M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering, Jeju, Korea, April 16-20, 2017, on USB (2017)

Marduk: A Monte Carlo Code for Analyzing Stochastic Neutron Population Dynamics

Thomas M. Sutton,[1] Andrew D. LaCharite,[1] and Anil K. Prinja[2]

[1]*Naval Nuclear Laboratory, Schenectady, New York, USA, thomas.sutton@unnpp.gov, andrew.lacharite@unnpp.gov*
[2]*Department of Nuclear Engineering, University of New Mexico, Albuquerque, New Mexico, USA, prinja@unm.edu*

Abstract *A Monte Carlo code called Marduk has been developed for the purpose of analyzing the stochastic behavior of neutron and precursor populations in nuclear systems having a weak or no external source during the initial phase of a transient due to a positive reactivity insertion. Its primary purpose is to determine the size-distribution of the neutron population, but it may also be used to determine quantities such as the probabilities of initiation and extinction. The code employs a lumped-parameter (point kinetics) model, and allows for a piece-wise linear, discontinuous variation in reactivity. It has the option of switching from a stochastic to a deterministic solution method once the population has become sufficiently large that its evolution has become essentially deterministic. The code uses a parallel algorithm with over 90% parallel efficiency when using several thousand processors.*

I. INTRODUCTION

The dynamic behavior of neutron populations in nuclear systems having a weak or no external source and producing negligible power is dominated by stochastic effects [1]. Solution of the usual reactor kinetics equations only provides the mean behavior of the populations, and does not address the distribution of possible behaviors. Marduk is a Monte Carlo code for analyzing the stochastic behavior of neutron and precursor populations in such nuclear systems. Its primary purpose is to determine the size-distribution of the population during the initial phase of a positive reactivity insertion transient, but it may also be used to determine quantities such as the probabilities of initiation (POI) and extinction (POE) [2]. The code employs a lumped-parameter (point kinetics) model, and allows for a piece-wise linear, discontinuous variation in reactivity. It optionally treats prompt and delayed neutrons as distinct populations, and it allows for an arbitrary number of delayed neutron precursor families. Since it is meant to be used for systems producing negligible power, there is no treatment of thermal feedback.

The study of the probability distribution of neutrons and precursors for subcritical and supercritical multiplying systems began in the earliest days of the nuclear era [3]. Early analytical methods were later superseded by deterministic codes for obtaining approximate numerical results [4]. Recently, Monte Carlo codes have been described that simulate prompt neutron and gamma ray production from fission chains for the purpose of producing counting distributions [5] and for modeling fast pulses in nuclear systems [6]. While a Monte Carlo algorithm for simulating stochastic neutron populations including space, energy, and direction dependence has been successfully implemented [7], such a method is far too computationally expensive for performing the large number of realizations required to get accurate neutron and precursor distributions.

The next section discusses the underlying theory that gives rise to the algorithm used by Marduk. The details of the algorithm are presented in Sec. III. Section IV illustrates some of Marduk's capabilities by providing examples of is solutions for various types of problems. Parallel efficiency is discussed in Sec. V. Section VI contains some concluding remarks. An Appendix shows how the stochastic method including distinct prompt and delayed neutron populations is consistent with the usual point kinetics formulation.

II. THEORETICAL DEVELOPMENT

To generate a distribution of neutron populations, Marduk runs multiple realizations of a transient—each using a different pseudorandom number sequence. The theoretical basis for the algorithm used to simulate a single realization is the forward master equation [8,9,10]:

$$\frac{dP(\mathbf{X},t)}{dt} = \sum_{\mathbf{X}'} W(\mathbf{X}' \to \mathbf{X},t) P(\mathbf{X}',t), \tag{1}$$

where $P(\mathbf{X},t)$ is the probability that at time t a system is in a state specified by the vector $\mathbf{X}$, $W(\mathbf{X}' \to \mathbf{X},t)$ is the transition probability per unit time at time t that a system in state $\mathbf{X}'$ transitions to state $\mathbf{X}$, $W(\mathbf{X},t) = \sum_{\mathbf{X}'} W(\mathbf{X} \to \mathbf{X}',t)$ is the probability per unit time at time t that a system in state $\mathbf{X}$ transitions to any other state. At $t = 0$ the state is $\mathbf{X}_0$. Defining $\Psi(\mathbf{X},t) \equiv W(\mathbf{X},t) P(\mathbf{X},t)$ as the probability per unit time at time t that the system will be undergoing a transition from state $\mathbf{X}$ to some other state, Eq. (1) may be used to obtain the integral equation

$$\begin{aligned}\Psi(\mathbf{X},t) &= T(0 \to t|\mathbf{X}_0) \\ &\quad + \int_0^t dt' T(t' \to t|\mathbf{X}) \sum_{\mathbf{X}'} C(\mathbf{X}' \to \mathbf{X},t') \Psi(\mathbf{X}',t'),\end{aligned} \tag{2}$$

where

$$T\left(t' \to t \middle| \mathbf{X}\right) \equiv W\left(\mathbf{X},t\right) e^{-\int_{t'}^{t} dt'' W(\mathbf{X},t'')} \tag{3}$$

is the conditional probability that a system in state $\mathbf{X}$ at time t' will transition out of that state per unit time at time *t*, and

$$C\left(\mathbf{X}' \to \mathbf{X}, t'\right) \equiv \frac{W\left(\mathbf{X}' \to \mathbf{X}, t'\right)}{W\left(\mathbf{X}', t'\right)}. \tag{4}$$

is the probability that a system transitioning out of state $\mathbf{X}'$ at time *t* will transition into state $\mathbf{X}$. Using a Neumann expansion, i.e.

