A bluff-and-fix algorithm for polynomial chaos methods *

Laura Lyman[1] and Gianluca Iaccarino[2]

[1] Institute for Computational and Mathematical Engineering, 475 Via Ortega, Stanford University, Stanford CA 94305, USA
lymanla@stanford.edu

[2] Department of Mechanical Engineering and Institute for Computational Mathematical Engineering, Stanford University, Building 500, Stanford, CA 94305
jops@stanford.edu

Abstract. Stochastic Galerkin methods can be used to approximate the solution to a differential equation in the presence of uncertainties represented as stochastic inputs or parameters. The strategy is to express the resulting stochastic solution using $M+1$ terms of a polynomial chaos expansion and then derive and solve a deterministic, coupled system of PDEs with standard numerical techniques. One of the critical advantages of this approach is its provable convergence as M increases. The challenge is that the solution to the $M+1$ system cannot easily *reuse* an already-existing computer solution to the M system. We present a promising iterative strategy to address this issue. Numerical estimates of the efficiency of the proposed algorithm (bluff-and-fix) demonstrate that it can be more efficient than using monolithic methods to solve the whole $M+1$ system directly.

Keywords: Polynomial chaos · Galerkin projections · Stochastic differential equations · Numerical PDE solvers · Spectral methods

1 Introduction

Uncertainty quantification (UQ) in physical models governed by systems of partial differential equations is important to build confidence in the resulting predictions. A common approach is to represent the sources of uncertainty as stochastic variables; in this context the solution to the original differential equations becomes random. Stochastic Galerkin schemes (SGS) are used to approximate the solution to parametrized differential equations. In particular, they utilize a functional basis on the parameter to express the solution and then derive and solve a *deterministic* system of PDEs with standard numerical techniques [2]. A Galerkin method projects the randomness in a solution onto a finite-dimensional basis, making deterministic computations possible. SGS are part of a broader class known as *spectral methods*.

The most common UQ strategies involve Monte Carlo (MC) algorithms, which suffer from a slow convergence rate proportional to the inverse square root of the number of samples [5]. If each sample evaluation is expensive —

* Supported by the US Department of Energy under the PSAAP II program.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

as is often the case for the solution of a PDE — this slow convergence can make obtaining tens of thousands of samples computationally infeasible [3]. Initial spectral method applications to UQ problems showed orders-of-magnitude reductions in the costs needed to estimate statistics with comparable accuracy [8].

In the present approach for SGS, the unknown quantities are expressed as an infinite series of orthogonal polynomials in the space of the random input variable. This representation has its roots in the work of Wiener [7], who expressed a Gaussian process as an infinite series of Hermite polynomials. Ghanem and Spanos [4] truncated Wiener's representation and used the resulting finite series as a key ingredient in a stochastic finite element method. SGS based on polynomial expansions are often referred to as *polynomial chaos* approaches.

Let $\mathcal{D} = [0,1] \times T$ be a bounded subset of the spatial and time domain $\mathbb{R}_x \times \mathbb{R}_{t\geq 0}$.[3] Then let $u : \mathcal{D} \to \mathbb{R}$ be continuous and differential in its space and time components; further, let $u \in \mathcal{L}^2(\mathcal{D})$, and $u(0,t) = 0$. This u represents the solution to a differential equation,

$$\mathcal{F}(u,x,t) = 0. \tag{1}$$

Here $\mathcal{F}$ is a general differential operator that contains both spatial and temporal derivatives. Often $\mathcal{F}$ is assumed to be nonlinear.

Let $\xi : \mathbb{R} \to \mathbb{R}$ be a zero-mean, square-integrable, real random variable. We assume uncertainty is present in the initial condition $u(\,\cdot\,,0)$ and represent it by setting

$$u(x,0;\xi) : \mathbb{R}_x \to \mathbb{R} \qquad u(x,0;\xi) = f(x,\xi)$$

where f is a known function of x and ξ. Accordingly, the solution $u(x,t;\xi)$ to $\mathcal{F}(u,x,t)$ is now a random variable indexed by $(x,t) \in \mathcal{D}$, meaning $u(x,t;\xi)$ is a stochastic process.

As statistics of ξ, we require that both $u(x,t;\xi)$ and $f(x,\xi)$ have existing second moments — and in accordance with $u(0,t;\xi) = 0$, we assume $f(0,\xi) = 0$ as well.[4] Note that these are the only restrictions; namely, even though f is chosen as sinusoidal in the example of Section 2, we do not require f to be periodic, bounded over the real line, zero on the whole boundary $\partial\mathcal{D}$, etc.

We consider a polynomial chaos expansion (PCE) and *separate* the deterministic and random components of u by writing

$$u(x,t;\xi) = \sum_{k=0}^{\infty} u_k(x,t)\Psi_k(\xi).$$

The $u_k : \mathcal{D} \to \mathbb{R}$ output deterministic coefficients, while Ψ_k are orthogonal polynomials with respect to the measure $d\xi$ induced by ξ. Let $\langle\,\cdot\,,\cdot\,\rangle$ denote the

[3] For convenience we set $\mathcal{D} = [0,1] \times T$, though all of the presented results follow immediately when $\mathcal{D} = [a,b] \times T$ for some arbitrary interval $[a,b] \subset \mathbb{R}_x$.

[4] These assumptions ensure that the needed conditions for applying the Cameron-Martin theorem [1] are met.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

inner product mapping $(\phi, \psi) \mapsto \int \phi(\xi)\psi(\xi)\, d\xi$, where the triple-product notation $\langle \phi\, \psi\, \varphi \rangle$ is understood as $\langle \phi, \psi\varphi \rangle = \langle \phi\psi, \varphi \rangle = \int \phi(\xi)\psi(\xi)\varphi(\xi)\, d\xi$. Then we require the Ψ_k to satisfy the properties

$$\langle \Psi_0, \Psi_0 \rangle = \mathbb{E}_\xi(\Psi_0) = 1 \qquad \langle \Psi_i, \Psi_j \rangle = c_i \delta_{ij}$$

where $c_i \in \mathbb{R}$ are nonzero and δ_{ij} is the Kronecker delta.

