# *Patch recovery and parallel rendezvous*

**David Neckels**

National Center for

Atmospheric Research

# *Patch recovery interpolation*

# Interpolating atmospheric winds

To compute the stress on the ocean surface, we require the atmospheric wind velocity on the ocean grid.

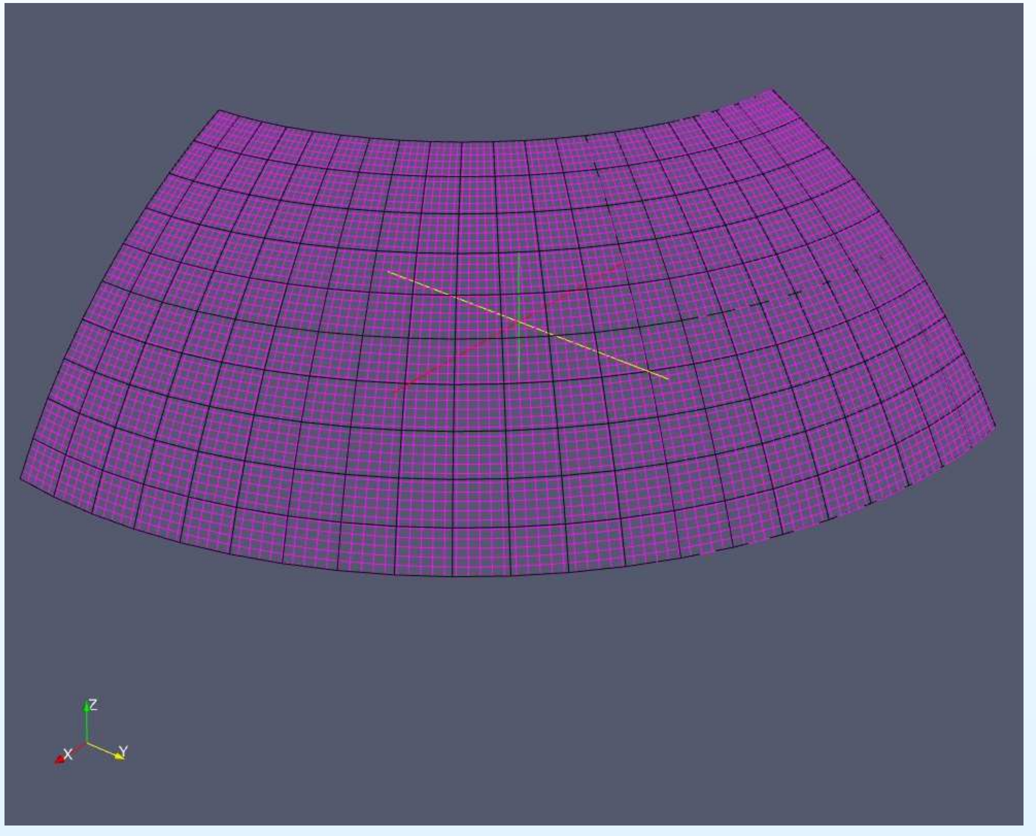
## Interpolating atmospheric winds

To compute the stress on the ocean surface, we require the atmospheric wind velocity on the ocean grid.

Typically the atmospheric grid scale is much coarser than the ocean grid scale

# Interpolating atmospheric winds

To compute the stress on the ocean surface, we require the atmospheric wind velocity on the ocean grid.

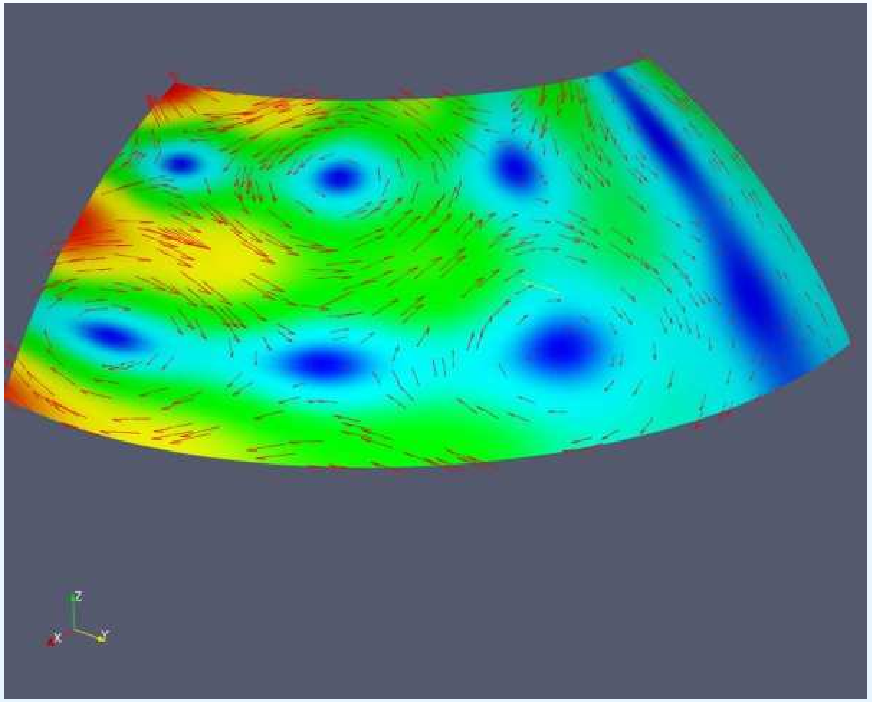


# Example:Interpolating atmospheric winds, ...

Consider an analytic flow pattern

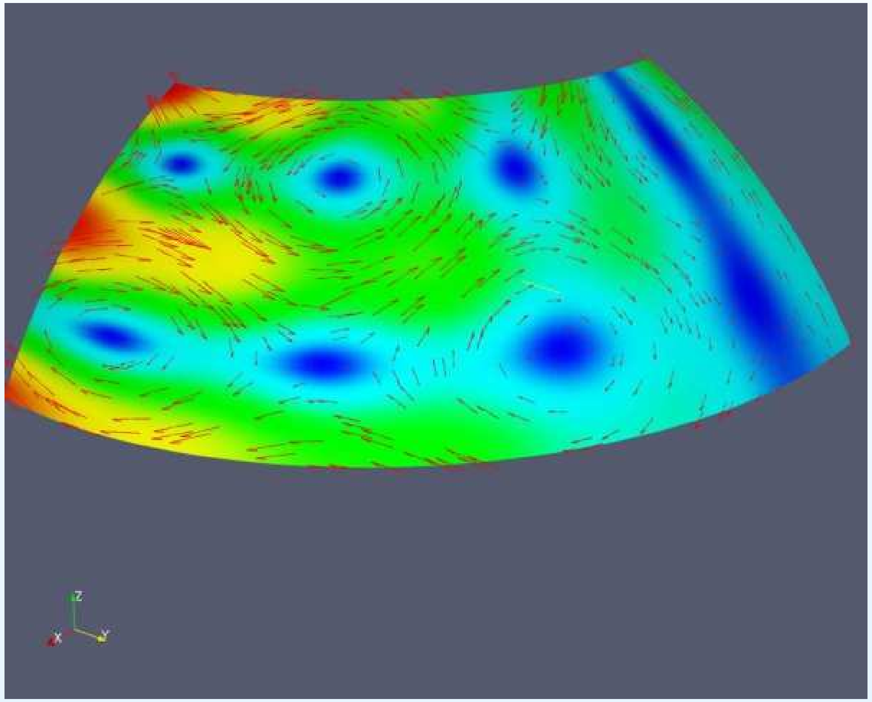
# Example:Interpolating atmospheric winds, ...

Consider an analytic flow pattern



# Example:Interpolating atmospheric winds, ...

Consider an analytic flow pattern



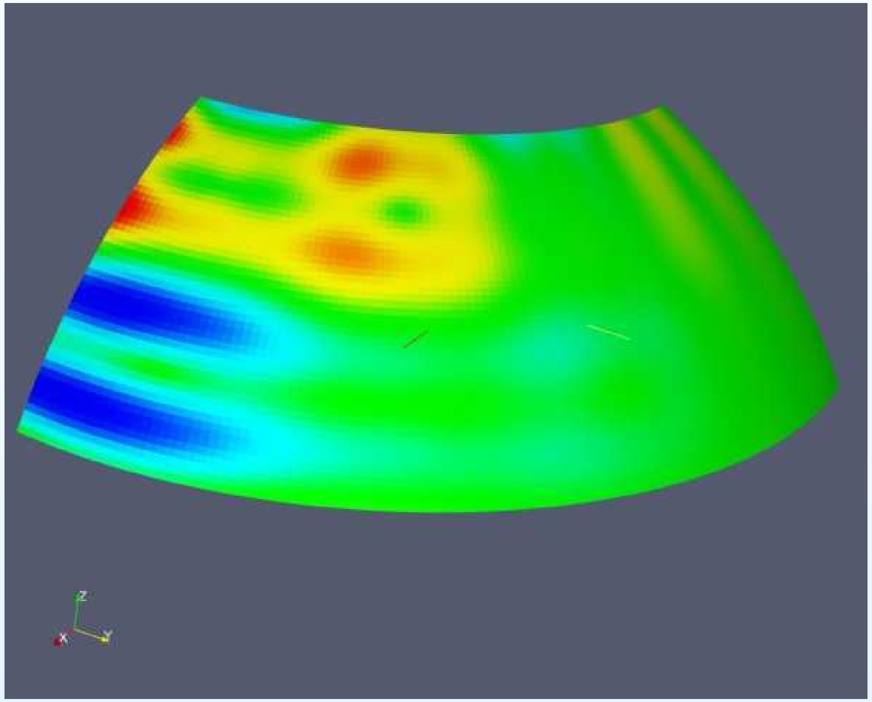

Of interest is the surface stress $\tau = C_p|U|U$, where $U = (u, v)$, especially $\nabla \times \tau$.

# Example: curl of tau

Curl of the analytic flow on the ocean grid is smooth

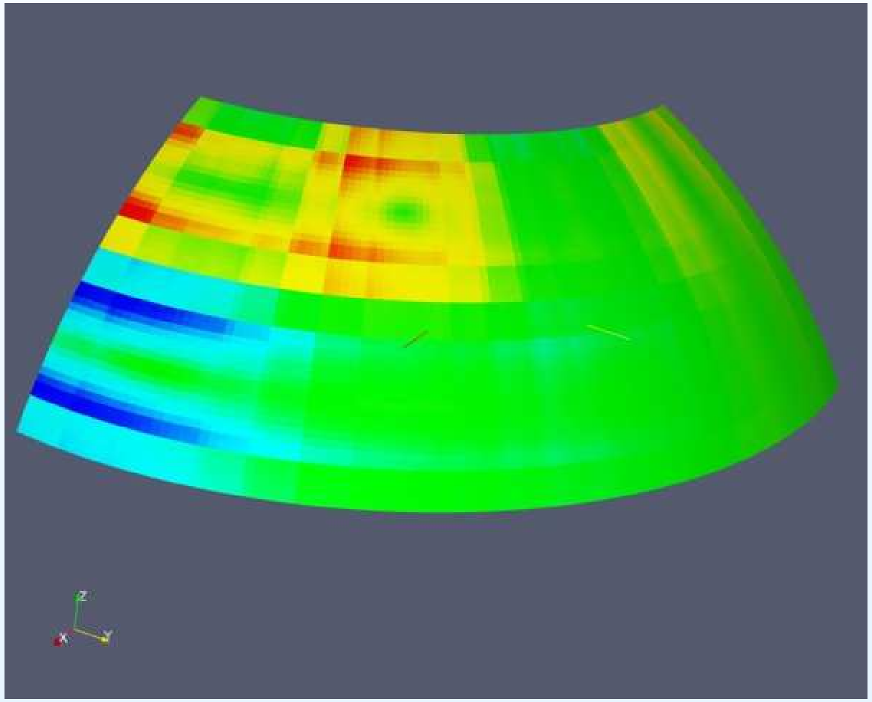
# Example: curl of tau

Curl of the analytic flow on the ocean grid is smooth



# Example: curl of tau

Curl of the analytic flow on the ocean grid is smooth




Curl of the standard bi-linear interpolant is not!

## Computation aspects of interpolation

To compute the interpolant from two distinct grids, there are several key steps

# Computation aspects of interpolation

To compute the interpolant from two distinct grids, there are several key steps

- Parallel rendezvous

# Computation aspects of interpolation

To compute the interpolant from two distinct grids, there are several key steps

- Parallel rendezvous
- Search (point in box)

# Computation aspects of interpolation

To compute the interpolant from two distinct grids, there are several key steps

- Parallel rendezvous
- Search (point in box)
- Interpolation method (bi-linear, conservative, patch...)

# Computation aspects of interpolation

To compute the interpolant from two distinct grids, there are several key steps

- Parallel rendezvous
- Search (point in box)
- Interpolation method (bi-linear, conservative, patch...)

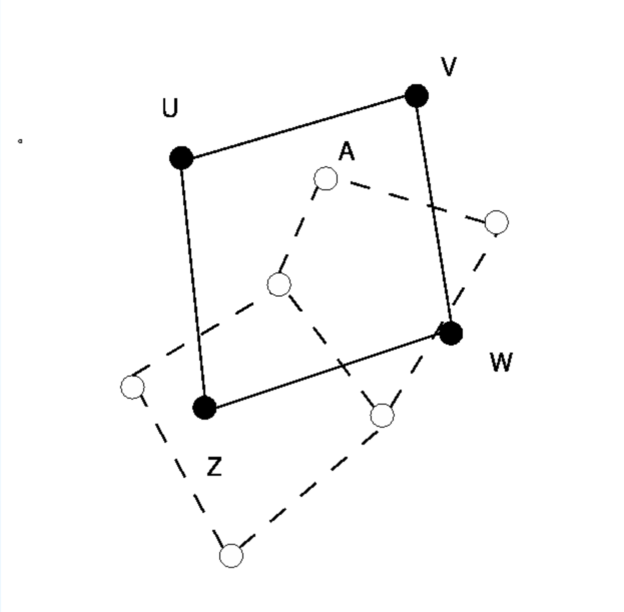
Each of these topics is its own talk. We begin with the Interpolation method.

