Version of Record: https://www.sciencedirect.com/science/article/pii/S0309170822001865
Manuscript_65ec86e5a728bd33a8faf1d3eaab087e

Variational encoder geostatistical analysis (VEGAS) with an application to large scale riverine bathymetry

Mojtaba Forghani[1*], Yizhou Qian[2], Jonghyun Lee[3], Matthew Farthing[4], Tyler Hesser[4], Peter K. Kitanidis[2,5], and Eric F. Darve[1,2]

[1] Department of Mechanical Engineering, Stanford University, CA
[2] Institute for Computational and Mathematical Engineering, Stanford University, CA
[3] Department of Civil and Environmental Engineering and Water Resources Research Center, University of Hawai'i at Mānoa, Honolulu, HI
[4] U.S. Army Engineer Research and Development Center, Vicksburg, MS
[5] Department of Civil and Environmental Engineering, Stanford University, CA

Abstract

Estimation of riverbed profiles, also known as bathymetry, plays a vital role in many applications, such as safe and efficient inland navigation, prediction of bank erosion, land subsidence, and flood risk management. The high cost and complex logistics of direct bathymetry surveys, *i.e.*, depth imaging, have encouraged the use of indirect measurements such as surface flow velocities. However, estimating high-resolution bathymetry from indirect measurements is an inverse problem that can be computationally challenging. Here, we propose a reduced-order model (ROM) based approach that utilizes a variational autoencoder (VAE), a type of deep neural network with a narrow layer in the middle, to compress bathymetry and flow velocity information and accelerate bathymetry inverse problems from flow velocity measurements. In our application, the shallow-water equations (SWE) with appropriate boundary conditions (BCs), *e.g.*, the discharge and/or the free surface elevation, constitute the forward problem, to predict flow velocity. Then, ROMs of the SWEs are constructed on a nonlinear manifold of low dimensionality through a variational encoder and the bathymetry inversion problem is derived on the low-dimensional latent space in a Hierarchical Bayesian setting. The reformulation allows variational inference with a small number (e.g., $\mathcal{O}(100)$ of ROM runs and efficient uncertainty quantification. We have tested our inversion approach on a one-mile reach of the Savannah River, GA, USA. Once the neural network is trained (offline stage), the proposed technique can perform the inversion operation orders of magnitude faster than traditional inversion methods that are commonly based on linear projections, such as principal component analysis (PCA), or the principal component geostatistical approach (PCGA). Furthermore, tests show that the algorithm can estimate the bathymetry with good accuracy even with sparse flow velocity measurements.

keywords: deep learning, inverse modeling, reduced-order models, riverine bathymetry, variational encoder, Bayesian estimation

1 Introduction

High-resolution riverine bathymetry plays an essential role in many practical applications such as the study of riverine morphodynamics, safe and efficient maritime transportation, aquatic habitat management, and flood risk management [1–5]. However, direct high-resolution bathymetric surveys, commonly performed via wading or watercraft-mounted multi-beam sonar equipment [3, 6], are time consuming and costly for long river reaches [7]. Therefore, several remote sensing

*Now at Meta, email: mojif@fb.com

techniques have been used in the literature to obtain informative flow properties from which the bathymetry can be estimated [8, 9]. These include airborne bathymetric LiDAR systems [6, 10–12], multi-spectral imagery [13, 14], satellite-derived bathymetry (SDB) using Google Earth engine [15], measurement of water surface elevation [9, 16], measurement of surface velocity through GPS drifters [8] or particle image velocimetry with digital video cameras [17], and thermal imagery [18, 19]. In this work, we estimate the bathymetry using surface flow velocity measurements, since they are relatively easy to acquire at a low cost in all river conditions (*e.g.*, not sensitive to high turbidity, high or low flows). Surface velocities are sensitive to river depth (see [20, 21]), which can be captured through empirical relationships or standard hydrodynamic models like the two-dimensional shallow water equations (SWEs; [22, 23])—the partial differential equations (PDEs) that calculate depth and depth-averaged flow velocities from bathymetries and the boundary conditions (BCs).

Our focus in this paper is on real-time bathymetry estimation using a SWE model, since it can produce high-resolution estimation of the river depth using riverine hydrodynamics instead of the mean cross-sectional depth estimates obtained from empirical relationships [20]. We will assume that the underlying river bed topography is invariant and the river flow is assumed to be at quasi-steady state during the flow velocity measurement campaign (collected in a short time, *e.g.*, over a day), which are typical conditions in bathymetry estimation problems [20, 22, 23]. These flow velocity observations can then be inverted through the SWEs model in order to estimate the bathymetry, which we treat as an "inverse problem."

In principle, the riverine bathymetry identification problem is underdetermined, meaning that usually there exist multiple possible configurations of bathymetry that are consistent with the observations (*i.e.*, flow velocity measurements)—the ill-posedness of inverse problems [20]. Beyond that, the inversion process must also account for the uncertainty introduced not only by the measurement error but also by the imperfect representation of riverine hydrodynamics in the numerical SWEs model. In addition to information contained in the data, inversion techniques typically utilize prior knowledge that reflects understanding of the bathymetry in order to weigh possible solutions among those consistent with data. These solutions can be evaluated in a probabilistic way, which is commonly treated in a Bayesian framework [24] by finding the probability density of feasible solutions that satisfy both model fitting and prior information [25]. For example, prior information is described through the statistical model in which unknown properties are distributed randomly with prescribed mean and spatial covariance functions, which form a probability density function (PDF).

As the volume of available data [26] for high-resolution bathymetry imaging becomes larger, high-resolution SWEs computational models should be implemented in order to capture small-scale spatial variability of the river bottom topography. Two main computational bottlenecks arise when gradient-based optimization approaches like variational data assimilation [27] are applied to high-dimensional and/or data-intensive problems: the high cost of construction of the Jacobian (sensitivity) matrix and the computational and storage costs for the dense prior covariance matrix. In particular, the computational costs associated with the Jacobian matrix construction amount to a number of simulation model runs proportional to the number of observations (with the adjoint-state method [28]) or the number of unknowns [29], which would be the most expensive part in the bathymetry estimation problem considered here.

To address these issues, ensemble-based methods [22, 30] have been widely used for the riverine bathymetry estimation problem because of their Jacobian-free implementation. For example, [23] presents an ensemble-based bathymetry estimation approach with the Regional Ocean Modeling System (ROMS) in which the river depth is estimated using synthetic flow velocity observations.

Using the same approach, [22] assimilates velocity measurements obtained from drifters, in order to estimate riverine bathymetry. However, the success of the ensemble-based method for the riverine bathymetry problem strongly depends on the choice of the initial "guess," *i.e.*, prior mean and covariance. Furthermore, the ensemble-based method typically suffers from spurious model correlation and ensemble spread reduction that require the so-called covariance localization [20, 31], especially when insufficient ensemble sizes are being used [20]. It has been shown that the ensemble-based method requires ensembles with many realizations when assimilating a larger number of observations [20, 32].

The principal component geostatistical approach (PCGA) [20, 33, 34] is another technique that, similar to ensemble-based methods, employs low-rank approximations of the prior covariance to avoid direct Jacobian matrix computation. However, unlike ensemble-based methods that use samples from the prior distribution, PCGA uses the eigendecomposition of the covariance matrix, and thus does not require any postprocessing (*i.e.*, localization). Additionally, PCGA implements successive iterations to achieve a better linearization unlike standard ensemble-based methods that perform one-step analysis leading to a solution dependent on the prior guess of the bathymetry. However, both PCGA and ensemble-based methods inherently require multiple forward model executions, which can be computationally too time-consuming to perform in real time with expensive high-resolution numerical solvers [35]. Furthermore, due to the high-dimensionality of the bathymetry problem, these approaches usually require some assumptions about the spatial structure of the solution [20] such as a spatial correlation modelled by multivariate Gaussian kernels, which limits the capability and generalizability of the inversion problem.

A number of recent works have used deep learning (DL)-based approaches in order to address some of the inverse modeling issues mentioned above. That is, they are capable of fast predictions by bypassing expensive numerical solvers, and they are more accurate and admit more general and complex forms of the solution (*e.g.*, universal approximation theory [24]). This has led to significant interest in leveraging the strength of DL for hydrology applications [36, 37], such as flood prediction [38], water level estimation [39], land cover classification [40], water quality monitoring [41], and water resources management [42]. There has also been a significant interest in the inverse modeling community in leveraging the strength of DL-based algorithms for bathymetry estimation [36]. For example, [7] proposed the PCA-DNN algorithm that works by combining a deep neural network (DNN) with a reduced order model (ROM) based on the principal component analysis (PCA). PCA-DNN takes flow velocities with given BCs as inputs and outputs bathymetries. In [43], the training dataset was generated using synthetic bathymetry profiles and the corresponding flow velocities were computed via a numerical solver of the SWEs (the forward problem). An important issue in deterministic approaches such as PCA-DNN is that using a deterministic map from velocity to bathymetry instead of solving the inverse modeling problem with a statistical framework limits the capability of these techniques to a specific set of measurements. Furthermore, the ill-posedness of the inverse problem leads to higher sensitivity of the inverse map to measurement noise.

In this paper, we propose a technique, referred to as variational encoder geostatistical analysis (VEGAS), **that incorporates both a statistical (Bayesian) approach to the inversion and a DNN-based ROM.** Our Bayesian approach attempts to find the solution of the inverse problem through an optimization process, whose iterations are performed via a computationally cheap ROM surrogate of the forward problem [44, 45]. In particular, we use a supervised variational encoder (SVE) [46] as a ROM of the SWEs. Then, we find the maximum a posteriori (MAP) estimate of the bathymetry (the optimization process) via the optimal reduced-dimension variables (latent space) of the trained SVE (see Section 2 for more details). Performing the Bayesian steps in the

low-dimensional space (latent space) of the ROM instead of the original high-dimensional space of the bathymetry/velocity reduces the computational cost of the inversion significantly. Such variational encoder-type structures have been used previously as a part of a Bayesian framework to infer the prior distribution, for example, when modeling unsteady fluid flows [47]. Since in our approach the ROM is combined with a Bayesian framework, it is capable of providing a posteriori PDF of the bathymetry. This is unlike deterministic methods such as PCA-DNN [7], whose prediction is deterministic and based on an inverse map. Furthermore, since our Bayesian approach models the observation uncertainty explicitly as part of its framework, it is robust to the measurement noise. Our Bayesian view to the inversion problem is similar to that of PCGA [33, 34]. However, PCGA uses a linear dimension reduction technique (similar to [7] and other linear ROM-based methods), while our ROM is based on a data-driven nonlinear dimension reduction, which has the potential to be more efficient and generalizable. Furthermore, PCGA uses numerical solvers to solve the forward problem (SWE) during its iterations, while here we use a DNN-based fast surrogate solver, which is orders of magnitude faster than traditional numerical solvers. This makes the inversion process possible without access to fast graphics processing units (GPUs) and special purpose solvers [48]. VEGAS can make online predictions possible both due to the computational advantages of the DNNs and the use of a ROM, which acts as a low-dimensional surrogate of the numerical solver of the forward model. The main contributions of this paper are summarized as follows:

1. To our knowledge, this is the first paper that uses a latent space obtained from nonlinear dimension reduction for fast and accurate inversion of the shallow water equations in surface hydrology.