By the Cameron-Martin theorem [1], the PCE of this random quantity converges in mean square,

$$\sum_{k=0}^{M} u_k(x,t)\Psi_k(\xi) \xrightarrow[\mathcal{L}^2]{M\to\infty} u(x,t;\xi).$$

This justifies the PCE and its truncation to a finite number of terms for the sake of computation. Substituting the truncation into Equation (1), we have

$$\mathcal{F}\left(\sum_{k=0}^{M} u_k(x,t)\Psi_k(\xi), x, t\right) = 0. \tag{2}$$

Furthermore, we can determine the initial conditions for the deterministic component functions. Multiplying $u(x,0;\xi)$ by any Ψ_k and integrating with respect to the ξ-measure $d\xi$ yields

$$\sum_{i=0}^{\infty} u_i(x,0) \int \Psi_i(\xi)\Psi_k(\xi)\, d\xi = \int f(x,\xi)\Psi_k(\xi)\, d\xi \;\; = \mathbb{E}_\xi[f(x,\xi)\Psi_k(\xi)]$$

$$u_k(x,0) = \frac{1}{c_k}\mathbb{E}_\xi[f(x,\xi)\Psi_k(\xi)].$$

The scalars c_k of course are dependent on the choice of polynomial Ψ_k. Similarly, we can "integrate away" the randomness in Eq. (2) by projecting onto each basis polynomial. This is discussed in detail in the next section.

2 Inviscid Burgers' equation

Our choice of orthogonal polynomials Ψ_k will rely on the distribution of the ξ random variable. Throughout this paper, we will choose $\xi \sim \mathcal{N}(0,1)$ and the Ψ_k to be Hermite polynomials; however, many of the results apply almost identically to other distributions and their corresponding polynomials.

Note that Hermite polynomials satisfy $\langle \Psi_k, \Psi_j \rangle = (k!)\delta_{kj}$ and by [6],

$$\langle \Psi_i \Psi_j \Psi_k \rangle = \begin{cases} 0 & \text{if } i+j+k \text{ is odd or } \max(i,j,k) > s \\ \frac{i!j!k!}{(s-i)!(s-j)!(s-k)!} & \text{else} \end{cases}$$

where $s = (i+j+k)/2$. Now let $u : \mathcal{D} \to \mathbb{R}$ be the solution to the inviscid Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \qquad u(x,0;\xi) = \xi \sin(x) \qquad u(0,t;\xi) = 0. \tag{3}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

The goal is to determine the solution u given the randomness in the initial conditions. Substituting in the finite PCE to (3), projecting on the polynomial basis, and integrating with respect to the ξ-measure yields

$$\sum_{i=0}^{M} \frac{\partial u_i}{\partial t} \langle \Psi_i \Psi_k \rangle + \sum_{i=0}^{M} \sum_{j=0}^{M} u_i \frac{\partial u_j}{\partial x} \langle \Psi_i \Psi_j \Psi_k \rangle = 0 \quad \text{for every } k \in \{0, \dots, M\}. \tag{4}$$

This is a system of $(M+1)$ non-linear and coupled PDEs, which we will refer to as an *M system*. Solving the original problem with randomness has now been transformed into solving a system that is completely *deterministic*.

For instance, the system for $M = 2$ is

$$\begin{aligned}
\frac{\partial u_0}{\partial t} &= -u_0 \frac{\partial u_0}{\partial x} - u_1 \frac{\partial u_1}{\partial x} - 2u_2 \frac{\partial u_2}{\partial x} \\
\frac{\partial u_1}{\partial t} &= -u_1 \frac{\partial u_0}{\partial x} - (u_0 + 2u_2) \frac{\partial u_1}{\partial x} - 2u_1 \frac{\partial u_2}{\partial x} \\
\frac{\partial u_2}{\partial t} &= -u_2 \frac{\partial u_0}{\partial x} - u_1 \frac{\partial u_1}{\partial x} - (u_0 + 4u_2) \frac{\partial u_2}{\partial x}.
\end{aligned}$$

As shown in Section 1, the initial conditions are

$$u_k(x, 0) = \frac{1}{k!} \sin(x) \int \xi \Psi_k(\xi) \, d\xi = \frac{1}{k!} \sin(x) \mathbb{E}_\xi[\xi \Psi_k(\xi)] \quad \text{for every } k \in \{0, \dots, M\}.$$

These initial conditions are easily computed. For example, when $0 \le k \le 10$ the only nonzero $u_k(x, 0)$ is $u_1(x, 0) = \sin(x)$.

For any $k \in \{0, \dots, M\}$, define the symmetric matrix $\mathbf{\Psi}_k^{(M)} \in \mathbb{R}^{M+1 \times M+1}$ such that

$$(\mathbf{\Psi}_k^{(M)})_{ij} = \langle \Psi_i \Psi_j \Psi_k \rangle \qquad \text{for } i, j \in \{0, \dots, M\}. \tag{5}$$

For the case $k = M$, we also name the particular matrix $L_M := \mathbf{\Psi}_M^M$ and let $(L_M)_{k\bullet}$ denote its kth row. When the Ψ_k are *Hermite* polynomials, we can easily prove that

$$L_M = \begin{bmatrix} 0 & \cdots & 0 & \langle \Psi_0 \Psi_M \Psi_M \rangle \\ \vdots & & \iddots & \vdots \\ 0 & \iddots & & \vdots \\ \langle \Psi_M \Psi_0 \Psi_M \rangle & \cdots & \cdots & \langle \Psi_M \Psi_M \Psi_M \rangle \end{bmatrix}.$$

