2022 APPLICATIONS OF MATHEMATICS 23 pp
Online first

ON SURROGATE LEARNING FOR LINEAR STABILITY ASSESSMENT OF NAVIER-STOKES EQUATIONS WITH STOCHASTIC VISCOSITY

Bedřich Sousedík, Baltimore, Howard C. Elman, College Park,
Kookjin Lee, Tempe, Randy Price, Fairfax

Received February 28, 2021. Published online March 1, 2022.

Dedicated to Professor Pavel Burda on the occasion of his 75th birthday.

Abstract. We study linear stability of solutions to the Navier-Stokes equations with stochastic viscosity. Specifically, we assume that the viscosity is given in the form of a stochastic expansion. Stability analysis requires a solution of the steady-state Navier-Stokes equation and then leads to a generalized eigenvalue problem, from which we wish to characterize the real part of the rightmost eigenvalue. While this can be achieved by Monte Carlo simulation, due to its computational cost we study three surrogates based on generalized polynomial chaos, Gaussian process regression and a shallow neural network. The results of linear stability analysis assessment obtained by the surrogates are compared to that of Monte Carlo simulation using a set of numerical experiments.

Keywords: linear stability; Navier-Stokes equations; generalized polynomial chaos; stochastic collocation; stochastic Galerkin method; Gaussian process regression; shallow neural network

MSC 2020: 35R60, 65C30, 60H35

1. Introduction

Models of mathematical physics are typically based on partial differential equations and they are often solved numerically using finite element methods. The models use parameters as input data, although exact parameter values are often not known and

The work was supported by the U.S. Department of Energy Office of Advanced Scientific Computing Research, Applied Mathematics program under award DE-SC0009301 and by the U.S. National Science Foundation under awards DMS1819115 and DMS1913201.

DOI: 10.21136/AM.2022.0046-21

they are modeled using random variables. This approach leads to so-called partial differential equations with uncertain data: given stochastic parameters, we wish to characterize their stochastic solutions. Probably the most popular method for solving these problems is Monte Carlo simulation, which is based on sampling: samples of input parameters give a set of independent deterministic problems, which are solved, and then the statistical moments of solution are obtained from ensemble averaging. This method is known to be slow (with errors for n samples behaving like $n^{-1/2}$), and since each sample requires solution of the full model, its computational costs will be high. Significant effort has been devoted to designing computationally cheaper alternatives to the full model called *surrogates* in order to decrease the overall computational cost. Arguably the most popular surrogate types are based on generalized polynomial chaos (gPC) in the engineering community [15], [37], and Gaussian process (GP) regression in the statistics community [29], [34].

Our focus is on linear stability analysis of parameterized dynamical systems. A steady solution u is *stable* if with a small perturbation of u, used as initial data in a transient simulation, the simulation reverts to u; otherwise it is *unstable*. This is of fundamental importance in studying dynamics, since unstable solutions may lead to turbulent flows or other inexplicable dynamic behavior [6], [30]. *Linear stability analysis* entails computing the rightmost eigenvalue of the Jacobian matrix at u; if this eigenvalue has positive real part, then u is unstable. In this study, we explore this issue using the parameterized Navier-Stokes equations. This is a challenging task, because it entails solving a nonlinear PDE close to a bifurcation point followed by solving of a nonsymmetric eigenvalue problem. Since it is also computationally intensive, we wish to find a less expensive surrogate. The Navier-Stokes equations with stochastic viscosity were studied, e.g., by [18], [28], [32], and techniques based on gPC for parameterized eigenvalue problems were studied, e.g., by [2], [4], [19]. A stochastic collocation method for linear stability analysis was studied in [10]. Most recently, an algorithm for solving nonsymmetric eigenvalue problems with uncertain data using an embedded (intrusive) stochastic Galerkin method and the same application as in the present study was proposed in [33].

Specifically, we design and compare several surrogates. There is only a handful of studies comparing gPC approaches and GP regression, see, e.g., [25], [26], [38]. One of our goals is to contribute to the discussion with this particularly challenging problem. For the construction of the gPC surrogate we use the stochastic collocation method, and in particular the variant based on the pseudospectral (nonintrusive) stochastic Galerkin method, see [3], [36], and for the GP surrogate we use the MATLAB function `fitrgp`. We note that it seems quite common to use software packages for GP regression, and very different results among the packages have been reported [12]. Therefore, in our numerical experiments we compare both gPC and GP surrogates

with results obtained from Monte Carlo simulation. Finally, following recent trends in using neural networks for solving PDE-based models, see, e.g., [27], [31], we study a surrogate based on a shallow neural network. We compare the performance of the surrogates using two benchmark problems, and we also compare the results with those of Monte Carlo simulation.

The paper is organized as follows. In Section 2 we recall the Navier-Stokes equations and the finite element discretization, in Section 3 we discuss the linear stability of the model, in Section 4 we formulate the Navier-Stokes equations with stochastic viscosity and introduce the surrogates, in Section 5 we present results of numerical experiments, and in Section 6 we summarize our work.

2. Steady-state Navier-Stokes equations

We begin by defining the model and notation for the deterministic steady-state Navier-Stokes equations, following [11]. We wish to find velocity $\vec{u}$ and pressure p such that

$$-\nu\nabla^2\vec{u} + (\vec{u}\cdot\nabla)\vec{u} + \nabla p = \vec{f}, \tag{2.1}$$

$$\nabla\cdot\vec{u} = 0, \tag{2.2}$$

in a spatial domain D, satisfying boundary conditions

$$\vec{u} = \vec{g} \quad \text{on } \Gamma_{\text{Dir}}, \quad \nu\nabla\vec{u}\cdot\vec{n} - p\vec{n} = \vec{0} \quad \text{on } \Gamma_{\text{Neu}}, \tag{2.3}$$

where $\partial D = \overline{\Gamma}_{\text{Dir}} \cup \overline{\Gamma}_{\text{Neu}}$, $\vec{n}$ denotes the normal vector, ν denotes the kinematic viscosity and $\vec{f}$ is a vector of external forces, and we assume sufficient regularity of the data. Properties of the flow are usually characterized by the Reynolds number

$$\text{Re} = \frac{UL}{\nu}, \tag{2.4}$$

where U is a characteristic velocity and L a characteristic length.

In the mixed variational formulation of (2.1)–(2.2) we wish to find $(\vec{u}, p) \in (V_{\text{E}}, Q_{\text{D}})$ such that

$$\int_D \nu\nabla\vec{u} : \nabla\vec{v} + \int_D (\vec{u}\cdot\nabla\vec{u})\cdot\vec{v} - \int_D p(\nabla\cdot\vec{v}) = \int_D \vec{f}\cdot\vec{v} \quad \forall\, \vec{v} \in V_{\text{D}}, \tag{2.5}$$

$$\int_D q(\nabla\cdot\vec{u}) = 0 \quad \forall\, q \in Q_{\text{D}}, \tag{2.6}$$

where $(V_{\text{D}}, Q_{\text{D}})$ is a pair of spaces satisfying an inf-sup condition and V_{E} is an extension of V_{D} containing velocity vectors that satisfy the Dirichlet boundary conditions [16].

