Optimizing Expensive Black-box Simulations

Juliane Mueller

Center for Computational Sciences and Engineering

JulianeMueller@lbl.gov

https://optimization.lbl.gov/

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…

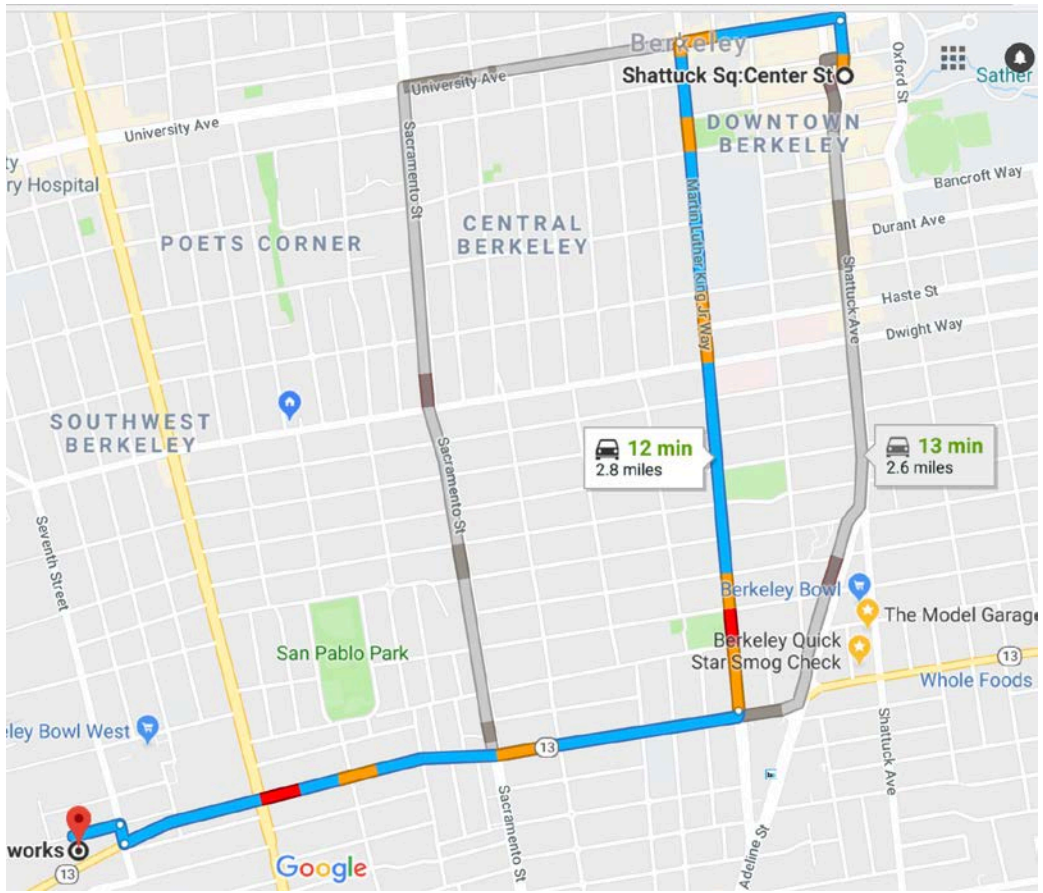
What is optimization?

- 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?

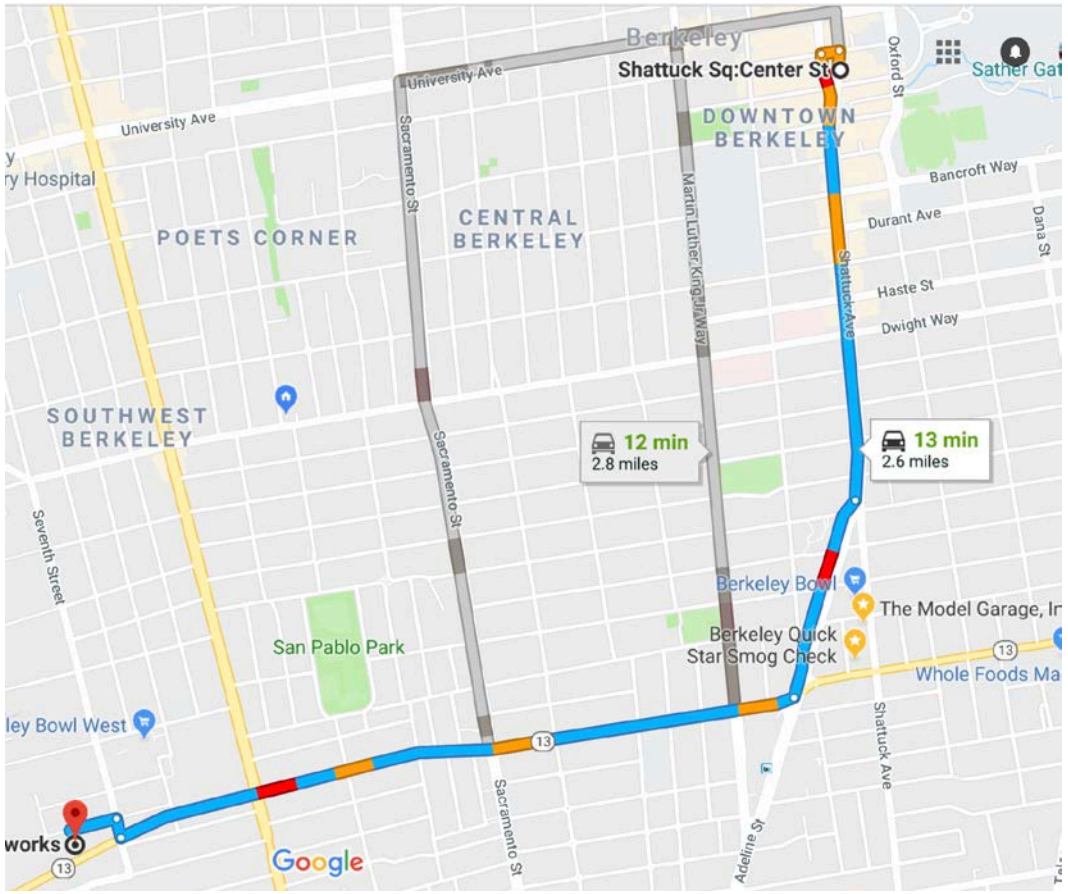
- We encounter optimization whenever we want to improve something, e.g.,
  - Get form A to B in the **fastest** way possible



Before we talk about black-box optimization…

What is optimization?

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

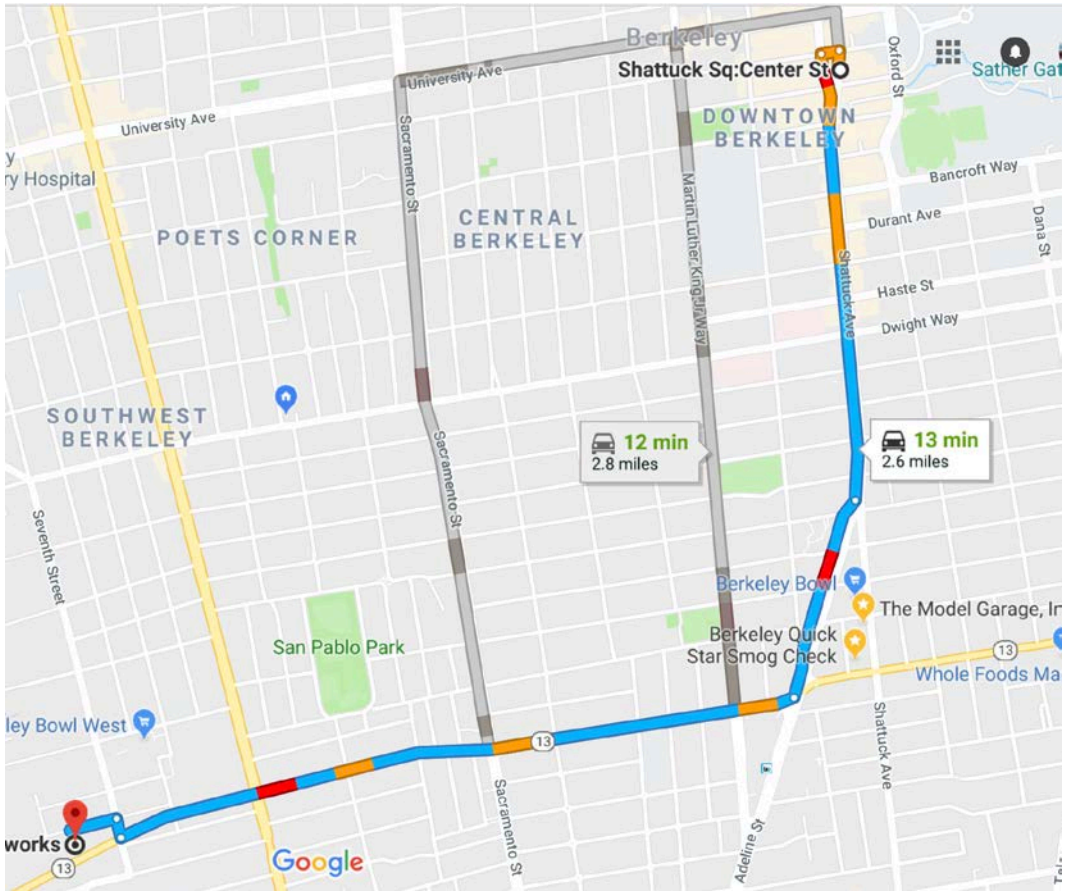


Find the “best” of something

Before we talk about black-box optimization…

What is optimization?

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

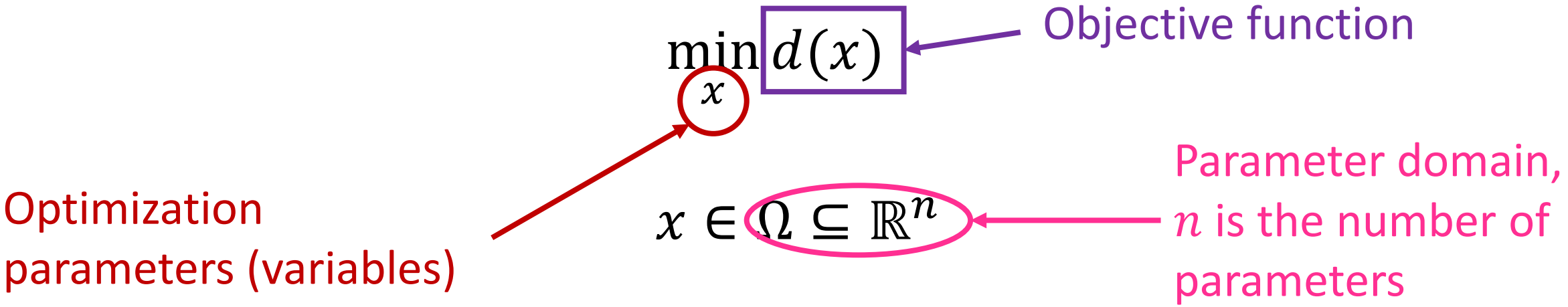


Formulate it as an optimization problem:

- 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 before we continue..

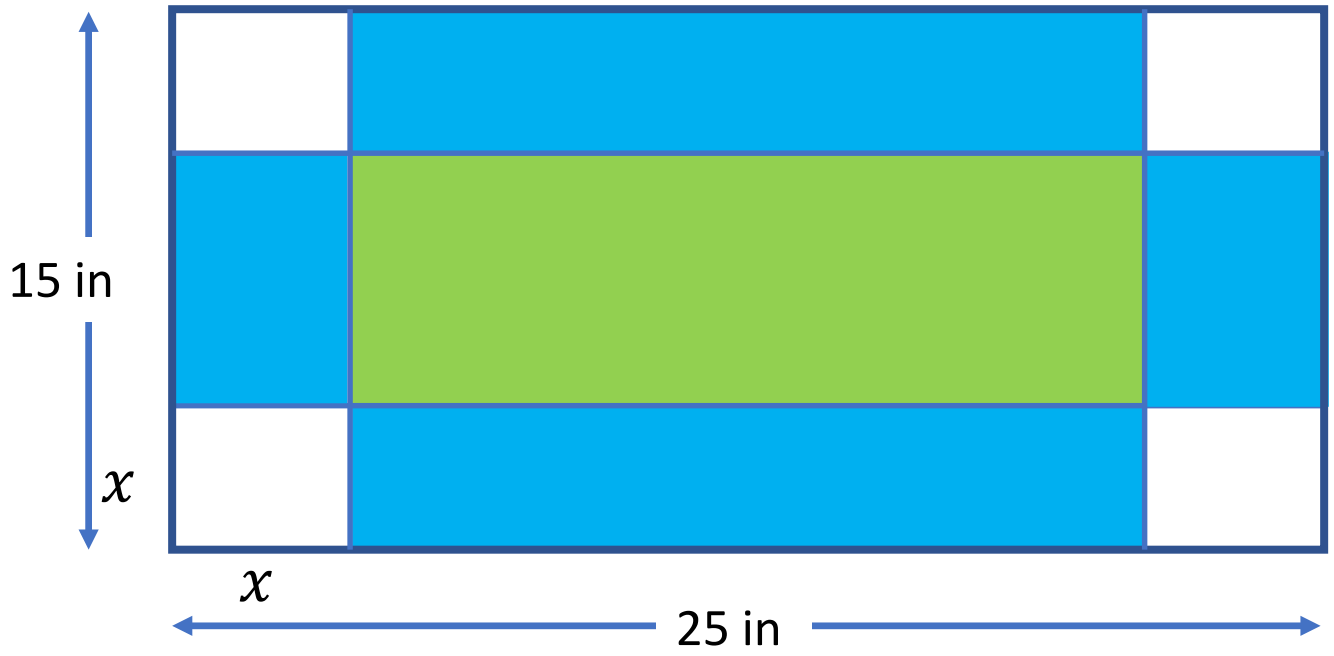


$g_j(x) \leq 0, j = 1, \ldots, J$ Inequality constraints

$h_k(x) = 0, k = 1, \ldots, K$ Equality constraints

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



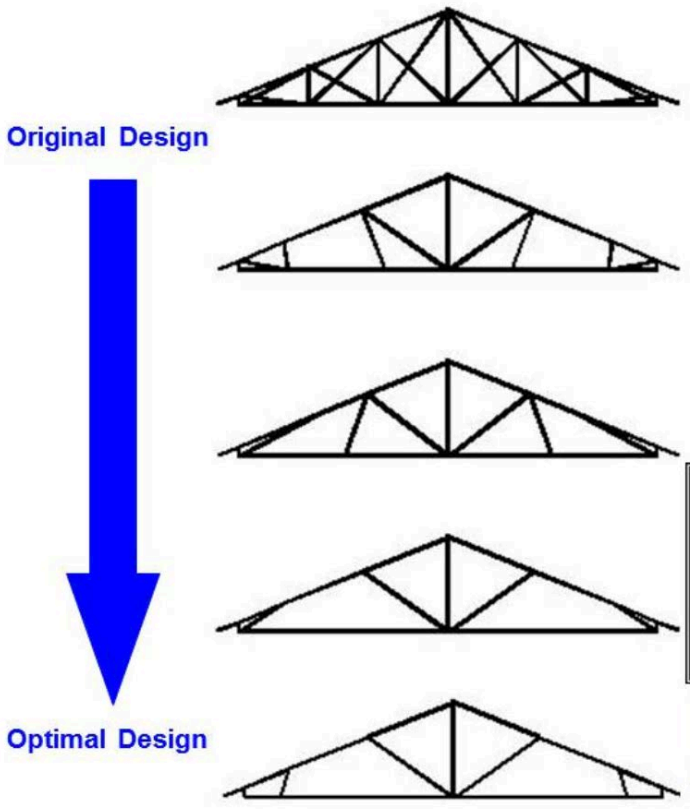
- x is the optimization parameter
- Objective function:
  Volume = width * length * height:
  $V(x) = (15 - 2x)(25 - 2x)x$
- Our constraints:
  - $x \geq 0$
  - $15 - 2x \geq 0 \rightarrow x \leq 7.5$
  - $25 - 2x \geq 0 \rightarrow x \leq 12.5$

