Utförd av:
Erik Kempe
Viktor Nerlander

Toriums påverkan på säkerhetsparametrar och konverteringskvot i Ringhals 4
Kärnkraft teknik och system

1 Abstract

In this report the reactor Ringhals 4 (R4) has been simulated nummericaly in Pyhton with diffrent mixtures of thorium and uranium. The purpose of these simulations was to study the effect on the conversion factor, moderator temperature feedback and fuel temperature feedback with differing ammounts of thorium mixed with uranium. The results found in this report yield that a higher thorium share of the total fuel composition is better for both investigated temperature coefficients and for the conversionrate. The share of delayed neutrons is slightly reduced by using thorium.

Innehåll

2 Introduktion

Torium har länge varit av intresse för kärnkraftsindustrin. Till att börja med så har ämnet en del intressant fysiska egenskaper. Ämnets oxiderade tillstånd, ThO_2, har en smältpunkt på $3300C$, bättre värmeledningsförmåga än UO_2 och expanderar mindre än UO_2. Dessa är tre väldigt eftertraktade egenskaper inom härdfysik eftersom att; man inte vill att bränslet ska smälta, man vill att värme överförs effektivt från bränsle till kylmedie och att ett bränsle som inte expanderar bättre kan hålla inne fissionsgaser [2].

Det finns betydligt mer torium än uran i jordskorpan. I vanlig gruvdrift för sällsynta jordartsmetaller leter man ofta efter monazit och då får man ofta torium som biprodukt och därför finns det redan etablerade processflöden av ämnet om intresse för bränsletillverkning skulle uppstå. Då torium återfinns koncentrerad kring monazit-mineraler är det även mer effektivt och därmed billigare att framställa oxiden jämfört med UO_2 [1]. Det är dock generellt sätt dyrare att använda torium som bränsle på grund av att man behöver framställa plutonium för att hjälpa till med fissionsprocessen [2].

Reaktorer som använder torium tillverkar betydligt mindre långlivat avfall till skillnad från reaktorer med uran som bränsle [5]. Innan ämnen som härstammar från torium-232 hinner bli tyngre varianter av plutonium har de passerat uran-233, uran-235 och plutonium-239 som alla har hög sannolikhet att fissioneras. Ämnena som härstammar från uran-238 får bara chansen att fissionera när de blir till plutonium-239 till skillnad från ämnen som härstammar från torium-232. Infångningen för torium-232 och uran-238 kan förstås som en motorväg där atomer kör framåt i takt med att de plockar upp neutroner och blir till tyngre grundämnen. Längs motorvägen finns det ibland avfartsramper där en viss andel av bilarna (ämnena) kör av (fissioneras). Thorium-232 har fler avfartsramper än uran-238 på vägen till att bli tyngre transuraner.

Ofta nämns att torium även är säkrare med avseende på kärnvapenspridning. Till att börja med så finns det ingen fissil isotop av torium vilket gör att man inte kan framställa vapnet genom anrikning. I framställningen av uran-233 tillverkas en liten andel uran-232 som har väldigt farliga gamma-strålar i sönderfallskedjan vilket gör materialet svårt att hantera och lätt att upptäcka [2].

I den här rapporten har reaktorn Ringhals 4 (R4) simulerats. R4 har simulerats i syfte att se hur konversionskvoter och säkerhetsmekanismer påverkas om en del torium-232 ersätter en viss del uran-238 i bränslet. Uran-238 och torium-232 är fertila isotoper vilket innebär att de kan bli till fissila isotoper vid infångning av neutroner. Uran-238 kan bilda neptunium-239 genom infågning av en neutron som sedan sönderfaller till plutonium-239. Torium-232 kan bilda protaktinium-233, som sedan sönderfaller till den fissila isotopen uran-233[8].

I ett kärnkraftverk brukar man tala om 3 viktiga återkopplingar; moderatortemperatur, bränsletemperatur och void[4]. R4 är en PWR[6] vilket innebär att vattnet har dubbelt så högt tryck jämfört med en BWR vilket leder till att ingen void skapas. Det är de bara återkopplingar med avseende på moderatortemperatur och bränsletemperatur som är relevanta här. Dessa återkopplingar ska vara negativa eftersom de motverkar förändringar i system vilket hjälper till att hålla reaktorn stabil. Moderatortemperaturåterkopplingen fungerar genom att en ökad temperatur leder till högre densitet av vattnet. Densiteten används för att räkna ut ett volymförhållande mellan moderator och bränsle vilket i sin tur används för att beräkna parametrar i multiplikationsfaktorn. Om volymförhållandet blir mindre resulterar detta i sämre moderering vilket ökar resonansinfångningen och på samma sätt fungerar bränsletemperaturåterkopplingen fast med bränsletemperatur som variabel. En grad Celsius ökning i bränslet leder till att resonanspassagefaktorn, p, ändras vilket påverkar kriticiteten k. Hur mycket k ändras av p och i vilken riktning temperaturförändringar i bränsle och moderator påverkar k beksrivs av dessa återkopplingskoefficienter som detta projekt har beräknat.

Fördröjda neutroner är nödvändiga för att kunna styra reaktorn då de ökar medellivslängden på härdens neutroner. De uppstår från fissionsprodukter som neutronemiterar och eftersom olika fissila bränslen har olika distribueringar av fissionsprodukter skiljer sig andelen fördröjda neutroner mellan bränslen.

3 Metod

Metodiken som användes i simuleringarna var iterativa numeriska beräkningar som utfördes i programmeringsspråket Python. De iterativa stegen var baserade på ekvationerna som finns i avsnitt (3.3) i rapporten. Med toriumhalt i denna rapport menas förhållandet mellan torium-232 och uran-238, andelen uran-235 hålls konstant på 3% av den totala mängden bränsle.

3.1 Storheter

I detta avsnitt presenteras de storheter som har använts i beräkningarna.

