

Surrogate Optimization of HPC Applications

Juliane Mueller

Center for Computational Sciences and Engineering

JulianeMueller@lbl.gov

<https://optimization.lbl.gov/>

Summer 2020

The takeaways from today's talk

1. A general understanding of what optimization is all about
2. An understanding of the importance of choosing the right tools (solvers) for your optimization problem
3. Optimization is needed in pretty much all science domains

Before we talk about black-box optimization...

What is optimization?

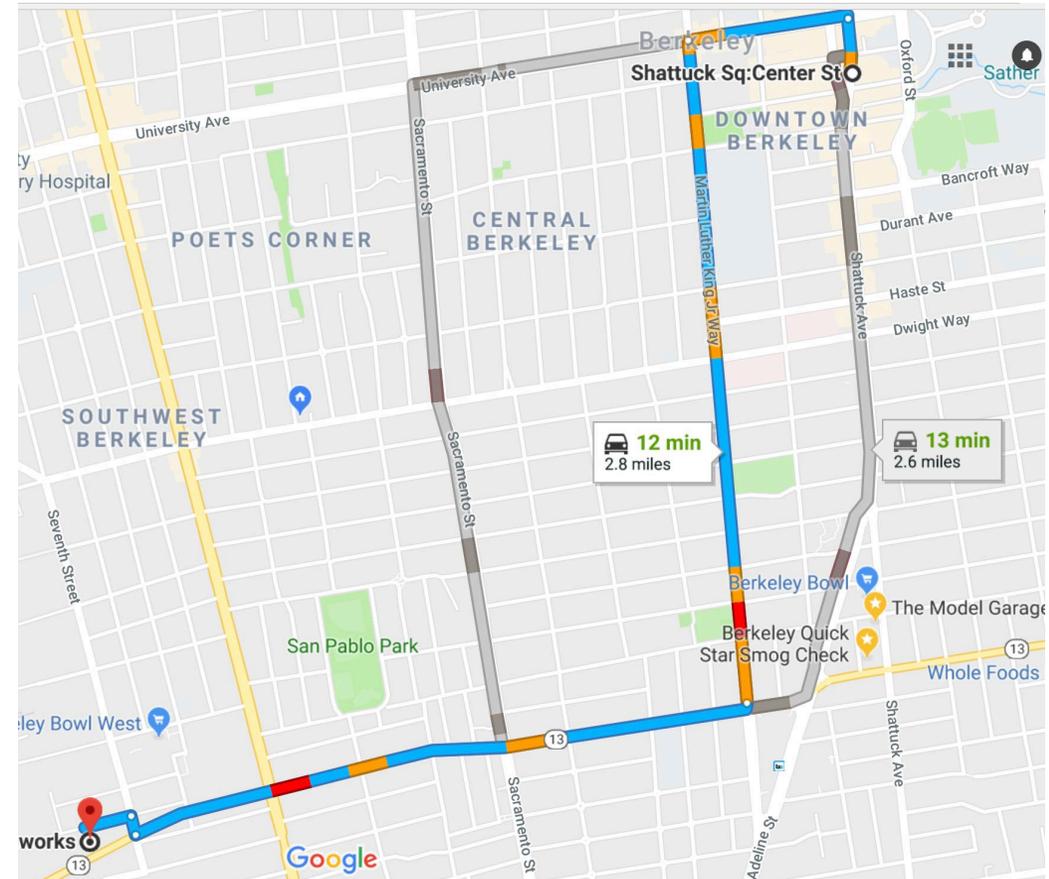
- We encounter optimization whenever we want to find the best of something, e.g.,
 - maximize the *Nonaka Metric* (calories per dollar spent)



Before we talk about *black-box* optimization...

What is optimization?

- We encounter optimization whenever we want to improve something, e.g.,
 - Get from A to B in the **fastest** way possible

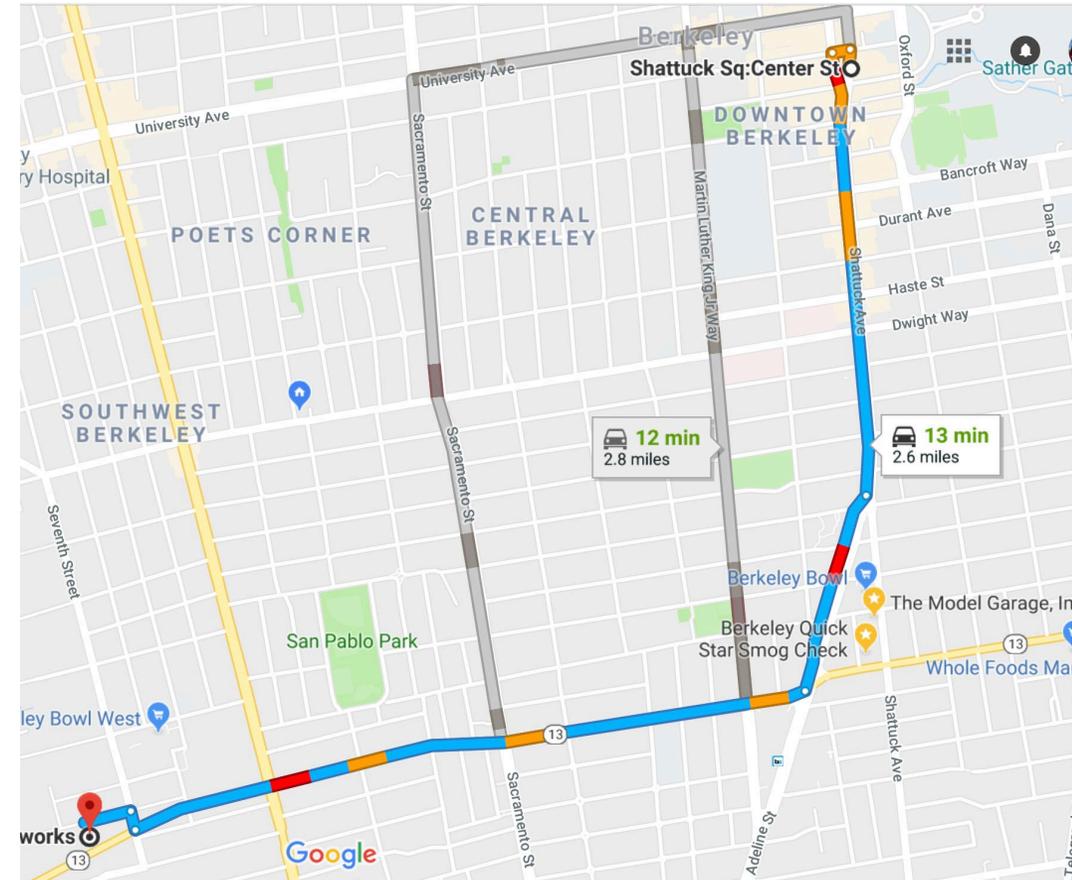


Before we talk about *black-box* optimization...

What is optimization?

- We encounter optimization whenever we want to improve something, e.g.,
 - Get from A to B in the **fastest** way possible
 - Get from A to B in the **shortest** way possible

Find the “best” of something



Before we talk about *black-box* optimization...

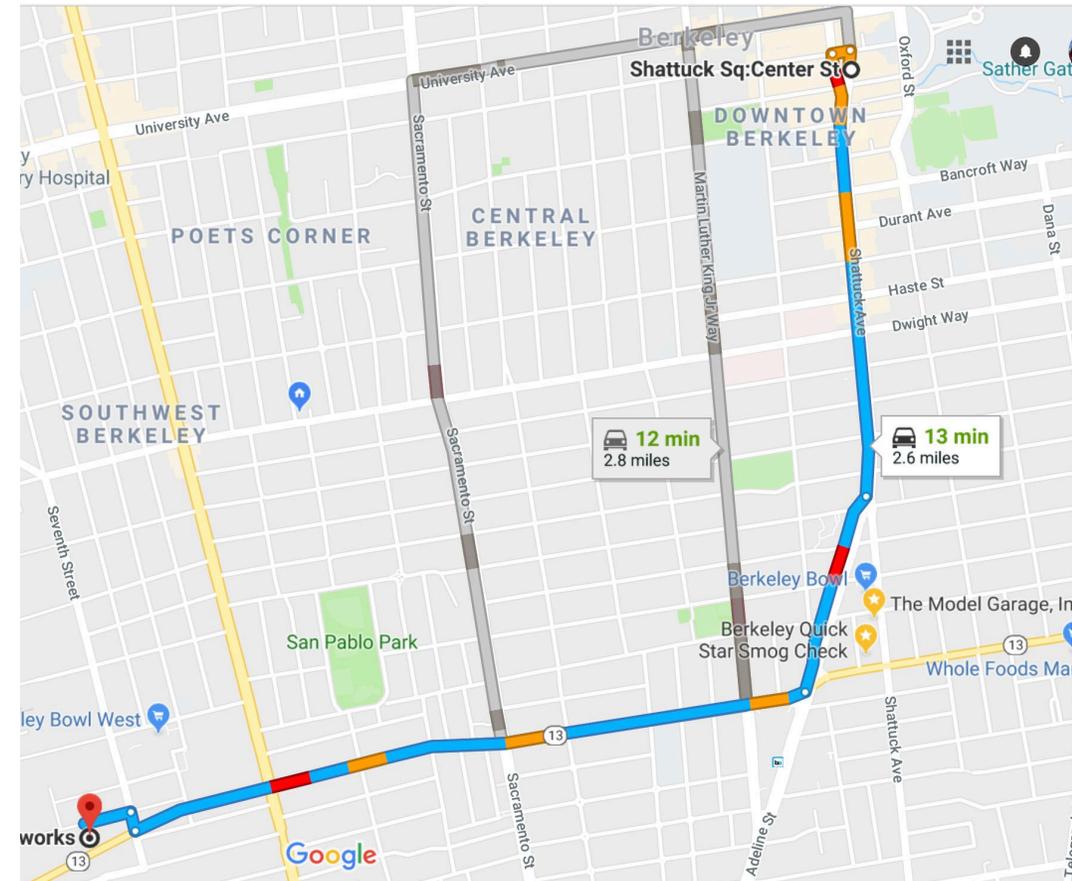
What is optimization?

- We encounter optimization whenever we want to improve something, e.g.,
 - Get from A to B in the **fastest** way possible
 - Get from A to B in the **shortest** way possible

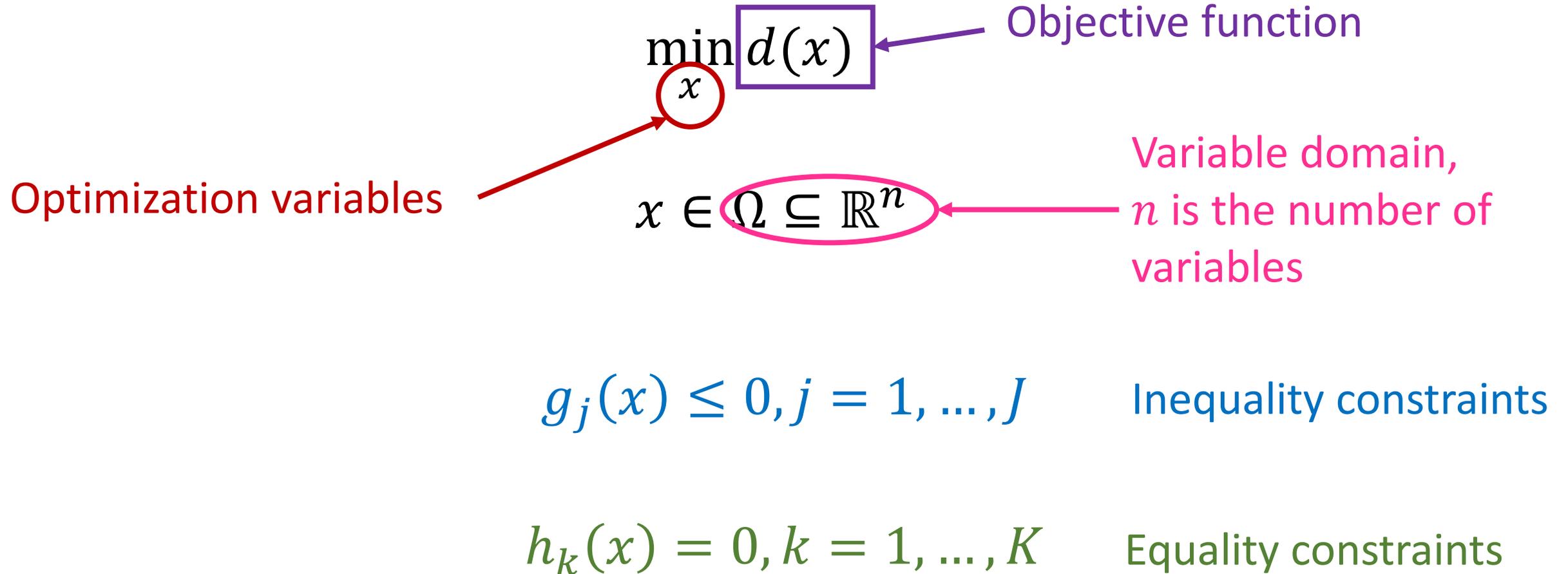
Formulate it as an optimization problem:

- Let d denote the distance between A and B
- d depends on which roads (x) you take, e.g.,
 - $x_{univ.ave.} = 0$ if we do not go University Ave.,
 - $x_{univ.ave.} = 1$ if we do go University Ave
- Find the values for x such that

$$\min_x d(x)$$



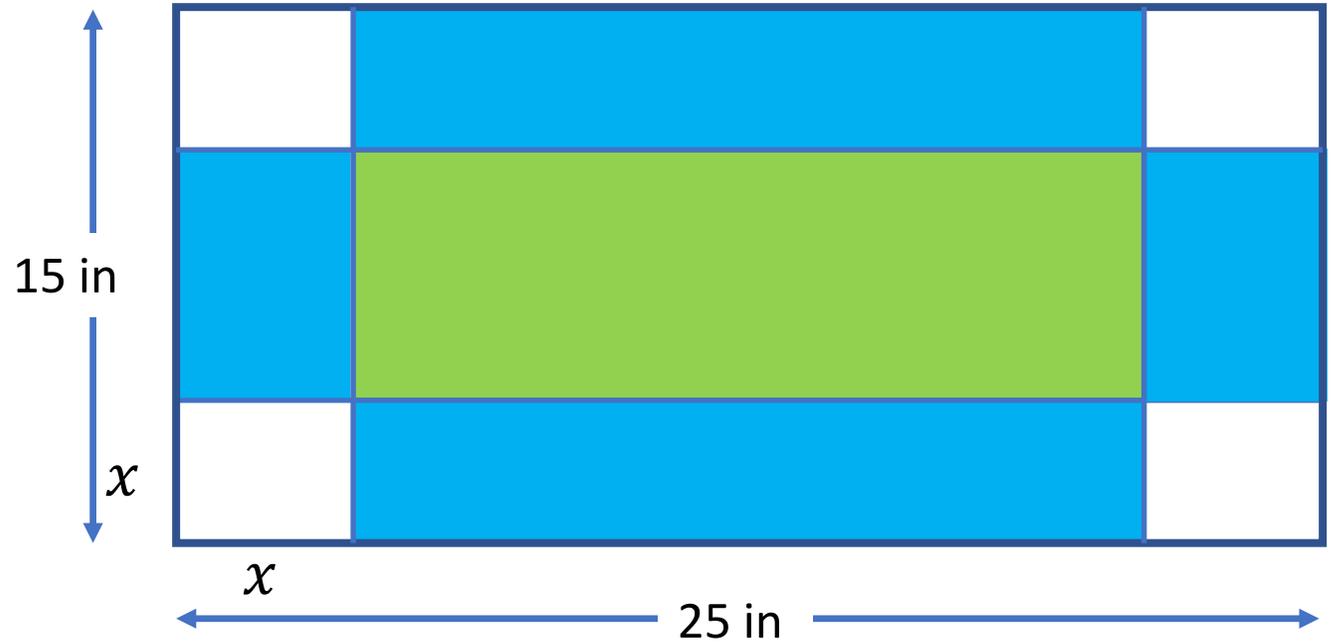
Some terminology...