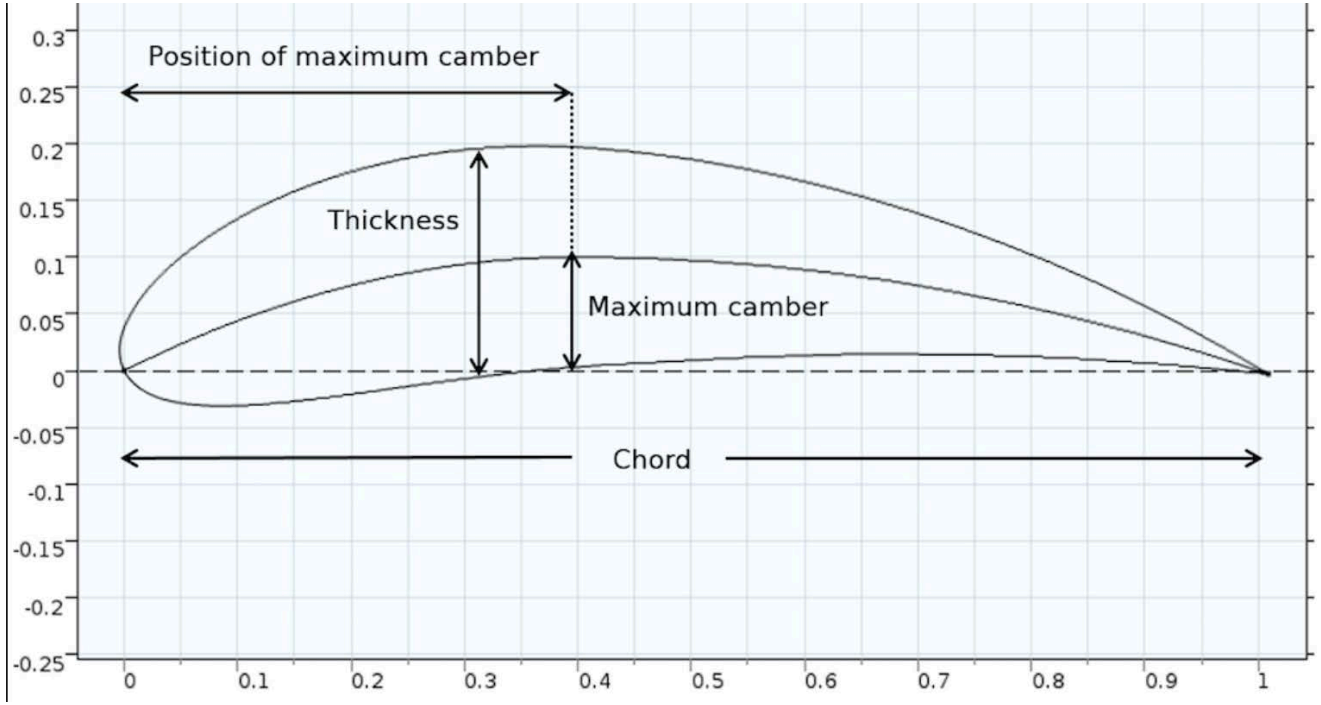
Our optimization problem is then:

$$\max V(x)$$
$$0 \leq x \leq 7.5$$

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

Optimization problems arise practically everywhere





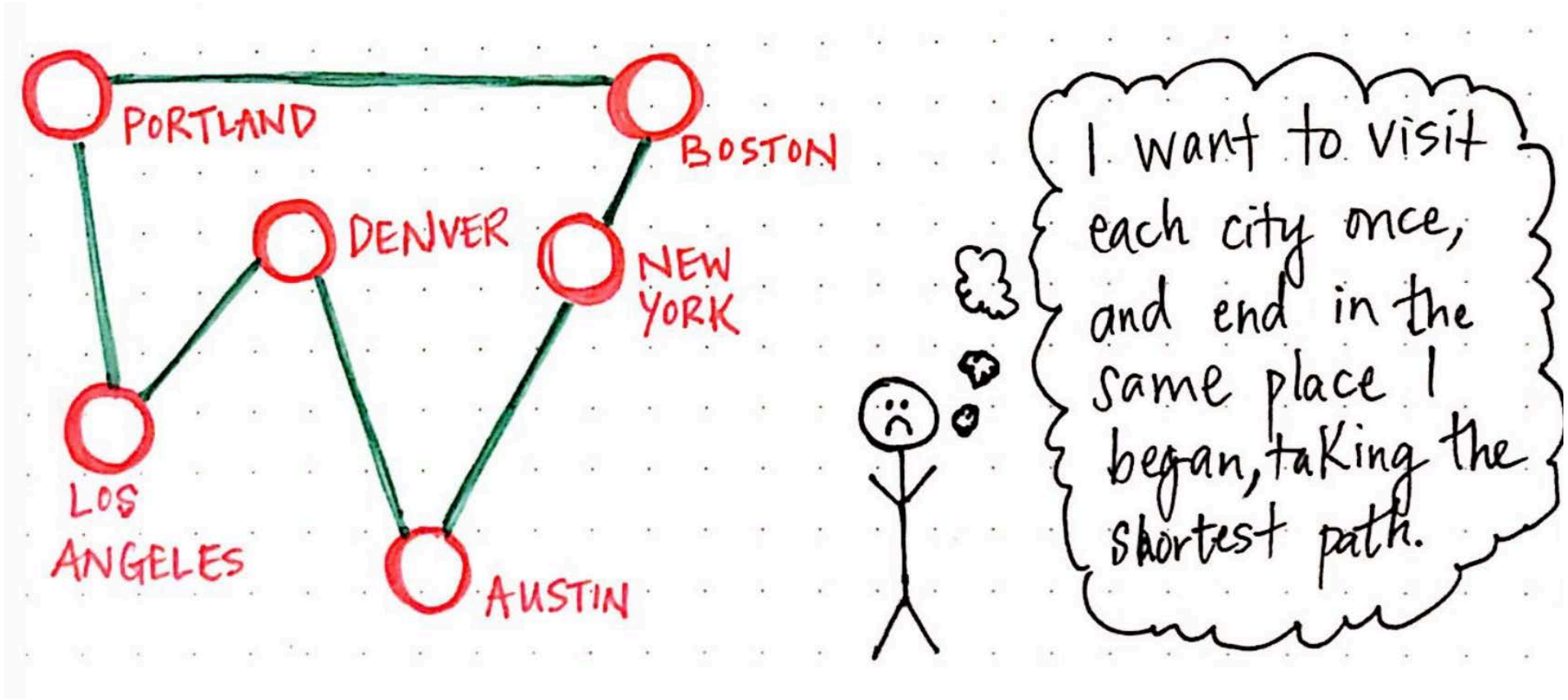
Structural optimization:

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

Airfoil design:

- Objective: maximize lift, minimize drag
- Parameters: see figure

Optimization problems arise practically everywhere



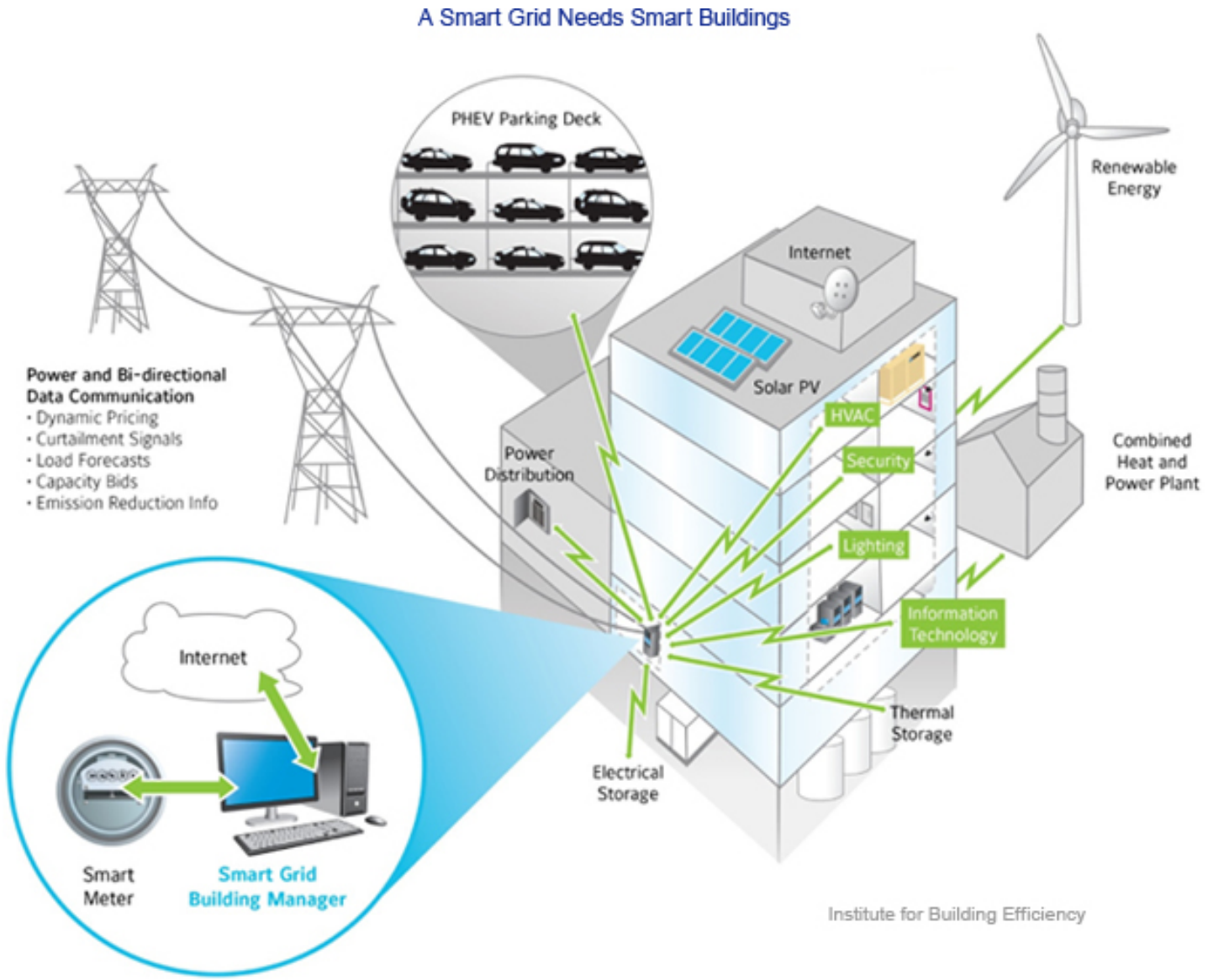
Traveling salesman problem:

- Objective: minimize the distance traveled
- Parameters: which links to traverse
- Constraints: each city must be visited

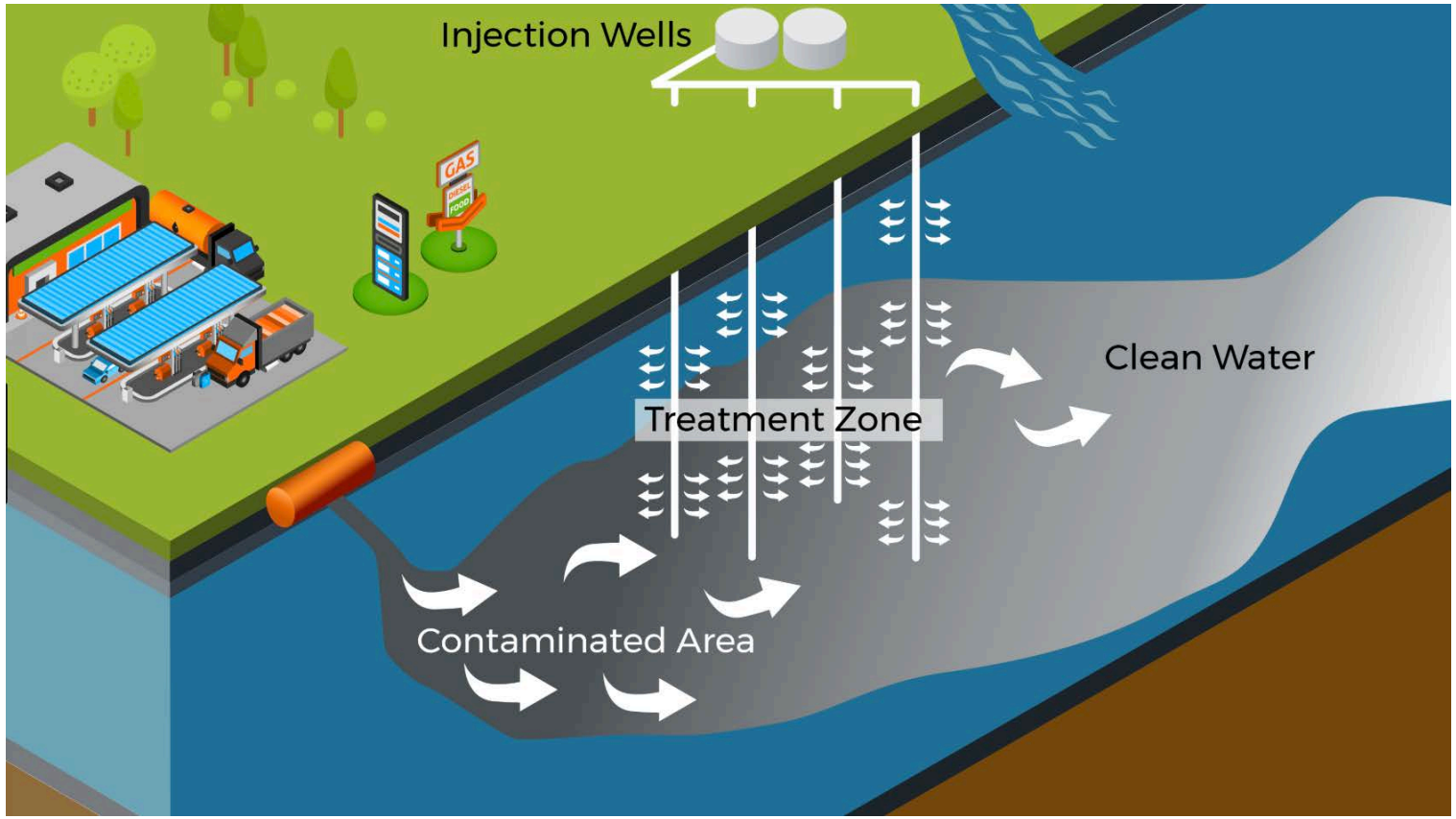
Optimization problems arise practically everywhere

Design of smart buildings:

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



Optimization problems arise practically everywhere



Groundwater remediation:

- Objective: minimize cleanup cost
- Parameters: 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 parameters 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