# Bi-linear interpolation

A standard interpolation scheme is the bi-linear scheme

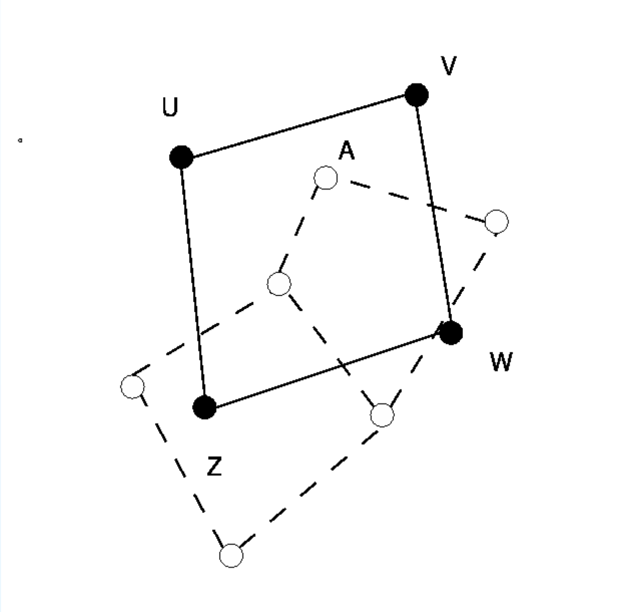
# Bi-linear interpolation

A standard interpolation scheme is the bi-linear scheme



# Bi-linear interpolation

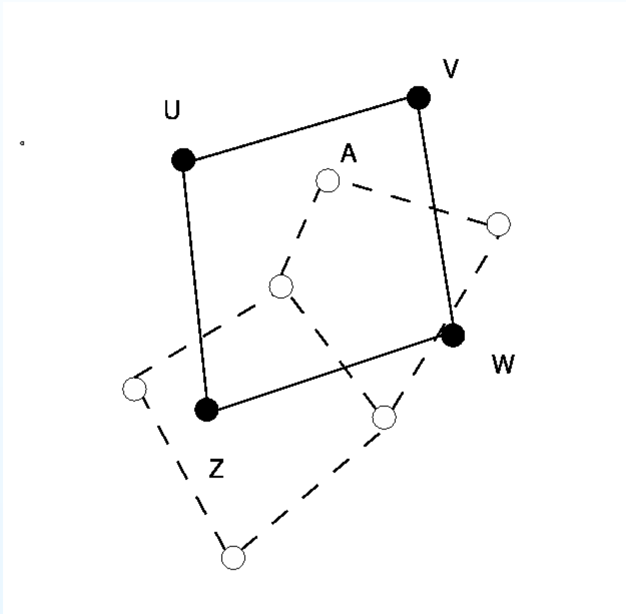
A standard interpolation scheme is the bi-linear scheme




The value at $A$ is a weighted sum of the values at $U, V, W, Z$, with the bi-linear shape functions $\phi$ as the weights.

# Bi-linear interpolation

A standard interpolation scheme is the bi-linear scheme

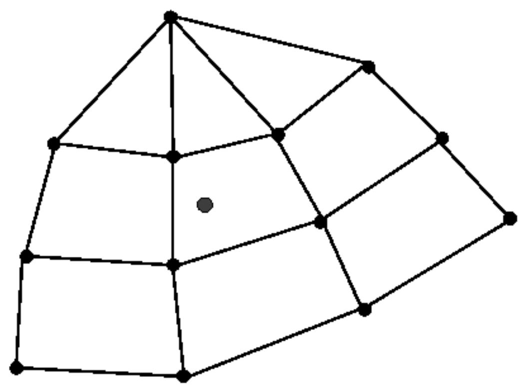



The value at $A$ is a weighted sum of the values at $U, V, W, Z$, with the bi-linear shape functions $\phi$ as the weights.

A reasonable approximation to $\nabla A$ is $U\nabla\phi_1 + \cdots + Z\nabla\phi_Z$.

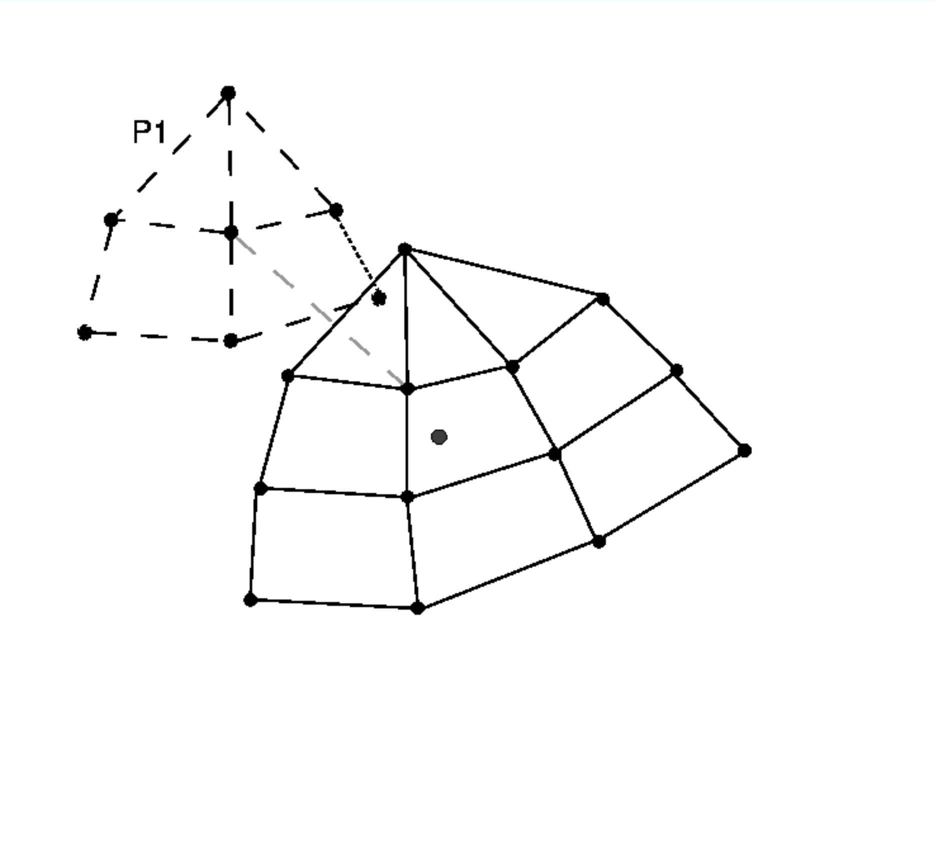
# Patch based methods

We form the interpolant at • using polynomials based on the node patches of the encompassing cell:



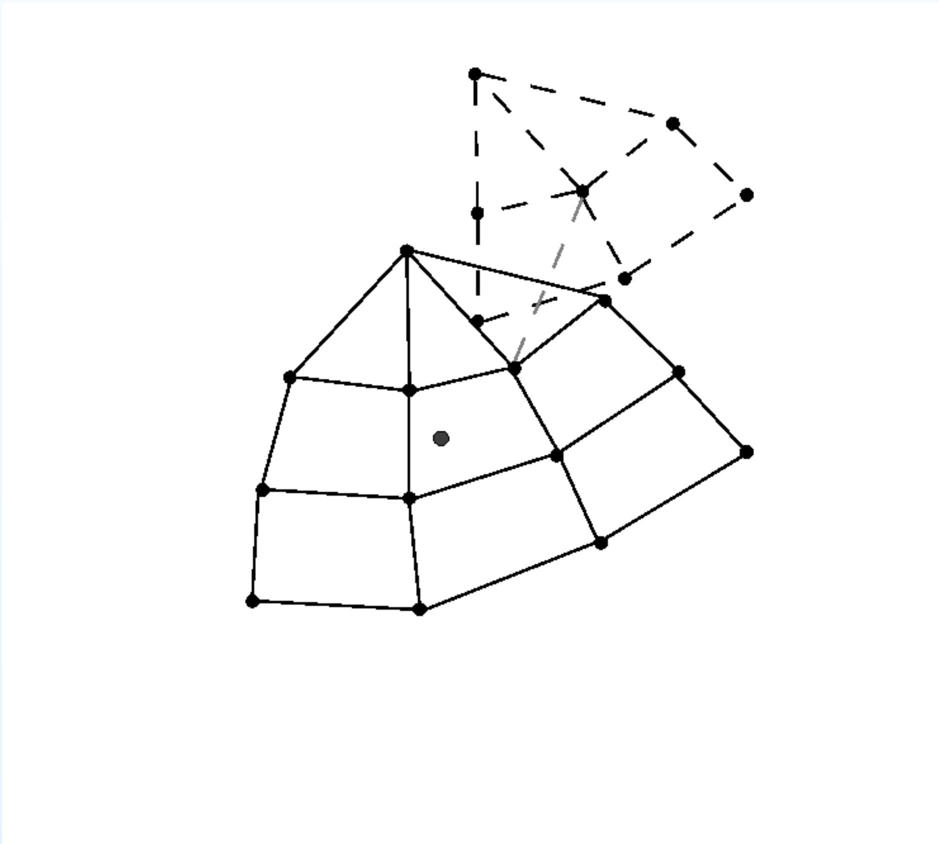
## Patch based methods

We form the interpolant at • using polynomials based on the node patches of the encompassing cell:



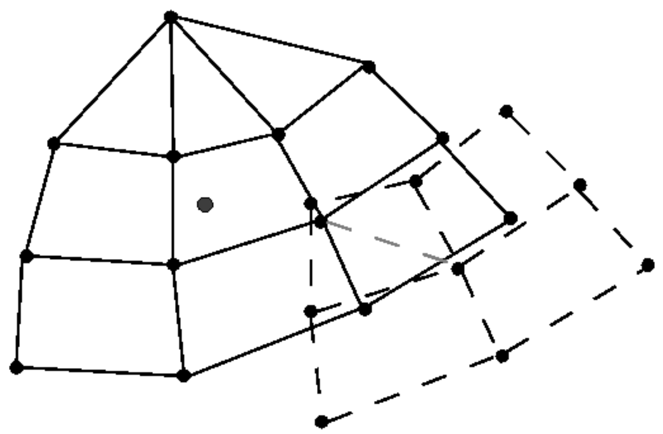
# Patch based methods

We form the interpolant at • using polynomials based on the node patches of the encompassing cell:



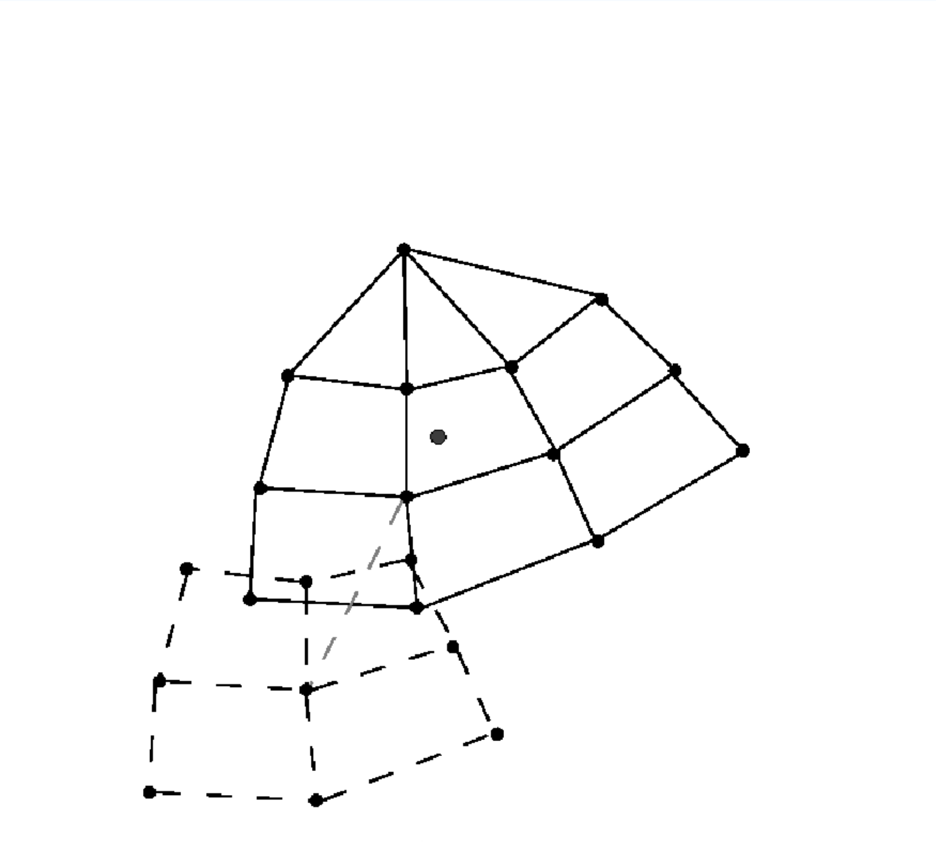
# Patch based methods

We form the interpolant at ● using polynomials based on the node patches of the encompassing cell:



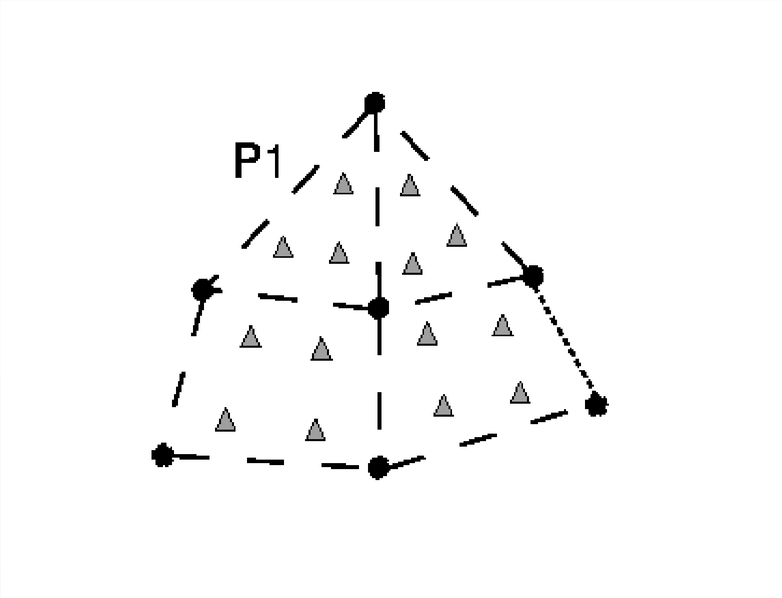
# Patch based methods

We form the interpolant at • using polynomials based on the node patches of the encompassing cell:



# Patch based methods,...

On each patch we sample the source function at a set of sample points (usually quadrature points) $\triangle$, using local bi-linear interpolation if necessary.




