Building Gaussian Process Statistical and Quantitative Learning Framework for Scientific Applications

X. Sherry Li
xsli@lbl.gov
Lawrence Berkeley National Laboratory

From Modeling to Learning with HPC
September 13-14, ICERM

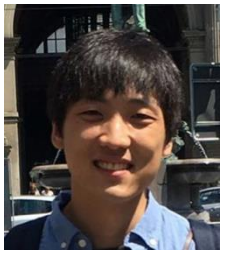
Team

Sherry Li
LBNL

Jim Demmel
UC Berkeley

Yang Liu
LBNL

Younghyun Cho
Santa Clara U.



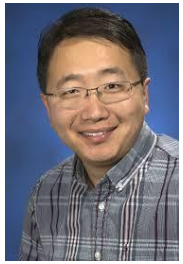
Hengrui Luo
Rice U.

David Trebotich
LBNL

Marcus Noack
LBNL

Xiaofeng Gu
BNL

Ji Qiang
LBNL



Yue Hao
BNL

Application drivers

- Optimizing HPC codes on real machines (runtime, energy, …)
  - Parameters setting greatly influence performance
  - Autotuning to find optimal parameters with limited number of runs
- Building a trustworthy digital twin for a physical system
  - Quantify the uncertainties of the simulation model with respect to the physical phenomena
- Setting the optimal operating configurations for scientific apparatuses and instruments

=> Bayesian statistical learning framework can treat the application as a black-box function and use Gaussian Process regression to estimate the mean function and the variance in distribution

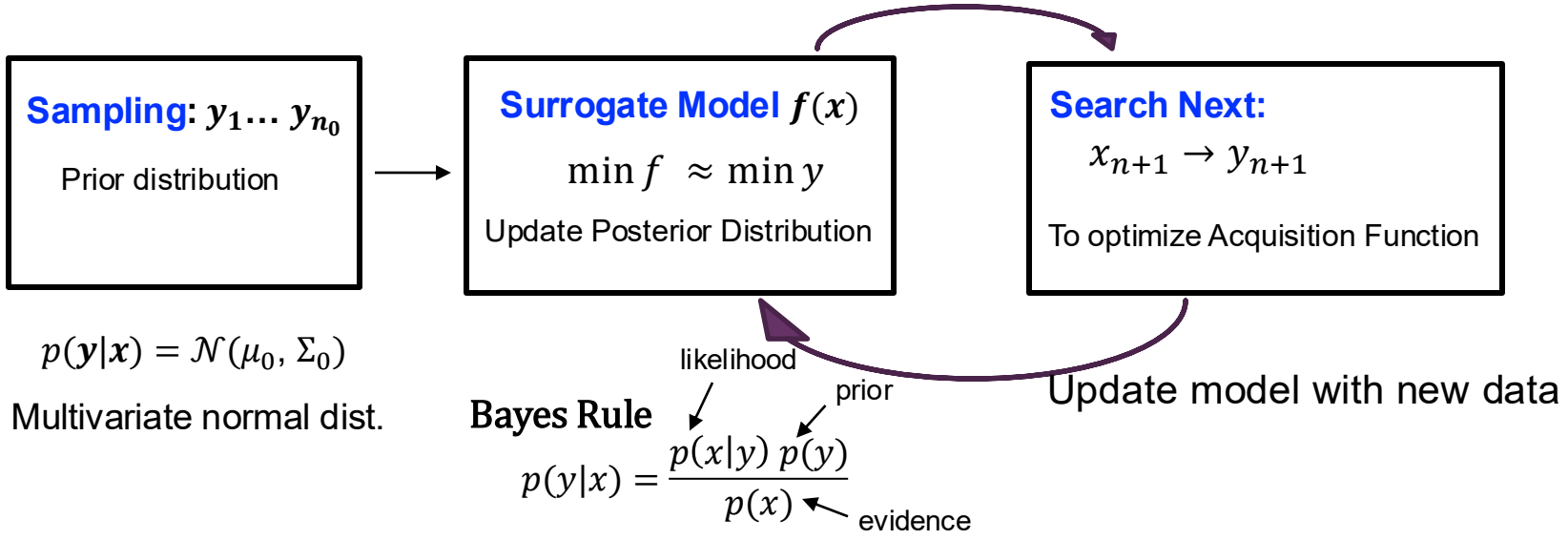
Outline

- Introduction of Gaussian Process
- Applications
  - Autotuning
  - Operation of particle accelerators
  - Decision-making in adaptive mesh refinement

- Software
  - GPTune: https://gptune.lbl.gov/
    - Multitask and transfer learning for black-box optimization with Bayesian statistics
  - gpCAM: https://gpcam.lbl.gov/
    - Uncertainty quantification, autonomous data acquisition, and HPC Bayesian optimization

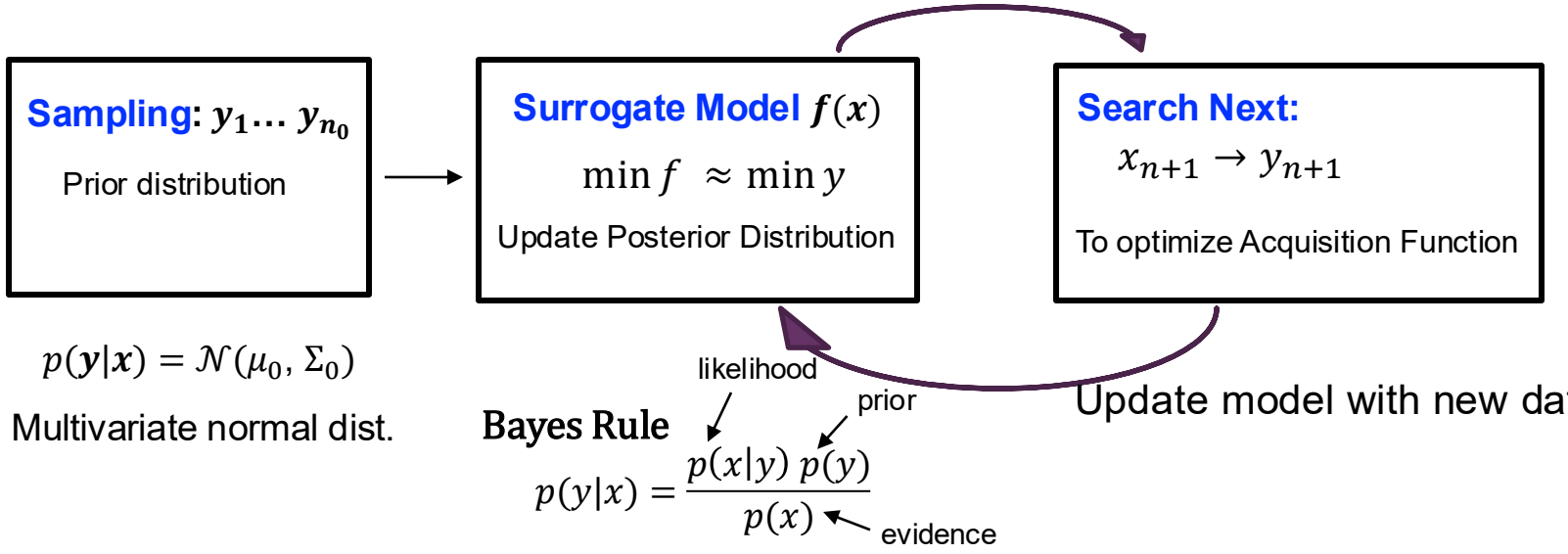
Statistical learning via Bayesian optimization

