Project Topics and References

How to pick a project

Tool vs Application

1. *Approach ONE:*
    - *Find a topic and identify an algorithm needed to parallelize*
2. *Approach TWO:*
    - Find an interesting parallel algorithm and apply it to topic!

Ok, BE REALISTIC TO GET GOING!

> The art of doing mathematics consists finding that special case which contains all the germs of generality.

David Hilbert
Mathematician, Physicist, Philosopher

Author of Geometry and the Imagination (1932)

Projects Topics

1. Heat Equation or Image processing (Chapter 11)
2. Better Solvers Red-Black vs MG vs CG Solvers (Chapter 10)
3. Molecular Dynamics — Neighbor Graphs
4. FFT and Convolutions (Chapter 9)
5. Random Graphs and Cluster Monte Carlo for Ising
6. Triangulated Lattices and Integration.
7. Quantum Computing on Linear Algebra

Projects Guidelines

1. Identify general ***topic*** discuss in lectures with team
2. Identify critical ***algorithm*** and class:
   - Laplacian on a 2d Grid - How to run
   - Molecules Dynamcs (balls) moving in 2d box
   - Random bits on Lattice — Monte Carlo
   - Other lattices. Triangular or random
3. ***Parallelize*** with MPI or OpenACC et al.
4. ***Timers or counters*** to analyzed performance (with errors?)
   - for different algorithms, sizes (N) and parallelization methods
5. GitHUB with
   - README, Code, Makefile Tables and Graphs and Slides and Report.

CHOSE FROM BIG 4

I. **MD** (Molecular Dynamic) Short Range in MPI

II. **Solver** MG vs CF vs RB on OpenACC

III. **Ising** Cluster Monte Carlo Graphs Algorithms

IV. **Heat Flow** in MPI

Molecular Dynamics

Improving Neighbor Table

https://en.wikipedia.org/wiki/Lennard-Jones_potential

Sahan Bandara

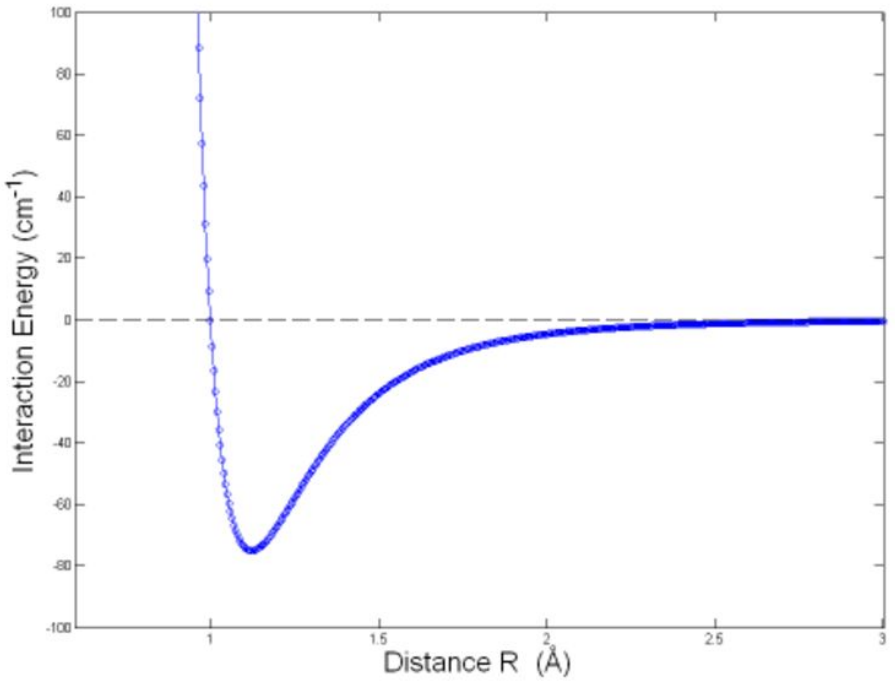
Problem Statement

- Molecular dynamics systems = very common model
  - Two-body interatomic interactions (Lennard-Jones potential function)
  - Periodic boundaries to represent “infinite” space
- Brute Force -> $O(N^2)$ performance when comparing atom pairs
- Improvement: Verlet Table Algorithm / Cell Linked List Algorithm
- Further improvement: https://arxiv.org/pdf/physics/0311055.pdf
  - Combines above techniques
  - Uses additional techniques to improve table update frequency/memory organization
- How does parallelization of system work?

Molecular Dynamics Basics

- Interaction between particles in a “infinite” space
  - Utilize periodic boundaries
- Conservation of total energy
  - $E = E_{kin} + E_{pot}$
  - $E_{pot} = \sum_{i=1..N}\sum_{j>i} e_{pot}(r_{ij})$ where r_{ij} = distance between particles “i” and “j" (Equation 2.2)[1]
- Lennard-Jones Potential
  - Neutral particles
  - Repulsion Forces (no collisions)

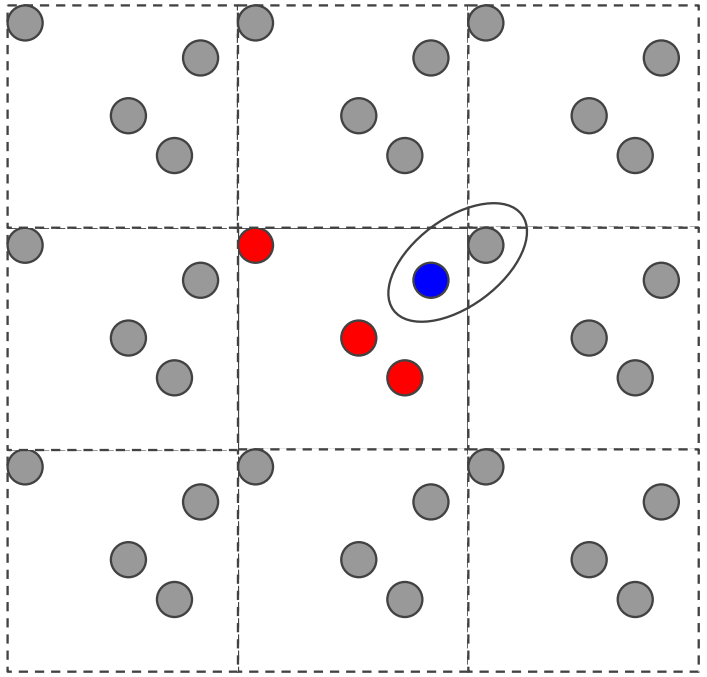
Simulation of Lennard-Jones potential [2]



