PoCET: a Polynomial Chaos Expansion Toolbox for Matlab

Felix Petzke * Ali Mesbah ** Stefan Streif *

* *Automatic Control & System Dynamics Lab at Technische Universität Chemnitz, Faculty for Electrical Engineering and Information Technology, Chemnitz, Germany (e-mail: {felix.petzke, stefan.streif}@etit.tu-chemnitz.de).*
** *Department of Chemical and Biomolecular Engineering, University of California, Berkeley, CA, USA (e-mail: mesbah@berkeley.edu).*

Abstract: We introduce PoCET: a free and open-scource Polynomial Chaos Expansion Toolbox for MATLAB, featuring the automatic generation of polynomial chaos expansion (PCE) for linear and nonlinear dynamic systems with time-invariant stochastic parameters or initial conditions, as well as several simulation tools. It offers a built-in handling of Gaussian, uniform, and beta probability density functions, projection and collocation-based calculation of PCE coefficients, and the calculation of stochastic moments from a PCE. Efficient algorithms for the calculation of the involved integrals have been designed in order to increase its applicability. PoCET comes with a variety of introductory and instructive examples. Throughout the paper we show how to perform a polynomial chaos expansion on a simple ordinary differential equation using PoCET, as well as how it can be used to solve the more complex task of optimal experimental design.

Keywords: polynomial chaos, simulation tools, optimal experiment design, optimal control

1. INTRODUCTION

Uncertainty quantification and propagation have always been irrefutably relevant topics in natural sciences and engineering, since almost every real-life process is at some point subject to disturbances or parametric uncertainties, determining its outcome in a somewhat random way. Of the several approaches available to describe and propagate probability densities, this paper focusses on the one devised by Norbert Wiener in 1938: the polynomial chaos expansion (PCE). In its core, a PCE is a series expansion method for density functions, based on distribution-specific polynomial basis functions. Through this expansion, any system with time-invariant uncertainties (e.g. in initial conditions or parameters) can be transformed into a set of deterministic equations. While the transformed system allows for very fast (online) simulation times compared to sampling-based approaches, the (offline) transformation itself can be computationally very expensive, since it requires solving a large number of integrals.

However, due to the increase of computational power in recent years, PCE has become more applicable to numerical problems. Its main applications range from uncertainty quantification (Witteveen and Bijl (2006); Eldred and Burkardt (2009); Ng and Eldred (2012); Savin and Couaillier (2019)), to general stochastic differential equations (Xiu and Karniadakis (2002), McKenzie (2012)) and systems with probabilistic uncertainties (Heuveline and Schick (2013); Kim et al. (2013)). Furthermore, approaches dealing with time-variant uncertainties have been developed (Gerritsma et al. (2010)). And specifically the possibility of fast uncertainty propagation has made PCE a widely applied tool for different online optimization prob-

lems like stochastic MPC (Mesbah et al. (2014); Paulson et al. (2014)) or experimental design (Streif et al. (2014); Mesbah and Streif (2015)).

This paper presents PoCET, a MATLAB toolbox specifically designed to perform a projection-based polynomial chaos expansion on dynamic systems with time-invariant uncertainties. Our aim was to automate the processes of choosing and setting up a sufficiently large polynomial basis, calculating the involved integrals and expanded system matrices, and writing the respective MATLAB function files required to solve the expanded system. While there are numerous toolboxes on uncertainty quantification in MATLAB already available (notably UQLab; Marelli and Sudret (2014)) they do not provide an automated method for uncertainty propagation in the Galerkin projection PCE framework for ODEs (cf. Sec. 2), which is particularly useful for polynomial systems (cf. Sec. 4.1).

PoCET is an open-source toolbox [1] that comes with detailed introductory and instructive examples and it is available for free. Note that PoCET requires MATLAB's Symbolic Math Toolbox for parsing; apart from that it is stand-alone.

Notation: N_ξ and P denote the number of independent uncertain variables and the order of the PCE, respectively. φ_n is a polynomial of order n from the polynomial basis $\mathbf{\Phi}$. States, their expansions, and the vector of their expansions are denoted by x, $\hat{x}_i$, and $\hat{\mathbf{x}}$, respectively. μ is a probability density function with corresponding support Ω and the m-th stochastic moment $\nu^{(m)}$. Gaussian variables are

---

[1] The latest version is available for download on GitHub. See `www.tu-chemnitz.de/etit/control/research/PoCET/` for more information.

denoted by $\xi \sim \mathcal{N}(\nu^{(1)}, \tilde{\nu}^{(2)})$, uniformly distributed ones by $\xi \sim \mathcal{U}(a, b)$, and beta-distributed ones by $\xi \sim \mathcal{B}(\alpha, \beta)$ or $\xi \sim \mathcal{B}_4(\alpha, \beta, l, u)$ for the standard or 4-parameter beta distributions, respectively. The Kronecker product is denoted by $\otimes$. We use the short-hand notation $\sum_{i_1=0}^N + \cdots + \sum_{i_m=0}^N = \sum_{i_1,\ldots,i_j=0}^N$ for a multi-indexed summation.

