Data Assimilation For Advection-Diffusion Flows By Interconnected Localized Filters: a case study of Allscale API

Emanuele Ragnoli, Fearghal O'Donncha, Albert Akhriev

IBM Research Ireland

September 25, 2017

The Problem

- An oil spill is the release of a liquid petroleum hydrocarbon into the environment, which is unfortunate but possible event.
- Marine fauna is especially vulnerable.
- Monitoring oil contamination over time in the affected area is an important part of clean up operation.
- Assuming that the level of pollution can measured at some locations within contaminated region (e.g. by remote sensing), the process of spreading the oil can be described by the well-known physical model.

The Model

- The equation for the process of substance (oil) dissemination driven by two physical phenomena – advection and diffusion:

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) - v_x\frac{\partial u}{\partial x} - v_y\frac{\partial u}{\partial y} + f(x, y, t),$$

  where $u = u(x, y, t)$ is oil concentration, D is diffusion coefficient, (v_x, v_y) are components of flow velocity, and f is contamination source term.
- In simplified set-up D is a constant, (v_x, v_y) depend on time only, f is a high concentration spot at $t = 0$ around one point and zero elsewhere, and $u(\Gamma) = \frac{\partial u(\Gamma)}{\partial \mathbf{n}} = 0$ on the boundary Γ of the domain Ω.

Data Assimilation

- Observations are available at few points of the domain .
- In order to obtain full picture of oil dispensing, the information should be spread from observation points to their unobservable neighbours.
- The information propagation is achieved by means of Kalman filtering.
- Kalman filter updates a (sub-)domain field of contaminant distribution utilizing information at observation points (sensors).
- Filtering drags the simulation process (which starts from zero field) towards the true state of the nature provided by measurements.

Computational Problem

- Had it been applied to the whole domain Ω, the Kalman filter would be prohibitively expensive as it comprises inversion of a dense matrix ($O(N^3)$).
- Also, the higher resolution (the larger problem size N) – the smaller time step (the larger number of iterations) is needed to satisfy the CFL condition. As such, each step should be fast enough to make reasonable computation time.

Domain sub-division

- Dividing the domain into sub-domains is a common circumvention to the large computational problems.
- Allscale API provides advanced facilities for domain partitioning and parallel framework for efficient solving a set of smaller sub-problems.
- In Amdados application we solve a number of sub-problems in parallel and independently.
- In order to approximate the global solution, a variant of Schwarz method iteratively refines the solution by exchanging the information at sub-domain boundaries, which is a relatively cheap operation.

Amdados application

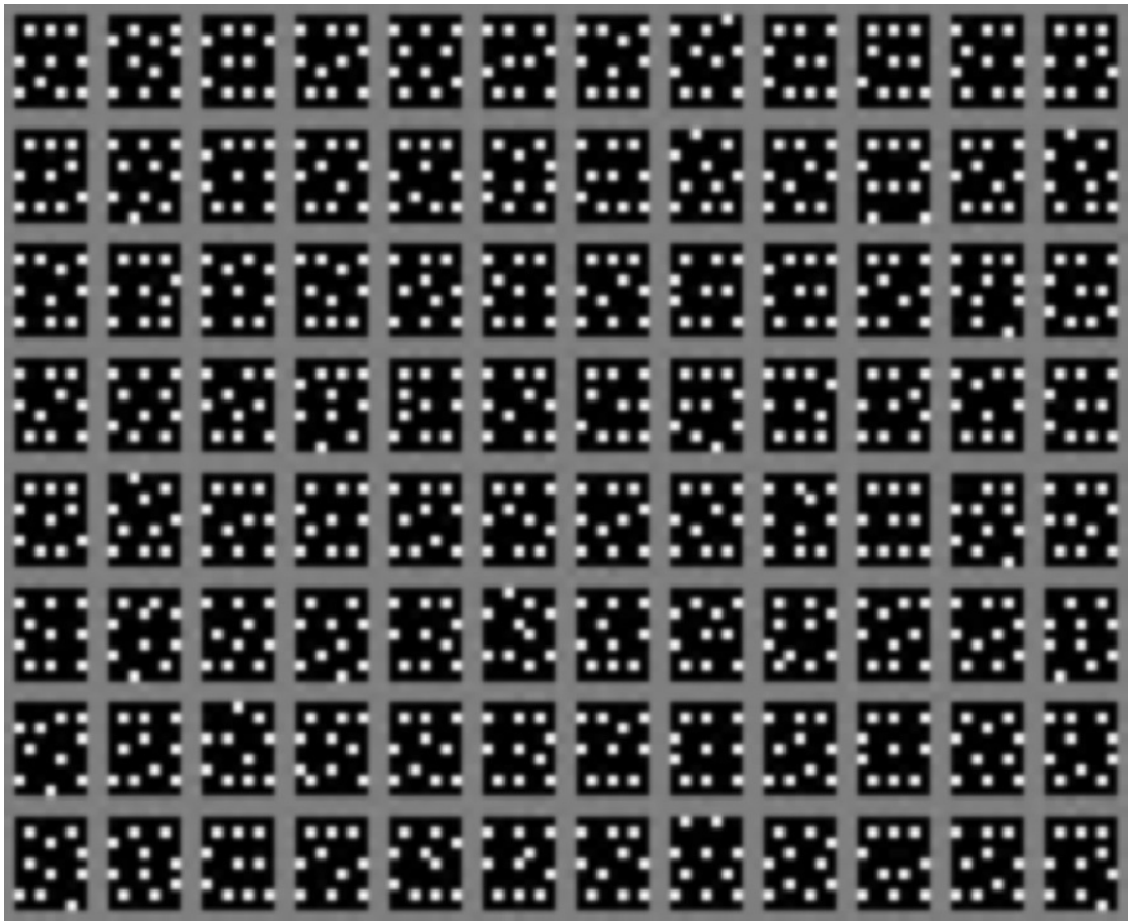
- Solves the data assimilation problem based on advection-diffusion equation.
- Demonstrates the power of Allscale API in solving the challenging real world problems.
- Currently implements an implicit advection-diffusion solver based om finite-difference discretization applied to a sub-domain.
- The global solution is refined by Kalman filtering coupled with Schwarz method that exchanges the information at the sub-domain boundaries.
- More advanced approaches (FEM, Minimax filtering) are currently investigated and partially implemented.