[1] Basics of Molecular Dynamics. Available from: C2_for.pdf
[2] Six Degree-of-Freedom Haptic Rendering for Biomolecular Docking - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/The-simulation-of-Lennard-Jones-potential_fig1_220110295 [accessed 1 May, 2018]

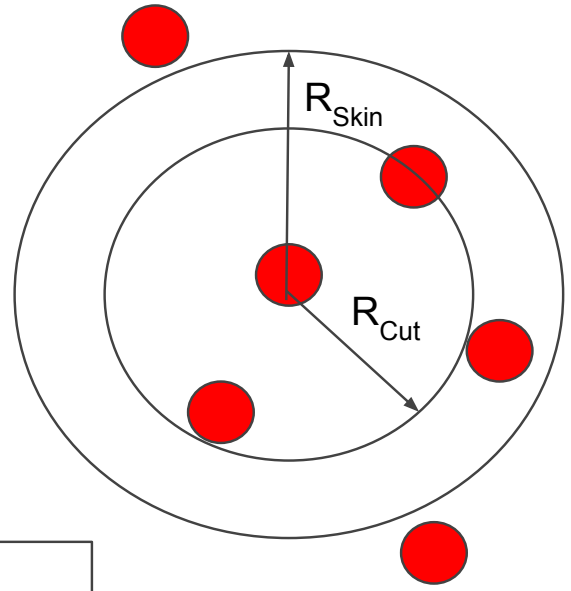
Periodic Boundaries

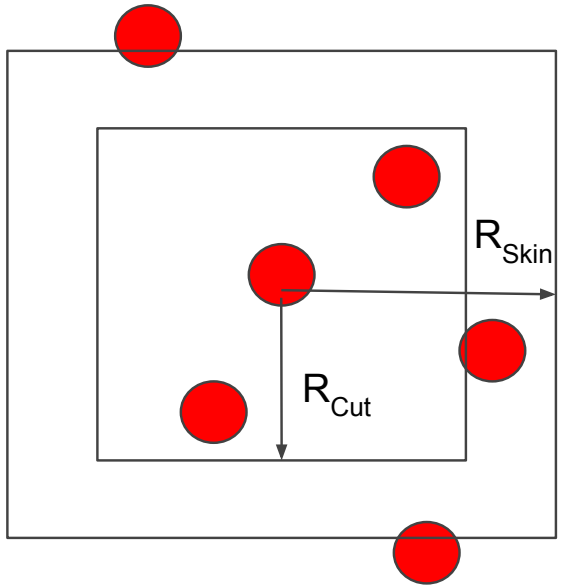
- Avoid surface effects
- Simulation particles see “images”
  - Image particles follow movement of simulation particle
- Pair interaction with closest particle
  - Minimum Image Criterion
- Constant number of particles
  - Simulation exit -> Image enter



Neighbor Table

- R_{Cut} -> Potential Cutoff
- R_{Skin} -> Considered Particle
- Circular boundaries
  - Used Square boundaries
- Data Structure for N particles:
  - Array (size N) of...
    - Pointers
    - Linked-Lists
    - Vectors
- Vector chosen for simplicity
  - Not efficiency
- Table reconstructed a set number of iterations





II. Solver MG vs CG vs RB

Solver Performance using Multigrid vs CG et al

See for CG https://authors.library.caltech.edu/records/e7fwb-j3238

https://en.wikipedia.org/wiki/Conjugate_gradient_method

https://en.wikipedia.org/wiki/Multigrid_method

A Linear Algebra Problem

- Ax = b
- Sparse matrix

- Very small eigenvalues (~0.0001)
  - Jacobi, Gauss-Seidel take too long
  - Conjugate gradient also takes too long
- Multigrid is a solution

What is a Multigrid?

- It is a recursive divide and conquer method used in many fast algorithms for computer science tasks and numerical methods for high performance computing.

- The main idea of multigrid is to accelerate the convergence of a basic iterative method (known as relaxation) by a *global* correction of the fine grid solution approximation from time to time, accomplished by solving a coarse problem.

III. Ising Random Cluster

Parallel Multigrid Percolation Cluster Detection

https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.62.1087

and lot of cool graph theoretic problem from Brower et al !

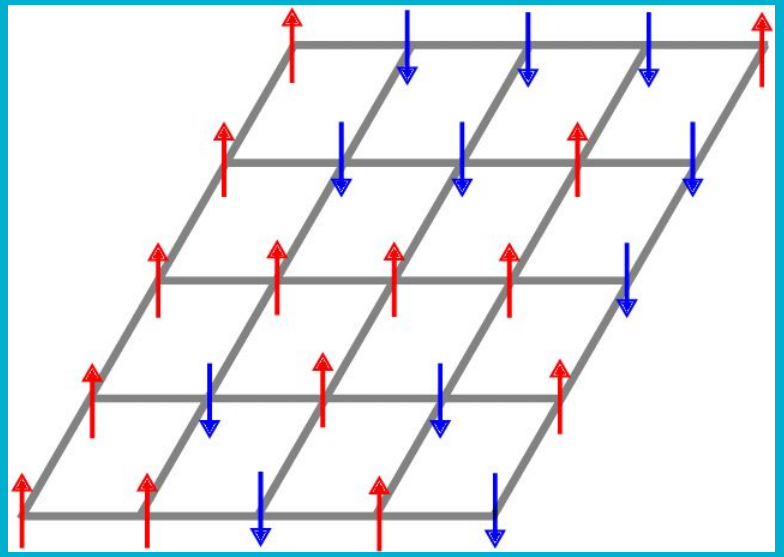
Ising Percolation Cluster Problem

Magnetic polarities tend to align.

We can count the energy of a plane of magnetic poles where we sum for each bonded/neighboring pair of poles -1 if they are the same and +1 if they are different by creating this as an undirected graph

We can relax this grid with the Swendsen-Wang algorithm by ‘percolating’ random bonds then randomly flipping where these bonded poles match

From a point we can serially find a cluster of like poles by BFS or DFS



Multigrid Algorithm

We are looking rather at relaxing based on a fixed point. We can derive the max distance of points in its cluster.