Call these samples $s_i$ at (local 2D) coordinates $p_i$.

## Local polynomial approximation

We fit a tensor product polynomial through these values, solving for the polynomial coefficients $c$

$$\min_c \sum_i (Q(c, p_i) - s_i)^2$$

# Local polynomial approximation

We fit a tensor product polynomial through these values, solving for the polynomial coefficients $c$

$$\min_c \sum_i (Q(c, p_i) - s_i)^2$$

Which yields the least squares system $A^\top Ac = A^\top s$ and $Q(p) = b(p)^T (A^\top A)^{-1} s$ where $b$ is the vector of the polynomial basis functions evaluated at the sample points.

# Local polynomial approximation

We fit a tensor product polynomial through these values, solving for the polynomial coefficients $c$

$$\min_c \sum_i (Q(c, p_i) - s_i)^2$$

Which yields the least squares system $A^\top A c = A^\top s$ and $Q(p) = b(p)^T (A^\top A)^{-1} s$ where $b$ is the vector of the polynomial basis functions evaluated at the sample points.

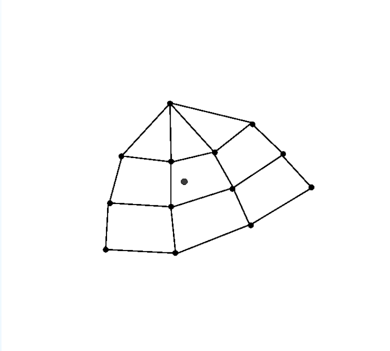
On a manifold, the local coordinates $p_i$ may either be full 3D coordinates, or the coefficients of the co-space of a reasonable normal.
This avoids pole type singularities in the patch algorithm (i.e. don't use lat/lon).

# Blending the patches

We use any partition of unity on the cell to blend the patches for a value $F(x) = \sum_j \psi_j(x)Q(x)$, for instance the bi-linear basis.

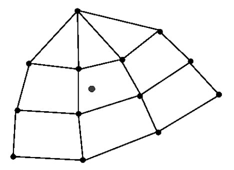
# Blending the patches

We use any partition of unity on the cell to blend the patches for a value $F(x) = \sum_j \psi_j(x) Q(x)$, for instance the bi-linear basis.



# Blending the patches

We use any partition of unity on the cell to blend the patches for a value $F(x) = \sum_j \psi_j(x) Q(x)$, for instance the bi-linear basis.



Explicitly, accounting for the local coordinate system $p = L(x)$ and the bi-linear interpolation to sample locations $s = \Phi f$, the interpolant is a linear function of the coefficients $f$ on this enlarged stencil,

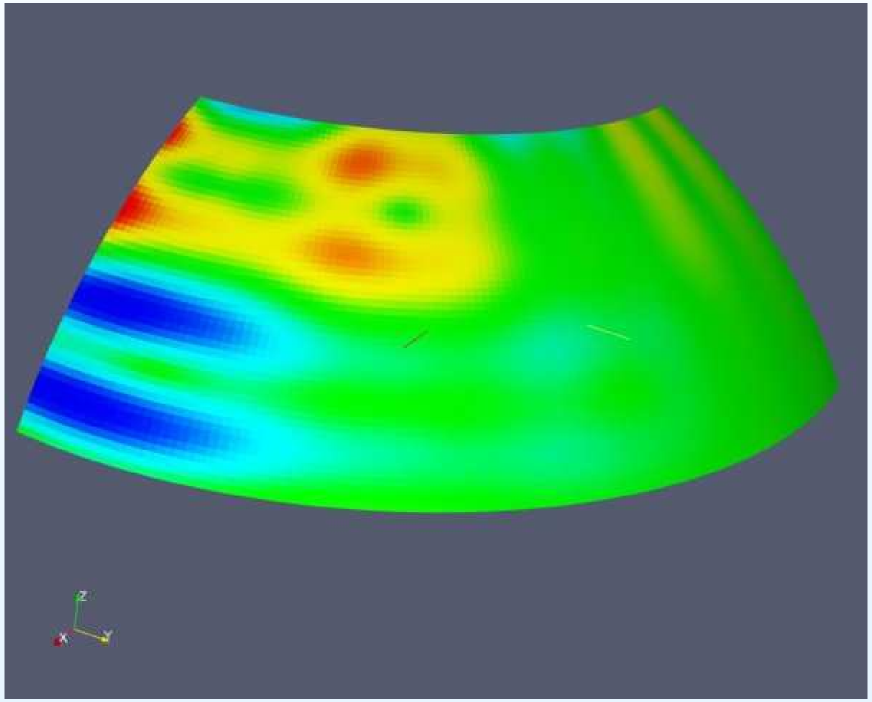
$$F(x) = \sum_j \left[\psi(x)\big(b \circ L(x)\big)^\top (A^\top A)^{-1} \Phi\right]_j f$$

# Back to curl of tau

Curl of the analytic flow on the ocean grid is smooth

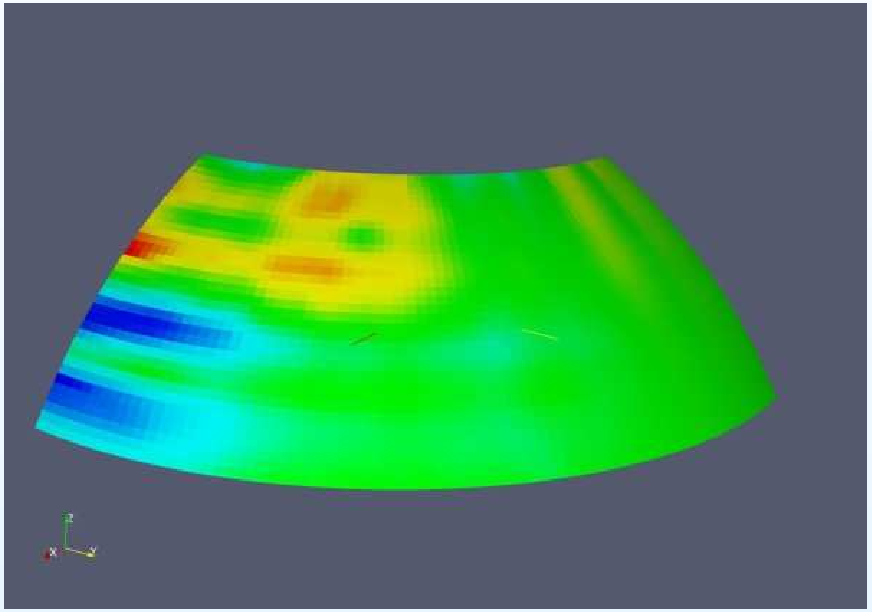
# Back to curl of tau

Curl of the analytic flow on the ocean grid is smooth



# Back to curl of tau

Curl of the analytic flow on the ocean grid is smooth



The patch recovery curl is more reasonable than the bi-linear!

## Some results from interpolation theory

Interpolating a function $f(x)$ into the space of continuous piecewise polynomial functions of order $p$ on a discretization $\mathcal{T}_h$, with cell diameters $h$, using exact values of $f$ at the nodes yields

## Some results from interpolation theory

Interpolating a function $f(x)$ into the space of continuous piecewise polynomial functions of order $p$ on a discretization $\mathcal{T}_h$, with cell diameters $h$, using exact values of $f$ at the nodes yields

$$||D^m(f - \mathcal{I}f)||_{L^2} \leq Ch^{(p+1)-m}||D^{p+1}f||_{L^2}$$

## Some results from interpolation theory

Interpolating a function $f(x)$ into the space of continuous piecewise polynomial functions of order $p$ on a discretization $\mathcal{T}_h$, with cell diameters $h$, using exact values of $f$ at the nodes yields

$$||D^m(f - \mathcal{I}f)||_{L^2} \leq Ch^{(p+1)-m}||D^{p+1}f||_{L^2}$$

i.e. for bi-linear interpolation

$$||f - \mathcal{I}f||_{L^2} \leq Ch^2||D^2 f||_{L^2}$$

## Some results from interpolation theory

Interpolating a function $f(x)$ into the space of continuous piecewise polynomial functions of order $p$ on a discretization $\mathcal{T}_h$, with cell diameters $h$, using exact values of $f$ at the nodes yields

$$||D^m(f - \mathcal{I}f)||_{L^2} \leq Ch^{(p+1)-m}||D^{p+1}f||_{L^2}$$

i.e. for bi-linear interpolation

$$||f - \mathcal{I}f||_{L^2} \leq Ch^2||D^2f||_{L^2}$$

and

$$||\nabla(f - \mathcal{I}f)||_{L^2} \leq Ch||D^2f||_{L^2}$$

## Some results from interpolation theory

Interpolating a function $f(x)$ into the space of continuous piecewise polynomial functions of order $p$ on a discretization $\mathcal{T}_h$, with cell diameters $h$, using exact values of $f$ at the nodes yields

$$||D^m(f - \mathcal{I}f)||_{L^2} \leq Ch^{(p+1)-m}||D^{p+1}f||_{L^2}$$

i.e. for bi-linear interpolation

$$||f - \mathcal{I}f||_{L^2} \leq Ch^2||D^2f||_{L^2}$$

and

$$||\nabla(f - \mathcal{I}f)||_{L^2} \leq Ch||D^2f||_{L^2}$$

Smoothness is required, at least of weak derivatives $||D^2f||_{L^2}$.

## An experiment

We perform a convergence study for the analytic function

$$f(x,y) = (1 - xy)\sin 3\pi x \cos 2\pi y)$$

on the unit square using patch and bi-linear interpolation.

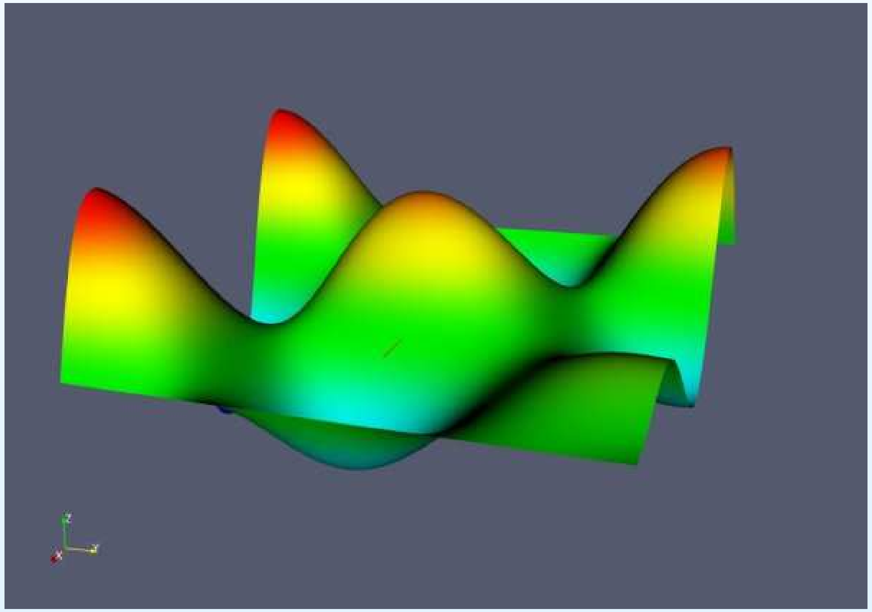
## An experiment

We perform a convergence study for the analytic function

$$f(x,y) = (1 - xy)\sin 3\pi x \cos 2\pi y)$$

on the unit square using patch and bi-linear interpolation.

