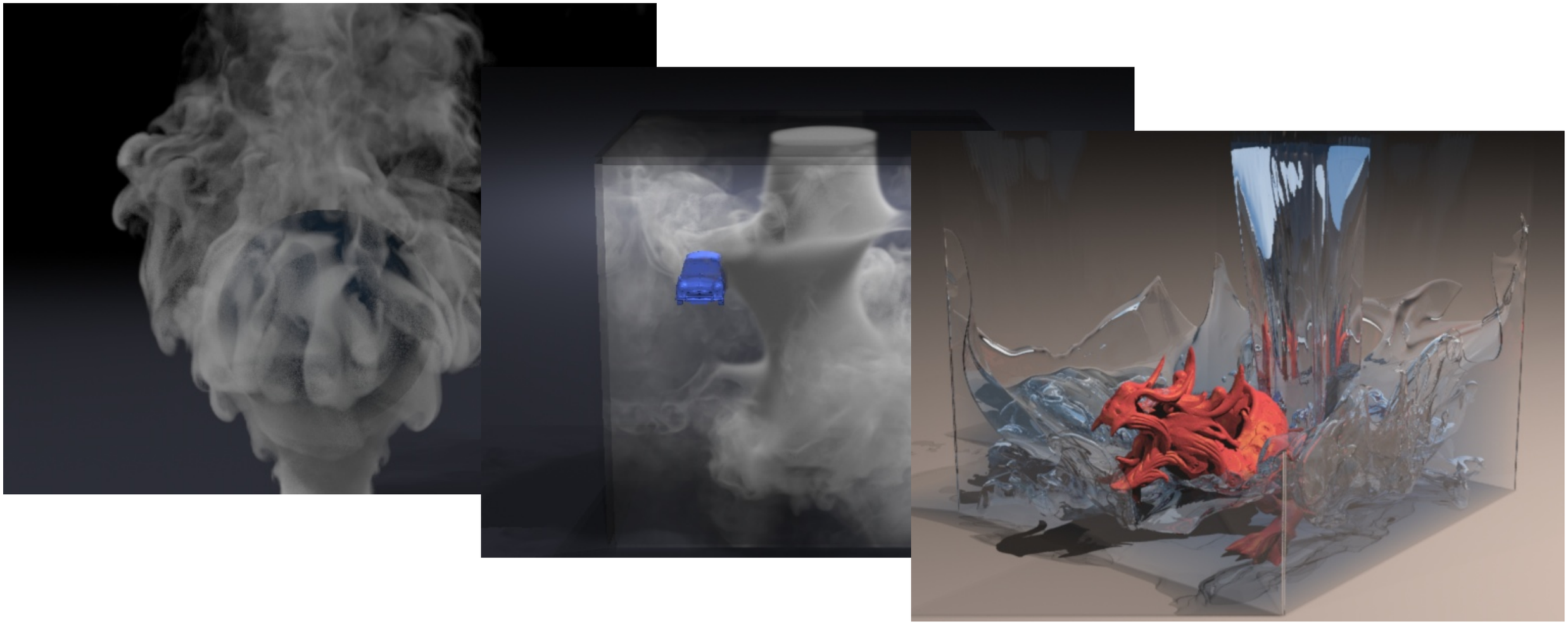


Parallel programming practices for the solution of Sparse Linear Systems (motivated by computational physics and graphics)

Eftychios Sifakis

CS758 Guest Lecture - 19 Sept 2012

Introduction

- Linear systems of equations are ubiquitous in engineering
- Their solution is often the bottleneck for entire applications
- NxN systems Ax=b arising from physics have special properties:
  - Almost always : Sparse - O(N) nonzero entries
  - Very often : Symmetric
  - Often enough : Positive definite, diagonally dominant, etc.
  - Dense matrix complexity ~$O(N^{2.38})$ does not apply
- For special matrices, solvers with cost as low as O(N) exist
  - The "hidden constant" becomes important, and can be lowered via parallelism
  - Efficient algorithms need to be very frugal with storage

Introduction

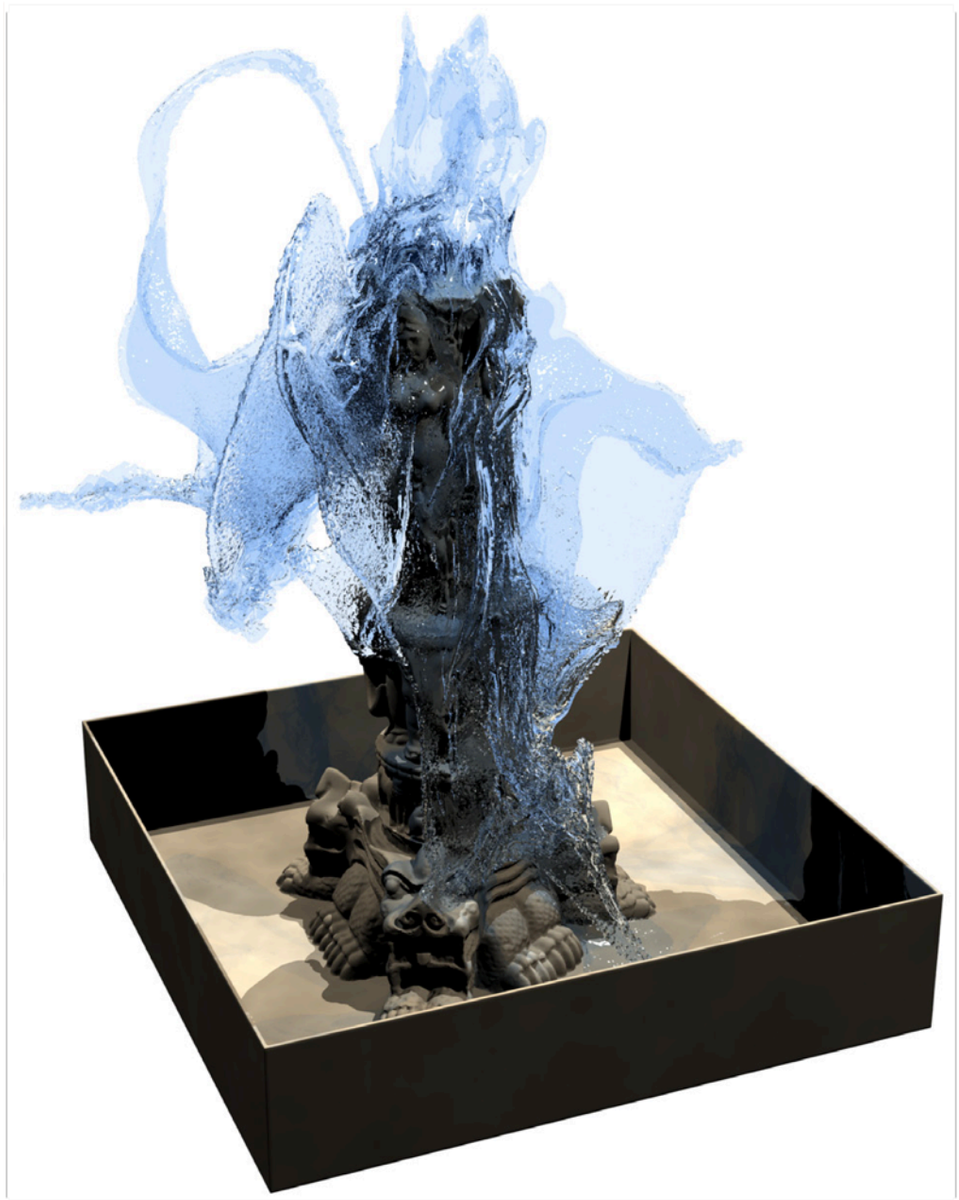
Example : The 3D Poisson equation

$$
\begin{pmatrix}
-6 & 1 & & & 1 & & & 1 & & & \\
1 & -6 & 1 & & & 1 & & & \ddots & & \\
 & 1 & -6 & 1 & & & \ddots & & & \ddots & \\
 & & \ddots & \ddots & \ddots & & & \ddots & & & 1 \\
1 & & & \ddots & \ddots & \ddots & & & \ddots & & \\
 & 1 & & & \ddots & \ddots & \ddots & & & \ddots & \\
 & & \ddots & & & \ddots & \ddots & \ddots & & & 1 \\
 & & & \ddots & & & \ddots & \ddots & \ddots & & \\
1 & & & & \ddots & & & \ddots & \ddots & \ddots & \\
 & \ddots & & & & \ddots & & & 1 & -6 & 1 \\
 & & \ddots & & & & 1 & & & 1 & -6 & 1 \\
 & & & 1 & & & & 1 & & & 1 & -6
\end{pmatrix}
\mathbf{x} = \mathbf{b}
$$