A simple example

Maximize the volume of a box by cutting away squares from each corner of a 25x15 inch rectangle and folding up the sides

- x is the optimization variable
- Objective function:
Volume = width * length * height:
$$V(x) = (15 - 2x)(25 - 2x)x$$
- Our constraints:
 - $x \geq 0$
 - $15 - 2x \geq 0 \rightarrow x \leq 7.5$
 - $25 - 2x \geq 0 \rightarrow x \leq 12.5$

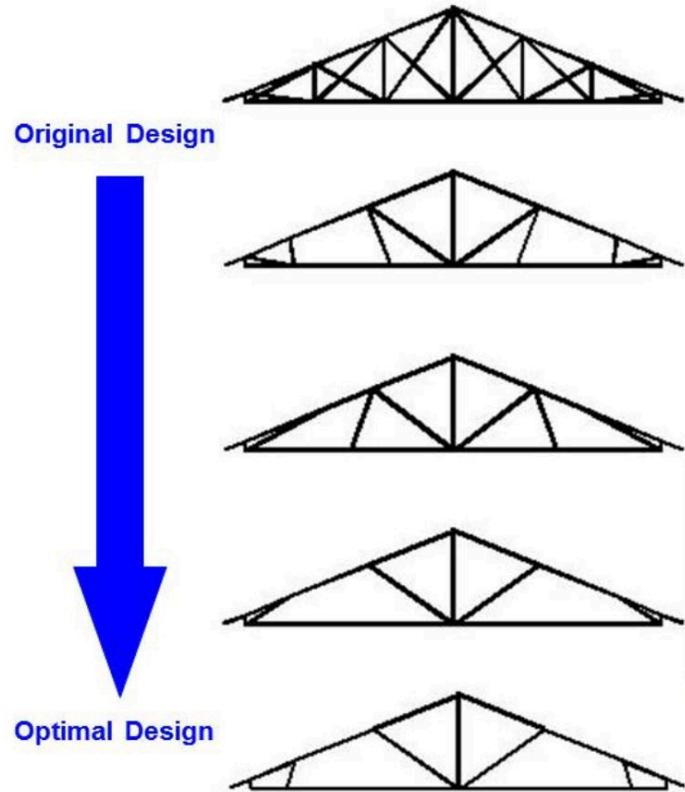


Our optimization problem is then:

$$\begin{aligned} \max V(x) \\ 0 \leq x \leq 7.5 \end{aligned}$$

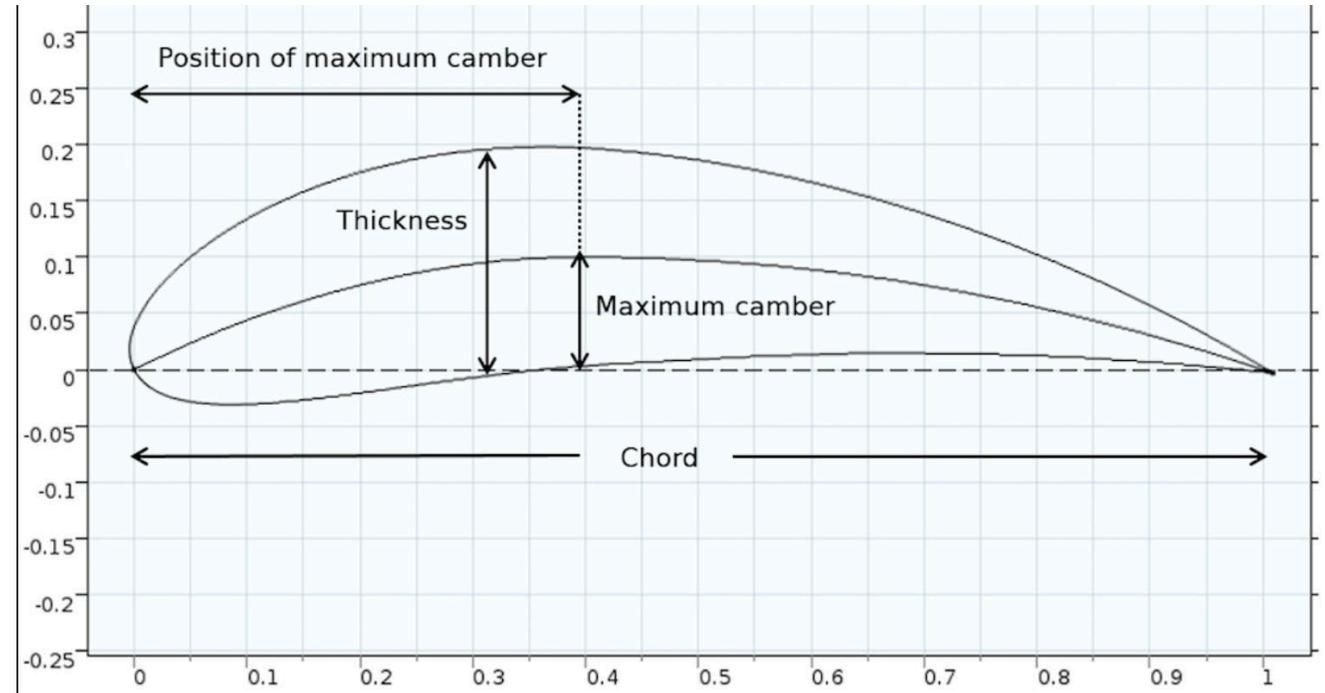
Solve it by computing the derivative of $V(x)$, setting it to 0 and finding x

Optimization problems arise practically everywhere



Structural optimization:

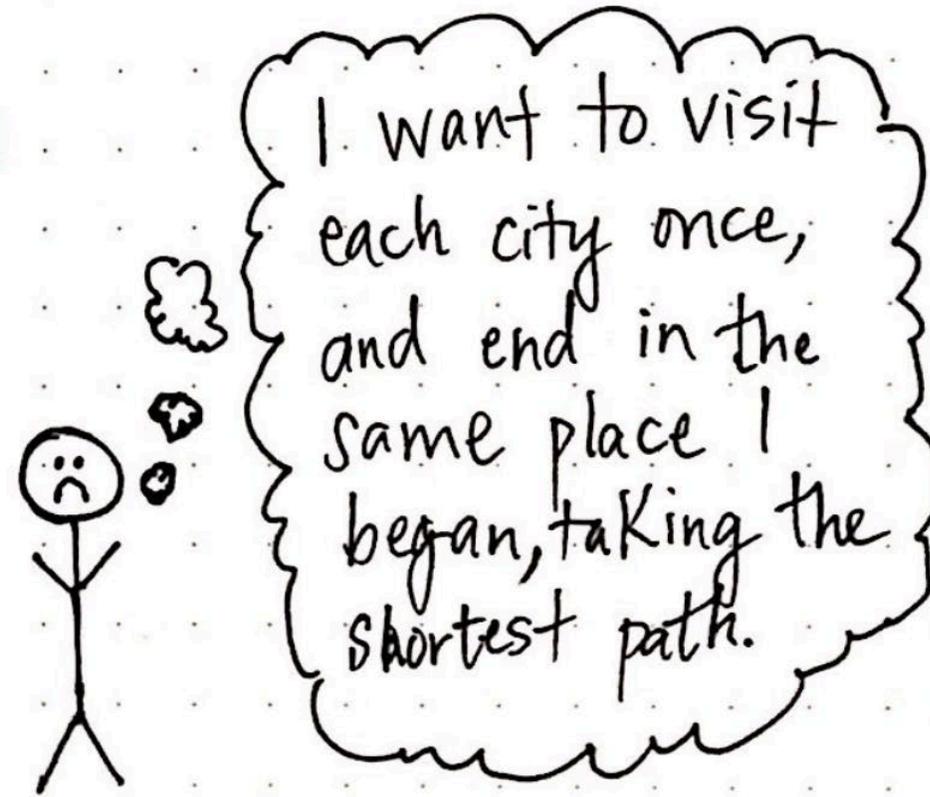
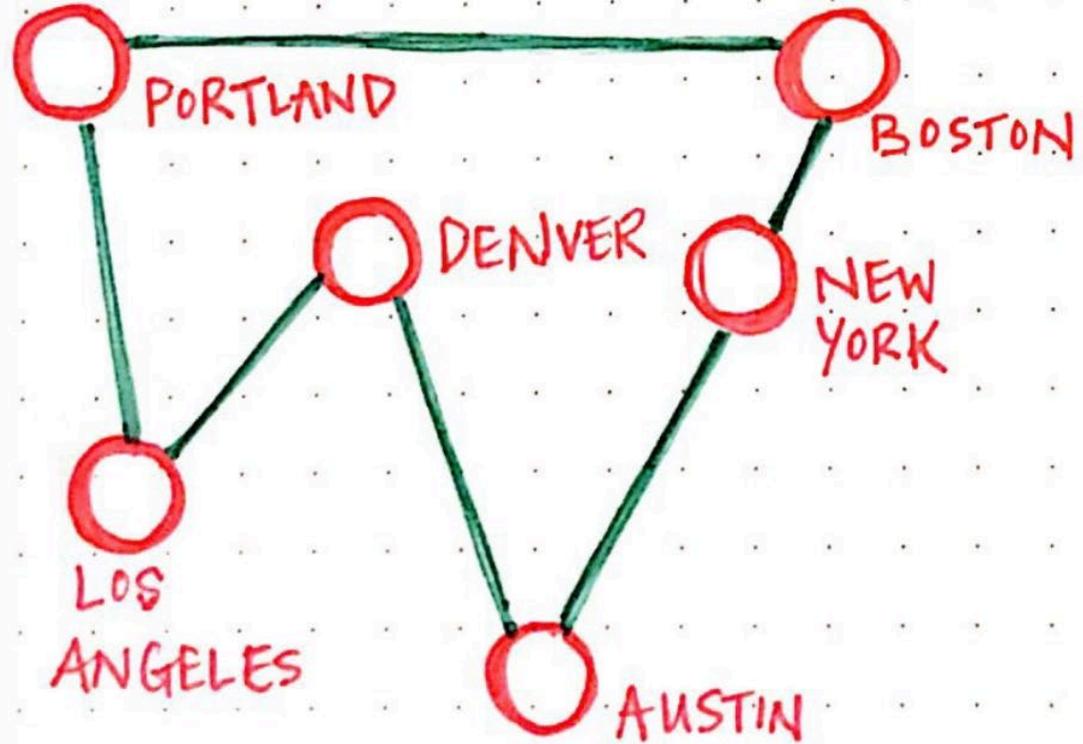
- Objective: minimize weight
- Variables: geometry
- Constraint: displacement under load



Airfoil design:

- Objective: maximize lift, minimize drag
- Variables: geometry, see figure

Optimization problems arise practically everywhere



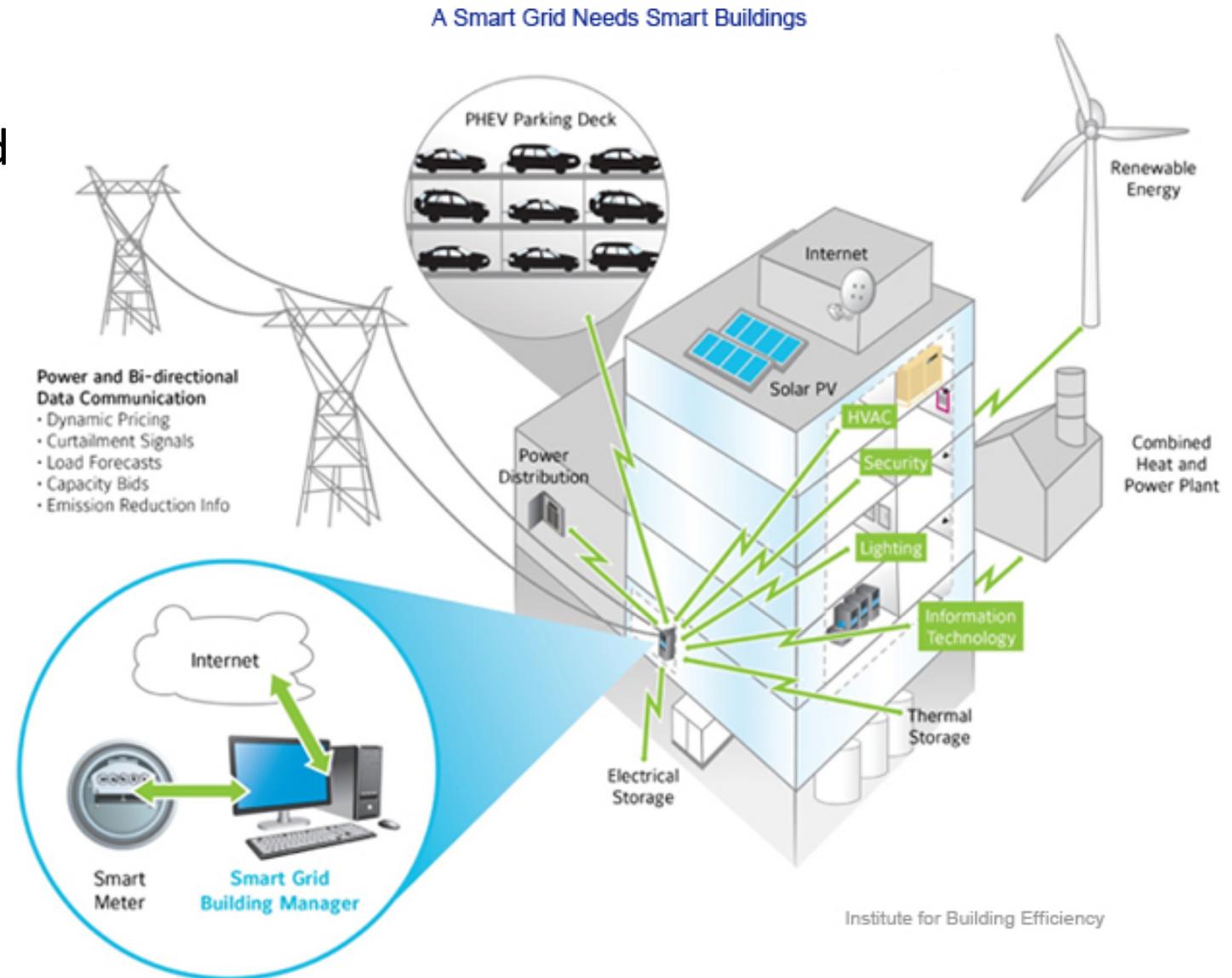
Traveling salesman problem:

- Objective: minimize the distance traveled
- Variables: which links to traverse
- Constraints: each city must be visited, finish where you started

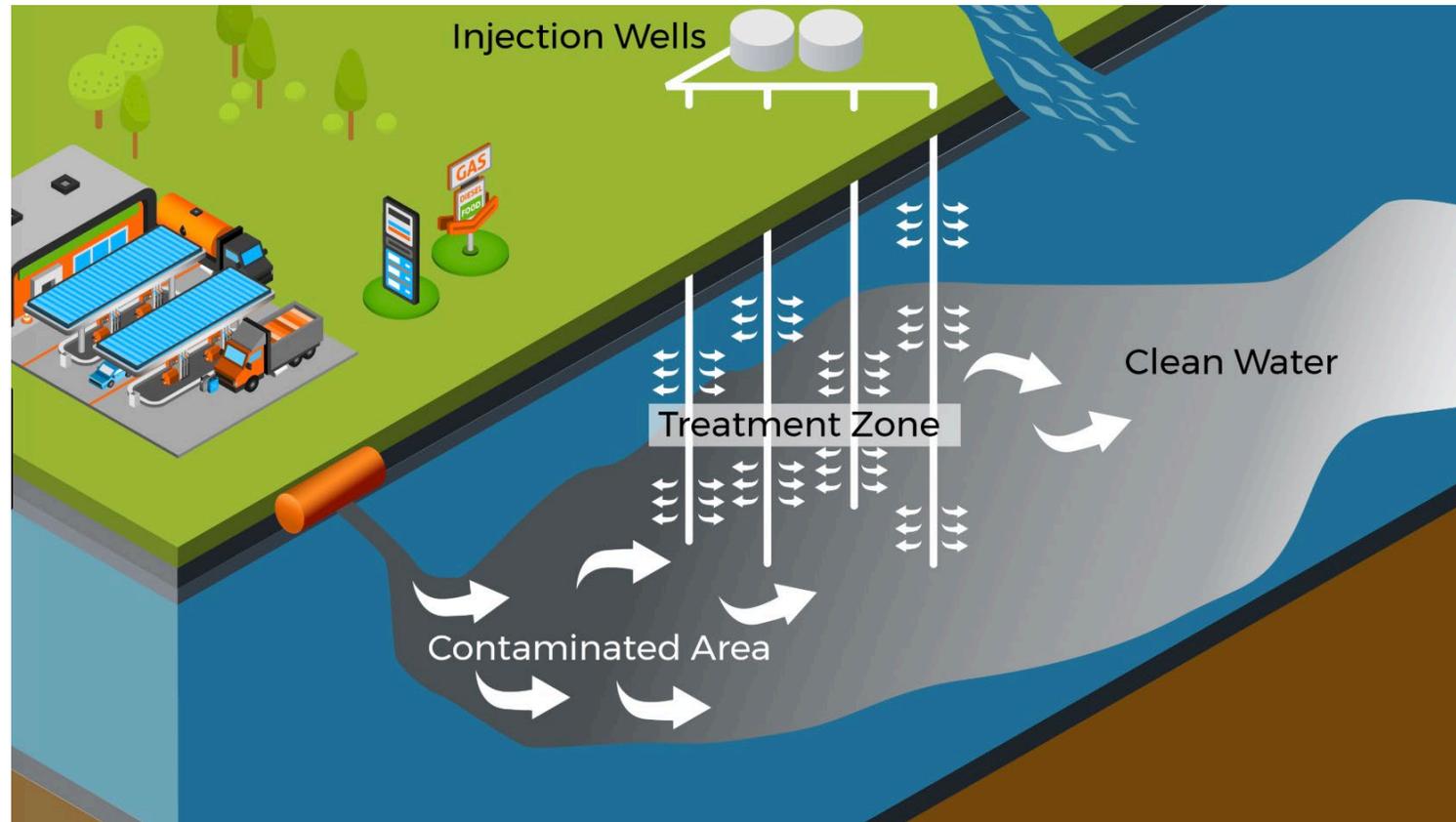
Optimization problems arise practically everywhere