- Problem: $\min_x y(t,x)$, t : task, x : parameter configuration
- Bayesian statistical inference is an iterative model-based approach



Statistical learning framework: Bayesian optimization

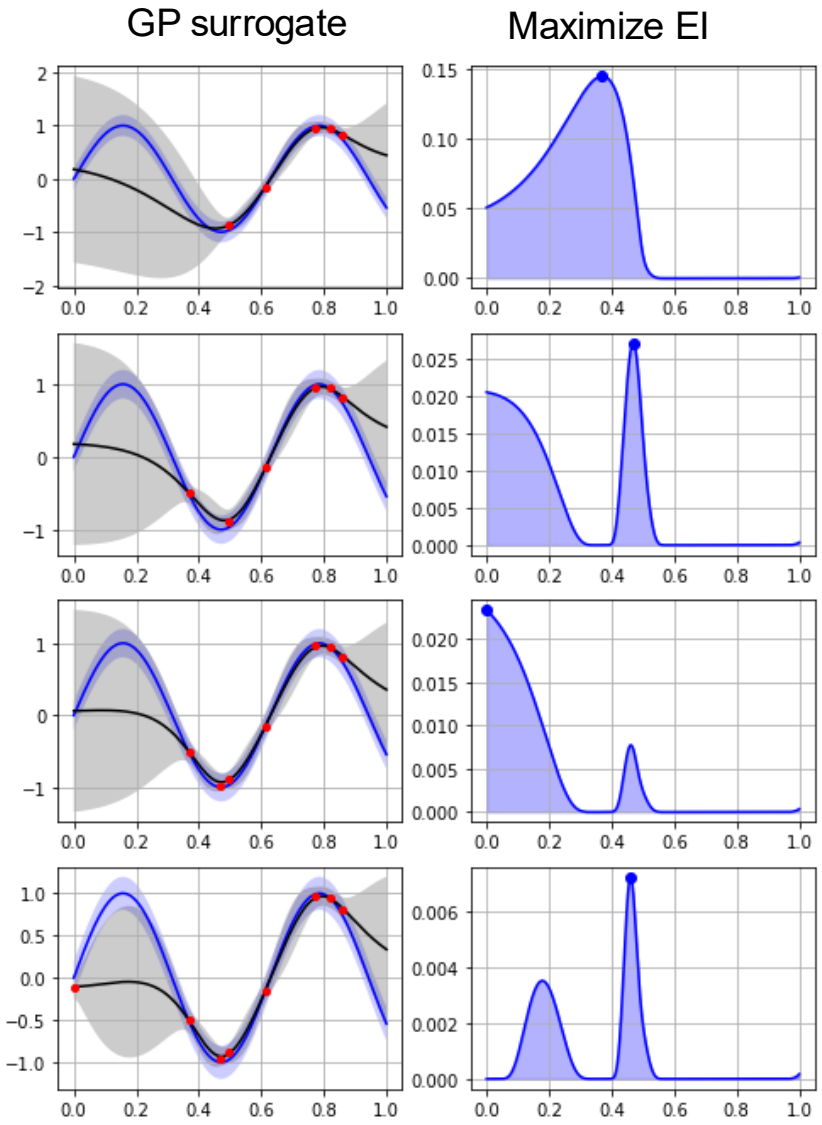
- Problem: $\min_x y(t,x)$, t : task, x : parameter configuration
- Bayesian statistical inference is an iterative model-based approach



- Gaussian Process regression
  - LCM (multitask learning)
  - cGP
- Deep GP
- Other models …

- Expected Improvement (EI)
  - Maximize EI: PSO, CMA-ES, …
- Other acquisition functions …

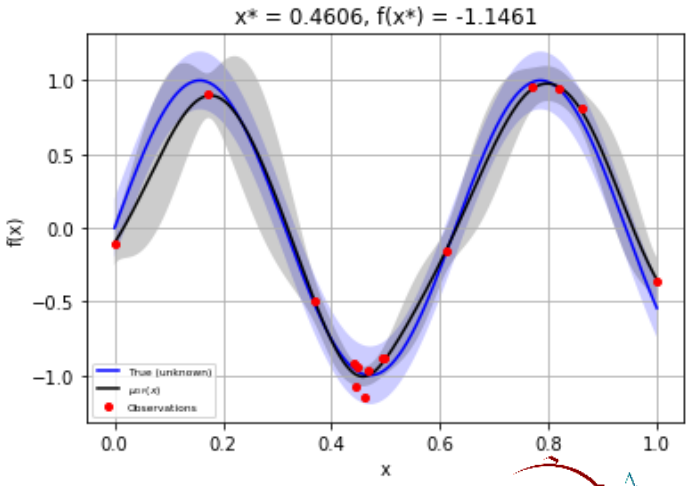
1D example: black-box function $y(x) = sin(10x)$



