Towards a particle-flow framework for uncertainty quantification, with applications in wind plant system dynamics and control

Towards a particle-flow framework for uncertainty quantification, with applications in wind plant system dynamics and control

Karl O. Merz
SINTEF Energy Research, Sem Sælands vei 11, 7034 Trondheim, Norway
E-mail: karl.merz@sintef.no

Abstract. The method of particle flow, originally developed for solving Bayes' formula, is extended to provide a general transformation between two probability distributions. It is shown that this can enable the use of a chaos expansion for uncertain or stochastic dynamic systems. The approach is demonstrated on a simple example. The method is potentially relevant for the real-time control of wind plants. For example, it could be used to obtain a probabilistic estimate of the wind field inside a wind farm using a combination of measurements from the turbines and modelling. Time lags and wake effects make this problem non-Gaussian, which the particle-flow method is well-suited to handle. It remains to be seen, however, whether there is a compelling reason to use a chaos expansion for stochastic dynamic analysis. Functions implementing the methods have been programmed in the Julia language.

1. Introduction

The dynamics of a wind power plant are stochastic and uncertain. Turbulent winds, including wakes from upstream turbines, and ocean waves are the dominant stochastic effects, while significant uncertanties are present in the aerodynamics; yaw sensor measurements; soil, structural, and hydrodynamic damping; temperature-dependent electrical properties; and other system parameters. This calls for explicit consideration of probabilities, or uncertainty, when making decisions regarding the design, analysis, or control of wind power plants.

The difficulty with probabilistic analysis is that it requires a global solution of the system dynamics. To be explicit, say that we have a discrete-time dynamic system

$$x^k = f(x^{k-1}, u^k, \alpha^k), \tag{1}$$

where the superscript represents the timestep, x is a vector of state variables, including uncertain parameters; u is a vector of inputs; and α represents the remaining deterministic system parameters. Then the joint probability density function $\varphi(x^{k-1}, u^k)$ is required for a complete characterization.[1] In principle, this density function spans the entire domain of the combined state and input variables: it is a global solution of the dynamics, akin to the attractor in the theory of nonlinear dynamic systems.

[1] To the extent that the inputs u are correlated over time, this effect can be represented through an appropriate definition of the state variables.

In order to make the probabilistic analysis of (1) tractable, we often take shortcuts. One common approach is to assume that $\varphi(x^{k-1}, u^k)$ is Gaussian, and so fully described by a mean vector and covariance matrix. Though often useful, the Gaussian assumption can also be catastrophically wrong. To pick one relevant example, say that the wind turbine controller has a speed exclusion zone, where the rotor speed is controlled so as to avoid a frequency band coinciding with a tower resonant mode. Near this operating regime, many of the important system response variables will have a bimodal probability distribution, one peak corresponding to the upper rotor speed and another to the lower rotor speed. This cannot be represented as Gaussian, and an attempt to obtain a Gaussian maximum likelihood estimate would produce a result in between the peaks, which was in reality highly improbable.

Monte Carlo methods allow marginal distributions to be constructed, without formulating $\varphi(x^{k-1}, u^k)$ explicitly. For instance, one might formulate the distribution $\varphi(y^k)$ of a scalar output

$$y^k = g(x^k, u^k, \alpha^k) \tag{2}$$

by repeated sampling of x^{k-1} and u^k in (1). Random or quasi-random Monte Carlo sampling is the preferred approach for high-dimensional problems. The technique is used as a building block in other methods, including Gaussian approaches like the ensemble Kalman filter (Evensen 2009), as well as the non-Gaussian approaches discussed below. Alone, the cloud of particles (sampled points in the state-input space) obtained by a Monte Carlo approach is not so useful: it needs to be combined with some other framework for the representation of probability, typically involving estimates of basis-function coefficients[2] through integrals over the domain.

We wish to have a framework that can represent general probability density functions of the type $\varphi(z)$, where $z = (x, u)$, in a compact-yet-useful way. This requires that we have a way to store the probability, and a way to operate on it repeatedly by passing it through functions like (1), applying Bayes' formula when new information becomes available. Several options exist. One may, for instance, apply a smoothing kernel to a cloud of particles (Mack and Rosenblatt 1979) in order to estimate the probability density at any point; but there are better approaches. The UQLab software (Lataniotis *et al.* 2021), for instance, decomposes a density function using a copula. The decomposition consists of independent marginal distributions along each coordinate axis of the z space, together with a special type of normalized multivariate distribution, the copula, that contains the dependency information. The key idea is that for many real-world processes the dependency can be reasonably approximated by a class of distributions that is easy to manipulate, a multivariate Gaussian being a simple example, and a type of tree data structure (vine copulas) being a more complicated example. In the general case, it can be difficult to formulate the copula accurately, given a set of sampled particles as a starting point (Merz 2021), and we do not apply the decomposition in the present work.