With a single V cycle of multigrid fine-coarse-fine we can distinguish where prior separate clusters can be combined by expanding a connectivity matrix by powers of two each iteration.

$$\Delta_{ij}^{(l)} = \begin{cases} M_{ij} & \text{if } i-j \text{ is a distance } \pm 2^l \text{ in a coordinate direction} \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

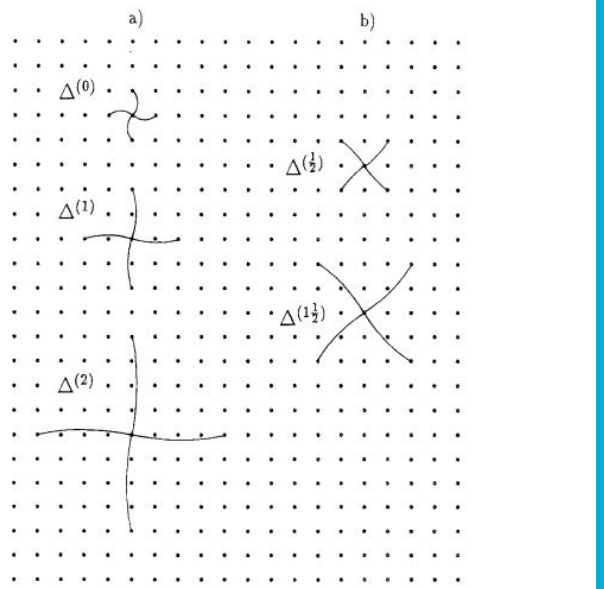


Fig. 1. (a) Multigrid connectivity matrices $\Delta^{(l)}$ connect neighbors at powers-of-two distances along coordinate axes. (b) The connectivity matrices for half-integer levels improve convergence by connecting neighbors along diagonals.

IV. HEAT EQUATION

(See Lect 11 in Class Lectures)

BACK UP PROJECTS

Suggestions if you are the are interesting to you!

Many start with analyzing a random graph: Easy to find problems to optimize with random guesses (e.g Monte Carlo moves relative to objective probability function.

(1) TSP, Coloring, Clusters for images etc.

EC500 E1: Parallel Programming and HPC

Parallelizing 2D Convolutions on GPUs

What is a Convolution?

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

A 1D convolution is a linear combination of two functions: ***f,*** some **signal** that you have recorded and ***g***, a **filter** to process the signal data in some desired fashion (i.e. smoothing, edge detection, or lowpass). You can extend the convolution into 2, 3, … N dimension. Convolution is a common data processing method that is used in a wide variety of domains.

The 2D convolution has a worst case time complexity of **$O(n^4)$**. So finding methods to improve performance is key.

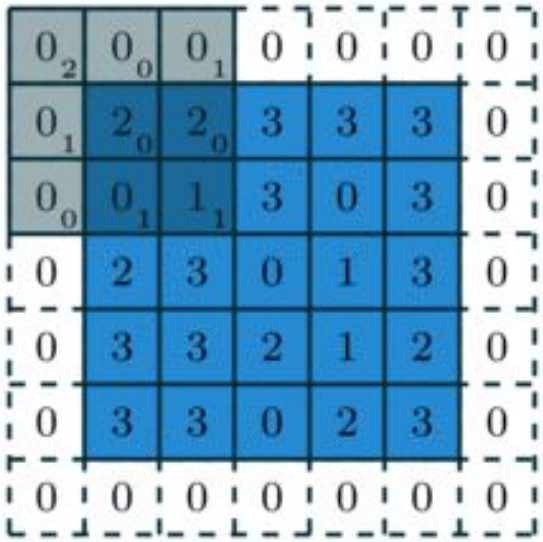
Goal

The goal of our project is to work with 2D convolution, a widely-used and simple, yet computationally inefficient algorithm, and explore methods to speed it up by implementing parallelization with GPU's

Work Planned vs Work Accomplished

- Build the 2D convolution function, FFT and the resulting matrices in both serial and parallel fashion
  - Building the 2D convolution matrix follows a specific pattern and can be parallelized
  - Reshaping the image, performing matrix math, and reshaping back to original size are all also parallelizable operations.
- Compare direct convolution to FFT2
  - Parallelize FFT
  - Test both parallelized methods and compare/contrast for different inputs (different sized images and kernels)
- Use OpenACC for GPU parallelization
  - Use OpenACC on the SCC

The 2D Convolution Example



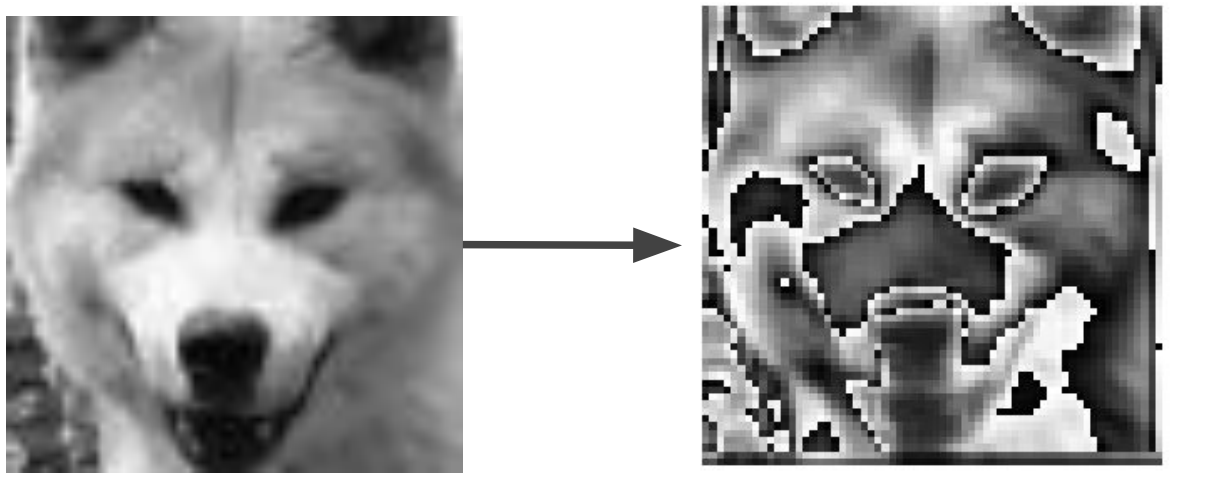
1	6	5
7	10	9
7	10	8

$$\mathbf{g}_{mn} = \sum_{m'=0}^{N-1} \sum_{n'=0}^{N-1} \mathbf{a}_{m-m',n-n'} \mathbf{f}_{m',n'}$$

2D convolutions generate new representations of the original data matrix based on a linear combination of its elements with that of a filter matrix or “Kernel” as it moves through the original data.

