Proceedings of the ASME 2014 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2011
August 17-20, 2014, Buffalo, NY, United States of America

DETC2014-35244

DRAFT: APPLYING ROBUST DESIGN OPTIMIZATION TO LARGE SYSTEMS

Matthew G. McIntire
Email: mcintima@engr.oregonstate.edu

Veronika Vasylkivska
Email: vasylkiv@science.oregonstate.edu

Christopher Hoyle
Department of Mechanical, Industrial, and Manufacturing Engineering
Oregon State University
Corvallis, OR 97331
Email: chris.hoyle@oregonstate.edu

Nathan Gibson
Department of Mathematics
Oregon State University
Corvallis, OR 97331
Email: gibsonn@math.oregonstate.edu

ABSTRACT

While Robust Optimization has been utilized for a variety of design problems, application of Robust Design to the control of large-scale systems presents unique challenges to assure rapid convergence of the solution. Specifically, the need to account for uncertainty in the optimization loop can lead to a prohibitively expensive optimization using existing methods when using robust optimization for control. In this work, a robust optimization framework suitable for operational control of large scale systems is presented. To enable this framework, robust optimization uses a utility function for the objective, dimension reduction in the uncertainty space, and a new algorithm for evaluating probabilistic constraints. The proposed solution accepts the basis in utility theory, where the goal is to maximize expected utility. This allows analytic gradient and Hessian calculations to be derived to reduce the number of iterations required. Dimension reduction reduces uncertain functions to low dimensional parametric uncertainty while the new algorithm for evaluating probabilistic constraints is specifically formulated to reuse information previously generated to estimate the robust objective. These processes reduce the computational expense to enable robust optimization to be used for operational control of a large-scale system. The framework is applied to a multiple-dam hydropower revenue optimization problem, then the solution is compared with the solution given by a non-probabilistic safety factor approach. The solution given by the framework is shown to dominate the other solution by improving upon the expected objective as well as the joint failure probability.

1 Introduction

Robust design is the optimization of a system to be insensitive in its objective and reliable in its constraints to parametric uncertainty [1]. In the state of the art [2, 3, 4], the objective to be optimized is a weighted combination of the mean and variance of the performance function. The limitations and expense present in this method are unacceptable when using robust optimization for the control of large-scale systems with hundreds of decision variables, and uncertain functions, rather than scalar parameters. First, uncertain functions must be discretized, resulting in hundreds of highly correlated uncertain parameters. This challenge can be overcome through the use of a Karhunen-–Loève, or KL, transformation, decomposing the uncertain functions to reduce the dimensionality of the uncertainty to just a few standard normal variables. Second, using the weighted mean and variance formulation does not produce acceptable results when the objective distribution is non-symmetric, as higher variation above the mean is penalized the same as higher variation below the mean. Logically, a distribution skewed toward higher performance should be preferred over a distribution with the same mean and variance that is skewed toward lower performance. Third, it is not straightforward to calculate the gradient of the objective using the weighted sum method, due to the use of the variance term in the objective. Using finite differencing to estimate the gradient and Hessian can be infeasible for large-scale control problems due to the large number of decision variables. These challenges are overcome through application of a utility function to the performance, enabling a direct calculation of expected utility and its gradient. Fourth, evaluating probabilistic constraints to ensure reliability typically does not use the available information resulting from the simulations run to calculate the robust objective. In this work, that information is reused to construct a surrogate model for each constraint which are evaluated using a new inverse reliability method. The proposed framework incorporates these four methods to enable robust optimization for operational control of large scale systems.

The framework is demonstrated on a case study for daily model-based control updates of the Columbia River Basin hydroelectric system. The Bonneville Power Administration is looking for a way to incorporate parametric uncertainty into the real-time planning optimization of flow through ten major dams on the Columbia and Snake Rivers in the states of Oregon and Washington. The system model contains thousands of decision variables, so any local search optimization must take advantage of all gradient information provided by the model.

2 Mathematical Background

The framework presented in this paper builds upon work from mathematics, economics, and engineering: uncertainty quantification, uncertainty propagation, utility theory, robust design, reliability-based design, and numerical optimization.

2.1 Uncertainty Quantification: the KL Transform

We describe the method we use to introduce the uncertainty into the system. Let $(\Omega, \mathscr{F}, \mathbb{P})$ be a complete probability space, where Ω is the set of outcomes, $\mathscr{F} \subset 2^{\Omega}$ is a σ-algebra of events and $\mathbb{P} : \mathscr{F} \to [0,1]$ is a probability measure. Assume that the parameters $\mathbf{P}$ can be described as a matrix-valued function of finite number N_{rv} of independent random variables $\{\xi_k\}_{k=1}^{N_{rv}}$, i.e.

$$\mathbf{P}(\omega) = \mathbf{P}(\xi_1(\omega), \xi_2(\omega), \ldots, \xi_{N_{rv}}(\omega)). \tag{1}$$

Let $\rho_k : \Gamma_k \to \mathbb{R}^+$, $k = 1,2,\ldots,N_{rv}$, denote the probability density function of the random variable ξ_k, with the image $\Gamma_k = \xi_k(\Omega) \subset \mathbb{R}$, $k = 1,2,\ldots,N_{rv}$. If the random variables $\{\xi_k\}_{k=1}^{N_{rv}}$ are independent then the joint probability density function ρ is given by the product of the corresponding densities

$$\rho(\mathbf{z}) = \prod_{k=1}^{N_{rv}} \rho_k(z_k), \quad \mathbf{z} \in \Gamma, \quad z_k \in \Gamma_k, \tag{2}$$

where $\Gamma = \prod_{k=1}^{N_{rv}} \Gamma_k \subset \mathbb{R}^{N_{rv}}$ is a support of the joint density function ρ. One way to obtain the finite dimensional representation of a component p_{ij} of the matrix $\mathbf{P}$ is possible with the help of Karhunen-Loève expansion. This representation works for time-dependent parameters. For the purpose of implementation several assumptions need to be satisfied: a particular component p_{ij} is a random process; several independent realizations of p_{ij} are available. Under these assumptions we can build a representation based on the spectral decompostion of the covariance of p_{ij}. For simplicity of exposition let $Q = p_{ij}$.

