High Performance Computing for Science and Engineering II

18.2.2019 - **Lecture 1: Introduction**

Instructors:

- Prof. Dr. Petros Koumoutsakos
- Dr. Sergio Martin

Teaching Assistants:

- Dr. Georgios Arampatzis (Head TA)
- Dr. Jacopo Canton
- Ermioni Papadopoulou
- Pantelis Vlachas
- Daniel Wälchli

CSElab

Computational Science & Engineering Laboratory

ETH zürich

Today's Class

Course Information:

- Lecture Schedule
- Class Resources
- Grading

Uncertainty Quantification & Optimization

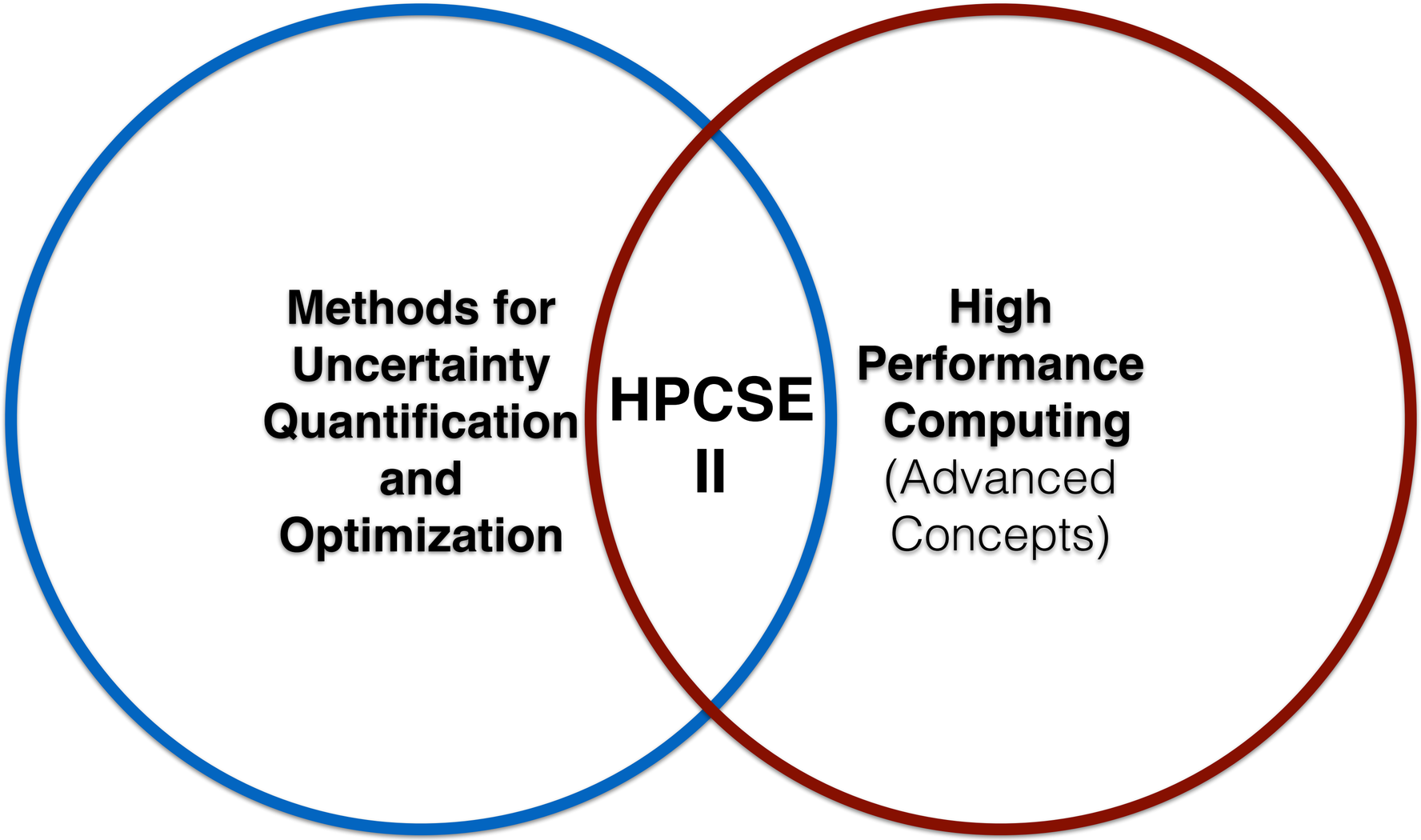
- An Overview.
- Why Uncertainty Quantification, Examples.
- The importance of High Performance Computing

Project / Homework

- An simple problem: n-Candles Problem
- (HPC) A review of algorithm optimization techniques.

Course Information

Course Contents



Schedule (Red = UQ, Blue = HPC)

18.02 (SM) Introduction
25.02 (PK) Bayesian UQ I - Likelihood & Priors
04.03 (PK) Bayesian UQ II - Laplacian Methods, Variational Inference
11.03 (PK) Markov Chain Monte Carlo
18.03 (SM) High Throughput Computation for UQ - UPC++ & Korali
25.03 (SM) Many-Core Processor Programming (CUDA) I (Basic Concepts)
01.04 (SM) Many-Core Processor Programming (CUDA) II (Optimizations)
08.04 (PK) Holiday?
15.04 (SM) Advanced MPI Concepts, MPI + OpenMP Hybridization.
29.04 (SM) MPI+CUDA, Heterogeneous Computing
06.05 (SM) Communication-Tolerant Programming, The MATE Model
13.05 (PK) TBD
20.05 (PK) TBD
27.05 --- Exam

Resources

Course Website:

- http://www.cse-lab.ethz.ch/teaching/hpcse-ii_fs19/
- Main resource for everything course-related.
- Homework will be posted here after class.

Discussion Board:

- All questions and comments should be sent through **Piazza**.
- You will receive an invitation to join.

Moodle:

- Exclusively for homework submission.
- https://moodle-app2.let.ethz.ch/course/view.php?id=10724 - For Engineers
- https://moodle-app2.let.ethz.ch/course/view.php?id=10725 - For CSE

Git Repository:

- https://gitlab.ethz.ch/hpcse_fs19/fs2019 - To download homework-related code.

Prerequisites

High Performance Computing

- Intermediate C/C++ skills.
- Performance Optimization Techniques.
- Concepts of concurrency and parallelism.
- Computer Architecture (Cache, multi-core processors)
- Basic knowledge of MPI / OpenMP / Vectorization.

Mathematics & Statistics

- Basic knowledge of PDE systems.
- Finite Difference Methods.
- Basic knowledge of statistics and probability theory.

There's plenty of review material in the course website!

Grading

Homeworks

- 6 Homework Assignments (coding and/or report).
- Accounts for 30% of the course grade (5% per Each HW).
- Submitted via Moodle (not email).
- Issued/graded every two weeks.
- We will provide intermediate milestones for assessment.
- Practice sessions: Mondays, 10.15 – 12.00 (HG G3) -- Starting Next Week.

Exam:

- 1 Exam on 27.05.2019. Computer and Paper-based Exam.
- Accounts for 70% of the course grade.
- **Tip:** If you do the **all** the homework, you will be much better prepared for the exam.

Working Together

Collaboration:

- Submitting as your own: work, or parts thereof, of another person (whether it is from a book, the web, a program library or ANY other source) without proper credit constitutes academic misconduct.

- You are encouraged to DISCUSS and WORK TOGETHER with other colleagues on the homework problems, especially through **Piazza**.

- If the source is properly cited then it is OK: homework credit should reflect use of other's work.

Plagiarism: One incident = Course Fail.

Uncertainty Quantification & Optimization

The Imperfect Paths to Knowledge

We, scientists and engineers, dedicate our work towards developing **tools** to better understand the **universe** and help us make **decisions**.

However: Every tool we have to understand the physical reality has an uncertainty associated to it. We need ways to measure/quantify them.

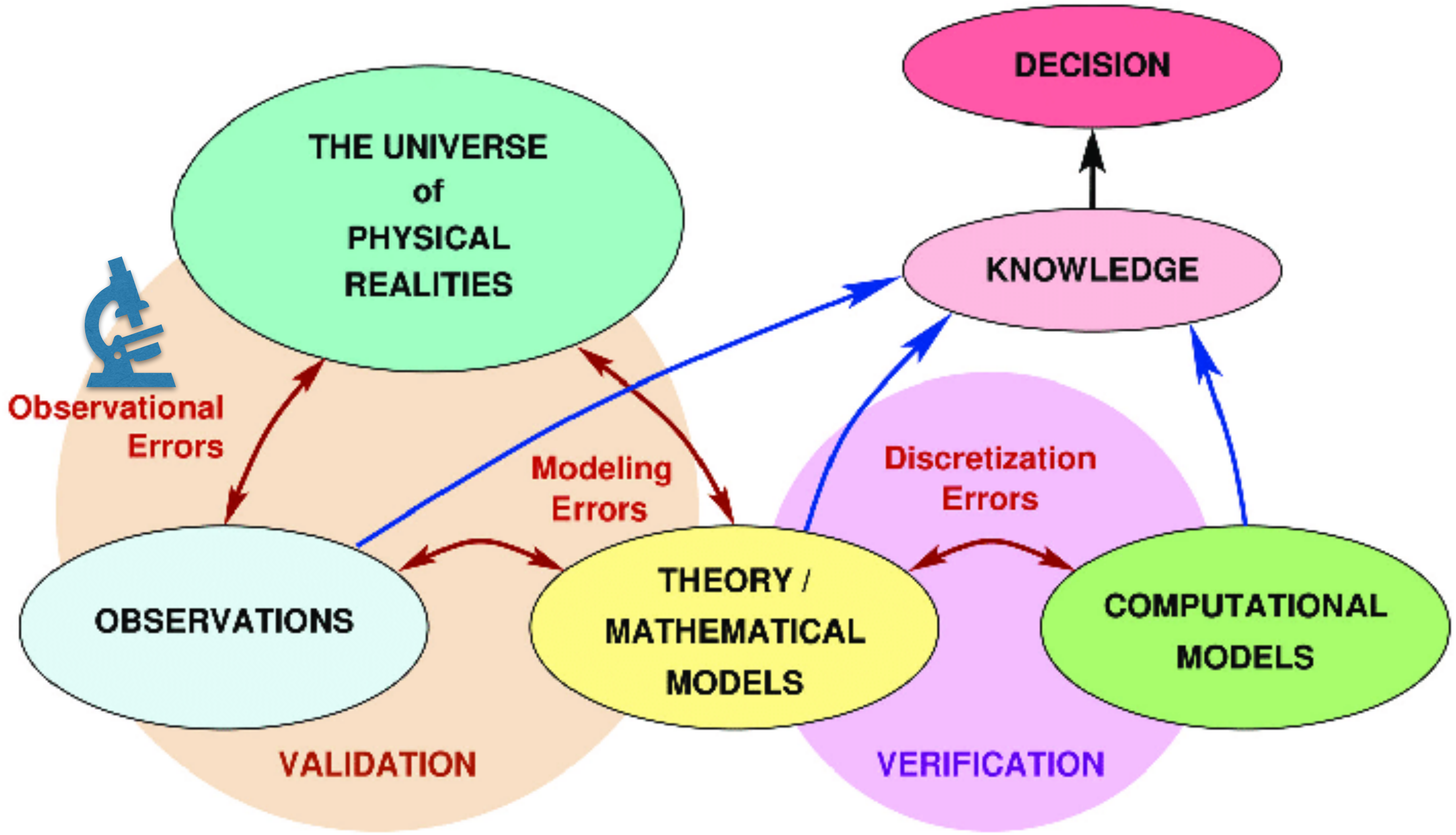


Image Credit: Americo Cunha Jr - Adapted from: R. Moser, T. Oden, O. Ghattas. UT Austin

Why Quantify Uncertainty? (I)

For Decision Making. Example: Hurricane.