Bränslestavsradie: $R_u = 0.41[cm]$ [6]

Densitet UO_2: $\rho_{UO2} = 10.97\ [g/cm^3]$ [10]

Densitet ThO_2: $\rho_{ThO2} = 10.0\ [g/cm^3]$ [9]

Termiskeffekt: $P = 3292[MW]$ [6]

Värmekapacitet:$Cp_{UO2} = 0.4[kJ/kg]$

Makr. tvärsnitt absoption: $\Sigma_a\ [cm^-1]$

Makr. tvärsnitt fission: $\Sigma_f\ [cm^-1]$

Makr. tvärsnitt spridning: $\Sigma_s\ [cm^-1]$

Moderatortemeratur: $T_m\ [K]$

Bränsletemperatur: $T_u\ [K]$

Ressonanspassagefaktor: p

Läckagefaktor: $P_s = 0.97$ [4]

Log. medelenergiförlusten: $\xi = 0.91$ [4]

Resonansintegral: $\sigma_{I,abs}\ [barn]$

Yta kring bränslekutsen: $S\ [cm^2]$

Volymbränsle: V_u

Volymmoderator: V_m

Kärntäthet: N

Konversionskvot: C

Snabba fissionsfaktorn: $\epsilon = 1.06$ [4]

Anrikning: $e = 0.03$

Frigjorda neutroner vid fission U233: $\nu_{U233} = 2.49$ [3]

Frigjorda neutroner vid fission U235: $\nu_{U235} = 2.436$ [3]

Frigjorda neutroner vid fission Pu239: $\nu_{Pu239} = 2.884$ [3]

Mikroskopiskt absorptionstvärsnitt Th232: $\sigma_{a,Th232} = 7.3[barn]$ [3]

Mikroskopiskt absorptionstvärsnitt U235: $\sigma_{a,U235} = 673.6[barn]$ [3]

Mikroskopiskt absorptionstvärsnitt U233: $\sigma_{a,U233} = 556[barn]$ [3]

Mikroskopiskt absorptionstvärsnitt Pu239: $\sigma_{a,Pu239} = 1382[barn]$ [3]

Mikroskopiskt absorptionstvärsnitt U238: $\sigma_{a,U238} = 2.6[barn]$ [3]

3.2 Antaganden

I asvnittet antaganden presenteras antaganden om modellen. Sant för alla nästkommande modelleringar är att reaktorn antogs vara ett globalt system utan lokala variationer.

3.2.1 Bibehållen kedjereaktion

Kedjereaktionen antogs vara bibehållen med en kriticitet = 1 utan att en fullständig neutronekonomi har simulerats.

3.2.2 Snabba fissionsfaktorn

I simuleringarna har snabba fissionsfaktorn antagits ha ett konstant värde på $\epsilon = 1.06$ som är ett vanligt värde för en lättvattenraktor med uran som bränsle [4]. Snabba fissionsfaktorn förändras inte när mängden uran-238 förändras i simuleringen.

3.2.3 Konversionskvot och isotoper

När konversionskvoten beräknades sattes vissa storheter till konstanta värden för att isolera de storheter som var av intresse. Återkoppling från moderatorn var inte önskvärt när konversionskvoten studerades därav sattes den till ett konstant värde likt de som erhållits från specifikationen. Bränsletemperaturen hölls också konstant eftersom att vanlig kylkapacitet av härden antas. Hänsyn har

endast tagits till uran-233 och plutonium-239 och andra fissila isotoper som skulle kunna skapas i reaktorn har försummats. Reaktorn kördes i 18 månader. Kärntätheter, N, räknades ut kontinuerligt i modellerna och det finns alltid data på hur mycket som finns av varje ämne. De ämnen som räknas ut är torium-232, protaktinium-233, uran-233, uran-235, uran-238 och plutonium-239. Antalet isotoper av diverse ämnen beräknades genom att ställa upp flöden som beskrev hur mycket av det ämnet som uppstod och hur mycket av ämnet som försvann. Uran-233 hade exempelvis ett positivt och ett negativt tillflöde. Det kunde skapas uran-233 genom sönderfall från protaktinium-233 och mängden uran-233 kunde minska genom att kärnor av ämnet klyvs.

3.2.4 Protaktinium

Protaktinium-233:s halveringstid var 27 dagar i simuleringarna[8]. Innan protaktinium-233 sönderfall var det möjligt att protaktinium-233 med sitt relativt stora tvärsnitt (900 barn)[3] fångade in en neutron och istället bildade protaktinium-234 som därefter sönderfall till den icke fissila isotpen uran-234. Antagandet i simuleringarna var att protaktinium-233 inte fångade in några neutroner utan enbart sönderfall till uran-233.

3.2.5 Moderatortemperaturåterkoppling och bränsletemperatursåterkoppling

För att modellera moderatorns förmåga att reglera reaktorn så hålls bränsletemperaturen konstant medan en rad olika temperaturer på vattnet används för att beräkna p och slutligen k. För bränsletemperaturen hölls vattnets temperatur och således moderatorns densitet vid ett konstant värde.

3.2.6 Fördröja neutroner

Vid beräkning av antal fördröjda neutroner är både bränsletemperaturen och moderatortemperaturen konstant. Genom att köra reaktorn i 18 månader och kontinuerligt beräkna andelen fördröjda neutroner beräknades ett medelvärde för den totala mängd fördröjda neutroner som härden innehöll.