Design of smart buildings:

- Objective: minimize energy demand
- Variables: building specifics (wall thickness, heating/cooling schedules)
- Constraints: energy use intensity



Optimization problems arise practically everywhere



Groundwater remediation:

- Objective: minimize cleanup cost
- Variables: the number and location of treatment wells, pumping strategy
- Constraints: remaining contaminant

Optimization problems come in many different flavors

We differentiate optimization problems based on their characteristics

- What type of variables do we have?
 - Continuous – integer – mixed-integer – binary – categorical
- What kind of objective function do we have?
 - Differentiable – analytic – simulation – convex – multimodal
- How many objective functions?
 - One
 - Several (multi-objective optimization)
- What kind of constraints do we have?
 - Equality – inequality – bound – simulation

Characteristics determine which solver to use

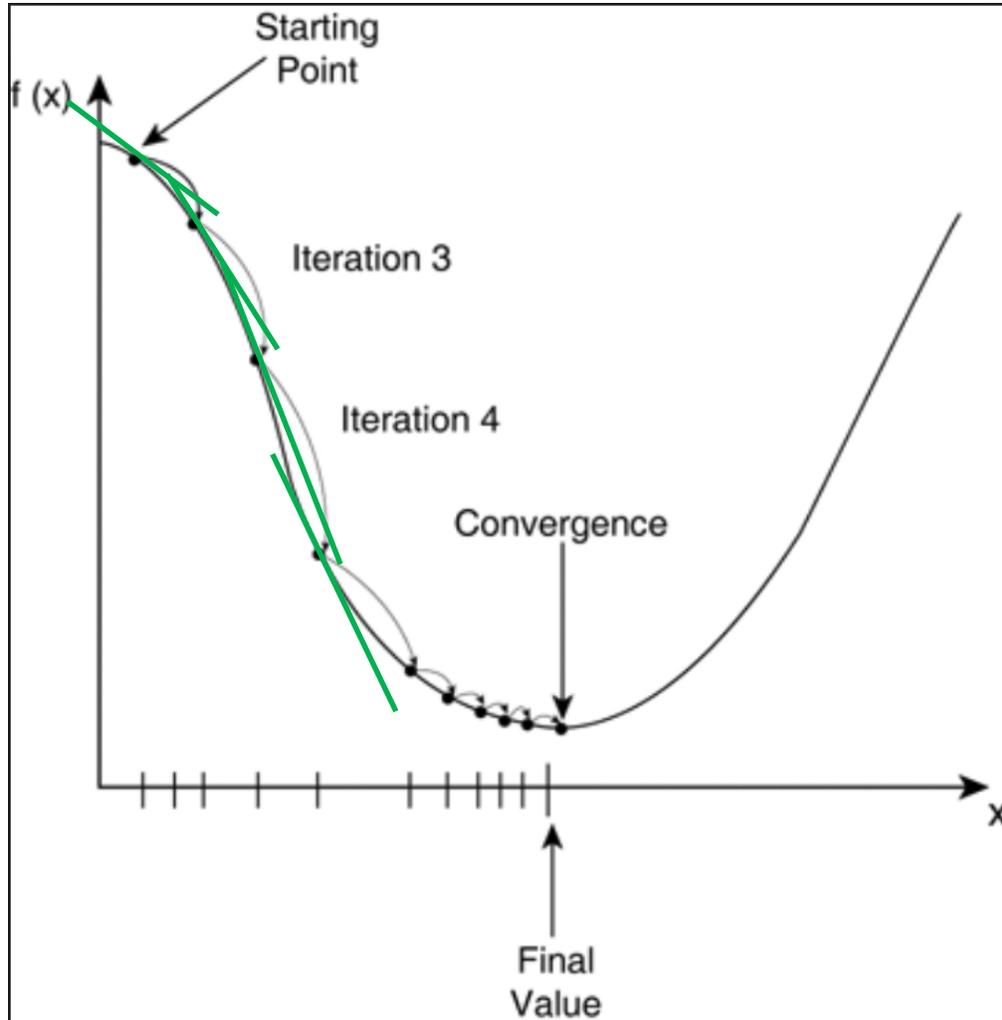
Optimization methods are developed for specific types of problems, for example:

- *Sequential quadratic programming* – for constrained nonlinear optimization
- *Simplex algorithm* – for linear programs
- *Steepest descent/gradient descent* – for nonlinear problems with derivatives
- *Heuristics* – for problems without derivative information (Tabu search, simulated annealing, genetic algorithms)
- *Surrogate model algorithms* – for problems that involve evaluating an expensive simulation

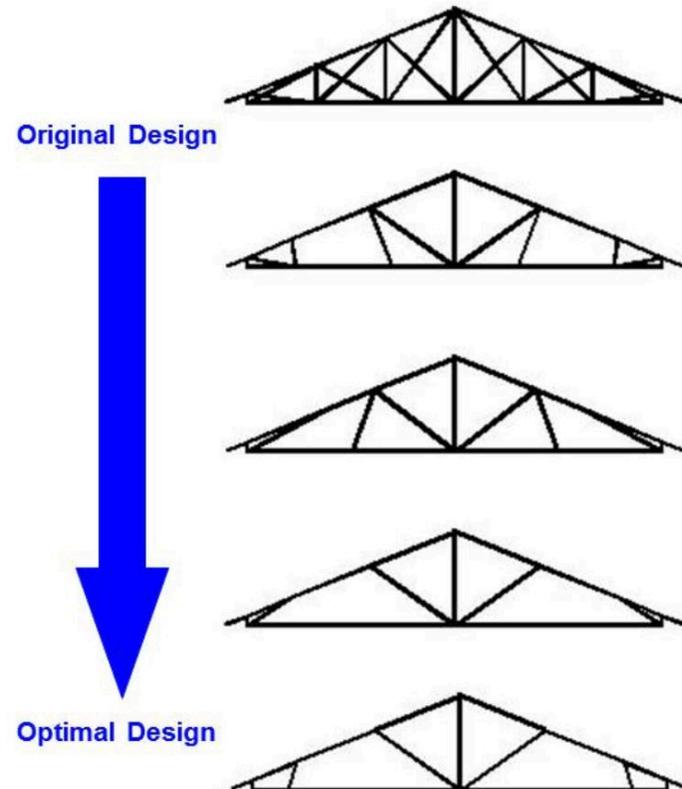
Selection of the optimization method requires a good understanding of the problem at hand

Optimization algorithms are iterative methods

Gradient based methods go downhill (minimization):



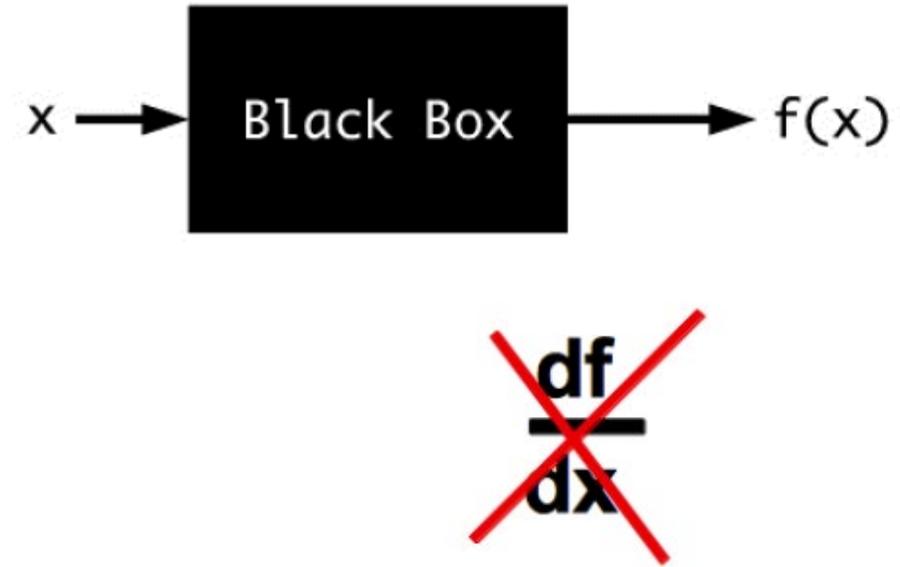
Repeatedly evaluate $f(x)$ to find the minimum



What's so special about the problems I work on?

The problems I work on have some nasty characteristics:

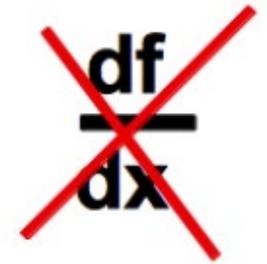
- The objective function involves a computer simulation
 - No analytic description of the objective function (black-box)
 - No gradient information



What's so special about the problems I work on?

The problems I work on have some nasty characteristics:

- The objective function involves a computer simulation
 - No analytic description of the objective function (black-box)
 - No gradient information
- The simulation is time consuming

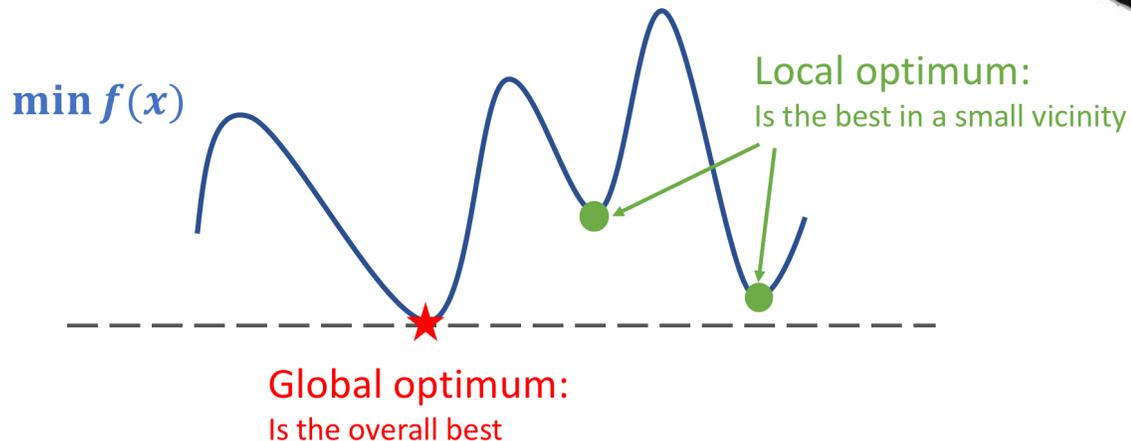
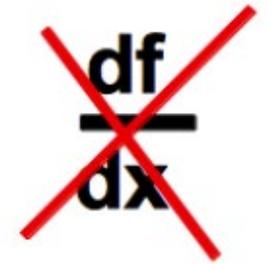


- We cannot do thousands of evaluations of $f(x)$
- We cannot approximate gradients

What's so special about the problems I work on?

The problems I work on have some nasty characteristics:

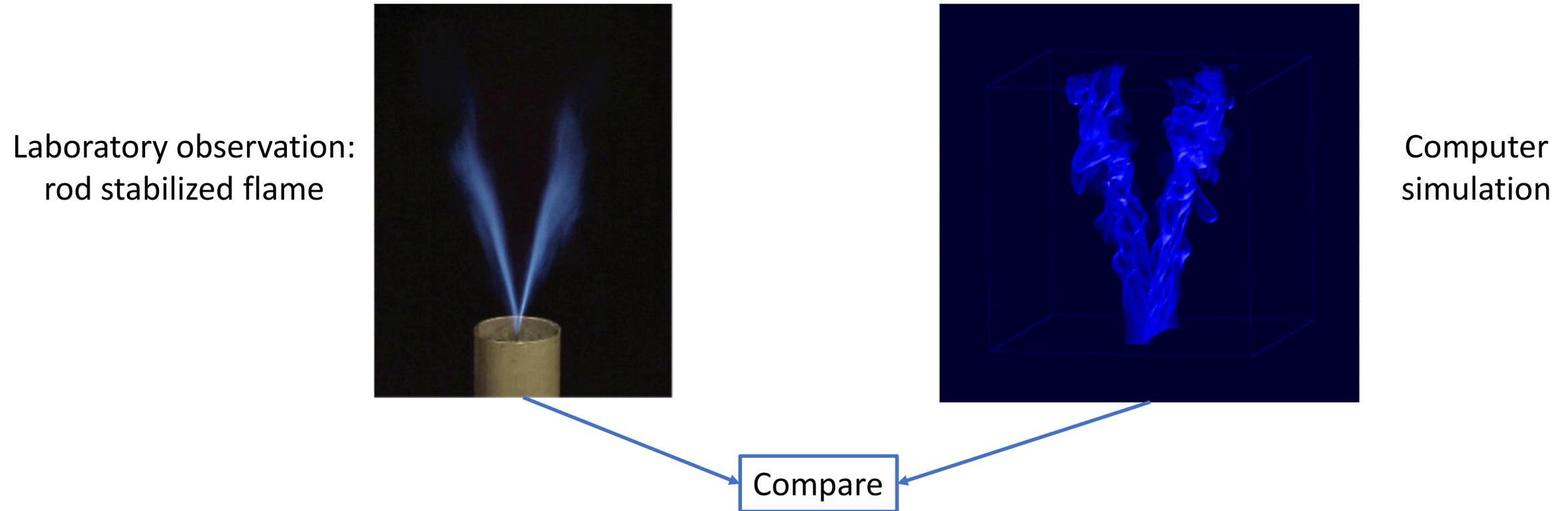
- The objective function involves a computer simulation
 - No analytic description of the objective function (black-box)
 - No gradient information
- The simulation is time consuming
- Multimodality



Goal: Find the global optimum within as few evaluations of $f(x)$ as possible

Applications arise in numerous science areas...

Practically anytime a simulation is involved



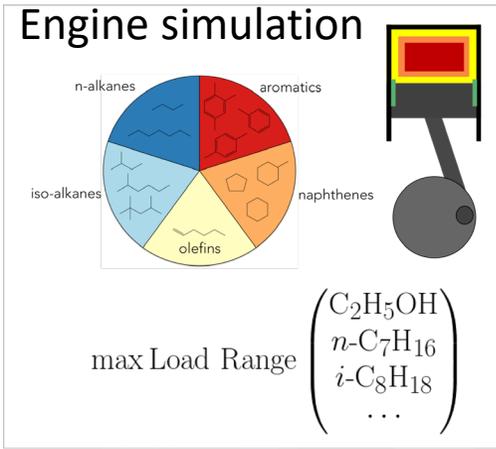
- “Tweak” (optimize) the simulation’s variables *such that the error between observed and simulated data is minimized*

Applications arise in numerous science areas...

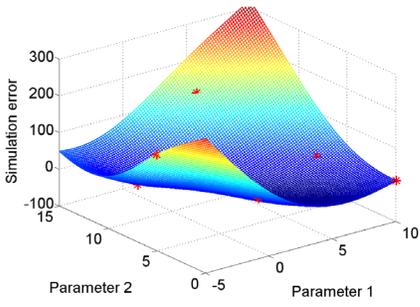
Practically anytime a simulation is involved

Quantity of interest

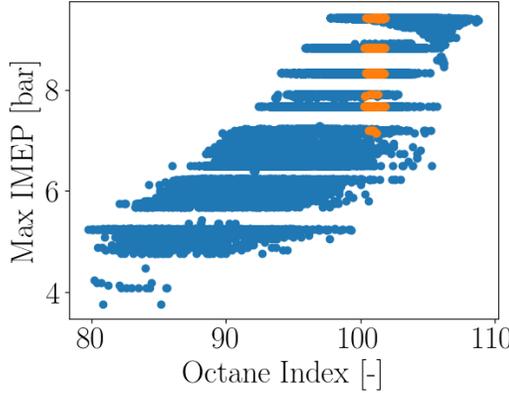
$$Q \begin{pmatrix} RPM \\ CR \\ C_2H_5OH \\ n-C_7H_{16} \\ i-C_8H_{18} \\ \dots \end{pmatrix}$$