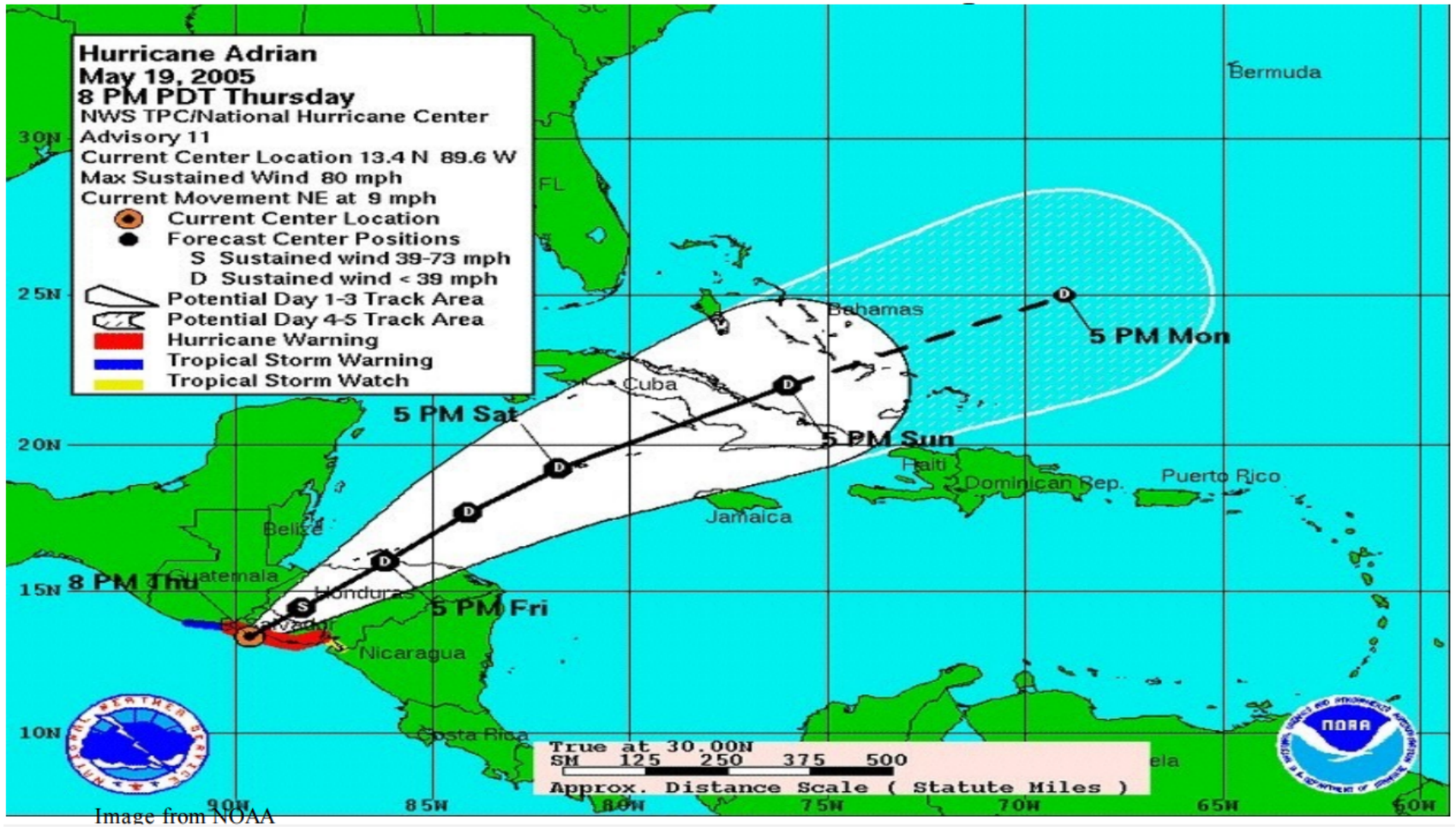


Image from NOAA

Image Credit: National Oceanic and Atmospheric Administration | U.S. Department of Commerce

Why Quantify Uncertainty? (II)

For Validation. Example: Speed of Light .

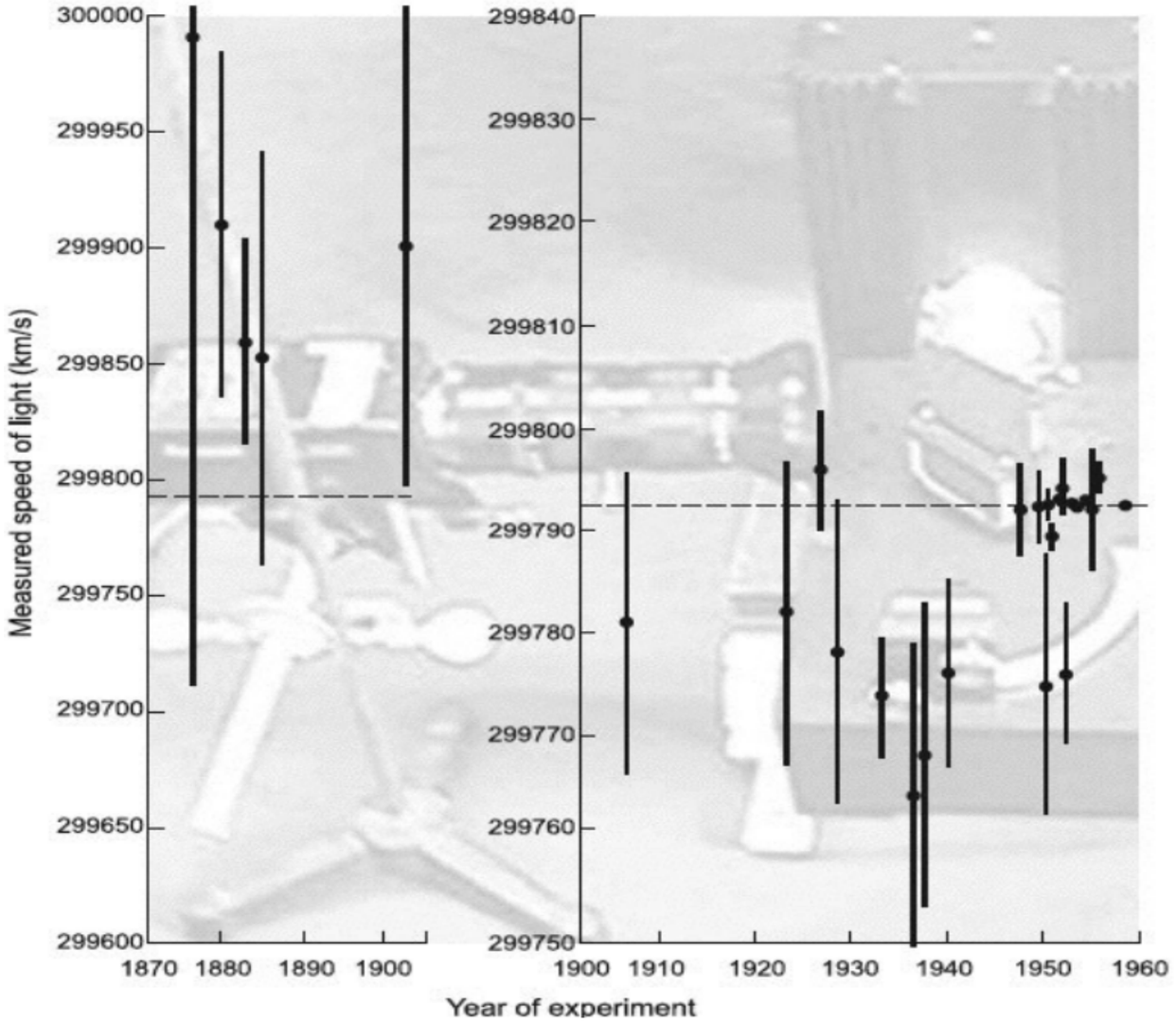


Image Credit: M. Christie et al., Los Alamos Science, #29, 2005

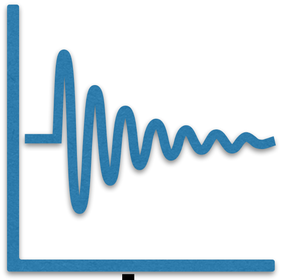
Definitions

- **System:** Any part of the world, natural or man-made, that we want to conceptually isolate to study. Inputs (**Parameters**) and outputs give its connections with the rest of the world.
  E.g., structural, mechanical, chemical, electrical, biological, and economic systems.

- **Mathematical Model:** A collection of laws and mathematical equations introduced to describe the behavior of a system (usually based on physical laws or observations).
  E.g., Algebraic equations, ordinary or partial differential equations, discrete equations.

- **Computational Model:** A numerical approximation or discretization of the mathematical model in a form that can be implemented in computers. Most mathematical models are too complicated to solve them exactly and **numerical approximations** are most of the time introduced to solve the problem in reasonable time.

How do we optimize/quantify uncertainty? (I)



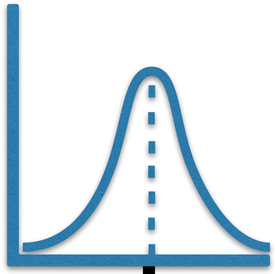
Better understanding of the system and models uncertainties.

UQ Methods
CMA-ES
Markov-Chain Monte-Carlo
TMCMC

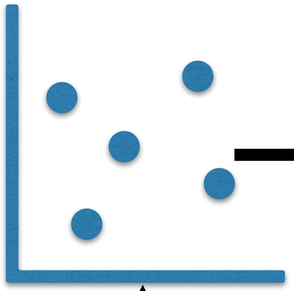
Stay Tuned!
More about these next week(s).

Improved

Parameter Guess



Computed Results



Experimental Measurements

Compare

Discretize / Approximate

Mathematical Model

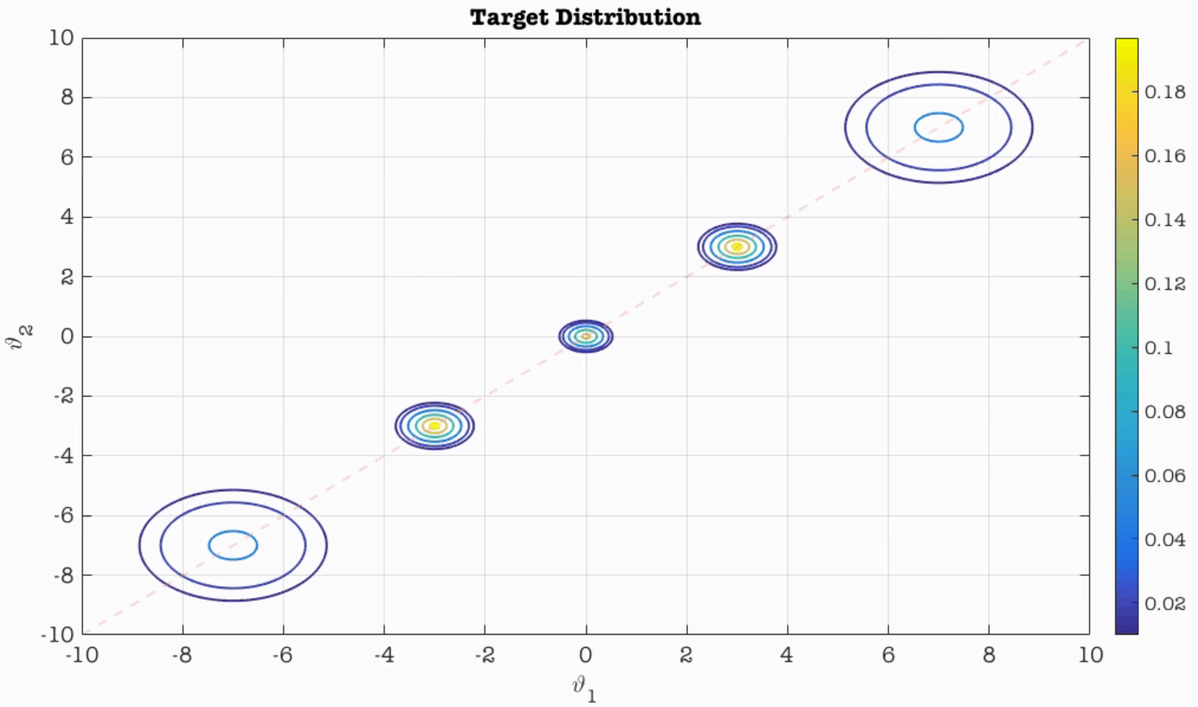
Computational Model

Physical System

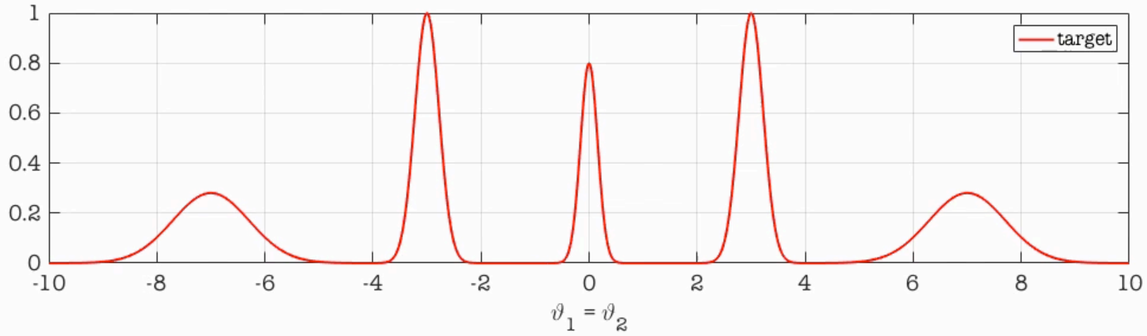
How do we optimize/quantify uncertainty? (III)