Applications

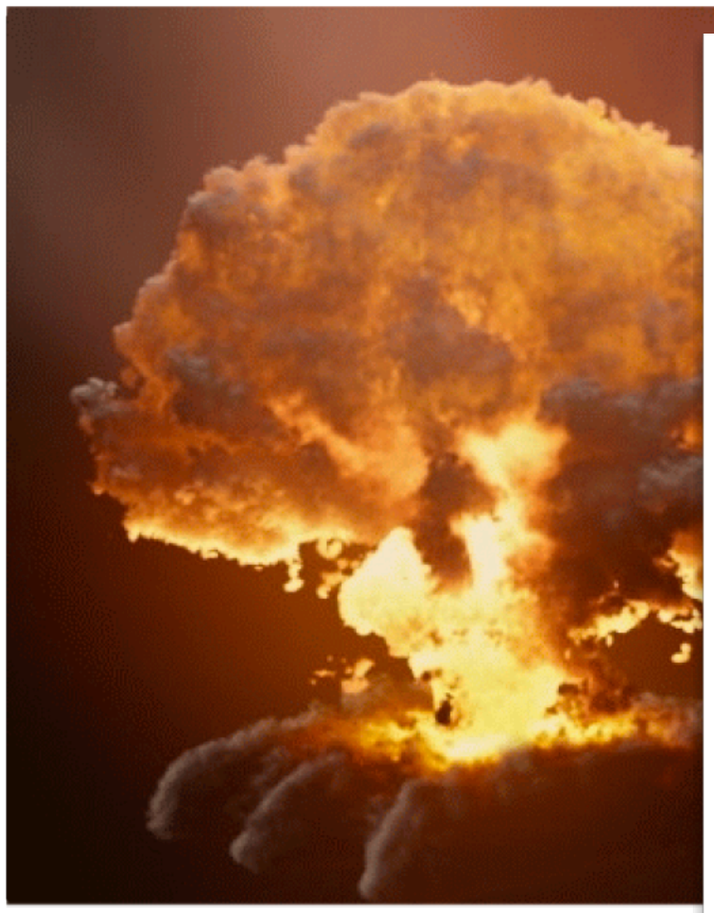
Rasmussen et al, Smoke Simulation for Large Scale Phenomena (SIGGRAPH 2003)

Applications



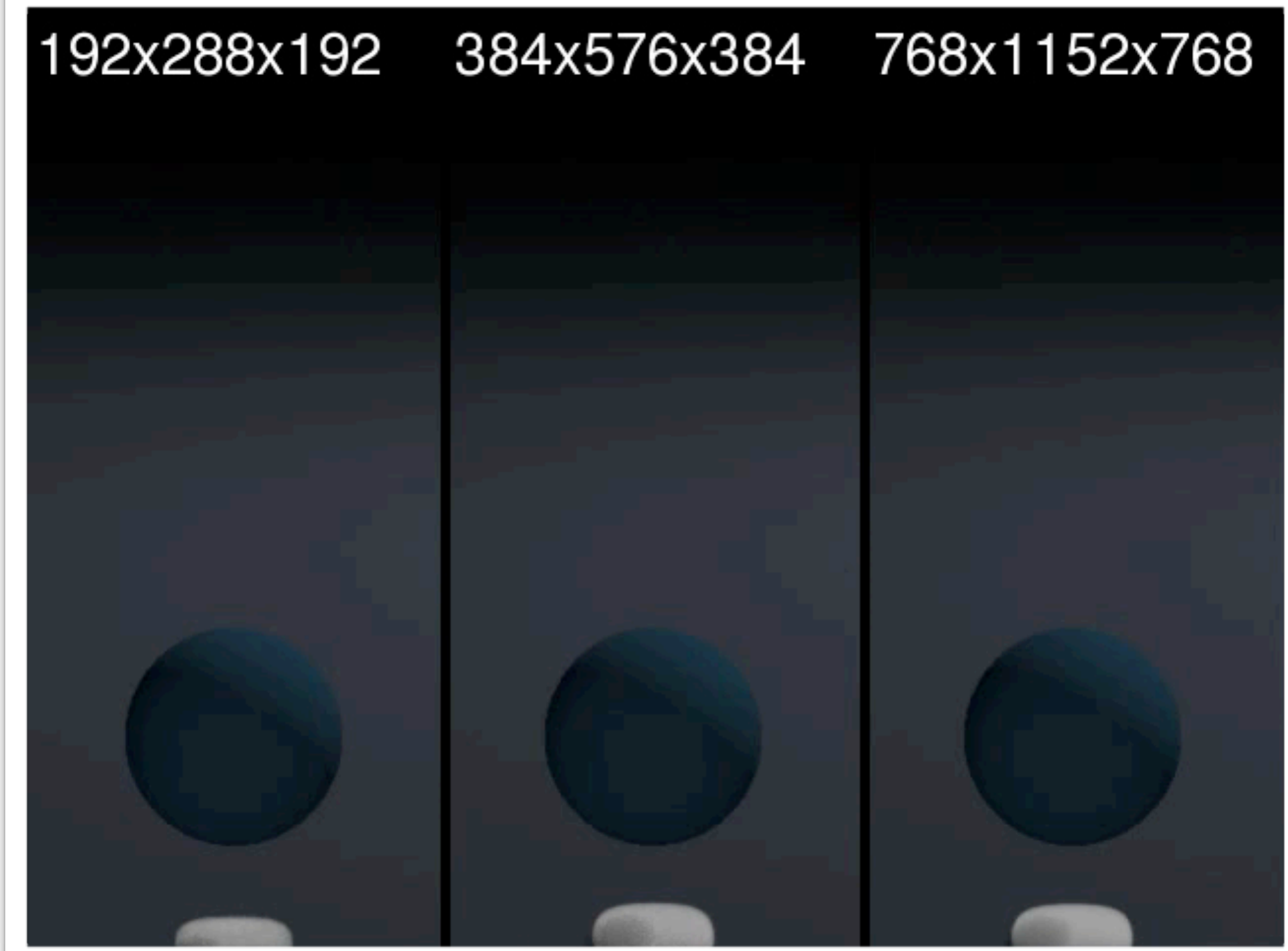
Nielsen et al, Out-Of-Core and Compressed Level Set Methods (ACM TOG 2007)

Applications



Horvath & Geiger, Directable, high-resolution simulation of fire on the GPU (ACM SIGGRAPH 2009)