One relevant reason for exploring parallelizing this operation is its wide use in modern computer vision and current deep learning applications.

The 2D Convolution Example



64x64 grayscale input image

64x64 edge detection output image

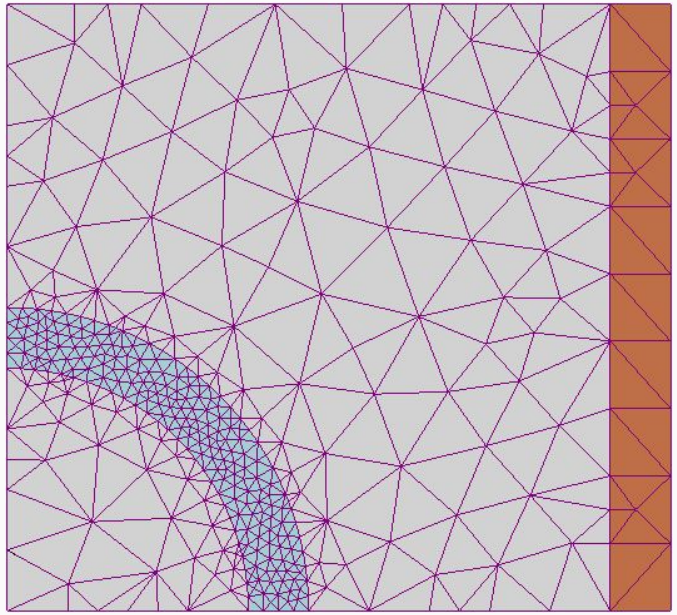
Example of using our 2D convolution function for edge detection on a 64x64 grayscale image of a dogs using a 3x3 local filter.

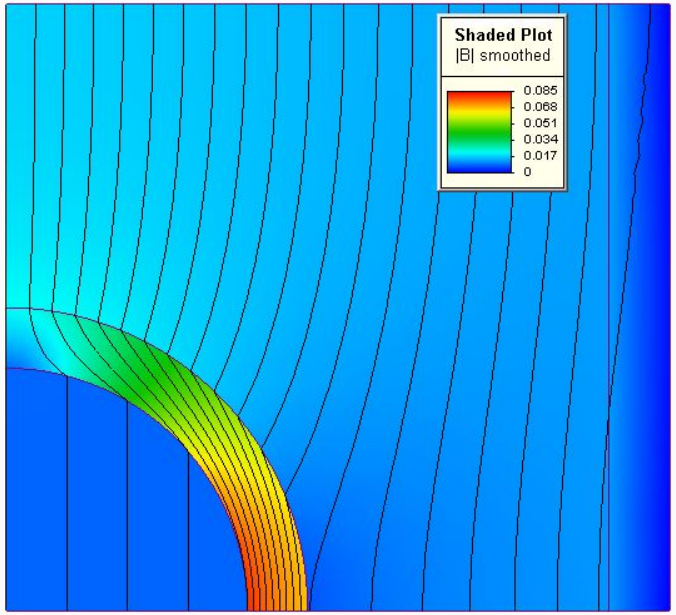
Triangulated Lattice and Integration.

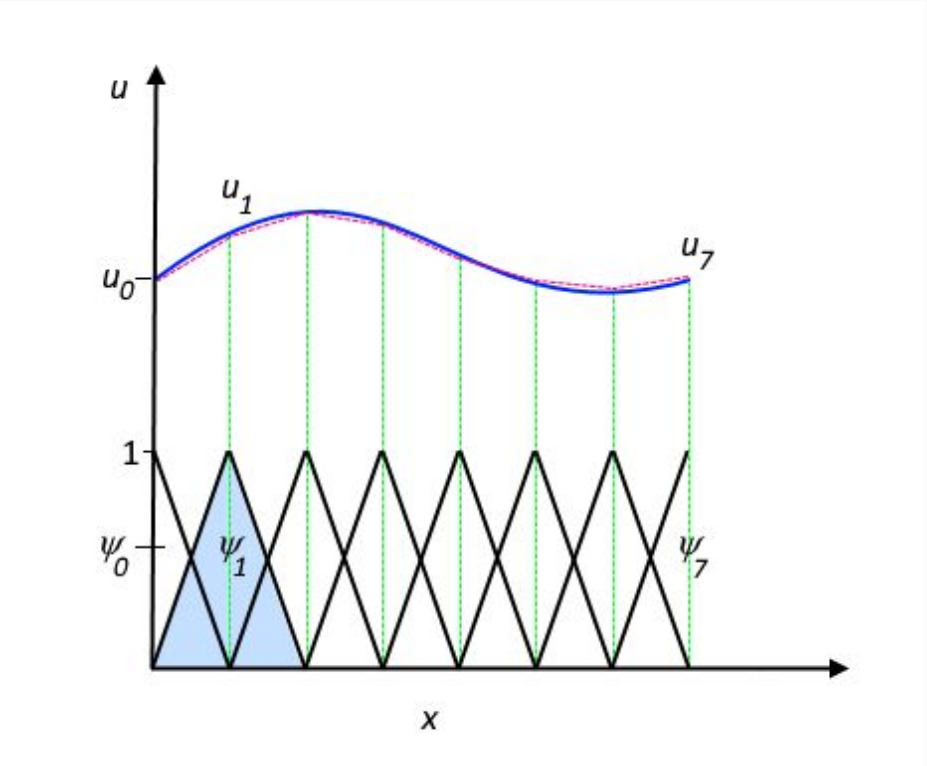
Spectrum of Finite Element Method Laplace operator on irregular triangular lattice

... or is it...?

