LA-UR-23-29604

Approved for public release; distribution is unlimited.

Title:	Data Remapping Between One-Dimensional Meshes
Author(s):	Ray, Navamita Hudgins, Jahi Michael
Intended for:	Slides for group presentation
Issued:	2023-08-21

Los Alamos NATIONAL LABORATORY

Data Remapping Between One-Dimensional Meshes

Jahi Hudgins

Florida A&M University

Mentors: Navamita Ray, Daniel “Danny” Shevitz

15th August, 2023

Acknowledgements

- Mentors for their help and guidance throughout the summer
- ASTERIX Consortium for the opportunity
  - Shonda Bernadin, Tommy Rockward
- Jacob Jones for constructive discussions
- Carter Mason for help with profiling studies

Outline

1 **Intro**

2 **Meshes and Fields**

3 **Point-wise Remap**

4 **Conservative Remap**

5 **Conclusion**

Introduction to Remap

- Data transfer between two meshes
- Uses range from image resizing to simulations of fluid mechanics
- Accuracy and performance are main issues
- Methods observed:
  - 1D Point-wise remap
  - 1D Conservative remap

Meshes and Fields

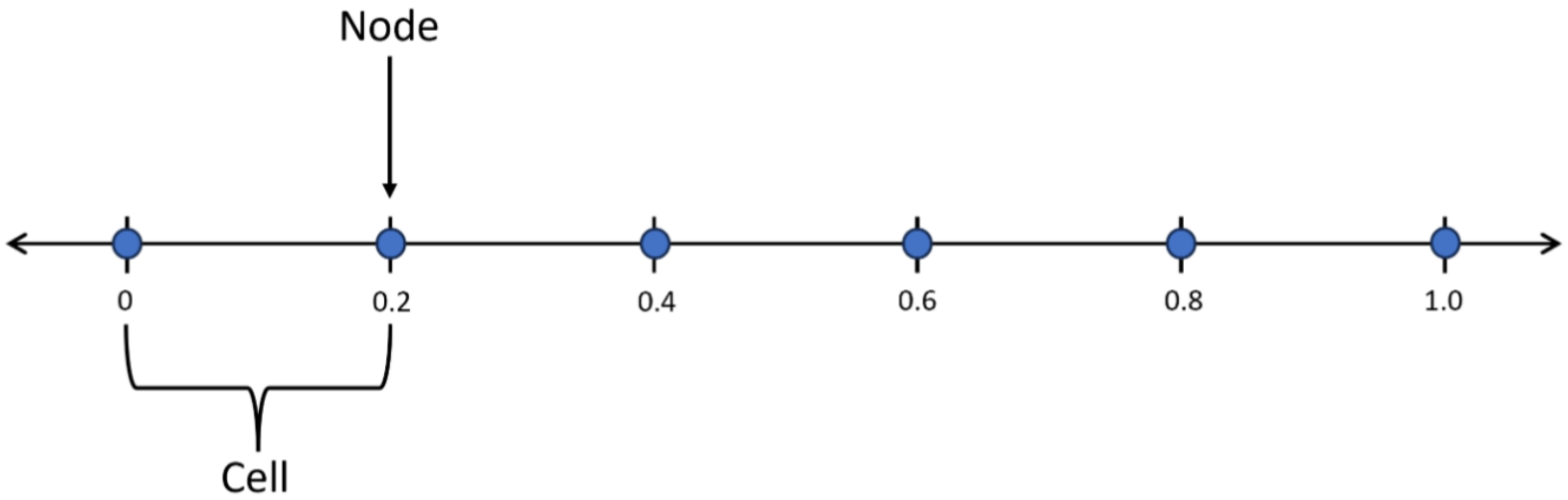
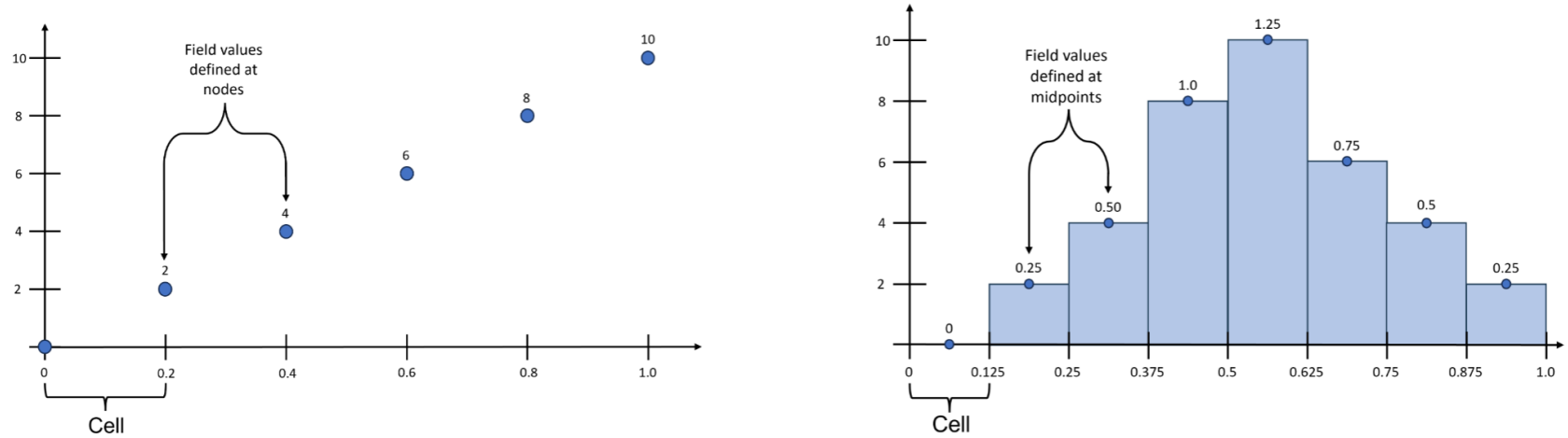