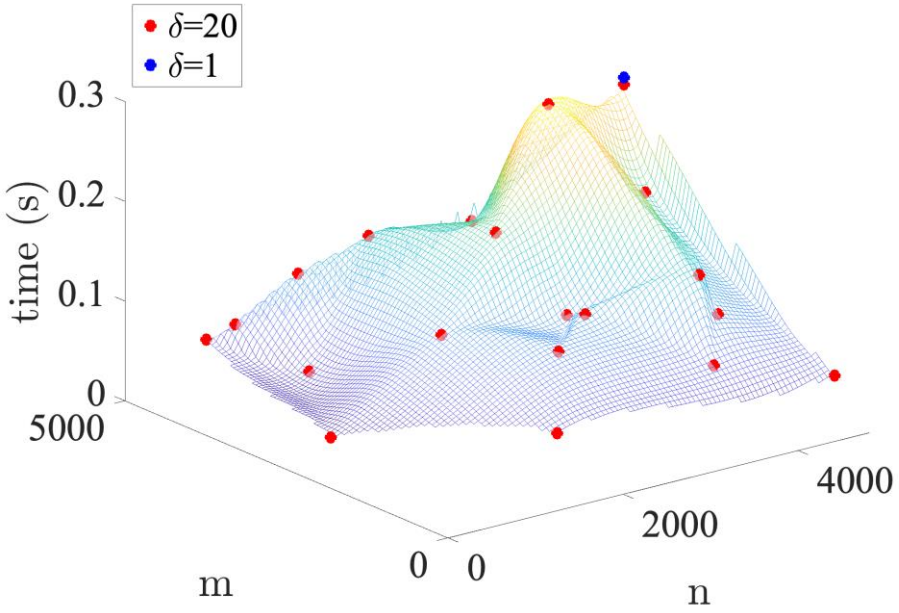
5 initial samples

4 additional steps

- Blue line: true function
- Red dots: function evaluations
- Black line: mean function of the fitted surrogate model
- Grey shaded area is 95% confidence interval



Modeling



Gaussian Process Regression

"Gaussian Processes for Machine Learning", Rasmussen and Williams 2006

Multitask learning LCM: GP for vector-valued functions

"Kernels for Vector-Valued Functions", Alvarez, Rosasco, Lawrence, 2012

Gaussian Process (GP) surrogate model

- GP defines a distribution over functions, specified by the mean function and covariance function $k(x, x')$ (kernel)

$$f(x) \sim GP\big(\mu(x), k(x, x')\big)$$

$$\mu(x) = \mathbb{E}[f(x]$$

$$k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))]$$

Gaussian Process (GP) surrogate model

- GP defines a distribution over functions, specified by the mean function and covariance function $k(x, x')$ (kernel)

$$f(x) \sim GP\big(\mu(x), k(x, x')\big)$$

$$\mu(x) = \mathbb{E}[f(x]$$

$$k(x, x') = \mathbb{E}[(f(x) - \mu(x))(f(x') - \mu(x'))]$$

- GP model prediction:

  Given s observation pairs:

$$X = [x^1, x^2, \ldots, x^s], \quad Y = [y(x^1), y(x^2), \ldots, y(x^s)]$$

  Add a new point x^*, update posterior prob. distribution

  - New mean for $y(x^*)$:

$$\mu_* = \mu(X) + K(x^*, X)\, K(X, X)^{-1}\, (Y - \mu(X))$$

  - New variance for $y(x^*)$:

$$\sigma_*^2 = K(x^*, x^*) - K(x^*, X)\, K(X, X)^{-1} K(x^*, X)^T$$

Multitask Learning LCM
– extending GP to vector-valued functions

- Consider a set of correlated objective functions $\{y_i(X)\}_{i\in 1..\delta}$ (i.e., multiple tasks) and GP models $\{f_i(X)\}_{i\in 1..\delta}$
- Linear Coregionalization Model (LCM) attempts to build a joint model : $\{u_q\}_{i\in 1..Q}$ (GP) encoding the shared behavior

$$f_i(x) = \sum_{q=1}^{Q} a_{i,q} u_q(x)$$

with

$$k_q(x, x') = \sigma_q^2 \exp\left(-\sum_{i=1}^{D} \frac{(x_i - x_i')^2}{l_i^q}\right)$$

Multitask Learning LCM – extending GP to vector-valued functions

- Consider a set of correlated objective functions $\{y_i(X)\}_{i\in 1..\delta}$ (i.e., multiple tasks) and GP models $\{f_i(X)\}_{i\in 1..\delta}$
- Linear Coregionalization Model (LCM) attempts to build a joint model : $\{u_q\}_{i\in 1..Q}$ (GP) encoding the shared behavior

$$f_i(x) = \sum_{q=1}^{Q} a_{i,q} u_q(x)$$

with

$$k_q(x, x') = \sigma_q^2 \exp\left(-\sum_{i=1}^{D} \frac{(x_i - x_i')^2}{l_i^q}\right)$$

"Big" covariance matrix includes both auto-covariance and cross-covariance

$$\Sigma\left(x_i^m, x_j^n\right) = \sum_{q=1}^{Q} a_{i,q} a_{j,q}\, k_q\left(x_i^m, x_j^n\right) + d_i \delta_{i,j}\, \delta_{m.n}$$

Modeling complexity

"Invert" K: dimension =

- Number of samples (single task learning)
- Number of samples X number of tasks (LCM)

- Dense: $O(N^{3})$
- Data-sparse (off-diagonal low rank): $O(N^{\alpha}\ polylog(N)\ )$
  - Apply HSS in STRUMPACK

    [Chavez, Liu, Ghysels, Rebrova, L, IPDPS 2020]
  - Apply H^2 in H2Pack

    [Huang, Xing, Chow, TOMS 2021]

Search Phase

- Where to place the new sample point?
  - Optimization based on surrogate model (easier!)

- Given a new sample point, need quickly update the model

Search for a point to maximize Acquisition Function

(… another optimization problem, but easier)

- Balance between exploitation and exploration
  - **Exploitation**: local search within promising regions
  - **Exploration**: global search of new regions with more uncertainty
- **Expected Improvement (EI)** – most commonly used AF

