Real-time Bayesian inference at extreme scale: A digital twin for tsunami early warning applied to the Cascadia subduction zone

Stefan Henneking
Oden Institute
The University of Texas at Austin
stefan@oden.utexas.edu

Sreeram Venkat
Oden Institute
The University of Texas at Austin
srvenkat@utexas.edu

Veselin Dobrev
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
dobrev1@llnl.gov

John Camier
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
camier1@llnl.gov

Tzanio Kolev
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
kolev1@llnl.gov

Milinda Fernando
Oden Institute
The University of Texas at Austin
milinda@oden.utexas.edu

Alice-Agnes Gabriel
Scripps Institution of Oceanography
University of California San Diego
algabriel@ucsd.edu

Omar Ghattas
Oden Institute, Walker Department of Mechanical Engineering
The University of Texas at Austin
omar@oden.utexas.edu

***Abstract*—We present a Bayesian inversion-based digital twin that employs acoustic pressure data from seafloor sensors, along with 3D coupled acoustic–gravity wave equations, to infer earthquake-induced spatiotemporal seafloor motion in real time and forecast tsunami propagation toward coastlines for early warning with quantified uncertainties. Our target is the Cascadia subduction zone, with one billion parameters. Computing the posterior mean alone would require 50 years on a 512 GPU machine. Instead, exploiting the shift invariance of the parameter-to-observable map and devising novel parallel algorithms, we induce a fast offline–online decomposition. The offline component requires just one adjoint wave propagation per sensor; using MFEM, we scale this part of the computation to the full El Capitan system (43,520 GPUs) with 92% weak parallel efficiency. Moreover, given real-time data, the online component exactly solves the Bayesian inverse and forecasting problems in 0.2 seconds on a modest GPU system, a ten-billion-fold speedup.**

***Index Terms*—Bayesian inverse problems, data assimilation, uncertainty quantification, digital twins, tsunami early warning, finite elements, real-time GPU supercomputing**

I. Justification for ACM Gordon Bell Prize

Fastest time-to-solution of a PDE-based Bayesian inverse problem with 1 billion parameters in 0.2 seconds, a ten-billion-fold speedup over SoA. Largest-to-date unstructured mesh FE simulation with 55.5 trillion DOF on 43,520 GPUs, with 92% weak and 79% strong parallel efficiencies in scaling over a 128× increase of GPUs on the full-scale *El Capitan* system.

This research was supported by DOD MURI grants FA9550-21-1-0084 and FA9550-24-1-0327, DOE ASCR grant DE-SC0023171, and NSF grants DGE-2137420, EAR-2225286, EAR-2121568, OAC-2139536. Work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-PROC-2004753). This research used resources from the Texas Advanced Computing Center (TACC), the National Energy Research Scientific Computing Center (NERSC, ALCC-ERCAP0030671), and Livermore Computing.

II. Performance Attributes

Performance attribute	This submission
Category of achievement	Scalability, time-to-solution, peak performance
Type of method used	Bayesian inversion, real-time computing, FEM
Results reported based on	Whole application including I/O
Precision reported	Double precision
System scale	Results measured on full-scale system
Measurement mechanism	Timers, DOF throughput, FLOP count

III. Overview of the Problem

A. Tsunami forecasting and the Cascadia subduction zone

Tsunamis are rare events but can be extremely deadly and cause catastrophic socioeconomic losses. In the past century tsunamis have claimed more than 400,000 lives, and individual tsunamis can result in hundreds of billions of dollars in losses, making them among the costliest natural disasters. State-of-the-art (SoA) tsunami early warning systems [1] rely on rapid characterization of source parameters primarily from seismic data [2], issuing alerts within minutes but based on simplified assumptions about the event. These systems typically infer earthquake moment magnitude and hypocenter location, and then trigger tsunami forecasts using precomputed scenarios or simplified fault models, assuming instantaneous, uniform slip on a predefined fault geometry. This approach is inadequate in the near field [3], where destructive tsunami waves can arrive on-shore in under ten minutes. In addition, current models fail to capture the complexities of earthquake rupture dynamics [4], including variable slip distributions and slow rupture speeds. This can lead to delayed, missed, or false warnings, especially when early seismic data misrepresent the true tsunami potential, such as during the 2011 Tōhoku, Japan tsunami or the 2024 Cape Mendocino, California earthquake,

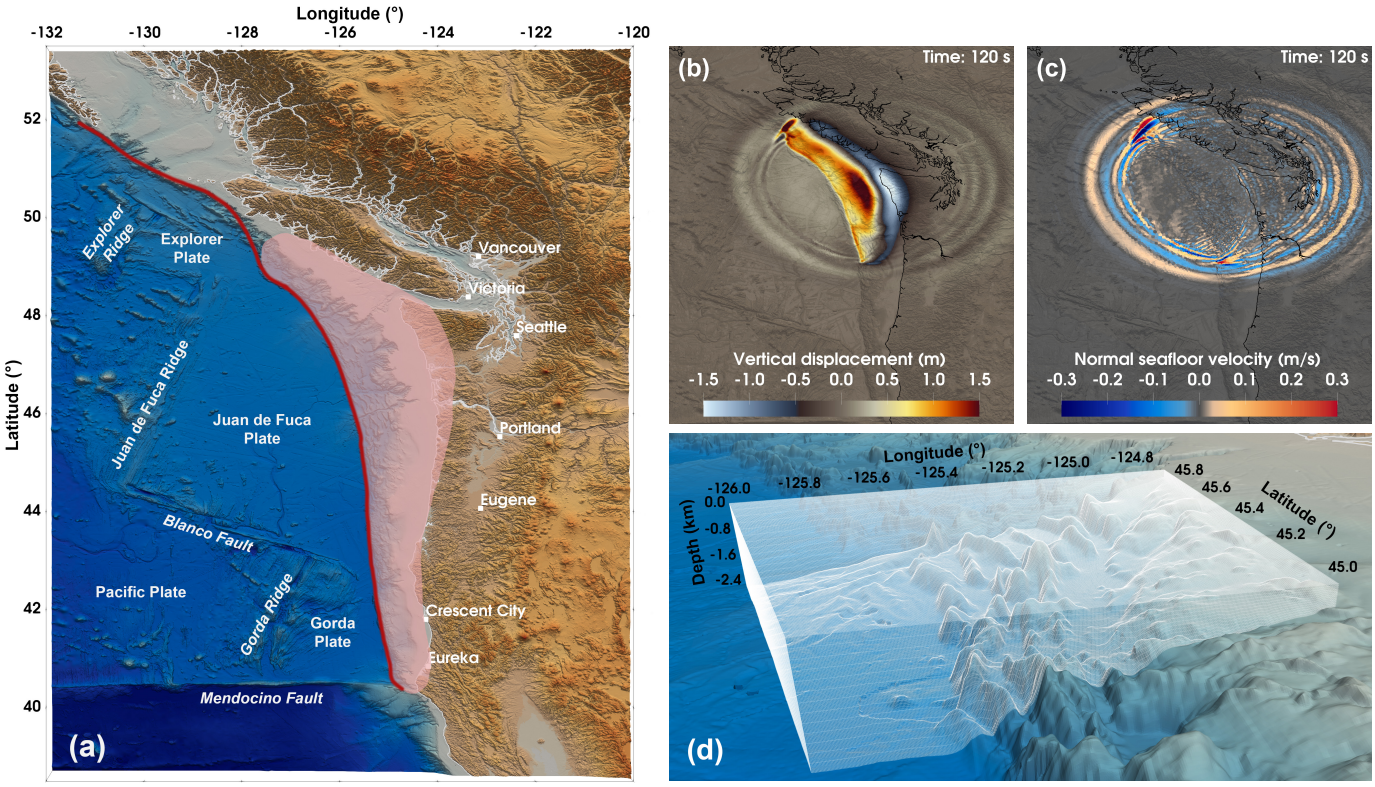

Fig. 1. (a) Topobathymetric map of the Cascadia subduction zone (CSZ) based on GEBCO's gridded bathymetric data set. The shaded area illustrates the potentially "locked" portion of the megathrust interface. The red line marks the trench where the subducting Explorer, Juan de Fuca, and Gorda plates begin their descent beneath the North American Plate. (b)–(c) Snapshots of vertical seafloor uplift and seafloor velocity from a physics-based 3D dynamic rupture and seismic wave propagation computation of a magnitude 8.7 earthquake scenario spanning the full margin of the CSZ. (d) Representative block of a 3D multi-block hexahedral mesh of the CSZ, depicting bathymetry-adapted meshing.

which did not cause a tsunami, despite five million people receiving evacuation alerts. To improve warning capabilities, real-time ocean bottom pressure data and GPS measurements can be integrated directly to constrain tsunami sources, without relying on assumed fault geometry or magnitude [5, 6].

SoA tsunami simulations typically solve the nonlinear shallow water equations, sometimes with added dispersive terms or empirical corrections, to model wave generation, propagation, and run-up. These models are widely used in tsunami hazard assessment and early warning but are known to omit important physical processes such as wave dispersion, 3D hydrodynamic interactions with bathymetry and coastal infrastructure, and the dynamics of the coupled Earth–ocean system [7]. Large-scale tsunami simulations using high-resolution bathymetry are constrained by simplifications in source modeling and may require minutes to hours per run, limiting their use in real-time settings [8].