Let $c(\vec{z};\vec{u},\vec{v}) \equiv \int_D (\vec{z}\cdot\nabla\vec{u})\cdot\vec{v}$. Because problem (2.5)–(2.6) is nonlinear, it is solved using a linearization scheme in the form of Newton or Picard iteration, derived as follows.[1] Consider a solution $(\vec{u},p)$ of (2.5)–(2.6) to be given as $\vec{u} = \vec{u}^n + \delta\vec{u}^n$ and $p = p^n + \delta p^n$. Substituting into (2.5)–(2.6) and neglecting the quadratic term $c(\delta\vec{u}^n;\delta\vec{u}^n,\vec{v})$ gives

$$\int_D \nu\nabla\delta\vec{u}^n : \nabla\vec{v} + c(\delta\vec{u}^n;\vec{u}^n,\vec{v}) + c(\vec{u}^n;\delta\vec{u}^n,\vec{v}) - \int_D \delta p^n(\nabla\cdot\vec{v}) = R^n(\vec{v}), \tag{2.7}$$

$$\int_D q(\nabla\cdot\delta\vec{u}^n) = r^n(q), \tag{2.8}$$

where

$$R^n(\vec{v}) = \int_D \vec{f}\cdot\vec{v} - \int_D \nu\nabla\vec{u}^n : \nabla\vec{v} - c(\vec{u}^n;\vec{u}^n,\vec{v}) + \int_D p^n(\nabla\cdot\vec{v}), \tag{2.9}$$

$$r^n(q) = -\int_D q(\nabla\cdot\vec{u}^n). \tag{2.10}$$

Step n of the *Newton iteration* obtains $(\delta\vec{u}^n,\delta p^n)$ from (2.7)–(2.8) and updates the solution as

$$\vec{u}^{n+1} = \vec{u}^n + \delta\vec{u}^n, \quad p^{n+1} = p^n + \delta p^n. \tag{2.11}$$

Step n of the *Picard iteration* omits the term $c(\delta\vec{u}^n;\vec{u}^n,\vec{v})$ in (2.7), giving

$$\int_D \nu\nabla\delta\vec{u}^n : \nabla\vec{v} + c(\vec{u}^n;\delta\vec{u}^n,\vec{v}) - \int_D \delta p^n(\nabla\cdot\vec{v}) = R^n(\vec{v}), \tag{2.12}$$

$$\int_D q(\nabla\cdot\delta\vec{u}^n) = r^n(q). \tag{2.13}$$

Next, let us consider the discretization of (2.1)–(2.2) by a div-stable mixed finite element method, and let the bases for the velocity and pressure spaces be denoted by $\{\phi_i\}_{i=1}^{n_u}$ and $\{\varphi_i\}_{i=1}^{n_p}$, respectively, $n_u > n_p$, and let us denote by $n_x = n_u + n_p$ the number of velocity and pressure degrees of freedom. In matrix terminology, each nonlinear iteration entails solving a linear system

$$\begin{bmatrix} \mathbf{F}^n & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathbf{u}^n \\ \delta\mathbf{p}^n \end{bmatrix} = \begin{bmatrix} \mathbf{R}^n \\ \mathbf{r}^n \end{bmatrix}, \tag{2.14}$$

which corresponds to (2.7)–(2.8), followed by an update of the solution

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \delta\mathbf{u}^n, \quad \mathbf{p}^{n+1} = \mathbf{p}^n + \delta\mathbf{p}^n. \tag{2.15}$$

[1] This gives direct computation of the steady solution. It is also possible to find such solutions by integrating to steady state; see, for example [1], [21].

For Newton's method, $\mathbf{F}^n$ is the (nonsymmetric) Jacobian matrix, a sum of the vector-Laplacian matrix $\mathbf{A}$, the vector-convection matrix $\mathbf{N}^n$, and the Newton derivative matrix $\mathbf{W}^n$,

$$\mathbf{F}^n = \mathbf{A} + \mathbf{N}^n + \mathbf{W}^n, \tag{2.16}$$

where

$$\begin{aligned}
\mathbf{A} &= [a_{ab}], \quad a_{ab} = \int_D \nu \nabla \phi_b : \nabla \phi_a, \\
\mathbf{N}^n &= [n^n_{ab}], \quad n^n_{ab} = \int_D (u^n \cdot \nabla \phi_b) \cdot \phi_a, \\
\mathbf{W}^n &= [w^n_{ab}], \quad w^n_{ab} = \int_D (\phi_b \cdot \nabla u^n) \cdot \phi_a.
\end{aligned}$$

For Picard iteration, the Newton derivative matrix $\mathbf{W}^n$ is dropped, and $\mathbf{F}^n = \mathbf{A} + \mathbf{N}^n$. The matrices are sparse and n_x is typically large. The divergence matrix $\mathbf{B}$ is defined as

$$\mathbf{B} = [b_{cd}], \quad b_{cd} = \int_D \phi_d (\nabla \cdot \varphi_c). \tag{2.17}$$

The residuals $\mathbf{R}^n$ and $\mathbf{r}^n$ at step n of both nonlinear iterations are given by discretization of (2.9)–(2.10), and they are computed as

$$\begin{bmatrix} \mathbf{R}^n \\ \mathbf{r}^n \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} - \begin{bmatrix} \mathbf{P}^n & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}^n \\ \mathbf{p}^n \end{bmatrix}, \tag{2.18}$$

where $\mathbf{P}^n = \mathbf{A} + \mathbf{N}^n$ and $\mathbf{f}$ is a discrete version of the forcing function of (2.1).[2]

3. Linear stability of the Navier-Stokes equations

Following [8] let us consider, in a general setup, the dynamical system

$$\mathbf{M} u_t = f(u, \nu), \tag{3.1}$$

where $f\colon \mathbb{R}^n \times \mathbb{R} \mapsto \mathbb{R}^n$ is a nonlinear mapping, $u \in \mathbb{R}^n$ is the state variable and u_t is its time derivative, $\mathbf{M} \in \mathbb{R}^{n \times n}$ is the mass matrix, and ν is a parameter. For a fixed value of ν, linear stability of the steady-state solution is determined by the spectrum of the eigenvalue problem

$$\mathbf{J} v = \lambda \mathbf{M} v, \tag{3.2}$$

[2] We use the convention that the right-hand sides of discrete systems incorporate Dirichlet boundary data for velocities.

where $\mathbf{J} = \frac{\partial f}{\partial u}(u(\nu), \nu)$ is the Jacobian matrix of f evaluated at ν. The eigenvalues have a general form $\lambda = \alpha + \mathrm{i}\beta$, where $\alpha = \operatorname{Re}\lambda$ and $\beta = \operatorname{Im}\lambda$, and there are two cases: if $\alpha < 0$, the perturbation decays with time, and if $\alpha > 0$, the perturbation grows. Therefore, a change of stability can be detected by monitoring the rightmost eigenvalues of (3.2).

We consider a special case of (3.1), the time-dependent Navier-Stokes equations (2.1)–(2.2),

$$\begin{aligned} \vec{u}_t &= \nu\nabla^2\vec{u} - (\vec{u}\cdot\nabla)\vec{u} - \nabla p, \\ 0 &= \nabla\cdot\vec{u}, \end{aligned} \tag{3.3}$$

subject to appropriate boundary and initial conditions. Mixed finite element discretization of (3.3) gives the following Jacobian and the mass matrix, see [8] and [11], Chapter 8 for more details:

$$\mathbf{J} = \begin{bmatrix} \mathbf{F} & \mathbf{B}^\top \\ \mathbf{B} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n_x \times n_x}, \quad \mathbf{M} = \begin{bmatrix} -\mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{n_x \times n_x}, \tag{3.4}$$

where $\mathbf{F}$ is defined as in (2.16) using the steady-state solution of (3.3), $\mathbf{B}$ is defined by (2.17), and $\mathbf{G}$ is the velocity mass matrix defined as

$$\mathbf{G} = [g_{ab}], \quad g_{ab} = \int_D \phi_b \phi_a,$$

which is symmetric positive definite. Since the mass matrix $\mathbf{M}$ is singular, problem (3.2) has an infinite eigenvalue. As suggested in [5], we replace the mass matrix $\mathbf{M}$ with the nonsingular, shifted mass matrix

$$\mathbf{M}_\delta = \begin{bmatrix} -\mathbf{G} & \delta\mathbf{B}^\top \\ \delta\mathbf{B} & \mathbf{0} \end{bmatrix}, \tag{3.5}$$

which maps the infinite eigenvalues of (3.2) to δ^{-1} and leaves the finite ones unchanged. Then the generalized eigenvalue problem (3.2) can be replaced by

$$\mathbf{J}v = \lambda\mathbf{M}_\delta v. \tag{3.6}$$

Efficient methods for estimating the rightmost pair of complex eigenvalues of (3.2) (or (3.6)) were studied in [8]. Here, our goal is different. We consider parametric uncertainty in the sense that the parameter $\nu \equiv \nu(\xi)$, where ξ is a set of random variables.

