# Scalable Bayesian Inference of Large Simulations via Asynchronous Prefetching Multilevel Delayed Acceptance

Maximilian Kruse
Karlsruhe Institute of Technology
Karlsruhe, Germany
maximilian.kruse@kit.edu

Zihua Niu
LMU Munich
Munich, Germany
zihua.niu@lmu.de

Sebastian Wolf
Technical University of Munich
Munich, Germany
wolf.sebastian@cit.tum.de

Mikkel B. Lykkegaard
Danish Technological Institute
Taastrup, Denmark
mbly@teknologisk.dk

Michael Bader
Technical University of Munich
Munich, Germany
bader@cit.tum.de

Alice-Agnes Gabriel
UCSD & LMU Munich
San Diego, USA & Munich, Germany
algabriel@ucsd.edu

Linus Seelinger
Karlsruhe Institute of Technology
Karlsruhe, Germany
linus.seelinger@kit.edu

## ABSTRACT

Bayesian inference enables greater scientific insight into simulation models, determining model parameters and meaningful confidence regions from observed data. With hierarchical methods like Multilevel Delayed Acceptance (MLDA) drastically reducing compute cost, sampling Bayesian posteriors for computationally intensive models becomes increasingly feasible. Pushing MLDA towards the strong scaling regime (i.e. high compute resources, short time-to-solution) remains a challenge: Even though MLDA only requires a moderate number of high-accuracy simulation runs, it inherits the sequential chain structure and need for chain burn-in from Markov chain Monte Carlo (MCMC). We present fully asynchronous parallel prefetching for MLDA, adding an axis of scalability complementary to forward model parallelization and parallel chains. A thorough scaling analysis demonstrates that prefetching is advantageous in strong scaling scenarios. We investigate the behavior of prefetching MLDA in small-scale test problems. A large-scale geophysics application, namely parameter identification for non-linear earthquake modelling, highlights interaction with coarse-level quality and model scalability.

## CCS CONCEPTS

• **Computing methodologies → Distributed computing methodologies**; **Modeling and simulation**.

## KEYWORDS

Uncertainty quantification, Bayesian inference, high performance computing, parallelization, geophysics, earthquake modeling

**ACM Reference Format:**
Maximilian Kruse, Zihua Niu, Sebastian Wolf, Mikkel B. Lykkegaard, Michael Bader, Alice-Agnes Gabriel, and Linus Seelinger. 2025. Scalable Bayesian Inference of Large Simulations via Asynchronous Prefetching Multilevel Delayed Acceptance. In *Platform for Advanced Scientific Computing Conference (PASC '25), June 16–18, 2025, Brugg-Windisch, Switzerland*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3732775.3733581

## 1 INTRODUCTION

Mathematical inverse problems provide a rich framework for gaining insight into a multitude of problems and phenomena in science and technology by combining models and data to infer unknown parameters, such as material properties, reaction rates, or effective potentials [4, 20, 33, 41]. Observational data is typically noisy and incomplete, while computational models only approximate the true system. Consequently, modern research increasingly focuses on incorporating these uncertainties into inference problems. A common approach to Uncertainty Quantification (UQ) is to formulate an inverse problem in the Bayesian framework [28], assigning a posterior probability to unknown parameters based on prior knowledge as well as goodness-of-fit between model prediction and observational data.

An attractive choice for solving Bayesian inverse problems (BIP) are Markov chain Monte Carlo (MCMC) [8, 13, 22, 25, 36] algorithms, as they impose minimal assumptions on the model. However, this generality comes at the price of requiring large numbers of model evaluations. This cost can easily become excessive, particularly for inverse problems governed by partial differential equations (PDE) that already require High Performance Computing (HPC) resources for a single evaluation.

Multilevel Markov chain Monte Carlo (MLMCMC) [11] exploits a hierarchy of models with varying fidelities, allocating most of the computational effort to less accurate but fast approximate models. MLMCMC thus achieves efficiency gains comparable to Multilevel Monte Carlo (MLMC) [21] in uncertainty propagation. MLMCMC relies on the assumption of independent samples from coarser levels, which serve as proposals for the finer levels. However, in practice,

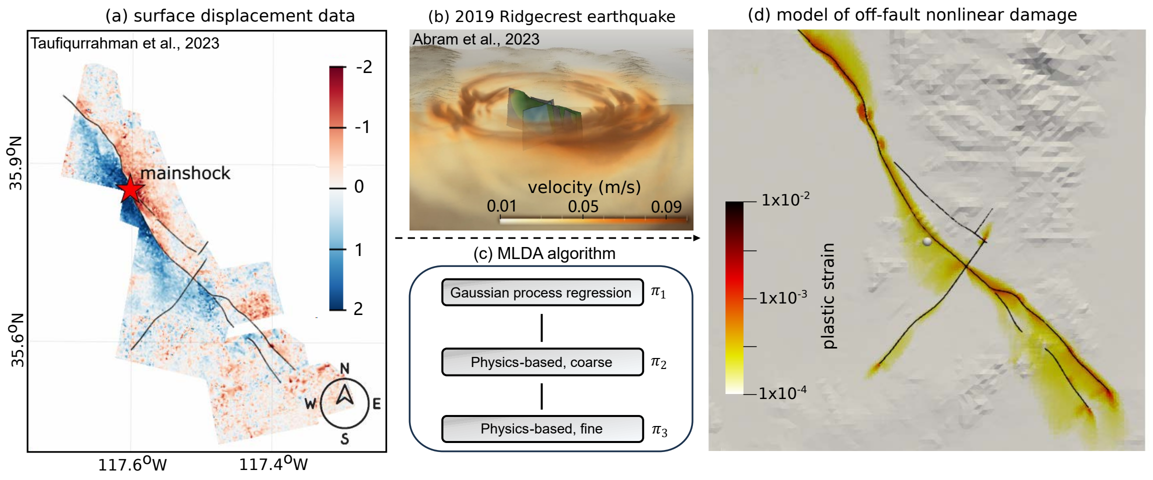


**Figure 1: Overview of the Multilevel Delayed Acceptance MCMC (MLDA) algorithm and its target application, the 2019 Ridgecrest earthquake. (a) Surface displacement data during the earthquake [50]. (b) Visualization of the simulated earthquake source and the generated seismic wave field [2]. (c) Structure of the proposed MLDA model hierarchy. (d) Modeled plastic deformation, which is controlled by physical parameters, the target of the inference in this work.**

this independence only holds approximately, introducing bias. The more recent MLDA [34], based on delayed-acceptance MCMC [7], addresses this limitation. Compared to MLMCMC, which has been successfully parallelized and deployed on HPC systems [47], MLDA introduces stronger data dependencies between levels, and is therefore more challenging to parallelize.

In this work, we introduce a novel parallelization approach for the MLDA algorithm based on prefetching [3, 49], the parallel evaluation of possible future Markov Chain states based on a binary decision tree. Prefetching acts as an additional level of parallelism on top of model parallelism and parallel execution of multiple MLDA chains, each of which naturally exhibit diminishing returns.

We discuss the theoretical basis for prefetching in a multilevel context. On an example problem, we investigate performance by comparing prefetching to parallelization with multiple chains. The results indicate that prefetching is in fact needed for optimal parallel efficiency in the strong scaling regime, i.e. where many compute resources are employed to rapidly achieve an inversion result.

Our prefetching MLDA implementation supports the UQ and Modeling Bridge (UM-Bridge) interface [45], leading to a modular pipeline that transparently scales to HPC clusters and allows linking to any simulation code. It is available open-source at [31].

Finally, we apply prefetching MLDA to a novel, computationally challenging inverse problem in geoscience. It aims to infer subsurface material parameters in a non-linear seismic model using earthquake data. We use three-dimensional, non-linear dynamic rupture models of the Ridgecrest earthquakes (similar to [50]), California's biggest earthquakes in more than 20 years which ruptured multiple segments of a complex fault system [43]. Our inversion aims to explain high-rate global positioning system datasets with earthquake physics. The model is implemented in the earthquake simulation software SeisSol [24, 30, 53], which integrates seamlessly with our computational pipeline. We construct the multilevel model hierarchy by varying the PDE discretization level, and augment it with an adaptive surrogate model based on Gaussian Process (GP) regression [42] on the coarsest level.

## 2 ASYNCHRONOUS PREFETCHING MLDA

Bayesian inverse problems define, for some parameter of interest $\theta \in \mathbb{R}^d$, a posterior distribution $\pi_{\text{post}}(\theta|d_{\text{obs}})$, given observations $d_{\text{obs}} \in \mathbb{R}^q$ of the system under consideration. In this work, we consider scenarios where the data is the perturbed output of a PDE model, given a specific parameter set as its input, $d^{\text{obs}} = G(\theta) + \eta$. $\eta \in \mathbb{R}^q$ is the realization of a noise variable with known statistics. This induces a likelihood $l(d_{\text{obs}}|\theta)$ to observe the data, given a parameter candidate. In combination with some prior distribution $\pi_{\text{prior}}(\theta)$, Bayes' theorem yields the posterior as

$$\pi(\theta) \equiv \pi_{\text{post}}(\theta|d_{\text{obs}}) \propto l(d_{\text{obs}}|\theta)\pi_{\text{prior}}(\theta). \tag{1}$$

*MLDA algorithm.* MCMC methods generate correlated samples from a target distribution $\pi$ (in our scenario a Bayesian posterior). Given a sample $\theta_n \in \mathbb{R}^d$, MCMC algorithms generate a new sample $\theta_{n+1}$ in a two-step procedure. Firstly, a proposal candidate $\tilde{\theta}_{n+1}$ is drawn from a distribution $q(\cdot|\theta_n)$, which is typically cheap to evaluate. The proposal is then accepted ($\theta_{n+1} = \tilde{\theta}_{n+1}$) or rejected ($\theta_{n+1} = \theta_n$) according to a transition probability $\alpha(\tilde{\theta}_{n+1}|\theta_n)$. Importantly, proposals tend to be cheap to compute, while the transition probability depends on costly target density evaluations $\pi(\tilde{\theta}_{n+1})$ and $\pi(\theta_n)$.