A chaos expansion is a convenient and flexible way of representing a probability density function. Uncertainty quantification software packages such as UQLab (Marelli *et al.* 2021), Dakota (Dalbey *et al.* 2020), and Chaospy (Feinberg and Langtangen 2015) implement the chaos expansion framework.[3] The idea behind a chaos expansion is that the probability is stored within a normalized reference space ξ, in a form that is easy to manipulate, such as a unit Gaussian distribution $\varphi(\xi) = \mathcal{N}(0, I)$. Then a function $z(\xi)$ is established, mapping the reference space ξ to the physical space z, in such a way that $\varphi(z)$ is recovered. The hope, often realized in practice, is that the function $z(\xi)$ is smooth and simple, such that it can be represented by a small number of basis functions such as low-order polynomials. Engineering approaches such as the Kalman filter, with its Gaussian assumption, can be expressed in terms

[2] This includes the mean and covariance in the Gaussian case, or more elaborate basis functions in the non-Gaussian case. Binning is an extremely crude example of a general set of basis functions.

[3] A chaos expansion may be combined with a copula decomposition, but this is not a requirement.

of low-order chaos expansions; turning it around, a chaos expansion can be thought of as an extendable and adaptable version of the typical, simplified methods of probabilistic analysis.

Adopting the chaos expansion framework, we look in Section 2 at some of its strengths and shortcomings, when applied to stochastic dynamic systems. We find that the key to success is a means to quickly find a suitable mapping $z(\xi)$, given a known density function $\varphi(\xi)$ and a sampled particle cloud representing $\varphi(z)$. The established approach for such operations, the Rosenblatt transformation (Rosenblatt 1952, Feinberg and Langangen 2015), is shown to have serious disadvantages as a numerical method. From the recent literature on particle filters (Daum and Huang 2007), we borrow the concept of particle flow (Section 3). It is shown how the problem of finding $z(\xi)$ can be formulated in terms of particle flow, improving upon the numerical properties of the Rosenblatt transformation. Section 4 then suggests some potential applications of the method in wind plant system dynamics and control.

2. Chaos expansion

Consider a chaos expansion consisting of a reference space ξ containing our choice of distribution $\varphi(\xi)$, for instance $\mathcal{N}(0, I)$, and a mapping $z(\xi)$, represented as

$$z(\xi) = c_i \psi_i(\xi). \tag{3}$$

Here c_i are coefficients, ψ_i are basis functions, and we are using Einstein summation notation for subscripts.[4] Note that the basis-function expansion (3) is a surrogate model for the process $z(\xi)$; this suggests that a variety of techniques for surrogate modelling, like radial basis functions or other machine-learning methods, are applicable. The method is not limited to polynomials.

A chaos expansion can be used to rapidly generate samples (pointwise probability masses) $p(z)$ of $\varphi(z)$, by sampling $\varphi(\xi)$ and applying (3); it can also be used to evaluate integrated quantities, like the pointwise probability density function

$$\varphi(\tilde{z}) = \int \delta\{\tilde{z} - z(\xi)\}\, \varphi(\xi)\, d\xi \tag{4}$$

or, more usefully, weighted integrals of $\varphi(z)$ over the z domain,

$$\int g(z)\, \varphi(z)\, dz = \int g\{z(\xi)\}\, \varphi(\xi)\, d\xi \tag{5}$$

by integrating over the ξ domain.

One of the strengths of a chaos expansion is the ability to handle degenerate probability distributions, like deterministic values or systems with saturation. For instance, a deterministic value $\varphi(\tilde{z}) = \delta(\tilde{z} - z)$ is represented simply by the constant function $z(\xi) = \tilde{z}$.

On the other hand, chaos expansions have shortcomings when applied to a dynamic system like (1). There are two major problems. First, as a nonlinear dynamic system convects in time, the attractor may become chaotic or fractal in nature. The result is a complex, fine geometry that cannot be represented by low-order basis functions.[5] The addition of stochastic noise may smooth out the fine structure in the attractor, but this does not fix the problem: points that start out nearby in both the z and ξ spaces remain nearby in ξ, but move apart over time in z, such that $z(\xi)$ becomes spiky.