Exact



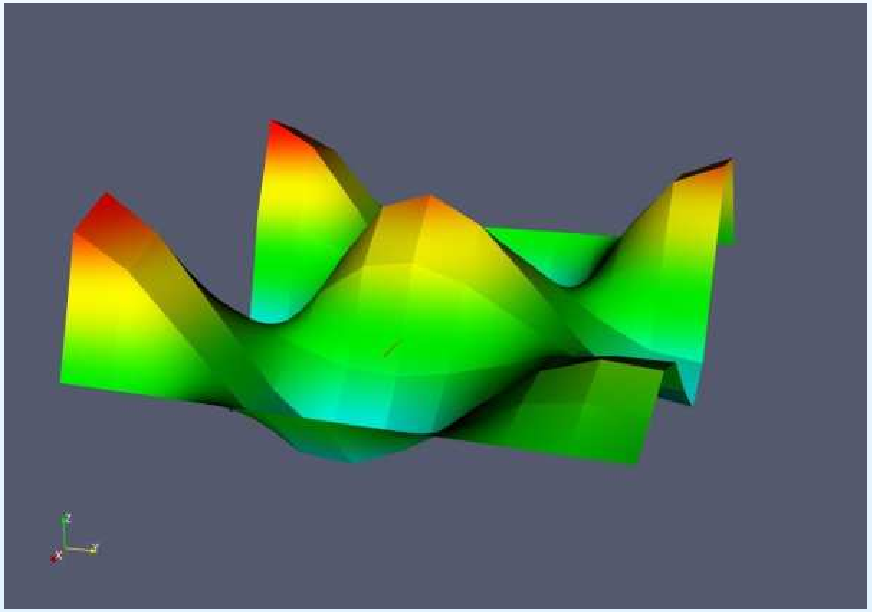
## An experiment

We perform a convergence study for the analytic function

$$f(x,y) = (1 - xy)\sin 3\pi x \cos 2\pi y)$$

on the unit square using patch and bi-linear interpolation.

Bilinear



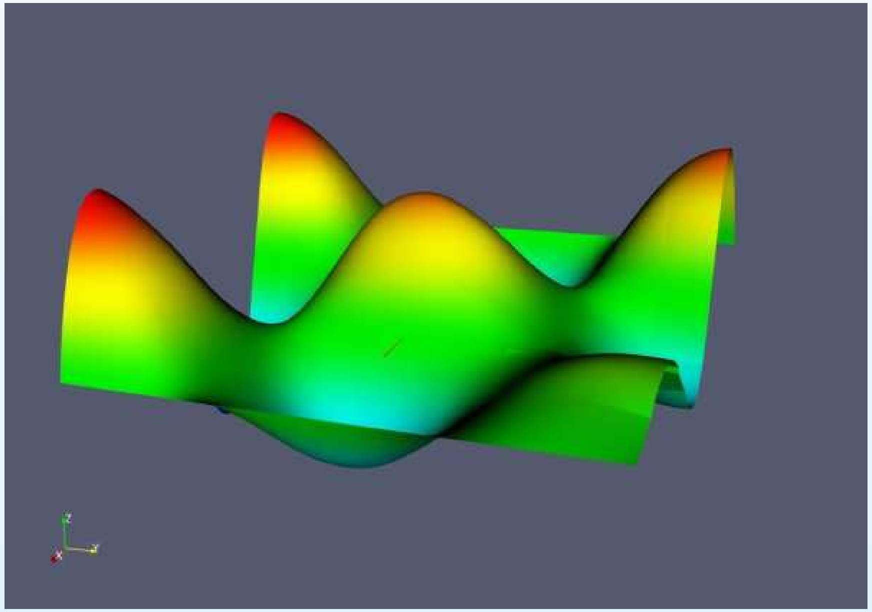
# An experiment

We perform a convergence study for the analytic function

$$f(x, y) = (1 - xy)\sin 3\pi x \cos 2\pi y)$$

on the unit square using patch and bi-linear interpolation.

Patch

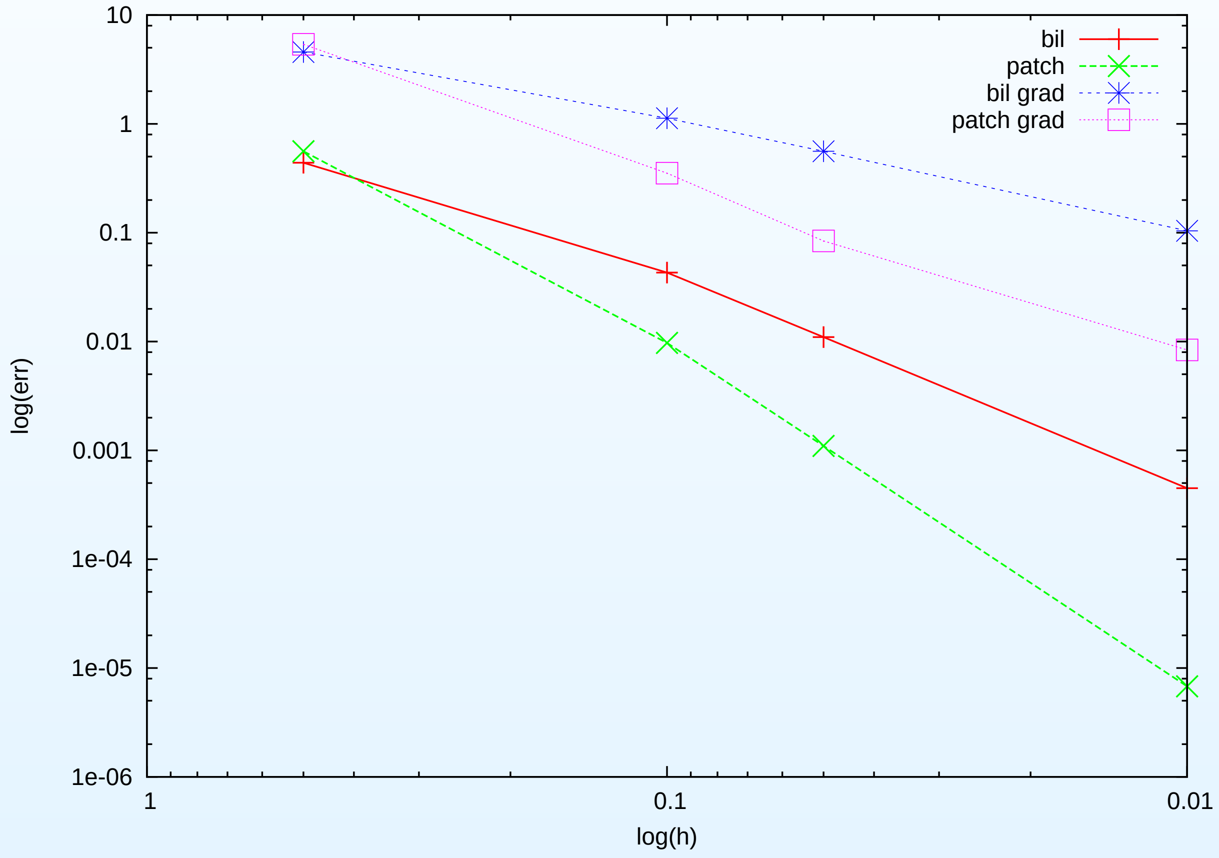


# Results

We compute the $L^2$ error on a super fine grid.

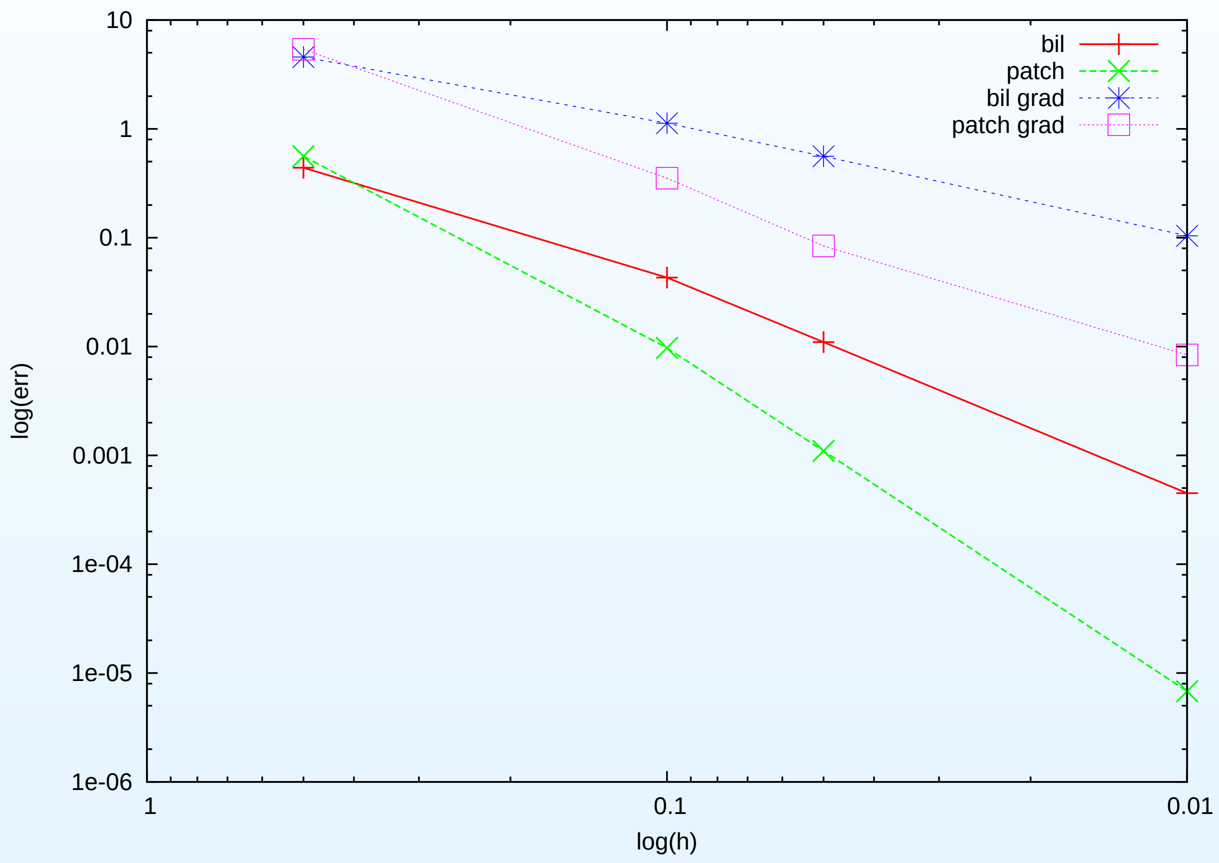
# Results

We compute the $L^2$ error on a super fine grid.



# Results

We compute the $L^2$ error on a super fine grid.



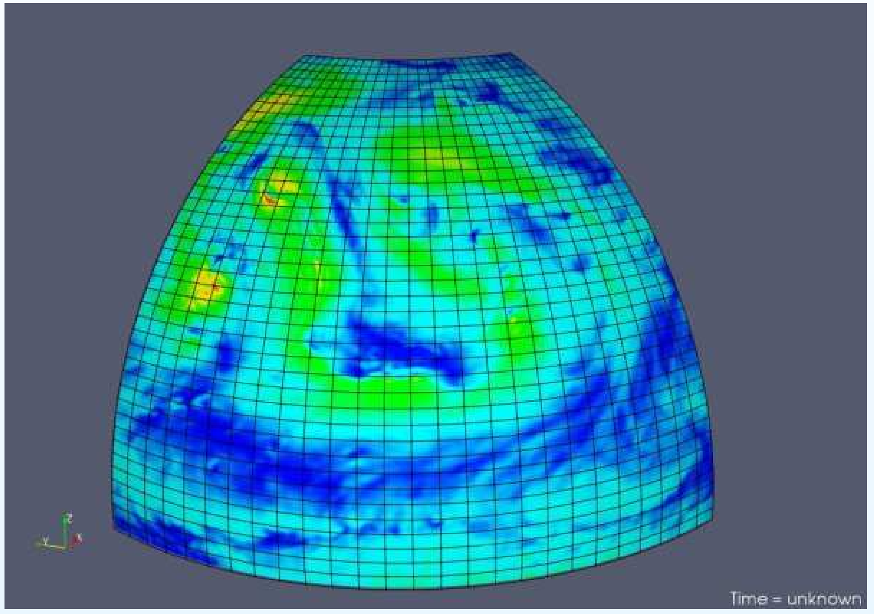

Rates are $P = 3.14, B = 1.96, \nabla P = 2.01, \nabla B = 1.01$.

