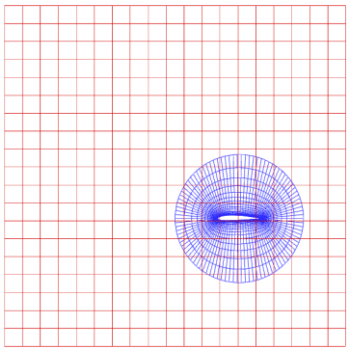
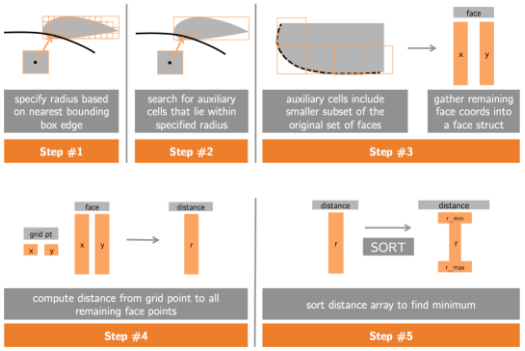
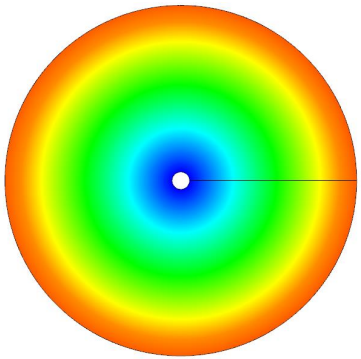
GPU acceleration of wall distance calculation for computational fluid dynamics codes







