I&EC research
Industrial & Engineering Chemistry Research

Article

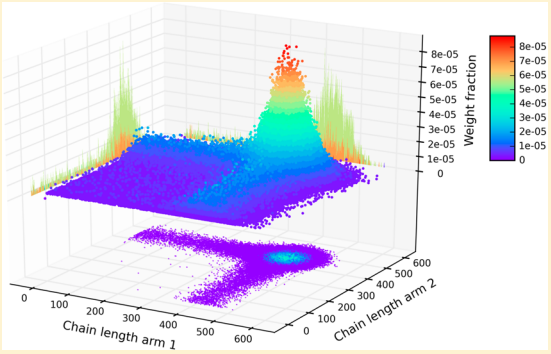
pubs.acs.org/IECR

# Modeling of RAFT Polymerization Processes Using an Efficient Monte Carlo Algorithm in Julia

Esteban Pintos,[†,‡] Claudia Sarmoria,[†,‡] Adriana Brandolin,[†,‡] and Mariano Asteasuain*[,†,‡]

[†]Planta Piloto de Ingeniería Química (PLAPIQUI), UNS−CONICET, Camino La Carrindanga km 7, 8000 Bahía Blanca, Argentina
[‡]Departamento de Ingeniería Química, Universidad Nacional del Sur (UNS), Avenida Alem 1253, 8000 Bahía Blanca, Argentina

**ABSTRACT:** A kinetic Monte Carlo model of a reversible addition−fragmentation chain transfer (RAFT) process is presented. The algorithm has been developed and implemented in Julia for the three main RAFT theories under current discussion (slow fragmentation, intermediate radical termination, and intermediate radical termination with oligomers). Julia is a modern programming language designed to achieve high performance in numerical and scientific computing. Thanks to a careful optimization of the code, it is possible to simulate a RAFT reaction scheme in short computing times for any of the three theories. The code is benchmarked against other programming languages (MATLAB, Python, FORTRAN, and C), showing that Julia presents advantages for this particular system. The model offers an efficient method for predicting average properties and molecular weight distributions of the polymer species, including the bivariate molecular weight distribution of the intermediate two-arm adduct. The proposed model can also be employed to obtain additional detailed information regarding the polymer microstructure at any reaction time.




## 1. INTRODUCTION

Controlled radical polymerization (CRP), also called reversible-deactivation radical polymerization (RDRP) according to the IUPAC recommendation,[1] offers the possibility of synthesizing materials with controlled molecular structures and well-defined architectures in operating conditions that are much less demanding than those required for living ionic polymerizations.[2] For example, some CRP techniques allow obtaining copolymers with controlled molecular weight, polydispersity, composition, and chain architecture under industrial conditions.[3] The distinguishing feature of this kind of polymerization is the establishment of a dynamic equilibrium between propagating radicals and a dormant species. Since this equilibrium is shifted toward the dormant species, the number of active chains is several orders of magnitude smaller than that of dormant species. As a result, chains grow slowly and at approximately the same speed, resulting in a narrow molecular weight distribution.

In order to establish this active−dormant species equilibrium, different approaches may be used. Depending on the type of approach, three main CRP techniques can be identified: nitroxide-mediated polymerization (NMP), atom transfer radical polymerization (ATRP), and reversible addition−fragmentation chain transfer (RAFT). In this work we focus on RAFT polymerization. This is a very versatile CRP technique due to its compatibility with a wide range of monomers and its relatively mild reaction conditions requirements.[4]

Even though considerable effort has been directed at understanding the reaction steps of the RAFT process, fundamental questions regarding the kinetic mechanism remain without a definite answer.[5,6] The basic RAFT polymerization mechanism is generally accepted. However, it has been observed that the propagation rate is slowed down by increasing the concentration of some chain transfer agents (CTA), notably dithiobenzoate RAFT agents.[7] A definite explanation of this characteristic retardation effect has yet to be established. The three main theories that have been proposed to explain this behavior are the following:

1. The *slow fragmentation (SF) model*[7,8] assumes that the retardation effect is due to a slow fragmentation of the intermediate two-arm adduct. The model predicts a rather small value for the fragmentation constant ($k_f$), which in turn leads to a large equilibrium constant ($K = k_a/k_f$). This is consistent with the value of $K$ found in experimental works. However, the model predicts concentrations of propagating radicals and intermediate adduct radicals that differ significantly from those found in the experiments.

2. In the case of the *intermediate radical termination (IRT) model*,[9] the rate retardation is assumed to be caused by a cross-termination reaction between the adduct radicals and propagating radicals. This theory accounts for the presence of

**Received:** April 27, 2016
**Revised:** July 12, 2016
**Accepted:** July 15, 2016
**Published:** July 15, 2016





DOI: 10.1021/acs.iecr.6b01639
*Ind. Eng. Chem. Res.* 2016, 55, 8534−8547

the three-arm polymer detected experimentally, something the SF theory does not do, but overestimates its concentration.

3. The *intermediate radical termination with oligomers (IRTO) model*[10] assumes that the cross-termination reaction postulated in the IRT theory occurs only between adduct radicals and oligomeric propagating radicals (up to two monomers long). The value of the equilibrium constant and the concentrations of the different species predicted by this model are in agreement with the experimental results. However, this theory predicts that the retardation effect should occur over the full conversion range. There is experimental evidence that would seem to contradict this assertion.[11]

In this scenario, mathematical modeling becomes a valuable tool, not only for helping to elucidate the mechanisms of the RAFT polymerization, but also as a partial substitute and complement of complex and time-consuming laboratory experiments. Additionally, it is very useful for the optimization of the synthesis process. In general, it is possible to divide the reported modeling techniques used to simulate polymerizations reactions into two main groups: stochastic and deterministic methods. The latter are based on the numerical solution of the underlying set of differential/algebraic equations associated with the population balances of the system. These methods have the general advantage of providing reliable and reproducible results in relatively short computing times. Their main disadvantages are (i) the amount of information that can be obtained on the microstructure of the polymer chains is limited in comparison to stochastic simulations, (ii) the formulation of the system of equations can result in very large, convoluted, and stiff systems, (iii) the existence of numerical noise and error, and (iv) sometimes the need to use simplification techniques, with the consequent loss of generality.

Several authors have addressed the modeling and simulation of CRP systems using deterministic techniques. There are thorough reviews that focus on the different mathematical modeling approaches for the different types of CRP.[12−14] Among the different articles on RAFT polymerization, it is important to highlight the work by Barner-Kowollik et al.,[8,15] who have modeled and studied RAFT polymerizations using the PREDICI[16] commercial software for different chain transfer agents and proposed the SF theory as a mechanism. Konkolewicz et al.[10,17,18] also modeled the complete molecular weight distribution (MWD) in RAFT polymerization processes using the IRTO theory. Zapata-González et al.[19,20] modeled the MWD in RAFT processes by direct integration of the complete set of population balances. They studied the application of numerical solutions to the direct integration of the population balances, using simplifying assumptions in order to reduce the complexity of the differential−algebraic equation (DAE) system. Fortunatti et al.[21,22] used probability generating functions (pgf) to model RAFT polymerizations. Their model was able to accurately predict the complete MWD and the bivariate MWD of the two-arm adduct. The use of the pgf technique allowed obtaining accurate results in short computational times.

Stochastic modeling approaches are mainly represented by the Monte Carlo (MC) technique. The advantage of this approach is that it is relatively simple to implement and can provide extremely detailed information about polymer microstructure and chain topological architecture that is usually not available with deterministic solutions. However, a disadvantage of the MC technique is the typically high computational cost required for obtaining accurate results.

MC methods have been extensively used in polymer reaction engineering for the prediction of distributed and morphological polymer properties. There are two recent reviews detailing the applications of these methods in polymer science.[23,24] The two main approaches for MC simulations of polymerization kinetics are based on the pioneering works of Gillespie[25] and Tobita.[26] The vast majority of existing stochastic simulation studies in polymer science are based on their algorithms.

Drache et al.[27] presented a MC algorithm to study the retardation effect of cumyl dithiobenzoate mediated methyl acrylate (MA) bulk RAFT polymerization. They used this model to estimate rate coefficients through fitting of experimental data. Drache and Drache[28] introduced a universal Monte Carlo simulator, termed "mcPolymer," that was used to simulate various examples of CRP, including the three most important mechanisms (NMP, ATRP, and RAFT). They focused on the software architecture of the program, including its data management and optimization approaches. Chaffey-Millar et al.[29] worked on a parallelized Monte Carlo implementation for the simulation of a complex RAFT polymerization example. The parallelized approach consisted in subdividing the initial amount of simulated molecules into smaller fractions and performing a parallel MC simulation for each of these fractions on the different workers of the available multicore processor. The simulations were performed for a prespecified fraction of the total reaction time (called "synchronization time"). After this synchronization time had elapsed, the fractions were merged, the molecules were subdivided again into new fractions, and they were sent to a worker to perform the MC simulation again. This procedure was repeated until the final reaction time was reached.