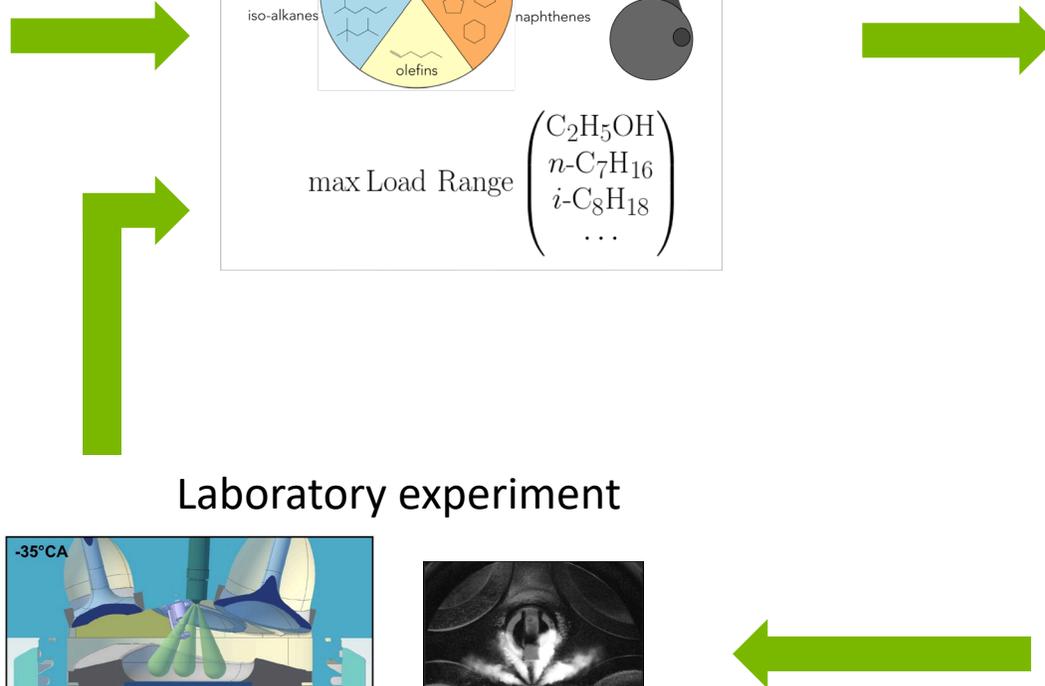
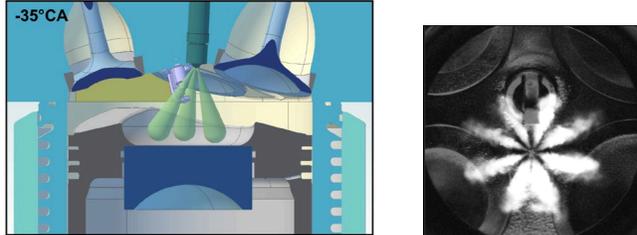
Optimization



Down-select fuels to try in lab



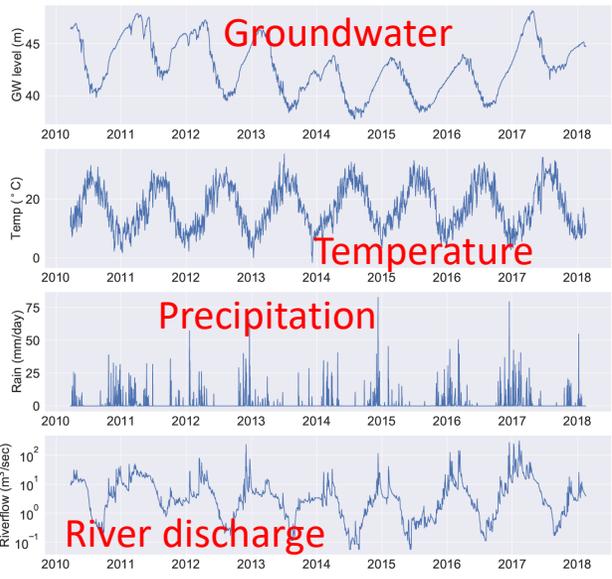
Laboratory experiment



Applications arise in numerous science areas...

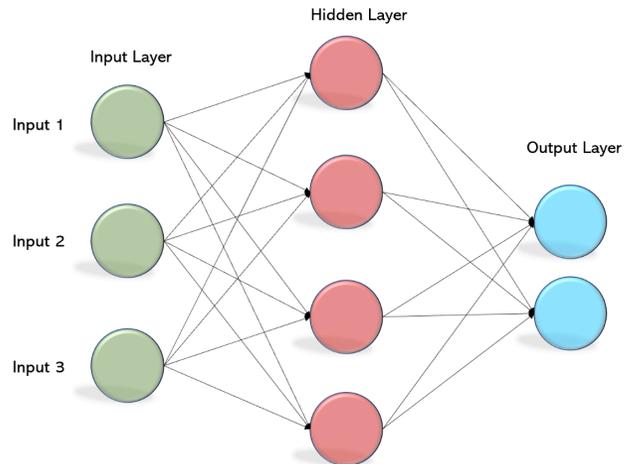
Practically anytime evaluating the objective function is expensive

E.g. Predict future groundwater levels based on historic data

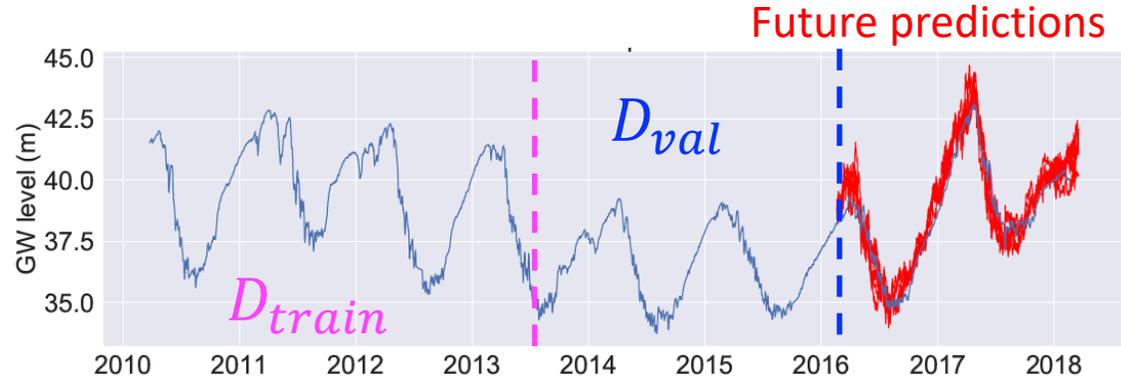


Optimization variables describe the architecture (=hyperparameters)

- # layers, nodes, epochs
- Batch size, dropout rate
- Anything else problem specific



Training a single DL model (a single architecture) can be very time consuming → only try very few architectures to find the best



Before we talk about how to solve these difficult problems, let's talk about how to *not* solve them

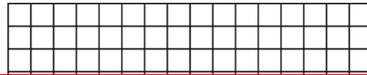
Variable sweeps (also grid sampling)

- Divide each variable's range into equidistant intervals → obtain a grid
- Evaluate your function at each grid point and declare the best point optimal

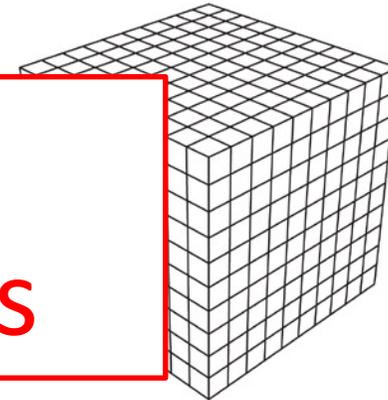
In one dimension



In two dimensions



In three dimensions



**Don't use this for
expensive evaluations**

What's the problem with this approach?

- It does not scale well:
 - For one variable, with 10 points → 10 evaluations
 - For two variables, with 10 points each → 10x10 evaluations
 - For d variables, with 10 points each → 10^d points
 - For example, 10 variables = 10^{10} evaluations, 1hour per evaluation = 10'000'000'000 hours or 416'666'666 days or 1'141'552 years

Random sampling

- Randomly select a bunch of points from the variable space and evaluate
- Declare the best point as your optimum

Drawback:

- *How many points do you evaluate? Which distribution?*
- *No mechanism to sample interesting regions of the variable space more thoroughly*

Don't use this for optimization

Tuning “by hand” and using intuition

- Manually
- Human i
- optimization
- Humans
- dimension

Don't do this if you ever want to graduate

- Expert's intuition *can* be helpful
- Often results in local search, missing better (unexpected solutions)

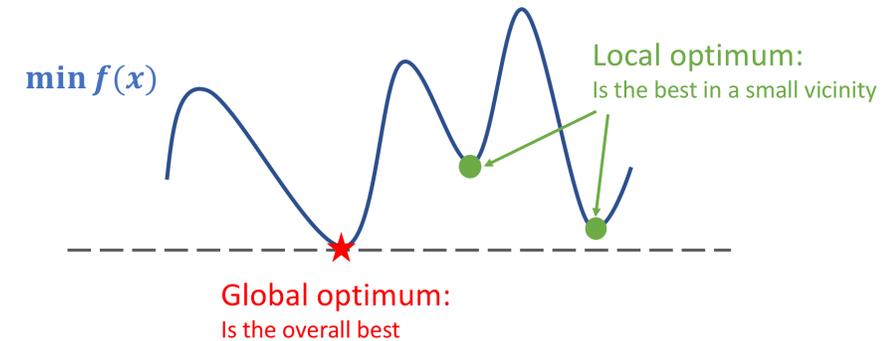
The screenshot shows a software interface with several panels. A red box highlights a central area containing a table with 13 rows and several columns. The table has a header row with labels '5th', '6th', '7th', '8th', '9th', '10th', '11th', '12th', '13th', and 'sub'. The columns contain numerical values and text labels like 'select', 'up/5', 'no diff', and '0'. The interface also includes a top navigation bar with tabs like 'Man Mode', 'Polarization', and 'Action', and a right-hand panel with a 'priority' list and other controls. Red circles with numbers 1 through 5 point to specific UI elements: 1 points to the 'Action' tab, 2 to the 'Man Mode' tab, 3 to the 'Polarization' tab, 4 to the 'reflectivity' dropdown, and 5 to the 'priority' list.

Don't use optimization methods that were not developed for the type of problem you have

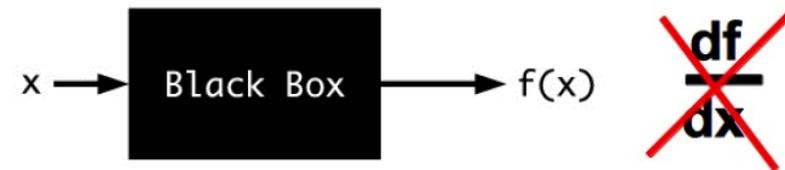
- Heuristics (GA's, SA,...):
 - Too inefficient for expensive simulations



- Local search methods for multi-modal problems:
 - Finds only local optimum

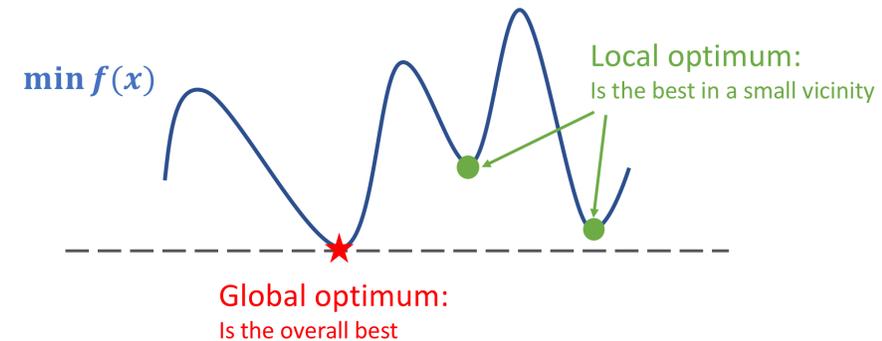


- Methods relying on gradient information:
 - Not useful if you don't have gradient



Takeaway: It is always worth it to invest time in searching for the appropriate optimization algorithm for your specific problem

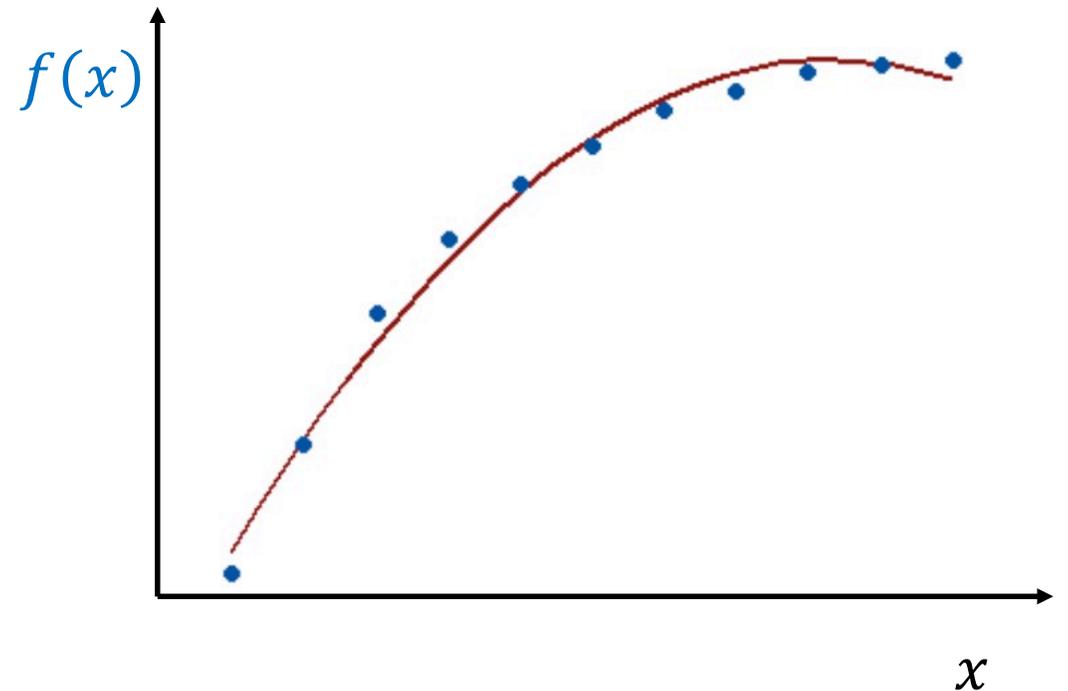
How to solve our difficult optimization problems?



Computationally *cheap surrogate models* to approximate the *expensive function*:

$$f(x) = s(x) + e(x)$$

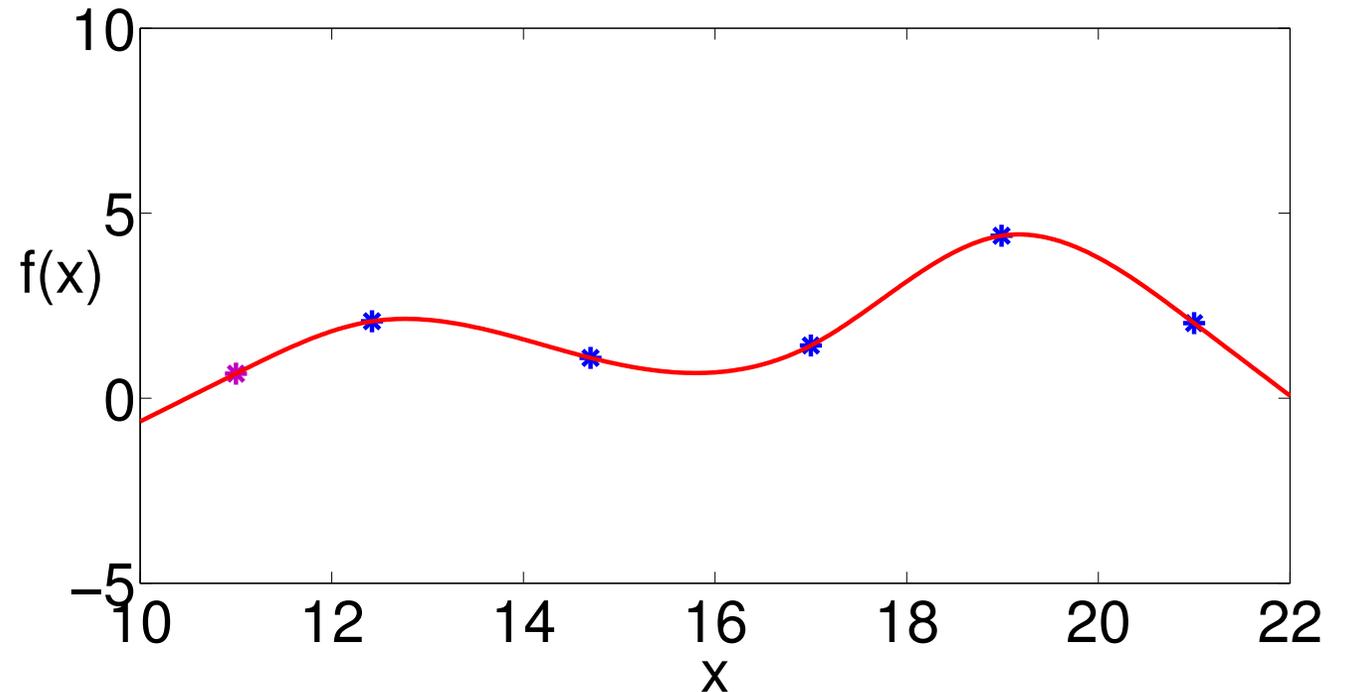
- $s(x)$ could be a
 - *polynomial regression model*