$$\Psi\left(\mathbf{X},t\right) = \sum_{j=0}^{\infty} \Psi_j\left(\mathbf{X},t\right), \tag{5}$$

where the index *j* denotes the number of transitions occurring prior to time *t*, we have

$$\begin{aligned} \Psi_0\left(\mathbf{X},t\right) &= T\left(0 \to t \middle| \mathbf{X}_0\right) \\ \Psi_j\left(\mathbf{X},t\right) &= \int_0^t dt' T\left(t' \to t \middle| \mathbf{X}\right) \sum_{\mathbf{X}'} C\left(\mathbf{X}' \to \mathbf{X}, t'\right) \Psi_{j-1}\left(\mathbf{X}', t'\right) \end{aligned}. \tag{6}$$

Equation (6) suggests the following Monte Carlo algorithm:

1. Sample the time t_1 to the first transition (event) from $T\left(0 \to t \middle| \mathbf{X}_0\right)$.
2. Sample the next state, $\mathbf{X}_1$, from $C\left(\mathbf{X}_0 \to \mathbf{X}, t_1\right)$.
3. Sample the time t_2 of the second event from $T\left(t_1 \to t \middle| \mathbf{X}_1\right)$.
4. Continue alternating between sampling the outcomes of events from *C* and times of the events from *T* until some specified stopping criterion is met.

III. ALGORITHM

At the highest level, the Marduk algorithm consists of an outer loop over a user-specified number of realizations. Each realization uses the same lumped-parameters and the same initial condition, but a different initial seed for the pseudorandom number generator. After all realizations have been performed, the results of all the realizations are analyzed to determine the distribution of the population size or other quantities of interest. In the next subsection, the details of the algorithm used to perform each realization are discussed.

1. Algorithm for Each Realization

For the lumped-parameter model, the state is given by the column vector $\mathbf{X} \equiv \mathrm{col}\left(n, n_2, m_1, \ldots, m_I\right)$, where n_1 is the number of prompt neutrons, n_2 is the number of delayed neutrons, and m_i is the number of delayed neutron precursors in family *i*. Prompt and delayed neutrons may be treated separately so that any difference in the likelihood that each type will cause a fission may be accounted for. Table I gives the parameters for the lumped-parameter model. These are supplied as input and are assumed to be time-independent.

Table I. Input for Lumped-Parameter Model

quantity	symbol
neutron generation time	Λ
mean rate of source events	S
probability of producing $\nu = 0, \ldots, \nu_{\max}$ prompt neutrons in a fission event	p_ν
delayed neutron fraction	β
effective delayed neutron fraction	β_{eff}
delayed neutron precursor abundance for family $i = 1, \ldots, I$	a_i
delayed neutron precursor decay constant for family $i = 1, \ldots, I$	$\lambda_{\mathrm{D},i}$

The average number of prompt neutrons per fission and the average number of all neutrons (prompt and effective delayed) per fission are, respectively, $\bar{\nu}_{\mathrm{p}} = \sum_{\nu=0}^{\nu_{\max}} p_\nu \nu$ and $\bar{\nu} = \bar{\nu}_{\mathrm{p}} / \left(1 - \beta_{\mathrm{eff}}\right)$, where $\nu_{\max}$ is the maximum fission multiplicity. The delayed neutron fraction for family *i* is $\beta_i = \beta a_i$. Finally, the ratio of the importance of delayed neutrons to that of prompt neutrons is $\gamma = \beta_{\mathrm{eff}} / \beta$.

Table II lists the types of events, their probabilities per unit time, and how each changes the state. In that table, the per-neutron rate of prompt-neutron-induced fission is $\lambda_{\mathrm{F}} = 1/\bar{\nu}\Lambda$, and the per-neutron rates of removal by means other than fission for prompt and delayed neutrons are, respectively, $\lambda_{\mathrm{C},1} = \left(1/k - 1/\bar{\nu}\right)/\Lambda$ and $\lambda_{\mathrm{C},2} = \left(1/k - \gamma/\bar{\nu}\right)/\Lambda$. Note that the effective multiplication factor, *k*, may be time-dependent.