1. Suppose we have M realizations of the parameter Q measured at time points $\{t_j\}_{j=0}^{n}$, where $t_j = t_0 + jh$, $h = \dfrac{T - t_0}{n}$, $j = 1,\ldots,n$, and $[t_0, T]$ is a time interval of interest. By $Q_i(t_j)$ we denote the value of the i-th realization of the parameter Q at the time point t_j.
2. Then we compute the sample mean vector $\bar{Q} = (\bar{Q}_1, \bar{Q}_2, \ldots, \bar{Q}_n)^T$ and an $(n \times n)$ covariance matrix C with elements $c_{j,k}$ of the process Q using the following formulas

$$\bar{Q}_j = \bar{Q}(t_j) = \frac{1}{M}\sum_{i=1}^{M} Q_i(t_j) \tag{3}$$

$$c_{j,k} = \frac{1}{M-1}\sum_{i=1}^{M} (Q_i(t_j) - \bar{Q}_j)(Q_i(t_k) - \bar{Q}_k). \tag{4}$$

3. It follows that $Q(t)$ can be represented in the form of its infinite series representation, called the Karhunen-Loève expansion

$$Q(t) = \bar{Q}(t) + \sum_{k=1}^{\infty} \sqrt{\lambda_k}\,\psi_k(t)\xi_k, \tag{5}$$

where $\{\lambda_k, \psi_k\}_{k=1}^{\infty}$ are the eigenpairs of the integral equation

$$\lambda\psi(t) = \int_{t_0}^{T} C(s,t)\psi(s)ds, \tag{6}$$

with $C(t_j, t_k) = c_{j,k}$; and $\{\xi_k\}_{k=1}^{\infty}$ is a sequence of uncorrelated random variables with mean 0 and variance 1 defined by

$$\xi_k = \frac{1}{\sqrt{\lambda_k}} \int_{t_0}^{T} [Q(t) - \bar{Q}(t)]\psi_k(t)dt, \quad k \geq 1. \tag{7}$$

We assume that the eigenvalues are arranged in decreasing order, that is, $\lambda_1 > \lambda_2 > \lambda_3 > \cdots$. In the case Q is a Gaussian random process, $\{\xi_k\}_{k=1}^{\infty}$ are independent and identically distributed normal random variables with mean 0 and variance 1.
4. If the positivity of the random process is an issue, the Karhunen-Loève expansion of the logarithm of a particular parameter can be considered instead.

The infinite representation of Q is not practical. The truncated representation is used instead

$$Q(t) \approx Q_N(t) = \bar{Q}(t) + \sum_{k=1}^{N} \sqrt{\lambda_k}\,\psi_k(t)\xi_k. \tag{8}$$

The number of terms N in the truncated representation of Q can be chosen in different ways. One may use a fact that $\sum_{n=1}^{\infty} \lambda_n = \int_{t_0}^{T} C(s,s)ds$ and based on this criteria choose the number of terms that would contribute to the major part of the variance. Another way to determine N is to look at the convergence rate of the eigenvalues and get rid of those that are close to 0, or insignificant in comparison with the first eigenvalue. For example, one can include the eigenvalues λ_n that satisfy

$$\lambda_n < a\lambda_1 \tag{9}$$

for some pre-defined constant $0 < a < 1$.

2.2 Utility Theory

In utility theory, for every uncertain outcome f resulting from a choice of control variable $\mathbf{X}$, there exists an associated certainty equivalent z, an assured outcome that the decision-maker would be ambivalent to choosing. That is, the decision maker would have no reason to choose one over the other: the uncertain outcome, or its certainty equivalent. Assuming the decision maker abides by the axioms of consistent choice [5], there exists a monotonic utility function U, defined implicitly

$$\mathrm{U}(z) = \mathrm{E}[\mathrm{U}(f)], \tag{10}$$

where E is the expectation operator. The utility function thus defines the "utility" u of a given outcome f

$$u = \mathrm{U}(f). \tag{11}$$

If the preferred outcome is high, the utility function must be monotonically increasing. If the preferred outcome is low, the utility function must be monotonically decreasing. Furthermore, if the decision maker is risk seeking, the utility function must be convex downward, and if the decision maker is risk averse, the utility function must be concave downward. If the decision maker is risk neutral, the utility function must be linear. It is shown that an exponential utility function approximates (10) for the case of *constant* risk aversion using the following argument adapted from [6].

First, define δ_z and Δ_f in terms of z, f, and the expected outcome, $\mathrm{E}(f)$.

$$\delta_z = z - \mathrm{E}(f), \tag{12}$$
$$\Delta_f = f - \mathrm{E}(f). \tag{13}$$

δ_z is the *risk premium*, the deviation of the decision maker's certainty equivalent from the expected outcome, and Δ_f is the random deviation of the outcome from its expected value. Rewrite (10) using equations (12) and (13)

$$\mathrm{U}(\mathrm{E}(f) + \delta_z) = \mathrm{E}[\mathrm{U}(\mathrm{E}(f) + \Delta_f)]. \tag{14}$$

Next, apply the Taylor expansion to both sides, and assume f is normally distributed, the mean $\mathrm{E}[\Delta_f] = 0$ and the variance $\mathrm{E}[\Delta_f^2] = \sigma_f^2$. Next, assume that the magnitude of the risk premium is quite a bit less than the magnitude of the uncertainty $\delta_z^2 \ll \sigma_f^2$, and that the uncertainty is small enough to disregard all terms third-order and higher. This leaves

$$\delta_z = \frac{\sigma_f^2}{2} \frac{\dfrac{d^2\mathrm{U}(\mathrm{E}(f))}{df^2}}{\dfrac{d\mathrm{U}(\mathrm{E}(f))}{df}}. \tag{15}$$

Now, constant risk aversion means that the risk premium is independent of the mean output. Consequently, any change to f that affects μ_f but not σ_f will not affect δ_z. Therefore,

$$\frac{\dfrac{d^2\mathrm{U}(f)}{df^2}}{\dfrac{d\mathrm{U}(f)}{df}} = c, \tag{16}$$

where c is a constant. The solution of (16) is

$$\mathrm{U}(f) = \begin{cases} a + be^{cf}, & c \neq 0 \\ a + bf, & c = 0 \end{cases}, \tag{17}$$

where a and b are also constants. Because utility exists only to rank outcomes on an interval (not ratio) scale, a is irrelevant, as is the magnitude (but not the sign) of b. The exponential utility function (17) can be parameterized according to the decision maker's risk aversion, and whether the preferred outcome is high or low. In the case of risk-neutrality, $c = 0$ and $b > 0$ for increasing preferences, $b < 0$ for decreasing. $b > 0$ in the case of risk tolerance and $b < 0$ in the case of risk aversion. In both risk-tolerant and risk-averse cases, the signs of b and c are the same for increasing preferences, and opposite for decreasing preferences. The magnitude of c increases in both directions from risk neutrality.

