



---

## **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

<b>Daftar Gambar</b>	<b>ii</b>
<b>Daftar Program</b>	<b>iii</b>
<b>1 Pendahuluan</b>	<b>2</b>
<b>2 Alur Perhitungan</b>	<b>4</b>
2.1 Pendahuluan . . . . .	4
2.2 Membaca <i>file input</i> . . . . .	8
2.3 Menghitung OPF saat irradiasi . . . . .	9
2.4 Menghitung DS saat kecelakaan . . . . .	10
2.5 Menghitung tekanan . . . . .	10
2.6 Fraksi gagal bahan bakar . . . . .	11
2.6.1 Fraksi gagal akibat berkurangnya <i>tensile strength</i> . . . . .	11
2.6.2 Fraksi gagal bahan bakar akibat <i>weight loss</i> . . . . .	13
<b>LAMPIRAN</b>	<b>1</b>
<b>Lampiran 1</b>	<b>2</b>

# Daftar Gambar

1.1	Ilustrasi bentuk bahan bakar <i>pebble</i> . . . . .	2
1.2	Komposisi elemen pelapis partikel . . . . .	3
2.1	Diagram alir perhitungan TRAIC . . . . .	4

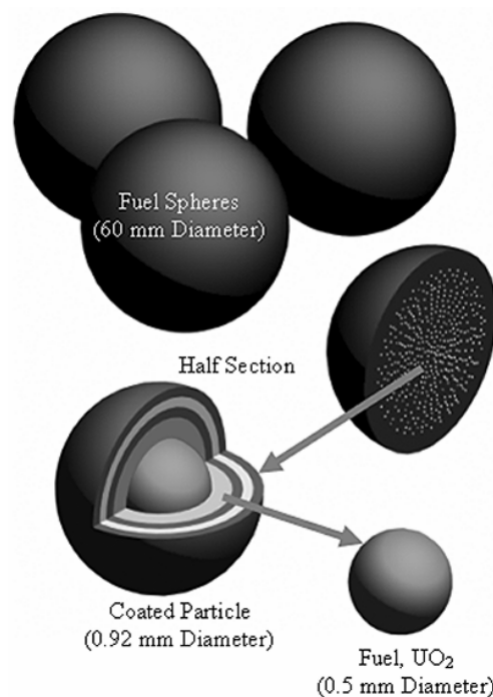
# Daftar Program

2.1	triac.py . . . . .	5
-----	--------------------	---

# 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.

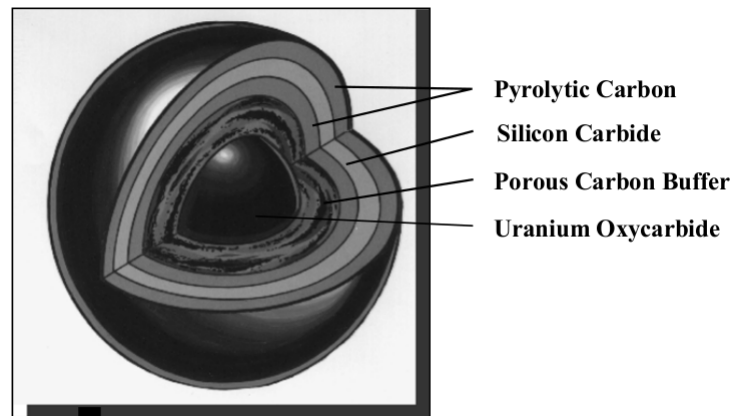


Gambar 1.1: Ilustrasi bentuk bahan bakar *pebble* [1]

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*).



