LA-UR-23-29598

Approved for public release; distribution is unlimited.

Title:	Data Remapping Between One Dimensional Meshes
Author(s):	Ray, Navamita Mason, Carter Bayo
Intended for:	Slides for internship work
Issued:	2023-08-21

Los Alamos NATIONAL LABORATORY

Data Remapping Between One-Dimensional Meshes

Carter Mason

Mentors: Navamita Ray, Danny Shevitz

Florida State University

15 August 2023

Acknowledgements

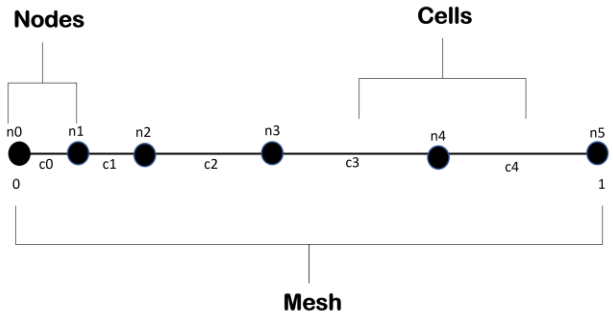
- Mentors for help and guidance throughout the summer.
- Asterix Program for funding and opportunity for LANL internship.
  - Shonda Bernadin, Tommy Rockwell
- Jacob Jones for advice and assistance throughout internship.

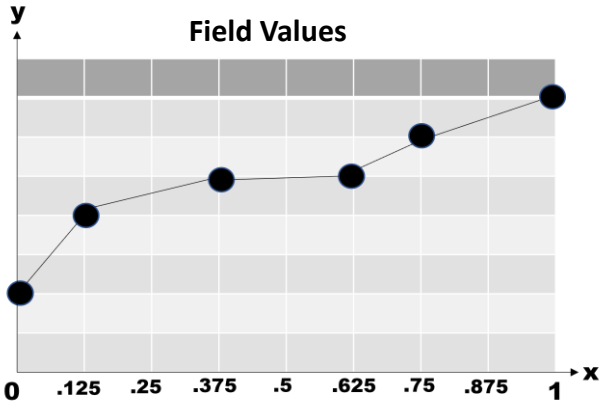
The Problem of Data Remap

- Remapping involves two meshes (source and target), where a discrete set of field values (data) is available on only one of the meshes (source mesh).
- Since the source and target mesh can be different, we need to find an approximation of the field on the target mesh using the data available on the source mesh.
- We will consider two algorithms for this problem:
  - Point-wise remap
  - Conservative remap.

Meshes and Fields

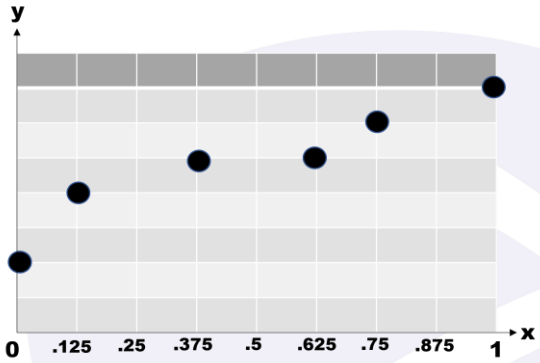
- Meshes
  - A mesh is a discretized approximation of a shape in any-dimension.
  - Contain nodes and cells.
  - 1-D for our purposes.
- Field
  - The corresponding y-values (data) at the given x-values on a mesh.



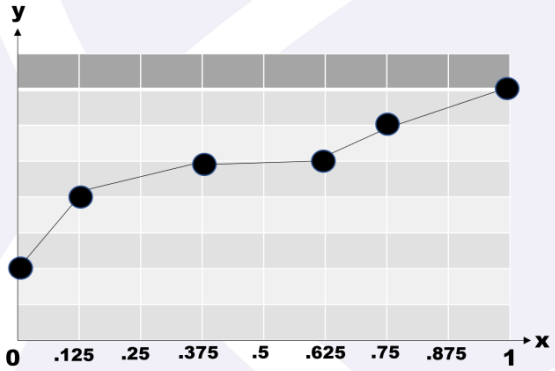


Point-wise Remap

- Remap of individual points of a target mesh.
- Field is defined at the nodes of the source mesh.
- We obtain an approximation over the source mesh by using piece-wise linear interpolation.
- We will use the linear approximation to interpolate a field of a target mesh, using our source field.

