Annals of Nuclear Energy 124 (2019) 88–97

Contents lists available at ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Development of a Molten Salt Reactor specific depletion code MODEC

Shaopeng Xia [a,b,c], Jingen Chen [a,b,c], Wei Guo [a,b,c], Deyang Cui [a,b], Jianlong Han [a,b,c], Jianhui Wu [a,b,*], Xiangzhou Cai [a,b,c,*]

[a] Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China
[b] Innovative Academies in TMSR Energy System, Chinese Academy of Sciences, Shanghai 201800, China
[c] University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Article history:
Received 24 April 2018
Received in revised form 15 August 2018
Accepted 23 September 2018

Keywords:
Molten Salt Reactor
Burnup calculation
Nonhomogeneous term
TTA
CRAM
Error analysis of ORIGEN-S

ABSTRACT

Molten Salt Reactor (MSR) has the characteristics of on-line reprocessing and continuously refueling, which bring significant differences from traditional reactors in burnup calculations. To handle its specific burnup characteristics, a Molten Salt Reactor specific depletion code - MODEC has been newly developed. MODEC implements two depletion algorithms to solve the basic burnup equations: the transmutation trajectory analysis (TTA) and the Chebyshev rational approximation method (CRAM). To simulate the on-line reprocessing, the fictive decay constant method is applied. And three different methods are implemented in MODEC to solve the nonhomogeneous burnup equations in the continuously refueling problems. Moreover, it can trace in real time the evolution of the in-stockpile nuclides which is extracted by on-line reprocessing. MODEC has three calculating modes for decay, constant flux and constant power calculations. By comparing with ORIGEN-S, the validity of performance of MODEC in conventional burnup calculations, burnup calculations with on-line reprocessing and burnup calculations with continuously refueling is proved. And a comparison of the three methods of solving nonhomogeneous burnup equations is presented and discussed. Additionally, a detailed analysis of error sources in ORIGEN-S is applied and an unpublished error source is found. Finally, a specific Monte Carlo burnup procedure for actual MSR burnup calculations is developed by coupling KENO-VI with MODEC, and the burnup benchmark of Molten Salt Fast Reactor (MSFR) is calculated to validate the specific Monte Carlo burnup procedure.

1. Introduction

Molten Salt Reactor (MSR) is one of the six candidates of Gen-IV advanced nuclear power system (U.S. DOE, 2002). The fuel in the MSR is dissolved in molten salt carrier which acts as coolant as well. The molten salt mixture circulates through the primary loop continuously, producing heat in the core and removing the heat by the external heat exchangers. Because of the unique fluid fuel form, the composition of fuel salt can be readily altered by on-line fuel salt reprocessing and continuously refueling during operation. The detailed reprocessing diagram referred to MSBR and MSFR (Nuttin et al., 2005; Mathieu et al., 2009) is shown in Fig. 1. In this reprocessing scheme, the gaseous and noble metal fission products are removed by helium bubbling, and the soluble fission products are removed by on-line reductive extraction. Besides, for a thorium-based MSR, Pa extracted from the reactor core by on-line reprocessing is stored in the stockpile for some time to let ^{233}Pa decay into ^{233}U, and the decay product ^{233}U is then reinjected into the core. Additionally, the fresh fuel should be fed continuously into the core to maintain critical operation. The on-line reprocessing and refueling system can bring good neutron economy, which makes it possible for Th/^{233}U breeding in thermal MSR. However, it causes significant differences in burnup calculation from the traditional solid fueled reactor.

The biggest difference in mathematics is that the characteristic of continuously refueling would bring a nonhomogeneous term into the conventional burnup equation. The depletion codes which can only solve the conventional burnup equations, such as WIMS (Lindley et al., 2017), DRAGON (Marleau et al., 2011) for simplified nuclides system, and CINDER (Wilson et al., 2008), DEPTH (She et al., 2013) for complicated nuclides system, cannot be applied directly to the MSR. ORIGEN2 (Croff, 1983) and ORIGEN-S (Gauld, 2011), which can solve the nonhomogeneous burnup equations, are suitable for MSR depletion calculation, while the depletion algorithm implemented in the two codes of truncation Taylor series expansion method with the separate treatment of short-lived

