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

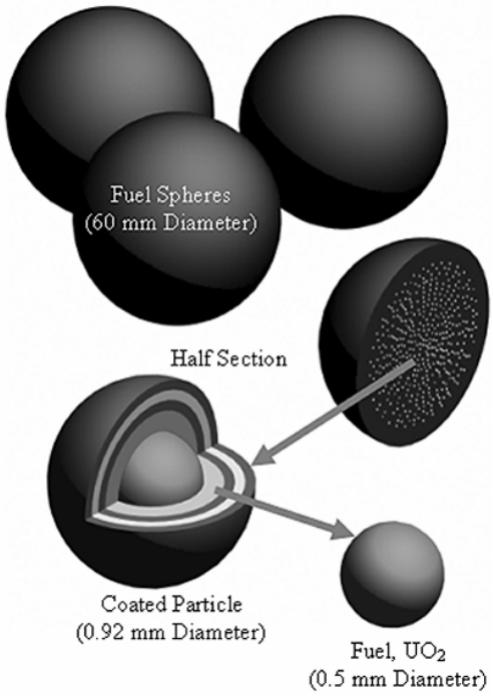
Daftar Gambar

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.

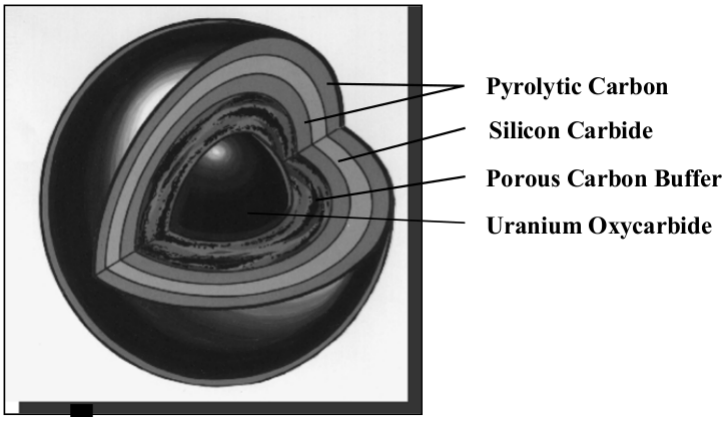


Gambar 1.1: Ilustrasi bentuk bahan bakar *pebble* [1]

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*).



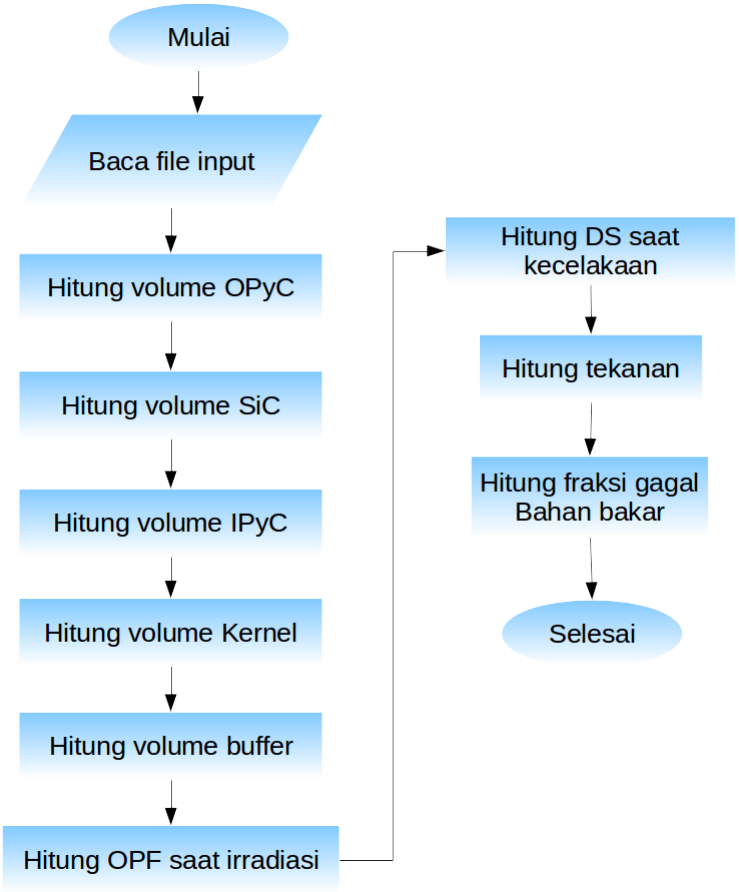
Gambar 1.2: Komposisi elemen pelapis partikel [1]

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



Gambar 2.1: Diagram alir perhitungan TRIAC

2.2 Membaca *file input*

Sub rutin ini ditujukan untuk membaca file input dengan format seperti terdapat pada Lampiran 3.4. 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 Listing 1 dan akan dijelaskan pada sub bab 3.1.

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 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 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. PANAMA [4] memodelkan fraksi gagal partikel bahan bakar dari sejauh mana lapisan silikon karbida mampu menahan tekanan akibat rilisnya gas produk fisi. Untuk menghitung tekanan yang timbul ketika kecelakaan terjadi pada waktu tertentu, sehingga menyebabkan panas tertentu, digunakan persamaan (2.9) [4].

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

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 partikel kernel $\left[\frac{m^3}{mole}\right]$, didefinisikan sebagai rasio berat 1 mol material kernel terhadap kerapatannya. Menurut Verfondern [4], V_m untuk $(Th,U)O_2$, UO_2 dan UCO masing-masing adalah $2.52 \cdot 10^{-5}[\frac{m^3}{mole}]$, $2.44 \cdot 10^{-5}[\frac{m^3}{mole}]$ dan $2.51 \cdot 10^{-5}[\frac{m^3}{mole}]$.