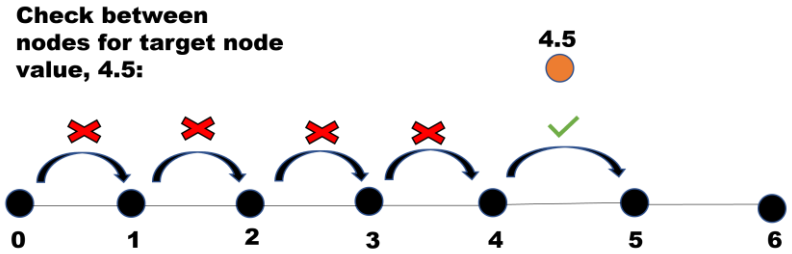


Before Approximation



After Approximation

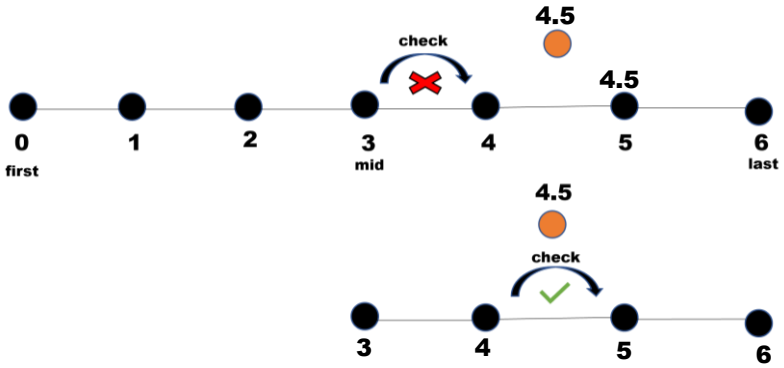
Linear Search for Point-wise Remap



Algorithm 1 Linear Search Algorithm

```
Xt = x-target value
Xs = x-source value (held in an array)
n = number of source cells
initialize i to 0
for i to n do
    if (Xt ≥ Xs[i]) and (Xt ≤ Xs[i+1]) then
        return i;
    end if
end for
return -1;                    ▷ target was not found in any source cell
```

Binary Search for Point-wise Remap



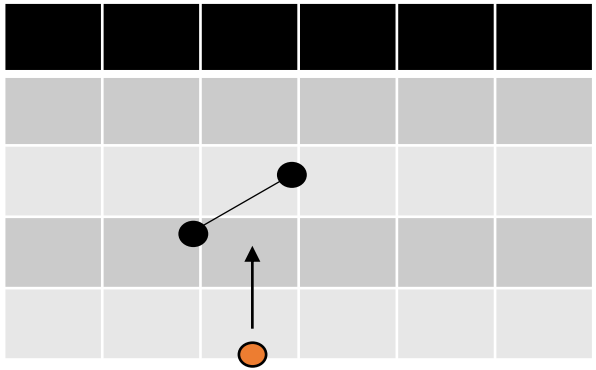
Algorithm 2 Binary Search Algorithm

```
Xt = x-target value
Xs = x-source value (held in an array)
n = number of source cells
initialize i to 0
initialize first to 0
initialize last to n
for i to n do
    initialize mid to first + ((last − first)/2)
    if (Xt ≥ Xs[mid]) and (Xt ≤ Xs[mid+1]) then
        return mid;
    end if
    if (Xt > Xs[mid+1]) then
        first = mid+1
        mid = (last-first)/2
    end if
    if (Xt < Xs[mid]) then
        last = mid;
        mid = (last-first)/2;
    end if
end for
return -1;                    ▷ target was not found in any source cell
```

Interpolation for Point-wise Remap

- Found slope and intercept of the linear approximation over found source cell.
- Found source cell which target cell lies in.
- Given a target point x_$targ$, we can find an approximate y_targ by using:

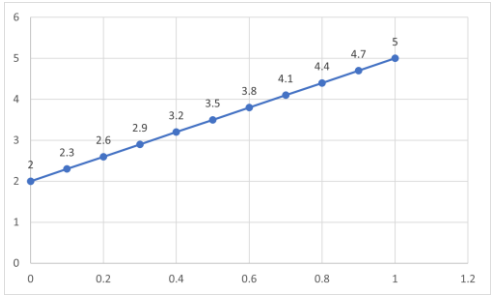
$$y_{targ} = m_{source} x_{targ} + b_{source}$$