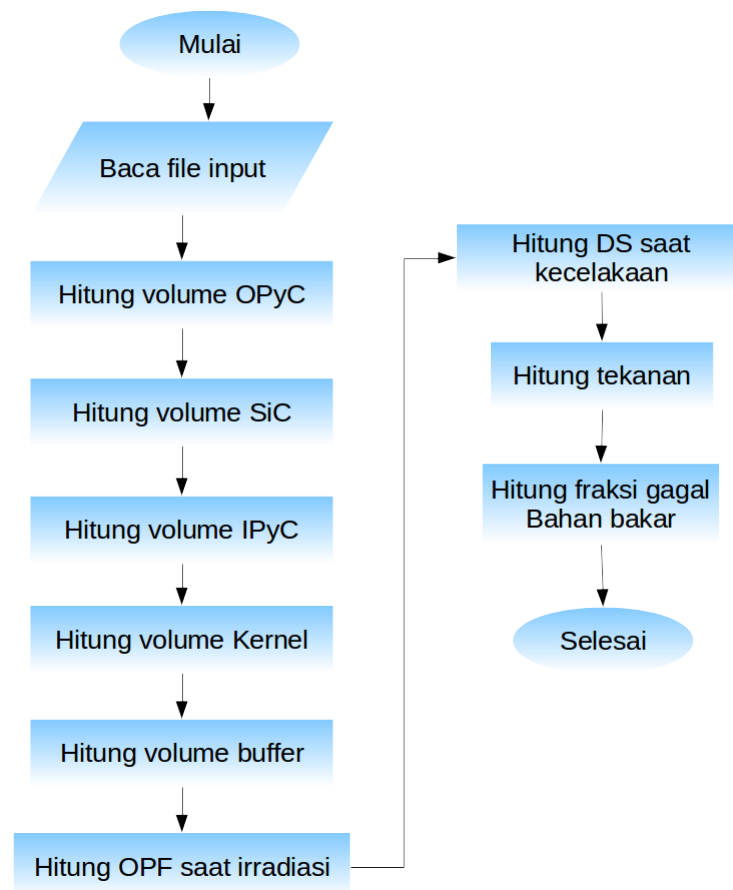
Gambar 1.2: Komposisi elemen pelapis partikel [1]

## 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 TRIAC

Listing 2.1: triac.py

```

1  import math, sys
2
3  def readdata(namafile):
4      f=open(namafile, "r")
5      statusGeometry=" [m]"
6      statusCharacteristics="SiC Tensile Strength [Pa]"
7      statusIrradiation="INPUT: Irradiation Temp. Hystory"
8      statusAccident="INPUT: Accident Temp. Hystory"
9      statusAll=0
10     dimensi=[]
11     characteristics=[]
12     irradiation=[]
13     accident=[]
14     i=0
15     x=0
16     for baris in f.readlines():
17         i=i+1
18         element=baris.split('\t')
19         if statusAll==0:
20             if element[0]==statusGeometry:
21                 for j in range(1,6):
22                     y=float(element[j])
23                     dimensi.append(y)
24                     statusAll=1
25
26             elif statusAll==1:
27                 if element[0]==statusCharacteristics:
28                     x=i+1
29                 elif i==x:
30                     try:
31                         for j in range(0,5):
32                             y=float(element[j])
33                             characteristics.append(y)
34                             statusAll=2
35                     except:
36                         x=i+1
37
38             elif statusAll==2:
39                 if element[0]==statusIrradiation:
40                     x=i+1
41                 elif element[0]==statusAccident:
42                     statusAll=3
43             else:
44                 temp=[]
45                 try:
46                     y=float(element[1])*24*3600
47                     temp.append(y)
48                     y=float(element[2])
49                     temp.append(y)
50                     irradiation.append(temp)
51                 except:
52                     x=i+1
53
54             elif statusAll==3:
55                 temp=[]
56                 try:
57                     y=float(element[1])*24*3600
58                     temp.append(y)
59                     y=float(element[2])
60                     temp.append(y)
61                     accident.append(temp)
62                 except:
63                     x=i+1
64
65     f.close()
66     return dimensi, characteristics, irradiation, accident
67
68
69 def OPF(irradiation, dt):