Finite Element Method - A 5s Background







[*Finite Element Method*, WikiPedia]

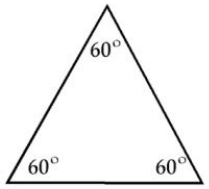
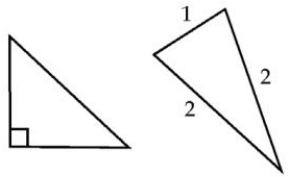
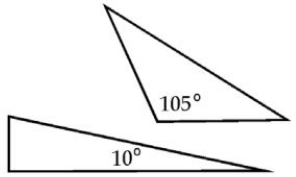
The ambiguous case of ‘fine’ elements

8.2 General Element Quality Checks

Element quality is a subject often talked about and never fully understood. The reason for this is complex but is related to the fact that quality is relative and the solution, by definition, is approximate. In the formulation of finite elements a local parametric coordinate system is assumed for each element type and how well the physical coordinate systems, both element and global, match the parametric dictates element quality. Below you see some graphics representing element quality and you should attempt to follow them, however, there will be a point of diminishing return if you try too hard to get every element within the acceptance criteria. Your judgment is your only guide in those cases. Always perform quality checks on the meshes you create. Check with “local experts” regarding the appropriate values for each element type required by your element checking computer programs. Be aware that, in these situations, “correct” answers can vary a great deal as illustrated in the following table where the range between “OK” and “very poor” is quiet wide.

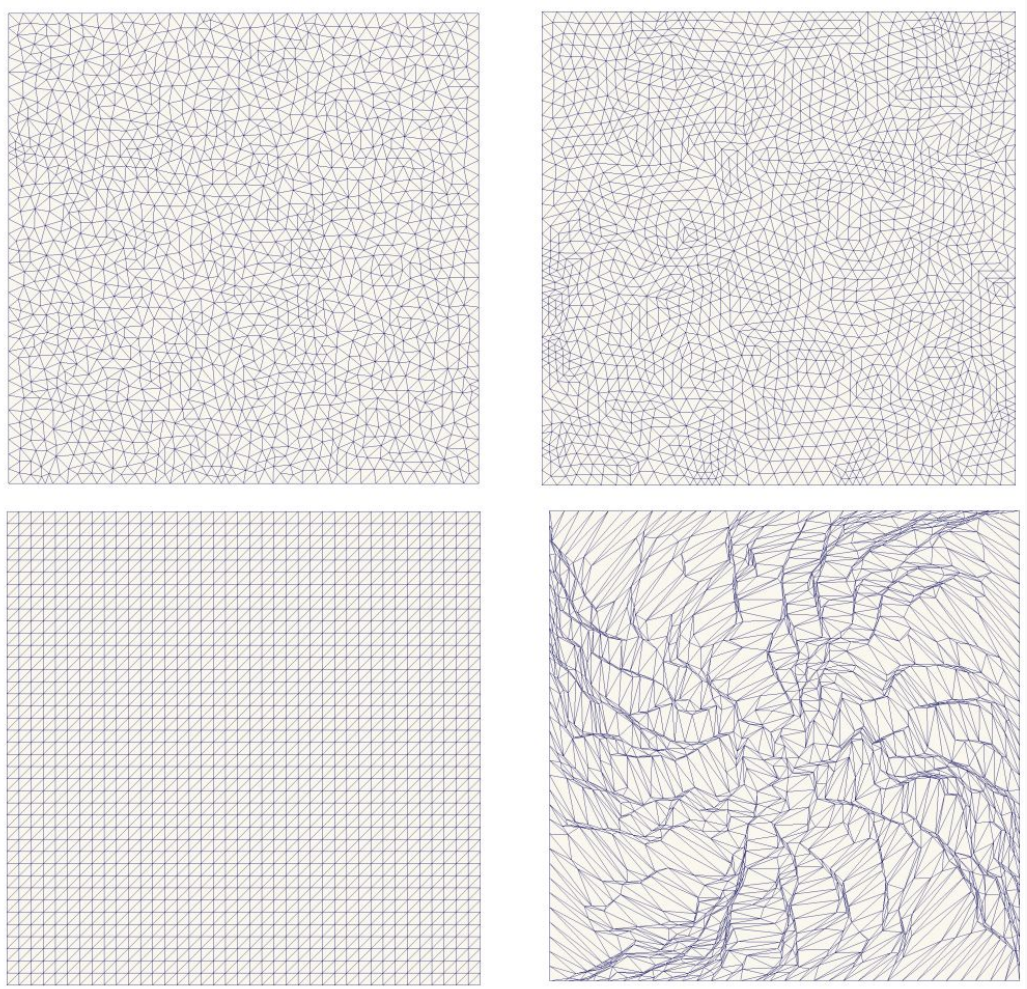
Solid elements use the determinant of the Jacobian Matrix and compare to the ideal value.

Some common element quality measures are detailed below:

BEST	OK	VERY POOR
		

[*Practical Finite Element Analysis*, Nitin S. Gokhale, 2008]

Numerical test in literature



[*Practical Finite Element Analysis*, Nitin S. Gokhale, 2008]