2. POLYNOMIAL CHAOS EXPANSION

This section very briefly introduces the main idea and mathematical foundation, and highlights common challenges and limitations. Thorough introductions can be found in McKenzie (2012); O'Hagan (2013); Sudret (2014).

2.1 General idea and mathematical foundation

The general (truncated) PCE of dimension N_ξ and order P of a random variable v is given by

$$x \approx \sum_{i=0}^{\tilde{P}-1} \hat{x}_i \varphi_i(\hat{\boldsymbol{\xi}}) = \hat{\boldsymbol{x}}^T \boldsymbol{\Phi}(\hat{\boldsymbol{\xi}}) \tag{1}$$

with $\tilde{P} = \frac{(N_\xi + P)!}{N_\xi! P!}$, $\hat{\boldsymbol{x}}, \boldsymbol{\Phi} \in \mathbb{R}^{\tilde{P}}$, and $\hat{\boldsymbol{\xi}} \in \mathbb{R}^{N_\xi}$. The polynomial basis $\boldsymbol{\Phi}$ consists of P orthogonal polynomials φ_n with

$$\langle \varphi_i, \varphi_j \rangle = \int_\Omega \varphi_i(\xi) \varphi_j(\xi) \rho(\xi) d\xi = \lambda_i \delta_{ij} \tag{2}$$

with $\lambda_i \in \mathbb{R}$ and $\delta_{ij} := \{1 \text{ if } i = j, \ 0 \text{ else}\}$. The actual polynomials φ_n depend on the probability densitiy function (PDF) of ξ and for many common distributions such orthogonal bases are known (cf. Savin and Couaillier (2019) and references therein). So far, PoCET supports Gaussian, uniform and beta distributions (i.e. Hermite, Legendre, and Jacobi polynomial bases, respectively).

Furthermore, there are two general approaches to calculate the PCE coefficients: non-intrusive (or projection based, cf. Eldred et al. (2008)), henceforth called **pPCE**, and intrusive (or collocation based) methods, henceforth called **cPCE**. PoCET was specifically designed for Galerkin projection based PCE, however, collocation is still supported since the former is only applicable to polynomial systems. A detailed comparison between both methods can be found in Eldred and Burkardt (2009).

2.2 PCE for dynamic systems

Consider the ordinary differential equation (ODE)

$$\dot{x}(t) = a(\xi) x(t) \tag{3}$$

where $a(\xi)$ is stochastic and time-invariant. The extended system is then given by (cf. equation (1))

$$\sum_{n=0}^{\tilde{P}-1} \dot{\hat{x}}_n \varphi_n = \sum_{j=0}^{\tilde{P}-1} \hat{a}_j \varphi_j \sum_{k=0}^{\tilde{P}-1} \hat{x}_k \varphi_k. \tag{4}$$

In order to solve this extended system of ODEs, we project the entire system onto $\varphi_i \in \boldsymbol{\Phi}$, which yields

$$\sum_{n=0}^{\tilde{P}-1} \dot{\hat{x}}_n \langle \varphi_n, \varphi_i \rangle = \sum_{j,k=0}^{\tilde{P}-1} \hat{a}_j \hat{x}_k \langle \varphi_j \varphi_k, \varphi_i \rangle, \tag{5}$$

and further, due to orthogonality of φ_n and φ_i,

$$\dot{\hat{x}}_i = \frac{1}{\langle \varphi_i, \varphi_i \rangle} \sum_{j,k=0}^{\tilde{P}-1} \hat{a}_j \hat{x}_k \langle \varphi_j \varphi_k, \varphi_i \rangle = \sum_{j=0}^{\tilde{P}-1} \boldsymbol{E}_j^{(1)} \hat{a}_j \hat{\boldsymbol{x}} \tag{6}$$

with

$$\boldsymbol{E}_j^{(1)} = \begin{bmatrix} e_{j00} & e_{j10} & \cdots & e_{jS0} \\ e_{j01} & e_{j11} & \cdots & e_{jS1} \\ \vdots & \vdots & \ddots & \vdots \\ e_{j0S} & e_{j1S} & \cdots & e_{jSS} \end{bmatrix}, \ e_{jki} := \frac{\langle \varphi_j \varphi_k, \varphi_i \rangle}{\langle \varphi_i, \varphi_i \rangle}, \tag{7}$$

and $S = \tilde{P} - 1$. The superscript (1) denotes the order of the coefficient matrix E, which is equal to the sum of orders of all states that appear in the respective monomial (cf. Sec. 4.1 for more details). In order to solve (6), we calculate the values of the coefficients $\hat{a}_j$ as

$$\hat{a}_j = \frac{\langle a(\xi), \varphi_j \rangle}{\langle \varphi_j, \varphi_j \rangle}. \tag{8}$$