```



```

70     x=len(irradiation)
71     print("Irradiation length:",x)
72     y=dt*24*3600
73     tb=1020*24*3600
74     z=0.0
75     fo=file('gti','w')
76     fo.write('i, y, t, ti, Ti, t-ti, a1, a, b, g, g1, z \n')
77     for i in range(x):
78         j=irradiation[i]
79         a1=8.3143*j[1]
80         a=-163000/(a1)
81         b=math.exp(a)
82         g=2*(8.32e-11)*b
83         g1=g*(tb-j[0])*y
84         z=z+g1
85         fo.write(str(i)+' ', ' '+str(y)+' ', ' '+str(tb)+' ', ' '+str(j[0])+' ', ' '+str(j[1])+' ', ' '+str(tb-j[0])+' ')
86     fo.close()
87
88     Tb=0.85e4/((2*math.log10(tb))-(math.log10(z))-10.08)
89     logDS=-2.3-(8116/Tb)
90     ds=math.pow(10,logDS)
91     tau=ds*tb
92     return z,Tb,ds,tau
93
94 def FTau(tau):
95     looping=0.0
96     for n in range(1,2000):
97         pangkat=math.pow(n,2)*math.pow(math.pi,2)*tau
98         A=math.exp(-(pangkat))
99         B=math.pow(n,4)*math.pow(math.pi,4)
100        looping=looping+((1-A)/B)
101
102    ftau=1-((6/tau)*looping)
103    return ftau
104
105 def OPFAccident(Tb,tb,T):
106     logOPF=-10.08-(8500/Tb)+(2*math.log10(tb))-(0.404*((10000/T)-(10000/(Tb+75))))
107     opfa=math.pow(10,logOPF)
108     return opfa
109
110 def Pressure(Tb,dsAccident,Vk,Vf,accident):
111     p=[]
112     for i in range(len(accident)):
113         x=accident[i]
114         y=dsAccident[i]
115         p.append((y[2]*Ff*OPFAccident(Tb,x[1]*Fb)/((Vf/Vk)*R*x[1]/Vm))
116     return p
117
118 def DS(T):
119     logDS=-2.3-(8116/T)
120     ds=math.pow(10,logDS)
121     return ds
122
123
124 if __name__=="__main__":
125     if len(sys.argv)==3:
126         f=sys.argv[1]
127         dt=int(sys.argv[2])
128     else:
129         f="example.pan.in"
130         dt=17
131
132     x=readdata(f)
133     dimensi=x[0]
134     characteristics=x[1]
135     irradiation=x[2]
136     accident=x[3]
137
138     VolRef1=(4/3)*math.pi*dimensi[0]*dimensi[0]*dimensi[0]
139     VolRef2=(4/3)*math.pi*dimensi[1]*dimensi[1]*dimensi[1]