3.3 Ekvationer

Här kommer ekvationer främst för resonanspassagefaktorn och konversionskvoterna redovisas. Samtliga makroskopiska tvärsnitt beräknas enligt ekvation(1), som multiplicerar det mikroskopiska tvärsnittet för isotopen σ med antalet kärnor N av samma isotop.

$$\Sigma_i = \sigma_i \cdot N_i \tag{1}$$

3.3.1 Resonaspassagefaktorn

Det globala tvärsnittet som används för att räkna ut resonanspassagefaktorn p, där T_U är bränslets temperatur, kan skrivas enligt ekvation(2), där B_1 beräknas enligt ekvation(3).

$$\sigma_{I,abs}(T_u) = \sigma_{I,abs}(300K) \cdot (1 + B_1(\sqrt{T_U} - \sqrt{300})) \tag{2}$$

$$B_1 = 6.1 \cdot 10^{-3} + \frac{0.94 \cdot 10^{-2}}{R_{uu}} \tag{3}$$

Resonansintegral för U-238 där R_u är bränslestavens radie och ρ_U beräknades enligt ekvation(4).

$$\sigma_{I,abs}(300K) = 3.0 + \frac{39.6}{\sqrt{R_u \cdot \rho_U}}[barn] \tag{4}$$

Resonansintegral för Th-232 där S är ytan kring bränslekutsen och m är massa Thorium (kvoten ska enligt Weitman ligga på 0.15-0.65 cm^2/g) [7] och beräknades enligt ekvation(5) .

$$\sigma_{I,abs}(300K) = 6.5 + \frac{15.6}{\sqrt{S/m_{ThO_2}}}[barn] \tag{5}$$

Det går att räkna ut det globala tvärsnittet för torium-232, som beror på resonansintegralen, genom att beräkna kvoten av resonansintegralerna så kan man ansätta att den kvoten är hur mycket det globala tvärsnittet för torium skiljer sig från uran. Bränslekutsdiametern, R_u, för kutsarna som används i R4 är 4.10 mm och densiteten för bränslet, ρ, är 10.0 g/cm^3 och 10.97 g/cm^3 för torium

och uran respektive. Integralerna blir då 21.7 för uran och 28.8 för torium. Vi kan då beräkna det globala tvärsnittet av torium enligt ekvation(6).

$$\sigma_{I,abs,Th}(T_u) = \sigma_{I,abs,U}(T_u)\frac{28.8}{21.7} = \sigma_{I,abs,U}(T_u) \cdot 1.33 \tag{6}$$

I och med resultat från ekvation (6), kärntätheten N, bränslevolymen V_u, medellogaritmiska energiförlusten ξ, makroskopiska spridningstvärsnittet av moderatorn Σ_S och moderatorvolymen V_m kunde en gemensam resonanspassagefaktor för torium-232 och uran-238, p, räknas ut enligt ekvation(7):

$$p(\sigma_{I,abs}, N) = exp(-(1-e)\frac{N\sigma_{I,abs}V_u}{(\xi \cdot \Sigma_S)V_m}) \tag{7}$$

3.3.2 Konversionskvot

En konversionskvot är hur många fissila kärnor som skapas för varje fissil kärna som förbrukas. Konversionskvoterna för uran-238 och torium-232 har beräknats enligt ekvation(8-9). Den första termen i bägge ekvationerna(8-9) beräknar hur många termiska neutroner som fångas i uran-238 respektive torium-232. Den andra termen i de bägge ekvationerna(8-9) beräknar hur många neutroner som fångas in i resonanserna i uran-238 respektive torium-232, då konversionkvoterna baseras på ett gemensamt uträknat p viktas hur många neutroner som fastnar i respektive ämne med antalet kärnor av respektive ämne.

$$C_U = \frac{\Sigma_a(N_{U238})}{\Sigma_a(N_{U235}) + \Sigma_a(N_{U233}) + \Sigma_a(N_{Pu239})} + \frac{\Sigma_f(N_{U233}) \cdot \nu_{U233} + \Sigma_f(N_{U235}) \cdot \nu_{U235} + \Sigma_f(N_{U239}) \cdot \nu_{Pu239}}{\Sigma_a(N_{U235}) + \Sigma_a(N_{U233}) + \Sigma_a(N_{Pu239)}} \cdot \varepsilon \cdot P_S(1-p \tag{8}$$

$$C_{Th} = \frac{\Sigma_a(N_{Th232})}{\Sigma_a(N_{U235}) + \Sigma_a(N_{U233}) + \Sigma_a(N_{Pu239})} + \frac{\Sigma_f(N_{U233}) \cdot \nu_{U233} + \Sigma_f(N_{U235}) \cdot \nu_{U235} + \Sigma_f(N_{U239}) \cdot \nu_{Pu239}}{\Sigma_a(N_{U235}) + \Sigma_a(N_{U233}) + \Sigma_a(N_{Pu239)}} \cdot \varepsilon \cdot P_S(1-p \tag{9}$$

4 Resultat

I avsnittet resultat presenteras resultaten för konversionskvoten med avseende på toriumhalt samt återkopplingskoeffecienterna med avseende på toriumhalten och andelen fördröjda neutroner med avseende på toriumhalten.

4.1 Moderatortemperaturkoeffecient

Vid 0% torium så förväntas samma moderatoråterkoppling vara den samma som för enbart uran som förväntas ligga mellan $-(5-25)$ pcm/C [4]. I takt med att mängden torium-232 ökar till 100% jämfört med uran-238 kommer moderatortemperaturkoffecienten att bli mer negativ, med ett slutvärde strax under -36pcm/C som ses i figur(1).

4.2 Bränsletemperaturkoeffecienten

Bränsletemperaturkoeffecienten beskrivs som funktion av toriumhalt i figur (2). Bränsletemperaturenkoeffecienten kommer att minska i takt med att toriumhalten ökar ända upp till 100%. Bränsletemperaturkoeffecienten kommer att minska från -2.71 pcm/C till -2.94 pcm/C .

4.3 Konversionskvot

Konversionskvoten kommer att öka från cirka 0.4 till 0.7 i takt med att toriumhalten ökar från 0 % till 100 %, vilket kan ses i figur(3). Konversionkvoten för torium-232 ökar snabbare än vad den minskar för uran-238 vilket innebär att den totala konversionen kommer att öka när toriumhalten ökar.

4.4 Fördröjda neutroner

Figur (4) beskriver relationen mellan toriumhalt och andel fördröjda neutroner. Andelen fördröjda neutroner kommer att minska från kring 0.00549 till kring 0.00543 i takt med att andelen torium ökar till från 0 % till 100 %.

[scale=0.85]moderator.pdf

Figur 1: Moderatortemperaturkoeffecient beroende på toriumhalt

[scale=0.85]btemp.pdf

Figur 2: Bränsletemperaturkoefficient beronde på toriumhalt.