A major drawback of MCMC methods is that they typically require a large number of evaluations of the target density to achieve a sufficient number of effectively uncorrelated samples. Multilevel algorithms may drastically reduce that cost. We employ the MLDA algorithm [34] in this work. The basic idea of MLDA (as for other multilevel methods) is to employ a hierarchy of models with different accuracy-cost tradeoff. We denote this hierarchy as subchain levels $s = 1, 2, \dots, K$ with corresponding target densities $\pi_s$. We presume that densities on lower levels are cheap to compute but coarse approximations of the posterior, whereas evaluations on higher levels are computationally more expansive, but more exact. We set $\pi = \pi_K$. MLDA generates high-quality proposals on subchain level $l$ by spawning an MCMC chain on subchain level $l-1$ at the current state. The final sample of that coarser-level subchain is then used as a proposal for level $l$. Applying this recursively across levels results in an increasingly fast rate of decorrelation for samples on finer levels. Through a careful choice of model hierarchy and subchain lengths (see [34] for details on convergence rates), MLDA can achieve a significant reduction in overall cost for sampling $\pi$.

*Prefetching MLDA.* Like single-level MCMC algorithms, MLDA is inherently sequential. A trivial method of parallelization is the generation of multiple independent chains. However, since each chain requires a burn-in period and mixing, parallel chains scale at diminishing returns. Thus, we additionally employ within-chain parallelization through prefetching [3, 5, 49]. The underlying concept of prefetching is to expand possible future states of a MCMC chain $\overline{\theta}_{n+j},\ j = 1, 2, \dots$ from a given state $\theta_n$ in a binary tree, representing all future accept/reject decisions. We refer to this structure as a *Markov tree*. Possible future states can be determined in advance, since proposals are typically cheap. We can then conduct the costly target density evaluations for possible future states in parallel. Now, given a pool of $N_p$ workers, $N_p$ target density evaluations across the Markov tree can be conducted simultaneously. With the obtained results, the Markov chain can potentially be advanced by $N_p > 1$ steps. Since we might precompute density evaluations in branches that the MCMC algorithm ultimately does not take, the efficiency of prefetching depends on a "clever" distribution of computational resources onto the possible future states of the chain.

We now introduce an extension of prefetching for MLDA, including fully asynchronous operations accommodating differing run times across levels. We begin by establishing a formal notation for a Markov tree and its nodes. A general Markov tree consists of a sequence of levels $l \in \mathbb{N}_0^+$, not to be confused with the subchain levels $s$ of the MLDA algorithm. We characterize a node $t$ in layer $T_l$ by a string,

$$t \in T_l \text{ with } T_l \subset \{a, r\}^l, \tag{2}$$

indicating the sequence of accepts ($a$) and rejects ($r$) leading to the state of that node. We denote the root level as $T_0$, which only contains the root node $t_0$. Moving on, we define the concatenation operation to create a descendant or child of a given node,

$$\oplus : T_l \times \{a, r\} \to T_{l+1}. \tag{3}$$

Informally, this means that a child node is generated from its parent by appending a letter from $\{a, r\}$ to its defining string. A complete Markov tree is given as the directed acyclic graph resulting from the concatenation of nodes over subsequent layers. Under abuse of notation, we resemble this by the union of layers,

$$T \coloneqq \bigcup_{l \in \mathbb{N}_0^+} T_l. \tag{4}$$

An exemplary Markov tree is depicted in fig. 2.

For the MLDA algorithm, tree nodes carry two additional quantities, their subchain levels $s(t) \in \{1, 2, \dots, K\}$ and their indices

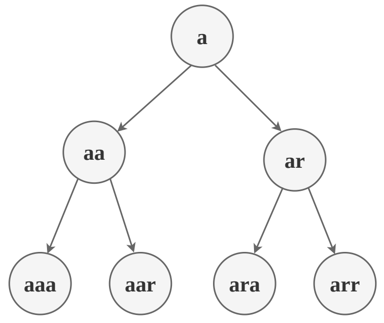


**Figure 2: Three levels of a Markov decision tree. Nodes are labelled with the sequence of accepts and rejects leading to that node.**

$i(t) \in \{1, 2, \dots, I_{s(t)}\}$ within the respective subchain. We define for the root node

$$s(t_0) = K, \quad i(t_0) = 1, \tag{5}$$

and for all $t \in T$, $x \in \{a, r\}$ through recursive concatenation

$$s(t \oplus x) = \begin{cases} s(t) & \text{if } s(t) = 1 \text{ and } i(t) < I_0 \\ s(t) + 1 & \text{if } s(t) < K \text{ and } i(t) = I_{s(t)} \\ s(t) - 1 & \text{else,} \end{cases} \tag{6}$$

and

$$i(t \oplus x) = \begin{cases} i(t) + 1 & \text{if } s(t) = 1 \text{ and } i(t) < I_0 \\ i(v_s(t \oplus x)) + 1 & \text{if } s(t) < K \text{ and } i(t) = I_{s(t)} \\ 1 & \text{else.} \end{cases} \tag{7}$$

Here, we have introduced the mapping $v_s : T \to T$ from a node to its closest ancestor on the same subchain level, i.e. $s(v_s(t)) = s(t)$.

We can now define the states of the nodes in the Markov tree. Assuming that proposals on the coarsest level of the MLDA hierarchy are generated by a distribution $q(\cdot|\cdot)$, we set $\overline{\theta}_{t_0} = \theta_n$ and define subsequent states recursively:

$$\text{If } s(t \oplus x) = s(t) = 1 : \quad \overline{\theta}_{t \oplus x} = \begin{cases} \tilde{\theta}_t \sim q(\cdot|\overline{\theta}_t) & \text{if } x = a, \\ \overline{\theta}_t & \text{if } x = r, \end{cases} \tag{8a}$$

$$\text{If } s(t \oplus x) = s(t) + 1 : \quad \overline{\theta}_{t \oplus x} = \begin{cases} \overline{\theta}_t & \text{if } x = a, \\ \overline{\theta}_{v_s(t \oplus x)} & \text{if } x = r, \end{cases} \tag{8b}$$

$$\text{If } s(t \oplus x) = s(t) - 1 : \quad \overline{\theta}_{t \oplus x} = \overline{\theta}_t. \tag{8c}$$

*Asynchronous prefetching.* In our work, we employ asynchronous prefetching, making use of computational resources as soon as they become available. This is absolutely crucial in MLDA, since model run times may vary extremely across levels: An entire subchain may complete before a previous finer-level node. To this end, we distinguish four computational states for each node in the Markov tree. Nodes in the set $O \subseteq T$ have not been assigned any computational resources yet. Nodes in $W \subseteq T$ are currently running evaluations of the target density. For nodes in $D \subseteq T$, the computation has been completed, and computational resources have been freed. And finally, nodes in $E \subset T$ have been eliminated through MCMC accept/reject decisions.

We make these decisions as soon as the target densities for all relevant nodes have been computed. If an MCMC move from the state $\theta_t$ is accepted, we prune $t \oplus r$ and all its children, i.e. we put these nodes into the set $E$. If the move is rejected, we prune $t \oplus a$ and its subtree. Note that the MLDA algorithm comprises different types of MCMC decisions, each requiring different target densities to be evaluated. For subchains on levels $s = 1, 2, \ldots, K-1$, this facilitates additional within-subchain pruning. On these levels, only the target densities at the first and last subchain index are needed for a subsequent MCMC decision on the next finer level. Consequently, as soon as a succession of three or more nodes in a subchain are uniquely connected, we can make the first node in that sequence the immediate parent of the last node, effectively pruning all intermediates. Given the recursive nature of MLDA's subchains, we would otherwise have to track an extreme amount of subchain nodes before advancing to the next fine-level state.

Next, we devise a strategy for optimally assigning computational resources to nodes in the Markov tree. For this, we assume that we can approximately predict the acceptance probability for a node on subchain level $s$ by a level-dependent estimate $\tilde{\alpha}_s$. Such estimates can be static guesses or adaptive (e.g., depending on the previous behavior of the Markov chain). In addition, we have to take into account that the acceptance probability is either zero or one if the MCMC decision has already been made. We hence define the approximate acceptance rate $\gamma$ as

$$\gamma(t \oplus a) := \begin{cases} \tilde{\alpha}_{s(t\oplus a)} & \text{if } t \oplus a, t \oplus r \notin E, \\ 1 & \text{if } t \oplus r \in E, \\ 0 & \text{if } t \oplus a \in E. \end{cases} \tag{9}$$

Moving on, we recursively construct the estimated probability $P_R(t)$ of a node $t$ to be reached by the actual Markov chain. We set $P_R(t_0) = 1$ and extend to other nodes as

$$P_R(\overline{\theta}_{t\oplus x}) = \begin{cases} P_R(\overline{\theta}_t)\gamma(t \oplus a) & \text{if } x = a, \\ P_R(\overline{\theta}_t)(1 - \gamma(t \oplus a)) & \text{if } x = r. \end{cases} \tag{10}$$

We can now formulate an optimization problem for finding a node $t^*$ whose target density should be evaluated next, as soon as computational resources become available. This is simply the node $t \in O$ that has maximum probability of being required for the propagation of the Markov chain. The probability of a node being required, in turn, is given by the probability of its immediate ancestor being reached. So we define the mapping $\nu : T \longrightarrow T$ from a node to its parent, and obtain the next candidate for target density evaluation according to

$$t^* = \underset{t \in O}{\operatorname{argmax}}\, P_R(\nu(t)). \tag{11}$$

Under this choice of optimality condition, the probabilities of child nodes are necessarily lower than those of their parents. Consequentially, the optimal node can always be found in the subtree of layers $l \leq L^*$, where $L^*$ is the first layer for which no target densities have been computed yet.

*Active subtree.* While considering an infinite tree on a theoretical level, we operate only on an active subtree $T_A \subset T$ in practice, adding and pruning nodes as the algorithm progresses. New node pairs $(t \oplus a, t \oplus r)$ are only added to the active tree if their parent meets one of the following conditions:

(a) $t$ is the root
(b) $t$ is an accept node, meaning that the last letter in its string is an "a", and $t \in D \cup W$
(c) $t$ is a reject node, meaning that the last letter in its string is an "r", and its accept sibling $t_s \in D \cup W$

This ensures that along any possible path, exactly one advance accept/reject decision that still needs target density evaluation is in the active subtree. As soon as computational resources are available, evaluation is started for the current $t^* \in O$. As soon as results of computations are returned and resources freed, available MCMC decisions are performed and subtrees pruned, including within-subchain pruning. Importantly, as soon as there is a unique path between the root node and a fine level child $t_N$, that child's state is a new sample of the Markov chain. We then discard the entire tree before that sample and make it the new root, i.e. $t_0 \longleftarrow t_N$.

