Dokumen Pengembangan TRIAC
(TRIso *Analysis Code*)

LABORATORIUM KOMPUTASI

PUSAT TEKNOLOGI DAN KESELAMATAN REAKTOR NUKLIR

Disusun oleh:
Arya Adhyaksa Waskita

Supervisor:
Dr. Topan Setiadipura

31 Juli 2017

Daftar Isi

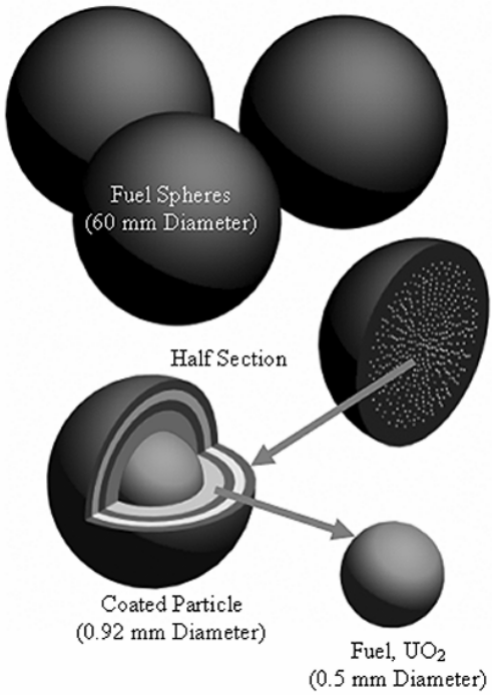
Daftar Gambar

Daftar Program

BAB 1

Pendahuluan

BATAN saat ini tengah berencana membangun reaktor riset baru berbasis HTGR (*High Temperature Gas-cooled Reactor*) [1] sebagai persiapan PLTN, yang akan dibangun di Indonesia di masa depan [2]. Salah satu yang perlu diperhatikan dalam pengembangan reaktor jenis ini adalah bahan bakarnya yang berjenis *pebble* yang bentuknya dapat diilustrasikan seperti pada Gambar 1.1. Bahan bakar harus dirancang sedemikian rupa sehingga rasio gagalnya bahan bakar selama operasi minimal.

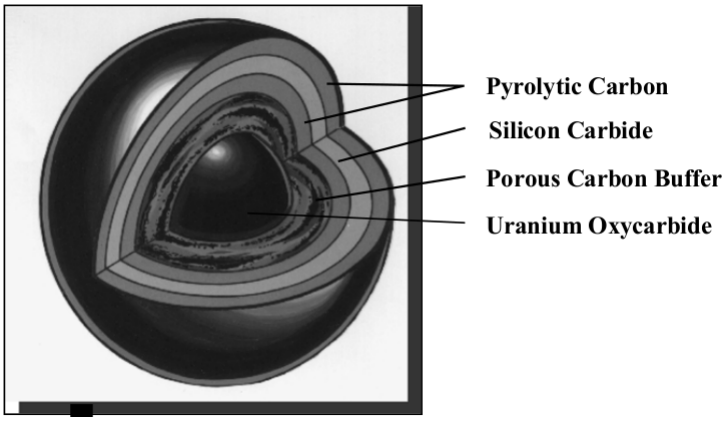


Gambar 1.1: Ilustrasi bentuk bahan bakar *pebble* [1]

Bahan bakar berjenis *pebble* ini memiliki komponen utama yang dalam Gambar 1.1 disebut sebagai *coated particle*. Komposisi elemen pelapis (*coated*) dapat diilustrasikan dalam Gambar 1.2. Dalam upaya menguasai teknologi reaktor berjenis HTGR melalui pengembangan RDE, salah tugas yang harus dilaksanakan adalah penguasaan analisis kegagalan bahan bakarnya, khususnya ketika terjadi kecelakaan.

Beragam model analisis telah dikembangkan, salah satunya yang dikembangkan oleh Wang [1]. Selain itu, terdapat sebuah model sederhana yang dikembangkan oleh Verfondern dalam PANAMA [3]. Pada model tersebut, bahan bakar disebut gagal jika kekuatan lapisan

SiC (*Silicon Carbide*) lebih kecil daripada tekanan internal dari lapisan di bawahnya (perhatikan Gambar 1.2). Model inilah yang akan diterapkan dalam TRIAC (*TRIso Analysis Code*).