Regarding computation times of the MC algorithms, the final simulation time depends on several variables, such as the reaction mechanism, reaction system, rate parameters, initial concentrations, final reaction time (or conversion), number of molecules, and computer hardware. A direct comparison of the different algorithms needs to be performed on the exact same system in order to get conclusive results. Nevertheless, some typical reaction times are provided: Drache et al.[27] reported computation times of 14 min for a RAFT system considering the IRT mechanism. They used $5 \times 10^8$ molecules for this simulation and reported that their conditions needed $8.95 \times 10^8$ MC iteration steps. The Monte Carlo simulations were executed by the mcPolymer program (Windows and Linux), programmed in C++ with a Tcl interface. The calculations were performed on a computer cluster comprising 20 CPUs (AMD Athlon XP 1900C to AMD Athlon 64 3200C), running under SuSE-Linux. Barner-Kowollik et al.[29] reproduced, to the best of their knowledge, the RAFT system presented by Drache et al.[27] with an algorithm that ran in approximately 450 s for a system with $1 \times 10^9$ molecules. They introduced a parallelization approach for the algorithm that, for a system size of $1 \times 10^{10}$ molecules, produced simulation times of around 5000 s for a single core simulation, 500 s for a simulation over eight cores, and 450 s for a simulation over 16 threads (eight cores with Intel's Hyper-Threading technology). Drache and Drache[28] reported several simulation time values for different CRP systems. Using $1 \times 10^9$ molecules, they reported 61 s and $3.4 \times 10^8$ iteration steps for NMP, 725 s and $2.7 \times 10^9$ iteration steps for ATRP, and 400 s and $1.05 \times 10^9$ iteration steps for RAFT with the IRT mechanism. They also implemented a parallelized


DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

algorithm for the RAFT polymerization with computation times similar to those reported by Barner-Kowollik et al.[29] They analyzed the performance of this approach and concluded that, compared to a single worker implementation, a larger number of molecules could be used in the parallelized algorithm to perform a simulation in a shorter time. The larger number of simulated molecules resulted in smoother concentration−time curves and better data for statistical analyses. However, the underlying single MC simulation was not accelerated, and each processor still needed a certain number of molecules to produce reliable results.[28] In their tests, these authors could not detect any influence of the synchronization time on the simulation results. This leads to the conclusion that a single split of the simulated molecules and a single merge at the final reaction time would lead to the same results.

Using an approach completely different from that of Gillespie, Tobita[26,30] developed a new class of MC method that employed appropriate probability density functions. This method was used for the prediction of macromolecular characteristics of the polymer, such as the full molecular weight distributions of the dead and live polymer chains and the spatial distributions of the branched and cross-linked polymer chains. The approach led to several publications covering the topics of branching and cross-linking in batch and continuous reactors, emulsion polymerization, copolymerization, studies of degradation by random scissions, and cross-linked networks syntheses among others.

In this work, a kinetic MC model of a RAFT reaction scheme was developed and explored in a novel programming language called Julia. The use of this language in combination with engineering of the program code allowed significant reductions in computational time. The underlying algorithm is based on the modeling technique developed by Gillespie.[25] Several code optimization techniques were applied to improve the performance of the simulation. The so-called *exact* method was used, where each molecule in a control volume of the reaction system is individually represented in the algorithm, and each chemical reaction is explicitly simulated while the time evolved is updated at each iteration step. Strategies other than the exact method have been reported, such as the $\tau$-leaping method,[31] presented by Gillespie. This technique allows decreasing simulation time at the cost of lower accuracy. These strategies were not included in this work for the sake of obtaining the highest possible level of detail, but could be considered in future works. The three mechanistic theories of RAFT polymerization under discussion (SF, IRT, and IRTO) were simulated for different conditions and compared using this model.

The present work is structured in the following manner: first, a quick overview and introduction of the Julia programming language is given. Second, the algorithm is described and explained from a computational and mathematical point of view. Some of the details and strategies used to improve the performance of the implementation are also explained. Next, the effect of the number of molecules on the prediction of the average properties and the full MWD for the three theories is discussed. Finally, the results from the implementation of the same algorithm in four other languages (MATLAB, Python, FORTRAN, and C) are compared.

## 2. METHODOLOGY

**2.1. Julia Programming Language.** Choosing a suitable programming language becomes very important when working with time-consuming algorithms. This is the case of MC simulations, which basically consist of long iteration loops that become computationally expensive. Additionally, for the particular implementation considered in this work, large arrays for data storage are required.

The first and most successful numerical computing language was FORTRAN (short for "formula translating system") released in 1957. FORTRAN accomplished remarkable advances toward the "translation" of high-level formulas into low-level machine code. The acceptance of FORTRAN in many areas of high-performance computing to this day, such as the LAPACK package for numerical linear algebra, is proof of its outstanding success.

The outlook of computing in general has changed radically since then. Most modern scientific computing languages such as MATLAB and its open-source alternatives Octave and SciLab, together with R, Python (with NumPy), and others, have gained widespread popularity over the years, and they all fall into the category known as dynamic languages or dynamically typed languages. This means that in these languages programmers write simple, high-level code without being required to declare variable types (such as int, float, or double). Declaration of variable types is characteristic of classic static languages like C, C++, and FORTRAN, also known as statically typed languages. This apparently minor difference gives dynamic languages a major benefit in programmer's time (productivity), at the relative expense of usually longer execution times (performance).

Julia[32] is a modern high-level dynamic programming language designed to achieve high performance in technical, numerical, and scientific computing. The language was designed and developed with the objective of providing a performance similar to that of C while using a high-level programming style. In order to achieve this, its developers combined several technologies, such as "multiple dispatch," "just-in-time (JIT) compilation," and "low-level virtual machine (LLVM) compiler infrastructure," among others.[33,34]

Julia may be seen as an open source high-performance alternative to languages like MATLAB, R, and Python, in addition to being seen as a high-productivity alternative to C, C++, and FORTRAN. Moreover, Julia is better suited for general purpose programming tasks than traditional scientific languages (C and FORTRAN), because it can be used not only for prototyping numerical algorithms but also for deploying them. On top of providing excellent performance, the resulting code is easier to work with, modify, and maintain.

**2.2. Monte Carlo Simulation.** The algorithm used in this work, based on the algorithm presented by Gillespie,[25] consists of six steps: (1) initialization, (2) reaction selection, (3) time step calculation, (4) reaction simulation, (5) update, and (6) iteration, which are described below.

*Step 1. Initialization.* The MC simulation follows the evolution of the population of molecules in a control volume of the reaction medium. The system is initialized by setting the total number of molecules ($N$) present in the control volume at time zero (in other words, setting the initial size of the control volume) and the value of the kinetic constants. In the MC simulation, reaction rates are defined in units of number of reacted molecules per unit time, according to an expression of the type

$$R_i = k_i X^c \qquad i = 1, ..., N_R \tag{1}$$



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

where $R_i$ is the reaction rate of the $i$th reaction, $k_i$ is the microscopic reaction rate coefficient, $X^c$ is the number of combinations of molecules that may take part in the $i$th reaction, and $N_R$ is the number of reaction steps of the kinetic mechanism. Experimental reaction rates are usually given in units of moles of reacted molecules per unit volume and unit time, and are functions of the concentrations of the reacting species; experimental kinetic constants ($k^{exp}$) are expressed in the corresponding units. Therefore, it is necessary to convert the reported experimental reaction rate coefficients to the microscopic reaction rate coefficient for each reaction. Denoting Avogadro's number as $N_A$ and the control volume of the system as $V$, this rate coefficient conversion is as follows.[25]

For first order reactions

$$k = k^{exp} \tag{2}$$

For bimolecular reactions

$$k = \frac{k^{exp}}{N_A V} \tag{3}$$

for different reacting species and

$$k = \frac{2k^{exp}}{N_A V} \tag{4}$$

for identical reacting species.

The control volume of the system is determined by $N$ and the density of the reaction medium. The density value is an experimentally obtained physical quantity, while $N$ is chosen by the user. The value of $N$ is critical in the MC technique since it affects the performance, reliability, and reproducibility of the simulation. The selection of the appropriate value of $N$ is discussed later on.

The reaction rate of the $i$th reaction is given by eq 1. For the reactions simulated in the present work, the computation of the reaction rates can be summarized as follows.

For first order reactions

$$R_i = k_i n_i \tag{5}$$

For bimolecular reactions

$$R_i = k_i n_i n_j \tag{6}$$

for different reacting species and

$$R_i = \frac{k_i n_i (n_i - 1)}{2} \tag{7}$$

for identical reacting species.

In these expressions, $n_i$ and $n_j$ are the number of molecules of species $i$ and $j$. In eq 5 $k_i$ has units of $s^{-1}$, while in eqs 6 and 7 its units are $s^{-1}\cdot molecule^{-1}$.

*Step 2. Reaction Selection.* The algorithm continues by choosing the next reaction that will take place in the system. This selection is random in nature but according to probabilities obtained from the reaction rates calculated in step 1. According to Gillespie,[25] the occurrence probability ($P_i$) of the $i$th reaction is defined as

$$P_i = \frac{R_i}{\sum_{i=1}^{N_R} R_i} \tag{8}$$

Reaction $\mu$ will be selected when the following inequality is satisfied:

$$\sum_{i=1}^{\mu-1} P_i < r_1 < \sum_{i=1}^{\mu} P_i \tag{9}$$

In this expression, $r_1$ is a uniformly distributed random number in the interval [0, 1). Random number generation in Julia is done via the SIMD-oriented fast Mersenne twister (SFMT) algorithm. This version of the Mersenne twister (MT) pseudorandom number generator is twice as fast as the original MT and has better dimensional equidistribution.[35]

Following this procedure exactly as has been described, $R_i$ and $P_i$ would need to be evaluated in each iteration as expressed by eqs 5−8. This is not computationally efficient for two reasons: (1) Not all the $R_i$ values are modified in each iteration. Therefore, there is no need to update all of them but only those affected by the changes in the value of $n_i$. (2) Floating-point division as expressed in eq 8 is not computationally efficient. It should be noted that the MC simulation involves millions of repetitions of the iteration loop. Therefore, the time spent in any operation is very significant.

The first problem has been addressed in the literature.[36] A reported solution consisted of introducing a dependency graph which lists the reaction rates that depend on the outcome of each reaction, enabling the algorithm to identify and modify only those reaction rates which require updating. A similar approach is used in this work, but instead of using a dependency graph to store additional information, the reaction rates are directly and explicitly updated in the same block of code where the update of the number of molecules is performed (step 4 of the MC algorithm). This approach omits the usage of a dependency graph and does not require any special data structure. Hence, it saves time and memory space related to the storage and access time to the graph. Although it requires more code to be written, this code could be easily autogenerated thanks to Julia's native metaprogramming capabilities.

