ADAPTIVE WICK–MALLIAVIN APPROXIMATION TO NONLINEAR SPDEs WITH DISCRETE RANDOM VARIABLES*

MENGDi zhENG†, Boris RozovsKy†, and George E. KArnIadakiS†

Abstract. We propose an adaptive Wick–Malliavin (WM) expansion in terms of the Malliavin derivative of order \( Q \) to simplify the propagator of general polynomial chaos (gPC) of order \( P \) (a system of deterministic equations for the coefficients of gPC) and to control the error growth with respect to time. Specifically, we demonstrate the effectiveness of the WM method by solving a stochastic reaction equation and a Burgers equation with several discrete random variables. Exponential convergence is shown numerically with respect to \( Q \) when \( Q \geq P - 1 \). We also analyze the computational complexity of the WM method and identify a significant speedup with respect to gPC, especially in high dimensions.

Key words. discrete stochastic process, general polynomial chaos, high dimensions, Wiener–Askey scheme

AMS subject classifications. 33C45, 60H07, 60H15, 60H35, 68U20

DOI. 10.1137/140975930

1. Introduction. Nonlinear SPDEs with discrete random variables (RVs) and jump processes are of practical use, since sources of stochastic excitations including uncertain parameters and boundary/initial conditions are typically observed at discrete values. Many complex systems of fundamental and industrial importance are significantly affected by the underlying fluctuations/variations in random excitations, such as the stochastic-volatility jump-diffusion model in mathematical finance [3, 4, 8, 9, 10, 51], stochastic simulation algorithms for modeling diffusion, reaction, and taxis in biology [14], truncated Levy flight model in turbulence [27, 35, 37, 46], quantum-jump models in physics [12], etc. Numerically, nonlinear SPDEs with discrete processes are often solved by general polynomial chaos (gPC) involving a system of coupled deterministic nonlinear equations [50], or the probabilistic collocation method (PCM) [17, 49, 52] involving \( N \) nonlinear corresponding PDEs obtained at the collocation points. In this paper, we use the Wick–Malliavin (WM) approximation [47] (defined in section two) of nonlinear SPDEs to improve the efficiency of gPC.

The Wick product was first proposed as a renormalization technique in quantum field theory [48], hence a nonlinear equation driven by singular noise can be interpreted in the renormalized Wick form, e.g., the viscous Burgers equation with a white noise source [19, 25]. Subsequently, Wick calculus was introduced in stochastic analysis [20, 21], as it can be a generalization of the Skorokhod–Malliavin integral [33]. Malliavin derivative is a notion of derivative in the Malliavin calculus, which extends the calculus of variations from functions to stochastic processes. This calculus allows the computation of derivatives and integration by parts of RVs, e.g., in mathematical finance to compute the sensitivities of financial derivatives [5, 28, 29, 30, 34, 38]. A

*Submitted to the journal’s Methods and Algorithms for Scientific Computing section July 7, 2014; accepted for publication (in revised form) April 7, 2015; published electronically August 4, 2015.

†Division of Applied Mathematics, Brown University, Providence, RI 02906 (mengdi.zheng@brown.edu, boris.rozovsky@brown.edu, george.karniadakis@brown.edu). The first and the third authors of this work were supported partially by ARL and University of Utah (CED3M), NSF/DMS-1216437, and by OSD-MURI grant FA9550-09-1-0613. The second author was supported by ARO grant W911NF-13-1-0012.
Taylor-like series expansion involving both Wick products and Malliavin derivatives was proposed as the WM expansion to approximate the polynomial nonlinearity in SPDEs [36]. This has been applied to moment statistics for SPDEs with polynomial nonlinearity and one RV with Gaussian or uniform distribution [47]. If an SPDE is driven by RVs or a stochastic process that can be represented by independent continuous RVs, in other distributions than Gaussian and uniform with finite moments, we can still apply the WM expansion in a similar way as long as we can generate orthogonal polynomials with respect to their measures [52]. For example, if a nonlinear SPDE is driven by a Brownian motion, by the Karhunen–Loève expansion [32], we can represent the Brownian motion by independent Gaussian RVs, and apply the WM approximation [47]. Our objective here is to solve SPDEs with polynomial nonlinearity and multiple discrete RVs by the WM approximation using gPC based on the Wiener–Askey scheme of hypergeometric polynomials [50].

The paper is organized as follows: in section 2, we define the WM expansion and derive the WM propagators for a stochastic reaction equation and a stochastic Burgers equation; in section 3, we present several numerical results for SPDEs with one RV and multiple RVs, including an adaptive procedure to control the error in time. We also compare the computational complexity between gPC and WM for the stochastic Burgers equation with the same level of accuracy. We conclude in section 4 with a short summary. In the appendix, we provide an iterative algorithm to generate coefficients in the WM approximation.

2. WM approximation. The WM propagator simplifies the gPC propagator by considering a fewer number of product terms from the polynomial nonlinearity. In this section, we present this simplification procedure and derive WM propagators for a stochastic reaction equation and a stochastic Burgers equation. The following procedure can be done for any discrete stochastic input with finite moments of all orders. To demonstrate the approximation procedure, we take the Poisson RV as an example.

2.1. WM series expansion. Given a discrete Poisson RV $\xi \sim \text{Pois}(\lambda)$ with measure $\Gamma(x) = \sum_{k \in S} \frac{e^{-\lambda}}{k!} \lambda^k \delta(x - k)$, on a finite support $S = \{0, 1, 2, \ldots, N\}$, there is an associated unique set of monic orthogonal polynomials $[11, 18]$, called Charlier polynomials, denoted as $\{c_k(x; \lambda), k = 0, 1, 2, \ldots\}$, such that

$$
\sum_{k \in S} \frac{e^{-\lambda}}{k!} \frac{\lambda^k}{k!} c_m(k; \lambda)c_n(k; \lambda) = \begin{cases} n!\lambda^n \delta_{mn} & \text{if } m = n, \\ 0 & \text{if } m \neq n. \end{cases}
$$

(2.1)

The monic Charlier polynomials associated with $\text{Pois}(\lambda)$ are defined as

$$
c_n(x; \lambda) = \sum_{k=0}^{n} \binom{n}{k} (-\lambda)^{n-k} x(x-1) \ldots (x-(k-1)), \quad n = 0, 1, 2, \ldots.
$$

(2.2)

Here $\binom{n}{k}$ is the binomial coefficient. The first few Charlier polynomials are

$$
c_0(x; \lambda) = 1,
$$

(2.3)

$$
c_1(x; \lambda) = x - \lambda,
$$

(2.4)

$$
c_2(x; \lambda) = x^2 - 2\lambda x - x + \lambda^2,
$$

(2.5)

$$
c_3(x; \lambda) = x^3 - 3\lambda x^2 - 3x^2 + 3\lambda^2 x + 3\lambda x + 2x - \lambda^3.
$$