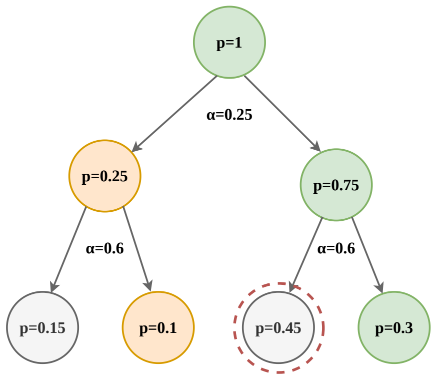


**Figure 3: Traversal of node probabilities through an exemplary Markov tree. Green color indicates finished posterior evaluation, orange indicates computations in progress. The most likely candidate, selected for the next posterior evaluation, is encircled in red.**

Overall, the presented MLDA algorithm iteratively performs a sequence of steps, until the desired number of fine-level samples is reached. These steps can be summarized as follows:

(1) Expand the Markov tree, request new posterior evaluations,
(2) update the Markov tree with finished jobs,
(3) compute available MCMC decisions,
(4) prune the Markov tree, and
(5) if possible, propagate the Markov chain.

## 3 SCALING MLDA FOR HPC SIMULATORS VIA UM-BRIDGE

Prefetching MLDA applies to a very wide range of inverse problems, since it only requires access to pointwise evaluations of the posterior (and no gradients of the forward model etc.). In order to make our software equally universal, we perform model evaluation calls through UM-Bridge. UM-Bridge is an easy-to-use framework for interacting with simulation codes regardless of programming language or platform, as demonstrated in [46].

It allows our Python implementation of prefetching MLDA to link to any simulation model from simple test problems to highly optimized simulations running on massively parallel HPC architectures. We illustrate the resulting setup in fig. 4.

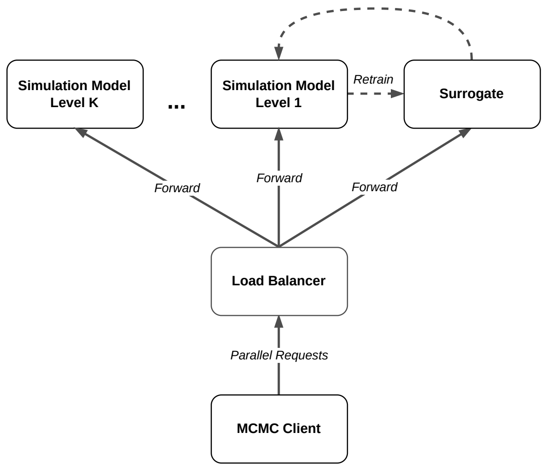


**Figure 4: Computational setup consisting of UQ client, cluster-side load balancer, adaptive surrogate model and parallel simulation instances.**

The prefetching MLDA Python code constitutes the client side in our setup, requesting model evaluations from the cluster via UM-Bridge. The load balancer, typically running on the login node of an HPC cluster, launches simulation model instances as needed and forwards evaluation requests to them.

In total, we employ three levels of parallelism:

- We run multiple independent MLDA samplers and collect their results (which is valid since they sample the same posterior), making use of `multiprocessing`.
- Each sampler performs asynchronous prefetching (section 2) for in-chain parallelism.
- The simulation model itself may be parallelized.

Importantly, since the generic load balancer handles simulation runs on the cluster, Python's `multiprocessing` framework is enough for prefetching MLDA to control parallel simulation runs across thousands of cores. Model parallelism is entirely transparent to the UQ side.

In our geophysics application (section 5), we employ a MLDA model hierarchy that includes a GP surrogate [42] in addition to numerical simulators. The GP approximates the simulation model on the coarsest simulation level at extremely low cost.

We ensure GP accuracy while restricting training to high-posterior areas through an adaptive procedure: We initially pretrain the surrogate from simulation runs on only a small number of Latin Hypercube points. During the UQ run, we employ the mean prediction as an approximation to the simulator. However, if GP variance exceeds a threshold (indicating high approximation error), we trigger an additional coarse-level simulation for that parameter, updating the GP for higher confidence in that area.

## 4 APPLICATION: BAYESIAN INFERENCE IN SEISMOLOGY

Demonstrating prefetching MLDA in a real-world application, we target a highly relevant inverse problem from computational seismology: Inferring information about earthquake sources from surface recordings only.

### 4.1 Modeling Earthquakes in SeisSol

The forward problem is solved by the earthquake simulation software SeisSol (www.seissol.org). The Earth is modeled as a 3D elastic body and the respective elastic wave equation is expressed in first-order formulation,

$$\partial_t u + A\partial_x u + B\partial_y u + C\partial_z u = 0, \tag{12}$$

where $u$ is the vector of unknowns (stress and particle velocities). The flux matrices $A$, $B$ and $C$ contain the material parameters (Lamé parameters and density). This hyperbolic partial differential equation is solved using the Discontinuous Galerkin method with Arbitrary high-order DERivatives time stepping (ADER-DG) [14].

SeisSol implements ADER-DG for elastic [14], viscoelastic [15, 51], anisotropic [10, 55] and poroelastic [9, 56] material models. In addition, it features plastic deformation [57] and the coupling of acoustic and elastic domains [30]. A key component of SeisSol are dynamic rupture earthquake sources [38, 39]. For this source type, the non-linear frictional failure along prescribed fault planes is simulated along with the seismic wave fields. The movement of the sliding fault induces wave motion in the surrounding bulk volume. Interaction of frictional failure and wave motion allows researchers to investigate earthquake physics. For example, dynamic models can explain how rupture *jumps* from one fault segment to another one [17, 50].

SeisSol has been used on several petascale supercomputers to model earthquake scenarios with several billion degrees of freedom, achieving a significant fraction of the theoretical peak performance [24, 30, 53]. On the node level, SeisSol relies on the code generator YATeTo [52], which generates high-performance code by mapping kernel descriptions (in Einstein sum convention) to highly optimized backends for small matrix operations. Through different backends, SeisSol achieves performance portability between different compute architectures including GPUs [12]. SeisSol uses a hybrid parallelization approach: On node level, we use either OpenMP (for CPUs) or CUDA/SYCL/ROCm (for GPUs). Between nodes, the mesh is partitioned and communication between ghost and copy cells is done with MPI. Asynchronous communication then hides communication behind computation.

In the Bayesian inverse problem considered (see section 4.2), the results of the forward model depend on several volumetric material parameters, e.g. Lamé parameters $\lambda$ and $\mu$ or plastic cohesion $c$, encoded in the parameter vector $\theta$. When the UQ sampler requires a forward model evaluation for a parameter $\theta$, a small wrapper script prepares input files for SeisSol. Parallel execution of SeisSol is then triggered. SeisSol produces artificial seismograms, which we compare to real-world recordings in order to form the Bayesian posterior eq. (15).

The wrapper code acts as an UM-Bridge server, making SeisSol easily accessible to the UQ code. Integrating UQ and model in such a way retains full flexibility w.r.t. model complexity (e.g. scattering at material interfaces or at the free surface), while cleanly separating it from the complexity of the UQ workflow.

## 4.2 2019 Ridgecrest, CA, earthquake scenarios

Modern measuring techniques enhance the quality and quantity of data available for characterizing earthquake dynamic rupture processes and include strong-motion and broadband seismometers, high-rate Global Navigation Satellite System (GNSS) instruments and space geodetic datasets [e.g., 18, 23].

However, while non-linear inversions of earthquake data for dynamic parameters to construct physically consistent earthquake models have been conducted [16, 19, 40, 44], the immense computational cost of each forward model restricted these to simplified model setups and inversions for on-fault parameters only. MLDA addresses the above limitations, drastically reducing the required number of full-complexity, high-resolution forward simulations.

As a demonstrator scenario, we examine linked dynamic rupture simulations of the 2019 $M_W$6.4 Searles Valley foreshock and the $M_W$7.1 Ridgecrest mainshock [50]. This earthquake sequence involves the rupture of a complex fault system comprising four major non-planar segments. To capture off-fault rock deformation, we embed the rupturing faults in elasto-plastic materials using a Drucker-Prager yield criterion [57]. The yielding strength $\tau_c$ is defined by two spatially varying material parameters, plastic cohesion $c$ and friction angle $\phi$, as

$$\tau_c = c\cos(\phi) - \sigma_m \sin(\phi), \tag{13}$$

where $\sigma_m = \sigma_{kk}/3$ is the mean of the stress tensor trace $\sigma_{ij}$. These plastic parameters strongly influence earthquake dynamic rupture processes and ground shaking. Following [50], we define spatially varying rock elastic moduli and prestress fields by combining a 3D community velocity model CVM-S4.26 [32] and a 2D community stress model representing the regional state of stress in the Southern California upper crust [58]. We also assume spatially varying plastic cohesion $c_0(x, y, z)$ as proportional to the shear modulus of the 3D velocity model. The plastic cohesion is $c = \gamma c_0(x, y, z)$, where $\gamma$ is a scaling factor to be inverted for using MLDA. The friction angle $\phi$ is held constant across the domain, for simplicity.

Under the above assumptions, the parameter vector is $\theta = (\gamma, \phi)^T \in \mathbb{R}^2$. Observational data for the inverse problem consists of three-dimensional displacement time series recorded at 10 GNSS stations near the fault system [35]. Each station provides surface displacement vectors $\delta_i(t)$, with $i$ representing east-west, north-south or up-down direction, at a sampling rate of 1 Hz. In total, we obtain observations $d^{\text{obs}} \in \mathbb{R}^q$, where $q = 3\times 10\times n_t$, and $n_t$ is the number of discrete points in the time series of each displacement component.

We aim to approximate the posterior distribution $\pi(\theta|d^{\text{obs}})$. We employ a uniform prior for all parameters, with the componentwise bounds defining a feasible rectangle $\Omega_{\text{prior}}$ within the parameter space. Furthermore, we assume that our data corresponds to the output of our forward model $G(\theta)$, perturbed by zero-centered Gaussian noise:

$$d^{\text{obs}} = G(\theta) + \eta, \quad \eta \sim \mathcal{N}(0, C),\ C = \sigma_d^2 I, \tag{14}$$

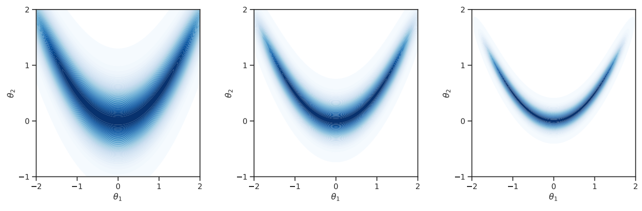


**Figure 5: Hierarchy of "banana"-shaped posterior densities.**

where $\sigma_d^2$ is a scalar constant and $I \in \mathbb{R}^{q\times q}$ denotes the identity matrix. In total, we define the Bayesian posterior

$$\pi(\theta|d^{\text{obs}}) = \begin{cases} \frac{1}{Z}\exp\Big(-\frac{1}{2}||d^{\text{obs}} - G(\theta)||^2_{C^{-1}}\Big), & \text{if } \theta \in \Omega_{\text{prior}}, \\ 0, & \text{else}, \end{cases} \tag{15}$$

where we have used the matrix-weighted norm $||\cdot||^2_{C^{-1}} = \langle\cdot, C^{-1}\cdot\rangle$. The normalization constant $Z$ only depends on the data and the chosen prior intervals.