Applications



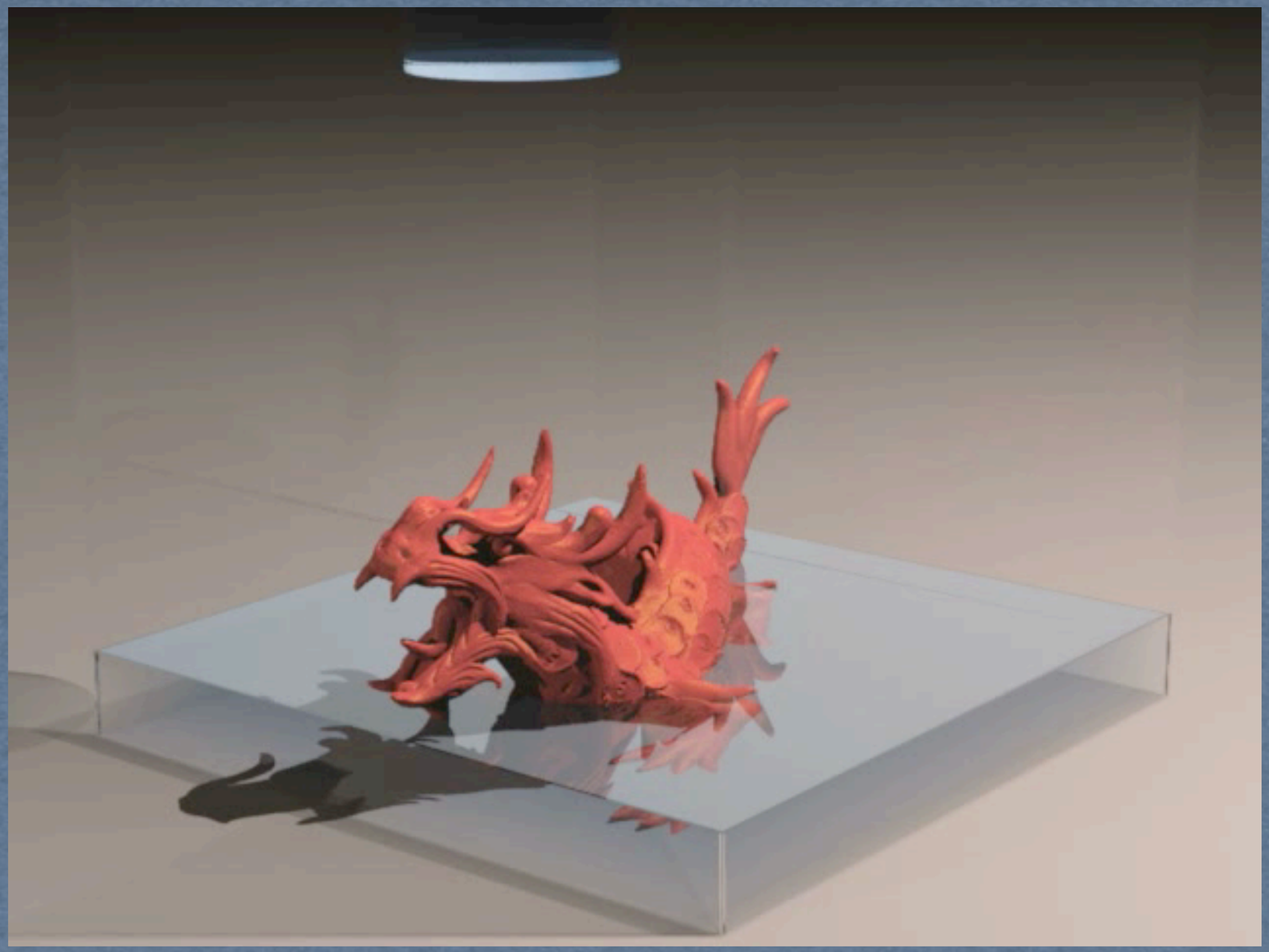
Introduction

700M equations
40 sec on 16 cores
16GB RAM required

Introduction

450M equations
25 sec on 16 cores, 12GB RAM required

Introduction



135M equations
12 sec on 16 cores, 3GB RAM required

Introduction

How good (or not) are these runtimes?

- For one of these problems the system ***Ax=b*** has
  - 700M equations, 700M unknowns
  - Storing ***x*** (in float precision) requires 2.8GB. Same for ***b***.
  - The matrix ***A*** has 7-nonzero entries per equation.
  - MATLAB requires ~40GB to store it, in its "sparse" format
- Our case study computes the solution
  - With ~40sec of runtime (on a 16-core SMP)
  - While using <16GB of RAM

    (caveat : it avoids storing ***A***)

Development Plan

Design

- Define your objectives
- Choose a parallel-friendly theoretical formulation
- Set performance expectations
- Choose a promising algorithm

Implement

- Implement a prototype
- Organize code into reusable kernels

Accelerate

- Reorder/combine/pipeline operations
- Reduce resource utilization (try harder ...)
- Parallelize component kernels

Clarifying objectives

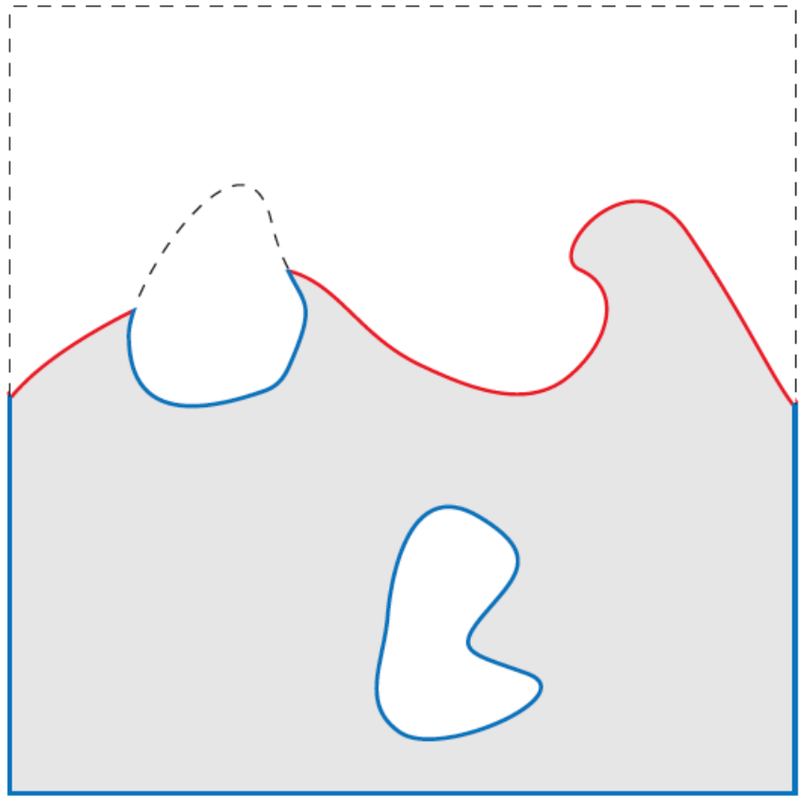
What kind of accuracy do we need?

- Solve ***Ax=b*** down to machine precision?
- Ensure that ***x*** is correct to *k* significant digits?
- Ensure that ***x*** (initial guess) is improved by *k* significant digits?

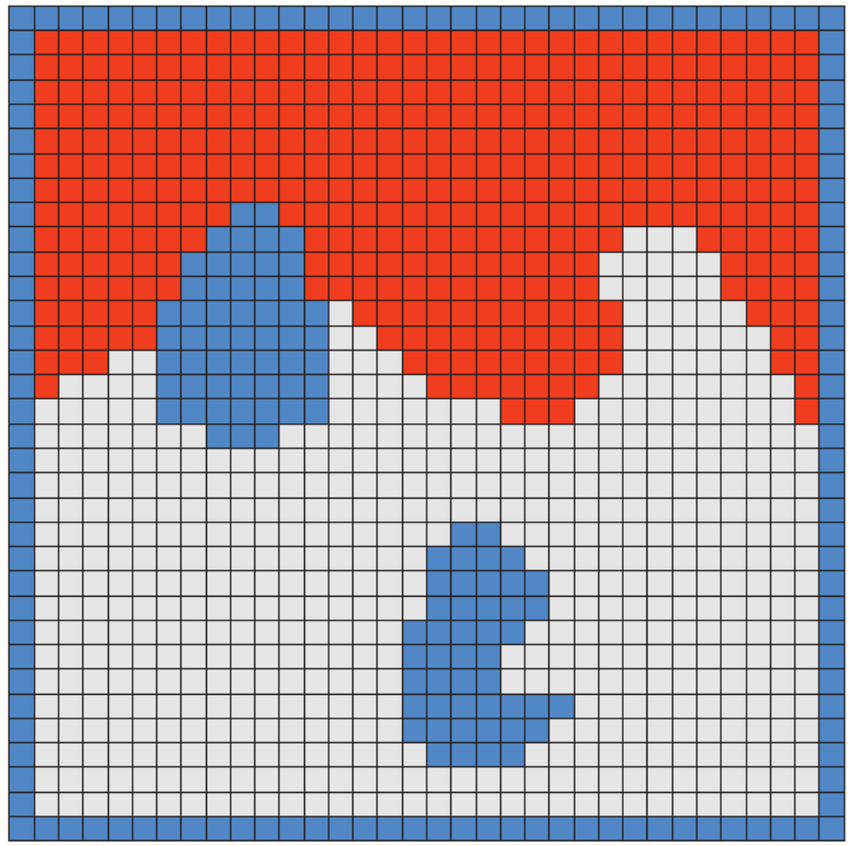
What is the *real* underlying problem we care about?

- The system ***Ax=b*** is rarely the ultimate objective
- Typically, it's means to an end
  - Solve the system *to create a simulation*
  - Solve the system *to generate a solution to a physical law*
- We have some flexibility to make ***Ax=b*** "better" for parallel algorithmic solution

Clarifying objectives

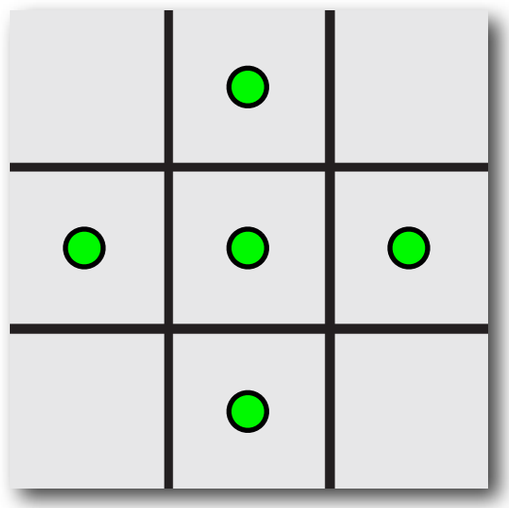


$$\Delta \mathbf{x} = \mathbf{b}$$



$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

$$\frac{-4u_{ij} + u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1}}{h^2} = f_{ij}$$



Development Plan

Design

- Define your objectives
- Choose a parallel-friendly theoretical formulation
- Set performance expectations
- Choose a promising algorithm

Implement

- Implement a prototype
- Organize code into reusable kernels

Accelerate

- Reorder/combine/pipeline operations
- Reduce resource utilization (try harder ...)
- Parallelize component kernels

Speedup vs. Efficiency

Well-intended evaluation practices ...