Table II. Types of Events

event	probability per unit time	effect on state n_1	n_2	m_i
source emission	S	$+1$		
removal of a prompt neutron (non-fission)	$\lambda_{\mathrm{C},1} n_1$	-1		
removal of a delayed neutron (non-fission)	$\lambda_{\mathrm{C},2} n_2$		-1	
fission by prompt neutron producing 0 precursors	$\lambda_{\mathrm{F}}\left(1-\beta\bar{\nu}\right) p_\nu n_1$	$+\nu-1$		
fission by prompt neutron producing a precursor	$\lambda_{\mathrm{F}}\beta_i\bar{\nu}\, p_\nu n_1$	$+\nu-1$		$+1$
fission by delayed neutron producing 0 precursors	$\gamma\lambda_{\mathrm{F}}\left(1-\beta\bar{\nu}\right) p_\nu n_2$	$+\nu$	-1	
fission by prompt neutron producing a precursor	$\gamma\lambda_{\mathrm{F}}\beta_i\bar{\nu}\, p_\nu n_2$	$+\nu$	-1	$+1$
precursor decay	$\lambda_{\mathrm{D},i} m_i$		$+1$	-1

For both operations, we need the total transition probability per unit time as a function of time. Mathematically, this is given by the summation of the quantities in the second column of Table II (including summations over ν and i, where appropriate). For computational efficiency, Marduk takes advantage of significant simplification that arises in the summation over the terms in the second through seventh rows of the table, computing the transition probability per unit time using

$$W\left(\mathbf{X},t\right)=\frac{n_1+n_2}{\Lambda}\left(1-\rho\left(t\right)\right)+\sum_i \lambda_{\mathrm{D},i} m_i + S\,. \qquad (7)$$

Note that in the above expression we have eliminated k in favor of reactivity using $\rho=\left(k-1\right)/k$. The time-dependence of this quantity is treated by defining reactivity intervals. Within an interval reactivity is either constant or linearly-varying. To sample the time to the next event, τ, we need the cumulative distribution function (CDF) corresponding to the probability distribution function given by Eq. (3). Denoting the current value of the total transition probability per unit time by W_0 and its derivative by W', we have $W\left(\mathbf{X},t+\Delta t\right)=W_0+W'\Delta t$. Using this functional dependence in Eq. (3) yields the required CDF. Letting ξ denote a pseudorandom number distributed uniformly between 0 and 1, the following sampling scheme is obtained:

$$\begin{aligned} &\tau \leftarrow -\frac{\ln\xi}{W_0}; && W'=0 \\ &\tau \leftarrow \frac{W_0}{W'}\left[\sqrt{1-2\frac{W'}{W_0^2}\ln\xi}-1\right]; && W'\neq 0. \end{aligned} \qquad (8)$$

If the sampling results in a time that lies beyond the current reactivity interval, time is advanced to the end of the current interval and a new time increment is sampled using the reactivity of the next interval

This method of advancing the time distinguishes Marduk from some similar codes. For example, the code described in Ref. 6 uses a fixed time-step size chosen so that the probability of a neutron or precursor undergoing more that one event in a time step is less than some small value such as 10^{-4} . At each timestep, at most one event is sampled for each neutron and precursor. The algorithm used by Marduk avoids this approximation.

The 'brute force' way of sampling the outcome of an event would be to compute the quantities in Table II, divide each by the total transition probability per unit time, then use the resulting probabilities to sample the outcome. The number of probabilities using this scheme is $3+I+2\left(I+1\right)\left(\nu_{\max}+1\right)$. For six delayed precursor families and $\nu_{\max}=7$, there would thus be 121 probabilities that must be computed for each event. A more efficient method is to employ a nested approach. From Eq. (7), it is seen that the transition probabilities per unit time of a source event, a precursor decay, or a neutron absorption event will already have been computed at this point in the calculation. Dividing each by W yields the associated probabilities, which are then used to sample which one of these types of events takes place. If a source event is sampled, then there is no need to compute any other probability. If the event is a precursor decay, then only the probabilities for each type of precursor need to be computed. Note that the corresponding transition probabilities per unit time have already been computed to get the second term on the right-hand-side of Eq. (7), so all that is required is a division of these by W. Again, the computation of the vast majority of the probabilities has been avoided. If the sampled event is an absorption, then the code goes to the next level and computes the probabilities that the absorption is a prompt neutron capture, a delayed neutron capture, or a fission. In the case that a capture event is sampled, then once again computation of most of the probabilities has been avoided. Only if a fission is sampled does the code get to the inner-most level in the nesting in which it is determined whether the neutron causing the fission is prompt or delayed,

whether or not a precursor is produced, and the number of prompt neutrons produced.

2. Modes of Operation

Marduk supports several modes of operation so that it can effectively handle various types of calculations.

A. Mode 1

In the first and simplest mode, the stochastic calculation for each realization is run from the initial state to the user-specified ending time. This mode is only practical for cases in which the population does not become excessively large. As the population increases, the mean time between events decreases. If the population were to become too large, the stochastic calculation would make very slow progress in terms of simulated time per unit computational time.