# A real wind field

We compare interpolation methods on realistic wind data

# A real wind field

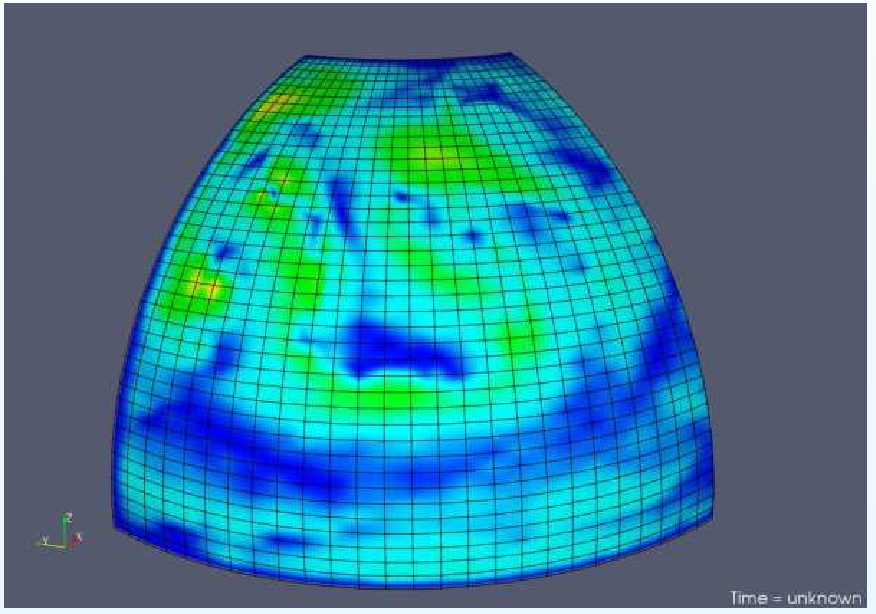
We compare interpolation methods on realistic wind data




The exact wind field ($|U|$)

## A real wind field

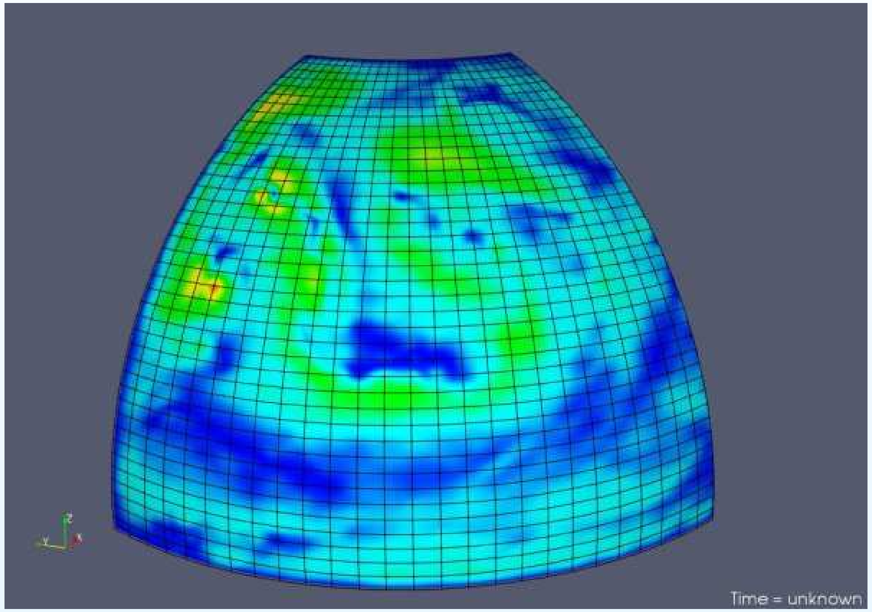
We compare interpolation methods on realistic wind data




The bi-linear interpolant

# A real wind field

We compare interpolation methods on realistic wind data



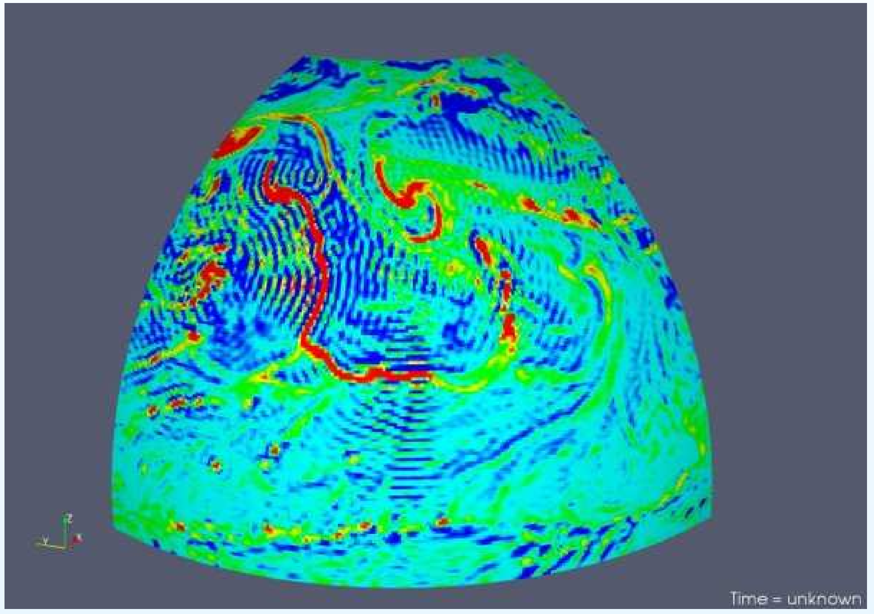

The patch interpolant

# Curl of the real wind field

We compare interpolation methods on realistic wind data

# Curl of the real wind field

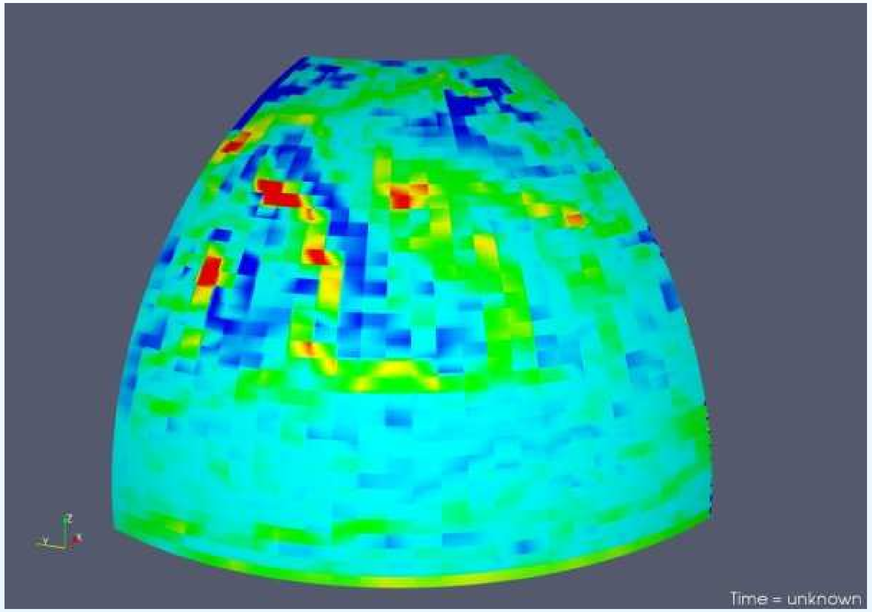
We compare interpolation methods on realistic wind data




The exact wind field ($\nabla \times U$)

# Curl of the real wind field

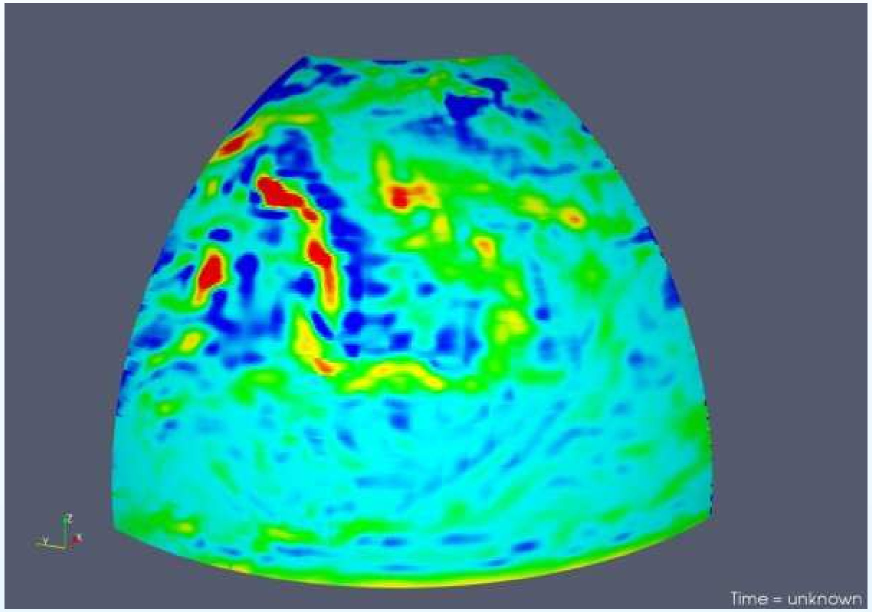
We compare interpolation methods on realistic wind data




The bi-linear interpolant

# Curl of the real wind field

We compare interpolation methods on realistic wind data




The patch interpolant

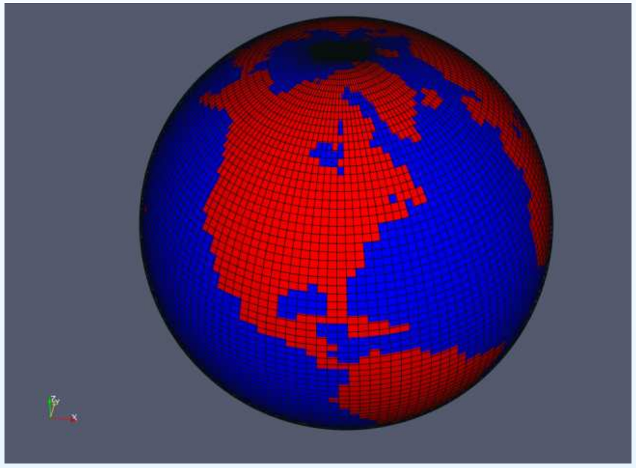
*Parallel rendezvous*

## Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.

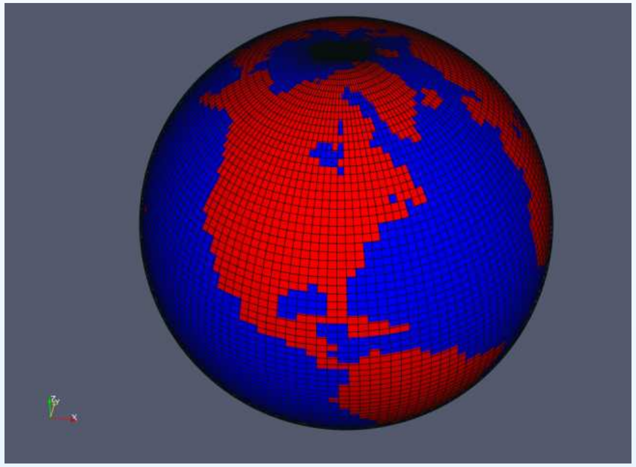
# Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



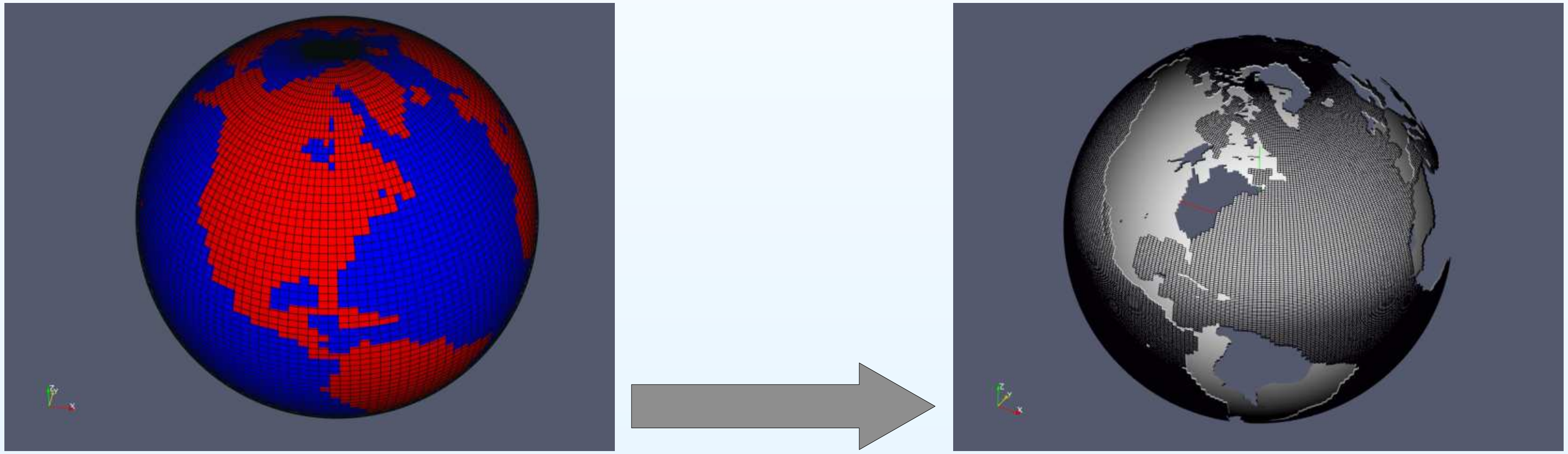
# Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



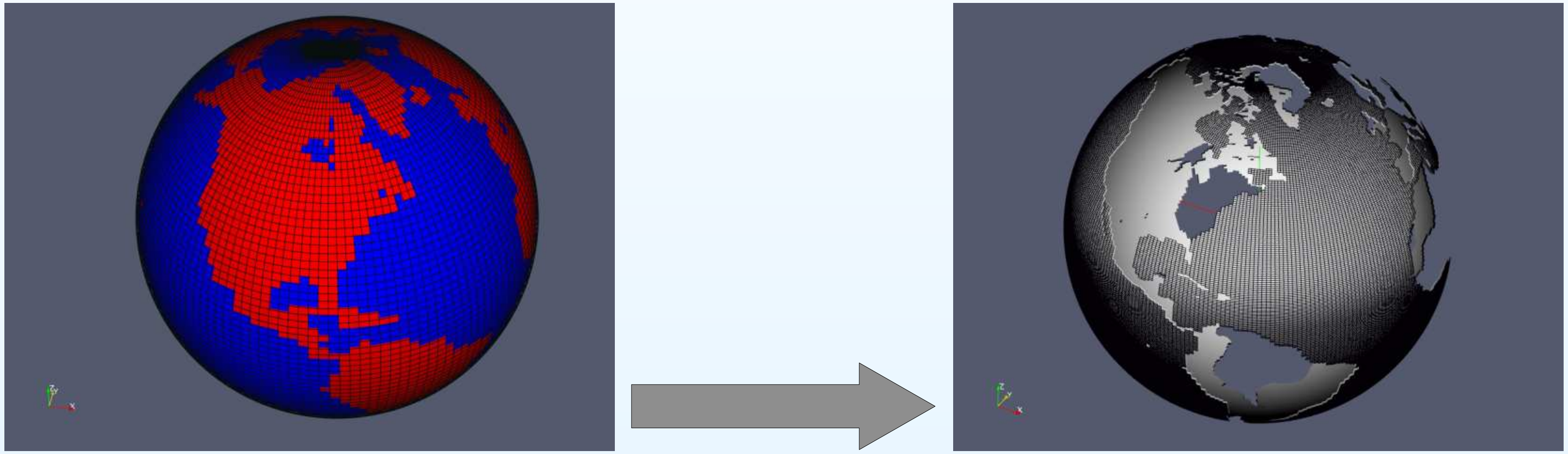
# Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