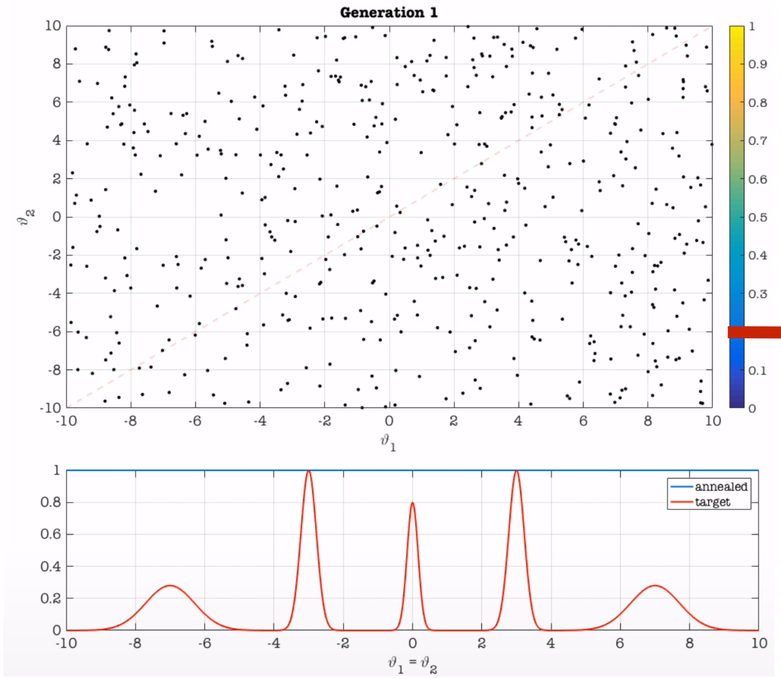
Uncertainty. Finding most likely distribution of a parameters *v1* and *v2*.

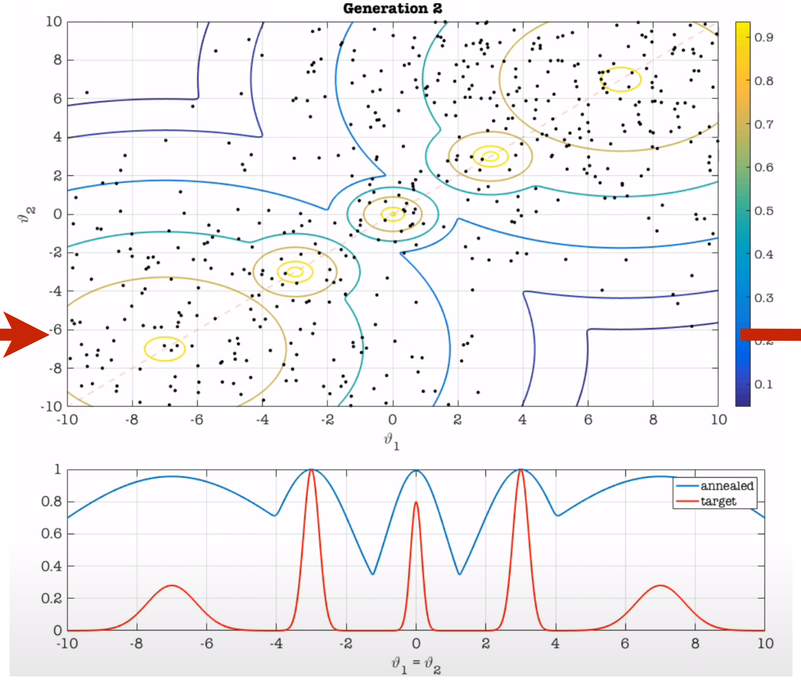
Parameter Space →

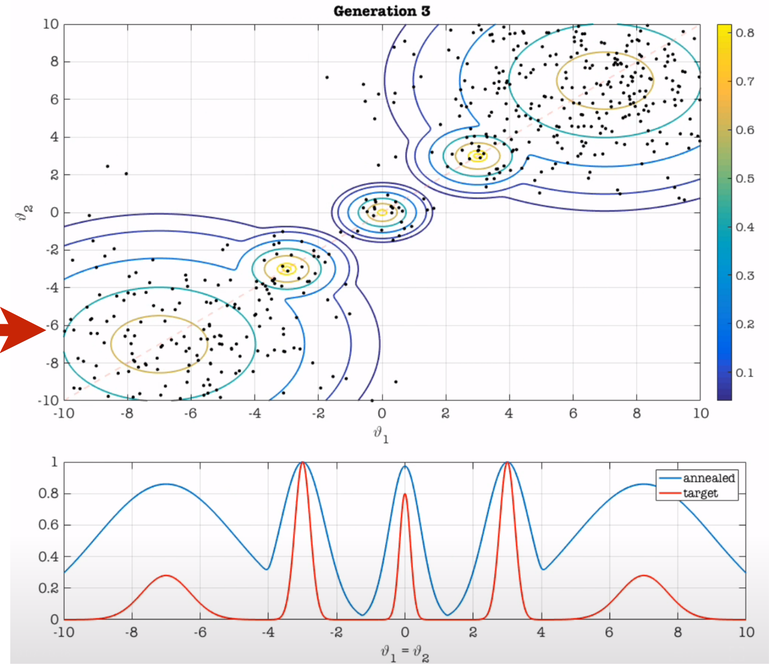


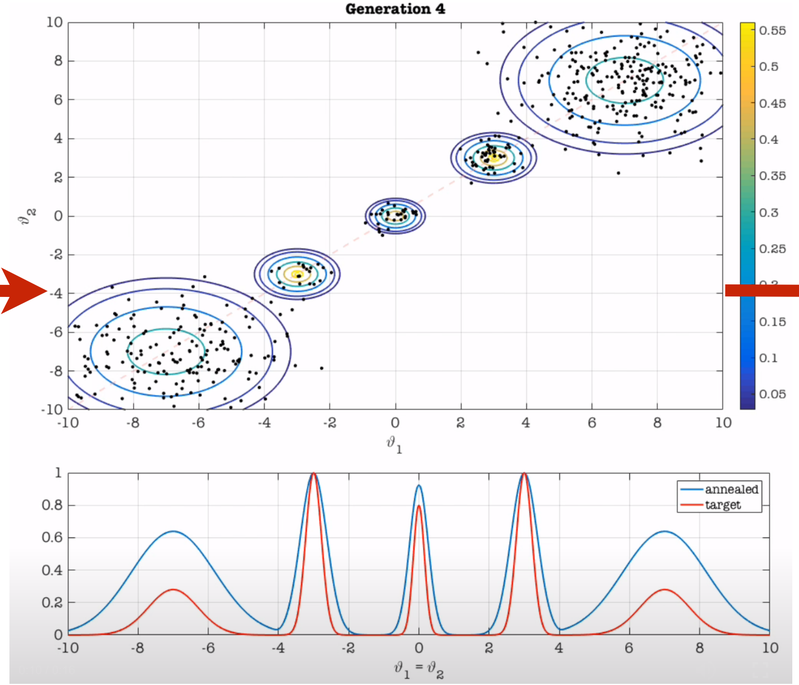
Likelihood *v1=v2*
i.e. Parameters Reflect Experimental Observations →