B. Mode 2

The second mode is used for positive reactivity insertion transients, and takes advantage of the fact that once the neutron population reaches a certain 'fiducial' level the subsequent time evolution is essentially deterministic. In this mode the stochastic calculation for a realization is run as a series of stages, with each stage consisting of a user-specified number of events. Following each stage the effective neutron number for the realization, defined as $n_{\text{eff}} \equiv n_1 + \gamma n_2$, is computed and compared to a user-supplied threshold. If the threshold has been exceeded, the current values of the neutron and precursor populations are supplied as input to a deterministic solver which then runs the problem to its conclusion. This allows for the solution of problems in which the population becomes too large for the practical use of a purely stochastic method.

C. Mode 3

The third mode of operation is for POI/POE calculations involving prompt neutrons only. A realization is stopped when either the neutron population goes identically to zero or exceeds a user-defined threshold. The fraction of realizations that stop by exceeding the threshold is the estimate of the POI.

3. Deterministic Solver

Marduk includes a deterministic solver that uses a fourth-order Runge-Kutta-Fehlberg method due to Kaps and Rentrop [11,12]. The equations solved are the usual point kinetics equations, and the quantities solved for are the expected values (denoted by angular brackets) of the effective number of neutrons and the effective number of precursors in each family, i.e., $\langle n_{\text{eff}} \rangle \equiv \langle n_1 \rangle + \gamma \langle n_2 \rangle$ and $\langle c_i \rangle \equiv \gamma \langle m_i \rangle, \quad i = 1, \ldots, I$. See the Appendix for more details on the relationship between the quantities solved for in the stochastic and deterministic calculations.

The deterministic solver is used in operational mode 2 to extend the stochastic solution to the end of the problem after the user-supplied effective neutron number threshold has been exceeded. It is also used to compute the denominator in calculations involving the ratio of the effective neutron number to its expected value. In either case, the fraction of the total computational time used by the deterministic solver is generally negligible.

4. Parallelization

The parallel algorithm uses one dedicated master process and *N* server processes that communicate using MPI. All communication is between the master process and the server processes—the server processes do not communicate with each other. At the beginning of the calculation, the master process assigns the first *N* realizations to the server processes. For the remainder of the calculation, when the master process receives a result from one of the server processes, it stores that result and assigns that process another realization. It continues to do this until all of the realizations have been completed. At that point the master process computes the desired output quantities from the results of the realizations. Parallel efficiency is discussed in Sec. V.

IV. EXAMPLE PROBLEMS AND RESULTS

1. Critical on (n,2n) Problem

The first problem is meant to provide confidence of Marduk's correctness, as the problem considered is one that can be solved analytically. The problem is an artificial one with only two possible reactions—capture and (n,2n)—that occur with equal probability. Delayed neutrons are neglected, and there is no fixed source. Since on the average one neutron is produced per neutron lost, the system is critical. Assuming one neutron present at $t = 0$, the probability of having *n* neutrons present at some later time *t* is

$$q_n(t) = \begin{cases} \dfrac{t/\ell}{2 + t/\ell}; & n = 0 \\ \dfrac{4(t/\ell)^{n-1}}{(2 + t/\ell)^{n+1}}; & n > 0 \end{cases}, \tag{9}$$

where $\ell = k\Lambda$ is the neutron lifetime. One million realizations were produced using operational mode 1, and the fraction having $n = 0, 1, 2, 3$ neutrons was computed for $t = \ell, 2\ell, \ldots, 6\ell$. These are plotted in Fig. 1 (points) along

with the corresponding probabilities given by Eq. (9) (lines). Figure 2 shows the probability of having a non-zero number of neutrons versus time out to 1024ℓ. As before, the analytic solution (line) is given along with the result obtained using Marduk (points). Both figures show excellent agreement between theory and simulation.

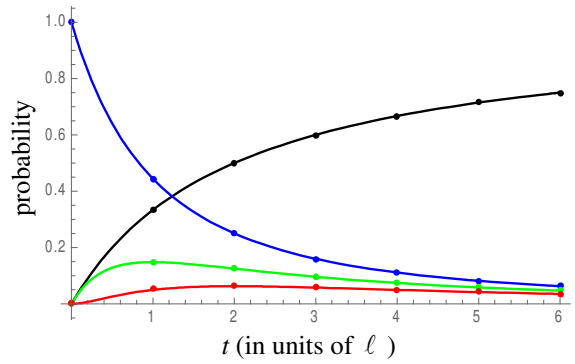

Fig. 1. Marduk and analytic values of $q_n(t)$ for $n = 0,1,2,3$ and $t \le 6\ell$.

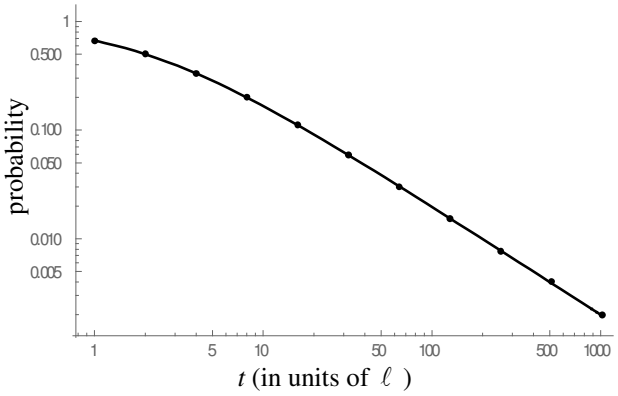

Fig. 2. Marduk and analytic values of $1 - q_0(t)$.