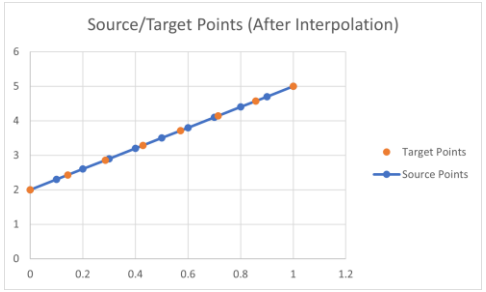


Point-wise Remap Results

Linear Equation ($y = 3x + 2$)

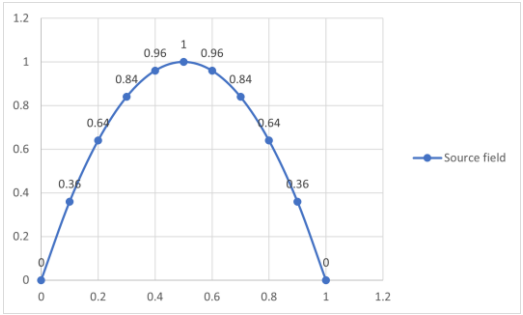
	Source	Target
#Cells	10	7
#Nodes	11	8

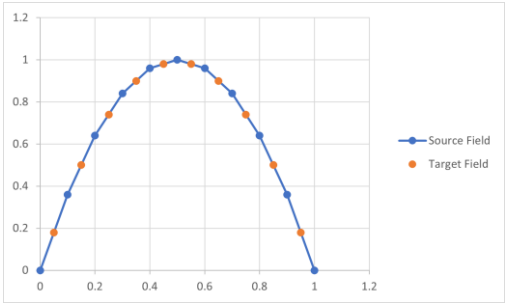




Quadratic Equation ($y = -4\left(x - \frac{1}{2}\right)^2 + 1$)

