**==LINT.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

```

subroutine lint(xa, ya, n1, n, x, y)
  implicit none
  C*** Start of declarations inserted by SPAG
  real*8 h, x, xa, y, ya
  integer k, khi, klo, n, n1
  C*** End of declarations inserted by SPAG
  -----
  c      subroutine for linear interpolation
  -----
  c      nomenclature:
  c      xa:   vector of length n of x values (independent variable)
  c      ya:   vector of length n of y values
  c      n:    length of xa and ya vectors
  c      x:    x value at which interpolation is required
  c      y:    returned value of dependent variable corresponding to x value
  -----
  dimension xa(n1), ya(n1)
  -----

  if ( xa(1).lt.xa(n) ) then
c increasing order list

```

39) 195

```

if ( x.le.xa(1) ) then
  y = ya(1)
else if ( x.ge.xa(n) ) then
  y = ya(n)
else
  klo = 1
  khi = n
20  continue
  if ( khi - klo.gt.1 ) then
    k = (khi + klo)/2
    if ( xa(k).gt.x ) then
      khi = k
    else
      klo = k
    end if
    go to 20
  end if
  h = xa(khi) - xa(klo)
  if ( h.eq.0. ) pause 'bad xa input.'
  y = ya(klo) + (x - xa(klo))/h*(ya(khi) - ya(klo))
end if

c decreasing order list
else if ( x.ge.xa(1) ) then
  y = ya(1)
else if ( x.le.xa(n) ) then
  y = ya(n)
else
  klo = 1
  khi = n
50  continue
  if ( khi - klo.gt.1 ) then
    k = (khi + klo)/2
    if ( xa(k).lt.x ) then
      khi = k
    else
      klo = k
    end if
    go to 50
  end if
  h = xa(khi) - xa(klo)
  if ( h.eq.0. ) pause 'bad xa input.'
  y = ya(klo) + (x - xa(klo))/h*(ya(khi) - ya(klo))
end if

return
end

```

**==MECH.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

subroutine mech(jj, time1, time2, temp, penet, ifail)
implicit none
C*** Start of declarations inserted by SPAG
  real*8 age, cdepth, cthick1, cthick2, d, e, filmthk, penet,
&         sfactor, strss, temp, tend, time1, time2, x, y, yieldstr,
&         z
  integer ifail, jj
C*** End of declarations inserted by SPAG
c-----

```

40/195

c subroutine to test WP for mechanical failure by fracture due to residual stress associated with welds in the presence of a flaw of a given depth produced by localized corrosion

c-----
common /bulk / age, tend, x, y, z, d, filmthk, e, cthick1,
& cthick2, sfactor
crrrr common /wpgmty/ wplen,wpdia
c-----

c nomenclature:
c temp: temperature at the wp surface [c]
c ifail: =0: wp not failed; =1: wp failed
c cdepth: crack depth or the depth of a corroded pit [m]
c penet:
c strss: tensile component of the residual stress [Pa]
c yieldstr: yield strength of A516 Grade 55 steel [Pa]
c pi: the constant pi
c r:
c sfactor: safety factor considering the increase of yield strength at the welds
c-----

c debugging parameters jj, time1, time2

crrrr print*, jj,time1,time2

ifail = 0
crrrr pi = 4. * atan(1.)
sfactor = 1.5
yieldstr = 205.E6

c limit temperature to the applicable range of the stress equ.

if (temp.lt.25.) temp = 25.
if (temp.gt.100.) temp = 100.
cdepth = penet
strss = yieldstr*sfactor

call fracfail(strss, cdepth, ifail)

cssss

c print*
c print*, 'punifo,cdepth,clt,strss=',punifo,cdepth,clt,strss
c print*, 'time1,penet',time1,penet
c print*

return
end

**==FRACFAIL.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

subroutine fracfail(strss, cdepth, ifail)
implicit none
C*** Start of declarations inserted by SPAG
real*8 cdepth, dki, dkic, pi, strss, yfunc, yvalue
integer ifail
C*** End of declarations inserted by SPAG

c-----
c fracture failure analysis based on stress intensity calculation
c-----

41/195

```
c nomenclature:
c cdepth:      crack depth or the depth of a corroded pit [m]
c yvalue:     geometry factor
c dkic:       fracture toughness of A516 Grade 55 steel
c             =250 [MPa-m**0.5]
c yfunc:      a function call
c strss:      tensile component of the residual stress [Pa]
c pi:         the constant pi
c ifail:
```

```
-----
pi = 4.*atan(1.)
dkic = 250.E6
yvalue = yfunc(cdepth)
dki = strss*yvalue*(pi*cdepth)**.5
if ( dki.gt.dkic ) ifail = 1
```

cssss

```
c print*
c print*,'dki,yvalue,cdepth=', dki,yvalue,cdepth
c print*
```

```
return
end
```

**==YFUNC.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```
real*8 function yfunc(tt)
implicit none
```

C*** Start of declarations inserted by SPAG

```
real alow, atop, tt, xx, yy
integer i
```

C*** End of declarations inserted by SPAG

```
dimension xx(30), yy(30)
```

```
-----
c function subprogram to determine the geometry factor for the
c fracture toughness failure criterion. This geometry value is for
c a surface crack in hollow cylinder, circular on outer area.
c this value is calculated by using nasa flagro program.
-----
```

c store the table data

```
xx(1) = 0.00001
xx(2) = 0.00005
xx(3) = 0.0001
xx(4) = 0.0002
xx(5) = 0.0005
xx(6) = 0.00075
xx(7) = 0.001
xx(8) = 0.002
xx(9) = 0.003
xx(10) = 0.004
xx(11) = 0.005
xx(12) = 0.006
xx(13) = 0.007
xx(14) = 0.008
xx(15) = 0.009
```

42/195

```

xx(16) = 0.0095
xx(17) = 0.0097
xx(18) = 0.0098
xx(19) = 0.0099
xx(20) = 0.00995
yy(1) = 0.8817
yy(2) = 0.8826
yy(3) = 0.8837
yy(4) = 0.8858
yy(5) = 0.8923
yy(6) = 0.8977
yy(7) = 0.9030
yy(8) = 0.9247
yy(9) = 0.9463
yy(10) = 0.9681
yy(11) = 0.9899
yy(12) = 1.0118
yy(13) = 1.0338
yy(14) = 1.0559
yy(15) = 1.0780
yy(16) = 1.0891
yy(17) = 1.0935
yy(18) = 1.0958
yy(19) = 1.0980
yy(20) = 1.0991

```

```

c      find location for tt

```

```

      if ( tt.le.xx(1) ) then
        yfunc = yy(1)
        return
      end if

```

```

      if ( tt.ge.xx(20) ) then
        yfunc = yy(20)
        return
      end if

```

```

      do 100 i = 1, 20
        if ( tt.le.xx(i) ) go to 200

```

```

100    continue

```

```

200    continue
        atop = xx(i)
        alow = xx(i - 1)
        yfunc = (tt - alow)/(atop - alow)*(yy(i) - yy(i-1))

```

```

      return
    end

```

```

**==INPUT.spg processed by SPAG 4.00Aa at 14:01 on 7 Jan 1997

```

```

subroutine input
implicit none

```

```

C*** Start of declarations inserted by SPAG

```

```

real*8 aa, age, betahy1, betahy2, betaox1, betaox2, cfactor,
&      clconc, clconca, clcrit1, clcrit2, constant1, cpass,

```

```

&      crate2, ctemp, cthick1, cthick2, curact, d, dtini
real*8 dtmax, e, ecor, ecrit, eexpt, errabs, errrel, filmthk,
&      gbthick, ghy1, ghy2, gox1, gox2, grainr, humdc, pttemi,
&      pttemo, rcoef, refph, rexpont
real*8 rkhy1, rkhy2, rkox1, rkox2, rptemi, rptemo, sfactor,
&      slpti, slpto, slpttemi, slpttemo, slrpi, slrpo, slrptemi,
&      slrptemo, sollay, spor, taus, tend, timintv
real*8 wpdia, wplen, x, xcouple, xipti, xipto, xirpi, xirpo,
&      xread, y, z
integer iflag, iflagcl, nhista, nintv, nseries, nset
C*** End of declarations inserted by SPAG
c-----
c      purpose: read input parameters
c      most variables are passed in common blocks and all the input is simple
c      non-formatted and could be easily upgraded at a later date.
c-----
c      parameter (nintv=20000)
c      common /bulk / age, tend, x, y, z, d, filmthk, e, cthick1,
&      cthick2, sfactor
crrrr      common /thist/timr(nintv), tavg(nintv), tcan(nintv), humd(nintv)
c      common /corrpar1/ ecor, ecrit, cpass, sollay, curact, aa(2, 3),
&      xread, refph, betaox1, betahy1, betaox2,
&      betahy2, rkox1, rkhy1, rkox2, rkhy2, gox1,
&      ghy1, gox2, ghy2, spor, taus, xcouple
c      common /corrpar2/ xipto, slpto, xirpo, slrpo, xipti, slpti,
c      &      xirpi, slrpi
c      common /solve / dtini, dtmax, errrel, errabs
c      common /temp1 / iflag, nset, timintv, humdc, ctemp, nhista
c      common /dryoxdc/ grainr, gbthick, constant1, nseries
c      common /new1 / clconca, clcrit1, clcrit2, clconc(nintv),
&      cfactor, iflagcl
c      common /wpgmt/ wplen, wpdia
c      common /cripotout/ xipto, pttemo, slpto, slpttemo, xirpo,
&      rptemo, slrpo, slrptemo
c      common /cripotin/ xipti, pttemi, slpti, slpttemi, xirpi, rptemi,
&      slrpi, slrptemi
c      common /prate / rcoef, rexpont, crate2, eexpt
c-----
c      nomenclature:
c      age:      age of the waste in years at time of emplacement in years
c      tend:     end time for simulation
c      d:        vapor diffusion coefficient [m^2/yr]
c      filmthk:  thickness of water film [m] on the wp
c      e:        evaporation rate [m^3/yr]
c      cthick1:  wp outer overpack initial thickness [m]
c      cthick2:  wp inner overpack initial thickness [m]
c      sfactor:  safety factor for uncertainties in mechanical properties
c      grainr:   grain radius [micrometer]
c      nseries:  number of terms in infinite series for wp oxidation
c      gbthick:  grain boundary thickness [micrometer]
c      constant1: a constant relating matrix and grain boundary
c                  diffusivities
c      betaox:   beta kinetics parameter for oxygen reduction
c      betahy:   beta kinetics parameter for water reduction
c      rkox:     rate constant for oxygen reduction [coulomb m/mole/yr]
c      rkhy:     rate constant for water reduction in [coulomb/m^2/yr]
c      gox:      activation energy for oxygen rate constant [J/mole]
c      ghy:      activation energy for water reduction [J/mole]
c      aa(1,3):  cpass = aa(1,1)+aa(1,2)*T+aa(1,3)*T**2
c      aa(2,3):  cpass = aa(2,1)+aa(2,2)*T+aa(2,3)*T**2

```

44/195

```
c      ecrit:    critical potential for localized corrosion [volts SHE]
c      refph:    reference ph
c      taus:    tortuosity/geometry correction for liquid diffusion
c              of oxygen through boiler scale
c      spor:    porosity of scale deposited on the wp (=1.0 for liq. film)
c      dtini:    initial time step [yr]
c      dtmax:    maximum time step [yr]
c      errrel:   relative error
c      errabs:   absolute error
c      ecor:    corrosion potential [volts SHE]
c      cpass:   passive current density [Coulomb/m^2/yr]
c      sollay:  thickness of scale layer [m]
c      curact:  active current density in [coulomb/m^2/yr]
c      timr(i): representative time [yr]
c      tavg(i): average repository temperature [Celsius]
c      tcan(i): waste package surface temperature [Celsius]
c-----
```

```
c--> read basic control parameters
```

```
c      first card is a comment line
c      read (9, *)

c      end time of simulation in years. simulations always start
c      at time = 0 years

c      read (9, *)
c      read (9, *)
c      read (9, *) tend
c      write (22, *) 'tend = ', tend
```

```
c--> container size and geometry
```

```
      read (9, *)
      read (9, *)
      read (9, *) wplen, wpdia
      write (22, *) 'wplen,wpdia', wplen, wpdia
      read (9, *) cthick1, cthick2
      write (22, *) 'cthick1,cthick2 = ', cthick1, cthick2
```

```
c--> thermal section
```

```
      read (9, *)
      read (9, *)
      read (9, *) iflag
c      there are total of 12 sets of tabular, i.e, nset = 12 maximum
      read (9, *) nset
      write (22, *) 'iflag,nset=', iflag, nset
c      time interval for screening input data
      read (9, *) timintv
      write (22, *) 'timintv=', timintv
c      read(9,*) nhista      ! used only when iflag is one
c      write(22,*) 'nhista=', nhista
```

```
c--> age of the waste [y]
```

```
      read (9, *)
      read (9, *)
      read (9, *) age
      write (22, *) 'age [y] = ', age
```


45/195

c--> dry oxidation at the wp overpack outer surface

```
read (9, *)
read (9, *)
read (9, *) grainr
read (9, *) nseries
read (9, *) gbthick
read (9, *) constant1
```

```
c          grain boundary diffusivities
write (22, *) 'grainr,nseries=', grainr, nseries
write (22, *) 'gbthick,constant1=', gbthick, constant1
```

c--> liquid condensation

```
read (9, *)
read (9, *)
read (9, *) humdc
read (9, *) filmthk
read (9, *) ctemp
write (22, *) 'humdc,filmthk,ctemp= ', humdc, filmthk, ctemp
```

c--> corrosion parameters

```
read (9, *)
read (9, *)
write (22, *)
write (22, *) 'corrosion parameters'
write (22, *)
```

c outer overpack

```
read (9, *) xipto
read (9, *) pttemo
read (9, *) slpto
read (9, *) slpttemo
read (9, *) xirpo
read (9, *) rptemo
read (9, *) slrpo
read (9, *) slrptemo

write (22, *) 'xipto,pttemo=', xipto, pttemo
write (22, *) 'slpto,slpttemo=', slpto, slpttemo
write (22, *) 'xirpo,rptemo=', xirpo, rptemo
write (22, *) 'slrpo,slrptemo=', slrpo, slrptemo
```

c inner overpack

```
read (9, *) xipti
read (9, *) pttemi
read (9, *) slpti
read (9, *) slpttemi
read (9, *) xirpi
read (9, *) rptemi
read (9, *) slrpi
read (9, *) slrptemi

write (22, *) 'xipti,pttemi=', xipti, pttemi
write (22, *) 'slpti,slpttemi=', slpti, slpttemi
write (22, *) 'xirpi,rptemi=', xirpi, rptemi
```

46/195

```

write (22, *) 'slrpi,slrptemi=', slrpi, slrptemi

read (9, *) betaox1, betahy1
read (9, *)
read (9, *) betaox2, betahy2
read (9, *)
write (22, *) 'betaox1, betahy1 = ', betaox1, betahy1
write (22, *) 'betaox2, betahy2 = ', betaox2, betahy2
read (9, *) rkox1, rkhy1
read (9, *) gox1, ghy1
read (9, *) rkox2, rkhy2
read (9, *) gox2, ghy2
write (22, *) 'rkox1,rkhy1,gox1,ghy1= ', rkox1, rkhy1, gox1,
&          ghy1
write (22, *) 'rkox2,rkhy2,gox2,ghy2= ', rkox2, rkhy2, gox2,
&          ghy2
read (9, *) aa(1, 1), aa(1, 2), aa(1, 3)
read (9, *) aa(2, 1), aa(2, 2), aa(2, 3)
write (22, *) 'aa11 ... aa13 = ', aa(1, 1), aa(1, 2), aa(1, 3)
write (22, *) 'aa21 ... aa23 = ', aa(2, 1), aa(2, 2), aa(2, 3)
read (9, *) eexpt
read (9, *) rcoef
read (9, *) rexpont
write (22, *) 'eexpt,rcoef,rexpont=', eexpt, rcoef, rexpont
read (9, *) crate2
read (9, *) xcouple
read (9, *) xread
write (22, *) 'crate2,xcouple,xread=', crate2, xcouple, xread
read (9, *) clconca
read (9, *) clcrit1
read (9, *) clcrit2
write (22, *) 'clconca=', clconca
write (22, *) 'clcrit1=', clcrit1
write (22, *) 'clcrit2=', clcrit2
read (9, *) cfactor
write (22, *) 'cfactor=', cfactor
read (9, *) refph
write (22, *) 'refph = ', refph
read (9, *) taus, spor
write (22, *) 'taus, spor = ', taus, spor

```

c--> numerical solver control parameters

```

read (9, *)
read (9, *)
read (9, *) dtini, dtmax
write (22, *) 'dtini, dtmax= ', dtini, dtmax
read (9, *) errrel, errabs
write (22, *) 'errrel, errabs = ', errrel, errabs
write (22, *)
&          '.....'

```

```

return
end

```

**==DRYOXDWP.spg processed by SPAG 4.00Aa at 14:02 on 7 Jan 1997

subroutine dryoxdwp(ipr, tyears, tempc, grainr, nseries,

47/195

```

      &          gbthick, constant1, penetd)
      implicit none
C*** Start of declarations inserted by SPAG
      real*8 constant1, ddd1, ddd2, dfsvtygb, dfsvtym, gbthick,
      &          grainr, penetd, pi, rgas1, sum, temp1, tempc, timehr,
      &          tyears, xxx1
      integer ipr, nnn, nseries
C*** End of declarations inserted by SPAG

```

```

c-----
c      nomenclature:
c      pi:          the constant pi
c      tempc:       temperature in celcius
c      temp1:       tempc in kelvin
c      dfsvtym:     matrix diffusivity
c      dfsvtygb    grain boundary diffusivity
c      constant1:   a constant relating matrix and grain boundary
c                  oxygen diffusivity in metal
c      grainr:      grain radius [micrometer]
c      gbthick:     grain bundary thickness [micrometer]
c      tyears:      time in years
c      thour:       tyears in hours
c      nseries:     number if terms in infinite series for wp oxidation
c                  boundary diffusivities
c
c      Note:
c      Dg:          Dm/constant1
c      rgas1:       universal gas constant
c                  = 1.987 (cal)/(g mol)(K)
c                  = 4.184e-3*1.987 (kJ)/(g mol)(K)
c-----

```

```

c      debudding parameter ipr
c      print*, ipr

c      print*, 'grainr,gbthick,constant1=', grainr, gbthick, constant1

      pi = 4.*atan(1.)
      rgas1 = 1.987

c      constant1 = 1.e-2
c      nseries = 25
c      gbthick = 0.7e-3          ! [micrometer]

      temp1 = 273. + tempc
      xxx1 = 169./(4.184E-3*rgas1*temp1)
c      print*, 'rgas1,temp1,xxx1=', rgas1, temp1, xxx1
      if ( xxx1.gt.65. ) xxx1 = 65.
      dfsvtym = 575.E-6*exp( - xxx1)
      dfsvtym = dfsvtym*1.E12*60*60
      dfsvtygb = dfsvtym/constant1
      ddd1 = 4.*dfsvtym/(grainr*gbthick*dfsvtygb)

      timehr = tyears*365.*24.
      sum = 0.
      do 100 nnn = 1, nseries
          ddd2 = dfsvtym*nnn**2*pi**2*timehr/grainr**2
          if ( ddd2.gt.65. ) ddd2 = 65.
          sum = sum + exp( - ddd2)
c      print*, 'ddd1,ddd2,exp(-ddd2)=', ddd1, ddd2, exp(-ddd2)
100 continue

```

4/3/95

```
penetd = 1./sqrt(ddd1*sum)
```

```
c convert from micrometer to meter:
```

```
penetd = penetd/1.E6
```

```
c print*,'@dryoxdwp:- ipr,tyears,tempc,diffvty,penetd'
```

```
c * ,ipr,tyears,tempc,dfsvtygb,penetd
```

```
return
```

```
end
```

57/1195

**==RELEASE.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

program release

implicit none

cssss IMPLICIT NONE

```

c-----
c Code:      EBSPAC (Engineering Barrier System Performance Assessment Code)
c
c Part:      ebspac_release.f. This is one of the two parts of EBSPAC
c
c Version:   1.0
c
c Date:      January 6, 1997
c
c Purpose:   The EBSPAC computer program has been developed jointly by the
c            Center for Nuclear Waste Regulatory Analyses (CNWRA) and the
c            U. S. Nuclear Regulatory Commission (NRC) for use in the
c            review of DOE's licence application for the HLW Geologic
c            Repository by the Office of NMSS, DWM.
c
c            ebspac_fail.f part of EBSPAC calculates waste package failure
c            time. ebspac_release.f part of EBSPAC calculates the rate of
c            release of nuclides from the repository in the near field.
c
c            The repository is assumed to be made up of cells and EBSPAC
c            provides results for one cell. Input to EBSPAC is read from
c            files which may either be created manually by the user or
c            generated by use of codes other than EBSPAC.
c
c Developers: ebspac_fail.f was developed by Sitakanta Mohanty
c            at the CNWRA with assistance from Tae Ahn (NRC, 301-415-5812),
c            G. Cragolino, P. Lichtner, N. Sridhar, and R. Janetzke (CNWRA)
c
c User:      Developed for use in NRC Iterative Performance Assessment
c            III. This computer code is managed under CNWRA's CODE
c            CONFIGURATION PROCEDURE. Any modifications to the source
c            code must be reported to code custodian (see below).
c            OTHER versions of this code will be released in the future.
c
c Disclaimer: This computer code has not been formally or informally
c            verified, is known to have numerous bugs, and the model
c            embodied in it are not validated.
c
c Contact:   Sitakanta Mohanty (210)522-5185
c            Center for Nuclear Waste Regulatory Analyses
c            Southwest Research Institute, San Antonio, Tx 78250
c-----

```

```

C*** Start of declarations inserted by SPAG
      real amass, amass0, amassc, amassl, c14inv, ccfr, cco3, cftime,
&      cfuel, cgap, ci1000, con3, con4, cstore, ctemp
      real relrate, cumrel, cumrfuel, cumrzirc, cvol, czmetal,
&      czoide, defrac, dr, dt, wetfrac, floref, flow, fueden
      real fuedif, fuewt, funnel, gscmrl, oxgnovpr, phvalue, qin, r0,
&      rdrift, r0z, radu, radsg, dcoefm, flowfactr, rhouz, rpor,
&      rr
      real sc14, sftime, simtimex, sumc14, crf, drf, sumrel, sumre2,
&      sumre3, rlratel, rlrates, rlrates, tcan, tavg, tcaqu,
&      tcool, temp1, temp2, tfail, tflo, tftc, thclad, thetim
      real amwp1, amwp2, amwp3
      real time1, tleach, tlt, totalc14, trid, ttemp, vmax, watrel,

```

pcl5

JOB 1740

release4.f

For: scratchy1!mohanty
Date: Mon Jan 6 14:35:55 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 07:08:03 1991
Started: Wed Nov 13 07:08:03 1991

**==RELEASE.spg processed by SPAG 4.00A 19:34 on 9 Dec 1996

59/195

program release
implicit none

SOFTWARE Analysis Tool Run by Software Developer

BSM
1/8/97

csss IMPLICIT NONE

```

c-----
c Code:      EBSPAC (Engineering Barrier System Performance Assessment Code)
c
c Part:      ebspac_release.f. This is one of the two parts of EBSPAC
c
c Version:   1.0
c
c Date:      January 6, 1997
c
c Purpose:   The EBSPAC computer program has been developed jointly by the
c            Center for Nuclear Waste Regulatory Analyses (CNWRA) and the
c            U. S. Nuclear Regulatory Commission (NRC) for use in the
c            review of DOE's licence application for the HLW Geologic
c            Repository.
c
c            ebspac_fail.f part of EBSPAC calculates waste package failure
c            time. ebspac_release.f part of EBSPAC calculates the rate of
c            release of nuclides from the repository in the near field.
c
c            The repository is assumed to be made up of cells and EBSPAC
c            provides results for one cell. Input to EBSPAC is read from
c            files which may either be created manually by the user or
c            generated by use of codes other than EBSPAC.
c
c Developers: ebspac_fail.f was developed by Sitakanta Mohanty
c            at the CNWRA with assistance from Tae Ahn (NRC),
c            G. Cragnolino, P. Lichtner, N. Sridhar, and R. Janetzke (CNWRA)
c
c User:      Developed for use in NRC Iterative Performance Assessment
c            III. This computer code is managed under CNWRA's CODE
c            CONFIGURATION PROCEDURE. Any modifications to the source
c            code must be reported to code custodian (see below).
c            OTHER versions of this code will be released in the future.
c
c Disclaimer: This computer code has not been formally or informally
c            verified, is known to have numerous bugs, and the model
c            embodied in it are not validated.
c
c Contact:   Sitakanta Mohanty (210)522-5185
c            Center for Nuclear Waste Regulatory Analyses
c            Southwest Research Institute, San Antonio, Tx 78250
c-----

```

C*** Start of declarations inserted by SPAG

```

real amass, amass0, amassc, amassl, c14inv, ccfr, cco3, cftime,
& cfuel, cgap, ci1000, con3, con4, cstore, ctemp
real relrate, cumrel, cumrfuel, cumrzirc, cvol, czmetal,
& czoxide, defrac, dr, dt, wetfrac, floref, flow, fueden
real fuedif, fuewt, funnel, gscmrl, oxgnovpr, phvalue, qin, r0,
& rdrift, r0z, radu, radsg, dcoefm, flowfactr, rhouz, rpor,
& rr
real sc14, sftime, simtimex, sumc14, crf, drf, sumre1, sumre2,
& sumre3, rlr1, rlr2, rlr3, tcan, tagv, tcaqu,
& tcool, temp1, temp2, tfail, tflo, tftc, thclad, thetim
real amwp1, amwp2, amwp3
real time1, tleach, tlt, totalc14, trid, ttemp, vmax, watrel,

```

60/195

```

&      wleach, ws, xco, xcon, dryfr, ktsize
      integer nbt
      integer i, ichns, icoun, ndefco, iiso, imax, imodel, iscon,
&      itim, itime, itype, ize, jmax, k, maxbin, maxchn
      integer maxele, maxi, maxj, maxmem, maxnuc, maxnxm, maxste,
&      maxtim, mtot, icfcon, nelem, noxz, nr, ntemp,
&      numiso, numwp
      integer nzones
      real xlarge, xxx, ppor
      real vsmall, sum
      real dtinit, dtmax, dtmin, eps, tiny
      real xnoloss, xnolosst, xmass, rlmass, amwp
      integer errnum
      integer ncon, kcon, icon, nchns, ni
      real rmaxc14
      real tmaxc14
      real sol, rd, rde, amall, halflife, curall, fggap
      integer itemp
      real xlint, ylint, zlint
      real wplen, wpdia, driftdia
      real c14rate
      real pi, xvola, dintl, xlintl, xfrac, rintl, waterht
      real fuepor
      real dtt1
      real ntype1, ntype2, ntype3

```

```

cssss      REAL*8 alam, atoms, atall, am
      real alam, atoms, atall, am
      real vabs, voidfrac
      integer nremain
      real oxwidth
      real sum1, iflaguo24
      real diff
      real dist, marker
      real xtemporary, gwtotalc14
      integer kc14, ic14
      real pleft

      logical lxaust
      character*6 tmpnam
      character*8 timeyr
      character*80 errmsg
      character*3 elem
      character*6 namall
      character*24 fdate