(2.6)

For numerical computation, here we consider the support $S$ to be from 0 to $N$ instead of 0 to $\infty$, such that $\mathbb{P}(\xi = N) \leq 1e^{-32}$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Since \( \{ c_k(x; \lambda), k = 0, 1, 2, \ldots \} \) belongs to the Askey scheme of hypergeometric polynomials [2], the product of any two polynomials can be expanded as [1]

\[
(2.7) \quad c_m(x)c_n(x) = \sum_{k=0}^{m+n} a(k, m, n)c_k(x), \quad m, n = 0, 1, 2, \ldots,
\]

where \( a(k, m, n) \) can be evaluated both analytically\(^2\) and numerically \([6, 16, 18, 40, 41]\). Numerically we may generate \( a(k, m, n) \) by

\[
(2.8) \quad a(k, m, n) = \frac{\sum_{j \in S} \frac{\lambda^j}{j!} c_k(j; \lambda)c_m(j; \lambda)c_n(j; \lambda)}{\sum_{j \in S} \frac{\lambda^j}{j!} c_k(j; \lambda)c_k(j; \lambda)}, \quad k = 0, 1, 2, \ldots, m + n.
\]

Analytically \( a(k, m, n) \) is given by [26]

\[
(2.9) \quad a(k, m, n) = \frac{\sum_{l=0}^{k(m+n-k)/2} \frac{\lambda^l}{l!}\sum_{m,n,k} \frac{k!\lambda^k}{k!\lambda^k} c_m(j; \lambda)c_m(j; \lambda)}{\sum_{l=0}^{k(m+n-k)/2} \frac{\lambda^l}{l!}\sum_{m,n,k} \frac{k!\lambda^k}{k!\lambda^k} c_m(j; \lambda)c_k(j; \lambda)}, \quad k = 0, 1, \ldots, m + n.
\]

Here \( \lambda, x, j \) is the floor function.

The alternative analytical method to generate \( a(k, m, n) \) in (2.8) is given in the appendix.

For convenience, let us denote \( a(m + n - 2p, m, n) \) by \( K_{mnp} \) as follows (for \( \xi \sim \text{Pois}(\lambda) \)),

\[
(2.10) \quad K_{mnp} = \frac{\sum_{l=0}^{p} \frac{\lambda^{2l}}{l!(n-2l)!} (m-n+1)(m-n+2)(m-n+2p)}{\lambda^{2p}(m+n-2p)}, \quad p = 0, 1/2, \ldots, \frac{m+n}{2}.
\]

Then (2.7) can be rewritten as

\[
(2.11) \quad c_m(x; \lambda)c_n(x; \lambda) = \sum_{p=0}^{m+n} K_{mnp}c_{m+n-2p}(x; \lambda),
\]

where \( p \) takes half-integer values as \( p = 0, 1/2, 1, \ldots, \frac{m+n}{2} \). Equation (2.11) is completely equivalent to (2.7).

Now let us define the \textit{Wick product} \( \odot \) as \([13, 22, 31, 33, 48]\)

\[
(2.12) \quad c_m(x; \lambda) \odot c_n(x; \lambda) = c_{m+n}(x; \lambda), \quad m, n = 0, 1, 2, \ldots,
\]

and define the \textit{Malliavin derivative} \( \mathcal{D}^p \) as\(^3\) \([33, 39]\)

\[
(2.13) \quad \mathcal{D}^p c_i(x; \lambda) = \frac{\partial}{(i-p)!} c_{i-p}(x; \lambda), \quad i = 0, 1, 2, \ldots, \quad p = 0, 1/2, 1, \ldots, i.
\]

\(^2\)For monic polynomials \( \{ c_i(x; \lambda), i = 0, 1, 2, \ldots \} \), we can derive \( a(m + n, m, n) \) to \( a(0, m, n) \) iteratively by matching the coefficient of \( x^{m+n} \) to \( x^0 \) for the left- and right-hand sides of (2.7), as an alternative method to deriving \( a(k, m, n) \), than in (2.8). We notice that \( a(m + n, m, n) = 1 \).

\(^3\)In this definition \( p \) has to take half-integer values in order to balance (2.17) with (2.11). Although here in the definition of Malliavin derivative \( c_{i-p} \) may take integer values, the Malliavin derivative will always appear with the weighted Wick product; therefore, after taking the Malliavin derivative and Wick product the resulting polynomial will always be an integer.
We define $D^{p_1 \cdots p_d}$ as the product of operators from $D^{p_1}$ to $D^{p_d}$:

$$
D^{p_1 \cdots p_d}c_i(x; \lambda) \cdots c_i(x; \lambda) = \Pi_{j=1}^d \frac{ij!}{(i_j - p_j)!} c_{i_j - p_j}(x; \lambda),
$$

(2.14)

$$
i_j = 0, 1, 2, \ldots, p_j = 0, 1/2, 1, \ldots, i_j.
$$

We define the weighted Wick product $\circ_p$ in terms of the Wick product as

$$
c_m \circ_p c_n = \frac{pm!n!}{(m + p)!(n + p)!} K_{m+p,n+p} c_m \circ c_n,
$$

(2.15)

and define $\circ_{p_1, \ldots, p_d}$ as

$$
(c_{m_1} \cdots c_{m_d}) \circ_{p_1, \ldots, p_d} (c_{n_1} \cdots c_{n_d}) = \Pi_{j=1}^d \frac{p_j!m_j!n_j!}{(m_j + p_j)!(n_j + p_j)!} K_{m_j + p_j, n_j + p_j, p_j} c_{m_j} \circ c_{n_j}.
$$

Therefore, (2.11) can be rewritten as

$$
c_m(x; \lambda) c_n(x; \lambda) = \sum_{p=0}^{m+n} \frac{D^p c_m \circ_p D^p c_n}{p!}.
$$

(2.17)

We note that the definition of the weighted Wick product (2.15) depends on the measure $\Gamma$. Assume that we are given two random fields $u$ and $v$ on the same probability space $(S, \mathcal{B}(S), \Gamma)$, with their expansions $u = \sum_{i=0}^{\infty} u_i c_i$ and $v = \sum_{i=0}^{\infty} v_i c_i$. Then, we can expand $uv$ by

$$
uv = \sum_{p=0}^{\infty} \frac{D^p u \circ_p D^p v}{p!}
$$

(index $p$ takes half-integer values), if we define

$$
D^p u = \sum_{i=0}^{\infty} u_i D^p c_i.
$$

(2.19)

Now let us introduce a nonnegative half integer $Q \in \{0, 1/2, 1, \ldots\}$ as the WM order, hence (2.18) can be approximated by the following WM expansion

$$
uv \approx \sum_{p=0}^{Q} \frac{D^p u \circ_p D^p v}{p!}, Q = 0, 1/2, 1, \ldots,
$$

(2.20)

and $p$ here also takes half-integer values.

