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

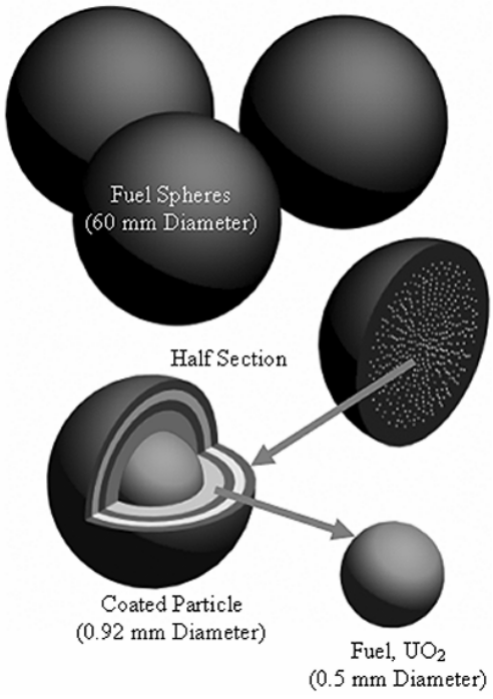
Daftar Gambar

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.

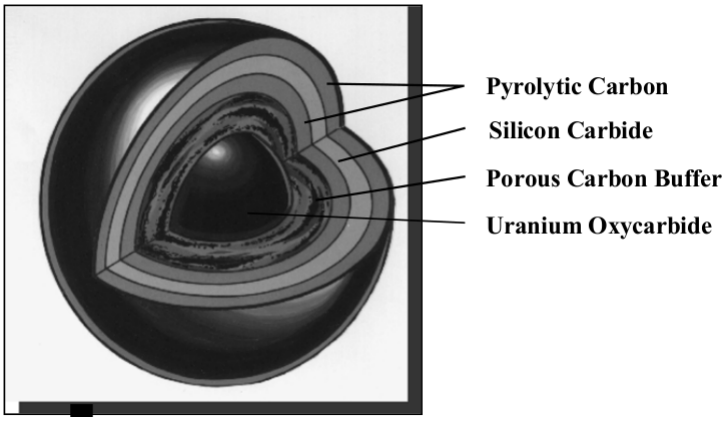


Gambar 1.1: Ilustrasi bentuk bahan bakar *pebble* [1]

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*).



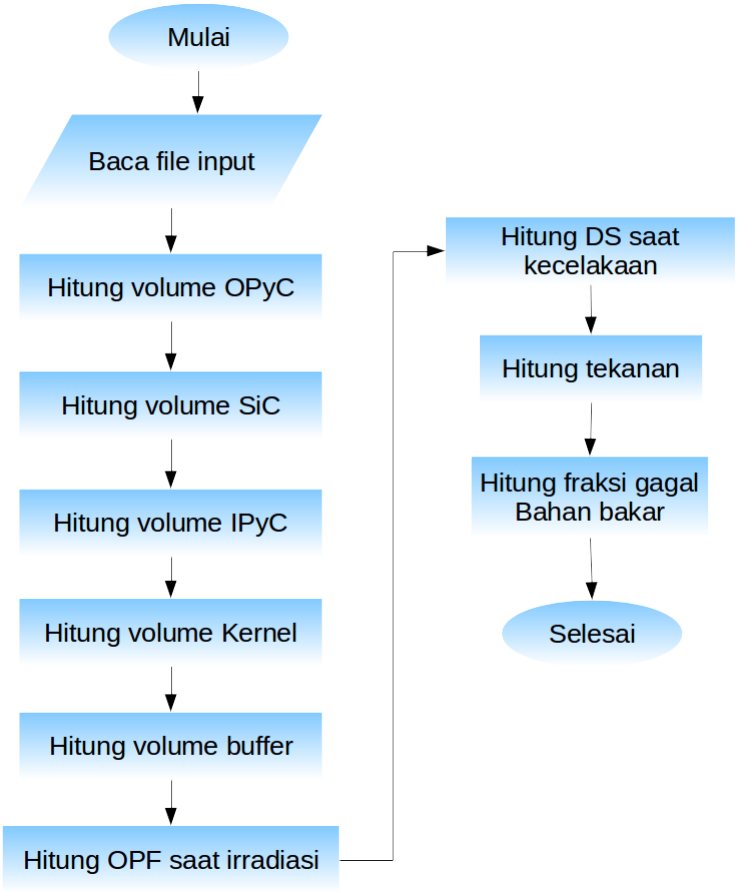
Gambar 1.2: Komposisi elemen pelapis partikel [1]

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 3.1 yang dibangun sepenuhnya berbasis pengetahuan yang diperoleh dari dokumen laporan teknis [4].



Gambar 2.1: Diagram alir perhitungan TRAIC

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 3.1

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 3.1). 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 3.1).