Alternatively, we can calculate these coefficients with a least-squares optimization approach using samples of the respective random variables as

$$\hat{\mathbf{a}}^\star = \arg\min_{\hat{\mathbf{a}}} (\mathbf{y} - \boldsymbol{\Phi}\hat{\mathbf{a}})^\top (\mathbf{y} - \boldsymbol{\Phi}\hat{\mathbf{a}}), \tag{9}$$

where $\mathbf{y} = [a(\xi_1), \ldots, a(\xi_q)]$ is the vector of observations and $\xi_1, \ldots, \xi_q$ are samples of the random variable ξ. This approach is generally referred to as stochastic regression or collocation.

2.3 Challenges and limitations

There are several common challenges regarding the implementation of a PCE. Setting up the extended system (6) requires solving $\tilde{P}^{p+2}$ integrals, where p is the highest order of monomials. In PoCET these integrals are solved using adaptive quadrature rules, which are well known for all of the supported polynomials (cf. Press et al. (1992)). Furthermore, the n-th stochastic moment from the PCE coefficients after solving the extended system can be calculated as

$$\nu^{(m)}(x) = \int_\Omega x^m(\xi) \mu(d\xi)$$

$$\approx \sum_{i_1,\ldots,i_m=0}^{\tilde{P}-1} \hat{x}_{i_1} \cdots \hat{x}_{i_m} \langle \varphi_{i_1} \cdots, \varphi_{i_m} \rangle \tag{10}$$

$$= \big(\underbrace{\hat{\boldsymbol{x}} \otimes \cdots \otimes \hat{\boldsymbol{x}}}_{m \text{ times}}\big)^T \hat{\boldsymbol{E}}_\nu^{(m)}$$

with $\hat{\boldsymbol{E}}_\nu^{(m)} = \big[\epsilon_{0\ldots0}, \ldots, \epsilon_{0\ldots S}, \ldots, \epsilon_{1\ldots S}, \ldots, \epsilon_{S\ldots S}\big]^\top \in \mathbb{R}^{\tilde{P}^m}$ and $\epsilon_{i_1\ldots i_m} := \langle \varphi_{i_1} \cdots \varphi_{i_{m-1}}, \varphi_{i_m} \rangle$ which requires solving another $\tilde{P}^m$ integrals. Lastly, modeling time-varying random signals (e.g. noise) still remains challenging and has only been approached in recent years (see, e.g., Paulson and Mesbah (2019)).

3. MAIN FUNCTIONALITIES OF PoCET

Our aim of designing PoCET was to facilitate the application of a polynomial chaos expansion to dynamic systems in MATLAB. In order to demonstrate its usage, we consider the autonomous system

$$\dot{x}(t) = -a(\xi) x(t), \ x(0) = 2, \\ a \sim \mathcal{B}(2, 2), \tag{11}$$

where a is beta-distributed on the interval $[0, 1]$ with shape parameters $\alpha = \beta = 2$. The following sections provide a step-by-step demonstration of how to define, expand, and simulate this system using PoCET.

3.1 System definition

There are four pre-defined building blocks we can use to define systems in PoCET: states, parameters, inputs, and outputs, which are each defined as structures (MATLAB stucts). States are time variant variables that are defined by an ODE or difference equation, respectively, as well as a possibly uncertain initial condition. Parameters are time invariant variables and therefore only defined by a probability distribution. Inputs are possibly time variant functions explicitly defined by the user. And outputs are time-invariant functions of the states.

Defining system (11) amounts to the code

```
states(1).name = 'x'; % variable name
states(1).pdf  = 'dirac'; % variable's
    probability density function
states(1).data = 2; % distribution's parameters
states(1).rhs  = '-a*x'; % right hand side of ODE

parameters(1).name = 'a';
parameters(1).pdf  = 'beta';
parameters(1).data = [2, 2];
```

where the expected input for the property `data` depends on the chosen distribution (e.g. a uniform distribution is defined by its lower and upper bound; cf. `help PoCET`).