We present a full-physics Bayesian inversion framework—a so-called *digital twin* [9]—that enables real-time, data-driven tsunami forecasting with dynamic adaptivity to complex source behavior. Our inversion framework employs ocean bottom pressure data, along with the 3D coupled acoustic–gravity wave equations, to infer earthquake-induced spatiotemporal seafloor motion in real time. The solution of this Bayesian inverse problem then provides the boundary forcing to forward propagate the tsunamis toward coastlines and issue forecasts of wave heights at specified locations, with quantified uncertainties. We apply our framework to the Cascadia subduction zone (CSZ), which stretches 1000 km from northern California to British Columbia (Fig. 1). The CSZ has been eerily quiet for over 300 years but is considered overdue for a magnitude 8.0–9.0 megathrust earthquake, with paleoseismic and geodetic evidence indicating a recurrence interval of roughly 250 to 500 years [10, 11]. In addition to recently proposed future offshore deployments (e.g., by SZ4D), the NEPTUNE cabled observatory [12] has provided continuous offshore observations since 2009 across the northern Cascadia margin, recording several tsunamis and offering valuable data to inform optimal sensor placement.

B. Bayesian inverse problems

Our work addresses a critically important class of problems that has received little attention in the Gordon Bell Prize competition: *inverse problems* [13, 14], in particular in the Bayesian framework [15, 16]. In a *forward problem*, the input *parameters* (e.g., initial or boundary conditions, coefficients, source terms) are specified; solution of the governing equations, here assumed to be partial differential equations (PDEs), then yields the state variables describing the behavior of the system. However, forward models typically contain uncertain parameters. Often these uncertain parameters are infinite-dimensional fields, i.e., they vary in space (and possibly time).

In inverse problems, we seek to infer uncertain parameter fields from measurements or observations of system states.

Infinite-dimensional inverse problems are usually ill-posed: many parameter fields can be found that are consistent with the observational data to within the noise. The classical output least-squares approach to the inverse problem posits a regularization functional that penalizes unwanted features of the parameter field (e.g., magnitude or roughness), yielding a unique solution [13]. In contrast, the *Bayesian approach* seeks to characterize the probability that *any* parameter field is consistent with the data and prior knowledge. The parameter is treated as a random field, and the solution of the inverse problem becomes a probability density, the *posterior distribution* [16]. As such the Bayesian approach is very powerful.

This power comes at a price, though: computing the Bayesian solution is generally intractable for infinite-dimensional inverse problems governed by PDEs. For example, computing just the mean of the posterior formally requires numerical integration in the discretized parameter dimension (in our case, one billion); *each evaluation* of the integrand requires solution of the forward problem, which can take hours or more, and numerous forward solutions are required. As a result, various approximations of the posterior, such as the Laplace approximation [14], must be invoked, and even then massive-scale computation is necessary, e.g. [17, 18, 19]. The specific Bayesian inverse problem we address here is the real-time inference and prediction of tsunamis for early warning, which adds a real-time requirement to the challenges described above. Our real-time inversion and forecasting goal would seem futile, since SoA inversion methods (see §IV) would require *50 years* on 512 NVIDIA A100 GPUs to solve this problem, whereas we have less than a minute to provide early warning. As we shall see in §V, we are able to achieve real-time performance through a combination of novel parallel inversion algorithms, advanced PDE discretization libraries, and exploitation of leading-edge GPU supercomputers.

C. Inference of seafloor motion with acoustic–gravity model

In this section, we describe our target inverse problem. The forward tsunami dynamics are modeled by the acoustic–gravity wave equations that couple the propagation of ocean acoustic waves with surface gravity waves. This PDE model is derived by linearization of the conservation of mass and momentum around hydrostatic pressure in the compressible ocean, and the coupling to the surface gravity wave comes from a modified free surface boundary condition at the sea surface [20]. The model takes the form of a coupled first-order system in the velocity vector field $\vec{u}(\vec{x},t)$, the pressure field $p(\vec{x},t)$, and the surface gravity wave height $\eta(\vec{x},t)$,

$$\begin{cases} \rho\,\partial_t \vec{u} + \nabla p = 0, & \Omega \times (0,T), \\ K^{-1}\partial_t p + \nabla\cdot\vec{u} = 0, & \Omega \times (0,T), \\ p = \rho g \eta, & \partial\Omega_{\mathrm{s}} \times (0,T), \\ \partial_t \eta = \vec{u}\cdot\vec{n}, & \partial\Omega_{\mathrm{s}} \times (0,T), \\ \vec{u}\cdot\vec{n} = -\partial_t b, & \partial\Omega_{\mathrm{b}} \times (0,T), \\ \vec{u}\cdot\vec{n} = Z^{-1} p, & \partial\Omega_{\mathrm{a}} \times (0,T), \end{cases} \tag{1}$$

with homogeneous initial conditions. Here, K and ρ are the bulk modulus and density of seawater, $Z = \rho c$ is the acoustic wave impedance, $c = \sqrt{K/\rho}$ is the speed of sound in seawater, g is gravitational acceleration, and $\partial_t b$ is the seafloor normal velocity with b the normal displacement. The spatial domain is denoted by Ω with boundaries $\partial\Omega_{\mathrm{s}}$ (sea surface), $\partial\Omega_{\mathrm{b}}$ (sea bottom), and $\partial\Omega_{\mathrm{a}}$ (lateral, absorbing boundaries); $\vec{n}$ is the outward unit normal; and the temporal domain is $(0,T)$.

The uncertain parameter field $m(\vec{x},t)$, to be inferred from the data, is the spatiotemporal seafloor normal velocity,

$$m(\vec{x},t) := -\partial b(\vec{x},t)/\partial t, \quad (\vec{x},t) \in \partial\Omega_{\mathrm{b}} \times (0,T).$$

The observables d are model predictions of data d^{obs} from pressure sensors mounted on the seafloor, i.e. $d = p(\vec{x}_j^d, t_i^d)$, where $\vec{x}_j^d \in \partial\Omega_{\mathrm{b}}$, $j = 1,\ldots,N_d$, are the seafloor sensor locations and $t_i^d \in (0,T)$, $i = 1,\ldots,N_t^d$, are the time instants at which observations are made. The *parameter-to-observable* (p2o) *map* $\mathcal{F}$ takes the seafloor velocity m as input, solves the acoustic–gravity wave equations (1), and extracts d, the pressures at the sensor locations and observation times. The *quantities of interest* (QoI) are the surface wave height predictions at locations and times of interest, i.e. $q = \eta(\vec{x}_j^q, t_i^q)$ where $\vec{x}_j^q \in \partial\Omega_{\mathrm{s}}$, $j = 1,\ldots,N_q$, and $t_i^q \in (0,T)$, $i = 1,\ldots,N_t^q$, are specific locations and time instants pertinent to tsunami early warning. The *parameter-to-QoI* (p2q) *map* $\mathcal{F}_q$ takes the inferred seafloor velocity m and forward predicts the QoI q.

The Bayesian inverse problem can then be stated as follows: given pressure recordings d^{obs} from sensors on the seafloor, infer the spatiotemporal seafloor velocity m in the subduction zone and its uncertainty. The tsunami posterior prediction problem is: given inferred parameters m and their uncertainty, forward predict the surface gravity wave heights q and their uncertainty. In §IV, we discuss why this problem is intractable with the current SoA.

IV. Current State of the Art

In this section, we discuss SoA methods for solving the Bayesian inverse problem described in §III-C. After discretization in space and time, Bayes' theorem gives

$$\pi_{\mathrm{post}}(\mathbf{m}|\mathbf{d}^{\mathrm{obs}}) \propto \pi_{\mathrm{like}}(\mathbf{d}^{\mathrm{obs}}|\mathbf{m})\pi_{\mathrm{prior}}(\mathbf{m}),$$

where $\pi_{\mathrm{prior}}(\mathbf{m})$ is the prior probability density of the seafloor velocity parameters $\mathbf{m}$, $\pi_{\mathrm{like}}(\mathbf{d}^{\mathrm{obs}}|\mathbf{m})$ is the likelihood of the observed seafloor pressure data $\mathbf{d}^{\mathrm{obs}}$, given $\mathbf{m}$, and $\pi_{\mathrm{post}}(\mathbf{m}|\mathbf{d}^{\mathrm{obs}})$ is the posterior density reflecting the probability of the parameters conditioned on the data.

Taking a Gaussian prior with mean $\mathbf{m}_{\mathrm{prior}}$ and covariance $\mathbf{\Gamma}_{\mathrm{prior}}$ (here block diagonal, with each block the inverse of an elliptic PDE operator in space representing a Matérn covariance [15]), along with centered Gaussian additive noise with noise covariance $\mathbf{\Gamma}_{\mathrm{noise}}$, the posterior is then given by:

$$\pi_{\mathrm{post}}(\mathbf{m}|\mathbf{d}^{\mathrm{obs}}) \propto \exp\left\{-\tfrac{1}{2}\|\mathbf{F}\mathbf{m}-\mathbf{d}^{\mathrm{obs}}\|^2_{\mathbf{\Gamma}_{\mathrm{noise}}^{-1}} - \tfrac{1}{2}\|\mathbf{m}-\mathbf{m}_{\mathrm{prior}}\|^2_{\mathbf{\Gamma}_{\mathrm{prior}}^{-1}}\right\}.$$

Here $\mathbf{F}$ is the discrete p2o map.

The maximum-a-posteriori (MAP) point $\mathbf{m}_{\text{map}}$ maximizes the posterior distribution over admissible parameters $\mathbf{m}$, or equivalently solves the quadratic optimization problem

$$\mathbf{m}_{\text{map}} := \arg\min_{\mathbf{m}} \tfrac{1}{2}\|\mathbf{F}\mathbf{m} - \mathbf{d}^{\text{obs}}\|^2_{\mathbf{\Gamma}^{-1}_{\text{noise}}} + \tfrac{1}{2}\|\mathbf{m} - \mathbf{m}_{\text{prior}}\|^2_{\mathbf{\Gamma}^{-1}_{\text{prior}}}.$$

The MAP point $\mathbf{m}_{\text{map}}$ is thus the solution of the linear system

$$\left(\mathbf{F}^*\mathbf{\Gamma}^{-1}_{\text{noise}}\mathbf{F} + \mathbf{\Gamma}^{-1}_{\text{prior}}\right)\mathbf{m}_{\text{map}} = \mathbf{F}^*\mathbf{\Gamma}^{-1}_{\text{noise}}\mathbf{d}^{\text{obs}} + \mathbf{\Gamma}^{-1}_{\text{prior}}\mathbf{m}_{\text{prior}}, \quad (2)$$

where $\mathbf{H} := (\mathbf{F}^*\mathbf{\Gamma}^{-1}_{\text{noise}}\mathbf{F} + \mathbf{\Gamma}^{-1}_{\text{prior}})$ is the Hessian of the negative log-posterior and $\mathbf{F}^*$ is the adjoint of the p2o map.