## Description of the Problem

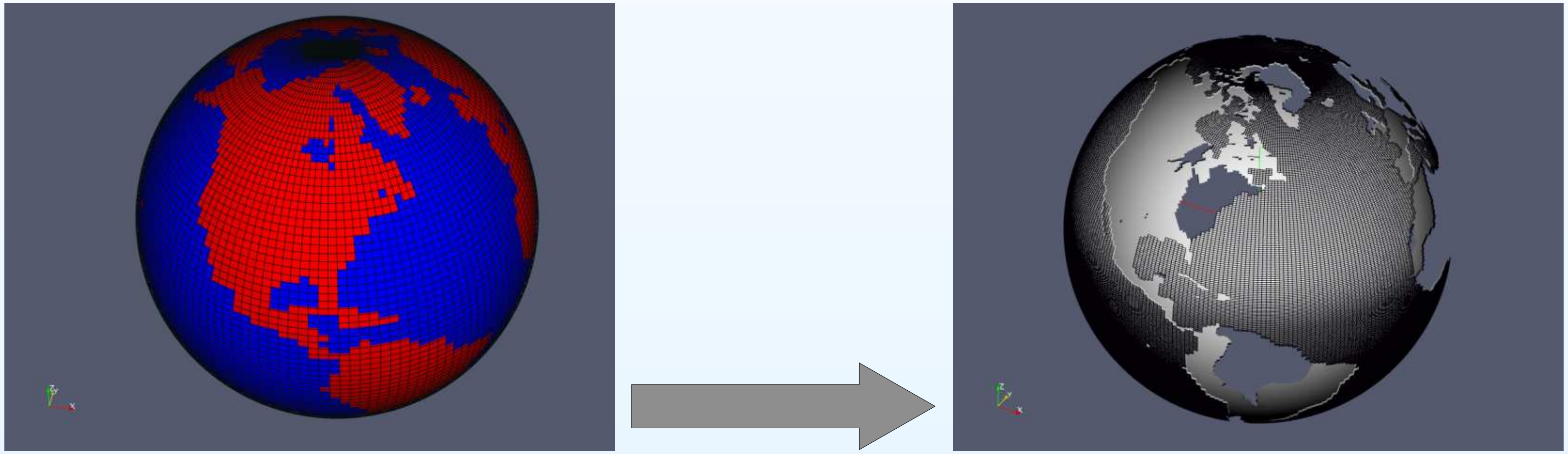
To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



How to calculate weights in an efficient/load balanced manner?

# Description of the Problem

To interpolate data from one grid(mesh) to another, where each is distributed, independently, in parallel.



How to calculate weights in an efficient/load balanced manner?

How to perform the interpolation in an efficient/load balanced manner?

# Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

# Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

# Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Destination points are shipped to the source grid decomposition for the search and weight calculation.

## Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Destination points are shipped to the source grid decomposition for the search and weight calculation.

These bounding boxes depend on a fixed coordinate system (which the two grids must negotiate), and optimal performance requires the parallel decomposition be roughly aligned with this coordinate system.

# Bounding box and load imbalance problem

In the most straightforward approach, bounding boxes for each processor's grid are shared amongst processors.

A destination cell (or point) locates the source processor with a cell(s) that contains it.

Destination points are shipped to the source grid decomposition for the search and weight calculation.

These bounding boxes depend on a fixed coordinate system (which the two grids must negotiate), and optimal performance requires the parallel decomposition be roughly aligned with this coordinate system.

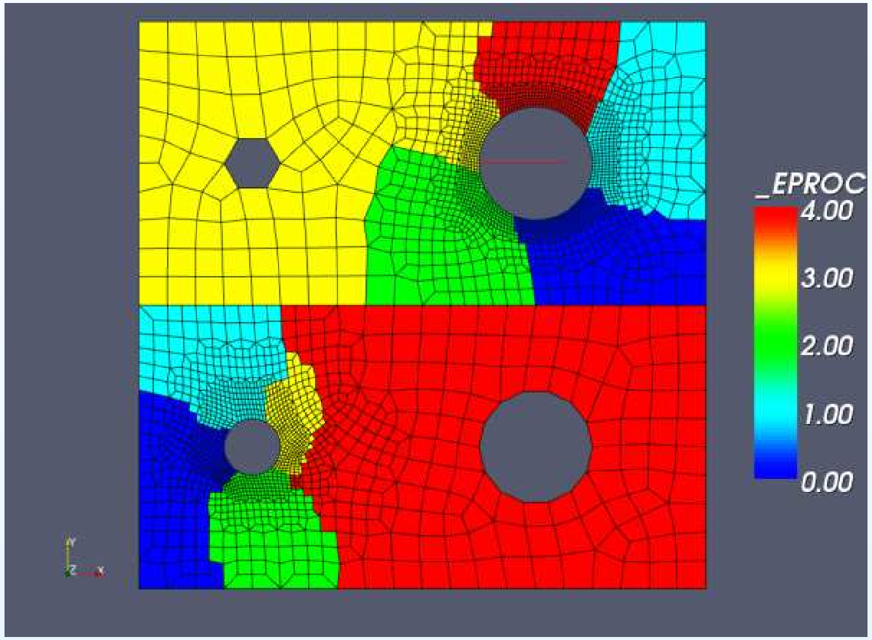
This condition is rarely satisfied.

# Load in balance problem

The interpolation problem is by nature geometric, but the grid decomposition is not necessary so

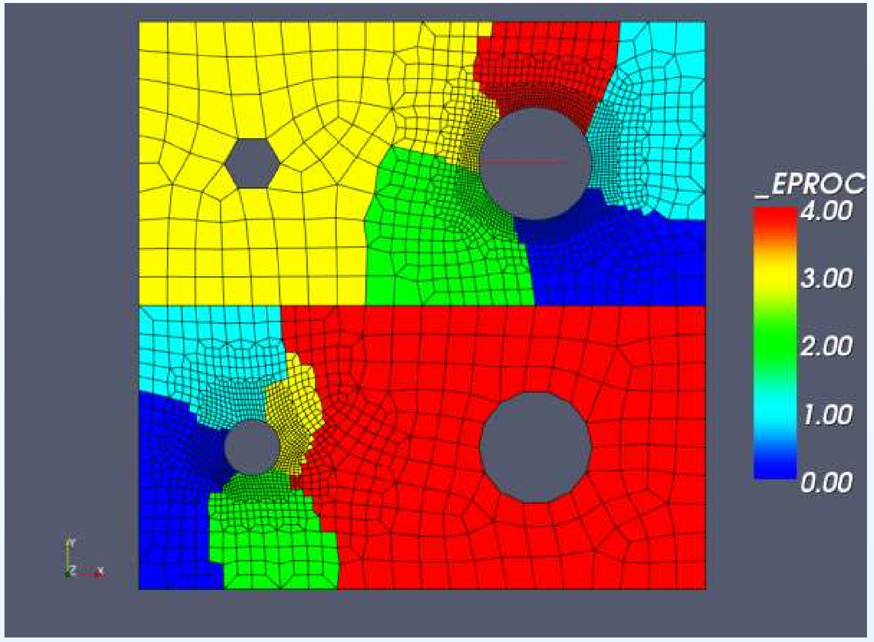
# Load in balance problem

The interpolation problem is by nature geometric, but the grid decomposition is not necessary so



# Load in balance problem

The interpolation problem is by nature geometric, but the grid decomposition is not necessary so



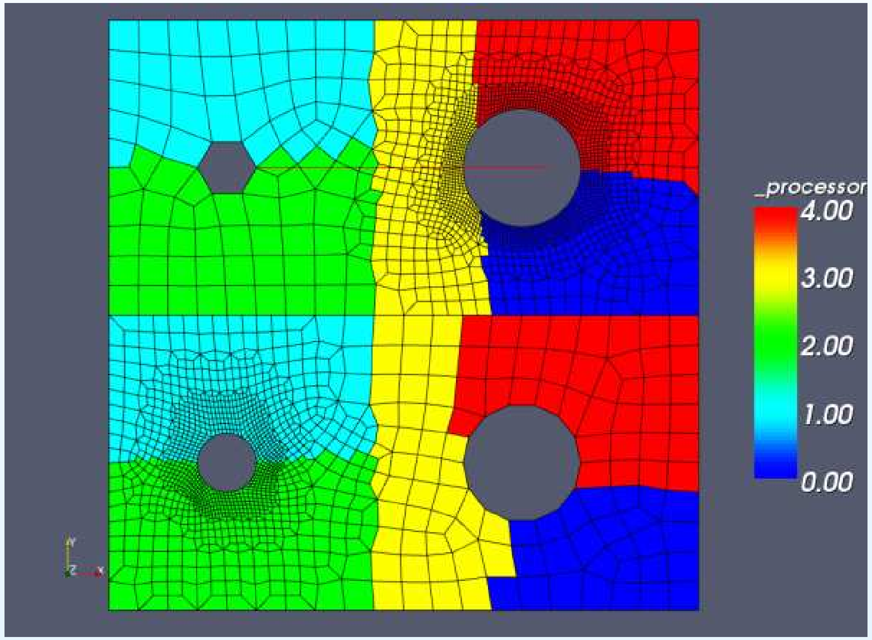

In the worst case, the entire source mesh may be shipped to one processor! This standard approach lacks robustness.

# A Geometric solution

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!

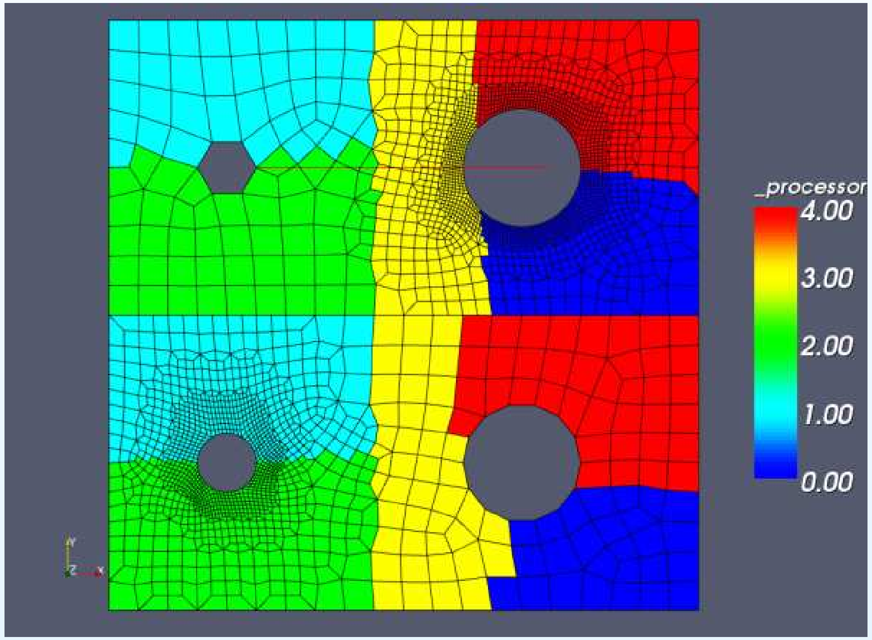
# A Geometric solution

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!



## A Geometric solution

We construct a new partition for each mesh such that the portions of each mesh on a given processor are geometrically collocated!



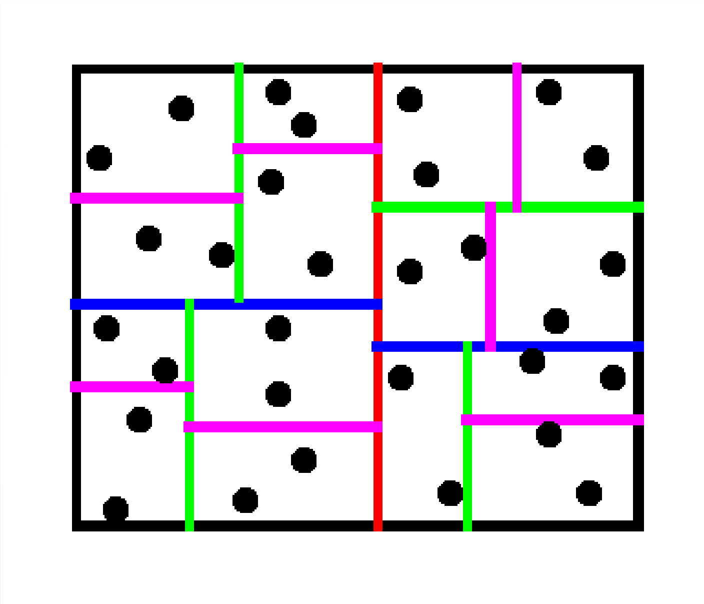

Also, the union of meshes is load balanced!

## RCB to the rescue

The Recursive Coordinate Bisection algorithm is a parallel algorithm for partitioning a set of geometric entities (possibly with weights). The package Zoltan provides this.

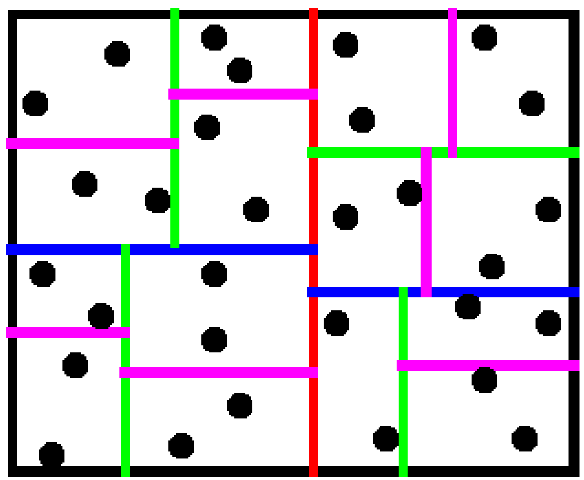
# RCB to the rescue

The Recursive Coordinate Bisection algorithm is a parallel algorithm for partitioning a set of geometric entities (possibly with weights). The package Zoltan provides this.