Now let us assume $\eta$ to be an RV with discrete measure of finite moments of all orders on a complete probability space $(S, \mathcal{B}(S), \Gamma)$. There is an associated unique set of monic orthogonal polynomials with respect to this measure [18], denoted as $\{P_i(\eta), i = 0, 1, 2, \ldots\}$ for $\eta \in S$, such that

$$
\int_S P_m(\eta)P_n(\eta)d\Gamma(\eta) \begin{cases} > 0 & \text{if } m = n, \\ = 0 & \text{if } m \neq n. \end{cases}
$$

(2.21)

Following the same procedure from (2.7) to (2.17), we can expand the product of

\[ A1875 \]

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
2.2. WM propagators. In this section, we will study a stochastic reaction equation and a stochastic Burgers equation, and derive their WM propagators.

2.2.1. Reaction equation. Let us consider the following reaction equation with a random coefficient:

\[
\frac{dy}{dt} = -\sigma k(t, \xi_1, \xi_2, \ldots, \xi_d)y(t; \omega), \quad y(0; \omega) = y_0, \tag{2.23}
\]

where \( \xi_1, \ldots, \xi_d \sim \text{Pois}(\lambda) \) are independent identically distributed (i.i.d.), and\(^5 \) \( k(t, \xi_1, \ldots, \xi_d) = \sum_{i_1, \ldots, i_d=0}^{\infty} a_{i_1, \ldots, i_d}(t)c_{i_1}(\xi_1; \lambda) \cdots c_{i_d}(\xi_d; \lambda) \); \( \sigma \) controls the variance of the reaction coefficient. Also the \( \{c_k(\xi; \lambda), k = 0, 1, 2, \ldots\} \) are monic Charlier polynomials associated with the Poisson distribution and with mean \( \lambda \) [15, 23, 42, 45].

Remark. Here we present the WM approximation method for the Poisson distribution; however, the method is not restricted to Poisson distribution, since we can generate orthogonal polynomials with respect to other discrete measures [6, 16, 18, 40], at least for the Wiener–Askey family of polynomials [1, 2].

By (2.20), the WM approximation to (2.23) is

\[
\frac{dy}{dt} \approx -\sigma \sum_{p_1, \ldots, p_d=0}^{Q_1, \ldots, Q_d} \frac{\mathcal{D}^{p_1}c_{i_1}(\xi_1; \lambda) \cdots \mathcal{D}^{p_d}c_{i_d}(\xi_d; \lambda)}{p_1! \cdots p_d!} \mathcal{D}^{p_1 \cdots p_d}y, \quad y(0; \omega) = y_0. \tag{2.24}
\]

Here \( Q_1, \ldots, Q_d \) are WM orders for RVs \( \xi_1, \ldots, \xi_d \), respectively. We expand the solution to (2.23) in a finite dimensional series as

\[
y(t; \omega) = \sum_{j_1, j_2, \ldots, j_d=0}^{P_1, \ldots, P_d} \hat{y}_{j_1, \ldots, j_d}(t)c_{j_1}(\xi_1) \cdots c_{j_d}(\xi_d), \tag{2.25}
\]

where \( P_1, \ldots, P_d \) are polynomial chaos expansion order for RVs \( \xi_1, \ldots, \xi_d \), respectively. By substituting (2.25) into (2.24) and the Galerkin projection onto \( c_{i_1}(\xi_1) \cdots c_{i_d}(\xi_d) \),

\[
\langle f(\xi_1, \ldots, \xi_d)c_{i_1}(\xi_1) \cdots c_{i_d}(\xi_d) \rangle = \int_{S_i} d\Gamma_1(\xi_1) \cdots \int_{S_d} d\Gamma_1(\xi_d)f c_{i_1}(\xi_1) \cdots c_{i_d}(\xi_d), \tag{2.26}
\]

\((S_1, \ldots, S_d)\) are the support and the measure of \( \xi_1 \), we obtain the WM propagator for problem (2.23) as

\[
\frac{d\hat{y}_{i_1, \ldots, i_d}(t)}{dt} = -\sigma \sum_{l_1, \ldots, l_d=0}^{P_1, \ldots, P_d} \sum_{m_1, \ldots, m_d=0}^{Q_1, \ldots, Q_d} K_{l_1, m_1+i_1-l_1, m_1} \cdots K_{l_d, m_d+i_d-l_d, m_d}, \tag{2.27}
\]

\[
\hat{y}_{i_1, \ldots, i_d}(0) = y_0 \delta_{i_1, 0} \delta_{i_2, 0} \cdots \delta_{i_d, 0},
\]

for \( i_1 = 0, 1, \ldots, P_1, \ldots, i_d = 0, 1, \ldots, P_d \).

\(^5\)Such \( k(t, \xi_1, \ldots, \xi_d) \) is meaningful to be considered because many stochastic processes have series representations, e.g., Karhunen–Loeve expansion for the Gaussian process [24, 32], and shot noise expansion for Levy pure jump processes [7, 43, 44].
2.2.2. The Burgers equation. Let us now consider the following Burgers equation with a random forcing term:

\begin{equation}
(2.28) \quad u_t + uu_x = \nu u_{xx} + \sigma \sum_{j=1}^{d} c_1(\xi_j) \psi_j(x,t), \quad x \in [-\pi, \pi],
\end{equation}

with initial condition

\begin{equation}
(2.29) \quad u(x,0) = 1 - \sin(x)
\end{equation}

and periodic boundary conditions. Here \( \xi_1, \ldots, d \sim \text{Pois}(\lambda) \) are i.i.d. RVs, and \( \sigma \) is a constant that controls the magnitude of the force. The WM approximation of (2.28) is

\begin{equation}
(2.30) \quad u_t + \sum_{p_1, \ldots, p_d=0}^{Q_1, \ldots, Q_d} \frac{1}{p_1! \ldots p_d!} D^{p_1 \ldots p_d} u_{x} \approx \nu u_{xx} + \sigma \sum_{j=1}^{d} c_1(\xi_j) \psi_j(x,t).
\end{equation}