[scale=0.95]1h.pdf

Figur 3: konversionskvoten baserat på mängden torium-232 jämför med mängden uran-238. Den totala konversionskvoten ökar i takt toriumhalten medan andelen U238 minskar.

[scale=0.95]beta.pdf

Figur 4: Andel fördröjda neutroner som funktion av toriumhalt.

5 Diskussion

I detta avsnitt diskuteras resultaten och antaganden som gjordes för simuleringarna.

5.1 Snabba fissionsfaktorn

Då snabba fissionsfaktorn har modellerats som ett konstant tal kommer det att påverka resultatet eftersom torium-232 och uran-238 inte kommer att ha exakt samma fissionstvärsnitt vid höga energinivåer.

5.2 Moderatortemperaturåterkoppling

I figur(1) syns det att moderatorremperaturkoeffecienten blir mer negativ när toriumhalten ökar. Detta beror på att torium-232 har ett högre infångningstvärsnitt och ett högre globalt tvärsnitt än uran-238 vilket leder till att fler neutroner fångas i resonanserna. Att moderatorremperaturkoeffecienten blir mer negativ innebär att reaktorn blir mer motståndskraftig mot förändringar vilket är positivt ur ett säkerhetsperspektiv.

5.3 Bränsletemperaturåterkoppling

Bränsletemperaturåterkopplingen kommer att få ett mer negativt värde i takt med att toriumhalten ökar. Då torium-232 har ett högre infångningstvärsnitt och ett högre globalt tvärsnitt betyder det att fler neutroner kommer att fastna resonanserna vilket leder till färre klyvningar i jämförelse med när halten uran-238 är högre. Resultatet är bra ur ett säkerhetsperspektiv då det gör reaktorn mer motståndskraftig mot förändringar.

5.4 Konversionskvot

Konversionskvoten kommer att öka i takt med att toriumhalten ökar enligt figur(3). Detta betyder att reaktorn kommer ha en högre utbränning ju mer torium man använder förutsatt att man kan hålla reaktorn kritisk. En högre konversionskvot var ett förväntat resultat redan när det globala tvärsnittet och infångningstvärsnittet konstaterats som större än för uran.

5.5 Fördröjda neutroner

Andelen fördröjda neutroner benämns som β och är plottat mot toriumhalt i figur (4). Uran-233 har 0.0026, Uran-235 har 0.0065 och plutonium-239 har 0.0021 andel fördröjda neutroner vid fissionering. Att andelen fördröjda neutroner minskar vid ökande toriumhalt trots att uran-233 har fler fördröjda neutroner än plutonium-239 beror på att konversionskvoten även ökar och skapar fler uran-233 kärnor vilket kommer att minska totala mängden fördröjda neutroner. I slutet avtar kurvans lutning i figur(4) något då vi inte får någon väsentlig ökning av uran-233 per procent ökning av toriumhalten.

5.6 Diskussion av antaganden

Det antas att all protaktinium-232 kärnor som skapas kommer att sönderfalla till uran-233, detta är nödvändigtvis inte sant. Protaktinium-232 har ett infångningstvärsnitt på 900 barn och en halveringstid på cirka 27 dagar. Protaktinium-232 kan fånga in neutroner och sedan söndefalla till uran-234 som inte är en fissil isotop, då beräkningarna som har genomförts inte har tagit hänsyn till hur detta kommer att påverka olika parametrar i verkligheten kommer resultatet troligtvis att skilja sig från de beräknade. I beräkningarna tas det inte hänsyn till att fissila isotoper som plutonium-239 och uran-235 kan fånga in neutroner och bli icke fissila som förstås skulle påverka resultatet. Att alla isotoper inte är med i beräkningarna kommer exempelvis påverka konversionskvoten eftersom antalet fissila och icke fissila kärnor kommer att skilja sig mot verkligheten. Hur återkopplingkoffecienterna förändras med toriumhalten påverkas av toriumhalten kommer även att skilja sig från verkligheten då dessa bygger på infågning av neutroner i olika kärnor och alla olika kärnor inte har räknats med. Andelen fördröjda neutroner kan även komma att skilja sig ifrån verkligheten då olika kärnor har olika andel fördröjda neutroner.

5.7 Fördröjda neutroner jämfört med återkoppling

I figur(2) och figur(1) syns det att återkopplingskoeffecienterna förbättras med en ökande mängd toriumhalt, vilket gör reaktorn mer motståndskraftig mot förändringar. I figur(4) syns det att andelen fördröja neutroner minskar med en ökande toriumhalt, det ger en smalare marginal mellan kriticitet och promptkriticitet och kan innebära en försämring av säkerheten. Återkopllingskoffecienterna kan bidra till en förbättrad säkerhet och en minskande del fördröjda neutroner kan bidra till en försämrad säkerhet. Vad som väger tyngst säkerhetsmässigt mellan återkopllingskoffecienterna och en minskad mängd fördröjda neutroner behövs undersökas vidare för att kunna ge ett entydligt svar på hur reaktornssäkerhet påverkas med olika toriumhalter.

5.8 Bibehålla kedjereaktion

I beräkningarna har inte neutronekonomin simulerats fullständigt vilket innebär att det inte går att säga om kedjereaktionen i reaktorn kommer att bibehållas med en ökande toriumhalt. Hur neutronekonomin påverkas behöver undersökas för att kunna säga om det är möjligt att ha en viss toriumhalt i bränslet.

6 Slutsats

Säkerhetsmässigt kommer både moderatortemperaturkoeffecienten och bränsletemperaturkoeffecienten att förändras på ett önskvärt sätt och göra reaktorn mer motståndskraftig. Andelen fördröjda neutroner kommer att förändras på ett icke säkerhetsmässigt önskvärt sätt då andelen minskar. Konversionskvoten kommer att öka vilket är positivt bränslet får en högre utbränning. Tre parametrar kommer att förändras på ett önskvärt sätt och en parameter, andelen fördröjda neutroner kommer att förändras på ett icke önskvärt sätt och därav rekommenderas det att torium inte tillsäts i R4 utan att det krävs vidare undersökning på hur mycket de olika säkerhetsparametrarn kommer att påverka reaktornsäkerhetn i jämförelse med varandra.