Nathan Wukie Vasanth Ganapathy Chris Park

Outline

- Background
- Brute-force algorithm
- Advancing boundary algorithm

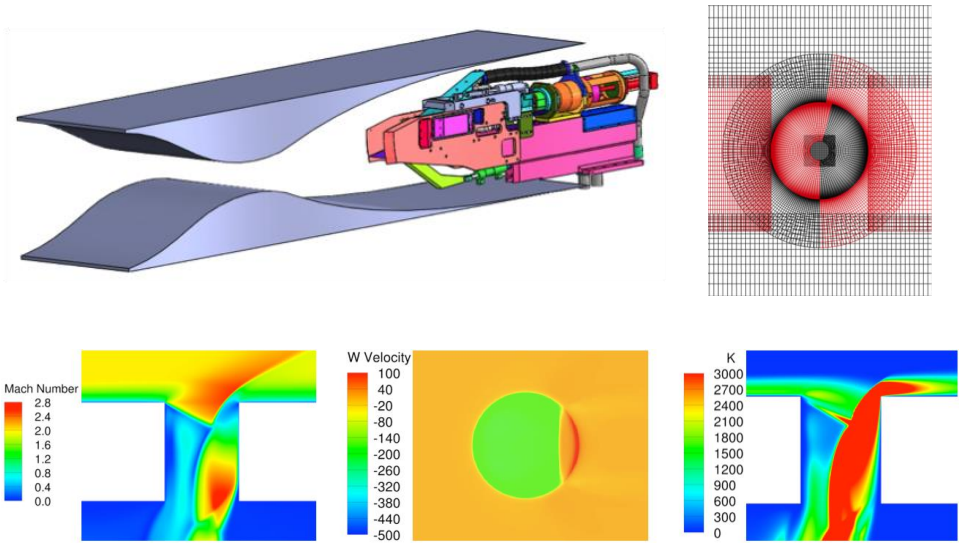
Background

- Reynolds-Averaged Navier-Stokes calculation

Conservation of mass

Conservation of momentum

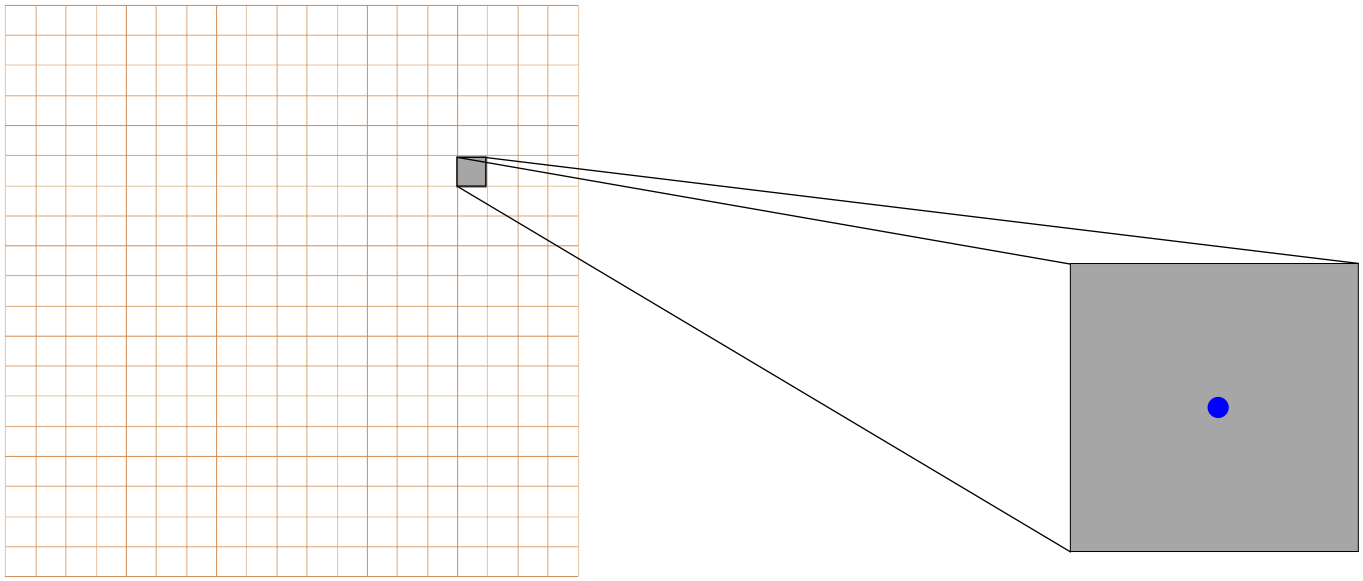
Conservation of energy



Wukie et al. 2012

Background

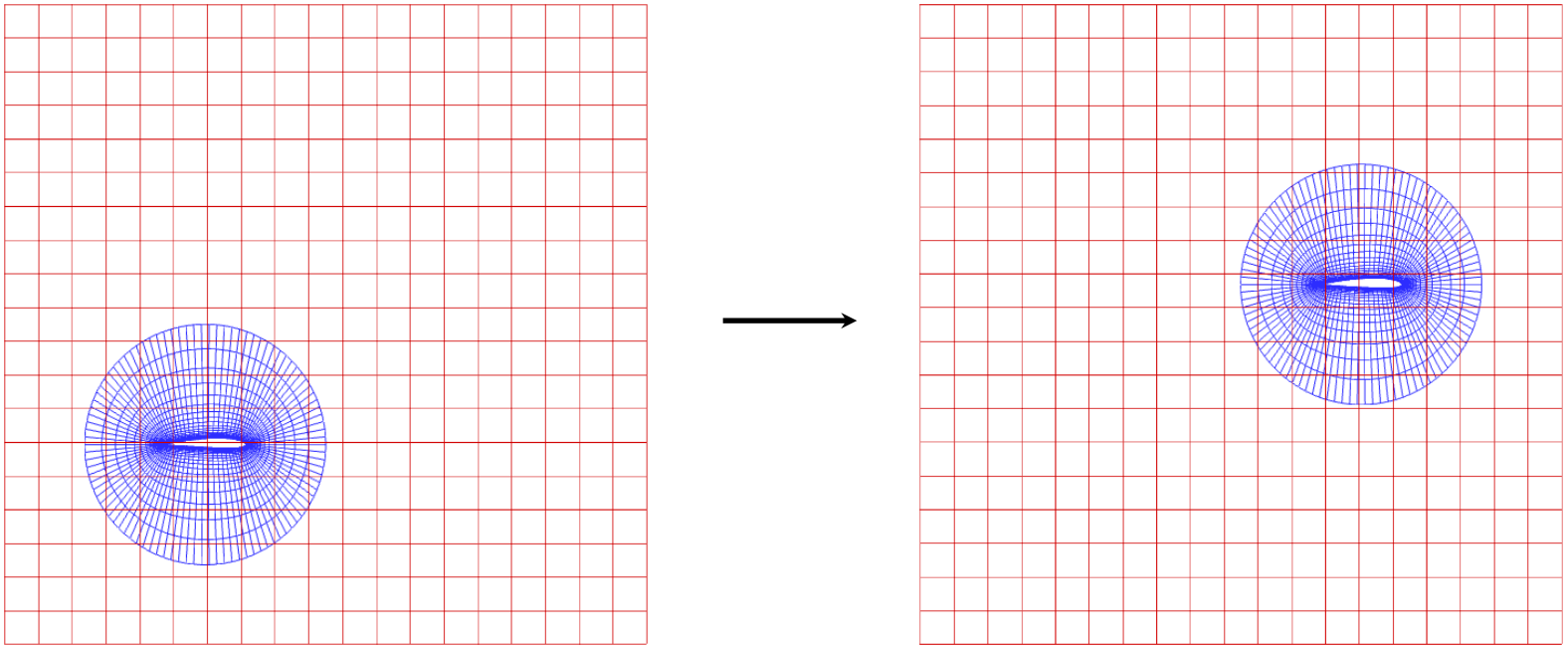
- Cell-centered, Finite Volume discretization