2. Constant Reactivity Problem

This problem includes six families of delayed neutron precursors and a fixed source. Initially there are no neutrons or precursors in the system. Reactivity is held constant at a value of 0.005, and the problem duration is 25 s. Table III provides the problem parameters. The quantity to be determined is the probability density function (PDF) for the final value of the ratio $r = n_{\text{eff}} / \langle n_{\text{eff}} \rangle$.

This problem was run for 150,000 realizations using operational mode 2 and a stage length of 10^8 events. The transition from the stochastic to the deterministic algorithm occurred following the first stage for which $n_{\text{eff}} > 10^5$. The calculation took approximately 8.4 hours using 401 processors (1 master and 400 servers). Figure 3 shows a binned representation of the PDF for the ratio r with a bin width of 0.05. The minimum and maximum r values are 5.17×10^{-3} and 16.64, and the average is 1.00084.

Table III. Parameters for Constant Reactivity Problem

quantity	value(s)
Λ (s)	10^{-4}
S (s^{-1})	1000
p_ν; $\nu = 0,\ldots,7$	0.0317, 0.172, 0.3363, 0.3038, 0.1268, 0.0266, 0.0026, 0.0002
β	0.0065
β_{eff}	0.0068
a_i, $i = 1,\ldots,6$	0.033, 0.219, 0.196, 0.395, 0.115, 0.042
$\lambda_{\text{D},i}$, $i = 1,\ldots,6$ (s^{-1})	0.0124, 0.0305, 0.111, 0.301, 1.14, 3.01

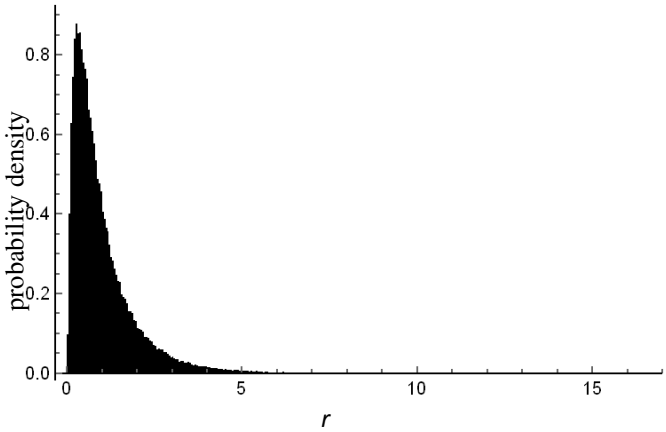

Fig. 3. Distribution of r for the constant reactivity problem.

To illustrate operational mode 3, Fig. 4 shows the n_{eff} trajectories corresponding to the realizations with the smallest and largest final values of r. The solid black line is the deterministic point kinetics solution for the entire transient. The red circles are the stochastic values for the realization corresponding to the smallest final value of r. Following the 15th stage, the effective neutron number exceeded 10^5, and the code transitioned to the deterministic algorithm (red line). The blue circles and line are the same quantities for the realization corresponding to the largest final value of r. For this realization the stochastic-to-deterministic transition took place following stage 16.

Figure 5 provides enlarged views of the transitions from the stochastic to deterministic solutions. To show that the assumption of deterministic behavior beyond the transition point is valid, additional calculations were performed that extend the stochastic solution to effective neutron numbers of 5×10^5. The results of these calculations are indicated by the open circles superimposed on the deterministic solutions. As can be seen, the deterministic solutions are in good agreement with the extended stochastic solutions.

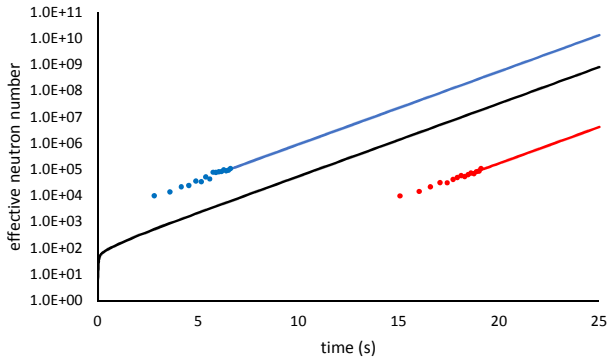

Fig. 4. Transition from stochastic to deterministic solutions for realizations with largest (blue) and smallest (red) values of *r*. The circles are the stochastic values at the stage ends, and the lines are the deterministic continuations. The black line is the deterministic solution for the entire transient.