2. The latent space in our construction contains information about both the input (bathymetry) and output (velocity) unlike other approaches. This provides additional benefits beyond a standard supervised autoencoder approach for simultaneous construction of a ROM for both forward prediction and inversion.

3. We perform inversion in the latent space rather than simply exploiting a learned input-output relationship from the deep neural network. That is, we formulate the inverse problem in a Hierarchical Bayesian Framework and use the theoretical framework for variational inference. Previously proposed methods in other areas such as hydrogeology [e.g., 49] have commonly used ensemble-based, statistical approximation. Compared to variational inversion, ensemble approaches usually require more forward model evaluations with covariance localization or inflation [34].

The remainder of the paper is as follows. In Section 2, we provide a brief overview of the methodology used in our work, that is, the SVE (forward solver) and the Bayesian inversion process (inverse solver). In Section 3, we discuss the process being used to generate the data, such as bathymetries, BCs, and flow velocities that will be provided to the DNN. In Section 4, we provide the results of applying our inversion algorithm on different datasets and discuss the results. Finally, in Section 5 we discuss the major findings from the work and consider potential future directions.

2 Methodology

In this work, first we generate a set of synthetic data that consists of data points in the form of BCs and bathymetries as inputs and flow velocities as outputs (labeled data). Specifically, as input BCs we provide the discharge at the upstream boundary and the free-surface elevation at the downstream boundary. We then use the two-dimensional (2D) shallow-water module of the U.S. Army Corps of Engineers' adaptive hydraulics (AdH) model [50] to solve the SWEs numerically and thus generate high-fidelity training set data for the offline stage. AdH provides a stabilized finite element (FE) approximation of depths and flow velocities on unstructured 2D meshes and uses as inputs bathymetry and BCs. Note that, in agreement with our assumptions about the data collection, the lateral geometry and bathymetry of the river have also been assumed to be fixed during the simulation [22, 23]. Once the dataset is generated, we train the SVE [46] in an *offline* stage (see Section 2.2). The *online* stage then identifies the optimal latent space variables of the SVE and provides the MAP estimate and uncertainty of the bathymetry, given the flow velocity measurements and the BCs (see Section 2.1). In this work, we evaluate the performance of the VEGAS both in cases that the measurement locations are dense and sparse.

In Section 2.1, we discuss the Bayesian approach (inverse problem solver; *online* stage) used in our work and its relation to the DNN-based ROM (SVE), and then in Section 2.2 we present the SVE architecture briefly. More details of the DNN used in our work are provided in the Supplementary Information file.

2.1 Inversion algorithm

We have used a Bayesian approach in this work as the inverse problem solver, in order to reconstruct the bathymetry from flow velocity observations. The forward problem can be defined in the form of the relationship

$$\mathbf{y} = \mathbf{h}(\mathbf{s}) + \boldsymbol{\varepsilon}, \tag{1}$$

where $\mathbf{s}$ is the input (*e.g.*, bathymetry), $\mathbf{y}$ is the observation (*e.g.*, flow velocities), $\boldsymbol{\varepsilon}$ is the observation and model uncertainty noise such as a Gaussian distribution with mean zero $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, where $\mathbf{R}$ is the model/observation error matrix, and $\mathbf{h}$ is the forward map. Here, we have assumed a noise level of $r_i = 0.05$ m/s for the velocity observation, where r_i^2 are diagonal terms of the matrix $\mathbf{R}$. The inverse problem equivalent of (1) can be defined as a problem with unknown m-dimensional variable $\mathbf{s}$ (bathymetry) and n (noisy) observation $\mathbf{y}$ (flow velocities in the two directions). The Bayes' rule allows us to evaluate a posterior distribution of $\mathbf{s}$ via

$$p(\mathbf{s}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{s})p'(\mathbf{s}) = \int_{\boldsymbol{\theta}} p(\mathbf{y}|\mathbf{s})p'(\mathbf{s}|\boldsymbol{\theta})p'(\boldsymbol{\theta})d\boldsymbol{\theta}, \tag{2}$$

where $p'(\cdot)$ represents the prior probability and $\boldsymbol{\theta}$ is a set of hyperparameters that models $\mathbf{s}$ in a hierarchical Bayesian framework.

Assume that we can parameterize $\mathbf{s} = D(\mathbf{z}, \boldsymbol{\theta})$, where $\mathbf{z}$ is a k ($\ll m$)-dimensional random variable. The variable $\mathbf{z}$ in the context of the inversion problem represents the low-dimensional representation of $\mathbf{s}$, which is equivalent to the latent space variable of the ROM (see Section 2.2 for more details). In particular, we can constrain variable $\mathbf{z}$ to follow a Gaussian distribution to ensure the regularity of the latent space:

$$\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}). \tag{3}$$

We also assume that D is a deterministic map from $\mathbf{z}$ to $\mathbf{s}$ and $p'(\boldsymbol{\theta})$ follows a delta distribution:

$$\boldsymbol{\theta} = \boldsymbol{\delta}(\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}). \tag{4}$$

This allows us to rewrite (2) in the form

$$\begin{aligned} p(\mathbf{z}|\mathbf{y}) &\propto p(\mathbf{y}|\mathbf{z})p'(\mathbf{z}) \\ &\propto \exp\left(-(\mathbf{y} - \mathbf{h}(G(\mathbf{z})))^\top \mathbf{R}^{-1}(\mathbf{y} - \mathbf{h}(G(\mathbf{z})))\right) \exp\left(-\mathbf{z}^\top \boldsymbol{\Sigma}^{-1}\mathbf{z}\right), \end{aligned} \tag{5}$$

where $\mathbf{s} = G(\mathbf{z})$ is the map from the low-dimensional latent space $\mathbf{z}$ to the unknown variable space (*e.g.*, bathymetry). Finally, the MAP estimate of (5) becomes

$$\mathbf{z}_{\text{map}} = \arg\min_{\mathbf{z}} \left((\mathbf{y} - \mathbf{h}(G(\mathbf{z})))^\top \mathbf{R}^{-1}(\mathbf{y} - \mathbf{h}(G(\mathbf{z}))) + \mathbf{z}^\top \boldsymbol{\Sigma}^{-1}\mathbf{z}\right). \tag{6}$$

The MAP estimate can be determined by the Gauss-Newton method with iterative linearizations [33]. For iteration count l and step size α, we have

$$\begin{aligned} \mathbf{z}^{l+1} &= \mathbf{z}^l + \alpha \left(\boldsymbol{\Sigma}^{-1} + (\mathbf{J}^l)^\top \mathbf{R}^{-1}\mathbf{J}^l\right)^{-1} \left(\boldsymbol{\Sigma}^{-1}\mathbf{z}^l + (\mathbf{J}^l)^\top \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{h}(G(\mathbf{z}^l))\right)\right) \\ &= (1-\alpha)\mathbf{z}^l + \alpha\boldsymbol{\Sigma}\mathbf{J}^l \left(\mathbf{J}^l\boldsymbol{\Sigma}(\mathbf{J}^l)^\top + \mathbf{R}\right)^{-1} (\mathbf{y} - \mathbf{h}(G(\mathbf{z}^l)) + \mathbf{J}^l\mathbf{z}^l), \end{aligned} \tag{7}$$

where $\mathbf{z}^l$ is the value of the latent space variable $\mathbf{z}$ at the l-th iteration and $\mathbf{J}^l$ is the Jacobian of the forward map (from the latent space to flow velocities) at the lth iteration (see below). The maximum of the total number of iterations (running time limit) as well as the minimum magnitude of the gradient (convergence of $\mathbf{z}$) can be considered as the stopping criteria during the Gauss-Newton iterations.

The Jacobian $\mathbf{J}^l$ at the lth iteration is also given by

$$\mathbf{J}^l = \left.\frac{\partial \mathbf{h}(G(\mathbf{z}))}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^l} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{s}}\right|_{\mathbf{s}=G(\mathbf{z}^l)} \left.\frac{\partial \mathbf{s}}{\partial \mathbf{z}}\right|_{\mathbf{z}=\mathbf{z}^l} = \left.\mathbf{J}_{\mathbf{h}}\right|_{\mathbf{s}=G(\mathbf{z}^l)} \left.\mathbf{J}_G\right|_{\mathbf{z}=\mathbf{z}^l}. \tag{8}$$

Note that $\frac{\partial \mathbf{s}}{\partial \mathbf{z}}$ can be evaluated analytically using automatic differentiation (AD). This information can be provided at no additional computational cost in common libraries such as TensorFlow [51] and PyTorch [52]. Since the dimension of $\mathbf{z}$, $k \leq 100$, is assumed to be significantly smaller than the dimension of $\mathbf{s}$, we can also use a finite difference formulation to calculate the Jacobian matrix as an alternative to AD:

$$\begin{aligned} \mathbf{J}^l\boldsymbol{\Sigma}(\mathbf{J}^l)^\top &= \mathbf{J}^l\boldsymbol{\Sigma}^{1/2}\left(\mathbf{J}^l\boldsymbol{\Sigma}^{1/2}\right)^\top \\ \mathbf{J}^l\boldsymbol{\Sigma}^{1/2} &= \left[\mathbf{J}^l\sigma_1 e_1, \mathbf{J}^l\sigma_2 e_2, \mathbf{J}^l\sigma_3 e_3, \ldots, \mathbf{J}^l\sigma_n e_n\right] \\ \mathbf{J}^l\sigma_i e_i &\approx \sigma_i \frac{\mathbf{h}(G(\mathbf{z} + \delta_i)) - \mathbf{h}(G(\mathbf{z}))}{\delta_i}, \end{aligned} \tag{9}$$

where e_i is a unit vector with ith element equal to 1 and δ_i is the finite difference discretization value. The number of forward model runs will be $k+1$ at each iteration. Since the optimization problem in (6) becomes nonlinear because of the operator $G(\mathbf{z})$, a line-search method has also been implemented. Also, for our numerical validation purposes, we assume throughout the remainder of the paper that $\boldsymbol{\Sigma}$ is the identity matrix.