Solution variables stored at cell-center

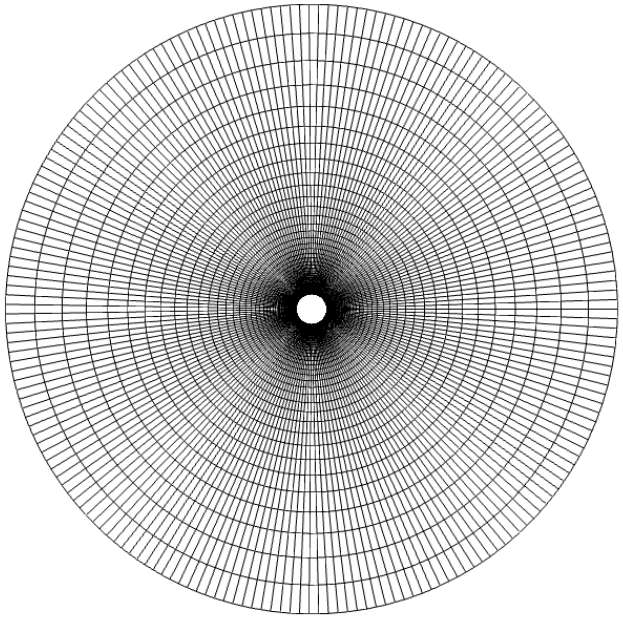
Background

- Moving mesh calculation

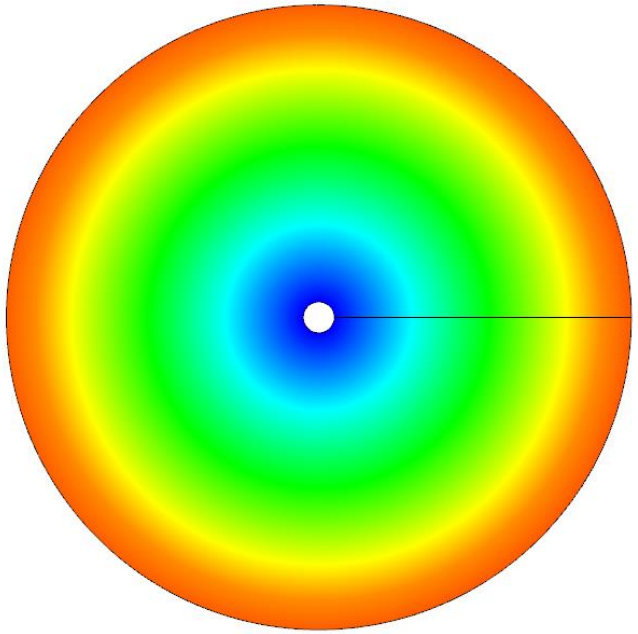


Background

- Test case: circle



Computational grid

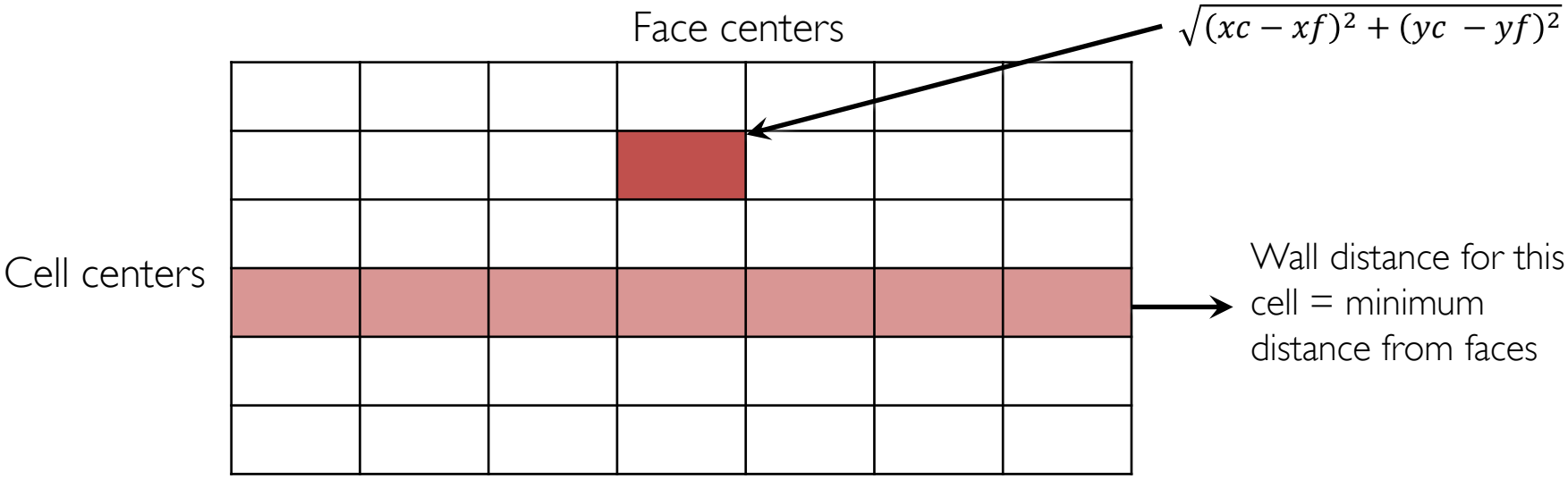


Wall distance field

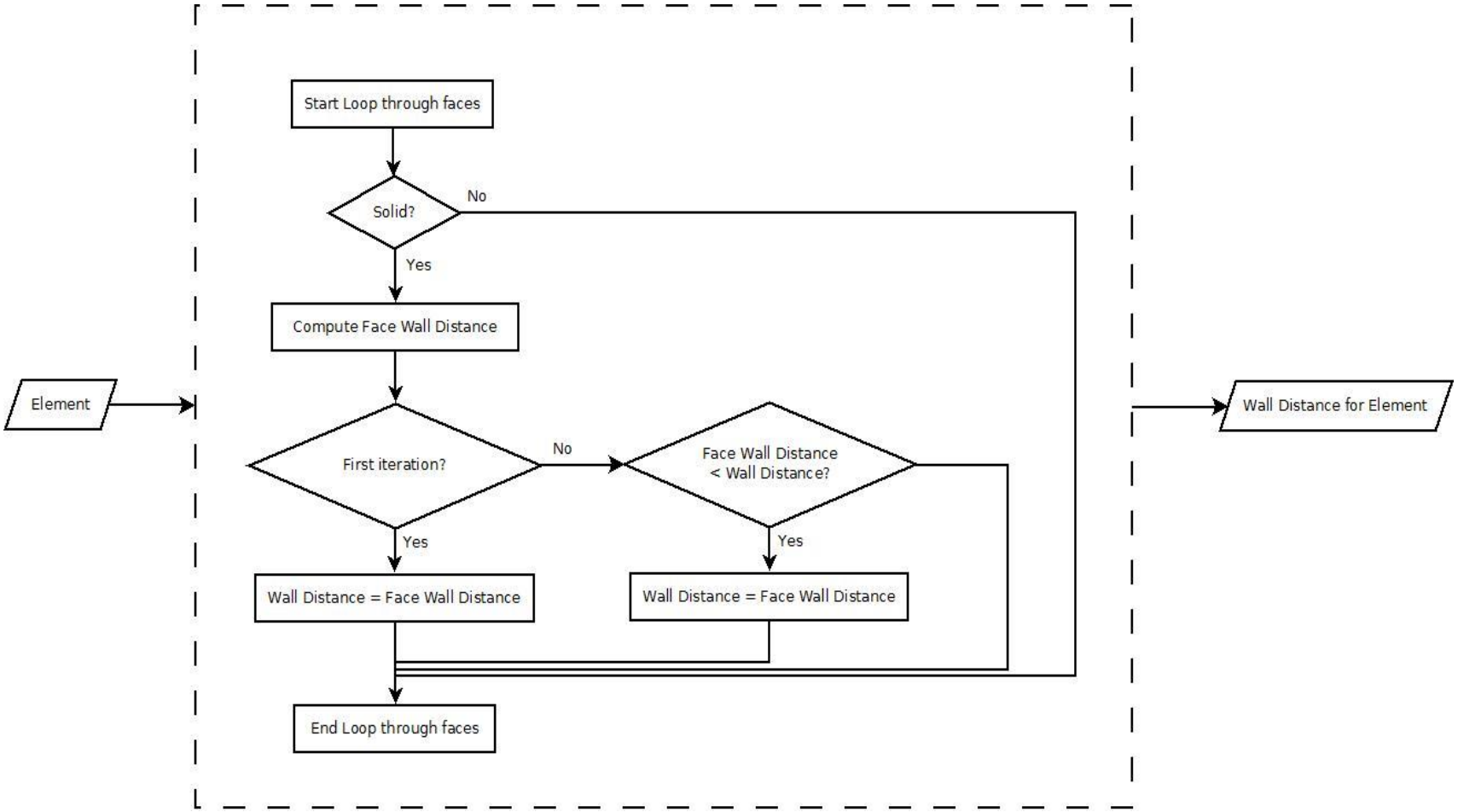
Brute-force algorithm

Brute-force Algorithm Outline

- For each cell element