> *"My serial implementation of algorithm X on machine Y ran in Z seconds. When I parallelized my code, I got a speedup of 15x on 16 cores ..."*

... are sometimes abused like this:

> *"... when I ported my implementation to CUDA, this numerical solver ran 200 times faster than my original MATLAB code ..."*

So, what is wrong with that premise?

Speedup vs. Efficiency

Watch for warning signs:

- Speedup across platforms grossly exceeding specification ratios
  - e.g. NVIDIA GTX580 vs. Intel i7-2600
  - Relative (peak) specifications :
    - GPU has about 3x higher (peak) compute capacity
    - GPU has about 8x higher (peak) memory bandwidth
  - Significantly higher speedups likely indicate:
    - Different implementations on the 2 platforms
    - Baseline code was not optimal/parallel enough
- "Standard" parallelization yields linear speedups on **many** *cores*
  - [Reasonable scenario] Implementation is CPU-bound
  - [Problematic scenario] Implementation is CPU-wasteful

Speedup vs. Efficiency

A different perspective ...

> *" ... after optimizing my code, the runtime is about 5x slower than **the best possible performance** that I could expect from this machine ..."*

*... i.e. 20% of maximum theoretical **efficiency!***

***Challenge :** How can we tell how fast the best implementation could have been? (without implementing it ...)*

Performance bounds and *"textbook efficiency"*

Example : Solving the quadratic equation

$$ax^2 + bx + c = 0$$

What is the *minimum* amount of time needed to solve this?

Data access cost bound

*"We cannot solve this faster than the time needed to read **a,b,c** and write **x**"*

"We cannot solve this faster than the time needed evaluate the polynomial, for given values of a,b,c and x"
(i.e. 2 ADDs, 2 MULTs plus data access)

Solution verification bound

Equivalent operation bound

"We cannot solve this faster than the time it takes to compute a square root"

Performance bounds and *"textbook efficiency"*

What about linear systems of equations?

$$\mathbf{Ax} = \mathbf{b}$$

"Textbook Efficiency"
(for elliptic systems)

*It is **theoretically possible** to compute the solution to a linear system (with certain properties) with a cost comparable to **10x the cost of verifying** that a given value **x** is an actual solution*

... or ...

*It is **theoretically possible** to compute the solution to a linear system (with certain properties) with a cost comparable to **10x the cost of computing** the expression **r=b-Ax***

and verifying that ***r=0***

(i.e. slightly over 10x of the cost of a matrix-vector multiplication)

Development Plan

Design

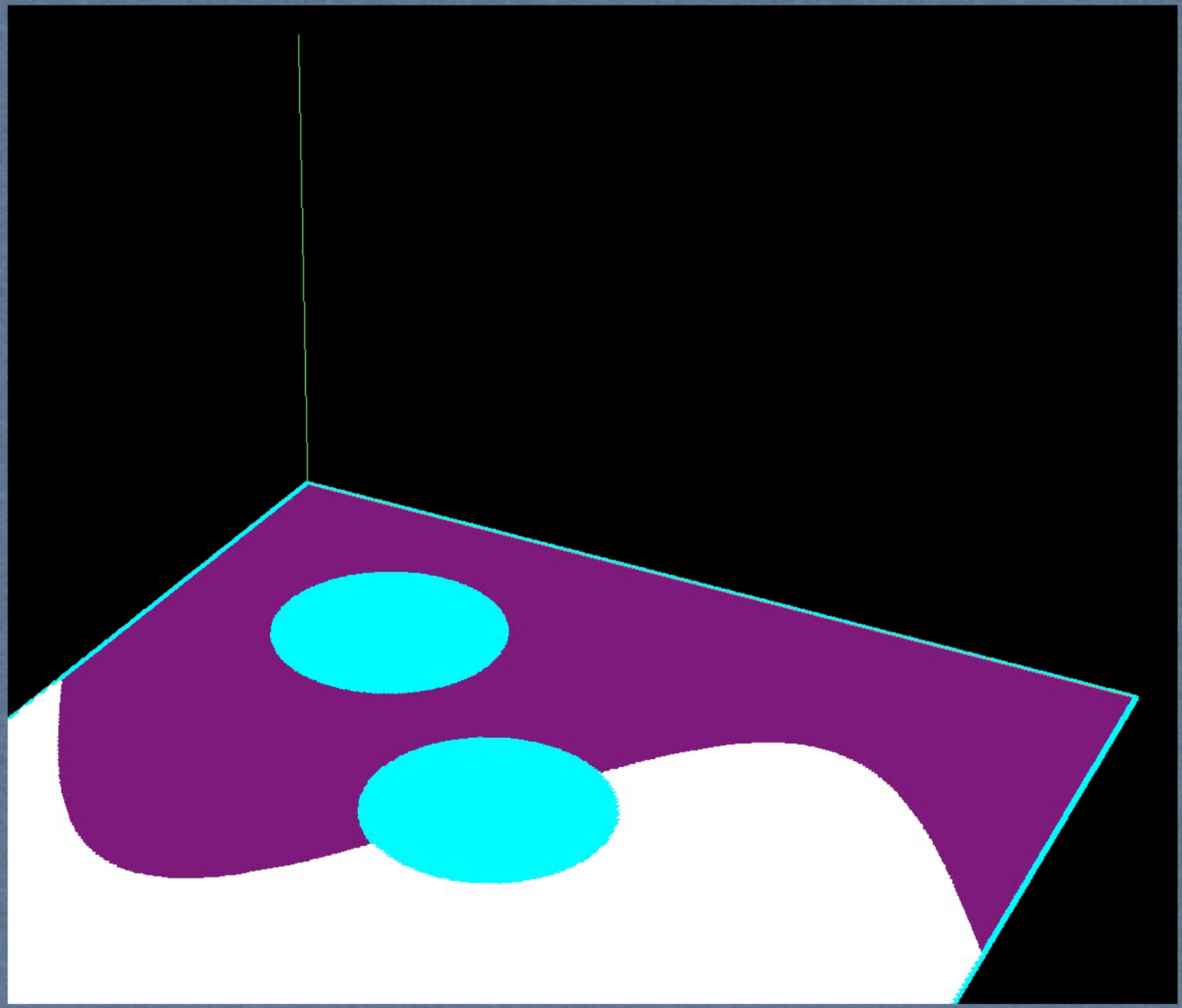
- Define your objectives
- Choose a parallel-friendly theoretical formulation
- Set performance expectations
- Choose a promising algorithm