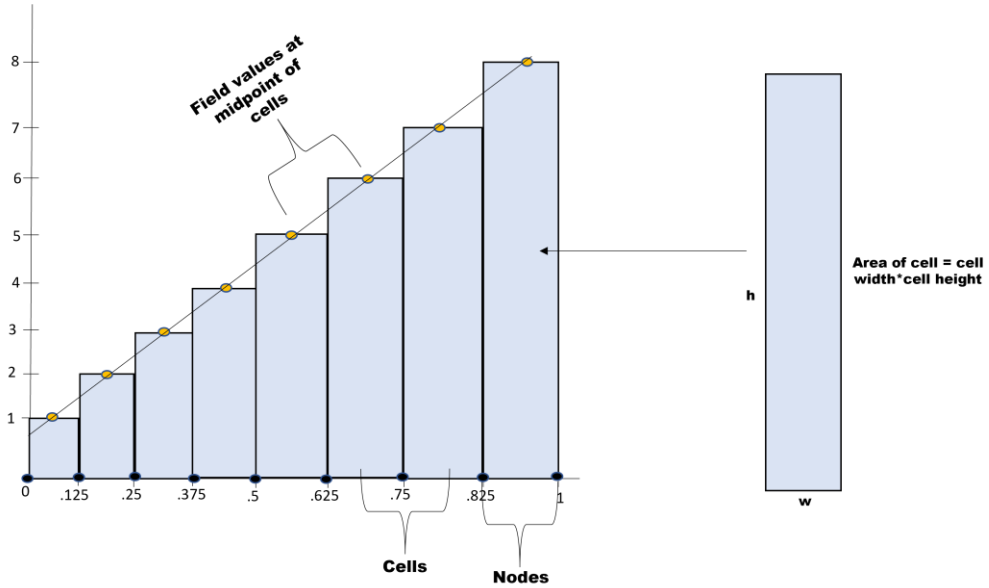
	Source	Target
#Cells	10	9
#Nodes	11	10



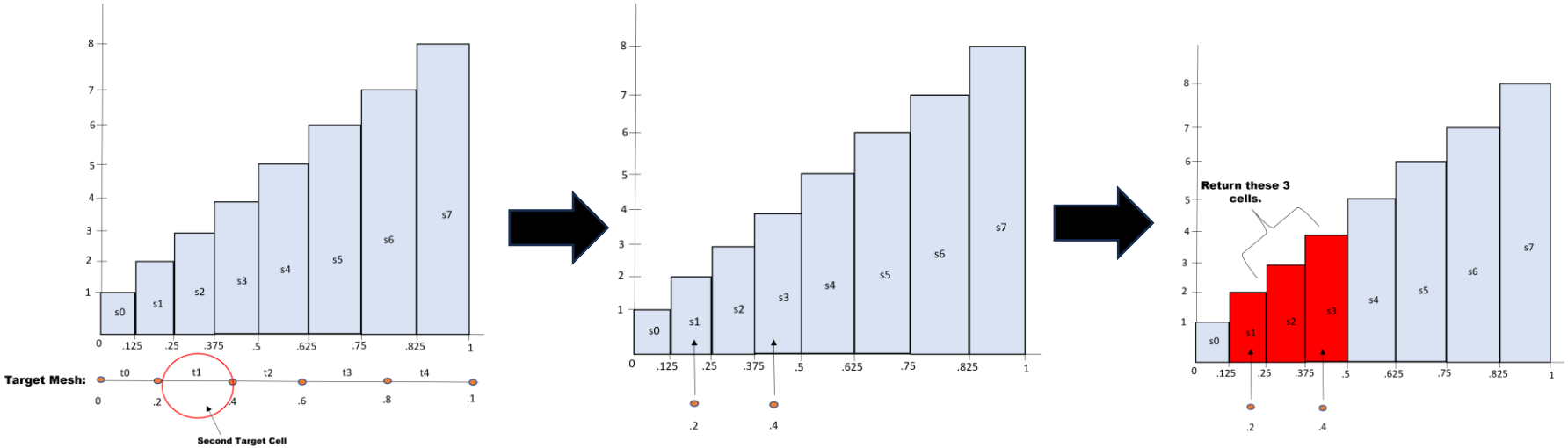


Conservative Remap

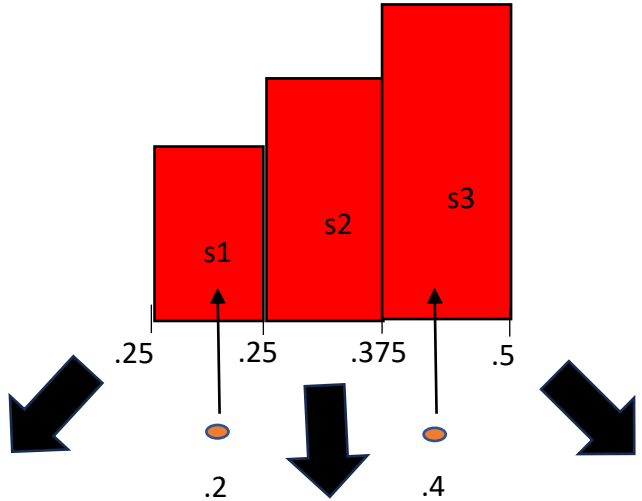
- Uses area under the curve of a source field to interpolate field data on a target mesh.
- Points at the middle of the cell make up the source field.
- 3-step process:
  - Search
  - Intersect
  - Interpolate