Substituting (16) back into (15) leaves

$$\delta_z = c\frac{\sigma_f^2}{2}, \tag{18}$$

which itself may be combined with (12)

$$z = \mathrm{E}(f) + \frac{1}{2}c\sigma_f^2. \tag{19}$$

to finally derive a representation of the certainty equivalent as a function of the mean and variance of an uncertain output.

2.3 Robust Design

Hazelrigg [5] encourages the maximization of expected utility as the decision-making objective in engineering design. However, due to the apparent difficulty in defining a utility function for a particular entity [3, 4], researchers in robust design have instead opted to utilize a weighted combination of the mean $\mathrm{E}(f)$ and variance σ_f^2 of the performance function as their objective [2]. This way, the decision-maker only has to specify one risk aversion coefficient, r, in a robust objective such as

$$\max \quad \mathrm{E}(f) - r\sigma_f^2, \tag{20}$$

where $r > 0$ corresponds to a risk-averse decision maker, $r < 0$ corresponds to a risk-seeking decision maker, and $r = 0$ corresponds to a risk-neutral decision maker.

It can easily be seen that (20) has the same form as (19). Thus, optimizing a robust objective which is a weighted combination of the mean and variance of a performance function is a good approximation to optimizing the certainty equivalent given an exponential utility function, the approximation of the utility function for constant risk aversion. Or, the expression in (20) is approximately equal to z in (10), given U has an exponential form.

2.4 Inverse Reliability Analysis (Probabilistic Constraints)

In deterministic optimization, we say that the problem must be subject to inequality constraints of the form

$$g_i \leq 0 \qquad \text{for } i = 1, \ldots, N_g, \tag{21}$$

where N_g is the number of constraints to be satisfied. We define a probabilistic constraint as

$$\mathbb{P}(g_i > 0) - \alpha_i \leq 0 \qquad \text{for } i = 1, \ldots, N_g, \tag{22}$$

where $\vec{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{N_g})$ is a vector of acceptable probabilities of failure, one for each constraint. Typical α-values range from 10^{-1} to 10^{-6}, depending on the consequences of violating each constraint. Joint failure probabilities are not considered in this method. We test the joint failure probability of a solution after the optimization completes.

In inverse reliability analysis [7], we utilize an idea from the First Order Reliability Method (FORM), the most probable point of failure (MPP). The MPP is defined for each constraint g_i as the most probable point in the uncertainty space such that $g_i = 0$. Assuming the uncertainty can be represented by a set of standard normal variables $\vec{\xi}$, the two-norm of the MPP, known as the reliability index β, can be used to calculate a first-order approximation of the probability of failure by taking its quantile in the standard normal distribution.

There are two relevant issues that preclude us using FORM to calculate probabilistic constraints. First, the search for the MPP may lead into areas of extremely low probability (such as in cases with a negligible probability of failure), where the simulation model may no longer be very accurate. Second, there is no known function that maps $\nabla_x g_i$ to $\nabla_x \mathbb{P}(g_i > 0)$, and we must utilize available gradient information for the framework to be efficient.

Inverse reliability analysis [7] assumes that there exists some control settings $\mathbf{X}$ for each constraint such that $\mathbb{P}(g_i > 0) = \alpha_i$; that is, that the MPP in that instance represents a probability of failure exactly that which is maximally allowable. It then assumes that we can approximate that unknown MPP (designated $\vec{\xi}_{\alpha_i}$) by considering the set of all possible MPPs corresponding to a probability of failure α_i, and choosing the one which at the current x produces the highest (worst) g_i. This is an optimization problem for each constraint at each iteration of the optimization over x.

$$\vec{\xi}_{\alpha_i} = \max_{\vec{\xi}} \quad g_i \quad \text{subject to} \quad \Phi(\|\vec{\xi}\|) = \alpha_i \tag{23}$$

We can then use the g_i evaluated at each $\vec{\xi}_{\alpha_i}$ as appropriate substitutes for the probabilistic constraints 22, as they will always cross zero simultaneously in x.

2.5 Polynomial Chaos Representation

To assist in solving our problem in the stochastic context we form a generalized Polynomial Chaos (PC) expansion of each constraint as a function of the random variables $\{\xi_k\}$ described in section 2.1 as a surrogate model. To illustrate the idea of the proposed uncertainty approach, we illustrate the process on one constraint, $g = g_i$ for some $i = 1, \ldots, N_g$. We want to determine the coefficients of a PC expansion of each constraint for some fixed time t. The representation of the constraint g in terms of a degree p expansion is

$$g^p(t, \vec{\xi}) = \sum_{k=0}^{M_p} v_k(t) \Phi_k(\vec{\xi}), \tag{24}$$

where $\vec{\xi} = (\xi_1, \xi_2, \ldots, \xi_{N_{rv}})$ is a vector of random variables in the representation of P, $(M_p + 1)$ is a number of basis functions used. The functions $\{\Phi_k\}_{k=0}^{M_p}$ are the orthogonal polynomials of a degree at most p in each of N_{rv} variables. The maximum possible number of polynomial basis functions in this case is $p^{N_{rv}}$. Note that the actual number of basis functions depends on if the same degree polynomials are used in each random dimension for the approximation.