7 Referenser

[1] International Atomic Energy Agency. "Thorium fuel cycle — Potentialbenefits and challenges". I: IAEA-TECDOC-1450 (2005), s. 1–3.

[2] World Nuclear Association. *Thorium*. URL: https://www.world-nuclear.org/information-library/current-and-future-generation/thorium.aspx.

[3] National Nuclear Data Center. URL: https://www.nndc.bnl.gov/sigma/.

[4] KSU. *Reaktorfysik - Kraftindustrins grundutbildningspaket*. 2015.

[5] Marvin Schaffer. "Abundant thorium as an alternative nuclear fuel: Important waste disposal and weapon proliferation advantages". I: *Energy Policy* 60 (sept. 2013). DOI: 10.1016/j.enpol.2013.04.062.

[6] Vattenfall. "Teknisk data Ringhals". I: (2015).

[7] J Weitman. "The Effective Resonance Integralof Thorium Oxide Rods". I: (1962).

[8] Wikipedia. *Thorium*. URL: https://en.wikipedia.org/wiki/Thorium.

[9] Wikipedia. *Thorium dioxide*. URL: https://en.wikipedia.org/wiki/Thorium_dioxide.

[10] Wikipedia. *Uranium dioxide*. URL: https://en.wikipedia.org/wiki/Uranium_dioxide.

8 Appendix

8.1 R4.py

```
import math
import matplotlib.pyplot as plt
from pyXSteam.XSteam import XSteam
import numpy as np
from sklearn.linear_model import LinearRegression
steamTable = XSteam(XSteam.UNIT_SYSTEM_MKS) # m/kg/sec/ C /bar/W


class Reaktor:

    def __init__(self, termiskEffekt, drifttryck, n_bransleelement, branslevikt, P, anrikning, N_Th232_N_U238):

        # Reaktorparametrar
        self.termiskEffekt = termiskEffekt  # [W]
        self.prev_effekt = termiskEffekt
        self.drifttryck = drifttryck  # [MPa]
        self.branslevikt = branslevikt  # Vikt per bransleelement [kg]
        self.bransleelement = n_bransleelement
        self.stav_per_ele = 264
        self.n_knippen_styr = 48
        self.stav_per_knippe = 24
        self.fuel_w = self.branslevikt * self.bransleelement
        self.radie = 0.41  # cm
        self.B1_th = 0.0082768
        self.fuel_T = 800  # K
        self.vm_vu = 3.02  # volymf rh llande fr. test.py
        self.xi = 0.91  # KSU s.82
        self.stavar = 264
        self.langd = 3.42 * 100  # utr knat med data fr n specifikationen\ bransleelement densitet & n_stavar
        self.thermCon = 0.024  # W/cm*K
        self.tot_kyl_flow = 142223  # kg/s
        self.sek_kontakt = self.langd/100/3  # sekunder som kylvattnet kommer ha kontakt med samma stav
        self.l_U = 0.08488246103215057  # utr knad i test.py
        self.l_Th = 0.048023252115223215  # utr knad i test.py
        self.l_Pu = 0.0324758978318962  # utr knad i test.py

        self.P = P  # Ickel ckagefaktor
        self.anrikning = anrikning  # Anrikningsgrad
        self.N_Th232_N_U238 = N_Th232_N_U238  # F rh llandet mellan thorium-232 och uran-238
        self.rho_UO2 = 10.97  # Densitet uranoxid g*cm^-3
        self.rho_ThO2 = 10  # Densitet Thouriumoxid g*cm^-3
        self.nu_U = 2.436  # Antalet neutroner som frig rs vid fission
        self.nu_Th = 2.497
        self.nu_Pu = 2.884
        self.epsilon = 1.06  # Snabba fissionsfaktorn
        self.k = 1
        self.reak = 0
        self.timeStep = 3600
        self.vatten_temp = 282.7  # kylvatten temp (t_in)
        self.U_vatten = 14.3  # W/cm/K
        self.c_p_UO2 = 0.4 * 1000  # J/(kg K)
        self.c_p_H2O = 0.419 * 1000  # J/(kg K)
        self.rho_set_w = steamTable.rho_pt(self.drifttryck*10, self.vatten_temp)
        self.nu_233 = 2.49
        self.nu_239 = 2.93


        # Antal atomk rnor
        self.N_Pa233 = 0
        self.N_Pu239 = 0
        self.N_U233 = 0
        self.N_U235 = self.calc_atom_karnor(self.rho_UO2, 235 + 2*16)*self.anrikning
        self.N_U238 = self.calc_atom_karnor(self.rho_UO2, 235 + 2 * 16) * (1-self.anrikning) * (1-self.N_Th232_N_U238)
        self.N_Th232 = self.calc_atom_karnor(self.rho_UO2, 232 + 2 * 16) * (1-self.anrikning) * (self.N_Th232_N_U238)