Figure 1: Example of a 1D mesh

Meshes and Fields

- Field



(a) 1D mesh with field values at nodes

(b) 1D mesh with field values at cell midpoints

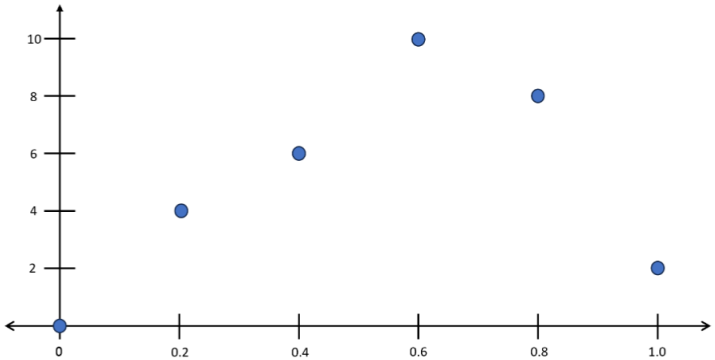
Figure 2: Field Definition Locations

Point-Wise Remap

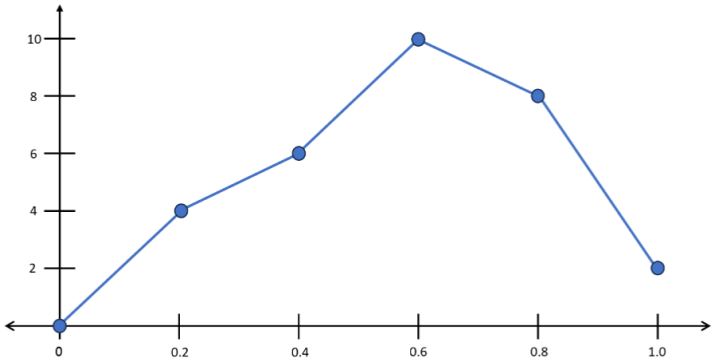
- Field values are defined at source nodes
  - Node-to-node transfer of data
- Piece-wise Linear Approximation
- Search
- Linear Interpolation