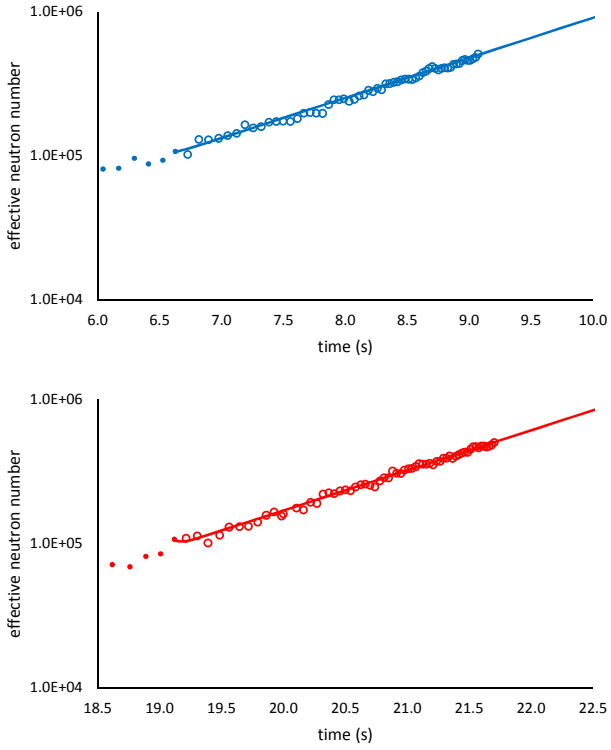

Fig. 5. Detail of the transition from stochastic to deterministic solutions for the realizations with the largest (top, blue) and smallest (bottom, red) values of *r*. The closed circles are the stochastic values and the line is the deterministic continuation from the calculation that produced Figs 3 and 4. The open circles are values from additional stochastic stages obtained by separate calculations.

3. Ramp Reactivity Insertion Problem

This problem demonstrates Marduk's ability to handle linear time-dependent reactivity insertions. There are initially no neutrons or precursors in the system. The complete set of problem parameters is given in Table IV. Reactivity increases at a constant rate from an initial value of −0.01 to 0.0065 over the course of 16.5 s. Marduk was run for 100 realizations using operational mode 1. Figure 6 shows the effective neutron population averaged over the realizations (points) as well as the deterministic point kinetics solution (line) versus time. As can be seen, there is good agreement.

Table IV. Parameters for Ramp Reactivity Problem

quantity	value(s)
Λ (s)	1.5×10^{-5}
S (s^{-1})	8800
$p_\nu;\quad \nu=0,\ldots,5$	0.027, 0.158, 0.339, 0.305, 0.133, 0.038
β	0.0065
β_{eff}	0.0068
$a_i,\quad i=1,\ldots,6$	0.033, 0.219, 0.196, 0.395, 0.115, 0.042
$\lambda_{\text{D},i},\quad i=1,\ldots,6$ (s^{-1})	0.0124, 0.0305, 0.111, 0.301, 1.14, 3.01

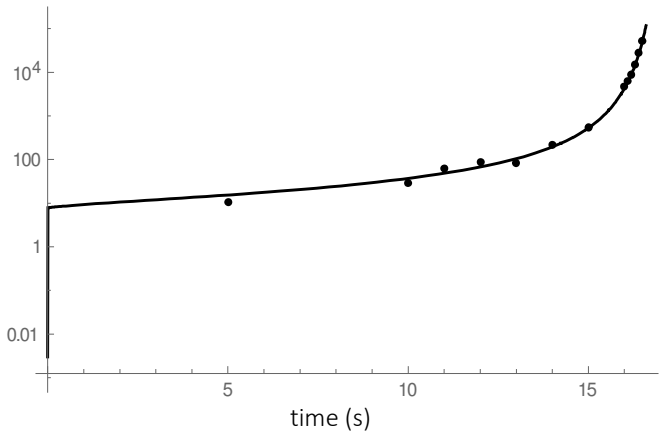

Fig. 6. Average neutron population for the ramp reactivity insertion problem. Points are averages over 100 stochastic realizations. The line is the deterministic solution.

4. Probability of Initiation Problem

This problem demonstrates Marduk's ability to determine the POI. The problem consists of a prompt-supercritical system with no external source, and neglects delayed neutrons. At $t=0$ there is one neutron present. For long times there are two possible outcomes: the neutron population goes to zero or becomes exponentially

increasing. The fraction of times the latter outcome occurs is the POI.

The POI may be determined analytically [2] as $1-x_0$, where x_0 is the root of

$$g(x) \equiv \sum_{\nu} c_\nu x^\nu - x \tag{10}$$

lying on the interval (0,1), and c_ν is the probability that an event in which the interacting neutron is lost produces ν neutrons as a result. These probabilities are related to the fission multiplicities by

$$\begin{aligned} c_0 &= 1-\frac{k}{\bar{\nu}}(1-p_0) \\ c_\nu &= \frac{k}{\bar{\nu}} p_\nu \; ; \quad \nu > 0 \end{aligned} \tag{11}$$

Exact values of the POI were calculated for four values of *k* using the multiplicity distribution given in Table V. For each *k* value, Marduk was run in operational mode 3 for one million realizations. Table VI shows the exact POI values, the Marduk values, the percent differences, and the relative standard deviation determined using the binomial distribution. In all four cases the Marduk values are in statistical agreement with the exact values.