Lemma 1. *Let $\mathbf{u} = (u_0, \dots, u_M)^T$ be the vector of $M+1$ functions, $\mathbf{v} = (u_0, \dots, u_{M-1})^T$, and $w = u_M$. Adopting the shorthand $(u_j)_x := \frac{\partial u_j}{\partial x}$ and $(\mathbf{u})_x := \frac{\partial \mathbf{u}}{\partial x}$, we can rewrite Equation (4) as*

$$\frac{\partial u_k}{\partial t} + \frac{1}{k!} \mathbf{u}^T \mathbf{\Psi}_k^{(M)} (\mathbf{u})_x = 0 \quad \textit{for every } k \in \{0, \dots, M\} \tag{6}$$

and when $k < M$ as

$$\frac{\partial u_k}{\partial t} + \frac{1}{k!} \mathbf{v}^T \mathbf{\Psi}_k^{(M-1)} (\mathbf{v})_x + (L_M)_{k\bullet} \frac{\partial}{\partial x} \left[w \left(\frac{\mathbf{v}}{\frac{1}{2} w} \right) \right] = 0 \tag{7}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

recalling that $(L_M)_{k\bullet}$ *is the* k*th row of* L_M*.*

Let D_M *be the diagonal* $M \times M$ *matrix such that* $D_{kk} = \frac{1}{k!}$ *and* $\widetilde{L}_M$ *be the* $M \times (M+1)$ *matrix comprised of the first* M *rows of* L_M*. Then let* $\mathbf{\Psi}^{(M-1)}$ *denote the* $M^2 \times M$ *matrix*

$$\mathbf{\Psi}^{(M-1)} := \left[\begin{array}{c} \mathbf{\Psi}_0^{(M-1)} \\ \hline \vdots \\ \hline \mathbf{\Psi}_{M-1}^{(M-1)} \end{array}\right].$$

Then (7) can be expressed in aggregate (i.e. for all $k < M$*) as*

$$\frac{\partial \mathbf{v}}{\partial t} + (D_M \otimes \mathbf{v}^T)\mathbf{\Psi}^{(M-1)}(\mathbf{v})_x + D_M \widetilde{L}_M \frac{\partial}{\partial x}\left[w\left(\frac{\mathbf{v}}{\frac{1}{2}(w)}\right)\right] = \mathbf{0} \tag{8}$$

where $\otimes$ *is the Kronecker product.*

Proof. See Appendix.

Equations (6) and (7), along with the last term in the LHS of Equation (8), will be used to inform an algorithm to solve the M system using the solution to the $M-1$ system. This is described in the following section.

3 Bluff-and-fix (BNF) algorithm

We will use the superscript $\mathbf{u}^{(M)}$ to denote the $(M+1)\times 1$ vector of functions that is the solution to the M system. Similarly, $u_k^{(M)}$ is the kth component function of $\mathbf{u}^{(M)}$. When a component function $u_k^{(M)}(x,t)$ is written without the superscript i.e. as $u_k(x,t)$, the value of M is considered to be a fixed but arbitrary positive integer.

Suppose we have computed $\boldsymbol{u}^{(M-1)}$. How can we incorporate this information into solving for $\boldsymbol{u}^{(M)}$? Firstly, the M system has the added equation for $\frac{\partial u_M}{\partial t}$, namely

$$\frac{\partial u_M^{(M)}}{\partial t} = -\frac{1}{M!}\sum_{i=0}^{M}\sum_{j=0}^{M}\langle \Psi_i \Psi_j \Psi_M \rangle u_i^{(M)} \frac{\partial u_j^{(M)}}{\partial x}. \tag{9}$$

If we had the first M coefficients $u_0^{(M)}, \ldots, u_{M-1}^{(M)}$ of $\boldsymbol{u}^{(M)}$ corresponding to the M system rather than to the $M-1$ system, we could substitute directly into the RHS of (9) and simply integrate the remaining equations. However, the numerical solutions $u_0^{(M-1)}, \ldots, u_{M-1}^{(M-1)}$ will differ from $u_0^{(M)}, \ldots, u_{M-1}^{(M)}$. To see why, recall Equation (6) in Lemma 1 and its notation to observe that $u_k^{(M-1)}$ is the solution to

$$\frac{\partial v_k}{\partial t} = -\frac{1}{k!}\mathbf{v}^T \mathbf{\Psi}_k^{(M-1)}(\mathbf{v})_x := F_{M-1}(\mathbf{v}, w) \tag{10}$$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

and by (7), $u_k^{(M)}$ for $k < M$ is the solution to

$$\frac{\partial v_k}{\partial t} = -\frac{1}{k!}\left(\mathbf{v}^T \mathbf{\Psi}_k^{(M-1)}(\mathbf{v})_x + (L_M)_{k\bullet}\frac{\partial}{\partial x}\left[w\left(\frac{\mathbf{v}}{\frac{1}{2}w}\right)\right]\right) := F_M(\mathbf{v}, w). \quad (11)$$

Thus, the numerical solutions $u_k^{(M-1)}$ and $u_k^{(M)}$ are different, because the right hand sides of (10) and (11) differ by the *function*

$$G_k(\mathbf{v}, w) := -\frac{1}{k!}(L_M)_{k\bullet}\frac{\partial}{\partial x}\left[w\left(\frac{\mathbf{v}}{\frac{1}{2}w}\right)\right]. \quad (12)$$

We can write all of the G_k functions together via the third term on the LHS of (8); that is,

$$\mathbf{G} := (G_0, \ldots, G_{M-1})^T \text{ so that } \mathbf{G}(\mathbf{v}, w) = -D_M \widetilde{L}_M \frac{\partial}{\partial x}\left[w\left(\frac{\mathbf{v}}{\frac{1}{2}(w)}\right)\right]. \quad (13)$$

3.1 One step bluff-and-fix

From (12), we know there is a discrepancy between $u_k^{(M-1)}$ and $u_k^{(M)}$ for $k < M$. Regardless, we can *bluff* and take the solutions we have $u_0^{(M-1)}, \ldots, u_{M-1}^{(M-1)}$ to solve for some approximation of $u_M^{(M)}$, which we can call $\widehat{u}_M^{(M)}$, and then back-substitute to solve the previous $(M-1)$ equations in the M system for $u_0^{(M)}, \ldots, u_{M-1}^{(M)}$ using $\widehat{u}_M^{(M)}$. This approach is potentially more efficient than calculating the solution of the M system directly via classic monolithic methods. However, we will opt for an algorithm with even less computation time.