```

C*** End of declarations inserted by SPAG

C-----

Ccccc

```

parameter (maxchn=50)
parameter (maxele=50)
parameter (maxmem=10)
parameter (maxnuc=50)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxste=300)
parameter (maxtim=500)
parameter (maxbin=500)
parameter (maxi=20, maxj=30)

real fracre(maxchn, maxmem, maxste)
real cumrelease(maxchn, maxmem, maxste)
real maxrel(maxchn, maxmem)

```


61/195

```
real timrel(maxchn, maxmem)
real tregular(maxbin)
```

```
common /ccont2/ defrac
common /ccont5/ sftime, iscon
common /ccoro1/ cftime, tcaqu, icfcon
common /cgeom / cvol, xcon, funnel, mtot
common /cnucl1/ elem(maxele), namall(maxnuc, maxmem)
common /cnucl2/ ncon(maxele), sol(maxele), rde(maxele),
&      amall(maxnuc, maxmem), halflife(maxnuc, maxmem),
&      curall(maxnuc, maxmem), fggap(maxnuc, maxmem),
&      kcon(maxele, maxnuc), icon(maxele, maxnuc),
&      rd(maxnuc, maxmem), ni(maxnuc), nchns,
&      ci1000(maxnuc, maxmem)
common /cnucl3/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
&      atoms(maxnuc, maxmem), am(maxnuc, maxmem),
&      c CFR(maxnuc, maxmem), tleach, nelem,
&      amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
&      amassl(maxnuc), tlt, tftc
common /difrel1/con3, con4, cstore(maxnxm, maxi), trid(maxi, 4),
&      rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
&      ppor(maxi)
common /array1/ sumre1(maxnuc, maxmem, maxste),
&      sumre2(maxnuc, maxmem, maxste),
&      sumre3(maxnuc, maxmem, maxste), time1(maxste),
&      rlratel(maxnuc, maxmem, maxste),
&      rlratel2(maxnuc, maxmem, maxste),
&      rlratel3(maxnuc, maxmem, maxste),
&      amwp1(maxnuc, maxmem, maxste),
&      amwp2(maxnuc, maxmem, maxste),
&      amwp3(maxnuc, maxmem, maxste)
common /cnume1/ simtimex, imax, jmax
common /cnume2/ xco(maxi), nzones, ize(maxi, maxj)
common /cnum3 / diff(maxi)
common /crhyd1/ rpor(maxi), flowfactr, flow, tflo(maxtim),
&      floref(maxtim)
common /cther0/ ntemp
common /cther1/ ttemp(maxtim), tcan(maxtim)
common /cther2/ ctemp
common /cwast1/ fuewt, fueden, fuedif, fuepor, amassc
common /c14 / sc14(maxbin), sumc14(maxbin), gscmrl(maxbin),
&      lxaust
common /icarbon/ noxz
common /rcarbon1/ rOz, rhouz, radu, radsg
common /rcarbon2/ cfuel, czmetal, czoxide, cgap
common /outpt / nbt
common /rkutta/ eps, tiny, dtinit, dtmin, dtmax

common /cumu / crf(maxnuc, maxmem), drf(maxnuc, maxmem)
common /wpvol / wplen, wpdia, driftdia, xvol, dintl, xlintl
common /leach / imodel, phvalue, oxgnovpr, cco3, xfrac
common /leach1/ wetfrac
common /oxidized/ oxwidth
common /uo24width/ sum1, iflaguo24
common /driftwall/ marker
```

```
dimension xlint(2), ylint(2), zlint(2)
dimension c14rate(maxbin)
dimension xno loss(maxnuc, maxmem), xmass(maxnuc, maxmem, maxste)
&      , rlmass(maxnuc, maxmem, maxste),
```

c nomenclature:

- c maxi: max number of x- or r-dir nodes
- c maxj: max number of y- or theta dir nodes
- c defrac: fraction of initially failed packages
- c idefco: integer value of the same thing
- c sftime: time at which scenario causes container failures in a cell
- c tcaqu: time at which aqueous condition begins in the cell
- c cftime: corrosion failure time [y]
- c icfcon: number failed containers in the cell due to corrosion
- c age: age of pit
- c cvol: cell volume
- c xcon: num of packages in a cell
- c wplen: length of package
- c wpdia: diameter of waste package
- c rspher: inner radius of spherical source for diffusion out of wp
- c funnel: area above waste package where water is assumed to funnel into waste package.
- c mtot: total number of containers in the cell
- c drf: rate of release, gram moles per year
- c rd: retardation factor for each isotope
- c dcoefm: molecular diffusion coefficients for pack and rock [m²/yr]
- c tfail: time of wp failure, years
- c tcool: time at which wp drops below boiling pt, years
- c tleach: time span for all u to be converted, years
- c qin: inflow of water to canister, [l/y]
- c wetfrac: position (height) of the outlet as a fraction of the wp id
- c dryfrac: fraction of the fuel available for dry oxidation
- c vmax: maximum water volume in wp before overflow [l]
- c dt: timestep [yr]
- c amass0: mass of fuel [kg]; amassc assigned to amass0
- c amass(k, i): solid mass of each radionuclide in fuel matrix [kg]
- c namall: radionuclide name
- c elem: element name
- c v: volume of water in canister, liters
- c vmin: initial v
- c atall: inventory of radionuclides [atoms];
obtained by converting curall(k,i) in ci
- c sumrel: release history:1=defective,2=scenario,3=normal
failure in the cell, the results are overwritten in
sumrel(maxnuc,maxmem,time step)
- c rlrates: release rate history:1=defective,2=scenario,3=normal
failure in the cell, the results are overwritten in
rlrates(maxnuc,maxmem,time step)
- c time1,2,3: times corresponding to releases for each type of
container failure
- c nr: number of zones for diffusion cal. (same as imax)
- c dr: delta x
- c ci1000(chain,member): 1,000 years inventory in curies
- c simtimex: maximum simulation time [y]
- c imax: nodes in x (or r) direction
- c jmax: nodes in y (or theta) direction
- c xco: x (or r) coordinates
- c yco: y (or theta) coordinates
- c nzones: number of material zones
- c izeone(i,j): zone number of node (i,j)
- c ppor: porosity at nodes outside the wp

```

C      ntflo:          num of entries in flow table
C      rpor:          porosity:: three layers damaged, disturbed, host rock
C      flow:          cell wise flow loading
C      tflo:          time of table entry i for floref
C      floref:        reference flow at tflo
C      ntemp:         num of entries in time versus temp table
C      ttemp(time):   times in table at which temperatures are given
C      tcan(time):    wp surface temp at above times
C      ctemp          crit. temp.[C] for the onset of liquid flow around wp
C      dcoefm:        molecular diffusion coefficient in rock and packing
C      fuewt:         formula wt for spent fuel
C      fuelen:        fuel density
C      fueldif:       fuel diffusion coefficient
C      amassc:        initial mass of uranium oxide fuel, kg
C      sc14:          c14 release rate [ci/kg SF]
C      sumc14:        cumulative c-14 release from the cell as a function of time [ci]
C      rhouz:         density of uo2 kg mole/m^3
C      oxz:           number of moles of uo2 oxidized/mole o (not o2)
C      r0z:           radius of SF particle [m]
C      radu:          initial radius of uo2 grain [m]
C      radsg:         subgrain radius after transgranular fracture [m]
C      thclad:        thickness of cladding, m
C      cfuel:         ci c14 / kg of fuel in fuel
C      czmetal:       ci c14 / kg of fuel in zirconium cladding
C      czoxide:       ci c14/kg of fuel in initial zirconium oxide and crud
C      cgap:          ci of c14 / kg of fuel in grain boundary and gap
C      gscarl:        gaseous cumulative release
C      lxaust:        flag indicating c14 inventory is exhausted.
C      iscon:         scenario container numbers by cell
C      nremain:       remaining waste packages after initially defective ones
C      idefco:        defective container numbers
C      sftime:        scenario container failure time
C      cftime:        corrosion failure time [y]
C      maxchn:        max number of decay chains
C      maxele:        max number of elements (radionuclides)
C      maxnuc:        max number of isotopes
C      maxmem:        max number of enteries in table: age, prob of failure
C      maxste:        max number of time steps
C      maxtim:        max (time-data set) values
C      maxbin:        max number of c14 bins

```

c Notes:

```

C      -> defective container time is zero

C      -> formula for convertng from curies -> atoms is as follows:
C          activity(bq) == lamda x N,
C          where, lamda == log(2.0)/halflife (sec);
C          1 ci == 3.7e10 bq
C          3.7e10*365*24*3600/alog(2.0) == 1.6834e18
C          xnoloss(k,i) == atoms(k,i)/(halflife(k,i)*1.6834E18)

C      -> waste dissolution and transport three types:initial defective
C          containers (1), failures by scenario(2), and corrosion/buckling
C          failures(3). liqrel called 3 times

```

CCCCCC

C Start here.

CCCCCC

C+++++++ open output files ++++++

64/195

```

C files:
c   input:  release4.inp
C   output: inv1000.out, relcum.out, relfrac.out, maxrel.dat,
C           ebs14.dat, treleasel.out, ebsnef.dat, maxc14.dat
C   output file numbers;
C           nunit(1) = unit 5           nunit(2) = unit 6
C           nunit(3) = unit 10          nunit(4) = unit 11
C           nunit(5) = unit 12          nunit(6) = unit 13
C           nunit(7) = unit 14          nunit(8) = unit 16
c           nunit(9) = unit 17

```

```

c   cumc14.out:   cum. release of c14 with time
c   ratec14.out:  time history of c14 release with time
c   release.out:  cum. liquid rel. for all r.n. w/ time
c   maxrel.dat:   maximum fractional release
c   inv1000.out:  no loss inventory of r.n. at 1000 yr
c   treleasel.out: same as ebsnef.dat, but has cumulative releases
c                 in stead of release rates
c   relcum.out:   cumulative release and cumulative release fraction
c   relfrac.out:  normalized release for ebs release standards

```

```

c top18 requirements
write(*,*) 'EBSPAC (Engineering Barrier System Performance Assessment Code)'
write(*,*) 'This is the release part of the EBSPAC code'
write(*,*) fdate()

```

```

call opnfil(20, 'ebsnef.dat', 'replace', .true., errnum, errmsg)
if ( errnum.ne.0 ) stop ' opening ebsnef.dat '
open (1, file='echo_release.out', status='unknown')
open (21, file='cumc14.out', status='unknown')
open (22, file='ratec14.out', status='unknown')
open (23, file='inv1000.out', status='unknown')
open (25, file='relcum.out', status='unknown')
open (24, file='relfrac.out', status='unknown')
open (26, file='treleasel.out', status='unknown')
open (28, file='release.out', status='unknown')
open (29, file='dignostic.out', status='unknown')
open (40, file='maxrel.dat', status='unknown')

```

C+++++++ initialize and prepare input ++++++

```

vsmall = 1.E-20
pi = 4.*atan(1.)
timeyr = 'time(yr)'

```

```

call init
call input

```

c assign diffusion coefficients to grid nodes from zone data

```

do 100 i = 1, nr
  dcoefm(i) = diff(izone(i,1))
  ppor(i) = rpor(izone(i,1))
csss  print*, 'dcoefm(i,') = ', dcoefm(i)
100  continue

```

c calculate volume of maximum sustainable liquid in the wp
c

65/195

```

do 200 icoun = 1, maxste
  do 150 ichns = 1, nchns
    do 120 iiso = 1, ni(ichns)
      sumre1(ichns, iiso, icoun) = 0.
      sumre2(ichns, iiso, icoun) = 0.
      sumre3(ichns, iiso, icoun) = 0.
      rlratel(ichns, iiso, icoun) = 0.
      rlratel2(ichns, iiso, icoun) = 0.
      rlratel3(ichns, iiso, icoun) = 0.
      amwp1(ichns, iiso, icoun) = 0.
      amwp2(ichns, iiso, icoun) = 0.
      amwp3(ichns, iiso, icoun) = 0.
120      continue
150      continue
200      continue

```

C+++++ radionuclide input data ++++++

```

write (1, *)
write (1, *) 'data echo: initial inventory of all nuclides'
write (1, *)

do 300 k = 1, nchns
  do 250 i = 1, ni(k)
    write (1, *) k, i, namall(k, i), curall(k, i), '[ci/wp]'
250      continue
    write (1, *)
300      continue

```

C time for onset of condensation

```

do 400 i = 2, ntemp
  temp1 = tcan(i - 1)
  temp2 = tcan(i)
csss      IF ( temp2.LT.temp1 .AND. temp1.LE.ctemp) GO TO 500
          if ( temp2.le.temp1 .and. temp1.le.ctemp ) go to 500
400      continue

500      continue
          tcaqu = ttemp(i - 1)

write (1, *) 'time to reach below boiling point [yr]=', tcaqu

```

C+++++ container failure of each type ++++++

```

idefco = int(mtot*defrac + 0.005)
write (1, *) 'mtot,idefco:', mtot, idefco

nremain = mtot - idefco

if ( iscon.gt.nremain ) then
  print *, '!!!Warning:-bad numbers for scenario failure'
  print *, 'make sure tpa code has right logic'
  print *, 'program forced to stop'
  stop
end if

if ( cftime.lt.sftime ) then
  icfcon = nremain

```

6/1/95

```

iscon = 0
else
icfcon = nremain - iscon
end if

```

C--> loop over failure types: 1-defective, 2-scenario, 3-corrosion
C modify this later so that in the future, all runs can be done
C in one lapse.

```

ntype1 = 0
ntype2 = 0
ntype3 = 0

```

```

do 600 itime = 1, ntemp
sumc14(itime) = 0.
c14rate(itime) = 0.

```

600 continue

```

do 800 itype = 1, 3

```

```

if ( itype.eq.1 ) then
tfail = 0.
numwp = ndefco
end if

```

```

if ( itype.eq.2 ) then
tfail = sftime
numwp = iscon
end if

```

```

cssss
if ( itype.eq.3 ) then
tfail=min(simtimex, cftime)
tfail = cftime
numwp = icfcon
end if

```

```

write (*, *)
write (*, *)
write (*, 9007)
write (*, '(a42,i10)') 'failure type:', itype
write (*, '(a42,i10)') 'number of failed wp:', numwp
write (*, '(a42,f12.0)') 'wp failure time [yr]:', tfail

```

C+++++++ inside the waste package ++++++

```

amass0 = amassc
tcool = tcaqu
write (*, '(a42,f12.0)') 'onset of flow [yr]:', tcaqu
if (itype.eq.2.and.numwp.eq.0) then
write(*,*)
write(*,*) 'no release due to scenario failure ',
& 'because it occurred after corrosion failure'
write(*,*)
endif

```

```

waterht = wetfrac*dintl
rintl = dintl/2.
voidfrac = xvold/(pi*rintl**2*xlintl)
if ( voidfrac.gt.1. ) then
print *, 'voidfrac=', voidfrac
print *, '!!!!!!program forcibly stopped for wrong data!'
stop

```

67/175

```

end if
xxx = (rintl - waterht)/rintl
vabs = rintl**2*xlintl*acos(xxx)
xfrac = vabs/(pi*rintl**2*xlintl)
vmax = vabs*voidfrac
write (1, *) 'rintl,waterht=', rintl, waterht
write (1, *) 'xlintl,xxx=', xlintl, xxx
write (1, *) 'xfrac,vmax=', xfrac, vmax

do 650 itime = 1, ntemp
  sc14(itime) = 0.
650  continue
  if ( numwp.gt.0 ) then

C+++++ calculate gas releases ++++++

      sum1 = 0.0
      iflaguo24 = 0
      call gasrel(rhouz, r0z, radu, radsg, tfail, thclad, tcan,
&          ttemp, cfuel, czmetal, czoxide, cgap, sc14,
&          cumrfuel, cumrzirc, maxtim, oxwidth)

C--> multiply results (amassc[kg/wp] x numwp) prior to output

      do 660 itime = 2, ntemp

c12/13/96 brought the next if statement from outside to inside
c          of the do loop
C--> mass balance check for gas releases

      if ( ttemp(itime).gt.tcaqu ) then
        dryfrac = 1. - xfrac
      else
        dryfrac = 1.
      end if

      write(1,*), 'ttemp,sc14,dryfrac=',ttemp(itime),
&          sc14(itime),dryfrac
c12/13/96 modified by the multiplying factor dry frac
      c14rate(itime) = c14rate(itime) +
&          sc14(itime)*numwp*amassc*dryfrac
660  continue
cdbg    print*, 'numwp,amassc',numwp,amassc

      totalc14 = (cfuel + czmetal + czoxide + cgap)
&          *(idefco + iscon + icfcon)*amassc
      write (1, *)
      write (1, *) 'cfuel, czmetal, czoxide, cgap', cfuel,
&          czmetal, czoxide, cgap
      write (1, *) 'XXX initial c14 inventory [ci]=', totalc14
      write (1, *) 'normalized cumulative gaseous releases'
      write (1, *) 'cumrfuel,cumrzirc = ', cumrfuel, cumrzirc

c          gaseous c14 goes only to gas module and not neftran

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

c          spherical-equivalent radius of the wp and drift of wp length

      r0 = 0.5*(0.5*wpdia**2 + wpdia*wplen)**0.5

```

68/155

```

rdrift = 0.5*(0.5*driftdia**2 + driftdia*wplen)**0.5
dist = r0
do 680 i = 1, nr
  dist = dist + dr(i)
  if ( dist.gt.rdrift ) then
    dist = dist - dr(i)/2.
    marker = i
    go to 700
  end if
680 continue
700 continue
qin = flow*flowfactr*funnel

write (1, *)
write (1, *) 'tfail,tcool, numwp = ', tfail, tcool, numwp
write (1, *) 'qin,xfrac, vmax = ', qin, xfrac, vmax
write (1, *) 'amass0 = ', amass0
write (1, *) 'liquid phase release'
write (1, *) 'release (ci), release/initial inventory '
write (1, *)

call liqrel(xfrac, xv0l, dt, tcool, r0, dist, amass0, vmax,
&          tfail, qin, itype, simtimex, tavg)
end if
if ( itype.eq.1 ) ntype1 = numwp
if ( itype.eq.2 ) ntype2 = numwp
if ( itype.eq.3 ) ntype3 = numwp

```

800 continue

+++++approximate mass balance check-up+++++

```

c calculate inventory at simtimex

write (1, *) 'no-loss inventory at [y]=' , simtimex

do itemp = 1, ntemp
  do 900 k = 1, nchns
    do 850 i = 1, ni(k)
      curies to atoms
      atall(k, i) = curall(k, i)*(halflife(k,i)*1.6834E18)
      alam(k, i) = alog(2.0)/halflife(k, i)
850 continue
900 continue

call decay(ttemp(itemp), atall, atoms, alam)

c total no-loss cell inventory at max. time
do 1000 k = 1, nchns
  do 950 i = 1, ni(k)
c--> convert atoms to curies
  xnoloss(k, i) = atoms(k, i)/(halflife(k,i)*1.6834E18)
  xnoloss(k, i)=xnoloss(k, i)*(ntype1+ntype2+ntype3)*xfrac
  if(k.eq.5.and.i.eq.4) xnolosst(itemp)=xnoloss(k,i)
950 continue
1000 continue
enddo

c--> cumulative mass released

```



```

do 1100 itemp = 2, ntemp

  do 1050 k = 1, nchns
    do 1020 i = 1, ni(k)
      xmass(k, i, itemp) = sumre1(k, i, itemp)
      & *ntype1 + sumre2(k, i, itemp)
      & *ntype2 + sumre3(k, i, itemp)*ntype3
      rlmass(k, i, itemp) = rlrates1(k, i, itemp)
      & *ntype1 + rlrates2(k, i, itemp)
      & *ntype2 + rlrates3(k, i, itemp)*ntype3
      amwp(k, i, itemp) = amwp1(k, i, itemp)
      & *ntype1 + amwp2(k, i, itemp)*ntype2 + amwp3(k, i, itemp)
      & *ntype3

```

```

c-->      convert from kg/cell to ci/cell

      xmass(k, i, itemp) = xmass(k, i, itemp)
      & *3.576095996E8/half-life(k, i)/amall(k, i)
      rlmass(k, i, itemp) = rlmass(k, i, itemp)
      & *3.576095996E8/half-life(k, i)/amall(k, i)
      amwp(k, i, itemp) = amwp(k, i, itemp)
      & *3.576095996E8/half-life(k, i)/amall(k, i)
      if(k.eq.5 .and. i.eq.4) write(29,9009) ttemp(itemp),
      & xnosst(itemp),amwp(k,i,itemp),xmass(k,i,itemp)

```

```

1020      continue
1050      continue

```

```

1100      continue

```

```

write (25, 9001)
write (*, *)
write (*, 9008)
write (*, 9001) simtimex
write (*, 9007)
write (*, 9006)
write (*, 9008)

```

```

do 1200 k = 1, nchns
  do 1150 i = 1, ni(k)
    pleft = (xnoloss(k,i)-xmass(k,i,ntemp))/xnoloss(k,i)
    write (*, 9002) namall(k, i), half-life(k, i), xnoloss(k, i),
    & amwp(k, i, ntemp), xmass(k, i, ntemp),pleft
    write (25, *) namall(k, i), xmass(k, i, ntemp),pleft

```

```

1150      continue

```

```

1200      continue

```

```

write (*, 9008)

```

```

write (26, *) 'cumulative release with time [ci/cell]'

```

```

do 1300 k = 1, nchns
  do 1250 i = 1, ni(k)
    write (26, *) namall(k, i)
    do 1220 itemp = 2, ntemp
      write (26, *) ttemp(itemp), xmass(k, i, itemp)

```

```

1220      continue

```

```

1250      continue

```

```

1300      continue

```

```

write (1, *) 'ntype1+ntype2+ntype3=', ntype1 + ntype2 + ntype3

```

c cumulative gas release

70/195

```
sum = 0.0
do 1400 itime = 2, ntemp
  dtt1 = ttemp(itime) - ttemp(itime - 1)
  sum = sum + c14rate(itime)*dtt1
cssss  sum= c14rate(itime)*dtt1
  sumc14(itime) = sum
1400  continue

cccc  stop

C+++++ equal interval outputs of releases ++++++

C  count number of isotopes for the ebsnef.dat file

  numiso = 0
  do 1500 k = 1, nchns
    do 1450 i = 1, ni(k)
      numiso = numiso + 1
1450  continue
1500  continue

  xtsize = simtimex/nbt

  do 1600 k = 1, nchns
    do 1550 i = 1, ni(k)
      cumrel = 0.
      relrate = 0.

      do 1540 itim = 1, nbt
        watrel = 0.0
        thetim = itim*xtsize
        tregular(itim) = thetim
        do 1510 itemp = 2, ntemp
          if ( ttemp(itemp).ge.thetim ) then
            xlint(1) = ttemp(itemp - 1)
            xlint(2) = ttemp(itemp)
            ylint(1) = rlmass(k, i, itemp - 1)
            ylint(2) = rlmass(k, i, itemp)
            call lint(xlint, ylint, 2, 2, thetim, relrate)
            zlint(1) = xmass(k, i, itemp - 1)
            zlint(2) = xmass(k, i, itemp)
            call lint(xlint, zlint, 2, 2, thetim, cumrel)
            go to 1520
          end if
1510  continue
1520  continue
        fracre(k, i, itim) = relrate
        cumrelease(k, i, itim) = cumrel

1540  continue
1550  continue
1600  continue

c-->  write cumulative release output for graphics plots

write (28, 9004) timeyr, ((namall(k,i),i=1,ni(k)), k=1, nchns)
do 1700 itim = 1, nbt
  write (28, 9005) tregular(itim),
&      ((cumrelease(k,i,itim),i=1,ni(k)), k=1,
```

71/195

```

& nchns)
1700 continue

write (20, '(a)') 'release rates from ebspac'
write (20, '(2i11,2x,a)') numiso, nbt, 'num nucls, ntemp'

do 1800 k = 1, nchns
  do 1750 i = 1, ni(k)
    loop over time steps
      tmpnam = namall(k, i)
1720 continue
      if ( tmpnam(1:1).eq.' ' ) then
        tmpnam = tmpnam(2:)
        go to 1720
      end if

      write (20, '(a6)') tmpnam

      do 1740 itim = 1, nbt
        write (20, 9003) tregular(itim), fracre(k, i, itim)
1740 continue
1750 continue
1800 continue

```

C+++++ maximum fractional release with time ++++++

```

call decay(1000.DO, atall, atoms, alam)
write (23, *) 'k,i,1000 yr inventory, ini. inv./cell'
do 1900 k = 1, nchns
  do 1850 i = 1, ni(k)
    ci1000(k, i) = atoms(k, i)/(halflife(k,i)*1.6834E18)
    ci1000(k, i) = ci1000(k, i)*mtot
    write (23, *) k, i, namall(k, i), ci1000(k, i), curall(k, i)
1850 continue
1900 continue

```

c--> find fractional releases

C--> find maximum fractional release for all nuclides

```

do 2000 k = 1, nchns
  do 1950 i = 1, ni(k)
    maxrel(k, i) = 0.0
    timrel(k, i) = 0.0
    do 1920 itim = 1, nbt
      xlarge = fracre(k, i, itim)/ci1000(k, i)
      if ( xlarge.gt.maxrel(k,i) ) then
        maxrel(k, i) = xlarge
        timrel(k, i) = tregular(itim)
      end if
1920 continue
1950 continue
2000 continue

```

C--> find maximum fractional release of c14

```

kc14 = 0
ic14 = 0
do 2100 k = 1, nchns
  do 2050 i = 1, ni(k)
    if ( namall(k,i).eq.' C14 ' ) then

```

72/195-

```

      kc14 = k
      ic14 = i
cssss      c14inv=ci1000(k,i)
cssss      IF (c14inv.le.vsmall) c14inv = vsmall
      go to 2200
      end if
2050      continue
2100      continue

2200      continue
      xtemporary = atall(kc14, ic14)
      gwtotalc14 = curall(kc14, ic14)*mtot + totalc14
      write(*, '(a50,f12.1)')
&          'zeroth yr inventory for liquid releases:',
&          curall(kc14,ic14)*mtot
      write(*, '(a50,f12.1)')
&          'zeroth yr inventory for gaseous releases:',totalc14
      if ( kc14.ne.0 .and. ic14.ne.0 ) then
          atall(kc14, ic14) = gwtotalc14*halflife(kc14, ic14)*1.6834E18
          call decay(1000.DO, atall, atoms, alam)
          c14inv = atoms(kc14, ic14)/(halflife(kc14,ic14)*1.6834E18)
          if ( c14inv.le.vsmall ) c14inv = vsmall
          write(*, '(a50,f12.1)')
&          '1000-yr c14 inv. for liq. rel [ci/cell]:',
&          ci1000(kc14,ic14)
          write(*, '(a50,f12.1)')
&          '1000-yr c14 inv. for gaseous rel. [ci/cell]:',
&          c14inv-ci1000(kc14,ic14)
      end if
      atall(kc14, ic14) = xtemporary

      rmaxc14 = 0.0
      tmaxc14 = 0.0
      write (21, *) 'time [yr]      cumulative c14 release [ci]'
      write (22, *) 'time [yr]      c14 release rate [ci]'
      do 2300 itime = 1, ntemp
          write (21, *) ttemp(itime), sumc14(itime)
          write (22, *) ttemp(itime), c14rate(itime)
          xlarge = c14rate(itime)/c14inv
          if ( xlarge.gt.rmaxc14 ) then
              rmaxc14 = xlarge
              tmaxc14 = ttemp(itime)
          end if
2300      continue

C-->      write maximum fractional releases to file maxc14.dat

      write (40, *) 'title: maximum fractional release of c14'
      write (40, *) 'tmaxc14 [yr]      rmaxc14 [fraction]'
      write (40, *) tmaxc14, rmaxc14
      write (40, *)
      write (40, *)

      write (40, *) 'maximum fractional release of all nuclides'
      write (40, *) ' name      timrel (year)      maxrel (fraction)'

      do 2400 k = 1, nchns
          do 2350 i = 1, ni(k)
              if ( k.eq.kc14 .and. i.eq.ic14 ) maxrel(k, i) = maxrel(k, i)
&              *ci1000(k, i)/c14inv