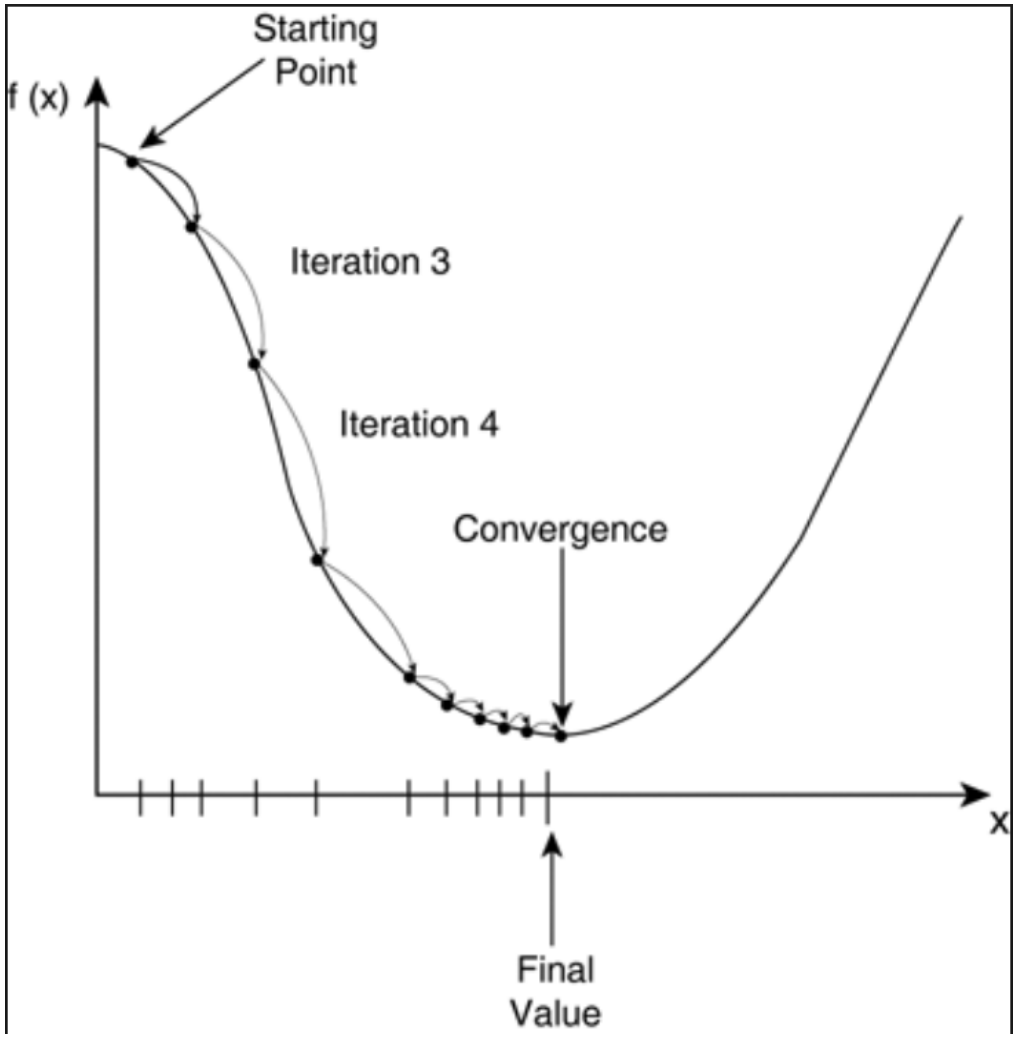
Optimization methods are developed for specific types of problems, for example:

- *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

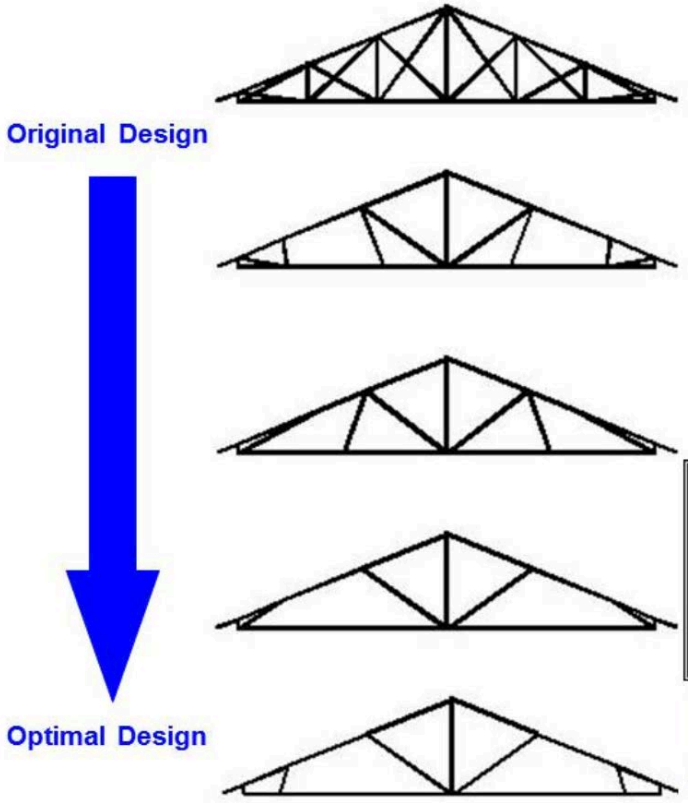
Selection of the optimization method requires a good understanding of the problem at hand

Optimization algorithms are iterative methods

Gradient based methods go downhill (minimization):



Several evaluations of $f(x)$ are required 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:

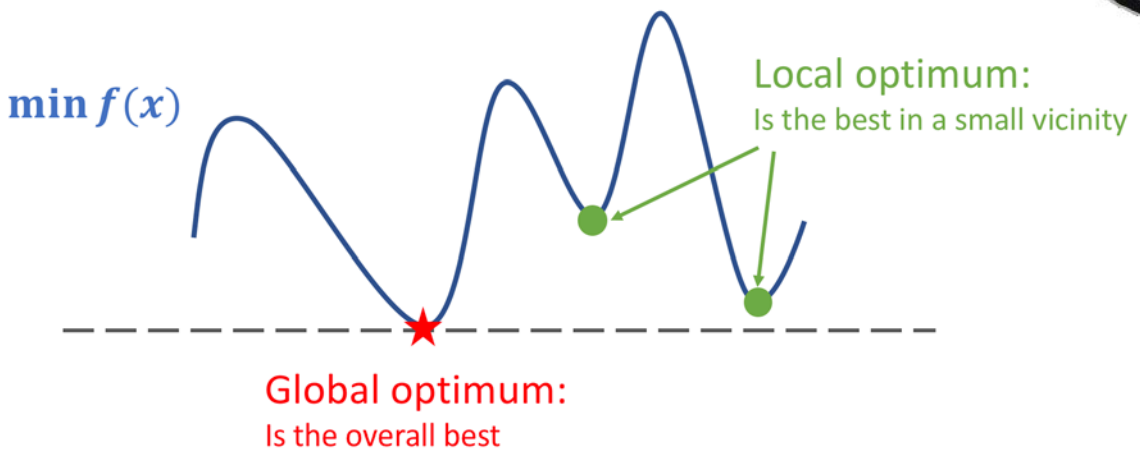
- The objective function involves a computer simulation
  - No analytic description of the objective function (black-box)
  - No gradient information
- The simulation is time consuming

- 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

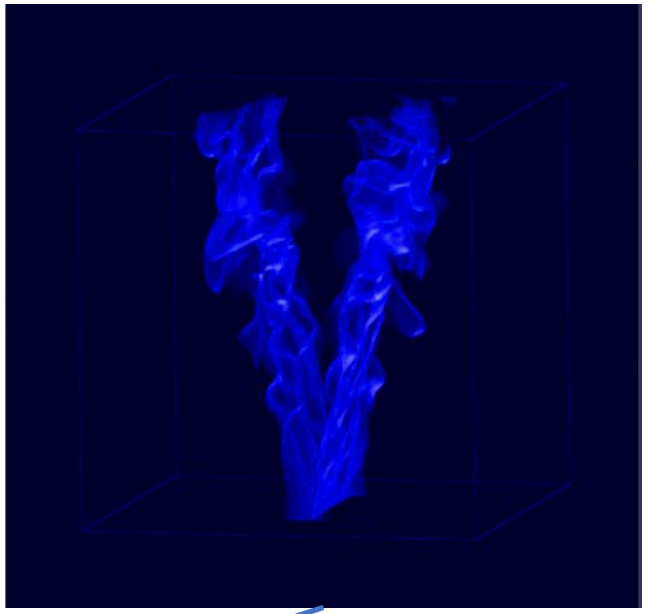


Goal: Find the global optimum within as few evaluations of $f(x)$ as possible

Example applications arise in numerous science areas...

Practically anytime a simulation is involved

Laboratory observation: rod stabilized flame



Computer simulation

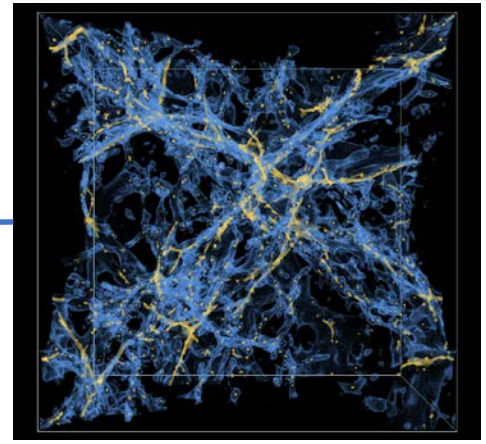
Compare

- "Tweak" (optimize) the simulation's parameters *such that the error* between observed and simulated data *is minimized*

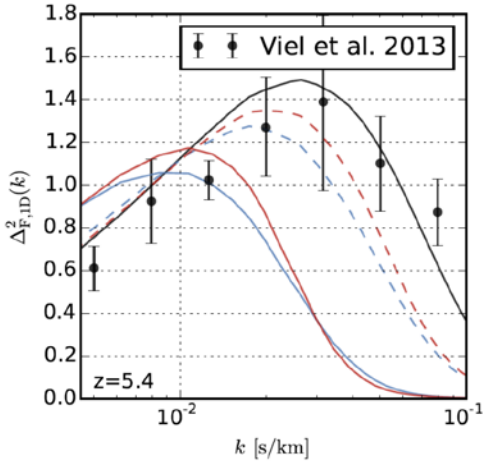
Example applications arise in numerous science areas...

Practically anytime a simulation is involved

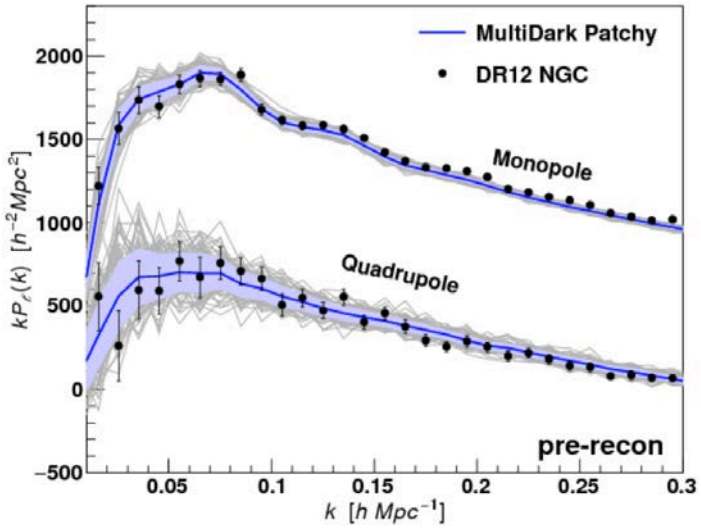
Cosmology simulation



Compute



Power spectrum observation data



Compare

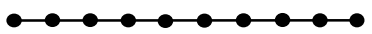
- Infer the parameters of the cosmology simulation that most likely explain the observation data

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

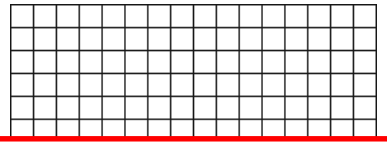
Parameter sweeps (also grid sampling)

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

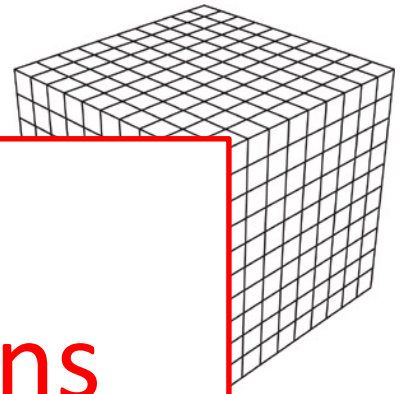
In one dimension



In two dimensions



In three dimensions



Don’t use this for expensive evaluations

What’s the problem with

- It does not scale well:
  - In one dimension, with 10 points → 10 evaluations
  - In two dimensions, with 10 points each dimension → 10x10 evaluations
  - In d dimensions, with 10 points each dimension → 10^d points
  - For example, 10 dimensions = 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 parameter space and evaluate
- Choose the best point as your optimum

Drawback:

- *How many points* do you evaluate*? How do you select* these points*?*
- *No mechanism to sample interesting regions of the parameter domain more thoroughly*

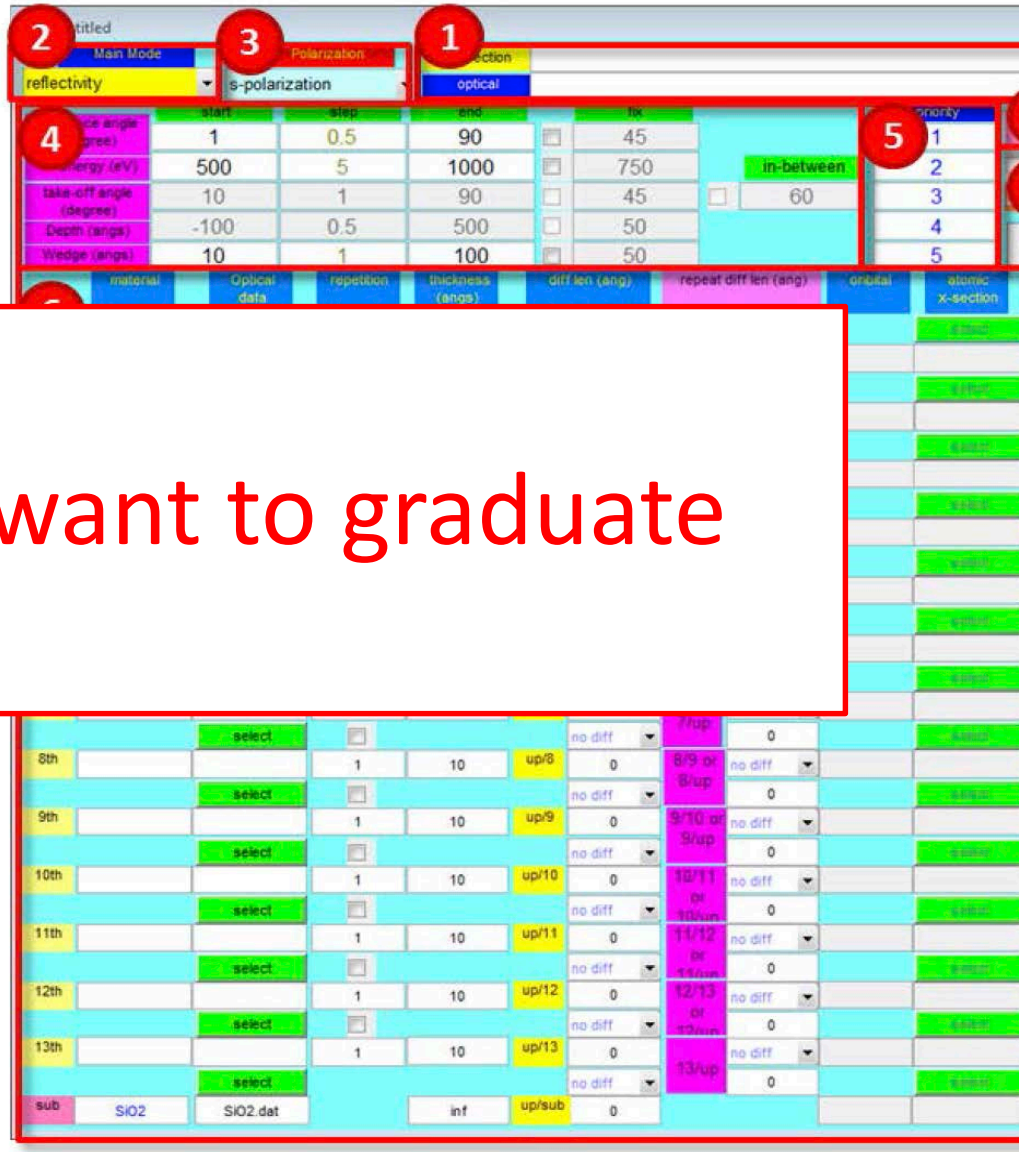
Don’t use this for optimization

Tuning “by hand” and using intuition

- Manually adjust parameter values
- Human is often the slowest element in optimizat
- Humans a dimensio

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

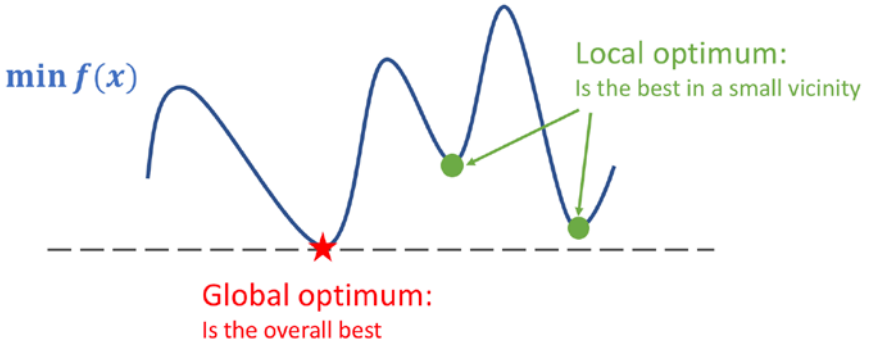
Don’t do this if you ever want to graduate