For the second problem, instead of generating a uniform random number in the [0, 1) interval and comparing this number with every $P_i$, a more efficient strategy consists of generating a random number in the interval [0, $R_t$], where $R_t = \sum_{i=1}^{N_R} R_i$, and comparing this new random number directly with the cumulative values of $R_i$. In this way, there is no need to perform floating-point divisions in the order of millions and a considerable performance increase is achieved. This strategy can be summarized by the following inequality:

$$\sum_{i=1}^{\mu-1} R_i < r_1 R_t < \sum_{i=1}^{\mu} R_i \tag{10}$$

Equation 10 replaces eq 9 for the reaction selection. Another improvement in this step of the algorithm was introduced by Cao et al.[37] These authors suggested that, in a system in which there are reactions that occur considerably more frequently than others, if the most frequent reactions are moved to the beginning of the reaction search order, the average search depth when selecting a reaction is reduced and thus computational time is substantially reduced. For instance, in the case of free radical polymerizations, propagation reactions are usually the most frequent ones to take place, while for RAFT polymerizations, addition/fragmentation reactions are also very frequent. This strategy was used in this work.

Finally, Barner-Kowollik et al.[29] suggested reducing the complexity of the reaction selection by using a binary tree data



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

structure. This approach was not used in this work due to the rather small number of reactions and a high capability of ordering the most frequent reactions beforehand. However, the approach could be appropriate for more complex polymerizations systems, with a larger number of reactions and more widely scattered reaction probabilities.

*Step 3. Time Step Calculation.* Once the reaction has been selected, the reaction time ($t$) needs to be updated in order to consider the amount of time that has elapsed between the occurrence of this event and the previous reaction in the system. Following the original Gillespie[25] algorithm, this is done via $t = t + \Delta t$, where

$$\Delta t = \frac{\ln(1/r_2)}{R_t} \tag{11}$$

In eq 11, $r_2$ is another random number generated from a uniform distribution. Profiling results of preliminary simulations showed that the line of code where eq 11 was performed was one of the most expensive lines of the entire simulation. Therefore, its analysis and improvement were worthwhile. It should be noted that profiling data was acquired using Julia's built-in profiler.

One may notice that eq 11 indicates that the time increment is a random number obtained from an exponential distribution with a rate parameter $\lambda = R_t$. That is

$$\Delta t = e_{R_t} = \frac{e_1}{R_t} \tag{12}$$

Here, $e_{R_t}$ is a random number generated according to the exponential distribution with $\lambda = R_t$, and $e_1$ is a random number generated according to the exponential distribution with $\lambda = 1$.

Taking advantage of Julia's standard library, it is possible to generate random numbers from an exponential distribution with $\lambda = 1$, which is $e_1$ in eq 12. The elimination of the logarithm calculation in each iteration improves considerably the simulation time.

*Step 4. Reaction Simulation.* In order to carry out the simulation of the selected chemical reaction, the molecules that will participate in this reaction are chosen in a random way. For instance, for a reaction between species A and B, a molecule of A and a molecule of B are randomly selected from the population of A and B in the control volume. The computation time required for the execution of this step, the random selection of the reacting molecules, is substantial in the total running time of the algorithm. Barner-Kowollik et al.[29] showed that the data structures used to represent and store the different species have a strong influence on this execution time. Therefore, the strategies used are worth explaining.

When the reactant species are unimeric, all molecules of the same species are identical, and randomly choosing a molecule is trivial. Hence, unimeric species can be represented in the algorithm by a single integer variable which stores the number of molecules of a particular species present in the control volume.

The case of polymeric reactants is more complicated because these molecules can have different chain lengths. For example, a polymer of chain length $n$ is different from another one of chain length $m$, even though both belong to the same species. Under these circumstances, there are various options for the representation of these molecules in the algorithm. Two of them will be further explored.

The first option consists of storing each molecule explicitly in a one-dimensional array. The index of the array corresponds to the molecule identity, and the value corresponds to the chain length associated with it. For example, if $P$ is an array that stores the dead polymer molecules, $P(3) = 100$ means that the third dead polymer molecule has a chain length of 100. With this option, the selection of random molecule has $O(1)$ time complexity but $O(N)$ space complexity. This highly desirable constant time complexity means that the time needed for choosing a molecule is independent of the number of molecules in the array and is thus independent of the array length. This is accomplished at the expense of a linear space complexity, which means that the size needed to store the array in memory increases linearly with the number of molecules.

The second option consists of storing a compressed representation, aiming at reducing storage for cases where the number of molecules is large enough to produce a memory overflow error or the computer runs out of memory. This is done at the expense of longer computation times. There are different data structures that would allow storing this compressed data. In this work a so-called *linear* array representation has been implemented. In this representation the index of the array corresponds to the chain length and the value corresponds to the number molecules of that length present in the system. In this case, $P(3) = 100$ would mean that there are 100 dead polymer molecules of chain length 3. For this situation, choosing a random molecule requires performing a linear search through the array with $O(N)$ time complexity but only $O(n_{max})$ constant space complexity, where $n_{max}$ is the maximum chain length of the species being simulated.

Both options were implemented, and the results obtained with both of them were compared. It is worth noting that other representations exist that have been implemented by other authors.[29,38] These options might be considered for more complex polymerization reactions in future works.

Once the reactant molecules have been selected, the reaction is simulated. This step is straightforward and only requires simple arithmetic. First, the number of molecules of the reacted species are subtracted from the control volume and, in a similar manner, the produced species are added to the system. Second, if there are polymeric species involved, the storage arrays need to be updated.

*Step 5. Update.* After the selected reaction has been simulated, the quantities of the molecules involved in that reaction have changed and therefore the reaction rates need to be updated. As explained before, since only a given number of species participate in the selected reaction, not all reaction rates need to be updated. In order to avoid redundant arithmetic operations, only those reaction rates which depend on the number of molecules of the species that were involved in the previous reaction are updated; the rest remain unchanged.

*Step 6. Iteration.* All the steps described up to this point correspond to one iteration of the algorithm. We then repeat from step 2 (Reaction Selection) until a stopping criterion is met. Common stopping criteria used were a final reaction time or conversion reached.

**2.3. Computer Hardware and Software.** All the simulations implemented in this work were run on a regular desktop computer equipped with an Intel Core i5-3330 processor running at 3 GHz and with 8 GB of RAM memory.

Julia Language version 0.3.10 was used to run the Monte Carlo simulations, and the Gadfly v0.3.12 package within the Julia environment was utilized for creating the plots. gPROMS



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

v3.5.3 was used for the deterministic simulations. The algorithm performance comparison was performed using MATLAB version R2011a, and the C and FORTRAN codes were compiled using the GNU Compiler Collection (GCC) version 4.9.2.

## 3. MATHEMATICAL MODEL

Given the theoretical nature of this work, the kinetic mechanism and the values of the kinetic constants used for the simulation are those typically found in the literature. The mathematical model is based on the kinetic mechanism indicated by the equations listed in Table 1.

**Table 1. Kinetic Mechanism of RAFT Polymerization**

| step | reaction |
|---|---|
| initiation | $I \xrightarrow{f,k_d} 2R_0^\bullet$ |
| | $R_0^\bullet + M \xrightarrow{k_{pi}} R_1^\bullet$ |
| propagation | $R_n^\bullet + M \xrightarrow{k_p} R_{n+1}^\bullet$ |
| pre-equilibrium | $R_n^\bullet + T \xrightarrow{k_a} {}^\bullet TR_n$ |
| | ${}^\bullet TR_n \xrightarrow{0.5k_f} R_n^\bullet + T$ |
| | ${}^\bullet TR_n \xrightarrow{0.5k_f} R_0^\bullet + TR_n$ |
| core equilibrium | $R_n^\bullet + TR_m \xrightarrow{k_a} R_n\dot{T}R_m$ |
| | $R_n\dot{T}R_m \xrightarrow{0.5k_f} R_n^\bullet + TR_m$ |
| | $R_n\dot{T}R_m \xrightarrow{0.5k_f} TR_n + R_m^\bullet$ |
| termination | $R_n^\bullet + R_m^\bullet \xrightarrow{k_{tc}} P_{n+m}$ |
| | $R_n^\bullet + R_m^\bullet \xrightarrow{k_{td}} P_n + P_m$ |
| cross-termination[a] | $R_n\dot{T}R_m + R_0^\bullet \xrightarrow{k_{ct}} P_{(n+m)}$ |
| | $R_n\dot{T}R_m + R_s^\bullet \xrightarrow{k_{ct}} P_{(n+m+s)}$ |

[a]The cross-termination reaction is used only in the IRT and IRTO models. For the IRT model $s = 1, ..., \infty$ and for the IRTO model $s = 1, 2$.

The chemical species involved are initiator (I), monomer (M), active radicals with $n$ units of M ($R_n^\bullet$), chain transfer agent (T), inactive radicals with $n$ units of M ($TR_n$), adduct radicals with two arms of lengths $n$ and $m$ ($R_n\dot{T}R_m$), and dead polymer chains of length $n$ ($P_n$). The same set of constants ($k_a$ and $k_f$) was used for both the pre-equilibrium and the core equilibrium; nevertheless, using a different set of values for each of these reaction stages would be straightforward and easily accomplished. The initiation constant was assumed to be equal to the propagation constant ($k_{pi} = k_p$), and the initiator efficiency is denoted as $f$. The simulation of the initiator decomposition reaction requires additional explanation due to the presence of the initiator efficiency factor ($f$). Once this reaction has been selected to be simulated, the number of molecules of the initiator is subtracted by 1 from the control volume. After this step an additional random number in the range [0, 1) is generated to take into consideration the efficiency factor. If this random number is less than or equal to $f$, the reactions proceeds as stated on Table 1, adding 2 molecules of active radicals ($R_0^\bullet$) to the control volume. Otherwise, if the random number is greater than $f$, 2 molecules of inert species are added instead. Kinetic constant values that have been previously reported in the literature[19] were used in this work. These values are detailed in Table 2.