4. The Navier-Stokes equations with stochastic viscosity

Let $(\Omega, \mathcal{F}, \mathcal{P})$ represent a complete probability space, where Ω is the sample space, $\mathcal{F}$ is a σ-algebra on Ω and $\mathcal{P}$ is a probability measure. We will assume that the randomness in the model is induced by a vector of independent, identically distributed (i.i.d.) random variables $\xi = (\xi_1, \ldots, \xi_{m_\xi})^\top$ such that $\xi\colon\ \Omega \to \Gamma \subset \mathbb{R}^{m_\xi}$. Let $\mathcal{B}(\Gamma)$ denote the Borel σ-algebra on Γ induced by ξ, and let ϱ denote the induced probability measure for ξ. The expected value of the product of measurable fuctions u and v that depend on ξ determines a Hilbert space $T_\Gamma \equiv L^2(\Gamma, \mathcal{B}(\Gamma), \varrho)$ with inner product

$$\langle u, v \rangle = \mathbb{E}[uv] = \int_\Gamma u(\xi) v(\xi) \varrho \, \mathrm{d}\xi, \tag{4.1}$$

where the symbol $\mathbb{E}$ denotes mathematical expectation.

In computations, we use a finite-dimensional subspace $T_P \subset T_\Gamma$ spanned by a set of polynomials $\{\psi_l(\xi)\}$ that are orthogonal with respect to ϱ, that is, $\langle \psi_k, \psi_l \rangle = \delta_{kl}$. This is referred to as the gPC basis; see [15], [37] for details and discussion. For T_P, we will use the space spanned by multivariate polynomials in $\{\xi_j\}_{j=1}^{m_\xi}$ of total degree p, which has dimension $n_\xi = \binom{m_\xi + p}{p}$. We follow the setup from [32] and assume that the viscosity ν is given by a stochastic expansion

$$\nu(\xi) = \sum_{l=1}^{n_\nu} \nu_l(x) \psi_l(\xi), \tag{4.2}$$

where $\{\nu_l(x)\}$ is a set of given deterministic spatial functions. We note that this is tantamount to taking the Reynolds number (2.4) to be stochastic.

4.1. Stochastic linear stability and Monte Carlo simulation. We are interested in a stochastic counterpart of the generalized eigenvalue problem (3.6), that is,

$$\mathbf{J}(\xi) v(\xi) = \lambda(\xi) \mathbf{M}_\delta v(\xi), \tag{4.3}$$

where $\mathbf{J}(\xi)$ is the nonsymmetric Jacobian matrix, which along with the eigenvalues $\lambda(\xi) \in \mathbb{C}$ and eigenvectors $v(\xi) \in \mathbb{C}^{n_x}$ depends on the vector ξ. The rightmost eigenvalue can be studied by Monte Carlo simulation, which entails the solution of a number of mutually independent deterministic problems at a set of sample points $\xi^{(i)}$, $i = 1, \ldots, n_{\mathrm{MC}}$. The sample points are generated randomly following the distribution of the random variables ξ, and they give realizations of the viscosity by evaluating (4.2). A realization of viscosity gives rise to deterministic functions $\vec{u}(\cdot, \xi^{(i)})$ and

$p(\cdot, \xi^{(i)})$ that satisfy the deterministic steady Navier-Stokes equations, and to finite-element approximations $\mathbf{u}^{(i)}$ and $\mathbf{p}^{(i)}$. The vector $\mathbf{u}^{(i)}$ is used to set up the Jacobian $\mathbf{J}(\xi^{(i)})$ and solving (4.3) provides a realization of the rightmost eigenvalue $\lambda(\xi^{(i)})$.

In Monte Carlo simulation this procedure is thus performed for every sample $i = 1, \ldots, n_{\mathrm{MC}}$, and the moments of the eigenvalue are obtained from ensemble averaging. We will also use the term *simulator* and denote it by η for the computer code computing the rightmost eigenvalue of (4.3) for given input parameters ξ. Since using the simulator is in general computationally expensive, we are interested in constructing an *emulator*, which is a computationally cheap *surrogate* of the full model that can be easily evaluated for any value of the input parameters. We will denote use of an emulator by $\lambda_\star(\xi) = \eta_\star(\xi)$, where the symbol $\star$ stands for any of the three approaches to emulation and surrogate construction discussed next.

4.2. Polynomial chaos surrogate. Both Monte Carlo and stochastic collocation methods are based on sampling. For stochastic collocation, the sample points $\xi^{(q)}$, $q = 1, \ldots, n_q$, consist of a set of predetermined *collocation points*. This approach derives from a methodology for performing quadrature or interpolation in multidimensional space using a small number of points, a so-called sparse grid [13], [24]. There are two ways to implement stochastic collocation, either by constructing a Lagrange interpolating polynomial, or, in the so-called pseudospectral approach, by performing a discrete projection into T_P [3], [36]. We use the second approach. In particular, we will search for expansions of the eigenvalue $\lambda(\xi)$ in the form

$$\lambda(\xi) = \sum_{k=1}^{n_\xi} \lambda_k \psi_k(\xi), \tag{4.4}$$

where $\lambda_k \in \mathbb{C}$ are coefficients corresponding to the basis $\{\psi_k\}$ defined by a discrete projection

$$\lambda_k = \langle \lambda, \psi_k \rangle, \quad k = 1, \ldots, n_\xi. \tag{4.5}$$

The coefficients in (4.4) are determined by evaluating (4.5) (see (4.1)), using numerical quadrature as

$$\lambda_k = \sum_{q=1}^{n_q} \lambda(\xi^{(q)}) \psi_k(\xi^{(q)}) w^{(q)}, \tag{4.6}$$

where $\xi^{(q)}$ are the quadrature (collocation) points and $w^{(q)}$ are quadrature weights. That is, the evaluations of coefficients in (4.5) entail solving a set of independent deterministic eigenvalue problems at a set of sample points. Details of the rule we use in our numerical experiments are discussed in Section 5, and we refer, e.g., to monograph [20] for more details.

Once the coefficients in (4.5) have been determined, the stochastic collocation emulator η_{SC} is

$$\lambda_{\mathrm{SC}}(\xi) = \eta_{\mathrm{SC}}(\xi) = \sum_{k=1}^{n_\xi} \lambda_k \psi_k(\xi). \tag{4.7}$$

See [2] for analysis showing convergence of this approximation for self-adjoint problems.

4.3. Surrogate based on Gaussian process regression. In Gaussian process regression we assume that if the process depends on $n_\star$ inputs in m_ξ dimensions, then the output is an $n_\star$-dimensional vector. Specifically, the output is modeled as

$$\lambda_{\mathrm{GP}}(\xi) = \eta_{\mathrm{GP}}(\xi) = \mu + z(\xi), \tag{4.8}$$

where we consider μ as a constant, which is also common in practice, and z is a Gaussian process to be determined. The distribution of the output is multivariate normal with mean μ. For the covariance function R we consider the so-called squared exponential kernel function, and we note that it is proportional to a correlation (or kernel) matrix C by a constant of proportionality σ_f^2 called the variance (σ_f is the standard deviation) via $R = \sigma_f^2 C$. Specifically, the correlation function C has the entries given by

$$C(\xi, \xi') = \exp\Big[-\frac{1}{2}\frac{(\xi - \xi')^\top (\xi - \xi')}{\sigma_l}\Big],$$

where σ_l is the correlation length. The prior for the simulator is

$$\eta_{\mathrm{GP}}^{\mathrm{prior}}(\xi) \sim \mathcal{N}(\mu, R(\xi, \xi)),$$

where $\mathcal{N}$ denotes the multivariate normal distribution. The parameters μ, σ_f and σ_l are estimated from the simulator runs at the experimental design points $\xi^{(t)}$, $t = 1, \ldots, n_d$, with results collected in a vector λ_{GP}^d. Let us define the correlation matrix C_d with entries $c_{ij} = C(\xi_i, \xi_j)$, where $i, j = 1, \ldots, n_d$, and let us denote by H a vector of ones with length n_d. Assuming a standard noninformative prior for variance parameters following [26], we estimate

$$\widehat{\mu} = (H^\top C_d^{-1} H)^{-1} H^\top C_d^{-1} \lambda_{\mathrm{GP}}^d, \quad \widehat{\sigma}_f = (\lambda_{\mathrm{GP}}^d - \widehat{\mu} H)^\top C_d^{-1} (\lambda_{\mathrm{GP}}^d - \widehat{\mu} H).$$

The correlation length is estimated by maximizing the logarithm of the likelihood L as

$$\widehat{\sigma}_l = \arg\max_{\sigma_l} [\log L(\sigma_l | \lambda_{\mathrm{GP}}^d)],$$

where the likelihood for the correlation length is

$$L(\sigma_l | \lambda_{\mathrm{GP}}^d) \propto (\widehat{\sigma}_f^2)^{-(n_d - n_\mu)/2} |C_d|^{-1/2} |H^\top C_d^{-1} H|^{-1/2},$$

where $|\cdot|$ is the determinant, and we use $n_\mu = 1$, since we consider constant μ in (4.8).