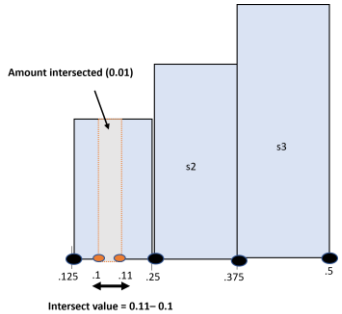


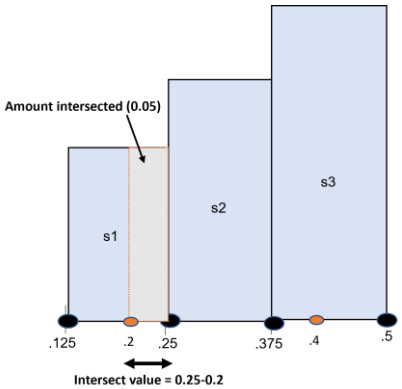
Search for Conservative Remap

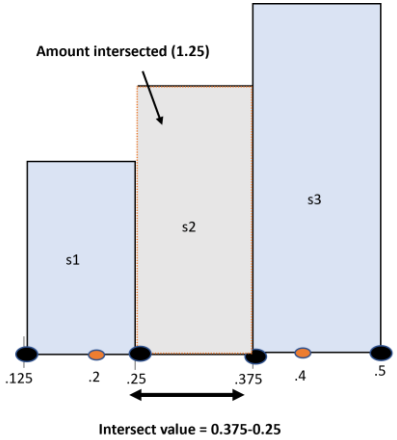


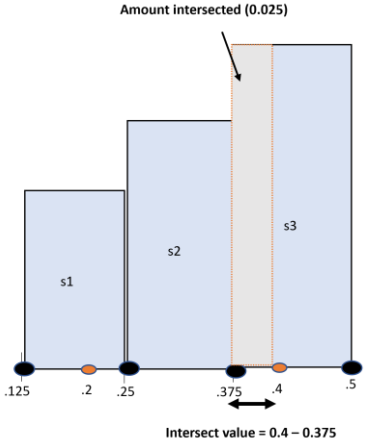
Intersect


s1
s2
s3
.25
.25
.375
.5
.2
.4


Amount intersected (0.01)
s2
s3
.125
.1
.11
.25
.375
.5
Intersect value = 0.11– 0.1


Amount intersected (0.05)
s1
s2
s3
.125
.2
.25
.375
.4
.5
Intersect value = 0.25-0.2


Amount intersected (1.25)
s1
s2
s3
.125
.2
.25
.375
.4
.5
Intersect value = 0.375-0.25


Amount intersected (0.025)
s1
s2
s3
.125
.2
.25
.375
.4
.5
Intersect value = 0.4 – 0.375

Interpolate

- We get the field value using target cell width and the target cell area.
- Target cell width:

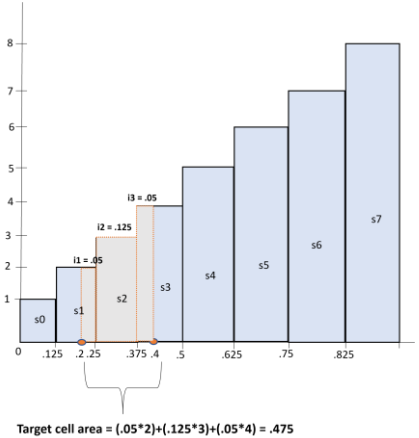
$$h_t = x_{end} - x_{start}$$

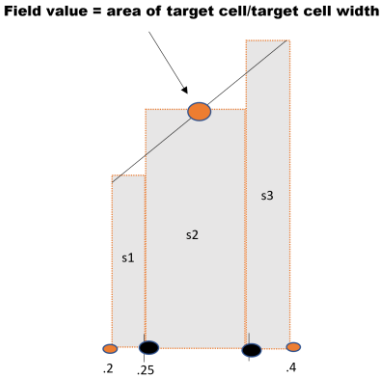
- Target cell area:

$$A_t = (\sum_{n=1}^{N_s} I_n^{st} * y_n^s)$$

- Target field value at the midpoint of the cell:

$$y_t = A_t / h_t$$

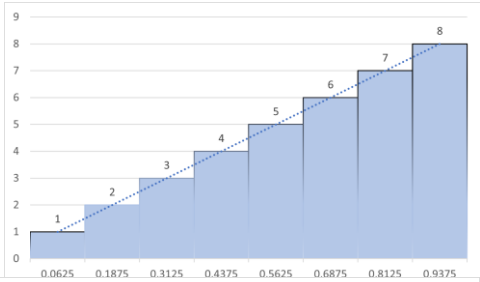


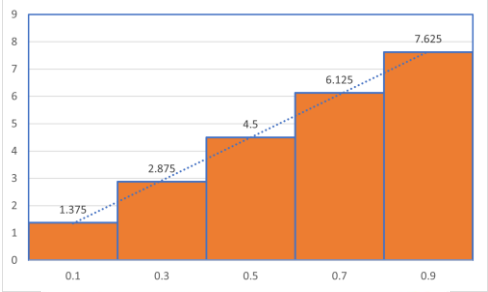


Results for Conservative Remap

Linear Equation ($y = 8x + 1/2$)