Piece-wise Linear Approximation

slope: $m = \frac{y_1 - y_0}{x_1 - x_0}$ *y-intercept:* $b = (-m * x_0) + y_0$



(a) Points before Linear Approximation



(b) Points after Linear Approximation

Figure 3: Piece-wise Linear Approximation

Linear Search

- Sequential algorithm
- Works on unsorted arrays
- Time complexity = O(N)

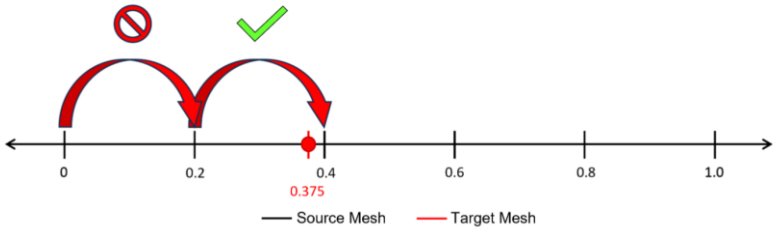


Figure 4: Point-wise Linear Search

Algorithm 1 Linear Search

Require: $nCell \geq 0$
  for $i \leftarrow 0$ **to** $nCell$ **do**
    if $target \geq source[i]$ and $target \leq source[i+1]$ **then**
      $return \leftarrow i$
    else
      $i \leftarrow i + 1$
    end if
  end for

Binary Search

- Repeatedly halves search interval
- Only works on sorted arrays
- Time complexity = O(log N)

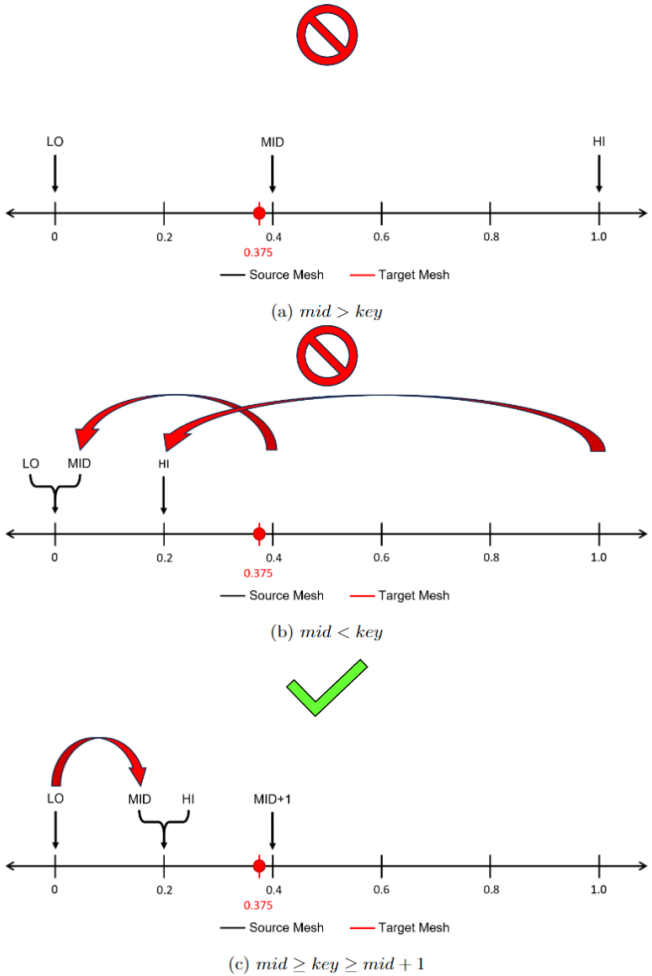
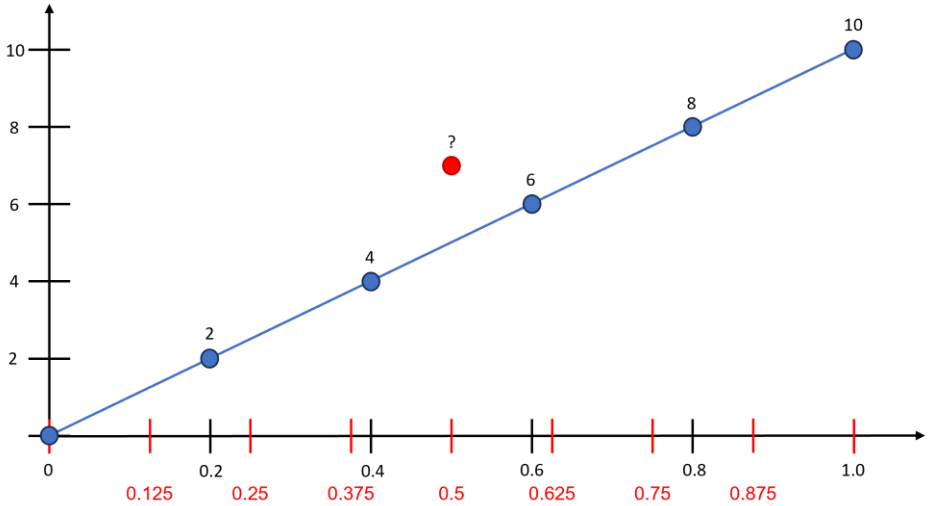