Stochastic noise also leads to the second problem: a white-noise input at a given timestep increases the dimension of the probability space by one, so the dimensionality of the problem

[4] ... not superscripts, which are used to index the timestep.

[5] This is noted in the literature as a lack of convergence when using a finite number of basis-function terms over long simulations. (Gerritsma 2010)

grows with time.[6] This increase in dimension can be avoided by using Monte-Carlo techniques, but again this causes neighboring z coordinates to depart from each other over time, creating a spiky $z(\xi)$.

Can we find a way to fix these problems? Imagine now that we have a nonlinear dynamic system, and start with a known chaos expansion $\varphi(\eta) = \mathcal{N}(0, I)$ and $z_0(\eta)$ at some initial time. We simulate in Monte-Carlo fashion for some timesteps, until we notice that $z(\eta)$ is becoming spiky. We now pause the simulation, generate $\varphi(z)$ – say, as a cloud of sampled particles obtained via $\varphi(\eta)$ and $z(\eta)$ – and ask: Does there exist a simple, smooth mapping from $\varphi(z)$ to a new reference space $z(\xi)$, with $\varphi(\xi) = \mathcal{N}(0, I)$? That is to say, we wish to *rejuvinate* the mapping between the z domain and the reference space ξ, in a way that gets rid of the spikiness that was present in $z(\eta)$.[7] Although $\varphi(\eta)$ and $\varphi(\xi)$ have the same form, there is an infinite number of mappings that carry one arbitrary probability distribution into another, so $z(\xi)$ does not have to resemble $z(\eta)$. We are therefore free to choose a nice $z(\xi)$: if we can do so in a computationally efficient way.

It is seen that the key to success is the ability to quickly find a simple transformation $z(\xi)$, given $\varphi(z)$ and $\varphi(\xi)$. The Rosenblatt transformation is an established method for transforming one arbitrary probability density function to another. The transformation constructs $z(\xi)$ such that

$$P(z_1) = P(\xi_1) \tag{6}$$
$$P(z_2|z_1) = P(\xi_2|\xi_1) \tag{7}$$
$$P(z_3|z_1 z_2) = P(\xi_3|\xi_1 \xi_2) \tag{8}$$

and so on, where

$$P(\tilde{v}|w) = \int_{-\infty}^{\tilde{v}} \varphi(v|w)\, dv. \tag{9}$$

The difficulty with (6) and the subsequent lines is immediately evident: a path integral such as (9) cannot be evaluated easily or accurately from a collection of nonuniformly-spaced particles, and integrating over the space along such regular paths is particularly troublesome in high dimensions. An alternate numerical method is needed.

3. Particle flow

Particle flow was introduced by Daum and Huang (2007), and the concept has since been developed further (Daum *et al.* 2018), with Crouse and Lewis (2020) providing an excellent review and derivation of the principal equations. The key insight is that one can frame the solution for the transformation from one probability field to another in terms of a pseudo-time differential equation, also called a homotopy. So far, the technique has been applied in the context of Bayesian state estimation. Consider Bayes' formula,

$$\varphi(x|y) = \frac{\varphi(y|x)\,\varphi(x)}{\varphi(y)}, \qquad \varphi(y) = \int \varphi(y|x)\,\varphi(x)\, dx. \tag{10}$$

[6] The Karhunen-Löeve expansion (Ghanem and Spanos 1991) of a Wiener process may help, but this only delays, not fixes, the problem.

[7] "What is the point," you may ask, "why not simply use a particle filter?" Although representing the probability density function as a cloud of particles is indeed akin to a particle filter, which does not require a chaos expansion, what we gain here is the ability to decouple the particle cloud from the equations of motion. That is, we can simulate $z(\eta)$ in an intrusive way, dealing directly with the basis-function coefficients as state variables, or in a non-intrusive way, where the particles are chosen so as to best represent $z(\eta)$. Then the particles used to realize $\varphi(z)$ can be generated by the chaos expansion, and need not all be convected through the equations of motion, which could be a substantial savings in computational effort.

