공정 시뮬레이션 출력변수의 확률분포 계산 알고리즘

An Algorithm for Calculation of Probability Distributions of Output Variables in Process Simulation

최 수 형
(Soo Hyoung Choi)

Abstract : Stochastic process analysis is often based on Monte Carlo simulations. As a more rigorous alternative, a deterministic algorithm based on numerical integration is proposed in this paper, which calculates the probability distributions of dependent random variables using the results of simulation with grid points of independent random variables. For performance evaluation, the proposed algorithm is applied to an example problem which can be analytically solved, and the result is compared with that of Monte Carlo simulation. The proposed algorithm is suitable for general process simulation problems with a few independent random variables, and expected to be applicable to areas such as safety analysis and quality control.

Keywords : process simulation, probability distribution, stochastic simulation, numerical integration

I. 서론

확률적 시뮬레이션 기법[1]은 소립자물리에서 거시경제까지 다양한 분야에 적용되고 있다. 확률적 공정 시뮬레이션은 주로 공정모델 및 외부조건에 불확실성이 존재할 때 수행되며 일반적으로 입력변수들의 확률분포로부터 시작하여 출력변수들의 확률분포를 구하는 과정이다. 출력변수들 중 임의의 하나를 다음과 같이 나타내기로 하자.

$$y = g(x_1, \ldots, x_n) \tag{1}$$

여기서 g는 공정에 대한 모델 함수이다. 또 $x_1, \ldots, x_n$은 입력변수, 즉 독립 확률변수로서, 이들의 확률분포는 $f_1(x_1), \ldots, f_n(x_n)$으로 주어진다. 이때 출력변수, 즉 종속 확률변수의 누적분포함수는 다음과 같이 정의된다.

$$F(y) = \int \cdots \int_{D(y)=\{(x_1,\ldots,x_n)\,|\,g(x_1,\ldots,x_n)\le y\}} f_1(x_1)\cdots f_n(x_n)dx_1\cdots dx_n \tag{2}$$

이 적분을 계산할 수 있다면 출력변수의 확률분포는 다음 식으로부터 구할 수 있다.

$$f(y) = dF(y) / dy \tag{3}$$

위 (2)에서 만약 $f_1, \ldots, f_n$, 및 g가 간단한 함수들이라면 해석적 적분이 가능하다. 그러나 많은 공정 시뮬레이션 문제에 있어서 g는 명시적으로 표현할 수 있는 함수가 아니다. 따라서 수치해석적 방법이 요구되며 본 논문에서는 두 가지 후보 (1) 몬테카를로 시뮬레이션 및 (2) 수치적분 방법을 고려한다.

몬테카를로 시뮬레이션은 주어진 입력변수들의 확률분포에 따라 표본을 추출하고 이를 근거로 시뮬레이션을 수행하는 방법이다. 이 방법은 어떤 결과를 그것이 일어날 확율대로 생성하므로 출력변수의 누적분포함수는 다음과 같이 근사화 할 수 있다.

$$F(y) \approx N(g(x_1, \ldots, x_n) \le y) / N \tag{4}$$

여기서 N은 괄호 안의 조건을 만족하는 표본 점 개수이다. 따라서 방법 1은 출력변수의 확률분포를 개략적으로 산정하는 데 효과적으로 사용될 수 있다. 그러나 만약 우리의 목표가 어떤 희귀한 사건 상황의 확률을 예측하는 것이라면 이 방법은 비효율적이 된다. 왜냐하면 그 사건이 충분히 여러 차례 발생할 때까지 시뮬레이션을 반복해야 하기 때문이다. 이와 달리 방법 2는 결정론적 계산에만 기초를 두고 있으며 따라서 인위적으로 설정한 극한상황에서의 시뮬레이션 결과들을 사용할 수 있다. 일반적으로 방법 2가 더 정밀할 수 있으나 만약 독립 확률변수들이 많다면 계산량이 너무 많아지게 된다. 이 경우 몬테카를로 적분법[2]을 적용할 수 있다. 이 방법은 미리 설정된 격자를 사용하지 않고 표본 점들을 추출하여 이 점들에서의 함수 값들로 적분을 수행한다. 여기에 속하는 가장 기초적인 알고리즘은 표본 추출시 평탄한 확률분포를 사용하며 이는 방법 1 및 2의 절충안이라 볼 수 있다.