Implement

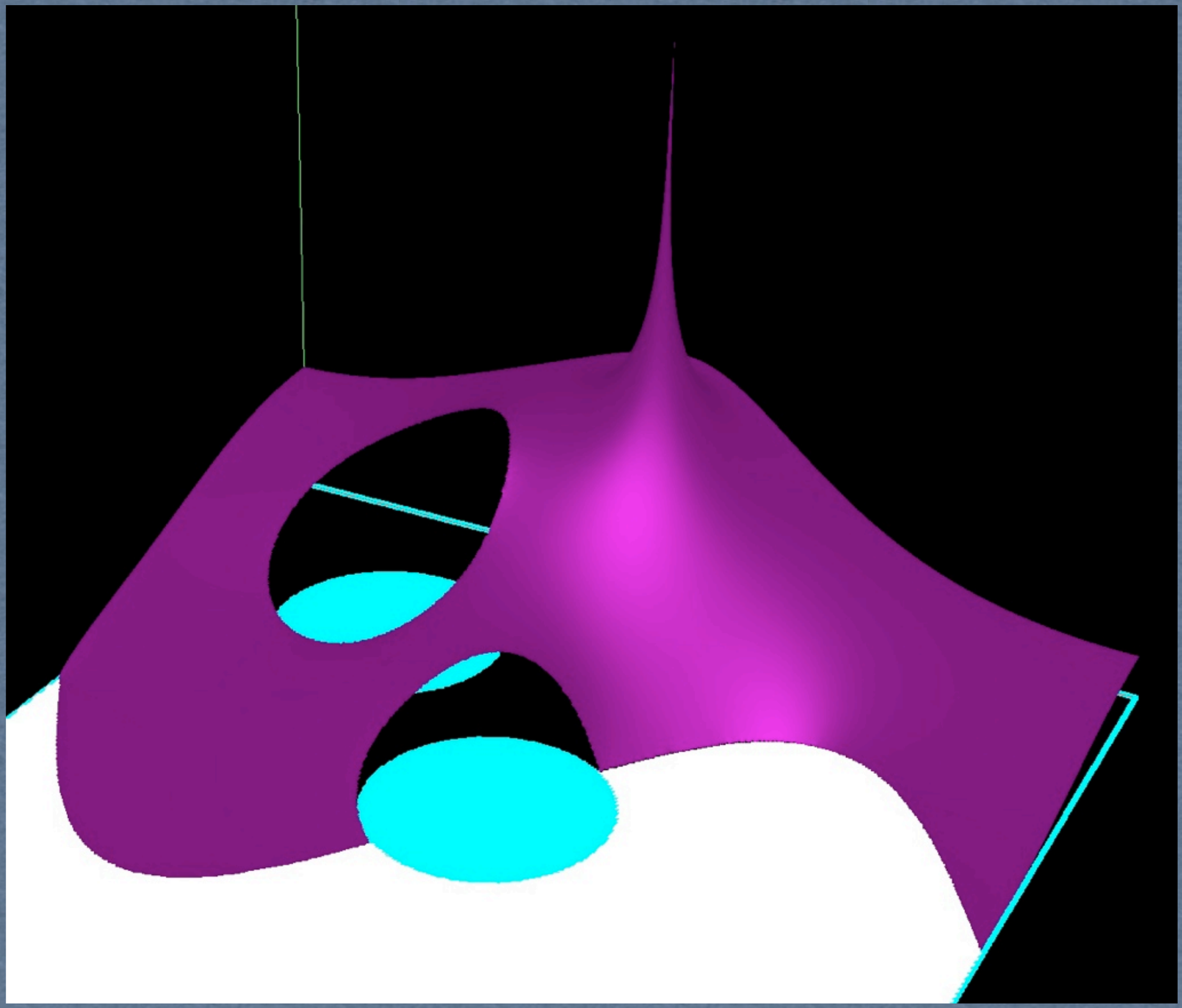
- Implement a prototype
- Organize code into reusable kernels

Accelerate

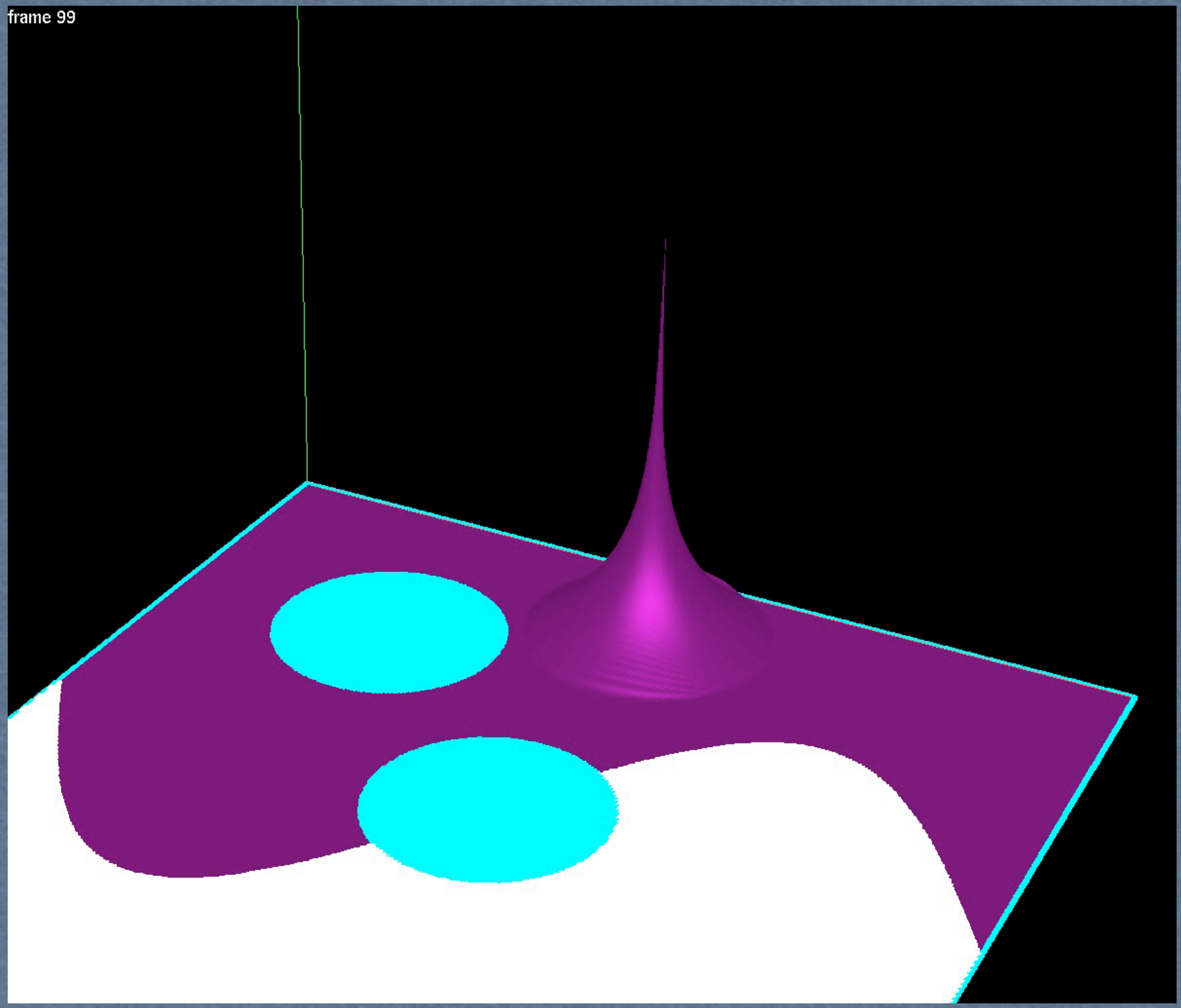
- Reorder/combine/pipeline operations
- Reduce resource utilization (try harder ...)
- Parallelize component kernels



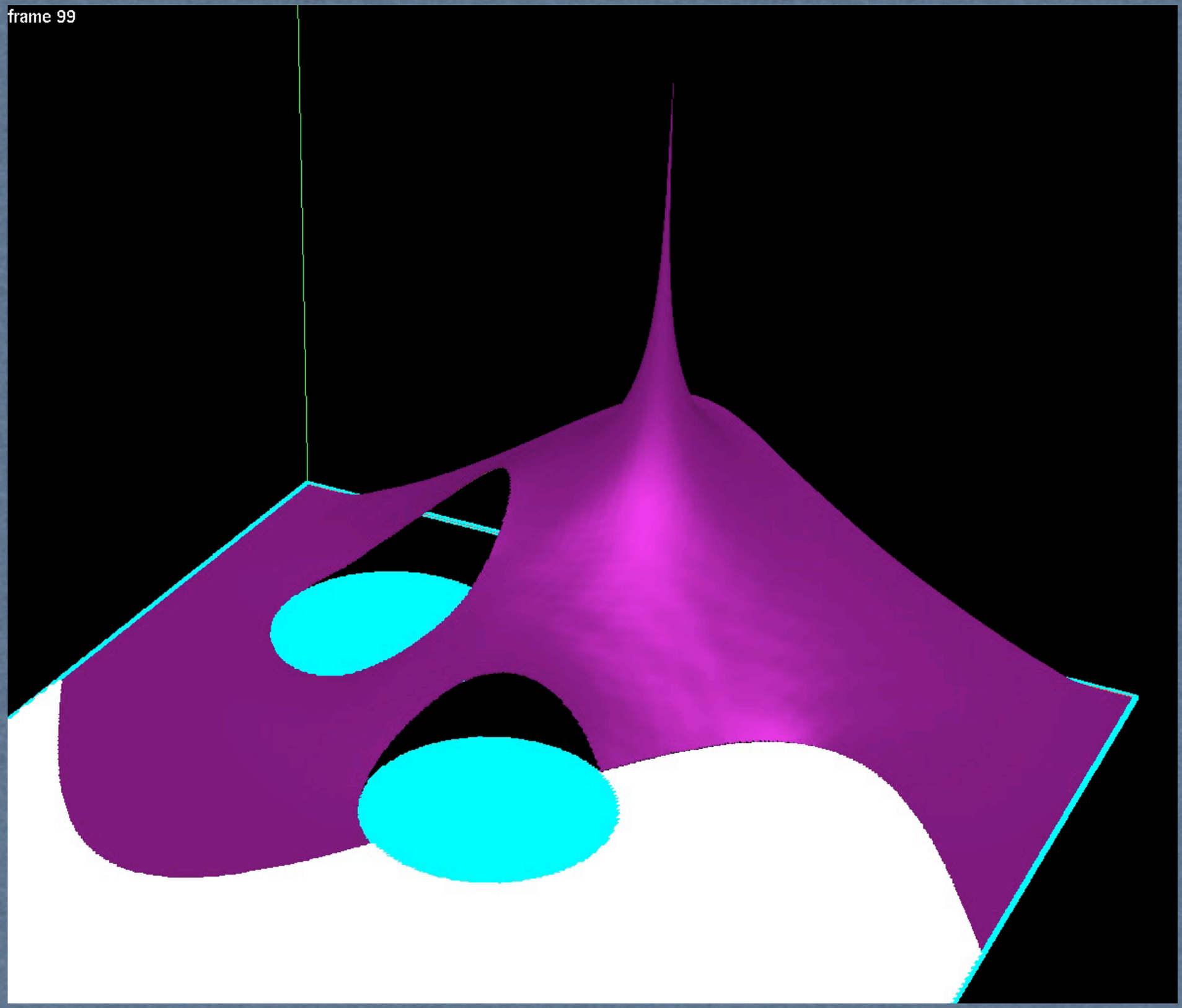
Sample 2D domain (512x512 resolution)