  - Calculate its distance from each of the solid faces
  - Wall distance is the minimum of these distances

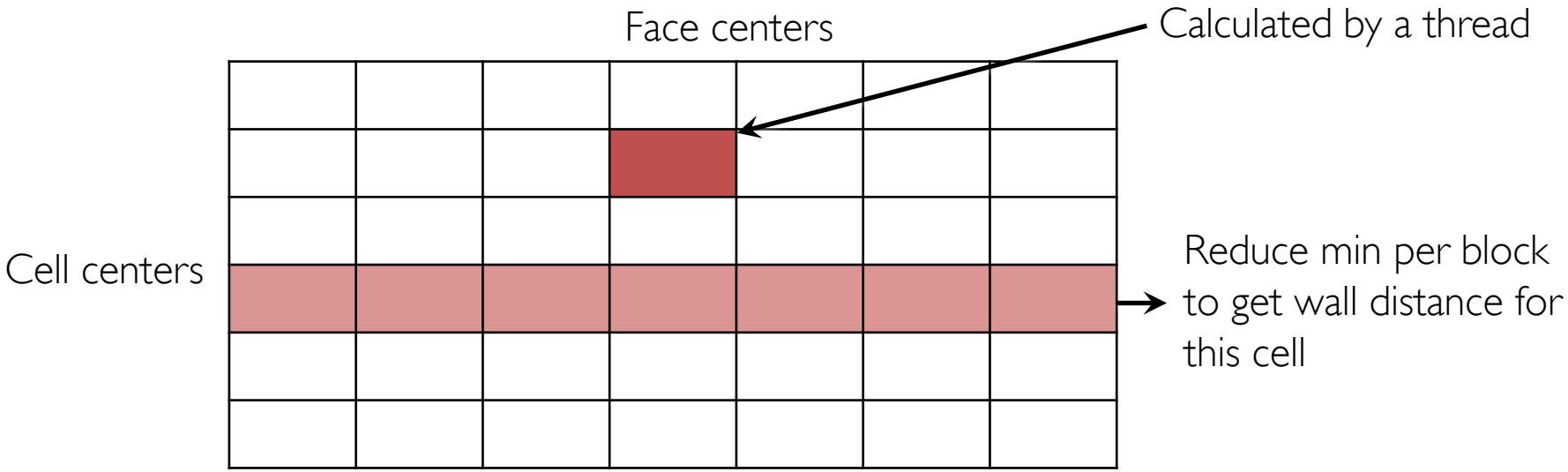


Brute-force – Serial



Brute-force – Parallel (Block Per Cell)

- Each block calculates wall distance for a cell
  - Each thread calculates distance from a face
  - Reduce minimum to get wall distance



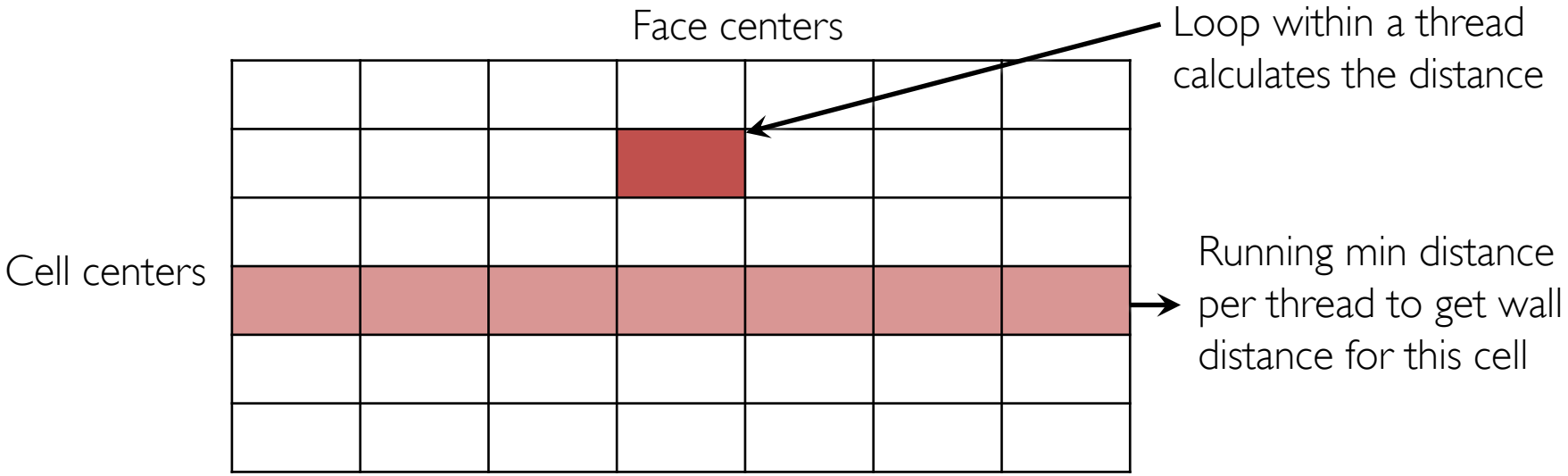
Brute-force – Parallel (Block Per Cell)