  For new point x_i^*, expected difference from current best is

$$\Delta(x_i^*) = \mu_i^* - y_i^{min}$$

$$EI(x_i^*) = \mathbb{E}\left[\left[y_i^* - y_i^{min}\right]^+\right]$$

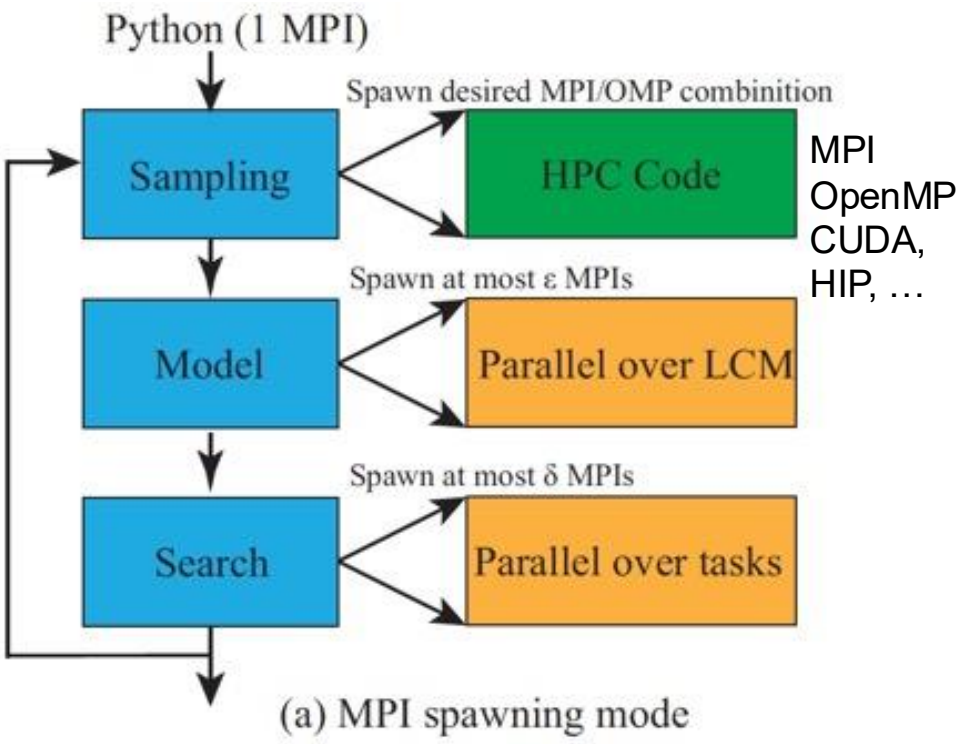
[Jones et al. 1998]

$$= [\Delta(x_i^*)]^+ + \sigma_i^* \varphi\left(\frac{\Delta(x_i^*)}{\sigma_i^*}\right) - |\Delta(x_i^*)|\Phi\left(\frac{\Delta(x_i^*)}{\sigma_i^*}\right)$$

  - $\varphi(.)$: probability density function
  - $\Phi(.)$: cumulative distribution function

GPTune tuning workflow on parallel machines

Parallel execution model



GPTune advanced features

- History database
- Multi-objective optimization
- Multi-fidelity optimization
- Users' performance models or hardware performance counters to guide tuning
- Clustered GP for non-smooth function surface
- CK-GPTune workflow

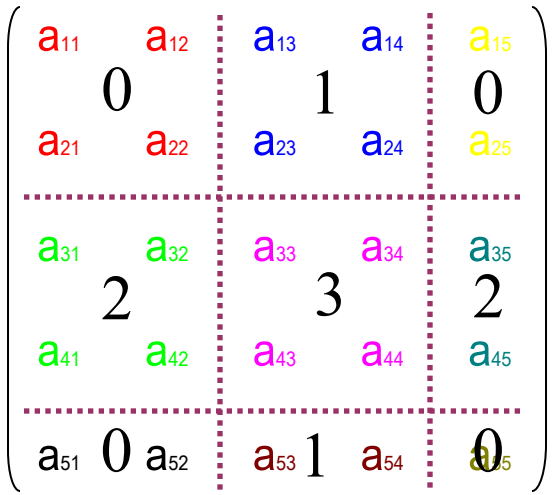
GPTune: Multitask Learning for Autotuning Exascale Applications, Proc. of Principles and Practice of Parallel Programming, 2021.
GPTuneBand: Multi-task and Multi-fidelity Autotuning for Large-scale High Performance Computing Applications, SIAM PP22.
Non-smooth Bayesian Optimization in Tuning Problems, arXiv preprint arXiv:2109.07563
Enhancing Autotuning Capability with a History Database, McSOC-2021, Special Session: Autotuning for Multicore & GPU.

Parallel dense QR factorization in ScaLAPACK

- 2D block-cyclic layout
- Task: {m, n}
- 3 Parameters: {mb, nb, p}
  (fixed nprocs=pxq)

- Conventional wisdom (Users' Guide):
  - Make block as square as possible
  - Make process grid as square as possible

$m = n = 5, mb = nb = 2, p$=2

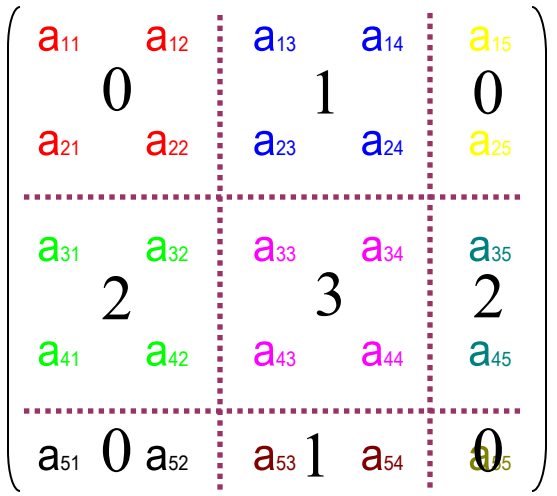


Parallel dense QR factorization in ScaLAPACK

- 2D block-cyclic layout
- Task: {m, n}
- 3 Parameters: {mb, nb, p}
  (fixed nprocs=pxq)

- Conventional wisdom (Users' Guide):
  - Make block as square as possible
  - Make process grid as square as possible

$m = n = 5, mb = nb = 2, p$=2



Cori machine @ NERSC:

n	mb	nb	p	q	time
2000	4	8	2	12	0.20
10000	1	26	16	192	1.61

Multi-task vs single task: fusion simulation codes

- Solve the extended magnetohydrodynamic equations. Used for calculating the equilibrium, stability, and dynamics of fusion plasmas. Critical for tokamaks design, e.g., ITER
- Linear solver can take > 80% simulation time
  - Both use SuperLU as block-Jacobi preconditioner to GMRES
- M3D-C1: **5 parameters;** NIMROD: **7 parameters**
- Task defined as # of time steps (t) in nonlinear iterations
- 80 samples: 1 x 80 vs. 4 x 20