## RCB to the rescue

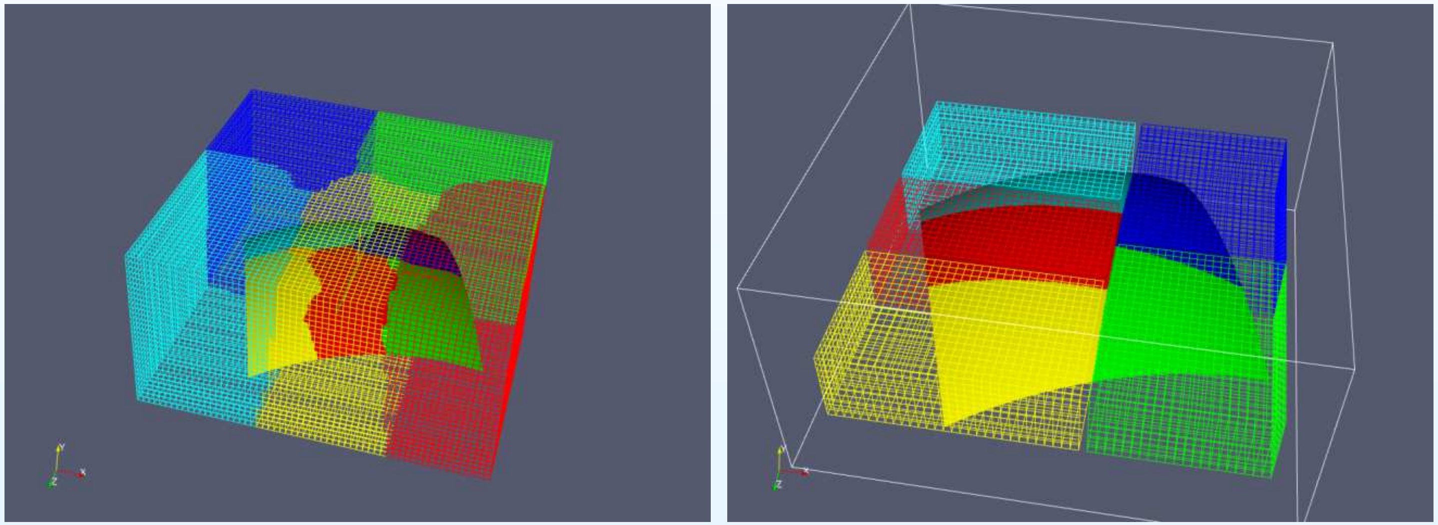
The Recursive Coordinate Bisection algorithm is a parallel algorithm for partitioning a set of geometric entities (possibly with weights). The package Zoltan provides this.



A parallel median-finding kernel is at the core of the algorithm.

# Intersecting grids

This algorithm is only applied to the geometric intersection of the meshes.

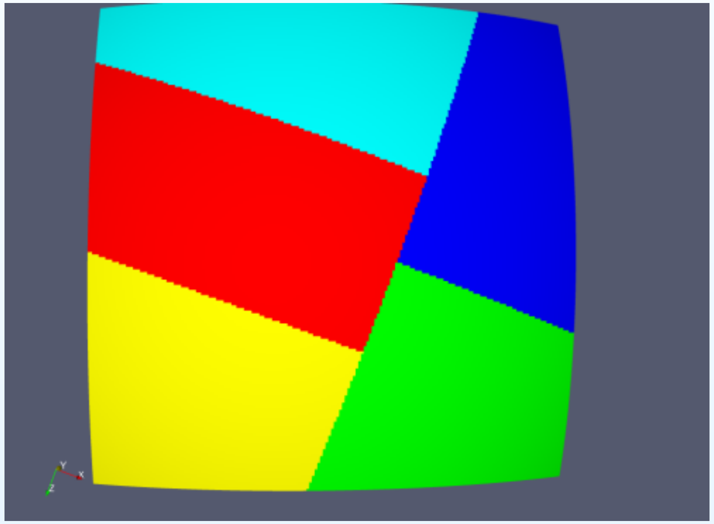


# Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.

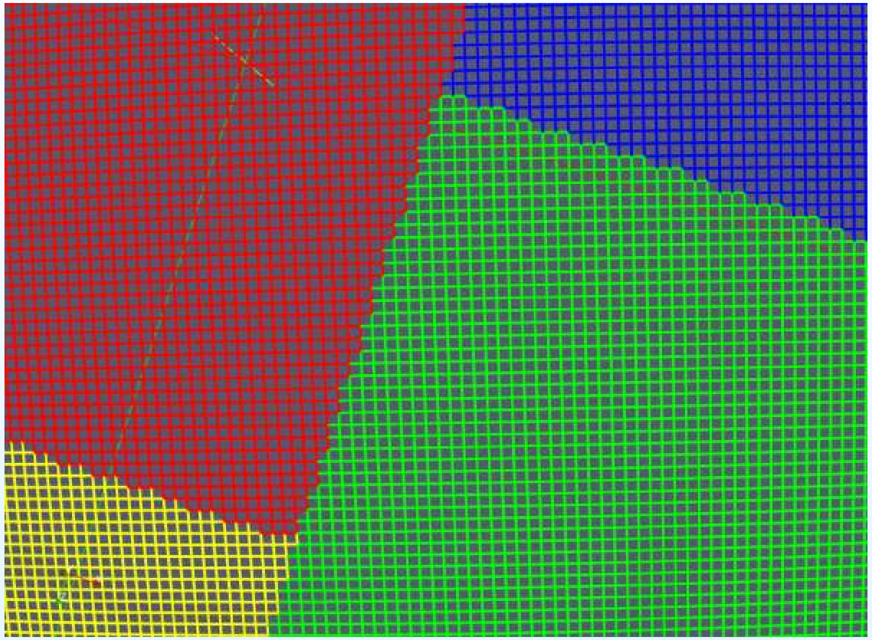
## Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.



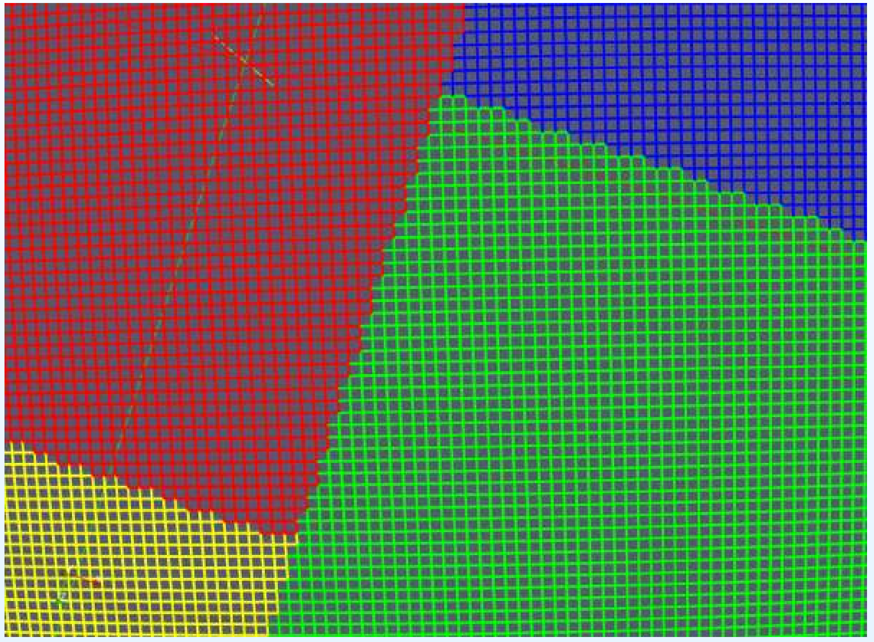
# Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.



# Non-regular decompositions

Representing the meshes in the Rendezvous decomposition is a challenge since, in general, the meshes will not have a regular decomposition in this space.



We need a representation for such decompositions.

# The rendezvous matrix application

The interpolation forms a commutative diagram

# The rendezvous matrix application

The interpolation forms a commutative diagram

$$
\begin{array}{ccc}
Src_R & \xrightarrow{\mathcal{C}} & Dst_R \\
\mathcal{A}\Big\uparrow & & \mathcal{B}\Big\uparrow \\
Src_s & \xrightarrow{\mathcal{I}} & Dst_d
\end{array}
$$

# The rendezvous matrix application

The interpolation forms a commutative diagram

$$\begin{array}{ccc} Src_R & \xrightarrow{\mathcal{C}} & Dst_R \\ \mathcal{A}\Big\uparrow & & \mathcal{B}\Big\uparrow \\ Src_s & \xrightarrow{\mathcal{I}} & Dst_d \end{array}$$

Where $\mathcal{A}$ and $\mathcal{B}$ are the mesh migration communication spec's and $\mathcal{C}$ is the local interpolation operator. The subscripts $_{s}, _{d}, _{R}$ are the source,destination and rendezvous decompositions. We have $\mathcal{I} = \mathcal{B}^{\top} \circ \mathcal{C} \circ \mathcal{A}$.

# The rendezvous matrix application

The interpolation forms a commutative diagram

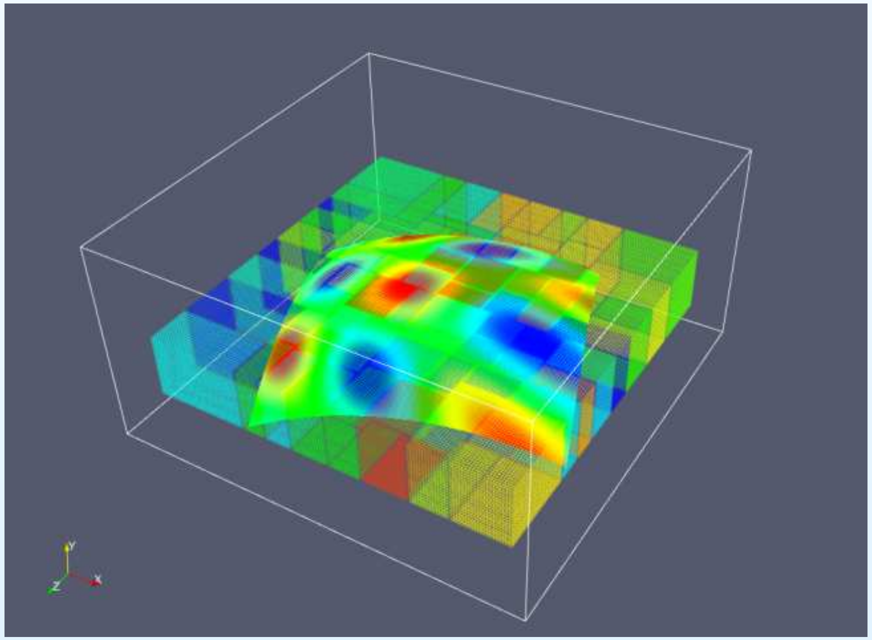
$$\begin{array}{ccc} Src_R & \xrightarrow{\mathcal{C}} & Dst_R \\ \mathcal{A}\Big\uparrow & & \mathcal{B}\Big\uparrow \\ Src_s & \xrightarrow{\mathcal{I}} & Dst_d \end{array}$$

Where $\mathcal{A}$ and $\mathcal{B}$ are the mesh migration communication spec's and $\mathcal{C}$ is the local interpolation operator. The subscripts $_{s}, _{d}, _{R}$ are the source,destination and rendezvous decompositions. We have $\mathcal{I} = \mathcal{B}^{\top} \circ \mathcal{C} \circ \mathcal{A}$.

We ship fields and results using the mesh migration comm spec's $\mathcal{A}$ and $\mathcal{B}^{\top}$.