After the parameters have been determined, the Gaussian process emulator η_{GP} is specified by a posterior distribution, which is a Student's t-distribution with $n_d - n_\mu$ degrees of freedom

$$\eta_{\mathrm{GP}}(\xi) \sim t_{n_d - n_\mu}(M^*(\xi)|R^*(\xi,\xi)). \tag{4.9}$$

The posterior mean and covariance functions in (4.9) are defined, respectively, as

$$M^*(\xi) = \widehat{\mu} + \widehat{R}(\xi)C_d^{-1}(\lambda_{\mathrm{GP}}^d - \widehat{\mu}H),$$
$$R^*(\xi,\xi') = \frac{\widehat{\sigma}_f^2}{n_d - n_\mu - 2}[C(\xi,\xi') - \widehat{R}(\xi)C_d^{-1}\widehat{R}(\xi')^\top + Q(\xi)(H^\top C_d^{-1}H)^{-1}Q(\xi')^\top],$$

where $\widehat{R}(\xi)$ is a (row) vector of correlations between ξ and the experimental design points, and $Q(\xi) = 1 - \widehat{R}(\xi)C_d^{-1}H$. In implementation, we use the MATLAB functions `fitrgp` and `predict` with more details given in discussion of numerical experiments in Section 5. We also note that even though the emulator η_{GP} readily provides uncertainty information through the posterior distribution (4.9), we explore η_{GP} by evaluating it directly so that it is treated in a manner consistent with the other emulators η_{SC} and η_{NN}, the latter of which is discussed next.

4.4. Neural network surrogate. The final surrogate is based on a shallow (as opposed to deep) neural network with a single hidden layer and hyperbolic tangent sigmoid transfer function `tansig`, which is mathematically equivalent to tanh, see [35]. The goal is to develop an emulator

$$\lambda_{\mathrm{NN}}(\xi) = \eta_{\mathrm{NN}}(\xi),$$

based on nonlinear regression and supervised learning. The network is trained as follows. We are given a training set of inputs and targets in the form $\{\xi^{(t)}, \lambda(\xi^{(t)})\}$, $t = 1, \ldots, n_t$, and the training data is split into groups used for training, testing and validation. The neural network emulator η_{NN} is initialized randomly, and the task of the training is to produce a network that produces small errors on the training set but also responds well to additional inputs. In that case we say that the network generalizes well. The process of training a neural network entails tuning the values of the weights and biases of the network to optimize network performance by minimizing the sum of squared errors

$$\frac{1}{n_t}\sum_{t=1}^{n_t}(\lambda(\xi^{(t)}) - \lambda_{\mathrm{NN}}(\xi^{(t)}))^2.$$

The specific algorithm we use for the training is the Bayesian regularization backpropagation, in which the weight and bias values are updated according to Levenberg-

Marquardt optimization, see [7], [22] for details. In implementation, we use MATLAB functions `fitnet`, `train` and `net` with more details given in Section 5.

4.5. Validation and assessment of the surrogate models. After the surrogates are built, we would like to assess and compare their quality. Our strategy is similar to that used by [26]. Specifically, for the validation of the surrogates constructed using the emulators we used Monte Carlo simulation, for which the input parameters $\xi^{(i)}$, $i = 1, \ldots, n_{\mathrm{MC}}$, are distinct from the input parameters used to build the surrogates. The validation metric is then given by root mean square error (RMSE) defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{n_{\mathrm{MC}}} \sum_{i=1}^{n_{\mathrm{MC}}} (\lambda_\star(\xi^{(i)}) - \lambda(\xi^{(i)}))^2},$$

where the symbol $\star$ denotes any of the SC, GP or NN emulators. We used the Monte Carlo sample points $\xi^{(i)}$, $i = 1, \ldots, n_{\mathrm{MC}}$. Since RMSE represents the distance between a surrogate and the Monte Carlo simulator across the input parameters space, low RMSE values are favorable.

Next, we compute the mean and variance of each surrogate, $\mu_\star$ and $\sigma_\star$, respectively, and we estimate those provided by the emulators using empirical formulas given as

$$\mu_\star = \frac{1}{n_{\mathrm{MC}}} \sum_{i=1}^{n_{\mathrm{MC}}} \lambda_\star(\xi^{(i)}), \quad \sigma_\star = \sqrt{\frac{1}{n_{\mathrm{MC}}} \sum_{i=1}^{n_{\mathrm{MC}}} (\lambda_\star(\xi^{(i)}) - \mu_\star))^2}.$$

Although for stochastic collocation both quantities above could be calculated directly from the gPC coefficients, here we used the above formulas also with η_{SC}. Since we want to detect instability, we also use the surrogates to estimate the probability that the rightmost eigenvalue is nonnegative as

$$\Pr(\lambda_\star \geqslant 0) \approx \frac{1}{n_{\mathrm{MC}}} \sum_{i=1}^{n_{\mathrm{MC}}} \mathbf{1}(\lambda_\star(\xi^{(i)}) \geqslant 0),$$

where $\mathbf{1}$ denotes the indicator (1 or 0) function. Finally, we also test the ability of the surrogate to reconstruct the probability density function of the simulator output, which we do using a kernel density estimator with Gaussian kernel provided by the MATLAB function `ksdensity`.

R e m a r k 4.1. We note that only one of these, the neural network emulator, exactly fits within the paradigm of "machine learning" methods in the sense that it constructs a neural network. However, we view all of them as methods based on learning, in the sense that the surrogate is built from data obtained from a training set, where

for stochastic collocation the learning process is the construction of the solution at the collocation points, and for Gaussian process regression, it is the construction of the mean, variance and correlation length from the simulation at the design points.

5. Numerical experiments

We implemented the Navier-Stokes solver in MATLAB version 9.7.0.1190202 (R2019b) using the IFISS 3.5 package [9], and we tested the simulator and the emulators using two benchmark problems: flow around an obstacle and an expansion flow around a symmetric step. These are representative examples that exhibit important types of bifurcation, a Hopf bifurcation for the first (where the critical eigenvalues are a complex conjugate pair) and a pitchfork bifurcation for the second (with a real critical eigenvalue) [6], [17]. For both examples, we consider perturbations of mean viscosities that are near the values leading to bifurcations.

For the solution of the steady Navier-Stokes problem in the simulator we used a hybrid strategy in which an initial approximation is obtained from the solution of the stochastic Stokes problem, after which several steps of Picard iteration are used to improve the solution, followed by Newton iteration. The convergence test was for the Euclidean norm of the algebraic residual (2.18) to satisfy

$$\left\| \begin{bmatrix} \mathbf{R}^n \\ \mathbf{r}^n \end{bmatrix} \right\| \leqslant 10^{-8} \left\| \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \right\|.$$

Next, the eigenvalue problems (3.6), in which $\mathbf{M}_\delta$ is defined by (3.5) with $\delta = -10^{-2}$ as in [8], were solved using the function `eigs` in MATLAB. The 300 eigenvalues with the largest real part of the deterministic eigenvalue problem with mean viscosity ν_1 for each of the two examples are displayed in Figure 3. The viscosity (4.2) is parameterized using $m_\xi = 2$ random variables. For the Monte Carlo method we used 10^3 sample points generated randomly following the distribution of the random variables ξ. For stochastic collocation we used Smolyak sparse grid and grid level 4. With these settings, there were $n_q = 29$ points on the sparse grid, and this set of quadrature points was used to design all three emulators η_{SC}, η_{GP} and η_{NN}, that is $n_q = n_d = n_t$. For the GP regression (and also for the training of the neural network) we standardize the data before the regression. To this end let μ^d and σ^d denote the mean and standard deviation of the rightmost eigenvalues $\lambda(\xi^{(q)})$ calculated using the simulator at the quadrature points $\xi^{(q)}$, $q = 1, \ldots, n_q$. The data points passed to the GP regression function `fitrgp` in MATLAB are scaled as

$$\lambda(\xi^{(q)}) \leftarrow \frac{\lambda(\xi^{(q)}) - \mu^d}{\sigma^d}, \quad q = 1, \ldots, n_q, \tag{5.1}$$

and the results $\lambda_{\mathrm{GP}}(\xi)$ of the emulator function `predict` are descaled as

$$\lambda_{\mathrm{GP}}(\xi) \leftarrow \sigma^d \lambda_{\mathrm{GP}}(\xi) + \mu^d. \tag{5.2}$$

For the neural network emulator we use function `fitnet` in MATLAB to construct a neural network with one hidden layer of 20 neurons, and we set the training algorithm to use Bayesian regularization. The training parameters used in the actual training function `train` are divided in the following way: 80% for training, 10% for testing and 10% for validation. While we do not have a general strategy to find the optimal size of the neural network, we empirically tried to find as small a network as possible that would still match the Monte Carlo simulation reasonably well. We used scaling (5.1) for the training, and descaling (5.2) for the emulator predictions given by the function `net` in MATLAB.