**Table 2. Kinetic Constants and Kinetic Parameters**

| parameter | value |
|---|---|
| $f$ | 0.5 |
| $k_d$ | 0.036 $h^{-1}$ |
| $k_{pi}$ | $3.6 \times 10^6$ $L{\cdot}mol^{-1}{\cdot}h^{-1}$ |
| $k_p$ | $3.6 \times 10^6$ $L{\cdot}mol^{-1}{\cdot}h^{-1}$ |
| $k_a$ | $3.6 \times 10^9$ $L{\cdot}mol^{-1}{\cdot}h^{-1}$ |
| $k_f$[a] | 36 $h^{-1}$ |
| $k_f$[b] | $3.6 \times 10^7$ $h^{-1}$ |
| $k_{tc}$ | $3.6 \times 10^{10}$ $L{\cdot}mol^{-1}{\cdot}h^{-1}$ |
| $k_{td}$ | $3.6 \times 10^{10}$ $L{\cdot}mol^{-1}{\cdot}h^{-1}$ |
| $k_{ct}$[a] | 0 |
| $k_{ct}$[b] | $3.6 \times 10^{10}$ $L{\cdot}mol^{-1}{\cdot}h^{-1}$ |

[a]SF theory. [b]IRT and IRTO theories.

### 3.1. Calculation of Molecular Properties.

Thanks to the straightforward implementation of the MC simulation and the simple representation of polymeric molecules in the algorithm, the calculations of the number-average molecular weight ($\overline{M}_n$) and weight-average molecular weight ($\overline{M}_w$) can be performed using their well-known definitions:

$$\overline{M}_n = \frac{M_{w,mon} \sum_i \sum_j j n_i^j}{\sum_i \sum_j n_i^j} \tag{13}$$

$$\overline{M}_w = \frac{M_{w,mon} \sum_i \sum_j j^2 n_i^j}{\sum_i \sum_j j n_i^j} \tag{14}$$

In eqs 13 and 14, subscript $i$ represents the different polymeric species (i.e., $i$ = active radicals, inactive radicals, adduct radicals, and dead polymer chains), $M_{w,mon}$ is the monomer molecular weight, $j$ is the chain length, and $n_i^j$ is the number of molecules of the polymeric species $i$ with chain length $j$. The value of superscript $j$ goes from 1 up to the maximum chain length of species $i$ within the population of the control volume

The complete molecular weight distribution is calculated in a similar manner:

$$MWD_j = \frac{\sum_i j n_i^j}{\sum_i \sum_j j n_i^j} \tag{15}$$

In eq 15 $MWD_j$ is the molecular weight distribution of the overall population of polymeric species expressed in weight fraction.

## 4. RESULTS AND DISCUSSION

### 4.1. Effect of the Number of Molecules.

As mentioned before, the value of parameter $N$ affects the accuracy and performance of the simulation. This value has to be small enough for the simulation to run in a reasonable time while being sufficiently large to yield reliable and accurate results. In particular, a large enough value is required so that the species with the lowest concentrations are present in statistically significant quantities. However, simulation time increases with $N$. This happens because, as can be seen from eqs 5−7, all reaction rates increase with $N$, and according to eq 11 larger



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

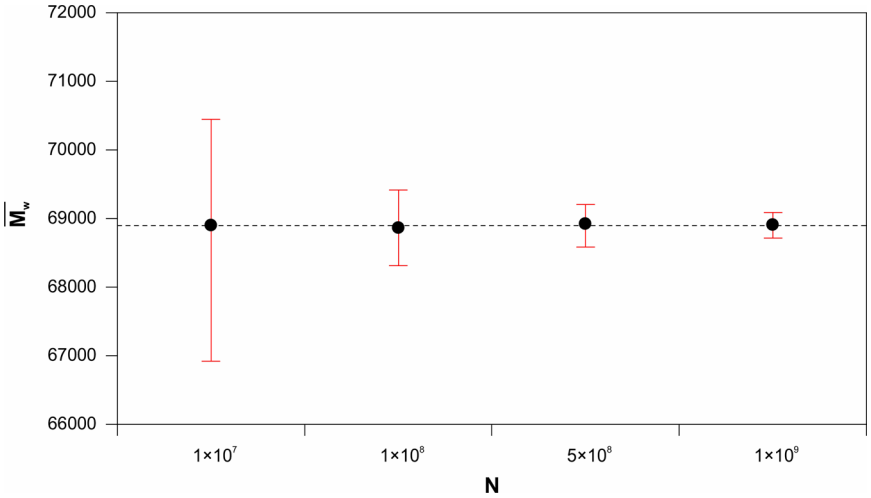



**Figure 1.** Weight-average molecular weight from 100 simulations with different values of $N$ for the SF theory. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, $t_f = 34$ h.

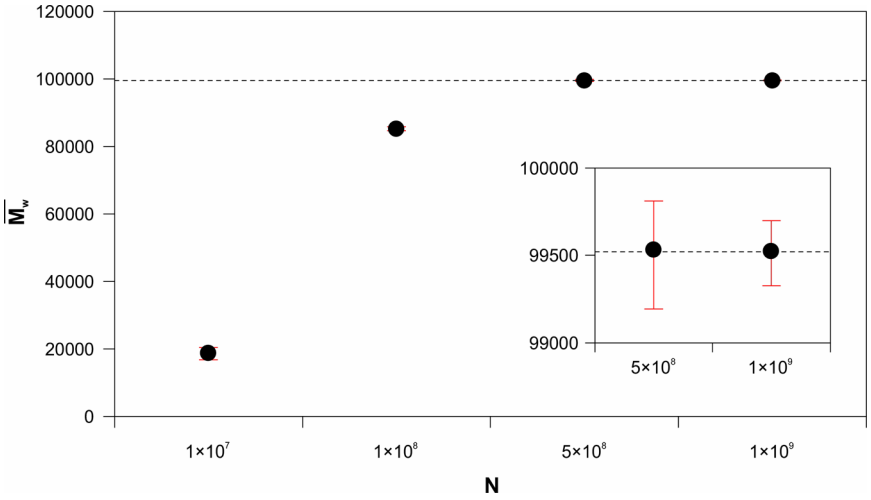



**Figure 2.** Weight-average molecular weight from 100 simulations with different values of $N$ for the IRT theory. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

rates result in smaller time steps. In consequence, as the value of $N$ grows, a greater number of iterations will be required to reach the specified reaction time or conversion.

In order to find optimal values of $N$ for predicting polymer properties, 100 simulations were performed with each of the following values of $N$, [$1 \times 10^7$, $1 \times 10^8$, $5 \times 10^8$, $1 \times 10^9$, $5 \times 10^9$, $1 \times 10^{10}$], for the models based on the three theories under study (SF, IRT, and IRTO). The stopping criterion used was the final reaction time ($t_f$), which was set to 34 h. The initial operating conditions were set to monomer $M_0 = 5$ mol·L$^{-1}$, initiator $I_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and chain transfer agent $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$. Figure 1 shows the outputs of the weight-average molecular weight ($\overline{M_w}$) for the SF model. The dots represent the average values of the 100 simulations, while the error bars are the maximum and minimum values obtained in these simulations. The dashed line represents the value calculated by the deterministic simulation. Similar figures were obtained for the number-average molecular weight ($\overline{M_n}$) and the conversion as well as for different sets of operating conditions. It can be seen that, for this theory, a number of molecules as low as $1 \times 10^8$ was able to represent the system, yielding accurate and reproducible results.

The situation is different with the models responding to the IRT and IRTO theories. Figure 2 summarizes the results for the IRT model. Notice that the plot has been zoomed for the values of $N$ of $5 \times 10^8$ and $1 \times 10^9$. A similar result was obtained with the IRTO model.

The most significant aspect of this data is that values of $N$ smaller than $5 \times 10^8$ fail to return accurate results. It can be noticed that, for the smaller values of $N$, the predicted $\overline{M_w}$ is lower than its actual value; the same was observed for $\overline{M_n}$ and conversion. This was not the case for the SF model, where even simulations performed with $1 \times 10^7$ predicted good values when averaged.

This behavior has been observed previously for MC simulations of free radical polymer systems.[39,40] It happens because the number of molecules of the different moieties in a MC simulation is a discrete value that is a fraction of $N$. The species with lowest concentrations in the reacting system may have zero members in the MC simulation if $N$ is too small, something that affects all reaction rates in which they are involved. In conventional free radical polymerization, these species are the propagating radicals. If $N$ is so small that the number of radicals in the MC simulation becomes zero,