The posterior is then a *Gaussian centered at the MAP point*, i.e. $\pi_{\text{post}}(\mathbf{m}|\mathbf{d}^{\text{obs}}) \propto \mathcal{N}(\mathbf{m}_{\text{map}}, \mathbf{\Gamma}_{\text{post}})$, with posterior covariance

$$\mathbf{\Gamma}_{\text{post}} := \mathbf{H}^{-1} = \left(\mathbf{F}^*\mathbf{\Gamma}^{-1}_{\text{noise}}\mathbf{F} + \mathbf{\Gamma}^{-1}_{\text{prior}}\right)^{-1}. \quad (3)$$

Furthermore, the posterior predictive of the wave height QoI $\mathbf{q}$, which depend linearly on the parameters via the discrete p2q map $\mathbf{F}_q$, can be determined as the Gaussian probability density $\pi_{\text{post}}(\mathbf{q}|\mathbf{d}^{\text{obs}}) \propto \mathcal{N}\left(\mathbf{q}_{\text{map}}, \mathbf{\Gamma}_{\text{post}(q)}\right)$, where $\mathbf{q}_{\text{map}} = \mathbf{F}_q\mathbf{m}_{\text{map}}$ and $\mathbf{\Gamma}_{\text{post}(q)} = \mathbf{F}_q\mathbf{\Gamma}_{\text{post}}\mathbf{F}_q^*$.

Writing expressions (2) for the posterior mean $\mathbf{m}_{\text{map}}$ and (3) for the posterior covariance $\mathbf{\Gamma}_{\text{post}}$ is easy; computing them is a monumental challenge. The coefficient matrix of (2), the Hessian $\mathbf{H}$, is a dense matrix of dimension of the parameters, here $10^9 \times 10^9$. Constructing it efficiently via $\mathbf{F}$ requires as many adjoint wave propagation solutions—here, each requiring $\sim$1 hour on 512 NVIDIA A100 GPUs—as there are spatiotemporal data, here $\sim$250,000; this amounts to 25 years. Even if $\mathbf{H}$ could be constructed and stored (4 exabytes), factoring it would require 10 years on a sustained 1 EFLOP/s machine. SoA methods for large-scale inverse problems employ matrix-free conjugate gradients (CG), preconditioned by the prior covariance (thus involving elliptic PDE solves), and yield convergence in a number of iterations of the order of the number of eigenvalues of the prior-preconditioned Hessian of the negative log likelihood, $\widetilde{\mathbf{H}}_{\text{like}} := \mathbf{\Gamma}^{1/2}_{\text{prior}}\mathbf{F}^*\mathbf{\Gamma}^{-1}_{\text{noise}}\mathbf{F}\mathbf{\Gamma}^{1/2}_{\text{prior}}$, that are above unity [14]. For highly ill-posed inverse problems, the eigenvalues decay rapidly and $\widetilde{\mathbf{H}}_{\text{like}}$ has low effective rank, and thus CG converges rapidly [14]. Moreover, a low-rank approximation of $\widetilde{\mathbf{H}}_{\text{like}}$ computed via a matrix-free randomized eigensolver, along with the Sherman–Morrison–Woodbury formula, can be used to compactly represent the posterior covariance, sample from the posterior distribution, and compute the posterior variance field, as has been done for some large-scale Bayesian inverse problems [17, 18].

Unfortunately, our operator is not low rank, due to the hyperbolic nature of the forward wave propagation problem (which tends to preserve information), and the location of the pressure sensors directly on the seafloor, whose motion we seek to infer. In our numerical experiments on smaller but representative regions [21], the effective rank is nearly of the order of the data dimension, and thus we can expect CG to take $\mathcal{O}$(250,000) iterations. Since each iteration requires the action of $\mathbf{F}$ and $\mathbf{F}^*$ on a vector, i.e., a pair of forward/adjoint wave propagations, we would expect CG to take $\mathcal{O}$(50) years (on 512 GPUs) for our target problem. And this is for computing $\mathbf{m}_{\text{map}}$ via (2); in the absence of low-rank properties, the even more daunting task of evaluating the uncertainty of the inverse solution (which is also needed for the posterior of the QoI, $\mathbf{\Gamma}_{\text{post}(q)}$) via the inverse of the Hessian in (3) is unthinkable. While approximations of high (global) rank Hessians have been attempted, they have not been viable for high-frequency wave problems in 10^9 parameter dimensions, e.g., [22].

Alternatively, we could relax the need to exactly solve (2) for $\mathbf{m}_{\text{map}}$ and instead seek a surrogate approximation of the p2o map, $\mathbf{F}$. However, this too is a monumental challenge: constructing surrogates in 10^9 parameter dimensions, where each training point amounts to a forward wave propagation ($\sim$1 hour on 512 GPUs), is essentially impossible using conventional surrogate methods. Specially-architected neural operator surrogates that exploit the geometry of p2o maps have been developed, scaled to high ambient dimensions, and deployed within Bayesian inverse problems, e.g., [23]. However, the efficiency of these surrogates (measured by number of training data needed) is predicated on low-rank representations of $\mathbf{F}$, again a property not enjoyed by our p2o maps. Thus, surrogates are not a viable option for our problem.

Finally, we might attempt to construct a projection-based reduced order model (ROM) of the forward acoustic–gravity wave equations, by projecting (1) onto a low-dimensional reduced basis. The ROM would have to be parameterized over 10^9 inputs, which is well beyond the reach of current methods. A more fundamental difficulty is that efficient ROMs for high-frequency wave propagation are not viable due to the Kolmogorov N-width problem [24], which asserts the non-existence of a low-dimensional subspace embedding.

Next, we present novel parallel algorithms that exploit the autonomous nature of the dynamics (1) and p2o and p2q maps $\mathbf{F}$ and $\mathbf{F}_q$ to induce an offline–online decomposition permitting real-time solution of both the Bayesian inverse problem for seafloor motion and subsequent wave height forecasting.

V. Innovations Realized

Our real-time Bayesian inference and prediction framework combines the following key ideas: (i) an offline–online decomposition of the inverse solution that precomputes various mappings to enable the real-time application of the inverse operator and entirely circumvents the need for PDE solutions during the inversion phase; (ii) representation of the inverse operator in data space instead of high-dimensional parameter space; (iii) extension of this real-time inference methodology to the goal-oriented setting for the prediction of QoI under uncertainty; (iv) exploitation of the shift invariance of (1) to reduce the required number of PDE solutions by several orders of magnitude; and (v) exploitation of this same property to perform efficient FFT-based and multi-GPU-accelerated Hessian matrix–vector products (matvecs). We employ this framework to construct a digital twin for tsunami early warning in the CSZ that solves a Bayesian inverse problem with over 10^9 parameters and 8,820 QoI *exactly* (up to rounding errors); both the parameter inference and the complete end-to-end, data-to-

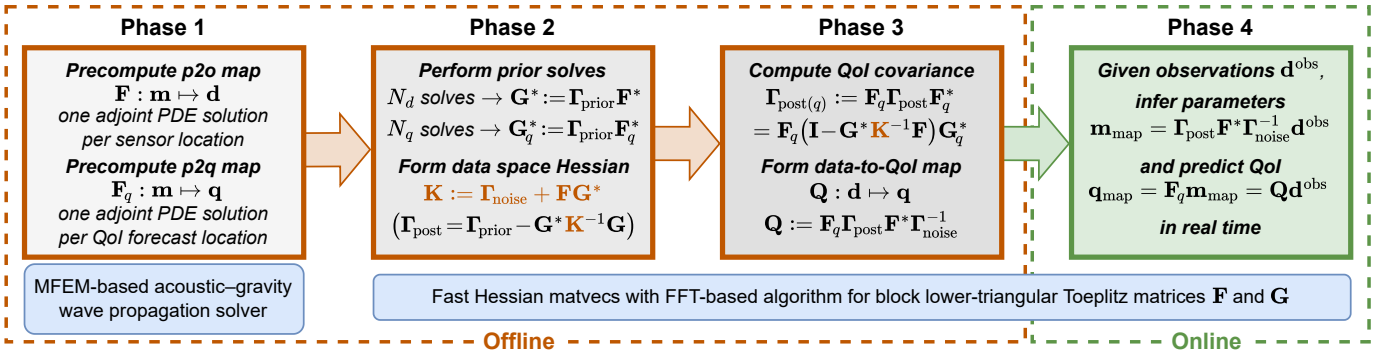

Fig. 2. Real-time Bayesian inference framework. The inverse solution is decomposed into several precomputation (offline) Phases 1–3 that are executed just once, and a real-time (online) Phase 4 of parameter inference and QoI prediction that is executed when an earthquake occurs and data are acquired. Phase 1 computes adjoint PDE solutions of the acoustic–gravity model (one PDE solution per sensor and QoI forecast location) to precompute the p2o and p2q block Toeplitz matrices. Phases 2–4 rely on fast FFT-based Hessian actions using this block Toeplitz structure and a transformation of the inverse operator from the high-dimensional parameter space to the much lower-dimensional data space. Compute times for each phase are given in Table III (§VII).

inference-to-prediction computations can be carried out within a fraction of a second.