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 TRIAC (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).

$$\phi_{total} = 1 - (1 - \phi_0) \cdot (1 - \phi_1) \cdot (1 - \phi_2) \tag{2.11}$$

2.6.1 Fraksi gagal akibat berkurangnya *tensile strength*

Fraksi gagal partikel triso dimodelkan dengan apa yang diistilahkan Verfondern sebagai model bejana tekan [4]. Hal ini disebabkan karena fraksi gagal dipengaruhi oleh variabel-variabel yang terenkapsulasi dalam parameter tenanan internal dan kekuatan lapisan silikon karbida. 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

m=parameter Weibull (dijelaskan selanjutnya)

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], dihitung menggunakan persamaan (2.9)

$\dot{v}$=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).

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

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

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. Nilai Γ_m ditentukan menggunakan persamaan (2.18).

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

$$\log \Gamma_m = 0.394 + \frac{0.065 \cdot 10^4}{T_B} \tag{2.18}$$

Sama seperti σ_{oo}, nilai m_{oo} juga diperoleh dengan mengukur parameter tersebut pada partikel yang belum diiradiasi. Tabel 2.1 menunjukkan nilai σ_{oo} dan m_{oo} pada beberapa jenis specimen sebelum dikenakan irradiasi [4].

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 (2.19), di mana nilai m_o diperoleh dari persamaan (2.17)

$$m = m_o \cdot \left(0.44 + 0.56 \cdot e^{-\dot{\eta} \cdot t}\right) \tag{2.19}$$

dan nilai $\dot{\eta}$ mengikuti persamaan 2.20 dengan pola yang sama seperti persamaan (2.14).

$$\dot{\eta} = 0.565 \cdot e^{\left(\frac{-187400}{R \cdot T}\right)} [s^{-1}] \tag{2.20}$$

Tabel 2.1: Nilai σ_{oo} dan m_{oo} untuk berbagai jenis specimen[4]

Specimen	Sebelum irradiasi		Setelah irradiasi	
	σ_{oo} [MPa]	m_{oo}	σ_o [MPa]	m_o
EO 1674	722	7.0	660	6.1
EO 1607	850	8.0	777	7.0
HT 150-167	600	6.0	549	5.3
EO 249-251	453	5.0	414	4.4
EO 403-405	867	8.4	793	7.4
EUO 1551	1060	8.5	969	7.4
ECO 1541	1080	6.4	987	5.6
EC 1338	998	7.4	912	6.5

2.6.2 Fraksi gagal bahan bakar akibat *weight loss*

Laju *weight loss* yang terjadi akibat tingginya temperatur saat terjadi kecelakaan mengikuti persamaan (2.21).

$$k = k_o \cdot e^{\frac{-Q}{R \cdot T}} \tag{2.21}$$

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 (2.22).

$$\zeta = \int_{t_1}^{t_2} k(T)dt \tag{2.22}$$

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

Secara numerik, persamaan (2.22) dapat dituliskan sebagai persamaa (2.23).

$$\zeta(t_2) = \zeta(t_1) + k(T_m) \cdot (t_2 - t_1) \tag{2.23}$$

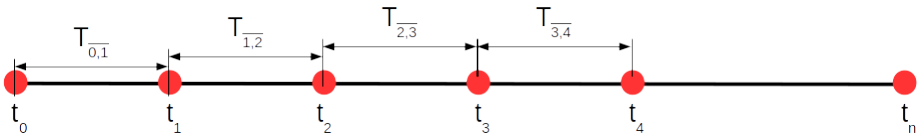
dengan $k(T_m) = \frac{375}{d_o} \cdot e^{\left(\frac{-556000}{R \cdot T_m}\right)}$.

Kemudian, fraksi gagal ϕ_2 sedemikian rupa sehingga nilainya ≤ 1. Karena itu, variabel ϕ_2 selanjutnya didefinisikan sebagai persamaan (2.24).

$$\phi_2(t,T) = 1 - e^{-\alpha \cdot \zeta^{\beta}} \tag{2.24}$$

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.



Gambar 2.2: Hubungan antara waktu dan temperatur pada perhitungan ϕ_1

2.6.3 Pertumbuhan fraksi gagal

Berdasarkan PANAMA [4], Verfondern memodelkan pertumbuhan fraksi gagal partikel triso akibat berkurangnya *tensile strength* adalah seperti persamaan (2.25) berikut.

$$\phi_1 = \phi_1(t_2, T_m) - \phi_1(t_1, T_m) \tag{2.25}$$

T_m merupakan temperatur rata-rata antara waktu t_1 dan t_2. Ilustrasinya disajikan dalam Gambar 2.2

Disebutkan Verfondern [4], nilai ϕ_1 saat kecelakaan dimulai (t_0 seperti pada Gambar 2.2) merupakan fungsi dari sejarah irradiasi. Sedangkan untuk waktu-waktu selanjutnya $(t_1, t_2, \cdots, t_n)$ merupakan akumulasi dari nilai ϕ_1 pada persamaan (2.25). Jika ϕ_1 pada t_1 lebih besar daripada ϕ_1 pada saat t_0, maka akumulasikan nilai ϕ_1. Jika sebaliknya, gunakan nilai ϕ_1 sebelumnya untuk perhitungan selanjutnya (nilai ϕ_1 tetap).