```

```
        # Halveringstider
        self.halveringstid_Pa233 = 26.975 * 24 * 3600  # [s]

        # Tv rsnitt
        self.barn = 1E-24
        self.sig_235_f = 576*self.barn
        self.sig_235_g = 97.6 * self.barn
        self.sig_239_f = 801 * self.barn
        self.sig_239_g = 281 * self.barn
        self.sig_238_g = 2.6 * self.barn
        self.sig_238_f = 1.76E-5*self.barn
        self.sig_232_a = 7.3*self.barn
        self.sig_233_f = 514*self.barn
        self.sig_233_g = 42*self.barn
        self.sig_233_a = self.sig_233_f + self.sig_233_g
        self.sig_235_a = self.sig_235_f + self.sig_235_g
        self.sig_238_a = self.sig_238_f + self.sig_238_g
        self.sig_239_a = self.sig_239_f + self.sig_239_g
        self.sig_w_a = 0.33344 * 2 * self.barn  # 2H #https://www.ncnr.nist.gov/staff/hammouda/distance_learning/chapter_9.pdf
        self.sig_w_s = 56.08 * self.barn

    def calc_konversion(self):  # vi kommer ha en f r b da
        self.c_U = (self.sig_238_a * self.N_U238) / (self.sig_235_a * self.N_U235 + self.sig_233_a * self.N_U233 + self.sig_239_a * self.N_Pu239) + (((self.sig_233_f * self.N_U233 * self.nu_233 + self.sig_235_f * self.N_U235 * self.nu_U + self.sig_239_f * self.N_Pu239 * self.nu_239) /(self.sig_235_a * self.N_U235 + self.sig_233_a * self.N_U233 + self.sig_239_a * self.N_Pu239)) * self.epsilon * self.P * (1 - self.p)) * ((self.N_U238) / (self.N_Th232 + self.N_U238))

        self.c_Th = (self.sig_232_a * self.N_Th232) / (self.sig_235_a * self.N_U235 + self.sig_233_a * self.N_U233 + self.sig_239_a * self.N_Pu239) + (((self.sig_233_f * self.N_U233 * self.nu_233 + self.sig_235_f * self.N_U235 * self.nu_U + self.sig_239_f * self.N_Pu239 * self.nu_239) /(self.sig_235_a *  self.N_U235 + self.sig_233_a * self.N_U233 + self.sig_239_a * self.N_Pu239)) * self.epsilon * self.P * (1 - self.p)) * ((self.N_Th232) / (self.N_Th232 + self.N_U238))

    def calc_p(self):  # Ber kning av resonaspassagefaktor
        B1_U = 6.1 * 10 ** -3 + 0.94 * 10 ** -2 / (self.radie * self.rho_UO2)
        B1_Th = self.B1_th  # utr knat med integralf rh llande
        sigma_300K_U = (3.0 + 39.6 / math.sqrt(self.radie * self.rho_UO2)) * 10 ** -24  # cm^2
        S = self.radie * 2 * math.pi * 1  # yta br nslekuts
        m = self.radie ** 2 * math.pi * 1 * self.rho_ThO2  # volym * densitet = massa
        sigma_300K_Th = (6.5 + 15.6 / math.sqrt(S/m)) * 10 ** -24  # cm^2
        sigma_fuel_T_U = sigma_300K_U * (1 + B1_U * (math.sqrt(self.fuel_T) - math.sqrt(300)))
        sigma_fuel_T_Th = sigma_300K_Th * (1 + B1_Th * (math.sqrt(self.fuel_T) - math.sqrt(300)))

        self.p = math.exp((-(1 - self.anrikning) * self.N_U238 * sigma_fuel_T_U * 1 / self.vm_vu /(self.xi * self.sig_w_s * self.calc_atom_karnor(self.rho_w * 1E-3, 18))) + -(1 - self.anrikning) * self.N_Th232 * sigma_fuel_T_Th * 1 / self.vm_vu /(self.xi * self.sig_w_s * self.calc_atom_karnor(self.rho_w * 1E-3, 18)))

        self.p_U = math.exp(-(1 - self.anrikning) * self.N_U238 * sigma_fuel_T_U * 1/self.vm_vu /(self.xi * self.sig_w_s * self.calc_atom_karnor(self.rho_w*1E-3, 18)))
        self.p_Th = math.exp(-(1 - self.anrikning) * self.N_Th232 * sigma_fuel_T_Th * 1/self.vm_vu /(self.xi * self.sig_w_s * self.calc_atom_karnor(self.rho_w*1E-3, 18)))

    def calc_fission(self):
        denominator_f = self.N_U235 * self.sig_235_f + self.N_Pu239 * self.sig_239_f + self.N_U233 * self.sig_233_f
        chans_235 = (self.N_U235 * self.sig_235_f) / denominator_f
        chans_233 = (self.N_U233 * self.sig_233_f) / denominator_f
        self.fission_235 = ((self.N_U235 * self.sig_235_f) / denominator_f) * self.FR * self.timeStep  # fissionerade 235
        self.fission_233 = ((self.N_U233 * self.sig_233_f) / denominator_f) * self.FR * self.timeStep  # fissionerade 235
        self.fission_239 = (1 - chans_235 - chans_233) * self.FR * self.timeStep  # fissionerade 239
        total_fission = self.fission_235 + self.fission_233 + self.fission_239
        self.N_Pu239 += total_fission * self.c_U - self.fission_239
```

```
116         self.N_Pa233 += total_fission * self.c_Th - (self.N_Pa233 * math.exp(-self.
    timeStep / self.halveringstid_Pa233))
117         self.N_U233 += - self.fission_233 + self.N_Pa233 * math.exp(-self.timeStep /
    self.halveringstid_Pa233)
118         self.N_U235 -= self.fission_235
119         self.N_U238 -= total_fission * self.c_U
120         self.N_Th232 -= total_fission * self.c_Th
121 
122         # skapade = total_fission * self.c_U + self.N_Pa233 * math.exp(-self.timeStep
     / self.halveringstid_Pa233)
123         # print(skapade / total_fission, self.c_U, self.c_Th)
124         #
125         # skapade = total_fission * self.c_U + total_fission * self.c_Th + self.
    N_Pa233 * math.exp(
126         #     -self.timeStep / self.halveringstid_Pa233)
127         # anvanda = self.fission_235 + self.fission_233 + self.fission_239
128         # print(skapade / anvanda, self.c_U, self.c_Th, self.p_U, self.p_Th)
129 
130     def calc_eta(self):  # termiska snabba fissionsfaktorn
131         den = self.N_Pu239*self.sig_239_a + self.N_U238*self.sig_238_a + self.N_U235*
    self.sig_235_a + \
132               self.N_U233*self.sig_233_a + self.N_Th232*self.sig_232_a
133         num = self.N_Pu239*self.sig_239_f*self.nu_Pu + self.N_U235*self.sig_235_f*
    self.nu_U + \
134               self.N_U233*self.sig_233_f*self.nu_Th
135         self.eta = num/den
136 
137     def calc_FR(self):  # Ber knar fissionsraten
138         self.FR = self.termiskEffekt / (3.2E-11) * self.rho_UO2/\
139                   (self.branslevikt*self.bransleelement*1000)  # Konveterar vikten
    till gram, ber knar fissionsraten
140         self.neutronflux = self.FR / (self.calc_atom_karnor(
141             self.rho_UO2, 233) * self.sig_233_f + self.calc_atom_karnor(self.rho_UO2,
     235) * self.sig_235_f)
142 
143     def calc_n_phi(self):  # Ber knar neutronfl det
144         self.n_phi = self.FR / (self.N_U235*self.sig_235_f)
145 
146     def calc_atom_karnor(self, densitet, atommassa):  # Ber knar neutronfl det
147         u = 1.66043E-24
148         N = densitet/(atommassa*u)
149         return N
150 
151     def calc_anrikning(self):  # Ber knar anrikning
152         self.anrikning = (self.N_U235 + self.N_Pu239 + self.N_U233)/(self.N_U235 +
    self.N_Pu239 + self.N_U233 +
153                                                                      self.N_U238 +
    self.N_Th232)
154 
155     def calc_reaktivitet(self):
156         self.k -= 0.31930829351276 # styrstavar
157         self.reak = (self.k - 1) / self.k
158 
159     def calc_effekt(self):
160         den = self.fission_233 + self.fission_235 + self.fission_239
161         self.l_viktad = self.l_U*self.fission_235/den + self.l_Th*self.fission_233/
    den + self.l_Pu*self.fission_239/den
162         self.termiskEffekt = self.termiskEffekt*math.exp(self.reak*self.timeStep/self
    .l_viktad)  # W
163 
164 
165     def calc_lin_heat_rate(self):
166         self.lin_Q = self.termiskEffekt/self.bransleelement/self.stavar/self.langd  #
     W/cm
167         self.tempDiff = self.lin_Q/(4 * math.pi * self.thermCon)  # slide 10 F10
168         T_yta = self.fuel_T - self.tempDiff