A workable strategy is based on a similar idea. Instead of re-computing $u_0^{(M)}, \ldots, u_{M-1}^{(M)}$ after obtaining $\widehat{u}_M^{(M)}$, we re-solve for the *least accurate* $u_k^{(M)}$ at the same time as solving for $\widehat{u}_k^{(M)}$. That is, we only correct the $\widehat{u}_k^{(M)}$ that we believe will be the worst approximations of their corresponding $u_k^{(M)}$. The $\mathcal{I}$ in Algorithm 1 is the collection of *correction indices* i.e. the indices k denoting which $u_k^{(M)}$ are corrected.

An algorithm to use the solution to an $M-1$ system to solve an M system.

Algorithm 1: one step bluff-and-fix(c, $\mathbf{u}^{(M-1)}$)

input:
- correction size $c \in \{1, \ldots, M\}$
- $\mathbf{u}^{(M-1)}$ obtained by standard monolithic methods

---

select correction indices $\mathcal{I} \subseteq \{0, \ldots, M-1\}$ such that $|\mathcal{I}| = c - 1$
$\mathcal{I} \leftarrow \mathcal{I} \cup \{M\}$
set approximate solutions to the M system equal to those of the $(M-1)$ system (**bluff**), if those solutions are not getting corrected:

$$\widehat{u}_k^{(M)} \leftarrow u_k^{(M-1)} \text{ for all } k \in \{0, \ldots, M\} \setminus \mathcal{I}$$

solve (fix) the coupled system $\{\widehat{u}_k^{(M)}\}_{k\in\mathcal{I}}$ of size c to obtain $\widehat{\mathbf{u}}^{(M)}$

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

Since the correction size c must be at least one, $M \in \mathcal{I}$ always. Here the language is less intuitive; a former approximation of $u_M^{(M)}$ is not being "corrected" per say, since $\widehat{u}_M^{(M)}$ is being determined for the first time — but when $c > 1$, solving for $\widehat{u}_M^{(M)}$ is occurring at the same time as some other $\widehat{u}_k^{(M)}$ are being fixed.

Algorithm 1 poses two immediate questions.

1. How large should we make the correction size $c \in \{1, \ldots, M\}$?
2. How should we choose the correction indices $\mathcal{I}$? That is, how do we pick which $\widehat{u}_k^{(M)}$ approximations should be fixed?

To address question 1, note that *if $c = M$, then the one step bluff-and-fix (BNF) is equivalent to solving the entire M system directly.* In this case, we use $\mathbf{u}^{(M-1)}$ to set *none* of the approximations $\widehat{u}_k^{(M)}$ (no bluffing), and the entire coupled system $\{\widehat{u}^{(M)}\}_{k=0}^{M}$ of $M+1$ equations is solved by standard numeric techniques.

The hypothesis is that by choosing which $\widehat{u}_k^{(M)}$ to correct judiciously, we can still well-approximate $\mathbf{u}^{(M)}$ when $c < M$. This brings us to second posed question. To determine the correction indices $\mathcal{I}$, we target the $u_k^{(M)}$ such that $u_k^{(M-1)}$ and $u_k^{(M)}$ are very different so that the approximation $\widehat{u}_k^{(M)} = u_k^{(M-1)}$ is a poor one). From (12), we know the difference between the numeric solutions $u_k^{(M)}$ and $u_k^{(M-1)}$ arises from the function G_k, so we would like $\mathcal{I}$ to ideally include $k^* = \arg\max_{k\in\{0,\ldots M-1\}} \|G_k\|$, where $\|\cdot\|$ denotes some function norm over $(x,t) \in \mathcal{D}$. From the definition of G_k, we do not know what values its input functions $(\mathbf{v}, w)$ or their derivatives will take over $(x,t) \in \mathcal{D}$; however, we *do* know the entries of the matrix L_M.

Now there is some choice. The function G_k is the difference between F_M and F_{M-1} from Equations (9) and (10), and all three of these equations are scaled by the $\frac{1}{k!}$ factor. We can keep this $\frac{1}{k!}$ factor and select the G_k that is large in an "absolute error" sense. Alternatively, we can ignore this scaling and choose the $\widehat{u}_k^{(M)}$ to fix such that the difference function G_k is significant *relative* to its corresponding F_M and F_{M-1}.

The former approach (call it the *absolute version*) targets $k_1^* = \max_{k\in\{0,\ldots,M-1\}} \frac{1}{k!}\|(\widetilde{L}_M)_{k\bullet}\|_\infty = \max_{k\in\{0,\ldots,M-1\}} \frac{1}{k!}\sum_{j=0}^{M} |(\widetilde{L}_M)_{k,j}|$. Equivalently, by Equation (13), this is selecting $k_1^* = \arg\max_{0\le k\le M-1} \sum_{j=1}^{M} |(D_M\widetilde{L}_M)_{kj}|$ i.e. the k_1^* indexing the row of $D_M\widetilde{L}_M$ with largest absolute row sum — where we recall that $\widetilde{L}_M$ is matrix of the first M rows of L_M. The latter approach (call it the *relative version*) simply picks the row indexing the largest row sum in $\widetilde{L}_M$ itself (i.e. not in $D_M\widetilde{L}_M$).

Selecting the row of $\widetilde{L}_M$ with maximal absolute row sum is simple; it is straight-forward to verify that row $M-1$ obtains the maximum (though may not do so uniquely) and that the row sums of $\widetilde{L}_M$ are non-decreasing as the row index k increases. The structure of the $D_L\widetilde{L}_M$ matrix does not lend itself to as obvious of a pattern.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

We opt for the *relative* version when constructing our algorithm, since its numeric results are overwhelmingly promising (as discussed in Section 4) and its implementation avoids an additional row sorting step; however, this is a potential area for future investigation. Since we require $|\mathcal{I}| = c$ for a given correction size parameter c, we simply pick the indices corresponding to the last c rows in the M system i.e. $\mathcal{I} = \{M - c + 1, \ldots, M\}$.

3.2 Iterative bluff-and-fix

An assumption of the one step bluff-and-fix algorithm is that you already have solved the fully coupled $M - 1$ system via some explicit time-stepping scheme (e.g. as fourth-order Runge Kutta). Realistically, we likely only want to solve a fully coupled M_0 system for when M_0 is small (say 2 or 3). How can we then use this information for approximating $\mathbf{u}^{(M)}$ for a larger $M > M_0$?