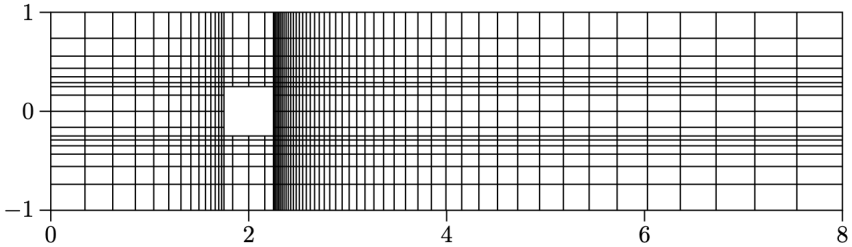

Figure 1. Finite element mesh for the flow around an obstacle problem.

5.1. Flow around an obstacle. For the first example, we consider flow around an obstacle in a similar setup as studied in [32]. The domain of the channel and the discretization are shown in Figure 1. The spatial discretization uses a stretched grid with 1008 $\boldsymbol{Q}_2 - \boldsymbol{Q}_1$ (*Taylor-Hood*) finite elements. There are 8416 velocity and 1096 pressure degrees of freedom. The viscosity $\nu(x, \xi)$ was taken to be a truncated lognormal process transformed from an underlying Gaussian process [14]. That is, $\psi_l(\xi)$, $l = 1, \ldots, n_\nu$, is a set of Hermite polynomials, which also specifies the expansion of viscosity (4.2) used in the simulator. Denoting the coefficients of the Karhunen-Loève expansion of the Gaussian process by $g_j(x)$ and $\iota_j = \xi_j - g_j$, $j = 1, \ldots, m_\xi$, the coefficients in expansion (4.2) are computed as

$$\nu_l(x) = \frac{\mathbb{E}[\psi_l(\iota)]}{\mathbb{E}[\psi_l^2(\iota)]} \exp\left[g_0 + \frac{1}{2}\sum_{j=1}^{m_\xi} (g_j(x))^2\right].$$

The covariance function of the Gaussian process, for points $X_1 = (x_1, y_1)$ and $X_2 = (x_2, y_2)$ in D, was chosen to be

$$C_{\mathrm{rf}}(X_1, X_2) = \sigma_g^2 \exp\Big(-\frac{|x_2 - x_1|}{L_x} - \frac{|y_2 - y_1|}{L_y}\Big), \tag{5.3}$$

where L_x and L_y are the correlation lengths of the random variables ξ_i, $i = 1, \ldots, m_\xi$, in the x and y directions, respectively, and σ_g is the standard deviation of the Gaussian random field. The correlation lengths were set to be equal to 25% of the width and height of the domain. The coefficient of variation CoV of the lognormal field, defined as $\mathrm{CoV} = \sigma_\nu/\nu_1$, where σ_ν is the standard deviation and ν_1 is the mean viscosity, was 1% or 10%. According to [23], in order to guarantee a complete representation of the lognormal process by (4.2) the degree of polynomial expansion of $\nu(x, \xi)$ should be twice the degree of the expansion of the solution. We follow the same strategy here. Therefore, the values of n_ξ and n_ν are, see, e.g., [15], p. 87 or [36], Section 5.2, $n_\xi = (m_\xi + p!)/(m_\xi p!)$, $n_\nu = (m + 2p)!/(m!(2p)!)$. For the gPC expansion of eigenvalues (4.4), the maximal degree of gPC expansion is $p = 3$, so then $n_\xi = 10$ and $n_\nu = 28$. We assumed that the random variables $\{\xi_l\}_{l=1}^{m_\xi}$ follow a normal distribution and used Smolyak sparse grid with Gauss-Hermite quadrature points for collocation. For the solution of the Navier-Stokes problem we used the hybrid strategy with 6 steps of Picard iteration followed by at most 15 steps of Newton iteration. We used mean viscosity $\nu_1 = 5.36193 \times 10^{-3}$, which corresponds to Reynolds number $\mathrm{Re} = 373$, and the rightmost eigenvalue pair is $0.0085 \pm 2.2551\mathrm{i}$, see the left panel in Figure 3. Table 1 presents the results of validation and assessment of the surrogates using the indicators from Section 4.5. It is evident that for both CoV 1% and 10% the values of RMSE are small for all surrogates with the smallest value for the stochastic collocation, where we note that we used the same values of $\xi^{(i)}$ in the Monte Carlo simulation and also for sampling the gPC surrogate (4.7). All values of μ and σ are in close agreement, and in particular, all values of RMSE are smaller than the corresponding values of μ (and σ) by at least two orders of magnitude. Also, all emulators indicate reliably the probability of the rightmost eigenvalue being nonnegative. Finally, Figure 4 displays the probability density function (pdf) estimates of the rightmost eigenvalue. The estimates were obtained using MATLAB function `ksdensity` for sampled gPC expansions. In all cases, we see an excellent agreement of the plots in the left panel corresponding to $\mathrm{CoV} = 1\%$ and in the right panel corresponding to $\mathrm{CoV} = 10\%$.

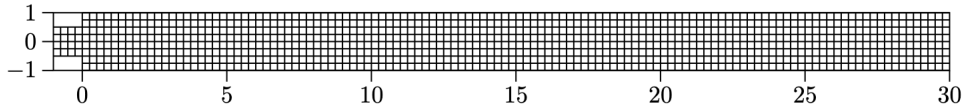

Figure 2. Finite element mesh for the expansion flow around a symmetric step.

5.2. Expansion flow around a symmetric step. For the second example, we consider an expansion flow around a symmetric step. The domain and its discretization are shown in Figure 2. The spatial discretization uses a uniform grid with 976 $\boldsymbol{Q}_2 - \boldsymbol{P}_{-1}$ finite elements, which provide a stable discretization for the rectangular

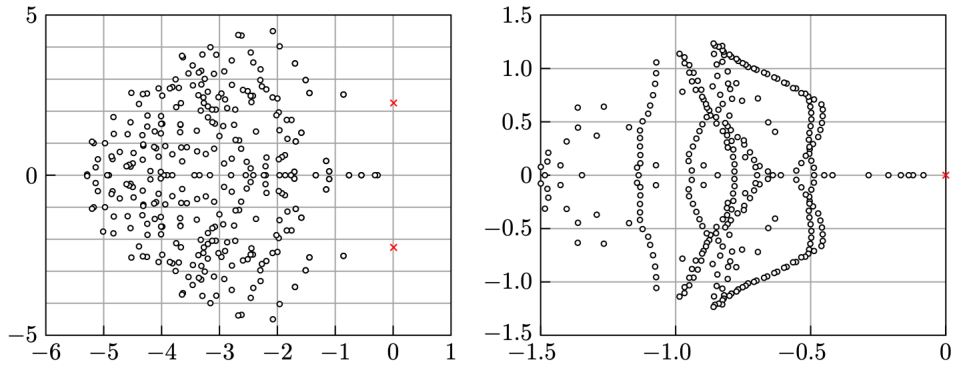

Figure 3. An image of the complex plane and 300 eigenvalues with the largest real part of the deterministic eigenvalue problem with mean viscosity (i.e., $\nu = \nu_1$ in (4.2)) for the two examples: flow around an obstacle (left) and expansion flow around a symmetric step (right). The rightmost eigenvalues are indicated by a red cross.

