An automated Bayesian computational optimization approach for designing antimicrobial nanosurfaces

Hanfeng Zhai

Cornell University

www.hanfengzhai.net

February 15, 2022

Background

- "*..., the number of people under the age of 5 will be outnumbered by those over 65. The World Health Organization projects that rapid global aging will severely exacerbate the burden of noncommunicable diseases for populations who can least afford treatments.* - JJ
- **Market!** Pfizer: \$154.229B; Moderna: \$7.336B; J&J: \$170.693B[1]
- Bacteria attachment (leads to generation of **biofilm**) is one of the major cause of disease and contamination during treatment in biomedical devices...! (everyone wants to live healthier and longer)
- Experimental investigations are **expensive**! We need efficiency, trustworthy **materials design at low cost**.
- Materials Genome Initiative[2]; Digital Twins[3]; Materials 4.0[4]; ... and, of course, **Machine Learning!**[5]

[1] all from Wikipedia, BTW the giant **Google LLC** worth \$420B!

[2] `https://www.mgi.gov/`

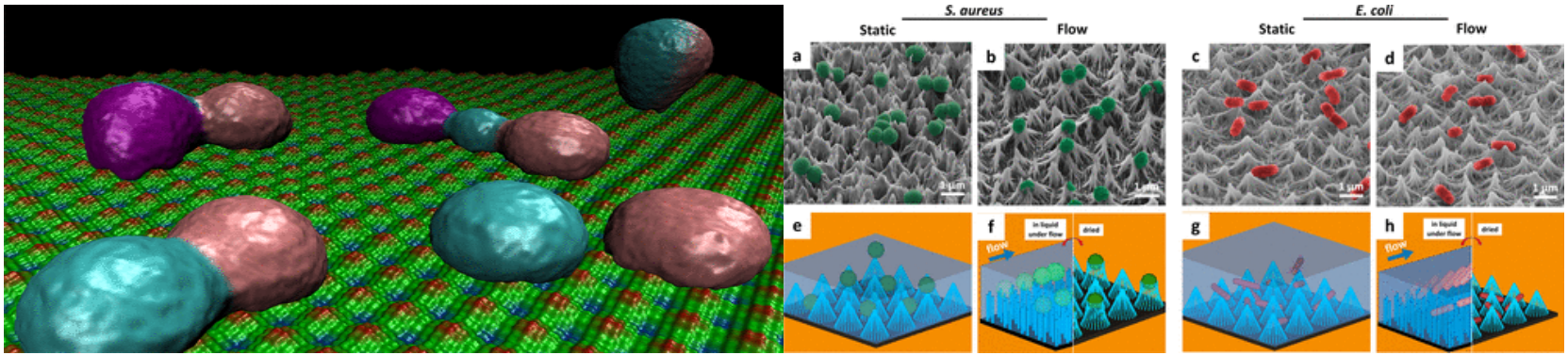
[3] `https://www.ibm.com/topics/what-is-a-digital-twin`

[4] Jose & Ramakrishna, *Appl. Mat. Today*, 2018

[5] Everyone knows it!!

Goal

- Unveil the mechanism of antimicrobial property of super-hydrophobic surfaces through numerical simulation (DEM). FEW people did it!
- Design the nanosurfaces (coatings) with simulation + ML algorithms, more specifically, **Bayesian optimization**[6].
- Prove the results to be trustworthy. Automate the process to make it more universally applicable.