```

73/195

```

        write (40, *) namall(k, i), tleach(k, i), maxrel(k, i)
2350    continue
2400    continue

        print *,
& 'output files: release.out, ebsnef.dat, relcum.out, relfrac.out'
& , 'maxrel.dat, ebs14.dat, maxc14.dat, ratec14.out, inv1000.out'

        stop
9001    format (15x, 'cumulative release [ci/cell] by'f10.1, ' years')
9002    format (5x, a8, 4(4x,e10.3),2x,f8.2)

9003    format (1x, f15.7, 1x, e15.7)

9004    format (a8, 50(a6,1x))
9005    format (1P50(e10.3,1x))
9006    format (6x, 'namall', 4x, 'halflife(yr)', 4x, 'xno loss(ci)', 4x,
&          'amwp(ci)', 4x, 'xmass(ci)',2x,'fracleft')

9007    format (80('-'))
9008    format (80('='))
9009    format(4(2x,e10.3))
        end
**==LIQREL.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

```

        subroutine liqrel(xfrac, xvola, dtt, tcool, r0, dist, amass0,
&          vtmp, tfail, qin, itype, tend, tavg)
C-----
C    liquid source-term model
C-----
        implicit none
c8888    IMPLICIT NONE
C*** Start of declarations inserted by SPAG
        real xvola
        real amass, amass0, amass1, ccf, ci1000, con3, con4, cstore,
&          derivs, dcoefm, dr, dt, dtinit, dtmax, dtmin, dtt, eps,
&          xfrac, pi, qin
        real qout, qtmp, r0, crf, drf, rkqc, rr, srate, sumre1, sumre2,
&          sumre3, rrate1, rrate2, rrate3, t, tcan, tavg, tcool,
&          tend, tfail, tftc, time, time1
        real amwp1, amwp2, amwp3
        real tiny, tleach, tlt, tmass0, tmp, trid, tstart, tstop, ttemp,
&          vmax, vmin, volw, vtmp, wleach, ws, ystart
        integer i, ifault, itype, j, k, maxele, maxi, maxmem, maxnuc,
&          maxnxm, maxste, maxtim, nbad, nelem, nok, nr
        integer ntemp, nvar
        integer ncon, kcon, icon, nchns, ni
        real sol, rd, rde, amall, halflife, curall, fggap
        real r0z, rhouz, radu, radsg
c8888    REAL*8 alam, atoms, atall, am
        real alam, atoms, atall, am
        real fuewt, fueden, fuedif, fuepor, amassc
        real sareap, sareat, uo2rate
        real sum
        integer kmax, kount

```

```

integer it
integer iflaga
real dydx
real dxsav, xp, yp
real timintv, addtime
integer idisplay
real oxwidth
real ppor
real dist, marker

```

C*** End of declarations inserted by SPAG

C-----

```

parameter (maxele=50, maxmem=10, maxnuc=50)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxste=300, maxtim=500)
parameter (maxi=20)

```

```

real dissipxx, rdpxx, adflux(maxnuc, maxmem),
& diffusion1(maxnuc, maxmem), diffusion2(maxnuc, maxmem)

```

```

dimension dydx(200)
dimension ystart(200)

```

```

common /cther1/ ttemp(maxtim), tcan(maxtim)
common /cwast1/ fuewt, fueren, fuerdif, fuepor, amassc
common /cther0/ ntemp

```

```

common /rcarbon1/ rOz, rhouz, radu, radsg
common /path / kmax, kount, dxsav, xp(200), yp(100, 200)
common /damvalues/ rdpxx, dissipxx, adflux, diffusion1,

```

```

& diffusion2
common /rkutta/ eps, tiny, dtinit, dtmin, dtmax
common /cnuc12/ ncon(maxele), sol(maxele), rde(maxele),

```

```

& amall(maxnuc, maxmem), halflife(maxnuc, maxmem),
& curall(maxnuc, maxmem), fggap(maxnuc, maxmem),
& kcon(maxele, maxnuc), icon(maxele, maxnuc),
& rd(maxnuc, maxmem), ni(maxnuc), nchns,
& cil000(maxnuc, maxmem)

```

```

common /cnuc13/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
& atoms(maxnuc, maxmem), am(maxnuc, maxmem),
& c CFR(maxnuc, maxmem), tleach, neleM,
& amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
& amassl(maxnuc), tlt, tftc

```

```

common /difrel1/ con3, con4, cstore(maxnxm, maxi), trid(maxi, 4),
& rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
& ppor(maxi)

```

```

common /array1/ sumre1(maxnuc, maxmem, maxste),
& sumre2(maxnuc, maxmem, maxste),
& sumre3(maxnuc, maxmem, maxste), time1(maxste),
& rlrates1(maxnuc, maxmem, maxste),
& rlrates2(maxnuc, maxmem, maxste),
& rlrates3(maxnuc, maxmem, maxste),
& amwp1(maxnuc, maxmem, maxste),
& amwp2(maxnuc, maxmem, maxste),
& amwp3(maxnuc, maxmem, maxste)

```

```

common /cumu / crf(maxnuc, maxmem), drf(maxnuc, maxmem)
common /oxidized/ oxwidth
common /driftwall/ marker

```

```

external derivs, rkqc

```

C-----

c Nomenclature:

```

C      r0z:          radius of SF particle [m]
C      amass0:       mass of fuel [kg]; amassc assigned to amass0
C      amass(k,i):   solid mass of each radionuclide in fuel matrix [kg]
C      amassl(ielem): mass of element ielem in water
C      ccfr:         conc. at volw (mass/vol)
C      ci1000:       1,000 years inventory of radionuclides in [ci]
C      con3:         coefficient for diffusion calculation
C      cstore:       concentration in cell, gm/l
C      derivs:       name of derivative subroutine for mass balance in wp
C      dcoefm:       mol. diffusion coefficient of packing & rock [m^2/yr]
C      dr:           delta x
C      dt:           timestep, years
C      dtinit:       initial time step used in the R-K algorithm
C      dtmax:        maximum allowed timestep
C      dtmin:        minimum allowed timestep
C      dtt:          time intervals (in the data from external temp module)
C      eps:          integration accuracy for runge kutta integration
C      xfrac:        fraction of fuel exposed
C      pi:           the constant pi
C      qin:          inflow of water to canister, m^3/yr
C      qout:         flow rate out of wp [l/y]
C      qtmp:         maximum of qin and a very small flow rate
C      r0:           effective spherical radius of canister
C      drf:          rate of release, gram moles per year
C      crf:          cumulative release at a given time [yr]
C      rkqc:         name of a subroutine
C      srate:        spent fuel matrix dissolution rate [kg/m2/yr]
C      sumre1:       crf for itype = 1
C      sumre2:       crf for itype = 2
C      sumre3:       crf for itype = 3
C      rlr1:         drf for itype = 1
C      rlr2:         drf for itype = 2
C      rlr3:         drf for itype = 3
C      t:           time [yr]
C      tcan(time):   wp surface temperature [C]
C      tavg(time):   drift wall temperature [C]
C      tcool:        time at which wp drops below boiling point
C      tend:         end time for simulation run
C      tfail:        time of wp failure [yr]
C      tftc:         maximum of wp fail time or boiling point
C      time1,2,3:    times corresponding to releases for each type of
C                   container failure
C      tiny:         a small number needed for odeint subroutine
C      tleach:        time span for all u to be converted, years
C      tlt:          absolute time until which leaching takes place
C                   i.e., tleach+tftc
C      tmass0:       remaining mass after leaching at each time step
C                   (initially same as amass0)
C      trid:         array containing coefficients for the tridiagonal matrix
C                   in the diffusion calculation
C      tstart:       beginning time for odeint subroutine
C      tstop:        end time for odeint subroutine
C      ttemp(time):  times in table at which temperatures are given
C      vmin:         initially volw=vmin
C      vmax:         maximum water volume in wp before it overflows, liters
C      vtmp:         maximum water volume in canister, m^3
C      wleach:       rate of radionuclide release (congruent dissolution)
C      ystart:       array containing all dependent variables for odeint
C      ifailt:       a flag indicating quick release of gap inventory

```

76/195

```

C      itype:          1=defective can run,2=scenario can run,3=normal failure run
C      maxele:        array dimension for maximum number of elements
C      maxi:          max number of x- or r-dir nodes
C      maxmem:        array dimension for maximum number of members in a chain
C      maxnuc:        array dimension for maximum number of nuclides
C      maxste:        max number of time steps
C      maxtim:        max (time-data set) values
C      nbad:          number of bas steps but corrected later in the R-K method
C      nelelem:       number of elements
C      nok:           number of steps accepted in the R-K calculations (see odeint)
C      nr:            number of zones for diffusion cal. (same as imax)
C      volw           volume of water in canister, m^3
C      ntemp:         num of entries in time versus temp table
C      nvar:          an index for number of variables being solved in odeint
C      ncon(ielem):   number of chains in which isotopes of element ielem occur
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)
C      nchns:        number of chains
C      ni
C      sol(ielem):    solubility limit of the element ielem
C      rd:            retardation factor for each isotope
C      rde
C      amall:         molecular weight of radionuclides
C      halflife:      half life of radionuclides [yr]
C      curall         radionuclide inventory per WP in curies
C      fggap:         gap fraction of each radionuclide
C      alam:          coefficient of decay for each radionuclide
C      atoms:        number of atoms of radionuclides at time t
C      atall:         number of atoms in the initial inventory curall curies
C      am:            mass of each radionuclide in the wp water at time t
C      fuewt:        formula wt for spent fuel
C      fueden:        fuel density
C      fuedif:        fuel diffusion coefficient
C      fuepor:        porosity in between SF particles
C      amassc:        initial mass of uranium oxide fuel, kg
C      sareap:        total surface area from one SF particle
C      sareat        total internal surface area in the wetted portion of sf
C      uo2rate:       leaching rate of sf matrix radionuclide [kg/yr]

```

```

C note: solubility is in kg/m^3 = mg/l to be input in radionuclide
C      data file

```

```

C-----

```

```

CCCCC
C      Start here.
CCCCC

```

```

      pi = 4.*atan(1.)
      iflaga = 0

```

```

Cjw make sure time constant for water in can is not too fast

```

```

      qtmp = max(qin, 1.E-6)
      if ( vtmp/qtmp.lt.5. ) then
        vmax = 5.*qtmp
      else
        vmax = vtmp

```

```

cqqqq      print*, 'vtmp/qtmp, and vmax=', vtmp/qtmp, vmax
      end if

```


77/195

```
write (1, *) 'above:vmax,vtmp,qtmp, vmax, vtmp, qtmp
```

C constant in transient diffusion equation

```
con3 = 4*pi*(r0 + dr(1)/2)**2*dcoefm(1)*ppor(1)/dr(1)
con4 = 4*pi*(dist)**2*dcoefm(marker)*ppor(marker)/dr(marker)
```

```
write (1, *) 'ppor(marker),con3,con4=', ppor(marker), con3, con4
```

C set up the tridiagonal matrix for diffusive flux calculation

```
call setdif(r0)
```

C calculate uo2 matrix leachng time

```
tmass0 = amass0*xfrac
tftc = amax1(tfail, tcool)
```

```
write (1, *) 'amass0=', amass0
addtime = 0.0
timintv = 50.
```

```
do 100 i = 2, ntemp
  time = ttemp(i)
  if ( time.gt.tftc ) then
    tavg = tcan(i)
    addtime = time - tftc
    dtt = ttemp(i) - ttemp(i - 1)
    if ( iflaga.eq.0 ) then
      timintv = dtt
      iflaga = 1
    end if
```

```
c print*, 'time,tcool,tftc,timintv,tavg=',
c & time,tcool,tftc,timintv,tavg
```

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

c--> waste matrix surface area and leach rate/unit mass UO2

```
cssss sareap = 4.*pi*((r0z-oxwidth)**2+(3.*r0z**2*oxwidth-3.
cssss & *r0z*oxwidth**2+oxwidth**3)/radu)
```

```
write (1, *) , 'grain radius [m] :', radu
write (1, *) , 'grain oxidation depth [m]:', oxwidth
```

```
sareap = 4.*pi*(radu - oxwidth)
& **2 + (3.*radu**2*oxwidth - 3.*radu*oxwidth**2 +
& oxwidth**3)/radu**3*(4.*pi*radu**2)
```

c--> waste matrix leach rate from total wet mass of UO2/wp

```
cssss sareat = (amass0*xfrac) * saream
sareat = sareap*(amass0*xfrac)/(4./3.*pi*radu**3*fueden)
uo2rate = srate*sareat
write (1, *) 'time,amass0,sareat,srate,uo2rate,tmass0'
write (1, *) time, amass0, sareat, srate, uo2rate, tmass0
```

```
tmass0 = tmass0 - uo2rate*dtt
if ( tmass0.lt.0. ) then
  if ( time.le.tftc + timintv ) then
    addtime = amass0*xfrac/uo2rate
```

78/195-

```

    else
      addtime = addtime - abs(tmass0)/uo2rate
    end if
    go to 200
  end if
  write (1, *) 'time, amass0, xfrac, uo2rate, tmass0'
  write (1, *) time, amass0, xfrac, uo2rate, tmass0
end if
100 continue

200 continue
tleach = addtime
tlt = tleach + tftc
write (*, '(a42.f12.0)') 'sf leaching time span [yr]:', tleach
write (*, '(a42.f12.0)') 'sf leaching onset time [yr]:', tftc
write (*, '(a42.f12.0)') 'sf leaching will end at [yr]:', tlt
write (*, 9001)

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

C-->    convert curies/wp -> atoms/wp

c      activity(bq)=lamda x N;
C      lamda->alog(2.0)/halflife (sec); 1 ci -> 3.7e10 bq
c      the next line is equivalent to: atall(k, i)=curall(k, i)
c      & *3.7E10*(365*24*3600*halflife(k,i))/alog(2.0)

      atall(k, i) = curall(k, i)*(halflife(k,i)*1.6834E18)

C-->    calculate decay coefficient [1/yr]

      alam(k, i) = alog(2.0)/halflife(k, i)

250    continue
300    continue

C      set minimum value of water in wp to a small value

      if ( vtmp.ne.vmax ) then
        tmp = vtmp/qtmp
        vmin = (5. - tmp)*qtmp
      else
        vmin = 1.e-6
      end if

      volw = vmin

      write (1, *) '@liqrel-namall, am241i(kg), it, crf, am(kg), volw'

      ifailt = 0

      do 400 k = 1, nchns
        do 350 i = 1, ni(k)
          am(k, i) = 1.E-20
          crf(k, i) = 1.E-20
350        continue
400      continue

      write (1, *) 'volw=', volw
```

```
sum = 0.0
idisplay = 0
```

```
do 600 it = 2, ntemp
```

```
  t = ttemp(it)
  dt = t - ttemp(it - 1)
  tavg = tcan(it - 1)
```

```
cssss      tavg=tcan(it)
```

```
C      set inventory for quick release at tfail
```

```
  if ( t.gt.tfail .and. t.gt.tftc ) then
```

```
    if ( ifailt.eq.0 ) then
      ifailt = 1
```

```
    call decay(t - dt, atall, atoms, alam)
```

```
C      radionuclides in grain+gap [kg]
```

```
  do 410 k = 1, nchns
    do 405 i = 1, ni(k)
      am(k, i) = am(k, i) + fggap(k, i)*(atoms(k,i)/6.02E23)
      &          *amall(k, i)*0.001
```

```
405      continue
```

```
410      continue
```

```
  ystart(1) = volw
  j = 1
```

```
  do 420 k = 1, nchns
```

```
    do 415 i = 1, ni(k)
```

```
      j = j + 1
```

```
      ystart(j) = am(k, i)
```

```
      write (1, *) 'ystart(', j, ')=', ystart(j)
```

```
415      continue
```

```
420      continue
```

```
  do 430 k = 1, nchns
```

```
    do 425 i = 1, ni(k)
```

```
      j = j + 1
```

```
      ystart(j) = crf(k, i)
```

```
      write (1, *) 'ystart(', j, ')=', ystart(j)
```

```
425      continue
```

```
430      continue
```

```
  nvar = j
```

```
end if
```

```
C ode solver from numerical recipes
```

```
  tstart = t - dt
```

```
  tstop = t
```

```
  kmax = 100
```

```
  dxsav = (tstop - tstart)/20.
```

```
  call odeint(ystart, nvar, tstart, tstop, eps, dtinit, dtmin,
```

```
&          dtmax, nok, nbad, tiny, derivs, rkqc, xfrac,
```

```
&          xvol, tfail, qin, vmax, amass0, r0z, tavg, qout,
```

```
&          dydx, it)
```

cssss

```
if (it.eq.ntemp-31)
```

80/195

```
volw = ystart(1)
```

```
j = 1
```

```
do 460 k = 1, nchns
```

```
do 440 i = 1, ni(k)
```

```
j = j + 1
```

```
am(k, i) = ystart(j)
```

440

```
continue
```

460

```
continue
```

```
do 480 k = 1, nchns
```

```
do 470 i = 1, ni(k)
```

```
j = j + 1
```

```
crf(k, i) = ystart(j)
```

470

```
continue
```

480

```
continue
```

```
do 500 k = 1, nchns
```

```
do 490 i = 1, ni(k)
```

```
j = j + 1
```

```
drf(k, i) = dydx(j)
```

490

```
continue
```

500

```
continue
```

C limit the flux and concentration to positive

```
do 520 k = 1, nchns
```

```
do 510 i = 1, ni(k)
```

```
if ( am(k,i).lt.0.0 ) am(k, i) = 0.0
```

```
if ( crf(k,i).lt.0.0 ) crf(k, i) = 0.0
```

510

```
continue
```

520

```
continue
```

C cumulative release of all nuclides [kg/wp]

```
do 540 k = 1, nchns
```

```
do 530 i = 1, ni(k)
```

```
if ( itype.eq.1 ) then
```

```
amwp1(k, i, it) = am(k, i)
```

```
sumre1(k, i, it) = crf(k, i)
```

```
rlrate1(k, i, it) = drf(k, i)
```

```
end if
```

```
if ( itype.eq.2 ) then
```

```
amwp2(k, i, it) = am(k, i)
```

```
sumre2(k, i, it) = crf(k, i)
```

```
rlrate2(k, i, it) = drf(k, i)
```

```
end if
```

```
if ( itype.eq.3 ) then
```

```
amwp3(k, i, it) = am(k, i)
```

```
sumre3(k, i, it) = crf(k, i)
```

```
rlrate3(k, i, it) = drf(k, i)
```

```
end if
```

```
if ( k.eq.5 .and. i.eq.4 .and. idisplay.eq.0 ) then
```

```
write (*, 9002)
```

```
write (*, 9003)
```

```
write (*, 9005)
```

```
write (*, 9002)
```

81/195

```

        end if
        if ( k.eq.5 .and. i.eq.4 ) write (*, 9004) t, crf(k, i),
&          rdpxx, dissipxx, adflux(k, i), diffusion1(k, i),
&          diffusion2(k, i)
530      continue
540      continue
        idisplay = idisplay + 1
        if ( idisplay.gt.21 ) idisplay = 0

        if ( t.gt.tend ) go to 700

    end if

    time1(it) = t

600  continue

700  continue
    return
9001  format (80('-'))
9002  format (80(' '))

9003  format (4x,'time', 4x,'cwpwater', 5x, 'rgenerate', 5x, 'rdecay',
&          6x, 'rconv', 5x, 'rdiff_r0', 4x, 'rdiff_ebs')
9005  format (5x, '[yr]', 6x, '[kg]', 7x, '[kg/yr]', 5x,
&          '[kg/yr]', 5x, '[kg/yr]', 5x, '[kg/yr]',
&          5x, '[kg/yr]')

9004  format (f10.1, 1P6e12.3)
    end

```

**==DERIVS.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

subroutine derivs(t, y, dydx, xfrac, xvola, tfail, qin, vmax,
&          amass0, tavg, qout, dtd, it)
    implicit none
c$$$    IMPLICIT NONE
    integer it
    real xvola

C*** Start of declarations inserted by SPAG
    real amass, amass0, amassl, amt1, amt2, ccf, ccf, ci1000,
&          con1, con3, con4, concn, cstore, dam, dcoefm, difflax1,
&          difflax2, diffusion1, diffusion2, dr, dt, dtd, dvdt, dydx
    real xfrac, fut, qin, qout, rdp, rf, adflux, rr, srate, sumre1,
&          sumre2, sumre3, rrate1, rrate2, rrate3, t, tavg, tfail,
&          tftc, time1, tleach
    real amwp1, amwp2, amwp3
    real tlt, trid, vfull, vmax, vnow, wleach, ws, wu, y, yyy
    real sareap, sareat, uo2rate
    integer i, ielem, j, jj, k, kountr, l, maxele, maxi, maxmem,
&          maxnuc, maxnxm, maxste, nelem, nr
    integer ncon, kcon, icon, nchns, ni
    real sol, rd, rde, amall, halflife, curall, fggap
    integer mmi, mmj
    real fracn
    real fuewt, fueден, fueдif, fuepor, amassc

```

```

real dissip, sum
real dissipxx, rdpxx
real rdp2, dissip2
real oxwidth
real ppor
integer iflow
real pi
real r0z, rhouz, radu, radsg
cssss  REAL*8 alam, atoms, atall, am
real alam, atoms, atall, am
real xxx1,xxx2,xxx3

```

C*** End of declarations inserted by SPAG

C-----

```

parameter (maxele=50, maxmem=10, maxnuc=50)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxste=300)
parameter (maxi=20)

dimension y(200), dydx(200)
dimension diffusion1(maxnuc, maxmem), diffusion2(maxnuc, maxmem)
dimension amt1(maxnuc, maxmem), dam(maxnuc, maxmem),
&         amt2(maxnuc, maxmem), adflux(maxnuc, maxmem),
&         rf(maxnuc, maxmem)

```

```

c  COMMON /parameters2/ rf
common /cwast1/ fuewt, fueden, fuedif, fuepor, amassc
common /damvalues/ rdpxx, dissipxx, adflux, diffusion1,
&                 diffusion2
common /rcarbon1/ r0z, rhouz, radu, radsg
common /oxidized/ oxwidth

common /cnucl2/ ncon(maxele), sol(maxele), rde(maxele),
&             amall(maxnuc, maxmem), halflife(maxnuc, maxmem),
&             curall(maxnuc, maxmem), fggap(maxnuc, maxmem),
&             kcon(maxele, maxnuc), icon(maxele, maxnuc),
&             rd(maxnuc, maxmem), ni(maxnuc), nchns,
&             ci1000(maxnuc, maxmem)
common /cnucl3/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
&             atoms(maxnuc, maxmem), am(maxnuc, maxmem),
&             ccfr(maxnuc, maxmem), tleach, nelelem,
&             amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
&             amassl(maxnuc), tlt, tftc
common /difrel1/con3, con4, cstore(maxnxm, maxi), trid(maxi, 4),
&             rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
&             ppor(maxi)
common /array1/ sumre1(maxnuc, maxmem, maxste),
&             sumre2(maxnuc, maxmem, maxste),
&             sumre3(maxnuc, maxmem, maxste), time1(maxste),
&             r1rate1(maxnuc, maxmem, maxste),
&             r1rate2(maxnuc, maxmem, maxste),
&             r1rate3(maxnuc, maxmem, maxste),
&             amwp1(maxnuc, maxmem, maxste),
&             amwp2(maxnuc, maxmem, maxste),
&             amwp3(maxnuc, maxmem, maxste)

```

data fut/1.0/

C-----

c Nomenclature:

```

c vwnow: volume of water in the wp at time t (same as y(1))
C vmax: maximum water volume in wp before overflow [l]
C qout: flow rate out of wp [l/y]
C con1: weir coefficient used for wp water overflow model
C qin: inflow of water to canister, [l/y]
c dvdt: time rate of change of water in the wp
c dydx: array containing derivatives for all equn. used in odeint
c nchns: number of chains
c amt1: amount of each r.n. in the wp water at time t
c ni(nchns): elements in the chain
C nelem: element of the chain (ni(ichns) assigned)
c amt2: amount of each r.n. outside the wp at time t(for mass balance)
c tlt: absolute time until which leaching takes place (tleach+tftc)
c wu: a multiplier for keeping leaching zero before time tlt
c t: time [yr]
c atall: inventory of radionuclides [atoms];
c obtained by converting curall(k,i) in ci
c atoms: number of atoms of radionuclides at time t
c alam: coefficient of decay for each radionuclide
c srate: the rate of dissolution of the uo2 matrix
C amass(k, i): solid mass of each radionuclide in fuel matrix [kg]
c fggap: gap fraction of each radionuclide
c amall: molecular weight of radionuclides
c fut: correction factor for present mass of fuel still left
C unconverted - presently, assume that mass or area of
C fuel doesn't change
C fueledn: fuel density
c fuepor: porosity in between SF particles
c sareap: specific internal surface area per unit bulk vol. of sf
c sareat: total internal surface area in the wetted portion of sf
c uo2rate: leaching rate of sf matrix radionuclide [kg/yr]
c wleach: leaching rate of radionuclide congruent w/ sf leaching
C amass0: mass of fuel [kg]; amassc assigned to amass0
c ncon(ielem):
c amassl(ielem): mass of element ielem in water
c concn: conc. given element in the wp water at a specified time
c sol(): solubility limit of an element (input data)
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)
c fracn: elemental fraction of radionuclide in the wp water (mass/mass)
c cefr: radionuclide conc. in the wp water at a specified time
c vfull: volume of water in the wp just before water overflows
C cefrd: conc. scaled to vfull[m3]
c dtd: time interval [yr]
c diffflux: mass rate of r.n. production due to diffusive flux out
c diffusion: diffflux values stored in the array diffusion
c rdp: mass rate of production of r.n. from parent in the wp water
c adflux(k,ii): mass rate of r.n. release due to advective flux out of wp
c dissip: decay loss rate of the radionuclide from the wp water
c dam(k,i): rate of change of mass in wp water
c rdp2: mass rate of production of r.n. from parent outside
c the wp (for mass balance calculation)
c rf(k,i): total outflux from the wp with time
c iflow: a flag indicating whether water from the wp is overflowing