Sebagai ilustrasi, saat menghitung nilai ϕ_1 di $t = t_1$, maka diperlukan nilai $\phi_1(t_0, T_m)$ (nilai pertama) dan $\phi_1(t_1, T_m)$ (nilai kedua). Nilai pertama adalah fungsi irradiasi, sedangkan nilai kedua diperoleh dari persamaan (2.12) dengan parameter-parameter yang sesuai. Selisih keduanya akan menentukan nilai ϕ_1 di titik $t = t_1$. Jika selisih nilai kedua dan pertama positif, selisih nilai tersebut diakumulasikan pada nilai ϕ_1 di $t = t_0$. Tetapi jika sebaliknya, maka nilai ϕ_1 di titik $t = t_1$ sama dengan nilai ϕ_1 di titik $t = t_0$. Skenario yang sama berlaku untuk titik-titik waktu selanjutnya.

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. Saat irradiasi, hubungan saling ketergantungan antar variabel adalah seperti Gambar 3.1. Sedangkan saat kecelakaan, hubungannya adalah seperti pada Gambar 3.2.

$$opf \rightarrow T_b \rightarrow DS \rightarrow \tau_i$$

Gambar 3.1: Hubungan ketergantungan antar variabel di fase irradiasi

$$T \rightarrow DS \rightarrow \tau \rightarrow Fd \rightarrow opf \rightarrow pressure \rightarrow \dot{v} \rightarrow \sigma_t \rightarrow \phi$$

Gambar 3.2: Hubungan ketergantungan antar variabel di fase kecelakaan

Selanjutnya, triac juga memerlukan sejumlah data yang harus diberikan oleh pengguna sebelum perhitungan dimulai. Selain data-data seperti yang akan dijelaskan dalam sub bab 3.2, diperlukan juga beberapa data lain. Karena triac mengadopsi perhitungan yang dilakukan dalam PANAMA [4], maka triac juga memerlukan data seperti yang diperlukan PANAMA. Tabel 3.1 menyajikan beberapa parameter serta nilainya yang diperlukan oleh triac, masing untuk HTR-Modul dan HTR-500.

Selain itu, triac juga memerlukan parameter lain berupa status interpolasi. Dengan status ini, sejarah irradiasi/kecelakaan akan diinterpolasi atau menggunakan nilai yang diberikan pengguna dari *file input*.

3.2 Pembacaan *file input*

Seluruh proses dalam triac didahului dengan membaca *file input* dengan format yang sama seperti pada Lampiran 3.4. Penerapan pembacaan *file input* adalah seperti pada Listing 1.

Di Listing 1, pembacaan *input data* dilakukan secara sekuensial dan manual. Nilai-nilai yang harus dibaca ditentukan berdasarkan informasi yang ada pada *file input*. Sebagai

Tabel 3.1: Tambahan data yang diperlukan triac[4]

Parameter	HTR-Modul	HTR-500
Jenis partikel	UO_2	UO_2
Burnup / FIMA / Fb	0.08	0.08
Fast fluence / Γ $[10^{25} m^{-2}]$	1.4	1.4
σ_{oo} [MPa]	834	834
m_{oo}	8.02	8.02

contoh, untuk membaca nilai geometri, digunakan karakter "[m]" sebagai penanda. Jika ditemukan karakter tersebut, maka di saat itulah pembacaan nilai geometri dilakukan. Hal inilah yang dimaksud sebagai pembacaan secara manual. Ketika karakter yang diperlukan berubah, maka modifikasi harus dilakukan pada modul ini.

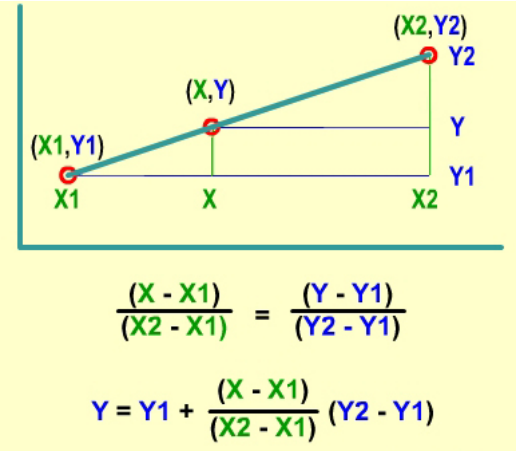
Selain nilai terkait geometri, diperlukan juga pembacaan untuk nilai *physical properties* serta sejarah operasi, baik saat operasi normal maupun kecelakaan. Pembacaan nilai yang berbeda dilakukan secara berurutan berdasarkan kemunculan nilai tersebut dalam *file input*. Hal inilah yang dimaksud dengan pembacaan secara sekuensial.

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

1. Data tentang geometri *pebble*. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusGeometry` (baris ke-4. 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-9).

2. Data tentang kekuatan SiC. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusCharacteristics` (baris ke-5 pada Listing 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-10).

3. Data tentang sejarah irradiasi. Data ini diidentifikasi menggunakan teks yang didefinisikan oleh variabel `statusIrradiation` (baris ke-6 pada Listing 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-11). Ilustrasinya adalah



Gambar 3.3: Ilustrasi interpolasi linier yang digunakan

seperti $[[0,593],[1468800,833],\ldots]$ dengan informasi waktu pengukuran dalam satuan detik. 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-7). 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-12). Ilustrasinya adalah seperti $[[0,1033],[2341.44,1033],\ldots]$ dengan informasi waktu pengukuran dalam satuan detik.

.

Namun, terlihat pada baris ke-76 dari Listing 1, terdapat total 5 variabel yang dikembalikan ke fungsi awal, dengan variabel kelima adalah $b-a$. Variabel ini adalah rentang waktu pengukuran data irradiasi.

Selain itu, untuk meningkatkan ketelitian perhitungan, disiapkan juga modul interpolasi secara linier. Modul ini disiapkan agar sejarah operasi normal dan kecelakaan sehingga dapat diperoleh hasil yang tepat. Penerapan dari modul interpolasi linier tersebut disajikan pada Listing 2.