Each PC expansion coefficient can be found as an expectation

$$v_k(t) = \mathrm{E}[g(t, \vec{\xi}) \Phi_k(\vec{\xi})] = \int_{\Gamma} g(t, \mathbf{z}) \Phi_k(\mathbf{z}) \rho(\mathbf{z}) d\mathbf{z}. \tag{25}$$

The computation of the coefficients in (25) can be done efficiently with the use of the stochastic collocation (SC) method. The outline of the application of the SC method to the PC expansion is given below:

1. Choose a set of collocation points $(\mathbf{z}_j, w_j)$, $\mathbf{z}_j \in \Gamma$, where $\mathbf{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,N_{cp}})$ is a j-th node and w_j is its corresponding weight, $j = 1, \ldots, N_{cp}$.
2. For each $j = 1, \ldots, N_{cp}$ obtain the surface values g
3. Approximate the PC expansion coefficients

$$v_k(t) = \mathrm{E}[g(t, \vec{\xi}) \Phi_k(\vec{\xi})] \approx \sum_{j=1}^{N_{cp}} w_j g(t, \mathbf{z}_j) \Phi_k(\mathbf{z}_j). \tag{26}$$

4. Finally, construct the N_{rv}-variate, pth-order PC approximation of the solution

$$g^p(t,\vec{\xi}) = \sum_{k=0}^{M_p} v_k(t)\Phi_k(\vec{\xi}). \tag{27}$$

The same coefficients v_k can be used to approximate the first two moments of the constraint g if needed, e.g.

$$\mathrm{E}[g(t,\vec{\xi})] \approx v_0(t), \quad \mathrm{Var}[g(t,\vec{\xi})] \approx \sum_{k=1}^{M_p} v_k(t)^2. \tag{28}$$

If the variance of the constraint (or the component of the solution) is not a concern, then calculation of the coefficients $\{v_k\}_{k=1}^{M_p}$ is not necessary. We are interested in the PC representation only when dealing with the probabilistic constraints. In this case PC representation is used as an initial polynomial surrogate of the response surface described by the contraint function g. The algorithm of calculating the probability associated with a particular constraint is described below in subsection 3.4. It is worth to mention that PC expansion mentioned above is not the only way to obtain the needed polynomial representation of the response surface. We discuss alternatives below.

3 Large-Scale System Robust Control Framework

The robust optimization framework has two main interfaces, one with the problem and one with the decision maker. These divisions allow us to better comprehend the full scope of material covered in this paper.

3.1 Problem Interface

The problem is considered by the robust optimization framework to include a black box simulation-based function, an objective function, and a constraint function. The simulation (Sim) is

$$(S, \nabla_{\mathbf{X}} S, \nabla^2_{\mathbf{X}} S) = \mathrm{Sim}(\mathbf{X}, \mathbf{P}), \tag{29}$$

where $\mathbf{X}$ is an N_t by N_x matrix of control variables, $\mathbf{P}$ is an N_t by N_p matrix of uncertain parameters, and S is an N_t by N_s matrix of system states. N_t is the number of time steps in the simulation, N_x is the number of discretized control functions, N_p is the number of discretized uncertain parameter functions, and N_s is the number of discretized state functions of interest. $\nabla_{\mathbf{X}}$ is the gradient with respect to $\mathbf{X}$; $\nabla^2_{\mathbf{X}}$ is the associated Hessian. The $\nabla_{\mathbf{X}}$ operator adds one dimension of size $n_t n_x$ to the data; $\nabla^2_{\mathbf{X}}$ adds two dimensions, each of size $N_t N_x$.

The objective function (Obj) is

$$(f, \nabla_{\mathbf{X}} f, \nabla^2_{\mathbf{X}} f) = \mathrm{Obj}(S, \nabla_{\mathbf{X}} S, \nabla^2_{\mathbf{X}} S), \tag{30}$$

where f is the scalar objective. The constraint function (Con) is

$$(\vec{g}, \nabla_{\mathbf{X}} \vec{g}, \nabla^2_{\mathbf{X}} \vec{g}) = \mathrm{Con}(S, \nabla_{\mathbf{X}} S, \nabla^2_{\mathbf{X}} S), \tag{31}$$

where $\vec{g}$ is an N_g-length vector of constraint values. N_g is the number of constraints to be satisfied.

Equations (29)-(31) can easily be expanded to include time-invariant control variables, parameters, and states, in addition to those with a discretized functional form. In implementation, the functions are assumed to calculate the derivatives only when called for.

3.2 Decision-Maker Interface

The framework expects from the decision maker a monotonic utility function and a vector of acceptable probabilities of failure. The utility function (U) is

$$(u, \nabla_{\mathbf{X}} u, \nabla^2_{\mathbf{X}} u) = \mathrm{U}(f, \nabla_{\mathbf{X}} f, \nabla^2_{\mathbf{X}} f) \tag{32}$$

defined over the objective space. Its inverse is defined as well, as

$$z = \mathrm{U}^{-1}(u), \tag{33}$$

where z is a certainty equivalent. We discuss the use of (33) in section 3.3.

The vector of acceptable probabilities of failure $\vec{\alpha}$ is an N_g-length vector of alpha levels, one for each constraint. We discuss probabilistic constraints in section 2.4.

3.3 Robust Objective

The following is a robust design objective defined as maximizing the certainty equivalent from (11)

$$\max \quad z = \mathrm{U}^{-1}(\mathrm{E}[u]). \tag{34}$$

Due to the monotonic nature of utility functions (in this case increasing, as $f(\mathbf{X})$ is a performance function, not a cost function), this will have the same solution $\mathbf{X}$ (not the same objective evaluation) as maximizing (10). In practice, however, it is often worth the effort to take the utility inverse of the expected utility, in order to optimize the certainty equivalent. If the original problem is close to linear, or quadratic as it is in this paper, then transforming it with the highly non-linear exponential function will severely hamper the speed of the optimization. The inverse utility operation returns the objective to a more well-behaved space. The inverse utility step is accomplished as follows

$$z = \mathrm{U}^{-1}(\mathrm{E}[u]), \tag{35}$$

$$\nabla_x(z) = \frac{\nabla_x(\mathrm{E}[u])}{\frac{d\mathrm{U}(z)}{dz}}, \tag{36}$$

$$\nabla^2_x(z) = \frac{\nabla^2_x(\mathrm{E}[u]) - \frac{d\mathrm{U}^2(z)}{dz^2}\nabla_x(z)^T\nabla_x(z)}{\frac{d\mathrm{U}(z)}{dz}}. \tag{37}$$

3.3.1 Gradient Calculation

Newton-based local search methods use the gradient and Hessian of the objective in determining the next solution to be tested. If an analytical gradient and Hessian can be calculated, or simulated together with the objective, the number of iterations, as well as the number of simulations per iteration, can be greatly reduced. Conversely, many more decision variables can by introduced. Using gradient information with the weighted sum objective formulation in (20), however, is unknown. Using a utility-based formulation of robust design alleviates this issue.