A. FFT-based Hessian matvecs

Solving the inverse problem (2) requires numerous Hessian matvecs, and each Hessian matvec comes at the cost of a pair of forward and adjoint PDE solutions. For large-scale problems, PDE solutions are computationally expensive and cannot be performed in real time. One key innovation that enables real-time inference is to recognize and exploit the fact that the p2o map is governed by an *autonomous dynamical system*. Specifically, the acoustic–gravity model (1) represents a linear time-invariant (LTI) dynamical system for which the PDE operator does not explicitly depend on time.

The LTI dynamical system structure of the governing equations implies that the discrete p2o map $\mathbf{F}$, which maps parameters $\mathbf{m}$ (spatiotemporal seafloor motion) to observables $\mathbf{d}$ (sea bottom pressure) via solution of the discretized equations of the acoustic–gravity PDE model (1), inherits a special structure. Let N_m denote the spatial parameter points, N_d the number of sensors ($N_d \ll N_m$), and N_t the temporal dimension of parameters and observations[1] ($N_t \gg 1$), i.e.,

- $\mathbf{m} \in \mathbb{R}^{N_m N_t}$ with blocks $\mathbf{m}_j \in \mathbb{R}^{N_m}$, $j = 1, 2, \ldots, N_t$;
- $\mathbf{d} \in \mathbb{R}^{N_d N_t}$ with blocks $\mathbf{d}_i \in \mathbb{R}^{N_d}$, $i = 1, 2, \ldots, N_t$.

Then, the discrete p2o map $\mathbf{F}$ can be written as a *block lower-triangular Toeplitz* matrix:

$$\begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \\ \mathbf{d}_3 \\ \vdots \\ \mathbf{d}_{N_t} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_{11} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{F}_{21} & \mathbf{F}_{11} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{F}_{31} & \mathbf{F}_{21} & \mathbf{F}_{11} & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{F}_{N_t,1} & \mathbf{F}_{N_t-1,1} & \cdots & \mathbf{F}_{21} & \mathbf{F}_{11} \end{bmatrix} \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \\ \mathbf{m}_3 \\ \vdots \\ \mathbf{m}_{N_t} \end{bmatrix},$$

where $\mathbf{F} \in \mathbb{R}^{(N_d N_t)\times(N_m N_t)}$ with blocks $\mathbf{F}_{ij} \in \mathbb{R}^{N_d \times N_m}$, $i, j = 1, 2, \ldots, N_t$.

[1]For simplicity of presentation, without loss of generality, we assume that the frequency of the observations and QoI predictions is equal to the temporal discretization of the parameters, i.e., $N_t = N_t^d = N_t^q$.

In other words, the block Toeplitz structure of $\mathbf{F}$ corresponds to a *time-shift invariance* of its blocks $\mathbf{F}_{ij}$. This shift invariance implies that (i) the p2o map can be precomputed from only N_d (number of sensors) adjoint PDE solutions (corresponding to the first block column of $\mathbf{F}$) and stored compactly at cost of $\mathcal{O}(N_m N_d N_t)$ memory; (ii) the block Toeplitz matrix can be embedded within a block circulant matrix that is block-diagonalized by the discrete Fourier transform [25]; and (iii) the p2o matvec then becomes a block-diagonal matvec operation in Fourier space.

These FFT-based p2o matvecs can be implemented efficiently on multi-GPU clusters. In particular, care is taken to arrange the data layouts of the matrices and vectors in each part of the computation to minimize the amount of GPU–GPU communication. By exchanging the order of space and time vector indices local to each processor, the need for strided memory accesses during the core computational kernels is avoided. A 2D GPU partitioning scheme is employed, where the dimensions of the processor grid are adaptively tuned according to the problem sizes and total number of GPUs in order to further reduce communication costs [26].

The main advantage of the FFT-based matvec algorithm is that it avoids PDE solves entirely, thus making it independent of the cost of state discretization, time-stepping constraints from Courant–Friedrichs–Lewy (CFL) condition, and computing on unstructured mesh-based data structures dictated by the PDE discretization. Moreover, it relies on arithmetic operations that involve data that is contiguous in memory and leverages routines from libraries such as cuBLAS that are well known to achieve high performance on GPUs. As we shall see, the initial cost of precomputing $\mathbf{F}$, for a practical number of sensor locations, is easily amortized in the context of solving Bayesian inverse problems, where the matrix-free Hessian actions (that involve costly PDE solutions) can now be replaced with these extremely fast FFT-based Hessian matvecs.

Finally, the entire discussion about the p2o map $\mathbf{F}$ analogously applies to the p2q map $\mathbf{F}_q \in \mathbb{R}^{(N_q N_t)\times(N_m N_t)}$ that maps the parameters $\mathbf{m}$ to the QoI $\mathbf{q} \in \mathbb{R}^{N_q N_t}$ (sea surface wave heights) at N_q forecast locations ($N_q \ll N_m$).

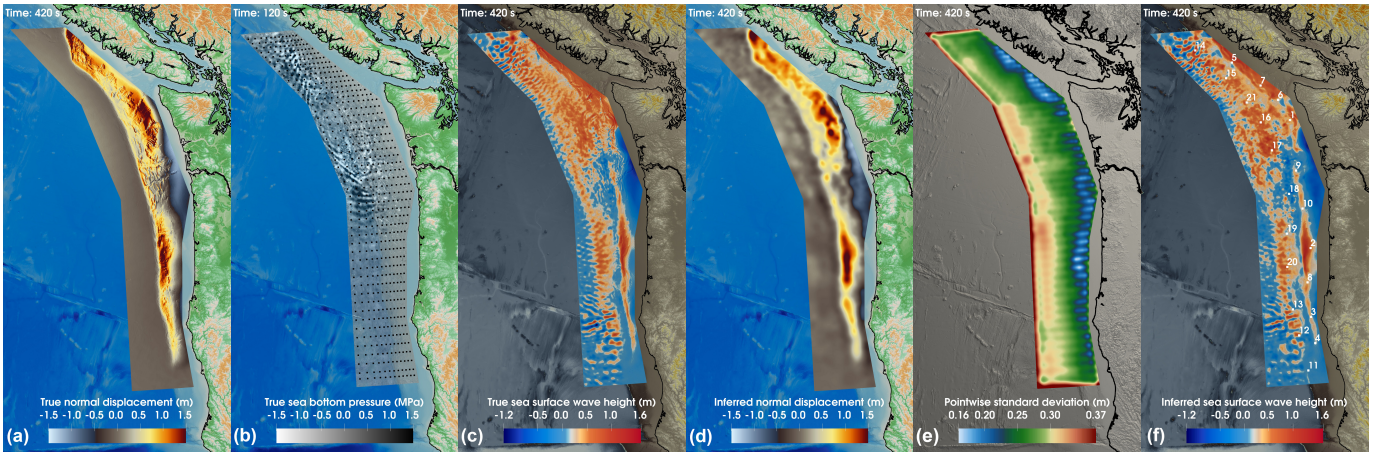

Fig. 3. Physics-based magnitude 8.7 dynamic rupture earthquake scenario for a margin-wide rupture in the CSZ, from left to right: (a) True seafloor displacement; (b) snapshot of true seafloor acoustic pressure field with 600 hypothesized sensor locations; (c) snapshot of true sea surface wave height; (d) inferred mean of seafloor displacement; (e) uncertainties plotted as pointwise standard deviations in meters of seafloor normal displacement; and (f) snapshot of reconstructed sea surface wave height with 21 locations for QoI predictions. Hyperlinks to animations of: (i) source (seafloor vertical uplift and normal velocity); (ii) forward solution (sea bottom pressure and surface wave height); and (iii) inverse solution (true and inferred seafloor normal displacement).

B. Real-time Bayesian inference and prediction framework

To enable real-time Bayesian inference, our framework decomposes the inverse solution into several precomputation (offline) Phases 1–3 and a real-time inference (online) Phase 4 [21], as depicted in Fig. 2.

Phase 1 performs $N_d + N_q$ (number of sensors plus QoI forecasts) adjoint PDE solutions to precompute the block Toeplitz matrices $\mathbf{F}$ and $\mathbf{F}_q$.

Phase 2 computes the block Toeplitz matrices $\{\mathbf{G}^*, \mathbf{G}_q^*\} = \mathbf{\Gamma}_{\text{prior}}\{\mathbf{F}^*, \mathbf{F}_q^*\}$ by performing $N_d + N_q$ solves of the inverse elliptic operator defining the prior covariance $\mathbf{\Gamma}_{\text{prior}}$. Then, the Sherman–Morrison–Woodbury formula is used to rewrite the posterior covariance as

$$\mathbf{\Gamma}_{\text{post}} = \left(\mathbf{F}^*\mathbf{\Gamma}_{\text{noise}}^{-1}\mathbf{F} + \mathbf{\Gamma}_{\text{prior}}^{-1}\right)^{-1} = \left(\mathbf{\Gamma}_{\text{prior}} - \mathbf{G}^*\mathbf{K}^{-1}\mathbf{G}\right),$$

where $\mathbf{K} = \mathbf{\Gamma}_{\text{noise}} + \mathbf{F}\mathbf{G}^*$. Doing so effectively shifts the inverse operator from the intractable high-dimensional parameter space (of dimension $N_m N_t$) to the still-large-but-tractable data space (of dimension $N_d N_t$); for that reason, we refer to $\mathbf{K}$ as the (prior-preconditioned) "data space Hessian." The dense $\mathbf{K}$ matrix is precomputed with $N_d N_t$ matvecs on unit vectors, each of which—thanks to the FFT-based matvecs of $\mathbf{F}$ and $\mathbf{G}^*$—can be executed within fractions of a second, after which $\mathbf{K}$ can be Cholesky-factorized.