Amdados: the observations

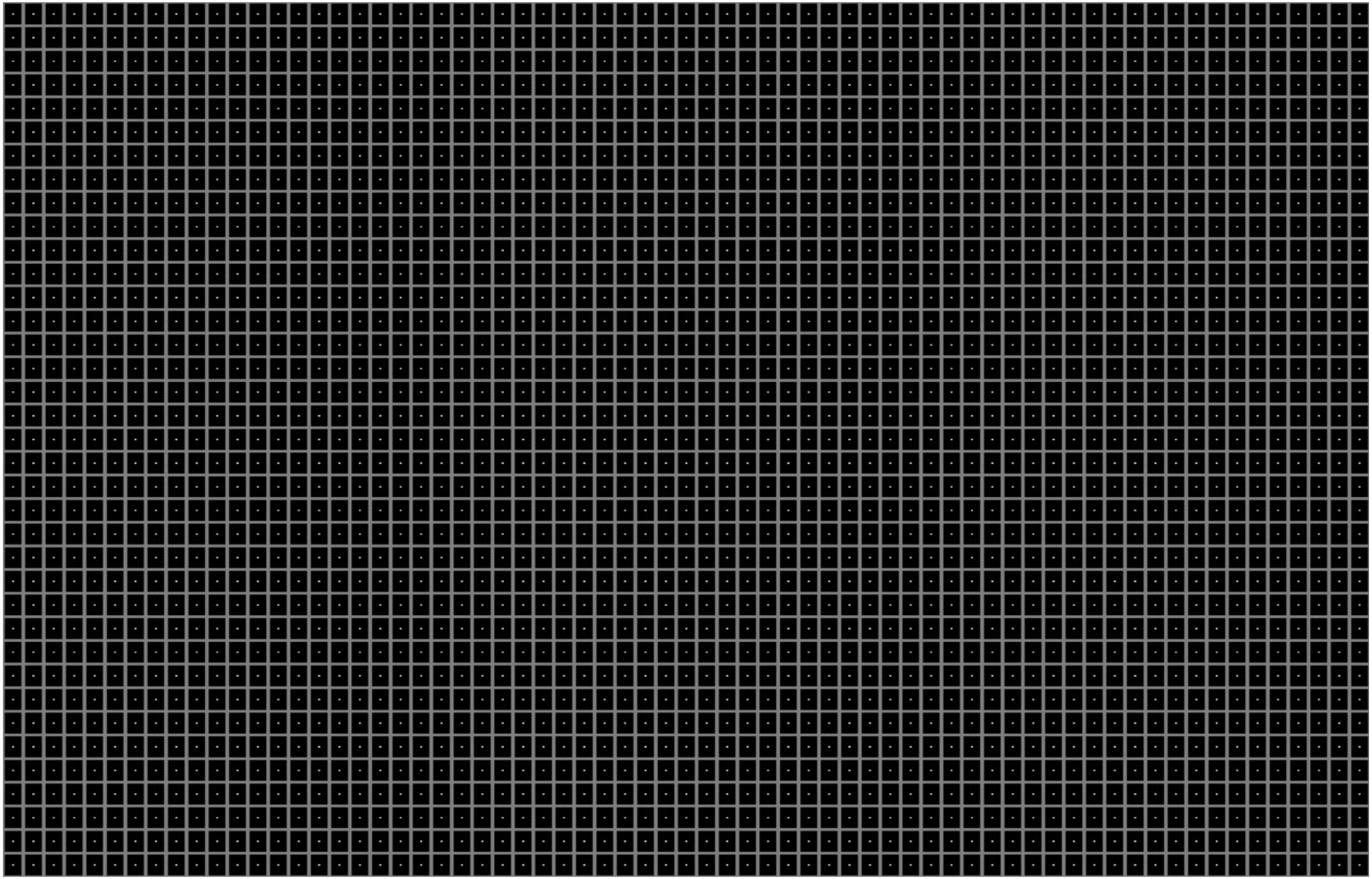
- In practice, the true state of the nature is available at a (small) subset of domain points where the sensors can measure the concentration.
- In our demo, the measurements (aka analytic solution) are synthesized by the global implicit forward solver written in Python3.
- The solver does not do domain decomposition (having limited scalability) and does not apply any filtering just forward time-marching of advection-diffusion PDE.
- The solution at each time step is recorded into the file of observations and used in simulation as the sensor measurements.