본 논문에서는 격자를 사용하는 수치적분 방법에 기초하여 소수의 독립 확률변수가 존재하는 문제들에 적합한 알고리즘을 제안하고자 한다. 성능평가를 위하여 이 알고리즘은 (2)의 적분을 해석적으로 구할 수 있는 예제로서 초기농도와 반응시간이 삼각 확률분포로 주어지는 일차반응 시스템 문제에 적용된다. 이 결과를 몬테카를로 시뮬레이션 결과 및 해석해와 비교한다. 제안된 알고리즘은 몬테카를로 시뮬레이션보다 더 정밀한 대안이며 안전성 해석 및 품질제어와 같은 분야에 적용할 수 있을 것으로 기대된다.

II. 제안된 알고리즘

1. 격자설정

독립 확률변수 x_i의 유효범위를 지정하고 이를 일정간격 Δx_i

논문접수 : 2002. 6. 25., 채택확정 : 2002. 8. 28.
최수형 : 전북대학교(soochoi@chonbuk.ac.kr)
※ 본 연구는 서울대학교 화학공정신기술연구소에 의하여 지원되었습니다.

로 나눈다. 이때 양 끝점 및 간격 사이 점들에서 시뮬레이션 하기로 한다. 따라서 변수 x_i의 간격 개수를 m_i라 하면 수행해야 할 시뮬레이션 경우의 수는 다음과 같다.

$$N = \prod_{i=1}^{n}(m_i + 1) \tag{5}$$

이렇게 결정된 각 점 $\mathbf{x}_k = (x_{1k}, \ldots, x_{nk})$, $k = 1, \ldots, N$ 에 대하여 사다리꼴 적분공식을 적용하기 위한 가중치를 다음과 같이 계산한다.

$$w_k = \left(\frac{1}{2}\right)^{n_k} \tag{6}$$

여기서 n_k는 점 $\mathbf{x}_k$ 의 원소들 중 최소 또는 최대 경계 값을 갖는 변수들의 개수이다.

2. 시뮬레이션

위에서 정한 입력변수 점 $\mathbf{x}_k$에 대하여 공정모델을 이용하여 다음과 같이 출력변수 값 y_k를 구한다.

$$y_k = g(\mathbf{x}_k) \tag{7}$$

이로부터 배열 $(w_k, \mathbf{x}_k, y_k)$, $k = 1, \ldots, N$을 얻는다.

3. 누적분포 결정

각 점 $\mathbf{x}_k$에 대하여 다음과 같이 적분에 사용될 항을 계산한다.

$$\Delta F_k = w_k \prod_{i=1}^{n} f_i(x_{ik})\Delta x_i \tag{8}$$

이로부터 배열 $(y_k, \Delta F_k)$, $k = 1, \ldots, N$을 얻는다. 이 배열을 y_k 값 오름차순으로, 즉 다음과 같이 정렬한다.

$$y_1 \leqq \ldots \leqq y_N \tag{9}$$

이제 다음과 같이 누적분포를 계산할 수 있다.

$$F(y_k) \approx F_k = \sum_{j=1}^{k} \Delta F_j \tag{10}$$

여기서 $F_N \approx 1$이며 $F_N = 1$로 만들기 위하여 다음과 같이 보정한다.

$$F(y_k) = F_k \leftarrow F_k / F_N \tag{11}$$

이제 누적분포 $F(y)$를 나타내는 N 개의 점 (y_k, F_k)를 얻었다.

지금까지 적분방법으로 차수가 낮은 사다리꼴 공식을 사용한 이유는 다음과 같다. 만약 심프슨 공식을 쓴다면 일부 점 $\mathbf{x}_j$는 가중치 $w_j = (4/3)^n$을 갖는다. 이 점들이 $F(y_k)$를 구하기 위한 (2)의 적분영역 $D(y_k)$ 내 정당한 위치에 속한다면 문제가 없겠으나 적분영역 경계에 가깝다면 (10)에서 이 가중치를 사용한 ΔF_j 는 과대평가된 것이다. 적분영역 경계를 파악하여 가중치를 조절하면 이상적이겠으나 본 연구에서는 (6)을 사용하여 일부 변수가 최소 또는 최대인 확실한 경계 점들을 제외하고는 가중치를 $w_j = 1$로 일정하게 함으로써 적분영역 경계에서 낮아져야 할 가중치가 오히려 높아지는 것을 막는 방법을 택하였다.