It has already been shown that the current robust design objective formulation (20) approximates the solution given by the expected utility when the utility function has an exponential shape. The power in using this form is the ability to utilize gradient information from the simulation. If the new robust objective is simply that presented in (34), then it follows that

$$\nabla_x(\mathrm{E}[u]) = \mathrm{E}[\nabla_x(u)], \tag{38}$$

and by the chain rule

$$\nabla_x(\mathrm{E}[u]) = \mathrm{E}\left[\frac{du}{df}\nabla_x(f)\right]. \tag{39}$$

Thus, the gradient of the expected utility is simply a function of the performance output and its gradient. The Hessian is slightly more complicated:

$$\nabla_x^2(\mathrm{E}[u]) = \mathrm{E}\left[\nabla_x(\frac{du}{df}\nabla_x(f))\right], \tag{40}$$

$$\nabla_x^2(\mathrm{E}[u]) = \mathrm{E}\left[\frac{du}{df}\nabla_x^2(f) + \frac{du^2}{df^2}\nabla_x(f)^T\nabla_x(f)\right], \tag{41}$$

due to the gradient being a vector. However, the Hessian of the expected utility is still simply a function of the performance output, its gradient, and its Hessian.

3.3.2 Uncertainty Propagation

We use the full factorial numerical integration (FFNI) [8] method or tensor-product quadrature [9] for uncertainty propagation. We apply it as follows:

1. Choose a set of collocation points $(\mathbf{z}_j, w_j)$, $\mathbf{z}_j \in \Gamma$, where $\mathbf{z}_j = (z_{j,1}, z_{j,2}, \ldots, z_{j,N_{cp}})$ is a j-th node and w_j is its corresponding weight, $j = 1, \ldots, N_{cp}$.
2. For each $j = 1, \ldots, N_{cp}$ evaluate the values of the parameter matrix $\mathbf{P}_j$ and run the corresponding (deterministic) simulation (Sim), in parallel, to obtain the system states and their partial derivatives as in (29)

$$(S_j, \nabla_{\mathbf{X}} S_j, \nabla_{\mathbf{X}}^2 S_j) = \mathrm{Sim}(\mathbf{X}, \mathbf{P}_j). \tag{42}$$

3. Evaluate the utility function and its partial derivatives by combining (30) and (32).

$$(u_j, \nabla_{\mathbf{X}} u_j, \nabla_{\mathbf{X}}^2 u_j) = \mathrm{U}(\mathrm{Obj}(S_j, \nabla_{\mathbf{X}} S_j, \nabla_{\mathbf{X}}^2 S_j)). \tag{43}$$

4. Determine the expected utility, along with the expected gradient and Hessian

$$\mathrm{E}(u) = \sum_{j=1}^{N_{cp}} w_j u_j, \tag{44}$$

$$\mathrm{E}(\nabla_{\mathbf{X}} u) = \sum_{j=1}^{N_{cp}} w_j \nabla_{\mathbf{X}} u_j, \tag{45}$$

$$\mathrm{E}(\nabla_{\mathbf{X}}^2 u) = \sum_{j=1}^{N_{cp}} w_j \nabla_{\mathbf{X}}^2 u_j. \tag{46}$$

5. Determine the certainty equivalent and its partial derivatives with (35)-(37). The certainty equivalent is the robust objective as in (34).

3.4 Probabilistic Constraints

At each iteration of the optimization, we specify an inner optimization problem (23) for each constraint to determine at which point in the uncertainty space to evaluate it and its gradient.

Sampling of the original system (29)-(31) is usually expensive so we use approximate surrogate models in the form of Polynomial Chaos (PC) expansions, as a cheap alternative during each inner optimization. We construct each surrogate (24) by applying (31) to the values of S at the collocation points already calculated during the robust objective calculation as described in section 3.3.2, step 2. Additional work required for sampling the system beyond the collocation points can be explained by the fact that in general PC expansion provides a good approximation only of two statistical moments (mean and variance). The collocation points are not always a good source of information about the behavior of the system at the tails of the distribution, often where the constraints become interesting. For simplicity of representation of the algorithm we consider a single variable of interest g present in the formulation of a probabilistic constraint,

$$\mathbb{P}(g > 0) < \alpha. \tag{47}$$

The outline of the algorithm is given below.

1. Assume we have a solution g as a function of the random vector $\vec{\xi} = (\xi_1, \xi_2, \ldots, \xi_{N_{rv}})$ through the system uncertain parameters.
2. We assume that for a given set of collocation points $\{\mathbf{z}_j\}_{j=1}^{N_{cp}}$ on a particular grid (full or sparse) we know all necessary components of the solution at these points, including $g(\mathbf{z}_j)$, $j = \overline{1, N_{cp}}$.
3. We build a PC representation of the constraint g of degree p, g^p, using the values calculated at $\{\mathbf{z}_j\}_{j=1}^{N_{cp}}$. For convenience of notation, we denote the PC expansion and further improved surrogate models by $\tilde{g}$. In general, if the failure region associated with a given probability constraint is far from the concentration of mass of the joint density function of the random vector $\vec{\xi}$, extra work needs to be done to build an adequate approximation of the solution around or in the failure region.

4. Find $\beta = \Phi^{-1}(\alpha)$, and $\vec{\xi}^*$ as a solution of the problem: $\max\limits_{\vec{\xi}} \tilde{g}(\vec{\xi})$ subject to $\|\vec{\xi}^*\| = \beta$.
5. Sample the system to get $g(\vec{\xi}^*)$. Update the surrogate model $\tilde{g}$ by accounting for the new data point $(\vec{\xi}^*, g(\vec{\xi}^*))$.
6. Repeat steps (4)-(5) as many as N_{rv} more times, stopping if the difference between two subsequent $\vec{\xi}^*$ is smaller than a prescribed tolerance. Among the possible modifications and improvements of the above algorithm are ways of estimating the probability (e.g. SORM, sampling) and use of full or sparse grids for the initial PC approximation. Furthermore, should this algorithm not terminate precisely enough, refinement of $\vec{\xi}^*$ may continue by using a secondary linear surrogate near the failure region.

4 System Model: Steady Pool Routing

The elements of the framework are demonstrated on a sufficiently complex example problem, an interconnected, multiple-reservoir hydroelectric system. The model uses pool routing, or a discrete one-dimensional physical model, to represent the system. The model assumes that water leaving a dam is instantaneously available at the next dam. The model, and many of its parameters, written in MATLAB, is based on a single-dam, deterministic optimization example published by Mathworks in 2012 [10].