Phase 3 computes the uncertainties for the QoI predictions by forming the dense covariance matrix $\mathbf{\Gamma}_{\text{post}(q)} = \mathbf{F}_q\mathbf{\Gamma}_{\text{post}}\mathbf{F}_q^*$. This requires $N_q N_t$ matvecs with $\mathbf{\Gamma}_{\text{post}(q)}$ on unit vectors; each matvec effectively solves a billion-parameter inverse problem (in the actions of $\mathbf{\Gamma}_{\text{post}}$ on a vector), which is made tractable thanks to our ability to solve the inverse problem in a fraction of a second. Analogously, we precompute the data-to-QoI map, $\mathbf{Q} := \mathbf{F}_q\mathbf{\Gamma}_{\text{post}}\mathbf{F}^*\mathbf{\Gamma}_{\text{noise}}^{-1}$, which, for a moderate number of QoI forecasts, can (in Phase 4) be used to make rapid real-time QoI predictions directly from observational data, thereby bypassing the need to explicitly reconstruct the seafloor velocity parameters (see §VII).

Phase 4 executes the (online) parameter inference and QoI prediction, which, given data $\mathbf{d}^{\text{obs}}$, reduces to computing the MAP points, $\mathbf{m}_{\text{map}} = \mathbf{\Gamma}_{\text{post}}\mathbf{F}^*\mathbf{\Gamma}_{\text{noise}}^{-1}\mathbf{d}^{\text{obs}}$ and $\mathbf{q}_{\text{map}} = \mathbf{F}_q\mathbf{m}_{\text{map}} = \mathbf{Q}\mathbf{d}^{\text{obs}}$ (taking $\mathbf{m}_{\text{prior}} = \mathbf{0}$), both of which, after the precomputations done in Phases 1–3, can be executed *in real time*.

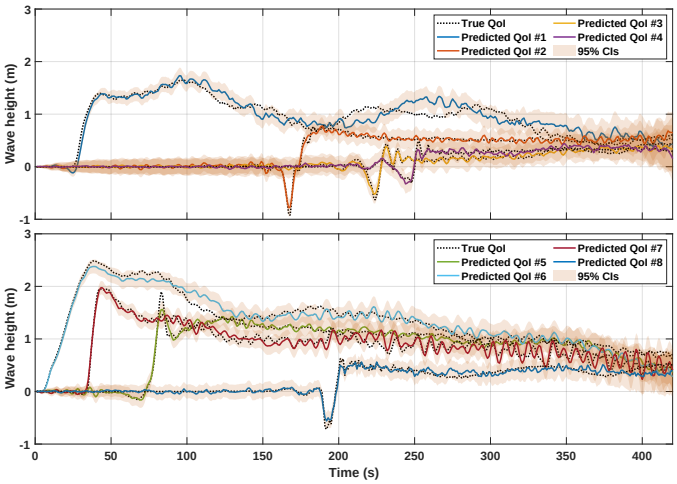

Fig. 4. Real-time QoI predictions with uncertainties illustrated as 95% credible intervals (CIs) inferred from noisy, synthetic data of 600 hypothesized seafloor acoustic pressure sensors for a margin-wide rupture in the CSZ. The QoI numbers (#1–#8) refer to (a subset of) the 21 QoI forecast locations marked in the inferred (reconstructed) sea surface wave height plot in Fig. 3.

C. Application to the Cascadia subduction zone

To assess the feasibility of our framework for near-field real-time tsunami forecasting in Cascadia without assuming simplified earthquake models, we generated seafloor displacements that capture the nonlinear interaction of seismic wave propagation and frictional rock failure during a realistic CSZ rupture scenario computed with a 3D dynamic earthquake rupture model [27, 4]. We then used these "true" seafloor displacements as a source for our acoustic–gravity forward model (i.e. normal displacements $b(\vec{x}, t)$ in (1)) to generate

synthetic observational data with 1% relative added noise, at $N_d = 600$ hypothesized seafloor pressure sensors. Using these sparse, noisy observations of acoustic pressure, we then solved the PDE-based inverse problem for the spatiotemporal parameter field representing seafloor velocities during the rupture event, and made predictions of the QoI (sea surface wave height) at $N_q = 21$ forecast locations. We note that even a small number of sea surface wave height forecasts is informative for early warning purposes [28].

To solve this inverse problem, we discretized the acoustic–gravity model (1) with high-order finite elements in MFEM (see §VI-B) using a hexahedral mesh with a spatial resolution of 300 meters (less in shallow areas of the CSZ) with 3.74 billion states and explicit Runge–Kutta time-stepping, with a timestep size dictated by the CFL condition, for a simulation time of 420 seconds. The parameter field was discretized with $N_m = 2{,}416{,}530$ spatial points and $N_t = 420$ temporal steps (1 Hz frequency) for a total parameter dimension of $\sim$1.015 billion. Results of the parameter inference and QoI prediction under uncertainty are shown in Figs. 3 and 4; note that the inverse solution is depicted in terms of the seafloor normal displacement, $\int_0^T m(\vec{x}, t)dt$, which is visually simpler to interpret than snapshots of the inferred spatiotemporal parameter field (seafloor velocities). The computation times and other performance-related aspects for this inverse problem are presented in §VII.

VI. How Performance Was Measured

Our inference and prediction framework consists of two code bases. The first is the "Cascadia application code," a C++ solver for the acoustic–gravity model (1) implemented in MFEM. This code is used in the (offline) Phase 1 of our framework to compute adjoint PDE solutions that define the p2o map $\mathbf{F}$ and p2q map $\mathbf{F}_q$. Both AMD (via HIP) and NVIDIA (via CUDA) GPU-based systems are supported; we report performance results for both types of systems.

The second code, which implements Phases 2–4 of our framework, is an FFT-based C++/CUDA code that exploits the block Toeplitz structure of $\mathbf{F}$ and $\mathbf{F}_q$, together with the other algorithmic innovations described in the preceding section, to solve the real-time Bayesian parameter inference and QoI prediction problems. This code makes extensive use of CUDA-exclusive libraries, including cuDSS and cuSOLVERMp, among others, so we report performance results on NVIDIA GPU-based systems.

All computations are performed in double precision; for ill-posed inverse problems like ours, single precision is unstable.

A. HPC systems

LLNL's *El Capitan* is an HPE Cray EX supercomputer with 11,136 nodes, each containing 4 AMD Instinct™ MI300A Accelerated Processing Units (APUs). The MI300A consists of 24 Zen4-based CPU cores, a CDNA3-based GPU, and 128 GB of HBM3 memory, integrated onto a single package. Each MI300A has a peak double precision throughput of 61.3 TFLOP/s for a total peak of the machine of 2.73 EFLOP/s. The system nodes are connected by an HPE Slingshot-200 dragonfly interconnect with at most three network hops between any two nodes.

NERSC's *Perlmutter* is an HPE Cray Shasta supercomputer with 1,536 nodes, each containing an AMD EPYC™ 7763 64-core CPU and 4 NVIDIA A100 GPUs. Each GPU has 40 GB of HBM2 memory and a peak double precision throughput of 9.7 TFLOP/s for a total system peak of 59.6 PFLOP/s. The system is connected by a Slingshot-11 dragonfly interconnect with at most three network hops between any two nodes.

B. MFEM library

MFEM is a C++ finite element (FE) library that provides high-performance discretization algorithms to a wide range of HPC application codes in national laboratories, industry, and academia [29]. In addition to SoA support for high-order methods, MFEM has been designed to be highly scalable on the full range of computing platforms, from laptops to GPU-accelerated supercomputers [30]. Building on efforts in the U.S. Exascale Computing Project [31], the library has put a special emphasis on GPU architectures. An important algorithmic innovation that enables MFEM to achieve high performance on GPU hardware is the concept of *FE operator decomposition* that exposes data parallelism in high-order FE computations [32], and enables additional optimizations on quadrilateral and hexahedral grids by taking advantage of the Cartesian product structure at the element level in the decomposition. Critically, MFEM's approach also leads to a significantly reduced memory requirement through *partial assembly* (PA), which stores an asymptotically optimal amount of data: $\mathcal{O}(1)$ per degree of freedom (DOF). Compared to the classical full (global sparse matrix) or element-level (local dense matrices) assembly algorithms, PA leads to orders-of-magnitude faster execution times on both CPU and GPU architectures. Another option supported by MFEM is *matrix-free* (MF) assembly where no data is stored and all computations are done on the fly; see Fig. 7. In this work, we focus on faster time-to-solution (as opposed to higher FLOP/s), so we use the PA approach to discretize the forward problem (1).

C. Cascadia application code

The acoustic–gravity forward problem (1), as well as its adjoint problem, were cast into a mixed variational formulation and then discretized with MFEM using the Galerkin FE method and explicit fourth-order Runge–Kutta (RK4) time-stepping. The FE discretization uses fourth-order continuous (H^1-conforming) scalar-valued pressure and third-order discontinuous (L^2-conforming) velocity components.

Table I summarizes the main steps of the Cascadia application for computing adjoint PDE solutions in Phase 1 of our framework. The timers represent wall time; they are implemented with standard POSIX clock functions, called after `{hip,cuda}DeviceSynchronize` and `MPI_Barrier`.