We start with the prior $\varphi(x)$; encounter some new information y, for which there is a model $\varphi(y|x)$ describing the likelihood; and wish to find the posterior $\varphi(x|y)$. Taking the logarithm of (10) gives

$$\log \varphi(x|y) = \log \varphi(x) + \log \varphi(y|x) - \log \varphi(y). \tag{11}$$

Next, introduce a pseudo-time parameter λ, with $0 \leq \lambda \leq 1$. Consider the equation, or homotopy,

$$\log \varphi(x(\lambda)) = \log \varphi(x(0)) + \lambda \log \varphi(y|x(0)) - \log \varphi(y, \lambda). \tag{12}$$

The term $\varphi(y, \lambda)$ is defined as a normalization, such that $\varphi(x(\lambda))$ remains a valid probability density function for all values of λ. Then, when $\lambda = 0$, the right-hand side is equal to the prior; and when $\lambda = 1$, it becomes the posterior. If we begin at $\lambda = 0$ with a collection of particles representing the prior, and these are convected in a way that satisfies (12), then at $\lambda = 1$ the particles will be located so as to represent the posterior.

Now, (12) is a scalar equation, whereas x is generally high-dimensional, containing the coordinates of a collection of particles in multidimensional space. The problem is therefore highly underdetermined: as noted previously, there are an infinite number of possible solutions for the transformation between two probability distributions. At first glance, this seems to be daunting; yet Daum *et al.* had the insight that, from a computational perspective, this freedom is more a blessing than a curse: one can develop a variety of feasible approaches, and choose the one that best satisfies requirements such as low computational effort and robustness.

To illustrate one possible approach (Daum *et al.* 2013), we can begin with the natural observation that the evolution of probability in the pseudo-time parameter λ should obey the Fokker-Planck-Kolmogorov equation. If we assume that there is an underlying dynamic (in λ) system that evolves according to the stochastic differential equation

$$dx = f(x, \lambda)\, d\lambda + \sigma(x, \lambda)\, dw, \tag{13}$$

where f is the drift coefficient, σ is the diffusion coefficient, and w is a white-noise process, then the Fokker-Planck-Kolmogorov equation gives the time evolution of the probability density function,

$$\frac{\partial \varphi(x)}{\partial \lambda} = -\frac{\partial (\varphi(x)\, f_i(x, \lambda))}{\partial x_i} + \frac{1}{2} \frac{\partial^2 (\varphi(x)\, Q_{ij})}{\partial x_i\, \partial x_j}, \quad Q_{ij} = \sigma_{ik} \sigma_{kj}. \tag{14}$$

Returning to (12), and taking the derivative with respect to λ, noting that

$$\frac{d \log a}{db} = \frac{1}{a} \frac{da}{db}, \tag{15}$$

we obtain

$$\frac{1}{\varphi(x)} \frac{\partial \varphi(x)}{\partial \lambda} = \log \varphi(y|x(0)) - \frac{\partial}{\partial \lambda} \log \varphi(y, \lambda). \tag{16}$$

Expanding the first term on the right-hand side of (14) and using (16),

$$\log \varphi(y|x(0)) - \frac{\partial}{\partial \lambda} \log \varphi(y, \lambda) = -\frac{\partial \log \varphi(x)}{\partial x_i} f_i - \frac{\partial f_i}{\partial x_i} + \frac{1}{2\varphi(x)} \frac{\partial^2 (\varphi(x)\, Q_{ij})}{\partial x_i\, \partial x_j}. \tag{17}$$

The remaining task is to convert this scalar equation into a vector equation for the components of f; here this is done by differentiating (17) with respect to each component x_k, giving

$$\frac{\partial \log \varphi(y|x(0))}{\partial x_k} = -\frac{\partial^2 \log \varphi(x)}{\partial x_i\, \partial x_k} f_i - \frac{\partial \log \varphi(x)}{\partial x_i} \frac{\partial f_i}{\partial x_k} - \frac{\partial^2 f_i}{\partial x_i\, \partial x_k} + \frac{\partial}{\partial x_k} \frac{1}{2\varphi(x)} \frac{\partial^2 (\varphi(x)\, Q_{ij})}{\partial x_i\, \partial x_j}. \tag{18}$$

Next a little trick is played: since there is complete freedom in the choice of Q_{ij}, we can take

$$-\frac{\partial \log \varphi(x)}{\partial x_i}\frac{\partial f_i}{\partial x_k} - \frac{\partial^2 f_i}{\partial x_i\,\partial x_k} + \frac{\partial}{\partial x_k}\frac{1}{2\varphi(x)}\frac{\partial^2(\varphi(x)\,Q_{ij})}{\partial x_i\,\partial x_j} = 0 \tag{19}$$

as its definition. The drift coefficient vector can then be solved from the remaining terms,

$$f_i = -\left(\frac{\partial^2 \log \varphi(x)}{\partial x_i\,\partial x_k}\right)^{-1}\frac{\partial \log \varphi(y|x(0))}{\partial x_k}. \tag{20}$$