If we expand the solution in a finite dimensional series as

\begin{equation}
(2.31) \quad u(x, t; \xi_1, \ldots, \xi_d) = \sum_{k_1, \ldots, k_d=0}^{p_1, \ldots, p_d} \tilde{u}_{k_1, \ldots, k_d}(x, t)c_{k_1}(\xi_1; \lambda) \cdots c_{k_d}(\xi_d; \lambda),
\end{equation}

then by substituting (2.31) into (2.30) and performing a Galerkin projection onto \( c_{k_1}(\xi_1) \cdots c_{k_d}(\xi_d) \), we derive the WM propagator for problem (2.28) as

\begin{equation}
(2.32) \quad \frac{\partial}{\partial t} \tilde{u}_{k_1, \ldots, k_d}(x, t) + \sum_{p_1, \ldots, p_d=0}^{Q_1, \ldots, Q_d} \sum_{m_1, \ldots, m_d=0}^{P_1, \ldots, P_d} \left( K_{m_1, k_1+2p_1-m_1, p_1} \cdots K_{m_d, k_d+2p_d-m_d, p_d} \tilde{u}_{m_1, \ldots, m_d} \frac{\partial}{\partial x} \tilde{u}_{k_1+2p_1-m_1, \ldots, k_d+2p_d-m_d} \right)
\end{equation}

\begin{equation}
= \nu \frac{\partial^2}{\partial x^2} \tilde{u}_{k_1, \ldots, k_d} + \sigma(\delta_{1, k_1} \delta_{0, k_d} \cdots \delta_{0, k_d} \lambda_{1} + \cdots + \delta_{0, k_1} \delta_{0, k_d} \lambda_{1, k_d})
\end{equation}

for \( k_1, \ldots, k_d = 0, 1, \ldots, P \), with the restriction \( 0 \leq k_1 + 2p_1 - m_1 \leq P \) for \( i = 1, \ldots, d \). The initial conditions (I.C.) and boundary conditions (B.C.) are given by

\begin{equation}
(2.33) \quad \begin{cases}
\tilde{u}_{0,0,\ldots,0}(x,0) = u(x,0) = 1 - \sin(x) \quad \text{(I.C.)}, \\
\tilde{u}_{k_1,\ldots,k_d}(x,0) = 0, \quad \text{if } (k_1, \ldots, k_d) \neq (0, \ldots, 0) \quad \text{(I.C.)}, \\
\tilde{u}_{k_1,\ldots,k_d}(\pi, t) = \tilde{u}_{k_1,\ldots,k_d}(\pi, t) \quad \text{(periodic B.C. on } [-\pi, \pi]).}
\end{cases}
\end{equation}

3. Results. In this section, we will provide numerical results for solving the reaction and the Burgers equations with different discrete random inputs by the WM method. We will compare the computational complexity of WM and gPC for the Burgers equation with multiple RVs.

3.1. Reaction equation with one RV. In Figure 1, we show results from computing the WM propagator given in (2.27) for the reaction equation with one Poisson RV \( d = 1 \) in (2.23)). We plot the errors of second moments at final time \( T \)s with respect to different WM expansion order \( Q \).\(^6\) The polynomial expansion order

\(^6\)In Figure 1 we show errors for \( Q \) taking integer values because the error line for \( Q = k \) is almost the same as \( k + \frac{1}{2} \). We observe similar behavior in Figure 4.
excluding the time to compute coefficients in equation (2.27), with polynomial order \( P = 10, \gamma_0 = 1, \lambda = 0.5. \) We used RK4 scheme with time step \( dt = 1e - 4 \); \( k(\xi) = \frac{c_0(\xi, \lambda)}{2!} + \frac{c_1(\xi, \lambda)}{3!} + \frac{c_2(\xi, \lambda)}{4!} \), \( \sigma = 0.1 \) (left); \( k(\xi) = \frac{c_0(\xi, \lambda)}{1!} + \frac{c_1(\xi, \lambda)}{2!} \), \( \sigma = 1 \) (right).

\( P \) in (2.25) was chosen sufficiently large in order to mainly examine the convergence with respect to \( Q \). We used the fourth-order Runge–Kutta method (RK4) to solve (2.27) with sufficiently small time steps. The error of the second moment at final time \( T \) is defined as:

\[
\text{l2err}(T) = \left| \frac{E[y_{ex}^2(T; \omega)] - E[y_{num}^2(T; \omega)]}{E[y_{num}^2(T; \omega)]} \right|.
\]

From Figure 1 with a fixed polynomial order \( P \), we take \( k(\xi) = a_0(t)c_0(\xi; \lambda) + a_1(t)c_1(\xi; \lambda) + a_2(t)c_2(\xi; \lambda); \) therefore, the WM order \( Q = 2 \) is the highest order that equates (2.23) with (2.24) in (2.20) when \( Q \geq 2, \) the WM propagator is exactly the same as the gPC propagator. We observe that in Figure 1, when \( Q \) increases by one, the error is improved by at least one order of magnitude when \( \sigma = 0.1 \), and four orders of magnitude when \( \sigma = 1 \). Therefore, with less computational cost than gPC, the WM method can achieve the same accuracy as gPC. In gPC, the polynomial order \( P \) serves as a resolution parameter for the stochastic system. In the WM method, for each \( P \) we may further refine the system by another resolution parameter \( Q \). We observe that the right plot in Figure 1 has a dip for error lines corresponding to \( Q = 1 \) and 2. When \( \sigma \) is larger, the solution of (2.23) decays faster, and hence this trend in the error; however, with polynomial order \( P \) we ignore the terms in the sum (2.25) with polynomial order larger than \( P \), which increases the error with respect to time. Because of this balance of decreasing and increasing errors, we observe that errors go down at first and then up in the right plot in Figure 1. On the left plot of Figure 1 we do not observe that the error goes down and up because \( \sigma \) is small and the solution decays slower so the error mainly increases with time. We can evaluate the coefficients \( K_{mnp} \) in (2.10) offline, and we compute only the WM propagator in (2.24) online. We consider the number of terms to evaluate \( k(t, \xi)y(t; \omega) \) in (2.23) in the WM propagator (2.24) as the primary contribution to the computational complexity. We consider the online CPU time in Table 1 as the CPU time to evaluate the right-hand side of (2.24) excluding the time to compute coefficients \( K_{mnp} \) in (2.10). In Table 1 we compare the complexity and corresponding computational time between gPC of different orders \( P \) and WM with a fixed order of \( P = 3, Q = 2 \) for the reaction equation (2.23) with one.
For gPC with different orders $P$ and WM with a fixed order of $P = 3, Q = 2$ in reaction equation (2.23) with one Poisson RV ($\lambda = 0.5, y_0 = 1$, $k(\xi) = \frac{\partial_k(x,\lambda)}{2^{1/2}} + \frac{\partial_k(x,\lambda)}{3!} + \frac{\partial_k(x,\lambda)}{4!}$, $\sigma = 0.1$, RK4 scheme with (time step $dt = 1e-4$), we compare (1) computational complexity ratio to evaluate $k(t, \xi)(t, \omega)$ between gPC and WM (upper); (2) CPU time ratio to compute $k(t, \xi)(t, \omega)$ between gPC and WM (lower). We simulated in MATLAB on Intel (R) Core (TM) i5-3470 CPU @ 3.20 GHz.