```

```

C-----
CCCCC
C Start here.
CCCCC

```

```

pi = 4.*atan(1.)
C--> use weir coefficient con1 in overflow model to smoothly
C converge volume to steady state

```

```

csss print*, 'xvol=',xvol
      vwnow = y(1)
      if ( vwnow.le.vmax ) then
        qout = 0.0
        iflow = 0
      else
        iflow = 1
        con1 = 0.8
        yyy = (vwnow/vmax)**2
        qout = qin*con1*yyy
      end if

```

```

dvdt = qin - qout
dydx(1) = dvdt

```

```

j = 1
do 100 k = 1, nchns
  do 50 i = 1, ni(k)
    j = j + 1
    amt1(k, i) = y(j)
50 continue
100 continue

```

```

do 200 k = 1, nchns
  do 150 i = 1, ni(k)
    j = j + 1
    amt2(k, i) = y(j)
150 continue
200 continue

```

```

c print*, 'amt1(5,4)=',amt1(5,4)
C--> calculate mass transfer from fuel to water

```

```

C leaching period multiplier

```

```

if ( t.lt.tlt ) then
  wu = 1.0
else
  wu = 0.0
end if

```

```

C rate of release of contained radionuclides

```

```

call decay(t, atall, atoms, alam)

```

```

C congruent UO2 leach rate

```

```

call leachrt(tavg, srate)

```

```

cssss
c print*, 'fueden,fuepor,rOz,amass0=',
c & fueden,fuepor,rOz,amass0

```

```

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

```


85/55

```

C      solid mass of each radionuclide in fuel matrix [kg]

      amass(k, i) = (1. - fggap(k,i))*(atoms(k,i)/6.02E23)
&      *amall(k, i)*0.001

c-->   waste matrix surface area and leach rate/unit mass UO2

cssss  sareap = 4.*pi*((r0z-oxwidth)**2+(3.*r0z**2*oxwidth-3.
cssss  &      *r0z*oxwidth**2+oxwidth**3)/radu)
      sareap = 4.*pi*(radu - oxwidth)
&      **2 + (3.*radu**2*oxwidth - 3.*radu*oxwidth**2 +
&      oxwidth**3)/radu**3*(4.*pi*radu**2)

c-->   waste matrix leach rate from total wet mass UO2/wp

cssss  sareat = (amass0*xfrac)*sareap/(4./3.*pi*r0z**3*fueden)
      sareat = sareap*(amass0*xfrac)/(4./3.*pi*radu**3*fueden)
      uo2rate = srate*sareat

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)/amass0)*uo2rate*wu*fut
c      if(k.eq.5.and.i.eq.4) then
c      print*, 'amass,amass0,wleach',amass(k,i),amass0,wleach(k,i)
c      endif

250    continue
300    continue

C-->   collect all radionuclides by element in wp water

C-->   mass of element ielem in the wp water

do 400 ielem = 1, nelelem
  sum = 0.0
  do 350 l = 1, ncon(ielem)
    mmi = kcon(ielem, l)
    mmj = icon(ielem, l)
    sum = sum + amt1(mmi, mmj)
350   continue
  amassl(ielem) = sum
400   continue

C-->   fraction of nuclide w.r.to its element

do 500 ielem = 1, nelelem
cssss  IF ( vwnow.GT.0. ) THEN
      if ( vwnow.ge.1.E-20 ) then
        concn = amassl(ielem)/vwnow
        if ( concn.gt.sol(ielem) ) concn = sol(ielem)
      else
        concn = 0.0
      end if
500   continue

C-->   mass of radionuclide per unit volume of wp water

```

86/195

```
do 450 l = 1, ncon(ielem)
  mmi = kcon(ielem, l)
  mmj = icon(ielem, l)
csss      IF ( amassl(ielem).GT.0.0 ) THEN
  if ( amassl(ielem).gt.1.E-20 ) then
    fracn = amt1(mmi, mmj)/amassl(ielem)
    ccfr(mmi, mmj) = concn*fracn
c          if (mmi.eq.5.and.mmj.eq.4)
c      &          print*, 'concn,fracn',concn,fracn
c          ccfr(kcon(ielem,l), icon(ielem,l))
c      &          =concn*amt1(kcon(ielem,l), icon(ielem,l))
c      &          /amassl(ielem)
  else
cssss      ccfr(mmi,mmj) = 0.0
            ccfr(mmi, mmj) = 1.E-20
  end if
450      continue

500      continue

C-->      correction for diffusive flux

kountr = 0

do 600 k = 1, nchns
  do 550 i = 1, ni(k)
    kountr = kountr + 1
    vfull = xvoll
    ccfrd = ccfr(k, i)*vwnow/vfull
cssss
c          if(it.eq.165) print*, 'ccfrd (same as c0),vwnow=',
c      &          ccfrd,vwnow

cssss      IF ( ccfrd.GT.0.0 ) THEN
            if ( ccfrd.gt.1E-20 .and. iflow.eq.1 ) then

Csm        see the next constraint again...
            dt = dtd

            call dflux(ccfrd, rd(k,i), alam(k,i), kountr, diffflux1,
&            diffflux2, dt, it)

            else
              diffflux1 = 0.0
              diffflux2 = 0.0
            end if

            diffusion1(k, i) = diffflux1
            diffusion2(k, i) = diffflux2

550      continue
600      continue

jj = 1

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

c          if (k.eq.5.and.i.eq.4) print*, 'dtd=',dtd
c-->      rate of production of am(k,i) from parent
```

87/195

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

c-->      decay loss rate of the radionuclide from the wp water
xxx2 = amt1(k,i)*6.02E23/amall(k,i)/0.001
dissip = xxx2*alam(k, i)
cssss      dissip = amt1(k, i)*alam(k, i)

c-->      radionuclide outflux from the wp: advective
ccc      if (k.eq.5.and.i.eq.4) print*,'ccfr,qout',ccfr(k,i),qout

      adflux(k, i) = ccfr(k, i)*qout

C-->      total outflux from the wp

      rf(k, i) = adflux(k, i) + diffusion1(k, i)
cssss      rf(k,i) = 0.0

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

      xxx3 = (rdp-dissip)
c          from atoms --> kg
      xxx3 = (xxx3/6.02E23)*amall(k,i)*0.001
      dam(k,i) = wleach(k,i) + xxx3 - rf(k,i)

cssss      dam(k, i) = wleach(k, i) + rdp - dissip - rf(k, i)

cssss      if(k.eq.5.and.i.eq.4) write(*,99001) t,wleach(k,i)
c          print*,'it=',it
c          if(it.eq.165) write(*,99001) t,wleach(k,i)
c          &      ,rdp,dissip,adflux(k,i),diffusion(k,i),dam(k,i)
      jj = jj + 1
cccc      if (k.eq.5.and.i.eq.4) print*,'jj=',jj
      dydx(jj) = dam(k, i)
      if ( k.eq.5 .and. i.eq.4 ) dissipxx = dissip
      if ( k.eq.5 .and. i.eq.4 ) rdp2xx = 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)
cssss      rdp2 = amt2(k, i - 1)*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)
cssss      dissip2 = amt2(k, i)*alam(k, i)
    xxx3 = (rdp2-dissip2)
c          from atoms --> kg
```

88/
1195

```
xxx3 = (xxx3/6.02E23)*amall(k)0.001
dydx(jj) = xxx3 + adflux(k, i) + diffusion1(k, i)
c if(k.eq.5.and.i.eq.4) write(*,99003)
c & jj,t,dt,dydx(jj),y(jj)
750 continue
800 continue

do 900 k = 1, nchns
do 850 i = 1, ni(k)
jj = jj + 1
dydx(jj) = adflux(k, i) + diffusion2(k, i)
850 continue
900 continue

return

9001 format ('t,lch,rdp,dsp,adflux,diffn,dam==', 12(e10.4,2x))
9002 format ('t,rdp,dissip,rdp-dissip,adflux,diffusion=', 12E10.2)
9003 format ('from derivs: ', i5, 4x, 12(e10.4,2x))
end

**==DECAY.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996
```

```
subroutine decay(t, xni0, xnij, alam)
implicit none
cssss IMPLICIT NONE
C-----
C chain decay by bateman equation to calculate contribution of the
c ith chain member to the jth chain member where j is a daughter
c member
C-----
C*** Start of declarations inserted by SPAG
real an, ci1000, p1, p2, sum, t, xxx
integer i, j, k, kchn, l, m, maxele, maxi, maxmem, maxnuc,
& maxnxm, maxste, memb
integer ncon, kcon, icon, nchns, ni
real sol, rd, rde, amall, halflife, curall, fggap
cssss REAL*8 xni0,xnij,alam
real xni0, xnij, alam
C*** End of declarations inserted by SPAG
C-----

parameter (maxele=50, maxmem=10, maxnuc=50)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxste=300)
parameter (maxi=20)

dimension an(maxmem)
dimension xni0(maxnuc, maxmem), xnij(maxnuc, maxmem),
& alam(maxnuc, maxmem)
common /cnuc12/ ncon(maxele), sol(maxele), rde(maxele),
& amall(maxnuc, maxmem), halflife(maxnuc, maxmem),
& curall(maxnuc, maxmem), fggap(maxnuc, maxmem),
& kcon(maxele, maxnuc), icon(maxele, maxnuc),
& rd(maxnuc, maxmem), ni(maxnuc), nchns,
& ci1000(maxnuc, maxmem)
```

89
1/95

c Nomenclature:

- C maxele: max number of elements (radionuclides)
- C maxnuc: max number of isotopes
- C maxmem: max number of entries in table: age, prob of failure
- C maxste: max number of time steps
- C maxi: max number of x- or r-dir nodes
- c nchns: number of chains
- c ni(nchns) elements in the chain
- C nelem: element of the chain (ni(ichns) assigned)
- C kchn: index for chain number
- C xnij: nventory of the jth element of kchn chain at time t
- c p1: first product in the Bateman equation
- c p2: second product in the Bateman equation
- c alam: coefficient of decay for each radionuclide
- c t: time [yr]
- c an(i): mass contribution of the ith element to the jth element
- c xni0: initial inventory of the jath elemnt of the kchn chain

CCCCC

C Start here.

CCCCC

do 200 kchn = 1, nchns

memb = ni(kchn)

do 100 j = 1, memb

xnij(kchn, j) = 0.0

do 40 i = 1, j

C first product in bateman equation

p1 = 1.0

if (j.gt.1) then

do 5 k = i, j - 1

p1 = p1*alam(kchn, k)

5 continue

end if

sum = 0.0

do 20 l = i, j

C second product in the bateman equation

p2 = 1.0

do 10 m = i, j

if (m.ne.l) p2 = p2*(alam(kchn,m) - alam(kchn,l))

10 continue

C the sum in bateman equation

Csm next line added

xxx = alam(kchn, l)*t

if (xxx.gt.65.0) xxx = 65.0

sum = sum + exp(- xxx)/p2

20 continue

C the contribution of the ith element to the jth element

an(i) = p1*xni0(kchn, i)*sum

```

40      continue

      do 60 i = 1, j
C          sum the contributions of all previous elements in chain
          xnij(kchn, j) = xnij(kchn, j) + an(i)
60      continue
100     continue
200     continue

      return
      end

```

**==DFLUX.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

      subroutine dflux(c0, rdi, al, kount, difflux1, difflux2, dt, it)
      implicit none

```

```

c-----
csss      IMPLICIT NONE
C-----
C          module for calculating diffusion flux with finite difference scheme
C-----

```

C*** Start of declarations inserted by SPAG

```

      integer it
      real al, amass, amassl, c0, ccsr, con3, con4, cp, cstore,
&        dcoefm, difflux1, difflux2, dr, dt, rdi, rr, sumre1,
&        sumre2, sumre3, rlrates, rlrates2, rlrates3, tftc, time1
      real amwp1, amwp2, amwp3
      real tleach, tlt, trid, wleach, ws
      real marker
      real ppor
      integer i, kount, maxi, maxmem, maxnuc, maxnxm, maxste, nelem,
&        nr, nrs

```

C*** End of declarations inserted by SPAG

```

C-----

```

```

      parameter (maxmem=10, maxnuc=50)
      parameter (maxnxm=maxnuc*maxmem)
      parameter (maxste=300)
      parameter (maxi=20)
      parameter (nrs=maxi)

```

```

csss      REAL*8 alam, atoms, atall, am
      real alam, atoms, atall, am
      dimension cp(nrs)
      common /cnuc13/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
&        atoms(maxnuc, maxmem), am(maxnuc, maxmem),
&        ccsr(maxnuc, maxmem), tleach, nelem,
&        amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
&        amassl(maxnuc), tlt, tftc
      common /difrel1/con3, con4, cstore(maxnxm, maxi), trid(maxi, 4),
&        rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
&        ppor(maxi)
      common /array1/ sumre1(maxnuc, maxmem, maxste),
&        sumre2(maxnuc, maxmem, maxste),
&        sumre3(maxnuc, maxmem, maxste), time1(maxste),
&        rlrates(maxnuc, maxmem, maxste),
&        rlrates2(maxnuc, maxmem, maxste),
&        rlrates3(maxnuc, maxmem, maxste),
&        amwp1(maxnuc, maxmem, maxste),

```

9/1/95

```

&          amwp2(maxnuc, maxmem, maxste),
&          amwp3(maxnuc, maxmem, maxste)
common /driftwall/ marker

```

c Nomenclature:

```

C maxele:      max number of elements (radionuclides)
C maxnuc:      max number of isotopes
C maxmem:      max number of entries in table: age, prob of failure
C maxste:      max number of time steps
C maxi:        max number of x- or r-dir nodes
c c0:          concentration inside the wp (inner b.c.)
c trid:        array containing coefficients for the tridiagonal matrix
c              in the diffusion calculation
c ws:          retardation coefficient R
c rdi:         retardation coefficient R
c dt:          time interval
C cstore:      concentration in cell, gm/l
c nr:          number of node points
c cp(i):       concentration at node point i
c con3:        coefficient for diffusion calculation
c: difflux1:   diffusive flux at the inner sphere radius[gm mol/yr]

```

CCCCC

C Start here.

CCCCC

```

trid(1, 4) = c0

```

C --> set middle points for radionuclide in question

```

do 100 i = 2, nr - 1
  trid(i, 2) = ws(i) + rdi*(al + 1.0/(dt))
  trid(i, 4) = rdi*cstore(kount, i)/(dt)
csss  print*, 'trid=', trid(i,1), trid(i,2), trid(i,3), trid(i,4)
100  continue

```

csss

```

c if (it.eq.165) then
c   print*, 'trid(i,1)=', (trid(i,1), i=1, nr-1)
c   print*, 'trid(i,2)=', (trid(i,2), i=1, nr-1)
c   print*, 'trid(i,3)=', (trid(i,3), i=1, nr-1)
c   print*, 'trid(i,4)=', (trid(i,4), i=1, nr-1)
c   print*
c   print*, '111111 dt, nr=', dt, nr
c   print*
c endif

```

C --> solve the tridiagonal matrix

```

call diag3(trid, nr, cp)

```

```

do 200 i = 1, nr
  cstore(kount, i) = cp(i)

```

csss

```

if ( cstore(kount,i).lt.1.E-20 ) cstore(kount, i) = 1.E-20

```

```

200 continue

```

csss

92/195

```

c      if (it.eq.165) then
c          print*,'cp(i)=',(cp(i),i=1,nr)
c          print*
c      endif

C-->  diffusie flux at the inner sphere radius, gm mol/yr

diffflux1 = con3*(cp(1) - cp(2))
diffflux2 = con4*(cp(marker) - cp(marker+1))

return
end

**==DIAG3.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

```

subroutine diag3(a, n, x)
implicit none
cssss  IMPLICIT NONE
C-----
c      module for implementing Thomas Algorithm
C-----
C*** Start of declarations inserted by SPAG
real a, b, g, w, x
integer i, j, maxi, n, np1
C*** End of declarations inserted by SPAG
C-----
parameter (maxi=20)

dimension a(maxi, 4), w(maxi), b(maxi), g(maxi), x(maxi)
C-----

c      Nomenclature:

C      maxi:      max number of x- or r-dir nodes
c      n:         same as nr in module dflux
c      x:         same as cp in module dflux

C-----
CCCCCC
C      Start here.
CCCCCC

w(1) = a(1, 2)
g(1) = a(1, 4)/w(1)
if ( n.le.1 ) then
    x(1) = g(1)
else

do 50 i = 2, n
cssss  print*,'trid=',a(i,1),a(i,2),a(i,3),a(i,4)
        b(i - 1) = a(i - 1, 3)/w(i - 1)
        w(i) = a(i, 2) - a(i, 1)*b(i - 1)
        g(i) = (a(i,4) - a(i,1)*g(i-1))/w(i)
50    continue

x(n) = g(n)
np1 = n + 1

```


93/195

```

do 100 i = 2, n
  j = np1 - i
  x(j) = g(j) - b(j)*x(j + 1)
100 continue

```

```
end if
```

```
return
end
```

**==GASREL.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

subroutine gasrel(rhouz, r0z, radu, radsg, tfail, thclad, tcan,
&                ttemp, cfuel, czmetal, czoxide, cgap, sc14,
&                rrtfuel, rrtzirc, maxtim, oxwidth)

```

```
implicit none
```

```
cssss IMPLICIT NONE
```

```

C-----
C gaseous releases from a single waste package
C-----

```

C*** Start of declarations inserted by SPAG

```

real al, cfuel, cgap, rrtfuel, rrtzirc, czmetal, czoxide, fclad,
&    r0z, radu, radsg, rc14lb, rxel33, rc14la, rhouz, sc14,
&    tcan, temp, tfail, thclad, time
real timef, ttemp
integer ifflag, itime, maxbin, ntemp, maxtim
real dtt1
real gapfrac
real xxx, oxwidth
real sum1, iflaguo24
real rc14lapr, rc14lbpr, rxel33pr, fcladpr

```

C*** End of declarations inserted by SPAG

```

C-----
external derivgas

parameter (maxbin=500)

common /cther0/ ntemp
common /uo24width/ sum1, iflaguo24

dimension tcan(maxtim), ttemp(maxtim)
dimension sc14(maxbin)

data al/1.21E-4/

```

```

C-----
c
c Nomenclature:

```

```

c al:          radioactive decay coefficient for c14
c rrtfuel:     release rate of c-14 from the sf
c rrtzirc:     release rate of c-14 from the cladding
c ntemp:       num of entries in time versus temp table
c ttemp(time): times in table at which temperatures are given
c tcan(time):  wp surface temp at above times
c dtt1:        time interval [yr]
c rc14la:      output fraction of oxidized clad [dimensionless]

```

94/1195

```
C rc14lb:      output fraction of sf matrix that oxidized
c rxe133:     release rate of Xe-133
C fclad:      output fraction of cladding that oxidized [dim.less]
c r0z:       radius of sf particle [m]
C radu:       radius of outer grain boundary layer[m]
c radsg:     subgrain radius after transgranular fracture [m]
C thclad:    thickness of cladding, meters
C czoxide:   curies of c14 in cladding and other metal oxide / kg sf
C cgap:      curies of c14 in gap and grain boundary / kg sf
C tfail:     time of canister failure, years
C sc14:      output array of c14 release rate, curies
C cfuel:     curies of c14 per kg of fuel
C czmetal:   curies of c14 per kg of metal
c gapfrac:   c14 release from cgap in dtt1 years
C maxbin:    max number of c14 bins
```

C-----

CCCCC

C Start here.

CCCCC

C convert to variables that can be used in the common

```
ifflag = 0
rrtfuel = 0.
rrtzirc = 0.
rc14la = 0.
rc14lb = 0.
rxe133 = 0.
fclad = 0.
timef = 0.
rc14lapr = 0.0
rc14lbpr = 0.0
rxe133pr = 0.0
fcladpr = 0.0
```

```
do 100 itime = 2, ntemp
  temp = tcan(itime - 1)
  time = ttemp(itime)
  dtt1 = ttemp(itime) - ttemp(itime - 1)
```

C normalized release of c-14 and xe-133 and
c differential normalized release from cladding oxidation

```
if ( time.gt.tfail ) then
  timef = tfail
  call uo24oxd(rc14la, radu, dtt1, temp, oxwidth)
  call u3o8oxd(rc14lb, rxe133, radu, radsg, dtt1, temp, timef)
  call zircoxd(thclad, fclad, dtt1, temp)
  write (1, *) 'ttemp,rc14la,rc14lb,rxe133', ttemp(itime),
& rc14la, rc14lb, rxe133
end if
```

```
cdbg print*
cdbg print*, '++++ rc14la,c14lapr:', rc14la, rc14lapr
cdbg print*, 'rc14lb,c14lbpr:', rc14lb, rc14lbpr
cdbg print*, 'rxe133,rxe133pr:', rxe133, rxe133pr
cdbg print*, 'fclad,fcladpr:', fclad, fcladpr
```

C--> normalized release of c-14 by diffusion from outer layer

95/195

```

      rrtfuel = ((rc14la-rc14lapr) + (rc14lb-rc14lbpr)
&          + (rxel33-rxel33pr))/dtt1
      rc14lapr = rc14la
      rc14lbpr = rc14lb
      rxel33pr = rxel33
      fcladpr = fclad

```

C--> release of c-14 by oxidation of cladding

```

      rrtzirc = (fclad - fcladpr)/dtt1

```

c cfuel, czmetal etc. are in ci/kg sf

```

      if ( ttemp(itime).gt.tfail ) then
        if ( ifflag.eq.0 ) then
          xxx = -tfail*al
          if ( xxx.lt. - 65. ) xxx = -65.
          gapfrac = (czoxide + cgap)*exp(xxx)/dtt1
cdbg      print*, 'gapfrac, sc14(itime):', gapfrac, sc14(itime)
          sc14(itime) = rrtfuel*cfuel + rrtzirc*czmetal + gapfrac
          ifflag = 1
        else
          sc14(itime) = rrtfuel*cfuel + rrtzirc*czmetal
          gapfrac = 0.0
        end if
      end if

```

100 continue

```

      return
end

```

**==ODEINT.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

      subroutine odeint(ystart, nvar, x1, x2, eps, h1, hmin, hmax,
&          nok, nbad, tiny, derivs, rkqc, xfrac, xvola,
&          tfail, qin, vmax, amass0, r0z, tavg, qout,
&          dydx, it)

```

implicit none

cssss IMPLICIT NONE

C-----

integer it

C*** Start of declarations inserted by SPAG

```

      real amass0, dydx, eps, errc, xfrac, xvola, h, h1, hdid, hmax,
&          hmin, hnext, qin, qout, tavg, tfail, tiny, vmax, x, x1, x2
      real y, yscal, ystart, zero
      real xsav, r0z, two
      integer kmax, kount
      real dxsav, xp, yp
      real fuewt, fuerden, fuerdif, fuepor, amassc
      integer i, ieq, ifudge, maxstp, nbad, nok, nstp, nvar

```

C*** End of declarations inserted by SPAG

C *****

```

      external derivs, rkqc
      parameter (maxstp=50000, two=2.0, zero=0.0)
      dimension ystart(200), yscal(200), y(200), dydx(200)

```

```
common /cwastr1/ fuewt, fueden, fuepor, amassc
common /path / kmax, kount, dxsav, xp(200), yp(100, 200)
```

```
C *****
C x: independent variable
C h1: guessed first step size
C nok: number of good/okay steps taken
C nbad: number of steps taken but retried and fixed
C y(i): dependent variables
C ystart(i): starting values for integration
C dydx: derivative of the dependent variable
C nvar: number of starting values
C tiny: a small real number
C dxsav: intermediate integration results are stored at intervals > dxsav
C yscal: the vector against which the error is scaled
C hdid: the step size which was actually accomplished
C hnext: the estimated next step size
C derivs: user supplied subroutine for calculating the right hand side
C of the derivative
C x1: lower limit of the integration
C x2: upper limit of the integration
C eps: integration to be done with required accuracy eps
C hmin: minimum allowed step size
C r0z: radius of sf particle
C xfrac: fraction of fuel exposed
C tfail: time of canister failure, years
C qin: inflow of water to canister, liters per year
C vmax: maximum water volume in canister before it overflows, liters
C amass0: mass of fuel[kg]; amassc assigned to amass0
C tavg: same as tcan (wp surface temperature)
C qout: water outflow rate from the wp
C ieq: equation controlling step size,
C errc: the critical error
```

```
C-----
CCCCC
C Start here.
CCCCC
C-----
```

```
      x = x1
      h = sign(h1, x2 - x1)
      nok = 0
      nbad = 0
      kount = 0

      do 100 i = 1, nvar
        y(i) = ystart(i)
100    continue

ccc   print*, 'from odeint at the beginning y(15)=',y(15)

      if ( kmax.gt.0 ) xsav = x - dxsav*two

200  continue
      do 400 nstp = 1, maxstp

c     print*, 'nstp, h, hmax=',nstp,h,hmax

      call derivs(x, y, dydx, xfrac, xvol, tfail, qin, vmax, amass0,
&      tavg, qout, h, it)
      do 250 i = 1, nvar
        yscal(i) = abs(y(i)) + abs(h*dydx(i)) + tiny
```

97/195

```

250      continue

ccc      print*, 'from odeint after first derivs y(15)=' , y(15)

      if ( kmax.gt.0 ) then
      if ( abs(x-xsav).gt.abs(dxsav) ) then
      kount = kount + 1
      xp(kount) = x
      do 260 i = 1, nvar
      yp(i, kount) = y(i)
260      continue
      xsav = x
      end if
      end if

      if ( (x+h-x2)*(x+h-x1).gt.zero ) h = x2 - x
csm pay attention to this extra constraint
      if ( h.gt.hmax ) h = hmax
      call rkqc(y, dydx, nvar, x, h, eps, yscal, hdid, hnext, ieq,
&          errc, derivs, xfrac, xvol, tfail, qin, vmax, amass0,
&          r0z, tavg, qout, it)
      if ( hdid.eq.h ) then
      nok = nok + 1
      else
      nbad = nbad + 1
      end if

ccc      print*, 'from odeint after rkqc h and y(15)=' , h, y(15)

      if ( (x-x2)*(x2-x1).ge.zero ) then
      do 280 i = 1, nvar
      ystart(i) = y(i)

cssss
c      if (it.gt.164) then
c      print*, '#### x,x2,x1=' , x, x2, x1
c      endif
280      continue
      if ( kmax.ne.0 ) then
      kount = kount + 1
      xp(kount) = x
      do 290 i = 1, nvar
      yp(i, kount) = y(i)
290      continue
      end if

c pay close attention to the next two lines
cssss      eps=eps/(10.**ifudge)
cssss      tiny=tiny/(10.**ifudge)
      go to 600

csm      RETURN
      end if

      if ( abs(hnext).lt.hmin ) then
      write (*, *) 'runge-kutta error messages.',
&          ' step size too small'
      write (*, *) 'i,y,dydx,yscal'
      do 300 i = 1, nvar
      write (*, *) i, y(i), dydx(i), yscal(i)
300      continue

      write (*, *) 'equation controlling step size is: ', ieq

```

98/195

```
write (*, *) 'controlling equation error is: ', errc
write (*, *) 'step size is: ', hdid
```

```
C jcw note: if it does not make it then we increase the allowable
C error and try again
```