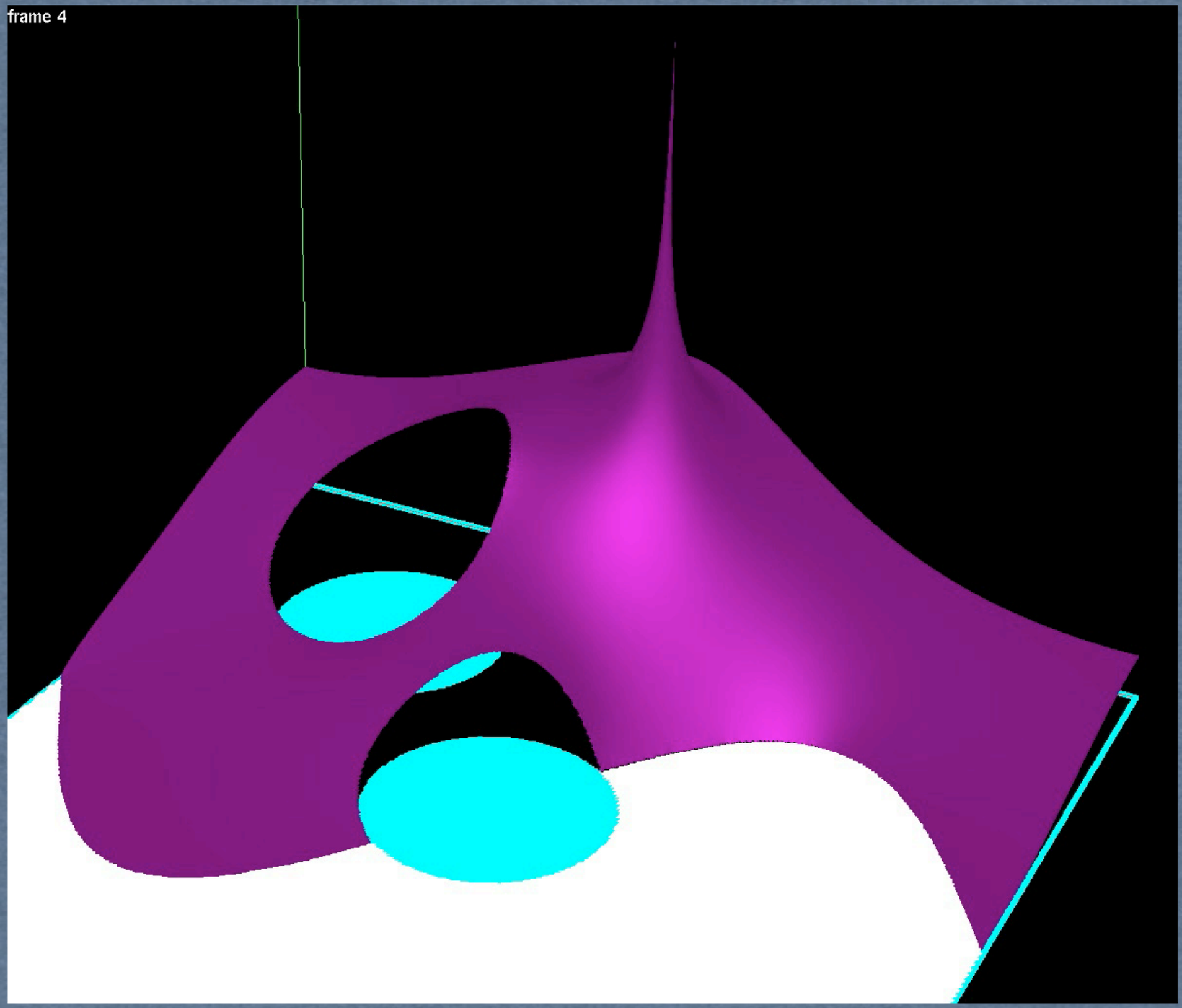
Exact solution



Conjugate Gradients (w/o preconditioning)



Conjugate Gradients (with a stock preconditioner)



Conjugate Gradients (with parallel multigrid preconditioner)

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Performance

- Converges in O(Nd) with stock preconditioners
- Converges in O(N) with multigrid preconditioners

Prerequisites

- Requires a symmetric system matrix
- Matrix needs to be positive definite
- (Other variants exist, too)

Benefits

- Low storage overhead
- Simple component kernels

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: **if** $(\nu < \nu_{\max})$ **then return**
4: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: **for** $k = 0$ **to** k_{max} **do**
6: $\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\alpha \leftarrow \rho/\sigma$
8: $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: **if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: **return**
12: **end if**
13: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\rho \leftarrow \rho^{\text{new}}$
16: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: **end for**
18: **end procedure**

Development Plan

Design

- Define your objectives
- Choose a parallel-friendly theoretical formulation
- Set performance expectations
- Choose a promising algorithm

Implement

- Implement a prototype
- Organize code into reusable kernels

Accelerate

- Reorder/combine/pipeline operations
- Reduce resource utilization (try harder ...)
- Parallelize component kernels

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Kernels

- Multiply()
- Saxpy()
- Subtract()
- Copy()
- Inner_Product()
- Norm()

procedure MGPCG($\mathbf{r}, \mathbf{x}$)
  $\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
  if $(\nu < \nu_{\max})$ **then return**
  $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
  for $k = 0$ **to** k_{max} **do**
    $\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
    $\alpha \leftarrow \rho/\sigma$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
    if $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
      $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
      return
    end if
    $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
    $\beta \leftarrow \rho^{\text{new}}/\rho$
    $\rho \leftarrow \rho^{\text{new}}$
    $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
  end for
end procedure

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Kernels

- Multiply()
- Saxpy()
- Subtract()
- Copy()
- Inner_Product()
- Norm()

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: **if** $(\nu < \nu_{\max})$ **then return**
4: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: **for** $k = 0$ **to** k_{max} **do**
6: $\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\alpha \leftarrow \rho/\sigma$
8: $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: **if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: **return**
12: **end if**
13: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\rho \leftarrow \rho^{\text{new}}$
16: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: **end for**
18: **end procedure**

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Kernels

- Multiply()
- Saxpy()
- Subtract()
- Copy()
- Inner_Product()
- Norm()

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: **if** $(\nu < \nu_{\max})$ **then return**
4: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: **for** $k = 0$ **to** k_{max} **do**
6: $\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\alpha \leftarrow \rho/\sigma$
8: $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: **if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: **return**
12: **end if**
13: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\rho \leftarrow \rho^{\text{new}}$
16: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: **end for**
18: **end procedure**

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Kernels

- Multiply()
- Saxpy()
- Subtract()
- Copy()
- Inner_Product()
- Norm()

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Kernels

- Multiply()
- Saxpy()
- Subtract()
- Copy()
- Inner_Product()
- Norm()

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

$$\mathcal{L}\mathbf{x} = \mathbf{f}$$

Kernels

- Multiply()
- Saxpy()
- Subtract()
- Copy()
- Inner_Product()
- Norm()

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: **if** $(\nu < \nu_{\max})$ **then return**
4: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: **for** $k = 0$ **to** k_{max} **do**
6: $\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\alpha \leftarrow \rho/\sigma$
8: $\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: **if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: **return**
12: **end if**
13: $\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\rho \leftarrow \rho^{\text{new}}$
16: $\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: **end for**
18: **end procedure**