```

```

140 VolOPyC=VolRef1-VolRef2
141
142 """ Volume SiC """
143 VolRef3=(4/3)*math.pi*dimensi[2]*dimensi[2]*dimensi[2]
144 VolSiC=VolRef2-VolRef3
145
146 """ Volume IPyC """
147 VolRef4=(4/3)*math.pi*dimensi[3]*dimensi[3]*dimensi[3]
148 VolIPyC=VolRef3-VolRef4
149
150 """ Volume Buffer & Volume Kernel """
151 VolKernel=(4/3)*math.pi*dimensi[4]*dimensi[4]*dimensi[4]
152 VolBuff=VolRef4-VolKernel
153 print( "Volume OPyC: " ,VolOPyC)
154 print( "Volume SiC: " ,VolSiC)
155 print( "Volume IPyC: " ,VolIPyC)
156 print( "Volume Buffer: " ,VolBuff)
157 print( "Volume Kernel: " ,VolKernel)
158
159 TB=OPF( irradiation , dt )
160 z=TB[0]
161 Tb=TB[1]
162 dsi=TB[2]
163 tauI=TB[3]
164 print( "OPF=" , z , "Tb=" , Tb , "DS=" , dsi , "TauI=" , tauI )
165
166 tb=1020*24*3600
167 Vk=VolKernel
168 Vf=0.5*VolBuff
169 Ff=0.31
170 R=8.3143
171 Vm=2.43796e-5
172 Fb=0.08
173
174 a=(0.5*(pow( dimensi [ 1 ] ,3)+pow( dimensi [ 2 ] ,3)))
175 r=pow( a , ( 1.0/3 ))
176 do=dimensi[1]-dimensi[2]
177 """ fluence = 1.4
178 fluenceTemp = 0.556 + ((650)/Tb)
179 fluenceS = pow( 10 , fluenceTemp )
180 sigma0 = (196.e6)*(1-(fluence/fluenceS)) """
181 sigma0 = 756
182 """ fluenceM = 0.394 + (650/Tb)
183 m = 2*(1-(fluence/fluenceM))
184 print( 'fluence = '+str(fluence) + ', fluenceTemp = '+str(fluenceTemp) + ', fluenceS = '+str(fluenceS) +
185 m = 8
186 print( FTau(0.05) , FTau(0.5) , FTau(1) , FTau(2))
187 print( 'ln(2) = ' + str( math . log (2)))
188 pressure = []
189 SigmaT = []
190 phil = 0
191 for i in range( len ( accident )):
192     dsa=DS( accident [ i ][ 1 ])
193     tauA=dsa*accident [ i ][ 0 ]
194     """ print ( ' tauA = '+ str ( tauA ) ) """
195     if accident [ i ][ 0 ] == 0:
196         Fd=FTau( tauI )
197     else :
198         Fd=(( ( tauI + tauA ) * FTau ( tauI + tauA )) - ( tauA * FTau ( tauA ) ) ) / tauI
199
200     opfa=OPFAccident( Tb , tb , accident [ i ][ 1 ])
201     n=(( Fd * Ff ) + opfa ) * Fb * ( Vk / Vm )
202     """ p = ((( Fd * Ff ) + opfa ) * Fb ) / ( ( Vf / Vk ) * R * accident [ i ][ 1 ] / Vm ) """
203     p = n * R * accident [ i ][ 1 ] / Vf
204     pressure . append ( p )
205     """ print ( tauA , tauI , Fd , p ) """
206
207     vdot = ( 5.87e-7 ) * math . exp ( - 179500 / ( 8.3143 * accident [ i ][ 1 ] ) )
208     a = ( 1 + ( vdot * accident [ i ][ 0 ] ) / do )
209     sigmaT = ( ( r * p ) / ( 2 * do ) ) * a

```

```

210 SigmaT.append(sigmaT)
211 print(str(i)+', T='+str(accident[i][1])+', p='+str(p)+' , sigmaT='+str(sigmaT)+' , sigma0='+str(sigma0))
212
213 a1=sigmaT/sigma0
214 a=pow(a1,m)
215 b=math.exp(-math.log(2)*a)
216 phi=1-b
217 print('a1='+str(a1)+' , a='+str(a)+' , b='+str(b)+' , phi='+str(phi)+' , phi-phi1='+str(phi-phi1))
218 phi1=phi
219 """ print('a='+str(a)+' , b='+str(b)+' , phi='+str(phi)) """
220
221
222 """ eta=0.565*math.exp(-187400/(8.3143*accident[i][1]))
223 m=m0*(0.44+(0.56*math.exp(-eta*accident[i][0])))
224
225
226
227 temp=sigmaT/sigma0
228 print(sigmaT,temp,m,eta)
229 gagal=1-math.exp(-math.log1p(2)*pow((sigmaT/sigma0),m))
230 phil.append(gagal) """

```

## 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 iradiasi disimpan dalam variabel yang diberi nama `irradiation` dalam bentuk `list`. Setiap elemen adalah `list` yang secara *nested* terdiri dari dua elemen yang mewakili data kolom kedua dan ketiga tiap akuisisi (baris ke-12 pada Listing 2.1). Ilustrasinya adalah seperti `[[0, 593], [17, 833], ...]`. Data tentang nomor urut tidak digunakan karena selain tidak diperlukan dalam perhitungan, akan menyulitkan proses interpolasi yang akan diterapkan berikutnya.