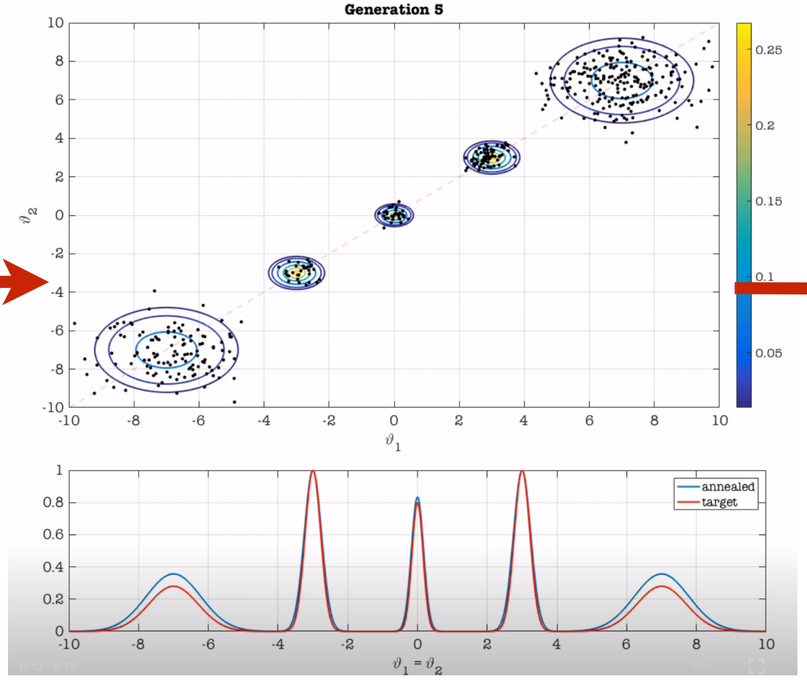


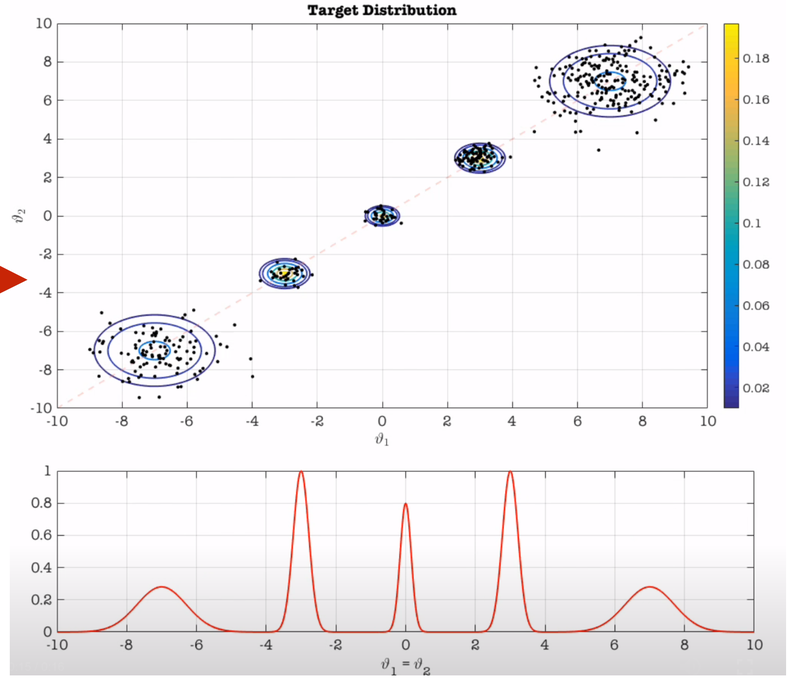

Generation 1
annealed
target


Generation 2
annealed
target


Generation 3
annealed
target

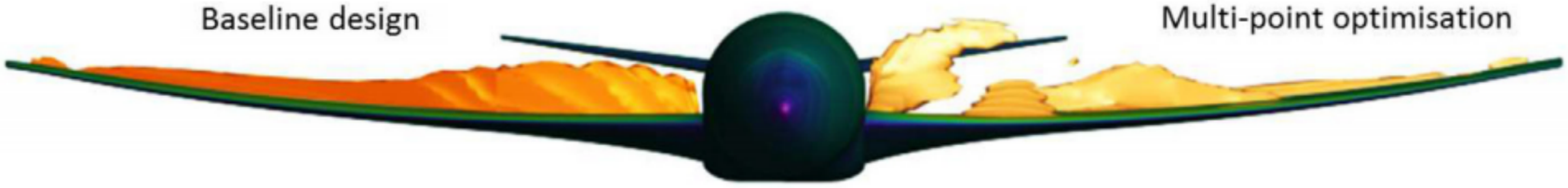

Generation 4
annealed
target


Generation 5
annealed
target

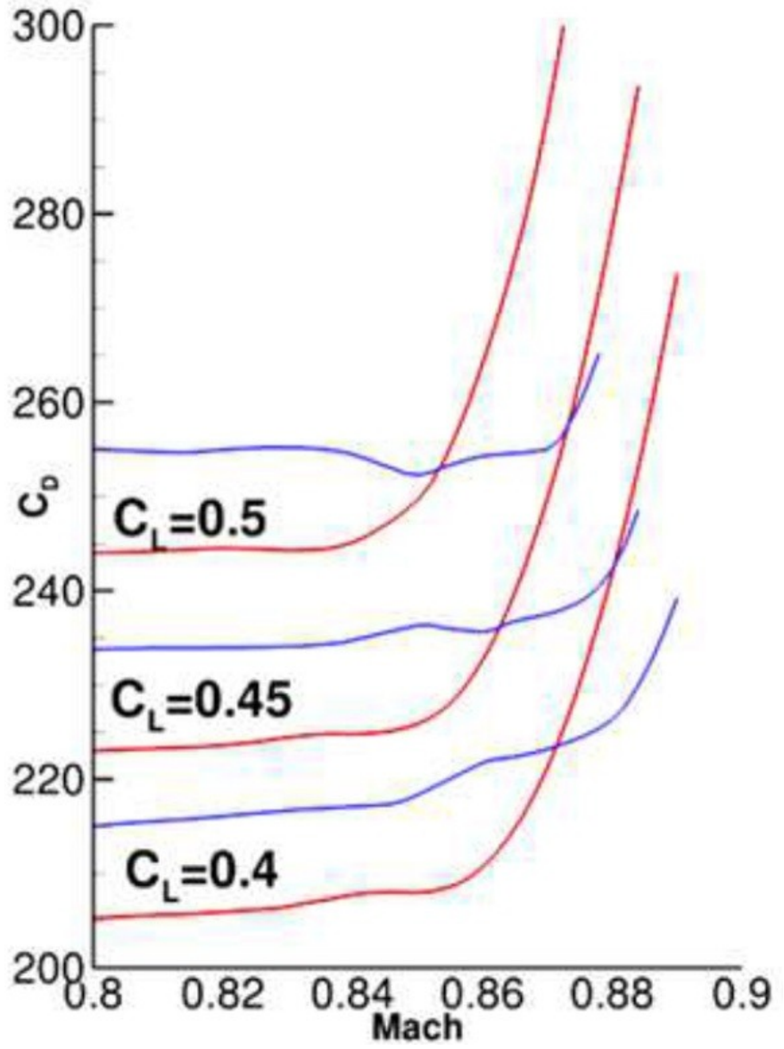

Target Distribution

Optimization

Example: Shaping wings for reduced drag.

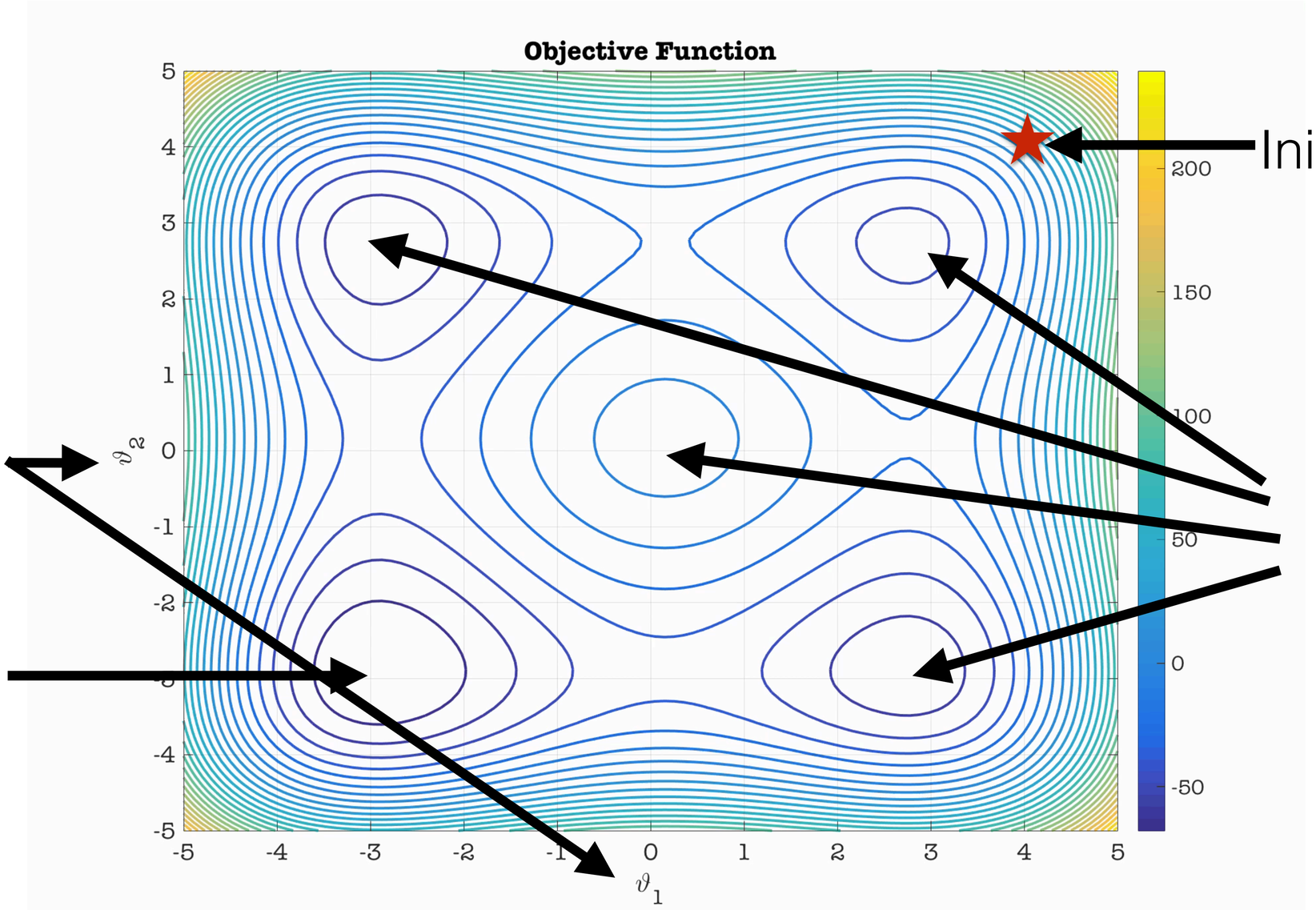


Source: State-of-the-art in aerodynamic shape optimisation methods. N. Skinner, H. Zare-Behtash



How do we optimize parameters?

Optimization. Minimizing a two-parameter function.



High-Throughput Computing

Fact I: Uncertainty Quantification and Optimization methods require many model evaluations.

Fact II: For problems with many parameters, the number of model evaluations can be vast.

Our Goals:

- Develop fast and scalable computational models
- Ways to execute many evaluations given limited time and computational resources.

Our Tools:

High-Throughput Computing

Efficiently running many tasks for long-periods of time, using many computational resources

In this Course You'll Learn:
Korali and UPC++

High Performance Computing

Build scalable algorithms that run as fast as possible in short periods of time (FLOPs)

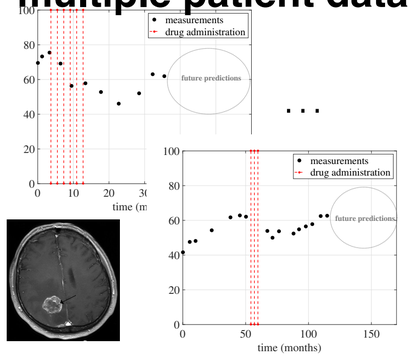
In HPCSEI: **MPI, OpenMP, PThreads**

In HPCSEII: **More MPI, CUDA, MPI+X Communication-Tolerant Programming**

Research @ CSELab

Pharmaceutical Applications: Patient Evolution Expectancy

multiple patient data



model

$$\frac{dC}{dt} = -\vartheta_1 C$$

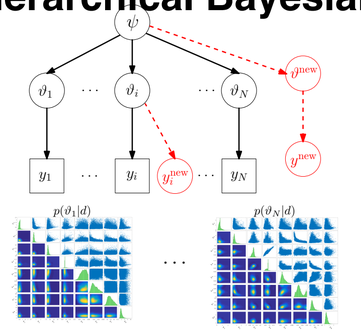
$$\frac{dP}{dt} = \vartheta_4 P(1 - \frac{P^*}{K}) + \vartheta_5 Q_P - \vartheta_3 P - \vartheta_1 \vartheta_2 CP$$

$$\frac{dQ}{dt} = \vartheta_3 P + \vartheta_1 \vartheta_2 CQ$$

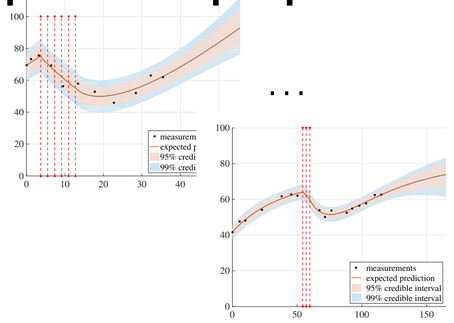
$$\frac{dQ_P}{dt} = \vartheta_1 \vartheta_2 CQ - \vartheta_5 Q_P - \vartheta_6 Q_P$$

$$C(0) = 0,\ P(0) = \vartheta_7,\ Q(0) = \vartheta_8,\ Q_P(0) = 0$$

hierarchical Bayesian

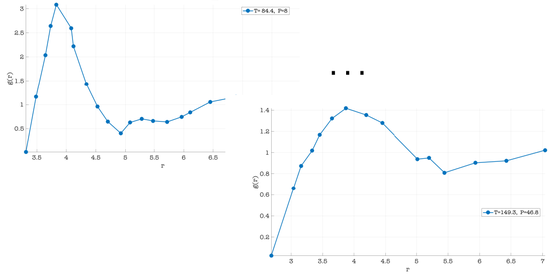


prediction per patient



Nuclear Chemistry: Density Distribution of Argon Gas

RDF from multiple thermodynamic conditions

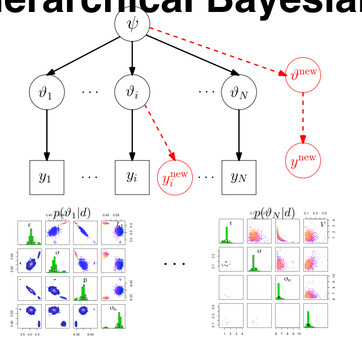


model

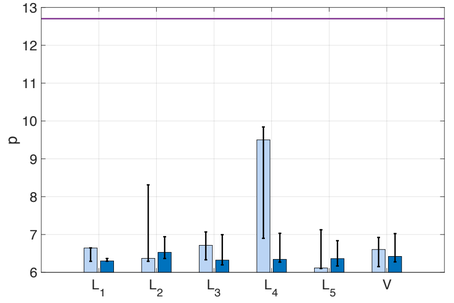
$$M\ddot{X} = -\nabla V(X)$$

$$V(r) = 4\varepsilon\left[\left(\frac{\sigma}{r}\right)^{p} - \left(\frac{\sigma}{r}\right)^{6}\right]$$

hierarchical Bayesian

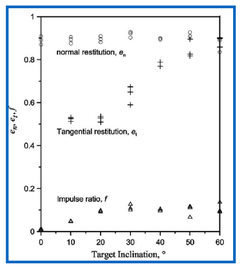


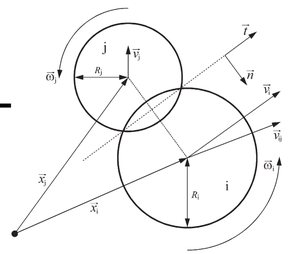
exponent value per data set



Material Science: Dynamics of Granular Materials

Coefficient of restitution of steel beads





model

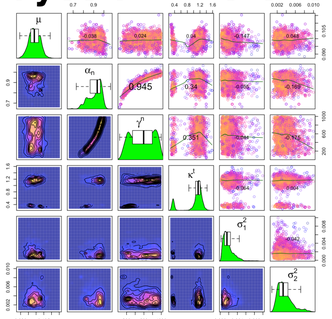
$$\mathbf{F}^n = -k^n \mathbf{y} - \gamma^n \frac{d\mathbf{y}}{dt}|\mathbf{y}|^{\alpha_n}$$

$$\mathbf{F}^t = \min\left(\frac{2}{3}k^t \boldsymbol{\xi}^t, \mu \mathbf{F}^n\right)$$

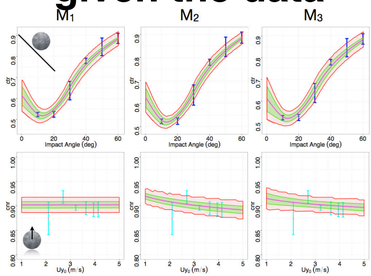
$$\boldsymbol{\xi}^t = \int_{t_0}^{t} \boldsymbol{v}_{rel}^t(\tau) d\tau$$

$$k^n = \frac{4}{3} E_{eff} \sqrt{R_{eff}|\mathbf{y}|}$$

Bayesian Inference



Most probable model given the data



A simpler case:

The n-Candles Problem

Also, your course project! 