## 5 RESULTS

### 5.1 Artificial Test Case

We first present results for prefetching MLDA based on an artificial hierarchy of 2D posterior densities. We define "banana"-shaped densities, which can be obtained from a simple transformation of a Gaussian density [28]. Based on a precision coefficient $c_i$, we write for the parameters $\theta_1, \theta_2 \in \mathbb{R}$ of the $i$-th member of the density hierarchy

$$\pi_i(\theta_1, \theta_2) \propto \exp\Big[-\tfrac{1}{2}c_i\big(20(\theta_1^2 - 2\theta_2)^2 + 2(\theta_2 - \tfrac{1}{4})^4\big)^2\Big]. \tag{16}$$

We construct a hierarchy of models with precision parameters $c_i = \{0.1, 0.3, 1.0\}$, which is depicted in fig. 5. Since evaluation of these densities is practically instantaneous, we introduce artificial sleep times $t_i = \{0.001s, 3s, 10s\}$ to emulate the workload of more realistic posterior evaluations. The compute time on the coarsest level is still negligible, mimicking a fast surrogate model.

Before evaluating the parallel efficiency of the prefetching algorithm, we demonstrate the potential usefulness of MLDA compared to vanilla Metropolis-Hastings (MH) MCMC [36], based on our test hierarchy. To this end, we directly draw samples from the fine-level distribution with the MH-MCMC algorithm. We utilize a simple random walk proposal, $q(\cdot|\theta) = \mathcal{N}(\theta, I)$. Furthermore, we generate an equal number of samples with the MLDA algorithm, using 30 samples for level-one subchains and three samples for level two. On level one, we employ the same proposal as for the MH-MCMC algorithm. For both approaches, we generate four chains consisting of 1000 samples each.

Verifying that our MLDA implementation delivers the expected statistical efficiency, we compare the autocorrelation functions (ACF), effective samples sizes (ESS) and rank-normalized $\hat{R}$ statistics [54] of the two obtained sample sets, exemplarily for $\theta_1$ (fig. 6). For the MH-MCMC algorithm, samples are highly correlated, up to a lag of $\sim$ 40 samples. The autocorrelation is reduced by a factor of more than 10 for the samples obtained with MLDA. Similarly, we observe that the ESS is about a factor 10 higher for

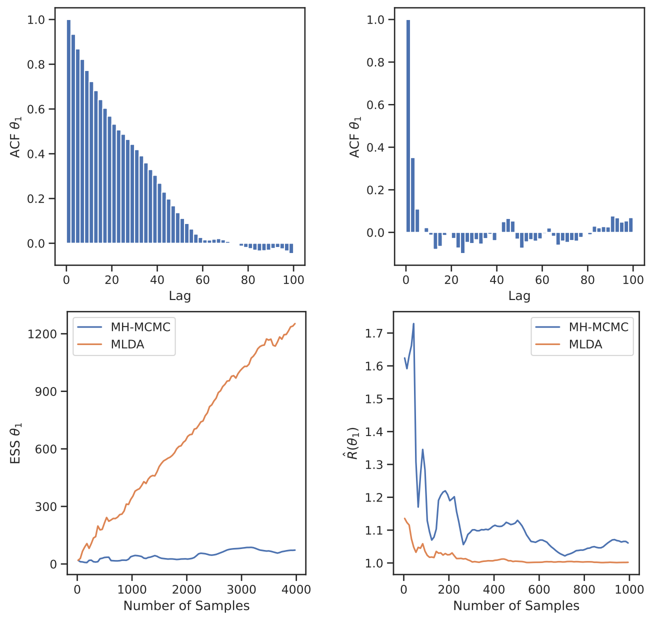


**Figure 6: ACFs for MH-MCMC (top-left) and MLDA samples (top-right). ESS (bottom-left) and $\hat{R}$ (bottom-right) comparison for both algorithms.**

the MLDA samples. Lastly, the $\hat{R}$ converges to one much faster for MLDA, indicating very short burn-in times. When implemented efficiently, this means that the utilization of a hierarchy of models can significantly reduce burn-in times and increase the number of effective samples for statistical estimation.

*Parallel speed-up.* Moving on, we discuss results regarding the core contribution of this work, parallelization through prefetching. We conduct MLDA runs as described above, with parallelization through prefetching for a varying number of up to 10 parallel threads. To assess parallel efficiency, we inspect the overall speed-up in run times, which is simply given as the ratio of the execution time without pre-fetching, $t_1$, and that for $N_{\text{th}}$ parallel threads, $t_{N_{\text{th}}}$,

$$S_{\text{th}}(N_{\text{th}}) = t_1/t_{N_{\text{th}}}. \tag{17}$$

This quantity implicitly includes system latency. As the dummy model benchmarks have been run on a single CPU, we do not expect latency to be relevant. For large-scale computations, on the other hand, latencies in the prefetching algorithm are negligible compared to the run times of single forward model evaluations.

Figure 7 summarizes the speed-up results. In the artificial problem, parallelization through prefetching yields an overall speed-up of about 2.5–3 in the tested range of threads. However, the speed-up reaches a plateau rather quickly, so that we anticipate a single-digit number of threads to be sensible in many cases.

Lastly, we point out that the current implementation of parallelized MLDA leaves some room for improvement. Specifically, ongoing simulation runs are not aborted when they become obsolete through an MCMC decision. Moreover, we employ a rather simplistic acceptance predictor for future states, which does not

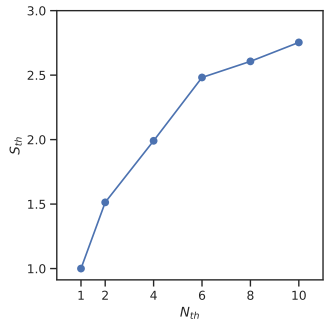


**Figure 7: Runtime speed-ups for prefetching-based parallelization of MLDA with the banana posterior hierarchy.**

take into account the random draws for the realization of the actual Markov Chain.

*Optimal parallelization strategy.* To assess the usefulness of prefetching, we need to compare it to the obvious competing parallelization approach for MCMC, namely running parallel chains. Generating parallel chains is an embarrassingly parallel task, but also exhibits diminishing returns: Burn-in has to be performed for every single chain, implying that an increasing amount of samples has to be discarded when more parallel chains are run. To formalize this, suppose we intend to generate $M_{\text{eff}}$ usable samples of a posterior distribution (not to be confused with the ESS presented above). From the total number of samples generated, we have to deduct $M_{\text{burn}}$ burn-in samples for every chain. Let $\nu = \frac{M_{\text{eff}}}{M_{\text{burn}}}$ further be the ratio of effective samples to burn-in samples. Given such a fixed ratio $\nu$, the speed-up through $N_{\text{ch}}$ parallel chains is

$$S_{\text{ch}}(N_{\text{ch}}) = \frac{N_{\text{ch}}}{1 + \frac{N_{\text{ch}}-1}{\nu+1}}. \tag{18}$$

Regarding the sampling procedure from the banana hierarchy, we choose for our assessment $M_{\text{burn}} = 7$ and $M_{\text{eff}} = \{50, 100, 200, 400\}$. We point out here that we do not rely on a quantitative assessment of the $\hat{R}$ statistics for the determination of $M_{\text{burn}}$. The statistic is not reliable quantitatively in the low sample and low chain regime [1]. We rather make the optimistic estimate that the chains reach stationarity after initial decorrelation, which is supported by the behavior of the traces themselves.

The resulting speed-ups for up to 32 parallel chains are presented in fig. 8 (left). Clearly, parallelization performance deteriorates when burn-in becomes significant compared to the overall number of samples that are computed per chain. This is also apparent from equation (18). For $\nu \gg N_{\text{ch}}$, we can expect (nearly) perfect speed-up. As we approach $\nu \sim \mathcal{O}(1)$, however, we can observe that also $S_{\text{ch}} \sim \mathcal{O}(1)$, regardless of the number of chains employed.

For the case $M_{\text{eff}} = 50$, we compare the performance gain of parallel chains with that of prefetching. Specifically, we consider two computations with different amounts of resources $N_{i,1}$, $N_{i,2}$, $i \in \{\text{th}, \text{ch}\}$, and the respective difference $\Delta N_i = N_{i,1} - N_{i,2}$. The unit of the resources is either the number of parallel chains or the number of threads for prefetching. Note that these two quantities are comparable, as they both denote multiples of the computational resources required for a posterior evaluation. For both simulations,

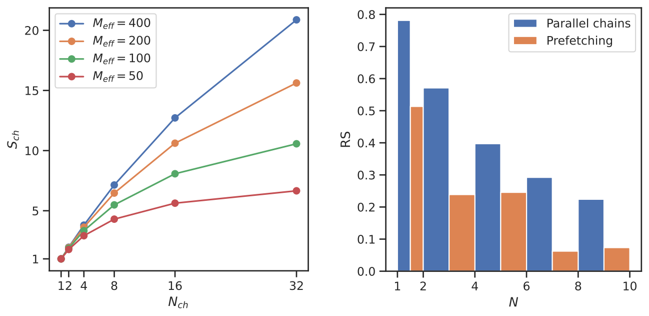
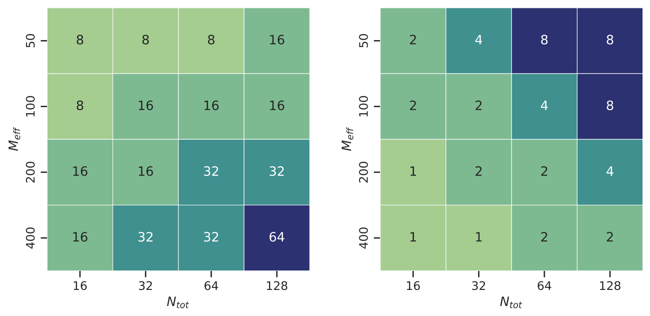