Computationally *cheap surrogate models* to approximate the *expensive function*:

$$f(x) = s(x) + e(x)$$

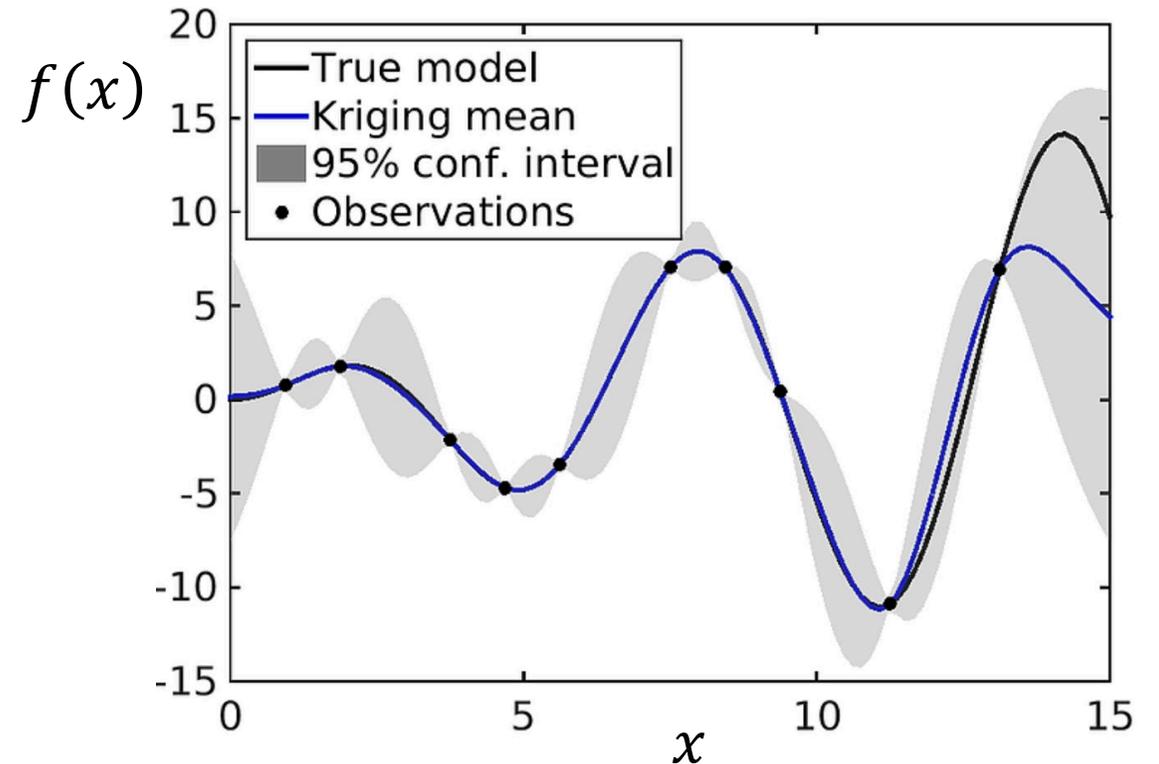
- $s(x)$ could be a
 - polynomial regression model
 - *Radial basis function model*



Computationally *cheap surrogate models* to approximate the *expensive function*:

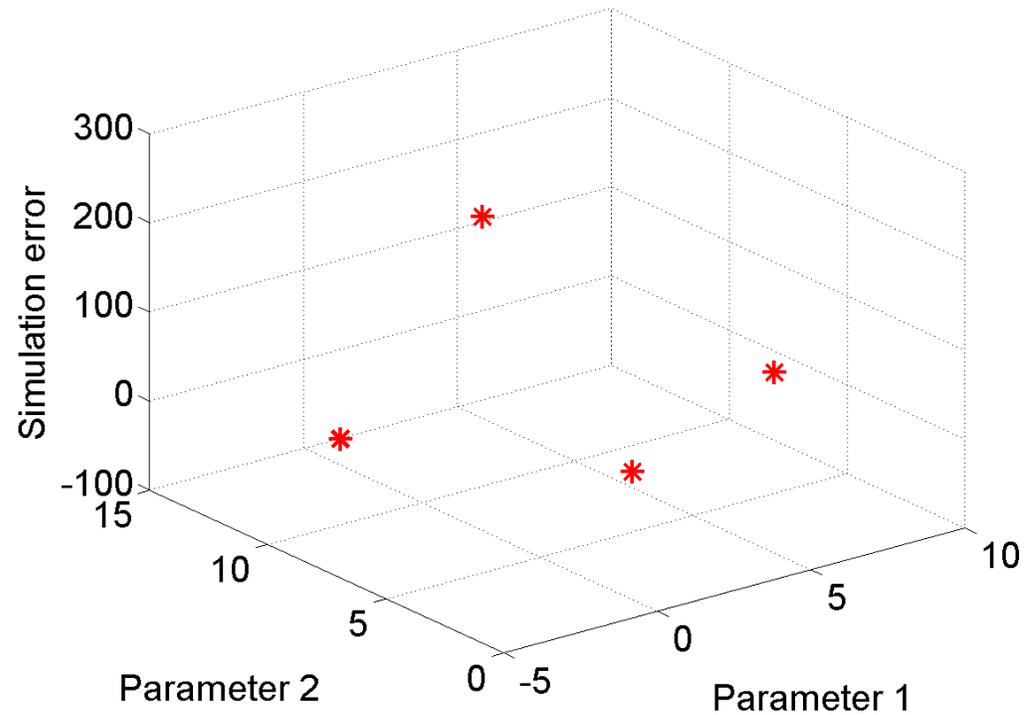
$$f(x) = s(x) + e(x)$$

- $s(x)$ could be a
 - polynomial regression model
 - Radial basis function model
 - *Kriging model (Gaussian process)*



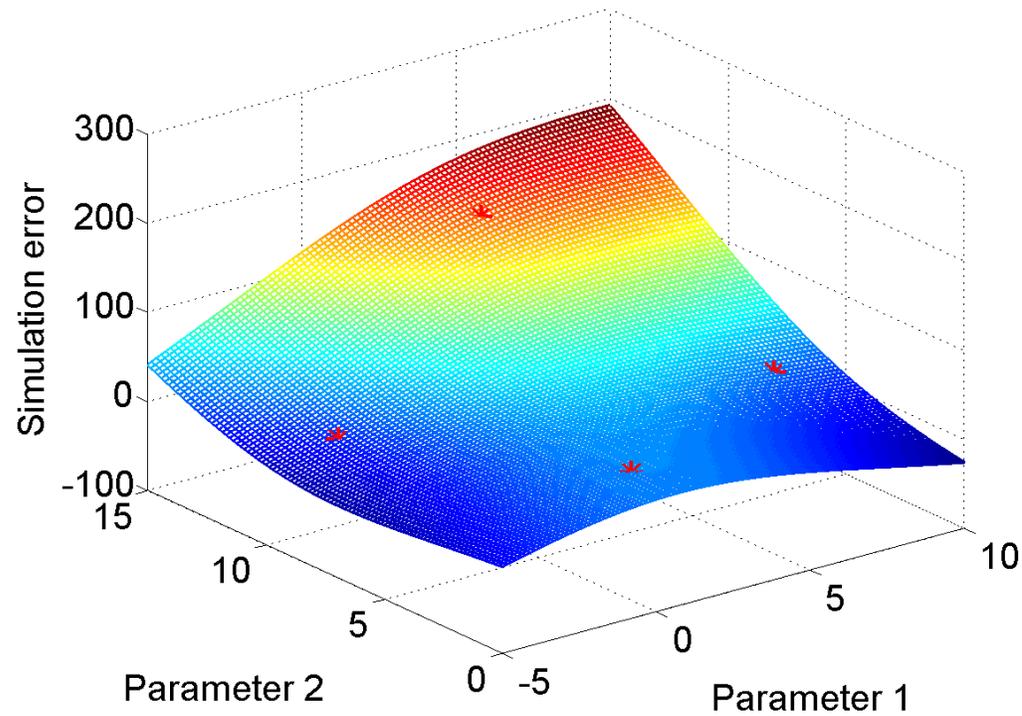
How does it work?

- Before fitting a surrogate model, we need some data
 - Use an *initial experimental design* and we evaluate the expensive simulation



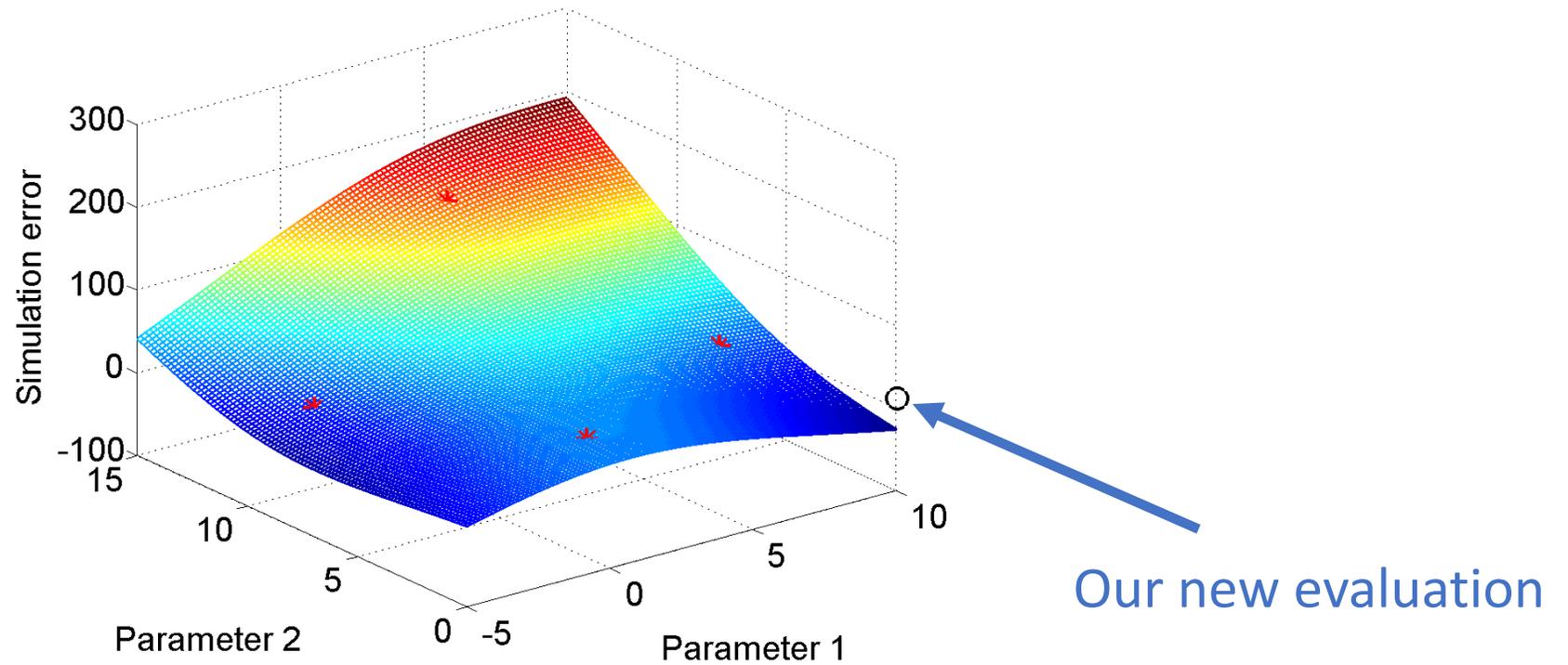
How does it work?

- Now we have enough information to fit a surrogate model
 - We use radial basis functions, but could be anything, really



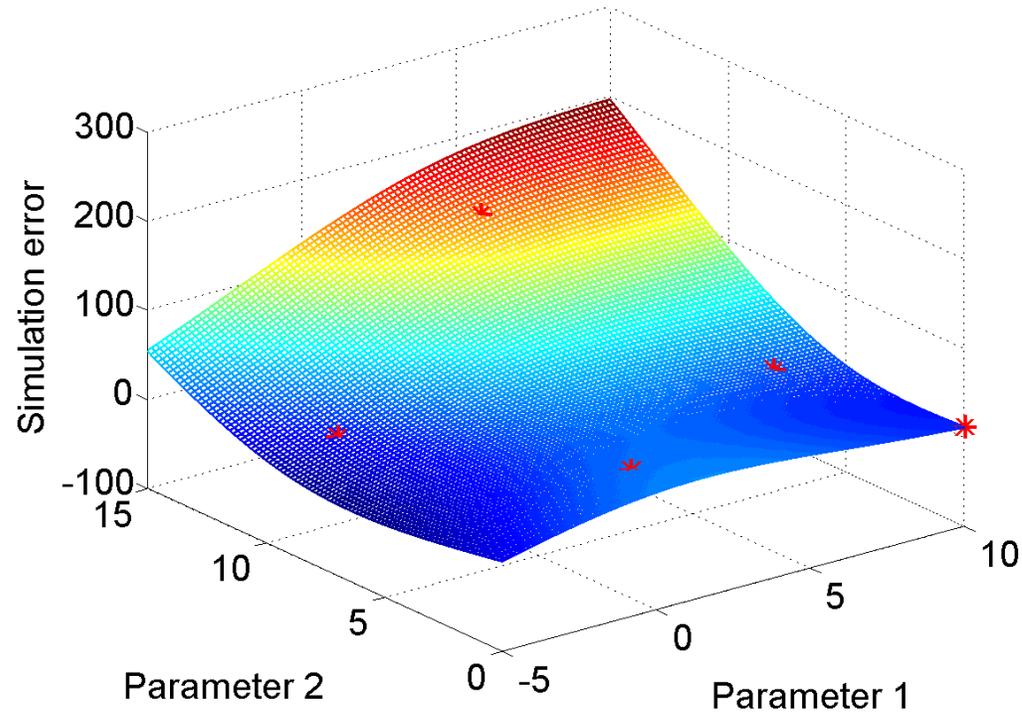
How does it work?

- We use the surrogate model to select a new point for evaluation



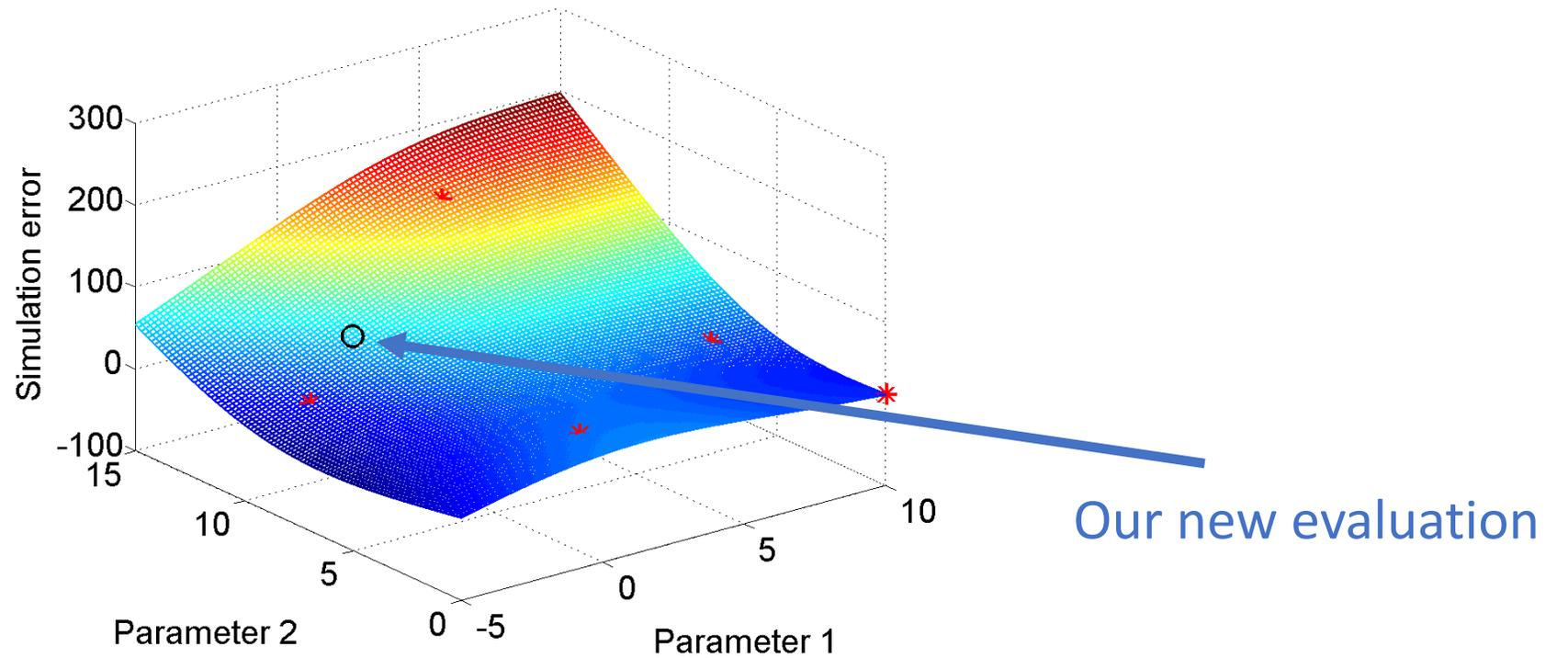
How does it work?