4.1 Defining relationships

4.1.1 Functional Form

The amount of water stored s in reservoir i at time t is

$$s_i(t) = s_i(0) + \int_0^t \Delta q_i(u)\,\mathrm{d}u, \tag{48}$$

where $\Delta q_i(t)$ is the net flow rate into reservoir i at time t. The independent variables are the sets of functions $q_{turb}(t)$ and $q_{spill}(t)$, representing the flow rates through the turbines and spillway, respectively. The net flow can be written in terms of them

$$\Delta q_i = q_{ext,i}(t) - q_{turb,i}(t) - q_{spill,i}(t) + \sum_{j\in A_i} \left(q_{turb,j}(t) + q_{spill,j}(t)\right) \tag{49}$$

where A_i is the set of all dams that feed directly into reservoir i, and $q_{ext,i}(t)$ is the total external tributary flow into reservoir i as a parameter. The head k at each dam is

$$k_i(t) = c_{1,i}s_i(t) + c_{2,i}, \tag{50}$$

where c_1 and c_2 are the coefficients of a linear model of the geometry of reservoir i. The performance measure, total revenue R, is

$$R(t) = \sum_{i=1}^{n} \int_0^t p(u)k_i(u)q_{turb,i}(u)\,du, \tag{51}$$

where $p(t)$ is the price of power.

4.1.2 Time-Discretized Form

Implementation of the model requires discretization in time. The functions from the previous section will now be turned into arrays. The volume of water $s_{end_{i,j}}$ stored in reservoir j of n at the end of time step i of m of duration h is defined recursively

$$s_{end_{i,j}} = s_{end_{i-1,j}} + h\Delta q_{i,j}, \tag{52}$$

where $\Delta q_{i,j}$ is the net flow rate into reservoir j during time step i. The average volume of water stored $s_{i,j}$ during each time step is

$$s_{i,j} = s_{end_{i,j}} - \tfrac{1}{2}h\Delta q_{i,j} \tag{53}$$

Equation (53) can be rewritten explicitly using matrices.

$$S = S_0 + \left(L - \tfrac{1}{2}I\right)h\Delta Q, \tag{54}$$

where I is the identity matrix, and L is the square lower triangular summation matrix, with all elements on or below the main diagonal equal to 1. S_0 is a replicated row vector containing the volume of water in each reservoir at time step $i = 0$, is an initial condition. ΔQ can be broken into components

$$\Delta Q = Q_{ext}A_{ext} + \left(Q_{turb} + Q_{spill}\right)\left(A_{out} - I\right). \tag{55}$$

where Q_{turb} is the flow rate through the turbines, Q_{spill} is the flow rate over the spillway, and Q_{ext} is the flow rate for the tributaries. A_{out} is the one-directional adjacency matrix of the river network. Each row represents a dam, and the corresponding column represents the reservoir. For example, $a_{out_{1,2}} = 1$ means that the outflow from dam 1 flows into the reservoir behind dam 2. Similarly, the columns of A_{ext} represent the reservoirs, but its rows represent the tributaries external to the system. For example, $a_{ext_{2,3}} = 1$ means that tributary 2 flows into the reservoir behind dam 3. Substituting (55) into (54)

$$S = S_0 + \left(L - \tfrac{1}{2}I\right)h\left(Q_{ext}A_{ext} + \left(Q_{turb} + Q_{spill}\right)\left(A_{out} - I\right)\right). \tag{56}$$

The average head K is modeled

$$K = C_1 \circ S + C_2, \tag{57}$$

where constants C_1 and C_2 parameterize a linear approximation of the average river cross subsection in each reservoir. Consider C_1 and C_2 to represent row vectors replicated to have the same size as S, in the same manner as S_0. The energy produced $\vec{e}$ is modeled

$$R = \vec{p}^T\left(K \circ Q_{turb}\right)\vec{1}, \tag{58}$$

where $\vec{1}$ is the one vector, used to sum over the columns, and $\vec{p}$ is the price of energy during each time step.

4.2 Gradient calculation

The gradient is a collection of partial derivatives with respect to each control variable. The control variables are arranged in a three-dimensional tensor Q, where $q_{t,k,l}$ is the flow at time t out of dam k through passage l. Currently, each dam has only two passages, turbine and spill, designated 1 and 2. Thus the gradient of S can be constructed from (56) as follows

$$\frac{\partial S}{\partial q_{t,k,l}} = h\left(L - \tfrac{1}{2}I\right)\left(\frac{\partial [q_{i,j,1}]}{\partial q_{t,k,l}} + \frac{\partial [q_{i,j,2}]}{\partial q_{t,k,l}}\right)(A_{out} - I), \tag{59}$$

where

$$[q_{i,j,1}] = Q_{turb}, \quad [q_{i,j,2}] = Q_{spill} \tag{60}$$

and

$$\frac{\partial q_{i,j,g}}{\partial q_{t,k,l}} = \begin{cases} 1, & \left\{i\ j\ g\right\} = \left\{t\ k\ l\right\}. \\ 0, & \text{else} \end{cases} \tag{61}$$

From (57)

$$\frac{\partial K}{\partial q_{t,k,l}} = C_1 \circ \frac{\partial S}{\partial q_{t,k,l}}. \tag{62}$$

Finally, from (58), since $\vec{p}$ is independent of $\vec{q}$

$$\frac{\partial R}{\partial q_{t,k,l}} = \vec{p}^{\,T}\left(K \circ \frac{\partial Q_{turb}}{\partial q_{t,k,l}} + Q_{turb} \circ \frac{\partial K}{\partial q_{t,k,l}}\right)\vec{1}. \tag{63}$$

4.3 Hessian calculation

First, due to $\partial Q_{turb}/\partial q_{t,k,l}$, $\partial S/\partial q_{t,k,l}$ and $\partial K/\partial q_{t,k,l}$ all being constant, their second partial derivatives are all zero. Therefore, from (63)

$$\frac{\partial^2 R}{\partial q_{t,k,l}\partial q_{u,v,w}} = \vec{p}^{\,T}\left(\frac{\partial K}{\partial q_{u,v,w}} \circ \frac{\partial Q_{turb}}{\partial q_{t,k,l}} + \frac{\partial Q_{turb}}{\partial q_{u,v,w}} \circ \frac{\partial K}{\partial q_{t,k,l}}\right)\vec{1}. \tag{64}$$

Each of the two matrix terms in the above equation has at most one non-zero element. This leads to a very fast implementation.