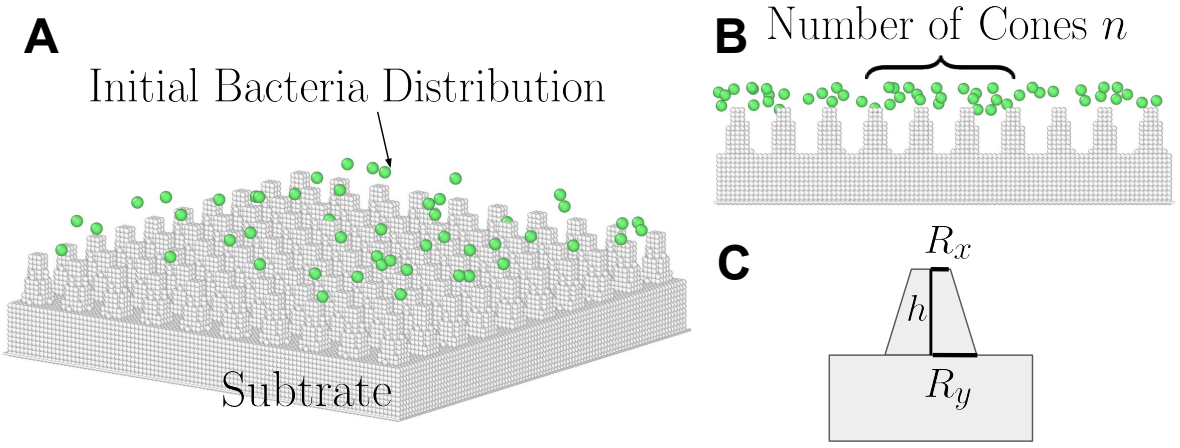


Zhang et al., *Langmuir*, 2019; Hizal et al., *ACS Appl. Mater. Interfaces*, 2017.

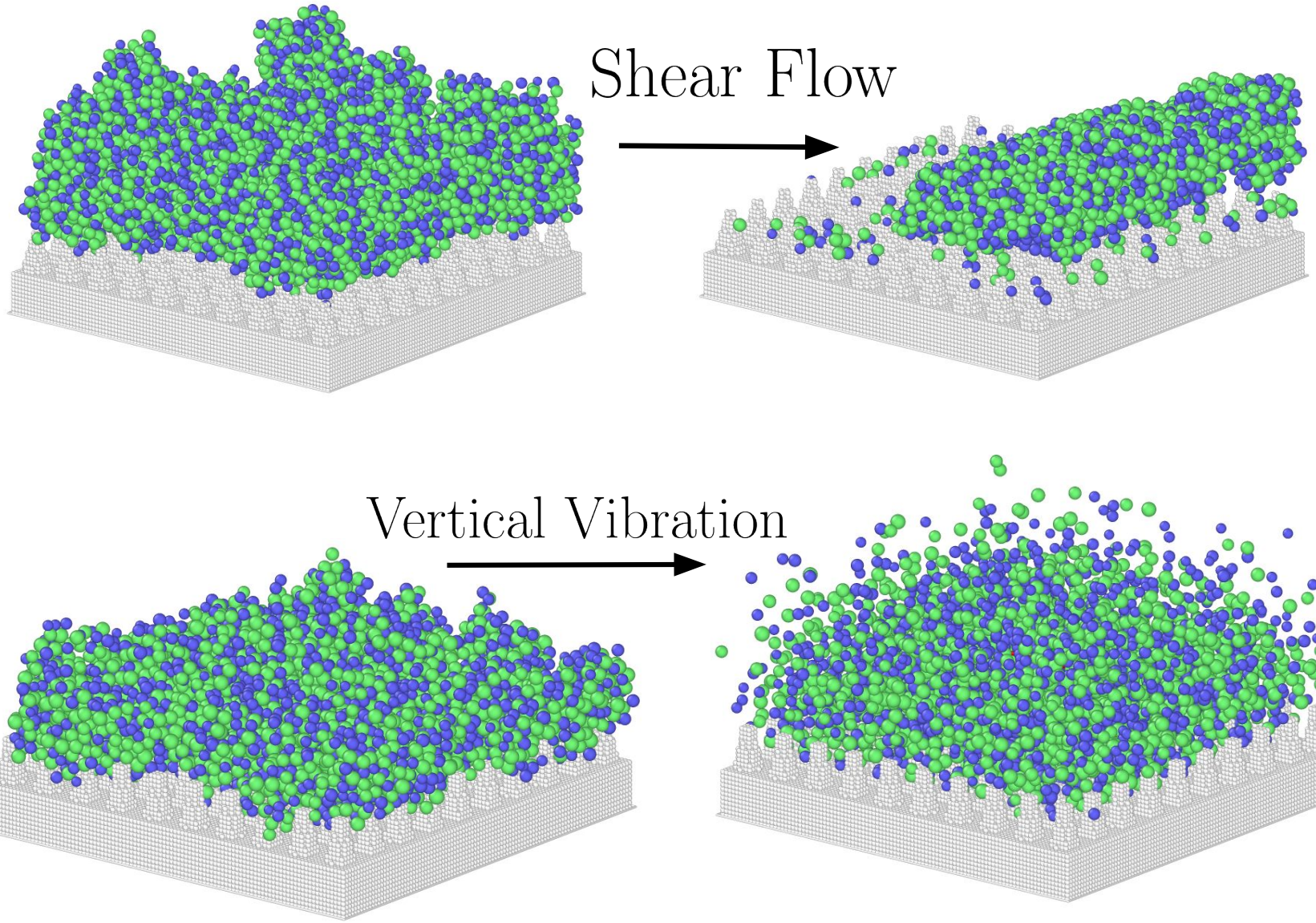
[6] widely used, heuristics, Bayesian statics

