User guide for nsPCE toolbox

Joel A. Paulson

July 17, 2019

1 Introduction

The nsPCE toolbox [1] is a Matlab-based code that implements methods for the construction of non-smooth polynomial chaos expansions (nsPCE) surrogate models. These surrogate models can be used to accelerate the solution of uncertainty quantification (UQ) problems of interest in complex biological systems including global sensitivity analysis and parameter estimation. The nsPCE method represents an extension of traditional PCE that is able to effectively capture singularities in the model response that can occur due to discrete/sudden changes in the system behavior. In particular, these discrete events are known to commonly occur in dynamic flux balance analysis (DFBA) models that couple intracellular fluxes, found from the solution of a constrained metabolic network model of the cellular metabolism, to the time-varying nature of the extracellular substrate and product concentrations. Although geared toward DFBA models, this code can in principle be applied to any model of interest including those described by other smooth or non-smooth functions. The proposed method on which this code is based is described in detail in [2].

2 Installation

The nsPCE code requires a DFBA solver. There are a variety of codes available, however, we suggest that users obtain a copy of DFBAlab [3] from `https://yoric.mit.edu/software/dfbalab/how-obtain-dfbalab` for two main reasons. First, the example files are based on DFBAlab, meaning one can directly copy-and-paste the required files and then modify them when applying the code to a new problem. Second, DFBAlab has numerical advantages over alternative implementations due to its use of lexicographic optimization that enables unique fluxes to be defined throughout the simulation and a limited number of calls to the linear program (LP) solver.

nsPCE also requires a number of sparse PCE operations. The current implementation of the code is based on UQLab [4], which can be freely downloaded for academic users at `https://www.uqlab.com/`. As such, we highly recommend users obtain a copy of UQLab to avoid modifications to the source code.

There should be a folder named "Functions" in your download of nsPCE. Add this folder to your Matlab path, and then you should be able to run the example scripts contained in the folder “Examples”. The provided examples contain a script for constructing the surrogate models (`main_pce`) and another script for solving a related parameter estimation problem (either `main_smc` or `main_map`).

3 Getting started: Example walkthrough

Here, we walkthrough how to setup the nsPCE code using the synthetic metabolic network problem as an example. This example as studied originally in [5, Chapter 8].

First, we define the relevant problem details. The intracellular reaction network consumes a carbon source C, a nitrogen source N, and an oxygen source O to produce lipids L, ethanol E, biomass X, ATP, and some oxidation product COX. Although used for illustrative purposes, this network is meant to mimic the behavior of living organisms in the sense that (i) E can only be produced in the absence of O, (ii) L can only be accumulated in the absence of N, (iii) there is a minimum ATP requirement, and (iv) the aerobic oxidation of C produces more energy than fermentation of C. The set of reactions can be summarized as

$$\begin{aligned}
v_C &: C_{ex} \to C, \\
v_N &: N_{ex} \to N, \\
v_O &: O_{ex} \to O, \\
v_{OX} &: C + O \to S_{ATP,OX} ATP + S_{OX,OX} COX_{ex}, \\
v_{Ferm} &: 4C \to S_{ATP,Ferm} ATP + E_{ex} + S_{OX,Ferm} COX_{ex}, \\
v_L &: 4C + S_{ATP,L} ATP \to L, \\
v_X &: S_{C,X} C + S_{N,X} N + S_{ATP,X} ATP \to X, \\
v_{ATP} &: ATP \to ATP_{maintenance},
\end{aligned} \tag{3.1}$$

where the subscript ex denotes extracellular metabolites and all of the reactions are assumed to be unidirectional. The unknown stoichiometric coefficients are denoted by $S_{i,j}$, where i represents the metabolite name and j represents the reaction name. The dynamic mass balance equations for the extracellular environment are given by