4.4 Constraints

Different types of the constraints are applied to the problem: bounds on the control variables, deterministic and probabilistic inequality constraints, and one deterministic equality constraint. The bounds determine the maximum turbine flow at each dam at each time step (65). The first set of deterministic inequality constraints defines the minimum total flow at each dam at each time step (66). The second set of inequality constraints prescribes the maximum change in turbine flow through each dam at each time step (67). The

0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

TABLE 1. River network adjacency matrix A_{out}

deterministic equality constraint defines the total system storage at the end of the simulation to some constant (68). The probabilistic inequality constraints specify the minimum (69) and maximum (70) storage levels at each dam at each time step. In equation form all constraints can be represented as follows

$$q_{i,j,1} \le q_{\max turb,i,j} \tag{65}$$
$$q_{i,j,1} + q_{i,j,2} \ge q_{\min total,i,j} \tag{66}$$
$$\Delta q_{i,j,1} \le q_{\max change,i,j} \tag{67}$$
$$\sum_j S_j(T) = \sum_j S_j(0) \tag{68}$$
$$\mathbb{P}(S_j < S_{\min,j}) < \alpha_{\min,j} \tag{69}$$
$$\mathbb{P}(S_j > S_{\max,j}) < \alpha_{\max,j}. \tag{70}$$

Here, T is time of the end of simulation. The deterministic constraints and bounds are treated in the usual way. The probabilistic constraints are taken into account with the algorithm described in subsection 3.4.

5 Case Study: Lower Columbia River

We solve the problem described in section 4 using two frameworks, in order to illustrate the superiority of the framework presented in this paper. The comparison is a safety factor approach, where a safety margin is applied to the constraints affected by uncertainty, so that deterministic optimization on mean values may be used.

5.1 Problem Parameters

In this study, we use dam and reservoir descriptions supplied by BPA. Of the ten major dams administrated by BPA, we limit this case study to the four closest to the Pacific Ocean, the Bonneville Dam, The Dalles Dam, the John Day Dam, and the McNary Dam. The system under study thus comprises four reservoirs, where the outflow from The McNary Dam enters the reservoir above the John Day Dam, and so on. The adjacency matrices A_{out} and A_{ext} from (55) are specified in tables 1 and 2. The reservoir shape linear regression coefficients from (50), estimated from the appropriate reservoir storage data, are specified in table 3. We discretize time hourly.

The model from section 4 uses steady pool routing, that is, water discharged from a dam is assumed to be immediately available at the next dam. Also, we approximate the storage curves (the relationship between water elevation and volume stored in a reservoir) for each reservoir with a linear regression to further simplify the problem.

1	0	0	0
0	1	0	0

TABLE 2. Tributary adjacency matrix A_{ext}

c_1	c_2
0.0571	68.7907
0.0403	95.8500
0.1887	76.5733
0.1000	54.2588

TABLE 3. Reservoir linear regression coefficients

We select initial conditions such that the 50-hour period begins with all reservoirs half full by volume. The terminal condition included is that the expected total water stored within the four reservoirs be equal to the initial total volume. Uncertainty is applied to a simulated inflow curve to represent inflow forecasts. The same inflow curves are used as inflows into the reservoirs behind the McNary Dam and John Day Dam. The lower two reservoirs are assumed to not have additional tributaries. We use the same simulated data as in [11], only scaled to better match the problem described here. Sample price data was used from [10].

The KL expansion is truncated based on the magnitude of the eigenvalues leaving three random variables to represent the uncertainty in the inflows.

5.2 Decision Parameters

5.2.1 Probabilistic (Robust) Case

The parametrization of the exponential utility function from (17) is problem specific. While values of $a = 0$ and $b = -1$ can be used regardless of the problem, assuming risk aversion, the value of c must be determined from the decision-maker's risk attitude toward the objective. The following is one of many possible ways to fix c.

If a change in the objective from f_1 to f_2 is worth the same increase in utility as a change in the objective from f_3 to f_4, given $|f_2 - f_1| - |f_4 - f_3|$ and $|f_2 + f_1| - |f_4 + f_3|$ both have the same sign, then c is the non-trivial solution to

$$e^{cf_1} - e^{cf_2} = e^{cf_3} - e^{cf_4}. \tag{71}$$

For the sake of the four-dam, 50 hour example used in this paper, suppose that an increase in revenue from 180000 to 360000 is worth the same increase in utility as an increase in revenue from 360000 to 720000. We substitute these values into (71).

For the α-levels, we specify all of them to be 0.05, corresponding to a 5% acceptable probability of failure for each constraint. Note that the constraints are not independent.

5.2.2 Deterministic (Safety Factor) Case

In order to make a fair comparison between our framework and a deterministic optimization, we applied a safety factor to the constraints represented in (69) and (70). These two probabilistic constraints were re-written in safety factor form as

$$S_j > S_{\min,j} + ms(S_{\max,j} - S_{\min,j}) \tag{72}$$
$$S_j < S_{\max,j} - ms(S_{\max,j} - S_{\min,j}) \tag{73}$$

with a margin of safety $ms = 0.05$.

	Safety Factor	Robust
Individual	0.1259	0.0491
	0.1198	0.0479
	0.1137	0.0478
	0.1075	0.0471
	0.1014	0.0471
	0.0953	0.0459
	0.0071	0.0458
	0.0001	0.0335
	0.0001	0.0020
		0.0011
Joint	0.1259	0.0622

TABLE 4. Failure probability estimates for individual constraints most violated, and joint failure probability estimates, for each solution

5.3 Comparison of Robust and Safety Factor Approaches

Both methods were run in Matlab on an 8-core desktop using the built-in `fmincon` interior-point optimization algorithm. The single-threaded safety factor optimization completed in 48 seconds, while the parallelized robust optimization completed in 509 seconds. Figure 1 shows a comparison between the two solutions of the expected volume in each reservoir. The expected revenues of the two solutions are 1.4372e7 for the probabilistic and 1.4295e7 for the deterministic.

Once the solutions were determined, a more accurate failure probability for each was estimated by Monte Carlo sampling. A sample size of 10^6 was simulated using each solution. Table 4 contains estimated individual failure probabilities for only those constraints with an estimate of 10^4 or greater. The last row contains the estimated joint failure probability for each solution. This clearly shows that although the robust solution has a measurable probability of violating more constraints, its joint probability of violating any constraint is much lower.