Physical System

Heat Distribution on a Metal Plate

Image Credit: lakeshorepublicmedia.org

We study the steady state (final) temperature distribution on a metal plate, given:

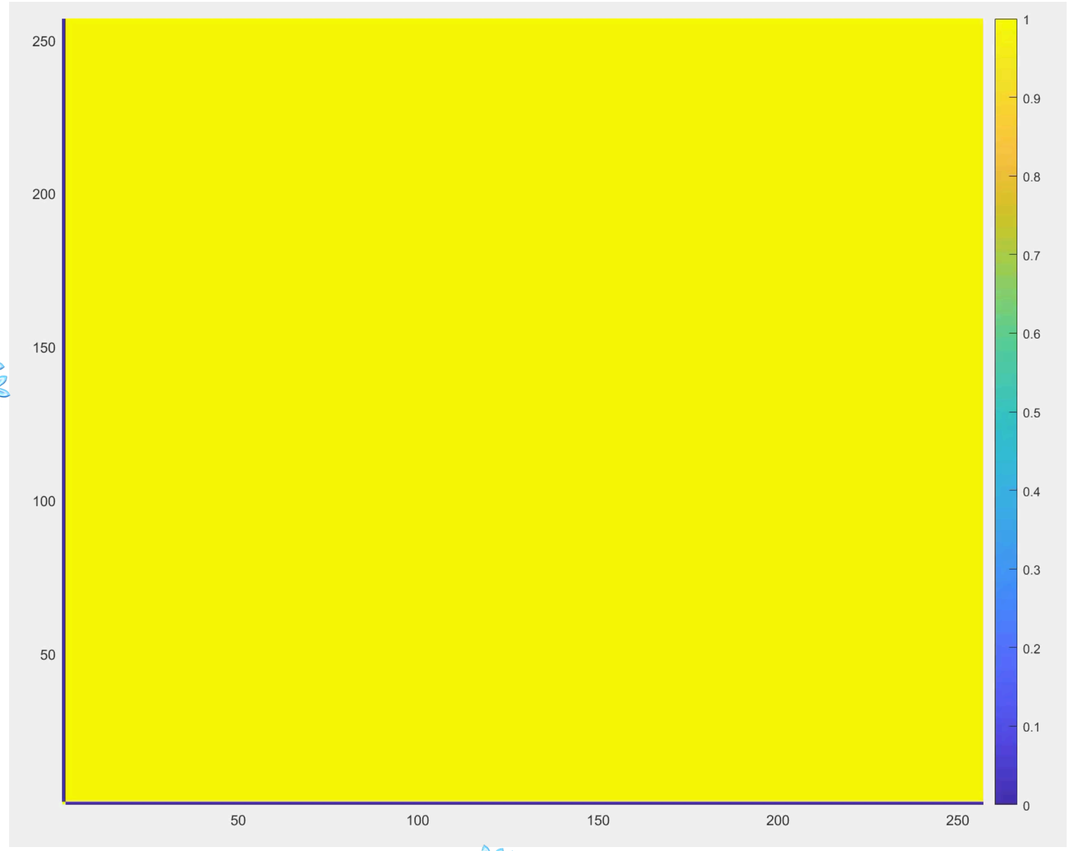
- Initial Temperature
- External Temperatures (Boundary)
- External Heat Sources.

Physical System

Example: 🔥Plate - ❄️Boundaries - No Heat Sources

❄️

❄️



❄️

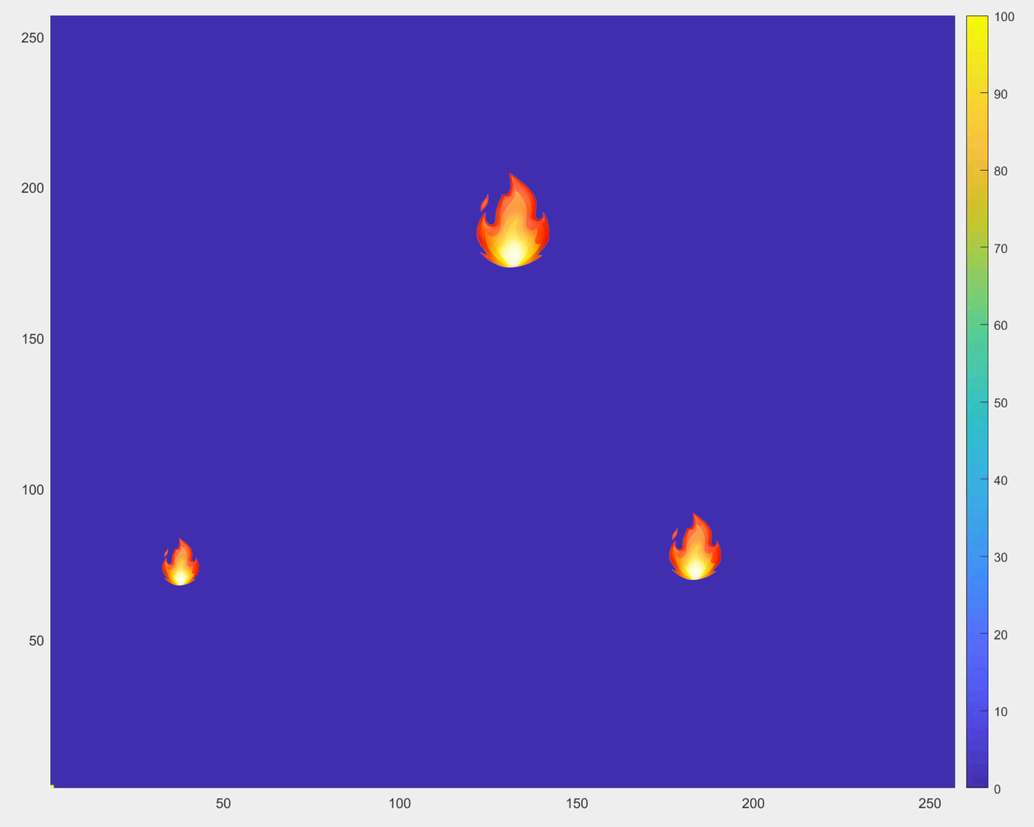
❄️

Physical System

Example: ❄️Plate - ❄️Boundaries - 3 Heat Sources🔥🔥🔥

❄️

❄️

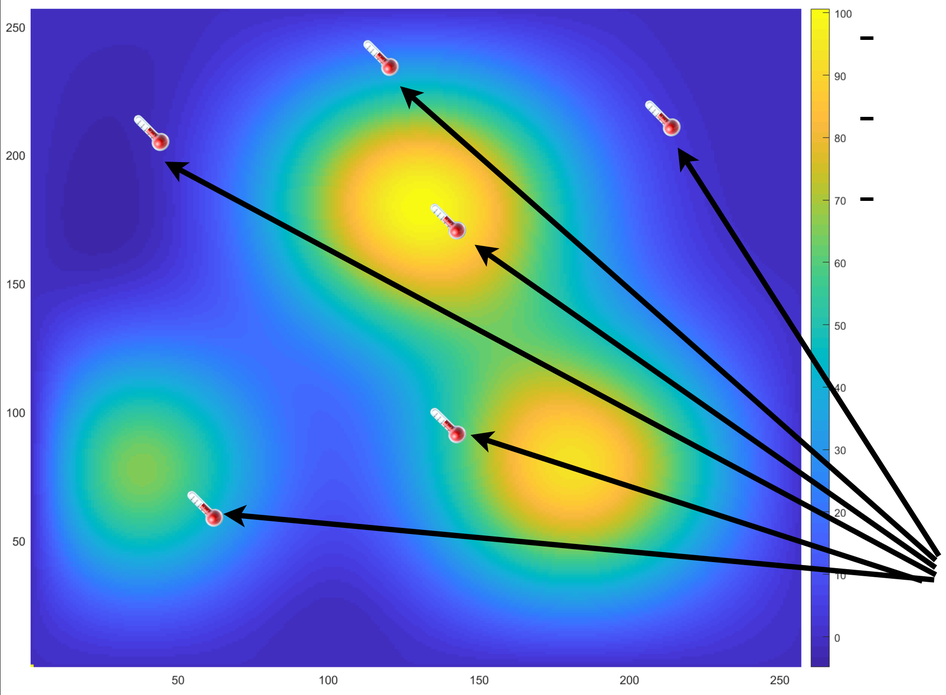


❄️

❄️

Course Project

The n-Candles Problem



- We know there are 3 sources of heat (candles)
- We don't know their location (x,y), intensity nor width.
- Neither we know the thermal diffusivity of the material.

Total: 13 unknown parameters

You receive:

A set of experimental observations.
(Temperature measured a different points)

Your Task:

- Determine the most likely values of each of the parameters.
- Quantify the overall uncertainty for each parameter.

Mathematical Model

2D Heat Equation

$$\frac{\partial T}{\partial t} - \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = f(x, y)$$

Rate of Change over Time

Thermal diffusivity (Constant)

Laplace Operator

External Heat Sources

Steady-state equation with $\frac{\partial T}{\partial t} = 0$

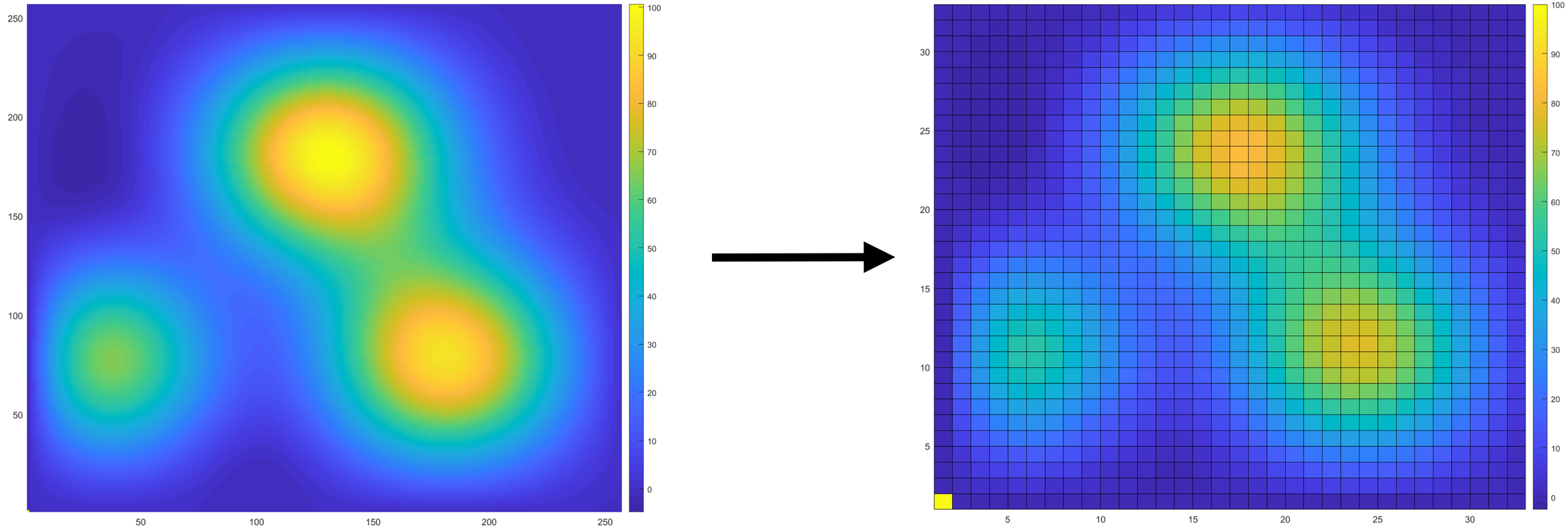
$$- \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) = f(x, y)$$

Computational Model (I)

Spatial Discretization:

$$[0,1] \times [0,1] \rightarrow N \times N \; Grid, \; with \; h = \Delta x = \Delta y = 1/N$$

Discretization Error!