2. Data tentang kekuatan SiC. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusCharacteristics` (baris ke-6 pada Listing 3.1). 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 3.1).

3. Data tentang sejarah irradiasi. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusIrradiation` (baris ke-7 pada Listing 3.1). 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 dua elemen yang mewakili data kolom kedua dan ketiga tiap akuisisi (baris ke-12 pada Listing 3.1). Ilustrasinya adalah seperti $[[0,593],[17,833],\ldots]$. 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 3.1). 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 3.1). Ilustrasinya adalah seperti $[[0,1033],[0.0271,1033],\ldots]$

.

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.

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

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].

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

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 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 partiekl 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.

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

2.6 Fraksi gagal bahan bakar

Tahapan terkahir 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 (ϕ_0). Dalam analisis ini, nilai ϕ_0 diasumsikan sama dengan 0.
2. Berkurangnya *tensile strength* lapisan SiC (ϕ_1). Hal ini dapat terjadi karena
   - proses irradiasi 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 (ϕ_2).

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

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

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).

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

dengan :

σ_o=*tensile strength* dari SiC [Pa] pada akhir irradiasi

σ_t=tekanan yang dialami SiC [Pa] akibat tekanan gas internal

Variabel tekanan internal pada SiC (σ_t) dihitung dengan 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.

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

dengan :

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

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

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

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

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

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

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.

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

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 irradiasi, lapisan SiC juga dapat terkorosi karena *grain Boundary*. Jika korosi akibat irradiasi tergantung pada sejarah irradiasi 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).

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

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

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

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).

$$k = k_o \cdot e^{\frac{-Q}{R \cdot T}} \tag{2.20}$$

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 ζ yang nilainya mengikuti persamaan (2.21).

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

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).

$$\zeta(t_2) = \zeta(t_1) + k(T_m) \cdot (t_2 - t_1) \tag{2.22}$$

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

Kemudian, fraksi gagal ϕ_2 sedemikian rupa sehingga nilainya ≤ 1. Karena itu, variabel ϕ_2 selanjutnya didefinisikan sebagai persamaan (2.23).

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

Nilai α dan β 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 °C. Sementara RDE tidak dirancang untuk sampai pada temperatur tersebut.

BAB 3

Penerapan

3.1 Pendahuluan

TRIA *Code* yang telah dijelaskan sebelumnya secara umum dapat dikelompokkan menjadi dua tugas utama, masing-masing adalah perhitungan di waktu irradiasi dan kecelakaan. Penerapannya adalah seperti pada Listing 3.1.

Listing 3.1: 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                 """print element"""
20                 if statusAll==0:
21                         if element[0]==statusGeometry:
22                                 for j in range(1,6):
23                                         y=float(element[j])
24                                         dimensi.append(y)
25                                 statusAll=1
26                                 print dimensi
27
28                 elif statusAll==1:
29                         if element[0]==statusCharacteristics:
30                                 x=i+1
31                         elif i==x:
32                                 try:
33                                         for j in range(0,5):
34                                                 y=float(element[j])
35                                                 characteristics.append(y)
36                                         statusAll=2
37                                         print characteristics
38                                 except:
39                                         x=i+1
40
41
42
```

```
43                  elif statusAll==2:
44                          if element[0]==statusIrradiation:
45                                  x=i+1
46                          elif i==x:
47                                  if element[0]==statusAccident:
48                                          statusAll=3
49                                  else:
50                                          temp=[]
51                                          try:
52                                                  y=float(element[1])
53                                                  if y>0:
54                                                          temp.append(y*24*3600)
55                                                          y=float(element[2])
56                                                          temp.append(y)
57                                                          irradiation.append(temp)
58                                                  x=i+1
59                                          except:
60                                                  print element
61                                                  x=i+1
62
63
64
65                  elif statusAll==3:
66                          temp=[]
67                          try:
68                                  y=float(element[1])*24*3600
69                                  temp.append(y)
70                                  y=float(element[2])
71                                  temp.append(y)
72                                  accident.append(temp)
73                          except:
74                                  x=i+1
75
76          return dimensi, characteristics, irradiation, accident
77
78
79  def OPF(irradiation,dt):
80          x=len(irradiation)
81          y=(17.0/dt)*24*3600
82          tb=1020*24*3600
83          z=0.0
84          for i in range(x):
85                  j=irradiation[i]
86                  a1=8.3143*j[1]
87                  a=-163000/(a1)
88                  b=math.exp(a)
89                  g=2*(8.32e-11)*b
90                  g1=g*(tb-j[0])*y
91                  z=z+g1
92
93          Tb=0.85e4/((2*math.log10(tb))-(math.log10(z))-10.08)
94          logDS=-2.3-(8116/Tb)
95          ds=math.pow(10,logDS)
96          taui=ds*tb
97          return z,Tb,ds,taui
98
99  def FTau(tau):
100         looping=0.0
101         for n in range (1,2000):
102                 pangkat=math.pow(n,2)*math.pow(math.pi,2)*tau
103                 A=math.exp(-(pangkat))
104                 B=math.pow(n,4)*math.pow(math.pi,4)
105                 looping=looping+((1-A)/B)
106
107         ftau=1-((6/tau)*looping)
108         return ftau
109
110 def OPFAccident(Tb,tb,T):
111         logOPF=-10.08-(8500/Tb)+(2*math.log10(tb))-(0.404*((10000/T)-(10000/(Tb+75))))
112         opfa=math.pow(10,logOPF)
```