Figure 6: Point-wise Binary Search [$key = 0.375$]

Linear Interpolation

- Receives source location of target node from search
- Calculates target field value using source cell slope and y-intercept

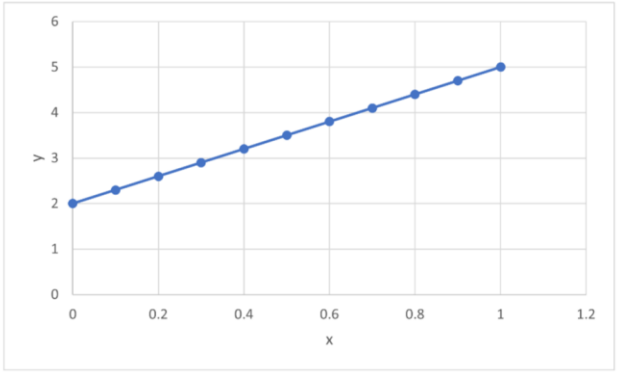
$$y_{trg} = (m_{src} * x_{trg}) + b_{src}$$



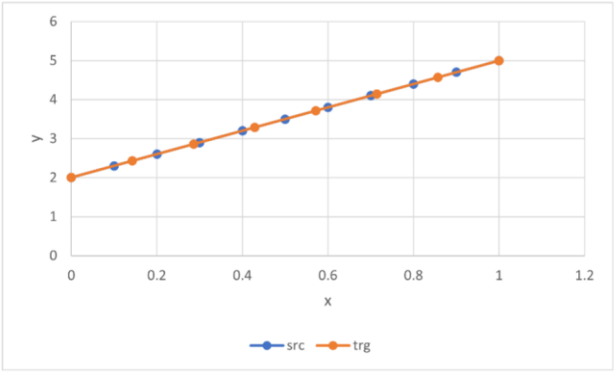
Verification Using a Linear Equation

$$y = 3x + 2$$

	Source Mesh	Target Mesh
# Cells	10	7
# Nodes	11	8



(a) Source Mesh



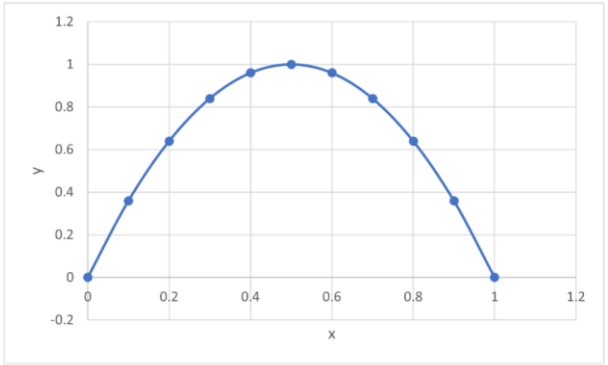
(b) Meshes Overlapped

Figure 5: Linear Function Verification

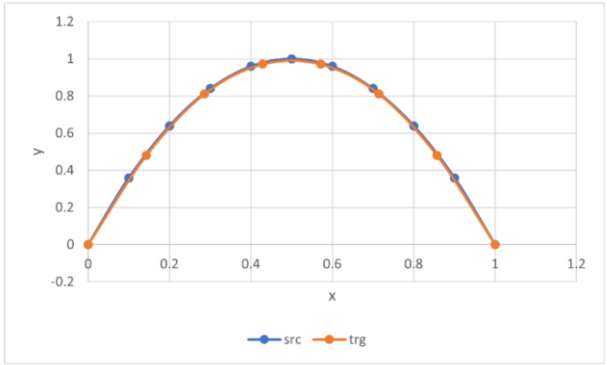
Verification Using a Quadratic Equation

$$y = -4(x - \frac{1}{2})^2 + 1$$