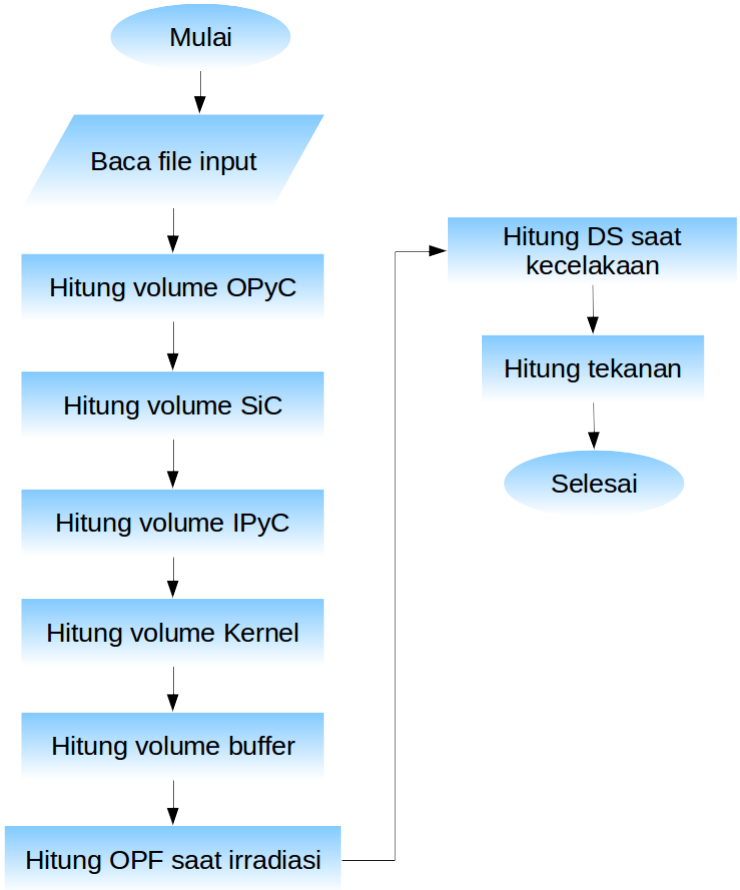
Gambar 1.2: Komposisi elemen pelapis partikel [1]

BAB 2

Alur Perhitungan

2.1 Pendahuluan

Secara umum, perhitungan TRIAC mengikuti diagram alir seperti pada Gambar 2.1 berikut. Sementara kode sumbernya disajikan dalam Listing 2.1 yang dibangun sepenuhnya berbasis pengetahuan yang diperoleh dari dokumen laporan teknis [4].