169         q_water = self.U_vatten * T_yta  # v rme    verfrd    till vattnet
170         heat_to_w = q_water * self.sek_kontakt
171         m = 0.42114504425  # utr knat  v rde  f  r  vatten kring br nslestaven FIXA
    DET  H  R?
172         self.t_out = heat_to_w/(m*self.c_p_H2O) + self.vatten_temp
173         self.rho_w = steamTable.rho_pt(self.drifttryck*10, (self.t_out + self.
    vatten_temp)/2)
174 
175     def calc_rho_w(self):
```

```
        self.rho_w = steamTable.rho_pt(self.drifttryck * 10, (self.t_out + self.
    vatten_temp) / 2)

    def calc_volymf(self):
        r_c = 3.355 / 2  # m, utr knad h rdradie
        r_b = self.radie / 100  # m, br nslekutsradie
        area_b = r_b ** 2 * math.pi * self.stav_per_ele * self.bransleelement
        area_s = r_b ** 2 * math.pi * self.n_knippen_styr * self.stav_per_knippe
        area_m = r_c ** 2 * math.pi - area_b - area_s
        area_m *= self.rho_w/self.rho_set_w
        self.v_b = area_b * self.langd * 1E4  # cm^3
        self.v_m = area_m * self.langd * 1E4  # cm^3
        self.vm_vu = area_m/area_b

    def calcdT_dt(self):
        self.dT_dt = ((self.termiskEffekt - self.prev_effekt) * self.timeStep) / (
    self.c_p_UO2 * self.fuel_w)
        self.fuel_T += self.dT_dt

    def calc_k(self):
        u_vikt = self.N_U238 / (self.N_U238 + self.N_Th232)
        th_vikt = self.N_Th232 / (self.N_U238 + self.N_Th232)
        #self.p = self.p_U * u_vikt + self.p_Th * th_vikt
        #self.k = self.eta * self.epsilon * (self.p_U * u_vikt + self.p_Th * th_vikt)
     * self.f * self.P
        self.k = self.eta * self.epsilon * self.p * self.f * self.P

    def calc_f(self):
        makro_b = self.N_Pu239*self.sig_239_a + self.N_U238*self.sig_238_a + self.
    N_U235*self.sig_235_a\
                  + self.N_U233*self.sig_233_a + self.N_Th232*self.sig_232_a
        makro_m = self.calc_atom_karnor(self.rho_w*1E-3, 18)*self.sig_w_a
        self.f = makro_b * self.v_b / (makro_b * self.v_b + makro_m * self.v_m)
        self.f *= 0.87


def main():
    #beta()
    #konvertering()
    #moderator()
    branslet()


vector = np.linspace(0, 1, 101)  # f r konverteringsgraden
bransletemperatur = np.linspace(780, 840, 840 - 780 + 1)
moderatortemperatur = np.linspace(282, 324, 324 - 282 + 1)
simuleringstid = np.linspace(0, 24 * 30 * 18, 24 * 30 * 18 + 1)
coef = []
data = []
data1, data2, data3 = [], [], []


def beta():
    for c, e in enumerate(vector):
        R4 = Reaktor(3292E6, 15.5, 157, 523, 0.97, 0.03, e)
        print(f'Th: {math.floor(e*100)} %')
        R4.calc_FR()
        R4.t_out = 323.9
        R4.calc_rho_w()
        R4.calc_p()
        data_asd = []
        data11, data22, data33 = [], [], []
        for count, ele in enumerate(simuleringstid):
            R4.calc_konversion()
            R4.calc_fission()
            den = R4.fission_235 + R4.fission_233 + R4.fission_239
            vikt_239 = R4.fission_239/den
            vikt_235 = R4.fission_235/den
            vikt_233 = R4.fission_233/den
            beta_U, beta_Pu, beta_Th = 0.0065, 0.0021, 0.0026
            beta_w = vikt_233*beta_Th + vikt_235*beta_U + vikt_239*beta_Pu
            data_asd.append(beta_w)
            data11.append(R4.fission_233)
            data22.append(R4.fission_235)
            data33.append(R4.fission_239)
        data1.append(np.mean(data11))
```

```
        data2.append(np.mean(data22))
        data3.append(np.mean(data33))
        data_asd = np.array(data_asd)
        data_asd = np.mean(data_asd)
        data.append(data_asd)
    plot_beta()


def konvertering():
    for c, e in enumerate(vector):
        R4 = Reaktor(3292E6, 15.5, 157, 523, 0.97, 0.03, e)
        R4.calc_lin_heat_rate()
        R4.calc_FR()
        mean1, mean2, mean3 = [], [], []
        for _ in range(1):
            R4.reak = 0
            R4.calc_p()
            R4.calc_konversion()
            R4.calc_fission()
            mean1.append(R4.c_Th + R4.c_U)
            mean2.append(R4.c_Th)
            mean3.append(R4.c_U)
        data1.append(np.mean(mean1)), data2.append(np.mean(R4.c_Th)), data3.append(np
    .mean(R4.c_U))
    plot_konvertering()


def branslet():
    for c, e in enumerate(vector):
        R4 = Reaktor(3292E6, 15.5, 157, 523, 0.97, 0.03, e)
        print(f'Th: {math.floor(e*100)} %')
        data = []
        R4.t_out = 323.9
        R4.calc_rho_w()
        R4.calc_volymf()
        for count, ele in enumerate(bransletemperatur):
            R4.fuel_T = ele
            R4.calc_p()
            R4.calc_f()
            R4.calc_eta()
            R4.calc_k()
            data.append(R4.k)
        model = LinearRegression()
        model.fit(bransletemperatur.reshape(-1, 1), data)
        coef.append(model.coef_[0]*1E5)
    plot_btemp()


def moderator():
    for c, e in enumerate(vector):
        R4 = Reaktor(3292E6, 15.5, 157, 523, 0.97, 0.03, e)
        p_t = []
        print(f'Th: {math.floor(e*100)} %')
        for count, ele in enumerate(moderatortemperatur):
            R4.t_out = ele
            R4.calc_rho_w()
            R4.calc_volymf()
            R4.calc_p()
            R4.calc_f()
            R4.calc_eta()
            R4.calc_k()
            p_t.append(R4.k)
        #plt.plot(moderatortemperatur, p_t)
        #plt.show()
        model = LinearRegression()
        model.fit(moderatortemperatur.reshape(-1, 1), p_t)
        coef.append(model.coef_[0]*1E5)
    plot_moderator()


def plot_foo():
    f = plt.figure()
    plt.plot(vector, data1, label='Fission U233')
    plt.plot(vector, data2, label='Fission U235')
    plt.plot(vector, data3, label='Fission Pu239')
    plt.title('Medelv rde antal fissioner')
```

```
    plt.ylabel('Fissioner')
    plt.xlabel('Toriumhalt')
    plt.legend()
    plt.grid()
    plt.show()
    f.savefig("fissioner.pdf", bbox_inches='tight')


def plot_beta():
    f = plt.figure()
    plt.plot(vector, data)
    plt.xlabel('Toriumhalt')
    plt.ylabel('beta')
    plt.title('Andel f rdr jda neutroner som funktion av toriumhalt')
    plt.grid()
    plt.show()
    f.savefig("beta.pdf", bbox_inches='tight')


def plot_btemp():
    f = plt.figure()
    plt.plot(vector, coef)
    plt.xlabel('Toriumhalt')
    plt.ylabel('Br nsletemperaturskoefficient [pcm/C]')
    plt.title('Br nsletemperaturens   terkoppling   som funktion av toriumhalt')
    plt.grid()
    plt.show()
    f.savefig("btemp.pdf", bbox_inches='tight')


def plot_moderator():
    f = plt.figure()
    plt.plot(vector, coef)
    plt.xlabel('Toriumhalt')
    plt.ylabel('Moderatortemperaturkoefficient [pcm/C]')
    plt.title('Moderators   terkoppling   som funktion av toriumhalt')
    plt.grid()
    plt.show()
    f.savefig("moderator.pdf", bbox_inches='tight')


def plot_konvertering():
    f = plt.figure()
    plt.plot(vector, data1, label='Total konverteringskvot')
    plt.plot(vector, data2, label='Konverteringskvot Torium')
    plt.plot(vector, data3, label='Konverteringskvot Uran')
    plt.title('Konverteringskvot som funktion av torium-halt')
    plt.xlabel('Toriumhalt')
    plt.ylabel('Konverteringskvot, c')
    plt.legend()
    plt.grid()
    plt.show()
    f.savefig("1h.pdf", bbox_inches='tight')


if __name__ == "__main__":
   main()
```