	Source Mesh	Target Mesh
# Cells	10	7
# Nodes	11	8



(a) Source Mesh



(b) Meshes Overlapped

Figure 6: Quadratic Function Verification

Profiling against 200 target cells

- Used <chronos> library
- Timing blocks around search function

$$t_{avg} = \frac{t_{end} - t_{start}}{\#cells}$$

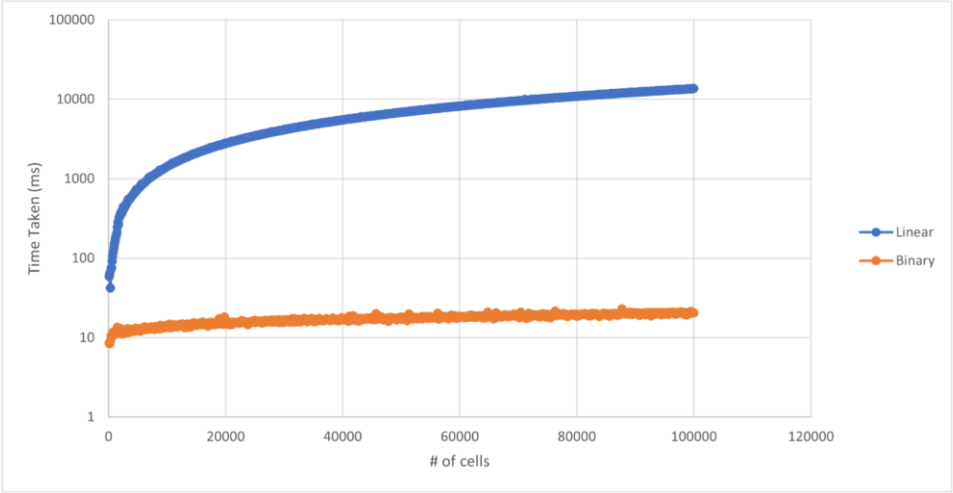
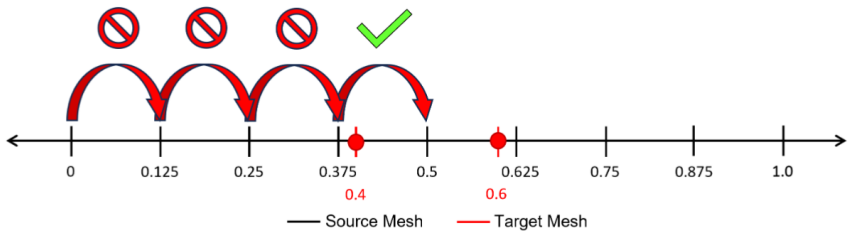


Figure 7: 10^2 to 10^5 Profiling

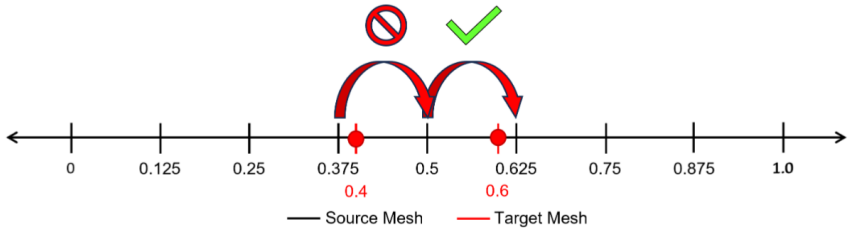
Conservative Remap

- Field values are defined at cell midpoints
- Field values are the area under the curve of the respective cell
  - Area is conserved between meshes
- Search
- Intersections
- Interpolation Using Area