Once the iterations are converged, for the uncertainty quantification (UQ) purposes, the posterior covariance of $\mathbf{z}$ can be calculated via

$$\begin{aligned} \mathbf{Q}_{\text{post}} &= \left(\boldsymbol{\Sigma}^{-1} + \mathbf{J}^\top\mathbf{R}^{-1}\mathbf{J}\right)^{-1} \\ &= \boldsymbol{\Sigma} - \boldsymbol{\Sigma}\mathbf{J}^\top\left(\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^\top + \mathbf{R}\right)\mathbf{J}\boldsymbol{\Sigma}, \end{aligned} \tag{10}$$

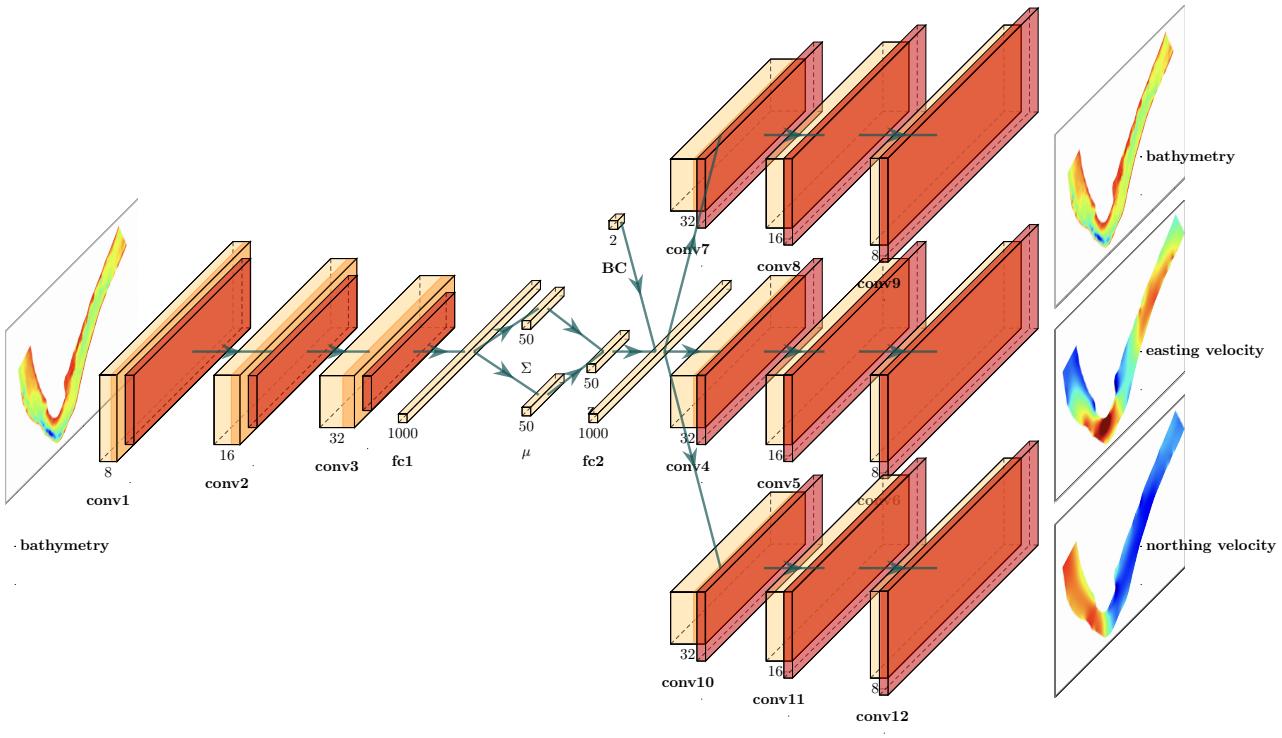
where $\mathbf{J}$ is the Jacobian, calculated either through AD or (9) at the converged $\mathbf{z}$.

2.2 ROM

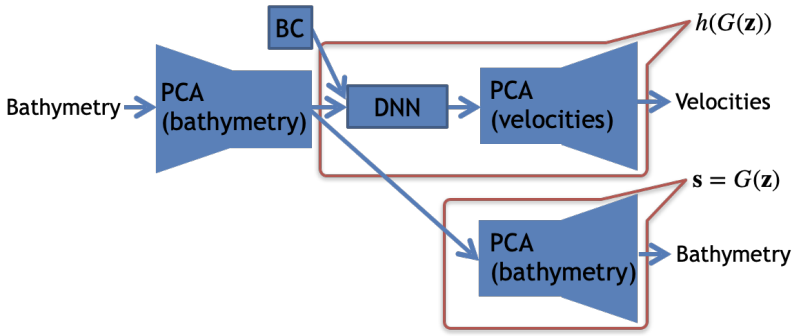
In this section, we introduce the SVE architecture, the DNN-based ROM that we used in VEGAS. The purpose of this DNN is to find the map $\mathbf{s} = G(\mathbf{z})$ from the low-dimensional latent space to the bathymetry and the map $\mathbf{h}(G(\mathbf{z}))$ from the latent space to the flow velocities (see (6)). Due to the close connection between the SVEs and variational autoencoders (VAEs), we first provide a brief overview of the VAE structure and its purpose in deep learning problems. VAEs are a class of DNNs that are primarily used for nonlinear dimension reduction in unsupervised learning [53, 54]. In VAE architectures, a high-dimensional input is fed as the input to the network; in the middle of the network (bottleneck), the dimension is reduced and two parallel layers provide the mean and variance of a multi-variate Gaussian distribution, from which the latent space variable is sampled; finally, the sampled latent space variable returns to the space of the output (which is the same as the input due to the unsupervised setup of VAE). This probabilistic structure imposes a strong regularization effect on the VAE architecture, which potentially leads to better generalization of the network [55]. The network, in its unsupervised fashion, learns a map from a dataset to itself, the so-called auto-associative neural networks (NNs). However, the strength of the VAE is in its ability to simultaneously learn a map from the input to the bottleneck layer, referred to as the encoder, as well as a map from the bottleneck layer to the output, the decoder.

While VAE has been primarily introduced as a nonlinear dimension reduction technique in unsupervised learning (input and output being the same data), its input and output can be replaced with a set of labeled datasets (such as bathymetry and flow velocity), representing a supervised DNN architecture. Such architectures are very useful for problems in which the dimension of the input and output are very large, since in general, it is very difficult to build a DNN without reducing the dimension of the layers for such problems; otherwise, the number of trainable parameters becomes very large, which leads to a DNN that is computationally very expensive to train. Furthermore, such dense networks can lead to severe overfitting without a suitably large dataset.

In this work, we use convolutional layers in the SVE structure [24] followed by a few fully connected (FC) layers that connect them to the bottleneck layer; this architecture is referred to as the convolutional SVE. Since in a convolutional network, filters are being applied to the whole input image (such as the 2D image of a riverbed profile), it reduces the size of the network significantly by taking advantage of the 2D nature of the input and homogeneity of the extracted features throughout the image. Fig. 1a shows the sketch of the SVE used in our work. The input in this figure is the bathymetry (the variable $\mathbf{s}$), while the output has three components, the flow velocity in two directions (function $\mathbf{h}(G(\mathbf{z}))$) and the bathymetry (function $\mathbf{s} = G(\mathbf{z})$). Note that the reason behind these outputs is that VEGAS requires the joint input/output distribution $(\mathbf{s} = \mathbf{h}(G(\mathbf{z})), \mathbf{y})$ (see (6)). More details on $\mathbf{h}(G(\mathbf{z}))$ and $G(\mathbf{z})$ and their relation to the inverse problem have been provided in Section 2.1. $\boldsymbol{\mu}$ in this figure is the generated mean vector, $\boldsymbol{\Sigma}$ is the generated variance, and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian distribution from which the latent variable $\mathbf{z}$ is generated ($\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$). We have also used a linear dimension reduction-based forward solver for comparison purposes in our work, in which case the dimension reduction and expansion are performed linearly via a PCA instead of nonlinearly using encoder and decoders; this is shown in Fig. 1b.



(a) SVE



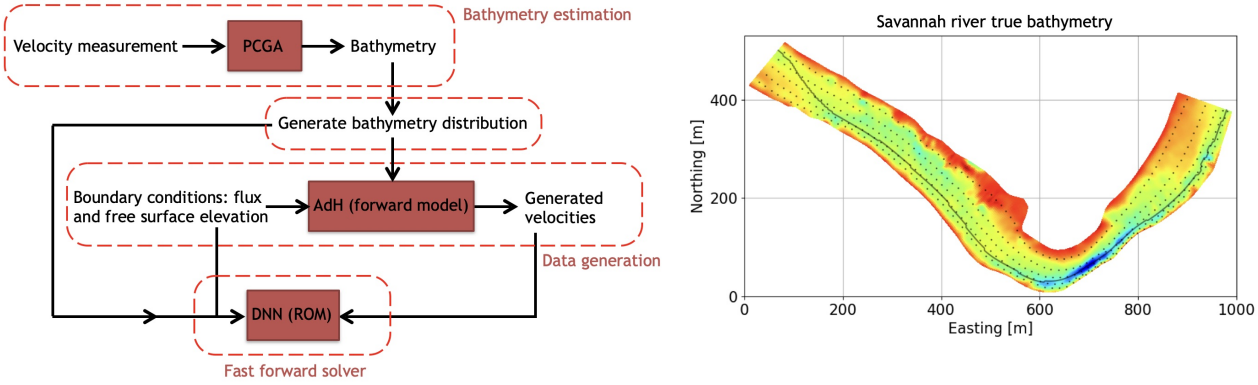
(b) PCA-based solver

Figure 1: (a) The schematic of the SVE architecture. In this approach, first, the dimension of the input (bathymetry) is reduced via an encoder, and then the low-dimensional data (the latent space) in the bottleneck layer is expanded to the output dimension. In SVE, the bottleneck layer generates a Gaussian distribution. The output of the decoder consists of both velocities and bathymetries. (b) The PCA-based solver replaces encoder/decoders with PCA, thus performing a linear dimension reduction/expansion.

3 Study site and data

In this section, we discuss the process used to obtain the training data for our DNN (the architectures in Fig. 1). Here, we are assuming that a sparse flow velocity measurement of the river of interest is available, and thus, we generate a training dataset that is relevant to this observation. Note that the VEGAS does not require the existence of such measurements at the time of the training process (offline stage). However, the purpose of this section is to propose an efficient way of generating a training dataset that takes such information (if available) into account, in order to generate a dataset that is relevant to our historical observation.

Fig. 2a shows the steps being used for generation of the dataset (training/validation/test) necessary for learning the ROM (the SVE network). At the first stage, an estimation of the bathymetry of the river is obtained from the sparse flow velocity measurement via PCGA [33, 34]. In this case, the river flow velocity measurement is performed during a short-term data collection campaign with a fixed bathymetry assumption. In order to allow our DNN to include a larger class of bathymetries into its prediction capability, we generate a bathymetry distribution from the estimated bathymetry of the PCGA (the "generate bathymetry distribution" step in Fig. 2a) via an additional sampling process based on typical river topography, before feeding them as inputs to the DNN (the SVE architecture in Fig. 1a). This allows the solver to be accurate for flow velocity prediction when future bathymetries, different from the posterior estimate, are provided as new inputs. The generated distribution along with the BCs are fed to the high-fidelity solver (the "AdH" block), which generates the data necessary for training the SVE (the "DNN" block). More detail of this process can be found in [46].



(a) (b)

Figure 2: (a) Schematic of the development of the DNN-based ROM. First, we estimate the posterior distribution of the bathymetry via the PCGA, then augment this distribution to a more general distribution in order to allow the solver to include a larger class of bathymetries into its prediction capability, and then use AdH to generate velocities. Finally, the bathymetries, BCs, and velocities are fed to the SVE which will be used as the ROM. (b) Reference (true) bathymetry of the Savannah river. The black line shows location of the thalweg, the deepest point along the river, at a given cross section. The dots are the 408 measurement locations used as inputs to the PCGA.

In this work, the VEGAS has been validated on a roughly one mile reach of the Savannah river near Augusta, GA (see Fig. 2b). First, we use AdH to generate (sparse) noisy flow velocity measurements synthetically, and then apply PCGA to these measurements. The PCGA-estimated

bathymetries are then augmented via a Gaussian kernel before providing them as inputs to the AdH. The augmentation ensures that a larger class of bathymetries (larger standard deviation than the posterior estimate of the PCGA) are included in the training set (see [46] and Supplementary Information for more details). One may use other deep learning-based priors such as Generative Adversarial Networks (GANs) in order to construct more localized topological structures, but this would require a larger, more exhaustive dataset of high-resolution riverine bathymetry profiles.