Computational Model (II)

Second-order central difference approximation to 2nd Derivatives:

$$\frac{\partial^2 T}{\partial x^2} \approx \frac{T(x+h,y) - 2T(x,y) + T(x-h,y)}{h^2}$$

$$\frac{\partial^2 T}{\partial y^2} \approx \frac{T(x,y+h) - 2T(x,y) + T(x,y-h)}{h^2}$$

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \approx \frac{T(x-h,y) + T(x,y+h) - 4T(x,y) + T(x-h,y) + T(x,y-h)}{h^2}$$

Approximation Error!

From Steady State Heat Eq: $-\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = f(x,y)$ and **α = 1**

$$-T(x-h,y) - T(x,y+h) + 4T(x,y) - T(x-h,y) - T(x,y-h) \approx h^2 f(x,y)$$

System of linear equations (Ax=b).

Computational Model (III)

$$A = \begin{bmatrix} 4 & -1 & & -1 & & & & \\ -1 & 4 & -1 & & -1 & & & \\ & -1 & 4 & -1 & & -1 & & \\ -1 & & -1 & 4 & -1 & & -1 & \\ & \ddots & & \ddots & \ddots & \ddots & & \ddots \\ & & -1 & & -1 & 4 & -1 & & -1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} T_{1,1} \\ T_{1,2} \\ \vdots \\ T_{N,N} \end{bmatrix}, \qquad \mathbf{b} = h^2 \begin{bmatrix} f_{1,1} \\ f_{1,2} \\ \vdots \\ f_{N,N} \end{bmatrix}$$

We approach $A\mathbf{x} = \mathbf{b}$ Iteratively: **Jacobi Method**

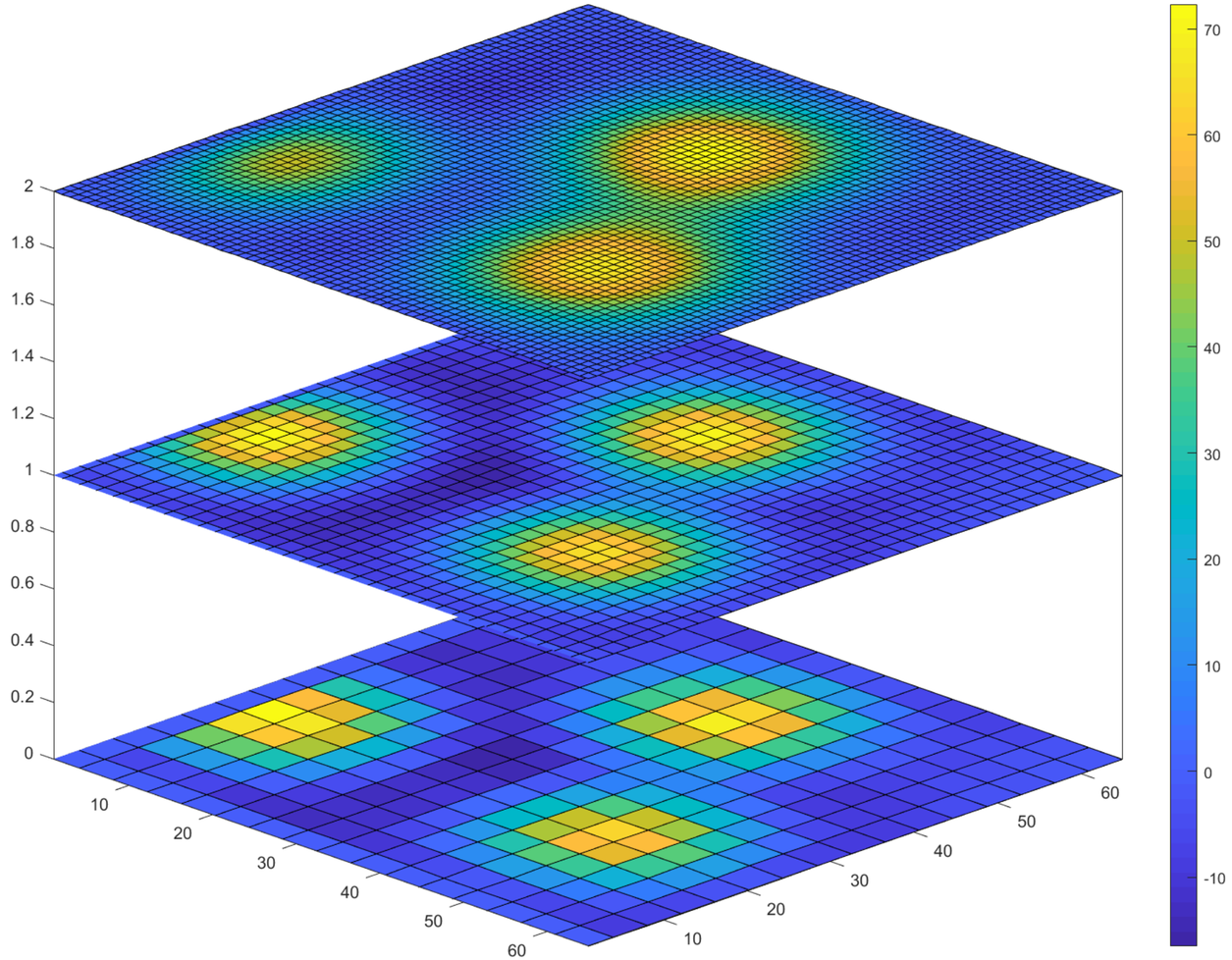
$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

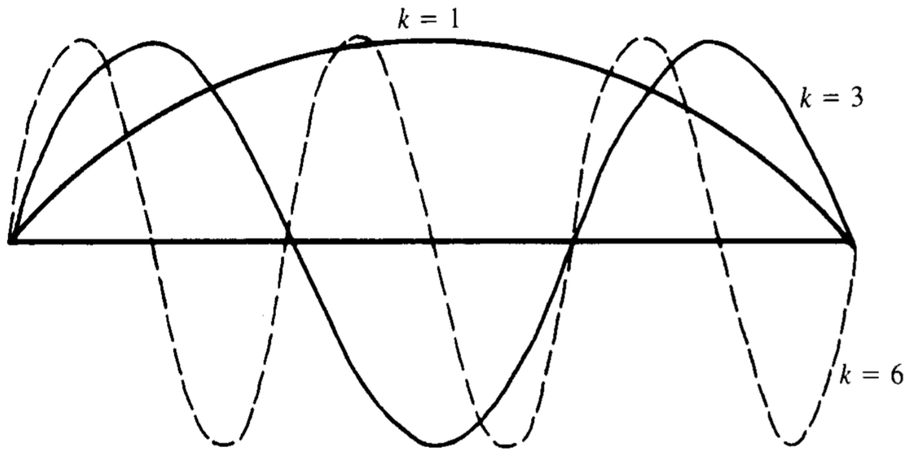
$$T_{i,j}^{(k+1)} = \frac{1}{4} \left(h^2 f_{i,j} - \sum_{j \neq i} a_{ij} T_{ij}^{(k)} \right), \quad i = 1, 2, \ldots, N \quad and \quad j = 1, 2, \ldots, N.$$

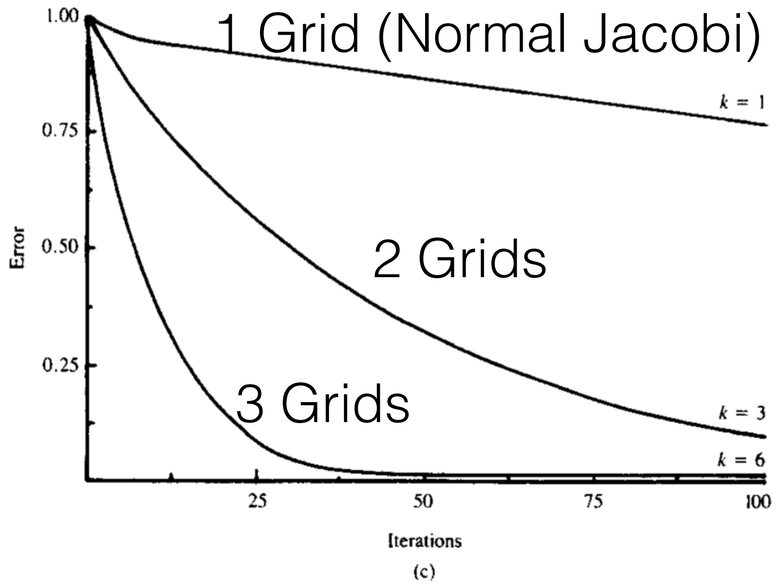
Problem: Jacobi converges too slowly.

Multigrid Method (I)

The **Multigrid method** accelerates convergence by '*smoothing*' the error at multiple frequencies.

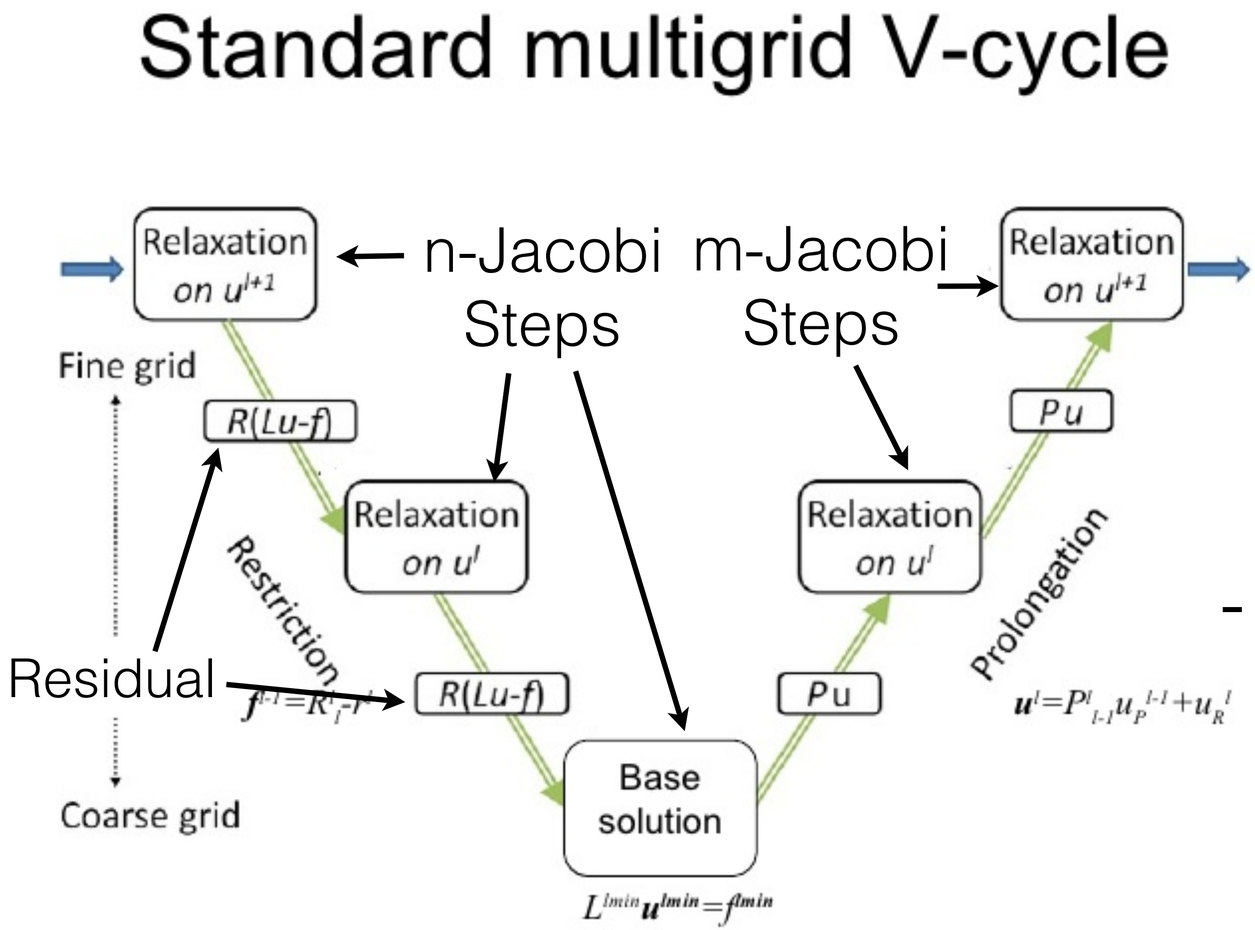






Source: A Multigrid Tutorial, 2nd Edition. Briggs, Henson, McCormick.