$$\begin{aligned}
\dot{X}(t) &= v_X(t) X(t), & X(0) &= X_0, \\
\dot{C}(t) &= -v_C(t) X(t), & C(0) &= C_0, \\
\dot{N}(t) &= -v_N(t) X(t), & N(0) &= N_0, \\
\dot{O}(t) &= -v_O(t) X(t), & O(0) &= O_0, \\
\dot{L}(t) &= v_L(t) X(t), & L(0) &= 0, \\
\dot{E}(t) &= v_{Ferm}(t) X(t), & E(0) &= 0, \\
C\dot{O}X(t) &= (S_{OX,OX} v_{OX}(t) + S_{OX,Ferm} v_{Ferm}(t)) X(t), & COX(0) &= 0, \\
\dot{\alpha}(t) &= \gamma(t), & \alpha(0) &= 0,
\end{aligned} \tag{3.2}$$

where α is a penalty state that remains equal to zero until the state trajectories become infeasible (e.g., when all of the metabolites are depleted) and the value $\gamma(t)$ is the instantaneous penalty that is automatically computed by `DFBAlab`. The intracellular network (3.1) is linked to the extracellular dynamics (3.2) through the substrate uptake rates, which are given by the following expressions

$$
\begin{aligned}
v_C^{UB}(\mathbf{s}) &= \max\left(0, v_{max,C}\frac{C}{K_C + C}\frac{1}{1+\frac{E}{K_{iE}}}\right),\\
v_N^{UB}(\mathbf{s}) &= \max\left(0, v_{max,N}\frac{N}{K_N + N}\right),\\
v_O^{UB}(\mathbf{s}) &= \max\left(0, v_{max,O}\frac{O}{K_O + O}\right),
\end{aligned}
\tag{3.3}
$$

where $\mathbf{s} = (X, C, N, O, L, E, COX, \alpha)$ is the vector of extracellular species. The unknown parameters are assumed to be

$$
\begin{aligned}
\boldsymbol{X} = [&v_{max,C},\ K_C,\ K_{iE},\ v_{max,N},\ K_N,\ v_{max,O},\ K_O, v_{ATP},\ X_0,\ C_0,\ N_0,\ O_0,\\
&S_{C,X},\ S_{N,X},\ S_{ATP,X},\ S_{ATP,OX},\ S_{ATP,Ferm},\ S_{ATP,L},\ S_{OX,OX},\ S_{OX,Ferm}]^\top.
\end{aligned}
\tag{3.4}
$$

A hierarchical set of objectives is used in the flux balance problem to ensure that unique reaction fluxes are obtained. The twenty uncertain parameters are assumed to be uniformly distributed between upper and lower bounds. These details are provided in the supplementary information of [2], as well as in the example code.

The basic steps required to build nsPCE surrogate models for this type of problem using this toolbox are summarized as follows:

1. Define relevant simulation files in the "dfba_model" folder.

2. Run random simulations to locate potential singularities in the system.

3. Define the quantities of interest function (`qoi_func`) and the time-to-discontinuity function (`tdisc_func`).

4. Run the `main_pce` script to construct the nsPCE surrogate models.

We now discuss each of these steps in detail. These steps can be exactly replicated for any new problem of interest as long as DFBAlab is used for simulation purposes.

3.1 The dfba_model folder

The three files that need to be created by the user are `RHS`, `DRHS`, and `evalForwardModel`.

The `RHS` and `DRHS` are DFBAlab specific files (see [3] and the DFBAlab user manual). Essentially, `RHS` encodes the extracellular substrate uptake bounds, which are shown in (3.3) for this example. Since these bounds are functions of the uncertain parameters, we must have a way to pass changing values for these parameters into this function. This is done with the INFO structure, as shown in lines 20-39. We use an if-else condition to ensure nominal values are used whenever INFO.param is not defined. The `DRHS` function encodes the right-hand side of the ODEs in (3.2). Again, the INFO structure is used to pass potentially uncertain parameters to this function (in this case $S_{OX,OX}$ and $S_{OX,Ferm}$). After defining these functions, we need to create the `evalForwardModel` function that returns the integrated DFBA state and time profiles over a set of samples of the uncertain parameters. Only two parts of this function need to be changed: (i) line 24 must be changed to load parameters from a specified file (this file can be generated and saved after running the `MC_simulation` discussed later) and (ii) lines 46-60 must be changed to ensure that the random parameters "X" are properly passed to INFO.param.