- We use the new piece of information to update the surrogate model



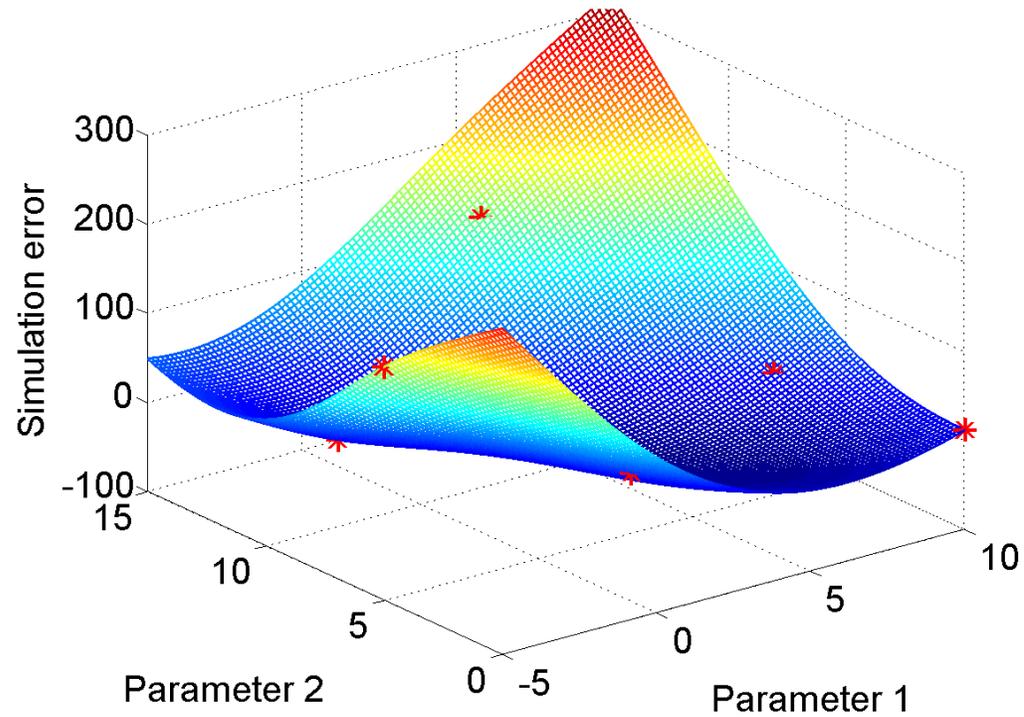
How does it work?

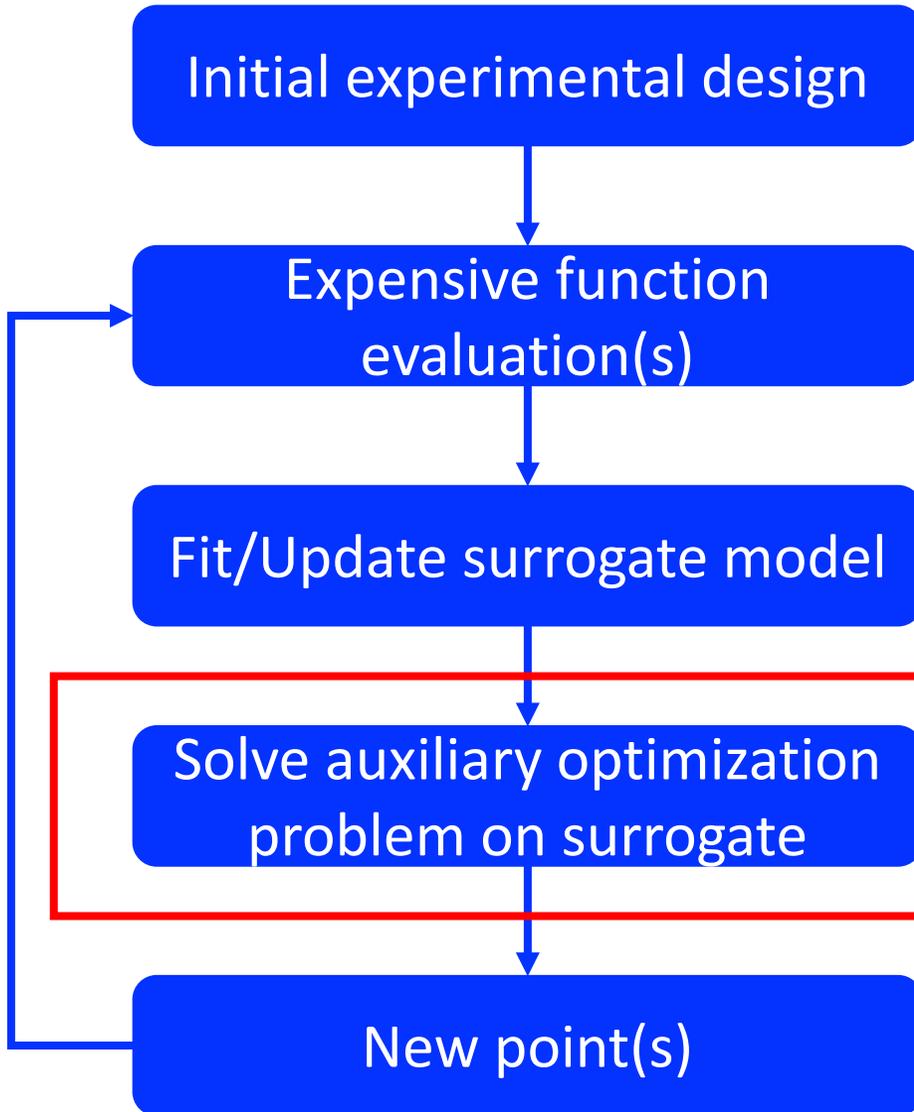
- Select a new point and evaluate the expensive simulation



How does it work?

- Update the surrogate model with the new piece of data





- Latin hypercube, Sobol sequence,...

- Here you call the simulation or do a time intensive experiment

- Radial basis function, Gaussian process, Deep learning model

- Maximize expected improvement; optimize tradeoff between predicted performance and distance; minimum of surrogate model

**This is where most of the research happens:
develop strategies for selecting new points**

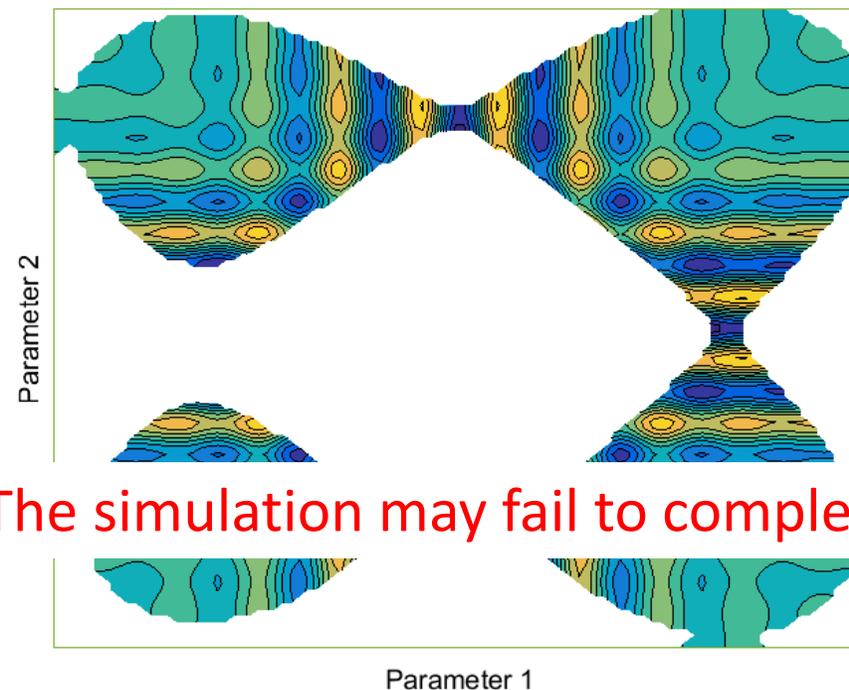
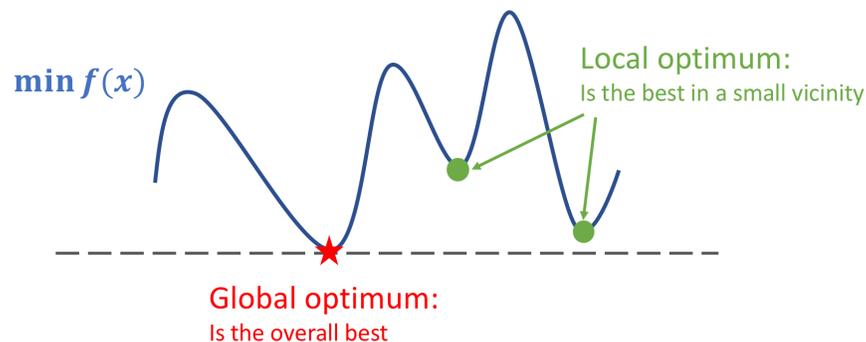
- STOP when you reached your function evaluation or compute budget

We keep doing this until we run out of time

- If we have about 100 hours to find a solution, and one simulation run takes an hour, that means we get at most 100 simulation evaluations done
- The best solution found at the end of the optimization is "the best we can do with the limited resources we have"
 - Is it globally optimal? – Maybe.
 - Is it locally optimal? – Sometimes.
 - Is it better than what the scientist used before? – Most likely.

An example where we used this method (successfully) – Combustion simulation

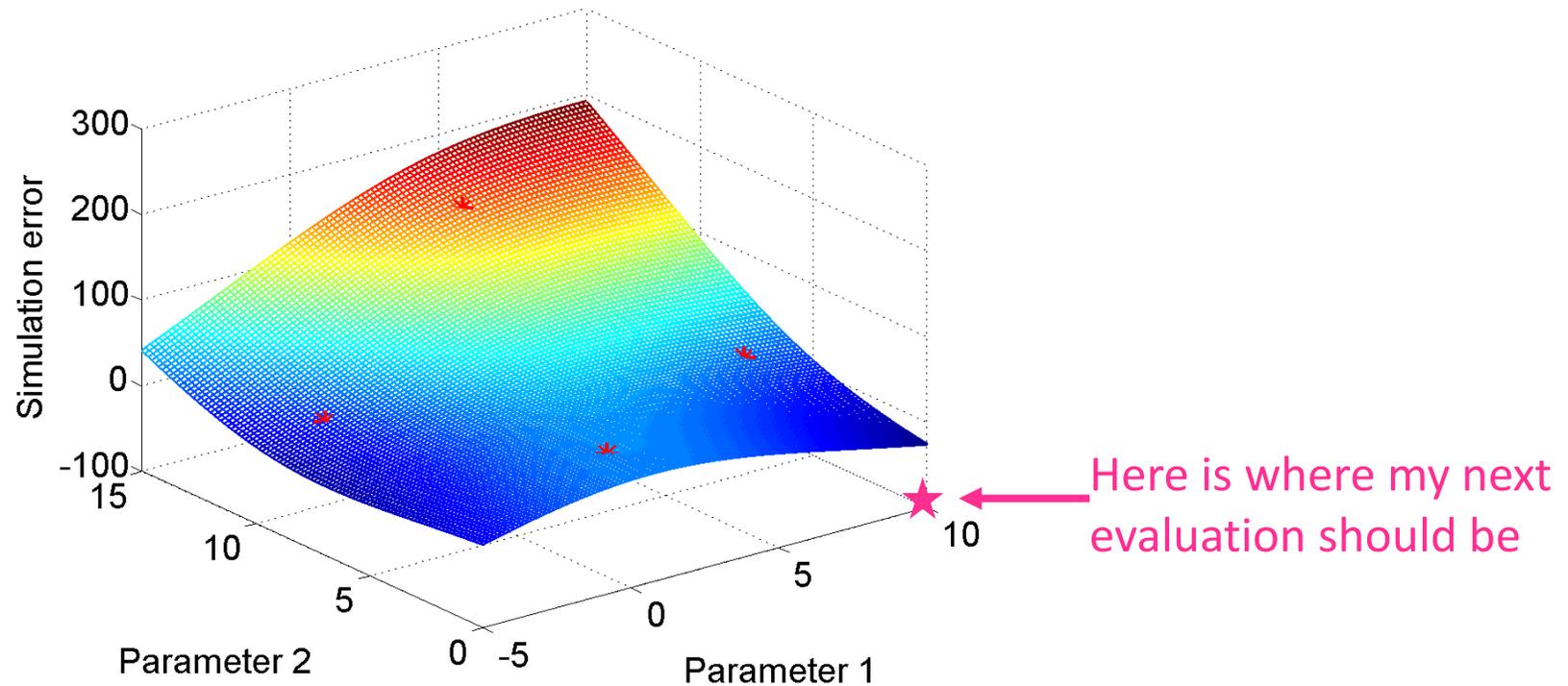
- Simulation of gas-phase chemical combustion
- Modeling involves fluid conservation laws, thermodynamic relationships, diffusive transport, chemical kinetics
- **Minimize the error** between simulation and observation



The simulation may fail to complete

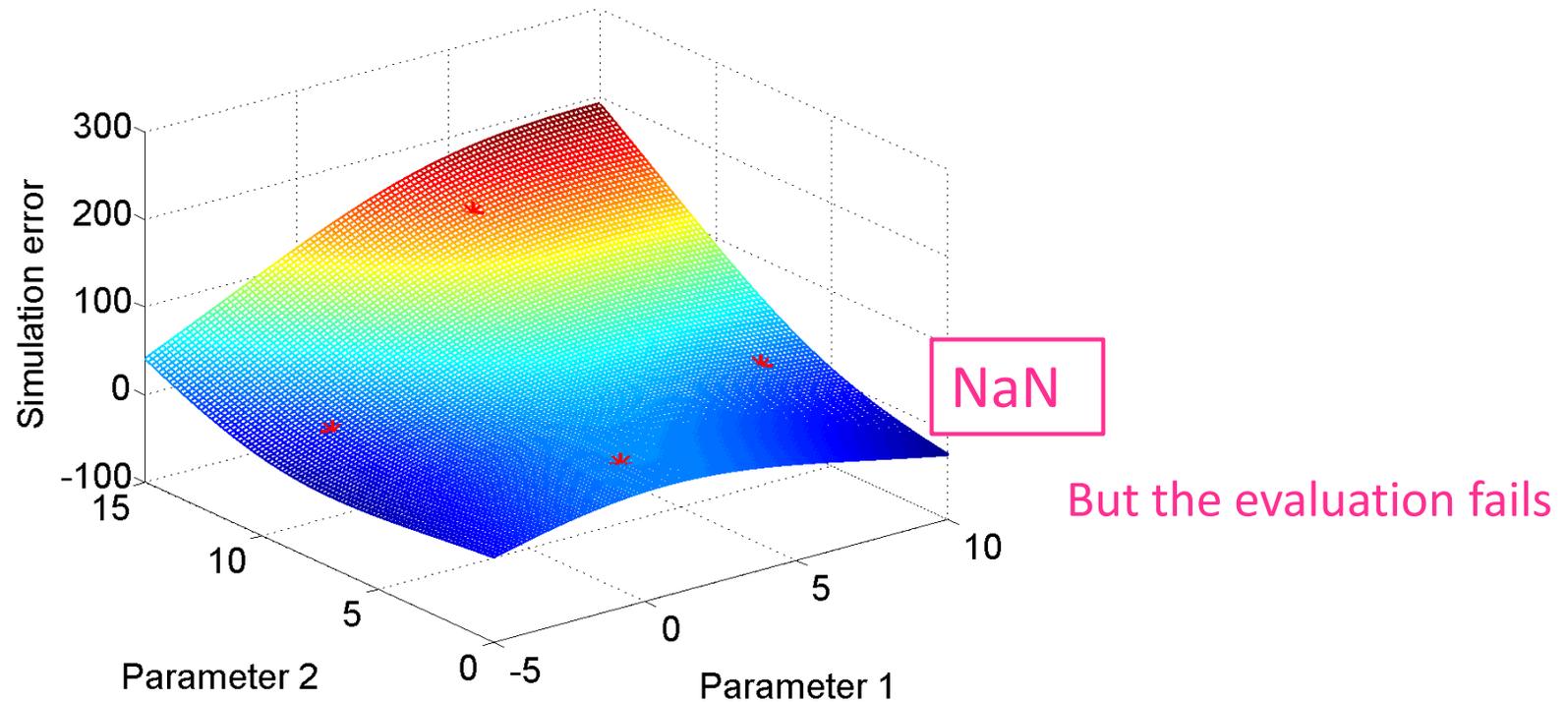
Combustion application and failing simulations

- With failing simulations – how do we build our surrogate model??



Combustion application and failing simulations

- With failing simulations – how do we build our surrogate model??