While performance is measured for the entire application, the computational expense is overwhelmingly dominated by the cost of the acoustic–gravity wave propagation solver. In

TABLE I
CASCADIA APPLICATION CODE: TIMERS

Timer	Main tasks
Initialization	Initialize MPI devices
Setup	Read and partition mesh
	Partially assemble the mixed FE operator
	Precompute parameter-to-state mapping
	Precompute state-to-observation operator
Adjoint p2o/p2q	Apply transpose observation operator (RHS)
	Solve adjoint acoustic–gravity wave propagation
	Retrieve p2o/p2q column vector from adjoint state
I/O	Write adjoint p2o/p2q column vector to file

particular, the dominant cost is the repetitive application (four applications per RK4 timestep) of the time-stepping operator

$$[\mathbf{u}_\delta \ \mathbf{p}_\delta]^T = \mathbf{M}^{-1}\left(-\mathbf{A}\,[\mathbf{u}\ \mathbf{p}]_i^T + [\mathbf{f}\ \mathbf{g}]_i^T\right),$$

where $[\mathbf{u}\ \mathbf{p}]_i^T$ and $[\mathbf{f}\ \mathbf{g}]_i^T$ are respectively the state and right-hand-side (RHS) vectors at time instance i, and $[\mathbf{u}_\delta\ \mathbf{p}_\delta]$ is the state increment used by RK4. The (lumped) mass matrix $\mathbf{M}$ and stiffness matrix $\mathbf{A}$ are discretizations of the block operators M and A defined by:

$$\left(M\begin{bmatrix}\vec{u}\\ p\end{bmatrix}, \begin{bmatrix}\vec{\tau}\\ v\end{bmatrix}\right) := \begin{bmatrix}(\rho\vec{u},\vec{\tau}) & 0\\ 0 & (K^{-1}p,v)+\langle(\rho g)^{-1}p,v\rangle_{\partial\Omega_s}\end{bmatrix}$$

and

$$\left(A\begin{bmatrix}\vec{u}\\ p\end{bmatrix}, \begin{bmatrix}\vec{\tau}\\ v\end{bmatrix}\right) := \begin{bmatrix}0 & (\nabla p,\vec{\tau})\\ -(\vec{u},\nabla v) & \langle Z^{-1}p,v\rangle_{\partial\Omega_a}\end{bmatrix}, \qquad (4)$$

where $\vec{u},\vec{\tau} \in (L^2(\Omega))^3$ and $p,v \in H^1(\Omega)$; $(\cdot,\cdot)$ denotes the (component-wise) $L^2(\Omega)$ inner product, and $\langle\cdot,\cdot\rangle_{\partial\Omega}$ is the $L^2(\partial\Omega)$ inner product over (part of) the boundary $\partial\Omega$.

TABLE II
SCALABILITY SETUP ON THE EL CAPITAN AND PERLMUTTER SYSTEMS

			Weak scaling	Strong scaling
Nodes	GPUs	Processor grid	Elements	Elements/GPU
85	340	5 × 17 × 4	1,693,450,240	4,980,736
170	680	10 × 17 × 4	3,386,900,480	2,490,368
340	1,360	10 × 34 × 4	6,773,800,960	1,245,184
680	2,720	20 × 34 × 4	13,547,601,920	622,592
1,360	5,440	20 × 68 × 4	27,095,203,840	311,296
2,720	10,880	40 × 68 × 4	54,190,407,680	155,648
5,440	21,760	40 × 136 × 4	108,380,815,360	77,824
10,880	43,520	80 × 136 × 4	216,761,630,720	38,912
47	188	1 × 47 × 4	295,698,432	1,572,864
94	376	2 × 47 × 4	591,396,864	786,432
188	752	2 × 94 × 4	1,182,793,728	393,216
376	1,504	4 × 94 × 4	2,365,587,456	196,608
752	3,008	4 × 188 × 4	4,731,174,912	98,304
1,504	6,016	8 × 188 × 4	9,462,349,824	49,152

The primary performance metric that matters from our application point of view is therefore *runtime per timestep*, which we report in our weak and strong scalability studies. Reported runtimes were averaged over ten consecutive timesteps (40 operator applications), discarding the initial ten timesteps as warm-up. The problem sizes we used to test scalability on the *El Capitan* and *Perlmutter* systems are summarized in Table II. Since runtime per timestep is the primary application-relevant performance metric, we optimized GPU-kernel performance for *DOF throughput* rather than total achieved FLOP/s (see §VII-B); FLOP counts and memory access rates on *El Capitan* were analyzed using the ROCm Compute profiler.

D. FFT-based Bayesian inference code

The FFT-based algorithm used to enable fast Hessian matvecs is completely agnostic to the particular application; it is a general purpose code that can be used to calculate matvecs with any block-triangular Toeplitz matrix. Examples of other digital twin applications that fit into this framework are discussed in §VIII. This code employs the cuFFT and cuBLAS libraries to compute batched FFTs and batched dense matvecs, respectively. Performance of the matvecs was measured with the NSight Compute profiler. The primary performance goal of this code is to achieve fast time-to-solution. FFT matvec wall timings were measured with `omp_get_wtime` after calling `cudaDeviceSynchronize` and `MPI_Barrier`. The implementation of the FFT matvec algorithm is available open-source at https://github.com/s769/FFTMatvec.

The prior solves used to form the $\mathbf{G}^*$ matrix were implemented with the cuDSS library. Linear solves with the dense, symmetric $\mathbf{K}$ matrix were implemented via Cholesky factorization through the cuSOLVERMp library.

VII. PERFORMANCE RESULTS

A. Scalability

We present weak and strong scalability results (Fig. 5) on the *El Capitan* and *Perlmutter* supercomputers. The computational cost of our inference and prediction framework is overwhelmingly dominated by the cost of the adjoint wave propagation PDE solutions in Phase 1 (offline) that are used to construct the p2o map $\mathbf{F}$ and p2q map $\mathbf{F}_q$. The computational cost of each adjoint PDE solution, in turn, is overwhelmingly dominated by the RK4 time-stepping solver; each time step involves four applications of the PDE operator. For this offline phase, scalability to large problem sizes is critical (in addition to throughput), since we wish to apply our framework to other settings beyond Cascadia where tsunamis can impact populations across long distances (such as occurred in the 2004 Sumatra–Andaman earthquake).

On *El Capitan*, the solver maintained 92% weak parallel efficiency over a 128-fold increase in problem size, from 85 nodes (340 AMD MI300A GPUs) to 10,880 nodes (43,520 GPUs) of the full system. The largest problem involved over 55.5 trillion DOF, which, to the best of our knowledge, is *the largest reported unstructured FE computation*. This computation used about 86% (4.8 PB) of the total available machine memory (5.57 PB). For the largest problem fitting on 340 GPUs (434 billion DOF), the solver achieved a parallel speedup of 100.9 over a 128-fold increase in GPUs, a strong parallel efficiency of 79%.

We also obtained excellent weak and strong scalability on the *Perlmutter* system over a 32-fold increase in GPUs, from 47 nodes (188 NVIDIA A100 GPUs) to 1,504 nodes (6,016 GPUs). Here, the solver achieved 100% weak efficiency for

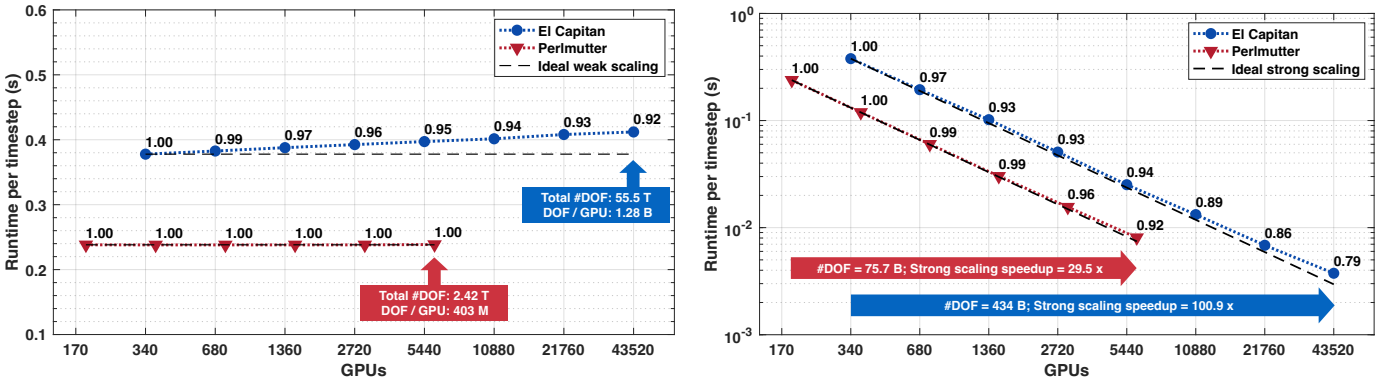

Fig. 5. Weak scalability (left) and strong scalability (right) results on *El Capitan*, from 85 nodes (340 AMD MI300A GPUs) to 10,880 nodes (43,520 GPUs), and on *Perlmutter*, from 47 nodes (188 NVIDIA A100 GPUs) to 1,504 nodes (6,016 GPUs). Numbers along the graph lines indicate parallel efficiency.

a problem with 403 million DOF per GPU and 92% strong efficiency for a problem involving 75.7 billion DOF.

While GPU-based systems have been the focus of our performance optimizations, we want to emphasize that our solver also exhibits excellent weak and strong scalability on CPU-based systems. For example, on TACC's *Frontera*[2] supercomputer, we achieve an outstanding weak efficiency of 95% over a 8,192-fold increase in problem size, from 1 node (56 cores) to 8,192 nodes (458,752 cores), with the largest problem involving 2.20 trillion DOF (4.80 million DOF per core), and a strong efficiency of 70% over a 128-fold increase in cores, from 3,584 cores to 458,752 cores.