Linear Search



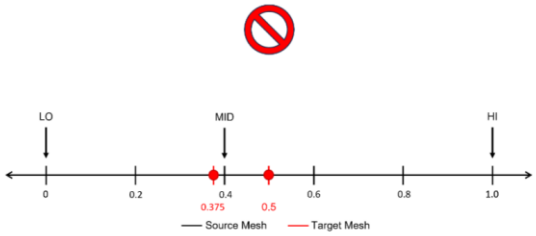
(a) Searching for head node



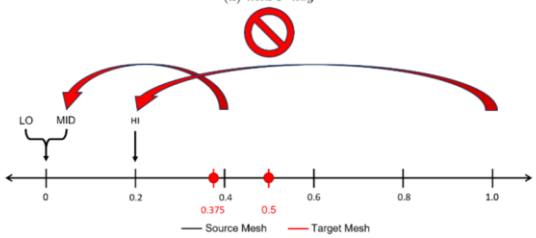
(b) Searching for tail node

Figure 9: Conservative Linear Search

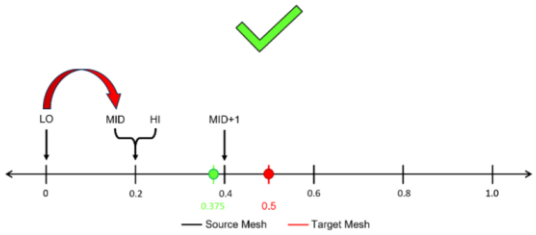
Binary Search



(a) $mid > key$

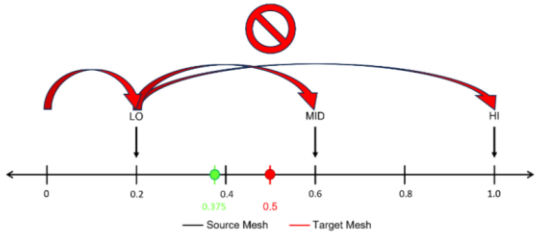


(b) $mid < key$

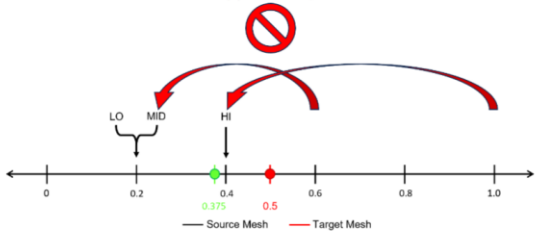


(c) $mid \geq key \geq mid + 1$

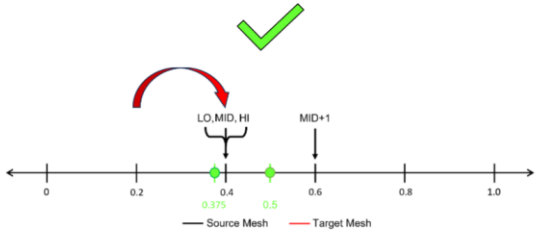
Figure 10: Conservative Binary Search Head [$key = 0.375$]