Gambar 2.1: Diagram alir perhitungan TRAIC

Listing 2.1: triac.py

```
import math,sys

def readdata(namafile):
        f=open(namafile,"r")
        statusGeometry="[m]"
        statusCharacteristics="SiC Tensile Strength [Pa]"
        statusIrradiation="INPUT: Irradiation Temp. Hystory"
        statusAccident="INPUT: Accident Temp. Hystory"
        statusAll=0
        dimensi=[]
        characteristics=[]
        irradiation=[]
        accident=[]
        i=0
        x=0
        for baris in f.readlines():
                i=i+1
                element=baris.split('\t')
                if statusAll==0:
                        if element[0]==statusGeometry:
                                for j in range(1,6):
                                        y=float(element[j])
                                        dimensi.append(y)
                                statusAll=1

                elif statusAll==1:
                        if element[0]==statusCharacteristics:
                                x=i+1
                        elif i==x:
                                try:
                                        for j in range(0,5):
                                                y=float(element[j])
                                                characteristics.append(y)
                                                statusAll=2
                                except:
                                        x=i+1

                elif statusAll==2:
                        if element[0]==statusIrradiation:
                                x=i+1
                        elif element[0]==statusAccident:
                                statusAll=3
                        else:
                                temp=[]
                                try:
                                        y=int(element[0])
                                        temp.append(y)
                                        y=float(element[1])
                                        temp.append(y)
                                        y=int(element[2])
                                        temp.append(y)
                                        irradiation.append(temp)
                                except:
                                        x=i+1

                elif statusAll==3:
                        temp=[]
                        try:
                                y=int(element[0])
                                temp.append(y)
                                y=float(element[1])
                                temp.append(y)
                                y=int(element[2])
                                temp.append(y)
                                accident.append(temp)
                        except:
                                x=i+1

        return dimensi, characteristics, irradiation, accident
```

```
70
71
72  def OPF(irradiation,dt):
73          x=len(irradiation)
74          print("Irradiation length:",x)
75          y=dt*24*3600
76          tb=1020*24*3600
77          z=0.0
78          for i in range(x):
79                  j=irradiation[i]
80                  g=2*(8.32e-11)*(math.exp(-163000/(8.3143*j[2])))*(tb-(j[1]*24*3600))*y
81                  z=z+g
82
83          Tb=0.85e4/((2*math.log10(tb))-(math.log10(z))-10.08)
84          logDS=-2.3-(8116/Tb)
85          ds=math.pow(10,logDS)
86          taui=ds*tb
87          return z,Tb,ds,taui
88
89  def FD(tauA,tauI):
90          summasi=tauA+tauI
91          looping=0.0
92          loopingPlus=0.0
93          loopingI=0.0
94          for n in range (1,2000):
95                  pangkat=math.pow(n,2)*math.pow(math.pi,2)*tauA
96                  A=math.exp(-(pangkat))
97                  B=math.pow(n,4)*math.pow(math.pi,4)
98                  looping=looping+((1-(A)/(B)))
99
100                 pangkatI=math.pow(n,2)*math.pow(math.pi,2)*tauI
101                 AI=math.exp(-(pangkatI))
102                 loopingI=loopingI*((1-(AI)/(B)))
103                 if tauA==0:
104                         """print(n,pangkatI,AI,B,AI/B,loopingI)"""
105                 loopingPlus=loopingPlus+(1-(math.exp(-(math.pow(n,2)*math.pow(math.pi,2)*summasi)))/(
106         if tauA==0:
107                 FTau=1-((6/tauI)*loopingI)
108                 print('tauI=',tauI)
109                 print('loopingI=',loopingI)
110                 print('ftauI=',FTau)
111                 Fd=FTau
112         else:
113                 FTau=1-((6/tauA)*looping)
114                 FTauPlus=1-((6/summasi)*loopingPlus)
115                 Fd=((summasi*FTauPlus)-(tauA*FTau))/tauI
116         return Fd
117
118 def OPFAccident(Tb,T):
119         tb=1020*24*3600
120         logOPF=-10.08-(8500/Tb)+(2*math.log10(tb))-(0.404*((10000/T)-(10000/(Tb+75))))
121         return math.pow(10,logOPF)
122
123 def Pressure(Tb,dsAccident,Vk,Vf,accident):
124         Ff=0.31
125         R=8.3143
126         Vm=2.43796e-5
127         Fb=0.08
128         p=[]
129         for i in range(len(accident)):
130                 x=accident[i]
131                 y=dsAccident[i]
132                 p.append((y[2]*Ff*OPFAccident(Tb,x[2])*Fb)/((Vf/Vk)*R*x[2]/Vm))
133         return p
134
135 def DS(accident,tauI):
136         x=len(accident)
137         ds=[]
138         for i in range(x):
139                 j=accident[i]
```

```
                    logDS=-2.3-(8116/j[2])
                    k=[]
                    l=math.pow(10,logDS)
                    k.append(l)
                    tauA=l*j[1]*24*3600
                    k.append(tauA)
                    Fd=FD(tauA,tauI)
                    k.append(Fd)
                    ds.append(k)
            return ds

if __name__=="__main__":
            if len(sys.argv)==3:
                    f=sys.argv[1]
                    dt=int(sys.argv[2])
            else:
                    f="example.pan.in"
                    dt=17

            x=readdata(f)
            dimensi=x[0]
            characteristics=x[1]
            irradiation=x[2]
            accident=x[3]
            """print("Dimensi:",dimensi)
            print("Karakteristik:",characteristics)
            print("Irradiasi:",irradiation)
            print("Accident:",accident)"""
            """Volume Outer Pyrolitic Carbon"""
            VolRef1=(4/3)*math.pi*dimensi[0]*dimensi[0]*dimensi[0]
            VolRef2=(4/3)*math.pi*dimensi[1]*dimensi[1]*dimensi[1]
            VolOPyC=VolRef1-VolRef2

            """Volume SiC"""
            VolRef3=(4/3)*math.pi*dimensi[2]*dimensi[2]*dimensi[2]
            VolSiC=VolRef2-VolRef3

            """Volume IPyC"""
            VolRef4=(4/3)*math.pi*dimensi[3]*dimensi[3]*dimensi[3]
            VolIPyC=VolRef3-VolRef4

            """Volume Buffer & Volume Kernel"""
            VolKernel=(4/3)*math.pi*dimensi[4]*dimensi[4]*dimensi[4]
            VolBuff=VolRef4-VolKernel
            print("Volume OPyC:",VolOPyC)
            print("Volume SiC:",VolSiC)
            print("Volume IPyC:",VolIPyC)
            print("Volume Buffer:",VolBuff)
            print("Volume Kernel:",VolKernel)
            TB=OPF(irradiation,dt)
            z=TB[0]
            Tb=TB[1]
            ds=TB[2]
            taui=TB[3]
            print("OPF=",z,"Tb=",Tb,"DS=",ds,"TauI=",taui)
            dsAccident=DS(accident,taui)
            for i in range(len(dsAccident)):
                    x=dsAccident[i]
                    print(x[0],x[1],x[2])

            p=Pressure(Tb,dsAccident,VolKernel,VolBuff/2,accident)
            print(p)
```