Seperti terlihat pada Lampiran 3.4, sejarah operasi normal atau disebut juga sebagai sejarah irradiasi, terdapat 3 kolom dalam *file input*. Demikian juga untuk sejarah ketika terjadi kecelakaan. Ketiganya adalah nomor urut, hari ke sekian dan temperatur. Dengan melakukan interpolasi, selisih hari yang digunakan dapat diperkecil. Dalam contoh *file input*, selisih pencatatan adalah 17 hari. Dengan interpolasi, kita dapat mengestimasi sejarah dalam selisih waktu yang lebih singkat.

Interpolasi yang diterapkan dapat diilustrasikan dalam Gambar 3.3 [1]. Argumen ketiga dari fungsi `linier (a,b,c)`, c, adalah jumlah partisi diantara nilai x_1 dan x_2. Nilai tersebut adalah *dt* yang merupakan argumen ketika mengeksekusi kode komputer TRIAC (Listing 4). Penggunaan fungsi interpolasi ini dilakukan di Listing 4 pada baris ke-53 s/d 66.

[1] `http://jadipaham.com/wp-content/uploads/2015/10/Rumus-interpolasi-linear.jpg`

3.3 TRIAC Core

Terdapat empat fungsi di dalam modul `core.py` seperti terlihat pada Listing 3. Fungsi-fungsi yang terdapat dalam modul ini dianggap sebagai fungsi yang sering digunakan dan relatif kompleks jika diletakkan dalam program utama TRIAC. Berikut adalah penjelasannya.

1. `OPF (irradiation,y,tb)` (baris ke-3 s/d 15). Fungsi tersebut membutuhkan tiga argumen, dengan argumen pertama adalah sejarah irradiasi dalam bentuk array. Jika melihat contoh yang disajikan pada Lampiran 3.4, 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 (dalam detik) dan temperatur irradiasi.

   Argumen kedua, *y* adalah rentang waktu pengukuran ketika massa irradiasi. Contoh pada Lampiran 1 menunjukkan bahwa pengukuran dilakukan setiap 17 hari. Ketika kita ingin rentang pengukuran ini lebih kecil dari 17, maka kita dapat memperolehnya dengan fungsi interpolasi. Waktu interpolasi ini harus dalam satuan detik. Sedangkan argumen ketiga adalah total masa irradiasi, yang dalam contoh Lampiran 1 adalah 1020 hari.

   Fungsi OPF akan menghitung akumulasi nilai *g* untuk setiap perubahan temperatur dan waktu irradiasi seperti dijelaskan pada persamaan (2.3). Akumulasi nilai *g* tersebut adalah nilai OPF seperti dijelaskan pada persamaan (2.1). Itu sebabnya kenapa fungsi ini diberi nama `OPF`.

   Kemudian, nilai OPF digunakan untuk menghitung nilai Tb seperti dijelaskan persamaan (2.4). Nilai Tb selanjutnya digunakan untuk menghitung nilai *ds* seperti dijelaskan persamaan (2.5) dan diterapkan oleh fungsi `DS` (baris ke-31 s/d 34 Listing 3). Akhirnya, nilai τ_i diperoleh dari nilai *ds* seperti dijelaskan persamaan (2.6). Tetapi, rentetan perhitungan tersebut tidak dilakukan di fungsi `OPF`, melainkan dalam program `triac.py` seperti pada Listing 4. Fungsi yang dibuat diusahakan untuk hanya mengerjakan satu fungsi saja.

2. `FTau (tau)` (baris ke-17 s/d 25). Fungsi ini digunakan untuk menghitung nilai f_τ seperti dijelaskan pada persamaan (2.8). Fungsi ini menerapkan nilai 2000 sebagai batas atas iterasi.

3. `OPFAccident (Tb,tb,T)` (baris ke-27 s/d 30). Fungsi ini akan menerima argumen kondisi kecelakaan melalui argumen ketiga (T). Sedangkan argumen pertama (Tb) diperoleh dari fungsi pertama, `OPF`.

4. `DS(T)` (baris ke-32 s/d 35). Fungsi ini adalah fungsi perantara untuk mendapatkan nilai τ_i.

5. `volume(r)`. Fungsi ini digunakan untuk menghitung volume setiap lapisan partikel triso.

6. `weibullParam(Tb,T,t,m00,gamma)`. Fungsi ini digunakan untuk menghitung parameter *m* seperti dijelaskan oleh persamaan (2.19). Fungsi ini digunakan sejak perhitungan memasuki kondisi kecelakaan. Diperlukan 4 argumen untuk mengeksekusi fungsi ini, masing-masing adalah T_b (diperoleh dari perhitungan terhadap sejarah irradiasi), T dan t (masing-masing adalah temperatur dan waktu sejak terjadinya kecelakaan), m_{oo} (parameter yang menggambarkan distribusi Weibull *tensile strength* untuk semua partikel triso sebelum diirradiasi), serta Γ.

7. `tekanan(Fd,opf,Vk,T,Vf)` (baris ke-51 s/d 56). Diperlukan 5 argumen untuk menghitung nilai tekanan yang telah dijelaskan di persamaan (2.9). Khusus untuk argumen opf, nilainya diperoleh dari fungsi `OPFAccident`.

8. `sigmaT(r,p,d,t,T)` (baris ke-58 s/d 61). Diperlukan 5 argumen untuk menghitung nilai σ_t. Argumen r dan d adalah parameter yang dijelaskan di persamaan (2.13). Khusus untuk argumen d, di persamaan (2.13) dinyatakan sebagai d_0. Argumen t dan T adalah waktu dan temperatur setelah terjadinya kecelakaan. Khusus untuk argumen T, nilainya tidak serta-merta sesuai dengan waktu t. Seperti dijelaskan pada Gambar 2.2, nilai temperatur yang digunakan dalam perhitungan adalah nilai rata-rata dari dua waktu akuisisi data yang beriringan. Sedangkan argumen p adalah variabel tekanan yang telah dijelaskan sebelumnya.