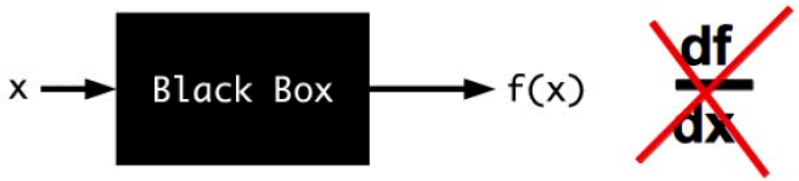
Don’t use optimization methods that were not developed for the type of problem you have

- Heuristics:
  - Too inefficient for expensive simulations

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

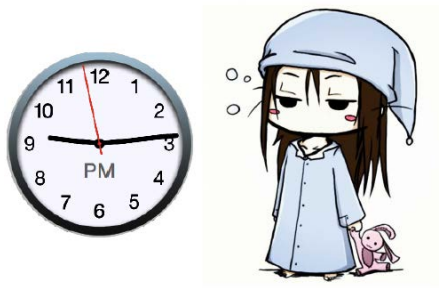


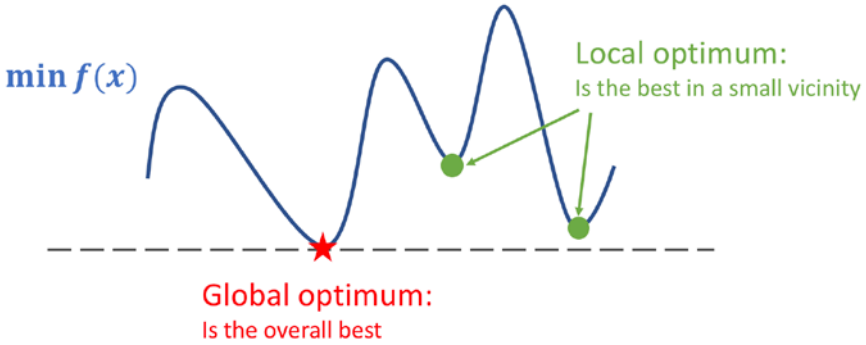
- Methods relying on gradient information:
  - Not useful if you don’t have gradient



<u>Takeaway:</u> It is always worth it to invest time in searching for the right optimization algorithm for your specific problem type

How to solve our difficult optimization problems?

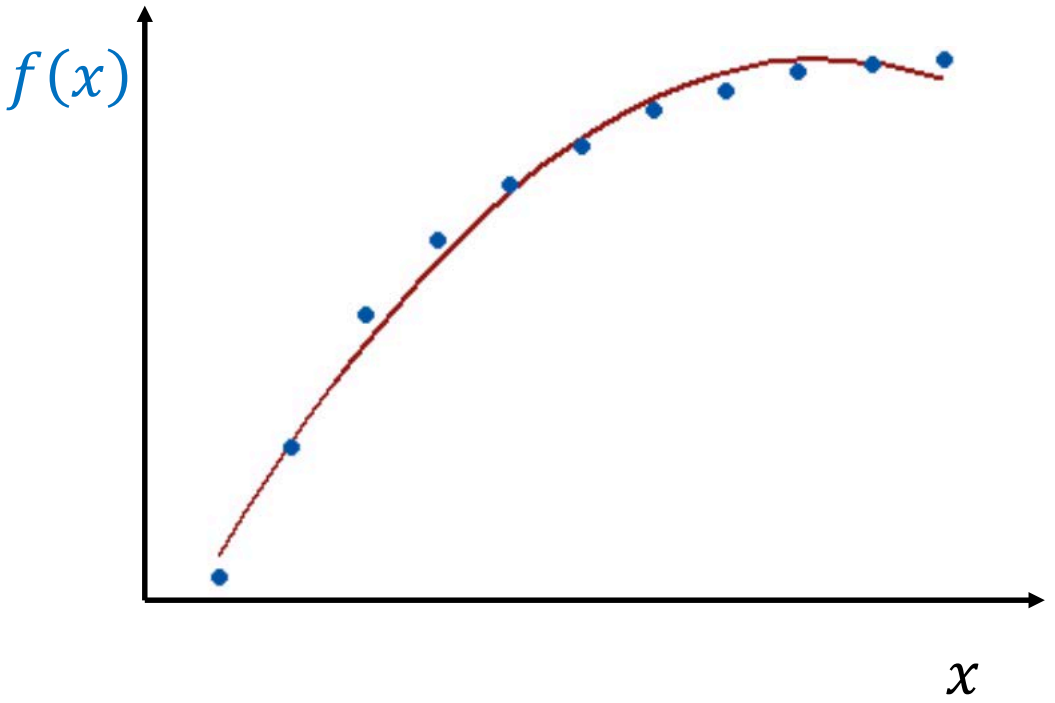




Computationally cheap *surrogate models* to approximate the expensive function:

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

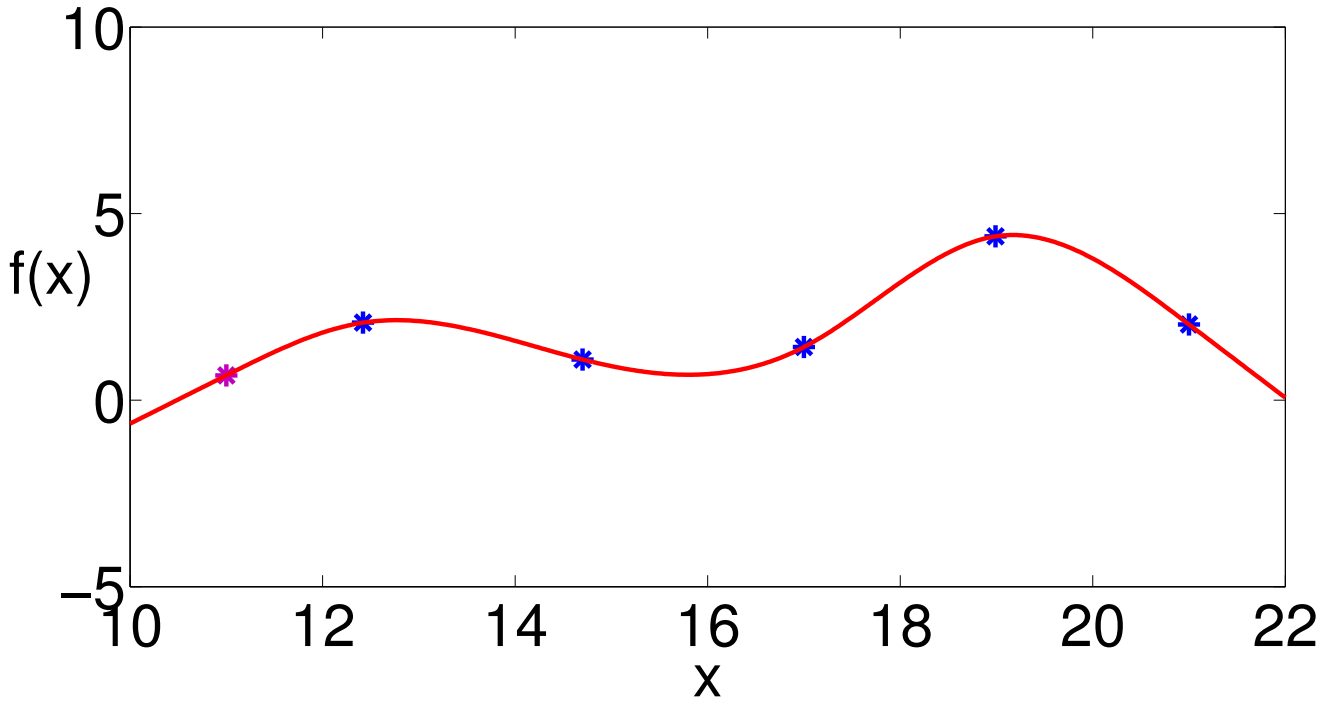
- $s(x)$ could be a
  - *polynomial regression model*



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

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

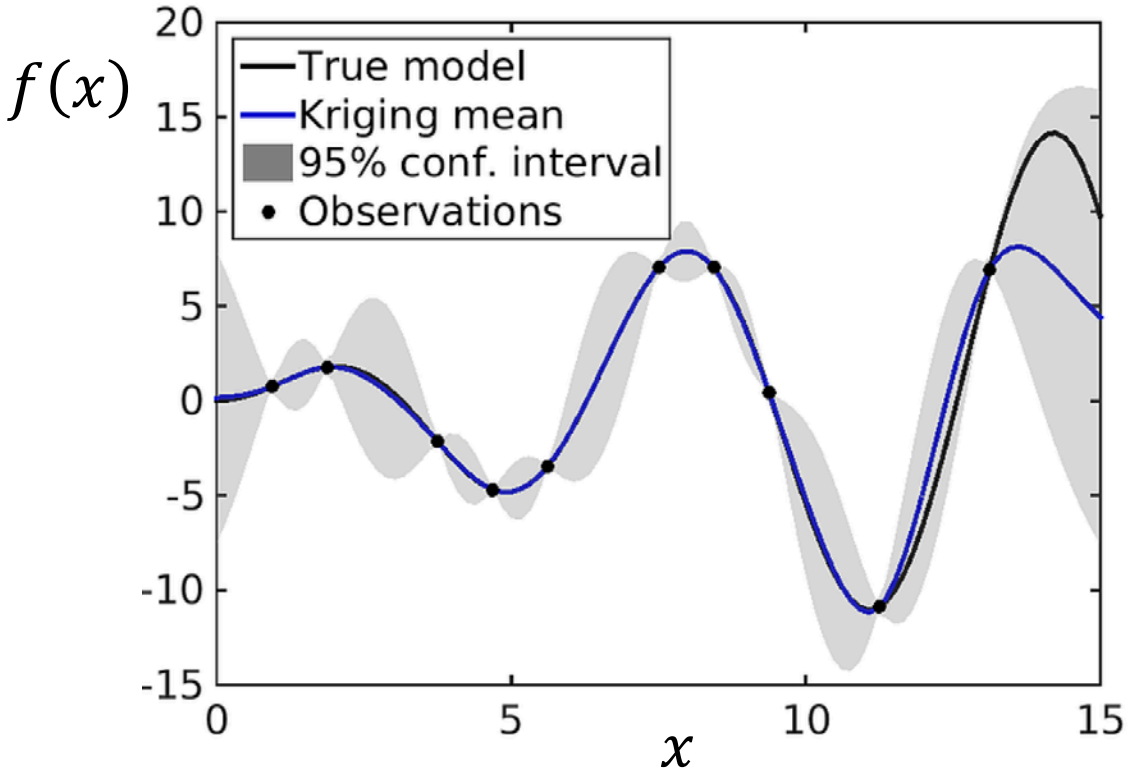
- $s(x)$ could be a
  - polynomial regression model
  - *Radial basis function model*



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

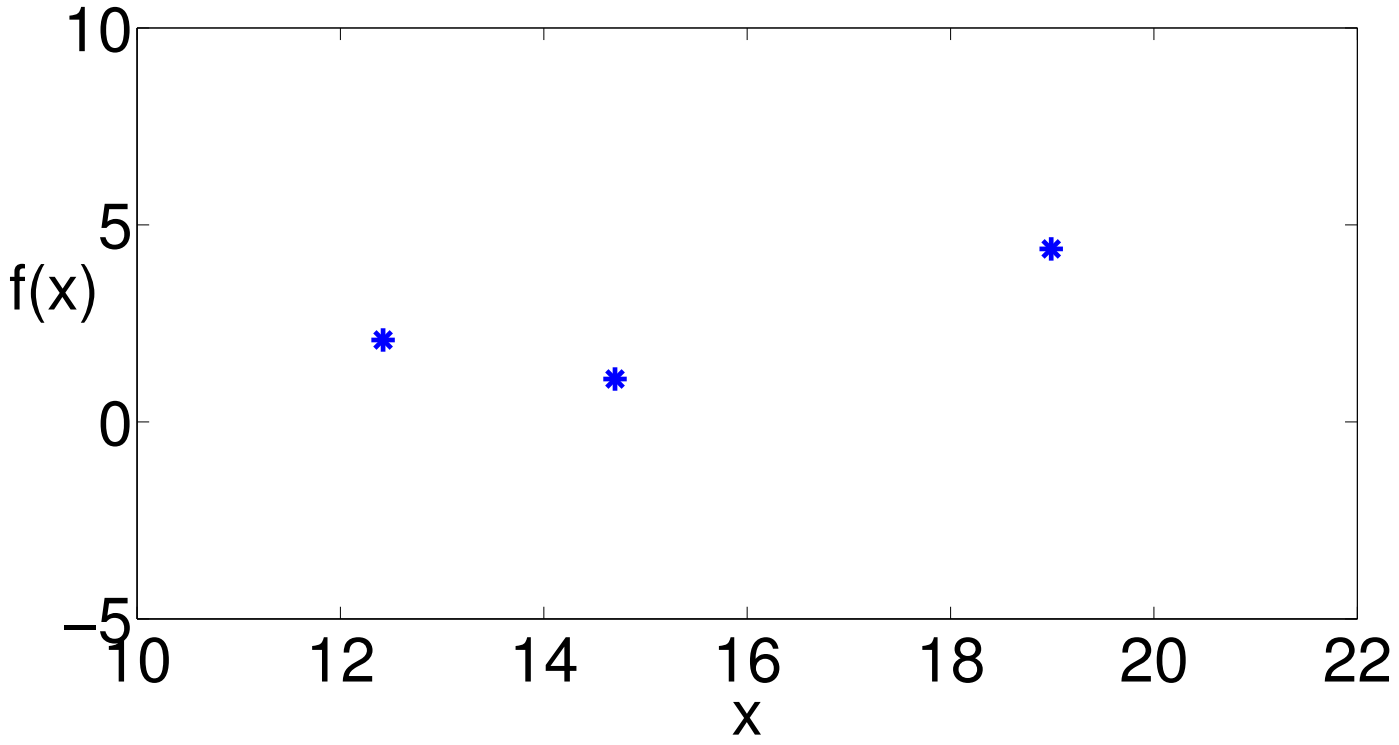
$$f(x) = s(x) + e(x)$$

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



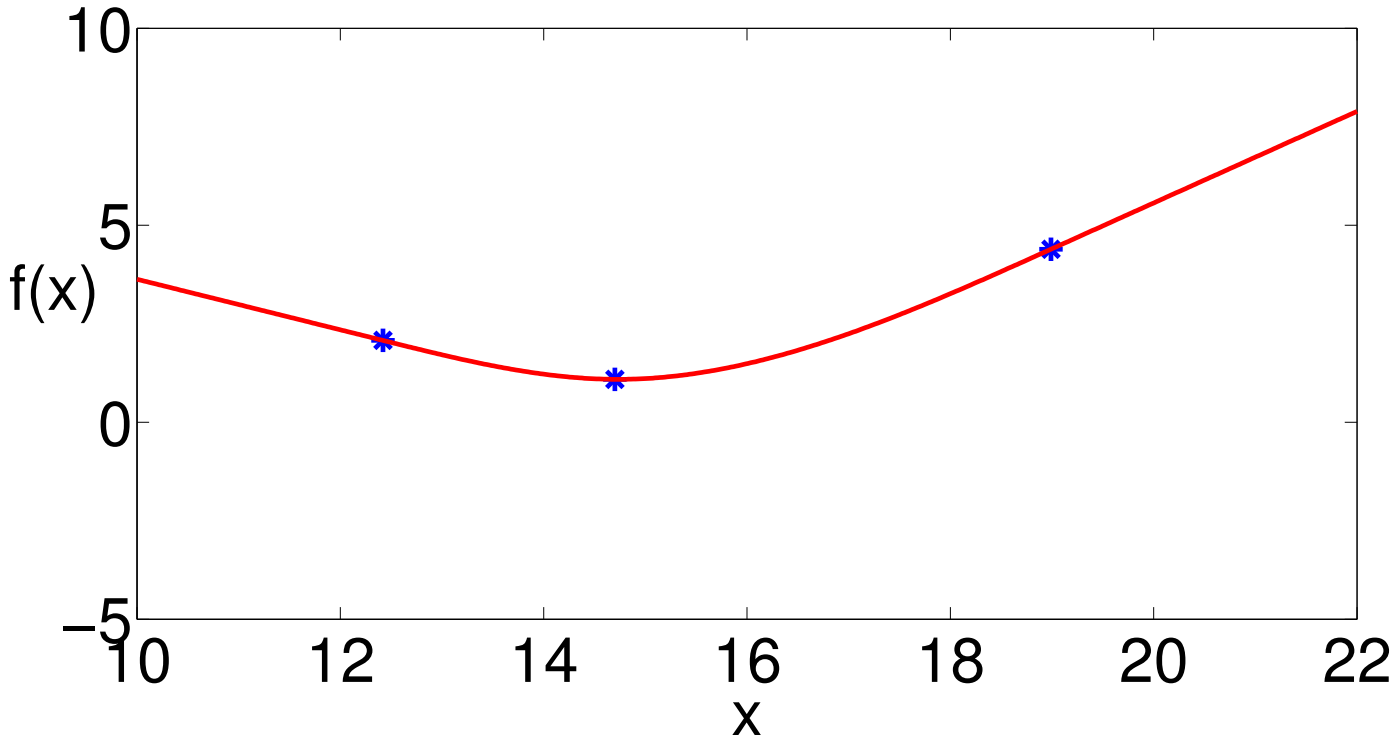
How does it work?