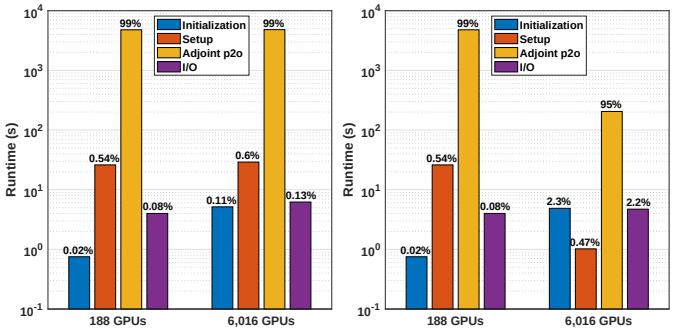

Fig. 6. Application timers (as defined in Table I) in the weak scalability (left) and strong scalability (right) limit on *Perlmutter*. Adjoint p2o and I/O timers, each measured for 200 timesteps, are projected to 20,000 timesteps. Numbers on bars indicate percentage of application runtime. Note log scale.

Since small timesteps are needed to resolve fast acoustic waves, the total number of timesteps for the wave propagation solver is usually large. For example, computation of the CSZ rupture scenario on our FE mesh requires on the order of at least $\mathcal{O}(10^4)$ time steps. Fig. 6 plots application timers (as defined in Table I), where adjoint p2o and I/O timers, each measured for 200 timesteps, are projected to 20,000 timesteps, to illustrate that the time for job initialization, setup, and I/O is negligible in the weak scalability limit but has minor impact in the strong scalability limit, where the entire application runtime is still overwhelmingly dominated by the solver.

[2]*Frontera* hosts 8,368 Intel Xeon Platinum 8280 ("Cascade Lake") compute nodes contained in 101 racks. Each node has 56 cores on two sockets and 192 GB of DDR4 RAM; see https://docs.tacc.utexas.edu/hpc/frontera/#system.

B. Peak performance

While the existing methods in MFEM already scaled well for the forward problem, we introduced additional GPU performance and memory optimizations to further reduce the runtime and increase the size of the problems we can simulate. Some of the performance optimizations for the two key kernels in (4) are shown in Fig. 7. These kernels account for the majority of the time in each stage of the RK4 time-stepping.

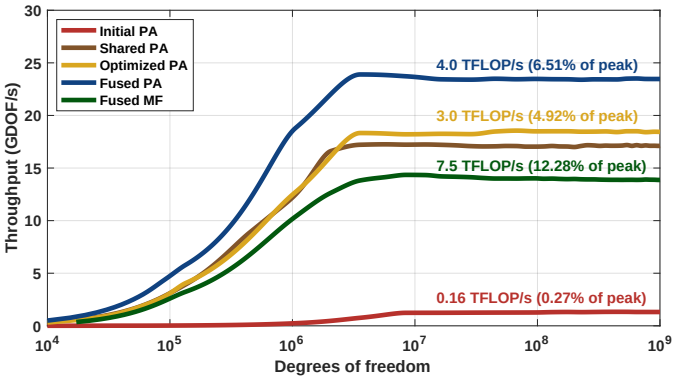

Fig. 7. Throughput, in billion degrees of freedom (GDOF) per second, for the kernels corresponding to the off-diagonal blocks in (4) on a single GPU of *El Capitan*. Measured FLOP/s rates with percentage of peak (61.3 TFLOP/s) are shown in the saturated regime for each curve. By utilizing the GPU shared memory we improved the initial partial assembly (PA) kernels (red) to a version (brown) that is 13× faster. Specifying explicit launch bounds for the kernels allowed us to additionally improve performance (yellow), which was the version of the kernels used in the scaling runs in Fig. 5. Further optimizations by fusing the actions of the two operators into one kernel allowed us to achieve a peak performance of 24 GDOF/s (blue) corresponding to 4 TFLOP/s, or 6.51% of FLOP/s peak. Although the matrix-free (MF) approach (green) attains higher FLOP/s, its time-to-solution (DOF throughput) remains lower than that of Fused PA.

We also performed extensive optimization of the memory usage, including: instrumenting the code to track memory usage separately on host and device; taking advantage of the unified memory on the MI300A APU by freeing host allocation so they can be used for GPU allocations; using sparsity in the RHS to avoid storing full vectors; fusing permutation operators with operator action kernels; recomputing determinants of Jacobian matrices instead of storing them; batching multiple small memory allocations together; switch-

ing to 64-bit integers for certain variables local to the GPU (e.g., multi-dimensional quadrature point data); and carefully reusing temporary vectors from RK4 for temporary vectors needed in the operator action. With all these optimizations for a problem of size 67 million DOF per APU, we improved unified memory usage from (5.2 host + 30.7 device) to (1.1 host + 5.64 device) GiB/APU, a reduction of 5.33×. These improvements allowed us to fit ∼1.28 billion DOF per APU.

Our primary performance metric is time-to-solution, which, for the PDE operator (4), is measured as DOF throughput. For a given problem size, higher throughput is directly proportional to faster computation time. As shown in Fig. 7, more FLOP/s do not imply faster time-to-solution: the best-performing implementation, Fused PA, has a 1.9× lower percentage of FLOP/s peak compared to Fused MF, but is 1.6× faster. The reason for this is that although Fused MF requires less memory, it incurs a higher computational cost, with an arithmetic intensity of 17.2 FLOP/byte compared to Fused PA's 2.5 FLOP/byte. Although the percentage of peak FLOP/s in Fig. 7 is not as high as for dense matrix operations, we note that it is much better than the results reported for the HPCG benchmark on comparable systems[3] (e.g., 0.7% of peak of *Frontier's* MI250Xs). Similar to the type of computations performed in FE algorithms, HPCG has a FLOP count and memory access rate proportional to the number of DOF.

All GPU kernels involved in the FFT matvec implementation (§VI-D) are memory-bound and achieve 80–95% of peak memory bandwidth on NVIDIA A100 and GH200 GPUs [26].

C. Time-to-solution

We measured time-to-solution for a margin-wide rupture scenario in the CSZ for which we presented inversion results in §V. These computations were performed on 128 *Perlmutter* nodes (512 NVIDIA A100 GPUs) and involved inference of more than one billion parameters from 252,000 synthetic, noisy observations obtained by 600 hypothesized seafloor pressure sensors, followed by real-time forecasting of the QoI (tsunami wave heights) under uncertainty for 21 QoI locations.

For this problem, Table III summarizes the compute times for each phase of our inference and prediction framework. As discussed above, the total computational expense is dominated by the PDE solutions in the (offline) Phase 1; in these offline computations, scalability is the most important metric, whereas time-to-solution is the metric that matters most for the Phase 4 (online) computations that are executed in real time.

Phase 1 relies on the PDE solver to precompute the p2o map $\mathbf{F}$ and p2q map $\mathbf{F}_q$. For the given configuration (600 sensors, 21 QoI locations), we performed a total 621 PDE solutions, each of which (on average) took 52 minutes on 512 A100 GPUs, for a total of 538 hours of compute time (note that each one of the 621 PDE solutions can be computed independently).

Once Phase 1 was completed, we exploited the block Toeplitz structure of the precomputed $\mathbf{F}$ and $\mathbf{F}_q$ to perform fast FFT-based Hessian matvecs. Each Hessian matvec, conventionally requiring a pair of forward and adjoint PDE solutions that take 104 minutes on 512 A100 GPUs per matvec, could now be performed in 0.024 seconds (averaged over 100 matvecs) on the same number of GPUs, a speedup of 260,000. We re-emphasize that these FFT-based Hessian matvecs are exact and do not incorporate any approximations. The 260,000× speedup derives primarily from the parallel algorithmic innovations, discussed in §V, that massively reduce the overall complexity of the Hessian matvec which, after the precomputations in Phase 1, avoids PDE solutions altogether.

[3]As of April 15, 2025, HPCG results have not been released for *El Capitan*.

TABLE III
COMPUTE TIME FOR EACH PHASE OF THE INFERENCE AND PREDICTION PERFORMED ON PERLMUTTER A100 GPU NODES. TIME-TO-SOLUTION FOR THE ONLINE COMPUTATION (PHASE 4) IS LESS THAN 0.2 SECONDS.

Phase	Task	GPUs	Compute time
1	form $\mathbf{F}:\mathbf{m}\mapsto\mathbf{d}$	512	600×52 m $\sim$ 520 h
	form $\mathbf{F}_q:\mathbf{m}\mapsto\mathbf{q}$	512	21×52 m $\sim$ 18 h
2	form $\mathbf{G}^*:=\mathbf{\Gamma}_{\text{prior}}\mathbf{F}^*$	16	600×4.5 s $\sim$ 45 m
	form $\mathbf{G}_q^*:=\mathbf{\Gamma}_{\text{prior}}\mathbf{F}_q^*$	16	21×4.5 s $\sim$ 1.5 m
	form $\mathbf{K}:=\mathbf{\Gamma}_{\text{noise}}+\mathbf{F}\mathbf{G}^*$	512	$252\text{k}\times 24$ ms $\sim$ 100 m
	factorize $\mathbf{K}$	25	22 s
3	compute $\mathbf{\Gamma}_{\text{post}(q)}$	512	$8{,}820\times 150$ ms $\sim$ 25 m
	compute $\mathbf{Q}:\mathbf{d}\mapsto\mathbf{q}$	512	$8{,}820\times 150$ ms $\sim$ 25 m
4	infer parameters $\mathbf{m}_{\text{map}}$	512	<0.2 s
	predict QoI $\mathbf{q}_{\text{map}}$	1	<1 ms

These rapid Hessian matvecs are the workhorse for the remaining offline computations (Phases 2–3). For example, forming the data space Hessian $\mathbf{K}$ requires 252,000 matvecs—an intractable task if each matvec required a forward/adjoint pair of PDE solutions—which we computed with the FFT-based matvecs in just 100 minutes. Finally, real-time inference of the parameters (seafloor motion) and forward prediction of the QoI (surface wave heights) was achieved in *less than 0.2 seconds* of compute time on 512 A100 GPUs.