	Source	Target
#Cells	8	5
#Nodes	9	6





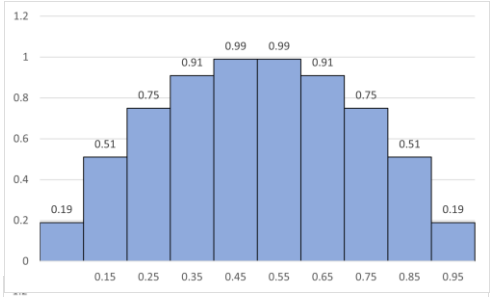
Area under curve preserved

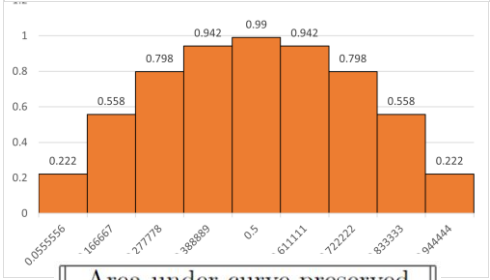
Source: 4.5000000000000000

Target: 4.5000000000000000

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

	Source	Target
#Cells	10	9
#Nodes	11	10





Area under curve preserved

Source: 0.6700000166893005

Target: 0.6700000166893005

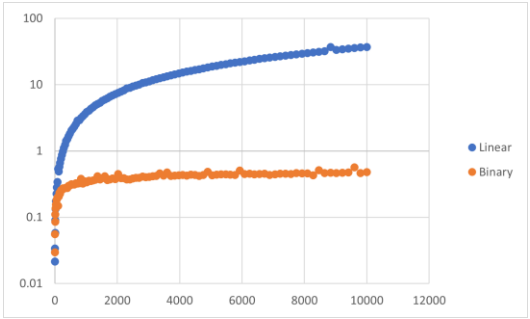
Profiling

- Used C++ ”Chronos” library.
- Used “chrono::high_resolution_clock::now()” function to register the start and end times for our searches.
- Used timing blocks around the loops for the searches.
- The average time:

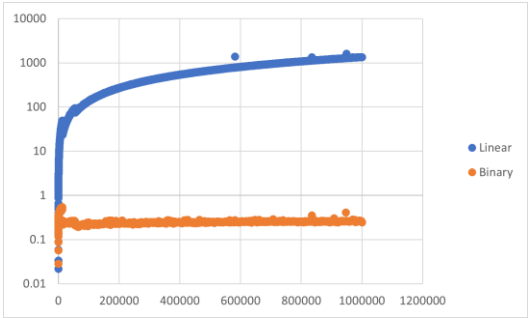
$$T_{avg} = \frac{T_{end} - T_{start}}{n_t}$$

- n_t = 100 target cells.

Point-wise

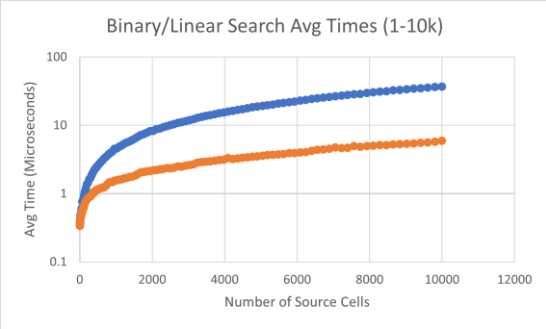


1-10,000 source cells

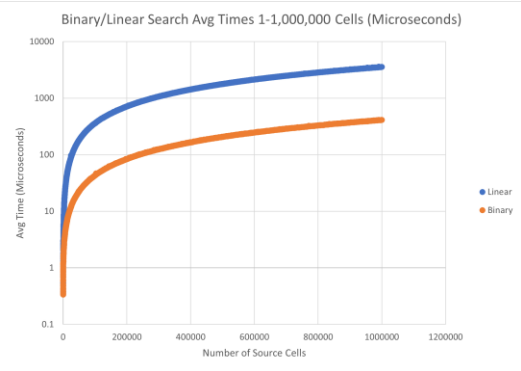


1-1,000,000 source cells

Conservative



1-10,000 source cells



1-1,000,000 source cells

Conclusion

- Completed study of two data remap algorithms
  - Pointwise interpolates node-based fields.
  - Conservative interpolates cell-center based fields.

- We found that while point-wise does allow us to complete remap, it doesn’t conserve mass, which is important when dealing with meshes containing specific properties that require the use and conservation of mass in order to be represented properly.

- We also studied performance of linear and binary searches.
  - Binary outperforms the linear search on sorted data.

Thank You! Questions?