Simulation setup

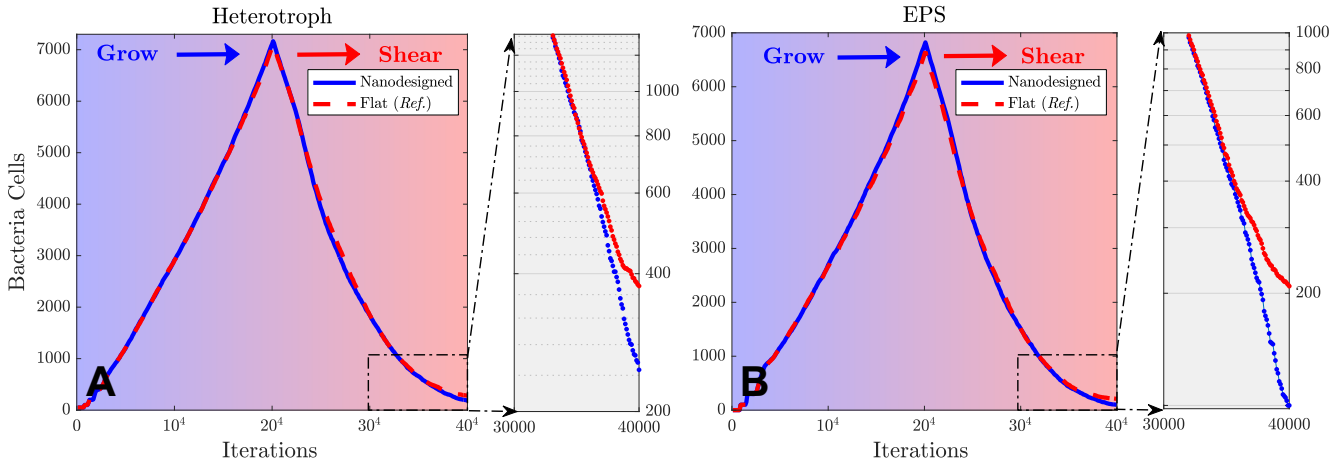
- **Design variables**: R_x, R_y, h, n
- **Simulation box**: Geometry: 4×10^{-3}m for x, y, z; *Boundary conditions*: fixed BCs on x, y, z; *Initial bacteria area*: 2×10^{-6}m
- **Bacteria cells**: *Initial No.*: 50; *Growth rate*: 0.00028; *Yield*: HET: 0.61 & EPS: 0.18. Monod growth model. HET: $K_s = 3.5 \times 10^{-5}$
- **Material properties**: *Heterotrophs*: $\rho = 150$; $d = 10^{-6}$; $m = 7.854 \times 10^{-17}$; *Substrate*: $\rho = 4410$; $d = 5^{-7}$; $m = 9.1875 \times 10^{-17}$.
- **Biofilm simulation**: *Grow*: $2 \times 10^4 (\times 10)$; *Shear*: $2 \times 10^4 (\times 2.5)$; *Vibration*: $1 \times 10^3 (\times 10^3)$;