4. 확률분포 계산

확률분포 $f(y)$는 (3)에서와 같이 $F(y)$를 y로 미분함으로써 얻을 수 있다. 문제는 위에서 얻은 N개의 점들은 $F(y)$에는 가까우나 이들을 모두 지나는 곡선의 미분은 대개 실제 $f(y)$와 전혀 다르다는 점이다. 이를 해결하기 위한 한 방법으로 본 논문에서 제안하는 방법은 다음과 같다. 우선 출력변수 구간 $y_1 \leqq y \leqq y_N$을 M ($< N$)개의 일정간격 Δy로 나눈다. 양 끝점 및 간격 사이 점들을 y^i, $i = 0, \ldots, M$이라 하면 $F(y^i) = F^i$ 값은 위에서 구한 (y_k, F_k) 배열을 이용하여 보간법으로 구한다. 가장 간단한 방법은 다음과 같은 선형 보간법이다.

$$F^i = \frac{F_{k+1} - F_k}{y_{k+1} - y_k}\left(y^i - y_k\right) + F_k \quad , i = 1, \ldots, M-1$$
$$F^0 = 0, F^M = 1 \tag{12}$$

여기서 y_k 및 y_{k+1}은 y^i를 사이에 둔 가까운 두 점이다. 확률분포 $f(y^i)$ f^i 값은 다음과 같이 수치미분으로 계산한다.

$$f^i = \frac{F^{i+1} - F^{i-1}}{y^{i+1} - y^{i-1}} \quad , i = 1, \ , M-1$$
$$f^0 = f^M = 0 \tag{13}$$

이 값들을 이용하여 전체구간에 대하여 심프슨 공식을 사용하여 적분하면 다음과 같다.

$$F = \left[\left(f^0 + f^M\right) + 4\sum_{i=1}^{m} f^{2i-1} + 2\sum_{i=1}^{m-1} f^{2i}\right]\frac{\Delta y}{3} \tag{14}$$

여기서 $M = 2m$이다. 이때 $F \approx 1$이며 $F = 1$로 만들기 위하여 다음과 같이 보정한다.

$$f(y^i) = fi \leftarrow fi / F \tag{15}$$

이로써 확률분포 $f(y)$를 나타내는 $M + 1$개의 점 (y^i, f^i)를 얻었다. 문제는 적절한 M 값인데 주어진 문제의 특성에 따라 다르겠으나 (5)의 m_i 값들과 비슷한 규모로 설정하는 것이 합리적이다.

III. 사례연구

제안된 알고리즘의 성능을 평가하기 위하여 간단한 예제를 풀어보자. 일차반응 A → B가 일어나는 시스템에서 반응속도상수는 $k = 1\ \text{min}^{-1}$이다. 반응물 초기농도는 $C_{A0} = 1$ mol/L ± 100% 이고 반응시간은 $\tau = 1$ min ± 100% 이며 다음과 같이 삼각 확률분포를 가정한다.

$$f_1(C_{A0}) = \begin{cases} C_{A0}, & 0 \leq C_{A0} \leq 1 \\ 2 - C_{A0}, & 1 < C_{A0} \leq 2 \end{cases} \tag{16}$$

$$f_2(\tau) = \begin{cases} \tau, & 0 \leq \tau \leq 1 \\ 2 - \tau, & 1 < \tau \leq 2 \end{cases} \tag{17}$$

반응물의 최종농도를 계산하면 다음과 같다.

$$C_A = g(C_{A0}, \tau) = C_{A0}\, e \tag{18}$$

목표는 최종농도의 확률분포 $f(C_A)$를 구하는 것이다.

1. 해석해

(2)의 해석적 적분에 의해 구한 누적분포 및 τ 이를 미분한 확률분포는 다음과 같다.