Amdados: the measurements

- The "sensors" are seeded randomly inside each sub-domain.
- Only observations at the sensor locations are used during simulation.
- White dots show "sensors"; gray lines outline the sub-domain borders.



Example: sensors in high resolution



Amdados: the simulation

- The goal is to synthesize contamination distribution inside the domain given measurements of the "true" field at a (small) sub-set of points where "sensors" are located.
- We start with zero field at $t = 0$ because the true state of nature is not known in advance.
- Gradually, the Kalman filter integrated with Schwarz method (for exchanging the information across sub-domain borders) makes the simulated field look quite similar the true one.
- In this way, a reasonable estimation of contamination level can be done in unobserved domain points on each time step.

Amdados: Kalman filter

- *Process model*: $\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t$, where $\mathbf{x}$ is the state field (space distribution of contaminant), $\mathbf{A}$ is the model matrix (finite-difference discretization) and $\mathbf{w}$ is the process noise.
- *Implicit method*: $\mathbf{B}\mathbf{x}_{t+1} = \mathbf{x}_t + \tilde{\mathbf{w}}_t$, $\mathbf{B} = \mathbf{A}^{-1}$.
- *Measurement residual*: $\mathbf{y}_t = \mathbf{z}_t - \mathbf{H}\mathbf{x}_t$ where $\mathbf{z}$ is the vector of observations at sensor locations, $\mathbf{H}$ is a low-rank observation matrix.
- *Prior state estimation*: $\hat{\mathbf{x}}_{t+1} = \mathbf{A}\mathbf{x}_t$, $\hat{\mathbf{P}}_{t+1} = \mathbf{A}_t\mathbf{P}_t\mathbf{A}_t^T + \mathbf{Q}_t$, where $\mathbf{P}$ is the process covariance matrix and $\mathbf{Q}$ is the process noise covariance.
- *Posterior estimation*: $\mathbf{x}_{t+1} = \hat{\mathbf{x}}_{t+1} + \mathbf{K}_t\mathbf{y}_t$, $\mathbf{P}_{t+1} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\hat{\mathbf{P}}_{t+1}$ where $\mathbf{K}$ is the Kalman gain matrix.
- Several posterior estimation steps are consecutively repeated in Schwarz method until sub-domain fields match at borders.