3.2 Monte Carlo simulations

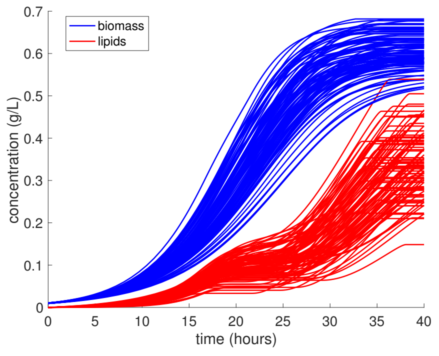
Before attempting to develop surrogate models for a system, it is useful to perform some exploratory simulations to check that things are working properly. This can be done with the `MC_simulation` script that is provided in the "dfba_model" folder. This script is broken up into three sections:

1. The first section defines the uncertainty description and simulation setup, e.g., relevant bounds of the parameter distribution, the type of sampling to be done, and the number of simulations to perform.
2. The second section provides all remaining details of the nominal model that are not in `RHS` and `DRHS`. This includes the reaction network (3.1), the hierarchical objectives for the flux balance problem, the initial conditions for the states, the time span of the simulation, and relevant solver tolerances.
3. The third section executes the Monte Carlo simulations wherein the system is integrated forward in time for each given set of the parameter values. The INFO structure is updated with the given parameter sample in lines 201-215.

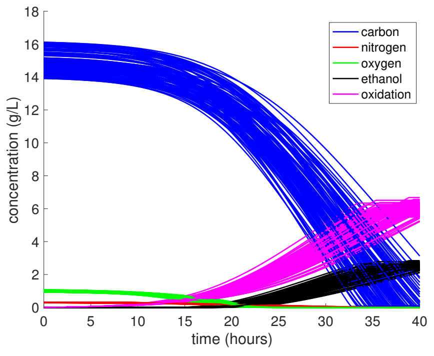
When implementing a new problem, the first two sections need to be fully updated. On the other hand, the third section only needs to be modified by ensuring INFO is being properly passed the parameter samples. Note that the plotting steps at the end of the script also need to be changed. It is usually best to create `evalForwardModel` after this script is working. In that way, you can directly save the variables from `MC_simulation` and copy lines 201-215 from `MC_simulation` to lines 46-60 of `evalForwardModel`. The `MC_simulation` should produce the three plots that are shown in Fig. 3.1, which represent 100 DFBA simulations for various randomly sampled parameter values.

3.3 Quantities of interest

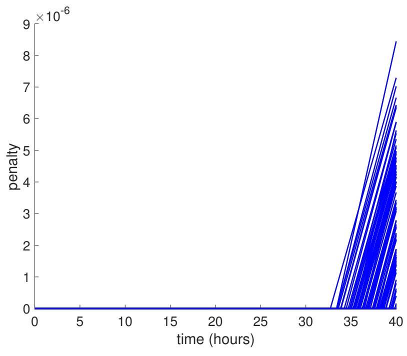
Two files need to be created by the user in the main folder, mainly `qoi_func` and `tdisc_func`. The quantities of interest function is simply whatever function of the DFBA that is of interest and thus is fairly straightforward to define. These will usually be the concentrations themselves at certain times, though other values could be chosen including, e.g., integral



(a)



(b)



(c)

Figure 3.1: **Monte Carlo simulation for the synthetic metabolic network.** The synthetic DFBA model with twenty uncertain parameters is integrated from time 0 to 40 hours for 100 different parameter realizations drawn independently from the uniform prior density. The time profiles are shown for **(a)** biomass and lipids, **(b)** the substrates and products, and **(c)** the penalty state.