For $0 \leq C_A \leq e^{-2}$,

$$F(C_A) = (e^4 - 2e^2 + 1)\, C_A^2 / 8$$
$$f(C_A) = (e^4 - 2e^2 + 1)\, C_A / 4$$

For $e^{-2} < C_A \leq = 2e^{-2}$,

$$F(C_A) = -(e^4 + 2e^2 - 1)\, C_A^2 / 8 + 2e^2\, C_A - (\ln C_A)^2 / 2 - (7/2) \ln C_A - 27/4$$
$$f(C_A) = -(e^4 + 2e^2 - 1)\, C_A / 4 + 2e^2 - (\ln CA + 7/2) / C_A$$

For $2e^{-2} < C_A \leq e^{-1}$,

$$F(C_A) = -(2e^2 - 1)\, C_A^2 / 8 + (\ln C_A)^2 / 2 - (7/2 - 2 \ln 2) \ln C_A + (\ln 2)^2 - 7 \ln 2 + 27/4$$
$$f(C_A) = -(2e^2 - 1)\, C_A / 4 + (\ln C_A - 2 \ln 2 + 7/2) / C_A$$

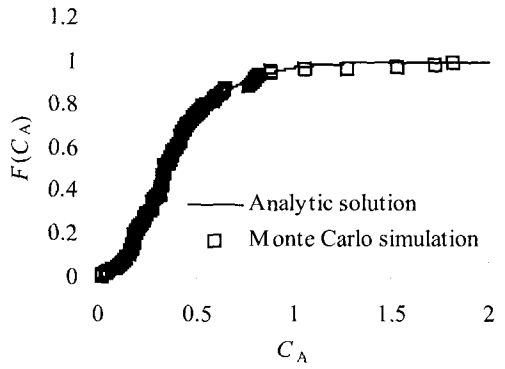


그림 1. 예제의 누적분포.

Fig. 1. Cumulative distribution of the example problem.

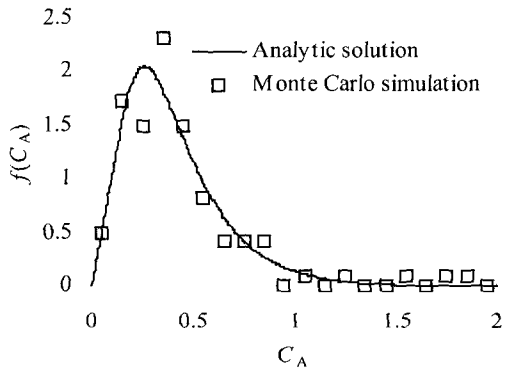


그림 2. 예제의 확률분포.

Fig. 2. Probability distribution of the example problem.

For $e^{-1} < C_A \leq 2e^{-1}$,

$$F(C_A) = (2e^2 + 1)\, C_A^2 / 8 - 4e\, C_A + (3/2)(\ln C_A)^2 - (2 \ln 2 - 17/2) \ln C_A + (\ln 2)^2 - 7 \ln 2 + 57/4$$
$$f(C_A) = (2e^2 + 1)\, C_A / 4 - 4e + (3 \ln C_A - 2 \ln 2 + 17/2) / C_A$$

For $2e^{-1} < C_A \leq = 1$,

$$F(C_A) = C_A^2 / 8 - (\ln C_A)^2 / 2 + (2 \ln 2 - 3/2) \ln C_A - (\ln 2)^2 + 3 \ln 2 - 3/4$$
$$f(C_A) = C_A / 4 - (\ln C_A - 2 \ln 2 + 3/2) / C_A$$

For $1 < CA \leq 2$,

$$F(C_A) = -C_A^2 / 8 + 2\, C_A - (\ln C_A)^2 + (2 \ln 2 - 3) \ln C_A - (\ln 2)^2 + 3 \ln 2 - 5/2$$
$$f(C_A) = -C_A / 4 + 2 - (2 \ln C_A - 2 \ln 2 + 3) / C_A$$

이들 누적분포 및 확률분포의 그래프는 각각 그림 1 및 2에 실선으로 표시되어 있다.