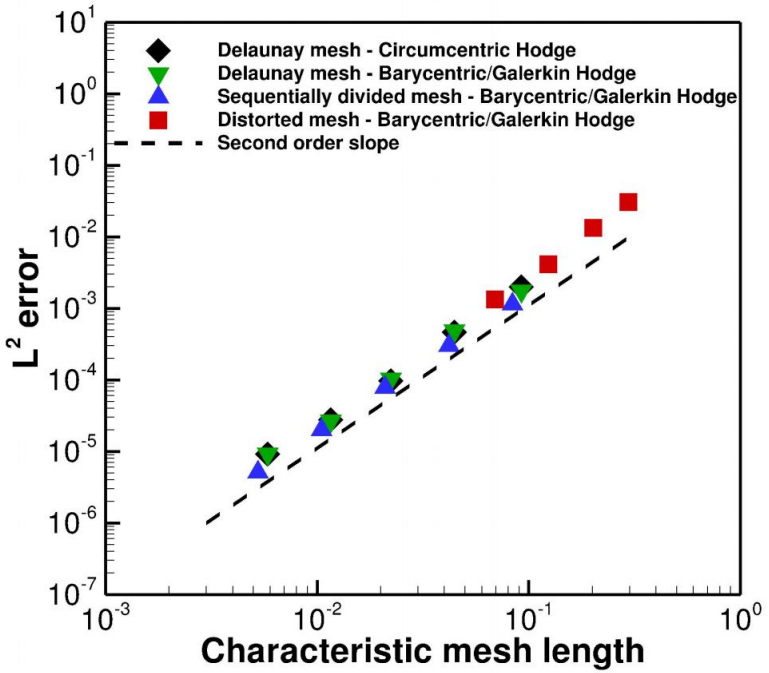
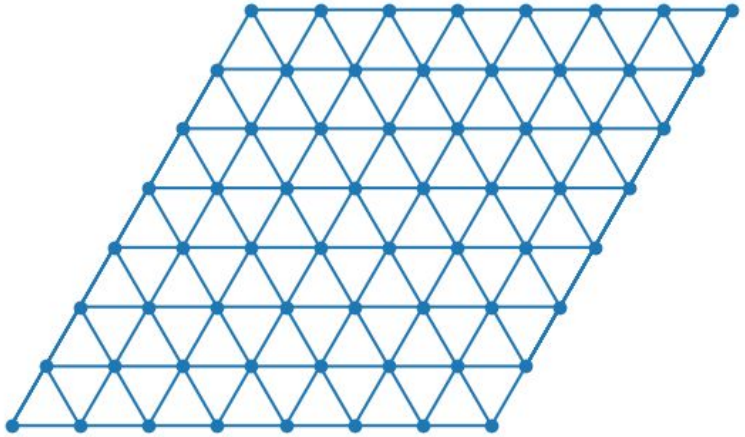
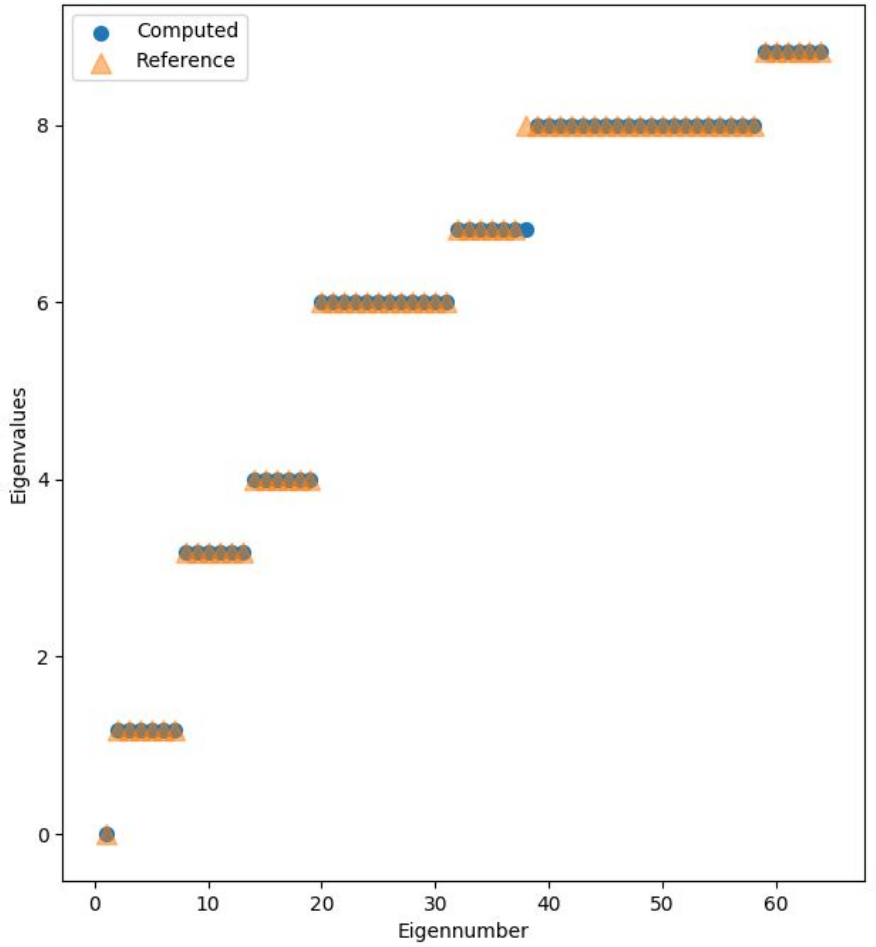


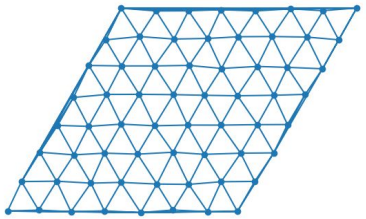
Figure: The numerical convergence of the L^2 error of p for the scalar Poisson Eq. with Neumann boundary conditions.

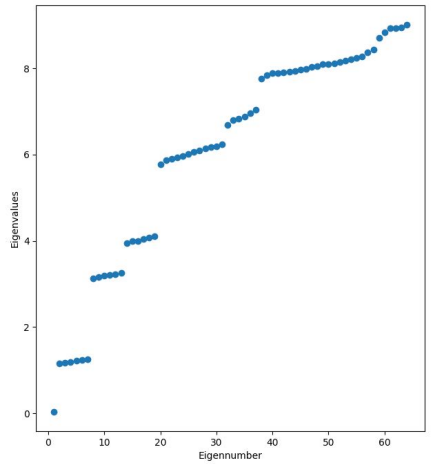
Numerical Test - Laplace operator on Triangular grid