2.2 Membaca *file input*

Sub rutin ini ditujukan untuk membaca file input dengan format seperti terdapat pada Lampiran 1. Sub rutin ini menggunakan skema yang kaku karena identifikasi nilai-nilai yang

akan dibaca ditentukan oleh suatu teks tertentu. Setelah teks yang menjadi penanda, nilai-nilai yang dibutuhkan dibaca. Tetapi, nilai tersebut dapat langsung berada dalam satu baris bersama dengan teks penanda, atau berada pada baris yang berbeda. Sub rutin ini terdapat pada baris ke-3 s/d baris ke-69 dalam Listing 2.1

Terdapat empat jenis data yang perlu dibaca dari *file input* dalam Lampiran 1, masing-masing adalah sebagai berikut.

1. Data tentang geometri *pebble*. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusGeometry` (baris ke-5 pada Listing 2.1). Di dalam data geometri, terdapat empat data berbeda, masing-masing secara berurutan adalah panjang jejari *pebble* terluar, OPyC (*Outer Pyrolitic Carbon*), SiC (*Silicon Carbide*), IPyC (*Inner Pyrolitic Carbon*), *buffer* dan kernel. Data geometri akan digunakan untuk menghitung volume setiap elemen pelapis (Gambar 1.2). Yang perlu diperhatikan adalah data jari-jari yang disajikan adalah jarak dari pusat bahan bakar sampai titik terluar dari setiap lapisan. Karena itu, volume suatu lapisan harus mempertimbangkan lapisan-lapisan di dalamnya. Data geometri disimpan dalam variabel diberi nama `dimensi` dan dalam bentuk `list` (baris ke-10 dalam Listing 2.1).

2. Data tentang kekuatan SiC. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusCharacteristics` (baris ke-6 pada Listing 2.1). Ada empat nilai yang perlu dibaca terkait kekuatan SiC, masing-masing adalah SiC *Tensile Strength* [Pa], *Weibull Modulus Burnup* [FIMA], *Fission Yield of stable fission gasses* [Ff], *Fast Neutron Fluence* dan rasio berat Th terhadap U-235 pada kernel. Data terkait kekuatan SiC disimpan dalam variabel yang diberi nama `characteristics` dalam bentuk `list` (baris ke-11 dalam Listing 2.1).

3. Data tentang sejarah irradiasi. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusIrradiation` (baris ke-7 pada Listing 2.1). Data ini merupakan data temperatur bahan bakar *pebble* pada selang waktu tertentu. Sebagai contoh, data yang disajikan pada Lampiran 1 diambil pada selang waktu 17 hari. Data sejarah irradiasi disimpan dalam variabel yang diberi nama `irradiation` dalam bentuk `list`. Setiap elemen adalah `list` yang secara *nested* terdiri dari tiga elemen yang mewakili data tiap kolom pada setiap akuisisi (baris ke-12 pada Listing 2.1). Ilustrasinya adalah seperti $[[1,0,593],[2,17,833],\ldots]$

4. Data tentang sejarah kecelakaan yang dalam hal ini adalah kondisi di mana temperatur bahan bakar lebih besar daripada 2000 °C. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusAccident` (baris ke-8 pada Listing 2.1). Data ini memiliki pola yang sama dengan data sejarah irradiasi. Data sejarah keselakaan disimpan dengan cara yang sama seperti data tentang sejarah irradiasi tetapi dengan nama `accident` (baris ke-13 pada Listing 2.1). Ilustrasinya adalah seperti $[[1,0,1033],[2,0.0271,1033],\ldots]$

.

2.3 Menghitung OPF saat irradiasi

OPF (*Oxygen Per Fission*) adalah jumlah atom oksigen yang terlepas selama fisi atom U^{235} atau Pu^{239}. Atom oksigen ini mempengaruhi terbentunya senyawa CO yang akan meningkatkan tekanan internal dalam bahan bakar. Pembentukan senyawa CO juga dipengaruhi oleh temperatur, waktu serta jenis partikel kernel.