- Implemented 3 variations
  - (A): Shared memory writes for face distances
  - (B): (A) + Shared memory reads to face (x, y) arrays
  - (C): Same as (B) with (x, y) arrays converted to an interspersed array for coalesced shared mem reads

    i.e. x0, y0, x1, y1, ...

Brute-force – Parallel (Thread Per Cell)

- Each thread calculates wall distance for a cell
  - Calculates distance from the cell to each of the faces
  - Keeps "running" min distance value as each faces distance gets calculated



Brute-force – Parallel (Thread Per Cell)

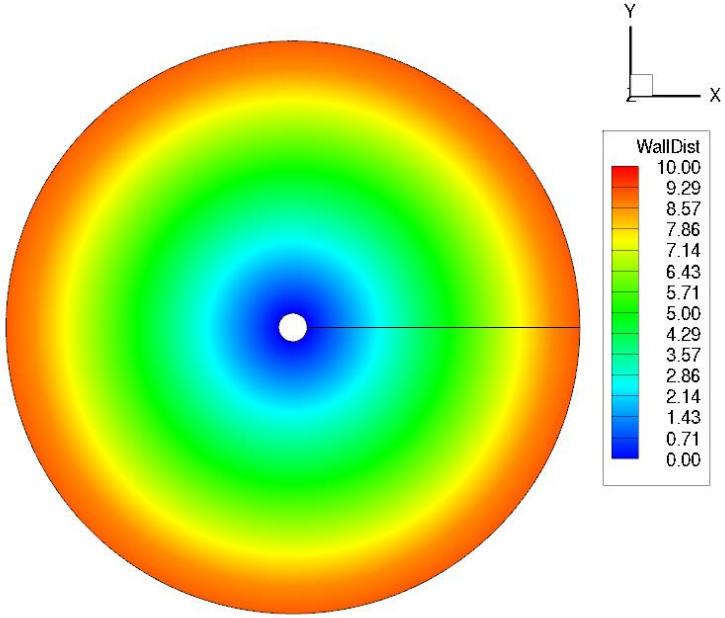
- Implemented 3 variations
  - (A): Each thread calculates wall distance for a cell
  - (B): Shared memory reads to face (x, y) arrays
  - (C): Same as (A) with (x, y) arrays converted to an interspersed array for coalesced global mem reads

    i.e. x0, y0, x1, y1, ...

Brute-force – Verification

- Output from serial algorithm visually inspected for correctness

- Outputs from parallel algorithms verified by comparing against serial algorithm output



Brute-force – Performance

Algorithm	Small Grid (msec)	Fine Grid (msec)	Extra Fine Grid (msec)
Serial	33	595	13636
Block Per Cell (1A) - Shared mem writes	1	28	-
Block Per Cell (1B) - Shared mem reads	1	30	-
Block Per Cell (1C) - Coalesced shared mem reads	1	31	-
Thread Per Cell (2A)	<1	7	173
Thread Per Cell (2B) - Shared mem reads	1	9	293
Thread Per Cell (2C) - Coalesced global mem reads	<1	7	166