9. `phi(sigma0, sigmaT, m)` (baris ke-63 s/d 67). Parameter inilah yang menjadi fokus triac, yaitu memprediksi fraksi gagal partikel triso seperti telah dijelaskan di persamaan (2.12).

3.4 Perhitungan TRIAC

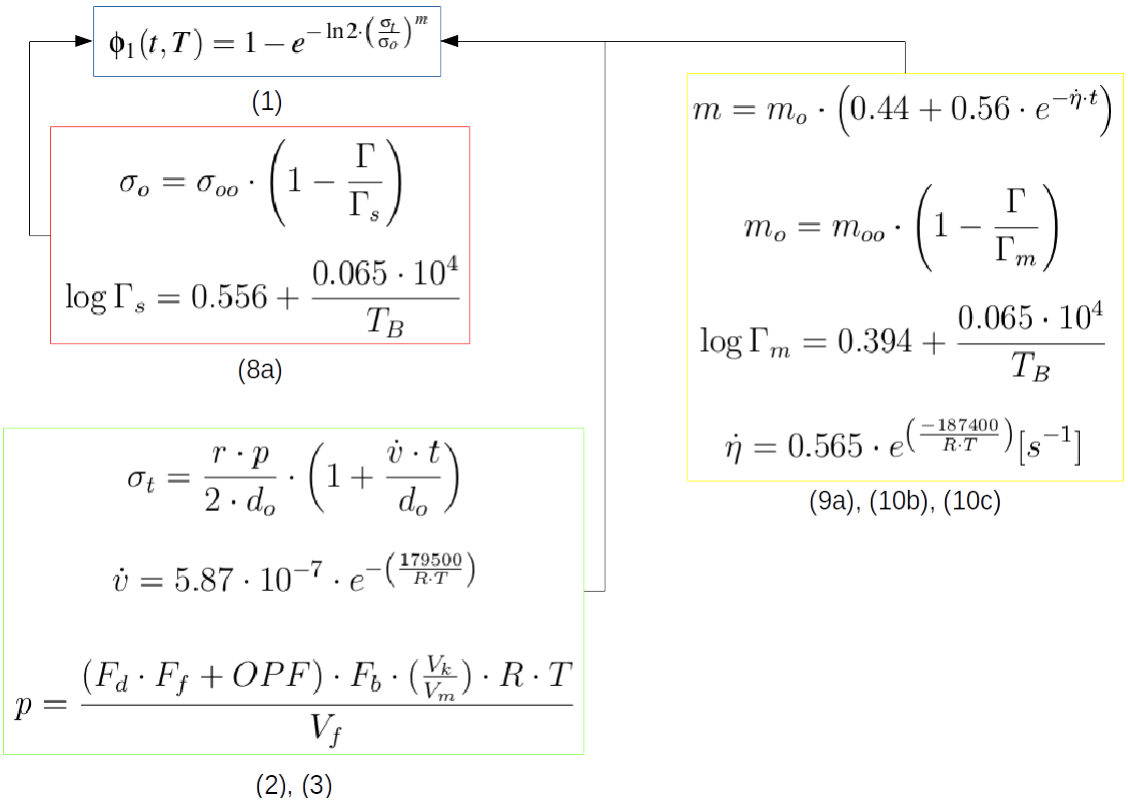
Bagian ini adalah inti dari perhitungan TRIAC yang alur eksekusinya diilustrasikan pada Gambar 2.1. Sedangkan hubungan interaksi antar fungsi untuk mendapatkan nilai fraksi gagal partikel triso setelah sekian waktu sejak terjadi kecelakaan dapat diilustrasikan seperti Gambar 3.4. Kotak dengan warna merah, kuning dan hijau pada Gambar 3.4 menunjukkan formula-formula yang hasilnya menjadi masukan untuk formula pada kotak berwarna biru. Sementara angka di bawah kotak-kotak tersebut adalah nomor formula dalam dokumen PANAMA [4].

Sedangkan Gambar 3.5 merupakan kelanjutan dari interaksi yang ditunjukkan Gambar 3.4, khususnya untuk menyediakan nilai masukan bagi parameter nilai tekanan yang dialami lapisan silikon karbida. Sama seperti Gambar 3.4, angka di bawah kotak-kotak berwarna yang berisi formula pada Gambar 3.5 menunjukkan nomor formula pada dokumen PANAMA [4]. Selain informasi nomor persamaan pada dokumen PANAMA, ditunjukkan pula bahwa kotak berwarna kuning merupakan variabel yang dipengaruhi oleh jenis partikel triso. Formula yang disajikan merupakan formula empiris untuk jenis partikel UO_2. Sementara untuk kotak berwarna hijau, selain dipengaruhi oleh jenis material partikel triso, juga dipengaruhi oleh kondisi apakah partikel triso sedang berada pada masa irradiasi (formula pertama) atau kecelakaan (formula kedua).