Nilai OPF didekati oleh persamaan (2.1). Nilai n dalam persamaan (2.1) sama dengan banyaknya data sejarah irradiasi. Nilai Δ_i merupakan selisih waktu dari sejarah irradiasi yang dicatat. Nilainya akan berubah dengan berubahnya rentang pencatatan temperatur irradiasi. Jika dalam contoh kasus yang disajikan pada Lampiran 1, rentang waktu pencatatan temperatur irradiasi dilakukan setiap 17 hari, maka Δ_i adalah 17 hari atau $17x24x3600$ detik. t_B adalah waktu irradiasi total bahan bakar, sedangkan $\overline{t_i}$ waktu irradiasi ketika pencatatan dilakukan.

$$OPF \simeq \sum_{i=1}^{n} g(\overline{t_i}) \cdot (t_B - \overline{t_i}) \cdot \Delta t_i \tag{2.1}$$

Tetapi, nilai OPF juga didefinisikan seperti persamaan (2.2), dengan nilai $g(\overline{t_i})$ didefinisikan oleh persamaan (2.3). Nilai R pada persamaan (2.3) adalah konstanta gas sebesar $8.3143[\frac{J}{mole \cdot K}]$.

$$OPF = \frac{g(T)}{2} \cdot t^2 \tag{2.2}$$

$$\frac{g(T)}{2} = 8.32 \cdot 10^{-11} \cdot e^{\frac{-163000}{R \cdot T}} \tag{2.3}$$

Nilai OPF selanjutnya digunakan untuk menghitung nilai temperatur irradiasi (T_B) dari persamaan (2.4). Formula empiris tersebut sesuai untuk jenis bahan bakar UO_2.

$$\log OPF = -10.08 - \frac{0.85 \cdot 10^4}{T_B} + 2 \cdot \log t_B \tag{2.4}$$

Sedangkan nilai T_B akan digunakan untuk menghitung DS, faktor berkurangnya koefisien difusi (s^{-1}) dari gas hasil fisi di dalam partikel kernel. Nilainya untuk bahan bakar UO_2 memenuhi persamaan (2.5).

$$\log DS = -2.30 - \frac{0.8116 \cdot 10^4}{T_B}. \tag{2.5}$$

Terakhir, DS akan digunakan untuk menghitung sebuah nilai tak berdimensi τ_i yang memenuhi persamaan (2.6).

$$\tau_i = DS(T_B) \cdot t_B \tag{2.6}$$

2.4 Menghitung DS saat kecelakaan

Seperti telah dijelaskan dalam sub bab 2.3, DS adalah faktor berkurangnya koefisien difusi gas hasil fisi dalam partikel kernel. Sekarang, faktor ini dihitung ketika kondisi kecelakaan terjadi. Kita memerlukan sejarah temperatur bahan bakar setelah kecelakaan terjadi serta τ_i. yang telah dihitung di persamaan (2.6).

Dengan menggunakan persamaan (2.6), kita dapat menghitung nilai DS dengan temperatur kecelakaan yang tercatat. Kemudian, kita perlu menghitung nilai τ_A dengan persamaan (2.6) tetapi dengan nilai temperatur dan waktu setelah terjadi kecelakaan. Selanjutnya, dengan modal nilai τ_i dan τ_A kita akan menghitung nilai Fd, yang merupakan faktor fisi gas Xe dan Kr (yang dominan). Nilai Fd dihitung dengan persamaan (2.7).

$$Fd = \frac{(\tau_i + \tau_A) \cdot f(\tau_i + \tau_A) - \tau_A \cdot f(\tau_A)}{\tau_i} \tag{2.7}$$

Sedangkan nilai $f(\tau)$ dihitung menggunakan persaamaan (2.8). Batas atas nilai n pada persaamaan (2.8) dapat menggunakan nilai yang cukup besar, misalnya 1000, atau ketika

dua nilai berdekatan yang dihasilkan hanya berselisih kurang dari 10^{-20}. Idealnya, suku penjumlahan sebanyak n akan semakin baik jika hasilnya mendekati 1.

$$f(\tau) = 1 - \frac{6}{\tau} \cdot \sum_{n=1}^{\infty} \left(1 - \frac{e^{-n^2 \cdot \pi^2 \cdot \tau}}{n^4 \cdot \pi^4} \right) \tag{2.8}$$

Di level implementasi, perhitungan *DS* saat kecelakaan didistribusi ke dalam beberapa fungsi seperti terlihat pada Listing 2.1. Masing-masing fungsi tersebut adalah `DS(accident,tauI)` dan `FD(tauA,tauI)`.