	M3D-C1 (32 cores)		NIMROD (192 cores)	
	Min time	Total time	Min time	Total time
Single-task	11.19	12310	112.7	14710
Multitask	11.17	7797	112.8	9559

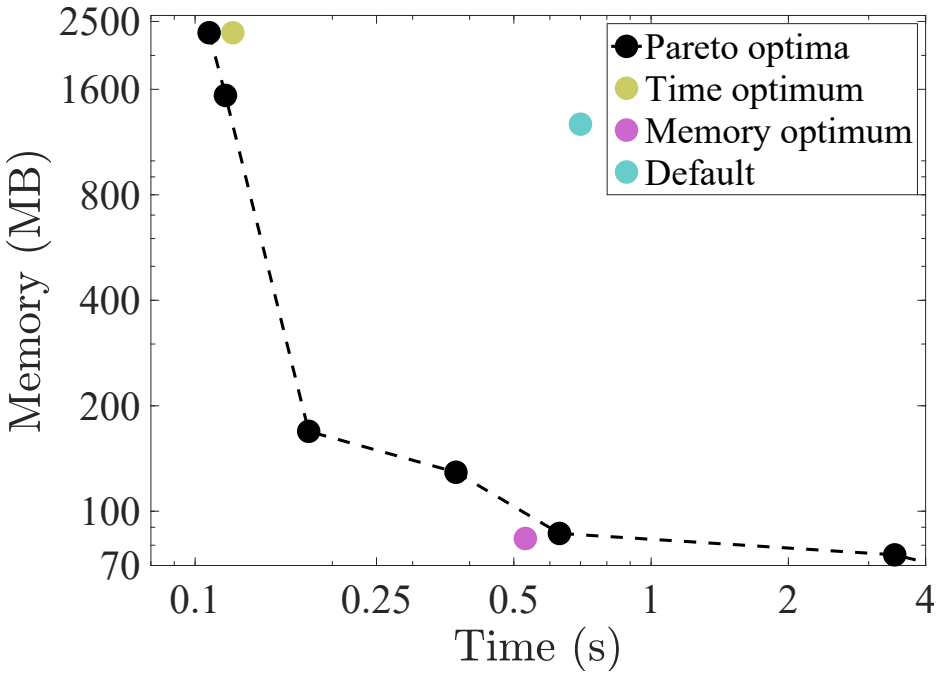
Single task: t = 3
Multitask: t = 1,1,1,3

Single task: t = 15
Multitask: t = 3,3,3,15

Multi-objective tuning for SuperLU_DIST

parallel sparse direct linear solver

- $\mathbb{IS} = [\text{matrix name}]$, $\mathbb{PS} = [\text{colperm, maxsuper, relax, nprow}]$
- Two objectives: $\mathbb{OS} = [\text{time}, \text{memory}]$
- Returns multiple parameter configurations
- Pareto optimal: best time and memory tradeoff (no other $\mathbb{PS}$ points dominate over this point in both objectives)



matrix "Si2"

80 samples

Tuning operational parameters for scientific instrument

DOE Nuclear Physics Office

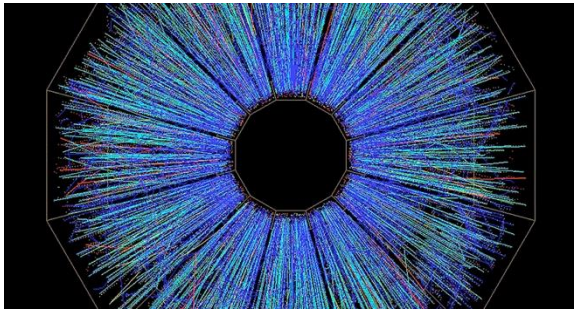
Collaboration between LBNL & Brookhaven Lab

[Gu, Hao, Qiang, Li, Liu, IPAC24]

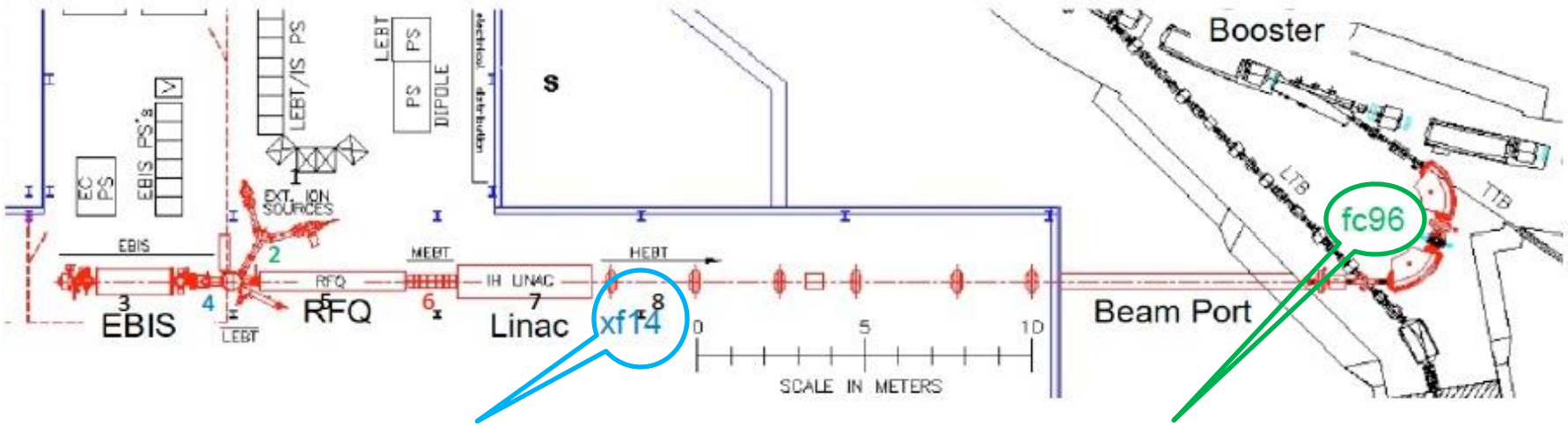
Electron Beam Ion Source (EBIS) at BNL is a heavy ion accelerator, producing high-intensity pulses of ions. It serves as a pre-injector system for

- Relativistic Heavy Ion Collider (RHIC)
- NASA Space Radiation Laboratory (NSRL)

Each beamline section has ~10 operational parameters that influence beam output intensity