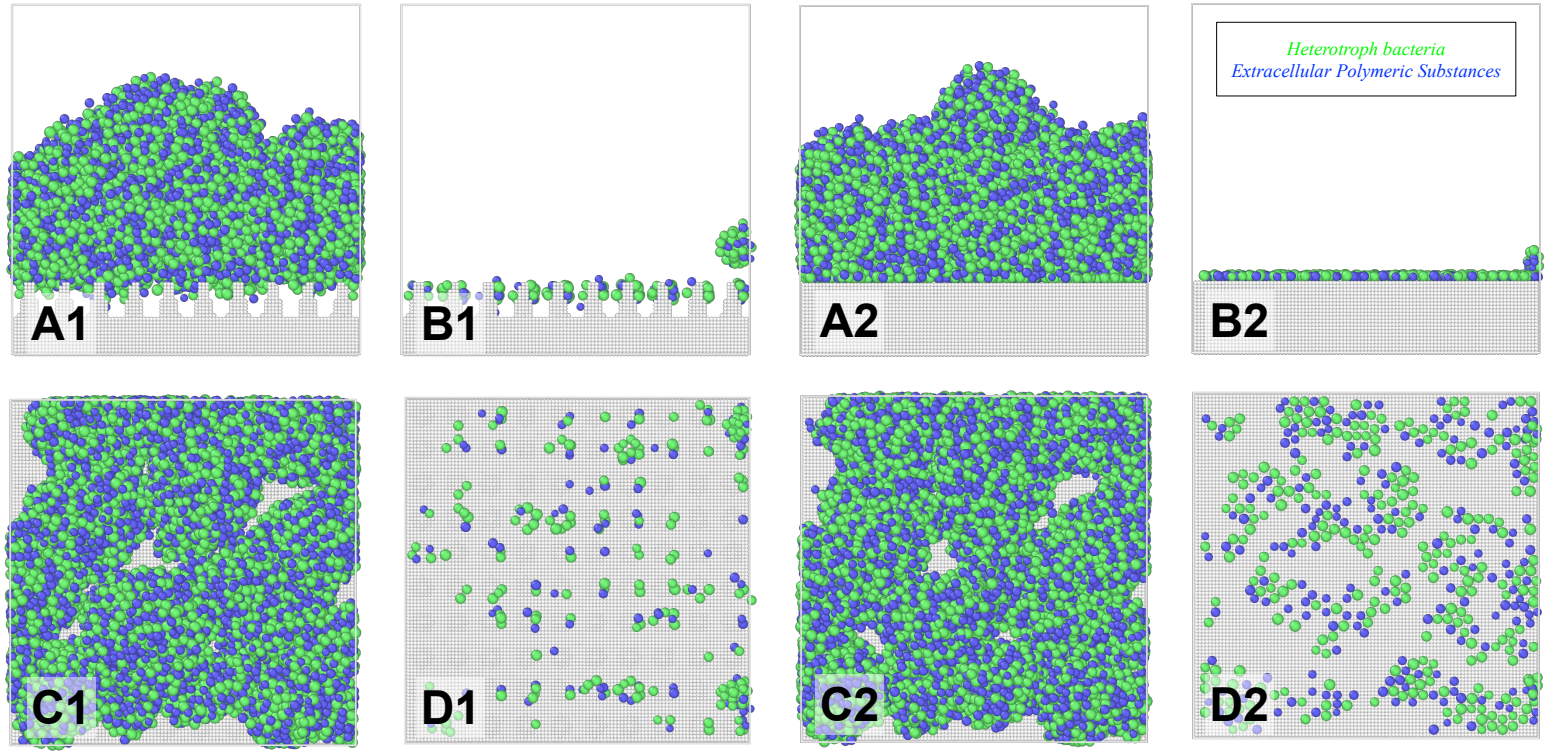


Biofilm removal: shear and vibration



Important question: why this kind of surface works at all?