We conclude by noting that computing the inverse solution with SoA methods (see §IV), without the innovations described in §V, would not only have been prohibitive in real time, but would have been intractable (50 years on 512 A100 GPUs). With the advances described in this paper, the computing time reduces to a one-time offline computation of 621 (N_d+N_q) adjoint wave propagations, requiring 538 hours on 512 GPUs and representing an ∼810× reduction in PDE solutions. Then, for each earthquake that excites the sensors, the online solution of the Bayesian inverse problem requires <0.2 s, a factor of 10^{10} speedup over the SoA CG algorithm.

VIII. IMPLICATIONS

We enable real-time probabilistic tsunami forecasts for near-field events such as a future magnitude 8–9 CSZ megathrust earthquake, where conventional systems may fail due to delayed or inaccurate source characterization. Our digital twin framework resolves complex rupture dynamics and tsunami wavefields at unprecedented speed and accuracy, enabling early predictions of inundation and evacuation zones within seconds, critical for life-saving response where warning time is otherwise measured in minutes or less. Notably, the real-time inversion itself requires only moderate computational resources. Moreover, if only surface wave heights at selected

locations are of interest, the forecasting step reduces to a precomputed, small, dense matrix–vector product—enabling deployment entirely without any HPC infrastructure.

While we demonstrate the potential for real-time forecasting, we remain limited by the sparsity of offshore sensors currently available in the CSZ; however, emerging technologies such as distributed acoustic sensing will improve observational coverage for resolving near-field tsunami source characteristics [33, 34]. Furthermore, expanding to fully-coupled acoustic–elastic simulations [35] allows us to employ our framework to invert for fault slip, and forward propagate seismic waves to compute—in real time—maps of the intensity of ground motion in populated regions. These *shake maps* provide critical information for early responders and post-earthquake recovery.

More generally, autonomous dynamical systems arise in many different settings beyond geophysical inversion. Our Bayesian inversion-based digital twin framework is thus more broadly applicable to such problems as acoustic, electromagnetic, and elastic inverse scattering; source inversion for transport of atmospheric or subsurface hazardous agents; satellite inference of emissions; and treaty verification, among others.

Acknowledgment

The authors thank Pythagoras Watson and Livermore Computing for assistance with *El Capitan* runs, Amanda Dufek and Muaaz Awan for assistance with *Perlmutter* runs, John Cazes for assistance with *Frontera* runs, Jonatan Glehman for assistance in generating earthquake simulation output, and Eric Dunham for insightful discussions about this work.

References

[1] O. Kamigaichi et al. “Earthquake early warning in Japan: Warning the general public and future prospects”. In: *Seismol. Res. Lett.* 80.5 (2009), pp. 717–726.

[2] B. Hirshorn et al. “Earthquake Source Parameters: Rapid Estimates for Tsunami Forecasts and Warnings”. In: *Complexity in Tsunamis, Volcanoes, and their Hazards*. Springer, 2021, pp. 299–333.

[3] R. J. LeVeque et al. “Developing a warning system for inbound tsunamis from the Cascadia subduction zone”. In: *OCEANS 2018 MTS/IEEE Charleston*. IEEE. 2018, pp. 1–10.

[4] C. Uphoff et al. “Extreme scale multi-physics simulations of the tsunamigenic 2004 Sumatra megathrust earthquake”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2017, pp. 1–16.

[5] H. Tsushima, R. Hino, H. Fujimoto, Y. Tanioka, and F. Imamura. “Near-field tsunami forecasting from cabled ocean bottom pressure data”. In: *J. Geophys. Res. Solid Earth* 114.B6 (2009).

[6] B. W. Crowell. “Chapter 6 - Earthquake and tsunami early warning with GNSS data”. In: *GNSS Monitoring of the Terrestrial Environment*. Ed. by Y. Aoki and C. Kreemer. Elsevier, 2024, pp. 111–127.

[7] L. S. Abrahams, L. Krenz, E. M. Dunham, A.-A. Gabriel, and T. Saito. “Comparison of methods for coupled earthquake and tsunami modelling”. In: *Geophys. J. Int.* 234.1 (2023), pp. 404–426.

[8] J. Behrens et al. “Probabilistic tsunami hazard and risk analysis: A review of research gaps”. In: *Front. Earth Sci.* 9 (2021), p. 628772.

[9] K. Willcox et al. *Foundational Research Gaps and Future Directions for Digital Twins*. National Academies of Sciences, Engineering, and Medicine study. https://doi.org/10.17226/26894. 2024.

[10] B. F. Atwater et al. “Summary of coastal geologic evidence for past great earthquakes at the Cascadia subduction zone”. In: *Earthq. Spectra* 11.1 (1995), pp. 1–18.

[11] C. Goldfinger et al. *Turbidite event history—Methods and implications for Holocene paleoseismicity of the Cascadia subduction zone*. Tech. rep. https://doi.org/10.3133/pp1661F. U.S. Geological Survey, 2012.

[12] C. R. Barnes, M. M. R. Best, F. R. Johnson, and B. Pirenne. “NEPTUNE Canada: Installation and initial operation of the world’s first regional cabled ocean observatory”. In: *Seafloor Observations: A New Vision of the Earth from the Abyss*. Springer, 2015, pp. 415–438.

[13] C. R. Vogel. *Computational Methods for Inverse Problems*. Frontiers in Applied Mathematics. Philadelphia, PA: SIAM, 2002, pp. xvi+183.

[14] O. Ghattas and K. Willcox. “Learning physics-based models from data: perspectives from inverse problems and model reduction”. In: *Acta Numer.* 30 (2021), pp. 445–554.

[15] A. M. Stuart. “Inverse problems: A Bayesian perspective”. In: *Acta Numer.* 19 (2010), pp. 451–559.

[16] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*. Vol. 160. Applied Mathematical Sciences. Springer-Verlag New York, 2005. xvi+339.

[17] T. Isaac, N. Petra, G. Stadler, and O. Ghattas. “Scalable and efficient algorithms for the propagation of uncertainty from data through inference to prediction for large-scale problems, with application to flow of the Antarctic ice sheet”. In: *J. Comput. Phys.* 296 (2015), pp. 348–368.

[18] T. Bui-Thanh, O. Ghattas, J. Martin, and G. Stadler. “A computational framework for infinite-dimensional Bayesian inverse problems Part I: The linearized case, with application to global seismic inversion”. In: *SIAM J. Sci. Comput.* 35.6 (2013), A2494–A2523.

[19] C. Cui et al. “GLAD-M35: A joint P and S global tomographic model with uncertainty quantification”. In: *Geophys. J. Int.* 239.1 (2024), pp. 478–502.

[20] G. C. Lotto and E. M. Dunham. “High-order finite difference modeling of tsunami generation in a compressible ocean from offshore earthquakes”. In: *Comput. Geosci.* 19.2 (2015), pp. 327–340.

[21] S. Henneking, S. Venkat, and O. Ghattas. “Goal-oriented real-time Bayesian inference for linear autonomous dynamical systems with application to digital twins for tsunami early warning”. In: *arXiv:2501.14911* (2025).

[22] I. Ambartsumyan et al. “Hierarchical matrix approximations of Hessians arising in inverse problems governed by PDEs”. In: *SIAM J. Sci. Comput.* 42.5 (2020), A3397–A3426.

[23] L. Cao, T. O’Leary-Roseberry, and O. Ghattas. “Derivative-informed neural operator acceleration of geometric MCMC for infinite-dimensional Bayesian inverse problems”. In: *J. Mach. Learn. Res.* (2025). To appear, pp. 1–67.

[24] C. Greif and K. Urban. “Decay of the Kolmogorov N-width for wave problems”. In: *Appl. Math. Lett.* 96 (2019), pp. 216–222.

[25] R. M. Gray. “Toeplitz and circulant matrices: A review”. In: *Found. Trends Commun. Inf. Theory* 2.3 (2006), pp. 155–239.

[26] S. Venkat, M. Fernando, S. Henneking, and O. Ghattas. “Fast and scalable FFT-based GPU-accelerated algorithms for Hessian actions arising in linear inverse problems governed by autonomous dynamical systems”. In: *arXiv:2407.13066* (2024).

[27] J. Glehman et al. “Partial ruptures governed by the complex interplay between geodetic slip deficit, rigidity, and pore fluid pressure in 3D Cascadia dynamic rupture simulations”. In: *EarthArXiv* (2024).

[28] C. M. Liu, D. Rim, R. Baraldi, and R. J. LeVeque. “Comparison of machine learning approaches for tsunami forecasting from sparse observations”. In: *Pure Appl. Geophys.* 178.12 (2021), pp. 5129–5153.

[29] R. Anderson et al. “MFEM: A modular finite element methods library”. In: *Comput. Math. Appl.* 81 (2021), pp. 42–74.

[30] J. Andrej et al. “High-performance finite elements with MFEM”. In: *Int. J. High Perform. Comput. Appl.* 38.5 (2024), pp. 447–467.

[31] *CEED: Center for Efficient Exascale Discretizations in the U.S. Exascale Computing Project (ECP)*. https://ceed.exascaleproject.org.

[32] P. Fischer et al. “Scalability of high-performance PDE solvers”. In: *Int. J. High Perform. Comput. Appl.* 34.5 (2020), pp. 562–586.

[33] D. Schmidt et al. *Earthquake and tsunami early warning on the Cascadia subduction zone: A feasibility study for an offshore geophysical monitoring network*. Tech. rep. https://hdl.handle.net/1773/50968. University of Washington, 2023.

[34] C. Becerril et al. “Towards tsunami early-warning with distributed acoustic sensing: Expected seafloor strains induced by tsunamis”. In: *Authorea Preprints* (2024).

[35] L. Krenz et al. “3D acoustic-elastic coupling with gravity: The dynamics of the 2018 Palu, Sulawesi earthquake and tsunami”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2021, pp. 1–14.