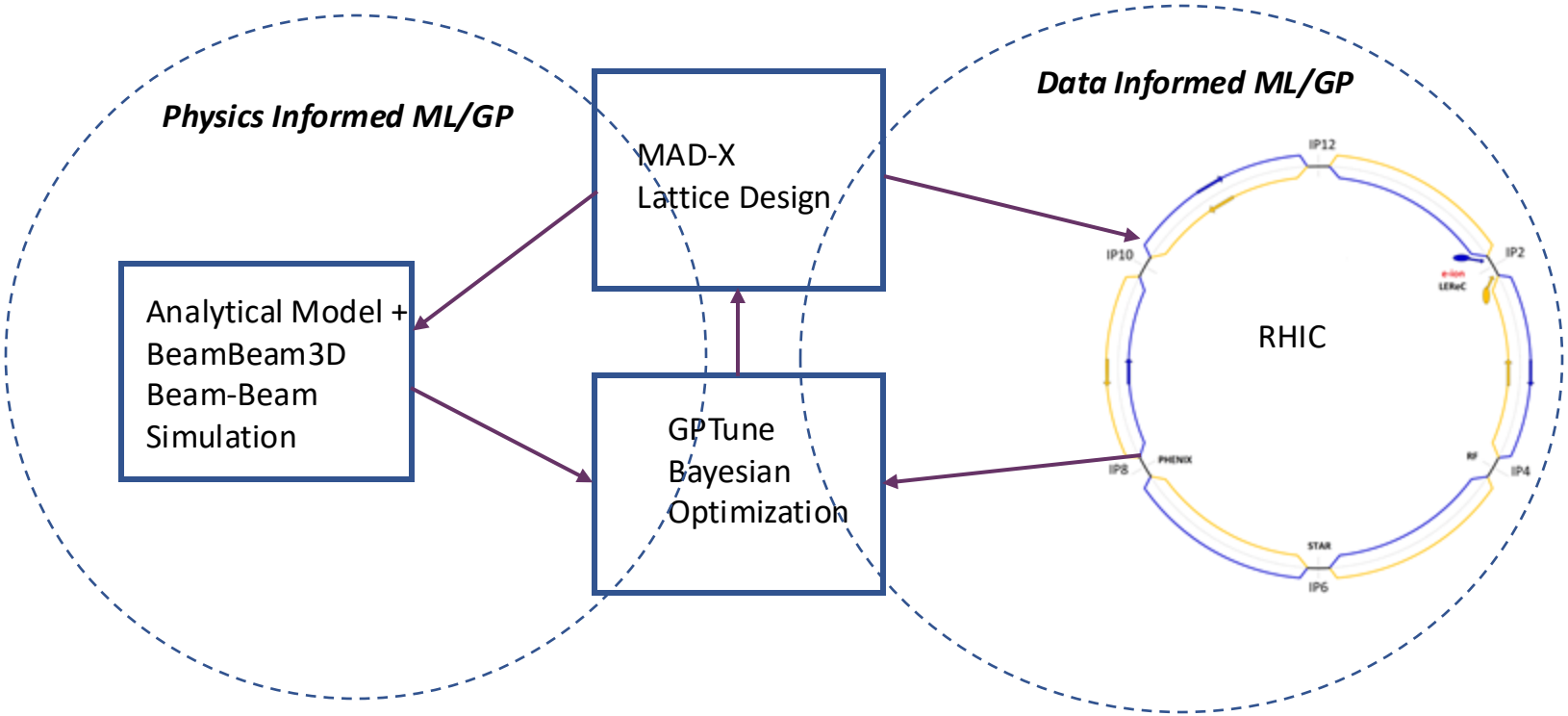
Collision of two 30-billion electron-volt gold beams in the STAR detector at RHIC



EBIS extraction line (10 params.)

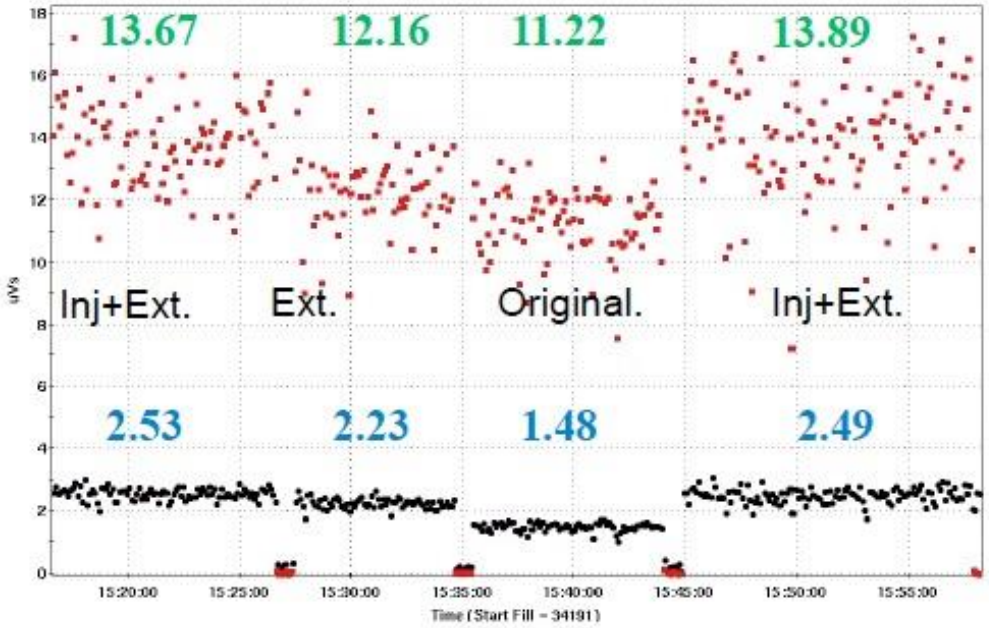
EBIS injection line (9 params.)

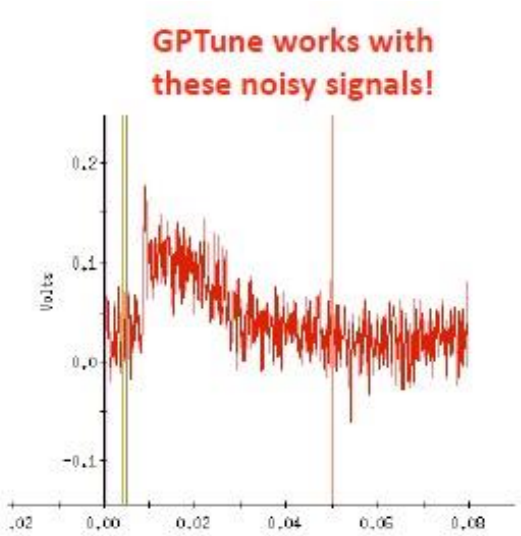
Advanced modeling for RHIC luminosity optimization



- We conducted a pioneering study, connecting GPTune to the measurements of Injection and Extraction beamlines
- Transfer learning improves the BO performance in RHIC luminosity optimization by using the GP model trained by the physics simulation.