3.2 Generating the expanded system

This step includes one of the main contributions of PoCET: the automatic generation of the extended system (6) from a specified ODE and writing the respective function files required for a simulation. First we define the desired order P of the PCE and then call the following functions using the structures defined in the previous step.

```
pce_order = 3; % desired order of the PCE
pce_sys = PoCETcompose(states,parameters,[],[],
    pce_order); % analyze input and generate PCE
PoCETwriteFiles(pce_sys,'PCEODE.m',[],'NOMODE.m')
    % create .m-files for ODE functions
```

The main work here is done by the function `PoCETcompose`, which checks the input structures for discrepancies, completeness, and applicability of projection based PCE. If the latter is true, it also computes all of the involved PCE coefficients $\boldsymbol{E}_j^{(p)}$ (cf. Eq. (7)) via Galerkin projection. The structure `pce_sys` contains these coefficients, as well as the initial conditions for the extended system. `PoCETwriteFiles` creates .m-function files in the current directory which contain the expanded ODE system (`PCEODE.m`) and the nominal ODE system (`NOMODE.m`), respectively, the former of which is used for pPCE simulations and the latter for cPCE or Monte Carlo simulations.

3.3 System simulation

We carry out the simulation of the expanded system using the system structure and the automatically generated ODE function files from Step 3.2 by employing either the function `PoCETsimGalerkin` or `PoCETsimCollocation` for pPCE or cPCE, respectively. Both functions output the solution of the PCE system, i.e. the values of the extended states $\hat{\mathbf{x}}_i(t)$ over the specified time horizon. Additionally, the function `PoCETsimMonteCarlo` provides a quick way to perform a Monte Carlo simulation of the specified system. Note that all simulation routines offered by PoCET are essentially wrappers for MATLAB's built-in ODE solvers and therefore all of the usual options can be specified. To do so, just create a structure including simulation time, step size, the desired solver, and its respective options.

```
simoptions.tspan = [0 1]; % simulation time span
simoptions.dt = 0.005; % output time step size
simoptions.solver = 'ode15s'; % desired solver
simoptions.setup = odeset; % solver options
```

Projection-based PCE   Using a Galerkin projection approach generally allows for very fast online computation times, since we already calculated all PCE coefficients offline in the previous step. We solve the expanded system by calling the function `PoCETsimGalerkin`.

```
gal_results = PoCETsimGalerkin(pce_sys,'PCEODE'
    ,[],simoptions); % system simulation
```

The structure `gal_results` contains fields for all states and outputs of the system which in turn contain the solution of their respective expansions over time. These can then be used for either quick sampling or to calculate the moments of the original variables. We can also solve the system manually using the generated ODE function files and the initial conditions from the system structure `pce_sys`, which generally yields faster computation times.

Collocation-based PCE and sampling   Collocation methods provide an alternative way to calculate the PCE coefficients using samples of the involved random variables (cf. Eq. (9)). PoCET has a dedicated routine called `PoCETsample` for drawing samples as they are used for several different purposes. As mentioned above, PoCET's cPCE routine `PoCETsimCollocation` uses the nominal system ODE for simulation as well as samples from the stochastic basis ξ.

```
n_samples = 10000; % number of samples
basis = PoCETsample(pce_sys,'basis',n_smpls); %
    draw samples for stochastic basis
col_results = PoCETsimCollocation(pce_sys,'NOMODE
    ', [],basis,simoptions); % simulate system
```

The structure `col_results` again contains the solutions of the expanded variables.

Monte Carlo simulation   Similar to cPCE, Monte Carlo simulations use the nominal ODE as well as samples on the involved random variables. In this case, however, we do not use samples of the stochastic basis ξ but samples of the actual random variable $a(\xi)$, which is done by calling `PoCETsample` with the option `variables`.

```
mc_samples = 10000; % number of samples
vars = PoCETsample(pce_sys,'variables',mc_samples
    ); % draw samples for system variables
mc_results = PoCETsimMonteCarlo(pce_sys,'NOMODE',
    [],vars,simoptions); % system simulation
```

The resulting structure `mc_results` contains the sampled trajectories of the original states.

3.4 Moment calculation and PDF fitting

After solving the expanded system we now show how to retrieve stochastic information about the original states $x(\xi)$. In order to compute their first m moments $\nu^{(m)}(x)$, we first have to calculate the coefficients $\hat{\boldsymbol{E}}_\nu^{(m)}$ (cf. Eq. (10)), which is done via the function `PoCETmomentCoeffs`. Afterwards we call `PoCETcalcMoments` to calculate the actual moments.

```
m = 4; % highest order to be calculated
MomCoeffs = PoCETmomentCoeffs(pce_sys,m); %
    calculate coefficients for moment calculation
results.x.moments = PoCETcalcMoments(pce_sys,
    MomCoeffs,results.x.pcvals); % calc. moments
```

The resulting field `moments` is an $m \times k$ matrix where k is the number of time steps in the simulation. Fig. 1 shows the resulting trajectories for the the first four moments in comparison to a Monte-Carlo simulation, which can be done as described above. For an analysis of the convergence properties of general polynomial chaos the interested reader is referred to Ernst et al. (2012).