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



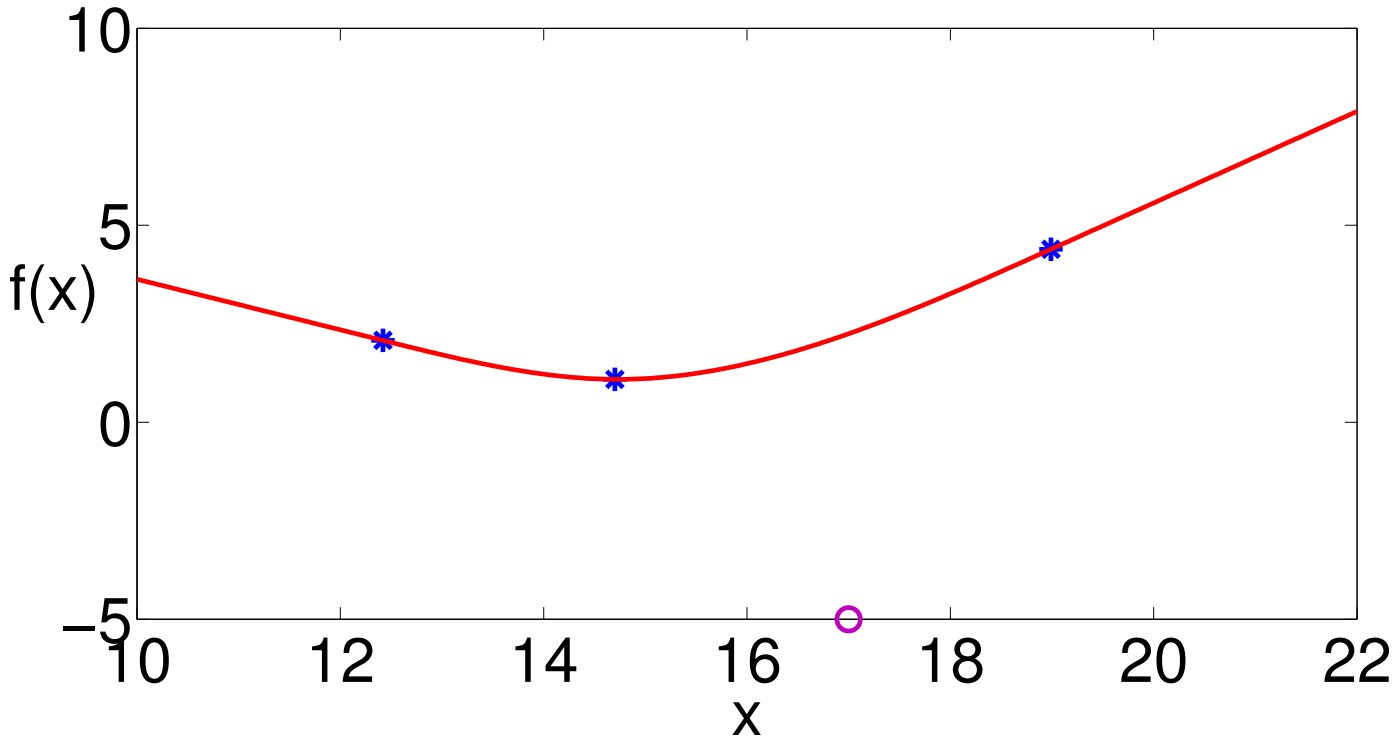
How does it work?

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



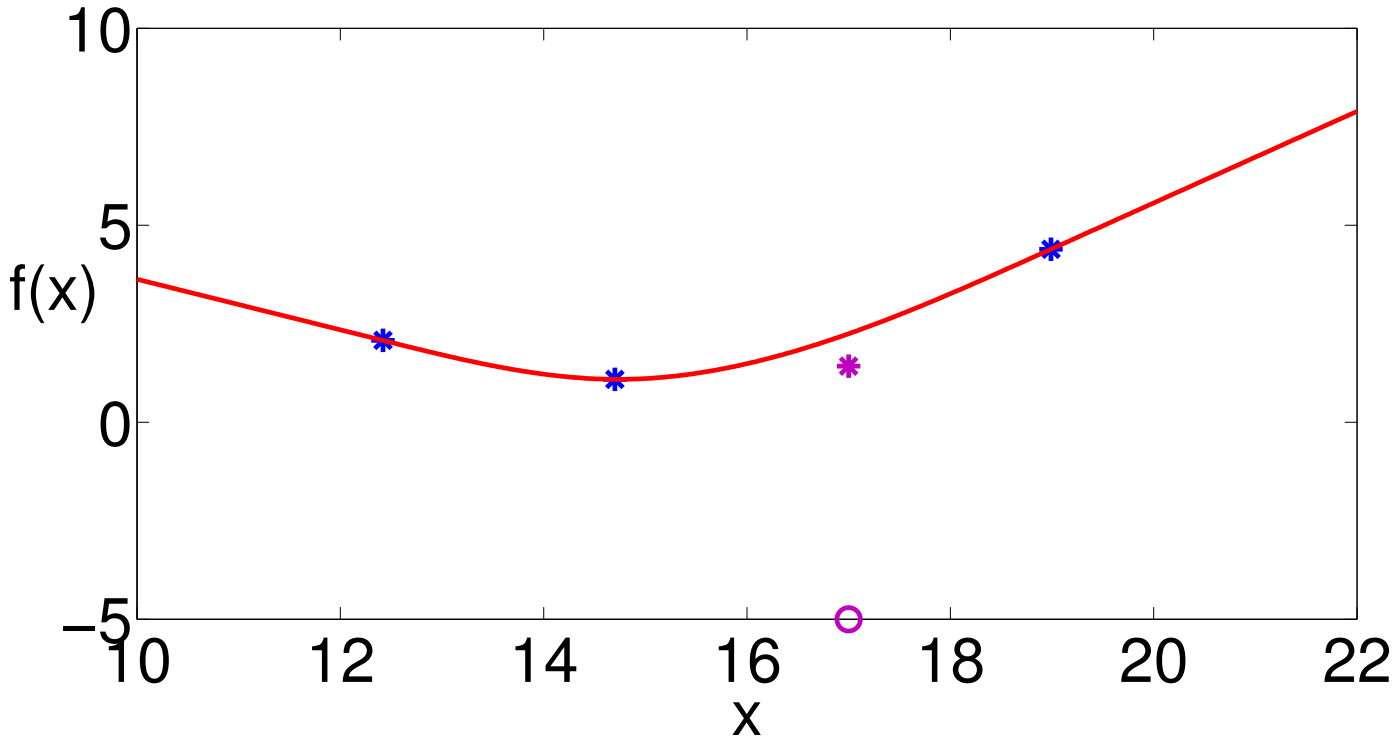
How does it work?

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



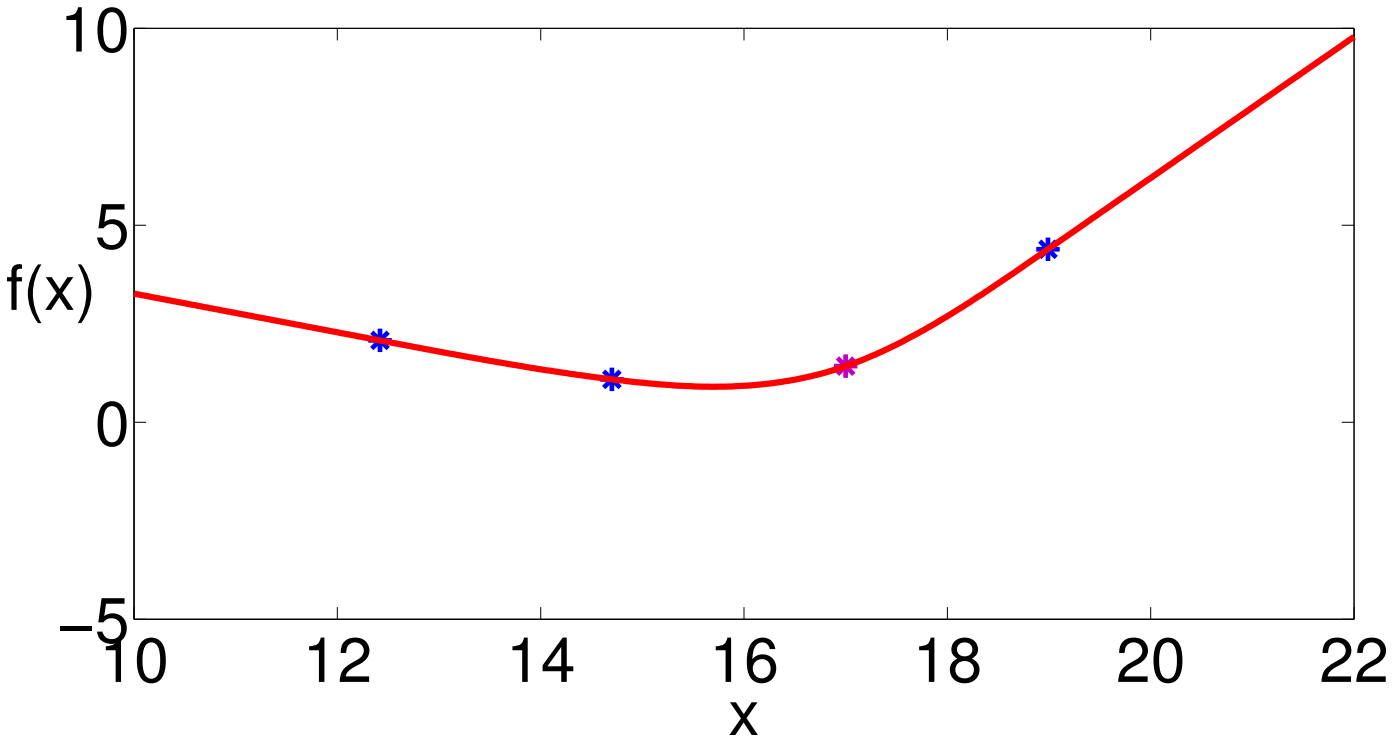
How does it work?

- We evaluate the expensive simulation for the newly chosen parameter value



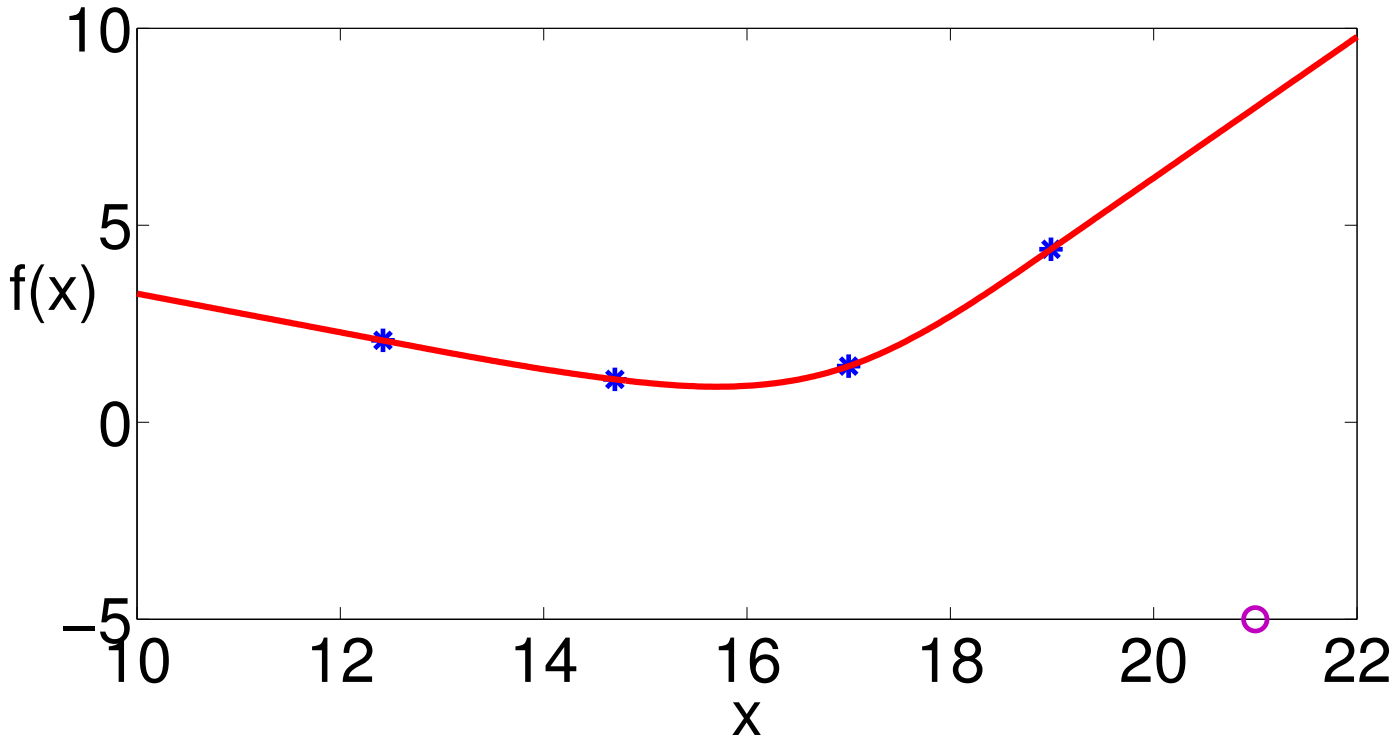
How does it work?