**Figure 8: Left: Speed-up through employment of parallel chains, for different values of $M_{\text{eff}}$ and $M_{\text{burn}} = 7$. Right: Speed-up rates of parallel chains vs. prefetching, for $M_{\text{eff}} = 50$.**



Left (optimal number of parallel chains):

| $M_{eff}$ \ $N_{tot}$ | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 50 | 8 | 8 | 8 | 16 |
| 100 | 8 | 16 | 16 | 16 |
| 200 | 16 | 16 | 32 | 32 |
| 400 | 16 | 32 | 32 | 64 |

Right (optimal number of threads for prefetching):

| $M_{eff}$ \ $N_{tot}$ | 16 | 32 | 64 | 128 |
|---|---|---|---|---|
| 50 | 2 | 4 | 8 | 8 |
| 100 | 2 | 2 | 4 | 8 |
| 200 | 1 | 2 | 2 | 4 |
| 400 | 1 | 1 | 2 | 2 |

**Figure 9: Optimal distribution of resources for different combinations of $M_{\text{eff}}$ and $N_{\text{tot}}$. Left: Optimal number of parallel chains. Right: Optimal number of threads for prefetching.**

we compare speed-ups $S_{i,1/2}$ as $\Delta S_i = S_{i,1} - S_{i,2}$. Finally, we define the relative speed-up as $\text{RS}_i = \Delta S_i / \Delta N_i$. The corresponding results are depicted in fig. 8 (right). While speed-up rates for parallel chains are generally higher than those for prefetching, both exhibit diminishing returns. As a result, we have to find an optimal, problem-dependent balance between the two.

To quantify the interplay between $N_{\text{ch}}$ and $N_{\text{th}}$, we consider the problem of optimal resource allocation for our artificial problem, under constraints that resemble realistic scenarios. Assume that our posterior evaluation requires one generic unit of computational resources. This might correspond to use-cases where posterior evaluations are obtained from software that is not parallelized or whose parallelization does not scale well. We further consider the scenario that these evaluations are expensive, e.g. for large simulations of physical systems. Thus, we anticipate to obtain only relatively small numbers of samples, and again choose $M_{\text{eff}} = \{50, 100, 200, 400\}$. Further suppose that these samples have to be obtained under severe time constraints, while a large number of resources $N_{\text{tot}}$ is available. This might indeed be a typical case for rapid prototyping and risk assessment in R&D projects. For our exposition, we choose $N_{\text{tot}} = \{16, 32, 64, 128\}$. Under these constraints, we intend to find the optimal combination of resources that minimizes evaluation time for the desired sample numbers. Formally, we compute

$$N^*_{\text{ch}}, N^*_{\text{th}} = \underset{N_{\text{ch}}, N_{\text{th}} \in \mathbb{N}_+}{\operatorname{argmax}} \; S_{\text{ch}}(N_{\text{ch}}) S_{\text{th}}(N_{\text{th}}), \text{ and } N_{\text{ch}} N_{\text{th}} = N_{\text{tot}}. \tag{19}$$

This is a simple combinatorial optimization problem, whose results are depicted in fig. 9 for the chosen combinations of $M_{\text{eff}}$ and $N_{\text{tot}}$.

The illustrations confirm our qualitative assessment on the interplay of $N_{\text{ch}}$ and $N_{\text{th}}$. For relatively large sample numbers $M_{\text{eff}}$, parallel chains are more efficient, although prefetching with a low number of threads might still yield some performance gain. On the other hand, prefetching is particularly effective for low $M_{\text{eff}}$, when burn-in becomes significant more quickly. In this case, up to eight threads can be employed.

To conclude our discussion, we comment on the projected speed-ups $S_{\text{tot}} = S_{\text{ch}} S_{\text{th}}$ for different combinations of resource distribution. Even if prefetching appears to be a favorable choice according to the solution of the above optimization problem, the simpler approach of parallel chains might still be a better option if performance gains are only marginal. Therefore, we evaluate $S_{\text{tot}}$ for the case $M_{\text{eff}} = 50$

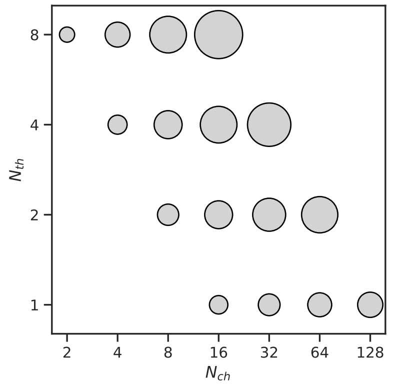


**Figure 10: Projected Speed-up for different combinations of $N_{\text{ch}}$ and $N_{\text{th}}$, for+ $M_{\text{eff}} = 50$. Sizes of the bubbles indicate the speed-up value.**

and varying $N_{\text{tot}}$, for all possible resource combinations $N_{\text{ch}}, N_{\text{th}}$. These speed-ups are visualized in the bubble plot 10.

The results indicate that, for given $M_{\text{eff}}$, the performance gain through prefetching is highly dependent on the total amount of available resources $M_{\text{tot}}$. If only few resources are available, only few parallel chains can be spawned, whose parallel efficiency is favorable compared to prefetching. In addition, we note that the projected speed-up in this scenario is relatively independent of the chosen resource combination. This is quite different for large $N_{\text{tot}}$. We observe that for $N_{\text{tot}} = 128$, prefetching with up to eight threads is clearly preferable to assigning all resources to parallel chains. Indeed, the potential performance gain through prefetching is almost 100%.

We conclude that prefetching is a viable approach for strong-scaling scenarios, i.e. when high computational resources are employed to compute a limited number of samples in a short time.

## 5.2 Large-Scale Seismology Application

As a real-world use-case, we apply our parallelized MLDA algorithm to the Bayesian inverse problem described in section 4.2, for the parameters $\theta = (\gamma, \phi) \in \mathbb{R}^2$. The approach employs a hierarchy of three models with increasing complexity. At the coarsest level, we

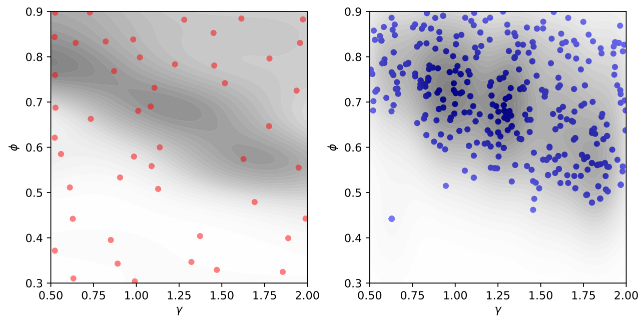


**Figure 11: Left: Surrogate model estimated probability density of the plastic cohesion $\gamma$ and the internal friction angle $\phi$. Red dots represent training points; Right: Accepted samples and density estimates from the finest MLDA hierarchy level.**

utilize a surrogate model based on GP regression, trained with data (both pretraining and online) obtained from the next higher level in the model hierarchy. This second, as well as the third and finest level, use full SeisSol simulations on computational meshes with ≈4 million tetrahedral elements. The polynomial order of the basis functions is set to three on the second level, and four on the third level, resulting in roughly double the runtime for the latter. In most cases, the modeled ground motions of the second level differ by less than 5% from those of the third level in the matrix-weighted norm defined in equation (15). However, the additional accuracy gained by using higher polynomial order is crucial to resolve whether the mainshock can be triggered by the foreshock in the 2019 Ridgecrest earthquake sequences [50]. Subchain lengths are set to 30 and two for level one and two, respectively. All following results have been obtained using the Frontera supercomputer [48]. The machine employs two Intel Xeon Platinum 8280 (Cascade Lake) processors per node, together offering 56 cores and operating at 2.7 GHz. The wall times for each evaluation on the 1st, 2nd, and 3rd levels are, respectively, <1 s, ∼10 CPU hours and ∼20 CPU hours. The total number of available compute nodes is 8368.

The results from the inversion procedure are depicted in fig. 11 (bottom). These include 440 fine-level samples, obtained from two independent chains, together with the density estimate for the two-dimensional posterior. To get these 440 accepted models in MLDA chains, the total numbers of evaluations from the 1st, 2nd, and 3rd levels are, respectively, 16135, 1112 and 525. Our main finding is that the plastic cohesion coefficient $\gamma$ has a near-uniform probability distribution in its parameter space, while the friction coefficient $\phi$ shows a clear tendency towards higher values. We also observe a slight negative correlation between the two parameters. The GP surrogate posterior density approximation (fig. 11, top) is in good agreement with the fine-level samples, indicating that the surrogate and coarser SeisSol simulations provide a reasonable approximation of the posterior.

We mention at this point that the employed model hierarchy has been chosen carefully, but nevertheless to a certain extent ad hoc. MLDA relies on the "similarity" or "overlap" between the distributions on different levels. This is impossible to quantify in advance for complex simulation models. However, MLDA gives statistical guarantees for fine-level samples, as long as the fine-level model resembles the desired, "true" posterior. Therefore, our choice of model hierarchy, while not guaranteed to be optimal, can be justified a-posteriori through the good decorrelation and mixing behavior of the resulting Markov chains.

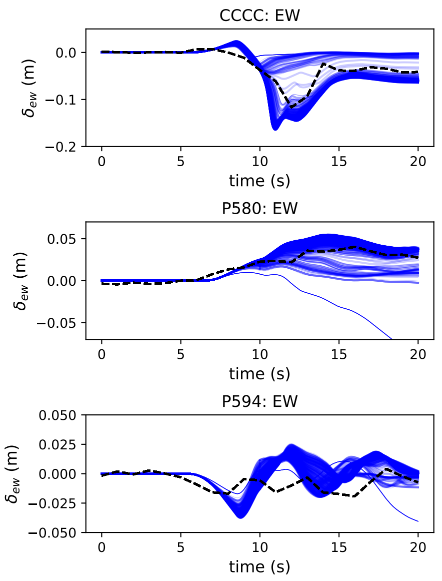


**Figure 12: Comparison between simulations (blue solid curves) and the observed displacement data $\delta$ in the east-west (EW) direction (black dashed curves) from three (CCCC, P580, P594) of the ten GNSS stations.**

We then assess the posterior predictive performance of the inferred parameter set by running fine-level forward simulations for all MCMC samples. We compare the resulting displacement in the east-west direction to the observational data at three GNSS stations, as shown in fig. 12. We observe that the posterior predictive ensemble reproduces the data reasonably well at the first two GNSS stations, with the prediction intervals encompassing the observations. However, the third station (P594) shows a slight discrepancy between the predictive ensemble and the data. We attribute this offset to the simplistic parametrization in our inverse problem, which cannot fully capture all complexities in the high-dimensional data. Nevertheless, the inferred parametrization provides reasonable predictions, including confidence intervals, for the Ridgecrest model.