Numerically, the particle flow can be solved with (Daum *et al.* 2018) or without (Daum and Huang 2013) the diffusion (19). In the latter case, one simply takes $\partial x_i/\partial\lambda = f_i$ as the state equation for the particle flow. This and similar particle-flow algorithms have been found to be highly effective in real-time Bayesian state estimation: the method is computationally efficient.

Why not apply a particle flow algorithm to transform between two arbitrary probability density functions, in lieu of the Rosenblatt transformation? Say that we have $\varphi(z)$ and $\varphi(\xi)$; define the homotopy

$$\varphi(z(\lambda)) = \varphi(\xi)\left(\frac{\varphi(z(1))}{\varphi(\xi)}\right)^{\lambda}, \tag{21}$$

which satisfies the desired initial and final conditions, and take the logarithm of both sides,

$$\log\varphi(z(\lambda)) = \log\varphi(\xi) + \lambda\left(\log\varphi(z(1)) - \log\varphi(\xi)\right). \tag{22}$$

Following the same procedure as (13) through (20), we end up with

$$f_i = -\left(\frac{\partial^2 \log \varphi(z)}{\partial z_i\,\partial z_k}\right)^{-1}\frac{\partial}{\partial z_k}\left(\log\varphi(z(1)) - \log\varphi(\xi)\right). \tag{23}$$

Convecting a cloud of particles from $\lambda = 0$ to $\lambda = 1$, the particles move from an initial set of coordinates ξ to a final set of coordinates z; the initial and final coordinates of each particle thus define pointwise a function $z(\xi)$, which is the desired mapping between the two probability density functions.

It is also possible to implement the transform in the reverse direction, from z to ξ. However, it was found to be advantageous to start with the reference distribution $\varphi(\xi)$. In the present implementation, quasi-random samples were drawn from the reference distribution based on a Poisson disk method (Bridson 2007), which resulted in a nicely-spread initial distribution of particles.

Note that the second-derivative term in (23) is implemented numerically using the right-hand side of (22); that is,

$$\frac{\partial^2 \log \varphi(z)}{\partial z_i\,\partial z_k} = \frac{\partial^2}{\partial z_i\,\partial z_k}\left\{\log\varphi(\xi) + \lambda\left(\log\varphi(z(1)) - \log\varphi(\xi)\right)\right\}. \tag{24}$$

This allows a smoothed kernel function fit to be computed once upfront, and used throughout the calculation of the trajectories, rather than fitting new kernel functions to the particles at each λ step. In order to prevent divergence of particles in the tails of the distribution, Gaussian kernel functions, which decay to zero as $|r| \to \infty$, were fit to the difference $\log\varphi(z(1)) - \log\varphi(\xi)$. This forces the tail behavior to match that of the reference distribution $\varphi(\xi)$, such that the few particles that start out in the tails remain at or near their initial positions.

Two methods of computing the derivatives were investigated, a k-nearest-neighbors algorithm (Choi *et al.* 2011, Meyer *et al.* 2001), and analytical gradients computed from the kernel function. The results were similar, and the analytical gradients have been used in the present work.

Figure 1 shows a simple example in which the technique is used to establish a function $z(\xi)$ mapping a bivariate unit normal distribution to a more complicated multimodal distribution. One hundred particles were used in the calculation. Plot (a) shows the reference distribution and (c) the target distribution, with (b) the trajectories of the particles. The final positions of the particles, when fit with a kernel function, produce the distribution (d). This matches well, although not precisely, with the target (c).