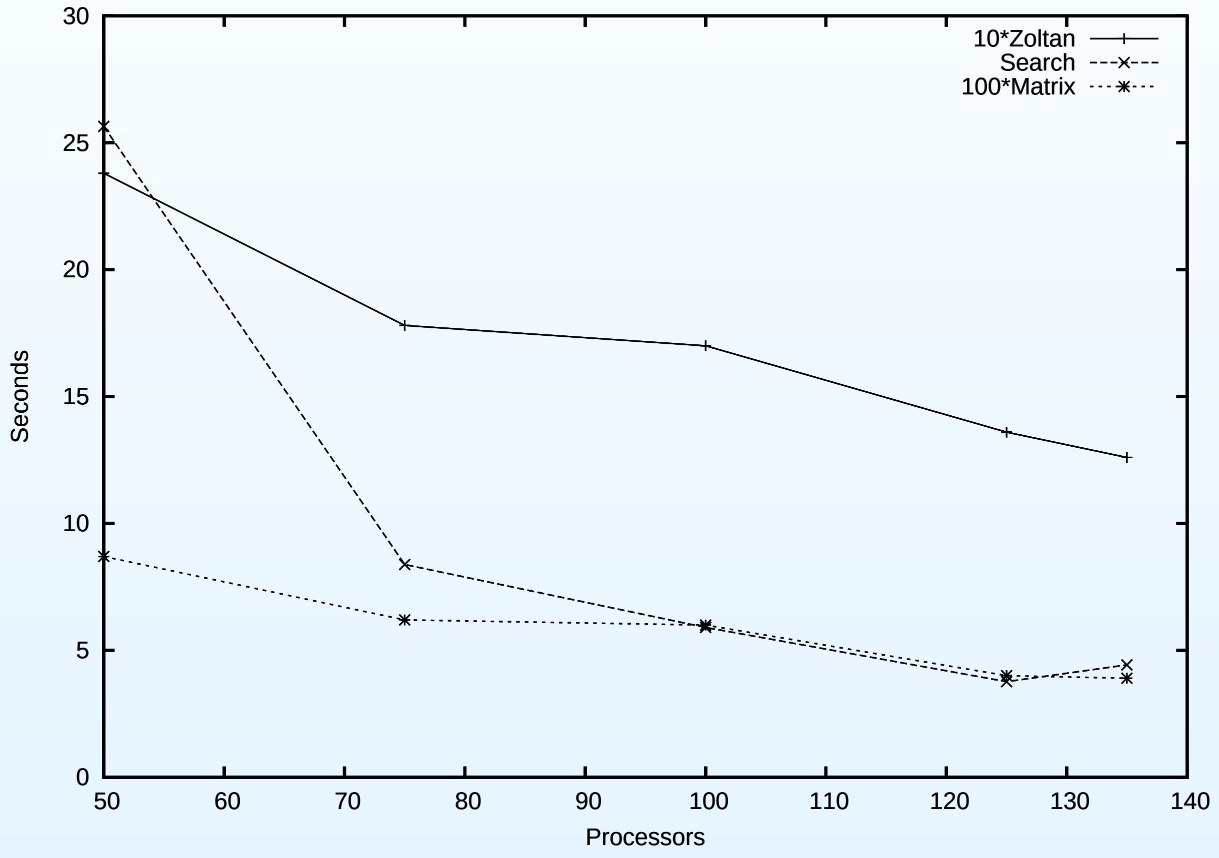
## Results

We interpolate from a 3d volume to a 2d manifold (bilinear). The volume contains 4M cells, the surface contains 1.9M cells. Only 984K source cells intersect the destination bounding box. Using UCAR's lightning cluster. 128 nodes, each with two 2.2GHz AMD Opteron processors, 4GB memory shared. 128-port Myrinet switch through single-port Myrinet PCI adaptor.



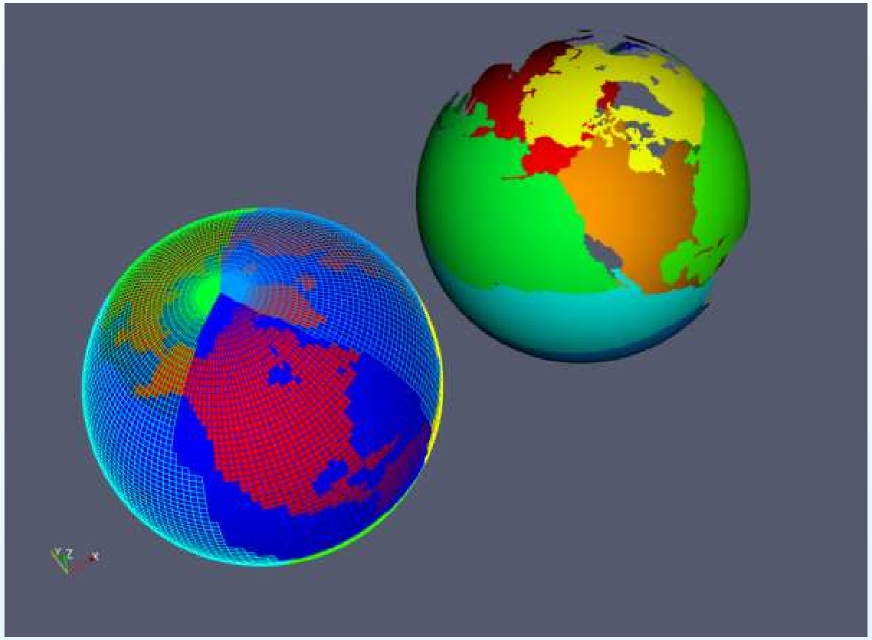
# Results, timing

## Timings:



# Example, analytic wind field

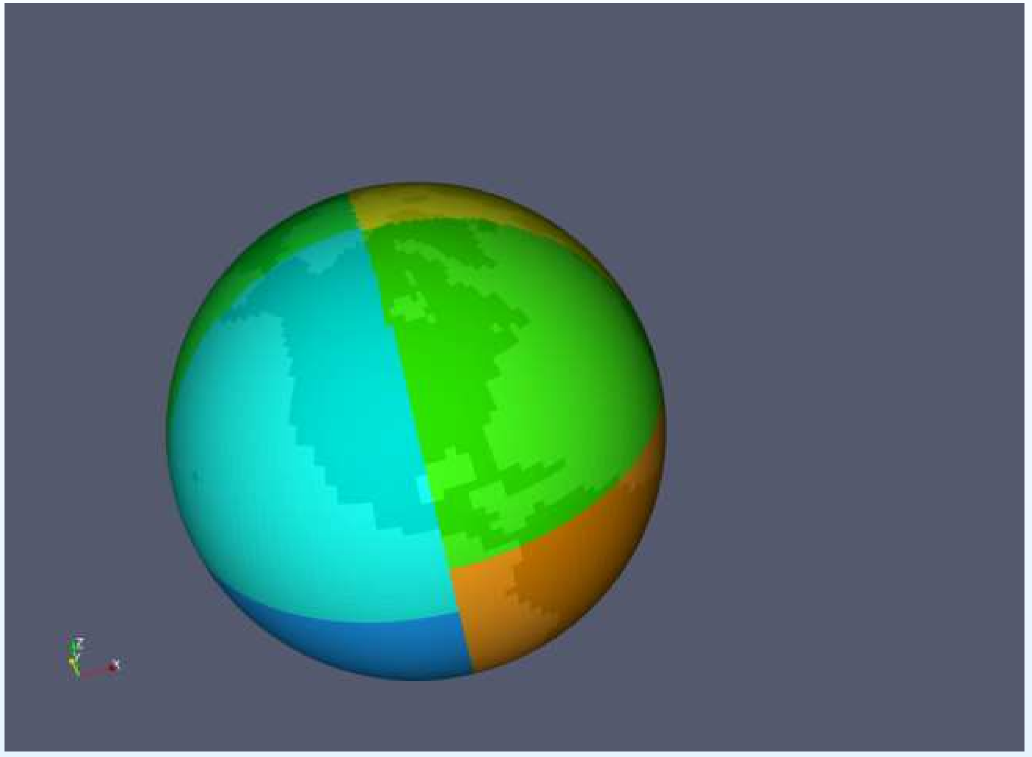
We interpolate an analytic wind field from a standard lat/lon earth grid to the POP ocean grid.



We use both bilinear and a patch-interpolation method.

# Rendezvous grid

The rendezvous grid decomposition for these meshes



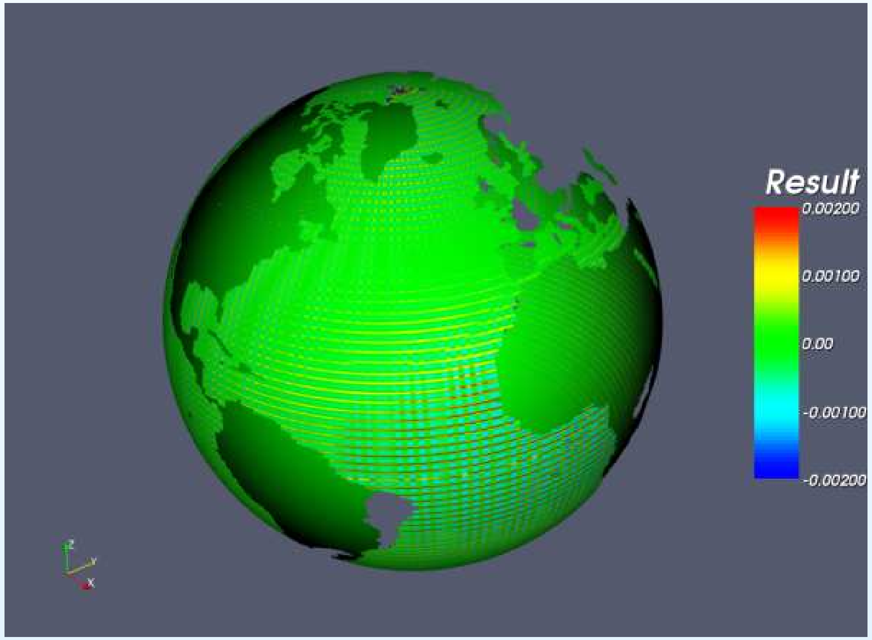
# Patch vs Bilinear gradients

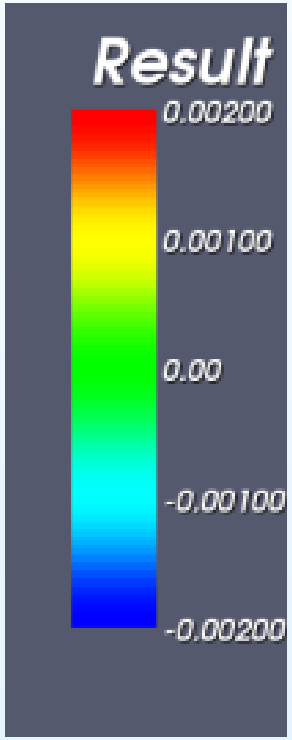
The patch method produces much more accurate derivatives, curl is shown here.

# Patch vs Bilinear gradients

The patch method produces much more accurate derivatives, curl is shown here.

Patch recovered Error

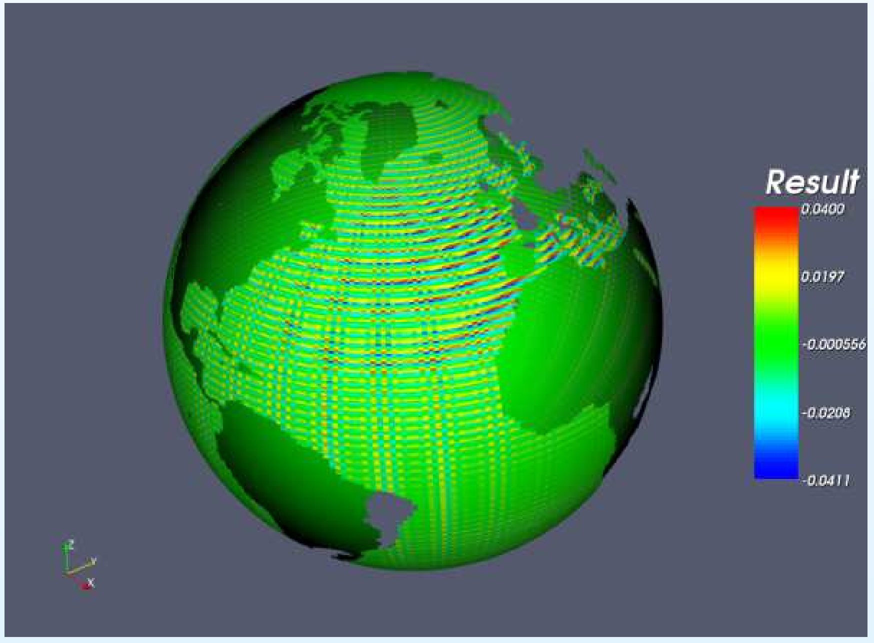


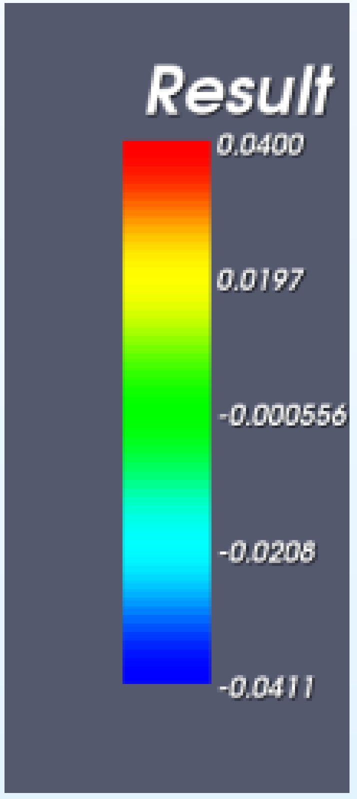



# Patch vs Bilinear gradients

The patch method produces much more accurate derivatives, curl is shown here.

Bilinear Error



# Conclusions

The patch recovery interpolation is a parallel-friendly interpolation method preserving derivatives

## Conclusions

The patch recovery interpolation is a parallel-friendly interpolation method preserving derivatives

The Rendezvous algorithm presents a straightforward and robust method to perform parallel regridding and (with the addition of a fractional area kernel) to compute the exchange grid.

## Conclusions

The patch recovery interpolation is a parallel-friendly interpolation method preserving derivatives

The Rendezvous algorithm presents a straightforward and robust method to perform parallel regridding and (with the addition of a fractional area kernel) to compute the exchange grid.

For some meshes, the matrix multiplication should be performed in rendezvous space, to combat load imbalance.

## References


- “A parallel rendezvous algorithm for interpolation between multiple grids,” Steve Plimpton, Bruce Henderickson, James Stewart. Proceedings of the 1998 ACM/IEEE conference on Supercomputing, 1998.
- “SIERRA Framework Version 3: Core Services Theory and Design,” H. Carter Edwards. Sandia National Laboratories, report SAND2002-2616, 2002.
- “Architecture of the Earth System Modeling Framework,” Hill, C., C. DeLuca, V. Balaji, M. Suarez, and A. da Silva. Computing in Science and Engineering, Volume 6, Number 1 (2004).
- ”Zoltan: Data Management Services for Parallel Dynamic Applications,” Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson and Courtenay Vaughan. Computing in Science and Engineering, Vol 4 Number 2, 2002.