<table>
<thead>
<tr>
<th>gPC order</th>
<th>$P = 4$</th>
<th>$P = 6$</th>
<th>$P = 8$</th>
<th>$P = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of complexity (gPC/WM)</td>
<td>1.4054</td>
<td>2.2162</td>
<td>3.027</td>
<td>3.8378</td>
</tr>
<tr>
<td>Ratio of CPU time (gPC/WM)</td>
<td>1.2679</td>
<td>1.8036</td>
<td>2.3893</td>
<td>2.875</td>
</tr>
</tbody>
</table>

RV (with the same parameters as on the left of Figure 1). Notice that the $l^2_{err}$ from WM with $P = 3, Q = 2$ is $1.5e-8$ and the $l^2_{err}$ from gPC with $P = 10$ is $1.4e-8$ (almost the same), however the online CPU time for gPC is $2.875$ times greater than that of WM.

#### 3.2. The Burgers equation with one RV

Now let us compute the WM propagator for a Burgers equation with one Poisson RV in (2.32). We solved the WM propagator by a second-order implicit-explicit (IMEX) time splitting scheme. For spatial discretization we used the Fourier collocation method. The reference solution was established by running the Burgers equation with $\xi$ taking all the possible values.

In this problem we define the $L_2$ norm of error for second moments as follows, for a certain final time $T$:

$$l^2_{u2}(T) = \frac{|E[u^2_{num}(x, T; \xi)] - E[u^2_{ex}(x, T; \xi)]|_{L^2([-\pi, \pi])}}{|E[u^2_{ex}(x, T; \xi)]|_{L^2([-\pi, \pi])}}.$$

In Figure 2, we observe monotonic convergence with respect to $Q$, that is, by increasing the WM order $Q$ by one, the error decreases effectively by five to six orders of magnitude at $T = 1$. If we use gPC in this problem, we will calculate $P + 1$ terms in $\sum_{i=0}^{P} u_i(x) \frac{\partial u_i}{\partial x}$ for $(P + 1)$ equations in the gPC propagator (343 terms in this problem). However by the WM method, in order to have good accuracy, say $1e-12$, as shown in Figure 2, we consider many fewer terms resulting from the nonlinear term $u_i \frac{\partial u_i}{\partial x}$ in the Burgers equation by only taking $Q = 3$ (231 terms in this problem).

In Figure 3, we plot the error defined in (3.2) with respect to polynomial expansion order $P$, for different WM order $Q$. We also compare it with the error by the PCM with $(P + 1)$ points. We observe that for a fixed polynomial order $P$ in gPC, the smallest $Q$ to match the error from the WM propagator to the same order with PCM is when $Q = P - 1$. For example, in Figure 3 when $P = 2$, the first error line by WM that touches the black solid line by PCM is the one that corresponds to $Q = 1$. Although this observation is only empirical, it allows us to compare the computational complexity between gPC and WM with the same level of accuracy, i.e., we are going to compare the computational cost later between gPC of polynomial order $P$ and WM of polynomial order $P$ and of WM order $Q = P - 1$. We also observe from Figure 3 the smallest value of $Q$ we need to model the stochastic Burgers equation with one

---

7 We used the second-order RK2 scheme for nonlinear terms and the forcing term, and the Crank–Nicolson scheme for the diffusion term.

8 Although the Poisson RV has an infinite number of points in the support, we only consider the points with probability more than $1e-16$.

9 gPC with polynomial order $P$ has the same magnitude of error with PCM implemented with $(P + 1)$ quadrature points, therefore, by plotting PCM with $(P + 1)$ quadrature points against WM with polynomial order $P$, we are comparing the gPC with WM at the same polynomial order $P$. 

---

**Table 1**

For gPC with different orders $P$ and WM with a fixed order of $P = 3, Q = 2$ in reaction equation (2.23) with one Poisson RV ($\lambda = 0.5, y_0 = 1$, $k(\xi) = \frac{\partial_k(x,\lambda)}{2^{1/2}} + \frac{\partial_k(x,\lambda)}{3!} + \frac{\partial_k(x,\lambda)}{4!}$, $\sigma = 0.1$, RK4 scheme with (time step $dt = 1e-4$), we compare (1) computational complexity ratio to evaluate $k(t, \xi)(t, \omega)$ between gPC and WM (upper); (2) CPU time ratio to compute $k(t, \xi)(t, \omega)$ between gPC and WM (lower). We simulated in MATLAB on Intel (R) Core (TM) i5-3470 CPU @ 3.20 GHz.

<table>
<thead>
<tr>
<th>gPC order</th>
<th>$P = 4$</th>
<th>$P = 6$</th>
<th>$P = 8$</th>
<th>$P = 10$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ratio of complexity (gPC/WM)</td>
<td>1.4054</td>
<td>2.2162</td>
<td>3.027</td>
<td>3.8378</td>
</tr>
<tr>
<td>Ratio of CPU time (gPC/WM)</td>
<td>1.2679</td>
<td>1.8036</td>
<td>2.3893</td>
<td>2.875</td>
</tr>
</tbody>
</table>
Fig. 2. The Burgers equation with one Poisson RV \( \xi \sim \text{Pois}(\lambda) \) (\( d = 1, \psi_1(x,t) = 1 \)): \( l_2u_2(T) \) error defined in (3.2) versus time, with respect to different WM order \( Q \). Here we take in (2.32) the polynomial expansion order \( P = 6, \lambda = 1, \nu = 1/2, \sigma = 0.1 \), IMEX (Crank–Nicolson/RK2) scheme with time step \( dt = 2e^{-4} \), and 100 Fourier collocation points on \([-\pi, \pi]\).

Fig. 3. \( P \)-convergence for the Burgers equation with one Poisson RV \( \xi \sim \text{Pois}(\lambda) \) (\( d = 1, \psi_1(x,t) = 1 \)): errors defined in (3.2) versus polynomial expansion order \( P \), for different WM order \( Q \), and by PCM with \( P + 1 \) points with the parameters \( \nu = 1, \lambda = 1 \), final time \( T = 0.5 \), IMEX (Crank–Nicolson/RK2) scheme with time step \( dt = 5e^{-4} \), 100 Fourier collocation points on \([-\pi, \pi] \), \( \sigma = 0.5 \) (left), and \( \sigma = 1 \) (right).

discrete RV for a specific polynomial order \( P \), to achieve the same accuracy with gPC of polynomial order \( P \). When \( Q \geq P - 1 \), we see from Figure 3 that even if we increase \( P \) the convergence rate versus \( P \) will be slower than \( P \)-convergence from gPC.