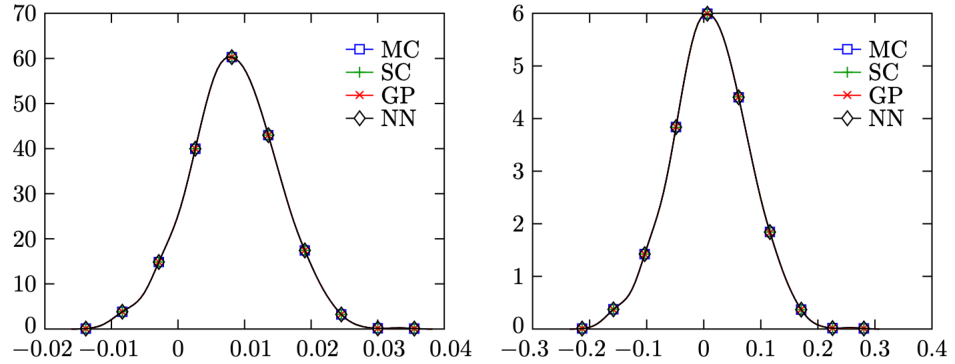

Figure 4. Plots of the pdf estimate of the real part of the rightmost eigenvalue obtained using Monte Carlo (MC), stochastic collocation (SC), Gaussian process regression (GP) and neural network (NN) for the flow around an obstacle with CoV = 1% (left) and CoV = 10% (right).

grid [11], p. 139. There are 8338 velocity and 2928 pressure degrees of freedom. For the viscosity we considered a random field with affine dependence on the random variables ξ given as

$$\nu(x,\xi) = \nu_1 + \sigma_\nu \sum_{l=2}^{n_\nu} \nu_l(x)\xi_{l-1}, \tag{5.4}$$

where ν_1 is the mean and $\sigma_\nu = \text{CoV} \cdot \nu_1$ the standard deviation of the viscosity, $n_\nu = m_\xi + 1$, and $\nu_{l+1} = \sqrt{3\lambda_l}v_l(x)$ with $\{(\lambda_l, v_l(x))\}_{l=1}^{m_\xi}$ are the eigenpairs of the eigenvalue problem associated with the covariance kernel of the random field. As in the previous example, we used the values CoV = 1% and 10%. We considered the

	MC	SC	GP	NN
		CoV = 1%		
RMSE	-	4.1859×10^{-8}	2.1709×10^{-6}	5.0301×10^{-7}
μ	8.3579×10^{-3}	8.3579×10^{-3}	8.3571×10^{-3}	8.3579×10^{-3}
σ	6.5356×10^{-3}	6.5356×10^{-3}	6.5355×10^{-3}	6.5356×10^{-3}
$\Pr(\lambda \geqslant 0)$		89.8%		
		CoV = 10%		
RMSE	-	9.4232×10^{-5}	3.9827×10^{-4}	2.9063×10^{-5}
μ	1.1279×10^{-2}	1.1277×10^{-2}	1.1235×10^{-2}	1.1277×10^{-2}
σ	6.5819×10^{-2}	6.5818×10^{-2}	6.5789×10^{-2}	6.5813×10^{-2}
$\Pr(\lambda \geqslant 0)$	56.5%		56.4%	56.5%

Table 1. Flow around an obstacle: validation of the surrogate models by Monte Carlo (MC) simulation using root mean square error (RMSE), their assessment using estimates of the mean μ, standard deviation σ, and probability that the rightmost eigenvalue is nonnegative. The surrogates are based on stochastic collocation (SC), Gaussian process regression (GP) and neural network (NN), and the measures are defined in Section 4.5.

covariance kernel (5.3), with correlation lengths set to 12.5% of the width and 25% of the height of the domain. We assumed that the random variables $\{\xi_l\}_{l=1}^{m_\xi}$ follow a uniform distribution over $(-1, 1)$. Note that (5.4) can be viewed as a special case of (4.2), which consists of only linear terms of ξ. For the parametrization of viscosity by (5.4), which then specifies the simulator, we used the same stochastic dimension m_ξ and degree of polynomial expansion p as in the previous example: $m_\xi = 2$ and $p = 3$, so then $n_\xi = 10$ and $n_\nu = m_\xi + 1 = 3$. We used a Smolyak sparse grid with Gauss-Legendre quadrature points for collocation. For the solution of the Navier-Stokes problem we used the hybrid strategy with 20 steps of Picard iteration followed by at most 20 steps of Newton iteration. We used mean viscosity $\nu_1 = 4.5455 \times 10^{-3}$, which corresponds to Reynolds number $\mathrm{Re} = 220$, and the rightmost eigenvalue is 5.7963×10^{-4} (the second largest eigenvalue is -8.2273×10^{-2}), see the right panel in Figure 3. Table 2 presents the results of validation and assessment of the surrogates using the indicators from Section 4.5. The trends are similar to those for the flow around an obstacle problem. For both CoV 1% and 10% the corresponding values of μ and σ are in close agreement. The values of RMSE are small for all surrogates and again, they are smaller than the corresponding values of μ (and σ) by at least two orders of magnitude. Finally, Figure 5 displays the probability density function (pdf) estimates of the rightmost eigenvalue. We note that both pdf estimates in this figure are "narrower" comparing to the pdf estimates for flow around an obstacle in Figure 4. Nevertheless there is an excellent agreement of all estimates in both left and right panels.

	MC	SC	GP	NN
		CoV = 1%		
RMSE	-	8.8129×10^{-10}	4.0545×10^{-7}	1.5824×10^{-8}
μ	5.7982×10^{-4}	5.7982×10^{-4}	5.7987×10^{-4}	5.7982×10^{-4}
σ	2.9150×10^{-4}	2.9150×10^{-4}	2.9151×10^{-4}	2.9149×10^{-4}
$\Pr(\lambda \geqslant 0)$	98.4%		98.5%	98.4%
		CoV = 10%		
RMSE	-	2.6183×10^{-7}	2.7106×10^{-6}	4.2076×10^{-7}
μ	4.9677×10^{-4}	4.9676×10^{-4}	4.9711×10^{-4}	4.9685×10^{-4}
σ	2.9048×10^{-3}	2.9048×10^{-3}	2.9050×10^{-3}	2.9048×10^{-3}
$\Pr(\lambda \geqslant 0)$		57.5%		

Table 2. Expansion flow around a symmetric step: validation of the surrogates by Monte Carlo (MC) simulation using root mean square error (RMSE), their assessment using estimates of the mean μ, standard deviation σ, and probability that the rightmost eigenvalue is nonnegative. The surrogates are based on stochastic collocation (SC), Gaussian process regression (GP) and neural network (NN), and the measures are defined in Section 4.5.

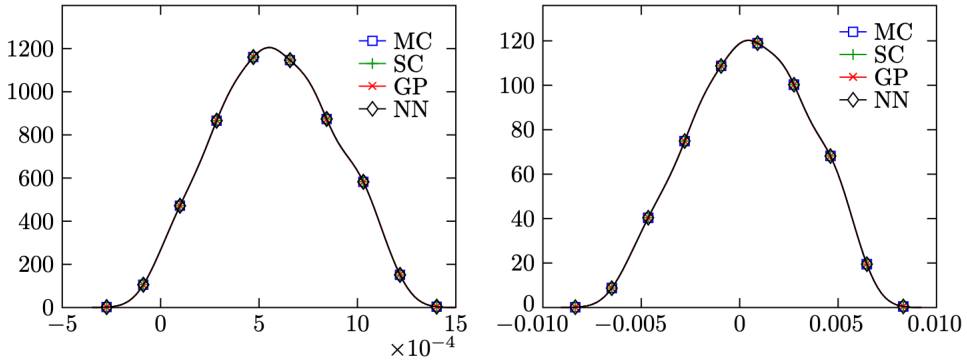

Figure 5. Plots of the pdf estimate of the real part of the rightmost eigenvalue obtained using Monte Carlo (MC), stochastic collocation (SC), Gaussian process regression (GP) and neural network (NN) for the expansion flow around a symmetric step with CoV = 1% (left) and CoV = 10% (right).