These bathymetries consist of $41 \times 501 = 20,541$ mesh nodes (501 nodes in the along-channel and 41 nodes in across-channel direction with a nominal 2.4 [m] resolution in either direction). Finally, these bathymetries along with the generated BCs—here assumed to be discharge and free-surface elevation (extracted from the United States Geological Survey (USGS) website [56]) are provided as inputs to AdH in order to obtain the flow velocities. These bathymetry/BC/flow velocity datasets are the training dataset that will be fed to the DNN in order to obtain the ROM (the SVE of Fig. 1a) (see [46] and Supplementary Information for more details).

4 Results and discussion

In this section, we provide the result of applying our inversion approach to the dataset generated in Section 3. In Section 4.1, we provide the result of inversion (bathymetry reconstruction) on the dataset of Section 3 when dense flow velocity measurements are available. We also compare the performance of VEGAS with a similar inversion approach whose DNN-based ROM uses linear dimension reduction technique [46] (see Fig. 1b). In Section 4.2, we provide similar results in the presence of sparse flow velocity measurement. And finally in Section 4.3, we present the inversion result on a test dataset that is sampled from a distribution different from the one that the SVE is trained on.

4.1 Performance of VEGAS in the presence of full measurement

In this section, we show the result of applying the inversion algorithm to the Savannah river domain. Here, we are assuming that the dense flow velocity measurement is available to the solver, therefore, $\dim(\mathbf{y}) = 20,541$ $(= 41 \times 501)$ for flow velocity in each direction. We will discuss the results in the presence of the sparse flow velocity measurement in Section 4.2. In Section 4.1.1, we study the influence of the dimension of the latent space on the inversion process and its ability to find a low-dimensional representation of the SWE dynamics.

Table 1 summarizes the root mean square errors (RMSEs) when estimating the bathymetries from the flow velocity measurements (inverse problem) using different methods (SVE and the PCA-based ROMs) in the presence of dense measurements. Here, we used 10% of our data for validation and a different set of unused 450 input-output pairs for testing. In order to have a fair comparison between different methods, we used the same latent space dimension in both methods, equal to 50 (see Section 4.1.1 for further detail regarding this choice). The errors for VEGAS are significantly lower than the PCA-based solver, due to the linear dimension reduction technique being used in the PCA-based approach. The inversion iterations with a preset maximum number of iterations of 10 takes about 1–3 min on an Intel(R) Xeon(R) CPU E5-2699 v3 @ 2.30 GHz machine with 8 cores. Fig. 3 compares performance of VEGAS with the PCA-based solver for three data points from the test set. We observe that in all cases VEGAS performs significantly better than the PCA-based approach, consistent with the result of table 1. Fig. 4 shows an example of the reference (true) and reconstructed latent space distribution for a test data point.

The figure compares the initial and final distribution of $\mathbf{z}$ with its true distribution. We observe that the algorithm has been successful in capturing the true distribution, starting from an initial random distribution.

Inversion method	Error (RMSE)		
	training set [m]	validation set [m]	test set [m]
VEGAS	**0.397**	**0.413**	**0.461**
PCA-based method	0.818	0.794	0.834

Table 1: Comparison between the reconstruction error of different inversion methods.

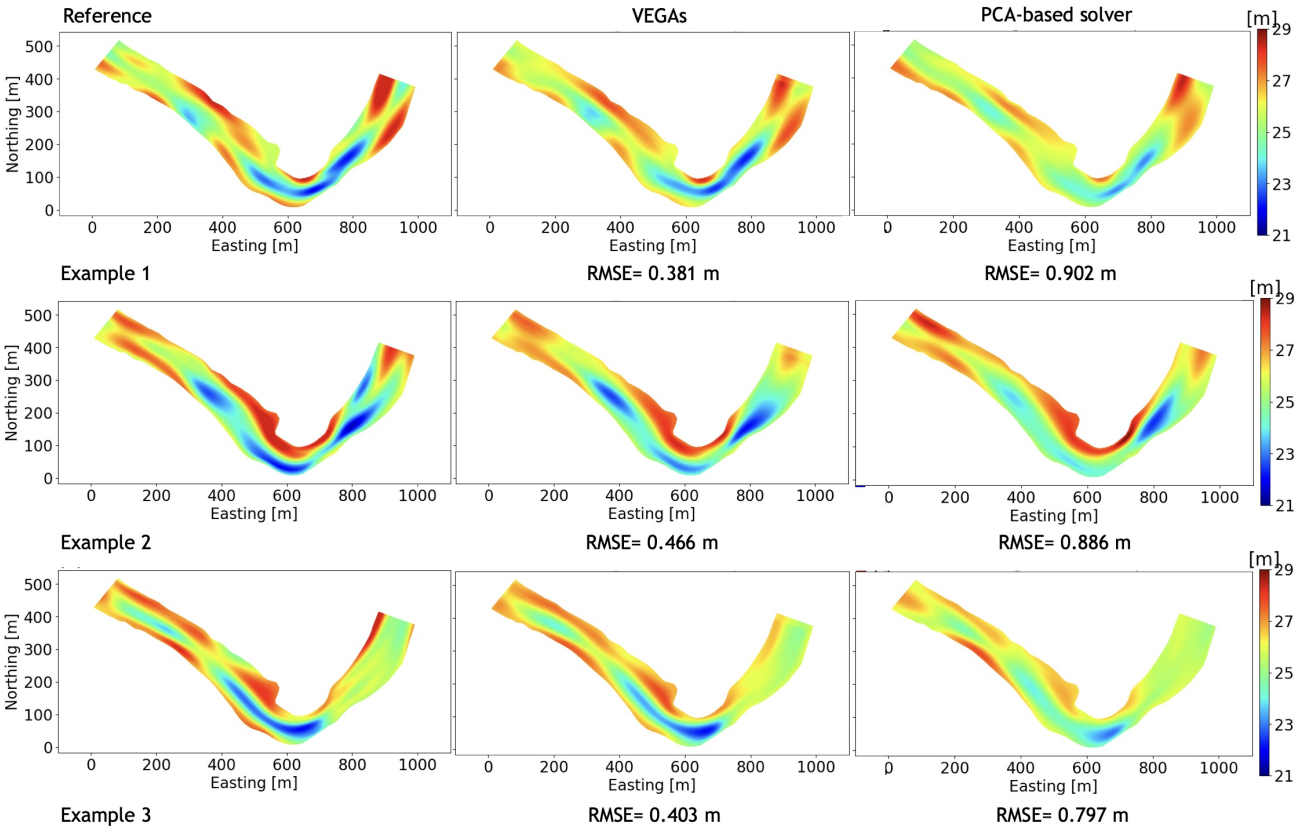


Figure 3: Comparison between the performance of VEGAS and the PCA-based solver for the bathymetry for three different test cases. Superior performance of VEGAS is noticeable in all cases. In both approaches, we chose a latent space dimension of $\dim(\mathbf{z}) = 50$.

We have also evaluated the performance of the inversion algorithm on the original reference profile for the Savannah river reach obtained from the U.S. Army Corps of Engineers survey. The result is shown in Fig. 5. We observe that the algorithm has been able to reconstruct the true profile with good accuracy. Here, we have assumed that the full flow velocity measurements are available to the inversion algorithm.

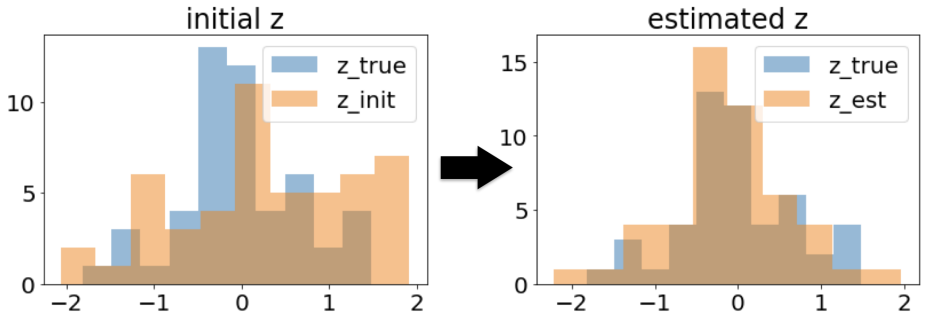


Figure 4: Initial versus final distribution of the latent space variable for a test set data point during the inversion process of VEGAS. The final estimated latent space variable matches the true distribution.

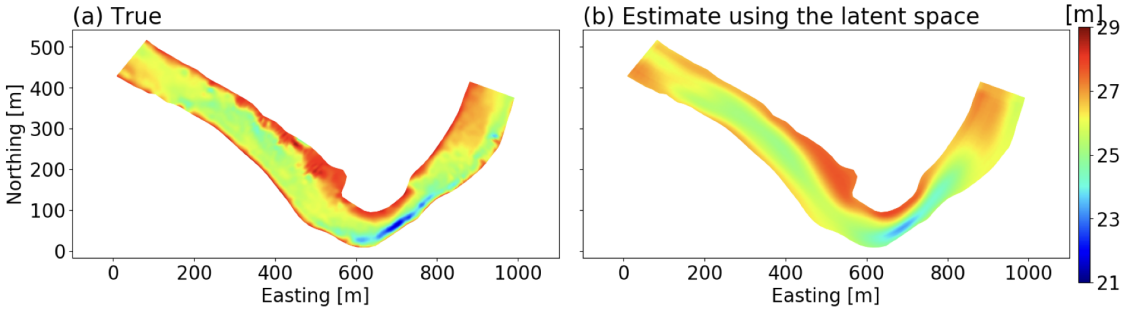
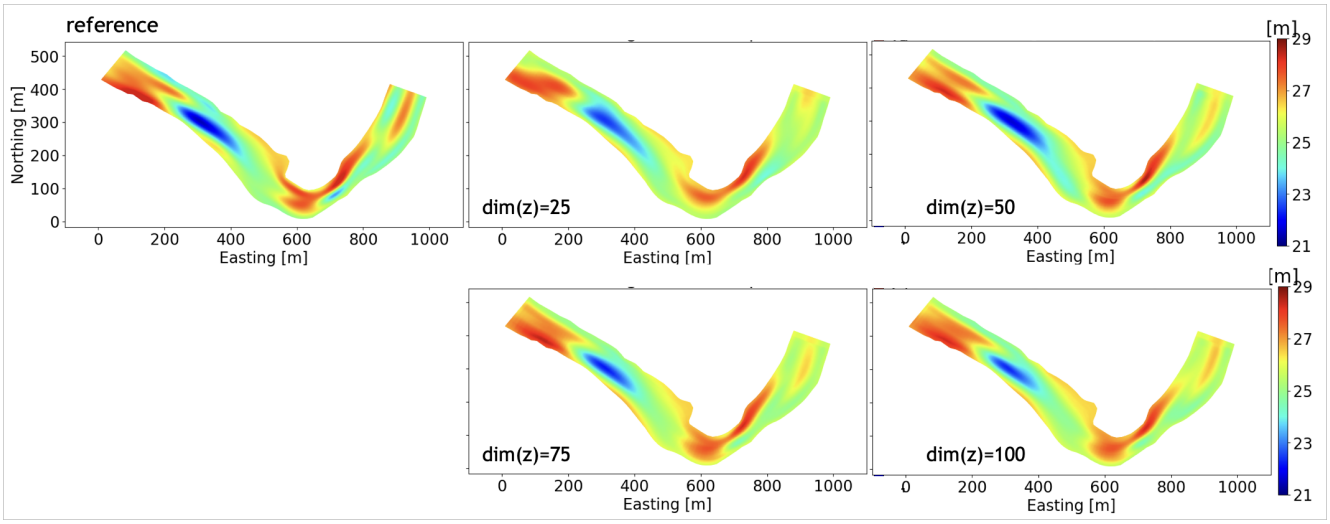


Figure 5: Reconstruction of the Savannah river profile.