In Figure 4, we investigate the \( Q \)-convergence of the WM approximation by plotting the error defined in (3.2) with respect to WM order \( Q \), for different polynomial expansion orders \( P \). The first observation is that when \( Q \) increases from integer \( k \) to the next larger half-integer \( k + \frac{1}{2} \), the error is not prominently improved, but the error is obviously improved when \( Q \) increases from integer \( k \) to integer \( k + 1 \). This is very similar to a phenomenon in spectral methods that the magnitude of error oscillates between even orders and odd orders. The second observation is that the choice of \( Q = P - 1 \) is optimum for the WM approximation, because in Figure 4 the error remains at the same magnitude when \( Q \) is taking values larger than \( P - 1 \). For example, we note the error line for the left figure in Figure 4 with respect to \( P = 2 \): the error
3.3. Reaction equation with multiple RVs. Now let us compute (2.27) with five i.i.d. Poisson RVs with mean $\lambda$ ($d = 5$). We solve problem (2.23) assuming a new model, where $k(\xi_1, \xi_2, \ldots, \xi_5, t) = \sum_{i=1}^5 \cos(it)c_1(\xi_i)$. The WM propagator in this problem was solved by the RK2 scheme. For a fixed polynomial expansion order $P$ in Figure 5, we plot the error defined in (3.1) with respect to time and for different WM order $Q$.

We observe in Figure 5 that by adding only one more Malliavin derivative order $Q$, the error is improved by two orders of magnitude at $T = 0.5$. When $Q = 1$, the WM propagator has a much simpler form than the gPC propagator. Figure 5 also demonstrates the ability of computing SPDEs with multiple RVs by the WM method. Notice that Levy processes have different types of series expansions by independent RVs, therefore, Figure 5 represents the first step towards dealing with nonlinear SPDEs with Levy processes (including Gaussian processes and pure jump processes that admit series representations).

Next let us compute (2.27) with one Poisson RV ($\xi_1$) with mean $\lambda$ and one binomial RV ($\xi_2$) with the number of trials $N$ and success probability $p$. We solve problem (2.23) assuming $k(\xi_1, \xi_2) = c_1(\xi_1)k_1(\xi_2)$, where $k_1(\xi_2)$ is the orthogonal polynomial to the binomial distribution for $\xi_2$. We derive the coefficients $K_{mnp}$ in (2.11) both for the Poisson distribution and the binomial distribution. The WM propagator in this case is still given by (2.27) with $d = 2$, except replacing the corresponding $K_{mnp}$ for $\xi_2$ by those generated from the binomial distribution. For a fixed polynomial order $P$ in Figure 6, we plot the error defined in (3.1) with respect to time and for different WM order $Q$. 

![Figure 4](image-url)
We observe in Figure 6 that by adding one more Malliavin derivative order $Q$, the error is improved by ten orders of magnitude at $T = 1$. Figure 6 also demonstrates the ability to compute SPDEs with multiple RVs with different distributions (hybrid-type).

### 3.4. The Burgers equation with multiple RVs.

Now let us compute (2.28) with three Poisson RVs with mean $\lambda$ ($d = 3$). We solve problem (2.28) with the random forcing term $\sigma \sum_{j=1}^{d} c_1(\xi_j) \psi_j(x,t) = \sigma \sum_{j=1}^{3} c_1(\xi_j) \cos(0.1jt)$. We solved the WM propagator (2.32) by the second-order IMEX time splitting scheme (RK2/Crank–Nicolson). For a fixed polynomial expansion order $P$ in Figure 5, we plotted the error defined in (3.2) with respect to time, for different WM order $Q$. Here we take $P_1 = P_2 = P_3 = P = 2$. 

![Figure 5](image-url)  
**Fig. 5.** Reaction equation with five Poisson RVs $\xi_1, \ldots, 5 \sim \text{Pois}(\lambda)$ ($d = 5$): error defined in (3.1) with respect to time, for different WM order $Q$, with parameters $\lambda = 1$, $\sigma = 0.5$, $y_0 = 1$, polynomial order $P = 4$, RK2 scheme with time step $dt = 1e - 3$, and $k(\xi_1, \xi_2, \ldots, \xi_5, t) = \sum_{i=1}^{5} \cos(it)c_1(\xi_i)$ in (2.23).

![Figure 6](image-url)  
**Fig. 6.** Reaction equation with one Poisson RV $\xi_1 \sim \text{Pois}(\lambda)$ and one Binomial RV $\xi_2 \sim \text{Bino}(N, p)$ ($d = 2$): error defined in (3.1) with respect to time, for different WM order $Q$, with parameters $\lambda = 1$, $\sigma = 0.1$, $N = 10$, $p = 1/2$, $y_0 = 1$, polynomial order $P = 10$, RK4 scheme with time step $dt = 1e - 4$, and $k(\xi_1, \xi_2, t) = c_1(\xi_1)k_1(\xi_2)$ in (2.23).
Fig. 7. The Burgers equation with three Poisson RVs $\xi_{1,2,3} \sim \text{Pois}(\lambda)$ ($d = 3$): error defined in (3.2) with respect to time, for different WM order $Q$, with parameters $\lambda = 0.1$, $\sigma = 0.1$, $y_0 = 1$, $\nu = 1/100$, polynomial order $P = 2$, IMEX (RK2/Crank–Nicolson) scheme with time step $dt = 2.5e^{-4}$.

We observe in Figure 7 that the error is not prominently decreased when we increase WM order $Q$ for one or two RVs, but the error is greatly decreased when we increase $Q$ for all three RVs. In this numerical experiment we have also computed the case that $Q_1 = Q_2 = Q_3 = \frac{1}{2}$, and similarly to Figure 4, the error line corresponding to that almost overlapped with the error line for $Q_1 = Q_2 = Q_3 = 0$ in Figure 7. This suggests that when we model the stochastic Burgers equations with multiple discrete RVs, the accuracy in some cases will not be greatly improved by increasing the WM order $Q$ by $\frac{1}{2}$. Therefore, in order to save computational cost in the WM method for the Burgers equations with multiple discrete RVs, we may use integer values for $Q$ for each RV instead of half-integer values.

3.5. Adaptive WM method. Now let us control the error growth with respect to time under a certain prespecified accuracy. We will show that it is possible to control the error below a certain threshold by increasing the gPC order $P$ and the WM order $Q$ ($P$-$Q$ refinement). Under a prespecified adaptive criterion value, we increase the polynomial order $P$ or the WM order $Q$, when the absolute value of error is greater than the adaptive criterion value ($P$-adaptivity and $Q$-adaptivity).