An algorithm to use the solution to an M_0 system to solve an M system for general $M > M_0$.

Algorithm 2: iterative bluff-and-fix$(c, \mathbf{u}^{(M_0)})$

input:
- correction size $c \in \{1, \ldots, M\}$
- $\mathbf{u}^{(M_0)}$ obtained by standard monolithic methods

---

$\widehat{\mathbf{u}}^{(M_0)} \leftarrow \mathbf{u}^{(M_0)}$

for $m = M_0 + 1, \ldots, M$ **do**

select correction indices $\mathcal{I} \subseteq \{0, \ldots, m-1\}$ such that $|\mathcal{I}| = c - 1$

$\mathcal{I} \leftarrow \mathcal{I} \cup \{m\}$

(bluff) set approximate solutions to the m system equal to those of the $(m-1)$ system, if those solutions are not getting corrected:

$$\widehat{u}_k^{(m)} \leftarrow u_k^{(m-1)} \text{ for all } k \in \{0, \ldots, m\} \setminus \mathcal{I}$$

(fix) solve the coupled system $\{\widehat{u}_k^{(m)}\}_{k \in \mathcal{I}}$ to obtain $\widehat{\mathbf{u}}^{(m)}$

end

The iterative bluff-and-fix algorithm uses some baseline $\mathbf{u}^{(M_0)}$ solution to get an approximation $\widehat{\mathbf{u}}^{(M_0+1)}$ of $\mathbf{u}^{(M_0+1)}$. The approximation $\widehat{\mathbf{u}}^{(M_0+1)}$ is then re-fed into the one step bluff-and-fix algorithm, instead of the "true" $\mathbf{u}^{(M_0+1)}$, to determine $\widehat{\mathbf{u}}^{(M_0+2)}$, and the process continues.

4 Numerical results

We report solutions for Burgers' equation with uncertain initial conditions, which are $u(x, 0; \xi) = \xi \sin(x)$ for $\xi \sim \mathcal{N}(0, 1)$. The equation is solved for $x \in [0, 3]$ on a uniform grid with $\Delta_x = 0.05$. Time integration is based on the Runge-Kutta 4-step (RK4) scheme with $\Delta_t = 0.001$.

Throughout this discussion, we will define *error* as the deviation of the solution approximated $\widehat{\mathbf{u}}^{(M)}$ produced by bluff-and-fix from the solution yielded from solving the full M system via RK4. *That is, our computations are not being*

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

compared with a true analytic solution but instead with the numerical solution from a standard monolithic method. In particular,

$$\text{absolute error in } \widehat{u}_k(\mathbf{x},\mathbf{t}) := \|\widehat{u}_k^{\,\mathrm{BNF}}(\mathbf{x},\mathbf{t}) - u_k^{\mathrm{RK4}}(\mathbf{x},\mathbf{t})\|_2 \tag{14}$$

$$\text{relative error in } \widehat{u}_k(\mathbf{x},\mathbf{t}) := \frac{\|\widehat{u}_k^{\,\mathrm{BNF}}(\mathbf{x},\mathbf{t}) - u_k^{\mathrm{RK4}}(\mathbf{x},\mathbf{t})\|_2}{\|u_k^{\mathrm{RK4}}(\mathbf{x},\mathbf{t})\|_2} \tag{15}$$

In (14) and (15), $u_k(\mathbf{x},\mathbf{t})$ is the matrix $(u_k(x_i,t_j))_{i,j}$ of u_k evaluated on the uniform grid of position $\mathbf{x}$ and time $\mathbf{t}$ points. The $\|\cdot\|_2$ denotes the matrix 2-norm.

When reporting the *average* absolute (resp. relative) error for an approximation $\widehat{\mathbf{u}}^{\,\mathrm{BNF}}$ for an M system, we mean taking the average of the absolute (resp. relative) errors of each $\widehat{u}_k^{\,\mathrm{BNF}}$ over all k indices.

4.1 Results for one step version

We compare the solution obtained by solving the full M system via RK4 against the computations obtained by inputting the $M-1$ solution $\mathbf{u}^{(M-1)}$ into the one step BNF algorithm. We test

1. correction size $c = 1$ (only $\widehat{u}_M^{(M)}$ is computed per step),
2. correction size $c = 2$ ($\widehat{u}_M^{(M)}$ is computed and $\widehat{u}_{M-1}^{(M)}$ is fixed per step), and
3. correction size $c = 3$ ($\widehat{u}_M^{(M)}$ is computed and both $\widehat{u}_{M-2}^{(M)}$ and $\widehat{u}_{M-1}^{(M)}$ are fixed per step).

The results are shown in Table 1.

Picking a small correction size (say one or two) can be sufficient for producing accurate solution approximations. For instance, using the solution $\mathbf{u}^{(4)}$ to produce $\widehat{\mathbf{u}}^{(5)}$ has average absolute error of under 1% for all c, around 4% average relative error for $c = 1$, and under 1% average relative error for $c = 2, 3$, as shown in the $M = 5$ row of Table 1. Figure 1 displays how the approximate solution $\widehat{u}_5^{(5)}$ in this system converges to the RK4 solution as the correction size c increases from 1 to 2. Similarly, using the solution $\mathbf{u}^{(6)}$ to produce $\widehat{\mathbf{u}}^{(7)}$ has average absolute error of under 1% for all c values, around 6% average relative error for $c = 1$, and about 2.5% average relative error for $c = 2$, as shown in the $M = 7$ row of Table 2.

Table 1: Average absolute & relative errors for correction sizes $c \in \{1, 2, 3\}$ in **one step** bluff-and-fix for $M = 3, \ldots, 8$.