(a) $mid > key$



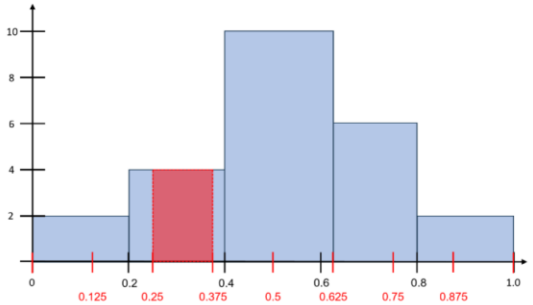
(b) $mid < key$



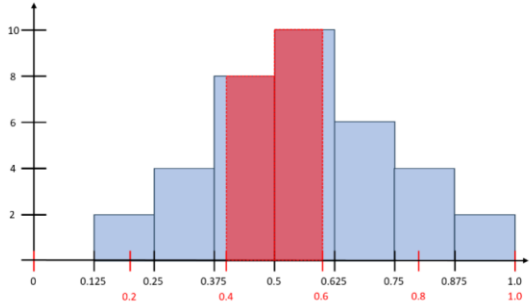
(c) $mid \geq key \geq mid + 1$

Figure 11: Conservative Binary Search Tail [$key = 0.5$]

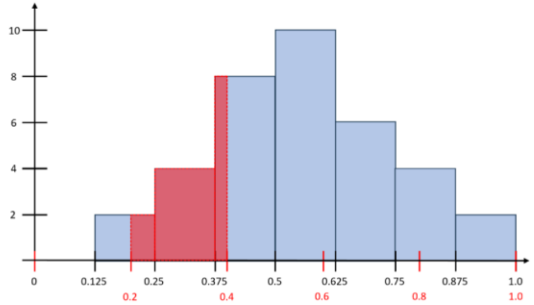
Intersections



(a) Case 1: Contained inside one source cell



(b) Case 2: Intersecting with 2 source cells



(c) Case 3: Intersecting with ≥ 3 source cells

Figure 8: Intersection Cases

Interpolation Using Area

- Target cell area is summation of intersected areas

$$A_{trg} = \sum_{n=1}^{N_c} I_{src} * y_{src}$$

- Target cell height is calculated using area and cell width

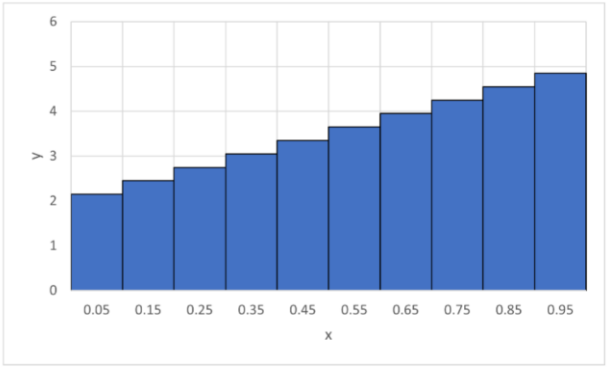
$$y_{trg} = A_{trg} / W_{trg}$$

Verification Using a Linear Equation

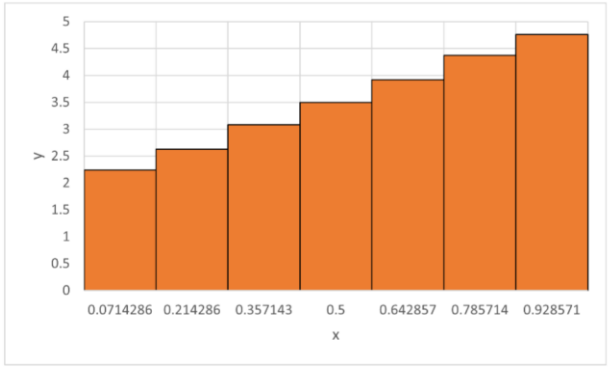
$$y = 3x + 2$$

	Source Mesh	Target Mesh
# Cells	10	7
# Nodes	11	8

	Area
Source Mesh	3.5000
Target Mesh	3.5000



(a) Source Mesh



(b) Meshes Overlapped

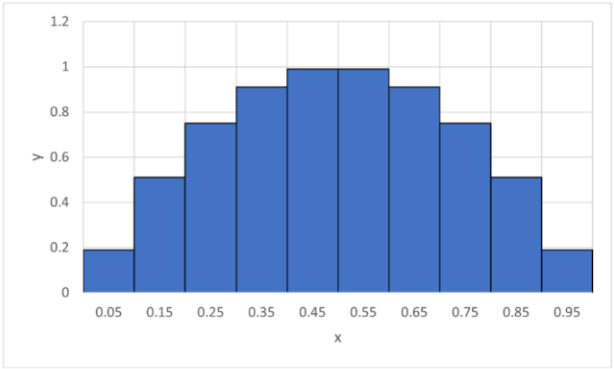
Figure 9: Linear Function Verification

Verification Using a Quadratic Equation

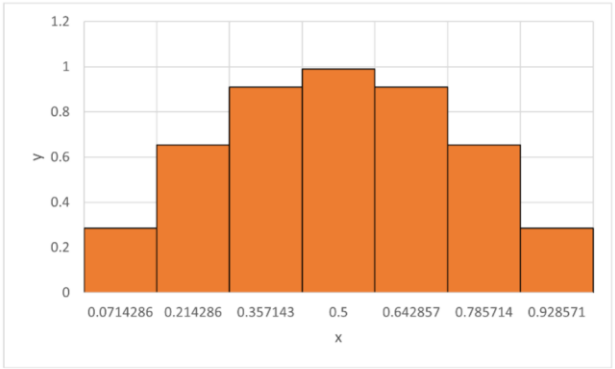
$$y = -4(x - \frac{1}{2})^2 + 1$$

	Source Mesh	Target Mesh