In order to better visualize the results it might be desirable to fit a probability density functions of the original states, instead of just plotting the moments. To do so, PoCET features a routine for recovering a 4-parameter beta distribution from the first four moments, based on Hanson (1991). In the example below, we fist calculate the respective moments for the last time step of the simulation, then recover the shape defining parameters α and β, as well as the lower and upper bounds of the support.

```
x_beta4 = calcBeta4(x_moments_final);
```

Fig. 2 shows the resulting 4-parameter beta distribution of the state x at the end of the simulation as well as a histogram of respective samples drawn by the Monte-Carlo simulation. Alternatively we could use the PCE coefficients from the system solution to create samples employing the function `PoCETsample`. These can then be used to fit a PDF with one of the available functions in MATLAB, like `fitdist` from the Statistics and Machine Learning Toolbox.
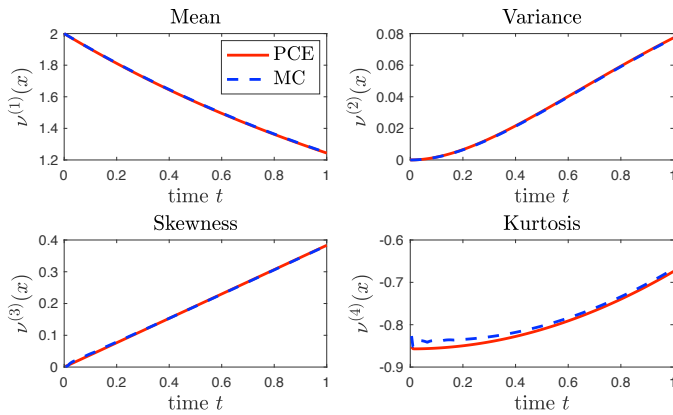

Fig. 1. Comparison of trajectories for the first four moments of of state x in (11) obtained by a pPCE of order 3 and a Monte-Carlo (MC) simulation using 10^6 samples. Simulation and moment calculation took $0.08\,\mathrm{s}$ for the pPCE and $333.29\,\mathrm{s}$ for the MC approach.
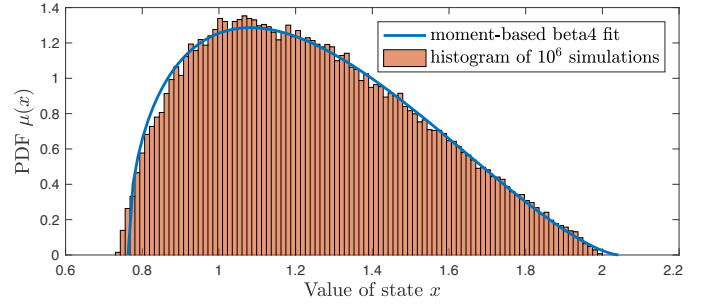

Fig. 2. PDF $\mu(x)$ after simulating system (11) for $1\,\mathrm{s}$ and comparison between a fitted 4-parameter beta distribution and a histogram of 10^6 samples.

3.5 Updating the extended system

As long as the distributions of the uncertainies do not change it is not necessary to recompute the entire PCE when their actual values change, since this only corresponds to an update of their respective PCE coefficients. For example, changing a in (11) from $a \sim \mathcal{B}(2,2)$ to $a \sim \mathcal{B}(4,6)$ can be achieved by calling

```
pce_sys = PoCETupdate(pce_sys,'a',[4,6]);
```

where the last input argument is the new `data` property of the parameter and therefore has to match its expected format (cf. Sec. 3.1).

4. ADVANCED EXAMPLES

This section provides a general overview of dealing with polynomial and general nonlinear systems as well as an extended example of how PoCET can be used for experimental design.