```
csm pay close attention to the next three lines
```

```
cssss ifudge=ifudge + 1
cssss eps=eps*10.
cssss tiny=tiny*10.
end if
```

```
h = hnext
400 continue
```

```
write (*, *) 'runge-kutta error messages'
write (*, *) 'i, y(i), dydx(i), yscal(i)'
write (*, *) 'time = ', x
```

```
do 500 i = 1, nvar
write (*, *) i, y(i), dydx(i), yscal(i)
500 continue
```

```
write (*, *) 'equation controlling step size is: ', ieq
write (*, *) 'controlling equation error is: ', errc
write (*, *) 'step size is: ', hdid
```

```
C note: if it does not make it then we increase the allowable
C error and try again
```

```
cssss ifudge=ifudge + 1
cssss eps=eps*10.
cssss tiny=tiny*10.
go to 200
C pause 'too many steps.'
```

```
600 continue
return
end
```

**==RKQC.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```
subroutine rkqc(y, dydx, n, x, htry, eps, yscal, hdid, hnext,
& ieq, errc, derivs, xfrac, xvol, tfail, qin,
& vmax, amass0, r0z, tavg, qout, it)
```

```
implicit none
```

```
cssss IMPLICIT NONE
```

```
C-----
C quality controlled runge kutta integration routine
C from numerical recipes
C-----
```

```
C*** Start of declarations inserted by SPAG
```

```
integer it
real r0z, xvol
real fuewt, fuerden, fuerdif, fuepor, amassc
real amass0, dydx, dysav, eps, errc, errcon, errmax, fcor,
```

99/1195

```

& xfrac, h, hdid, hh, hnext, h, one, pgrow, pshrnk, qin,
& qout, safety
real tavg, tfail, vmax, x, xsav, y, ysav, yscal, ytemp
integer i, ieq, n, nmax

```

```

C*** End of declarations inserted by SPAG

```

```

C-----
parameter (nmax=1000, pgrow= - 0.20, pshrnk= - 0.25,
& fcor=1.00/15.00, one=1.00, safety=0.9, errcon=6.D-4)

dimension y(200), dydx(200), yscal(200), ytemp(nmax), ysav(nmax)
& , dysav(nmax)

common /cwast1/ fuewt, fueden, fuedif, fuepor, amassc

external derivs

```

```

C-----

```

```

c Nomenclature:

```

```

c x: independent variable
c htry: step size to be attempted
c n: length of the dependent variable vector y
c ytemp: an array containing error estimates
c derivs: name of the subroutine with the derivative of the eq.
c to be integrated
c y(i): dependent variable vector
c dydx: derivative of the dependent variable
c eps: integration to be done with required overall tolerance eps
c hdid: step size which was actually accomplished
c hnext: estimated next stepsize
c xfrac: fraction of fuel exposed
c tfail: time of canister failure, years
c qin: inflow of water to canister, liters per year
c vmax: maximum water volume in canister before it overflows, liters
c amass0: mass of fuel[kg]; amassc assigned to amass0
c r0z: radius of sf particle
c tavg: same as tcan (wp surface temperature)
c qout: water outflow rate from the wp

```

```

C-----

```

```

xsav = x

do 100 i = 1, n
  ysav(i) = y(i)
  dysav(i) = dydx(i)
100 continue

h = htry
200 continue
hh = 0.5*h

ccc print*, 'before first rk4 call y(15),dydx(15)=',
ccc & y(15),dydx(15)

call rk4(ysav, dysav, n, xsav, hh, ytemp, derivs, xfrac, xvola,
& tfail, qin, vmax, amass0, r0z, tavg, qout, it)
x = xsav + hh

call derivs(x, ytemp, dydx, xfrac, xvola, tfail, qin, vmax,
& amass0, tavg, qout, hh, it)

```

700/1195

```
ccc print*, 'before 2nd rk4 call y(15),dydx(15)=' ,
ccc & y(15),dydx(15)
call rk4(ytemp, dydx, n, x, hh, y, derivs, xfrac, xvol, tfail,
& qin, vmax, amass0, r0z, tavg, qout, it)
x = xsav + h
if ( x.eq.xsav ) then
write (*, *) 'x,xsav=#####=' , x, xsav
write (*, *) 'stepsize not significant in rkqc'
write (*, *) 'The code is forcibly stopped'
stop
end if
```

```
ccc print*, 'before 3rd rk4 call y(15),dydx(15)=' ,
ccc & y(15),dydx(15)
call rk4(ysav, dysav, n, xsav, h, ytemp, derivs, xfrac, xvol,
& tfail, qin, vmax, amass0, r0z, tavg, qout, it)
errmax = 0.
errc = 0.
ieq = 0
```

```
ccc stop
```

```
do 300 i = 1, n
ytemp(i) = y(i) - ytemp(i)
```

csss

```
c print*, 'eps,yscal(i)',eps,yscal(i)
```

```
if ( abs(ytemp(i)/yscal(i)).gt.errmax ) then
ieq = i
errc = ytemp(i)
end if
errmax = max(errmax, abs(ytemp(i)/yscal(i)))
```

```
300 continue
```

```
errmax = errmax/eps
```

```
if ( errmax.gt.one ) then
```

```
h = safety*h*(errmax**pshrnk)
go to 200
```

else

```
hdid = h
```

```
if ( errmax.gt.errcon ) then
```

```
hnext = safety*h*(errmax**pgrow)
```

else

```
hnext = 4.*h
```

```
end if
```

```
end if
```

```
do 400 i = 1, n
```

```
y(i) = y(i) + ytemp(i)*fcor
```

```
400 continue
```

csss

```
c if (it.eq.165) then
```

```
c print*, 'it,ieq in rkqc is=',it,ieq
```

```
c stop
```

```
c endif
```

```
return
```

```
end
```



```

subroutine rk4(y, dydx, n, x, h, yout, derivs, xfrac, xvol,
&          tfail, qin, vmax, amass0, r0z, tavg, qout, it)
implicit none
c      IMPLICIT NONE
C-----
C      runge kutta routine from numerical recipes
C-----

C*** Start of declarations inserted by SPAG
integer it
real r0z, xvol
real amass0, dydx, dym, dyt, xfrac, h, h6, hh, qin, qout, tavg,
&    tfail, vmax, x, xh, y, yout, yt
integer i, n, nmax
real fuewt, fuerden, fuerdif, fuepor, amassc
C*** End of declarations inserted by SPAG
C-----

external derivs
parameter (nmax=1000)
dimension y(200), dydx(200), yout(200), yt(nmax), dyt(nmax),
&    dym(nmax)
common /cwast1/ fuewt, fuerden, fuerdif, fuepor, amassc
C-----
c      n:          number of variables
c      h:          r-k to advance the solution over an interval h
c      xh:
c      y(i):       dependent variables
c      dydx(i):    derivatives of dependent variables
c      yout(i):    an array containing the incremented y values
c      r0z:        radius of the sf particle [m]
c      xfrac:      fraction of fuel exposed
c      tfail:      time of canister failure, years
c      qin:        inflow of water to canister, liters per year
c      vmax:       maximum water volume in canister before it overflows, liters
c      amass0:     mass of fuel[kg]; amassc assigned to amass0
c      tavg:       isame as tcan (wp surface temperature)
c      qout:       water flow rate out of the wp
C-----
CCCCCC
C      Start here.
CCCCCC
C-----
      hh = h*0.5
      h6 = h/6.
      xh = x + hh

      do 100 i = 1, n
         yt(i) = y(i) + hh*dydx(i)
100    continue

      call derivs(xh, yt, dyt, xfrac, xvol, tfail, qin, vmax, amass0,
&          tavg, qout, hh, it)

      do 200 i = 1, n
         yt(i) = y(i) + hh*dyt(i)
200    continue

```

102/195

```

call derivs(xh, yt, dym, xfrac, xvola, tfail, qin, vmax, amass0,
&          tavg, qout, hh, it)

```

```

do 300 i = 1, n
  yt(i) = y(i) + h*dym(i)
  dym(i) = dyt(i) + dym(i)
300 continue

```

```

call derivs(x + h, yt, dyt, xfrac, xvola, tfail, qin, vmax,
&          amass0, tavg, qout, hh, it)

```

```

do 400 i = 1, n
  yout(i) = y(i) + h6*(dydx(i) + dyt(i) + 2.*dym(i))
400 continue

```

```

return
end

```

**=U3080XD.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

```

subroutine u3o8oxd(rc14lb, rxel33, radu, radsg, dtime, t, timef)
implicit none

```

```

c-----
c      IMPLICIT NONE

```

C-----
C cumulative release from dry oxidation of SF matrix to U308
C-----

C*** Start of declarations inserted by SPAG

```

real dfsvty133, dfsvty14, grainr, pi, ratio, rc14lb, rxel33,
&    sum, t, temp, dtime, timef, timehr, xxx1, xxx2
integer n, nseries
real radu, radsg
real sgrainr

```

C*** End of declarations inserted by SPAG

```

C-----
c      nseries:      number of terms to be used in the infinite series
c      pi:          the constant pi
c      radsg:       subgrain radius after transgranular fracture [m]
c      grainr:      grain radius [micrometer]
c      radu:        grainr [m]
c      sgrainr:     subgrain radius after transgranular fracture[micrometer]
c      temp:        temperature [K]
c      t:           temperature [C]
c      timehr:      time [hr]
c      dtime:       time [y]
c      dfsvty14:    diffusion coefficient for c-14 releases
c      n:
c      rc14lb:      output fraction of SF matrix that oxidized
c      dfsvty133:   diffusion coefficient for xe-133 release
c      rxel33:      release rate of Xe-133

```

```

c      example values:
c      grainr=10./2. ! [micrometer]

```

C
C note: gas phase release in U308 (spherical model, Crank,1975)
C the Dmatrix used in the following applies to

103 / 195

C Xe-133, Cl-36, Tc-99, I-129 relea
C-----
CCCCC
C Start here.
CCCCC

nseries = 25
pi = 4.*atan(1.)
grainr = radu*1E6
sgrainr = radsg*1E6

temp = t + 273.
timehr = dtime*365.*24.

if (temp.ge.250. + 273.) then

C group 1: c-14 release

sum = 0.
xxx1 = 26.6E3/1.987/temp
dfsvty14 = 0.6*grainr**2/(2.6E-9*exp(xxx1))

do 50 n = 1, nseries
xxx2 = dfsvty14*n**2*pi**2*timehr/sgrainr**2
if (xxx2.gt.35) go to 100
sum = sum + (1./n**2)*exp(- xxx2)

50 continue

100 continue
ratio = 1. - 6./pi**2*sum
rc14lb = ratio

C group 2: xe-133 release

C at this point, treated as c-14.

sum = 0.
xxx1 = -18.6E3/(1.987*temp)
dfsvty133 = 1.5E-10*exp(xxx1)
dfsvty133 = dfsvty133*1.E-4*1.E12*60*60

do 150 n = 1, nseries
xxx2 = dfsvty133*n**2*pi**2*timehr/(sgrainr)**2
if (xxx2.le.35) sum = sum + (1./n**2)*exp(- xxx2)

150 continue

ratio = 1. - (6./pi**2)*sum
rx133 = ratio

else
rc14lb = 0.
rx133 = 0.
end if

return
end

```
subroutine uo24oxd(c14lc, radu, dtime, t, oxwidth)
```

```
implicit none
```

```
C*** Start of declarations inserted by SPAG
```

```
integer iflaguo24
```

```
C*** End of declarations inserted by SPAG
```

```
cssss IMPLICIT NONE
```

```
C-----
```

```
C normalized release rate from dry oxidation of
```

```
C spent fuel uo2 to uo2.4
```

```
C-----
```

```
C*** Start of declarations inserted by SPAG
```

```
real c14invf, c14lc, oxwidth, ratek, sfpr, t, temp, dtime,
```

```
& timehr
```

```
real radu
```

```
real sum1, oxwidth1
```

```
common /uo24width/ sum1, iflaguo24
```

```
C*** End of declarations inserted by SPAG
```

```
C-----
```

```
c Nomenclature:
```

```
C rc14la: output fraction of oxidized clad [dimensionless]
```

```
C dtime: time [y]
```

```
C timehr: time in hr
```

```
C t: temperature of fuel [c]
```

```
c temp: temperature of fuel [K]
```

```
C radu: radius of SF grain [m]
```

```
c sfpr: radu in micrometer
```

```
c oxwidth: oxidized zone width in a spherical particle[micrometer]
```

```
c c14invf: fractional initial inventory of c-14 in a WP
```

```
c ratek: rate constant [micrometer/hr]
```

```
c c14lc: c14 release in unoxidized SF and UO2.4
```

```
c note: sfpr=0.2/100.*1.E6 ! convert to micrometer
```

```
C-----
```

```
CCCCCC
```

```
C Start here.
```

```
CCCCCC
```

```
sfpr = radu*1.E6
```

```
c14invf = 1.
```

```
temp = t + 273.
```

```
timehr = dtime*365.*24.
```

```
ratek = 1.04E8*exp( - 24.E3/1.987/temp)
```

```
oxwidth1 = sqrt(2.*ratek*timehr)
```

```
sum1 = sum1 + oxwidth1
```

```
if ( sum1.gt.sfpr ) then
```

```
sum1 = 1.0
```

```
iflaguo24 = 1
```

```
end if
```

```
write (1, *) , 'oxd. width and grain rad. [micrometer]=', sum1,
```

```
& sfpr
```

```
if ( iflaguo24.eq.0 ) c14lc = (1./sfpr**3)
```

```
& *(3.*sfpr**2*sum1 - 3.*sfpr*sum1**2 + sum1**3)*c14invf
```

```
oxwidth = sum1*1.E-6
```

write (1, *) 'oxwidth [m]=,temp-2 oxwidth, temp - 273

return
end

**==ZIRCOXD.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

105
/195

subroutine zircoxd(thclad, fclad, dtime, t)
implicit none

cssss IMPLICIT NONE

C-----
C release from zirconium metal upon oxidation model
C of dalgaard ref einziger 1991
C-----

C*** Start of declarations inserted by SPAG
real dweight, fclad, t, temp, thclad, dtime, timedays, zdensity,
& zdensity1
real xxx1

C*** End of declarations inserted by SPAG
C-----

c
c nomenclature:
c
c zdensity: zircalloy density [gm/cc]
c zdensity1: zircalloy density [mg/dm3]
c thclad: thickness of cladding [dm]
c dtime: time [y]
c timedays: time in days
c t: temperature of the cladding [c]
c dweight: weight gain of the oxidizing zircalloy layer [mg/dm2]
c fclad: output fraction of cladding that oxidized [dim.less]
c
c note: the weight gain is assumed to be proportional to the cladding
c oxidized layer thickness. Therefore, the oxidation stops
c when the thickness of the oxidized layer equals that of
c the cladding

C-----
CCCCC
C Start here.
CCCCC

zdensity = 6.56
zdensity1 = (zdensity*1000.)*1000.
thclad = 0.6*1.E-2
temp = t
timedays = dtime*365
cccc dweight=75.3*exp(- 556./temp) + 1.12E8*exp(- 12529./temp)
cccc & *timedays
xxx1 = -12529./temp
if (xxx1.lt. - 65.) xxx1 = -65.
dweight = 1.12E8*exp(xxx1)*timedays
fclad = dweight/(zdensity1*thclad)

return
end

```

subroutine leachrt(temp, srate)
  implicit none
  cssss      IMPLICIT NONE
  C-----
  C      spent fuel matrix dissolution [Stewart and gray, 1994,
  C      Gray and Wilson, 1995]
  C-----
  C*** Start of declarations inserted by SPAG
    real cco3, oxgnovpr, phvalue, rate1, rate2, xfrac, srate, temp,
    &      tempk, xxx
    integer imodel
  C*** End of declarations inserted by SPAG
    common /leach / imodel, phvalue, oxgnovpr, cco3, xfrac
    real srate1, srate2
    real aaa
  C-----

  c      Nomenclature:

  C      model 1###
  C      cco3:      carbonate concentration [moles/liter]
  C      oxgnovpr:  dissolved oxygen (0.002-0.2 atm overpressure)
  C      phvalue:   pH
  C      time:      time [days]
  C      temp:      temperature [C]
  C      tempk:     temp in K
  C      rate1:     spent fuel matrix dissolution rate from
  C                model 1 [mg/m2/day]
  C      srate1:    rate1 in kg/m2/yr
  C      model 2###
  C      rate2:     spent fuel matrix dissolution rate from model 2
  C                [gm/day/205 cm2 surface area]
  C      srate2:    rate2 in kg/yr/m2
  C
  C      srate:     spent fuel matrix dissolution rate [kg/m2/yr]

  C-----
  C      example values:
  C      phvalue:   9.
  C      oxgnovpr:  0.2 ppm
  C      cco3:      2.e-3 mol/liter
  C
  C      note:  ph = -log(h)
  C-----

  CCCCCC
  C      Start here.
  CCCCCC

  tempk = temp + 273.

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

  if ( imodel.eq.1 ) then
    xxx = 9.310 + 0.142*log10(cco3) - 16.7*log10(oxgnovpr)

```

107
/95

```

&      - 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 = 3.45
      rate2 = aaa*1.E4*exp( - 34.3/8.314E-3/tempk)
      srate2 = rate2*1.E-6*365.
      srate = srate2
end if

return
end

```

**==OPNFIL.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996

C

```

subroutine opnfil(unit, filnam, status, lplain, errnum, errmsg)
implicit none

```

csrss IMPLICIT NONE

C-----

```

C      name: opnfil - open file
C
C      purpose:
C      this module opens a formatted file on the unit specified with the
C      attributes provided in the argument list.
C
C      method:
C      a check is first made to determine the availability of the
C      requested unit and file. if either is already in use, then
C      processing is terminated and control is returned to the
C      calling routine. otherwise, 'status' is checked for the
C      desired operation.
C
C      status      operation
C      -----
C      read        open an existing file for input. the file must
C                  exist in a closed state for a successful completion.
C      new          open a new file for output. the file must not
C                  exist for a successful completion.
C      replace     open a new file for output. the file may or may not
C                  exist. if it does exist it is deleted before
C                  opening the new file.
C      append      open a new or existing file for output. the file
C                  may or may not exist. if it does exist it is
C                  opened and positioned at the end of the file so
C                  any new information is appended to the old file.
C                  if it does not exist it is opened as a new file.
C
C
C

```

108/195-

C the fourth argument controls the word attribute of vms type
C file systems. a value of .false. will enable the use of
C fortran carriage control characters as the file is written.
C these files are not convenient for buffering data between
C programs since the first column will have extraneous data.
C for this purpose a value of .true. should be used to provide
C a plain vanilla file without carriage control characters.

C an error number and message are returned to give the status
C of the requested operation. the message string is defined
C in the calling module and should be of length 64 for complete
C messages to be returned.

error number	meaning
0	no error, successful open.
1	requested unit is in use by another file.
2	requested file is in use by another unit or user.
3	requested input file was not found.
4	requested new file is not new, it already exists.
5	an unknown or invalid open request was provided in the argument list.
-1	system failure. the system has not acted in a manner consistent with the assumptions used in the module design.

C arguments:

C input:

C unit = integer, fortran unit number.
 C filnam = character, file name of file to be opened.
 C status = character, file disposition and relation to
 C any existing files with the same name.
 C lplain = logical, flag for the file system attribute to be
 C associated with this file. this is only useful with
 C vms type file systems where fortran carriage control
 C causes different handling characteristics when used
 C with system utilities. true = no fortran carriage
 C control characters in column 1.

C output:

C errnum = integer, error number indicating the success or
 C the conflict of the request with the file system.
 C errmsg = character, error message text.

C history:

C 04-23-92 original text. ron janetzke

Ccccc

integer errnum
 integer unit
 logical lplain
 character*(*) filnam
 character*(*) status
 character*(*) errmsg

Ccccc

C lexist = file exist flag.
 C lfoopen = file open flag.
 C luopen = unit open flag.
 C attrib = carriagecontrol attribute (vms type file systems only).

Ccccc
logical lexist
logical llopen
logical luopen

C-----

Ccccc
C start here.

Ccccc
C the error variables are initialized. with proper operation of
C the 'opnfil' subroutine they should be reset to other values
C before returning. if they are not reset, that would indicate
C a problem with assumptions made about how the operating system
C handles the inquire or open statements.

Ccccc
errnum = -1
errmsg = 'open operation incomplete. check opnfil source code.'

Ccccc
C check the availability of the unit.

Ccccc
inquire (unit=unit, opened=luopen)
if (luopen) then
errnum = 1
errmsg =
& 'unit is open. use of opnfil requires a closed unit.'

return
end if

Ccccc
C check the availability of the file.

Ccccc
inquire (file=filnam, exist=lexist, opened=llopen)
if (llopen) then
errnum = 2
errmsg = 'file is in use, probably on another unit or '//
& 'by another user.'

return
end if

Ccccc
C input file read.
C
C it is suggested that this status be used only for opening pre-
C existing files for input.

Ccccc
if (status.eq.'read' .or. status.eq.'read') then
if (lexist) then
open (unit=unit, file=filnam, status='old')
errnum = 0
errmsg = 'opnfil successful.'
else
errnum = 3
errmsg = 'requested read file was not found.'
end if

Ccccc
C output file new.
C
C it is suggested that this status be used only for opening

110 / 155

```
C files for output.
Ccccc
  else if ( status.eq.'new' .or. status.eq.'new' ) then
    if ( lexist ) then
      errnum = 4
      errmsg = 'new file creation blocked by existing file.'
    else
      open (unit=unit, file=filnam, status='new')
      errnum = 0
      errmsg = 'opnfil successful.'
    end if
```

```
Ccccc
C output file replace.
C
C it is suggested that this status be used only for opening
C files for output.
```

```
Ccccc
  else if ( status.eq.'replace' .or. status.eq.'replace' ) then
    if ( lexist ) then
      open (unit=unit, file=filnam, status='old')
      close (unit=unit, status='delete')
    end if

    open (unit=unit, file=filnam, status='new')
    errnum = 0
    errmsg = 'opnfil successful.'
```

```
Ccccc
C output file append.
C
C it is suggested that this status be used only for opening
C files for output.
Ccccc
  else if ( status.eq.'append' .or. status.eq.'append' ) then
    if ( lexist ) then
      open (unit=unit, file=filnam, status='old', access='append')
    else
      open (unit=unit, file=filnam, status='new')
    end if

    errnum = 0
    errmsg = 'opnfil successful.'
```

```
Ccccc
C unknown file status request.
Ccccc
  else
    errnum = 5
    errmsg = 'file status request is not valid.'
  end if
```

```
Ccccc
  return
end
**==SETDIF.spg processed by SPAG 4.00Aa at 19:34 on 9 Dec 1996
```

subroutine setdif(r0)

11/1/95

implicit none

cssss IMPLICIT NONE

C-----
C set coefficient matrix for the diffusive flow out of wp.
C-----

C*** Start of declarations inserted by SPAG

real a1, a2, a3, amass, amassl, c CFR, con3, con4, cstore,
& dcoefm, dr, dr12, dr32, r0, r12, r32, rr, sumre1, sumre2,
& sumre3, rlr12, rlr32, rlr33, tftc
real amwp1, amwp2, amwp3
real time1, tleach, tlt, trid, wleach, ws
real ppor
integer i, j, maxi, maxmem, maxnuc, maxnxm, maxste, nelem, nr

C*** End of declarations inserted by SPAG

C-----
parameter (maxmem=10)
parameter (maxnuc=50)
parameter (maxste=300)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxi=20)

cssss REAL*8 alam, atoms, atall, am
real alam, atoms, atall, am
common /cnucl3/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
& atoms(maxnuc, maxmem), am(maxnuc, maxmem),
& c CFR(maxnuc, maxmem), tleach, nelem,
& amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
& amassl(maxnuc), tlt, tftc
common /difrel1/ con3, con4, cstore(maxnxm, maxi), trid(maxi, 4),
& rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
& ppor(maxi)
common /array1/ sumre1(maxnuc, maxmem, maxste),
& sumre2(maxnuc, maxmem, maxste),
& sumre3(maxnuc, maxmem, maxste), time1(maxste),
& rlr12(maxnuc, maxmem, maxste),
& rlr32(maxnuc, maxmem, maxste),
& rlr33(maxnuc, maxmem, maxste),
& amwp1(maxnuc, maxmem, maxste),
& amwp2(maxnuc, maxmem, maxste),
& amwp3(maxnuc, maxmem, maxste)

C-----
c Nomenclature:

C a1,2,3: transformed radial coordinate
C cstore: concentration in cell, gm/l
C dcoefm: molecular diffusion coefficient of pack and rock (m2/yr)
C dr: delta x
C dr12,dr32: distance betwe. the center and left & right distance, resp.
C r0: outer radius of sphere
C r12,r32: position of the cenetr of the left and right blocks
C rr: zone distances from the edge of teh wp [m]
C trid: array containing coefficients for the tridiagonal matrix
C in the diffusion calculation
C ws:
C maxi: max number of x- or r-dir nodes
C maxmem: max number of enteries in table: age, prob of failure
C maxnuc: max number of isotopes
C maxste: max number of time steps
C nr: number of zones for diffusion calculation (same as imax)
C

112/195

```

C Note: assumes a spherical shell surrounding spherical source term.
c advection considered seperately, and results of advection and
c diffusion added assumes molecular diffusion only in the sphere
C accounts for decay of parent, but no generation of daughters
C in the sphere.
C inner boundary at r.n. concentrations inside the wp;
c outer boundary at r.n. concentration=0.0

```

```

C-----

```

```

CCCCCC

```

```

C Start here.