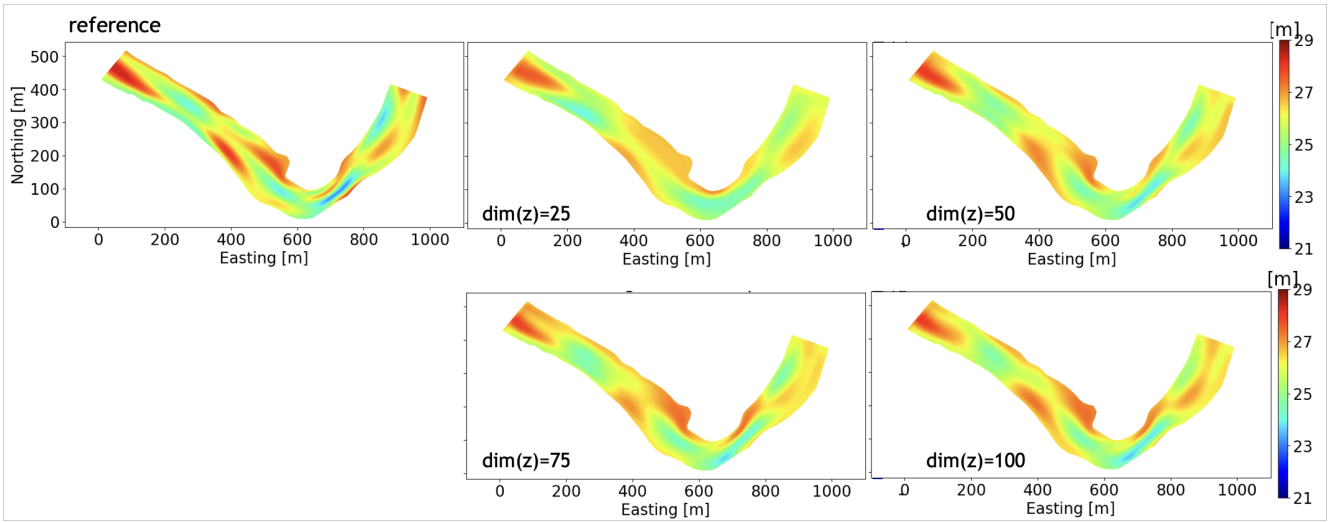
4.1.1 Latent space dimension

In this section, we study the effect of dimension of the latent space on the quality of the reconstruction and whether the dimension used in previous sections ($\dim(\mathbf{z}) = 50$) is sufficiently large to capture the dynamics of the forward/inversion problem. Fig. 6 shows examples of the bathymetry reconstruction for two different test set data points for different latent space dimensions ($\dim(\mathbf{z}) = 25, 50, 75, 100$). We observe that in general the quality of the reconstruction improves as we increase the latent space dimension. However, the improvement observed in the case of $\dim(\mathbf{z}) = 50$ compared to $\dim(\mathbf{z}) = 25$ is more significant. This trend is noticeable in the other test data points as well.

In order to better quantify the effect of latent space dimension on the reconstruction quality and, more importantly, determine the correct number of latent space dimensions to capture the dynamics of inverse and forward solvers, we also plot the RMSE of the training, validation, and test datasets for the ROM (the SVE), as well as the RMSE of the test dataset for the inversion algorithm (the VEGAS), as a function of the latent space dimension. Fig. 7 shows these results. We can observe that for the SVE, the flow velocities RMSE has stopped improving at dimensions in the range 50–70 (subfigures (a) and (b)). This dimension is found to be higher for the bathymetry (subfigure (c)), implying that a larger latent space dimension is required to represent the bathymetry accurately. However, almost no improvement is observed in the quality of the inversion problem (bathymetry reconstruction) after dimensions in the range 50–60 (subfigure (d)). This could likely be due to the fact that the VEGAS is based on, first, finding the low-dimensional representation of the velocity measurements (related to subfigures (a) and (b)), and then, expanding this representation to the bathymetry space (subfigure (c)). Therefore, although the bathymetry ROM



(a)



(b)

Figure 6: Examples of the bathymetry reconstruction on two test set data points for different latent space dimensions.

improves past $\dim(\mathbf{z}) = 50$, its influence on the reconstruction quality is not very significant, since the accuracy of the estimated $\mathbf{z}$ does not improve at larger latent space dimensions. These observations imply that the chosen latent space dimension ($\dim(\mathbf{z}) = 50$) is sufficiently large to capture the dynamics of the forward/inverse problem for the training/validation/test sets that we have considered in this work. This is also consistent with the result of Fig. 6 (improvement after $\dim(\mathbf{z}) = 50$ is negligible).

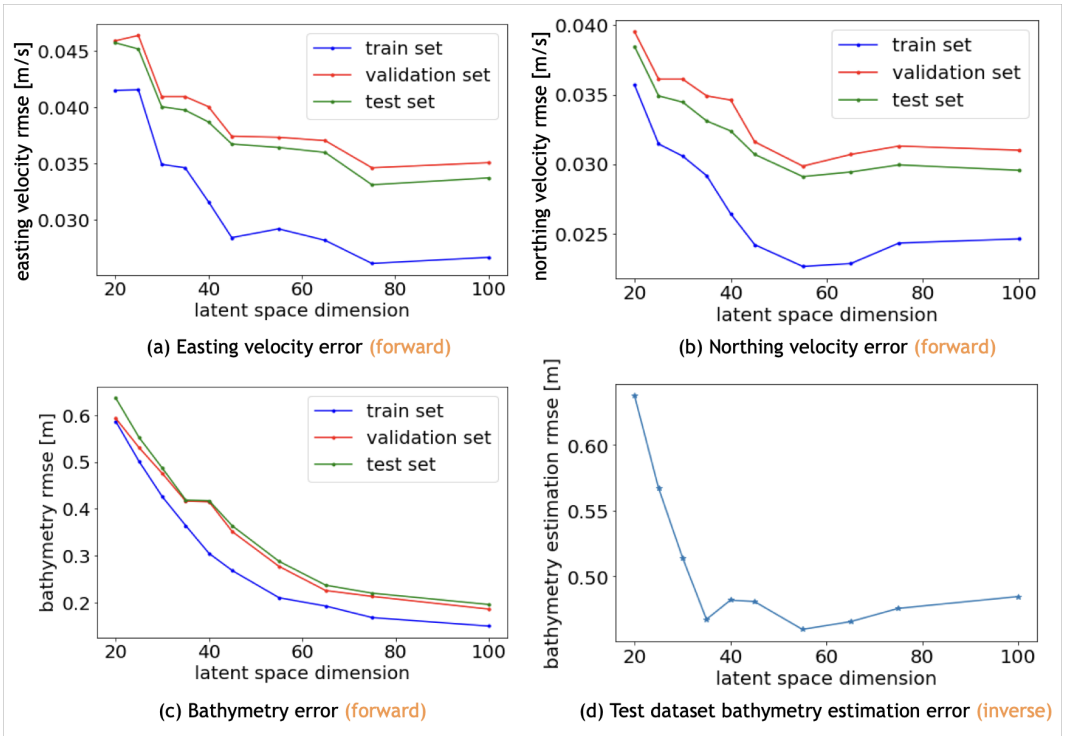


Figure 7: RMSE of the SVE prediction for different latent space dimensions for the flow velocity in the easting direction (a), northing direction (b), and the bathymetry (c). And RMSE of the inversion (VEGAS) on the test dataset (d). Although the errors decrease past $\dim(\mathbf{z})= 50$ in the case of bathymetry (subfigure (c)), no significant improvement in the VEGAS performance is observed after this point.

We have also plotted the eigendecomposition of the Hessian of the loss function with respect to the latent space components to better understand the effect of the latent space dimension. When the Hessian matrix eigenvalues reach small values, it implies the existence of directions that are not influential in the dynamics. The results are shown in Fig. 8 for the ROM (SVE) for different latent space dimensions. The plots show the eigendecompositions of the Hessian, obtained via applying singular value decomposition (SVD) to the Hessian of the different terms of the loss function, that is, velocity in easting direction, velocity in northing direction, and bathymetry, calculated with respect to the components of the latent space variable $\mathbf{z}$. Note that the loss function of the DNN (the SVE) has three terms (Fig. 1a), corresponding to its three outputs (the two flow velocities and the bathymetry). The Hessian matrices are calculated numerically using a finite difference method (see also Supplementary Information file for more detail on the loss term and Hessian calculations used in this work). We observe that for the flow velocities, the Hessian reaches zero at smaller latent space dimensions, in particular, 35 for the easting velocity and 30 for northing velocity, while for the bathymetry it happens at larger latent space dimensions, 75.

This is also consistent with our observation in Fig. 6 and Fig. 7. Reaching zero values at smaller latent space dimensions for the flow velocities implies that they require smaller number of latent space dimension compared to the bathymetry.

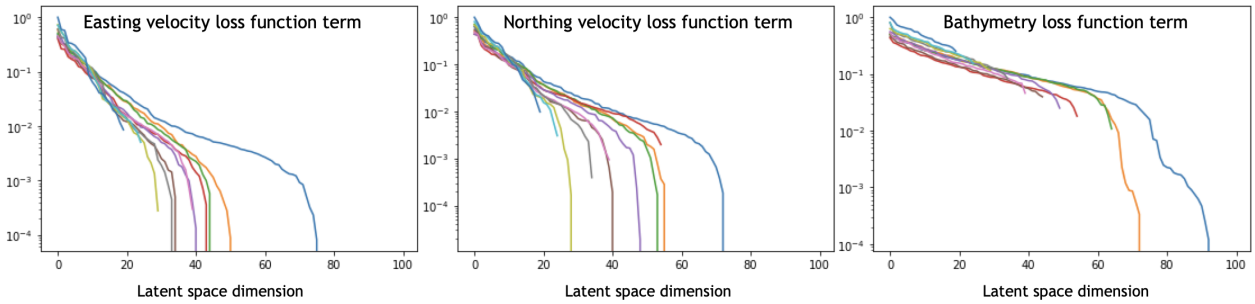


Figure 8: Eigendecomposition of the Hessian matrix for different terms of the loss function of the SVE with respect to the latent space variable. Each (colored) curve represents a SVE with a particular latent space dimension in the range dim($\mathbf{z}$)= 20–100. Decomposition of the easting velocity loss term (a), northing velocity loss term (b), and bathymetry loss term (c) for different latent space dimensions. Reaching zero values at smaller latent space dimensions for the flow velocities implies that they require fewer number of latent space dimension compared to the bathymetry.

4.2 Performance in the presence of sparse measurements

The reconstruction results that we have shown so far assumed that full flow velocity measurements are available to the inversion algorithm, that is, the assumption is that $41 \times 501 \times 2$ measurements are provided to the VEGAS, where 41 and 501 are the number of grid points in the northing and easting directions and 2 accounts for flow velocities in the easting and northing directions. Fig. 9 shows an example of a data point from the test dataset with its VEGAS prediction when full bathymetry measurement (20,541 for flow velocity in each direction), 2,000 measurement points, 500 measurement points, 200 measurement points, and 100 measurement points are available to the algorithm. The measurement points are equally distanced along and across the river (see Fig. 2b for an example of data points location). We observe that, as expected, the error increases as we use a smaller number of measurements. However, the error even with significantly sparse measurement is reasonable. Fig. 10 shows a similar comparison for the RMSE of the test dataset. Although the error increases as we use less number of measurements, we observe that even using very sparse measurements, such as the leftmost point which uses only 0.5% of the measurements (100 measurement points versus 20,541 measurement points), the reconstruction error is reasonable. This is an important property of VEGAS, since in many practical applications, flow velocity measurements only at a limited number of locations are available and therefore, accurate reconstruction of the bathymetry using such sparse measurements is an important feature of any inversion algorithm.