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



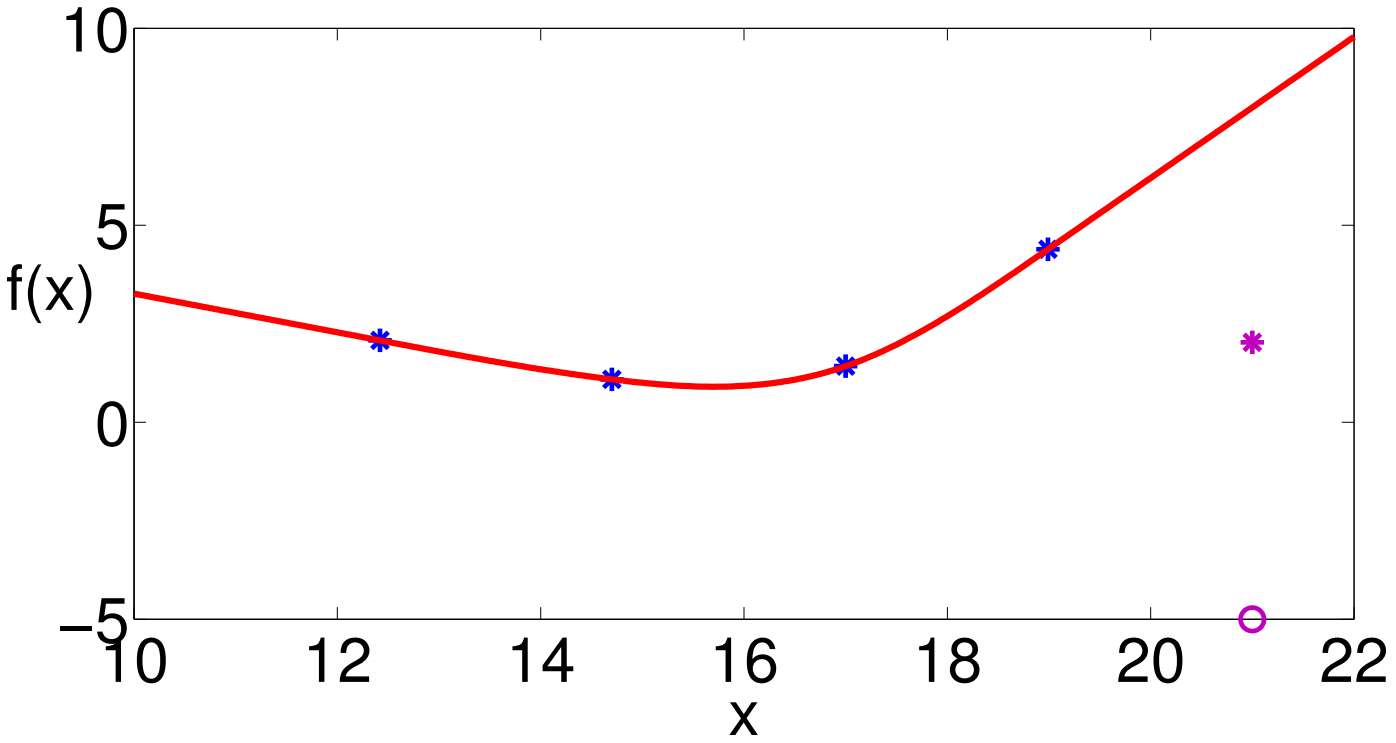
How does it work?

- And then we do the same thing over and over again
- Use the surrogate model to choose a new parameter value



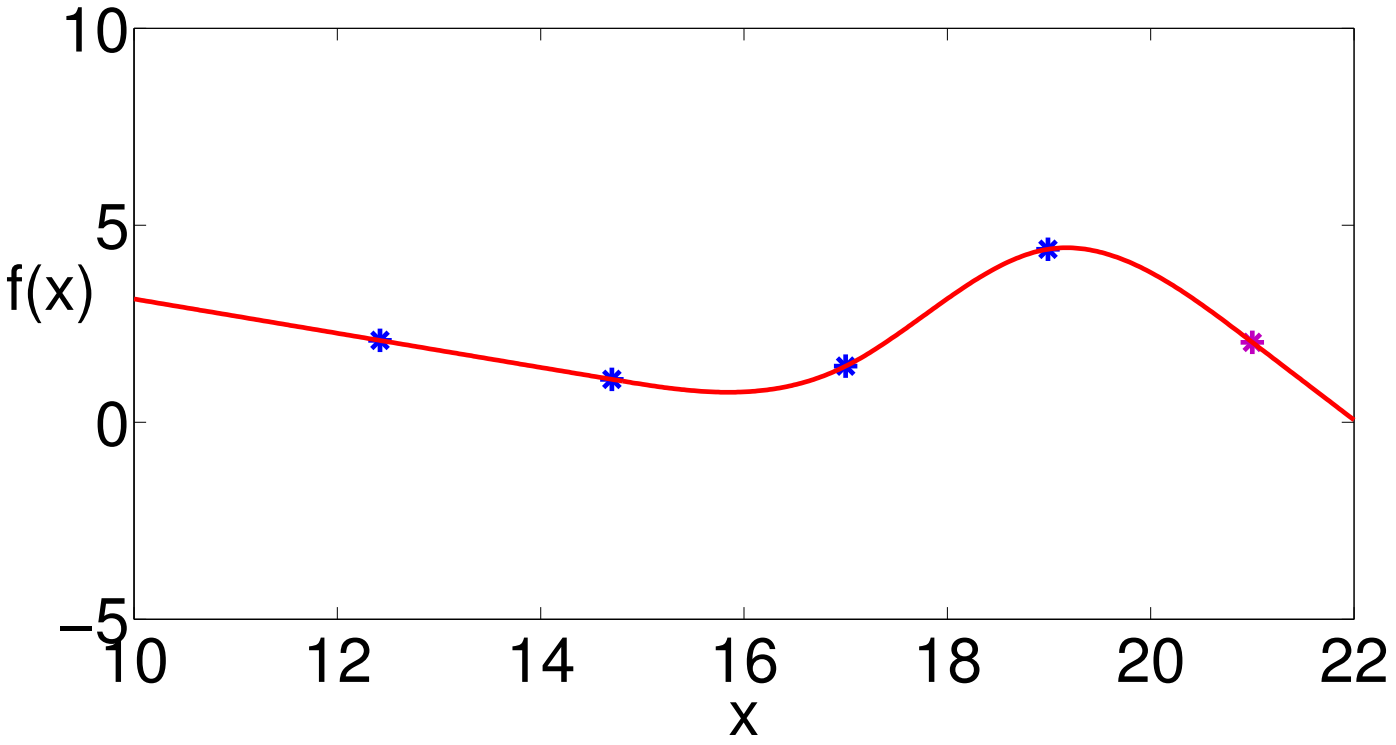
How does it work?

- Evaluate the expensive simulation for the new parameter value



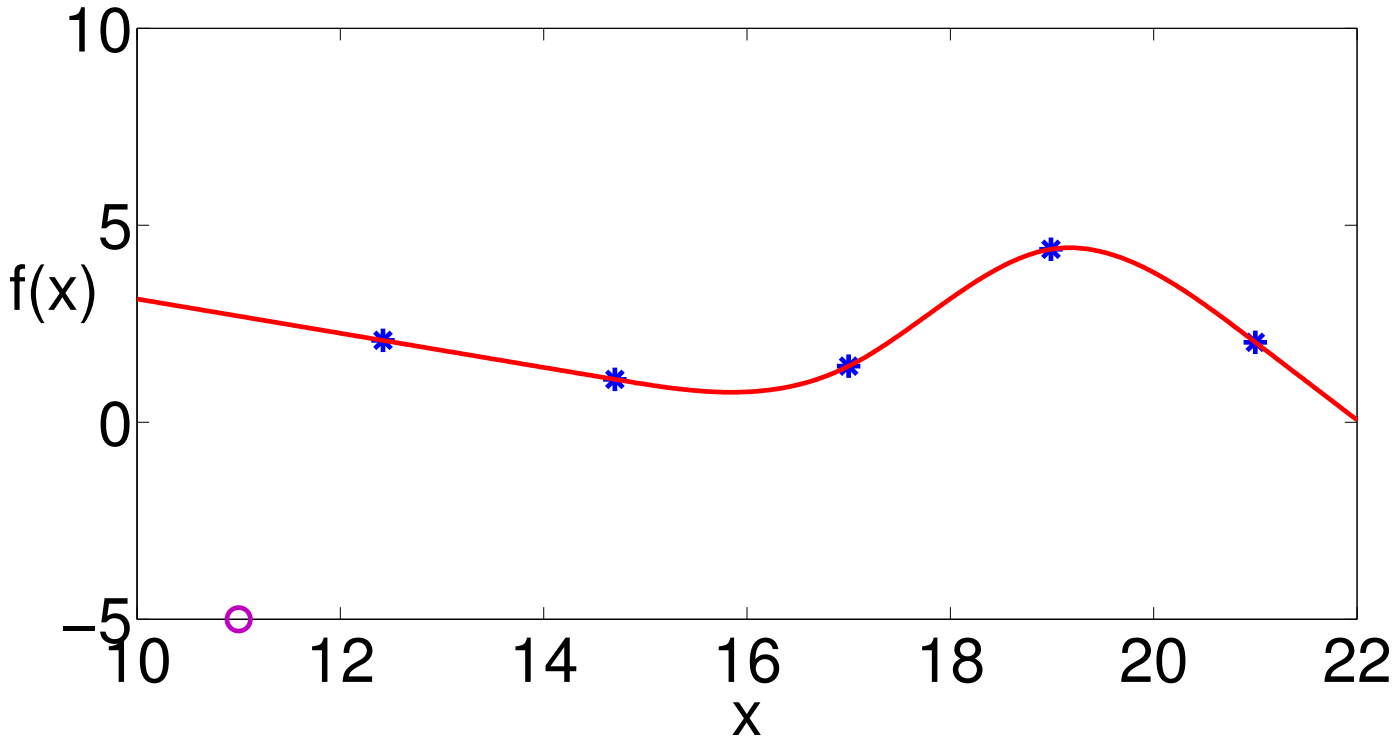
How does it work?

- Update the surrogate model with the new piece of data



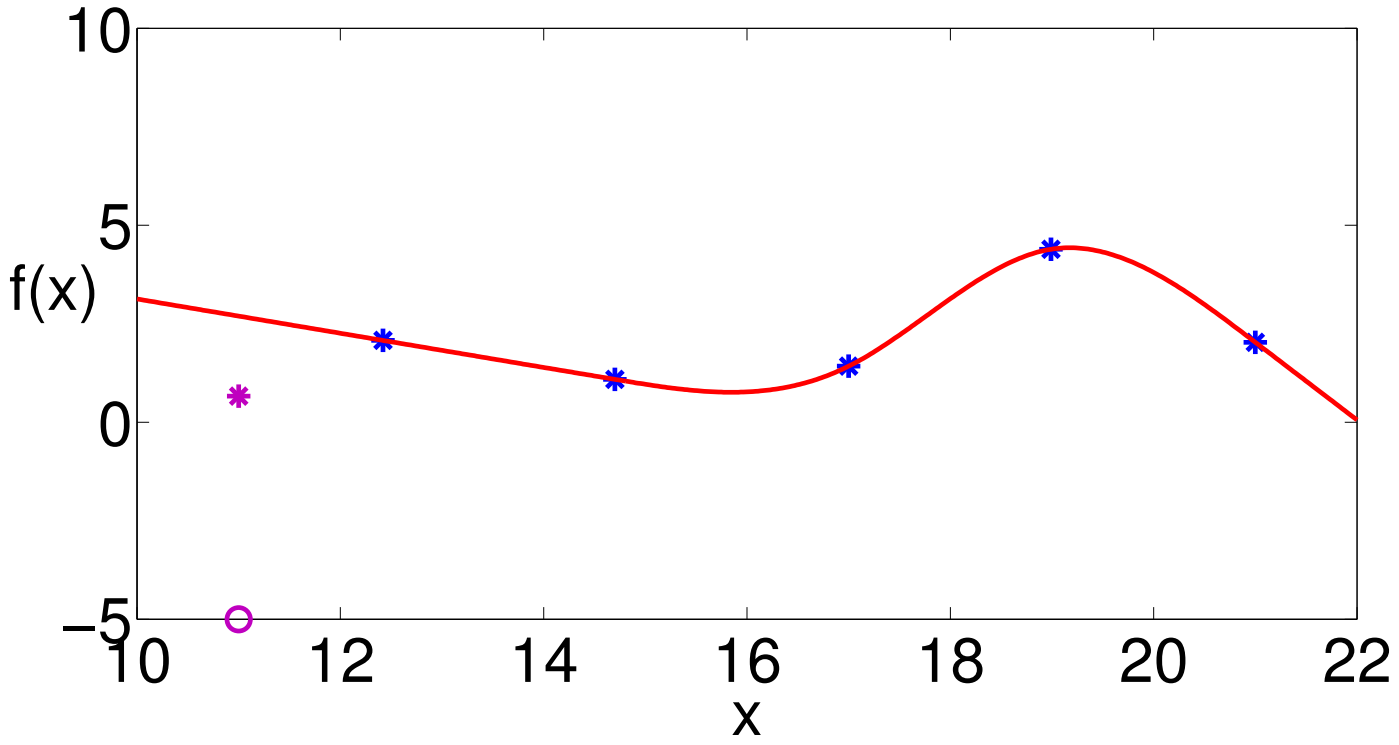
How does it work?

- Use the surrogate model to choose a new parameter value



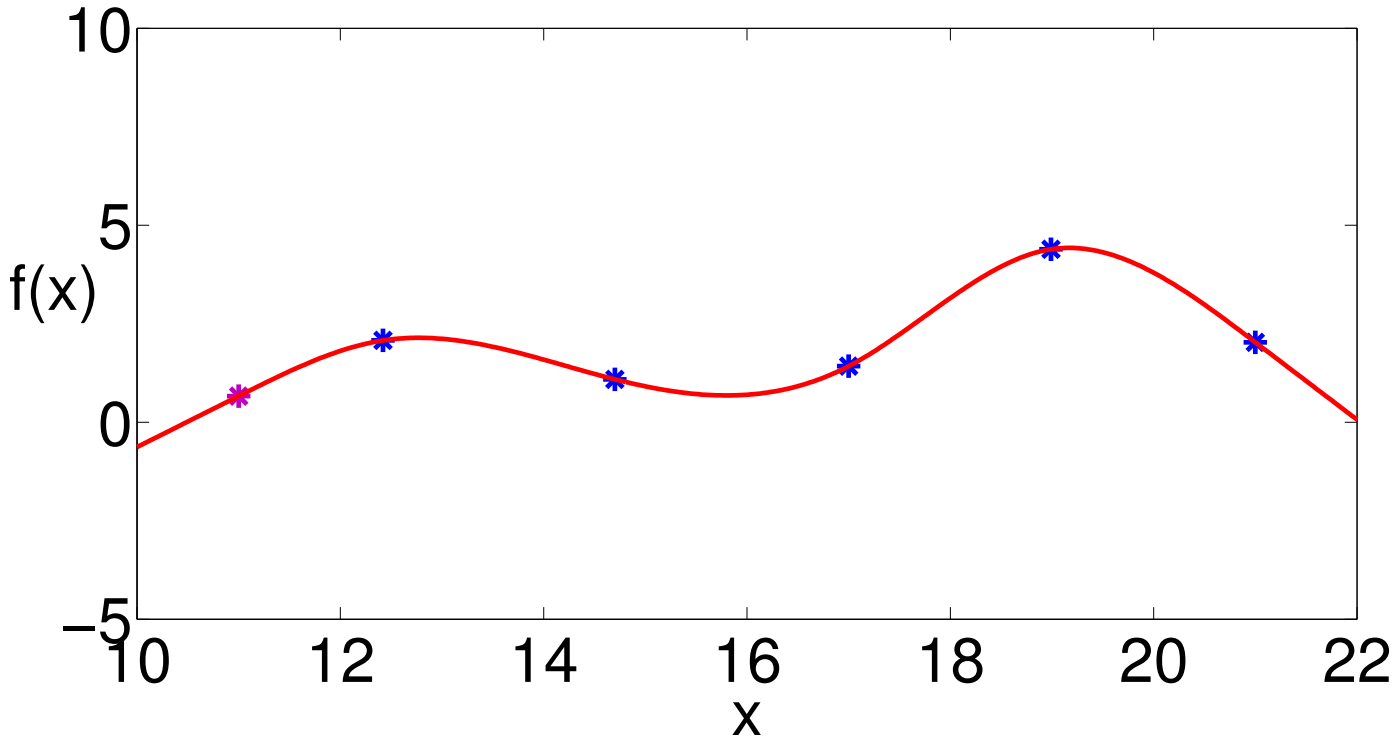
How does it work?

- Evaluate the expensive simulation for the new parameter value



How does it work?