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

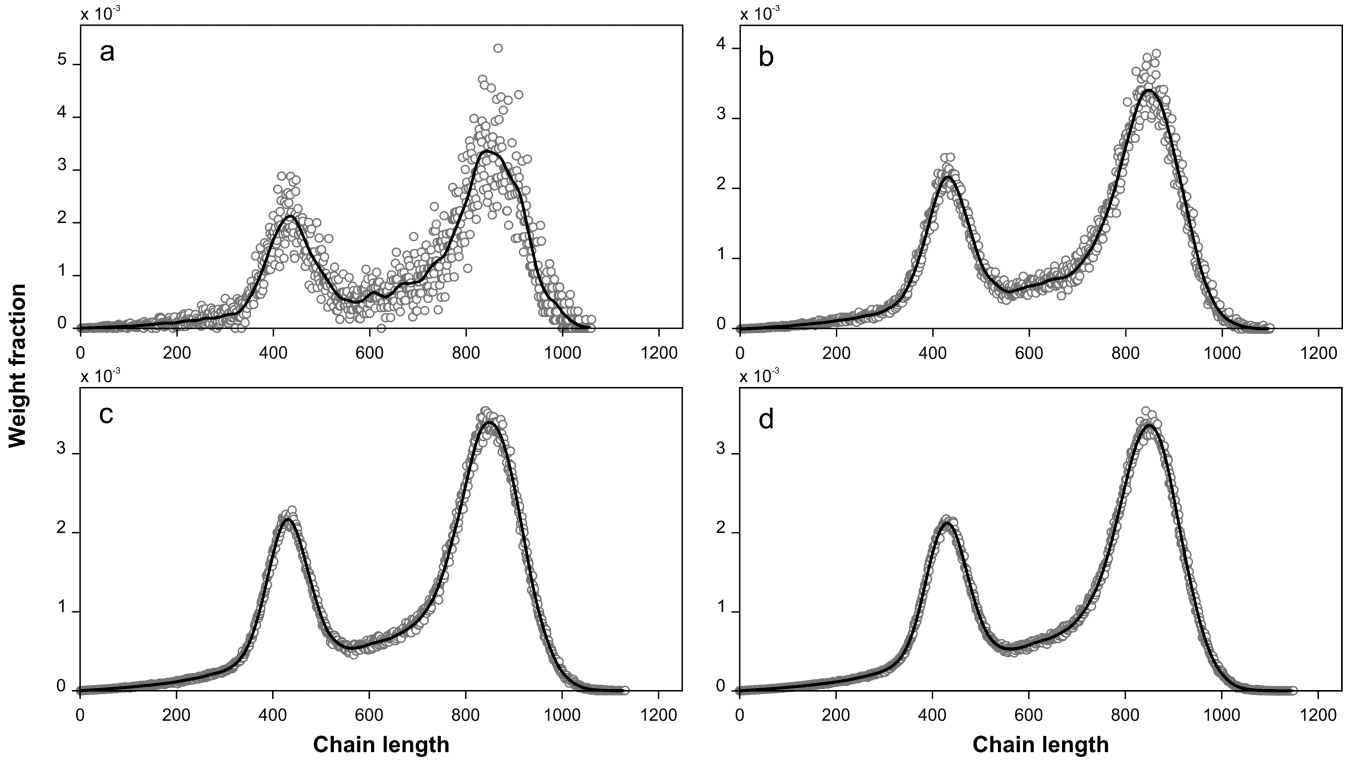



**Figure 3.** Scattered and smoothed MWD for different values of $N$: (a) $1 \times 10^7$, (b) $1 \times 10^8$, (c) $5 \times 10^8$, and (d) $1 \times 10^9$. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

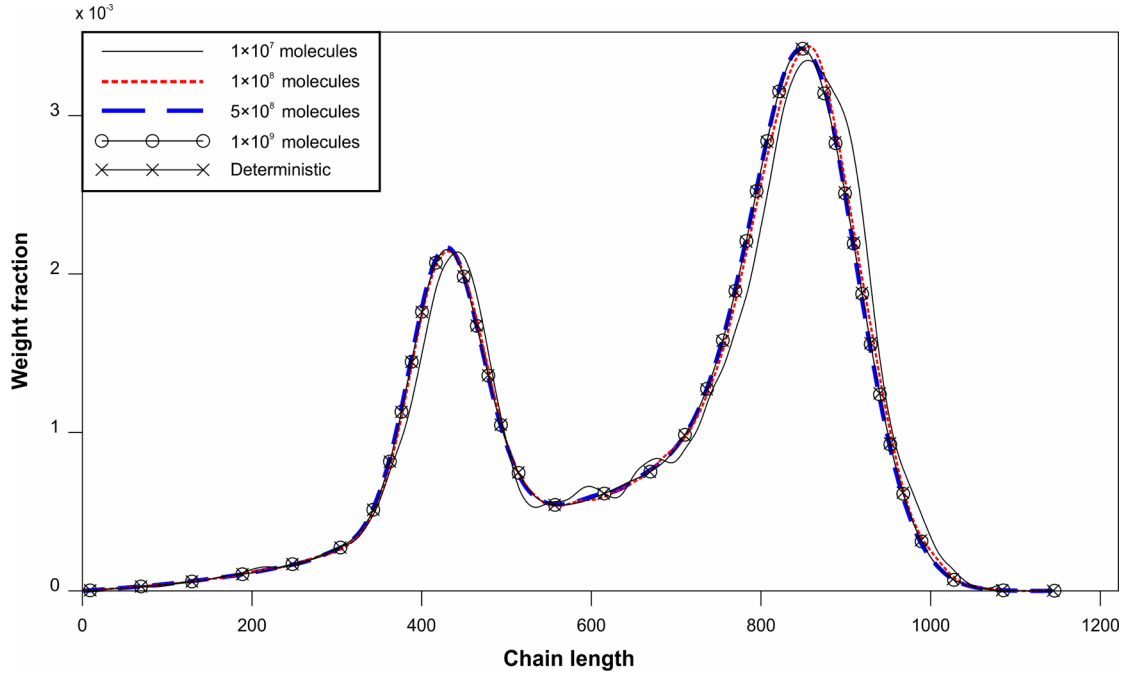



**Figure 4.** Full MWD for the SF theory. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

propagation cannot take place until another initiation event occurs, which artificially reduces conversion. Another effect is that the MC reaction rates in which radicals are involved would be zero as well. This affects the value of $R_t$ considerably, resulting in large time step values. As a consequence, the simulation incorrectly behaves as if the reactions were slower (since large time steps lead to fewer reaction events within the same reaction time). In the case of CRP, molecular weight increases linearly with conversion, so the elimination of radicals due to the mathematical artifact of an excessively small $N$ would lead to molecular weights with values lower than expected. In the IRT or IRTO models, the intermediate adduct fragments quickly and hence its concentration is low, comparable to that of propagating radicals. Therefore, its population may also become zero for low values of $N$. When this happens, the RAFT equilibrium between dormant and active species cannot



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

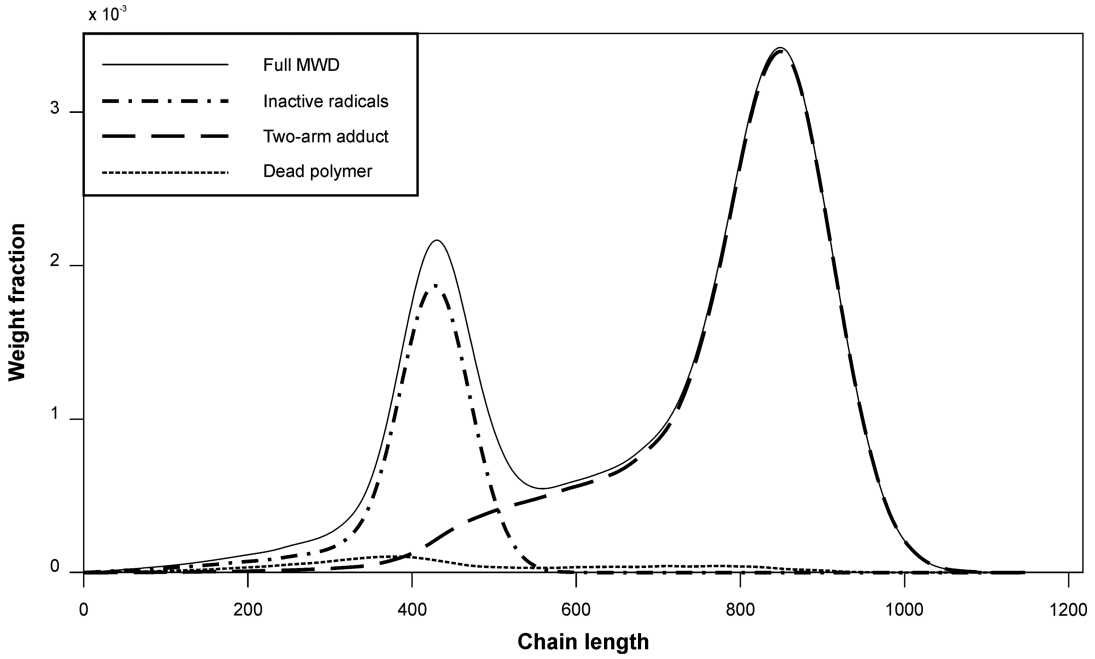



**Figure 5.** Contributions to the full MWD from the individual polymer species. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·$L^{-1}$, and $t_f = 34$ h.

be established, and the MC simulation is not statistically representative of the actual system. On the other hand, in the SF model the intermediate adduct fragments slowly and its concentration is much higher than in the models for the other theories. In consequence, it is not affected so much by relatively low values of $N$. In addition, its fragmentation provides a source of propagating radicals that contributes to maintain a nonzero population of this species.

Another important point to remark is that different $\overline{M}_w$ values have been shown in the previous figures for the SF, IRT, and IRTO models for the same operating conditions. The same situation can be observed for $\overline{M}_n$ and conversion. This is because the same set of kinetic parameters is used for the three theories. In case of aiming at predicting experimental data, each theory would require appropriate fitting of the kinetic parameters.

The effect of the number of molecules $N$ on the prediction of the full MWD was considered next. Figure 3 shows the full MWD obtained with the SF model with values of $N$ between $1 \times 10^7$ and $1 \times 10^9$. The graph includes the scattered output of the MC simulations and a smoothed curved obtained from them. The smoothing was performed by means of a locally weighted scatterplot regression (LOESS) with a smoothing factor $\alpha = 0.10$. As expected, the level of scattering is reduced as $N$ increases.

In order to validate the MC models, we compared the MC results with those of an independent method. Figure 4 shows the smoothed curves obtained from Figure 3, compared with the MWD computed by direct integration of the corresponding deterministic system of differential equations.

We used gPROMS v3.5.3 for solving the differential−algebraic equation (DAE) system by a direct integration method. This proprietary software uses standard mathematical solvers for the solution of mixed sets of differential and algebraic equations. For our particular system the DASOLV solver was used. This solver is based on variable time step, variable order backward differentiation formulas (BDF) and is designed to deal with large, sparse systems. Linear systems are solved with the MA48 solver, designed for large, sparse, and asymmetric systems.