4.3 Performance on unseen test set distribution

The test and training datasets that we have used so far were sampled from the same distribution, the distribution explained in Section 3. However, as it was shown in Fig. 2a, our distribution was

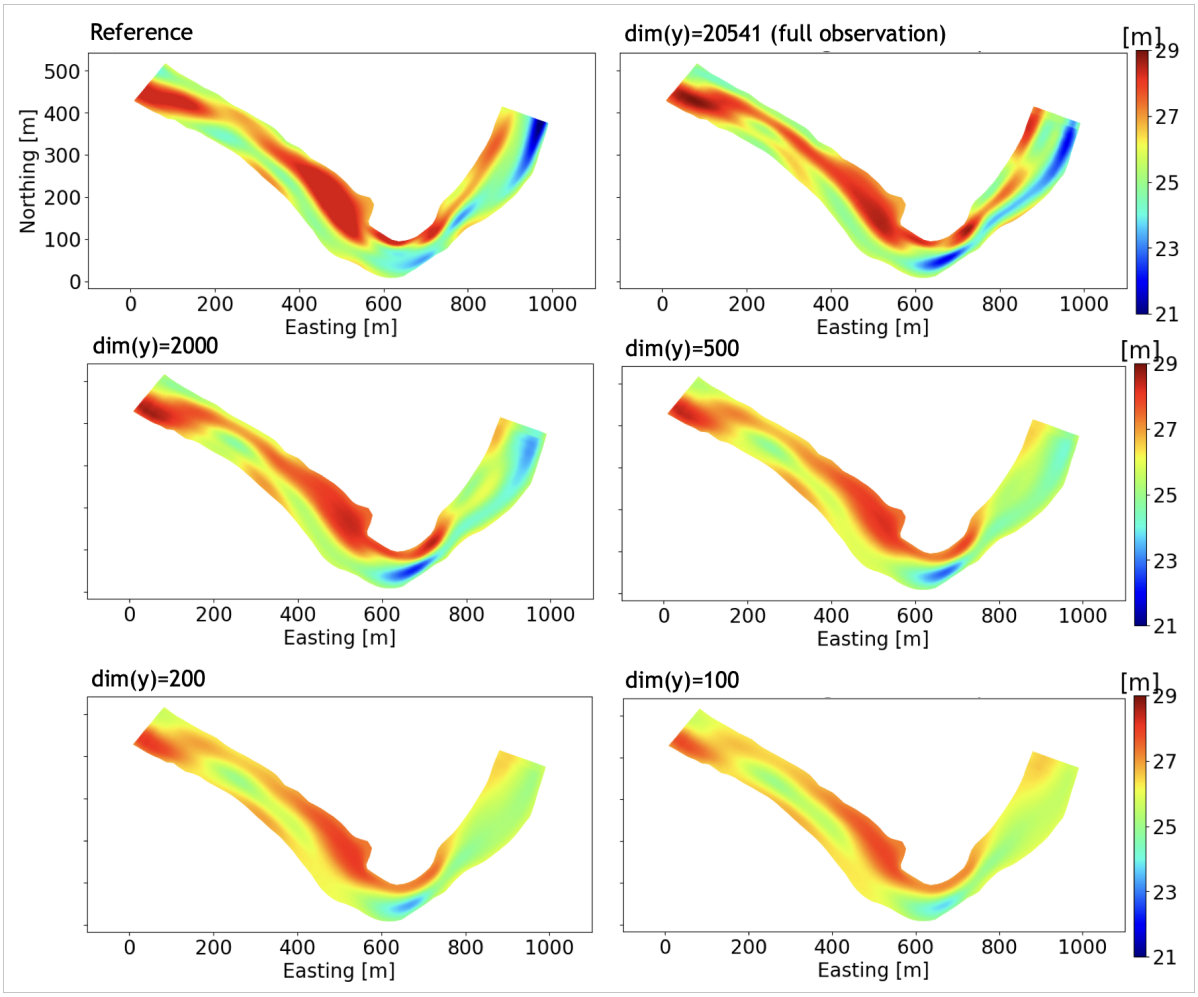


Figure 9: The effect of sparsity in flow velocity measurement on the reconstruction quality for a data point in the test dataset. The error increases as we use fewer velocity measurements $\mathbf{y}$ (see Section 2.1 for details). However, in all cases, the algorithm was mostly successful in capturing important features of the bathymetry, such as the locations with the largest and smallest depth.

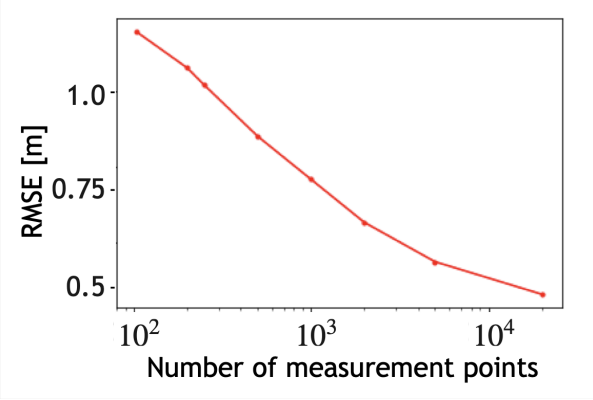


Figure 10: The effect of sparsity in the flow velocity measurements on the reconstruction quality for the test dataset.

obtained via augmenting the PCGA estimated bathymetry of the Savannah river. In this section, we are interested in evaluating the performance of VEGAS on datasets that are coming from a distribution that is different from the distribution that the DNN was trained on.

In this section, we generate our training dataset by adding **a Gaussian kernel to a parabolic mean bathymetry** and test the performance of VEGAS trained with this distribution, on the Savannah river as well as our previously used dataset (PCGA posterior with added Gaussian kernel). Note that the advantage of this approach is that the training set does not require any initial flow velocity measurement.

The results for different test set data points are shown in Fig. 11. The top figure shows the reconstruction on a test data point that was sampled from the same distribution that the training set was based on, parabolic with added Gaussian kernel. As expected, the error in this case is very low, since the test data point is sampled from the same distribution that the training set was based on. In the middle figure (Fig. 11b), the error is higher, since this data point is sampled from the previously mentioned dataset (PCGA posterior with added Gaussian kernel) which is different from the training set distribution. Finally, we show the reconstruction performance on the Savannah river. We observe that even in cases where the test data points are sampled from a distribution different from the training set distribution, the algorithm is capable of providing reasonably accurate reconstructions. The RMSE of the test set for the "parabolic + Gaussian kernel" distribution is 0.48 m and for the "PCGA posterior + Gaussian kernel" is 1.2 m. The larger RMSE of the latter is due to the fact that this test set is sampled from a distribution that is different from the training set distribution. Fig. 12 also shows an example of the reconstruction on a test data point from "PCGA posterior + Gaussian kernel" distribution (similar to Fig. 11b) with sparse flow velocity measurements (2,000 measurement points). The quality has deteriorated, as expected. However, the reconstruction was able to extract the most important features of the profile.

Although the errors in the cases that the test set distribution is different from the training set is higher than the cases that these two distributions are the same, as seen in Fig. 11, this larger error is expected as the two distributions become farther from each other. Here, we are quantifying the effect of the distance between the training and test set distributions on the test set error. We are using the normalized Mahalanobis distance for this purpose, defined as

$$d = \sqrt{\frac{(\mathbf{x} - \boldsymbol{\mu}_{\text{train}})^\top \boldsymbol{\Sigma}_{\text{train}}^{-1} (\mathbf{x} - \boldsymbol{\mu}_{\text{train}})}{m}}, \tag{11}$$

where $\mathbf{x}$ is the sample test data point, $\boldsymbol{\mu}_{\text{train}}$ is the mean of the training set distribution, $\boldsymbol{\Sigma}_{\text{train}}$ is its covariance matrix, and m is the dimension of the dataset. For the analyses considered here, $\boldsymbol{\mu}_{\text{train}}$ and $\boldsymbol{\Sigma}_{\text{train}}$ are the distribution properties of the "parabolic + Gaussian kernel" training set, while $\mathbf{x}$ can belong to either "parabolic + Gaussian kernel" (*e.g.*, Fig. 11a) or "PCGA posterior + Gaussian kernel" (*e.g.*, Fig. 11b) distribution. We have plotted the RMSE of the bathymetry reconstruction for these two test sets as a function of the distances for different input variables in Fig. 13. The vertical axis in all figures is the RMSE of the bathymetry reconstruction using VEGAS. The horizontal axes are Mahalanobis distance between data points from either of these two test sets and the training set for easting velocity (left), northing velocity (middle), and the latent space (right). As the distance of test set data points become larger (orange cluster compared to blue cluster), the RMSE also increases, as expected. By comparing the difference between RMSE and Mahalanobis distance of the mean of the two clusters, we observe that doubling distance has approximately led to RMSEs twice as large.