Optimization algorithm

Problem definition

Suppose that the simulation is denoted by $\mathcal{S}$, the total numbers of bacteria cells are denoted by $\mathcal{M}$; by changing R_x, R_y, h & n we aim to

$$R_x^*, R_y^*, h^*, n^* := \underset{R_x, R_y, h, n \in \mathcal{X}}{\arg\min} \; \mathcal{M}(\mathcal{S}(R_x, R_y, h, n))$$

where $R_x, R_y, h, n \in \mathcal{X}_B$ (bounded design space); with $\mathcal{S}$ means we let the biofilm grow and sheared off for 20k steps in NUFEB, respectively.

Gaussian process regression

Suppose a mapping from $R_x, R_y, h, n \xrightarrow{\mathcal{S}} \mathcal{M}$, a Gaussian process ($\mathcal{GP}$) fit the mapping based on probability distribution:

$$\mathcal{M} \sim \mathcal{N}(\mathcal{S}(R_x, R_y, h, n), \nu)$$

where we define an acquisition function: $a : \mathcal{X} \to \mathbb{R}^+$, TBC...

Optimization algorithm

Gaussian process priors

[a] The next point is found via a proxy optimization:

$$\{R_x, R_y, h, n\}_{next} = \arg\max_{R_x, R_y, h, n} a(R_x, R_y, h, n)$$

where the acquisition function can be simpilified denoted as

$$a(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta)$$

which can be chosen in several different ways. Under GP prior, a is dependent on predictive mean function $\mu(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)$ and predictive variance function $\sigma^2(\mathbf{x}; \{\mathbf{x}_n, \mathcal{X}\}, \theta)$. We denote the current best value as $\mathbf{x}_{best} = \arg\min_{\mathbf{x}_n} \mathcal{S}(\mathbf{x}_n)$; the cumulative distribution function of the standard normal: $\Phi(\cdot)$.

[a] Snoek et al., *NIPS*, 2012

Optimization algorithm: Acquisition function

- **Probability of improvement**
  One of the most intuitive strategy is to maximize the probability of the improving over the best current value, which can be written under $\mathcal{GP}$[7]:

$$a(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta) = \Phi(\gamma(\mathbf{x})), \ \gamma(\mathbf{x}) = \frac{\mathcal{S}(\mathbf{x}_{best} - \mu(\mathbf{x}; \{\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta\}))}{\sigma(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta)}$$

- **Expected improvement**
  One could max the EI over the current best, in the form of $\mathcal{GP}$:

$$a(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta) = \sigma(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta)(\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}); 0, 1))$$

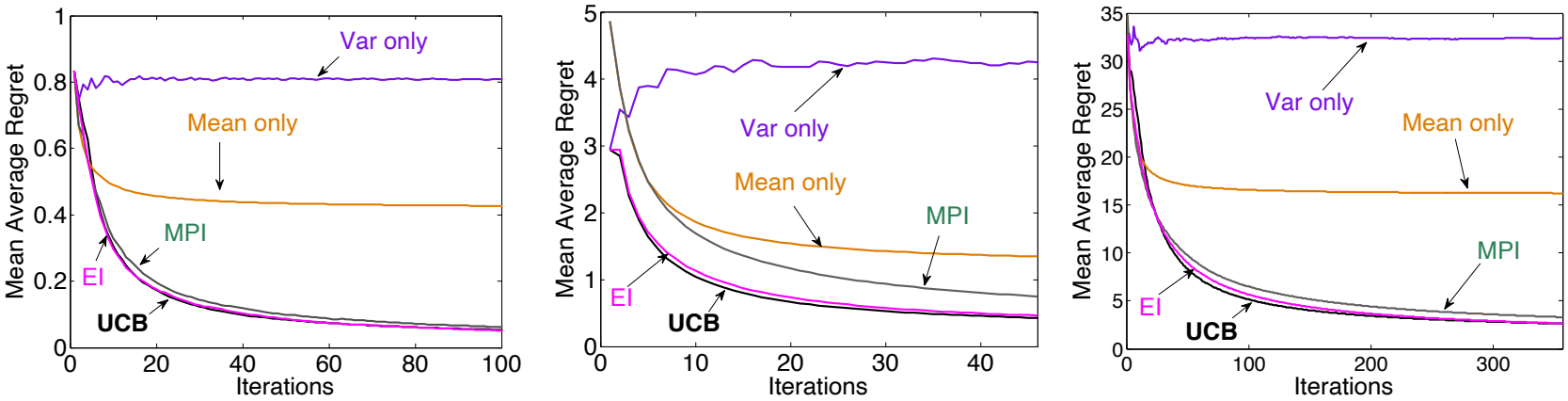
- **Upper confidence bound**
  One could exploit the lower confidence bounds to construct acquisition functions that minimize regret over optimization[8]:

$$a(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta) = \mu(\mathbf{x}; \{\mathbf{x}_n, y_n\},) - \kappa\sigma(\mathbf{x}; \{\mathbf{x}_n, \mathcal{M}\}, \theta)$$

[7] Kushner, *J. Bas. Eng.*, 1964
[8] Srinivas et al., *ICML*, 2010

Acquisition function: Why?

One thing the reviewer is very likely to ask is, Why "UCB"???



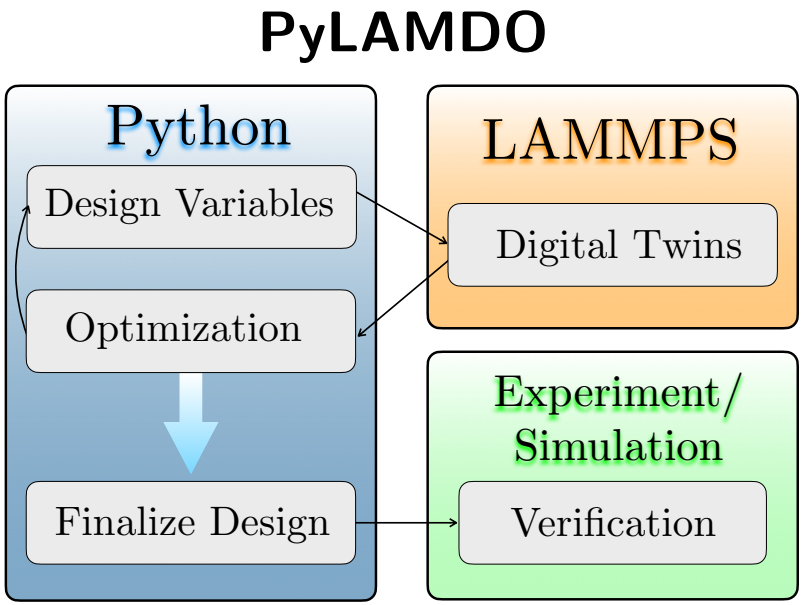
(a) *Squared exponential* (b) *Temperature data* (c) *Traffic data*

Figure 4. Comparison of performance: GP-UCB and various heuristics on synthetic (a), and sensor network data (b, c).

Srinivas et al., *ICML*, 2010, contends:

... For temperature data, the GP-UCB algorithm and EI heuristic clearly outperform the others, and do not exhibit significant difference between each other. On synthetic and traffic data MPI does equally well. In summary, GP-UCB performs at least on par with the existing approaches which are not equipped with regret bounds.

Algorithm implementation



Python-LAMMPS interface for Multiscale, Multiphysics, Materials Design & Optimization.