One of the limitations of the direct integration method applied to polymerization reactions is that the size of the resulting EDO system is proportional to the maximum chain length value of all species, so using large chain length values results in large EDO systems. Memory limitations in our computer system constrained our simulation to a maximum chain length value of 1800. For the simulation of the SF system, a value of 1200 was sufficient and yielded a system of 18 082 equations.

As shown in Figure 4, the MWD corresponding to the SF model is bimodal, with peaks at approximately 430 and 850 chain length units. As $N$ increases, the quality of the prediction improves. It may be observed in Figure 4 that using $N = 1 \times 10^7$ results in an MWD that is not accurate, since the positions of the peaks are not predicted correctly, and several oscillations appear in the smoothed distribution at values of chain length larger than 400. These oscillations are caused by the smoothing of the highly scattered data predicted with this value of $N$. Increasing $N$ to $1 \times 10^8$ molecules improves the predicted MWD considerably. However, it still shows some inexactitudes in the position of the second peak.

The MWD obtained with $N = 5 \times 10^8$ and higher reproduce very well the true distribution. Please note the practically exact overlapping between the $N = 1 \times 10^9$ and the deterministic curve. For $N \geq 5 \times 10^8$ all simulations converge to the true distribution. We conclude that $N = 5 \times 10^8$ is a good value for obtaining a representative MWD through the MC simulation of this reaction system by the SF theory. This simulation took 49 s to execute on a regular desktop computer equipped with an Intel Core i5-3330 processor running at 3 GHz and 8 GB of RAM memory and without using parallelization programming techniques.

As an example of additional model predictive capabilities, Figure 5 shows the MWD of the individual polymer species


DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

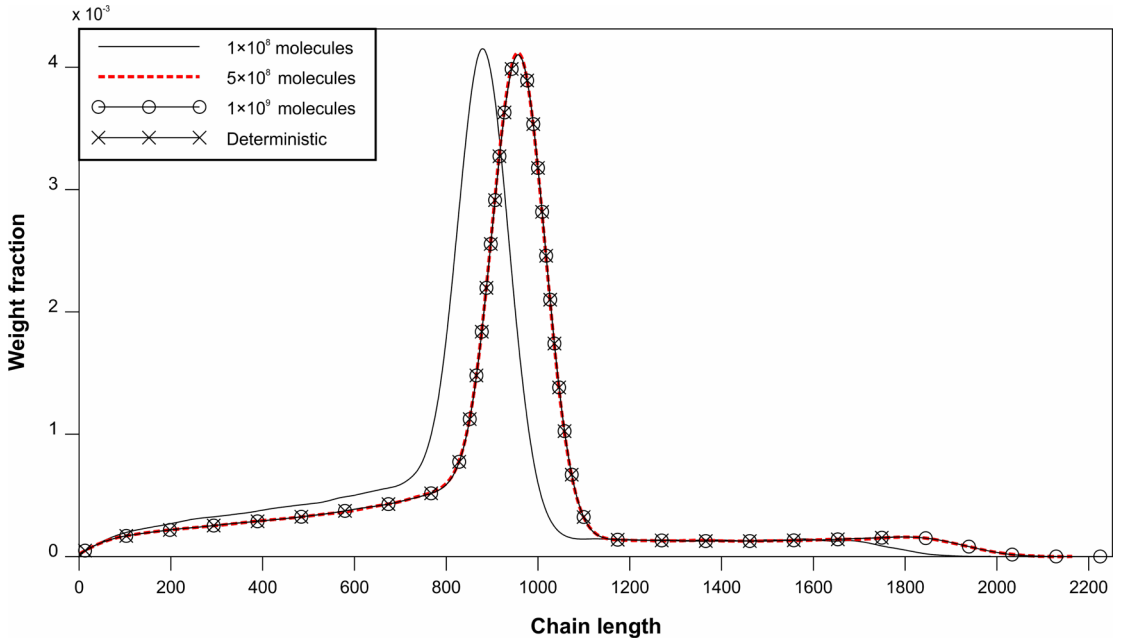



**Figure 6.** Full MWD for the IRTO theory. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

that make up the MWD, i.e. inactive radicals, intermediate two-arm adduct, and dead polymer. It can be seen that the first peak of the bimodal MWD of the global polymer corresponds to the population of inactive radicals, while the second peak corresponds to that of the intermediate two-arm adduct. It can be observed that the position of the second peak is twice the value of the first one, since the adduct is formed by the union of two molecules from the population represented by the first peak. Active radicals were not included in Figure 5 because their concentration is not significant.

The situation for the IRT and IRTO theories is different again. Figure 6 presents the results for the smoothed MWD for the IRTO model. The MWD corresponding to $N = 1 \times 10^7$ was rejected and not included in the graph due to significant noise and inaccuracy. It can be seen that the simulation with $1 \times 10^8$ molecules predicts lower molecular weights compared to the real distribution, something consistent with the average molecular weight predictions. However, starting from $N = 5 \times 10^8$, the distributions converge with good precision to the true distribution. Similar results were obtained with the IRT model.

It should be noted that the IRTO (as well as the IRT) model predicts a unimodal distribution because the fragmentation constant $k_f$ is large and thus the concentration of the two-arm adduct is low. The peak observed in the MWD distribution shown in Figure 6 corresponds to the most abundant species of inactive radicals. The deterministic simulation shown in Figure 6 showed a perfect overlapping with both stochastic simulations (the ones with $5 \times 10^8$ and $1 \times 10^9$ molecules) up to the chain length of 1800. As explained, this is a limitation of the direct integration method and larger values produced out-of-memory errors in our computer system. The resulting system of equations for this model consisted of 21 689 equations. The simulation times for the deterministic solution were 260 s for the SF system (Figure 4) and 420 s for the IRTO system (Figure 6).

**4.2. Bivariate Distribution of the Two-Arm Adduct.** One of the distinguishing characteristics of RAFT polymerizations is the presence of a two-arm intermediate species. This species is an adduct of two growing polymer chains that are linked to the chain transfer agent. Information on the complete MWD of this intermediate adduct may prove useful in studying the different proposed mechanisms for RAFT polymerization. This adduct is described by a two-dimensional (2D) distribution that takes into consideration the chain lengths of each of its arms. The attainment of the 2D distribution by the direct integration approach is prohibitive in terms of memory requirements of regular desktop computers. Its prediction using other deterministic simulations is a very laborious task. For instance, Fortunatti et al.[21] used probability generating functions to model the 2D distribution of the intermediate adduct. To the best of our knowledge, our work is the first to report this 2D distribution obtained from a stochastic simulation.

We show in Figure 7 the bivariate distribution of the two-arm adduct for the same operating conditions used for Figure 4. It can be observed that the position of the peak of this bivariate distribution (chain length of arm 1 = chain length of arm 2 = 425), is half the value of the position of the second peak of the MWD shown in Figure 4. This is so because the MWD in Figure 4 corresponds to the MWD considering the total length of the chains, and most intermediate adduct molecules are formed by two arms of roughly the same length. It is also interesting to note the presence of small tails in the MWD leaving at right angles from the main peak. These tails correspond to adduct molecules that have acquired one arm from the short molecular weight tail and the other one from the most abundant chain length of the inactive species (see Figure 5). The population of adducts with two short arms is extremely low.

This detailed information about the RAFT system is available at no extra simulation cost, in the short time of 49 s.

In the case of the IRT and IRTO models, it is more difficult to obtain the bivariate MWD of the intermediate adduct. Since the concentration of this adduct is much smaller according to these theories than with the SF approach, the number of adduct molecules resulting from MC simulations with $N = 5 \times 10^8$ or 1



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

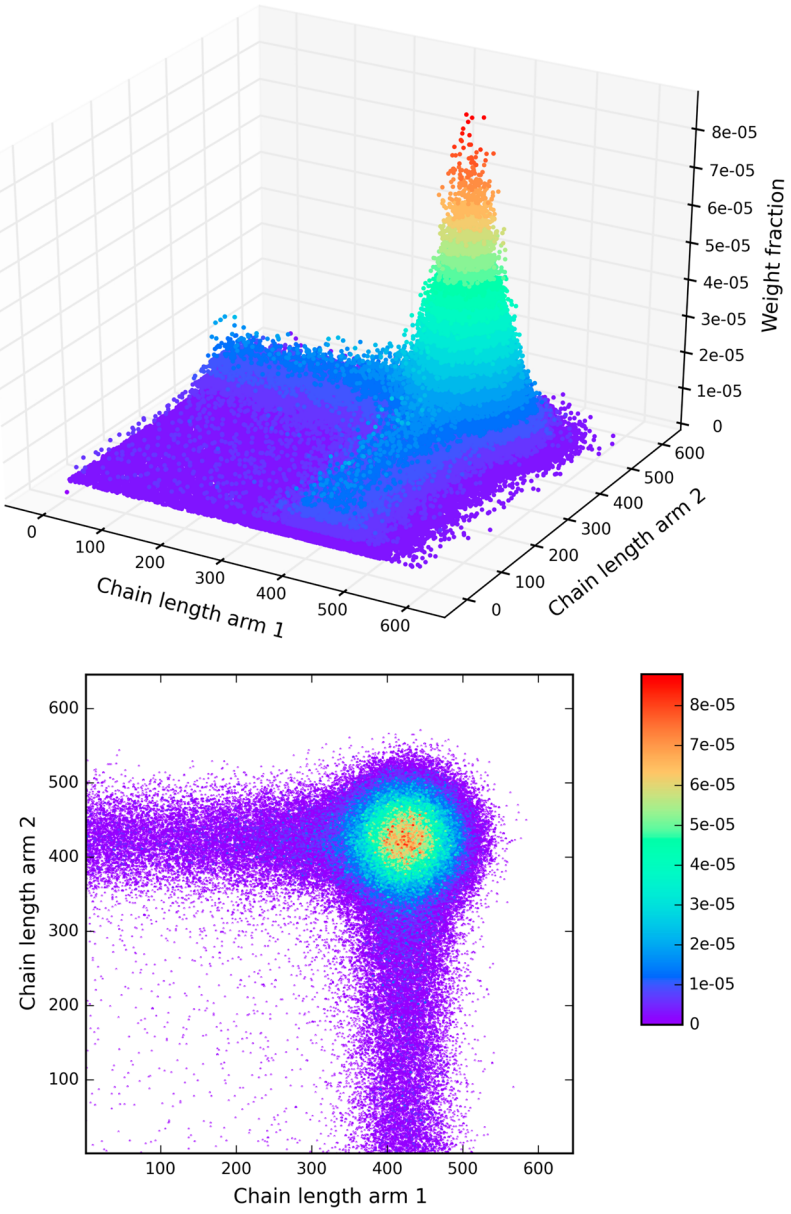



**Figure 7.** Surface and contour plots of the bivariate distribution of the two-arm adduct in the SF theory. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