- Update the surrogate model with the new piece of data



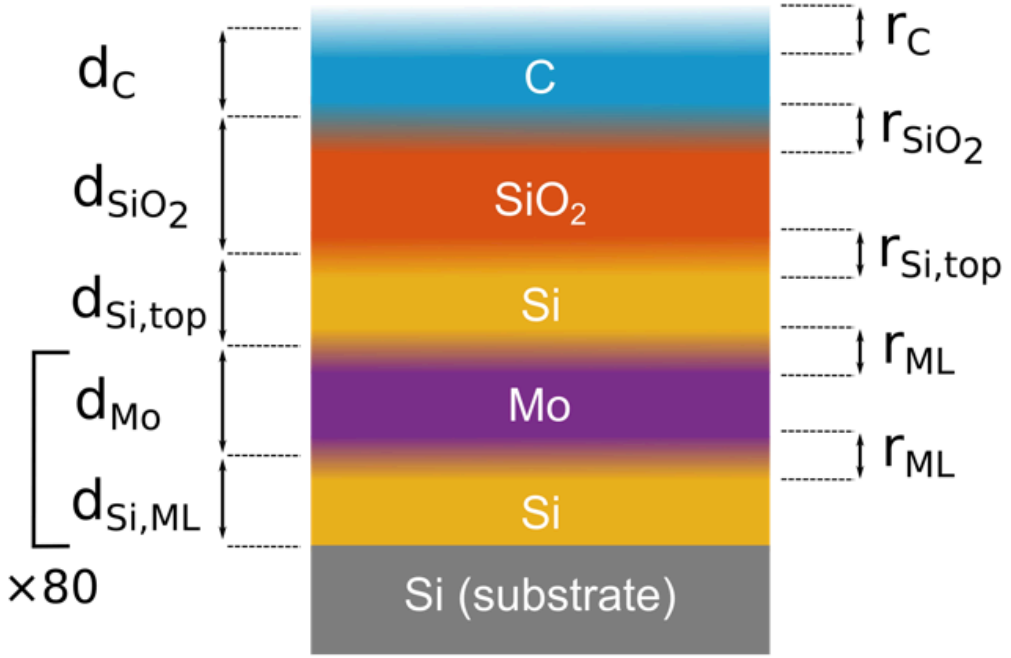
We keep doing this until we run out of time

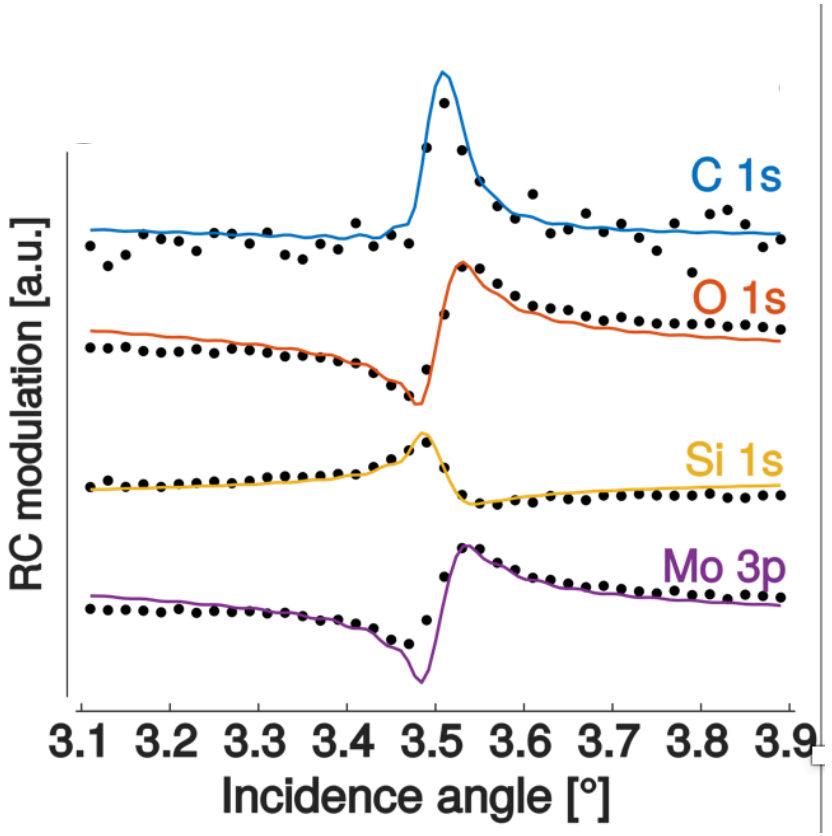
- If we have about 100 hours to find a solution, and one simulation run takes an hour, that means we get at most 100 simulation evaluations done
- The best solution found at the end of the optimization is ”the best we can do with limited resources”

    - Is it globally optimal? – Maybe.
    - Is it locally optimal? – Sometimes.
    - Is it better than what the scientist used before? – Most likely.

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

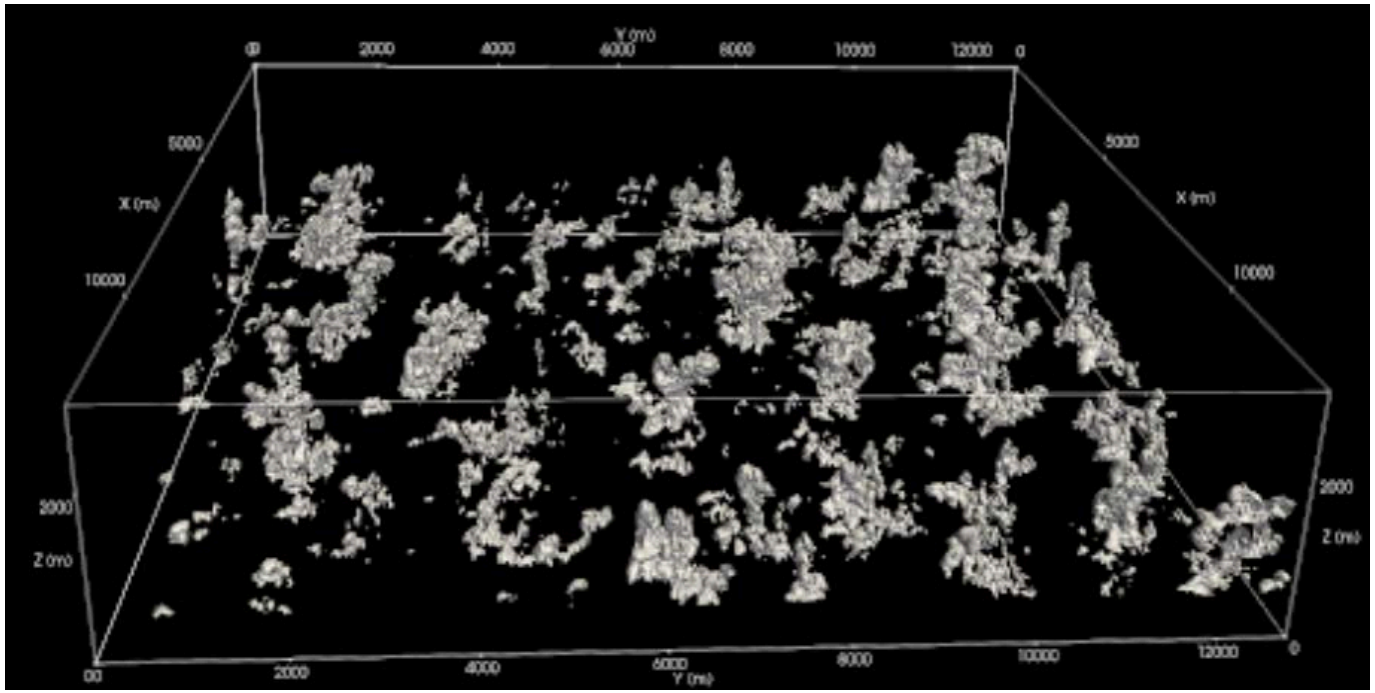
- Use X-ray standing wave to determine the thicknesses of the layers (d's and r's)
- **Minimize errors** between simulation (graphs) to observation (black dots)

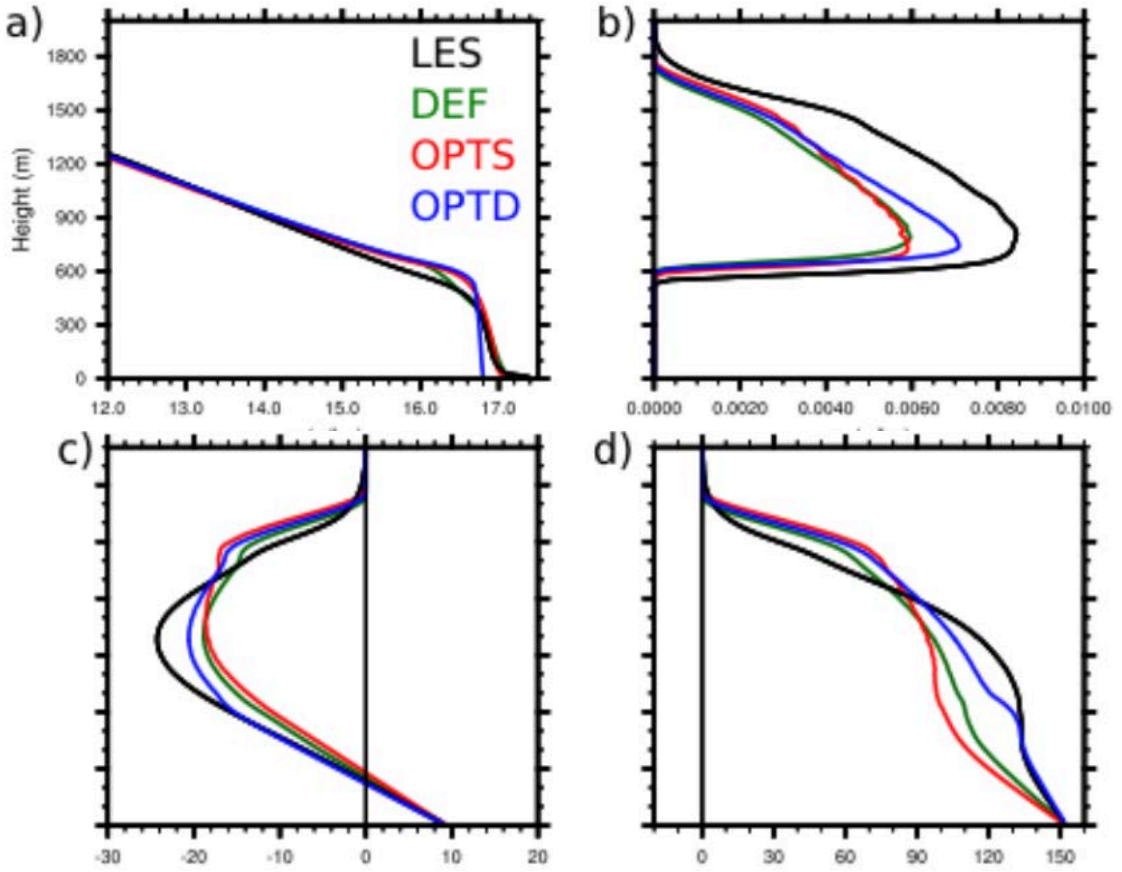




Some examples where we used this method (successfully) – Cloud simulations

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





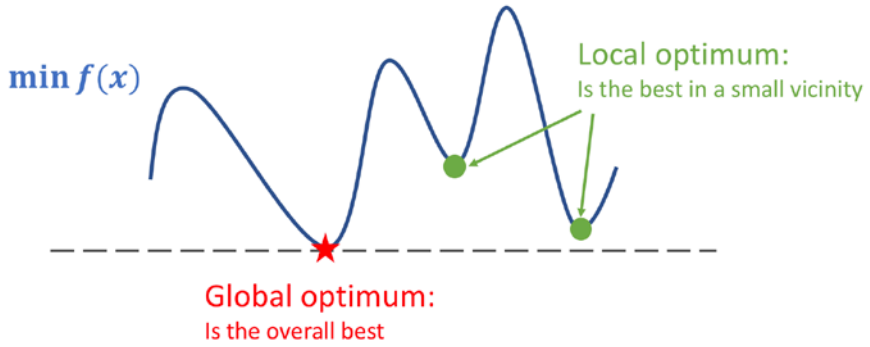
LES = benchmark OPTS, OPTD = optimizations