```
113         return opfa
114
115 def Pressure(Tb,dsAccident,Vk,Vf,accident):
116         p=[]
117         for i in range(len(accident)):
118                 x=accident[i]
119                 y=dsAccident[i]
120                 p.append((y[2]*Ff*OPFAccident(Tb,x[1])*Fb)/((Vf/Vk)*R*x[1]/Vm))
121         return p
122
123 def DS(T):
124         logDS=-2.3-(8116/T)
125         ds=math.pow(10,logDS)
126         return ds
127
128 def interpolasi(a,b,c):
129         x1=a[0]
130         y1=a[1]
131         x2=b[0]
132         y2=b[1]
133
134         i=[]
135         selisih=(x2-x1)/c
136         x=x1
137
138         for k in range(1,c):
139                 j=[]
140                 x=x+selisih
141                 y=(((x-x1)/(x2-x1))*(y2-y1))+y1
142                 j.append(x)
143                 j.append(y)
144                 i.append(j)
145         return i
146
147 if __name__=="__main__":
148         if len(sys.argv)==3:
149                 f=sys.argv[1]
150                 dt=int(sys.argv[2])
151         else:
152                 f="example.pan.in"
153                 dt=19
154
155         x=readdata(f)
156         dimensi=x[0]
157         characteristics=x[1]
158         irradiation=x[2]
159         accident=x[3]
160
161         VolRef1=(4/3)*math.pi*dimensi[0]*dimensi[0]*dimensi[0]
162         VolRef2=(4/3)*math.pi*dimensi[1]*dimensi[1]*dimensi[1]
163         VolOPyC=VolRef1-VolRef2
164
165         """Volume SiC"""
166         VolRef3=(4/3)*math.pi*dimensi[2]*dimensi[2]*dimensi[2]
167         VolSiC=VolRef2-VolRef3
168
169         """Volume IPyC"""
170         VolRef4=(4/3)*math.pi*dimensi[3]*dimensi[3]*dimensi[3]
171         VolIPyC=VolRef3-VolRef4
172
173         """Volume Buffer & Volume Kernel"""
174         VolKernel=(4/3)*math.pi*dimensi[4]*dimensi[4]*dimensi[4]
175         VolBuff=VolRef4-VolKernel
176         print("Volume OPyC:",VolOPyC)
177         print("Volume SiC:",VolSiC)
178         print("Volume IPyC:",VolIPyC)
179         print("Volume Buffer:",VolBuff)
180         print("Volume Kernel:",VolKernel)
181
182         irradiation2=[]
```

```
183         accident2=[]
184         lenIR=len(irradiation)
185         lenAcc=len(accident)
186
187         fi=file('irradiasi2','w')
188         irradiation2.append(irradiation[0])
189         b=0
190         fi.write(str(b)+', '+str(irradiation2[0])+'\n')
191         b=b+1
192         for x in range(1,lenIR):
193                 temp=[]
194
195                 i=irradiation[x-1][1]
196                 j=irradiation[x][1]
197                 if i!=j:
198                         temp=interpolasi(irradiation[x-1],irradiation[x],dt)
199                         for y in range(len(temp)):
200                                 irradiation2.append(temp[y])
201                                 fi.write(str(b)+', '+str(irradiation2[b])+'\n')
202                                 b=b+1
203                 irradiation2.append(irradiation[x])
204                 fi.write(str(b)+', '+str(irradiation2[b])+'\n')
205                 b=b+1
206
207         fi.close()
208
209         fi=file('accident2','w')
210         b=0
211         fi.write(str(b)+', '+str(irradiation2[0])+'\n')
212         b=b+1
213         accident2.append(accident[0])
214         for x in range(1,lenAcc):
215                 temp=[]
216
217                 i=accident[x-1][1]
218                 j=accident[x][1]
219                 if i!=j:
220                         temp=interpolasi(accident[x-1],accident[x],dt)
221                         for y in range(len(temp)):
222                                 accident2.append(temp[y])
223                                 fi.write(str(b)+', '+str(irradiation2[b])+'\n')
224                                 b=b+1
225                 accident2.append(accident[x])
226                 fi.write(str(b)+', '+str(irradiation2[b])+'\n')
227                 b=b+1
228
229
230         TB=OPF(irradiation2,1)
231         z=TB[0]
232         Tb=TB[1]
233         dsi=TB[2]
234         tauI=TB[3]
235         print("OPF="+str(z)+", Tb="+str(Tb)+", DS="+str(dsi)+", TauI="+str(tauI))
236
237         tb=1020*24*3600
238         Vk=VolKernel
239         Vf=0.5*VolBuff
240         Ff=0.31
241         R=8.3143
242         Vm=2.43796e-5
243         Fb=0.08
244
245         a=(0.5*(pow(dimensi[1],3)+pow(dimensi[2],3)))
246         r=pow(a,(1.0/3))
247         do=dimensi[1]-dimensi[2]
248
249         sigma0=756.e6
250         m=6.93
251         print(FTau(0.05),FTau(0.5),FTau(1),FTau(2))
252         print('ln(2)='+str(math.log(2)))
```