Development Plan

Design

- Define your objectives
- Choose a parallel-friendly theoretical formulation
- Set performance expectations
- Choose a promising algorithm

Implement

- Implement a prototype
- Organize code into reusable kernels

Accelerate

- Reorder/combine/pipeline operations
- Reduce resource utilization (try harder ...)
- Parallelize component kernels

Preconditioned Conjugate Gradients

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x},\ \mu \leftarrow \bar{\mathbf{r}},\ \nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu,\ \mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)},\ \rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p},\ \sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z},\ \mu \leftarrow \bar{\mathbf{r}},\ \nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu,\ \mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)},\ \rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p},\ \mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Preconditioned Conjugate Gradients

1: **procedure** MGPCG($\mathbf{r}, \mathbf{x}$)
2: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mathcal{L}\mathbf{x}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
3: $\quad$**if** $(\nu < \nu_{\max})$ **then return**
4: $\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{p} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho \leftarrow \mathbf{p}^T\mathbf{r}$
5: $\quad$**for** $k = 0$ **to** k_{max} **do**
6: $\quad\quad\mathbf{z} \leftarrow \mathcal{L}\mathbf{p}$, $\sigma \leftarrow \mathbf{p}^T\mathbf{z}$
7: $\quad\quad\alpha \leftarrow \rho/\sigma$
8: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \alpha\mathbf{z}$, $\mu \leftarrow \bar{\mathbf{r}}$, $\nu \leftarrow \|\mathbf{r} - \mu\|_\infty$
9: $\quad\quad$**if** $(\nu < \nu_{\max}$ **or** $k = k_{\max})$ **then**
10: $\quad\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$
11: $\quad\quad\quad$**return**
12: $\quad\quad$**end if**
13: $\quad\quad\mathbf{r} \leftarrow \mathbf{r} - \mu$, $\mathbf{z} \leftarrow \mathcal{M}^{-1}\mathbf{r}^{(\dagger)}$, $\rho^{\text{new}} \leftarrow \mathbf{z}^T\mathbf{r}$
14: $\quad\quad\beta \leftarrow \rho^{\text{new}}/\rho$
15: $\quad\quad\rho \leftarrow \rho^{\text{new}}$
16: $\quad\quad\mathbf{x} \leftarrow \mathbf{x} + \alpha\mathbf{p}$, $\mathbf{p} \leftarrow \mathbf{z} + \beta\mathbf{p}$
17: $\quad$**end for**
18: **end procedure**

Development Plan

Design

- Define your objectives
- Choose a parallel-friendly theoretical formulation
- Set performance expectations
- Choose a promising algorithm

Implement

- Implement a prototype
- Organize code into reusable kernels

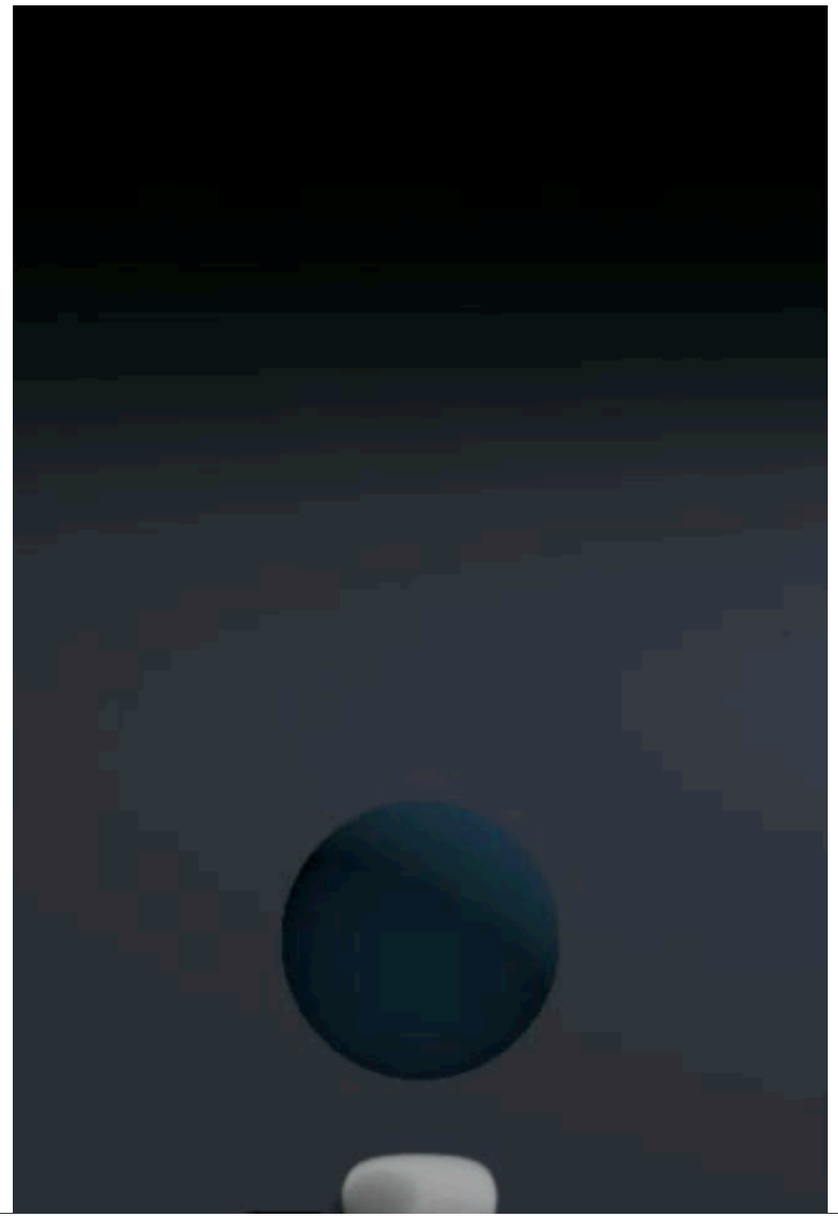
Accelerate

- Reorder/combine/pipeline operations
- Reduce resource utilization (try harder ...)
- Parallelize component kernels

Results and performance

Cost of 1 PCG Iteration By Simulation

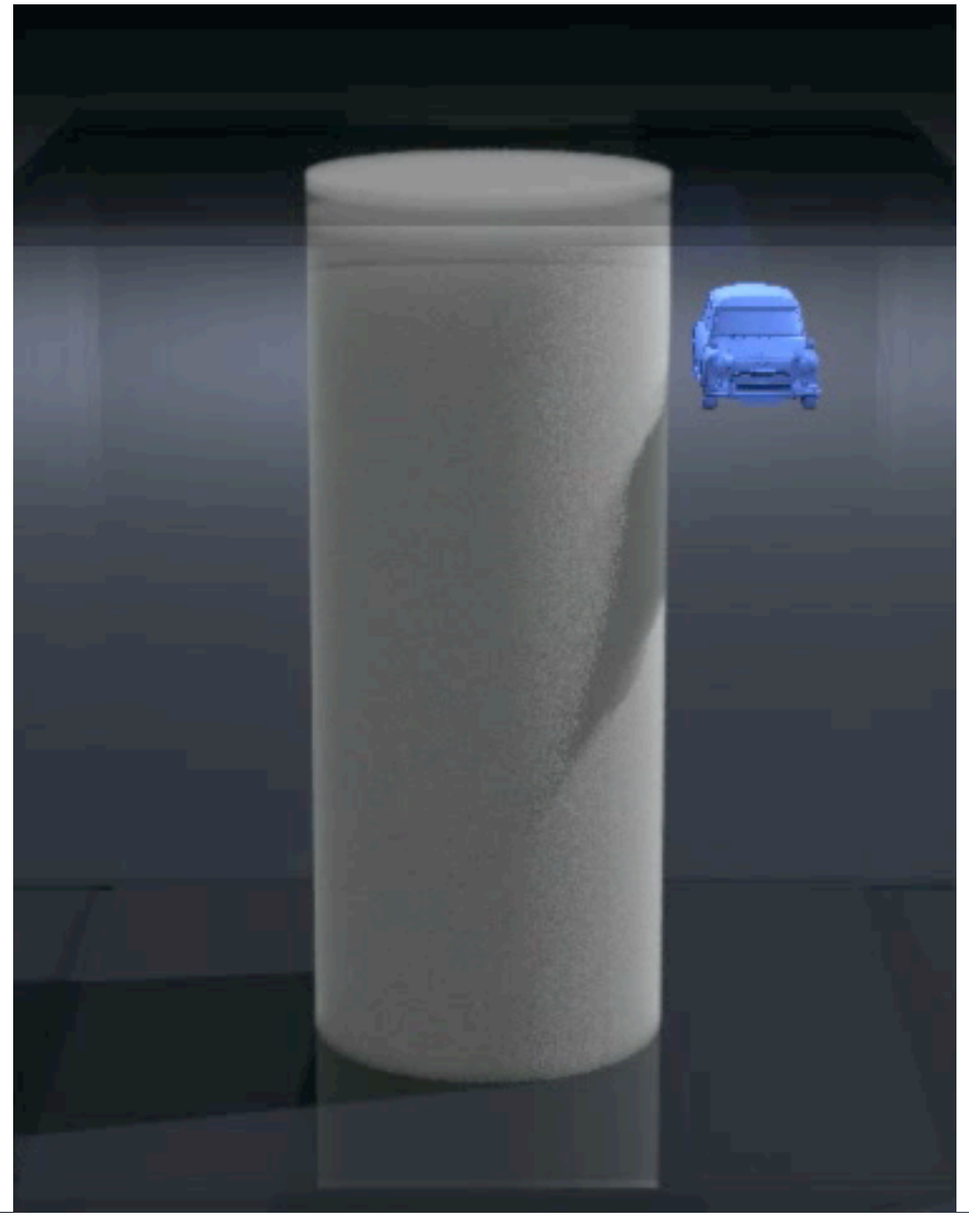
Simulation and Resolution	*1-core*	*16-core*	*Speedup*
Smoke flow past sphere			
64x64x64	39ms	23ms	1.7
96x96x96	127ms	47ms	2.7
128x128x128	299ms	67ms	4.5
192x192x192	983ms	167ms	5.9
256x256x256	2s 110ms	289ms	7.3
384x384x384	7s 380ms	875ms	8.4
512x512x512	15s 500ms	1s 930ms	8.0
768x768x768	51s 300ms	6s 90ms	8.4
768x768x1152	76s 800ms	9s 120ms	8.4
Smoke past car			
768x768x768	51s 200ms	6s 70ms	8.4
Free-surface water			
512x512x512	12s 900ms	1s 940ms	6.6