2. 몬테카를로 시뮬레이션

입력변수 확률분포 $f(x)$에 따라 표본 값 x를 추출하려면 0과 1 사이의 난수 u를 생성하여 $x = F^{-1}(u)$로 놓으면 된다. 여기서 F^{-1}는 누적분포의 역함수이다. 이 방법으로 (16) 및 (17)에 따라 $N = 121$ 개의 표본 점 (C_{A0}, τ)를 생성한 후 이 점들에서의 C_A 값들을 (18)로부터 구하였다. (4)를 적용하여 구한 누적분포함수는 그림 1과 같다. 출력변수의 확률분포함수는 다음과 같이 구한다

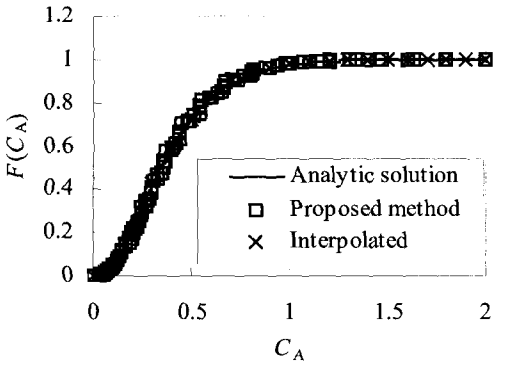


그림 3. 제안된 방법의 누적분포.

Fig. 3. Cumulative distribution by proposed method.

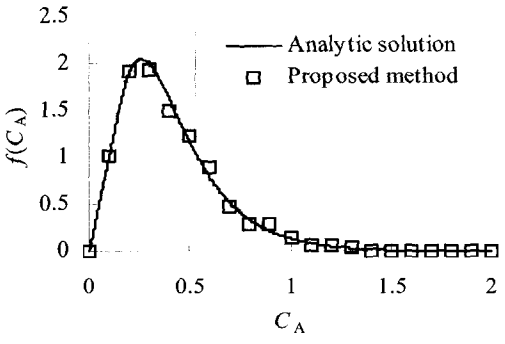


그림 4. 제안된 방법의 확률분포.

Fig. 4. Probability distribution by proposed method.

과 같이 구한다. 변수 y의 범위를 M개의 일정구간Δy 로 나누고 각 구간에 속하는 결과 개수 N_i를 센다. 각 구간의 중심점 y_i에서의 확률밀도는 $f_i = (N_i / N) / \Delta y$이다. 범위 $0 \leq C_A \leq 2$를 $M = 20$개로 나누어 구한 C_A의 확률분포함수는 그림 2와 같다.

3. 제안된 방법

입력변수 C_{A0} 및 τ 의 범위를 각각 $m_1 = m_2 = 10$개의 구간으로 나누었다. 따라서 출력변수 C_A를 계산해야 할 격자 점들은 (5)에 의하여 모두 $N = 121$개이다. 이 점들을 정렬하여 얻은 누적분포함수는 그림 3과 같다. 여기에 $M = 20$ 개로 구간을 나누어 선형보간법을 적용한 점들도 표시하였다. 이 점들을 근거로 수치미분을 하여 구한 확률분포함수는 그림 4와 같다.

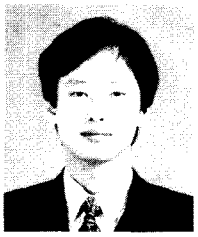
IV. 결론

예제를 통하여 몬테카를로 시뮬레이션과 제안된 수치적분 방법을 비교해 보았다. 출력변수 계산회수가 동일하므로 계산량은 비슷하다고 볼 수 있으나 제안된 방법이 더 정확한 결과를 내었다. 주된 이유는 몬테카를로 시뮬레이션이 확률 높은 상황에 편중되는 것과 달리 제안된 방법은 전 영역에 걸친 상황을 골고루 다루기 때문이다. 일반적으로 확률적 공정해석은 확률적 방법인 몬테카를로 시뮬레이션에 기반을 두는 경우가 많다. 이 방법은 어려운 수학을 피할 수 있는 편리한 도구이기는 하나 반드시 최선책이라고는 할 수 없다. 가능하다면 본 논문에서 제안한 것과 같이 수학적 기법이 상대적으로 많이 포함된 결정론적 방법을 사용함으로써 보다 효율적으로 정확한 결과를 얻을 수 있기 때문이다.

참고문헌

[1] N. Balakrishnan, V. B. Melas, and S. Ermakov, editors, *Advances in Stochastic Simulation Methods*, Birkhuser, Boston, 2000.

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C++: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, 2002.



최 수 형

1961년 1월 10일 생. 1984년 서울대학 화학공학과(공학사). 1986년(공학석사). 1990년University of Missouri-Rolla(Ph. D.). 1993년~현재 전북대학교 화학공학부 교수. 관심분야는 공정시스템.