GPTune online optimization improves EBIS performance

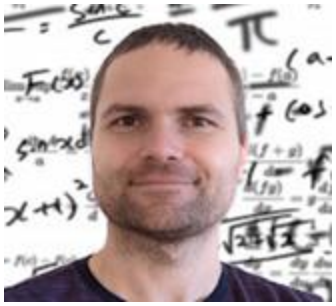




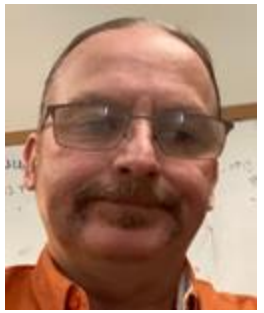
Intensity detector	Original	Ext. optimized with xf14	Gain	+ Inj. Optimized with FC96	Total Gain
xf14 (uVs)	1.48	2.23	42~50%	2.53/2.49	68~71%
fc96 (uVs)	11.22	12.61	8.4%	13.67/13.89	22-24%

Adaptive Mesh Refinement (AMR) – Chombo

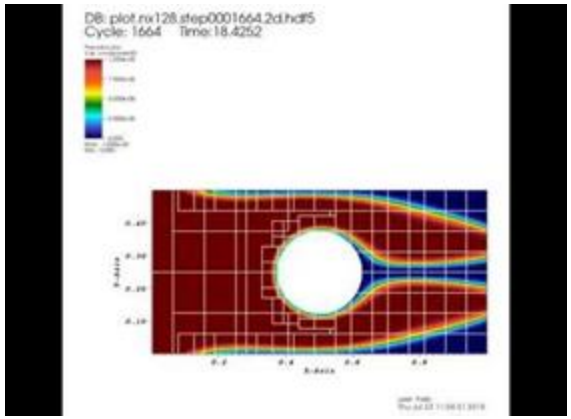
David
Trebotich



Marcus
Noack



Dan
Graves



Traditional AMR

- Deterministic — label data *a posteriori* to interrogate during simulation cf. refinement criteria (e.g., vorticity)
- Regridding interval is fixed, typically based on number of steps relative to grid resolution and stability
- Desired resolution is predetermined by user — cannot refine more or less than initial refinement choice
- Traditionally limited to box or patch data structures in finite difference / volume discretizations
- Requires problem-dependent expertise and *user experience* to get to an effective simulation

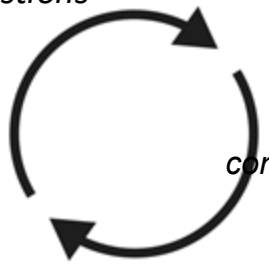
*Objective: make AMR more **anticipatory** to better predict what, when and where to refine with less user interaction and experience*

⇒ **Stochastic AMR: AMR via UQ-driven decision-making**

gpAMR is a stochastic AMR approach that uses Gaussian Processes and resulting UQ as refinement criterion

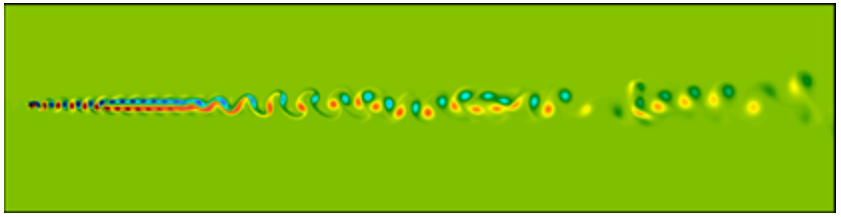
Refine and solve next PDE step

communicate suggestions

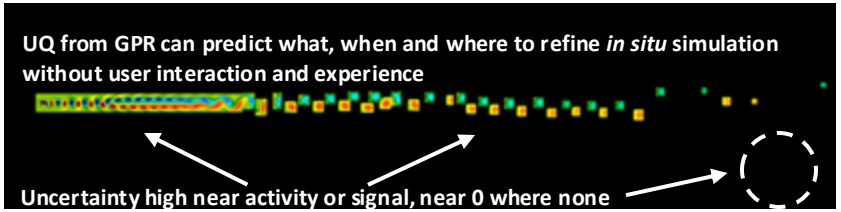


communicate latest solution

Query GP (UQ) for refinement or coarsening



Canonical turbulent flow past cylinder with vortex street instability layer



AMR finest grid data tracking with wake structures and individual vortices. Only 5% of domain is gridded at finest level

However, vanilla GPs does not work …

BERKELEY LAB

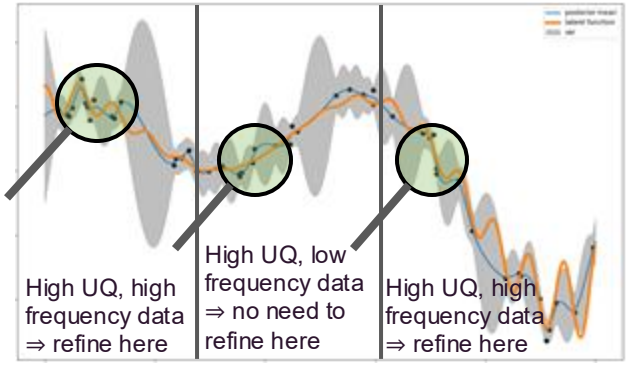
gpAMR: UQ-Driven Adaptive Mesh Refinement

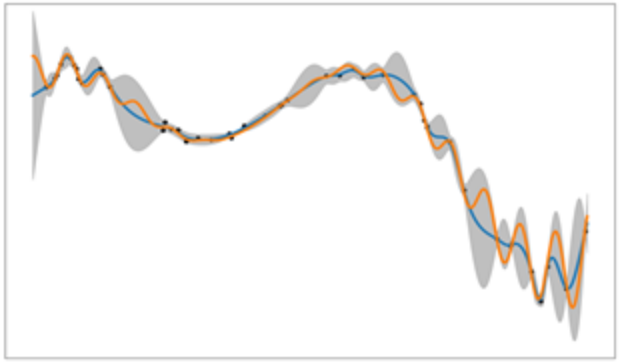
- GPs in their basic form use stationary kernels which depend only on distance between two points

$$k_{stat}(\mathbf{x}_1, \mathbf{x}_2) = \sigma_s^2 \left(1 - \frac{\sqrt{3d}}{l}\right) \exp\left[-\frac{\sqrt{3d}}{l}\right]$$

  Stationary kernels in a GP predict high uncertainty in places of data sparsity which is not what you want in AMR because it would end up refining everywhere.