4.1 Higher-order polynomial and nonlinear systems

We can handle polynomial systems directly using a projection-based PCE, since multiplication does not interfere with the orthogonality of the basis polynomials. However, the number of integrals to be solved on setting up the extended system increases significantly. If we consider, for instance, the system

$$\dot{x} = a(\xi)x^3 \tag{12}$$

we calculate its expansion analogously to (5) as

$$\dot{\hat{x}}_i = \frac{1}{\langle \varphi_i, \varphi_i \rangle} \sum_{j,k,l,m=0}^{\tilde{P}-1} a_j \hat{x}_k \hat{x}_l \hat{x}_m \langle \varphi_j \varphi_k \varphi_l \varphi_m, \varphi_i \rangle$$
$$= \sum_{j=0}^{\tilde{P}-1} \boldsymbol{E}_j^{(3)} a_j \hat{\boldsymbol{x}} \otimes \hat{\boldsymbol{x}} \otimes \hat{\boldsymbol{x}}. \tag{13}$$

The dimension of the coefficient matrix is $\boldsymbol{E}_j^{(3)} \in \mathbb{R}^{\tilde{P} \times \tilde{P}^3}$, i.e. $\tilde{P}^4$ integrals have to be solved (cf. Eq. (7)). In order to apply a PCE to a general nonlinear system, it is necessary to use the collocation approach in PoCET, since the involved algorithms to solve the projections in Eq. (5) heavily exploit orthogonality of the polynomials φ_i, which does not hold for general nonlinear functions with φ_i as arguments. However, most nonlinear systems can be transformed into a polynomial system using immersions, as decribed in Ohtsuka (2005). Note, however, that this

approach is only viable for low-dimensional nonlinear systems, since it amounts to an additional state space expansion on top of the one done by the PCE.

4.2 Optimal experimental design

One of the major advantages of PCE is the low computational effort associated with system simulation compared to sampling-based methods. This makes it highly suitable for tasks that require several simulations like model predictive control problems or optimal experimental design. In the following we consider an optimal experimental design problem previously analyzed in Henri (1902); Streif et al. (2014), with the Michaelis-Menten and Henri mechanisms as two assumed model hypotheses for an enzyme-catalyzed reaction. Both reaction models consider a substrate S and an enzyme E, which form an enzyme-substrate complex C and a final product P (cf. Rumschinski et al. (2010)). The model dynamics follow the law of mass action, where x_1 and x_2 denote the concentrations of the substrate and complex, respectively. The first model candidate, describing the Henri mechanism, is defined as

$$\dot{x}_1^H = \left(p_1^H + p_3^H\right)\left(x_2^H - 1\right)x_1^H + \left(p_2^H + u\right)x_2^H$$
$$\dot{x}_2^H = p_1^H\left(1 - x_2^H\right)x_1^H - \left(p_2^H + u\right)x_2^H, \tag{14}$$

while the second one, describing the Michaelis-Menten mechanism, is defined as

$$\dot{x}_1^M = p_1^M\left(x_2^M - 1\right)x_1^M + \left(p_2^M + u\right)x_2^M$$
$$\dot{x}_2^M = p_1^M\left(1 - x_2^M\right)x_1^M - \left(p_3^M + p_2^M + u\right)x_2^M, \tag{15}$$

where the input u is assumed to affect the reaction rate p_2^* in an additive manner. Furthermore, we assume uncertainties in the initial concentrations (with $x_1^*(0) \sim \mathcal{B}_4(3,3,0.96,0.98)$ and $x_2^*(0) \sim \mathcal{B}_4(3,3,0.01,0.03)$, respectively) and uncertain estimates for the parameter values (with $p_i^H \sim \mathcal{U}(0.9,1.1)$ and $p_i^M \sim \mathcal{U}(0.9,1.15)$ for all $i = 1,2,3$) to be available. Under these uncertainties and assuming that we take a measurement 10 s after the reaction started, the two models are barely distinguishable with a small number of measurements since the resulting PDFs have a very large overlap, as shown in Figs. 3(a) and 4(a). This makes it almost impossible to decide which model hypothesis is the right one.

Our goal is to find a piecewise constant input signal u that discriminates the two models. Defining such an input in PoCET is done in the system definition. While the ODEs for the states are defined as shown in Step 3.1. This time, however, we also define an input structure via the code

```
inputs.name = 'u';
inputs.rhs  = 'piecewise(u_t,u_v,t)'; % define
    input as piecewise constant signal
inputs.u_t = [0,1,2,3,4]; % time values at which
    input value changes
inputs.u_v = [0,0,0,0,0]; % input values to be
    taken at times defined in u_t
```

where `piecewise(u_t,u_v,t)` is a function that returns $u_v(k)$ for all times $t \in [u_t(k), u_t(k+1)]$. Note that the additional fields in the input structure follow no predefined scheme – their names essentially act as additional variables that can be changed in the online simulation. If we want to apply a stairs-shaped input to the system above, we call

```
gal_results = PoCETsimGalerkin(pce_sys,'PCEODE.m'
    ,[],simoptions,'u_v',[1,2,3,4,5]);
```

where the last input has to match the one defined in the `inputs` structure above.