8.2 test.py

```
import numpy as np

halflife = [55.72, 22.72, 6.22, 2.3, 0.610, 0.230]
yield_fraction = [0.00052, 0.00346, 0.00310, 0.00624, 0.00182, 0.00066]
yield_fraction = [i/np.min(yield_fraction) for i in yield_fraction]

# Uran
decay_U = [0.0124, 0.0305, 0.111, 0.301, 1.14, 3.01]
tau_U = [1/i for i in decay_U]
beta_list_U = [0.000215, 0.001424, 0.001274, 0.002568, 0.000748, 0.000273]
beta_U = np.sum(beta_list_U)

# Thorium
decay_Th = [0.0126, 0.0337, 0.139, 0.325, 1.13, 2.5]
```

```
tau_Th = [1/i for i in decay_Th]
beta_list_Th = [0.000224, 0.000777, 0.000655, 0.000723, 0.000133, 0.000088]
beta_Th = np.sum(beta_list_Th)

# Plutonium
decay_Pu = [0.0128, 0.0301, 0.124, 0.325, 1.12, 2.69]
tau_Pu = [1/i for i in decay_Pu]
beta_list_Pu = [0.000073, 0.000626, 0.000443, 0.000685, 0.000181, 0.000092]
beta_Pu = np.sum(beta_list_Pu)

b = 0
for count, ele in enumerate(decay_Pu):
    x = tau_Pu[count]
    y = beta_list_Pu[count]
    b += x*y

print(b)


# l_U = 0.08488246103215057
# l_Th = 0.048023252115223215
# l_Pu = 0.0324758978318962

I_d = (1-beta_U)*10**-4 + b

#print(I_d)
```