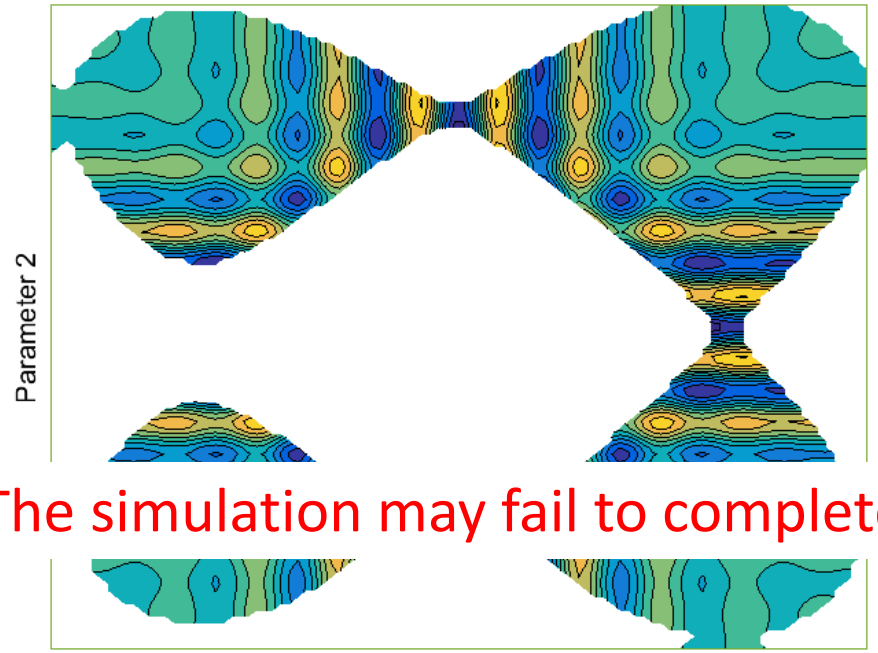
DEF = default

Get as close to the black line as possible

Some examples 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

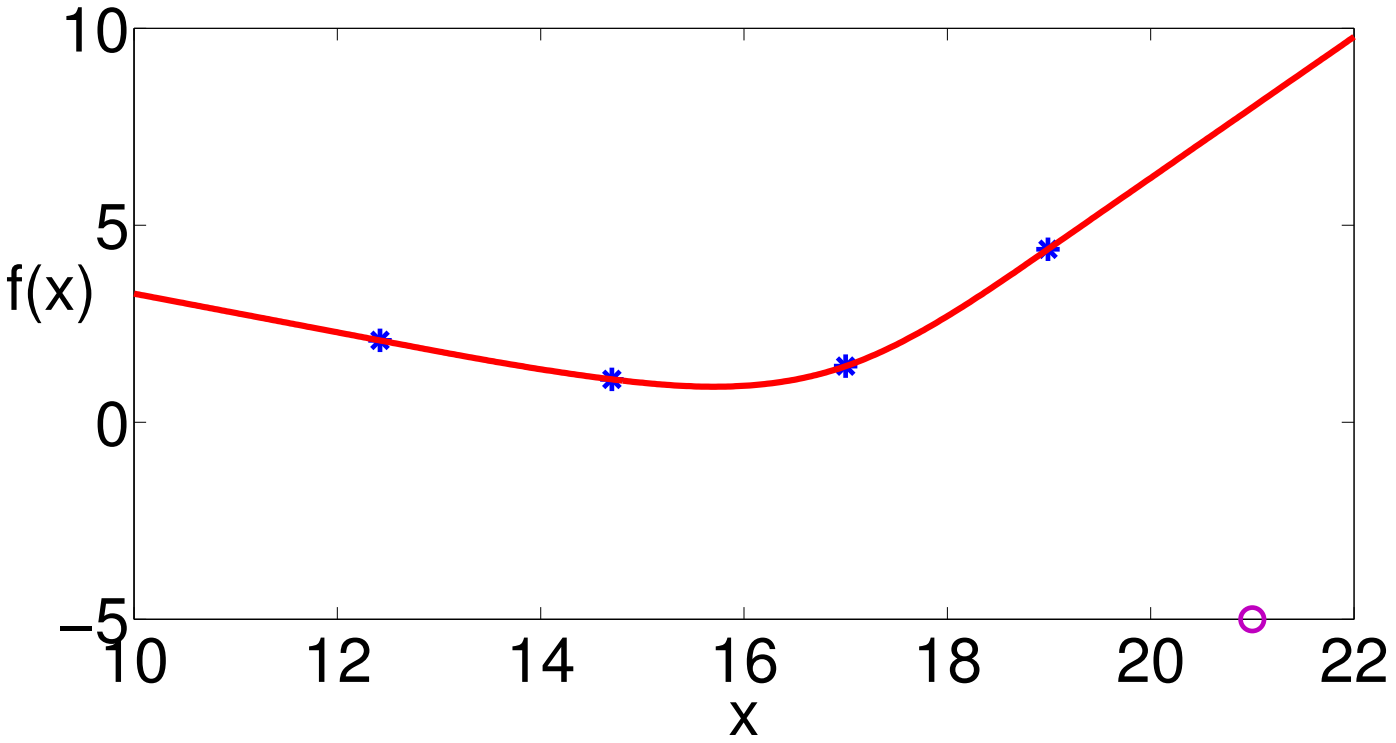




The simulation may fail to complete

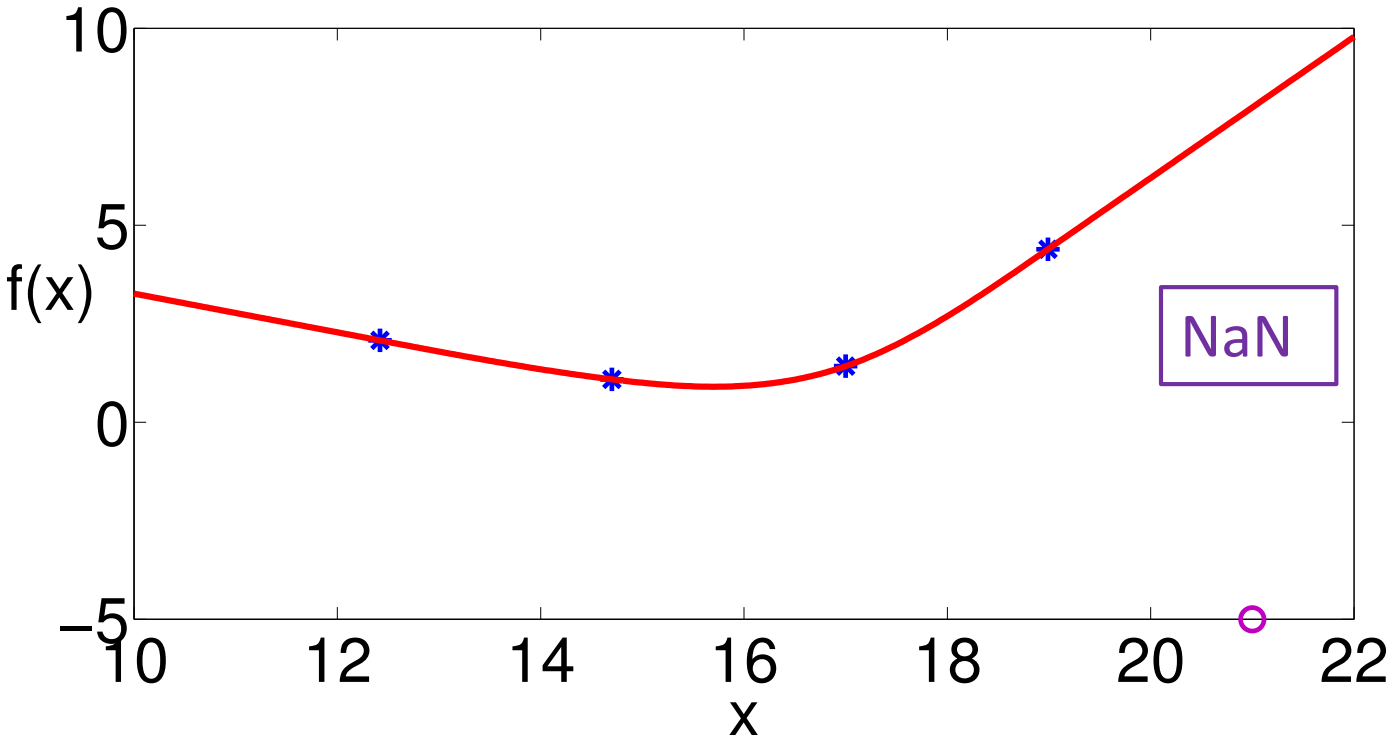
Some examples where we used this method (successfully) – Combustion simulation

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



Some examples where we used this method (successfully) – Combustion simulation

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

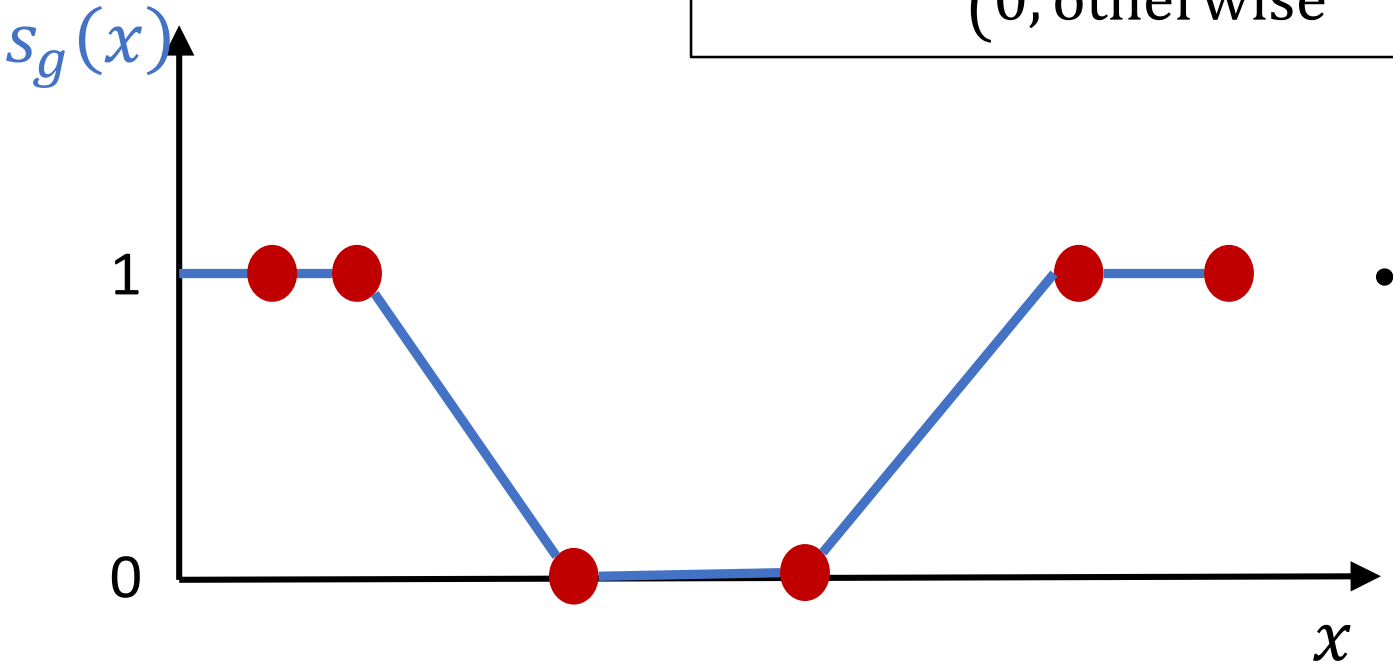


Some examples where we used this method (successfully) – Combustion simulation

- Need some trickery to deal with this:

- Build a function that predicts how likely it is that a parameter value 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

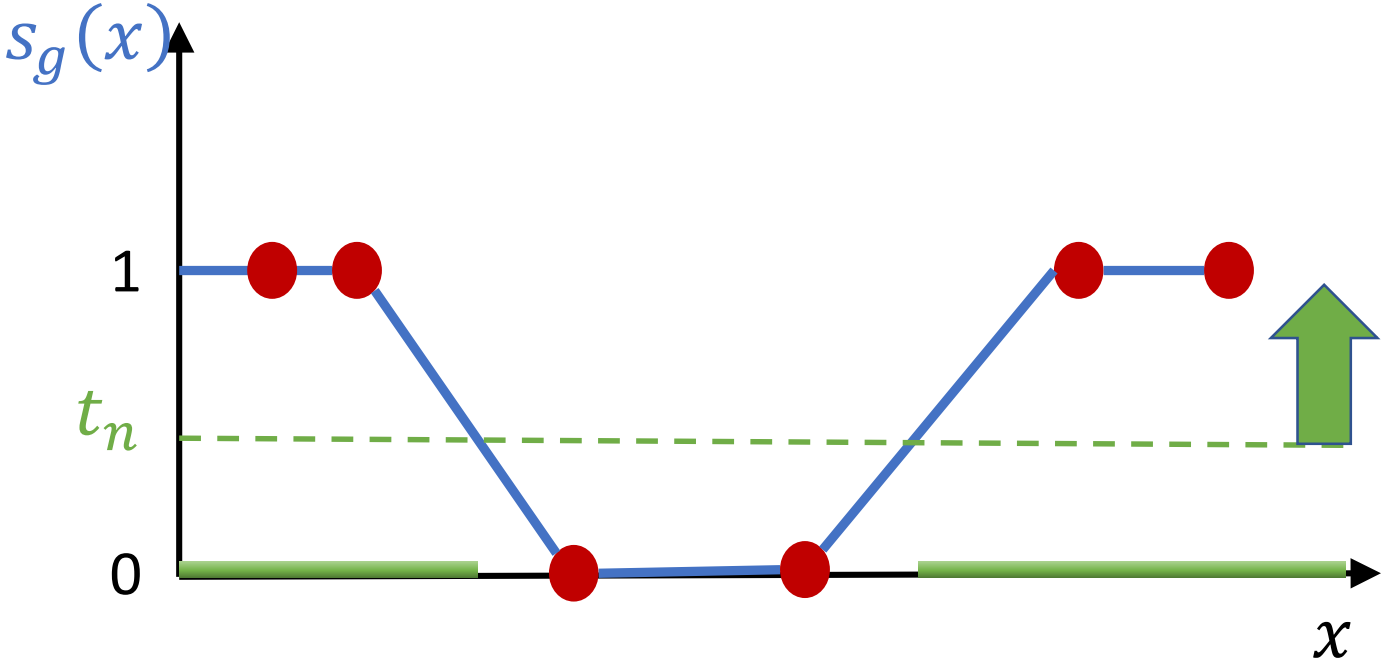
Some examples where we used this method (successfully) – Combustion simulation

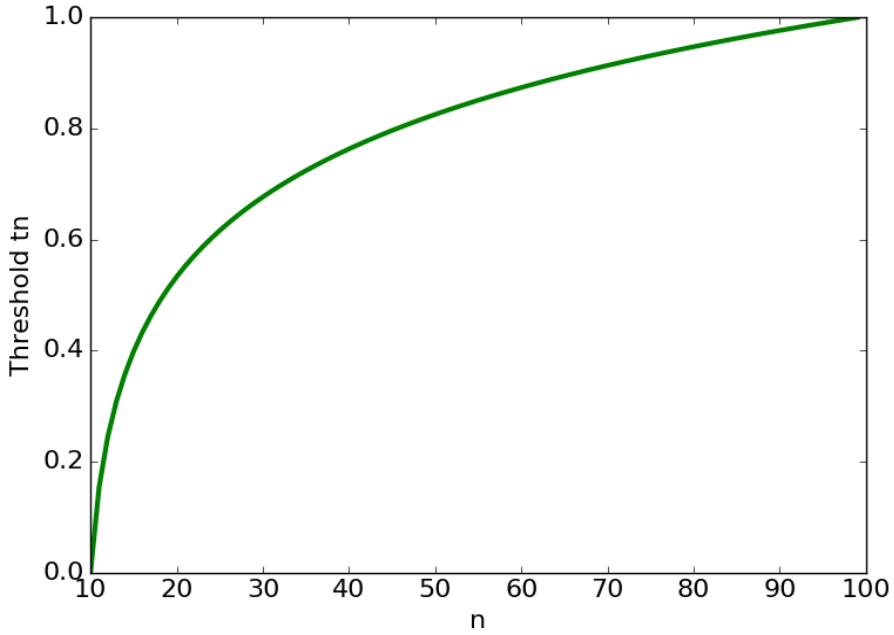
- Need some trickery to deal with this:
- Define a threshold

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

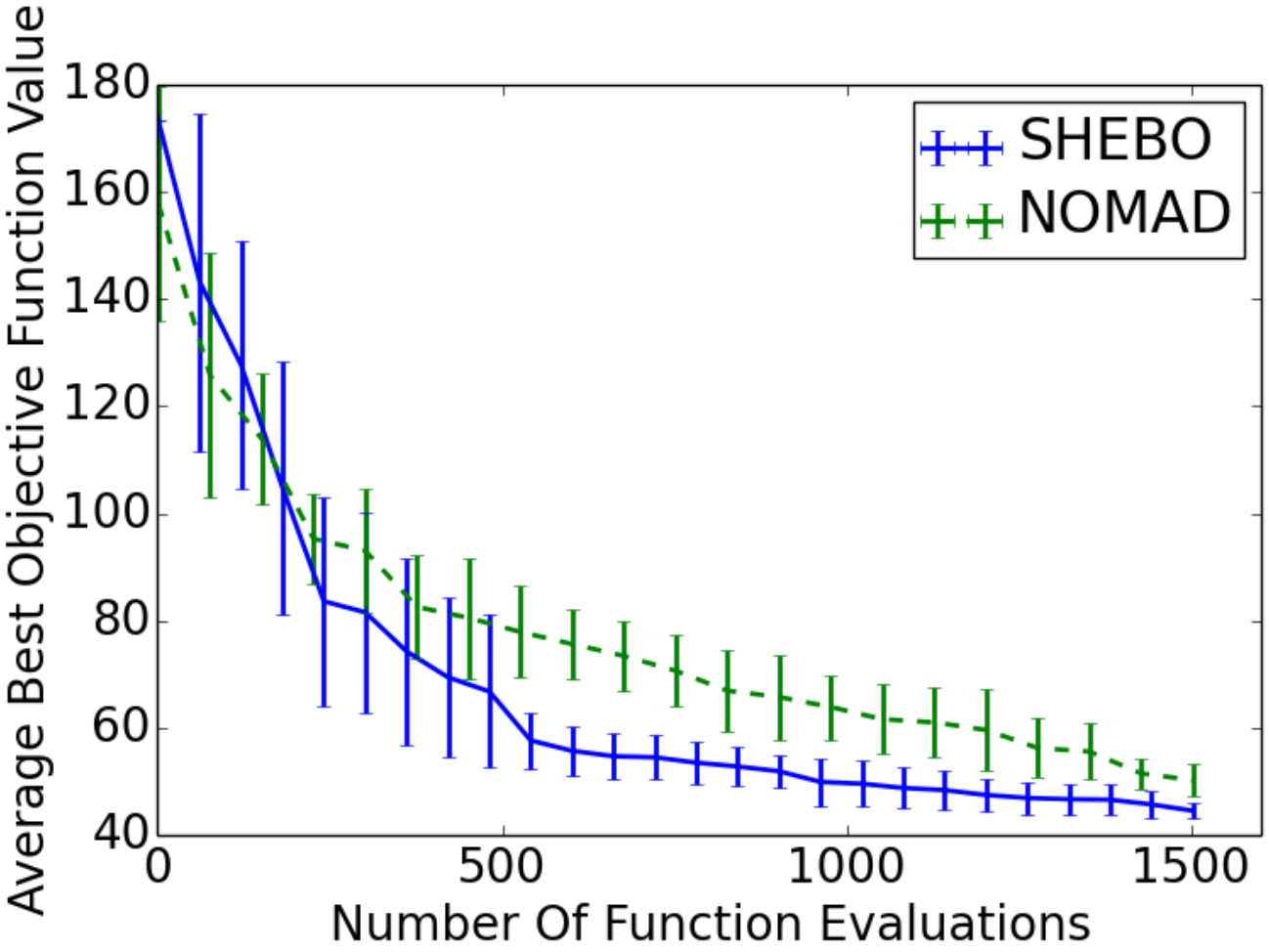
t_n dynamically increases a the number of evaluations n grows

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





Some examples where we used this method (successfully) – Combustion simulation



- 31 optimization parameters
- Objective function = error between simulation and observation

Convergence plots:

- Shows error as it evolves with function evaluations
- Lower graphs are better (minimization)

- More than 50% of the evaluations failed

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
- Physics event generator tuning – 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?