Table V. Fission multiplicity distribution for the POI problem.

ν	p_ν
0	0.0317
1	0.1720
2	0.3363
3	0.3038
4	0.1268
5	0.0266
6	0.0026
7	0.0002

Table VI. Probability of Initiation

k	POI	Marduk value	difference (%)	standard deviation (%)
1.01	0.01036	0.01035	−0.10	1.00
1.05	0.05083	0.05085	0.04	0.43
1.10	0.09937	0.09928	−0.09	0.30
1.50	0.42670	0.42679	0.02	0.12

V. PARALLEL EFFICIENCY

To measure parallel efficiency, the constant reactivity problem from Sec. IV.2. was run multiple times, with the number of server processes ranging from 1 to 2000. For each calculation, the number of realizations was ten times the number of server processes. Table VII gives the execution time for each calculation, as well as the parallel efficiency defined as the ratio of the single-server time to the multi-server time. As can be seen, the parallel efficiency is over 90% up to 2000 server processes. The calculations were run on a cluster of 12-core Intel Xeon E5-2680v3 2.5 GHz (Haswell) Processors.

Table VII. Parallel Performance

number of server processes	number of realizations	elapsed time (s)	parallel efficiency
1	10	807	1.00
10	100	846	0.95
100	1,000	866	0.93
1000	10,000	885	0.91
2000	20,000	871	0.93

VI. CONCLUSION

A code called Marduk has been developed to analyze the behavior of neutron and precursor populations in nuclear systems having a weak or no external source. It can treat prompt and delayed neutrons as distinct populations, and it can accommodate an arbitrary number of delayed neutron precursor families. It can also run prompt-neutron-only problems, such as determination of the POI. It can solve problems in which reactivity changes with time in a piece-wise linear, discontinuous fashion.

The code runs multiple realizations for each problem to obtain a distribution of outcomes from which the quantities of interest are obtained. Examples of such quantities are the distribution of the effective neutron number and the POI. Each realization uses a Monte Carlo method to sample the times between events, as well as the outcomes of the events. Once the population exceeds a threshold above which the behavior is essentially deterministic, the code has the option to switch to a much more computationally-efficient deterministic point-kinetics algorithm. The code is parallelized using MPI, with a single master process that assigns realizations to multiple server processes. Parallel efficiency of greater than 90% has been demonstrated using up to 2000 server processes.

Results have been presented for four problems. Two of the problems have analytic solutions, and the Marduk values agree with these. For the other two problems the average of many realizations performed with Marduk were compared to numerical solutions of the point kinetics equations, and excellent agreement was found.

APPENDIX: CONSISTENT DETERMINISTIC EQUATIONS

Here we demonstrate that the expected value of the effective neutron number computed stochastically by Marduk is identical to the value obtained from the solution of the usual point kinetics equations. Following MacMillan

[13], we rewrite Eq. (1) explicitly in terms of the quantities pertinent to the problem of interest, i.e.

$$\begin{aligned}\frac{dP(n_1,n_2,\mathbf{m},t)}{dt} &= S\left[P(n_1-1,n_2,\mathbf{m},t)-P(n_1,n_2,\mathbf{m},t)\right]\\ &+\lambda_{\mathrm{C},1}(n_1+1)P(n_1+1,n_2,\mathbf{m},t)\\ &-\frac{n_1}{k\Lambda}P(n_1,n_2,\mathbf{m},t)\\ &+\lambda_{\mathrm{C},2}(n_2+1)P(n_1,n_2+1,\mathbf{m},t)\\ &-\frac{n_2}{k\Lambda}P(n_1,n_2,\mathbf{m},t)\\ &+\sum_{\nu=0}^{\nu_{\max}}\frac{1}{\Lambda\bar{\nu}}(n_1-\nu+1)(1-\beta\bar{\nu})\,p_\nu\\ &\times P(n_1-\nu+1,n_2,\mathbf{m},t)\\ &+\sum_{\nu=0}^{\nu_{\max}}\sum_{i=1}^{I}\frac{1}{\Lambda\bar{\nu}}(n_1-\nu+1)\beta_i\bar{\nu}\,p_\nu\\ &\times P(n_1-\nu+1,n_2,\mathbf{m}-\boldsymbol{\delta}_i,t)\\ &+\sum_{\nu=0}^{\nu_{\max}}\frac{\gamma}{\Lambda\bar{\nu}}(n_2+1)(1-\beta\bar{\nu})\,p_\nu\\ &\times P(n_1-\nu,n_2+1,\mathbf{m},t)\\ &+\sum_{\nu=0}^{\nu_{\max}}\sum_{i=1}^{I}\frac{\gamma}{\Lambda\bar{\nu}}(n_2+1)\beta_i\bar{\nu}\,p_\nu\\ &\times P(n_1-\nu,n_2+1,\mathbf{m}-\boldsymbol{\delta}_i,t)\\ &+\sum_{i=1}^{I}\lambda_{\mathrm{D},i}(m_i+1)P(n_1,n_2-1,\mathbf{m}+\boldsymbol{\delta}_i,t)\\ &-\sum_{i=1}^{I}\lambda_{\mathrm{D},i}m_iP(n_1,n_2,\mathbf{m},t),\end{aligned} \tag{12}$$