4. Data tentang sejarah kecelakaan. 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 iradiasi. Data sejarah kecelakaan disimpan dengan cara yang sama seperti data tentang sejarah iradiasi tetapi dengan nama `accident` (baris ke-13 pada Listing 2.1). Ilustrasinya adalah seperti `[[0, 1033], [0.0271, 1033], ...]`

### 2.3 Menghitung OPF saat iradiasi

OPF (*Oxygen Per Fission*) adalah jumlah atom oksigen yang terlepas selama fisi atom  $U^{235}$  atau  $Pu^{239}$ . Atom oksigen ini mempengaruhi terbentuknya 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 iradiasi. Nilai  $\Delta_i$  merupakan selisih waktu dari sejarah iradiasi yang dicatat. Nilainya akan berubah dengan berubahnya rentang pencatatan temperatur iradiasi. Jika dalam contoh kasus yang disajikan pada Lampiran 1, rentang waktu pencatatan temperatur iradiasi dilakukan setiap 17 hari, maka  $\Delta_i$  adalah 17 hari atau  $17 \times 24 \times 3600$  detik.  $t_B$  adalah waktu iradiasi total bahan bakar, sedangkan  $\bar{t}_i$  waktu iradiasi ketika pencatatan dilakukan.

$$OPF \simeq \sum_{i=1}^n g(\bar{t}_i) \cdot (t_B - \bar{t}_i) \cdot \Delta t_i \quad (2.1)$$

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

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

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

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

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

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).

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

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

$$\tau_i = DS(T_B) \cdot t_B \quad (2.6)$$

## 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  $\tau_i$ , 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  $\tau_A$  dengan persamaan (2.6) tetapi dengan nilai temperatur dan waktu setelah terjadi kecelakaan. Selanjutnya, dengan modal nilai  $\tau_i$  dan  $\tau_A$  kita akan menghitung nilai  $Fd$ , yang merupakan faktor fisi gas Xe dan Kr (yang dominan). Nilai  $Fd$  dihitung dengan persamaan (2.7).

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

Sedangkan nilai  $f(\tau)$  dihitung menggunakan persamaan (2.8). Batas atas nilai  $n$  pada persamaan (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.

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

## 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) [1].

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

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 terjadi fisi saat terjadi kecelakaan

$F_b$  = 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 partikel kernel  $\left[ \frac{m^3}{mole} \right]$

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

Khusus untuk variabel  $OPF$ , karena perhitungan tekanan dilakukan ketika terjadi kecelakaan, digunakanlah persamaan (2.10). Persamaan (2.10) mirip dengan persamaan (2.4) dengan penambahan suku ke-3.

$$\log OPF = -10.08 - \frac{0.85 \cdot 10^4}{T_B} + 2 \cdot \log t_B - 0.04 \cdot \left( \frac{10^4}{T} + \frac{10^4}{T_B + 75} \right) \quad (2.10)$$

## 2.6 Fraksi gagal bahan bakar

Tahapan terakhir dari analisis ini adalah perhitungan fraksi gagal bahan bakar. Secara umum, fraksi gagal bahan bakar dipengaruhi sejumlah sebab. Dalam analisis yang dilakukan TRAIC (dan juga PANAMA sebagai acuannya), gagalnya bahan bakar dapat disebabkan oleh 3 sebab. Ketiganya adalah sebagai berikut.