Amdados: Schwarz iterations

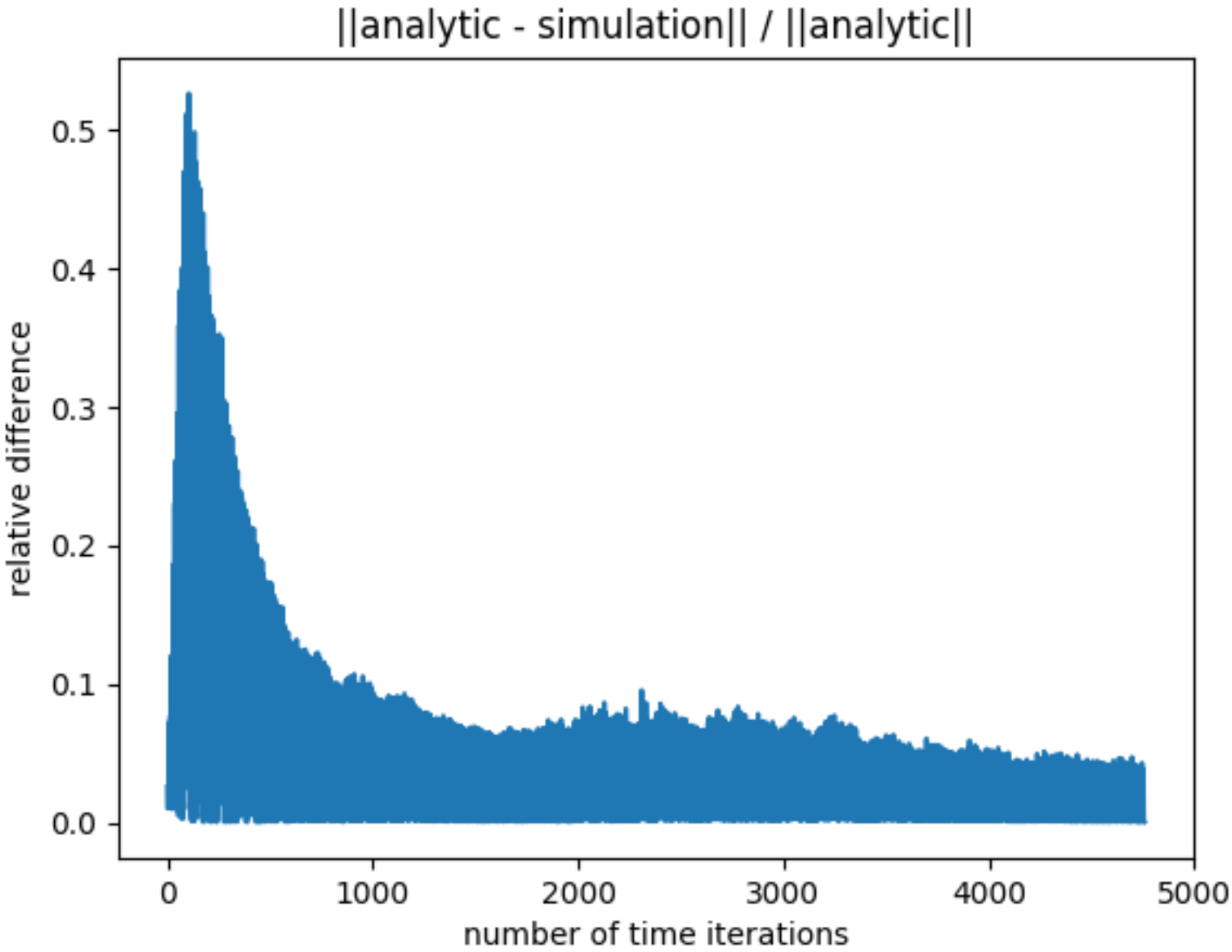
- Each sub-problem is propagated forward in time independently. As such, new estimations can significantly disagree along sub-domain borders.
- The Schwarz method solves a boundary value problem for a PDE approximately by splitting it into boundary value problems on smaller domains and iteratively combining the results. We use adaptive AND/ARN variant by Gastaldi et al., 1998.
- If flow coming into a sub-domain across a border (Up, Down, Left or Right), the border values of the neighbor sub-domain replace the corresponding border values of this one.
- If flow going out of the sub-domain, the Neumann conditions are imposed on the respective border.