Multigrid Method (II)



Source: Streaming Multigrid For Gradient Domain Operations On Large Images. M. Kazhdan, H. Hoppe

Requires special consideration for communication and cache optimizations (Great for an HPC Course project!)

Recommended Reads:

- Multigrid Methods (G. Strang)
- A Multigrid Tutorial, 2nd Edition. Briggs Henson, McCormick.

How about other cycles? (e.g., W)

Homework I: The Basics

Steps:

1. Download Homework I PDF from our website. [**Follow Instructions!**]
   http://www.cse-lab.ethz.ch/teaching/hpcse-ii_fs19/
2. Download/Clone our C/C++ Multigrid solver code from the git repository
   https://gitlab.ethz.ch/hpcse_fs19/fs2019.
3. Find optimal Multigrid configuration:
   1. How many grids, given a problem size.
   2. How many relaxations upon restriction/prolongation.
4. Apply algorithm optimization techniques to improve its performance.
   1. Cache/Memory Locality.
   2. Vectorization
   3. Pointer / Constant Optimizations.
5. Answer a few questions and fill out report.

Homework Grading

How we will grade:

1. You will only work on a single *heat2d.cpp* file containing the important parts to optimize.
2. **Submit** heat2d.cpp and report.pdf only (through Moodle).
3. We will test time-to-convergence (less = better) of your submission on **Euler** compute nodes.
4. **Self-grading**:
    1. If your code passes a two correctness (--verify) tests: **Problem2** and **Problem3** and,
    2. Reaches a certain baseline performance.
       The code is -almost- guaranteed to pass (unless it fails our tests for some other reason).
5. **First-Week Milestone**: A goal to be achieved by next Monday (practice).
   Not graded, but highly recommended.
6. **Optional assignments**. Not graded, but really really recommended.
7. **PDF report:** will be manually graded (should be easy if you worked with the code).

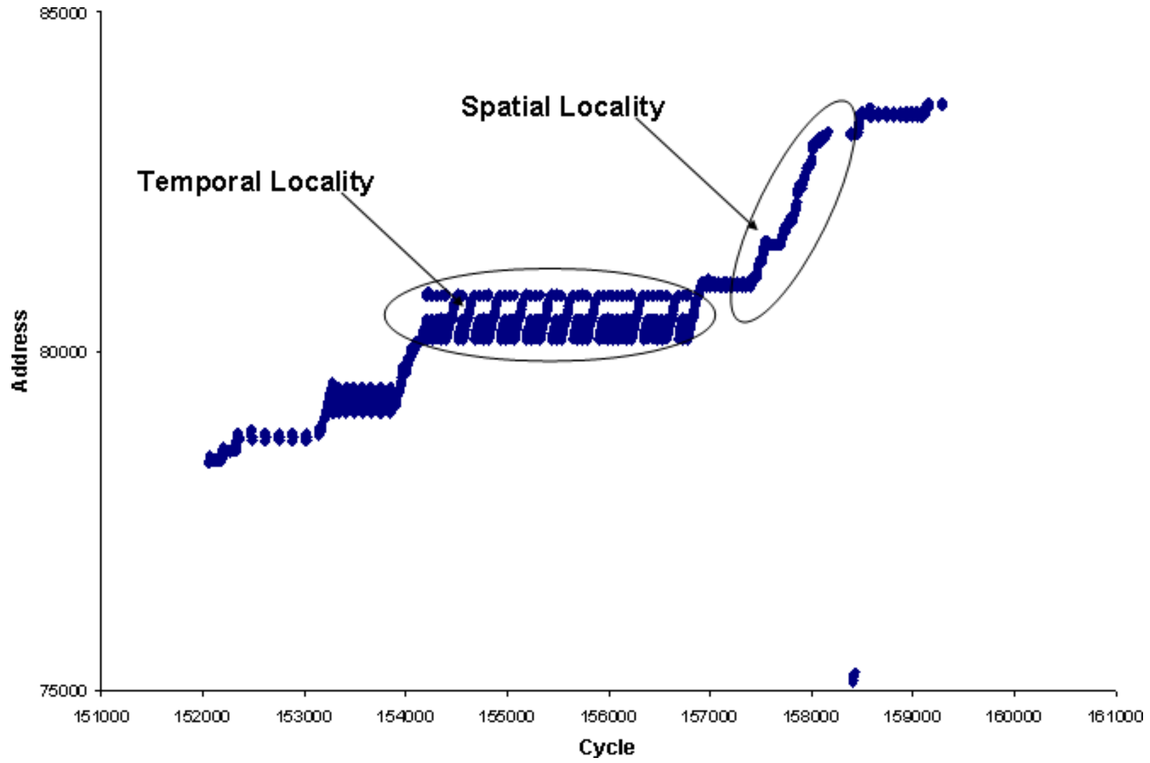
Friendly Competition: We will post the TOP 10 Homework I performances in our page.

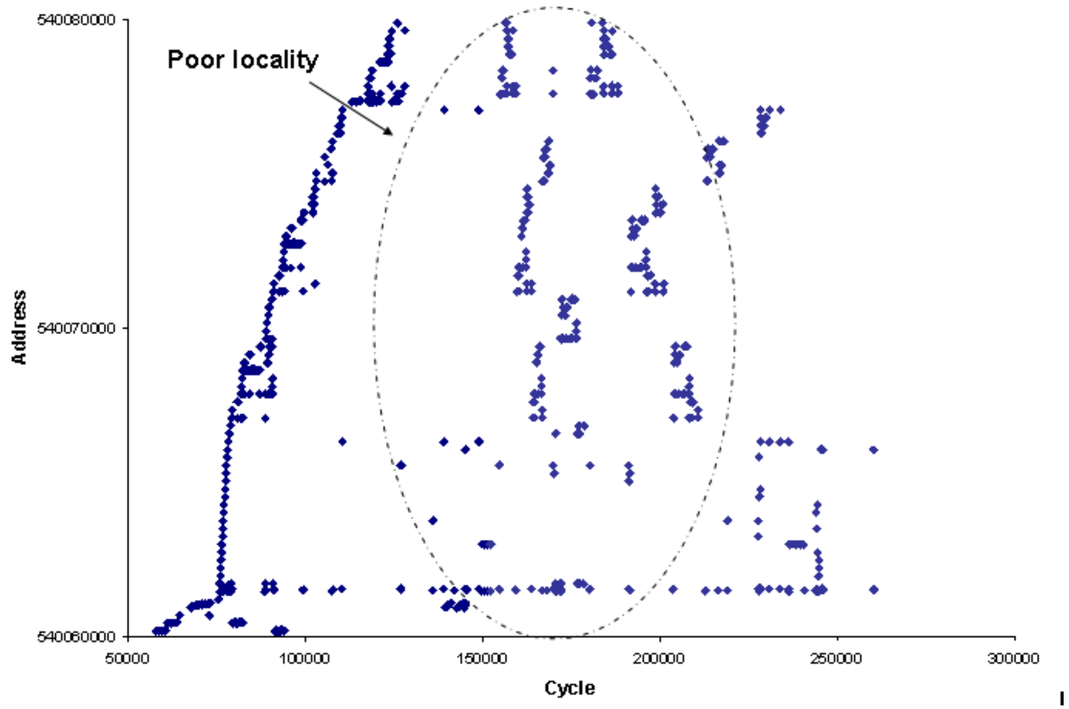
(Lets us know if you'd like to opt-out)

Review:
Algorithm Optimization

Cache Locality (I)

Cache structures in modern processors benefit from both **Temporal** and **Spatial** locality.





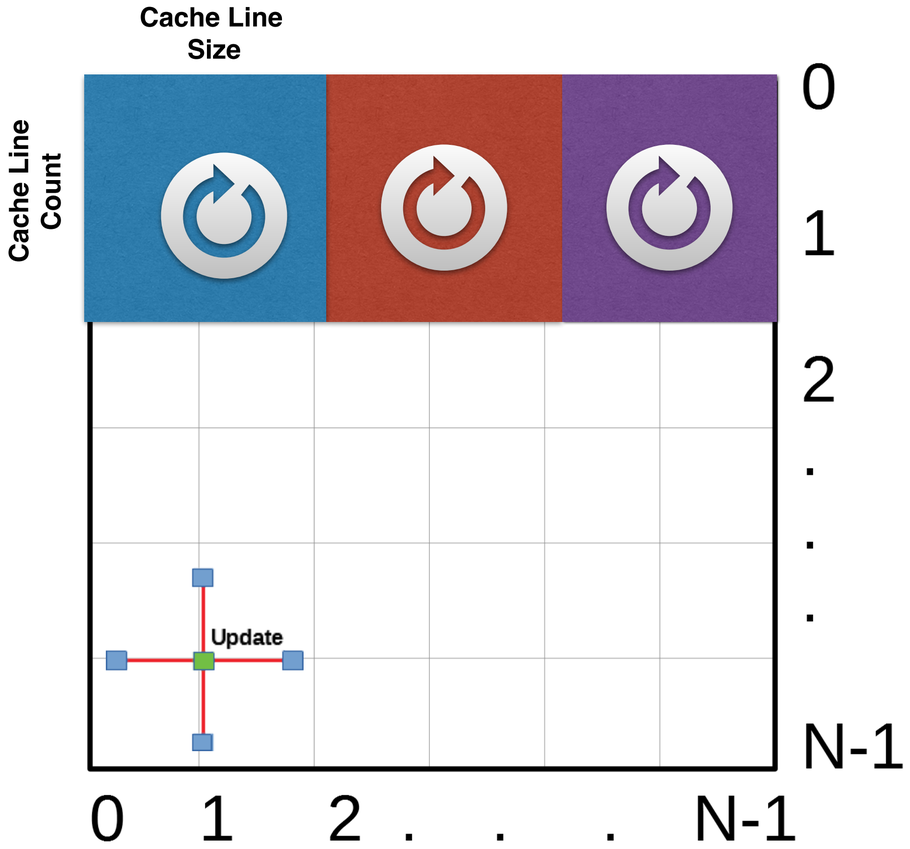
High Cache Line Reuse

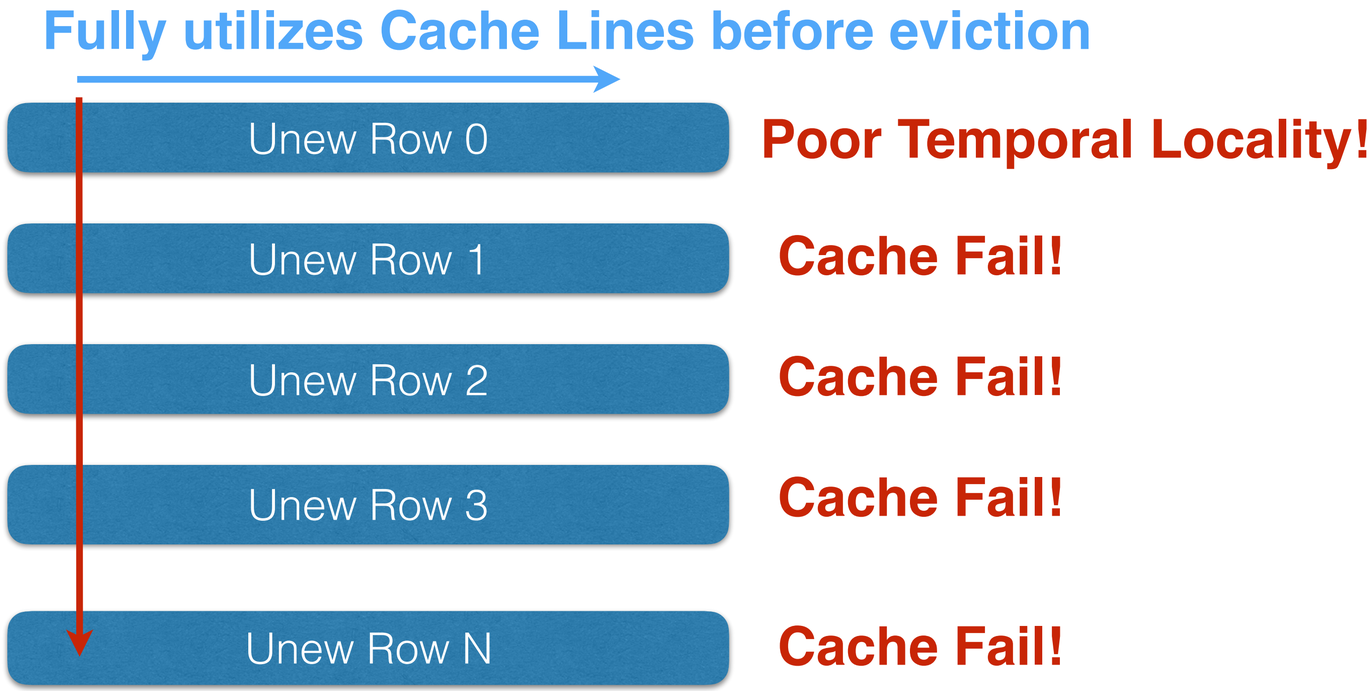
Frequent Cache Fails

Source: Optimizing for instruction caches, part 1, Amir Kleen, Livadariu Mircea, Itay Peled, Erez Steinberg, Moshe Anschel, Freescale

Cache Locality (III)

Problem 2) Solver iterates the 2D space row-wise first, instead of column-wise.