Since maximizing the expected revenue and minimizing the joint failure probability are competing objectives, the probabilistic framework results in a solution which dominates the deterministic one.

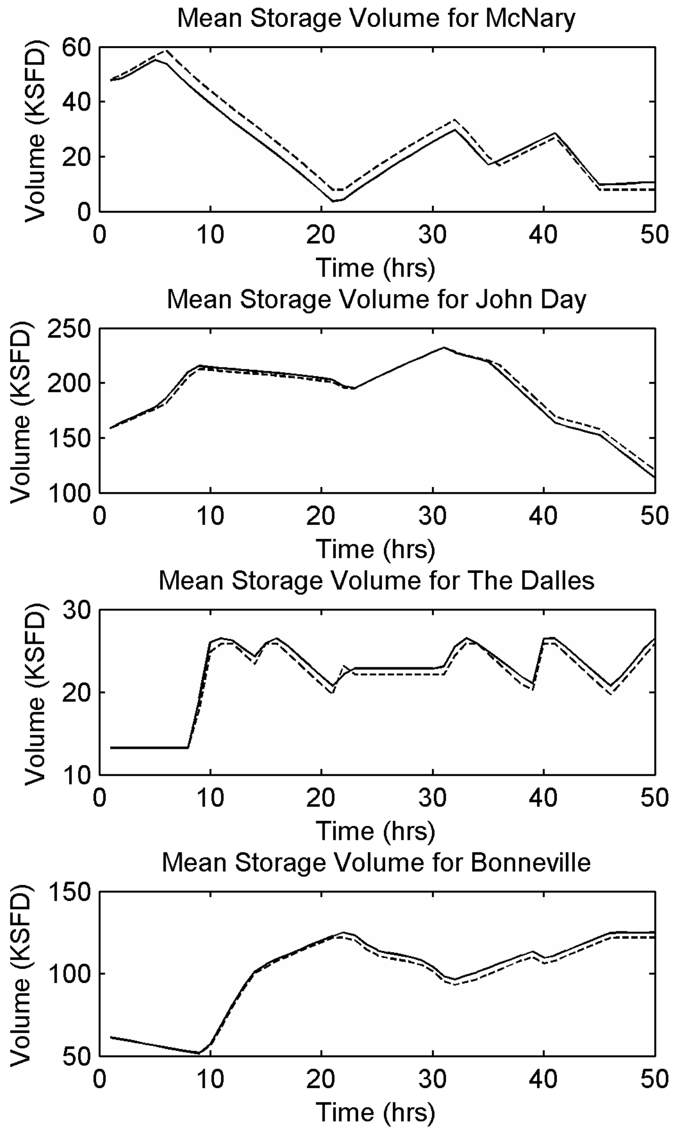

FIGURE 1. Comparison of mean storage volumes between robust (solid) and safety factor (dashed) solutions

6 CONCLUSIONS

It has been shown that the robust design objective formulation currently in use in the literature is equivalent to the maximization of expected utility, given an exponential utility function and normal random variables. However, direct application of the utility function to the performance function, rather than the application of a weighted combination of mean and variance, allows for the use of the gradient and Hessian of the performance in the robust optimization, and relaxes the restriction on symmetric variation.

Also, a complete framework for optimization of a robust objective with probabilistic constraints has been demonstrated. The application of a PC surrogate to each constraint allows for hundreds of failure probability estimations each iteration with minimal additional simulations.

A preliminary test has been done to demonstrate the viability of the method, and its superiority to deterministic approaches.

Future work includes adding more sources of uncertainty to the model, and using a more complete and realistic river network model. This will test the viability of the framework on a more nonlinear problem. The model used in this case study is a substitute for a more complete model currently in development at Oregon State University [12]. Furthermore, due to the direct application of the utility function to the objective, a non-exponential function may be used, in order to remove the assumption of constant risk aversion.

REFERENCES

[1] Beyer, H.-G., and Sendhoff, B., 2007. "Robust optimization–a comprehensive survey". *Computer methods in applied mechanics and engineering,* ***196***(33), pp. 3190–3218.

[2] Chattratichat, J., Darlington, J., Pantellides, C. C., Rustem, B., and Tanyi, B. A., 1996. "Parallel nonlinear optimisation for decision making under uncertainty". In Proceeding of the Sixth Parallel Computing Workshop, Kawasaki, Japan, November, Citeseer, pp. 12–13.

[3] Resende, C. B., Heckmann, C. G., and Michalek, J. J., 2012. "Robust design for profit maximization with aversion to downside risk from parametric uncertainty in consumer choice models". *Journal of Mechanical Design,* ***134***, p. 100901.

[4] Chen, W., 1998. "Quality utility–a compromise programming approach to robust design". PhD thesis, University of Illinois.

[5] Hazelrigg, G. A., 2012. *Fundamentals of Decision Making for Engineering Design and Systems Engineering*.

[6] Kirkwood, C. W., 1991. Notes on attitude toward risk taking and the exponential utility function.

[7] Du, X., Sudjianto, A., and Chen, W., 2004. "An integrated framework for optimization under uncertainty using inverse reliability strategy". *Journal of Mechanical Design,* ***126***, p. 562.

[8] Lee, S. H., and Chen, W., 2009. "A comparative study of uncertainty propagation methods for black-box-type problems". *Structural and Multidisciplinary Optimization,* ***37***(3), pp. 239–253.

[9] Lee, S., Chen, W., and Kwak, B., 2009. "Robust design with arbitrary distributions using gauss-type quadrature formula". *Structural and Multidisciplinary Optimization,* ***39***(3), pp. 227–243.

[10] DeLand, S., 2012. Optimization of hydroelectric flow with matlab. On the WWW. URL `http://www.mathworks.com/company/newsletters/articles/optimization-of-hydroelectric-flow-with-matlab.html`.

[11] Gibson, N., Gifford-Miears, C., Leon, A. S., and Vasylkivska, V., 2013. "Efficient computation of unsteady flow in complex river systems with uncertain inputs". *International Journal of Computer Mathematics*(just-accepted), pp. 1–18.

[12] Leon, A. S., Kanashiro, E. A., and González-Castro, J. A., 2013. "A fast approach for unsteady flow routing in complex river networks based on performance graphs". *Journal of Hydraulic Engineering,* ***139***(3), pp. 284–295.