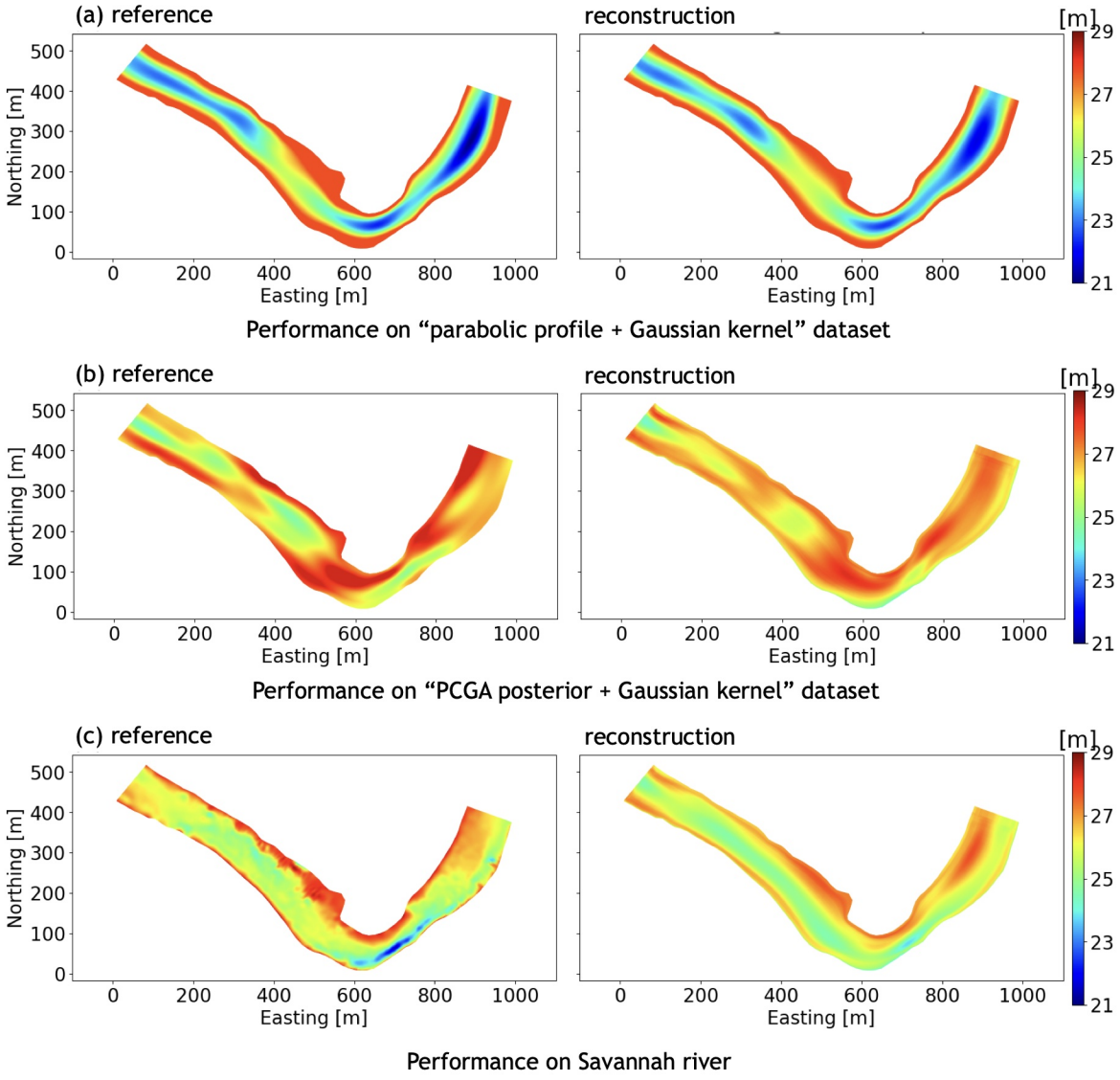


Figure 11: Performance of VEGAS trained on "parabolic + Gaussian kernel" and tested on (a) a data point from the same distribution, (b) a data point from "PCGA estimated posterior mean + Gaussian kernel" distribution, (c) Savannah river.

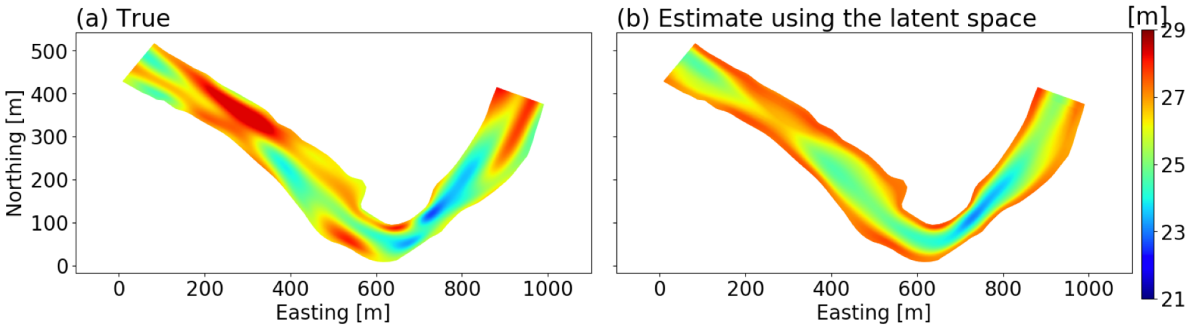


Figure 12: Performance of VEGAS trained on "parabolic + Gaussian kernel" and tested on a data point from "PCGA estimated posterior mean + Gaussian kernel" distribution with *sparse flow velocity measurements* (10% of measurements).

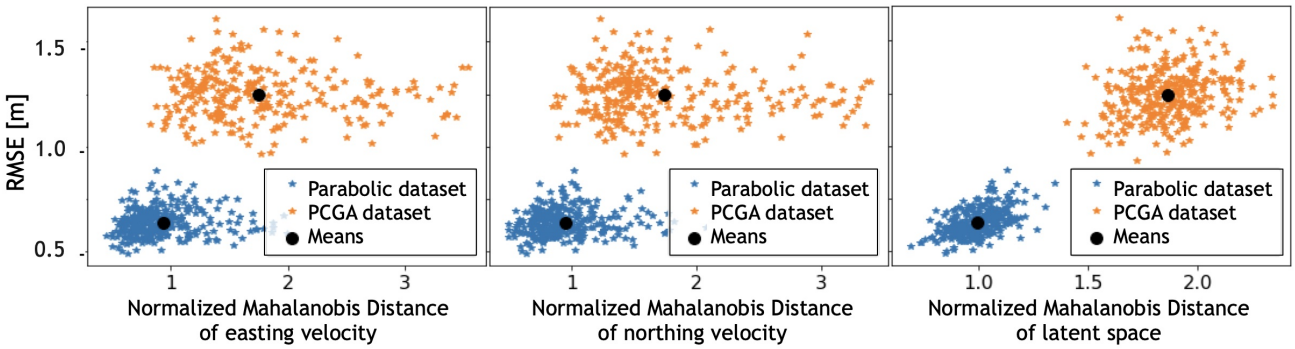


Figure 13: Comparison between the RMSE of the reconstructed bathymetries using VEGAS with the test set "parabolic + Gaussian kernel" ("Parabolic dataset" label in the figure) and "PCGA posterior + Gaussian kernel" distribution ("PCGA dataset" label in the figure) as a function of the Mahalanobis distances for easting velocity (left), northing velocity (middle), and latent space (c). "Parabolic + Gaussian kernel" is the training set for this figure.

We have plotted similar RMSE-Mahalanobis distance comparisons for test sets with the same distribution as the training set with variable standard deviations in Fig. 14. In particular, we have generated test sets with distributions similar to training set distribution, "parabolic + Gaussian kernel" distribution, with different standard deviation values, and have plotted RMSE-Mahalanobis distance plots for them, similar to the plots of Fig. 13. The standard deviation of the bathymetries in the training set is 1.2 m (the blue points), while the other test sets have the standard deviations of 2.13, 3.05, and 4.57 m. As expected, the larger the standard deviation is, the larger the RMSE in the bathymetry reconstruction. We also observe that in general, similar to the plots in Fig. 13, the ratio between RMSEs and Mahalanobis distance of flow velocities is approximately maintained (mean points in the left and middle figures).

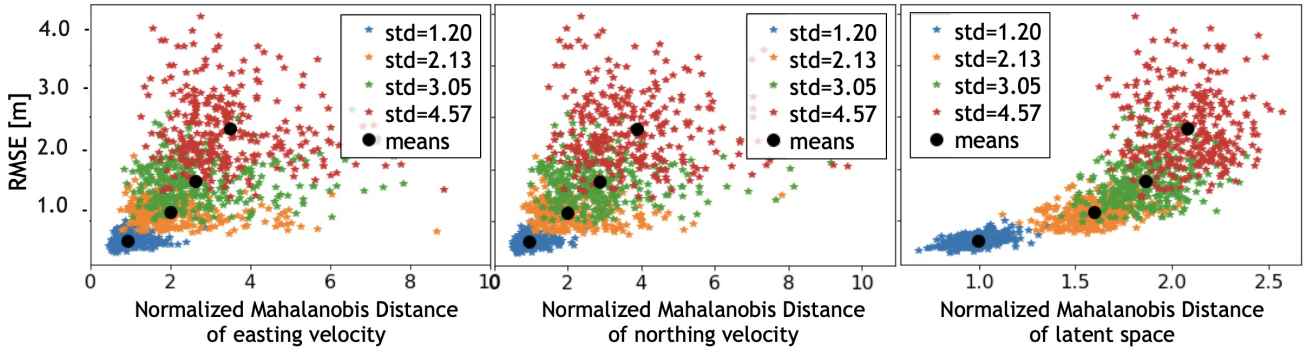


Figure 14: Comparison between the RMSE of the reconstructed bathymetries with the test set "Parabolic + Gaussian kernel" with different standard deviations as a function of the Mahalanobis distances of easting velocity (left), northing velocity (middle), and latent space (c). The training set here has a standard deviation of 1.2 m (similar to the test dataset shown in blue), and the distances are calculated with respect to this distribution.

5 Conclusion

In this work, we have presented a framework for fast estimation of riverine bathymetry from flow velocity measurements with variable boundary conditions, by combining a DNN as a ROM with a Bayesian approach for inversion. Once the neural network is trained after the offline stage, the predictions (inversion) can be done in less than a few minutes, making online bathymetry reconstruction possible. Our results demonstrate a computational efficiency about three orders of magnitude faster than common inversion algorithms such as the PCGA. Furthermore, our algorithm, VEGAS, has greater flexibility in terms of new velocity observations, due to both its nonlinear dimension reduction technique and its Bayesian viewpoint to the inversion problem.

The use of a DNN-based forward solver instead of numerical solvers like AdH has led to significant speed-ups in the inversion iteration process. Our forward solver (the SVE) is about three orders of magnitude faster than the AdH based on our previous observations [46]. Furthermore, using SVE in our DNN architecture introduces a powerful tool that captures nonlinearities present in the forward dynamics. Comparisons of VEGAS with the PCA-based solver show its significantly higher accuracy thanks to its nonlinear dimension reduction capability.

Adding a Gaussian kernel to the PCGA posterior estimate of the bathymetry (Section 3) leads to generating a training set distribution that is relevant to the river under study. However, as it was discussed and shown in Section 4.3, even when no such information is available and the training set is generated from a simple parabolic mean bathymetry profile, the inversion can produce relatively accurate bathymetries when applied to data points sampled from different unknown distributions. This is very important for construction of VEGAS models where no flow velocity/bathymetry measurements are available.

We have also shown that the inversion algorithm is capable of reasonably accurate reconstruction of the bathymetries even with sparse flow velocity measurements. This is a very important feature of the algorithm for practical applications, since in many cases, no high-resolution measurement of the flow velocity of the river is available, and instead the algorithm has to use the available (sparse) measurements to estimate the bathymetry. We have observed (Section 4.2) that even with 0.5% of the measurements, the algorithm is capable of capturing important features of the bathymetry.

The bathymetries provided to our DNNs were obtained by augmenting the posterior distribution of the PCGA inversion (the Gaussian kernel and the scaling factor). In the future, we will study incorporation of more complex and general distributions of bathymetries into our training sets, thus generalizing the forward (and consequently, inverse) solver to a larger class of bathymetries. Another assumption in this work is that the geometry of the reach was fixed. Important extensions include updating the methodology to allow for significant changes in the lateral geometry (due for example to bank overflow) and extending the approach to allow application to multiple classes of rivers.

Finally, for the cases with sparse flow velocity measurements, here we have studied situations where the measurements are available at equally-distance grid points. However, depending on the uncertainty observed in the bathymetry reconstruction, one should be able to find specific cross sections or measurement locations that are more influential in reducing the error in predictions in order to guide the collection of new flow velocity data. This would fit naturally into design of experiments approaches [57] and is an area of future interest.

Code sources

The PCGA codes can be found in the https://github.com/jonghyunharrylee/pyPCGA github repository, and the VEGAS codes can be found in the https://github.com/moji1369/VEGAS github repository.