M	Avg. Abs. Error			Avg. Rel. Error		
	$c=1$	$c=2$	$c=3$	$c=1$	$c=2$	$c=3$
3	0.1515	0.05593	0.01114	0.03196	0.004364	0.001683
4	0.03481	0.01823	0.007423	0.03711	0.005768	0.0008361
5	0.009536	0.006371	0.003599	0.04300	0.009232	0.001656
6	0.003279	0.002619	0.001859	0.05112	0.01530	0.003765
7	0.001330	0.001195	0.0009746	0.05986	0.02541	0.008166
8	0.0006404	0.0006118	0.0005623	0.06080	0.02630	0.01116

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

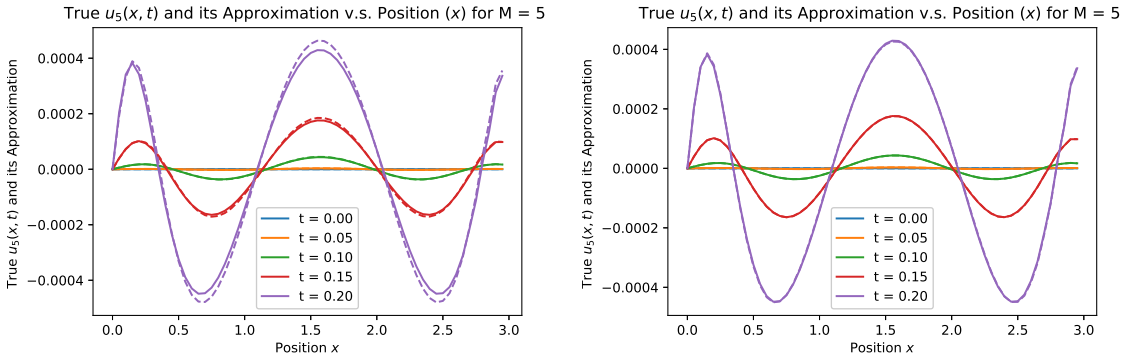

Fig. 1: The standard ("true") RK4 solution $u_5^{(5)}$ (solid lines) against the $\widehat{u}_5^{(4)}$ produced by one step bluff-and-fix (dashed lines) at time points $t = 0, 0.05, 0.10, 0.15, 0.20$ seconds for $0 \leq x \leq 3$. LHS figure shows $\widehat{u}_5^{(5)}$ for $c = 1$, and RHS figure shows $\widehat{u}_5^{(5)}$ improvement for $c = 2$.

What if average or relative error can be further reduced by choosing a different subset of $\widehat{u}_k^{(M)}$ to fix? To evaluate how well the correction indices $\mathcal{I}$ were selected, we can examine the absolute and relative error in the approximation $\widehat{u}_k^M = u_k^{(M-1)}$ for all k over various M values. Then we can see whether the least accurate $\widehat{u}_k^{(M)}$ for every M were appropriately chosen for correction. These results are displayed in Table 2.

Table 2: Selection of correction indices $\mathcal{I}$ ranked by priority in one step bluff-and-fix against the ideal $\widehat{u}_k^{(M)}$ to correct (in this numeric example) for $M \in \{3, \ldots, 8\}$. Matching indices are shown in blue.

M	Ranking of $k < M$ by Numeric Error (Descending) in $\widehat{u}_k^{(M)} = u_k^{(M-1)}$		Order of Selection in $\mathcal{I}$
	Absolute Error	Relative Error	
3	2, 1, 0	2, 0, 1	2, 1, 0
4	3, 2, 1, 0	3, 2, 0, 1	3, 2, 1, 0
5	4, 3, 2, 1, 0	4, 3, 2, 0, 1	4, 3, 2, 1, 0
6	3, 4, 5, 2, 1, 0	5, 4, 3, 2, 0, 1	5, 4, 3, 2, 1, 0
7	3, 4, 5, 2, 6, 1, 0	6, 5, 4, 3, 2, 0, 1	6, 5, 4, 3, 2, 1, 0
8	3, 4, 2, 5, 1, 6, 7, 0	7, 6, 5, 4, 3, 2, 0, 1	7, 6, 5, 4, 3, 2, 1, 0

We see from Table 2 that one step bluff-and-fix is often spot-on for guessing which approximations $\widehat{u}_k^{(M)}$ are least accurate. When defining the "worst" approximation by relative error, BNF always selects correctly for $c < M - 1$ — and while the ideal indices when $c = M - 1$ or $c = M$ are not all chosen, this is likely no issue, as in practice a correction size that large would not be used. (Recall that $c = M$ is equivalent to using regular RK4.) Also, it should be noted that bluff-and-fix can *still* produce an accurate solution approximation when

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

an "incorrect" index is chosen — it just might not be the best approximation *possible* given that value of c.

In addition, we test the computational cost of one step and iterative bluff-and-fix for different correction sizes and M values. The runtimes are measured using the `%timeit` command in `iPython` with parameters `-r 10 -n 10` to obtain an average with standard deviation over 100 realizations. All computations are performed on a machine with a 1.8 GHz Intel Dual-Core processor. The results are displayed in Table 3.

From Table 3, we observe that bluff-and-fix *consistently* requires less computation time than the standard Runge-Kutta 4-step. That being said, recall that one step bluff-and-fix assumes that the *solution to the smaller system is readily available*. Any additional time that was spent to obtain this $M-1$ system solution $\mathbf{u}^{(M-1)}$ is not accounted for when obtaining runtime measurements — and when M is large, this additional time is not trivial. Such concerns are addressed when timing the iterative version of the algorithm in the following section, which only assumes that we know $\mathbf{u}^{(M_0)}$ for some small M_0 value.

Table 3: Runtimes of **one step** bluff-and-fix to approximate $\mathbf{u}^{(M)}$ when given $\mathbf{u}^{(M-1)}$ compared with runtimes of solving full M systems via RK4. Tested over $M = 3, \ldots, 8$ with correction sizes $c = 2, 3$. Each time measurement is averaged over 100 loops to provide a confidence interval.

M	Avg. One Step BNF Runtime		Avg. RK4 Runtime
	$c = 2$	$c = 3$	
3	146 ms ± 7.81 ms	175 ms ± 14.1 ms	181 ms ± 7.81 ms
4	162 ms ± 13.2 ms	183 ms ± 22.5 ms	222 ms ± 22.4 ms
5	167 ms ± 21.3 ms	184 ms ± 4.42 ms	260 ms ± 15.2 ms
6	167 ms ± 5.42 ms	189 ms ± 3.96 ms	315 ms ± 37.6 ms
7	191 ms ± 21.1 ms	202 ms ± 6.01 ms	336 ms ± 13.1 ms
8	216 ms ± 31.5 ms	217 ms ± 5.38 ms	377 ms ± 3.77 ms

4.2 Results for iterative version

Now we present results for Algorithm 2 when using the baseline solution $\mathbf{u}^{(M_0)}$ for $M_0 = 2$ to approximate solutions to the M systems for $M = 3, \ldots, 8$. Correction sizes $c = 1, 2, 3$ are all tested. The results are displayed in Table 4.