2.5 Menghitung tekanan

Tekanan adalah variabel yang penting dalam tahapan analisis ini karena akan menentukan fraksi gagal bahan bakar. Untuk menghitung tekanan yang timbul ketika kecelakaan terjadi pada waktu tertentu, sehingga menyebabkan panas tertentu, digunakan persamaan (2.9).

$$p = \frac{(F_d \cdot F_f + OPF) \cdot F_b}{\left(\frac{V_f}{V_k}\right) \cdot R \cdot \frac{T}{V_m}} \tag{2.9}$$

dengan :

F_d = fraksi relatif gas fisi yang lepas

F_f = produk fisi yang dihasilkan dari gas fisi stabil, F_f=0.31

OPF = jumlah atom oksigen setiap fisi

F_b = *burnup* logam berat (FIMA)

V_f = fraksi void [m^3], terkait dengan 50% volume buffer

V_k = volume kernel [m^3]

V_m = volume molar dalam partiekl kernel $\left[\frac{m^3}{mole}\right]$

R = konstanta gas, 8.3143 $\left[\frac{J}{(mole \cdot K)}\right]$

Daftar Referensi

[1] J. Wang, "An integrated performance model for high temperature gas cooled reactor coated particle fuel," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.

[2] "Reaktor daya eksperimental (rde)," http://www.batan.go.id/index.php/id/reaktor-daya-eksperimental-rde, diakses: 17-07-2017.

[3] K. Verfondern, J. Cao, T. Liu, and H.-J. Allelein, "Conclusions from v&v studies on the german codes panama and fresco for htgr fuel performance and fission product release," *Nuclear Engineering and Design*, vol. 271, pp. 84 – 91, 2014, sI : HTR 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0029549313005992

[4] K. Verfondern and H. Nabielek, "The mathematical basis of the panama-i code for modelling pressure vessel failure of triso coated particles under accident conditions," Julich Research Center, Germany, Tech. Rep., 1990.

LAMPIRAN

Lampiran 1

TRIAC-BATAN
TRISO Analysis Code of BATAN
"Developed by Computational Laboratory, Center for Nuclear Reactor Technology and Safety, BATAN"

Case Title: (describe your problem case here)

TRISO Geometry:
Outer radius CFP SiC IPyC buffer kernel center

[m] 4.60E-04 4.20E-04 3.85E-04 3.45E-04 2.50E-04 0
Properties and Operation Parameters:

SiC Tensile Strength [Pa] Weibull Modulus Burnup [FIMA] "Fission Yield of stable fission gasses, Ff" Fast Neutron Fluence Weight ratio of th to U-235 in kernel

8.34E+08 8.02 0.09 0.31 2.4
Properties and Operation Parameters related with thermal decomposition:

Alpha Beta
0.0001 4
-1 1401.6 0.1 10
INPUT: Irradiation Temp. Hystory

1	0	593
2	17	833
3	34	1023
4	51	1093
5	68	1123
6	85	833
7	102	1023
8	119	1093
9	136	1123
10	153	833
11	170	1023
12	187	1093
13	204	1123
14	221	833
15	238	1023
16	255	1093
17	272	1123
18	289	833
19	306	1023
20	323	1093
21	340	1123
22	357	833
23	374	1023
24	391	1093
25	408	1123
26	425	833
27	442	1023
28	459	1093
29	476	1123
30	493	833
31	510	1023
32	527	1093
33	544	1123
34	561	833
35	578	1023
36	595	1093
37	612	1123

```
38    629   833
39    646   1023
40    663   1093
41    680   1123
42    697   833
43    714   1023
44    731   1093
45    748   1123
46    765   833
47    782   1023
48    799   1093
49    816   1123
50    833   833
51    850   1023
52    867   1093
53    884   1123
54    901   833
55    918   1023
56    935   1093
57    952   1123
58    969   833
59    986   1023
60    1003  1093
61    1020  1123

      0
      -1    180   1
INPUT: Accident Temp. Hystory

1     0     1033
2     0.0271      1033
3     0.2208      1068
4     1     1160
5     10    1571
6     20    1728
7     30    1752
8     35    1749
9     60    1690
10    90    1605
11    120   1526
12    180   1395

      0
```