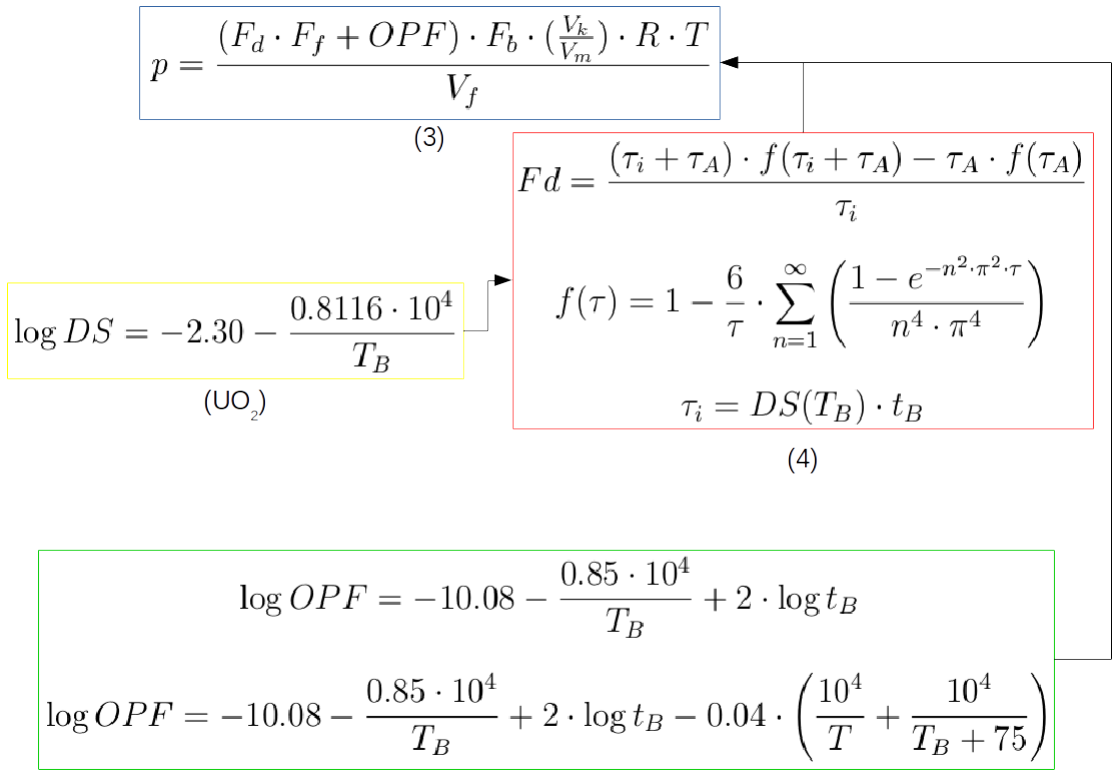
Program ini akan menerima dua argumen selain nama programnya sendiri, yaitu nama *file input* (baris ke-8) serta jumlah interpolasi yang diinginkan (baris ke-9) dalam rentang pengukuran yang sudah ada. Tetapi, jika pengguna tidak memberikan argumen tersebut, program akan dieksekusi dengan *file input* dan jumlah interpolasi yang telah ditetapkan (baris ke-11 dan 12).

Proses selanjutnya adalah membaca informasi dari *file input*. Ada lima informasi yang harus diperoleh dari *file input*, masing-masing adalah informasi geometri, karakteristik material, sejarah irradiasi dan kecelakaan serta rentang pengukuran temperatur saat irradiasi. Setelah data-data tersebut diperoleh, langkah selanjutnya adalah perhitungan geometri partikel triso. Proses ini dilakukan dalam baris ke-23 s/d 37.

Langkah selanjutnya setelah perhitungan geometri adalah interpolasi. Tetapi, karena opsi tanpa interpolasipun harus diakomodasi, maka ada kondisi yang harus dipenuhi seperti pada baris ke-44 dan baris ke-96. Jika nilai $dt > 1$, maka interpolasi harus dilakukan.



Gambar 3.4: Interaksi antar fungsi untuk mendapatkan fraksi gagal partikel triso



Gambar 3.5: Interaksi antar fungsi untuk mendapatkan nilai tekanan yang dialami lapisan silikon karbida

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
```

Lampiran 2: InputData.py

Listing 1: InputData.py

```
import sys, math, re
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
        a=0.0
        b=0.0
        c=0
        for baris in f.readlines():
                i=i+1
                element=baris.split('\t')
                if len(element)!=0:
                        if statusAll==0:
                                if element[0]==statusGeometry:
                                        for j in range(1,7):
                                                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 i==x:
                                        if element[0]==statusAccident:
                                                statusAll=3
                                        else:
                                                temp=[]
                                                try:
                                                        l=float(element[1])
                                                        c=c+1
                                                        if l>=0:
                                                                temp.append(l*24*3600)
                                                                if c==1:
```

```
                                                                    a=1
                                                                if c==2:
                                                                    b=1
                                                                m=float(element[2])
                                                                temp.append(m)
                                                                irradiation.append(temp)
                                                            x=i+1
                                                        except:
                                                            x=i+1

                            elif statusAll==3:
                                temp=[]
                                try:
                                        y=float(element[1])*24*3600
                                        temp.append(y)
                                        y=float(element[2])
                                        temp.append(y)
                                        accident.append(temp)
                                except:
                                        x=i+1

    return dimensi, characteristics, irradiation, accident, b-a
```

Lampiran 3: interpolasi.py

Listing 2: Interpolasi.py

```
def linier(a,b,c):
        x1=a[0]
        y1=a[1]
        x2=b[0]
        y2=b[1]

        i=[]
        selisih=(x2-x1)/c
        x=x1

        for k in range(1,c):
                j=[]
                x=x+selisih
                y=(((x-x1)/(x2-x1))*(y2-y1))+y1
                j.append(x)
                j.append(y)
                i.append(j)
        return i
```

Lampiran 4: core.py

Listing 3: core.py