82x speedup vs. serial

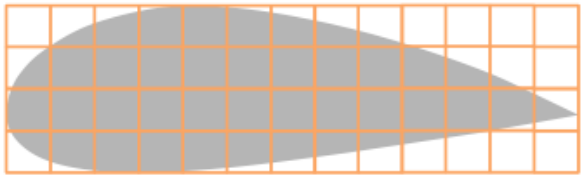
Advancing boundary algorithm

Advancing boundary algorithm

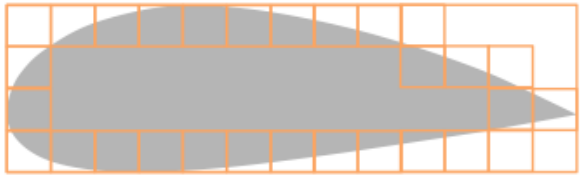
- Background: Pre-processor

Goal: create a smaller subset of faces that need searched

create bounding box



create auxiliary grid

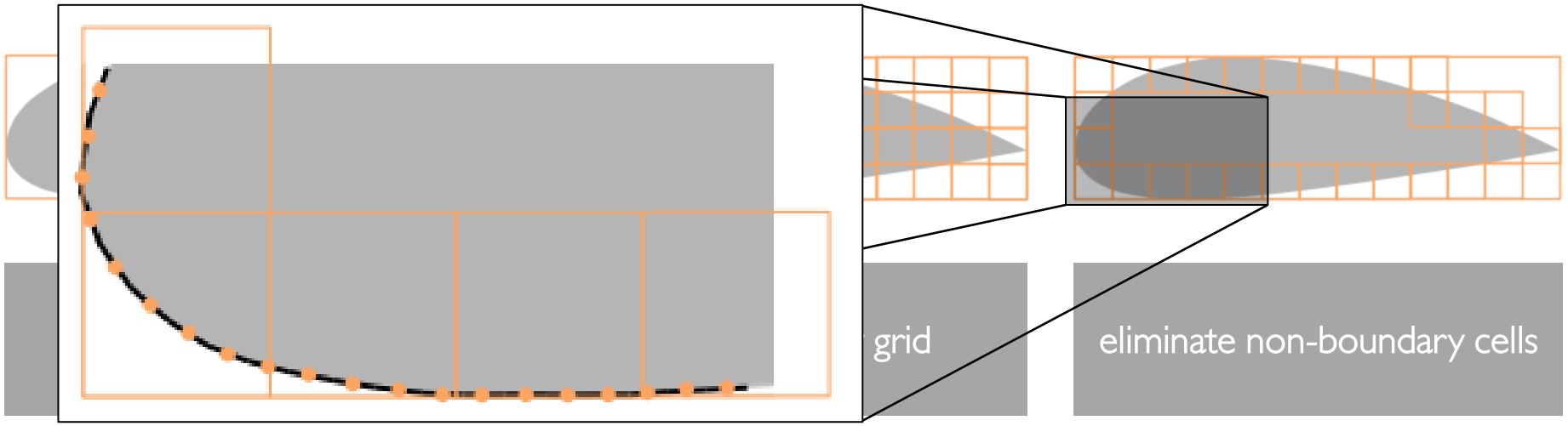


eliminate non-boundary cells

Advancing boundary algorithm

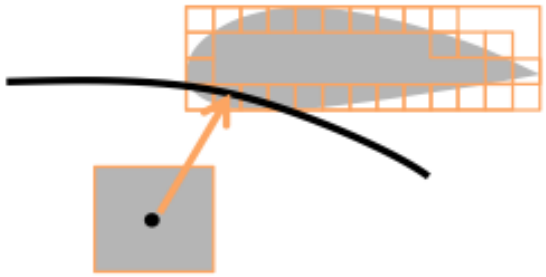
- Background: Pre-processor

Goal: create a smaller subset of faces that need searched



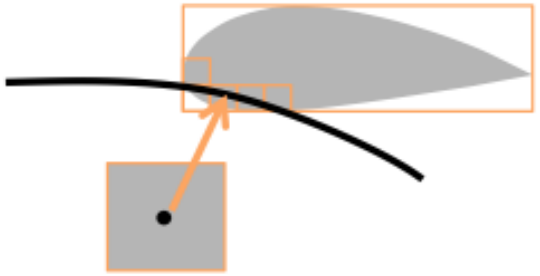
Advancing boundary algorithm

- Background: GPU-Kernel



specify radius based on nearest bounding box edge

Step #1

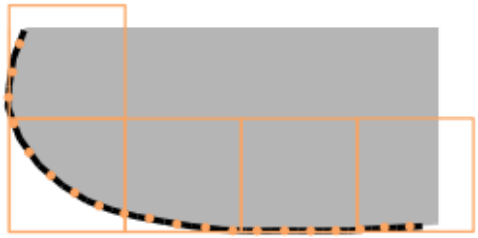


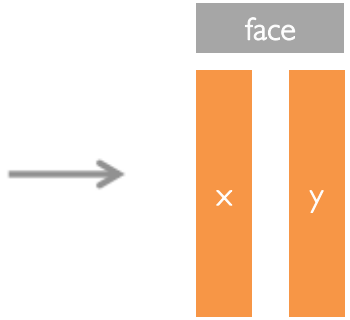
search for auxiliary cells that lie within specified radius

Step #2

Advancing boundary algorithm

- Background: GPU-Kernel

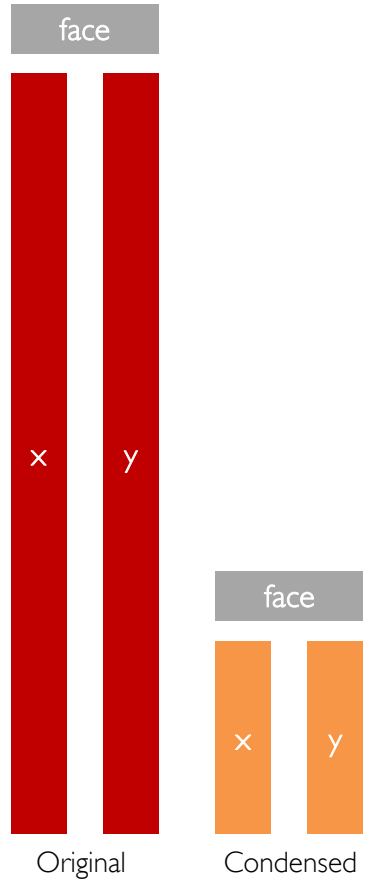




auxiliary cells include smaller subset of the original set of faces

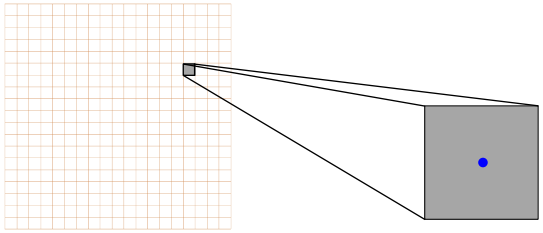
gather remaining face coords into a face struct