Computational time. We briefly mention our experience with running the MATLAB functions on a MacBook Pro laptop with a 3.5 GHz Intel Core i7 processor and 16 GB RAM. The computation of the rightmost eigenvalue for one sample of ξ using the simulator took at least 30s, depending on the value of ξ and settings of the inner solvers for the nonlinear iteration and call of the eigenvalue solver. On the other hand, a run of the emulators to evaluate the three surrogates took only be-

tween 0.02s and 0.04s for all 10^3 sample points, which were used for validation and assessment. The learning part (construction of an emulator) took 0.18s in the case of η_{GP} using the function `fitrgp`, and 1.25s in the case of η_{NN} using the function `train`. The construction of η_{SC} was implemented as a part of the simulator, however it can be seen, comparing (4.6) to (4.7), that if $n_q \approx n_\xi$, the construction of η_{SC} is inexpensive, and in particular the timings of the construction of η_{SC} and its use are similar. Finally, we note that all three emulators were trained using only $n_q = 29$ samples that require run of the simulator. Therefore, since the overhead associated with the training and use of the emulators is very small, the computational savings provided by the emulators are dramatic.

m_ξ	1	2	3	4	5
n_ξ	4	10	20	35	56
n_q	4	29	69	137	241

Table 3. Sizes of the gPC bases n_ξ and numbers of the quadrature points n_q for stochastic dimensions m_ξ and gPC degree $p = 3$.

MC	μ		8.8125×10^{-3}	
	σ		7.1136×10^{-3}	
	$\Pr(\lambda \geqslant 0)$		89.7%	
GP	n_d ($\approx n_d/n_q$)	6 ($\approx 20\%$)	8 ($\approx 30\%$)	29 (100%)
	RMSE	5.7386×10^{-6}	3.8676×10^{-6}	2.4706×10^{-6}
	μ	8.8134×10^{-3}	8.8103×10^{-3}	8.8117×10^{-3}
	σ	7.1124×10^{-3}	7.1135×10^{-3}	7.1135×10^{-3}
	$\Pr(\lambda \geqslant 0)$		89.7%	
NN	n_t ($\approx n_t/n_q$)	6 ($\approx 20\%$)	8 ($\approx 30\%$)	29 (100%)
	RMSE	6.1469×10^{-3}	7.7102×10^{-5}	1.4824×10^{-7}
	μ	13.6704×10^{-3}	8.8029×10^{-3}	8.8125×10^{-3}
	σ	4.8058×10^{-3}	7.1469×10^{-3}	7.1135×10^{-3}
	$\Pr(\lambda \geqslant 0)$	100%	89.4%	89.7%
SC	n_q		29	
	RMSE		3.0072×10^{-8}	
	μ		8.8125×10^{-3}	
	σ		7.1136×10^{-3}	
	$\Pr(\lambda \geqslant 0)$		89.7%	

Table 4. Effect of reducing the number of training points on the GP and NN surrogates for the flow around an obstacle problem with the channel of length 12 and with $m_\xi = 2$. The same quantities are used as in Table 1, and they were defined in Section 4.5.

MC	μ		8.7886×10^{-3}	
	σ		9.3753×10^{-3}	
	$\Pr(\lambda \geqslant 0)$		82.4%	
GP	n_d ($\approx n_d/n_q$)	13 ($\approx 5\%$)	25 ($\approx 10\%$)	241 (100%)
	RMSE	1.7554×10^{-3}	1.2636×10^{-5}	1.2535×10^{-5}
	μ	8.8913×10^{-3}	8.7898×10^{-3}	8.7914×10^{-3}
	σ	9.1899×10^{-3}	9.3740×10^{-3}	9.3742×10^{-3}
	$\Pr(\lambda \geqslant 0)$	83.0%	82.4%	
NN	n_t ($\approx n_t/n_q$)	13 ($\approx 5\%$)	25 ($\approx 10\%$)	241 (100%)
	RMSE	7.6909×10^{-3}	9.3273×10^{-5}	5.1869×10^{-6}
	μ	9.3639×10^{-3}	8.7848×10^{-3}	8.7886×10^{-3}
	σ	1.7564×10^{-3}	9.3215×10^{-3}	9.3717×10^{-3}
	$\Pr(\lambda \geqslant 0)$	100%	82.3%	82.4%
SC	n_q		241	
	RMSE		1.7987×10^{-7}	
	μ		8.7886×10^{-3}	
	σ		9.3754×10^{-3}	
	$\Pr(\lambda \geqslant 0)$		82.4%	

Table 5. Effect of reducing the number of training points on the GP and NN surrogates for the flow around an obstacle problem with the channel of length 12 and with $m_\xi = 5$. The same quantities are used as in Table 1, and they were defined in Section 4.5.

5.3. Effect of larger stochastic dimensions. We also studied the effect of reducing the number of training (or design) points for the Gaussian process (GP) regression and neural network (NN) surrogates using a problem with increasing stochastic dimension. We do not drop any quadrature (collocation) points from the stochastic collocation (SC) method, since it would yield an incorrect quadrature rule. In particular, we considered the flow around an obstacle problem in a similar setup as in Section 5.1 except with a channel of length 12 (instead of 8, cf. Figure 1). There are then $12,640$ velocity and 1640 pressure degrees of freedom, and the rightmost eigenvalue corresponding to the problem with the mean viscosity is a pair $0.0090 \pm 2.2550\mathrm{i}$. We considered a sequence of stochastic dimensions $m_\xi = 2, 3, 4, 5$. Sizes of the gPC bases and numbers of the quadrature points are given in Table 3. Other settings were the same as in Section 5.1. We selected a fraction of the quadrature points to train the two surrogates for each of the stochastic dimensions in order to test the robustness in training of the Gaussian process regression and neural network surrogates. For example, we selected every 10th quadrature point to be included in the training set, so that

then the ratio $n_t/n_q = 10\%$. Tables 4 and 5 summarize the results for $m_\xi = 2$ and $m_\xi = 5$, respectively. From Table 4 it can be seen that by using only 6 training points, i.e., reducing the ratio n_t/n_q to approximately 20%, the GP surrogate already provides relatively a quite accurate estimate compared to the results of the Monte Carlo simulation, whereas the results of the NN surrogate are not satisfactory. Increasing the number of the training points to 8 leads to a dramatic improvements of the NN surrogate. Nevertheless, by including all quadrature points into the training set, the approximation provided by the NN appears to be slightly more accurate then the one provided by the GP regression, but overall the most accurate is the result provided by the stochastic collocation. The same trends can be observed also from Table 5 for the case with $m_\xi = 5$, except that in this case only approximately 5% of the quadrature points are needed for the GP regression to provide a reasonable surrogate, and approximately 10% are needed for the NN. Therefore it appears that either of the GP or NN surrogates may provide an attractive alternative to the stochastic collocation for the high-dimensional problems.

6. Conclusion

We studied linear stability of Navier-Stokes equations with stochastic viscosity. This leads to a generalized eigenvalue problem, and we are interested in characterization of the rightmost eigenvalue. We designed three emulators for constructing the rightmost eigenvalue surrogate. The first surrogate was based on generalized polynomial chaos, and it was constructed using stochastic collocation, or its pseudospectral variant (sometimes called nonintrusive stochastic Galerkin method), which uses integration on Smolyak sparse grid and numerical quadrature. For the second and third surrogates we used functions available in MATLAB. The second surrogate was based on Gaussian process regression, and we used function `fitrgp`. The third surrogate was based on shallow neural network, and we used function `fitnet` with Bayesian Regularization backpropagation. We found that the set of quadrature points used for the generalized polynomial chaos surrogate is also suitable for training the other two emulators (based on Gaussian processes and neural network), and we also found that certain scaling of the learning data points, and subsequent descaling of the predictions, proposed by these emulators, improves the quality of the surrogates. Finally, for the benchmark problems, all three surrogates were in excellent agreement with the Monte Carlo simulation, and we also found that the number of training points used for the Gaussian process regression and the neural network can be further reduced without compromising the quality of the surrogates.

A c k n o w l e d g e m e n t. Bedřich Sousedík would like to thank Professors Pavel Burda and Jaroslav Novotný for many inspiring discussions about bifurcations and Navier-Stokes equations that inspired this work. Part of the work was completed while Randy Price was a graduate student at the University of Maryland, Baltimore County.