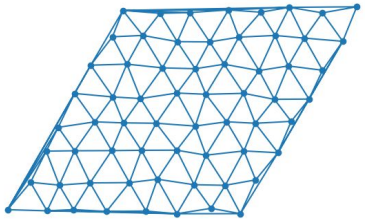


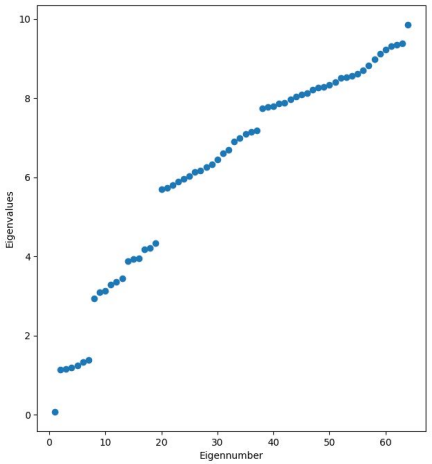


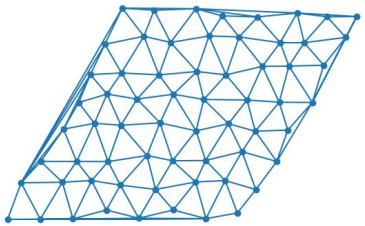
Unstructured Mesh testing

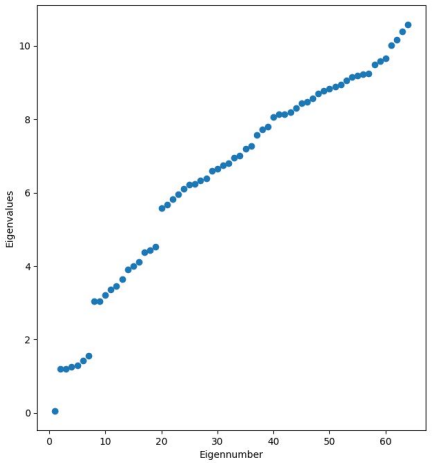


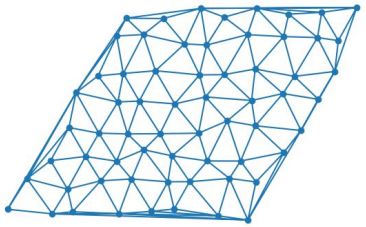


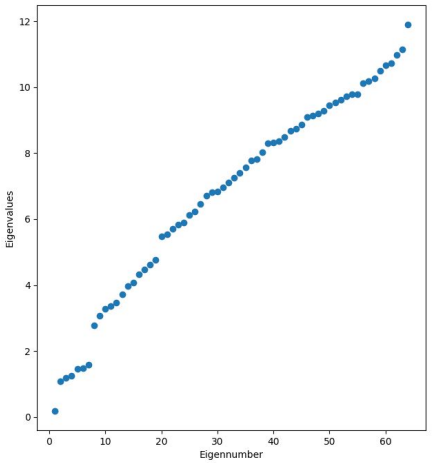












Traveling salesman Problem

TSP: Introduction

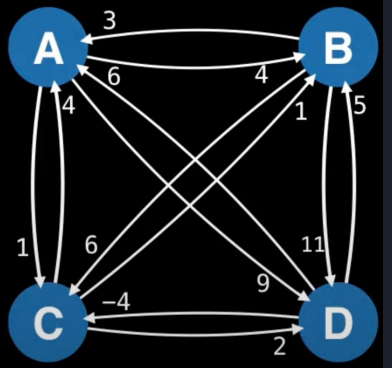
Let $L=\sum \sum x(i, j)\, d(i, j)$; We want to find:

$$L^* = \min \sum \sum x(i, j)\, d(i, j)$$

In other words - given a complete graph, find the shortest **Hamiltonian cycle** in the graph.

NP-complete - no known efficient (i.e., polynomial time) algorithm.

	A	B	C	D
A	0	4	1	9
B	3	0	6	11
C	4	1	0	2
D	6	5	-4	0



Proposed implementation of algorithms

Naive solution/Dynamic Programming

1. *(Exact) Naive: Brute force search over N!*
2. (YZ)(Exact) Held - Karp (Dynamic Programming) O(n^2 2^n)
3. (TY) (Approximate) Nearest Neighbour
4. (SW)(Approximate) Lin - Kernighan
5. (WZ) (Approximate) Christofides - 1.5* L*
6. (NS) (Approximate) Simulated Annealing
7. (NS) (Approximate) Ant colony optimization

Algorithm Main Ideas

1. **Naive - O(n!)**

   Try all permutations and see which one is the shortest path (using brute force search).

2. **Held-Karp - $O(n^2 2^n)$**

   Based on dynamic programming and solve TSP “bottom-up”.

   $G(i, s) = \min \{ C_{ik} + G(i, s-\{k\})\}$

3. **Nearest Neighbour**

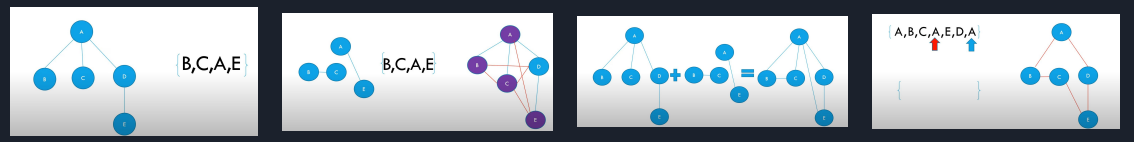
   One of the first algorithms solves TSP approximately. The salesman starts at a random city and repeatedly visits the nearest city until all have been visited.

Algorithm Main Ideas

1. **(Approximate) Lin - Kernighan O(N^2.2)**

   Lin–Kernighan is one of the best heuristics for solving the symmetric travelling salesman problem. It belongs to the class of local search algorithms.

2. **(Approximate) Christofides** - 1.5* L*: If the cities follow the triangle inequality, which is a+b>c, that this algorithm has 3/2 approximation guarantee.



3. **(Approximate) Simulated Annealing** - arrive at solution by (~exhaustively) exploring the state space
   - Random, iterative improvements to the path length
   - Always accept improvements; decaying probability of accepting setbacks to encourages state exploration
4. **(Approximate) Ant colony optimization** - based on behavior of real-life ant colonies
   - Randomly explore space, leaving “pheromones” to help later ants find good paths

Project Design Requirements

Program language: C++, Python (Networkx for visualization).

Dataset: TSPLIB

Test plan: Use (some) of the proposed algorithms to generate approximate solutions to various TSP datasets.

- How far from the optimal are the resulting paths?
- Performance of each algorithm?
- Statistics for the randomized algorithms
- Agreement with theoretical upper and lower bounds?

References:

For some introductory reading:

https://en.wikipedia.org/wiki/Travelling_salesman_problem
https://en.wikipedia.org/wiki/Christofides_algorithm

Detailed Algorithmic Implementations

"Simulated Annealing", Per Brinch Hansen June 1992
Ant Colony Optimization, Marco Dorigo and Thomas Stützle

Dataset Documentation

https://www.math.uwaterloo.ca/tsp/data/index.html
http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