* Corresponding authors at: Shanghai Institute of Applied Physics, Chinese Academy of Sciences, Shanghai 201800, China.
E-mail addresses: wujianhui@sinap.ac.cn (J. Wu), caixiangzhou@sinap.ac.cn (X. Cai).

https://doi.org/10.1016/j.anucene.2018.09.032
0306-4549/

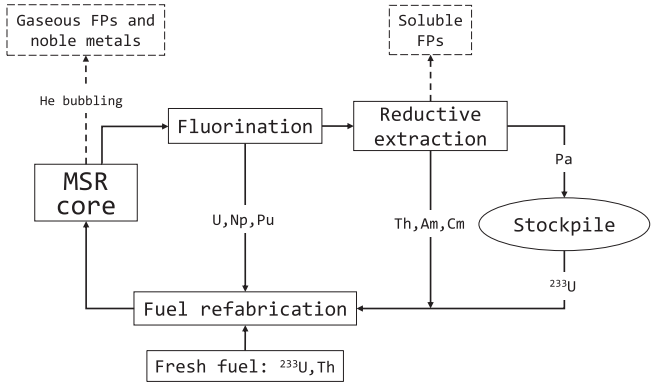

Fig. 1. Reprocessing diagram of MSR.

nuclides causes a loss of accuracy (Songqian et al., 2016). SERPENT-2 (Aufiero et al., 2013) has been recently extended and employed to investigate burnup evolution of the Molten Salt Fast Reactor (MSFR) (Merle-Lucotte et al., 2011). However, the extended SERPENT-2 cannot directly solve the nonhomogeneous burnup equations but convert the nonhomogeneous term to the homogeneous term based on the conservation of the total molar fraction of heavy metals in the fuel salt.

In order to address the unique burnup features of MSRs and meanwhile, maintain a high precision in depletion calculations, a specific depletion code MODEC for MSRs is developed. In MODEC, several advanced and high-precision algorithms for burnup equations and nonhomogeneous systems are employed, which are provided in detail in Sections 2 and 3. Then, its reliability is verified in Section 4, and finally, in Section 5, conclusions are given.

2. Mathematical model and method

2.1. Algorithms of conventional burnup equation

The conventional burnup equations consist of the following set of first-order liner differential governing equations:

$$\begin{cases} \frac{dn_i(t)}{dt} = \sum_j \lambda_{j,i} n_j - \lambda_i n_i, \\ \lambda_{j,i} = f_{j\to i}\sigma_j^{tot}\phi + \gamma_{j\to i}\lambda_j^{decay}, \quad \lambda_i = \sigma_i^{tot}\phi + \lambda_i^{decay}. \end{cases} \tag{1}$$

where n_i is the concentration of nuclide i; $f_{j\to i}$ is the probability of a neutron reaction from nuclide j into nuclide i; σ_j^{tot} and σ_i^{tot} are the microscopic one-group total cross sections of nuclide j and nuclide i, respectively; ϕ is the average neutron flux; $\gamma_{j\to i}$ is the branching ratio for the decay of nuclide j into nuclide i; γ_i^{decay} is the decay constant of nuclide i.

To solve the burnup system described by Eq. (1), two basic methods are implemented in MODEC. The first is transmutation trajectory analysis (TTA). The basic idea of TTA is to decompose the complex burnup web into a set of linear nuclide-chains, and then solve them one by one. A recursive form of the TTA (Huang et al., 2016) is implemented in MODEC. The solution of the ith node takes the following form:

$$n_i(t) = \sum_{j=1}^{a_i}\sum_{k=0}^{b_{i,j}} \gamma_{i,j,k} t^k e^{-\tilde{\lambda}_j t} \tag{2}$$

And the recursion formulas of the associated variables are given as follows:

(1) For the first node:

$$\begin{aligned} &a_1 = 1 \quad b_{1,1} = 0 \\ &\tilde{\lambda}_j = \lambda_1 \quad \gamma_{1,1,0} = n_1(0) \end{aligned} \tag{3}$$