$\times 10^9$ is too small for visualizing the bivariate distribution. The small number of adduct molecules is too dispersed in the chain length of arm 1-chain length of arm 2 grid to obtain a smooth surface. Even with $N = 1 \times 10^{12}$ the number of adduct molecules is not enough for obtaining the bivariate MWD of this species, since only a single molecule is predicted for most combinations of chain lengths of each arm. The simulation with this value of $N$ took over 7 days of CPU time. A larger value would be necessary for obtaining a smooth bivariate distribution. Nevertheless, even though the number of adduct molecules resulting from the MC simulation of the IRT or IRTO models with $N = 5 \times 10^8$ or $1 \times 10^9$ is insufficient for a good estimation of the bivariate MWD of the intermediate species, it is large enough for predicting accurate values of the global MWD of the polymer and the average molecular properties, as was shown before.

**4.3. Computation Times.** One of the main drawbacks of MC methods is the high computational time required to obtain accurate results. Thanks to the use of Julia and the algorithmic improvements performed, the computational times are generally better than those reported by other authors for similar reaction systems without using parallel computing technologies.[28] The time required for each MC simulation is shown in Table 3. The $O(1)$ representation was used in all the simulations. For this system and for the three theories studied in this work, a simulation using $N = 5 \times 10^8$ that yields good results takes less than 1 min for the SF theory and around 4 min for the IRT and IRTO theories. This time difference could be explained by two main factors. First, the larger number of reactions involved in the IRT and IRTO models makes the reaction selection slightly more time-consuming. Second, the fragmentation constant ($k_f$) is significantly larger in the IRT and the IRTO models than in the SF model. Finally, the cross-termination constant, $k_{ct}$, also takes a large value and it is not present in the SF model. These large kinetic constants contribute to smaller time steps between reaction events,



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

Table 3. Computation Times (in seconds)[a]

| $N$ | SF | IRT | IRTO |
|---|---|---|---|
| $1 \times 10^7$ | 0.84 | 0.34 | 0.5 |
| $1 \times 10^8$ | 9.4 | 25 | 33 |
| $5 \times 10^8$ | 49 | 210 | 271 |
| $1 \times 10^9$ | 110 | 487 | 568 |
| $5 \times 10^9$ | 820 | 2861 | 3031 |
| $1 \times 10^{10}$ | 1752 | 5541 | 6314 |

[a]Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

resulting in a larger number of iterations to reach the final reaction time. For instance, the number of iterations of the SF and IRT simulations for $N = 5 \times 10^8$ is $8.1 \times 10^8$ and $3.3 \times 10^9$, respectively. This is approximately 4 times more iterations for the IRT model than for the SF one, which is roughly the same difference observed in the computation times. Similar numbers were obtained for operating conditions other than those reported in Table 3.

**4.4. Comparison with Other Programming Languages.** Finally, the running time of the two different array representations ($O(1)$) and ($O(N)$) in Julia were compared. Additionally, the $O(1)$ representation in Julia was compared with the implementations written in other languages: Python, MATLAB, FORTRAN, and C. The results for the case of $N = 5 \times 10^8$ and the SF theory are summarized in Table 4.

Table 4. Implementation Comparison[a]

| implementation | time [s] |
|---|---|
| Julia | 49 |
| Julia $O(N)$ | 428 |
| MATLAB | 225 |
| Python | >17000 |
| FORTRAN | 178 |
| C | 71 |

[a]The $O(1)$ representation was used in all the simulations unless stated otherwise. Operating conditions: $M_0 = 5$, $I_0 = 5 \times 10^{-3}$, $CTA_0 = 5 \times 10^{-3}$ mol·L$^{-1}$, and $t_f = 34$ h.

It is worth noting that, in the case of Python, a pure CPython implementation was used employing only the NumPy[41] package for the array representation of the molecules. There are packages for Python that provide speed improvements, such as Cython[42] and Numba; these were not tested but could be considered in future works.

In MATLAB, the option to provide speed improvements comes by writing MEX files which can be loaded and executed by the MATLAB interpreter. MEX files are subroutines written in C, C++, or FORTRAN, which goes against the purpose of using pure high-level languages and thus were not tested in this work. Recent versions of MATLAB make use of a JIT compiler in order to accelerate for loops as well.

It could be argued that these results are not surprising as they coincide with the guesses of an experienced programmer. The fact that this algorithm implementation in Julia is faster than the one written in C could be surprising and criticized as well. An experienced C programmer might find ways to adapt the code to the peculiarities of the language and be able to produce a faster implementation. In order to make this comparison as unbiased as possible, the exact same algorithm was coded on the different languages, without adapting each code to the strengths of each language. The difficulty of each implementation is subjective and depends on the experience of the programmer with each language and also to his or her tendency to a particular programming paradigm. Finally, it is our opinion that the code written in Julia is much easier to read, follow, and debug than that of C or FORTRAN; it provides an outstanding performance, especially compared to those of Python and MATLAB.

## 5. CONCLUSIONS

An efficient kinetic MC simulation was implemented for a RAFT polymerization system in Julia. Three models were developed to simulate and analyze the main mechanistic RAFT theories currently under discussion (SF, IRT, and IRTO). It was shown that the models are able to accurately predict average properties like the number- and weight-average molecular weights and conversion, as well as MWD in just seconds of running time for the SF theory and under 5 min for the IRT and IRTO theories.

Simulations with different numbers of molecules were performed with each of the models, and it was found that for the studied reaction systems $N = 5 \times 10^8$ molecules provides good results for predicting the MWD. It is worth noting that all computation times were obtained using a regular desktop computer equipped with an Intel Core i5-3330 processor and 8 GB of RAM memory and without the use of parallelization techniques.

As a glimpse of the capabilities of MC simulations, the models are also capable of calculating increasingly complex properties like the contributions to the full MWD of the individual polymer species and the full bivariate MWD of the RAFT intermediate adduct. These complex properties are usually very laborious to obtain through deterministic simulations, and to the best of our knowledge, our work is the first to report them as obtained from a stochastic simulation.

It would be interesting to apply the developed MC model to more complex polymerization reactions, such as a RAFT copolymerization system. The resulting polymer would have a microstructure with more features that could be of interest, including copolymer structure, composition, and branching. Due to the larger number of reactions and microstructure alternatives present in a copolymerization reaction system, we could anticipate that a different data structure for the storage of polymeric molecules and a more efficient reaction selection algorithm than those used in this work would be required to maintain an optimal simulation time. The Julia language is more than capable of handling these complications. Modifications to the code are simple to perform, and the model can be easily adapted and modified to simulate different and more complex reaction systems. The postprocessing needed to obtain additional properties should be straightforward as well.

## AUTHOR INFORMATION

### Corresponding Author

*E-mail: masteasuain@plapiqui.edu.ar.

### Notes

The authors declare no competing financial interest.



DOI: 10.1021/acs.iecr.6b01639
Ind. Eng. Chem. Res. 2016, 55, 8534−8547

## ■ ACKNOWLEDGMENTS

The authors wish to thank the National Research Council of Argentina CONICET (PIP 0653) and Universidad Nacional del Sur UNS (PGI 24/M136) for financial support.

## ■ REFERENCES

(1) Jenkins, A. D.; Jones, R. G.; Moad, G. Terminology for Reversible-Deactivation Radical Polymerization Previously Called "controlled" radical Or "living" radical Polymerization (IUPAC Recommendations 2010). *Pure Appl. Chem.* **2009**, *82*, 483−491.

(2) Davis, K. A.; Matyjaszewski, K. *Statistical, Gradient, and Segmented Copolymers by Controlled/Living Radical Polymerizations*; Advances in Polymer Science; Springer: Berlin, 2002.

(3) Grishin, D. F.; Grishin, I. D. Controlled Radical Polymerization: Prospects for Application for Industrial Synthesis of Polymers (Review). *Russ. J. Appl. Chem.* **2011**, *84*, 2021.

(4) Chiefari, J.; Chong, Y. K.; Ercole, F.; Krstina, J.; Jeffery, J.; Le, T. P.; Mayadunne, R. T.; Meijs, G. F.; Moad, C. L.; Moad, G.; Rizzardo, E.; Thang, S. H. Living Free-Radical Polymerization by Reversible Addition-Fragmentation Chain Transfer: The RAFT Process. *Macromolecules* **1998**, *31*, 5559.

(5) Barner-Kowollik, C.; Buback, M.; Charleux, B.; Coote, M. L.; Drache, M.; Fukuda, T.; Goto, A.; Klumperman, B.; Lowe, A. B.; Mcleary, J. B.; Moad, G.; Monteiro, M. J.; Sanderson, R. D.; Tonge, M. P.; Vana, P. Mechanism and Kinetics of Dithiobenzoate-Mediated RAFT Polymerization. I. The Current Situation. *J. Polym. Sci., Part A: Polym. Chem.* **2006**, *44*, 5809.