```
253         pressure=[]
254         SigmaT=[]
255         phi1=0
256
257         for i in range(len(accident2)):
258                 dsa=DS(accident2[i][1])
259                 tauA=dsa*accident2[i][0]
260                 if accident2[i][0]==0:
261                         Fd=FTau(tauI)
262                 else:
263                         Fd=(((tauI+tauA)*FTau(tauI+tauA))-(tauA*FTau(tauA)))/tauI
264
265                 opfa=OPFAccident(Tb,tb,accident2[i][1])
266                 n=((Fd*Ff)+opfa)*Fb*(Vk/Vm)
267                 p=n*R*accident2[i][1]/Vf
268                 pressure.append(p)
269
270                 vdot=(5.87e-7)*math.exp(-179500/(8.3143*accident2[i][1]))
271                 a=(1+((vdot*accident2[i][0])/do))
272                 sigmaT=((r*p)/(2*do))*a
273                 SigmaT.append(sigmaT)
274
275                 a1=sigmaT/sigma0
276                 a=pow(a1,m)
277                 b=math.exp(-math.log(2)*a)
278                 phi=1-b
279                 phi1=phi1+phi
```

3.2 Saat irradiasi

Untuk perhitungan pertama, kita harus bisa mendapatkan nilai OPF (persamaan 2.1), T_B (persamaan 2.4), *DS* persamaan 2.5) dan τ_i (persamaan 2.6). Pada Listing 3.1, tugas tersebut didelegasikan pada fungsi `OPF (irradiation, dt)` (baris ke-72 s/d 90). Fungsi tersebut membutuhkan dua argumen, dengan argumen pertama adalah sejarah irradiasi dalam bentuk array. Jika melihat contoh yang disajikan pada Lampiran 3.2, data tersebut terletak setelah baris berisi `INPUT: Irradiation Temp. Hystory`. Array akan berdimensi dua, yaitu setiap elemen array merupan array dengan dua elemen, masing-masing adalah waktu dan temperatur irradiasi.

Argumen kedua sejatinya menunjukkan selisih waktu antara pengambilan data yang satu dengan pengambilan data berikutnya. Nilai tersebut diterapkan dalam Listing 3.1 sebagai variabel *y*. Karena selisih waktu antar pengambilan data yang dimaksud pada contoh Lampiran 3.2 dinilai cukup tinggi (17 hari), maka interpolasi dicoba untuk diterapkan juga. Hal ini dilatarbelakangi hasil simulasi yang disajikan dalam literatur [4]. Sebagai ilustrasi, jika tanpa interpolasi, nilai temperatur irradiasi $T_B = 1057.41\,^{\circ}\text{C}$. Bila dijalankan dengan menggunakan skenario selisih antar pengambilan data adalah $\frac{17}{2} = 8.5$ hari, dihasilkan nilai $T_B = 1046.55\,^{\circ}\text{C}$. Sedangkan, nilai T_B yang dihitung dengan skenario selisih antar pengambilan data adalah 1 hari (interpolasi jumlah pembagian sebanyak 17 menghasilkan nilai $1040.07\,^{\circ}\text{C}$. Nilai T_B selanjutnya akan semakin kecil dengan semakin kecilnya jarak pengambilan data (semakin besar pembagian untuk interpolasi).

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: Contoh file input

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

```
38    629   1023
39    646   1093
40    663   1123
41    680   593
42    697   833
43    714   1023
44    731   1093
45    748   1123
46    765   593
47    782   833
48    799   1023
49    816   1093
50    833   1123
51    850   593
52    867   833
53    884   1023
54    901   1093
55    918   1123
56    935   593
57    952   833
58    969   1023
59    986   1093
60    1003  1123
61  1020    593

      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
```