(2) For all j where $\tilde{\lambda}_j \neq \lambda_{i+1}$:

$$\begin{aligned} &b_{i+1,j} = b_{i,j} \quad \gamma_{i+1,j,b_{i+1,j}} = \frac{\lambda_{i+1}\gamma_{i,j,b_{i,j}}}{\lambda_{i+1}-\lambda_j} \\ &\lambda_{i+1,j,k} = \frac{\lambda_{i+1}\gamma_{i,j,k} - (k+1)\gamma_{i+1,j,k+1}}{\lambda_{i+1}-\gamma_j} \quad (0 \leqslant k \leqslant b_{i+1,j}) \end{aligned} \tag{4}$$

(3) If $\exists \hat{j} \in \{1,2,\ldots,a_i\}, \tilde{\lambda}_{\hat{j}} = \lambda_{i+1}$,then:

$$\begin{aligned} &a_{i+1} = a_i \quad b_{i+1,\hat{j}} = b_{i,\hat{j}} + 1 \\ &\gamma_{i+1,\hat{j},k} = \frac{\lambda_{i+1,i}\gamma_{i,\hat{j},k-1}}{k} \quad \left(0 \leqslant k \leqslant b_{i+1,\hat{j}}\right) \\ &\gamma_{i+1,\hat{j},0} = n_{i+1}(0) - \sum_{j=1,j\neq\hat{j}}^{a_{i+1}} \gamma_{i+1,j,0} \end{aligned} \tag{5}$$

Otherwise $\forall \hat{j} \in \{1,2,\ldots,a_i\}, \tilde{\lambda}_j \neq \lambda_{i+1}$, then:

$$\begin{aligned} &a_{i+1} = a_i + 1 \quad b_{i+1,a_{i+1}} = 0 \\ &\tilde{\lambda}_{a_{i+1}} = \lambda_{i+1} \quad \gamma_{i+1,a_{i+1},0} = n_{i+1}(0) - \sum_{j=1,j\neq\hat{j}}^{a_i} \gamma_{i+1,j,0} \end{aligned} \tag{6}$$

To avoid the unnecessary long chains, the trajectory passage (Cetnar, 2006) is introduced to cut off the unimportant chains:

$$P_n = \frac{n_{i+1}(t)}{n_1(0)_{\lambda_{i+1}=0}} < \varepsilon \tag{7}$$

The second method implemented in MODEC is Chebyshev rational approximation method (CRAM) (Pusa and Leppänen, 2010), which is one of the matrix exponential methods. The burnup equations described by Eq. (1) can be written in a matrix form:

$$\frac{d\vec{n}}{dt} = \mathbf{A}\vec{n} \tag{8}$$

where $\vec{n}$ is the vector of nuclide concentrations, and $\mathbf{A}$ is the transition matrix containing the rate coefficients of nuclide transmutation by decay and/or neutron reaction. The solution of the matrix-form burnup equations has an exponential form:

$$\vec{n}(t) = e^{\mathbf{A}t}\vec{n}_0 \tag{9}$$

where $e^{\mathbf{A}t}$ is the matrix exponential, and $\vec{n}_0$ is the vector of initial nuclide concentrations.

The CRAM method approximates the exponential by a rational function for the interval $(-\infty, 0]$. And the rational function is then expressed in a pole-residue form:

$$e^z \approx r_k(z) = \frac{P_k(z)}{Q_k(z)} = \alpha_0 + \sum_{i=1}^{k}\frac{\alpha_i}{z-\theta_i} = \alpha_0 + 2Re\sum_{i=1}^{k/2}\frac{\alpha_i}{z-\theta_i} \tag{10}$$

where P_k and Q_k are polynomials of order k; α_i and θ_i are the residues and poles. Since all eigenvalues of the matrix $\mathbf{A}t$ are negative (Pusa and Leppänen, 2010), the matrix exponential $e^{\mathbf{A}t}$ can be computed by CRAM. Thus, Eq. (9) can be written in CRAM as:

$$\begin{aligned} \vec{n}(t) &= e^{\mathbf{A}t}\vec{n}_0 \approx \hat{r}_{k,k}(-\mathbf{A}t)\vec{n}_0 \\ &= \alpha_0\vec{n}_0 + 2Re\left(\sum_{i=1}^{k/2}(\theta_i\mathbf{I} + \mathbf{A}t)^{-1}\alpha_i\vec{n}_0\right) \end{aligned} \tag{11}$$