References

[1] *E. Åkervik, L. Brandt, D. S. Henningson, J. Hœpffner, O. Marxen, P. Schlatter*: Steady solutions of the Navier-Stokes equations by selective frequency damping. Phys. Fluids *18* (2006), Article ID 068102. doi

[2] *R. Andreev, C. Schwab*: Sparse tensor approximation of parametric eigenvalue problems. Numerical Analysis of Multiscale Problems. Lecture Notes in Computational Science and Engineering 83. Springer, Berlin, 2012, pp. 203–241. zbl MR doi

[3] *I. Babuška, F. Nobile, R. Tempone*: A stochastic collocation method for elliptic partial differential equations with random input data. SIAM Rev. *52* (2010), 317–355. zbl MR doi

[4] *P. Benner, A. Onwunta, M. Stoll*: A low-rank inexact Newton-Krylov method for stochastic eigenvalue problems. Comput. Methods Appl. Math. *19* (2019), 5–22. zbl MR doi

[5] *K. A. Cliffe, T. J. Garratt, A. Spence*: Eigenvalues of block matrices arising from problems in fluid mechanics. SIAM J. Matrix Anal. Appl. *15* (1994), 1310–1318. zbl MR doi

[6] *K. A. Cliffe, A. Spence, S. J. Tavener*: The numerical analysis of bifurcation problems with application to fluid mechanics. Acta Numerica *9* (2000), 39–131. zbl MR doi

[7] *F. Dan Foresee, M. T. Hagan*: Gauss-Newton approximation to Bayesian learning. Proceedings of International Conference on Neural Networks (ICNN'97). Vol. 3. IEEE, Piscataway, 1997, pp. 1930–1935. doi

[8] *H. C. Elman, K. Meerbergen, A. Spence, M. Wu*: Lyapunov inverse iteration for identifying Hopf bifurcations in models of incompressible flow. SIAM J. Sci. Comput. *34* (2012), A1584–A1606. zbl MR doi

[9] *H. C. Elman, A. Ramage, D. J. Silvester*: IFISS: A computational laboratory for investigating incompressible flow problems. SIAM Rev. *56* (2014), 261–273. zbl MR doi

[10] *H. C. Elman, D. J. Silvester*: Collocation methods for exploring perturbations in linear stability analysis. SIAM J. Sci. Comput. *40* (2018), A2667–A2693. zbl MR doi

[11] *H. C. Elman, D. J. Silvester, A. J. Wathen*: Finite Elements and Fast Iterative Solvers: With Applications in Incompressible Fluid Dynamics. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 2014. zbl MR doi

[12] *C. B. Erickson, B. E. Ankenman, S. M. Sanchez*: Comparison of Gaussian process modeling software. Eur. J. Oper. Res. *266* (2018), 179–192. zbl MR doi

[13] *T. Gerstner, M. Griebel*: Numerical integration using sparse grids. Numer. Algorithms *18* (1998), 209–232. zbl MR doi

[14] *R. G. Ghanem*: The nonlinear Gaussian spectrum of log-normal stochastic processes and variables. J. Appl. Mech. *66* (1999), 964–973. doi

[15] *R. G. Ghanem, P. D. Spanos*: Stochastic Finite Elements: A Spectral Approach. Springer, New York, 1991. zbl MR doi

[16] *V. Girault, P.-A. Raviart*: Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms. Springer Series in Computational Mathematics 5. Springer, Berlin, 1986. zbl MR doi

[17] *W. J. F. Govaerts*: Numerical Methods for Bifurcations of Dynamical Equilibria. SIAM, Philadelphia, 2000. zbl MR doi

[18] *K. Lee, H. C. Elman, B. Sousedík*: A low-rank solver for the Navier-Stokes equations with uncertain viscosity. SIAM/ASA J. Uncertain. Quantif. *7* (2019), 1275–1300. zbl MR doi

[19] *K. Lee, B. Sousedík*: Inexact methods for symmetric stochastic eigenvalue problems. SIAM/ASA J. Uncertain. Quantif. *6* (2018), 1744–1776. zbl MR doi

[20] *O. P. Le Maître, O. M. Knio*:, Scientific Computation. Springer, Dordrecht (2010) Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics. zbl MR doi

[21] *J. C. Loiseau, M. A. Bucci, S. Cherubini, J.-C. Robinet*: Time-stepping and Krylov methods for large-scale instability problems. Computational Modelling of Bifurcations and Instabilities in Fluid Dynamics. Computational Methods in Applied Sciences 50. Springer, Cham, 2019, pp. 33–73. MR doi

[22] *D. J. C. MacKay*: Bayesian interpolation. Neural Comput. *4* (1992), 415–447. doi

[23] *H. G. Matthies, A. Keese*: Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. Comput. Methods Appl. Mech. Eng. *194* (2005), 1295–1331. zbl MR doi

[24] *E. Novak, K. Ritter*: High dimensional integration of smooth functions over cubes. Numer. Math. *75* (1996), 79–97. zbl MR doi

[25] *A. O'Hagan*: Polynomial chaos: A tutorial and critique from a statistician's perspective. Available at `http://tonyohagan.co.uk/academic/pdf/Polynomial-chaos.pdf` (2013).

[26] *N. E. Owen, P. Challenor, P. P. Menon, S. Bennani*: Comparison of surrogate-based uncertainty quantification methods for computationally expensive simulators. SIAM/ASA J. Uncertain. Quantif. *5* (2017), 403–435. zbl MR doi

[27] *G. C. Y. Peng, M. Alber, A. Buganza Tepole et al.*: Multiscale modeling meets machine learning: What can we learn? Arch. Comput. Methods Eng. *28* (2021), 1017–1037. MR doi

[28] *C. E. Powell, D. J. Silvester*: Preconditioning steady-state Navier-Stokes equations with random data. SIAM J. Sci. Comput. *34* (2012), A2482–A2506. zbl MR doi

[29] *C. E. Rasmussen, C. K. I. Williams*: Gaussian Processes for Machine Learning. Adaptive Computation and Machine Learning. MIT Press, Cambridge, 2006. zbl MR doi

[30] *P. J. Schmid, D. S. Henningson*: Stability and Transition in Shear Flows. Applied Mathematical Sciences 142. Springer, New York, 2001. zbl MR doi

[31] *J. Sirignano, K. Spiliopoulos*: DGM: A deep learning algorithm for solving partial differential equations. J. Comput. Phys. *375* (2018), 1339–1364. zbl MR doi

[32] *B. Sousedík, H. C. Elman*: Stochastic Galerkin methods for the steady-state Navier-Stokes equations. J. Comput. Phys. *316* (2016), 435–452. zbl MR doi

[33] *B. Sousedík, K. Lee*: Stochastic Galerkin methods for linear stability analysis of systems with parametric uncertainty. Avaliable at `https://arxiv.org/abs/2202.12485` (2022), 27 pages.

[34] *M. L. Stein*: Interpolation of Spatial Data: Some Theory for Kriging. Springer Series in Statistics. Springer, New York, 1999. zbl MR doi

[35] *T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, D. L. Alkon*: Accelerating the convergence of the back-propagation method. Biol. Cybern. *59* (1988), 257–263. doi

[36] *D. Xiu*: Numerical Methods for Stochastic Computations: A Spectral Method Approach. Princeton University Press, Princeton, 2010. zbl MR doi

[37] *D. Xiu, G. E. Karniadakis*: The Wiener-Askey polynomial chaos for stochastic differential equations. SIAM J. Sci. Comput. *24* (2002), 619–644. zbl MR doi

[38] *L. Yan, X. Duan, B. Liu, J. Xu*: Gaussian processes and polynomial chaos expansion for regression problem: Linkage via the RKHS and comparison via the KL divergence. Entropy *20* (2018), Article ID 191, 22 pages. doi

Authors' addresses: *Bedřich Sousedík*, Department of Mathematics and Statistics, University of Maryland, Baltimore County, 1000 Hilltop Circle, Baltimore, MD 21250, USA, e-mail: sousedik@umbc.edu; *Howard C. Elman*, Department of Computer Science and Institute for Advanced Computer Studies, Iribe Center, 8125 Paint Branch Dr, University of Maryland, College Park, MD 20742, USA, e-mail: helman@umd.edu; *Kookjin Lee*, School of Computing and Augmented Intelligence, Arizona State University, 699 S Mill Ave, Tempe, AZ 85281, USA, e-mail: kookjin.lee@asu.edu; *Randy Price*, Center for Mathematics and Artificial Intelligence and Center for Computational Fluid Dynamics, George Mason University, 4400 University Drive, MS: 3F2, Exploratory Hall, room 4102, Fairfax, VA 22030, USA, e-mail: rprice25@gmu.edu.