where $\mathbf{m}\equiv\mathrm{col}(m_1,\ldots,m_I)$ and $\boldsymbol{\delta}_i$ is a vector of length I with a 1 as the i^{th} location and zeros elsewhere. The expected values of the various populations are given by

$$\begin{aligned}\langle n_i\rangle &= \sum n_i P(n_1,n_2,\mathbf{m},t);\quad i=1,2\\ \langle m_i\rangle &= \sum m_i P(n_1,n_2,\mathbf{m},t);\quad i=1,\ldots,I\end{aligned}, \tag{13}$$

where the summations are over all possible values of the number of prompt neutrons, delayed neutrons, and delayed neutron precursors. Multiplying Eq. (12) through by n_1 and performing the summations yields an equation for $\langle n_1\rangle$ in terms of the expected values of the other variables. Equations for the expected values of the other variables can be obtained in an analogous way. The coupled set of equations so obtained is

$$\begin{aligned}\frac{d\langle n_1\rangle}{dt} &= \frac{1}{\Lambda}\left(\frac{k-1}{k}-\beta_{\text{eff}}\right)\langle n_1\rangle+\frac{\gamma(1-\beta_{\text{eff}})}{\Lambda}\langle n_2\rangle+S\\ \frac{d\langle n_2\rangle}{dt} &= -\frac{\langle n_2\rangle}{k\Lambda}+\sum_{i=1}^{I}\lambda_{\mathrm{D},i}\langle m_i\rangle\\ \frac{d\langle m_i\rangle}{dt} &= \frac{\beta_i}{\Lambda}\left(\langle n_1\rangle+\gamma\langle n_2\rangle\right)-\lambda_{\mathrm{D},i}\langle m_i\rangle;\quad i=1,\ldots,I\end{aligned}. \tag{14}$$

The equations for the expected number of prompt and delayed neutrons can be combined to get a single equation for the expected effective neutron number. Defining $\langle c_i\rangle\equiv\gamma\langle m_i\rangle$ and $\beta_{\text{eff},i}\equiv\gamma\beta_i$, we obtain

$$\begin{aligned}\frac{d\langle n_{\text{eff}}\rangle}{dt} &= \frac{\rho-\beta_{\text{eff}}}{\Lambda}\langle n_{\text{eff}}\rangle+\sum_{i=1}^{I}\lambda_{\mathrm{D},i}\langle c_i\rangle+S\\ \frac{d\langle c_i\rangle}{dt} &= \frac{\beta_{\text{eff},i}}{\Lambda}\langle n_{\text{eff}}\rangle-\lambda_{\mathrm{D},i}\langle c_i\rangle;\quad i=1,\ldots,I\end{aligned}, \tag{15}$$

which are just the usual point kinetics equations.

REFERENCES

1. M. M. R. WILLIAMS, *Random Processes in Nuclear Reactors*, Pergamon Press, Oxford (1974).
2. G. I. BELL, "Probability Distribution of Neutrons and Precursors in a Multiplying Assembly," *Annals of Physics*, **21**, 243 (1963).
3. D. HAWKINS and S. ULAM, "Theory of Multiplicative Processes. 1.," LA-171, Los Alamos Scientific Laboratory (1944).
4. G. I. BELL, *et al*., "Probability Distribution of Neutrons and Precursors in Multiplying Medium, II," *Nucl. Sci. Eng.*, **16**, 118 (1963).
5. K. S. KIM, *et al*., "Time Evolving Fission Chain Theory and Fast Neutron and Gamma-Ray Counting Distributions," *Nucl. Sci. Eng.*, **181**, 225 (2015).
6. C. M. COOLING, *et al*., "Coupled Probabilistic and Point Kinetics Modelling of Fast Pulses in Nuclear Systems," *Ann. Nucl. Energy*, **94**, 665 (2016).
7. T. J. TRAHAN, *et al*., "A Monte Carlo Algorithm for Fission Chain Analysis of Dynamic Stochastic Systems," *Trans. Am. Nucl. Soc*., **113**, 661 (2015).
8. I. PAZSIT and L. PAL, *Neutron Fluctuations*, Elsevier, Amsterdam, (2008).
9. C. W. GARDINER, *Handbook of Stochastic Methods*, Springer-Verlag, Berlin, (1983).
10. N. G. VAN KAMPEN, *Stochastic Processes in Physics and Chemistry*, North-Holland, Amsterdam, (1981).
11. P. KAPS and P. RENTROP, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Inglewood Cliffs, N. J. (1971).

12. W. H. PRESS, *et al*., *Numerical Recipes in FORTRAN, 2nd Edition*, Cambridge University Press, Cambridge (1992).
13. D. B. MACMILLAN, "Probability Distribution of Neutron Populations in a Multiplying Assembly," *Nucl. Sci. Eng.*, **39**, 329 (1970).