Connected Components or Coloring Problem

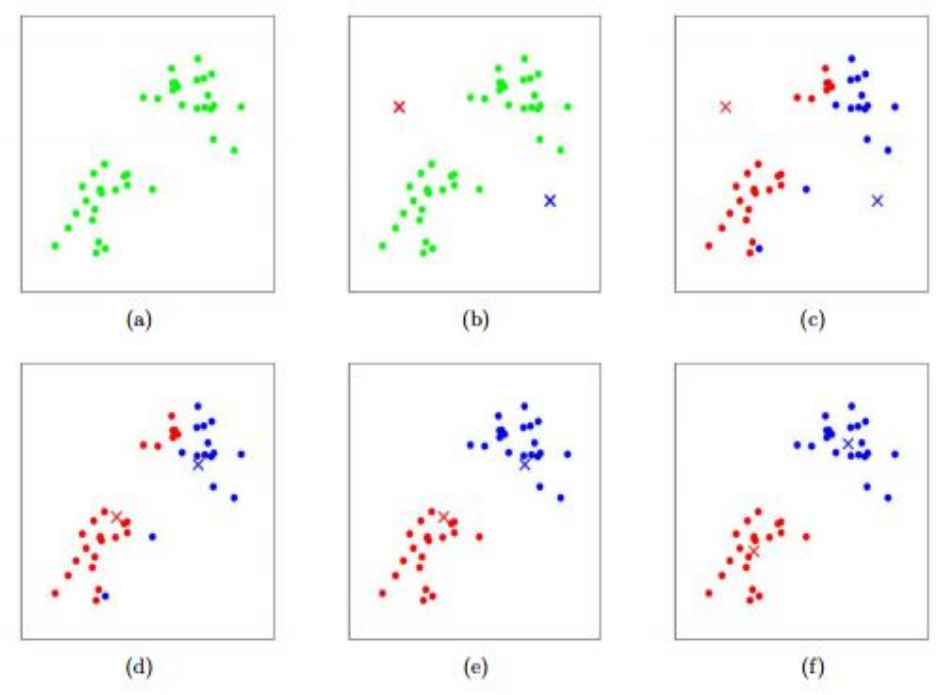
K-means Clustering Algorithm

—

Andreas Francisco
Tu Timmy Hoang

Algorithm

1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

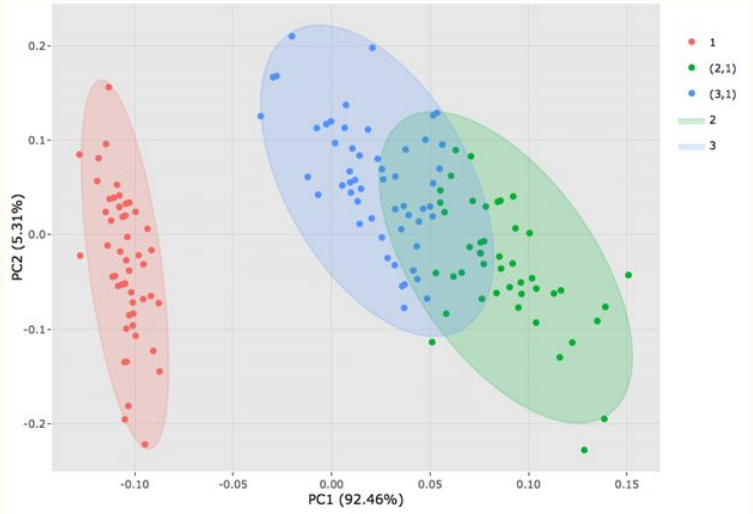


Parallelization

- Parallelize each distance from a centroid using openACC or openMPI.
- Reducing data computation by splitting data.
- Possibly using buffers to deal with edge cases in data division.

Fuzzy Algorithm

- Now we want to partition the points into Cj sets but points can be in more than one set so the representation of each point is a sum of the likelihood a point belongs to each cluster Cj.
- Continue computing the new centroids until the coefficient between two iterations of a point being in a cluster Cj has not changed more than epsilon.
- Compute the final centroids of each cluster and return the solution.
- Parallelize it the same as regular Algorithm.



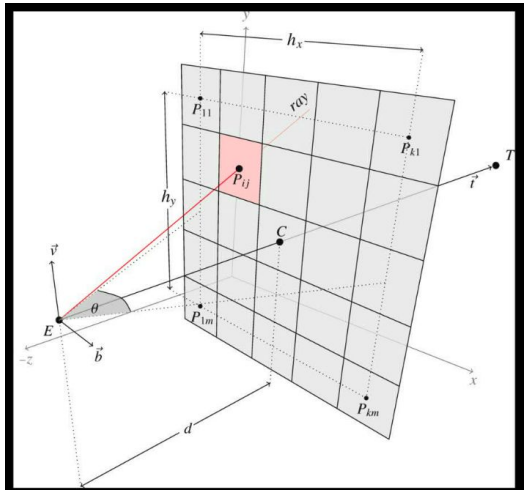
Comparison

- Check speed of convergence for the serial implementation of Hard Clustering and Fuzzy Clustering.
- Compare the accuracy of the solutions between them using a message passing interface.

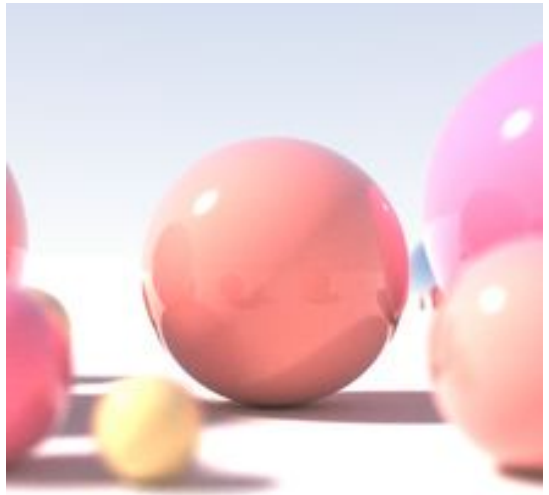
RayTracing

Description

Ray Tracing: Technique for producing high-fidelity images, where light rays are simulated entering the “eye” (viewport) for each pixel on the screen as they interact with the scene



Wikipedia contributors, "Ray tracing (graphics)," *Wikipedia, The Free Encyclopedia,* https://en.wikipedia.org/w/index.php?title=Ray_tracing_(graphics)&oldid=1078731359 (accessed March 31, 2022).



Goals

- Write scalar and parallel versions of ray tracing code to produce an image
- Determine the speedup when these different methods are used (number of threads/processes, MPI vs OpenACC, etc.)

Methods

- Follow the Ray Tracing in One Weekend tutorial to create scalar code
- Using this baseline, write OpenACC and MPI equivalents
- Measure the runtime for each on a variety of scenes, thread counts, etc.

Quantum Computing on the IBM simulator

Brower can give an introductory lecture but …..

THE ARE COMING SOON TO YOUR NEIGHBORHOOD

EuERA Cambridge/Harvard/MIT start up

https://www.quera.com/

IBM at RPI

https://www.ibm.com/quantum/technology