Acknowledgements

This research was supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research under the Collaboratory on Mathematics and Physics-Informed Learning Machines for Multiscale and Multiphysics Problems (PhILMs) project, PhILMS grant DE-SC0019453. This work was also supported by an appointment to the Faculty and Postdoctoral Fellow Research Participation Program at the U.S. Engineer Research and Development Center, Coastal and Hydraulics Laboratory administered by the Oak Ridge Institute for Science and Education through an inter-agency agreement between the U.S. Department of Energy and ERDC. The Chief of Engineers has granted permission for the publication of this work.

References

1. Zolezzi, G. & Seminarao, G. Downstream and upstream influence in river meandering. Part 1. General theory and application to overdeepening. *Journal of Fluid Mechanics* **438,** 183–211 (2001).
2. Lanzoni, S., Siviglia, A., Frascati, A. & Seminara, G. Long waves in erodible channels and morphodynamic influence. *Water Resources Research* **42,** W06D17 (2006).
3. Casas, A., Benito, G., Thorndycraft, V. & Rico, M. The topographic data source of digital terrain models as a key element in the accuracy of hydraulic flood modelling. *Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group* **31,** 444–456 (2006).
4. Westaway, R., Lane, S. & Hicks, D. The development of an automated correction procedure for digital photogrammetry for the study of wide, shallow, gravel-bed rivers. *Earth Surface Processes and Landforms,* 209–226 (2000).
5. Lane, S., Richards, K. & Chandler, J. Developments in monitoring and modelling small-scale river bed topography. *Earth Surface Processes and Landforms* **19,** 349–368 (1994).
6. Marcus, W. A. Mapping of stream microhabitats with high spatial resolution hyperspectral imagery. *Journal of geographical systems* **4,** 113–126 (2002).
7. Ghorbanidehno, H. *et al.* Deep learning technique for fast inference of large-scale riverine bathymetry. *Advances in Water Resources,* 103715 (2020).
8. Emery, L. *et al.* Autonomous collection of river parameters using drifting buoys. *Oceans MTS/IEEE* **Seattle, US,** 1–7 (2010).
9. Garambois, P. A. & Monnier, J. Inference of effective river properties from remotely sensed observations of water surface. *Advances in Water Resources* **79,** 103–120 (2015).
10. Hilldale, R. C. & Raff, D. Assessing the ability of airborne LiDAR to map river bathymetry. *Earth Surface Processes and Landforms* **33,** 773–783 (2008).

11. McKean, J. *et al.* Remote sensing of channels and riparian zones with a narrow-beam aquatic-terrestrial lidar. *Remote Sensing* **1,** 1065–1096 (2009).

12. Agrafiotis, P., Skarlatos, D., Georgopoulos, A. & Karantzalos, K. DepthLearn: learning to correct the refraction on point clouds derived from aerial imagery for accurate dense shallow water bathymetry based on SVMs-Fusion with LiDAR point clouds. *Remote Sensing* **11,** 2225 (2019).

13. Misra, A., Vojinovic, Z., Ramakrishnan, B., Luijendijk, A. & Ranasinghe. Shallow water bathymetry mapping using Support Vector Machine (SVM) technique and multispectral imagery. *International Journal of Remote Sensing* **39,** 4431–4450 (2018).

14. Wang, L., Liu, H., Su, H. & Wang, J. Bathymetry retrieval from optical images with spatially distributed support vector machines. *GIScience and Remote Sensing* **56,** 323–337 (2019).

15. Traganos, D., Poursanidis, D., Aggarwal, B., Chrysoulakis, N. & Reinartz, P. Estimating Satellite-Derived Bathymetry (SDB) with the Google Earth Engine and Sentinel-2. *Remote Sensing* **10,** 859 (2018).

16. Yoon, Y. *et al.* Estimating river bathymetry from data assimilation of synthetic swot measurements. *Journal of Hydrology* **464,** 363–375 (2012).

17. Muste, M., Fujita, I. & Hauet, A. Large-scale particle image velocimetry for measurements in riverine environments. *Water Resources Research* **44,** W00D19 (2008).

18. Puleo, J. A., McKenna, T. E., Holland, K. T. & Calantoni, J. Quantifying riverine surface currents from time sequences of thermal infrared imagery. *Water Resources Research* **48,** W01527 (2012).

19. De Lima, R. L., Abrantes, J. R., de Lima, J. L. & de Lima, M. I. P. Using thermal tracers to estimate flow velocities of shallow flows: laboratory and field experiments. *Journal of Hydrology and Hydromechanics* **63,** 255–262 (2015).

20. Lee, J. *et al.* Riverine bathymetry imaging with indirect observations. *Water Resources Research* **54,** 3704–3727 (2018).

21. Neyshabur, B., Bhojanapalli, S., McAllester, D. & Srebro, N. *Exploring generalization in deep learning* in *Advances in Neural Information Processing Systems* (2017), 5947–5956.

22. Landon, C., Wilson, G. W., Ozkan-Haller, H. T. & MacMahan, J. H. Bathymetry estimation using drifter-based velocity measurements on the Kootenai river, Idaho. *Journal of Atmospheric and Oceanic Technology* **31,** 503–514 (2014).

23. Wilson, G. W. & Ozkan-Haller, H. T. Ensemble-based data assimilation for estimation of river depths. *Journal of Atmospheric and Oceanic Technology* **29,** 1558–1568 (2012).

24. Bishop, C. M. *Pattern recognition and machine learning* (Springer, 2006).

25. Kaipio, J. & Somersalo, E. Statistical inverse problems: discretization, model reduction and inverse crimes. *Journal of Computational and Applied Mathematics* **198,** 493–504 (2007).

26. Gleason, C. J. & Smith, L. C. Toward global mapping of river discharge using satellite images and at-many-stations hydraulic geometry. *Proceedings of the National Academy of Sciences of the United States of America* **111,** 4788–4791 (2014).

27. Strub, I. S., Percelay, J., Tossavainen, O. & Bayen, A. M. Comparison of two data assimilation algorithms for shallow water flows. *Networks and Heterogeneous Media* **4,** 409–430 (2009).

28. Cacuci, D. G., Ionescu-Bujor, M. & Navon, I. M. Sensitivity and uncertainty analysis. *Boca Raton, FL: Chapman and Hall* (2003).

29. Sun, N. & Sun, A. Model calibration and parameter estimation: for environmental and water resource systems. *Berlin, Germany:Springer* (2015).

30. Wilson, G. W., Ozkan-Haller, H. T. & Holman, R. A. Data assimilation and bathymetric inversion in a two-dimensional horizontal surf zone model. *Journal of Geophysical Research: Oceans* **115** (2010).

31. Hamill, T. M., Whitaker, J. S. & Snyder, C. Distance-dependent filtering of background error covariance estimates in an ensemble Kalman filter. *Monthly Weather Review* **129,** 2776–2790 (2001).

32. Li, J. Y., Kokkinaki, A., Ghorbanidehno, H., Darve, E. F. & Kitanidis, P. K. The compressed state Kalman filter for nonlinear state estimation: application to large-scale reservoir monitoring. *Water Resources Research* **51,** 9942–9963 (2015).

33. Kitanidis, P. K. & Lee, J. Principal component geostatistical approach for large dimensional inverse problems. *Water Resources Research* **50,** 5428–5443 (2014).

34. Lee, J. & Kitanidis, P. K. Large-scale hydraulic tomography and joint inversion of head and tracer data using the principal component geostatistical approach (PCGA). *Water Resources Research* **50,** 5410–5427 (2014).

35. Ghorbanidehno, H., Kokkinaki, A., Lee, J. & Darve, E. Recent developments in fast and scalable inverse modeling and data assimilation methods in hydrology. *Journal of Hydrology* **591,** 125266 (2020).

36. Poggio, T., Mhaskar, H., Rosasco, L., Miranda, B. & Liao, Q. Why and when can deep- but not shallow- networks avoid the curse of dimensionality: a review. *International Journal of Automation and Computing* **14,** 503–519 (2017).

37. Sit, M. *et al.* A Comprehensive review of deep learning applications in hydrology and water Resources. *EarthArXiv* (2019).

38. Kratzert, F., Klotz, D., Brenner, C., Schulz, K. & Herrnegger, M. Rainfall-runoff modelling using long short-term memory (LSTM) networks. *Hydrology and Earth System Sciences* **22,** 6005–6022 (2018).

39. Jeong, J. & Park, E. Comparative applications of data-driven models representing water table fluctuations. *Journal of Hydrology* **572,** 261–273 (2019).

40. Abdi, G., Samadzadegan, F. & Reinartz, P. Deep learning decision fusion for the classification of urban remote sensing data. *Journal of Applied Remote Sensing* **12,** 016038 (2018).

41. L. Li, P. J., Xu, H., Lin, G., Guo, D. & Wu, H. Water quality prediction based on recurrent neural network and improved evidence theory: a case study of Qiantang River, China. *Environmental Science and Pollution Research* **26,** 19879–19896 (2019).

42. Karimi, H. *et al.* Comparison of learning-based wastewater flow prediction methodologies for smart sewer management. *Journal of Hydrology* **577,** 123977 (2019).

43. Stefanescu, R., Sandu, A. & Navon, I. Comparison of POD reduced order strategies for the nonlinear 2D shallow water equations. *International Journal of Numerical Methods in Fluids* **76,** 497–521 (2014).

44. Allen, M., Weickum, G. & Maute, K. Application of reduced order models for the stochastic design optimization of dynamic systems. *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, USA,* 1–19 (2004).

45. Galbally, D., Fidkowski, K., Willcox, K. & Ghattas, O. Nonlinear model reduction for uncertainty quantification in large-scale inverse problems. *International Journal for Numerical Methods in Engineering* **81,** 1581–1608 (2010).

46. Forghani, M. *et al.* Application of deep learning to large scale riverine surface flow velocity estimation. *Stochastic Environmental Research and Risk Assessment* **35,** 1069–1088 (2021).

47. Akkari, N., Casenave, F., Daniel, T. & Ryckelynck, D. Data-targeted prior distribution for variational autoencoder. *Fluids* **6,** 343 (2021).

48. Crossley, A., Lamb, R. & Waller, S. Fast solution of the shallow water equations using GPU technology. *Third International Symposium of British Hydrological Society (BHS), Newcastle, London, UK* (2010).

49. Kang, X. *et al.* Hydrogeophysical characterization of nonstationary DNAPL source zones by integrating a convolutional variational autoencoder and ensemble smoother. *Water Resources Research* **57,** e2020WR028538 (2021).

50. Savant, G., Berger, C., McAlpin, T. O. & Tate, J. N. Efficient implicit finite-element hydrodynamic model for dam and levee breach. *Journal of Hydraulic Engineering* **137,** 1005–1018 (2010).

51. Van Merriënboer, B., Wiltschko, A. B. & Moldovan, D. *Tangent: automatic differentiation using source code transformation in Python* in *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA* (2017).

52. Paszke, A. *et al. Automatic differentiation in PyTorch* in *31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA* (2017).

53. Kramer, M. A. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal* **37,** 233–243 (1991).

54. Hinton, G. E. & Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science* **313,** 504–507 (2006).

55. Kingma, D. P. & Welling, M. An introduction to variational autoencoders. *Foundations and Trends in Machine Learning* **12,** 307–392 (2019).

56. *USGS 02197000 Savannah river at Augusta, GA* `https://waterdata.usgs.gov/ga/nwis/uv?site_no=02197000`.

57. Durakovic, B. Design of experiments application, concepts, examples: state of the art. *Periodicals of Engineering and Natural Sciences* **5,** 421–439 (2017).