Results and performance

Smoke Past Sphere 768^3 PCG Iteration Breakdown

PCG Iteration Substep	*1-core*	*16-core*	*Speedup*
(Re-)Initialization*	13s 200ms	2s 370ms	5.6
V-Cycle (finest level breakdown)			
Int. Smoothing and Residuals	7s 990ms	1s 170ms	6.8
Bdry. Smoothing and Residuals	0s 983ms	0s 160ms	6.1
Restriction	3s 430ms	0s 287ms	12.0
Prolongation	2s 950ms	0s 398ms	7.4
Bdry. Smooth (upstroke)	0s 719ms	0s 103ms	7.0
Int. Smooth (upstroke)	11s 700ms	1s 150ms	10.2
V-Cycle total (1 iteration)	32s 200ms	3s 910ms	8.2
PCG, line 6	11s 600ms	0s 895ms	13.0
PCG, line 8	2s 270ms	0s 453ms	5.0
PCG, line 13 (inc. V-Cycle)	32s 200ms	3s 910ms	8.2
PCG, line 16	5s 160ms	828ms	6.2
PCG total (1 iteration)	51s 300ms	6s 90ms	8.4



Most kernels reach 60-70% of max theoretical efficiency (based on architectural limitations)

Preconditioning overhead : ~60% of iteration cost

Results and performance

"Black Box"(†) and Pipelined(*) CG (serial)

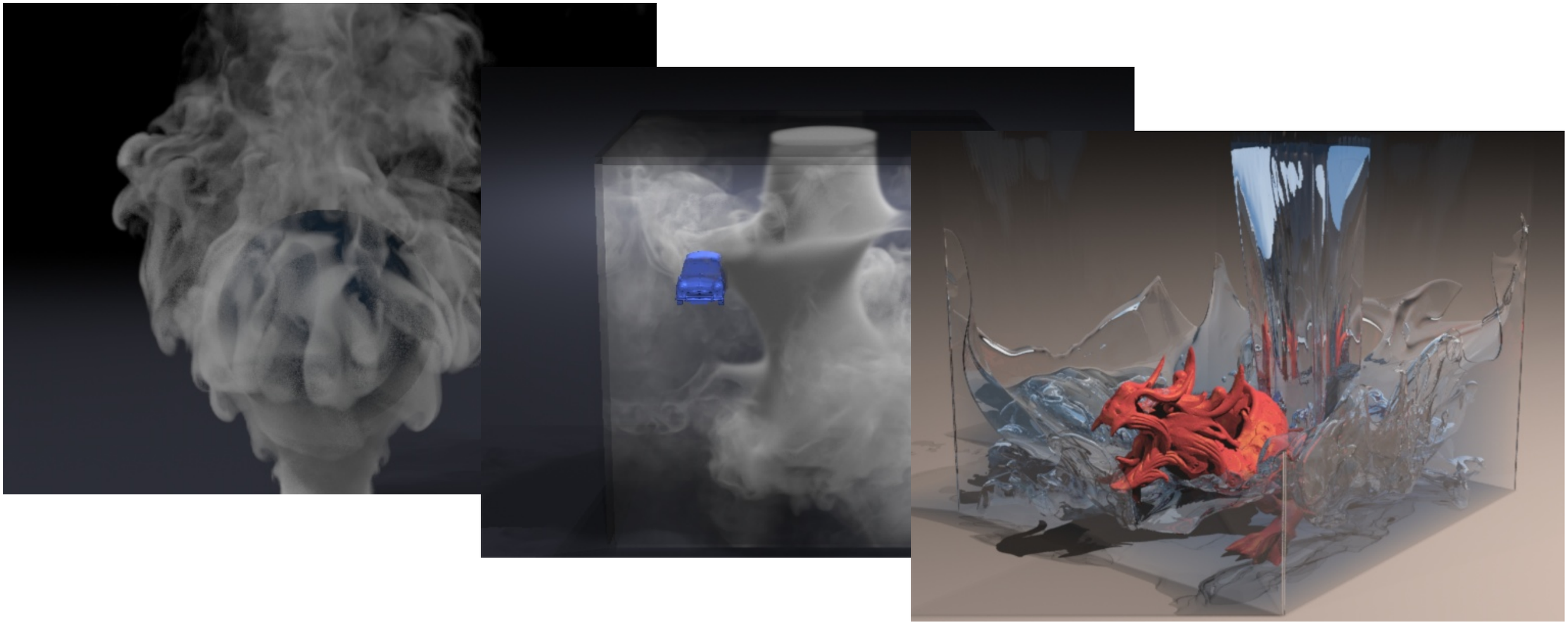
Resolution	64^3	96^3	128^3	192^3	256^3	384^3	512^3
Initialization Time (s)	0.0585	0.0617	0.1403	0.3931	1.0628	2.9837	6.9207
Iterations to r=1e-4	110	165	221	332	445	667	965
Iterations to r=1e-8	181	309	367	623	822	1261	1538
Time to r=1e-4 (s)†	2.32	8.73	42.45	149.94	503.97	2558.44	9010.28
Time to r=1e-4 (s)*	1.39	7.80	21.24	113.86	340.24	1999.10	5540.92
Time to r=1e-8 (s)†	3.81	16.35	70.49	281.37	930.92	4836.87	14360.43
Time to r=1e-8 (s)*	2.09	12.70	35.10	212.68	663.44	3524.00	8831.03

Incomplete Cholesky PCG (serial)

Resolution	64^3	96^3	128^3	192^3	256^3	384^3	512^3
Initialization Time (s)	0.20	0.55	1.24	4.01	9.61	32.23	76.47
Iterations to r=1e-4	36	52	72	107	138	213	278
Iterations to r=1e-8	57	78	104	149	194	295	395
Time to r=1e-4 (s)	0.94	5.04	17.00	88.22	283.10	1526.29	4679.32
Time to r=1e-8 (s)	1.50	7.56	24.55	122.85	397.98	2113.88	6648.68

Multigrid PCG (serial)

Resolution	64^3	96^3	128^3	192^3	256^3	384^3	512^3
Initialization Time (s)	0.08	0.10	0.24	0.63	1.63	5.01	12.21
Iterations to r=1e-4	9	9	11	10	12	12	13
Iterations to r=1e-8	15	16	17	18	19	20	21
Time to r=1e-4 (s)	0.57	2.82	3.70	10.74	25.92	89.71	211.88
Time to r=1e-8 (s)	0.88	4.80	5.57	18.50	39.54	144.47	332.71



Parallel programming practices for the solution of Sparse Linear Systems (motivated by computational physics and graphics)

Eftychios Sifakis

CS758 Guest Lecture - 19 Sept 2012