Using this, we set up a model-based optimization problem for `u_v` employing the `fmincon` function and the Bhattacharyya distance (cf. Kailath (1967)) as a similarity measure for the resulting PDFs. We implemented the latter as a nonlinear constraint in order to enforce the discrimination of the two models. The optimization took 11.5 s, including a total of 185 evaluations of the nonlinear constraint (i.e. 370 simulations). Fig. 3(b) shows the final PDFs under the discriminating input. Fig. 4(b) gives an even stronger result: since it shows the outer bounds of the concentrations x_2^* we can see that any measurement taken after 5 s or later would allow to decide which model hypothesis is the right one. The complete example with additional comments on the usage is included in PoCET, while more details about the employed methods can be found in Streif et al. (2014).
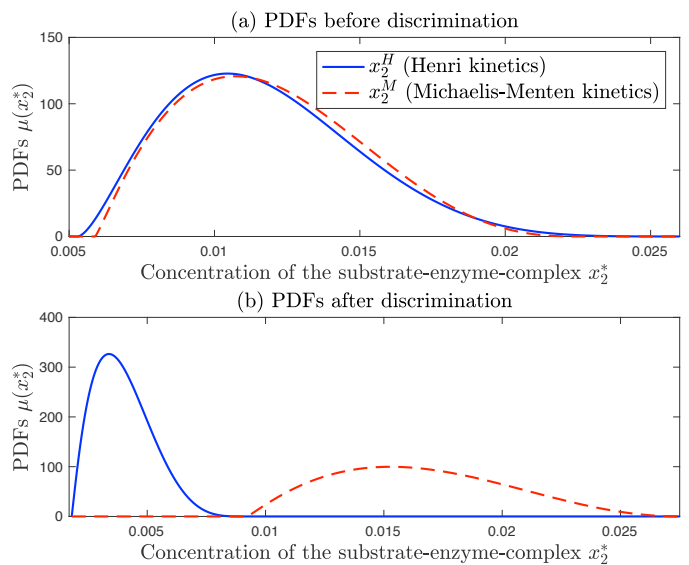
Fig. 3. Probability density functions of x_2^* after 10 s without (above) and with (below) discriminating input.
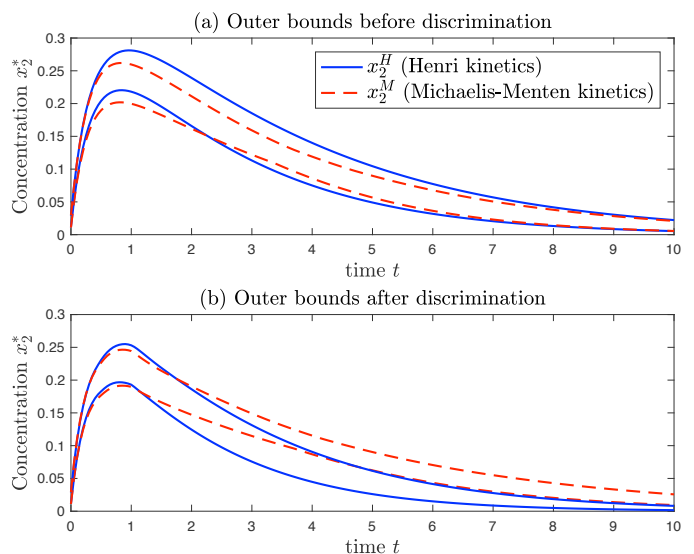
Fig. 4. Outer bounds of the PDFs of the substrate-enzyme-complex concentrations over time.

5. FURTHER REMARKS AND OUTLOOK

This paper introduced PoCET, a polynomial chaos expansion toolbox for Matlab. Its main contribution is the automated expansion of polynomial ODE systems, which allows for propagation of probabilistic uncertainties within the projection-based PCE framework. PoCET features a very straight-forward system definition and requires little to no knowledge of the PCE framework itself. It relies on the Symbolic Math Toolbox for input parsing but is otherwise stand-alone, open-source, and available for free (see www.tu-chemnitz.de/etit/control/research/PoCET/ for more information). Due to its modular design, it can be adapted into many different directions (e.g. arbitrary PCE by adjusting the quadrature rules for the integration), used for various applications (e.g. stochastic MPC or fault detection), or combined with other uncertainty quantification tools that support polynomial chaos, like UQLab (Marelli and Sudret (2014)).

REFERENCES

Eldred, M. and Burkardt, J. (2009). Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification. In *47th AIAA Aerospace Sciences Meeting*.