```

```

CCCCCC

```

```

      do 100 i = 1, maxnxm
        do 50 j = 1, maxi
          cstore(i, j) = 1.E-20
50      continue
100     continue

      rr(1) = r0

      write (1, *) 'nr in setdif =', nr

      do 200 i = 2, nr
        rr(i) = rr(i - 1) + dr(i - 1)
200     continue

```

```

C set up some of the coefficients for the tridiagonal matrix

```

```

trid(1, 1) = 0.0
trid(1, 2) = 1.0
trid(1, 3) = 0.0

```

```

do 300 i = 2, nr - 1
  dr32 = (dr(i) + dr(i+1))/2
  dr12 = (dr(i-1) + dr(i))/2
  r12 = (rr(i-1) + rr(i))/2
  r32 = (rr(i) + rr(i+1))/2
  a1 = dcoefm(i - 1)*r12**2/(rr(i)**2*dr(i-1)*dr12)
  a3 = dcoefm(i + 1)*r32**2/(dr(i)*rr(i)**2*dr32)
  a2 = a1 + a3
  trid(i, 1) = -a1
  trid(i, 3) = -a3
  ws(i) = a2

```

```

300 continue

```

```

C zero concentration at outer boundary

```

```

trid(nr, 1) = 0.0
trid(nr, 2) = 1.0
trid(nr, 3) = 0.0
trid(nr, 4) = 0.0

```

```

return
end

```

113 / 1195

```

subroutine init
implicit none
c-----
csss      IMPLICIT NONE
C-----

```

```

C*** Start of declarations inserted by SPAG
  real amassc, cftime, ctemp, cvol, floref, flowfactr, flow,
&      fueledn, fueledif, fuewt, funnel, rpor, sftime, simtimex,
&      tcaqu, tflo, deffrac
  real xco, xcon
  integer i, imax, iscon, ize, j, jmax, maxi, maxj, maxtim,
&      mtot, icfcon, ntemp, nzones
  real wplen, wpdia, driftdia
  real xvol, dintl, xlintl
  real fuepor

```

```

C*** End of declarations inserted by SPAG
C-----

```

```

Ccccc
  parameter (maxtim=500)
  parameter (maxi=20, maxj=30)

  common /ccont2/ deffrac
  common /ccont5/ sftime, iscon
  common /ccoro1/ cftime, tcaqu, icfcon
  common /cgeom / cvol, xcon, funnel, mtot
  common /cnume1/ simtimex, imax, jmax
  common /cnume2/ xco(maxi), nzones, ize(maxi, maxj)
  common /crhyd1/ rpor(maxi), flowfactr, flow, tflo(maxtim),
&      floref(maxtim)
  common /cther2/ ctemp
  common /cwast1/ fuewt, fueledn, fueledif, fuepor, amassc
  common /wpvol / wplen, wpdia, driftdia, xvol, dintl, xlintl
  common /cther0/ ntemp

```

```

C-----

```

```

c
c      Nomenclature:
c
c      cvol:          cell volume
c      xcon:          num of packages in a cell
c      wplen:         length of package
c      wpdia:         diameter of waste package
c      ctemp:         crit. temp.[C] for the onset of liquid flow around wp
c      ntemp:         num of entries in time versus temp table
c      rpor:          porosity:: three layers damaged, disturbed, host rock
c      flow:          flow rate into the container
c      floref:        reference flow at tflo
c      icfcon:        number failed containers in the cell due to corrosion
c      tcaqu:         time at which aqueous conditions begin in cell i
c      cftime:        corrosion failure time [y]
c      sftime:        time at which scenario causes container failures in a cell
c      fuewt:         formula wt for spent fuel
c      fueledn:       fuel density
c      fueledif:      fuel diffusion coefficient
c      imax:          nodes in x (or r) direction
c      jmax:          nodes in y (or theta) direction
c      simtimex::     maximum simulation time [y]
c      xco:           x (or r) coordinates
c      ize(i,j):      zone number of node (i,j)

```

```

C-----

```

114/115

```
c      Notes:
C      1 cal=4.186 joule
C      gas constant, r = 8.134 joules/mol-k
```

C-----

```
CCCCC
C      Start here.
CCCCC
```

C--> geometric elements: length,width,height for wp and drift

```
cvol = 1.
xcon = 1.
wplen = 1.
wpdia = 1.
```

C--> thermal parameters

```
ctemp = 97.
ntemp = 0
```

C--> hydrologic parameters

```
do 100 i = 1, maxi
  rpor(i) = 0.1
100 continue
```

```
flow = 0.
```

```
do 200 i = 1, maxtim
  floref(i) = 0.
200 continue
```

C--> container data

```
icfcon = 0
tcaqu = 0.
cftime = 1.E30
sftime = 1.E20
```

C--> radionuclide properties

```
fuewt = 0.
fueden = 1.
fuedif = 1.
```

C--> grid and time step information

```
imax = 1
jmax = 1
simtimex = 0.
```

```
do 300 i = 1, maxi
  xco(i) = 1.
300 continue
```

```
do 400 i = 1, maxi
  do 350 j = 1, maxj
    ize(i, j) = 1
350 continue
400 continue
```

115/195

return
end

**==INPUT.spg processed by SPAG 4.00Aa at 19:35 on 9 Dec 1996

subroutine input
implicit none

cssss IMPLICIT NONE

C-----

C*** Start of declarations inserted by SPAG

```

real amass, amassc, amassl, ccf, cco3, cftime, cfuel, cgap,
& con3, con4, cstore, ctemp, cvol, czmetal, czoxide, defrac,
& dcoefm, dr
real diff
real fuepor
real xfrac
real floref, flowfactr, flow, fueden, fuedif, fuewt, funnel,
& oxgnovpr, phvalue, ppor, r0z, radu, radsg, rhouz, rpor, rr
real wetfrac, simtimex, sumre1, sumre2, sumre3, rlratel,
& rlratel2, rlratel3, tcan, tcaqu, tflo, tftc, thclad, time1,
& tleach, tlt, trid, ttemp
real amwp1, amwp2, amwp3
real wleach, ws, xco, xcon, yco, sftime
integer nbt
integer i, ib, ie, imax, imodel, it, iz, ione, j, jb, je, jmax,
& m, maxi, maxj, maxmem, maxnuc, maxnxm, maxste, maxtim
integer mtot, icfcon, nelem, nr, ntemp, ntflo, nzones,
& iscon
real dtinit, dtmax, dtmin, eps, tiny
real wplen, wpdia, driftdia, xvol, dintl, xlintl
cssss REAL*8 alam, atoms, atall, am
real alam, atoms, atall, am

```

C*** End of declarations inserted by SPAG

C-----

```

parameter (maxmem=10)
parameter (maxnuc=50)
parameter (maxste=300)
parameter (maxtim=500)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxi=20, maxj=30)

common /ccont2/ defrac
common /ccont5/ sftime, iscon
common /ccoro1/ cftime, tcaqu, icfcon
common /cgeom / cvol, xcon, funnel, mtot
common /cnucl3/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
& atoms(maxnuc, maxmem), am(maxnuc, maxmem),
& ccf(maxnuc, maxmem), tleach, nelem,
& amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
& amassl(maxnuc), tlt, tftc
common /difrel1/con3, con4, cstore(maxnxm, maxi), trid(maxi, 4),
& rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
& ppor(maxi)
common /array1/ sumre1(maxnuc, maxmem, maxste),
& sumre2(maxnuc, maxmem, maxste),
& sumre3(maxnuc, maxmem, maxste), time1(maxste),
& rlratel(maxnuc, maxmem, maxste),

```

116/
1/55

```

&          rlr2(maxnuc, maxmem, maxste),
&          rlr3(maxnuc, maxmem, maxste),
&          amwp1(maxnuc, maxmem, maxste),
&          amwp2(maxnuc, maxmem, maxste),
&          amwp3(maxnuc, maxmem, maxste)

common /cnum1/ simtimex, imax, jmax
common /cnum2/ xco(maxi), nzones, ize(maxi, maxj)
common /cnum3 / diff(maxi)
common /cther0/ ntemp
common /crhyd1/ rpor(maxi), flowfactr, flow, tflo(maxtim),
&          floref(maxtim)
common /cther1/ ttemp(maxtim), tcan(maxtim)
common /cther2/ ctemp
common /cwest1/ fuewt, fueden, fuedif, fuepor, amassc
common /rcarbon1/ rOz, rhouz, radu, radsg
common /rcarbon2/ cfuel, czmetal, czoxide, cgap
character*32 , temfil, elefil, hydfil
common /leach / imodel, phvalue, oxgnovpr, cco3, xfrac
common /leach1/ wetfrac
common /rkutta/ eps, tiny, dtinit, dtmin, dtmax
common /outpt / nbt
common /wpvol / wplen, wpdia, driftdia, xvola, dintl, xlintl
real tave(maxtim)
integer nunit(9)

```

c Nomenclature:

```

C amassc:      initial mass of uranium oxide fuel, kg
C cco3:        carbonate concentration [moles/liter]
C cftime:      corrosion failure time [y]
C cfuel:       ci of c14 / kg of fuel in fuel
C cgap:        ci of c14 / kg of fuel in grain boundary and gap
C ctemp:       crit. temp.[C] for the onset of liquid flow around wp
C cvol:        cell volume
C czmetal:     ci of c14 / kg of fuel in zirconium cladding
C czoxide:     ci of c14 / kg of fuel in initial zirc. oxide and crud
C defrac:      fraction of initially failed packages
C dcoefm:      mol. diffusion coefficient for pack and rock [m^2/yr]
C dr:          delta x
c fuepor:      porosity in between SF particles
C floref:      reference flow at tflo
C flowfactr:   flow loading in a cell
c flow:        same as floref(1)
C fueden:      fuel density
C fuedif:      fuel diffusion coefficient
C fuewt:       formula wt for spent fuel
C funnel:      area above wp where water is assumed to funnel
C oxgnovpr:    dissolved oxygen (0.002-0.2 atm overpressure)
c phvalue:     pH
C ppor:        porosity of packing
C rOz:         initial radius of uo2 grain, m
C radu:        radius of uo2 fragment, m
C radsg:       subgrain radius after transgranular fracture [m]
C rhouz:       density of uo2 kg mole/m^3
C rpor:        porosity:: three layers damaged, disturbed, host rock
C wetfrac:     the fractional liquid level in the wp below which
c              the waste matrix is wet
C simtimex:    maximum simulation time [y]
C sumre(1,2,3): release history:1=defective,2=scenario,3=normal

```


117/190

```

C          failure in the cell, the results are overwritten in
C          sumrel(maxnuc,maxmem,time step)
C          rlrates(1,2,3): release rate history:1=defective,2=scenario,3=normal
C          failure in the cell, the results are overwritten in
C          rlrates(maxnuc,maxmem,time step)
C          tcan(time,cell):wp surface temp at above times
C          tcaqu:         time at which aqueous condition begins in the cell
C          tflo:         time of table entry i for floref
C          thclad:       thickness of cladding, m
C          time1,2,3:    times corresponding to releases for each type of
C                       container failure
C          ttemp(time):  times in table at which temperatures are given
C          xco:         x (or r) coordinates
C          xcon:        num of packages in a cell
C          yco:         y (or theta) coordinates
C          sftime:     time at which scenario causes wp failures in a cell
C          nbt:        output time interval
C          imax:       nodes in x (or r) direction
C          imodel:    an index for identifying leaching rate model
C          ize(i,j):   zone number of node (i,j)
C          iz:        zone identifier
C          ib:        beginning x-node for a zone
C          jb:        beginning y-node for a zone
C          ie:        ending x-node for a zone
C          je:        ending y-coordinate for a zone
C          jmax:      nodes in y (or theta) direction
C          maxi:     max number of x- or r-dir nodes
C          maxj:     max number of y- or theta dir nodes
C          maxmem:   max number of entries in table: age, prob of failure
C          maxnuc:   max number of isotopes
C          maxste:   max number of time steps
C          maxtim:   max (time-data set) values
C          mtot:    total number of containers in the cell
C          icfcon:  number failed containers in the cell due to corrosion
C          nr:     number of zones for diffusion cal. (same as imax)
C          ntemp:  num of entries in time versus temp table
C          ntflo:  num of entries in flow table
C          nzones: number of material zones
C          iscon:  integer variable of the same thing
C          dtinit: initial time step used in the R-K algorithm
C          dtmax:  maximum allowed timestep
C          dtmin:  minimum allowed timestep
C          eps:   integration accuracy for runge kutta integration
C          tiny:  a small number needed for odeint subroutine
C          wplen: wp length [m]
C          wpdia: diameter of waste package [m]
C          xvola: interior volume of the wp [m^3]
C          dintl: internal diameter of the wp [m]
C          xlintl: internal length of the wp [m]

```

```

C-----
CCCCCC
C      Start here.
CCCCCC

```

```

C-->  unit numbers for i/o

nunit(1) = 5
nunit(2) = 6
nunit(3) = 10

```

118 / 195

```
nunit(4) = 11
nunit(5) = 12
nunit(6) = 13
nunit(7) = 14
nunit(8) = 16
nunit(9) = 17
open (unit=4, file='release.inp', status='old')
```

C==> cell parameters

```
C--> read # of waste packages/cell, scenario failure WPs,
c    time of scenario failure, and # of defective (pre-failed) WPs
read (4, *)
read (4, *)
read (4, *)
read (4, *) xcon
read (4, *) iscon
read (4, *) sftime
read (4, *) defrac
write (1, *) 'xcon,defrac=', xcon, defrac

mtot = int(xcon + 0.05)
```

C==> waste package size

```
C--> read radius of the sphere with equivalent surface area as that
c    of the WP and the maximum volume before WP water overflows

read (4, *)
read (4, *)
read (4, *) dintl, xlintl
read (4, *) xvola
write (1, *)
write (1, *) 'xvola, dintl, xlintl=', xvola, dintl, xlintl
```

C==> environmental parameters

C--> thermal

```
C    temperature and rel humidity data as a function of time
C    ttemp: time; tcan: ave temp
C    filename is temphumd.dat:- four values per row
C    (time, wp surface temp, drift wall temp, relative humidity)
read (4, *)
read (4, *)
read (4, *) temfil
write (1, *) 'name of temeprature file- ', temfil
open (nunit(3), file=temfil, status='old')
read (nunit(3), *) simtimex
read (nunit(3), *) ntemp
read (nunit(3), *)
write (1, *) 'ntemp=', ntemp
write (1, *) 'it    ttemp    tcan    tave    humd'

do 100 it = 1, ntemp
    read (nunit(3), *) it, ttemp(it), tcan(it), tave(it)
    write (1, *) it, ttemp(it), tcan(it), tave(it)
100 continue
```

119/195

```
read (nunit(3), *) cftime
write(1,*) 'corrosion failure time cftime[yr]=', cftime
```

c passing wp dimensions (for this version)

```
read (nunit(3), *) wplen, wpdia
write (1, *) 'wp dia and length [m]=', wplen, wpdia
```

C--> read ctemp: crit. temp.[C] for the onset of liquid flow around wp

```
read (4, *) ctemp
write (1, *) 'ctemp=', ctemp
```

C==> liquid flow rate

```
read (4, *)
read (4, *)
read (4, *) hydfil
read (4, *) funnel
```

C--> read infiltration and load factors from 'floebs.dat'

c ntflo is hardwired until future development

```
ntflo = 2
open (nunit(5), file=hydfil, status='old')
read (nunit(5), *) flowfactr
```

```
do 200 i = 1, ntflo
  read (nunit(5), *) tflo(i), floref(i)
  write (1, *) 'i,tflo,floref=', i, tflo(i), floref(i)
```

200 continue

c assigned the flow rate at time =0 as the steady state flow rate

c in this version of EBSPAC

```
flow = floref(1)
```

```
write (1, *) 'ntflo=', ntflo
write (1, *) 'flow=', flow
write (1, *) 'flowfactr=', flowfactr
```

C==> waste material

C--> spent fuel properties

C read amount of SF in a WP [kg], SF mol. wt, SF density, SF diffusion

c coefficient??, SF porosity

```
read (4, *)
read (4, *)
read (4, *)
read (4, *) amassc
read (4, *) fuewt, fueden, fuedif, fuepor
write (1, *) 'amassc=', amassc
write (1, *) 'fuewt=', fuewt
write (1, *) 'fueden=', fueden
write (1, *) 'fuedif=', fuedif
write (1, *) 'fuepor=', fuepor
```

C==> SF leaching parameters

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

```

read (4, *)
read (4, *) imodel, phvalue, oxgnovpr, cco3, wetfrac
write (1, *) 'imodel,phvalue,oxgnovpr,cco3,wetfrac=', imodel,
&          phvalue, oxgnovpr, cco3, wetfrac

```

C--> nuclide inventory

```

C      read nuclide inventory from 'ebspac.nuc' file.
C      nuclide names from the file 'ebspac.nuc'

```

```

read (4, *)
read (4, *)
read (4, *) elefil
write (1, *) elefil

```

C--> read properties of radionuclides from a table with four
C columns: nuclide name, half life,specific activity,
C and parent-daughter relationship.parent=1,daughter=11,111,1111 etc.
C these are independent of time and cell numbers

```

call rdelem(nunit(9), elefil)

```

C==> Parameters for C-14 gas release

```

C      data provided for r0,rhou,dref1,e,oxz,
C      radu,thclad,cfuel,czmetal,czoxide,cgap

```

```

read (4, *)
read (4, *)
read (4, *) r0z
read (4, *) rhouz
read (4, *) radu
read (4, *) radsg
read (4, *) thclad
read (4, *) cfuel
read (4, *) czmetal
read (4, *) czoxide
read (4, *) cgap
write (1, *) 'r0z, rhouz, radu,radsg', r0z, rhouz, radu, radsg
write (1, *) 'thclad, cfuel, czmetal, czoxide, cgap', thclad,
&          cfuel, czmetal, czoxide, cgap

```

C==> grid and time step info

C--> spatial grid

```

read (4, *)
read (4, *)
read (4, *)
read (4, *)
read (4, *) imax, jmax
write (1, *) 'imax=', imax
write (1, *) 'jmax=', jmax

```

```

nr = imax

```

C--> read x-coordinates

```

read (4, *)
read (4, *) (xco(i), i=1, imax)

```

121/1995

```
dr(1) = xco(1)

do 300 i = 2, nr
  dr(i) = xco(i) - xco(i - 1)
300 continue

C--> read y-coordinates

read (4, *)
read (4, *) (yco, i=1, jmax)

C--> set zones (material types)
C syntax: zone 1 from (2,2) to (5,5)

read (4, *)
read (4, *)
read (4, *) nzones

do 400 m = 1, nzones
  read (4, *) iz, ib, jb, ie, je
  do 350 j = jb, je
    do 320 i = ib, ie
      ize(i, j) = iz
320 continue
350 continue
400 continue

C==== rock and backfill hydrological parameters

c rpor(zone): porosity is assumed to be constant in time
c zone 1 is teh backfill zone
C porosity is assumed to be constant in time
C porosity = 1 means air gap between wp and drift

read (4, *)
read (4, *)
read (4, *) (rpor(i), i=1, nzones)
write (1, *) 'rpor(i):', (rpor(i), i=1, nzones)

C==> Radionuclide transport properties

C--> diffusion coefficients in rock

read (4, *)
read (4, *)
read (4, *) (diff(i), i=1, 4)

c--> drift diameter

read (4, *) driftdia

C==> parameters to control solution algorithm

read (4, *)
read (4, *)
read (4, *) dtinit, dtmin, dtmax
read (4, *) eps, tiny

c--> parameters for results output
```

```
read (4, *)
read (4, *)
read (4, *) nbt
```

```
return
end
```

**==RDELEM.spg processed by SPAG 4.00Aa at 19:35 on 9 Dec 1996

```
subroutine rdelem(unit, elefil)
implicit none
```

```
cssss IMPLICIT NONE
```

```
C-----
c module to input the radionuclide inventory and properties
C-----
```

C*** Start of declarations inserted by SPAG

```
real amass, amassl, c CFR, ci1000, con3, con4, cstore, dcoefm,
& dr, rr, sumre1, sumre2, sumre3, rlr1, rlr2, rlr3,
& tftc, time1, tleach, tlt, trid, wleach
real amwp1, amwp2, amwp3
real ws
real ppor
integer i, ielem, inuc, k, l, maxele, maxi, maxmem, maxnuc,
& maxnxm, maxste, nelem, nn, nr
integer unit
integer ncon, kcon, icon, nchns, ni
```

```
real sol, rd, rde, amall, halflife, curall, fggap
cssss REAL*8 alam, atoms, atall, am
real alam, atoms, atall, am
```

C*** End of declarations inserted by SPAG

```
C-----
parameter (maxele=50)
parameter (maxmem=10)
parameter (maxnuc=50)
parameter (maxste=300)
parameter (maxnxm=maxnuc*maxmem)
parameter (maxi=20)
character*(*) elefil
character*3 elem
character*6 namall
common /cnucl1/ elem(maxele), namall(maxnuc, maxmem)
common /cnucl2/ ncon(maxele), sol(maxele), rde(maxele),
& amall(maxnuc, maxmem), halflife(maxnuc, maxmem),
& curall(maxnuc, maxmem), fggap(maxnuc, maxmem),
& kcon(maxele, maxnuc), icon(maxele, maxnuc),
& rd(maxnuc, maxmem), ni(maxnuc), nchns,
& ci1000(maxnuc, maxmem)
common /cnucl3/ atall(maxnuc, maxmem), alam(maxnuc, maxmem),
& atoms(maxnuc, maxmem), am(maxnuc, maxmem),
& c CFR(maxnuc, maxmem), tleach, nelem,
& amass(maxnuc, maxmem), wleach(maxnuc, maxmem),
& amassl(maxnuc), tlt, tftc
common /difrel1/con3, con4, cstore(maxnxm, maxi), trid(maxi,4),
& rr(maxi), dr(maxi), ws(maxi), nr, dcoefm(maxi),
& ppor(maxi)
common /array1/ sumre1(maxnuc, maxmem, maxste),
```

1231
/175-

```

&      sumre2(maxnuc, maxmem, maxste),
&      sumre3(maxnuc, maxmem, maxste), time1(maxste),
&      rlrates1(maxnuc, maxmem, maxste),
&      rlrates2(maxnuc, maxmem, maxste),
&      rlrates3(maxnuc, maxmem, maxste),
&      amwp1(maxnuc, maxmem, maxste),
&      amwp2(maxnuc, maxmem, maxste),
&      amwp3(maxnuc, maxmem, maxste)

```

C-----

```

C
C      Nomenclature:
C
C      maxi:      max number of x- or r-dir nodes
C      namall:    the radionuclide name
C      maxele:    max number of elements (radionuclides)
C      maxnuc:    max number of isotopes
C      maxmem:    max number of entries in table: age, prob of failure
C      maxste:    max number of time steps
C      maxtim:    max (time-data set) values
C      nelem:     number of elements
C      elefil:    parameter to read radionuclide inventory file
C      elem(ielem): name of all element (different from namall)
C      ielem:     element identifier
C      ncon(ielem): number of chains in which isotopes of element ielem occurs
C      sol(ielem): solubility limit of elemets
C      rde(ielem): retardation factor for elements
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)
C      rd:        retardation factor for nuclides (same as elements)
C      nchns:     number of chains
C      ni(i):     size of chains
C      namall(k,i): name of all radionuclides
C      amall:     molecular weight of radionuclides
C      halflife:  half life of radionuclides [yr]
C      curall    radionuclide inventory per WP in curies
C      fggap:    gap fraction of each radionuclide
C      inuc:     a counter for counting the number of nuclides

```

C-----

```

CCCCCC
C      Start here.
CCCCCC
c-->  open file for radionuclide inventory, elefil is the
C      file name parameter

      open (unit=unit, file=elefil, status='old')

C      read number of elements

      read (unit, *) nelem

      do 100 ielem = 1, nelem

C          read element name, num.chains in which it appears, and
C          solubility gram moles/liter, and retardation coefficient

      read (unit, 9001) elem(ielem), ncon(ielem), sol(ielem),
&      rde(ielem)
      write (1, *) 'ielem,elem,ncon,sol,rde', ielem, elem(ielem),

```

124/195

```

&          ncon(ielem), sol(ielem), rde(ielem)

c          read l and ielem locations in chains of all same elements

do 50 l = 1, ncon(ielem)

    read (unit, *) kcon(ielem, l), icon(ielem, l)
    write (1, *) 'ielem,l,kcon(ielem,l),icon(ielem,l)=', ielem,
&          l, kcon(ielem, l), icon(ielem, l)

c          fill in the radionuclide retardation from the element

    rd(kcon(ielem,l), icon(ielem,l)) = rde(ielem)

50  continue
100 continue

c--> isotope data

c          read number and size of chains
c
c          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
200  continue

return
9001 format (a3, 2x, i5, 2F10.0)
9002 format (e10.0, 4x, a6, 4E10.0)
end

**==LINT.spg processed by SPAG 4.00Aa at 19:35 on 9 Dec 1996

```

```

subroutine lint(xa, ya, n1, n, x, y)
implicit none
C*** Start of declarations inserted by SPAG
real h, x, xa, y, ya
integer k, khi, klo, n, n1
C*** End of declarations inserted by SPAG
c-----
c          module for linear interpolation between two specified limits
c-----
c          dimension xa(n1), ya(n1)
c-----
c
c Nomenclature:
c
c          xa:    vector of length n of x values (independent variable)

```


125/
195

c ya: vector of length n of y values
c n: length of xa and ya vectors
c x: x value at which interpolation is required
c y: returned value of dependent variable corresponding to x value
c

```
-----  
      if ( xa(1).lt.xa(n) ) then  
c increasing order list  
      if ( x.le.xa(1) ) then  
        y = ya(1)  
      else if ( x.ge.xa(n) ) then  
        y = ya(n)  
      else  
        klo = 1  
        khi = n  
20      continue  
        if ( khi - klo.gt.1 ) then  
          k = (khi + klo)/2  
          if ( xa(k).gt.x ) then  
            khi = k  
          else  
            klo = k  
          end if  
          go to 20  
        end if  
        h = xa(khi) - xa(klo)  
        if ( h.eq.0. ) pause 'bad xa input.'  
        y = ya(klo) + (x - xa(klo))/h*(ya(khi) - ya(klo))  
      end if  
c decreasing order list  
      else if ( x.ge.xa(1) ) then  
        y = ya(1)  
      else if ( x.le.xa(n) ) then  
        y = ya(n)  
      else  
        klo = 1  
        khi = n  
50      continue  
        if ( khi - klo.gt.1 ) then  
          k = (khi + klo)/2  
          if ( xa(k).lt.x ) then  
            khi = k  
          else  
            klo = k  
          end if  
          go to 50  
        end if  
        h = xa(khi) - xa(klo)  
        if ( h.eq.0. ) pause 'bad xa input.'  
        y = ya(klo) + (x - xa(klo))/h*(ya(khi) - ya(klo))  
      end if  
  
      return  
      end
```

pcl5

JOB 1786

example_release.inp

For: scratchy1!mohanty
Date: Mon Jan 6 18:09:24 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:41:44 1991
Started: Wed Nov 13 10:41:44 1991

136/195

```

\input data file for ebspac release code: ebspac_release.f
|
\Cell information
2335.          ! xcon: number of WP
1000          ! iscon: # of WP in scenario failure
4500.         ! sftime: time of scenario failure [y]
0.00          ! defrac: defective fraction of WPs/cell
|
\WP information
1.421, 4.572  ! dintl: wp ID, xintl: internal length [m]
4.83          ! xvola: wp internal vol[m3]
|
\Thermal data
'temphumd.dat' ! temfil: temp. file (output from ebspac_fail.f)
97.           ! ctemp: BP of water at atm. condition [C]
|
\Flow parameters
'floebd.dat'  ! hydfil: flow parameters file
7.           ! funnel: funnel factor
|
\SF materials
\
2800.         ! amassc: SF mass [kg]
250., 10000., 1.0E-08, 0.5 ! fuewt,fueden,fuedif,fuepor
\
\Fuel leaching
2, 9.0, 0.2, 2.e-3, 5.e-1 ! imodel,phvalue,oxgnovpr [atm],cco3 [mol/L],wetfrac
|
\ Radionuclide inventory
'ebspac.nuc'  ! elefil: nuclide names, halflife,inventory
|
\C-14 generation
0.5e-3        ! r0z: initial radius of SF particle [m]
37.           ! rhou: density of SF [kg mole/m^3]
1.e-5         ! r1z: radius of the SF grain [m]
1.e-6         ! r2z: subgrain fragment radius after trans. frac. [m]
6.1e-4        ! thclad: thickness of cladding [m]
7.2e-4        ! cfuel: c-14 [ci] /kg SF
4.89e-4       ! czmetal: c-14 [ci] /kg SF in Zry clad & other metals
2.48e-5       ! czoxide: c-14/kg SF in initial Zry oxide & crud
6.2e-6        ! cgap: c-14/kg SF in grain and gap
|
\NUMERICAL
\
\Grids
10, 10        ! imax,jmax: # of grid nodes in i,j directions
\X-COOR of grid nodes
.1, .5, 1.0, 2.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0
\Y-COOR of grid nodes
.1, .5, 1.0, 2.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0
\ Note: zones apply to the above grid, zones are the same for all cell
\ZONES
4             ! nzones: no. of zones for material types
1  1  1  1  1 ! iz,ib,jb,ie,je: for zone 1
2  2  1  2  1 ! iz,ib,jb,ie,je: for zone 2
3  3  1  3  1 ! iz,ib,jb,ie,je: for zone 3
4  4  1  10 1 ! iz,ib,jb,ie,je: for zone 4
|
\Rock parameters
0.14, 0.34, 0.34, 0.34 ! rpor(1..nzones): rock porosity