In Figure 8, we address the long term integration issue of gPC by computing the WM propagator in reaction equation (2.27) with two Poisson RVs with mean $\lambda$, for a fixed $Q$. We plot the error defined in (3.1) with respect to time and we adaptively increase $P$ to keep the error under the indicated adaptive criterion. We observe that increasing the polynomial order $P$ is an effective way to control the error as time progresses for SPDEs with multiple RVs. Besides dealing with the long term integration problem, varying $P$ also allows us to use a smaller polynomial order $P$ at early times, hence expending less computational cost. In gPC, we may also keep the error lower than a value by increasing $P$; however, increasing $P$ in the gPC propagator costs much more than increasing $P$ in the WM propagator with a small $Q$.

In Figure 9 we compute the WM propagator in the Burgers equation (2.32) with one Poisson RV, with mean $\lambda = 1$. We plot the error defined in (3.2) with respect to time both in the case that we fix $Q$ or $P$ to control the error to be under the indicated adaptive criterion by increasing $P$ or $Q$. We observe that increasing the WM expansion order $Q$ is also an effective way to control the error as time progresses.
3.6. Computational complexity. We demonstrate next that the WM propagator is more cost effective in evaluating the statistics of solution than the gPC propagator. Because the computational complexity depends on the form of the equation itself, we analyze this case by case. First let us consider the Burgers equation as an example to compare the complexity of WM and gPC.

3.6.1. The Burgers equation with one RV. To compare WM and gPC methods for the Burgers equation with one RV ($\xi \sim \text{Pois}(\lambda)$), we simply write the gPC and WM propagators separately and compare how they differ from each other. We consider this equation

\begin{equation}
(3.3) \quad u_t + uu_x = \nu u_{xx} + \sigma c_1(\xi; \lambda), \quad x \in [-\pi, \pi].
\end{equation}
The gPC propagator for this problem is

\[
\frac{\partial \tilde{u}_k}{\partial t} + \sum_{m,n=0}^{P} \tilde{u}_m \frac{\partial \tilde{u}_n}{\partial x} (c_m c_n c_k) = \nu \frac{\partial^2 \tilde{u}_k}{\partial x^2} + \sigma \delta_{1k}, k = 0, 1, \ldots, P, \tag{3.4}
\]

where \( (c_m c_n c_k) = \int_{S} d\Gamma(\xi)c_k(\xi)c_m(\xi)c_n(\xi). \)

The WM propagator for this problem is

\[
\frac{\partial \hat{u}_k}{\partial t} + \sum_{p=0}^{Q} \hat{u}_i \frac{\partial \hat{u}_{k+2p-1}}{\partial x} K_{i,k+2p-1,Q} = \nu \frac{\partial^2 \hat{u}_k}{\partial x^2} + \sigma \delta_{1k}, k = 0, 1, \ldots, P. \tag{3.5}
\]

The only difference between gPC and WM propagators is between the term \( \sum_{m,n=0}^{P} \tilde{u}_m \frac{\partial \tilde{u}_n}{\partial x} (c_m c_n c_k) \) in gPC and the term \( \sum_{p=0}^{Q} \hat{u}_i \frac{\partial \hat{u}_{k+2p-1}}{\partial x} K_{i,k+2p-1,Q} \) in WM. Assuming that we are going to solve (3.4) and (3.5) with the same time stepping scheme and the same spatial discretization, for each time step, let us also assume that the computational complexity of computing one term like \( \tilde{u}_i \frac{\partial \tilde{u}_j}{\partial x} \) is \( \alpha \), while the complexity for the rest of the linear terms is 1.

Under this assumption, in (3.4) for gPC, we have \( (P+1) \) equations in the system, each one with complexity \( 1 + (P+1)^2 \alpha \), and therefore the total complexity is \( (P+1)[1 + (P+1)^2\alpha] \).

In (3.5) for WM, we still have \( (P+1) \) equations in the system. By denoting the number of terms like \( \tilde{u}_i \frac{\partial \tilde{u}_j}{\partial x} \) in the whole WM propagator as \( C(P,Q) \), the total complexity will be \( (P+1) + C(P,Q)\alpha \), and we compute \( C(P,Q) \) numerically. We demonstrate how to count \( C(P,Q) \) when \( P = 4, Q = 1/2 \) in Figure 10: there are five \( 4 \times 4 \) grids, for \( k = 0, 1, 2, 3, 4 \), respectively, for \( \tilde{u}_i \frac{\partial \tilde{u}_j}{\partial x} \) in all the five equations in the WM propagator. The horizontal axis represents the index \( i \) for \( \tilde{u}_i \) and the vertical axis represents the index \( j \) for \( \tilde{u}_j \). We marked the terms like \( \tilde{u}_i \frac{\partial \tilde{u}_j}{\partial x} \) in the \( k \)th equation in the WM propagator by drawing a circle at the \((i,j)\) dot on the \( k \)th grid. In this way we may visualize the nonlinear terms in the propagator and hence visualize the main computational complexity. In the WM method for \( P = 4, Q = 1/2 \), only the circled dots are considered in the propagator, however, in the gPC method for \( P = 4 \), all the dots on the five grids are considered in the propagator. Hence, we can see how many fewer terms like \( \tilde{u}_i \frac{\partial \tilde{u}_j}{\partial x} \) we need to consider in WM comparing to gPC.

When \( P \) is sufficiently large, the ratio for complexity of WM to gPC is approximately \( \frac{C(P,Q)}{(P+1)^3} \), ignoring lower order terms on \( P \). Since we observed in Figures 3 and 4 that when \( Q = P - 1 \), the errors computed from WM propagators are at the same accuracy level as from gPC propagators, we calculate the ratio of complexity between WM and gPC for the Burgers equation with one RV \( \frac{C(P,Q = P-1)}{(P+1)^3} \) when \( Q = P - 1 \) (so that WM and gPC have the same level of accuracy) and \( P \geq 2 \) as

\[
C(P,Q = P - 1) = 1 - \frac{10 + \frac{1}{2}P(P+1)(P+2)}{(P+1)^3}. \tag{3.6}
\]

### 3.6.2. The Burgers equation with \( d \) RVs.

Now suppose we are going to solve the Burgers equation with \( d \) RVs (each RV \( \xi_i \) has polynomial expansion order \( P_i \) and WM order \( Q_i \)):

\[
u_t + uu_x = \nu u_{xx} - \sigma c_{m_1}(\xi_1) \cdots c_{m_d}(\xi_d), \quad x \in [-\pi, \pi]. \tag{3.7}
\]
Fig. 10. Terms in \( \sum_{p=0}^{Q} \sum_{i=0}^{P} \hat{u}_i \frac{\partial u_{k+2p-i}}{\partial x} K_{i,k+2p-i,p} \) for each PDE in the WM propagator for the Burgers equation with one RV in (3.5) are denoted by dots on the grids: here \( P = 4, Q = \frac{1}{2}, k = 0, 1, 2, 3, 4 \). Each grid represents a PDE in the WM propagator, labeled by \( k \). Each dot represents a term in the sum \( \sum_{p=0}^{Q} \sum_{i=0}^{P} \hat{u}_i \frac{\partial u_{k+2p-i}}{\partial x} K_{i,k+2p-i,p} \). The small index next to the dot is for \( p \), \( x \) direction is the index \( i \) for \( \hat{u}_i \), and \( y \) direction is the index \( k + 2p - i \) in \( \frac{\partial u_{k+2p-i}}{\partial x} \). The dots on the same diagonal line have the same index \( p \).