After discussing the inference results, we now evaluate the performance of our MLDA sampler in the context of the geophysics application. As for the artificial setup, we present the ESS, ACF, and $\hat{R}$ statistic of the obtained samples in fig. 13. The results indicate that employing the MLDA algorithm yields similar improvement as for our previous experiment.

*Optimal parallelization strategy.* Lastly, we assess prefetching for the inversion, compared to other means of parallelization. The procedure is analogous to the analysis of the artificial model hierarchy in section 5.1. We introduce, however, an additional degree of freedom, namely the speed-up $S_{\text{sim}}(N_{\text{sim}})$ of single model evaluations

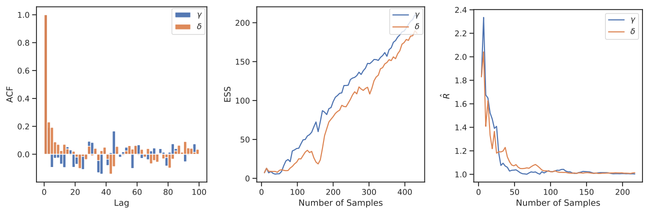


Figure 13: ACF, ESS, and $\hat{R}$ for MLDA run on the Ridgecrest model.

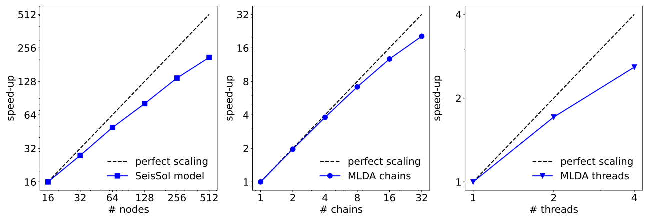


Figure 14: Strong scaling for the Ridgecrest model in SeisSol (left), increasing number of MLDA chains (middle), and increasing number of threads used within an MLDA chain (right).

when parallelized on $N_{\text{sim}}$ processing units:

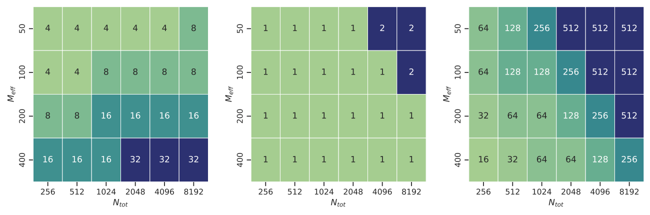
$$S_{\text{sim}}(N_{\text{sim}}) = t_1 / t_{N_{\text{sim}}}. \tag{20}$$

In this concrete setting, processing units correspond to compute nodes. Importantly, $S_{\text{sim}}$ is a gross quantity incorporating runtime speed-ups across all levels, weighted by the number of evaluations on each level. Thus, it is specific to the MLDA run configuration. It further implies that we utilize the same number of nodes for the parallelization on levels two and three (surrogate model evaluations are run in serial). The speed-ups for the conducted MLDA run, along with the speed-ups for parallel chains and prefetching threads, are depicted in fig. 14.

We can now formulate an optimization problem for the allocation of computational resources, similar to equation (19),

$$N_{\text{sim}}^*, N_{\text{ch}}^*, N_{\text{th}}^* = \underset{N_{\text{sim}}, N_{\text{ch}}, N_{\text{th}} \in \mathbb{N}_+}{\operatorname{argmax}} S_{\text{sim}}(N_{\text{sim}}) S_{\text{ch}}(N_{\text{ch}}) S_{\text{th}}(N_{\text{th}}),$$
$$\text{and } N_{\text{sim}} N_{\text{ch}} N_{\text{th}} = N_{\text{tot}}. \tag{21}$$

We investigate the distribution of resources for $M_{\text{burn}} = 7$, $M_{\text{eff}} = \{50, 100, 200, 400\}$, and $N_{\text{tot}} = \{256, 512, 1024, 2048, 4096, 8192\}$. The maximum amount of available resources corresponds to almost the entire Frontera compute cluster. The results in fig. 15 show that, in contrast to the artificial problem above, prefetching is not as advantageous here. Only for low sample numbers and high computational resources, activating prefetching is optimal. We attribute this to two aspects: (1) Although SeisSol is not run in its optimal resource regime, it scales very well onto significant portions of large HPC clusters, leaving little room for parallelization on the UQ side. In particular, prefetching is favorable if the forward model is not well parallelizable, and a large portion of the resources dedicated to a model evaluation can be assigned to either prefetching threads or parallel chains. This is clearly not the case here. (2) The GP surrogate proves extremely effective, leading to highly uncorrelated MLDA samples. This implies that only a very short burn-in phase is



| $M_{eff}$ \ $N_{tot}$ | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|
| 50 | 4 | 4 | 4 | 4 | 4 | 8 |
| 100 | 4 | 4 | 8 | 8 | 8 | 8 |
| 200 | 8 | 8 | 16 | 16 | 16 | 16 |
| 400 | 16 | 16 | 16 | 32 | 32 | 32 |

| $M_{eff}$ \ $N_{tot}$ | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|
| 50 | 1 | 1 | 1 | 1 | 2 | 2 |
| 100 | 1 | 1 | 1 | 1 | 1 | 2 |
| 200 | 1 | 1 | 1 | 1 | 1 | 1 |
| 400 | 1 | 1 | 1 | 1 | 1 | 1 |

| $M_{eff}$ \ $N_{tot}$ | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|
| 50 | 64 | 128 | 256 | 512 | 512 | 512 |
| 100 | 64 | 128 | 128 | 256 | 512 | 512 |
| 200 | 32 | 64 | 64 | 128 | 256 | 512 |
| 400 | 16 | 32 | 64 | 64 | 128 | 256 |

Figure 15: Optimal distribution of resources for different combinations of $M_{\text{eff}}$ and $N_{\text{tot}}$. Left: Optimal number of parallel chains. Middle: Optimal number of threads for prefetching. Right: Optimal number of nodes for running a single SeisSol simulation.

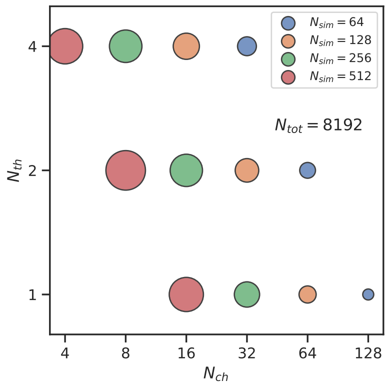


Figure 16: Projected Speed-up for different combinations of $N_{\text{ch}}$, $N_{\text{th}}$ and $N_{\text{sim}}$, for $M_{\text{eff}} = 50$, and $N_{\text{tot}} = 8192$. Sizes of the bubble indicate the speed-up value, colors indicate different values of $N_{\text{sim}}$.

necessary, making parallel chains unusually favorable compared to prefetching. We conclude that, although the application has greatly benefited from MLDA and our computational pipeline, it is not an ideal use-case for prefetching.

This is also confirmed by our assessment of the projected speed-ups for $M_{\text{eff}} = 50$ and $N_{\text{tot}} = 8192$, visualized in fig. 16. Although this is a scenario that favors prefetching (small sample size, large amount of resources), we again observe that heavy parallelization of SeiSol is favored. Distribution of additional resources onto prefetching threads does not yield advantages over parallel chains. In this scenario, parallel chains should be employed, as their implementation is unarguably simpler.

## 6 CONCLUSIONS AND PERSPECTIVES

In this work, we have presented a novel parallelization strategy for the MLDA algorithm via fully asynchronous prefetching. We have introduced the necessary theoretical foundations for prefetching in a multilevel setting. In addition, we have presented a modular computational pipeline that makes our workflow suitable for large-scale applications. We have further introduced the utilization of GP surrogates into the computational model hierarchy.

To assess the viability of the method, we have compared prefetching to the parallel simulation of multiple chains for an example problem. Our findings indicate that prefetching can be a valuable resource for scenarios where the strong scaling behavior of the sampling procedure is relevant. This is typically the case for expensive computational models, large computational resources, but limited time-to-result.

Beyond the example use-case, we have applied our parallelized prefetching algorithm to a large-scale problem in geoscience. Although the parametrization considered is quite simplistic, it demonstrates the viability for Bayesian inference involving large-scale simulations.

While prefetching does not seem to yield significant performance improvements in this particular case, we anticipate it to be valuable in scenarios where the forward model is not as well parallelized as SeisSol. It therefore is a promising addition for the solution of statistical inverse problems in realistic scenarios. Our generic computational pipeline facilitates this transfer, allowing for incorporation of a wide range of computational models.

We point out here that alternative methods of parallel MCMC methods are available, such as multi-proposal MH [6], population-based MCMC [27], and bias removal through couplings [26]. The key difference is that prefetching in itself is not a parallel algorithm, but a technical approach to parallelizing an otherwise sequential algorithm like MLDA. In general, prefetching is applicable for other sequential (portions of) MCMC algorithms, but we deem it particularly useful in the multilevel context, where computation times vary significantly across the model hierarchy. MLDA has some potential overlap with other MCMC algorithms. For instance, tempering for the generation of a model sequence in population-based MCMC can also be applied for the construction of a model hierarchy in MLDA. Assessing which algorithm works "better" is highly problem-specific, and beyond the scope of this work.

The performance assessment of parallelized MLDA in this work has solely focussed on the optimal speed-up of execution, given a fixed amount of resources. We have deliberately foregone the investigation of computational cost, e.g. the optimization of CPU hours. Prefetching always entails a "waste" of resources, meaning that it should not be employed when overall computational cost is the significant metric. An interesting point for future work is the interplay of speed-up and cost, which could establish a sort of Pareto optimality, as show-cased in [29].