```
1  import math
2
3  def OPF(irradiation,y,tb):
4          x=len(irradiation)
5          print("Irradiation length:",x)
6          z=0.0
7          for i in range(x):
8                  j=irradiation[i]
9                  a1=8.3143*j[1]
10                 a=-163000/(a1)
11                 b=math.exp(a)
12                 g=2*(8.32e-11)*b
13                 g1=g*(tb-j[0])*y
14                 z=z+g1
15         return z
16
17 def FTau(tau):
18         looping=0.0
19         for n in range (1,2000):
20                 pangkat=math.pow(n,2)*math.pow(math.pi,2)*tau
21                 A=math.exp(-(pangkat))
22                 B=math.pow(n,4)*math.pow(math.pi,4)
23                 looping=looping+((1-A)/B)
24         ftau=1-((6/tau)*looping)
25         return ftau
26
27 def OPFAccident(Tb,tb,T):
28         logOPF=-10.08-(8500/Tb)+(2*math.log10(tb))-(0.404*((10000/T)-(10000/(Tb+75))))
29         opfa=math.pow(10,logOPF)
30         return opfa
31
32 def DS(T):
33         logDS=-2.3-(8116/T)
34         ds=math.pow(10,logDS)
35         return ds
36
37 def volume(r):
38         return (4/3)*math.pi*r*r*r
39
40 def weibullParam(Tb,T,t,m00,gamma):
41         logGammaM=0.394+(650/Tb)
42         gammaM=math.pow(10,logGammaM)
43         m0=m00*(1-(gamma*1e25/gammaM))
44         R=8.3143
45         a=-187400/(R*T)
46         b=math.pow(math.e,a)
47         etaDot=0.565*b
48         c=math.pow(math.e,-etaDot*t)
49         return m0*(0.44+(0.56*c))
50
51 def tekanan(Fd,opf,Vk,T,Vf):
52         R=8.3143
53         Ff=0.31
54         Fb=0.8
```

```
55          Vm=2.43796e-5
56          print(R,Ff,Fb,Vm)
57          return ((Fd*Ff)+opf)*Fb*(Vk/Vm)*R*T/Vf
58
59  def sigmaT(r,p,d,t,T):
60          R=8.3143
61          nu=(5.87e-7)*math.exp(-179500/(R*T))
62          return ((r*p)/(2*d))+((r*p*nu*t)/(2*d*d))
63
64  def phi1(sigma0, sigmaT, m):
65          a=sigmaT/sigma0
66          b=math.pow(a,m)
67          c=math.log(2)*b
68          return 1-math.exp(-c)
```

Lampiran 5: triac.py

Listing 4: triac.py

```
import math,sys,shutil
from InputData import readdata
from Interpolasi import linier
from core import *

if __name__=="__main__":
        if len(sys.argv)==3:
                f=sys.argv[1]
                dt=int(sys.argv[2])
        else:
                f="intriac.txt"
                dt=50

        x=readdata(f)
        dimensi=x[0]
        characteristics=x[1]
        irradiation=x[2]
        accident=x[3]
        rentang=x[4]

        lenIR=len(irradiation)
        tb=irradiation[lenIR-1][0]
        VolRef1=volume(dimensi[0])
        VolRef2=volume(dimensi[1])
        VolOPyC=VolRef1-VolRef2

        """Volume SiC"""
        VolRef3=volume(dimensi[2])
        VolSiC=VolRef2-VolRef3

        """Volume IPyC"""
        VolRef4=volume(dimensi[3])
        VolIPyC=VolRef3-VolRef4

        """Volume Buffer & Volume Kernel"""
        VolKernel=volume(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)
        print('panjang irradiasi=',lenIR,'irradiasi[59]=',irradiation[59])

        if dt>1:
                fi=file('irradiation2','w')
                irradiation2=[]
                irradiation2.append(irradiation[0])
                fi.write('0'+', '+str(irradiation[0][0])+', '+str(irradiation[0][1])+'\n')
                b=1
                for x in range(1,lenIR):
                        temp=[]

```

```
55                          i=irradiation[x-1][1]
56                          j=irradiation[x][1]
57                          if i!=j:
58                                  temp=linier(irradiation[x-1],irradiation[x],dt)
59                                  for y in range(len(temp)):
60                                          irradiation2.append(temp[y])
61                                          fi.write(str(b)+', '+str(temp[y][0])+', '+str(temp[y][1])+'\n
62                                          b=b+1
63                          irradiation2.append(irradiation[x])
64                          fi.write(str(b)+', '+str(irradiation[x][0])+', '+str(irradiation[x][1])+'\n')
65                          b=b+1
66
67                          if x==lenIR-1:
68                                  print('irradiasi terakhir',irradiation[x])
69                  irradiation=irradiation2
70                  irradiation2=[]
71                  y=(float(rentang)/dt)*24*3600
72                  opf=OPF(irradiation,y,tb)
73                  print("OPF="+str(opf)+'\n')
74                  fi.close()
75
76          else:
77                  y=rentang*24*3600
78                  opf=OPF(irradiation,y,tb)
79                  print("OPF="+str(opf)+'\n')
80
81          Tb=0.85e4/((2*math.log10(tb))-(math.log10(opf))-10.08)
82          dsi=DS(Tb)
83          tauI=dsi*tb
84          print("OPF="+str(opf)+", Tb="+str(Tb)+", DS="+str(dsi)+", TauI="+str(tauI))
85
86          Vk=VolKernel
87          Vf=0.5*VolBuff
88          Ff=0.31
89          R=8.3143
90          Vm=2.43796e-5
91          Fb=0.08
92          gamma=1.4
93
94          a=(0.5*(pow(dimensi[1],3)+pow(dimensi[2],3)))
95          r=pow(a,(1.0/3))
96          do=dimensi[1]-dimensi[2]
97          sigma0=756e6
98          m=6.93
99
100         pressure=[]
101         SigmaT=[]
102         phi1=0
103         fphi=file('fPHI','w')
104         fparam=file('fparam','w')
105         fi=file('accident2','w')
106
107         if dt>1:
108                 accident2=[]
109                 lenAcc=len(accident)
110                 b=1
111                 accident2.append(accident[0])
112                 fi.write('0'+', '+str(accident[0][0])+', '+str(accident[0][1])+'\n')
113                 for x in range(1,lenAcc):
114                         temp=[]
115
116                         i=accident[x-1][1]
117                         j=accident[x][1]
118                         if i!=j:
119                                 temp=linier(accident[x-1],accident[x],dt)
120                                 for y in range(len(temp)):
121                                         accident2.append(temp[y])
122                                         fi.write(str(b)+', '+str(temp[y][0])+', '+str(temp[y][1])+'\n
123                                         b=b+1
124                         accident2.append(accident[x])
```