1. Pabrikasi ( $\phi_0$ ). Dalam analisis ini, nilai  $\phi_0$  diasumsikan sama dengan 0.
2. Berkurangnya *tensile strength* lapisan SiC ( $\phi_1$ ). Hal ini dapat terjadi karena
  - proses iradiasi maupun
  - meningkatnya temperatur secara signifikan ketika terjadi kecelakaan) atau disebut juga *grain boundary*.
3. Dekomposisi termal pada temperatur tinggi yang menyebabkan terjadinya *weight loss* pada lapisan SiC ( $\phi_2$ ).

Ketiga sebab terjadinya kegagalan bahan bakar tersebut mengikuti persamaan (2.11).

$$\phi_{total} = 1 - (1 - \phi_1) - (1 - \phi_2) \quad (2.11)$$

### 2.6.1 Fraksi gagal akibat berkurangnya *tensile strength*

Nilai fraksi gagal bahan bakar pada waktu  $t$  setelah terjadinya kecelakaan diperoleh dengan persamaan (2.12).

$$\phi_1(t, T) = 1 - e^{-\ln 2 \cdot \left( \frac{\sigma_t}{\sigma_o} \right)^m} \quad (2.12)$$

dengan :

$\sigma_o$  = *tensile strength* dari SiC [Pa] pada akhir iradiasi

$\sigma_t$  = tekanan yang dialami SiC [Pa] akibat tekanan gas internal

Variabel tekanan internal pada SiC ( $\sigma_t$ ) dihitung dengan persamaan (2.13). Pada persamaan (2.13), jari-jari lapisan SiC merupakan rerata karena lapisan SiC memang memiliki ketebalan yang nilai awalnya diwakili oleh variabel  $d_o$ .

$$\sigma_t = \frac{r \cdot p}{2 \cdot d_o} \cdot \left( 1 + \frac{\dot{v} \cdot t}{d_o} \right) \quad (2.13)$$

dengan :

$r$ =rerata jari-jari SiC,  $(0.5 \cdot (r_a^3 + r_i^3))^{\frac{1}{3}}$  [m]

$d_o$ =ketebalan awal lapisan SiC,  $r_a - r_i$  [m]

$p$ =tekanan gas fisi dalam partikel [Pa]

$\dot{v}$ =laju korosi sebagai fungsi temperatur (T),  $[\frac{m}{s}]$

Sedangkan variabel laju korosi ( $\dot{v}$ ) dihitung dengan persamaan (2.14), mirip dengan persamaan (2.3) dengan perbedaan pada konstanta.

$$\dot{v} = 5.87 \cdot 10^{-7} \cdot e^{-\left(\frac{179500}{R \cdot T}\right)} \quad (2.14)$$

Selanjutnya, variabel *tensile strength* lapisan SiC, penurunan nilainya mengikuti persamaan (2.15). Variabel  $\sigma_{oo}$  merupakan *tensile strength* awal sebelum diiradiasi. Nilainya merupakan sesuatu yang dapat diukur. Sedangkan  $\Gamma$  dan  $\Gamma_s$  masing-masing merupakan *fluence* neutron cepat  $[10^{25} m^{-2} EDN]$  dan *fluence* yang dipengaruhi temperatur iradiasi. Nilai  $\Gamma_s$  ditentukan menggunakan persamaan (2.16). Nilai minimum  $\sigma_{oo}$  merupakan nilai awal *tensile strength* dan diasumsikan sama dengan 196 [MPa]. Tentunya, dengan perlakuan iradiasi yang sama, lapisan SiC dengan nilai awal *tensile strength* terkecil akan memiliki nilai akhir *tensile strength* yang juga kecil.

$$\sigma_o = \sigma_{oo} \cdot \left(1 - \frac{\Gamma}{\Gamma_s}\right) \quad (2.15)$$

$$\log \Gamma_s = 0.556 + \frac{0.065 \cdot 10^4}{T_B} \quad (2.16)$$

*Tensile strength* lapisan SiC yang dihitung menggunakan persamaan (2.15) merupakan nilai yang berlaku pada satu *coated particle*. Padahal, ada sangat banyak *coated particle* yang dioperasikan. Karena itu, diperlukan perhitungan yang mempertimbangkan variabel ini untuk semua distribusi *coated particles*. Dengan pendekatan yang sama seperti persamaan (2.15), persamaan (2.17) dibangun.