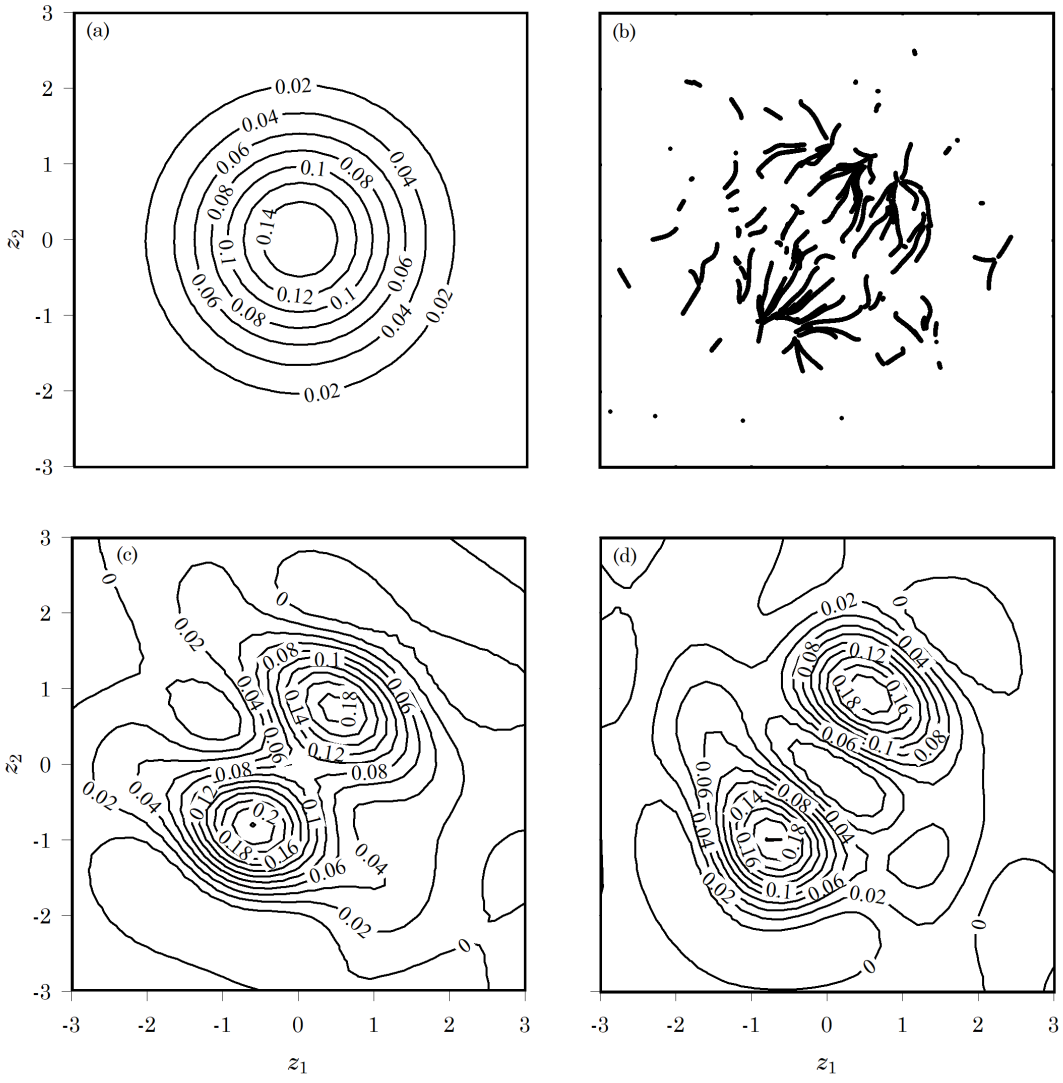

Figure 1. The reference distribution (a), particle trajectories (b), target distribution (c), and final distribution (d).

Figure 2 illustrates the function $z(\xi)$. The particles are plotted over a rendered interpolating surface, obtained using triangulation. The distribution of the particles is consistent, and could be fit with a relatively low-order basis function.

4. Potential applications in wind plant dynamics and control

Why might we want to use particle flow for wind plant dynamics and control? First and foremost, it can be used to implement Bayes' formula (10) for nonlinear state estimation, with applications

