Stony Brook University

Conservative and accurate data remapping for coupling climate models (WRF and CAM)

A Dissertation Presented

by

Ying Chen

to

The Graduate School

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

May 2012

Stony Brook University

The Graduate School

Ying Chen

We, the dissertation committee for the above candidate for the Doctor of Philosophy degree, hereby recommend acceptance of this dissertation.

Xiangmin (Jim) Jiao – Dissertation Advisor
Associate Professor, Department of Applied Mathematics and Statistics

Xiaolin Li – Chairperson of Defense
Professor, Department of Applied Mathematics and Statistics

Wei Zhu
Professor, Department of Applied Mathematics and Statistics

Minghua Zhang
Dean
School of Marine and Atmospheric Sciences

This dissertation is accepted by the Graduate School.

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Conservative and accurate data remapping for coupling climate models (WRF and CAM)

by

Ying Chen

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2012

The two-way coupling of WRF and CAM for regional climate modeling requires transferring data (such as wind velocities, temperature, moisture, etc.) between the grids of WRF and CAM. These grids are in general non-matching, and the data transfer should be both physical conservative and numerically accurate. Our method contains two parts: constructing the common refinement and conservative data transfer using the common refinement. The former computes the cell intersections and provides the necessary data structures for the latter. We project the grids of WRF and CAM onto a plain surface using their corresponding map projection methods (such as: Lambert conformal projection). Then common refinement method will locate the edge intersection points of two grids, and determine sub-facets to create a new mesh based on the union of the intersection points and vertices of the original two grids within the overlapping area. After obtaining the common refinement mesh, we use a weighted-residual formulation, which minimizes the L-2 norm of the error, to complete the data transfer. The L-2 minimization method satisfies physical conservation and

at the same time is as accurate as interpolation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Numerical Atmospheric Modeling is a useful approach to simulate every possible case in future climate change or weather prediction. Depending on simulation area, numerical atmospheric models can be categorized into two classifications: global climate system and regional weather model. Global climate systems take care of the whole atmosphere in a long term period, while regional weather models focus on short-term and small-scale weather phenomena. In general, we can say global climate systems are close systems, and they should satisfy fundamental physical principles, such as conservation of mass, conservation of momentum, conservation of thermodynamic energy, and the radiative transfer equation. In contrast, regional weather models don't guarantee physical conservation. Community Atmosphere Model (CAM) and Weather Research and Forecasting Model (WRF) are the most outstanding typification for global climate system and regional weather model, respectfully.

The Community Atmosphere Model (CAM with version 5) is developed by the climate community in collaboration with the National Center for Atmospheric Re-

search (NCAR). Like its predecessors, CAM is designed to be a modular and versatile model suitable for climate studies by the general scientific community. CAM5 can be run either as a stand-alone AGCM (coupled to the CLM land model) or as a component of the Community Climate System Model (CCSM). When coupled to the CCSM it interacts with fully prognostic land, sea-ice, and ocean models. It is also being employed in biogeochemical and physical-chemistry studies in experimental configurations. Meanwhile, the Weather Research and Forecasting (WRF) model is a numerical weather prediction (NWP) and atmospheric simulation system designed for both research and operational applications. The development of WRF has been a multi-agency effort to build a next-generation mesoscale forecast model and data assimilation system to advance the understanding and prediction of mesoscale weather and accelerate the transfer of research advances into operations.

Due to the imperfectness of both climate systems and weather models, the idea of coupling WRF and CAM together to overcome their weakness is quite straightforward. In theory, WRF will contribute accuracy in small weather phenomena to CAM, so that CAM will perform better in long-term climate simulation. At the same time, CAM can provide boundary condition to WRF under physical conservation, so that WRF can produce more accurate output in regional weather prediction. In order to test our intuition, we use the simulated Frontal Clouds during the ARM March 2000 as an example. The results supports our ideas very well.

However, there are many issues we need to addresses during the coupling. One of them is the data remapping issue. Since CAM and WRF are based on different resolution of grids, then to transfer data between those two grids conservatively and accurately is really a challenge.

1.1 Problem Description

1.1.1 2-D Mesh

A 2-D mesh is a discretization of the geometry of a 2-D plane. It divides a plane into many small elements. Such elements are polygons in general, including triangles and rectangles. There are three kinds of topological objects in a mesh, which are vertices, edges, and facets. Hence, each element consists of several vertices, several edges and one facet. Edges are formed by connecting vertices, while facets are surrounded by joined edges. The realization of the connection between any two vertices can be a line segment, or a curved edge which is a circular arc in our application. A 2-D mesh is then defined as collections of these vertices, edges and facets embedded in R^2.