$$m_o = m_{oo} \cdot \left(1 - \frac{\Gamma}{\Gamma_m}\right) \quad (2.17)$$

dengan  $\log \Gamma_m = 0.394 + \frac{0.065 \cdot 10^4}{T_B}$  dan nilai  $m_{oo} = 2$  sebagai nilai terkecilnya. Nilai  $m_o$  pada persamaan (2.17) kemudian akan disubstitusi ke persamaan (2.12) sebagai  $m$ .

Selain korosi karena proses iradiasi, lapisan SiC juga dapat terkorosi karena *grain Boundary*. Jika korosi akibat iradiasi tergantung pada sejarah iradiasi yang dialami bahan bakar dan terjadi sebelum kecelakaan, maka korosi karena *grain Boundary* terjadi setelah kecelakaan. Penurunan nilai distribusi *tensile strength* akibat meningkatnya temperatur karena kecelakaan mengikuti persamaan (2.18).

$$m = m_o \cdot \left(0.44 + 0.56 \cdot e^{-\dot{\eta} \cdot t}\right) \quad (2.18)$$

di mana nilai  $\dot{\eta}$  mengikuti persamaan 2.19 dengan pola yang sama seperti persamaan (2.14).

$$\dot{\eta} = 0.565 \cdot e^{-\left(\frac{187400}{R \cdot T}\right)} [s^{-1}] \quad (2.19)$$

### 2.6.2 Fraksi gagal bahan bakar akibat *weight loss*

Laju *weight loss* yang terjadi akibat tingginya temperatur saat terjadi kecelakaan mengikuti persamaan (2.20).

$$k = k_o \cdot e^{\frac{-Q}{RT}} \quad (2.20)$$

dengan  $Q = 556 \left[ \frac{kJ}{mol} \right]$  dan  $k_o$  adalah faktor frekuensi yang tergantung pada jenis partikel.

Selanjutnya, diasumsikan bahwa partikel TRISO tergantung pada apa yang disebut sebagai "*action integral*", dan disimbolkan dengan  $\zeta$  yang nilainya mengikuti persamaan (2.21).

$$\zeta = \int_{t_1}^{t_2} k(T) dt \quad (2.21)$$

dengan  $K(T)$  adalah nilai yang menggambarkan sejarah kondisi partikel yang bergantung pada temperatur dan waktu.

Secara numerik, persamaan (2.21) dapat dituliskan sebagai persamaa (2.22).

$$\zeta(t_2) = \zeta(t_1) + k(T_m) \cdot (t_2 - t_1) \quad (2.22)$$

dengan  $k(T_m) = \frac{375}{d_o} \cdot e^{\left( \frac{-556000}{R \cdot T_m} \right)}$ .

Kemudian, fraksi gagal  $\phi_2$  sedemikian rupa sehingga nilainya  $\leq 1$ . Karena itu, variabel  $\phi_2$  selanjutnya didefinisikan sebagai persamaan (2.23).

$$\phi_2(t, T) = 1 - e^{-\alpha \cdot \zeta^\beta} \quad (2.23)$$

Nilai  $\alpha$  dan  $\beta$  kemudian ditentukan secara empiris. Dan berdasarkan penelitian empiris sebelumnya terhadap partikel  $UO_2$ , diperoleh nilai  $\alpha = \ln 2 = 0.693$ , sedangkan nilai  $\beta = 0.88$ .

Dalam TRIAC, faktor fraksi gagal ini tidak akan dipertimbangkan. Hal ini disebabkan karena kondisi ini terjadi pada temperatur di atas  $2000^\circ\text{C}$ . Sementara RDE tidak dirancang untuk sampai pada temperatur tersebut.



# 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