```

137/195

```
|  
\ Radionuclide transport  
5.6e-5, 5.6e-5, 5.6e-5, 5.6e-5 ! rdiff(1..nzones):diffusion coef. [m2/yr]  
5.0 ! driftdia [m]  
|  
\ Solution algorithm control parameters (Runge-Kutta)  
25., 0.0, 10. ! dtinit [yr], dtmin [yr], dtmax [yr]  
1.0e-2, 1.0e-10 ! eps, tiny  
|  
\output parameters  
200 ! nbt: number of time intervals for output  
|  
\END
```

pcl5

JOB 1774

ebspacrun

For: scratchy1!mohanty
Date: Mon Jan 6 17:53:37 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:25:46 1991
Started: Wed Nov 13 10:25:46 1991



QMS 3825 Print System

QMS 3825 Print System

139/195

```
f77 fail.f -o fail.x  
f77 -r8 release4.f -o release4.x  
rm echo1.out  
rm echo2.out  
time fail.x > echo1.out  
time release4.x > echo2.out
```

pcl5

JOB 1771

example_fail.inp

For: scratchy1!mohanty
Date: Mon Jan 6 17:43:02 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:15:11 1991
Started: Wed Nov 13 10:15:11 1991

```

\example input file for ebspac_fail
|
\simulation time
10000.          ! tend: simulation time length [yr]
\              ! when iflag=1 (defined later)
\
5.682, 1.802    ! wplen,wpdia: wp length and diameter [m]
0.1, 0.02      ! cthick1,cthick2: wp layers 1&2 thicknesses [m]
|
\choose source of temperature data
2              ! iflag 1: emp. equation, 2: tab.data
1              ! nset (temp.-rel hum. relationship to use)
49.9999999    ! timintv (used when iflag=2)
|
\other temperature parameters
0.             ! age of fuel (not used in this version)
|
\Dry oxidation of wp outer overpack
5.             ! grainr: metal grain radius [micrometer]
25            ! nseries (terms in the infinite series)
0.7e-3       ! gbthick [micrometer]
1.e-2        ! constant: used in the dry oxidation equn.
|
\evaporation-condensation
0.65         ! humdc: critical relative humidity
2.e-3       ! filmthk: thickness of water film [m]
97.         ! ctemp: boiling point of water [C]
|
\Corrosion Parameters(Ep: pitting potential [mV]; Erp: repassivation potential [mV])
-584.8      ! xipto: outer overpack Ep intercept
3.92       ! pttemo: temp. coef. of outer overpack Ep intercept
-24.5      ! slpto: outer overpack Ep slope
-1.1       ! slpttemo: temp. coef. of outer overpack Ep slope
-620.3     ! xirpo: outer overpack Erp intercept
0.47       ! rptemo: temp. coef. of outer overpack Erp intercept
-95.2      ! slrpo: outer overpack Erp slope
0.88       ! slrptemo: temp. coef. of outer overpack Erp slope
200.       ! xipti: inner overpack Ep intercept
0.         ! pttemi: temp. coef. of inner overpack Ep intercept
-240.      ! slpti: inner overpack Ep slope
0.         ! slpttemi: temp. coef. of inner overpack Ep slope
422.8      ! xirpi: inner overpack Erp intercept
-4.1       ! rptemi: temp. coef. of inner overpack Erp intercept
-64.       ! slrpi: inner overpack Erp slope
-0.80      ! slrptemi: temp. coef. of inner overpack Erp slope
0.75, 0.5  ! betaox1, betahy1: beta kinetics parameters
\          ! for oxygen and water for WP outer overpack
0.75, 0.5  ! betaox2, betahy2: beta kinetics parameters for
\          ! oxygen and water for WP inner overpack
3.80e12, 1.6e-1 ! rkox1 [c*m/y/mol], rkhy1 [c/m2/yr]
37300., 25000. ! gox1 [J/mol], ghy1 [J/mol]
3.0e10, 3.2    ! rkox2 [c*m/y/m], rkhy2 [c/m2/yr]
40000., 25000. ! gox2 [J/mol], ghy2 [J/mol]
3.15e5, 0.0, 0.0 ! aa(1,1) [C/m2/yr], aa(1,2), aa(1,3)
6.30e4, 0.0, 0.0 ! aa(2,1) [C/m2/yr], aa(2,2), aa(2,3)
-0.46        ! eexpt: in Vshe
8.66e-3     ! rcoef:
0.45        ! rexpont:
2.05e-4     ! crate2: m/yr corrosion rate
0.0         ! xcouple, a fractional coupling strength

```

141/195

142/195

```
0.0          ! xread
3.e-1       ! clconc: chloride concentration [mol/L]
3.e-4       ! clcrit1: crit. chloride conc. for 1st layer [mol/L]
2.e-3       ! clcrit2: crit. chloride conc. for 2nd layer [mol/L]
100.        ! cfactor: chloride dilution factor
9.0         ! refph: reference pH
1.0, 1.0    ! taus:deposit tortuosity,spor:deposit porosity
|
\Runge-kutta control parameters
1.e-3, 1.e0 ! dtini, dtmax
1.e-2, 1.e-30 ! errrel (same as eps), errabs (same as tiny)
|
\end
////2000          ! nhista (used when iflag=1)
```

143 / 195

pcl5

JOB 1767

echo1.out

For: scratchy1!mohanty
Date: Mon Jan 6 17:39:45 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:11:54 1991
Started: Wed Nov 13 10:11:54 1991

144/155

CALCULATION OF WASTE PACKAGE FAILURE TIME

```

=====
                end of simulation time [yr]:      10000.
    No. of rows of data to pass to release.f:      196
-----
i  layer  tstop      tcan      ecrit      ecorr  chloride  rthick      mode
      [yr]      [C]      [Vshe]    [Vshe]    flag      [m]
=====
1      50.42     161.90     0.0000     0.0000     0    1.1999993E-01  dry oxd
1      100.50     155.60     0.0000     0.0000     0    1.1999986E-01  dry oxd
1      152.40     150.70     0.0000     0.0000     0    1.1999980E-01  dry oxd
1      206.00     147.40     0.0000     0.0000     0    1.1999974E-01  dry oxd
1      259.40     145.80     0.0000     0.0000     0    1.1999968E-01  dry oxd
1      312.80     144.70     0.0000     0.0000     0    1.1999962E-01  dry oxd
1      366.20     143.90     0.0000     0.0000     0    1.1999956E-01  dry oxd
1      419.70     143.00     0.0000     0.0000     0    1.1999950E-01  dry oxd
1      473.10     142.30     0.0000     0.0000     0    1.1999944E-01  dry oxd
1      533.80     141.00     0.0000     0.0000     0    1.1999938E-01  dry oxd
1      585.70     139.90     0.0000     0.0000     0    1.1999932E-01  dry oxd
1      637.50     138.80     0.0000     0.0000     0    1.1999926E-01  dry oxd
1      689.30     137.70     0.0000     0.0000     0    1.1999920E-01  dry oxd
1      741.10     136.60     0.0000     0.0000     0    1.1999914E-01  dry oxd
1      793.00     135.60     0.0000     0.0000     0    1.1999908E-01  dry oxd
1      844.80     134.60     0.0000     0.0000     0    1.1999902E-01  dry oxd
1      896.60     133.70     0.0000     0.0000     0    1.1999896E-01  dry oxd
1      948.40     132.70     0.0000     0.0000     0    1.1999890E-01  dry oxd
1     1000.00     131.80     0.0000     0.0000     0    1.1999884E-01  dry oxd
1     1065.00     130.50     0.0000     0.0000     0    1.1999878E-01  dry oxd
1     1129.00     129.30     0.0000     0.0000     0    1.1999872E-01  dry oxd
1     1194.00     128.00     0.0000     0.0000     0    1.1999866E-01  dry oxd
1     1258.00     126.80     0.0000     0.0000     0    1.1999860E-01  dry oxd
1     1323.00     125.60     0.0000     0.0000     0    1.1999854E-01  dry oxd
1     1388.00     124.40     0.0000     0.0000     0    1.1999848E-01  dry oxd
1     1452.00     123.30     0.0000     0.0000     0    1.1999842E-01  dry oxd
1     1517.00     122.10     0.0000     0.0000     0    1.1999836E-01  dry oxd
1     1581.00     121.00     0.0000     0.0000     0    1.1999830E-01  dry oxd
1     1646.00     119.90     0.0000     0.0000     0    1.1999824E-01  dry oxd
1     1710.00     118.80     0.0000     0.0000     0    1.1999818E-01  dry oxd
1     1775.00     117.70     0.0000     0.0000     0    1.1999813E-01  dry oxd
1     1839.00     116.70     0.0000     0.0000     0    1.1999807E-01  dry oxd
1     1904.00     115.60     0.0000     0.0000     0    1.1999801E-01  dry oxd
1     1986.00     114.40     0.0000     0.0000     0    1.1999795E-01  dry oxd
1     2036.00     113.60     0.0000     0.0000     0    1.1999789E-01  dry oxd
1     2086.00     112.90     0.0000     0.0000     0    1.1999783E-01  dry oxd
1     2136.00     112.30     0.0000     0.0000     0    1.1999777E-01  dry oxd
1     2186.00     111.60     0.0000     0.0000     0    1.1999771E-01  dry oxd
1     2236.00     111.00     0.0000     0.0000     0    1.1999765E-01  dry oxd
1     2286.00     110.30     0.0000     0.0000     0    1.1999759E-01  dry oxd
1     2336.00     109.70     0.0000     0.0000     0    1.1999753E-01  dry oxd
1     2386.00     109.10     -0.5694     0.1950     1    6.8837903E-02  local
1     2436.00     108.50     -0.5695     0.1945     1    5.0405446E-02  local
1     2486.00     107.90     -0.5695     0.1940     1    3.6636310E-02  local
1     2536.00     107.30     -0.5695     0.1935     1    2.5225966E-02  local
2     2586.00     106.80     0.0631     0.1932     1    1.4995669E-02  local
2     2636.00     106.20     0.0653     0.1932     1    4.7456686E-03  local
2     2686.00     105.70     0.0671     0.1932     1   -5.5043314E-03  local

```

145/195

```
=====
wp wetting time [yr]:      2336.
wp failure time [yr]:     2636.
penetration by dry oxidation [m]: 2.467E-06
echoed input data in:    echo_fail.dat
output data are in files: temphumd.dat and corrode.out
=====
```

146/191

pcl5

JOB 1768

temphumd.dat

For: scratchy1!mohanty
Date: Mon Jan 6 17:40:03 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:12:11 1991
Started: Wed Nov 13 10:12:11 1991

10000.0000000000

147/195

196

i, timr, tcan, tavg, humd

1	0.0002	21.8700	21.0000	0.9480
2	50.4200	161.9000	147.6000	0.1399
3	100.5000	155.6000	146.4000	0.1645
4	152.4000	150.7000	143.8000	0.1872
5	206.0000	147.4000	141.6000	0.2046
6	259.4000	145.8000	140.7000	0.2138
7	312.8000	144.7000	140.0000	0.2203
8	366.2000	143.9000	139.6000	0.2253
9	419.7000	143.0000	139.1000	0.2310
10	473.1000	142.3000	138.6000	0.2355
11	533.8000	141.0000	137.6000	0.2443
12	585.7000	139.9000	136.7000	0.2520
13	637.5000	138.8000	135.8000	0.2600
14	689.3000	137.7000	134.9000	0.2683
15	741.1000	136.6000	133.9000	0.2770
16	793.0000	135.6000	133.0000	0.2851
17	844.8000	134.6000	132.2000	0.2935
18	896.6000	133.7000	131.3000	0.3014
19	948.4000	132.7000	130.5000	0.3104
20	1000.0000	131.8000	129.6000	0.3187
21	1065.0000	130.5000	128.5000	0.3313
22	1129.0000	129.3000	127.3000	0.3434
23	1194.0000	128.0000	126.2000	0.3572
24	1258.0000	126.8000	125.0000	0.3705
25	1323.0000	125.6000	123.9000	0.3843
26	1388.0000	124.4000	122.8000	0.3988
27	1452.0000	123.3000	121.7000	0.4126
28	1517.0000	122.1000	120.6000	0.4284
29	1581.0000	121.0000	119.5000	0.4435
30	1646.0000	119.9000	118.5000	0.4592
31	1710.0000	118.8000	117.4000	0.4755
32	1775.0000	117.7000	116.4000	0.4926
33	1839.0000	116.7000	115.4000	0.5087
34	1904.0000	115.6000	114.4000	0.5272
35	1986.0000	114.4000	113.1000	0.5483
36	2036.0000	113.6000	112.4000	0.5629
37	2086.0000	112.9000	111.7000	0.5760
38	2136.0000	112.3000	111.1000	0.5876
39	2186.0000	111.6000	110.4000	0.6014
40	2236.0000	111.0000	109.8000	0.6136
41	2286.0000	110.3000	109.2000	0.6282
42	2336.0000	109.7000	108.6000	0.6410
43	2386.0000	109.1000	108.0000	0.6541
44	2436.0000	108.5000	107.4000	0.6675
45	2486.0000	107.9000	106.8000	0.6813
46	2536.0000	107.3000	106.3000	0.6954
47	2586.0000	106.8000	105.7000	0.7074
48	2636.0000	106.2000	105.1000	0.7221
49	2686.0000	105.7000	104.6000	0.7347
50	2736.0000	105.1000	104.1000	0.7501
51	2786.0000	104.6000	103.5000	0.7632
52	2836.0000	104.0000	103.0000	0.7793
53	2886.0000	103.5000	102.5000	0.7930
54	2936.0000	103.0000	102.0000	0.8070
55	2986.0000	102.5000	101.5000	0.8213
56	3036.0000	102.0000	101.0000	0.8359
57	3086.0000	101.6000	100.6000	0.8478

148/115

58	3136.0000	101.1000	100.1000	0.8630
59	3186.0000	100.7000	99.6900	0.8753
60	3236.0000	100.3000	99.2700	0.8879
61	3286.0000	99.8400	98.8500	0.9026
62	3336.0000	99.4300	98.4500	0.9160
63	3386.0000	99.0300	98.0500	0.9292
64	3436.0000	98.6300	97.6600	0.9427
65	3486.0000	98.2500	97.2800	0.9557
66	3536.0000	97.8700	96.9000	0.9654
67	3586.0000	97.5000	96.5300	0.9653
68	3636.0000	97.1300	96.1700	0.9656
69	3686.0000	96.7700	95.8200	0.9659
70	3736.0000	96.4200	95.4700	0.9658
71	3786.0000	96.0800	95.1200	0.9654
72	3836.0000	95.7400	94.7900	0.9657
73	3886.0000	95.4000	94.4500	0.9656
74	3936.0000	95.0700	94.1300	0.9659
75	3986.0000	94.7500	93.8100	0.9658
76	4036.0000	94.4300	93.4900	0.9657
77	4086.0000	94.1100	93.1800	0.9660
78	4136.0000	93.8000	92.8700	0.9660
79	4186.0000	93.5000	92.5700	0.9659
80	4236.0000	93.2000	92.2700	0.9658
81	4286.0000	92.9000	91.9800	0.9661
82	4336.0000	92.6100	91.6900	0.9661
83	4386.0000	92.3300	91.4100	0.9660
84	4436.0000	92.0400	91.1300	0.9663
85	4486.0000	91.7600	90.8500	0.9663
86	4536.0000	91.4900	90.5800	0.9662
87	4586.0000	91.2200	90.3100	0.9662
88	4636.0000	90.9500	90.0500	0.9665
89	4686.0000	90.6900	89.7900	0.9664
90	4736.0000	90.4300	89.5300	0.9664
91	4786.0000	90.1700	89.2700	0.9663
92	4836.0000	89.9200	89.0200	0.9662
93	4886.0000	89.6700	88.7800	0.9666
94	4936.0000	89.4300	88.5300	0.9661
95	4986.0000	89.1800	88.2900	0.9665
96	5036.0000	88.9400	88.0500	0.9664
97	5086.0000	88.7000	87.8100	0.9664
98	5136.0000	88.4600	87.5700	0.9663
99	5186.0000	88.2200	87.3400	0.9666
100	5236.0000	87.9800	87.1100	0.9670
101	5286.0000	87.7500	86.8800	0.9669
102	5336.0000	87.5200	86.6500	0.9669
103	5386.0000	87.2900	86.4200	0.9668
104	5436.0000	87.0600	86.2000	0.9672
105	5486.0000	86.8400	85.9800	0.9671
106	5536.0000	86.6200	85.7600	0.9671
107	5586.0000	86.4000	85.5400	0.9670
108	5636.0000	86.1800	85.3200	0.9670
109	5686.0000	85.9600	85.1100	0.9673
110	5736.0000	85.7500	84.9000	0.9673
111	5786.0000	85.5400	84.6900	0.9672
112	5836.0000	85.3300	84.4800	0.9672
113	5886.0000	85.1200	84.2800	0.9675
114	5936.0000	84.9200	84.0700	0.9671
115	5986.0000	84.7100	83.8700	0.9674
116	6036.0000	84.5100	83.6700	0.9674
117	6086.0000	84.3100	83.4800	0.9677

149/195

118	6136.0000	84.1100	83.2800	0.9677
119	6186.0000	83.9200	83.0900	0.9676
120	6236.0000	83.7200	82.9000	0.9680
121	6286.0000	83.5300	82.7100	0.9680
122	6336.0000	83.3400	82.5200	0.9679
123	6386.0000	83.1500	82.3300	0.9679
124	6436.0000	82.9700	82.1500	0.9678
125	6486.0000	82.7800	81.9600	0.9678
126	6536.0000	82.6000	81.7800	0.9678
127	6586.0000	82.4200	81.6000	0.9677
128	6636.0000	82.2400	81.4300	0.9681
129	6686.0000	82.0600	81.2500	0.9680
130	6736.0000	81.8800	81.0700	0.9680
131	6786.0000	81.7100	80.9000	0.9680
132	6836.0000	81.5300	80.7300	0.9683
133	6886.0000	81.3600	80.5600	0.9683
134	6936.0000	81.1900	80.3900	0.9683
135	6986.0000	81.0200	80.2200	0.9682
136	7036.0000	80.8500	80.0600	0.9686
137	7086.0000	80.6900	79.8900	0.9682
138	7136.0000	80.5200	79.7300	0.9685
139	7186.0000	80.3600	79.5700	0.9685
140	7236.0000	80.1900	79.4100	0.9688
141	7286.0000	80.0300	79.2500	0.9688
142	7336.0000	79.8700	79.0900	0.9688
143	7386.0000	79.7200	78.9300	0.9684
144	7436.0000	79.5600	78.7800	0.9687
145	7486.0000	79.4000	78.6200	0.9687
146	7536.0000	79.2500	78.4700	0.9687
147	7586.0000	79.1000	78.3200	0.9686
148	7636.0000	78.9400	78.1700	0.9690
149	7686.0000	78.7900	78.0200	0.9690
150	7736.0000	78.6400	77.8700	0.9689
151	7786.0000	78.4900	77.7200	0.9689
152	7836.0000	78.3500	77.5800	0.9689
153	7886.0000	78.2000	77.4300	0.9688
154	7936.0000	78.0600	77.2900	0.9688
155	7986.0000	77.9100	77.1500	0.9692
156	8036.0000	77.7700	77.0100	0.9692
157	8086.0000	77.6300	76.8700	0.9691
158	8136.0000	77.4900	76.7300	0.9691
159	8186.0000	77.3500	76.5900	0.9691
160	8236.0000	77.2100	76.4500	0.9690
161	8286.0000	77.0700	76.3200	0.9694
162	8336.0000	76.9400	76.1800	0.9690
163	8386.0000	76.8000	76.0500	0.9694
164	8436.0000	76.6700	75.9100	0.9689
165	8486.0000	76.5300	75.7800	0.9693
166	8536.0000	76.4000	75.6500	0.9693
167	8586.0000	76.2700	75.5200	0.9693
168	8636.0000	76.1400	75.3900	0.9692
169	8686.0000	76.0100	75.2600	0.9692
170	8736.0000	75.8800	75.1400	0.9696
171	8786.0000	75.7500	75.0100	0.9696
172	8836.0000	75.6200	74.8800	0.9695
173	8886.0000	75.5000	74.7600	0.9695
174	8936.0000	75.3700	74.6400	0.9699
175	8986.0000	75.2500	74.5100	0.9695
176	9036.0000	75.1200	74.3900	0.9698
177	9086.0000	75.0000	74.2700	0.9698

178	9136.0000	74.8800	74.1500	0.9698
179	9186.0000	74.7600	74.0300	0.9698
180	9236.0000	74.6400	73.9100	0.9698
181	9286.0000	74.5200	73.7900	0.9697
182	9336.0000	74.4000	73.6700	0.9697
183	9386.0000	74.2800	73.5600	0.9701
184	9436.0000	74.1600	73.4400	0.9701
185	9486.0000	74.0500	73.3200	0.9696
186	9536.0000	73.9300	73.2100	0.9700
187	9586.0000	73.8200	73.1000	0.9700
188	9636.0000	73.7000	72.9800	0.9700
189	9686.0000	73.5900	72.8700	0.9700
190	9736.0000	73.4800	72.7600	0.9699
191	9786.0000	73.3600	72.6500	0.9703
192	9836.0000	73.2500	72.5400	0.9703
193	9886.0000	73.1400	72.4300	0.9703
194	9936.0000	73.0300	72.3200	0.9703
195	9986.0000	72.9200	72.2100	0.9702
196	10000.0000	72.8900	72.1800	0.9702
	2636.0000000000			
	5.68200000000000			
	1.80200000000000			

150 / 195

151/195

pcl5

JOB 1769

corrode.out

For: scratchy1!mohanty
Date: Mon Jan 6 17:40:15 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:12:34 1991
Started: Wed Nov 13 10:12:34 1991

152/195

ilayer	tstop [yr]	tcan [C]	ecrit [Vshe]	ecorr [Vshe]	chloride flag	rthick [m]	mode
1	50.42	161.90	0.0000	0.0000	0	1.1999993E-01	
1	100.50	155.60	0.0000	0.0000	0	1.1999986E-01	
1	152.40	150.70	0.0000	0.0000	0	1.1999980E-01	
1	206.00	147.40	0.0000	0.0000	0	1.1999974E-01	
1	259.40	145.80	0.0000	0.0000	0	1.1999968E-01	
1	312.80	144.70	0.0000	0.0000	0	1.1999962E-01	
1	366.20	143.90	0.0000	0.0000	0	1.1999956E-01	
1	419.70	143.00	0.0000	0.0000	0	1.1999950E-01	
1	473.10	142.30	0.0000	0.0000	0	1.1999944E-01	
1	533.80	141.00	0.0000	0.0000	0	1.1999938E-01	
1	585.70	139.90	0.0000	0.0000	0	1.1999932E-01	
1	637.50	138.80	0.0000	0.0000	0	1.1999926E-01	
1	689.30	137.70	0.0000	0.0000	0	1.1999920E-01	
1	741.10	136.60	0.0000	0.0000	0	1.1999914E-01	
1	793.00	135.60	0.0000	0.0000	0	1.1999908E-01	
1	844.80	134.60	0.0000	0.0000	0	1.1999902E-01	
1	896.60	133.70	0.0000	0.0000	0	1.1999896E-01	
1	948.40	132.70	0.0000	0.0000	0	1.1999890E-01	
1	1000.00	131.80	0.0000	0.0000	0	1.1999884E-01	
1	1065.00	130.50	0.0000	0.0000	0	1.1999878E-01	
1	1129.00	129.30	0.0000	0.0000	0	1.1999872E-01	
1	1194.00	128.00	0.0000	0.0000	0	1.1999866E-01	
1	1258.00	126.80	0.0000	0.0000	0	1.1999860E-01	
1	1323.00	125.60	0.0000	0.0000	0	1.1999854E-01	
1	1388.00	124.40	0.0000	0.0000	0	1.1999848E-01	
1	1452.00	123.30	0.0000	0.0000	0	1.1999842E-01	
1	1517.00	122.10	0.0000	0.0000	0	1.1999836E-01	
1	1581.00	121.00	0.0000	0.0000	0	1.1999830E-01	
1	1646.00	119.90	0.0000	0.0000	0	1.1999824E-01	
1	1710.00	118.80	0.0000	0.0000	0	1.1999818E-01	
1	1775.00	117.70	0.0000	0.0000	0	1.1999813E-01	
1	1839.00	116.70	0.0000	0.0000	0	1.1999807E-01	
1	1904.00	115.60	0.0000	0.0000	0	1.1999801E-01	
1	1986.00	114.40	0.0000	0.0000	0	1.1999795E-01	
1	2036.00	113.60	0.0000	0.0000	0	1.1999789E-01	
1	2086.00	112.90	0.0000	0.0000	0	1.1999783E-01	
1	2136.00	112.30	0.0000	0.0000	0	1.1999777E-01	
1	2186.00	111.60	0.0000	0.0000	0	1.1999771E-01	
1	2236.00	111.00	0.0000	0.0000	0	1.1999765E-01	
1	2286.00	110.30	0.0000	0.0000	0	1.1999759E-01	
1	2336.00	109.70	0.0000	0.0000	0	1.1999753E-01	
1	2386.00	109.10	-0.5694	0.1950	1	6.8837903E-02	
1	2436.00	108.50	-0.5695	0.1945	1	5.0405446E-02	
1	2486.00	107.90	-0.5695	0.1940	1	3.6636310E-02	
1	2536.00	107.30	-0.5695	0.1935	1	2.5225966E-02	
2	2586.00	106.80	0.0631	0.1932	1	1.4995669E-02	
2	2636.00	106.20	0.0653	0.1932	1	4.7456686E-03	
2	2686.00	105.70	0.0671	0.1932	1	-5.5043314E-03	

Rews Uniquely Identified
153/195

pcl5

JOB 1770

echo2.out

For: scratchy1!mohanty
Date: Mon Jan 6 17:40:34 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:12:43 1991
Started: Wed Nov 13 10:12:43 1991



QMS 3825 Print System

QMS 3825 Print System

154/155

 failure type: 1
 number of failed wp: 0
 wp failure time [yr]: 0.
 onset of flow [yr]: 3686.

 failure type: 2
 number of failed wp: 0
 wp failure time [yr]: 4500.
 onset of flow [yr]: 3686.

no release due to scenario failure because it occurred after corrosion failure

 failure type: 3
 number of failed wp: 2335
 wp failure time [yr]: 2636.
 onset of flow [yr]: 3686.
 sf leaching time span [yr]: 186.
 sf leaching onset time [yr]: 3686.
 sf leaching will end at [yr]: 3872.