# Cells	10	7
# Nodes	11	8

	Area
Source Mesh	0.67000
Target Mesh	0.67000



(a) Source Mesh



(b) Meshes Overlapped

Figure 10: Quadratic Function Verification

Profiling Against 200 Target Cells

- Used <chronos> library
- Timing blocks around search function

$$t_{avg} = \frac{t_{end} - t_{start}}{\#cells}$$

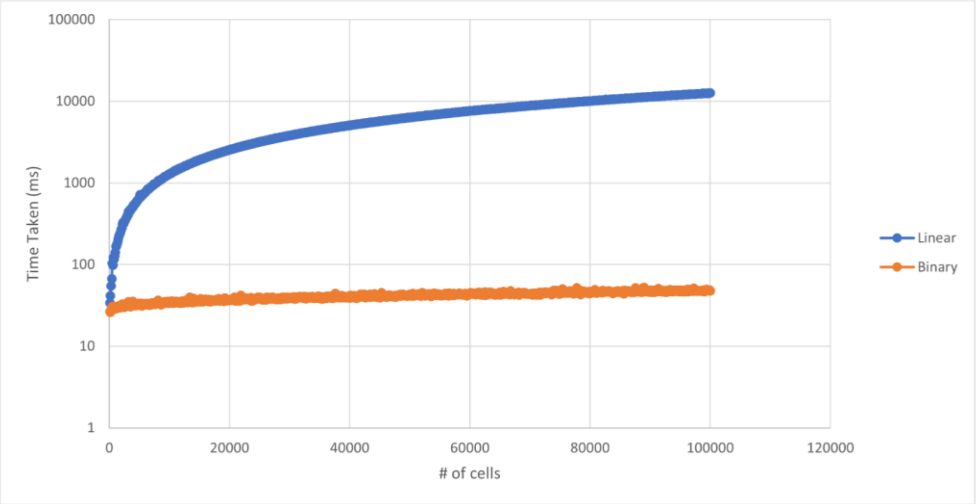


Figure 10: 10^2 to 10^5 Profiling

Conclusion

- Implemented two different remapping algorithms
  - Point-wise: node-to-node data remapping
  - Conservative: conserves the area under the curve

- Studied performance of both linear and binary searches
  - Found that the binary search was more efficient

Thanks! Any Questions?

Algorithm 2 Binary Search

Require: $nCell \geq 0$
  $lo \leftarrow 0$
  $hi \leftarrow nCell$
  while $lo \leq hi$ **do**
    $mid \leftarrow lo + ((hi - lo)/2)$
    if $target \geq source[mid]$ and $target \leq source[mid + 1]$ **then**
      $return \leftarrow mid$
    else if $source[mid] < target$ **then**
      $lo \leftarrow mid + 1$
    else if $source[mid] > target$ **then**
      $hi \leftarrow mid - 1$
    else
      $return \leftarrow -1$
    end if
  end while