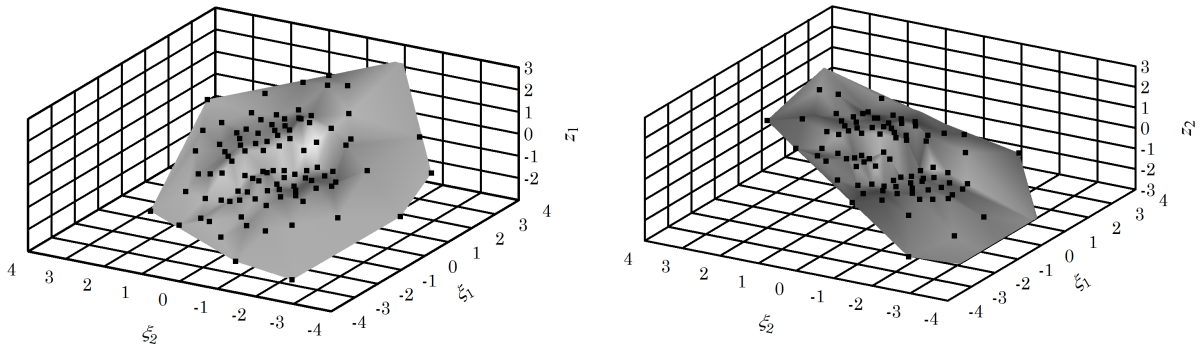

Figure 2. The functions $z_1(\xi)$ and $z_2(\xi)$.

in real-time control; this is the purpose for which particle flow was originally developed. For example, one might wish to obtain a reliable estimate of the effective wind speed and direction over a given wind turbine's rotor, using a combination of sensors from both the turbine itself and a cluster of neighboring turbines. Time lags, and especially wake effects, may make this problem non-Gaussian. Or, say, we might want to estimate the fatigue cycle accumulation rate at a bolted connection at the blade root, based on rudimentary measurements of the rotor speed, blade pitch, generator power, and nacelle accelerations, with some uncertainty regarding the distribution of aerodynamic loading along the blade.

In the context of uncertainty quantification and stochastic dynamics, particle flow in the form of (22), together with a chaos expansion $\varphi(\xi)$ and $z(\xi)$, provides a technique for compressed storage and retrieval (sampling) of a non-Gaussian probability density function. So, where might it be useful to have knowledge of the joint probability distribution over a collection of state variables or outputs?

One possible application is in forecasting under uncertainty. For example, one might wish to use upstream turbines to detect changes in the incoming wind conditions such as gusts or weather fronts, and react in a coordinated way across the wind plant. Yet there is uncertainty in the magnitude and timing of the events. Using a simplified model for wind propagation, one might form joint distributions of the predicted wind speed and direction, rotor speed, and blade pitch at each wind turbine, in order to judge the likelihood of exceeding safe operating limits and to take anticipatory action. The control system of a wind turbine includes nonlinear features such as saturation, dead-band, speed exclusion zones, and control-mode transitions, which make the response non-Gaussian, especially in the vicinity of the rated wind speed where loading is highest.

Another possible application is in offline fatigue analysis. Here we are interested in a component of stress at some critical location in the structure. Given the joint probability distribution of the stress component's trough-to-peak amplitude and trough-to-peak time interval, one can estimate the fatigue cycle accumulation rate. This requires propagating the system in time, probabilistically, through a half cycle of oscillation. Eiken (2007) developed the technique for single-degree-of-freedom systems, using cell-to-cell mapping to handle propagation of probability. Cell-to-cell mapping is flexible, but it involves meshing the entirety of space, and so does not scale well with dimension. A particle-flow chaos espansion approach, with either intrusive (basis-function) or particle-based simulations, could be useful in extending such an analysis to higher-dimensional systems.

5. Conclusions

The particle-flow framework has been adapted so as to provide a general transformation between two probability density functions. The method is well-suited for nonuniformly spaced particles, in contrast with the commonly-used Rosenblatt transformation. The ability to transform from one probability distribution to another, in a computationally-efficient way, enables one to generate the mapping $z(\xi)$ between a general distribution $\varphi(z)$ and some easy-to-use reference distribution $\varphi(\xi)$. This can in turn be applied to uncertain and stochastic dynamic systems, in essence storing the probability in the reference distribution, and periodically rejuvinating the mapping $z(\xi)$ so as to eliminate the emergence of sharp features.

The principle was demonstrated on a simple example. The theory is sound, but the idea needs to be fleshed out and explored for a variety of real-world problems. As Daum *et al.* (2007, 2013, 2018) have discussed, the practicality of particle flow methods depends very much on the details of the implementation. Many research questions remain open. Is the method efficient and accurate enough, with a reasonably small number of particles, or with an intrusive chaos expansion? How does it scale with dimension? The present implementation involves a kernel function fit; can a version be developed that does not rely on a kernel, and so can better handle singularities like saturation bounds or deterministic values? Is the proposed geodesic flow (23) the best choice, or do alternative methods give better accuracy? Is there a compelling reason to use chaos expansions for stochastic dynamic systems, given the ability to rejuvinate the mapping $z(\xi)$?