We observe how quickly the solution approximation from iterative bluff-and-fix converges to the "true" RK4 solution as the correction size is increased. For example, the average relative error in $\widehat{\mathbf{u}}^{(7)}$ drops from $\sim 27\%$ ($c = 1$) to $\sim 5.7\%$ ($c = 2$) to $\sim 1.4\%$ ($c = 3$), and the average absolute error falls from $\sim 8.7\%$ to $\sim 3.5\%$ to $\sim 1\%$.

Furthermore, choosing a small correction size, such as $c = 2$, is highly accurate even when M is as large as 8, with average relative error in this case always below 8.5% and average absolute error always below 6%. Future work will be focused on testing the algorithm for larger values of M to assess whether such a small c value can maintain these promising results.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

Table 4: Average absolute & relative errors in **iterative** bluff-and-fix when using the baseline solution $M_0 = 2$ to approximate solutions to the M systems for $M = 3, \ldots, 8$. Results for correction sizes $c = 1, 2, 3$ are shown.

M	Avg. Abs. Error			Avg. Rel. Error		
	$c = 1$	$c = 2$	$c = 3$	$c = 1$	$c = 2$	$c = 3$
3	0.1515	0.05593	0.01115	0.03196	0.004363	0.001682
4	0.1337	0.05301	0.01420	0.07467	0.01019	0.001924
5	0.1145	0.04634	0.01357	0.1313	0.02034	0.003412
6	0.09894	0.04033	0.01227	0.2013	0.03642	0.006904
7	0.08676	0.03547	0.01096	0.2740	0.05793	0.01400
8	0.07711	0.03147	0.009704	0.3586	0.08464	0.02020

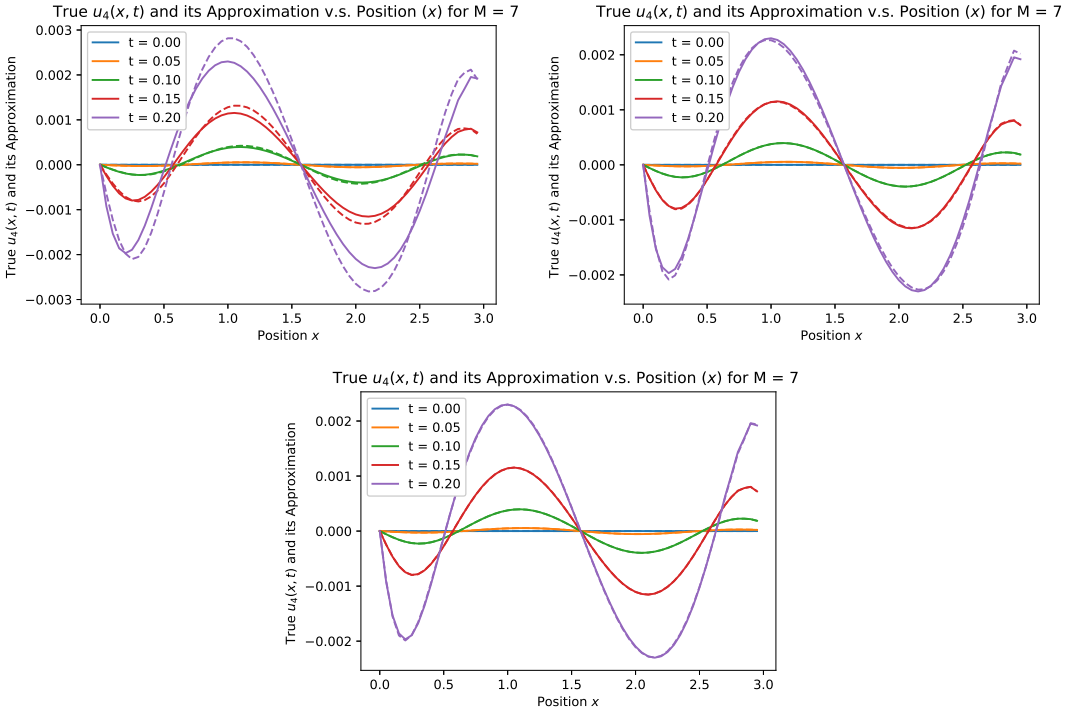

Fig. 2: The standard ("true") RK4 solution $u_4^{(8)}$ (solid lines) against the $\widehat{u}_4^{(8)}$ produced by iterative bluff-and-fix (dashed lines) from a baseline $M_0 = 2$ solution. Solved over a uniform grid with $0 \leq x \leq 3$ and $0 \leq t \leq 0.25$, with solutions at times $t = 0, 0.05, 0.10, 0.15, 0.20$ displayed. We can see how the approximation $\widehat{u}_4^{(8)}$ converges to the RK4 solution as the correction size is increases from one (top left figure) to two (top right figure) to three (bottom figure).

As before, we test the computational cost via `%timeit` in `iPython` for different correction sizes and M values, averaging each measurement over 100 realizations to provide a confidence interval. The computation time of iterative bluff-and-fix is on average 853 ms $\pm$ 39.5 ms, 1.12 s $\pm$ 121 ms, and 1.28 s $\pm$ 63.5 ms for

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

$c = 1, 2, 3$, respectively. We can see that the added cost from correcting $\widehat{u}_{M-1}^{(M)}$, as opposed to just solving for $\widehat{u}_M^{(M)}$, is on average only 267 milliseconds. This suggests that the reduction in error between $c = 1$ and $c = 2$ potentially comes at a cheap cost, especially given that average absolute error and relative error is under 6% and 8.5% (respectively) after this transition.