- Use novel acquisition functions for multi-point suggestions of AMR "tagging" in the PDE-based simulation





$$k_{non}(\mathbf{x}_1, \mathbf{x}_2) = f(\mathbf{x}_1) f(\mathbf{x}_2)\, k_{stat}(\mathbf{x}_1, \mathbf{x}_2)$$

$$f(x) = \sum_i^N \alpha_i \beta(\mathbf{x}_i, \mathbf{x}; w)$$

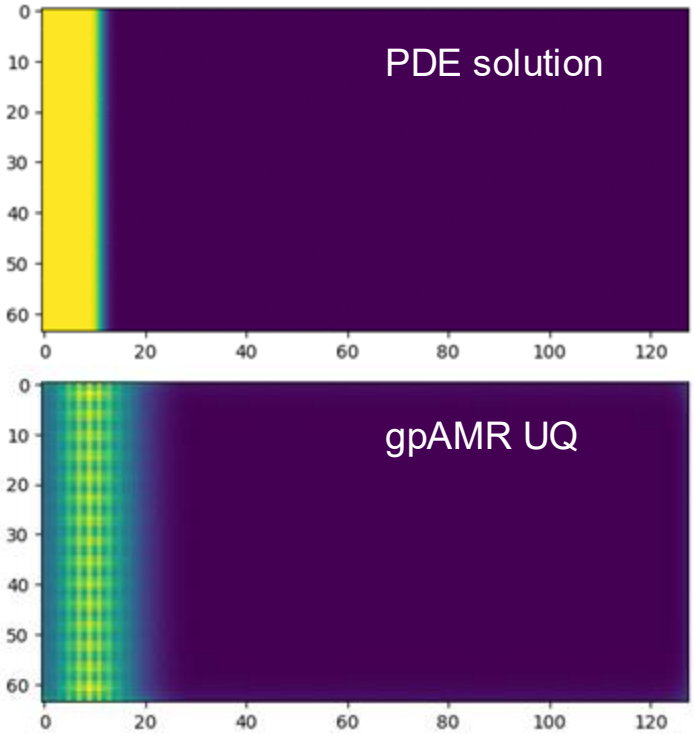
$$\beta(\mathbf{x}_i, \mathbf{x}; w) = \exp\left[-||\mathbf{x}_i - \mathbf{x}||\, w^2\right]$$

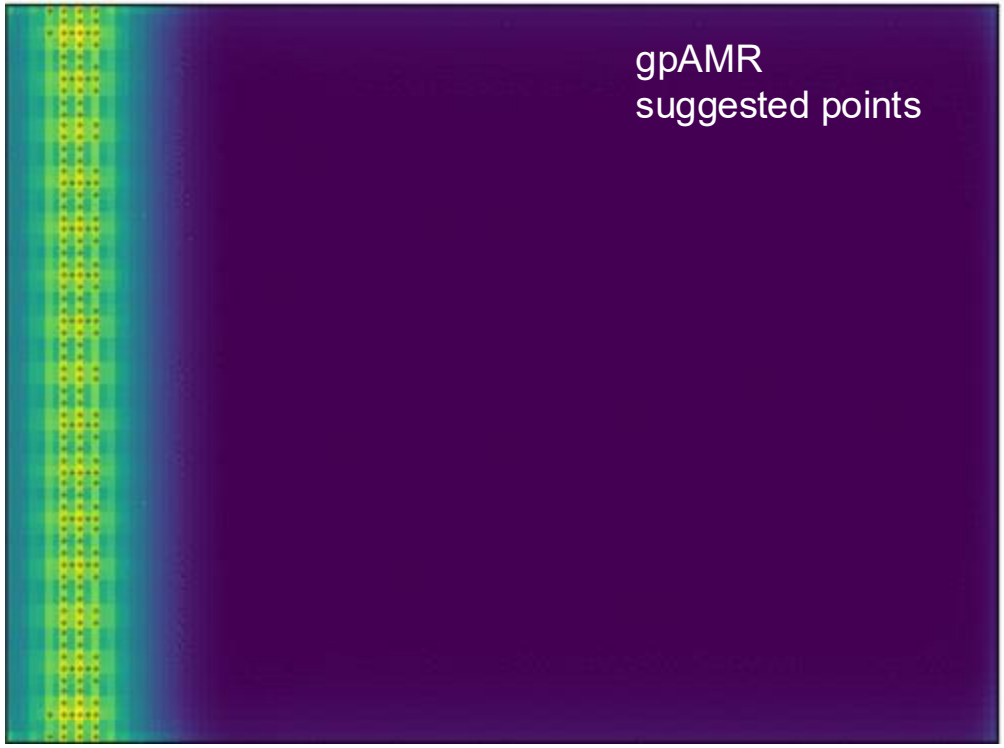
BERKELEY LAB

gpAMR demonstration for linear advection

Integrated **gpCAM** with the PDE-based simulation code **Chombo**

AMR Benchmark: 1D Linear Advection





gpAMR demonstration for 2D flow past a cylinder

AMR Benchmark: 2D Turbulent Flow Past a Cylinder

Summary of key research areas in GP

- Kernel design: how to represent covariance
  - Stationary and non-stationary
  - Input uncertainty with prob. distribution: e.g., Wasserstein distance
  - Physics-informed, PDE-aware
  - Fast linear algebra kernels to enable large scale GP with millions of data points
- Higher dimensional problems (e.g., D > 20)
- Non-smooth surface
  - Clustering GP: partition into piecewise smooth regions
- Deep GP: combine GP with neural networks
- Functional or tensor output
- Sparse GP: model approximations to speed up optimization

Acknowledgement

Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research's Applied Mathematics program under Contract No. AC02-05CH11231.

BERKELEY LAB

THANK YOU