(6) Klumperman, B.; van den Dungen, E. T. A.; Heuts, J. P. A.; Monteiro, M. J. RAFT-Mediated Polymerization-A Story of Incompatible Data? *Macromol. Rapid Commun.* **2010**, *31*, 1846.

(7) Moad, G.; Chiefari, J.; Krstina, J.; Mayadunne, R. T. A.; Postma, A.; Rizzardo, E.; Thang, S. H.; et al. Living Free Radical Polymerization with Reversible Addition−fragmentation Chain Transfer (the Life of RAFT). *Polym. Int.* **2000**, *49*, 993.

(8) Barner-Kowollik, C.; Quinn, J. F.; Nguyen, T. L. U.; Heuts, J. P. A.; Davis, T. P. Kinetic Investigations of Reversible Addition Fragmentation Chain Transfer Polymerizations: Cumyl Phenyldithioacetate Mediated Homopolymerizations of Styrene and Methyl Methacrylate. *Macromolecules* **2001**, *34*, 7849.

(9) Monteiro, M. J.; de Brouwer, H. Intermediate Radical Termination as the Mechanism for Retardation in Reversible Addition-Fragmentation Chain Transfer Polymerization. *Macromolecules* **2001**, *34*, 349.

(10) Konkolewicz, D.; Hawkett, B. S.; Gray-Weale, A.; Perrier, S. RAFT Polymerization Kinetics: How Long Are the Cross-Terminating Oligomers? *J. Polym. Sci., Part A: Polym. Chem.* **2009**, *47*, 3455.

(11) Ting, S. R. S.; Davis, T. P.; Zetterlund, P. B. Retardation in RAFT Polymerization: Does Cross-Termination Occur with Short Radicals Only? *Macromolecules* **2011**, *44*, 4187.

(12) Goto, A.; Fukuda, T. Kinetics of Living Radical Polymerization. *Prog. Polym. Sci.* **2004**, *29*, 329.

(13) Mastan, E.; Zhou, D.; Zhu, S. Modeling Molecular Weight Distribution and Effect of Termination in Controlled Radical Polymerization: A Novel and Transformative Approach. *J. Polym. Sci., Part A: Polym. Chem.* **2014**, *52*, 639.

(14) Al-Harthi, M. A. Highlight on the Mathematical Modeling of Controlled Free Radical Polymerization. *Int. J. Polym. Sci.* **2015**, *2015*, 1.

(15) Barner-Kowollik, C.; Quinn, J. F.; Morsley, D. R.; Davis, T. P. Modeling the Reversible Addition−fragmentation Chain Transfer Process in Cumyl Dithiobenzoate-Mediated Styrene Homopolymerizations: Assessing Rate Coefficients for the Addition−fragmentation Equilibrium. *J. Polym. Sci., Part A: Polym. Chem.* **2001**, *39*, 1353.

(16) Wulkow, M. The Simulation of Molecular Weight Distributions in Polyreaction Kinetics by Discrete Galerkin Methods. *Macromol. Theory Simul.* **1996**, *5*, 393.

(17) Konkolewicz, D.; Hawkett, B. S.; Gray-Weale, A.; Perrier, S. RAFT Polymerization Kinetics: Combination of Apparently Conflicting Models. *Macromolecules* **2008**, *41*, 6400.

(18) Konkolewicz, D.; Siauw, M.; Gray-Weale, A.; Hawkett, B. S.; Perrier, S. Obtaining Kinetic Information from the Chain-Length Distribution of Polymers Produced by RAFT. *J. Phys. Chem. B* **2009**, *113*, 7086.

(19) Zapata-González, I.; Saldívar-Guerra, E.; Ortiz-Cisneros, J. Full Molecular Weight Distribution in RAFT Polymerization. New Mechanistic Insight by Direct Integration of the Equations. *Macromol. Theory Simul.* **2011**, *20*, 370.

(20) Zapata-González, I.; Saldívar-Guerra, E.; Flores-Tlacuahuac, A.; Vivaldo-Lima, E.; Ortiz-Cisneros, J. Efficient Numerical Integration of Stiff Differential Equations in Polymerisation Reaction Engineering: Computational Aspects and Applications. *Can. J. Chem. Eng.* **2012**, *90*, 804.

(21) Fortunatti, C.; Sarmoria, C.; Brandolin, A.; Asteasuain, M. Modeling of RAFT Polymerization Using Probability Generating Functions. Detailed Prediction of Full Molecular Weight Distributions and Sensitivity Analysis. *Macromol. React. Eng.* **2014**, *8*, 781.

(22) Fortunatti, C.; Sarmoria, C.; Brandolin, A.; Asteasuain, M. Prediction of the Full Molecular Weight Distribution in RAFT Polymerization Using Probability Generating Functions. *Comput. Chem. Eng.* **2014**, *66*, 214.

(23) Meimaroglou, D.; Kiparissides, C. Review of Monte Carlo Methods for the Prediction of Distributed Molecular and Morphological Polymer Properties. *Ind. Eng. Chem. Res.* **2014**, *53*, 8963.

(24) Brandão, A. L. T.; Soares, J. B. P.; Pinto, J. C.; Alberton, A. L. When Polymer Reaction Engineers Play Dice: Applications of Monte Carlo Models in PRE. *Macromol. React. Eng.* **2015**, *9*, 141.

(25) Gillespie, D. T. Exact Stochastic Simulation of Coupled Chemical Reactions. *J. Phys. Chem.* **1977**, *81*, 2340.

(26) Tobita, H. Molecular Weight Distribution in Free Radical Polymerization with Long-Chain Branching. *J. Polym. Sci., Part B: Polym. Phys.* **1993**, *31*, 1363.

(27) Drache, M.; Schmidt-Naake, G.; Buback, M.; Vana, P. Modeling RAFT Polymerization Kinetics via Monte Carlo Methods: Cumyl Dithiobenzoate Mediated Methyl Acrylate Polymerization. *Polymer* **2005**, *46*, 8483.

(28) Drache, M.; Drache, G. Simulating Controlled Radical Polymerizations with mcPolymer—A Monte Carlo Approach. *Polymers* **2012**, *4*, 1416.

(29) Chaffey-Millar, H.; Stewart, D.; Chakravarty, M. M. T.; Keller, G.; Barner-Kowollik, C. A Parallelised High Performance Monte Carlo Simulation Approach for Complex Polymerisation Kinetics. *Macromol. Theory Simul.* **2007**, *16*, 575.

(30) Tobita, H. Monte Carlo Simulation of Emulsion Polymerization—linear, Branched, and Crosslinked Polymers. *Acta Polym.* **1995**, *46*, 185.

(31) Gillespie, D. T. Approximate Accelerated Stochastic Simulation of Chemically Reacting Systems. *J. Chem. Phys.* **2001**, *115*, 1716.

(32) Bezanson, J.; Karpinski, S.; Shah, V. B.; Edelman, A. Julia: A Fast Dynamic Language for Technical Computing. 2012, arXiv:1209.5145 [cs.PL], ArXiv.org.

(33) Bezanson, J.; Chen, J.; Karpinski, S.; Shah, V.; Edelman, A. Array Operators Using Multiple Dispatch: A Design Methodology for Array Implementations in Dynamic Languages. 2014, arXiv:1407.3845 [cs.PL], ArXiv.org.

(34) Bezanson, J.; Edelman, A.; Karpinski, S.; Shah, V. B. Julia: A Fresh Approach to Numerical Computing. 2014, arXiv:1411.1607 [cs.MS], ArXiv.org.

(35) Saito, M.; Matsumoto, M. Simd-Oriented Fast Mersenne Twister: A 128-Bit Pseudorandom Number Generator. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*; Springer: 2008.

(36) Gibson, M. A.; Bruck, J. Efficient Exact Stochastic Simulation of Chemical Systems with Many Species and Many Channels. *J. Phys. Chem. A* **2000**, *104*, 1876.



DOI: 10.1021/acs.iecr.6b01639
*Ind. Eng. Chem. Res.* 2016, 55, 8534−8547

(37) Cao, Y.; Li, H.; Petzold, L. Efficient Formulation of the Stochastic Simulation Algorithm for Chemically Reacting Systems. *J. Chem. Phys.* **2004**, *121*, 4059.
(38) Van Steenberge, P. H. M.; D'hooge, D. R.; Reyniers, M.-F.; Marin, G. B. Improved Kinetic Monte Carlo Simulation of Chemical Composition-Chain Length Distributions in Polymerization Processes. *Chem. Eng. Sci.* **2014**, *110*, 185.
(39) Gao, H.; Oakley, L. H.; Konstantinov, I. A.; Arturo, S. G.; Broadbelt, L. J. Acceleration of Kinetic Monte Carlo Method for the Simulation of Free Radical Copolymerization through Scaling. *Ind. Eng. Chem. Res.* **2015**, *54*, 11975.
(40) Ali Parsa, M.; Kozhan, I.; Wulkow, M.; Hutchinson, R. A. Modeling of Functional Group Distribution in Copolymerization: A Comparison of Deterministic and Stochastic Approaches. *Macromol. Theory Simul.* **2014**, *23*, 207.
(41) Van Der Walt, S.; Colbert, S. C.; Varoquaux, G. The NumPy Array: A Structure for Efficient Numerical Computation. *Comput. Sci. Eng.* **2011**, *13*, 22.
(42) Behnel, S.; Bradshaw, R.; Citro, C.; Dalcin, L.; Seljebotn, D. S.; Smith, K. Cython: The Best of Both Worlds. *Comput. Sci. Eng.* **2011**, *13*, 31.



DOI: 10.1021/acs.iecr.6b01639
*Ind. Eng. Chem. Res.* 2016, 55, 8534−8547