of a state over time or the product of states. The main thing to recognize is that this function is meant to take in the DFBA states/times determined by the `evalForwardModel` function, so that it essentially should interpolate these profiles to the desired time of interest. We implemented things in this manner so that the code is efficient whenever multiple time points of interest are considered, which is very common in parameter estimation problems. The `tdisc_func` represents the time that discontinuities happen in the simulation, and is used to partition the parameter space into separate elements. Thus, this function can be neglected when using global models. The easiest way to define `tdisc_func` is to try to locate singularities by eye in Fig. 3.1. We can see a few different possibilities in this example that are a result of a substrate (oxygen, nitrogen, or carbon) being fully consumed. We decided to use the time that the penalty state becomes positive to define `tdisc_func`, as this signifies the consumption of all substrates and results in sharp changes to the lipid and oxidation product concentrations.

3.4 Running the main script

Once all of these functions have been defined, the `main_pce` can be run to build the desired nsPCE surrogate models. Note that all user inputs, which are to be manipulated by the user, are included in the first section of the script. A user should not need to modify any of the remaining parts of the script. Please see the comments for specific details on how to define these variables. We do note that there are a number of UQlab parameters that can be adjusted, if desired. Please reference the UQlab manual for more details on these options as well as proper syntax. The surrogate construction will usually take at least a few minutes, depending on the complexity of the model.

4 Suggestions on the use of nsPCE

The main advantage of nsPCE is that it can handle singular models very effectively, meaning that the equations should have some form of non-smooth behavior to see significant improvements over global PCE. It is usually good practice to start with global PCE, since it is the simplest special case of nsPCE, to see if the desired accuracy can be achieved with high-order polynomial functions directly. If not, then one should try to choose a `tdisc_func` and run the "ns" version of the code.

Since the code is adaptive in the sense that samples are constantly being added to the experimental design whenever the desired tolerance is not met, one must be careful to not run into an infinite loop wherein the tolerance is not achievable due to the finite order of the surrogate. A good rule-of-thumb is to start with high tolerances (e.g., 10^{-2}) and a low N_{max} (e.g., 1000) to ensure that the algorithm is producing reasonable results before attempting to run a more expensive case.

One will usually need to tune the hyperparameters in the nsPCE algorithm in order to achieve the best possible results. Since this requires running the code multiple times, it can be useful to save results from previous runs and use them to start a subsequent run. This procedure is not automated in the `main_pce` can be implemented without too many modifications. A common case is to save the fitted time-to-discontinuity PCE and corresponding experimental design and then load this information on the next run, as opposed to re-fitting this function. It can also be useful to separately build the surrogate models for each quantity of interest, especially whenever there is a large contrast in the complexity of the output functions, to avoid having to use all samples for every time and output.

5 Citations

When using this code, please cite [2]. Also, please remember to cite any relevant packages that are used for DFBA simulation and the sparse PCE operations.

References

[1] Paulson JA. nsPCE; 2019. `https://github.com/joelpaulson/nsPCE`.

[2] Paulson JA, Martin-Casas M, Mesbah A. Fast uncertainty quantification of dynamic flux balance analysis models using non-smooth polynomial chaos expansions. PLoS Computational Biology. 2019; p. under review.

[3] Gomez JA, Höffner K, Barton PI. DFBAlab: A fast and reliable MATLAB code for dynamic flux balance analysis. BMC Bioinformatics. 2014;15:409.

[4] Marelli S, Sudret B. UQLab: A framework for uncertainty quantification in Matlab. In: Vulnerability, Uncertainty, and Risk: Quantification, Mitigation, and Management; 2014. p. 2554–2563.

[5] Gomez JA. Simulation, sensitivity analysis, and optimization of bioprocesses using dynamic flux balance analysis. Massachusetts Institute of Technology; 2018.