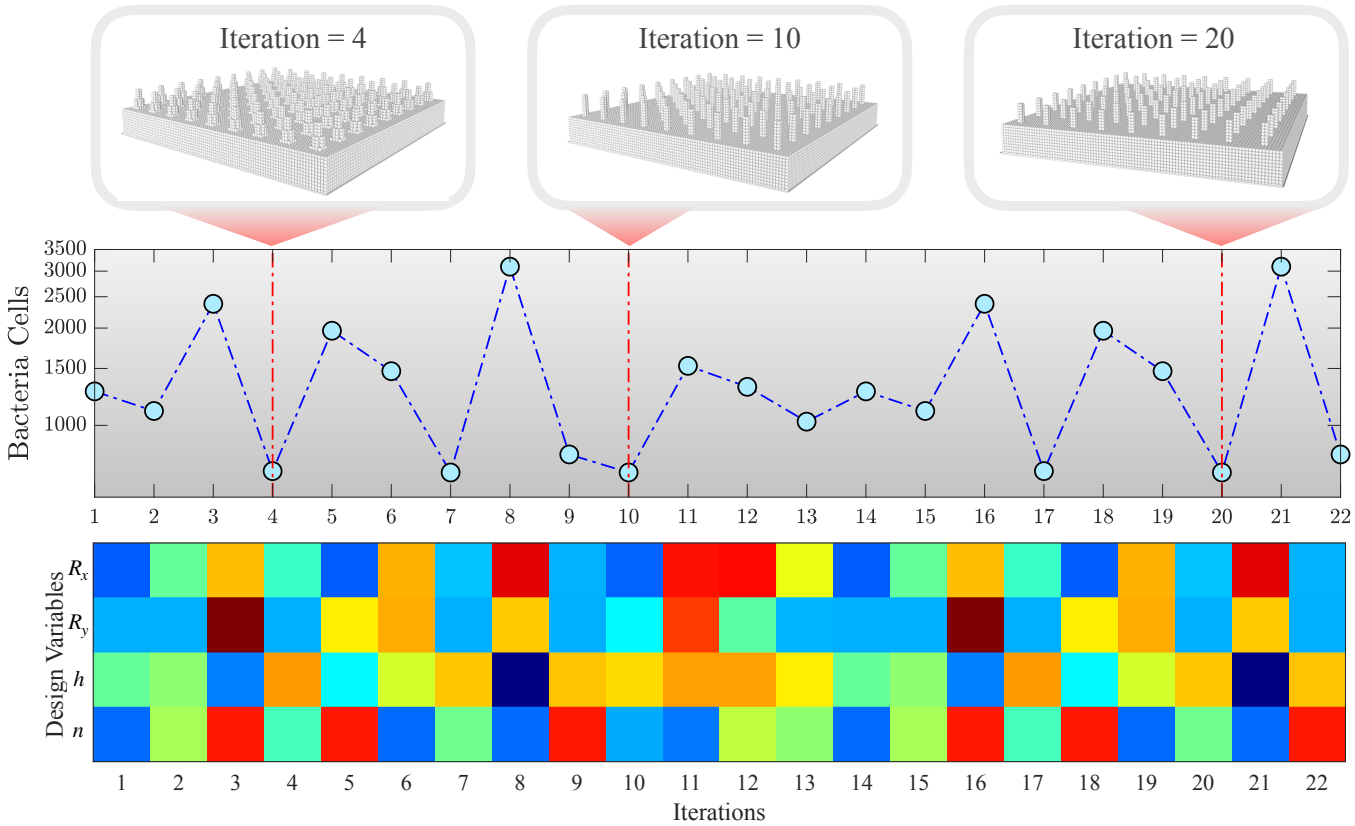
```
pbounds = 'R_x':  (0.15,
0.95), 'R_y':  (0.15, 0.95),
'h':  (12, 20), 'n':  (5.0,
10.9) ... utility =
UtilityFunction(kind="ucb",
kappa=2.5, xi=1)  ...
opt = BayesianOptimization(
f=biofilm_shear,
pbounds=pbounds, verbose=2,
random_state=1, ) ...
opt.maximize( init_points=10,
n_iter=90, alpha=1e-3, )
```

Stampede Implementation

50 nodes, 100 cores per node, time limit 48 hrs, Python 3.9.6; intel 19.1.1

Results

Example attempt: simple biofilm shear case



Ongoing problems

- Setting bounds for cones' upper and lower radius.
- Tuning parameters for vibration simulation, i.e., frequency & magnitude.
- (Technical) Stampede nodes are down / drained ...

The End

Any Questions...?