By gPC, we will have \( \Pi_{i=1}^{d} (P_i + 1) \) equations in the propagator, and if \( P_i \) are all equal to \( P \), there will be \( (P + 1)^d \) equations in the gPC propagator. We will have \( (\Pi_{i=1}^{d} (P_i + 1))(\Pi_{i=1}^{d} (P_i + 1)^2) \) terms like \( \hat{u}_k \frac{\partial u_i}{\partial x} \). When all the RVs are having the same \( P \), this number is \( (P + 1)^{3d} \).

By WM, we will still have the same number of equations in the propagator system, but the number of terms like \( \hat{u}_k \frac{\partial u_i}{\partial x} \) is \( \Pi_{i=1}^{d} C(P_i, Q_i) \). Let us assume all the RVs having the same \( P \) and \( Q \). This formula can be written as \( (C(P, Q))^d \).

When \( P \) is sufficiently large (for simplicity we assume \( P_i = P, Q_i = Q \) for all \( i = 1, 2, \ldots, d \)), the ratio for complexity of WM to gPC is approximately \( \frac{C(P, Q)^d}{(P + 1)^{3d}} \), ignoring lower order terms on \( P \). We computed the ratio of complexity in Figure 11 for \( d = 2, 3 \).

Besides Figure 11, we also want to point out the following observation. From Figure 3 we observed numerically that when \( Q \geq P - 1 \), the error from the WM method with polynomial order \( P \) is of the same order as the error from gPC with
polynomial order $P$. So let us consider the computational cost ratio $\frac{C(P,Q)}{(P+1)!}$ between the two methods for WM with order $Q = P - 1$, and gPC with order $P$, in Table 2.

We conclude from Figure 11 and Table 2 that (1) the larger the $P$, the bigger the cost ratio between WM to gPC ($\frac{C(P,Q)}{(P+1)!}$); (2) the higher the dimensions, for the same order $P$ and $Q$, the lower the ratio $\frac{C(P,Q)}{(P+1)!}$. In other words, the higher the dimensions, the less WM is going to cost than gPC for the same accuracy.

4. Conclusions. We presented a new WM expansion to approximate polynomial nonlinear terms in SPDEs with random inputs of arbitrary discrete measure with finite moments, on which orthogonal polynomials can be constructed numerically [41, 16, 18, 6, 40]. Specifically, we derived WM propagators for a stochastic reaction and a Burgers equation in (2.27) and (2.32) with multiple discrete RVs. The error was effectively improved by at least two to eight orders of magnitude when the WM order $Q$ was increased into a larger integer in Figures 1 and 2. Linear and nonlinear SPDEs with multiple RVs were considered in Figures 5 and 7 as the first step towards application of the WM method to nonlinear SPDEs with stochastic processes, such as Levy processes with jumps. We found the smallest WM order $Q$ for gPC polynomial order $P$ in the WM method to be $Q = P - 1$ in order to achieve the same order of magnitude of error in gPC with polynomial order $P$ or PCM with $(P+1)$ collocation points, by computing the Burgers equation with one Poisson RV in Figure 3. When $Q$ was larger than $(P - 1)$, the error remained almost constant as in Figure 4. We proposed an adaptive WM method in section 3.5, by increasing the gPC order $P$
and the WM order $Q$ as a possible solution to control the error growth in long term integration in gPC, shown in Figures 8 and 9. With $Q = P - 1$ we estimated and compared the computational complexity between the WM method and gPC for a stochastic Burgers equation with $d$ RVs in section 3.5. The WM method required much less computational complexity than gPC, especially in higher dimensions, as in Table 2. However WM is still more expensive than PCM or sparse PCM.

**Appendix. Algorithm to generate coefficients $a(k, m, n)$ numerically.**

We now present an algorithm that we used to compute coefficients $a(k, m, n)$ numerically in (2.7). Let us take the monic Charlier polynomials \{\(c_i(\xi; \lambda), i = 0, 1, 2, \ldots\)\} for a \(\xi \sim \text{Pois}(\lambda)\) as an example. Charlier polynomials are known as

\[
\kappa_n(\xi; \lambda) = \sum_{j=0}^{n} \xi^j \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} \lambda^{-k} s(k, j) = \sum_{j=0}^{n} C_{nj} \xi^j,
\]

where $s(k, j)$ is a Stirling number of the first kind, and $C_{nj}$ is the coefficient for $\xi^j$ in $\kappa_n(\xi; \lambda)$. Let us determine the value of $a(m + n, m, n)$ to $a(0, m, n)$ by matching coefficients for both sides of (2.7) from $x^{m+n}$ to $x^0$ one by one. Equation (2.7) in terms of Charlier polynomials is written as

\[
\kappa_{m}(\xi; \lambda)\kappa_{n}(\xi; \lambda) = \sum_{k=0}^{m+n} a(k, m, n) c_k(\xi; \lambda), \quad m, n = 0, 1, 2, \ldots.
\]

Since the \{\(c_i(\xi; \lambda), i = 0, 1, 2, \ldots\)\} are monic polynomials, the coefficient of $x^{m+n}$ on the left-hand side is 1, and on the right-hand side it is $a(m + n, m, n)$. Therefore $a(m + n, m, n) = 1$.

Next by matching the coefficients of $x^{m+n-1}$ on both sides, we have $C_{m,m}C_{n,n-1} + C_{m,m-1}C_{m,n} = a(m + n, m, n)C_{m+n, m+n-1} + a(m + n - 1, m, n)C_{m+n-1, m+n-1}$, therefore $a(m + n - 1, m, n) = C_{n,n-1} + C_{n,m-1} - C_{m+m, m+n-1}$.

In the same way by matching the coefficients of $x^{m+n-2}$ on both sides, and by knowing the value of $a(m + n, m, n)$ and $a(m + n - 1, m, n)$, we have $a(m + n - 2, m, n) = (C_{m,m-2} + C_{m,m-1}C_{n,n-1} + C_{n,n-2}) - (C_{m+m, m+n-2} + a(m + n - 1, m, n)C_{m+m-1, m+n-2})$.

If we continue this procedure, all the coefficients $a(k, m, n), k = 0, 1, 2, \ldots, m + n$, can be generated. Although we take the Charlier polynomials as an example here, this procedure obviously is applicable to any other orthogonal polynomials.

**REFERENCES**


Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.