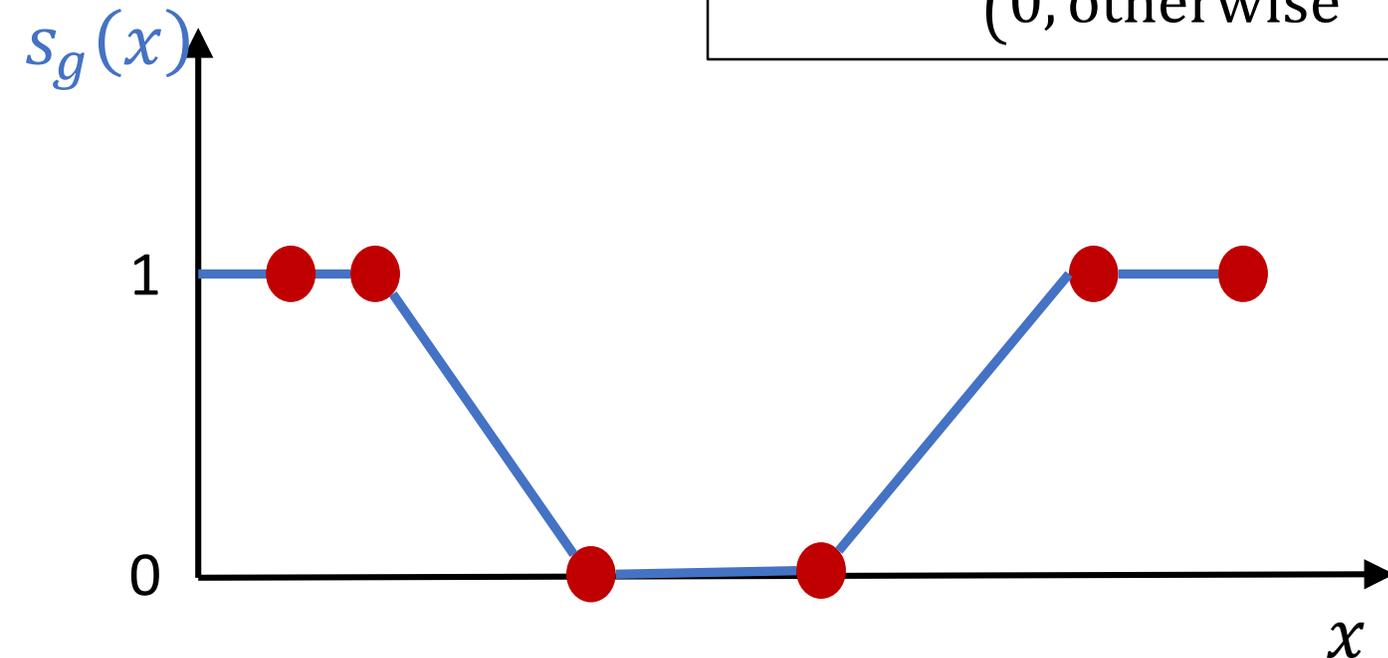


I cannot update my surrogate model !

Introduce a constraint to deal with failed simulations

- Need some trickery to deal with this (1-dimensional illustration):
- Build a function that predicts how likely it is that a variable vector will lead to successful evaluation

$$g(x) = \begin{cases} 1, & \text{if } f(x) \text{ evaluates successfully} \\ 0, & \text{otherwise} \end{cases}$$



- Use a piecewise linear approximation $s_g(x)$ to predict evaluability at every point x

Iteratively increase the evaluability threshold

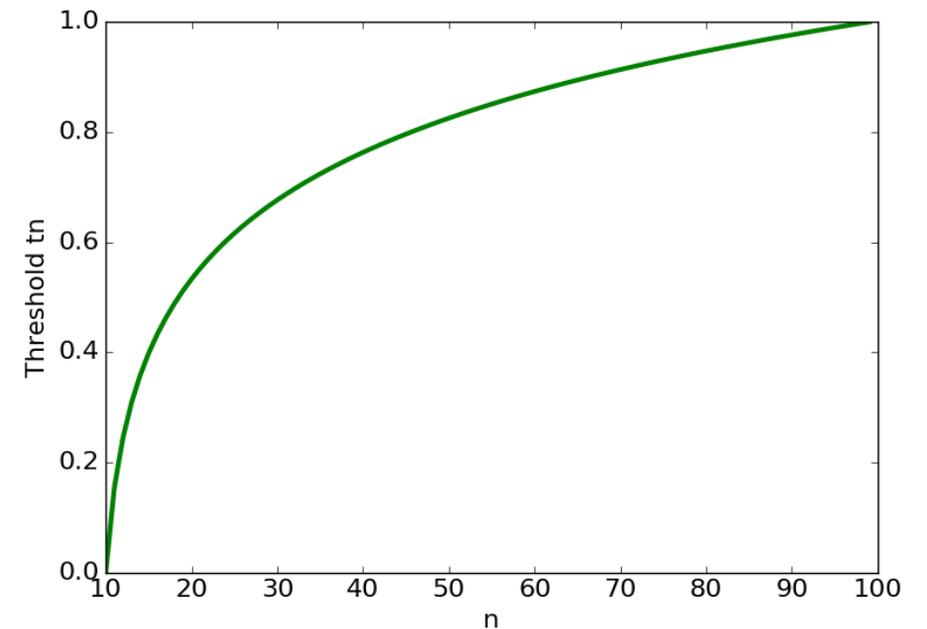
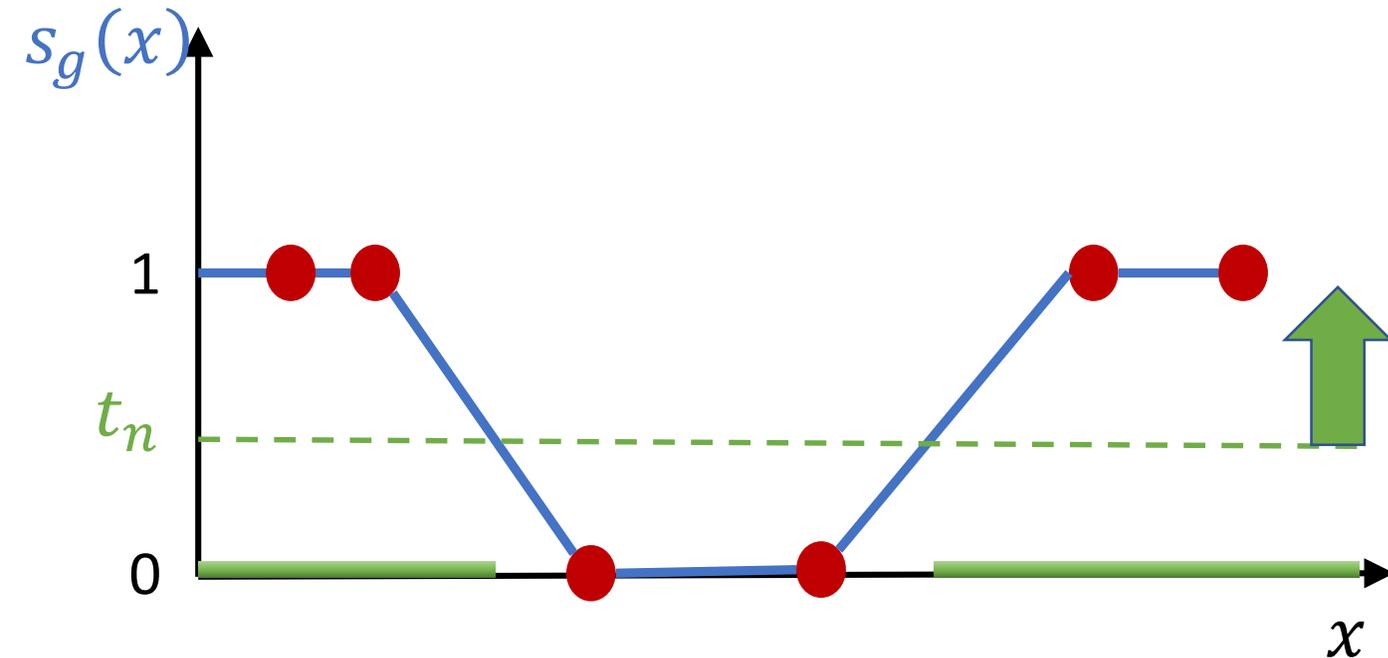
- Need some trickery to deal with this:

- Define a threshold

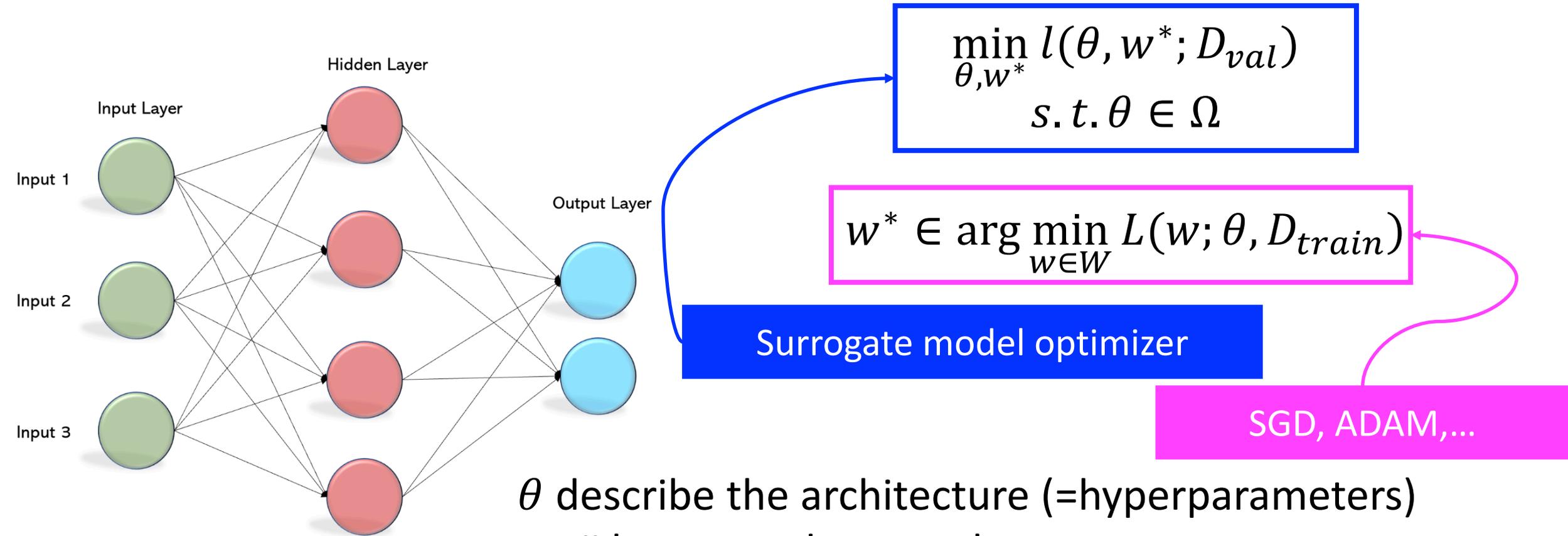
$$t_n = \frac{\log(n - n_0 + 1)}{\log(n_{max} - n_0)}$$

- And require $s_g(x) \geq t_n$

t_n dynamically increases as the number of evaluations n grows



Optimizing the architecture of Deep Learning Models by bilevel optimization *with integers*



θ describe the architecture (=hyperparameters)

- # layers, nodes, epochs
- Batch size, dropout rate
- Anything else problem specific

Our approach yields excellent results for predicting groundwater in California

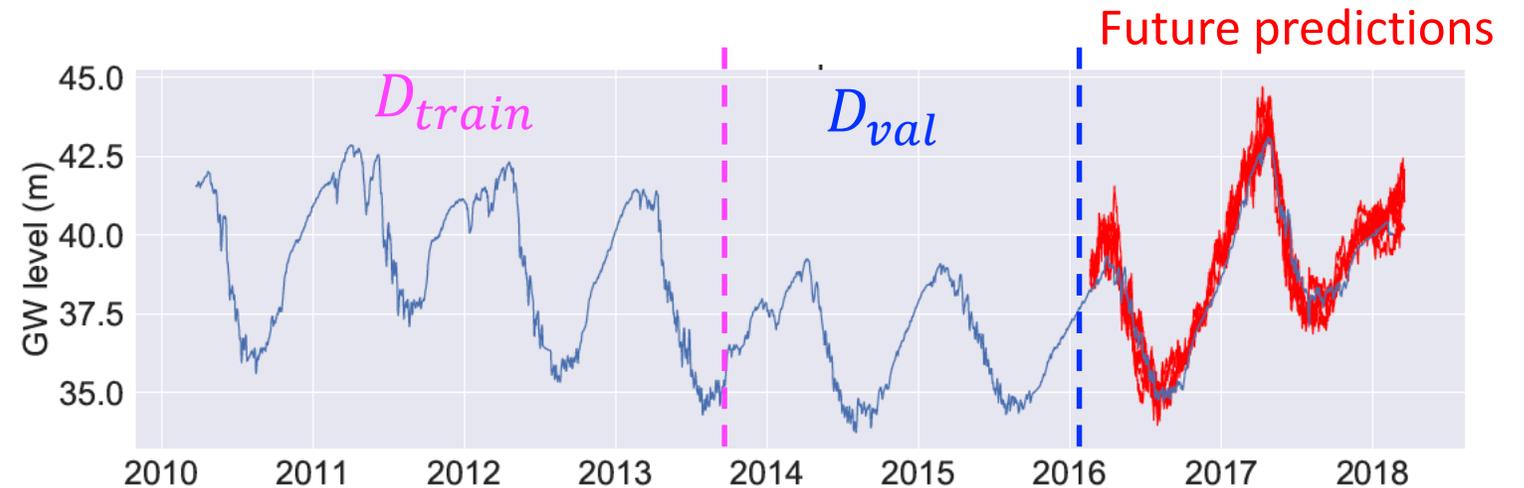
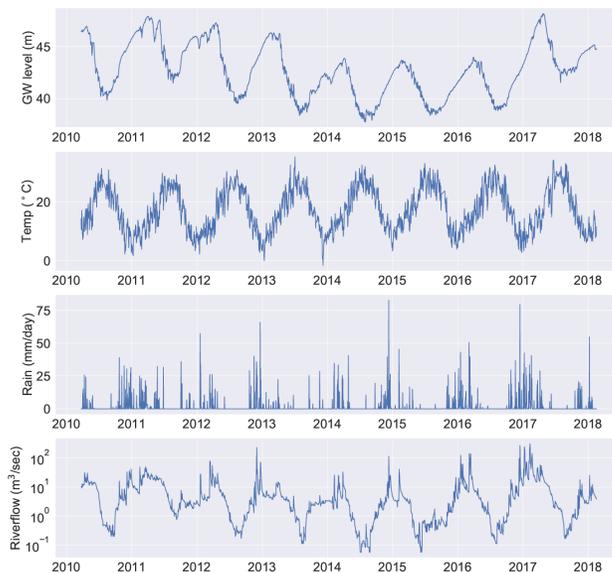
- Inform with
- Temperature
 - Precipitation
 - Time of year

With CRD and EESA at LBNL

- Historic data
- Temperature
 - Precipitation
 - River discharge
 - Groundwater levels
 - Time of year

HPO of deep learning model (tried CNN, MLP, LSTM, RNN)

- Future predictions for
- Groundwater levels

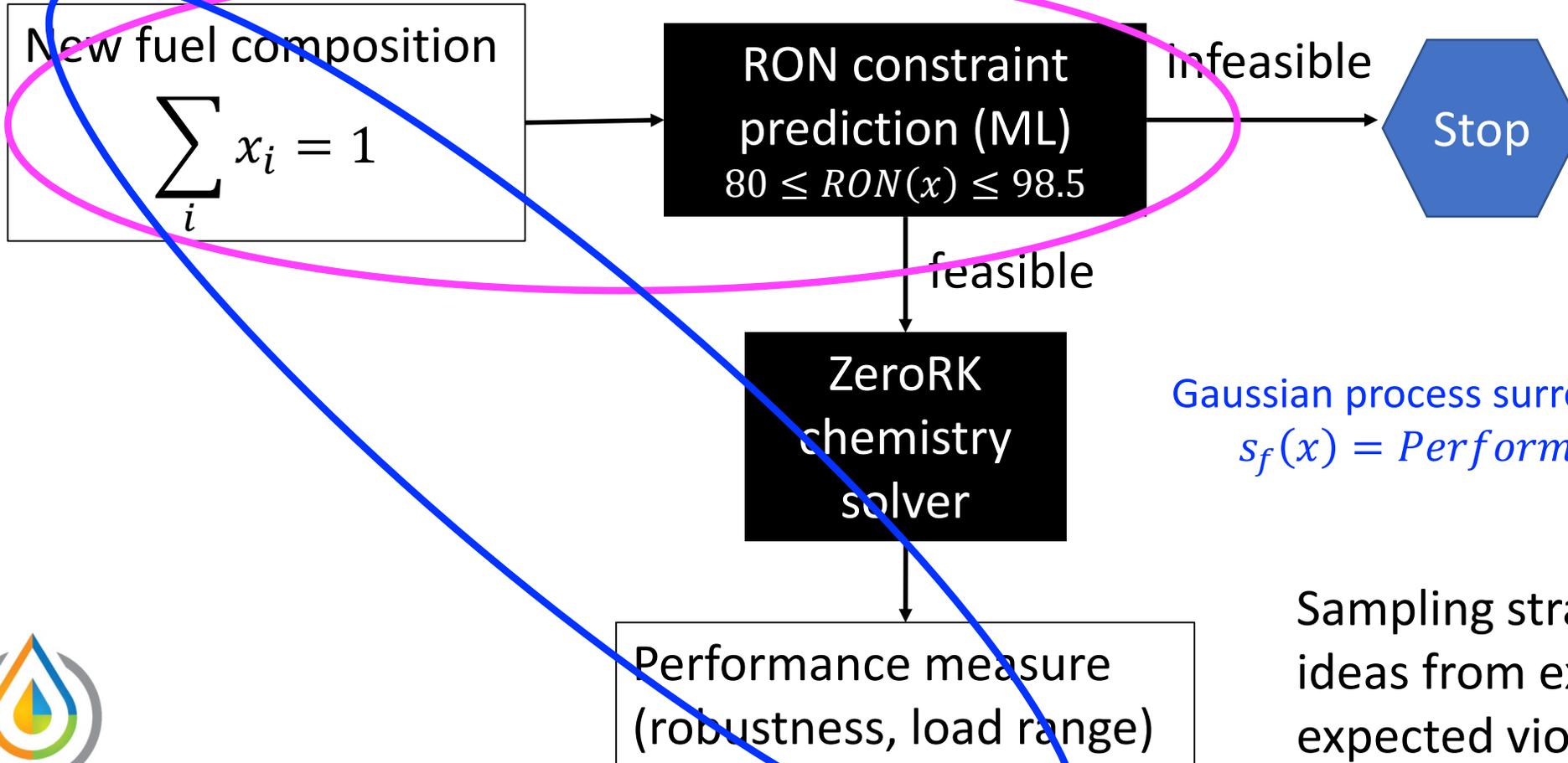


Constrained optimization to discover new fuels

Gaussian process surrogate for constraint:

$$s_g(x) = RON(x) + e_g(x)$$

Ongoing, with Lapointe
and McNenley @LLNL



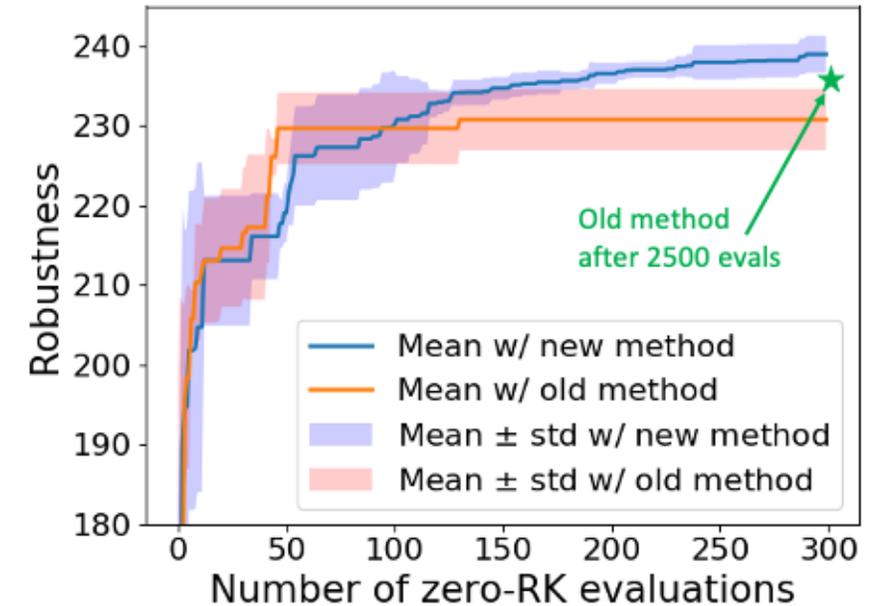
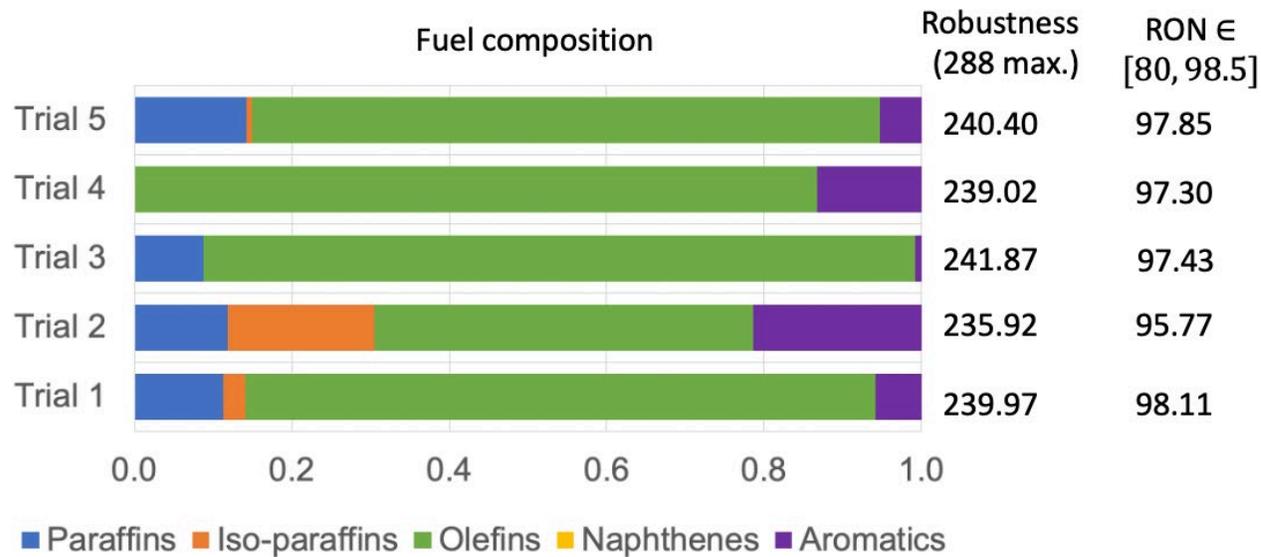
Gaussian process surrogate for objective:

$$s_f(x) = Performance(x) + e_f(x)$$

Sampling strategies are based on ideas from expected improvement, expected violation, and probability of feasibility

Different fuel blends yield similar performance

- Solve 2 different optimization problems (max load range; max robustness)
- 5 trials with the algorithm (bc algorithm is stochastic)
- 9 fuel components (5 fuel groups)



- Different fuel compositions with similar performance
- High amounts of olefins, low /no naphthenes, yield highest robustness

- Convergence plot: higher is better
- Old method = genetic algorithm
- New method = Gaussian process surrogate model algorithm

Other applications

- Reliability redundancy optimization – maximize system reliability
- Global climate model – calibrate parameters related to CH4 model
- Airfoil design – maximize lift and minimize drag simultaneously
- Watershed management – retire agricultural lands to reduce Ph runoff
- Particle physics – match simulations with observations
- Engine efficiency – design better engines and better fuels
- Renewable energies – maximize energy generated by kites, hydropower
- Scheduling – how to assemble products in line most efficiently
- And many more....

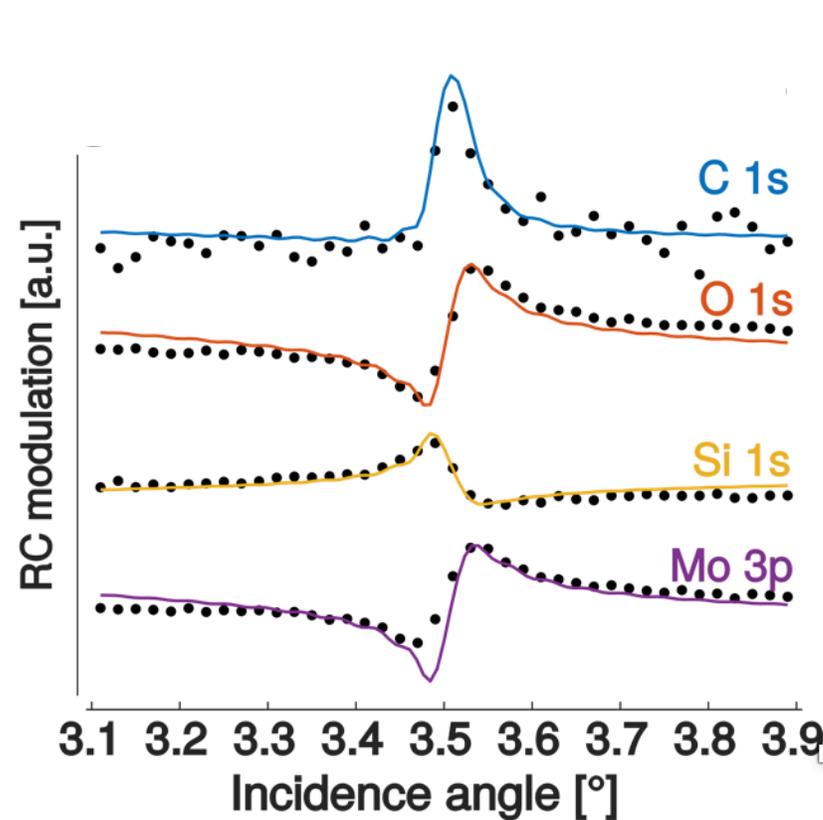
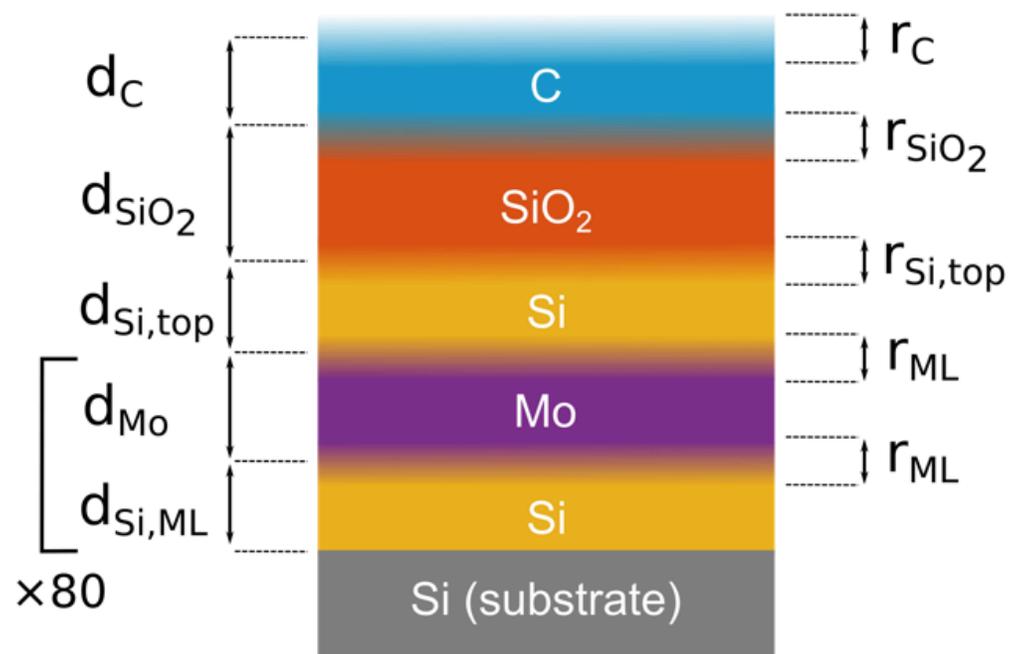
Bottomline: optimization is needed almost everywhere
Study optimization!

What did you learn today?

- What is the goal of doing optimization?
- Why do I not want to use grid sampling for objective functions that take a long time to evaluate?
- What are example applications that can/should make use of optimization?

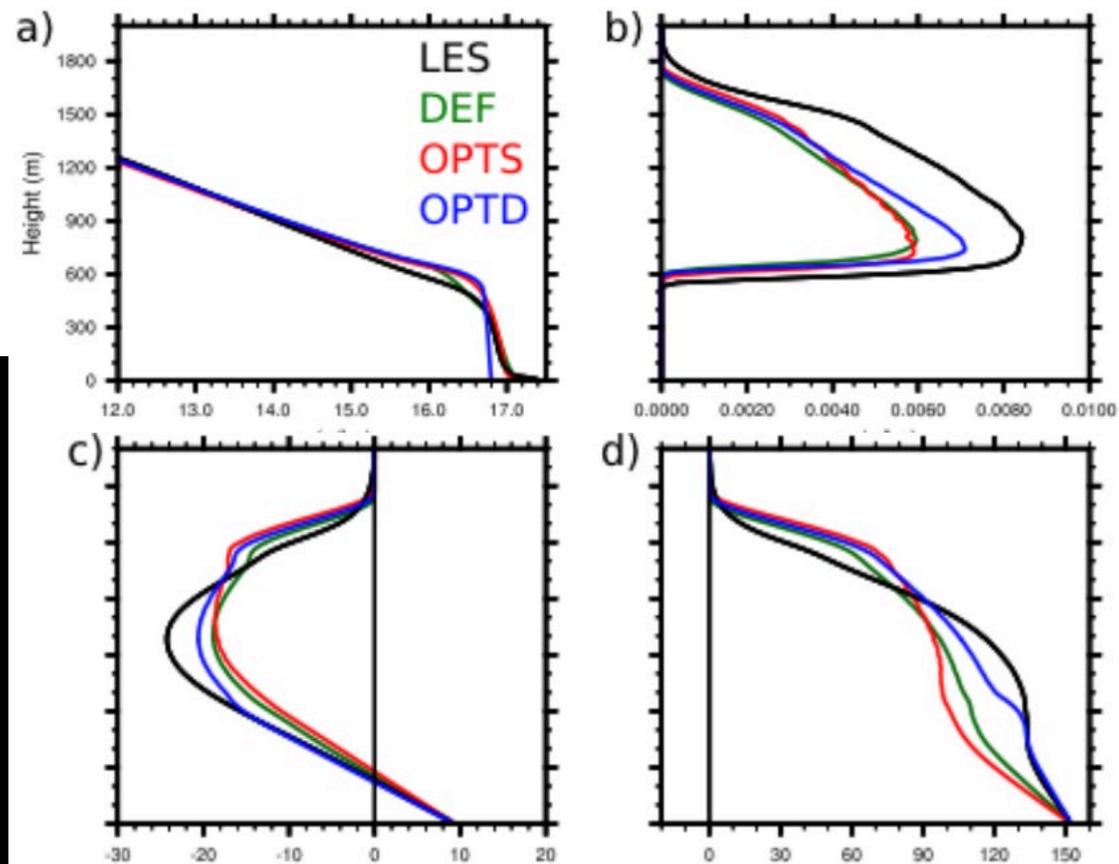
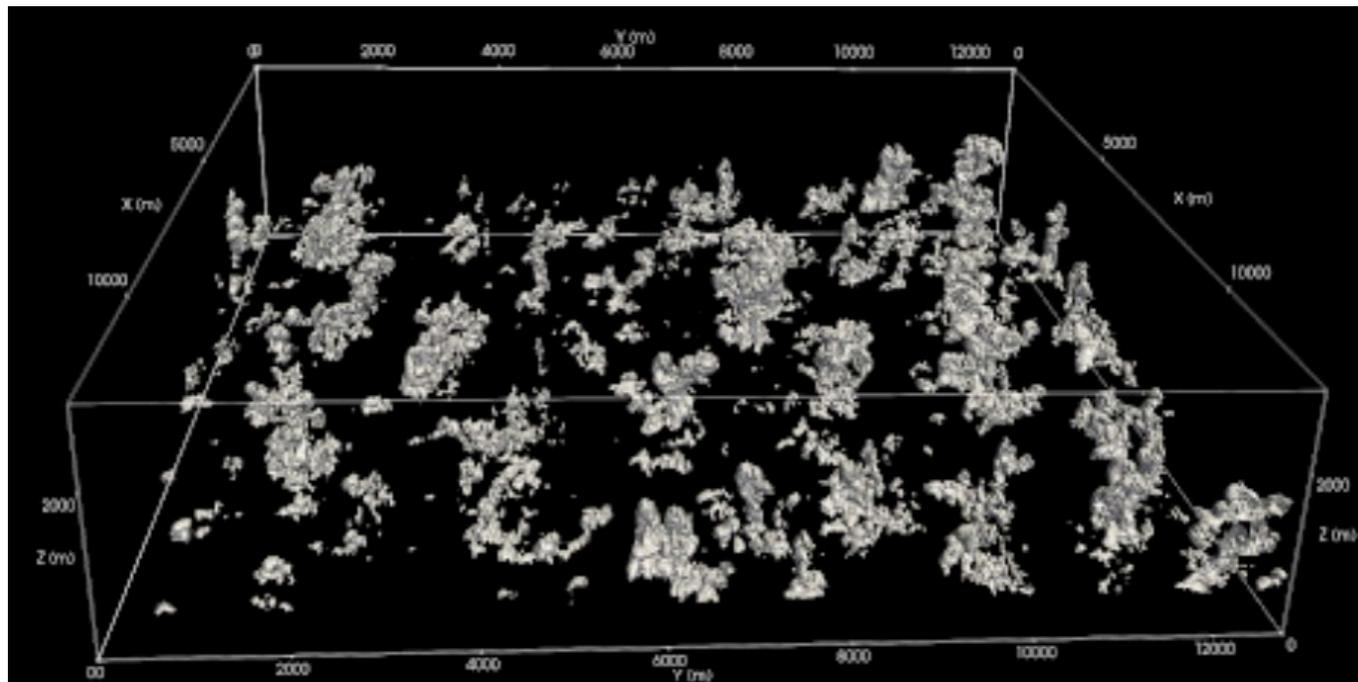
Some examples where we used this method (successfully) – Materials science/Chemistry

- Use X-ray standing wave to determine the thicknesses of the layers (d's and r's)
- **Minimize errors** between simulation (graphs) to observation (black dots)



Some examples where we used this method (successfully) – Cloud simulations

- Simulate how clouds form
- Compare simulations to observation data sets (**minimize error**)



LES = benchmark **OPTS**, **OPTD** = optimizations
DEF = default

Get as close to the black line as possible