Eldred, M., Webster, C., and Constantine, P. (2008). Evaluation of non-intrusive approaches for Wiener-Askey generalized polynomial chaos. In *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*.

Ernst, O.G., Mugler, A., Starkloff, H.J., and Ullmann, E. (2012). On the convergence of generalized polynomial chaos expansions. *ESAIM: Mathematical Modelling and Numerical Analysis*, 46(2), 317–339.

Gerritsma, M., Van der Steen, J.B., Vos, P., and Karniadakis, G. (2010). Time-dependent generalized polynomial chaos. *Journal of Computational Physics*, 229(22), 8333–8363.

Hanson, B. (1991). Method of moments estimates for the four-parameter beta compound binomial model and the calculation of classification consistency indexes. *ACT Research Report Series*, 91, 1–21.

Henri, V. (1902). General theory of the action of some glycoside hydrolases. *C. R. H. Acad. Sci. Paris*, 135, 916–919.

Heuveline, V. and Schick, M. (2013). A hybrid generalized polynomial chaos method for stochastic dynamical systems. *International Journal for Uncertainty Quantification*, 4, 37–61.

Kailath, T. (1967). The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communication Technology*, 15(1), 52–60.

Kim, K.K.K., Shen, D.E., Nagy, Z.K., and Braatz, R.D. (2013). Wiener's polynomial chaos for the analysis and control of nonlinear dynamical systems with probabilistic uncertainties. *IEEE Control Systems Magazine*, 33(5), 58–67.

Marelli, S. and Sudret, B. (2014). UQLab: A framework for uncertainty quantification in Matlab. In *2nd International Conference on Vulnerability, Risk Analysis and Management*, 2554–2563.

McKenzie, B. (2012). *Polynomial Chaos Expansions for Random Ordinary Differential Equations*. MA thesis, Oregon State University.

Mesbah, A. and Streif, S. (2015). A probabilistic approach to robust optimal experiment design with chance constraints. In *9th International Symposium on Advanced Control of Chemical Processes*, volume 48, 100–105. Whistler.

Mesbah, A., Streif, S., Findeisen, R., and Braatz, R.D. (2014). Stochastic nonlinear model predictive control with probabilistic constraints. In *American Control Conference*, 2413–2419.

Ng, L. and Eldred, M. (2012). Multifidelity uncertainty quantification using non-intrusive polynomial chaos and stochastic collocation. In *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*.

O'Hagan, A. (2013). Polynomial chaos: A tutorial and critique from a statistician's perspective. Submitted to SIAM/ASA Journal of Uncertainty Quantification.

Ohtsuka, T. (2005). Model structure simplification of nonlinear systems via immersion. *IEEE Transactions on Automatic Control*, 50(5), 607–618.

Paulson, J.A., Mesbah, A., Streif, S., Findeisen, R., and Braatz, R.D. (2014). Fast stochastic model predictive control of high-dimensional systems. In *53rd IEEE Conference on Decision and Control*, 2802–2809.

Paulson, J. and Mesbah, A. (2019). An efficient method for stochastic optimal control with joint chance constraints for nonlinear systems. *International Journal of Robust and Nonlinear Control*, 29, 5017–5037.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., and Flannery, B.P. (1992). *Numerical Recipes in C*. Cambridge University Press, Cambridge, USA, second edition.

Rumschinski, P., Borchers, S., Bosio, S., Weismantel, R., and Findeisen, R. (2010). Set-based dynamical parameter estimation and model invalidation for biochemical reaction networks. *BMC Systems Biology*, 4(69).

Savin, E. and Couaillier, V. (2019). Generalized polynomial chaos for non-intrusive uncertainty quantification in computational fluid dynamics. *Notes on Numerical Fluid Mechanics and Multidisciplinary Design*, 140, 123–141.

Streif, S., Petzke, F., Mesbah, A., Findeisen, R., and Braatz, R.D. (2014). Optimal experimental design for probabilistic model discrimination using polynomial chaos. In *19th IFAC World Congress*, volume 47, 4103–4109. Cape Town, South Africa.

Sudret, B. (2014). *Risk and Reliability in Geotechnical Engineering*, chapter Polynomial chaos expansions and stochastic finite element methods, 265–300. CRC Press.

Witteveen, J.A.S. and Bijl, H. (2006). Modeling arbitrary uncertainties using gram-schmidt polynomial chaos. In *44th AIAA Aerospace Sciences Meeting and Exhibit*.

Xiu, D. and Karniadakis, G.E. (2002). The wiener-askey polynomial chaos for stochastic differential equations. *SIAM Journal of Scientific Computation*, 24, 619–644.