```
                        fi.write(str(b)+', '+str(accident[x][0])+', '+str(accident[x][1])+'\n')
                        b=b+1
                accident=accident2
                accident2=[]
                fi.close()

        """Tambahan dari triacc per tanggal 8-12-2017"""
        dsa=DS(accident[0][1])
        print('temperatur ke-0 dan ke-1 accident: ',accident[0][1], accident[0][1])
        tauA=dsa*accident[0][0]
        Fd=FTau(tauI)

        n=((Fd*Ff)+opf)*Fb*(Vk/Vm)
        p=n*R*accident[0][1]/Vf
        pressure.append(p)
        sigmaT=(r*p)/(2*do)
        SigmaT.append(sigmaT)

        a1=sigmaT/sigma0
        a=pow(a1,m)
        b=math.exp(-math.log(2)*a)
        phi0=1-b
        phi=phi0
        phiphi=phi0          #akumulasi phi
        phi12=phi
        print('phi0='+str(phi0))

        PHI=[]
        PHI1=[]
        PHI1aw=[]
        PHI1m1=[]
        PHI1m2=[]
        PHI1uj=[]
        PHI2=[]
        PHI12=[]
        PHI.append(phi)
        PHI12.append(phi12)
        """batas update per tanggal 8-12-2017"""

        fi=file('dsa','w')
        PHI=[]
        PHI1=[]
        PHI2=[]
        PHI12=[]

        print('array waktu accident = ', len(accident))
        print('Waktu irradiasi= ', tb/(24*3600))
        print('Temp. rerata irradiasi= ', Tb, dt)

        for i in range(1,len(accident)):
                Tuj=accident[i][1]          #temperatur pada ujung mesh, titik ke-i
                Taw=accident[i-1][1]        #temperatur pada awal mesh, titik ke (i-1)
                if i==1:
                        Tm=(Tuj+Tb)/2
                else:
                        Tm=(Tuj+Taw)/2

                taw=accident[i-1][0]
                tm=(accident[i-1][0]+accident[i][0])/2
                tuj=accident[i][0]

                dsaaw=DS(Taw)
                dsam1=DS(Tm)
                dsam2=DS(Tm)
                dsauj=DS(Tuj)

                tauAaw=dsaaw*taw
                if tauAaw==0:
                        Fdaw=FTau(tauI)
                else:
```

```
                        Fdaw=(((tauI+tauAaw)*FTau(tauI+tauAaw))-(tauAaw*FTau(tauAaw)))/tauI

                tauAm1=dsam1*taw
                if tauAm1==0:
                        Fdm1=FTau(tauI)
                else:
                        Fdm1=(((tauI+tauAm1)*FTau(tauI+tauAm1))-(tauAm1*FTau(tauAm1)))/tauI

                tauAm2=dsam2*tuj
                Fdm2=(((tauI+tauAm2)*FTau(tauI+tauAm2))-(tauAm2*FTau(tauAm2)))/tauI

                tauAuj=dsauj*tuj
                Fduj=(((tauI+tauAuj)*FTau(tauI+tauAuj))-(tauAm1*FTau(tauAuj)))/tauI

                opfaaw=OPFAccident(Tb,tb,Taw)
                opfam1=OPFAccident(Tb,tb,Tm)
                opfam2=OPFAccident(Tb,tb,Tm)
                opfauj=OPFAccident(Tb,tb,Tuj)
                print(Fdaw,opfaaw,Vk,Taw,Vf)

                paw=tekanan(Fdaw,opfaaw,Vk,Taw,Vf)
                pm1=tekanan(Fdm1,opfam1,Vk,Tm,Vf)
                pm2=tekanan(Fdm2,opfam2,Vk,Tm,Vf)
                puj=tekanan(Fduj,opfauj,Vk,Tuj,Vf)

                sigmaTaw=(r,paw,do,taw,Taw)
                sigmaTm1=(r,pm1,do,tm,Tm)
                sigmaTm2=(r,pm2,do,tm,Tm)
                sigmaTuj=(r,puj,do,tuj,Tuj)

                maw=weibullParam(Tb,Taw,taw,m00,gamma)
                phiaw=phi1(sigma0, sigmaTaw, maw)

                mm1=weibullParam(Tb,Tm,tm,m00,gamma)
                phim1=phi1(sigma0, sigmaTa1, mm1)

                mm2=weibullParam(Tb,Tm,tm,m00,gamma)
                phim2=phi1(sigma0, sigmaTm2, mm2)

                muj=weibullParam(Tb,Tuj,tuj,m00,gamma)
                phiuj=phi1(sigma0, sigmaTuj, muj)
                """batas update per tanggal 14-02-2018"""

                if i==1:
                        phi12=phi12+phiaw

                if (phim2-phim1)>0:
                        phi12=phi12+(phim2-phim1)
                else:
                        phi12=phi12

                fphi.write(str(i)+','+str(accident[i][0]/(24*3600))+','+str(phiaw)+','+str(phim1)+','

                fparam.write(str(i)+','+str(accident[i][0]/(24*3600))+','+str(Taw)+','+str(Tm)+','+st
        fphi.close()
        fparam.close()
```