time [yr]	cwpwater [kg]	rgenerate [kg/yr]	rdecay [kg/yr]	rconv [kg/yr]	rdiff_r0 [kg/yr]	rdiff_ebs [kg/yr]
3736.0	1.397E-07	1.581E+15	1.560E+15	3.132E-09	1.118E-11	2.373E-21
3786.0	3.017E-07	1.598E+15	1.577E+15	3.165E-09	1.130E-11	2.494E-21
3836.0	4.686E-07	1.614E+15	1.593E+15	3.198E-09	1.141E-11	2.521E-21
3886.0	6.128E-07	1.290E+14	1.270E+14	2.550E-10	8.997E-13	1.806E-21
3936.0	6.268E-07	3.244E+11	6.328E+10	1.270E-13	4.438E-16	1.519E-25
3986.0	6.393E-07	2.622E+11	3.916E+10	7.861E-14	2.805E-16	1.333E-27
4036.0	6.518E-07	2.243E+11	3.351E+10	6.727E-14	2.400E-16	1.499E-27
4086.0	6.644E-07	1.915E+11	2.865E+10	5.752E-14	2.053E-16	2.477E-27
4136.0	6.771E-07	1.633E+11	2.451E+10	4.921E-14	1.756E-16	2.990E-27
4186.0	6.898E-07	1.444E+11	2.167E+10	4.350E-14	1.552E-16	2.286E-27
4236.0	7.026E-07	1.277E+11	1.913E+10	3.841E-14	1.370E-16	1.713E-27
4286.0	7.154E-07	1.128E+11	1.688E+10	3.390E-14	1.210E-16	1.214E-27
4336.0	7.282E-07	9.961E+10	1.490E+10	2.991E-14	1.067E-16	8.299E-28
4386.0	7.411E-07	8.792E+10	1.314E+10	2.637E-14	9.412E-17	5.205E-28
4436.0	7.540E-07	7.757E+10	1.158E+10	2.325E-14	8.295E-17	2.435E-28
4486.0	7.670E-07	6.842E+10	1.020E+10	2.047E-14	7.305E-17	1.731E-29
4536.0	7.800E-07	5.903E+10	9.118E+09	1.830E-14	6.532E-17	1.439E-27
4586.0	7.931E-07	5.217E+10	8.062E+09	1.618E-14	5.776E-17	1.205E-27
4636.0	8.062E-07	4.610E+10	7.126E+09	1.431E-14	5.105E-17	9.981E-28
4686.0	8.193E-07	4.072E+10	6.296E+09	1.264E-14	4.511E-17	8.150E-28
4736.0	8.325E-07	3.595E+10	5.562E+09	1.117E-14	3.985E-17	6.528E-28
4786.0	8.457E-07	3.174E+10	4.912E+09	9.861E-15	3.519E-17	5.093E-28

time [yr]	cwpwater [kg]	rgenerate [kg/yr]	rdecay [kg/yr]	rconv [kg/yr]	rdiff_r0 [kg/yr]	rdiff_ebs [kg/yr]
--------------	------------------	----------------------	-------------------	------------------	---------------------	----------------------

155 / 195

4836.0	8.589E-07	2.802E+10	4.337E+09	8.706E-15	3.107E-17	3.824E-28
4886.0	8.722E-07	2.472E+10	3.828E+09	7.685E-15	2.743E-17	2.703E-28
4936.0	8.855E-07	2.182E+10	3.379E+09	6.783E-15	2.421E-17	1.712E-28
4986.0	8.989E-07	1.925E+10	2.982E+09	5.987E-15	2.137E-17	8.697E-29
5036.0	9.123E-07	1.700E+10	2.633E+09	5.285E-15	1.886E-17	7.676E-29
5086.0	9.257E-07	1.501E+10	2.325E+09	4.667E-15	1.665E-17	6.777E-29
5136.0	9.391E-07	1.326E+10	2.053E+09	4.122E-15	1.471E-17	5.986E-29
5186.0	9.526E-07	1.172E+10	1.814E+09	3.643E-15	1.300E-17	5.289E-29
5236.0	9.661E-07	1.036E+10	1.604E+09	3.220E-15	1.149E-17	4.675E-29
5286.0	9.796E-07	9.166E+09	1.419E+09	2.849E-15	1.017E-17	4.134E-29
5336.0	9.931E-07	8.115E+09	1.256E+09	2.521E-15	8.997E-18	3.659E-29
5386.0	1.007E-06	7.189E+09	1.112E+09	2.233E-15	7.969E-18	3.240E-29
5436.0	1.020E-06	6.375E+09	9.860E+08	1.979E-15	7.064E-18	2.872E-29
5486.0	1.034E-06	5.658E+09	8.748E+08	1.756E-15	6.268E-18	2.547E-29
5536.0	1.048E-06	5.028E+09	7.770E+08	1.560E-15	5.567E-18	2.262E-29
5586.0	1.061E-06	4.473E+09	6.909E+08	1.387E-15	4.950E-18	2.011E-29
5636.0	1.075E-06	3.985E+09	6.152E+08	1.235E-15	4.408E-18	1.790E-29
5686.0	1.089E-06	3.557E+09	5.486E+08	1.101E-15	3.931E-18	1.596E-29
5736.0	1.102E-06	3.180E+09	4.901E+08	9.840E-16	3.512E-18	1.426E-29
5786.0	1.116E-06	2.849E+09	4.387E+08	8.808E-16	3.143E-18	1.276E-29
5836.0	1.130E-06	2.558E+09	3.935E+08	7.900E-16	2.819E-18	1.144E-29
5886.0	1.144E-06	2.302E+09	3.537E+08	7.102E-16	2.534E-18	1.028E-29

time cwpwater rgenerate rdecay rconv rdiff_r0 rdiff_ebs
[yr] [kg] [kg/yr] [kg/yr] [kg/yr] [kg/yr] [kg/yr]

5936.0	1.158E-06	2.076E+09	3.188E+08	6.400E-16	2.284E-18	9.261E-30
5986.0	1.171E-06	1.878E+09	2.880E+08	5.782E-16	2.063E-18	8.364E-30
6036.0	1.185E-06	1.704E+09	2.609E+08	5.238E-16	1.869E-18	7.575E-30
6086.0	1.199E-06	1.550E+09	2.371E+08	4.759E-16	1.698E-18	6.880E-30
6136.0	1.213E-06	1.415E+09	2.161E+08	4.338E-16	1.548E-18	6.267E-30
6186.0	1.227E-06	1.295E+09	1.976E+08	3.966E-16	1.415E-18	5.728E-30
6236.0	1.241E-06	1.190E+09	1.812E+08	3.639E-16	1.299E-18	5.252E-30
6286.0	1.255E-06	1.097E+09	1.669E+08	3.350E-16	1.195E-18	4.833E-30
6336.0	1.269E-06	1.015E+09	1.542E+08	3.095E-16	1.104E-18	4.463E-30
6386.0	1.283E-06	9.431E+08	1.430E+08	2.870E-16	1.024E-18	4.136E-30
6436.0	1.297E-06	8.792E+08	1.330E+08	2.671E-16	9.532E-19	3.847E-30
6486.0	1.311E-06	8.226E+08	1.243E+08	2.495E-16	8.904E-19	3.592E-30
6536.0	1.325E-06	7.725E+08	1.165E+08	2.339E-16	8.348E-19	3.366E-30
6586.0	1.339E-06	7.281E+08	1.097E+08	2.201E-16	7.856E-19	3.166E-30
6636.0	1.353E-06	6.887E+08	1.036E+08	2.079E-16	7.419E-19	2.988E-30
6686.0	1.367E-06	6.536E+08	9.814E+07	1.970E-16	7.031E-19	2.830E-30
6736.0	1.381E-06	6.225E+08	9.333E+07	1.874E-16	6.686E-19	2.690E-30
6786.0	1.395E-06	5.947E+08	8.904E+07	1.788E-16	6.379E-19	2.565E-30
6836.0	1.409E-06	5.699E+08	8.521E+07	1.711E-16	6.105E-19	2.453E-30
6886.0	1.423E-06	5.477E+08	8.179E+07	1.642E-16	5.860E-19	2.353E-30
6936.0	1.437E-06	5.278E+08	7.873E+07	1.581E-16	5.641E-19	2.264E-30
6986.0	1.451E-06	5.100E+08	7.599E+07	1.525E-16	5.444E-19	2.184E-30

time cwpwater rgenerate rdecay rconv rdiff_r0 rdiff_ebs
[yr] [kg] [kg/yr] [kg/yr] [kg/yr] [kg/yr] [kg/yr]

7036.0	1.465E-06	4.939E+08	7.352E+07	1.476E-16	5.267E-19	2.112E-30
7086.0	1.479E-06	4.794E+08	7.129E+07	1.431E-16	5.107E-19	2.047E-30
7136.0	1.493E-06	4.662E+08	6.928E+07	1.391E-16	4.963E-19	1.988E-30
7186.0	1.508E-06	4.543E+08	6.745E+07	1.354E-16	4.832E-19	1.935E-30
7236.0	1.522E-06	4.435E+08	6.579E+07	1.321E-16	4.714E-19	1.887E-30
7286.0	1.536E-06	4.336E+08	6.428E+07	1.291E-16	4.605E-19	1.843E-30
7336.0	1.550E-06	4.245E+08	6.290E+07	1.263E-16	4.506E-19	1.803E-30

156/195

7386.0	1.564E-06	4.162E+08	6.100E+07	1.237E-16	4.415E-19	1.766E-30
7436.0	1.578E-06	4.085E+08	6.046E+07	1.214E-16	4.331E-19	1.731E-30
7486.0	1.592E-06	4.014E+08	5.938E+07	1.192E-16	4.254E-19	1.700E-30
7536.0	1.607E-06	3.947E+08	5.837E+07	1.172E-16	4.182E-19	1.671E-30
7586.0	1.621E-06	3.886E+08	5.744E+07	1.153E-16	4.115E-19	1.643E-30
7636.0	1.635E-06	3.828E+08	5.657E+07	1.136E-16	4.053E-19	1.618E-30
7686.0	1.649E-06	3.774E+08	5.575E+07	1.119E-16	3.994E-19	1.594E-30
7736.0	1.663E-06	3.722E+08	5.498E+07	1.104E-16	3.939E-19	1.572E-30
7786.0	1.677E-06	3.674E+08	5.425E+07	1.089E-16	3.887E-19	1.550E-30
7836.0	1.692E-06	3.628E+08	5.356E+07	1.075E-16	3.837E-19	1.530E-30
7886.0	1.706E-06	3.584E+08	5.290E+07	1.062E-16	3.790E-19	1.511E-30
7936.0	1.720E-06	3.542E+08	5.227E+07	1.049E-16	3.745E-19	1.493E-30
7986.0	1.734E-06	3.502E+08	5.167E+07	1.037E-16	3.702E-19	1.475E-30
8036.0	1.748E-06	3.463E+08	5.110E+07	1.026E-16	3.661E-19	1.458E-30
8086.0	1.763E-06	3.426E+08	5.054E+07	1.015E-16	3.621E-19	1.442E-30

.....
time cwpwater rgenerate rdecay rconv rdiff_r0 rdiff_ebs
[yr] [kg] [kg/yr] [kg/yr] [kg/yr] [kg/yr] [kg/yr]
.....

8136.0	1.777E-06	3.390E+08	5.001E+07	1.004E-16	3.583E-19	1.427E-30
8186.0	1.791E-06	3.355E+08	4.949E+07	9.935E-17	3.545E-19	1.412E-30
8236.0	1.805E-06	3.322E+08	4.898E+07	9.834E-17	3.509E-19	1.397E-30
8286.0	1.819E-06	3.289E+08	4.849E+07	9.736E-17	3.474E-19	1.383E-30
8336.0	1.834E-06	3.256E+08	4.802E+07	9.640E-17	3.440E-19	1.369E-30
8386.0	1.848E-06	3.225E+08	4.755E+07	9.547E-17	3.407E-19	1.355E-30
8436.0	1.862E-06	3.194E+08	4.710E+07	9.455E-17	3.374E-19	1.342E-30
8486.0	1.876E-06	3.164E+08	4.665E+07	9.366E-17	3.342E-19	1.329E-30
8536.0	1.891E-06	3.135E+08	4.622E+07	9.279E-17	3.311E-19	1.316E-30
8586.0	1.905E-06	3.106E+08	4.579E+07	9.193E-17	3.281E-19	1.304E-30
8636.0	1.919E-06	3.078E+08	4.537E+07	9.108E-17	3.250E-19	1.291E-30
8686.0	1.933E-06	3.050E+08	4.496E+07	9.026E-17	3.221E-19	1.279E-30
8736.0	1.947E-06	3.022E+08	4.455E+07	8.944E-17	3.192E-19	1.268E-30
8786.0	1.962E-06	2.995E+08	4.415E+07	8.864E-17	3.163E-19	1.256E-30
8836.0	1.976E-06	2.969E+08	4.376E+07	8.785E-17	3.135E-19	1.244E-30
8886.0	1.990E-06	2.942E+08	4.337E+07	8.707E-17	3.107E-19	1.233E-30
8936.0	2.004E-06	2.916E+08	4.298E+07	8.630E-17	3.080E-19	1.222E-30
8986.0	2.018E-06	2.891E+08	4.261E+07	8.554E-17	3.052E-19	1.211E-30
9036.0	2.033E-06	2.865E+08	4.223E+07	8.479E-17	3.026E-19	1.200E-30
9086.0	2.047E-06	2.840E+08	4.186E+07	8.404E-17	2.999E-19	1.189E-30
9136.0	2.061E-06	2.816E+08	4.150E+07	8.331E-17	2.973E-19	1.179E-30
9186.0	2.075E-06	2.791E+08	4.114E+07	8.259E-17	2.947E-19	1.168E-30

.....
time cwpwater rgenerate rdecay rconv rdiff_r0 rdiff_ebs
[yr] [kg] [kg/yr] [kg/yr] [kg/yr] [kg/yr] [kg/yr]
.....

9236.0	2.089E-06	2.767E+08	4.078E+07	8.187E-17	2.922E-19	1.158E-30
9286.0	2.104E-06	2.743E+08	4.043E+07	8.116E-17	2.896E-19	1.147E-30
9336.0	2.118E-06	2.719E+08	4.008E+07	8.046E-17	2.871E-19	1.137E-30
9386.0	2.132E-06	2.696E+08	3.973E+07	7.976E-17	2.846E-19	1.127E-30
9436.0	2.146E-06	2.673E+08	3.939E+07	7.908E-17	2.822E-19	1.117E-30
9486.0	2.160E-06	2.650E+08	3.905E+07	7.840E-17	2.798E-19	1.107E-30
9536.0	2.174E-06	2.627E+08	3.871E+07	7.772E-17	2.774E-19	1.097E-30
9586.0	2.189E-06	2.604E+08	3.838E+07	7.705E-17	2.750E-19	1.088E-30
9636.0	2.203E-06	2.582E+08	3.805E+07	7.639E-17	2.726E-19	1.078E-30
9686.0	2.217E-06	2.560E+08	3.773E+07	7.574E-17	2.703E-19	1.068E-30
9736.0	2.231E-06	2.538E+08	3.740E+07	7.509E-17	2.680E-19	1.059E-30
9786.0	2.245E-06	2.516E+08	3.708E+07	7.445E-17	2.657E-19	1.050E-30
9836.0	2.259E-06	2.495E+08	3.676E+07	7.381E-17	2.634E-19	1.040E-30
9886.0	2.274E-06	2.473E+08	3.645E+07	7.318E-17	2.611E-19	1.031E-30
9936.0	2.288E-06	2.452E+08	3.614E+07	7.255E-17	2.589E-19	1.022E-30

9986.0 2.302E-06 2.431E+08 3.58E+07 7.193E-17 2.567E-19 1.013E-30
 10000.0 2.306E-06 2.423E+08 3.578E+07 7.184E-17 2.564E-19 1.908E-31

157/195

=====

cumulative release [ci/cell] by 10000.0 years

namall	halflife(yr)	xnoloss(ci)	amwp(ci)	xmass(ci)	fracleft
CM246	0.473E+04	0.195E+02	0.587E-06	0.197E+02	-0.01
PU242	0.386E+06	0.514E+04	0.636E-02	0.519E+04	-0.01
U238	0.447E+10	0.104E+04	0.113E-03	0.105E+04	-0.01
CM245	0.850E+04	0.182E+03	0.291E-04	0.184E+03	-0.01
AM241	0.432E+03	0.193E+03	0.259E-06	0.195E+03	-0.01
NP237	0.214E+07	0.202E+04	0.240E-02	0.204E+04	-0.01
AM243	0.738E+04	0.198E+05	0.171E-02	0.200E+05	-0.01
PU239	0.241E+05	0.763E+06	0.470E+00	0.771E+06	-0.01
PU240	0.654E+04	0.575E+06	0.491E-01	0.581E+06	-0.01
U236	0.234E+08	0.109E+04	0.102E-03	0.110E+04	-0.01
U234	0.245E+06	0.600E+04	0.608E-03	0.606E+04	-0.01
TH230	0.770E+05	0.524E+03	0.201E-03	0.529E+03	-0.01
RA226	0.160E+04	0.409E+03	0.485E-06	0.413E+03	-0.01
PB210	0.223E+02	0.407E+03	0.716E-07	0.411E+03	-0.01
CS135	0.230E+07	0.114E+04	0.109E-02	0.115E+04	-0.01
I129	0.157E+08	0.969E+02	0.653E-05	0.979E+02	-0.01
TC99	0.213E+06	0.389E+05	0.241E-02	0.393E+05	-0.01
NI59	0.800E+05	0.107E+05	0.931E-03	0.108E+05	-0.01
C14	0.573E+04	0.150E+04	0.453E-05	0.151E+04	-0.01
SE79	0.650E+05	0.112E+04	0.574E-04	0.113E+04	-0.01
NB94	0.203E+05	0.185E+04	0.316E-03	0.187E+04	-0.01

=====

zeroth yr inventory for liquid releases: 10040.5
 zeroth yr inventory for gaseous releases: 8107.1
 1000-yr c14 inv. for liq. rel [ci/cell]: 10036.7
 1000-yr c14 inv. for gaseous rel. [ci/cell]: 8104.1

output files: release.out, ebsnef.dat, relcum.out, relfrac.out
 maxrel.dat, ebs14.dat, maxc14.dat, ratec14.out, inv1000.out

158 } 195

pcl5

JOB 1775

ebspac.nuc

For: scratchy1!mohanty
Date: Mon Jan 6 17:53:55 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:26:03 1991
Started: Wed Nov 13 10:26:03 1991

159/195

15
 CM 2 .24E-6 100.0
 1 1
 2 1
 PU 3 .01E-3 100.0
 1 2
 3 2
 4 1
 U 3 0.95E-3 2.0
 1 3
 4 2
 5 1
 AM 2 .24E-6 200.0
 2 2
 3 1
 NP 1 .237E-3 50.0
 2 3
 TH 1 2.3E-10 100.0
 5 2
 RA 1 6.8E-8 30.0
 5 3
 PB 1 1. 20.0
 5 4
 CS 1 1. 200.0
 6 1
 I 1 1. 1.0
 7 1
 TC 1 1. 1.0
 8 1
 NI 1 0.59E-3 2.0
 9 1
 C 1 1. 1.0
 10 1
 SE 1 7.9E-11 1.0
 11 1
 NB 1 1. 20.0
 12 1

12 3 3 2 2 4 1 1 1 1 1 1 1

246.0	CM246	4.73E03	0.07224	0.
242.0	PU242	3.86E05	4.47857	0.
238.0	U238	4.47E09	0.89038	0.
245.0	CM245	8.50E03	0.35322	0.
241.0	AM241	4.32E02	4590.53	0.
237.0	NP237	2.14E06	0.8064	0.
243.0	AM243	7.38E03	43.4528	0.
239.0	PU239	2.41E04	862.391	0.
240.0	PU240	6.54E03	1422.21	0.
236.0	U236	2.34E07	0.67178	0.
234.0	U234	2.45E05	5.283	0.
230.0	TH230	7.70E04	3.61E-4	0.
226.0	RA226	1.60E03	1.0E-6	0.
210.0	PB210	2.23E01	1.3E-1	0.
135.0	CS135	2.30E06	0.9796	0.000
129.0	I129	1.57E07	0.083	0.000
99.0	TC99	2.13E05	34.38	0.000
59.0	NI59	8.00E04	9.97	0.
14.0	C14	5.73E03	4.30	0.000
79.0	SE79	6.50E04	1.0663	0.000
94.0	NB94	2.03E04	2.2259	0.

12345678901234567890123456789012345678901234567890

\ Pu238 inventory is put into U234 inventory.

160 / 195

161 / 195

pcl5

JOB 1777

cumc14.out

For: scratchy1!mohanty
Date: Mon Jan 6 17:57:24 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:29:33 1991
Started: Wed Nov 13 10:29:33 1991

time [yr] cumulative c14 release [ci

2.00000000000000-04	0.
50.420000000000	0.
100.500000000000	0.
152.400000000000	0.
206.000000000000	0.
259.400000000000	0.
312.800000000000	0.
366.200000000000	0.
419.700000000000	0.
473.100000000000	0.
533.800000000000	0.
585.700000000000	0.
637.500000000000	0.
689.300000000000	0.
741.100000000000	0.
793.000000000000	0.
844.800000000000	0.
896.600000000000	0.
948.400000000000	0.
1000.000000000000	0.
1065.000000000000	0.
1129.000000000000	0.
1194.000000000000	0.
1258.000000000000	0.
1323.000000000000	0.
1388.000000000000	0.
1452.000000000000	0.
1517.000000000000	0.
1581.000000000000	0.
1646.000000000000	0.
1710.000000000000	0.
1775.000000000000	0.
1839.000000000000	0.
1904.000000000000	0.
1986.000000000000	0.
2036.000000000000	0.
2086.000000000000	0.
2136.000000000000	0.
2186.000000000000	0.
2236.000000000000	0.
2286.000000000000	0.
2336.000000000000	0.
2386.000000000000	0.
2436.000000000000	0.
2486.000000000000	0.
2536.000000000000	0.
2586.000000000000	0.
2636.000000000000	0.
2686.000000000000	1598.7004081295
2736.000000000000	2695.9752127578
2786.000000000000	3496.6721938441
2836.000000000000	4060.0139550623
2886.000000000000	4433.3760243497
2936.000000000000	4662.7913577320
2986.000000000000	4786.9828967300
3036.000000000000	4840.5272322924
3086.000000000000	4854.1798480462
3136.000000000000	4854.1798480462
3186.000000000000	4854.1798480462

162 / 195

164/195

6236.000000000	4854.1798480462
6286.000000000	4854.1798480462
6336.000000000	4854.1798480462
6386.000000000	4854.1798480462
6436.000000000	4854.1798480462
6486.000000000	4854.1798480462
6536.000000000	4854.1798480462
6586.000000000	4854.1798480462
6636.000000000	4854.1798480462
6686.000000000	4854.1798480462
6736.000000000	4854.1798480462
6786.000000000	4854.1798480462
6836.000000000	4854.1798480462
6886.000000000	4854.1798480462
6936.000000000	4854.1798480462
6986.000000000	4854.1798480462
7036.000000000	4854.1798480462
7086.000000000	4854.1798480462
7136.000000000	4854.1798480462
7186.000000000	4854.1798480462
7236.000000000	4854.1798480462
7286.000000000	4854.1798480462
7336.000000000	4854.1798480462
7386.000000000	4854.1798480462
7436.000000000	4854.1798480462
7486.000000000	4854.1798480462
7536.000000000	4854.1798480462
7586.000000000	4854.1798480462
7636.000000000	4854.1798480462
7686.000000000	4854.1798480462
7736.000000000	4854.1798480462
7786.000000000	4854.1798480462
7836.000000000	4854.1798480462
7886.000000000	4854.1798480462
7936.000000000	4854.1798480462
7986.000000000	4854.1798480462
8036.000000000	4854.1798480462
8086.000000000	4854.1798480462
8136.000000000	4854.1798480462
8186.000000000	4854.1798480462
8236.000000000	4854.1798480462
8286.000000000	4854.1798480462
8336.000000000	4854.1798480462
8386.000000000	4854.1798480462
8436.000000000	4854.1798480462
8486.000000000	4854.1798480462
8536.000000000	4854.1798480462
8586.000000000	4854.1798480462
8636.000000000	4854.1798480462
8686.000000000	4854.1798480462
8736.000000000	4854.1798480462
8786.000000000	4854.1798480462
8836.000000000	4854.1798480462
8886.000000000	4854.1798480462
8936.000000000	4854.1798480462
8986.000000000	4854.1798480462
9036.000000000	4854.1798480462
9086.000000000	4854.1798480462
9136.000000000	4854.1798480462
9186.000000000	4854.1798480462

9236.0000000000	4854.1798480462
9286.0000000000	4854.1798480462
9336.0000000000	4854.1798480462
9386.0000000000	4854.1798480462
9436.0000000000	4854.1798480462
9486.0000000000	4854.1798480462
9536.0000000000	4854.1798480462
9586.0000000000	4854.1798480462
9636.0000000000	4854.1798480462
9686.0000000000	4854.1798480462
9736.0000000000	4854.1798480462
9786.0000000000	4854.1798480462
9836.0000000000	4854.1798480462
9886.0000000000	4854.1798480462
9936.0000000000	4854.1798480462
9986.0000000000	4854.1798480462
10000.0000000000	4854.1798480462

165

164/195

pcl5

JOB 1778

maxrel.dat

For: scratchy1!mohanty
Date: Mon Jan 6 18:01:07 CST 1997
Submit queue: IF 1 / Ethernet / UHSW
Submitted: Wed Nov 13 10:33:16 1991
Started: Wed Nov 13 10:33:16 1991

title: maximum fractional release of c14
tmaxc14 [yr] rmaxc14 [fraction]
2686.000000000 1.7625492907432D-03

1167/195

maximum fractional release of all nuclides

name	timrel (year)	maxrel (fraction)
CM246	3750.0000000000	1.5783522754812D-03
PU242	3750.0000000000	2.7141401662353D-03
U238	3750.0000000000	2.7344900329338D-03
CM245	3750.0000000000	2.0132696472912D-03
AM241	3750.0000000000	6.8843540618551D-06
NP237	3750.0000000000	5.8255911073670D-03
AM243	3750.0000000000	1.9220819013092D-03
PU239	3750.0000000000	2.4650694328810D-03
PU240	3750.0000000000	1.8372085777538D-03
U236	3750.0000000000	3.2638820937253D-03
U234	3750.0000000000	2.7056665626686D-03
TH230	3800.0000000000	0.93875542526128
RA226	3800.0000000000	154.56445111361
PB210	3800.0000000000	2.0504217152778D-03
CS135	3750.0000000000	2.7291136188881D-03
I129	3750.0000000000	2.7355903705135D-03
TC99	3750.0000000000	2.7029032191084D-03
NI59	3750.0000000000	2.6471868520012D-03
C14	3750.0000000000	9.6233839563820D-04
SE79	3750.0000000000	2.6289370073924D-03
NB94	3750.0000000000	2.4044897868145D-03