We further intend to explore the employed MLDA model hierarchy more systematically for SeisSol applications. While such an assessment can only be done numerically, different model configurations, an alternative number of levels, and/or different sub-chain lengths might yield a further improvement in sampling efficiency. Improvements could further be obtained through technical adjustments in the model hierarchy. Coarser level simulations might be sufficiently accurate with single-precision floating point arithmetics. Heterogenous distribution of computational resources among the levels could also speed up overall runtimes. Lastly, an additional benefit could be obtained through adaptive error modeling, as proposed in [34]. This approach improves similarity among levels in a quantifiable metric during the sampling procedure.

In addition, we intend to apply the presented methods to a higher-dimensional parametrization of the geophysics model, as the current approach appears to be too simplistic to resolve all features in the observational data. MLDA and MCMC methods have some inherent limitations regarding scalability in higher dimensions, but they are independent of our computational setup and parallelization strategy. In practice, this amounts to a higher number of posterior evaluations for an efficient exploration of the parameter space.

## ACKNOWLEDGMENTS

This work has been funded by the Innovation Study ScalaMIDA, which has received funding through the Inno4scale project, which is funded by the European High Performance Computing Joint Undertaking (JU) under Grant Agreement No 101118139. The JU receives support from the European Union's Horizon Europe Programme.

M.K. acknowledges additional support by the state of Baden-Württemberg through bwHPC. Z.N. and A.-A.G. acknowledge additional support from the National Science Foundation and TACC's LCCF-CSA (grant numbers OAC-2139536, OAC-2311208, EAR - 2225286, EAR-2121568), from the European Union's Horizon 2020 research and innovation programme (Marie-Skłodowska-Curie grant agreement number 955515, SPIN ITN), from Horizon Europe (ChEESE-2P, grant number 101093038, DT-GEO, grant number 101058129, Geo-INQUIRE, grant number 101058518), and the National Aeronautics and Space Administration (80NSSC20K0495). Additional computing resources were provided by the Institute of Geophysics of LMU Munich [37].

We thank Robert Scheichl (Heidelberg University, Germany) for fruitful discussions about prefetching and MLDA, Sebastian Krumscheid (KIT, Germany) for his insights on MCMC convergence and diagnostics, and Ian Wang (TACC) for his tremendous HPC Frontera support.

## REFERENCES

[1] Steve Brooks (Ed.). 2011. *Handbook for Markov chain Monte Carlo* (Boca Raton). Taylor & Francis.

[2] Gregory Abram, Alice Gabriel, Francesca Samsel, Nico Schliwa, Yinzhi Wang, and Sean Cunningham. 2019. Conversing Faults: The 2019 Ridgecrest Earthquake. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC23)*. Texas Advanced Computing Center, University of Texas; University of California, San Diego; Ludwig-Maxmilians-Universität München. https://sc23.supercomputing.org/proceedings/sci_viz/sci_viz_pages/svs102.html

[3] Elaine Angelino, Eddie Kohler, Amos Waterland, Margo Seltzer, and Ryan P. Adams. 2014. Accelerating MCMC via parallel predictive prefetching. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence* (Quebec City, Quebec, Canada) *(UAI'14)*. AUAI Press, Arlington, Virginia, USA, 22–31.

[4] Simon Arridge, Peter Maass, Ozan Öktem, and Carola-Bibiane Schönlieb. 2019. Solving inverse problems using data-driven models. *Acta Numerica* 28 (2019), 1–174. https://doi.org/10.1017/S0962492919000059

[5] A. E Brockwell. 2006. Parallel Markov chain Monte Carlo Simulation by Pre-Fetching. *Journal of Computational and Graphical Statistics* 15, 1 (2006), 246–261. https://doi.org/10.1198/106186006X100579

[6] Ben Calderhead. 2014. A general construction for parallelizing Metropolis-Hastings algorithms. *Proceedings of the National Academy of Sciences of the United States of America* 111, 49 (Dec. 2014), 17408–17413. https://doi.org/10.1073/pnas.1408184111

[7] J. Christen and Colin Fox. 2005. Markov chain Monte Carlo Using an Approximation. *Journal of Computational and Graphical Statistics* 14 (12 2005), 795–810. https://doi.org/10.1198/106186005X76983

[8] Tiangang Cui, Kody J.H. Law, and Youssef M. Marzouk. 2016. Dimension-independent likelihood-informed MCMC. *J. Comput. Phys.* 304 (2016), 109 – 137. https://doi.org/10.1016/j.jcp.2015.10.008

[9] Josep de la Puente, Michael Dumbser, Martin Käser, and Heiner Igel. 2008. Discontinuous Galerkin Methods for Wave Propagation in Poroelastic Media. *GEOPHYSICS* 73, 5 (Sept. 2008), T77–T97. https://doi.org/10.1190/1.2965027

[10] Josep de la Puente, Martin Käser, Michael Dumbser, and Heiner Igel. 2007. An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes - IV. Anisotropy. *Geophysical Journal International* 169, 3 (June 2007), 1210–1228. https://doi.org/10.1111/j.1365-246X.2007.03381.x

[11] Tim Dodwell, Chris Ketelsen, Robert Scheichl, and Aretha Teckentrup. 2019. Multilevel Markov Chain Monte Carlo. *SIAM Rev.* 61 (01 2019), 509–545. https://doi.org/10.1137/19M126966X

[12] Ravil Dorozhinskii and Michael Bader. 2021. SeisSol on Distributed Multi-GPU Systems: CUDA Code Generation for the Modal Discontinuous Galerkin Method. In *The International Conference on High Performance Computing in Asia-Pacific Region (HPC Asia 2021)*. Association for Computing Machinery, New York, NY, USA, 69–82. https://doi.org/10.1145/3432261.3436753

[13] Simon Duane, A.D. Kennedy, Brian J. Pendleton, and Duncan Roweth. 1987. Hybrid Monte Carlo. *Physics Letters B* 195, 2 (1987), 216 – 222. https://doi.org/10.1016/0370-2693(87)91197-X

[14] Michael Dumbser and Martin Käser. 2006. An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes - II. The Three-Dimensional Isotropic Case. *Geophysical Journal International* 167, 1 (Oct. 2006), 319–336. https://doi.org/10.1111/j.1365-246X.2006.03120.x

[15] Michael Dumbser, Martin Käser, and Eleuterio F. Toro. 2007. An Arbitrary High-Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes – V. Local Time Stepping and p-Adaptivity. *Geophysical Journal International* 171, 2 (2007), 695–717. https://doi.org/10.1111/j.1365-246X.2007.03427.x

[16] Eiichi Fukuyama and Takeshi Mikumo. 1993. Dynamic rupture analysis: Inversion for the source process of the 1990 Izu-Oshima, Japan, earthquake (M = 6.5). *Journal of Geophysical Research: Solid Earth* 98 (1993).

[17] Alice-Agnes Gabriel, Dmitry I. Garagash, Kadek H. Palgunadi, and P. Martin Mai. 2024. Fault size–dependent fracture energy explains multiscale seismicity and cascading earthquakes. *Science* 385 (2024).

[18] Alice-Agnes Gabriel, Thomas Ulrich, Mathilde Marchandon, James Biemiller, and John Rekoske. 2023. 3D Dynamic Rupture Modeling of the 6 February 2023, Kahramanmaraş, Turkey M w 7.8 and 7.7 Earthquake Doublet Using Early Observations. *The Seismic Record* 3, 4 (2023), 342–356.

[19] F. Gallovič, L. Valentová, J.-P. Ampuero, and A.-A. Gabriel. 2019. Bayesian Dynamic Finite-Fault Inversion: 2. Application to the 2016 Mw 6.2 Amatrice, Italy, Earthquake. *Journal of Geophysical Research: Solid Earth* 124 (2019).

[20] Omar Ghattas and Karen Willcox. 2021. Learning physics-based models from data: perspectives from inverse problems and model reduction. *Acta Numerica* 30 (2021), 445–554. https://doi.org/10.1017/S0962492921000064

[21] Michael B. Giles. 2008. Multilevel Monte Carlo Path Simulation. *Oper. Res.* 56, 3 (May 2008), 607–617. https://doi.org/10.1287/opre.1070.0496

[22] W. K Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57, 1 (1970), 97–109.

[23] J. N. Hayek, M. Marchandon, D. Li, L. Pousse-Beltran, J. Hollingsworth, T. Li, and A.-A. Gabriel. 2024. Non-Typical Supershear Rupture: Fault Heterogeneity and Segmentation Govern Unilateral Supershear and Cascading Multi-Fault Rupture in the 2021 7.4 Maduo Earthquake. *Geophysical Research Letters* 51, 20 (2024), e2024GL110128. https://doi.org/10.1029/2024GL110128 arXiv:https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2024GL110128 e2024GL110128 2024GL110128.

[24] A. Heinecke, A. Breuer, S. Rettenberger, M. Bader, A. Gabriel, C. Pelties, A. Bode, W. Barth, X. Liao, K. Vaidyanathan, M. Smelyanskiy, and P. Dubey. 2014. Petascale High Order Dynamic Rupture Earthquake Simulations on Heterogeneous Supercomputers. In *SC14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 3–14. https://doi.org/10.1109/SC.2014.6

[25] Matthew D. Homan and Andrew Gelman. 2014. The No-U-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *J. Mach. Learn. Res.* 15, 1 (Jan 2014), 1593–1623.

[26] Pierre E. Jacob, John O'Leary, and Yves F. Atchadé. 2020. Unbiased Markov Chain Monte Carlo Methods with Couplings. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 82, 3 (July 2020), 543–600. https://doi.org/10.1111/rssb.12336

[27] Ajay Jasra, David A. Stephens, and Christopher C. Holmes. 2007. On population-based simulation for static inference. *Statistics and Computing* 17, 3 (Aug. 2007), 263–279. https://doi.org/10.1007/s11222-007-9028-9

[28] Jari Kaipio and Erkki Somersalo. 2005. *Statistical and computational inverse problems.* Applied mathematical sciences, Vol. 160. Springer.

[29] Tzanio Kolev, Paul Fischer, Misun Min, Jack Dongarra, Jed Brown, Veselin Dobrev, Tim Warburton, Stanimire Tomov, Mark S Shephard, Ahmad Abdelfattah, Valeria Barra, Natalie Beams, Jean-Sylvain Camier, Noel Chalmers, Yohann Dudouit, Ali Karakus, Ian Karlin, Stefan Kerkemeier, Yu-Hsiang Lan, David Medina, Elia Merzari, Aleksandr Obabko, Will Pazner, Thilina Rathnayake, Cameron W Smith, Lukas Spies, Kasia Swirydowicz, Jeremy Thompson, Ananias Tomboulides, and Vladimir Tomov. 2021. Efficient exascale discretizations: High-order finite element methods. *The International Journal of High Performance Computing Applications* 35, 6 (Nov. 2021), 527–552. https://doi.org/10.1177/10943420211020803 Publisher: SAGE Publications Ltd STM.