Amdados: the algorithm

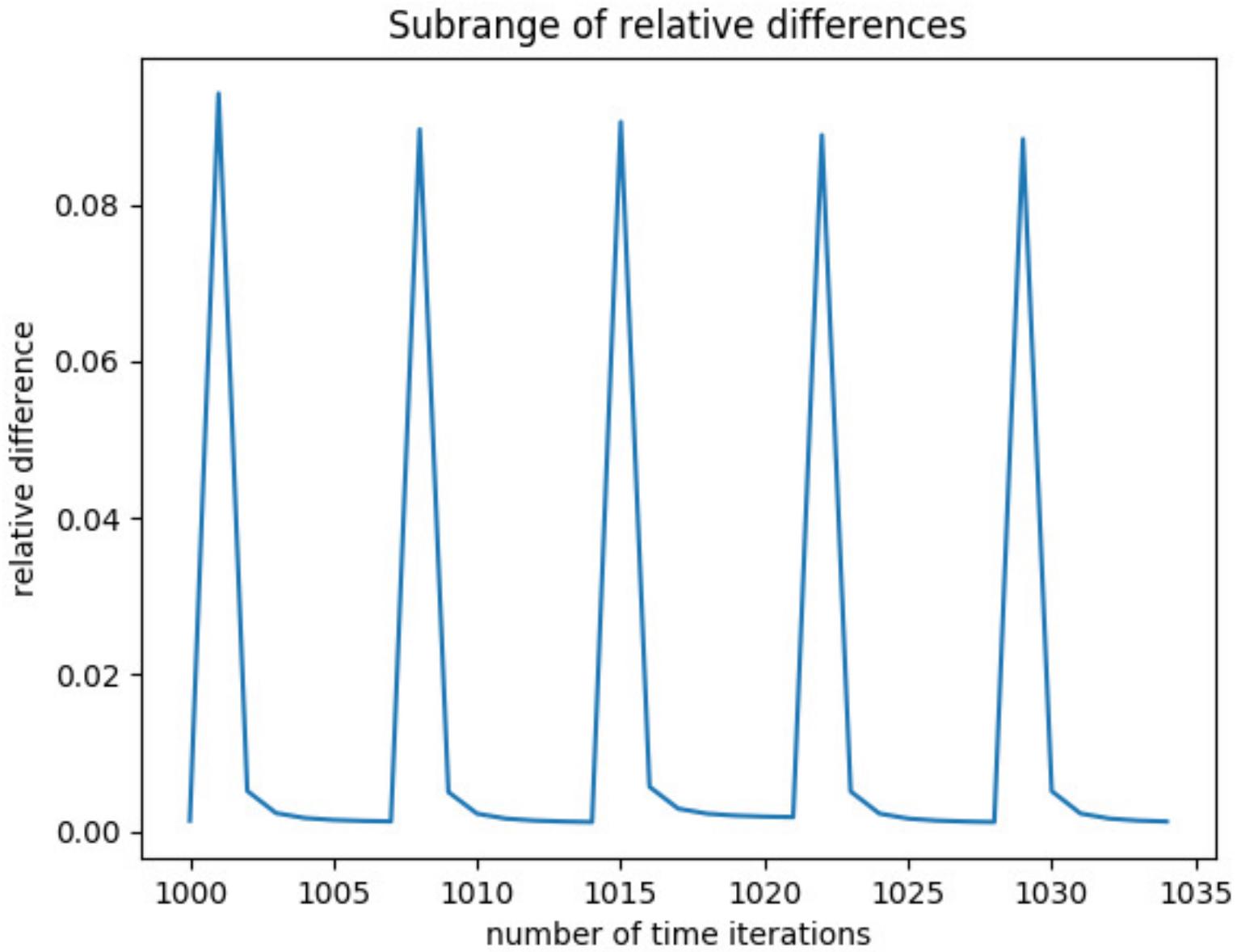
1: Start from zero field $\mathbf{x}_0 = 0$ and near-identity covariance $\mathbf{P}_0 \sim \mathbf{I}$.
2: **for** t $= 0{:}N_t$ **do**
3: Propagate state one step ahead $\hat{\mathbf{x}}_{t+1}$, $\hat{\mathbf{P}}_{t+1}$.
4: Get observation vector $\mathbf{z}_t$.
5: **for** t $= 1{:}N_{schwarz}$ **do**
6: Get posteriors $\mathbf{x}_{t+1}$, $\mathbf{P}_{t+1}$ by solving Kalman filter.
7: Apply Schwarz update.
8: Apply boundary conditions at the global border.
9: **end for**
10: Replace by the last posteriors: $\{\mathbf{x}_t, \mathbf{P}_t\} \leftarrow \{\mathbf{x}_{t+1}, \mathbf{P}_{t+1}\}$.
11: $t \leftarrow t + 1$
12: **end for**

N O T E: **fixed** number of Schwarz iterations – global reduction is discouraged by Allscale API.

Schwarz iterations: relative error



Closer look at the relative error



Thank You!