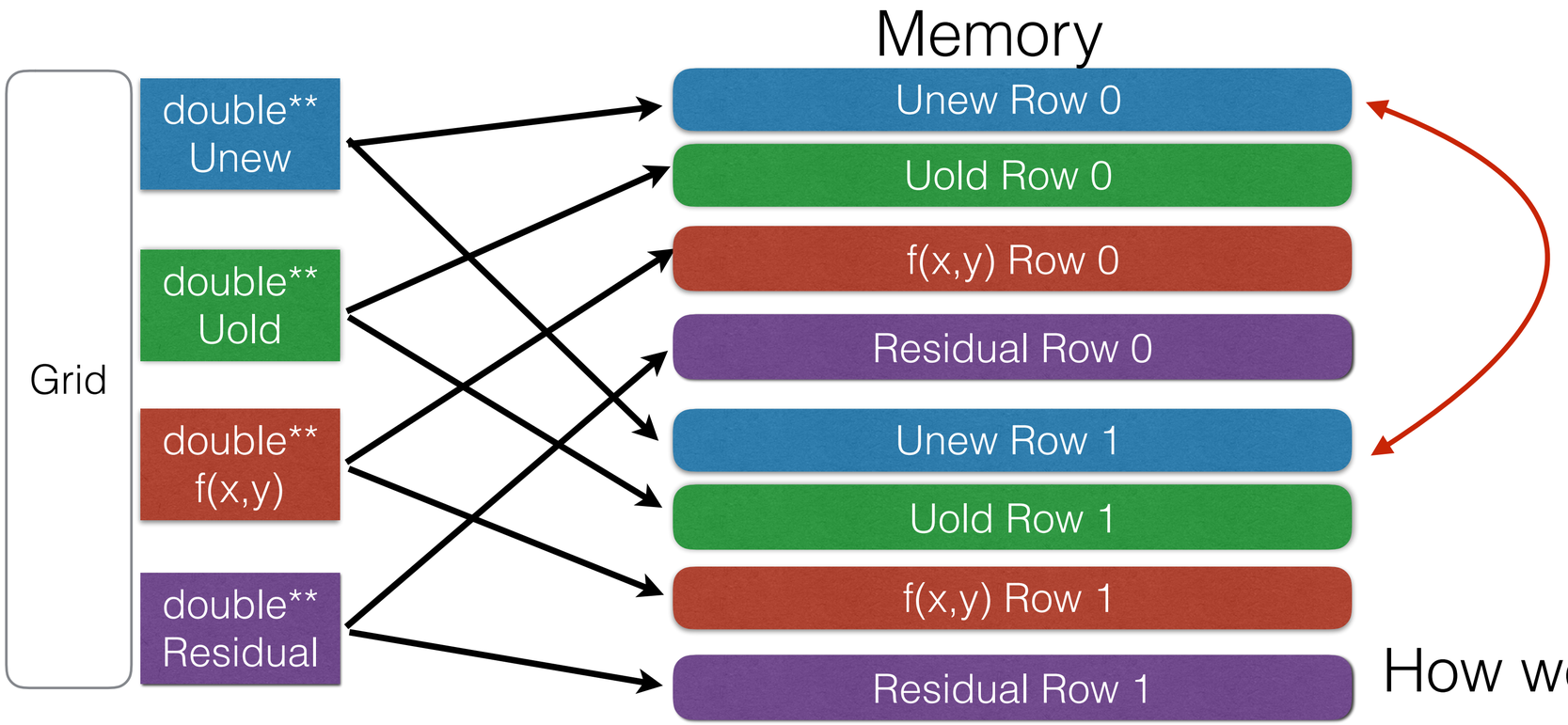


- Can you improve locality even further?
  Hint: **Cache Blocking (L1, L2 or both)**

Cache Locality (II)

The base Heat2d.cpp has two severe problems:

Problem 1) Rows for Unew, Uold, f(x,y), and Residual are intertwined:



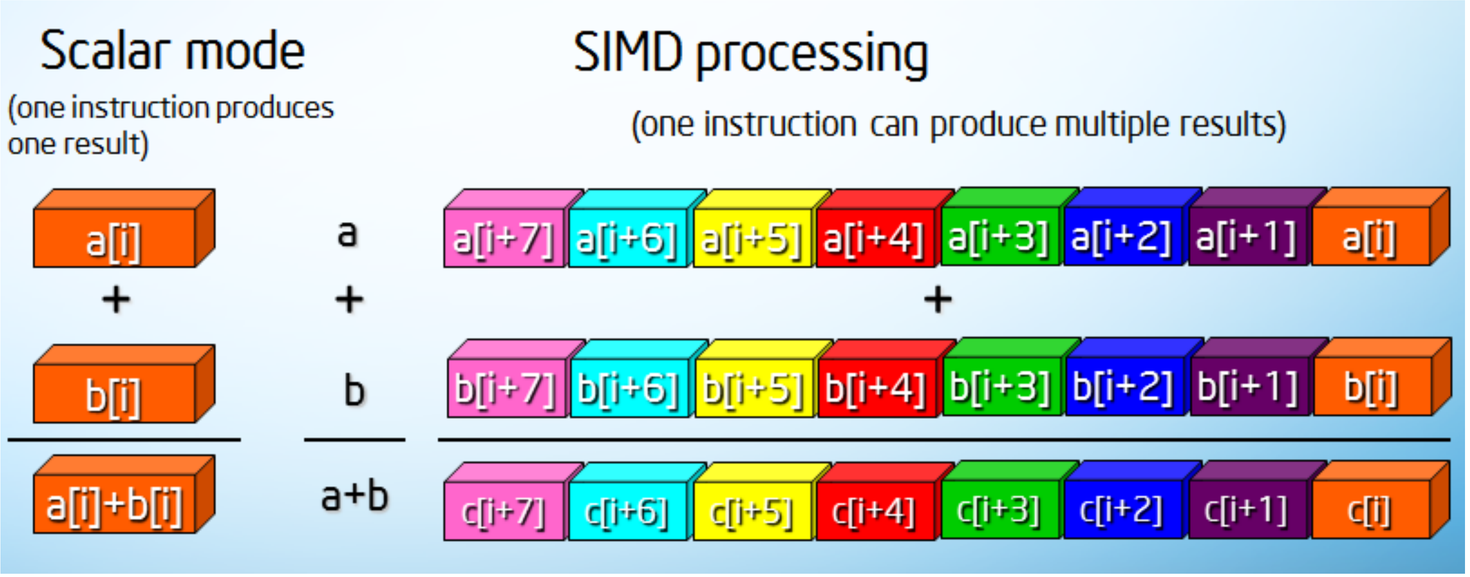
Poor Spatial Locality!

How would you solve this problem?

Vectorization (Review)

Single Instruction, Multiple Data Paradigm

Apply the same instruction to an array of elements, instead one-by-one.



Source: Vectorization - Find out what it is, Find out More!, Shannon Cepeda (Intel)

Three ways to go about it:

1. Compiler-Enabled Auto-Vectorization (GNU or Intel). (Easiest option - It's ok)
2. Intel SPMD Program Compiler (ISPC)
3. Intel Intrinsics

Auto-Vectorization

Visit: https://software.intel.com/en-us/cpp-compiler-auto-vectorization-tutorial

Load the Intel module and compile with -O3 and **-qopt-report=3** to generate a detailed vectorization report:

```
icc -O3 -D NOFUNCCALL -qopt-report=3 -qopt-report-phase=vec Multiply.c
```

```
LOOP BEGIN at Multiply.c(49,9)
      remark #15305: vectorization support: vector length 2
      remark #15399: vectorization support: unroll factor set to 4
      remark #15300: LOOP WAS VECTORIZED
      remark #15442: entire loop may be executed in remainder
      remark #15448: unmasked aligned unit stride loads: 2
      remark #15475: --- begin vector cost summary ---
      remark #15476: scalar cost: 10
      remark #15477: vector cost: 4.000
      remark #15478: estimated potential speedup: 2.380
      remark #15488: --- end vector cost summary ---
   LOOP END
```

The Auto-Vectorization optimizer requires that

- Allocations are aligned to 16-bit
- No pointer aliasing exists
- No inter-loop dependencies are violated.

Even if your program satisfies these conditions,
The compiler has no way to know unless we provide **hints**!

Investigate the effect of these hints:
-D NOALIAS
#pragma vector aligned
#pragma ivdep

for GNU: `-ftree-vectorize -ftree-vectorizer-verbose=X`;

Pointer Tricks

Problem:
The Jacobi method requires exchanging *Uold* <- *Unew* at every smoothing step.

```
// Update Uold <- Unew
for (int j = 0; j < N; j++)
  for (int i = 0; i < N; i++)
    Un[i][j] = U[i][j];
```

And then *Unew* is updated with new values calculated by the stencil.

```
// Apply central difference stencil
for (int j = 1; j < N-1; j++)
  for (int i = 1; i < N-1; i++)
    U[i][j] = (Un[i-1][j] + Un[i+1][j] + Un[i][j-1] + Un[i][j+1])/4 + f[i][j]*pow(h,2)/4;
```

How can we optimize the *Uold* <- *Unew* update step?

How about the Gauss-Seidel Method?

Constant Optimizations

Typically handled by the compiler. But we can help! (can we?)

```
// Apply central difference stencil
for (int j = 1; j < N-1; j++)
  for (int i = 1; i < N-1; i++)
    U[i][j] = (Un[i-1][j] + Un[i+1][j] + Un[i][j-1] + Un[i][j+1])/4 + f[i][j]*pow(h,2)/4;
```

Hint 1: pow(h,2) is a costly function.
Hint 2: Although it is a loop invariant, it is not taken out by the compiler. Why?
Hint 3: Division takes many more CPU cycles than multiplication.
Hint 4: Can we use associativity to simplify this operation?

Amdahl's Law (Review)

Typically used to evaluate the impact of parallelization:
E.g., How much speedup will we gain if we parallelize a section of our application.

Can also be used to estimate the impact of optimizations in sequential codes.
E.g., How much speedup will we gain by applying vectorization.

Example: Execution time of program made of A, B, and C sections.

Question:
If vectorization reduces computing time by half and we can only apply it to one section.
What's the maximum speedup we could get?

$$Speedup_A = \frac{90}{25 + 15 + \frac{50}{2}} = 1.38x \quad Speedup_B = \frac{90}{50 + 15 + \frac{25}{2}} = 1.16x \quad Speedup_C = \frac{90}{50 + 25 + \frac{15}{2}} = 1.09x$$

Amdahl's Law

Let's Analyze: Multigrid Case.

Time (s)	**Grid0**	**Grid1**	**Grid2**	**Grid3**	Total
Smoothing	35.711	7.913	2.047	0.224	45.895
Residual	9.819	2.338	0.557	0.070	12.784
Restriction	1.776	0.425	0.113	0.000	2.315
Prolongation	0.000	12.057	2.855	0.750	15.661
L2Norm	2.150	0.000	0.000	0.000	2.150
Total	49.456	22.733	5.572	1.044	79.525