[30] Lukas Krenz, Carsten Uphoff, Thomas Ulrich, Alice-Agnes Gabriel, Lauren S. Abrahams, Eric M. Dunham, and Michael Bader. 2021. 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 (SC '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3458817.3476173

[31] Maximilian Kruse, Zihua Niu, Sebastian Wolf, Mikkel Lykkegaard, Michael Bader, Alice-Agnes Gabriel, and Linus Seelinger. 2024. *Scalable Bayesian Inference of Large Simulations via Asynchronous Prefetching Multilevel Delayed Acceptance.* https://doi.org/10.5281/zenodo.14315615

[32] En-Jui Lee, Po Chen, Thomas H Jordan, Phillip B Maechling, Marine AM Denolle, and Gregory C Beroza. 2014. Full-3-D tomography for crustal structure in southern California based on the scattering-integral and the adjoint-wavefield methods. *Journal of Geophysical Research: Solid Earth* 119, 8 (2014), 6421–6451.

[33] Baoshan Liang, Jingye Tan, Luke Lozenski, David A. Hormuth, Thomas E. Yankeelov, Umberto Villa, and Danial Faghihi. 2023. Bayesian inference of tissue heterogeneity for individualized prediction of glioma growth. *IEEE Transactions on Medical Imaging* 42, 10 (2023), 2865–2875. https://doi.org/10.1109/TMI.2023.3267349 Publisher: IEEE.

[34] M. B. Lykkegaard, T. J. Dodwell, C. Fox, G. Mingas, and R. Scheichl. 2023. Multilevel Delayed Acceptance MCMC. *SIAM/ASA Journal on Uncertainty Quantification* 11, 1 (2023), 1–30. https://doi.org/10.1137/22M1476770 arXiv:https://doi.org/10.1137/22M1476770

[35] Diego Melgar, Timothy I Melbourne, Brendan W Crowell, Jianghui Geng, Walter Szeliga, Craig Scrivner, Marcelo Santillan, and Dara E Goldberg. 2020. Real-time high-rate GNSS displacements: Performance demonstration during the 2019 Ridgecrest, California, earthquakes. *Seismological Research Letters* 91, 4 (2020), 1943–1951.

[36] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. 1953. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics* 21, 6 (1953), 1087–1092. https://doi.org/10.1063/1.1699114

[37] Jens Oeser, Hans-Peter Bunge, and Marcus Mohr. 2006. Cluster Design in the Earth Sciences Tethys. In *High Performance Computing and Communications*, Michael Gerndt and Dieter Kranzlmüller (Eds.). Springer Berlin Heidelberg.

[38] Christian Pelties, Josep de la Puente, Jean-Paul Ampuero, Gilbert B. Brietzke, and Martin Käser. 2012. Three-Dimensional Dynamic Rupture Simulation with a High-Order Discontinuous Galerkin Method on Unstructured Tetrahedral Meshes. *Journal of Geophysical Research: Solid Earth* 117, B2 (2012), 1–15. https://doi.org/10.1029/2011JB008857

[39] C. Pelties, A.-A. Gabriel, and J.-P. Ampuero. 2014. Verification of an ADER-DG Method for Complex Dynamic Rupture Problems. *Geoscientific Model Development* 7, 3 (May 2014), 847–866. https://doi.org/10.5194/gmd-7-847-2014

[40] S. Peyrat and K. B. Olsen. 2004. Nonlinear dynamic rupture inversion of the 2000 Western Tottori, Japan, earthquake. *Geophysical Research Letters* 31 (2004).

[41] Simone Puel, Eldar Khattatov, Umberto Villa, Dunyu Liu, Omar Ghattas, and Thorsten W. Becker. 2022. A mixed, unified forward/inverse framework for earthquake problems: fault implementation and coseismic slip estimate. *Geophysical Journal International* 230, 2 (2022), 733–758. https://doi.org/10.1093/gji/ggac050 Oxford University Press.

[42] Carl Edward Rasmussen and Christopher K. I. Williams. 2006. *Gaussian processes for machine learning.* MIT Press.

[43] Zachary E Ross, Benjamín Idini, Zhe Jia, Oliver L Stephenson, Minyan Zhong, Xin Wang, Zhongwen Zhan, Mark Simons, Eric J Fielding, Sang-Ho Yun, et al. 2019. Hierarchical interlocked orthogonal faulting in the 2019 Ridgecrest earthquake sequence. *Science* 366, 6463 (2019), 346–351.

[44] Nico Schliwa, Alice-Agnes Gabriel, Jan Premus, and František Gallovič. 2024. The linked complexity of coseismic and postseismic faulting revealed by seismo-geodetic dynamic inversion of the 2004 Parkfield earthquake. *Journal of Geophysical Research: Solid Earth* 129 (2024), e2024JB029410. https://doi.org/10.1029/2024JB029410

[45] Linus Seelinger, Vivian Cheng-Seelinger, Andrew Davis, Matthew Parno, and Anne Reinarz. 2023. UM-Bridge: Uncertainty quantification and modeling bridge. *Journal of Open Source Software* 8, 83 (2023), 4748. https://doi.org/10.21105/joss.04748

[46] Linus Seelinger, Anne Reinarz, Mikkel B. Lykkegaard, Robert Akers, Amal M.A. Alghamdi, David Aristoff, Wolfgang Bangerth, Jean Bénézech, Matteo Diez, Kurt Frey, John D. Jakeman, Jakob S. Jørgensen, Ki-Tae Kim, Benjamin M. Kent, Massimiliano Martinelli, Matthew Parno, Riccardo Pellegrini, Noemi Petra, Nicolai A.B. Riis, Katherine Rosenfeld, Andrea Serani, Lorenzo Tamellini, Umberto Villa, Tim J. Dodwell, and Robert Scheichl. 2025. Democratizing uncertainty quantification. *J. Comput. Phys.* 521 (2025), 113542. https://doi.org/10.1016/j.jcp.2024.113542

[47] Linus Seelinger, Anne Reinarz, Leonhard Rannabauer, Michael Bader, Peter Bastian, and Robert Scheichl. 2021. High Performance Uncertainty Quantification with Parallelized Multilevel Markov Chain Monte Carlo. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC21) (SC '21)*. Association for Computing Machinery. https://doi.org/10.1145/3458817.3476150

[48] Dan Stanzione, John West, R. Todd Evans, Tommy Minyard, Omar Ghattas, and Dhabaleswar K. Panda. 2020-07-26. Frontera: The Evolution of Leadership Computing at the National Science Foundation. In *Practice and Experience in Advanced Research Computing 2020: Catch the Wave* (New York, NY, USA) *(PEARC '20)*. Association for Computing Machinery, 106–111. https://doi.org/10.1145/3311790.3396656

[49] Ingvar Strid. 2010. Efficient parallelisation of Metropolis–Hastings algorithms using a prefetching approach. *Computational Statistics & Data Analysis* 54, 11 (2010), 2814–2835. https://doi.org/10.1016/j.csda.2009.11.019 The Fifth Special Issue on Computational Econometrics.

[50] Taufiq Taufiqurrahman, Alice-Agnes Gabriel, Duo Li, Thomas Ulrich, Bo Li, Sara Carena, Alessandro Verdecchia, and František Gallovič. 2023. Dynamics, interactions and delays of the 2019 Ridgecrest rupture sequence. *Nature* 618, 7964 (2023), 308–315.

[51] Carsten Uphoff and Michael Bader. 2016. Generating High Performance Matrix Kernels for Earthquake Simulations with Viscoelastic Attenuation. In *2016 International Conference on High Performance Computing Simulation (HPCS)*. 908–916. https://doi.org/10.1109/HPCSim.2016.7568431

[52] Carsten Uphoff and Michael Bader. 2020. Yet Another Tensor Toolbox for Discontinuous Galerkin Methods and Other Applications. *ACM Trans. Math. Software* 46, 4 (Oct. 2020), 34:1–34:40. https://doi.org/10.1145/3406835

[53] Carsten Uphoff, Sebastian Rettenberger, Michael Bader, Elizabeth H. Madden, Thomas Ulrich, Stephanie Wollherr, and Alice-Agnes Gabriel. 2017. 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 (SC '17)*. ACM, New York, NY, USA, 21:1–21:16. https://doi.org/10.1145/3126908.3126948

[54] Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. [n. d.]. Rank-Normalization, Folding, and Localization: An Improved R^ for Assessing Convergence of MCMC (with Discussion). *Bayesian Analysis* 16, 2 ([n. d.]), 667–718. https://doi.org/10.1214/20-BA1221

[55] Sebastian Wolf, Alice-Agnes Gabriel, and Michael Bader. 2020. Optimization and Local Time Stepping of an ADER-DG Scheme for Fully Anisotropic Wave Propagation in Complex Geometries. In *Computational Science – ICCS 2020 (Lecture Notes in Computer Science)*, Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M. A. Sloot, Sérgio Brissos, and João Teixeira (Eds.). Springer International Publishing, Cham, 32–45. https://doi.org/10.1007/978-3-030-50420-5_3

[56] Sebastian Wolf, Martin Galis, Carsten Uphoff, Alice-Agnes Gabriel, Peter Moczo, David Gregor, and Michael Bader. 2022. An Efficient ADER-DG Local Time Stepping Scheme for 3D HPC Simulation of Seismic Waves in Poroelastic Media. *J. Comput. Phys.* 455 (April 2022), 1–29. https://doi.org/10.1016/j.jcp.2021.110886

[57] Stephanie Wollherr, Alice-Agnes Gabriel, and Carsten Uphoff. 2018. Off-Fault Plasticity in Three-Dimensional Dynamic Rupture Simulations Using a Modal Discontinuous Galerkin Method on Unstructured Meshes: Implementation, Verification and Application. *Geophysical Journal International* 214, 3 (Sept. 2018), 1556–1584. https://doi.org/10.1093/gji/ggy213

[58] Wenzheng Yang and Egill Hauksson. 2013. The tectonic crustal stress field and style of faulting along the Pacific North America Plate boundary in Southern California. *Geophysical Journal International* 194, 1 (2013), 100–117.

Received 12 September 2024; revised 15 March 2025; accepted 15 April 2025