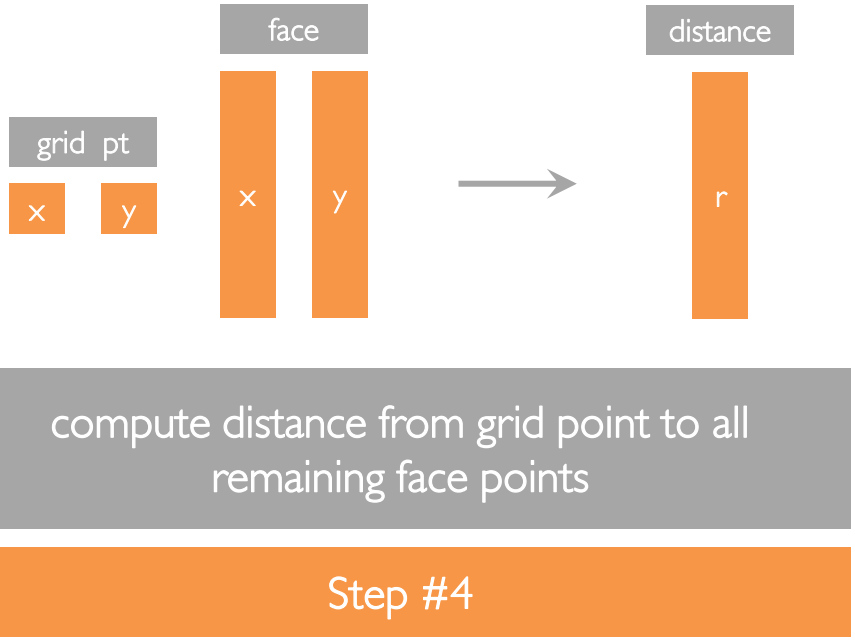
Step #3



Advancing boundary algorithm

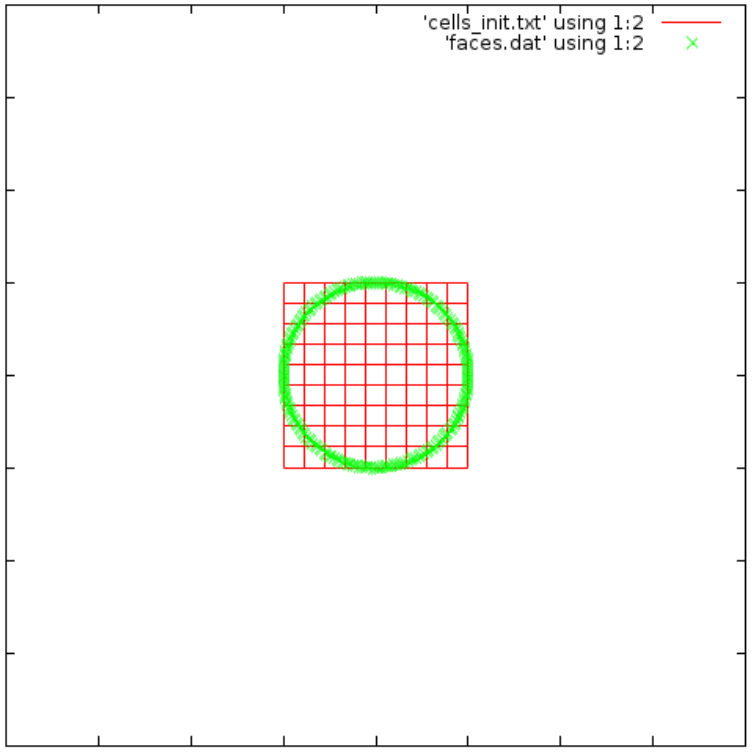
- Background: GPU-Kernel

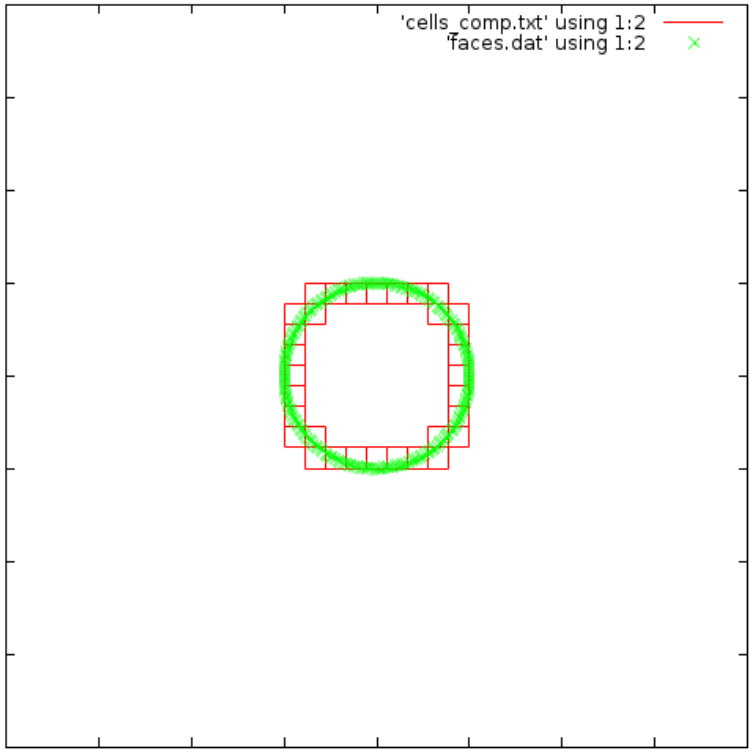




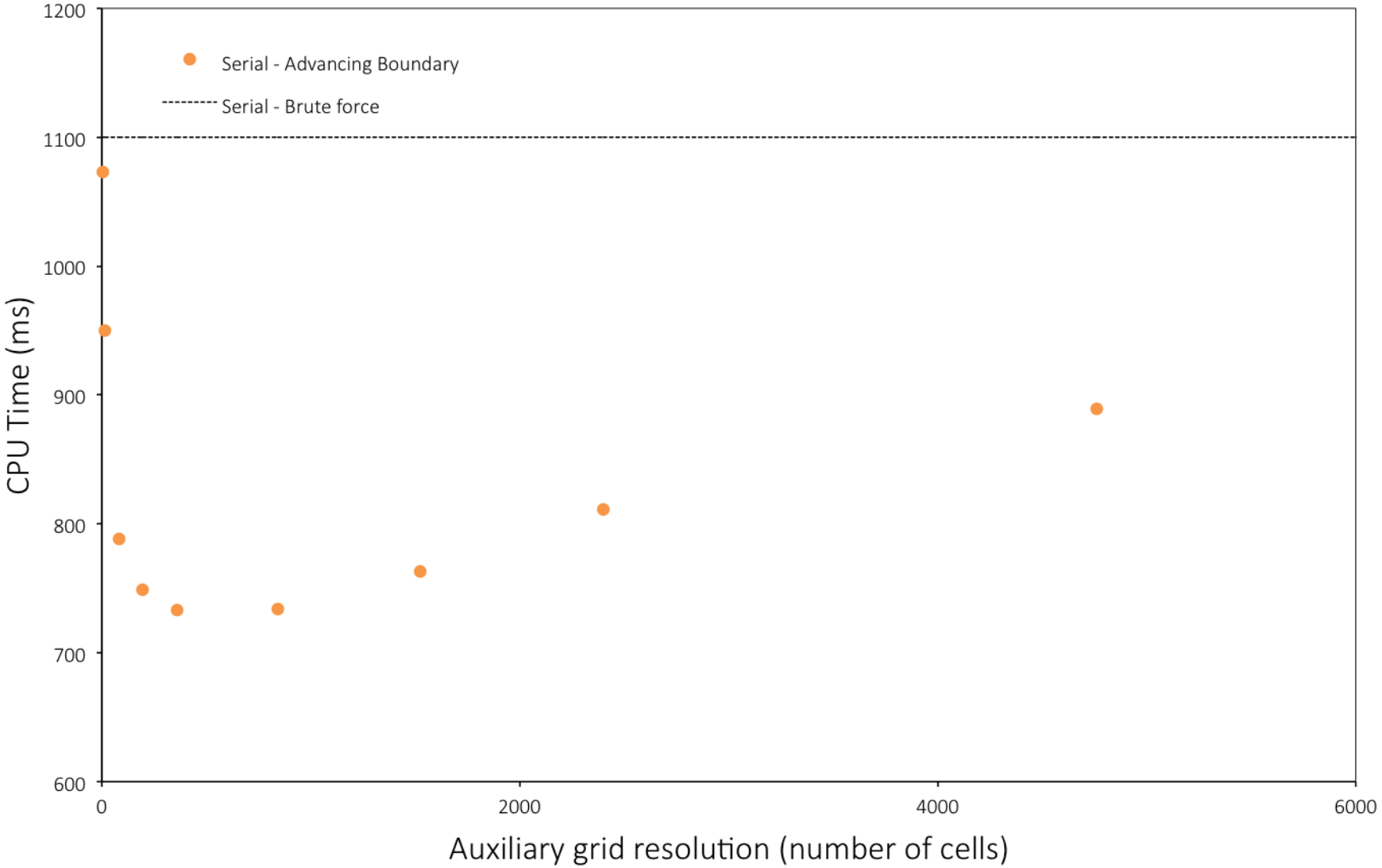
Advancing boundary algorithm

- Implementation: Pre-processing





Auxiliary grid resolution optimization



Advancing boundary algorithm

- Implementation: Aux Cell Structs

```
struct cell{

    // Cell Boundaries
    double xmin, xmax;
    double ymin, ymax;
    double xcenter, ycenter;

    // Included faces linked list
    struct face * root;

};
```

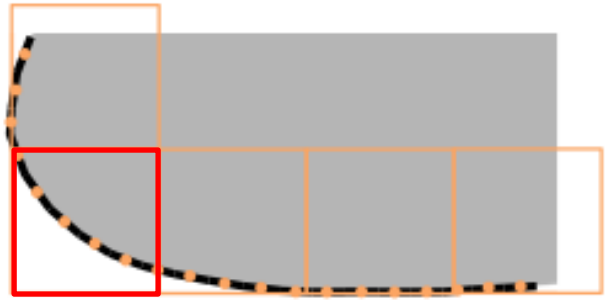
```
struct cell_t2{

    // Cell Boundaries
    double xmin, xmax;
    double ymin, ymax;
    double xcenter, ycenter;

    // Included faces linked list
    int faceNum;

    int storage;
    double * xface;
    double * yface;

};
```



```
struct cell_t3{

    // Cell Boundaries
    double xmin, xmax;
    double ymin, ymax;
    double xcenter, ycenter;

    // Included faces linked list
    int numFaces;

    double face_x[100];
    double face_y[100];

};
```

```
struct cell_pt3{

    // Cell Boundaries
    double xmin, xmax;
    double ymin, ymax;
    double xcenter, ycenter;

    // Included faces linked list
    int numFaces;

    double faces[200];

};
```

```
struct cell_pt1{

    // Cell Boundaries
    double xmin, xmax;
    double ymin, ymax;
    double xcenter, ycenter;

    // Included faces linked list
    int numFaces;

    int storage;
    int faceIndex[100];

};
```

Advancing boundary algorithm

Serial	Description	Average Time (ms)
T1	Baseline, linked-list of faces	8796
T2	New aux cell struct	11044
T3	New aux cell struct	10007

Parallel	Description	Average Time (ms)	Speedup	
			Serial CPU vs. GPU	Parallel CPU vs. GPU
T1	Baseline, no shared memory	134	65	5.5
T2	Memory management	114	77	6.4
T4		114	77	6.4
T5		112	78	6.5
T6		113	78	6.5
pt2_T5	New aux cell struct	106	83	6.9
pt3_T5	New aux cell struct	102	86	7.2

Conclusions

- Successful GPU implementation of two wall distance algorithms

- Order-of-magnitude type speedups realized

Thank you…