Using RK4 to *only* solve the $M = 8$ system is faster than using iterative bluff-and-fix from the baseline $M_0 = 2$ solution: the former spends 426 ms $\pm$ 48.6 ms per loop, as opposed to around 850 milliseconds. However, it is possible that for larger M values, the iterative bluff-and-fix algorithm will eventually out-perform RK4 in terms of runtime. This is an area for future investigation.

Furthermore, iterative bluff-and-fix has the advantage of producing approximate solutions to all of the M systems for $M = 3, \ldots, 8$ along the way. When repeatedly solving the full M system via RK4 for $M = 3, \ldots, 8$, the average runtime is 2.05 s $\pm$ 145 ms per loop — meaning bluff-and-fix with correction sizes $c = 1$ (averaged at 853 milliseconds), $c = 2$ (averaged at 1.21 seconds), and $c = 3$ (averaged at 1.28 seconds) is *far* more efficient for this type of goal.

5 Conclusion

Polynomial chaos (PC) methods are effective for incorporating and quantifying uncertainties in problems governed by partial differential equations. In this paper, we present a promising algorithm (one step bluff-and-fix) for utilizing the solution to a polynomial chaos M system arising from inviscid Burgers' equation to approximate the solution to the corresponding $(M+1)$ system. We expand the algorithm to an iterative version, which utilizes the solution to an M_0 system to approximate the solution to an M system for a general $M > M_0$. Bluff-and-fix is shown to be effective in producing accurate approximations, even when its correction size parameter is small, for both its one step and iterative versions. In the one step version, these approximations are produced more efficiently than doing so with a standard monolithic numeric scheme. While iterative bluff-and-fix from some baseline M_0 can be less efficient than solving the full M system directly, it has the advantage of producing approximations to all of the m systems along the way for $M_0 \leq m \leq M$ — and does so faster than the monolithic method solves all of the full m systems one by one. In general, it could be beneficial to know the solution to an M system for a consecutive range of M values, because then one could observe when the difference between consecutive system solutions is small, which provides a rough sense of the M value sufficient for solution convergence.

Future work will be focused on generalizing and testing the algorithm on other nonlinear PDEs with uncertain initial conditions. We also plan to investigate different choices of the uncertainty representation ξ. It is expected that the process for selecting which solutions to "fix" will need to be generalized, since those choices rely entirely on the structure of the L_M matrix, which depends on the choice of orthogonal polynomial.

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55

6 Appendix

Proof of Lemma 1. The computation for (6) is straight-forward. To prove (7) write $\sum_{i=0}^{M}\sum_{j=0}^{M} u_i \frac{\partial u_j}{\partial x}\langle \Psi_i\Psi_j\Psi_k\rangle$ as

$$\begin{aligned}
&\sum_{i,j=0}^{M-1}\langle\Psi_i\Psi_j\Psi_k\rangle u_i(u_j)_x + \sum_{j=0}^{M-1}\langle\Psi_M\Psi_j\Psi_k\rangle[u_M u_j]_x + \langle\Psi_M^2\Psi_k\rangle u_M(u_M)_x \\
&= \sum_{i,j=0}^{M-1}(\boldsymbol{\Psi}_k^{(M-1)})_{ij}u_i(u_j)_x + \sum_{j=0}^{M-1}(L_M)_{k,j}\left[u_M u_j\right]_x + (L_M)_{k,M}\left[\tfrac{1}{2}u_M(u_M)\right]_x \\
&= \mathbf{v}^T\boldsymbol{\Psi}_k^{(M-1)}(\mathbf{v})_x + (L_M)_{k\bullet}\frac{\partial}{\partial x}\left[w\left(\frac{\mathbf{v}}{\frac{1}{2}w}\right)\right].
\end{aligned}$$

where the chain rule and symmetry of the triple-product were applied in the first equality. Finally, note that the first and third terms on the LHS of (7) in aggregate are clearly $\frac{\partial \mathbf{v}}{\partial t}$ and $D_M\widetilde{L}_M\frac{\partial}{\partial x}\left[w\left(\frac{\mathbf{v}}{\frac{1}{2}w}\right)\right]$, respectively. By definition of the Kronecker product, $D_M\otimes\mathbf{v}^T$ is the $M\times M^2$ matrix

$$D_M\otimes\mathbf{v}^T = \begin{bmatrix} \frac{1}{(0)!}\mathbf{v}^T & & \\ & \ddots & \\ & & \frac{1}{(M-1)!}\mathbf{v}^T \end{bmatrix}$$

and so $(D_M\otimes\mathbf{v}^T)\boldsymbol{\Psi}^{(M-1)}$ is the $M\times M$ matrix whose kth row is $\frac{1}{k!}\mathbf{v}^T\boldsymbol{\Psi}_k^{(M-1)}$. Thus, $(D_M\otimes\mathbf{v}^T)\boldsymbol{\Psi}^{(M-1)}(\mathbf{v})_x$ is the desired $M\times 1$ vector whose kth entry is $\frac{1}{k!}\mathbf{v}^T\boldsymbol{\Psi}_k^{(M-1)}(\mathbf{v})_x$, completing the proof of (8).

References

1. Cameron, R., Martin, W.: The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals. *Annals of Mathematics.* **48**, 385-392 (1947).
2. Constantine, P.: A primer on stochastic Galerkin methods. Academic homepage, `https://www.cs.colorado.edu/~paco3637/docs/constantine2007primer.pdf`. Last accessed 6 Feb 2020
3. Constantine, P.: Spectral methods for parametrized matrix equations. Ph.D. thesis, Stanford University (2009)
4. Ghanem, R., Spanos, P.: Stochastic finite elements: a spectral approach. Springer-Verlag, New York (1991)
5. Owen, A.: Monte Carlo theory, methods and examples, Ch. 1. Academic homepage, `https://statweb.stanford.edu/~owen/mc/`. Last accessed 15 April 2020
6. Szegö, G: Orthogonal Polynomials, 2nd Edition. American Mathematical Society, New York (1959)
7. Wiener, N.: The homogenous chaos. *Amer. J. Math.* **60**(4), 897–936 (1938)
8. Xiu, D., Karniadakis, G.: The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM J. Sci. Comput.* **24**, 619-644 (2002)

ICCS Camera Ready Version 2020
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-50436-6_55