Some applications were suggested in the domain of wind plant system dynamics and control. Daum *et al.*'s original version of particle flow, for solving Bayes' formula, will undoubtedly be useful in real-time state estimation, with applications in digital twins and control. A chaos expansion framework might be applied for forecasting under uncertainty, fatigue analysis, or myriad other applications, however the utility remains to be demonstrated.

Functions implementing the particle flow and chaos expansion methods have been programmed in the Julia language.

Acknowledgements

Financial support for this work has been provided by the Norwegian Research Council and the industrial partners of NorthWind: Norwegian Research Centre on Wind Energy (grant no. 321954), with a background in the Norwegian projects OPWIND: Operational Control for Wind Power Plants (grant no. 268044) and CONWIND: Research on Smart Operation Control Technologies for Offshore Wind Farms (grant no. 304229), as well as the European Union Horizon 2020 programme (TotalControl, grant no. 727680).

Thanks to Valentin Chabaud and John Olav Tande for reviewing and commenting on the initial draft of the manuscript.

References

[1] Bridson R 2007 Fast Poisson disk sampling in arbitrary dimensions *Proc. ACM SIGGRAPH 2007 (San Diego)* 22 (New York: Association for Computing Machinery)

[2] Choi S, Willett P, Daum F and Huang J 2011 Discussion and application of the homotopy filter *Proc. SPIE 8050, Signal Processing, Sensor Fusion, and Target Recognition 2011 (Orlando)* 805021 (Bellingham: The International Society for Optics and Photonics)

[3] Crouse D, Lewis C 2020 *Consideration of particle flow filter implementations and biases* Report NRL/MR/5344--19-9938 (Washington DC: Naval Research Laboratory)

[4] Dalbey KR *et al.* 2020 *Dakota, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis, Version 6.13 Theory Manual* Report SAND2020-12496 (Albequerque: Sandia National Laboratory)

[5] Daum F and Huang J 2007 Nonlinear filters with log-homotopy *Proc. SPIE 6699, Signal and Data Processing of Small Targets 2007 (San Diego)* 669918 (Bellingham: The International Society for Optics and Photonics)

[6] Daum F and Huang J 2013 Particle flow with non-zero diffusion for nonlinear filters *Proc. SPIE 8745, Signal Processing, Sensor Fusion, and Target Recognition XXII 2013 (Baltimore)* 87450P (Bellingham: The International Society for Optics and Photonics)
[7] Daum F, Huang J and Noushin A 2018 New theory and numerical results for Gromov's method for stochastic particle flow filters *Proc. 21st International Conference on Information Fusion (FUSION) 2018 (Cambridge)* 108-115 (International Society of Information Fusion)
[8] Eiken O 2007 *Fatigue damage in an offshore wind turbine using probability density evolution* MS Thesis, Department of Structural Engineering (Trondheim: Norwegian University of Science and Technology)
[9] Evensen G 2009 The ensemble Kalman filter for combined state and parameter estimation *IEEE Control Systems Magazine* June 2009 83-104
[10] Feinberg J and Langtangen HP 2015 Chaospy: an open source tool for designing methods of uncertainty quantification *Journal of Computational Science* **11** 46-57
[11] Gerritsma M, van der Steen J-B, Vos P and Karniadakis G 2010 Time-dependent generalized polynomial chaos *Journal of Computational Physics* **229** 8333-8363
[12] Ghanem RG and Spanos PD 1991 *Stochastic Finite Elements: A Spectral Approach* (New York: Springer-Verlag)
[13] Lataniotis C, Torre E, Marelli S and Sudret B 2021 *UQLab User Manual – The Input Module* Report UQLab-V1.4-102 (Zürich: ETH Zürich)
[14] Mack YP and Rosenblatt M 1979 Multivariate k-nearest neighbor density estimates *Journal of Multivariate Analysis* **9** 1-15
[15] Marelli S, Lüthen N and Sudret B 2021 *UQLab User Manual – Polynomial chaos expansions* Report UQLab-V1.4-104 (Zürich: ETH Zürich)
[16] Merz K 2021 *Basis function methods for probabilistic analysis* Memo AN 21.12.33 (Trondheim: SINTEF Energy Research)
[17] Meyer TH, Eriksson M and Maggio RC 2001 Gradient estimation from irregularly spaced data sets *Mathematical Geology* **33** 693-717
[18] Rosenblatt M 1952 Remarks on a multivariate transformation *Annals of Mathematical Statistics* **23** 470-472