1.1.2 Formulating the Problem

In order to fully couple CAM and WRF, state variables (wind stress, temperature, etc) and heat and water fluxes should be transferred between models periodically, and such fields should be remapped from one model mesh to another. In particular, fluxes should be remapped conservatively in order to maintain the total amount of energy and water in the whole system. Both CAM and WRF use finite volume method. CAM supports several kinds of mesh, but regular Latitude-Longitude grid is the most important one. While in WRF, the earth surface is first projected onto a plain surface by using several kinds of Map Projection methods, so the computational mesh is indeed regular quadrilateral. The location of data is crucial for data transfer. All state variables and fluxes are cell-centered in CAM, that is different

in WRF, where the wind velocities are located in the cell side. Both cell-centered and staggered data are associated at a finite number of discrete points in a mesh, referred to as control points. Together with the basis functions (or shape functions, typically piece wise linear or quadratic) associated with the nodes, these nodal values determine a piece wise polynomial interpolant over the underlying space of the mesh. Let m be the number of source control points and n be the number of target control points. Let $f = \sum_{i=1}^{m} f_i \phi_i$ denote the source function, where $f_i = f(s_i)$ andϕ_i are the data value and basis function, respectfully, where s_i stands for the ith source control point. Similarly, let $g = \sum_{i=1}^{n} g_i \psi_i$ denote the target function, where $g_i = g(t_i)$ and ψ_i are related to the ith target control point t_i. Therefore, the data remapping problem is just to find out the value of g_i , given f_i , ϕ_i and ψ_i . It is obvious that $g - f$ determines the error between the data before and after remapping, and whether $\int (g - f)dx = 0$ tells whether the data remapping is conservative or not.

1.1.3 Coordinate System

In order to transfer data between CAM and WRF, we need to compute the intersection between CAM's mesh and WRF's mesh in order to set up a geometric relationship between those two input meshes. With the intersection, then we can operate interpolation method or remapping method to transfer data across two domains. Therefore, we need to clarify the geomeric properties for CAM's mesh and WRF's mesh. CAM's mesh consists with regular rectangles in spherical coordinate, and the edges of those rectangles are along with either latitude or longitude. While WRF's domain is simulated by a rectangular mesh in Carteisan plain with certain

kinds of map projection method, like Lambert projection, Mercator projection and so on.

There are two ways to compute the intersections, the first way is based on spherical coordinate which is used by SCRIP package (P. Jones, 1999), and the other way is within the Cartesian coordinate. For the first kind, there are several disadvantages, and the most essential one is due to edge linearly approximation. That is because although CAM's mesh is a rectangular mesh, WRF's elements are sectors in spherical coordinate, and the edges of WRF's elements don't need to be along with latitude ,longitude or any great circles, which makes it difficult to accurately approximate curved edges of WRF's mesh. Then WRF's edges will be linearly approximated when computing the edge-to-edge intersection, which may introduce big errors to the data transfer. In Cartesian coordinate, in contrast, WRF's mesh are regular rectanglular mesh, and CAM's elements are sectors whose information are clearly defined by certain map projection method, so that CAM's curved edges can be precisely fixed. Then we can accurately compute the intersection of two meshes based on sector-to-rectangle intersection and arc-to-segment intersection.

1.2 Software Architecture

Our data remapping package will be integrated into CCSM system, as a module between CAM and WRF. CCSM is referred to Community Climate System Model, which is a Global Climate Model developed by the University Corporation for Atmospheric Research (UCAR). There are several sub-models in CCSM system: atmosphere(CAM), land surface(CLM), ocean(POP), sea ice(CICE) and land-ice component (GLC). In our project, we will insert WRF into CCSM framework as

shown in Figure 1.1. And our remapping package will also added into CCSM, as a bridge between CAM and WRF, as shown in Figure 1.2 .

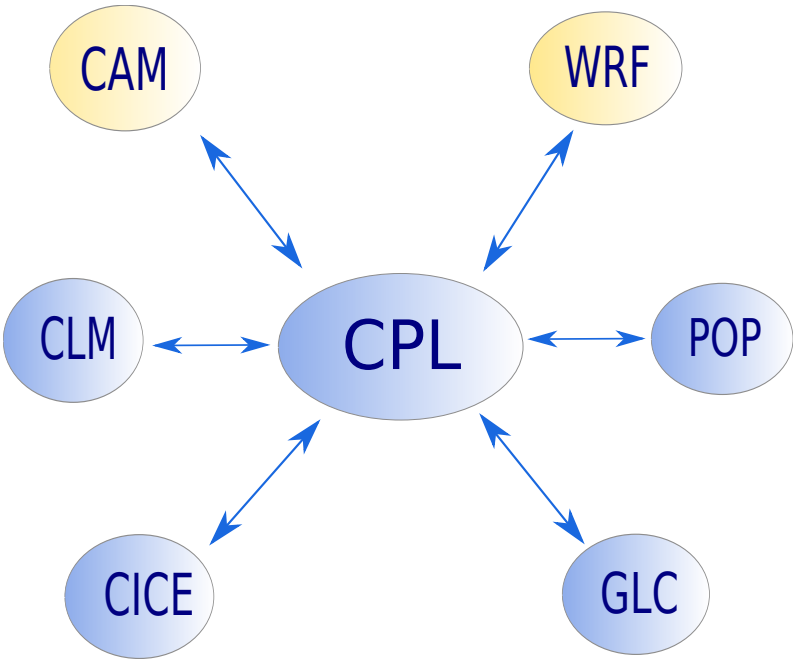

Figure 1.1: Software architecture

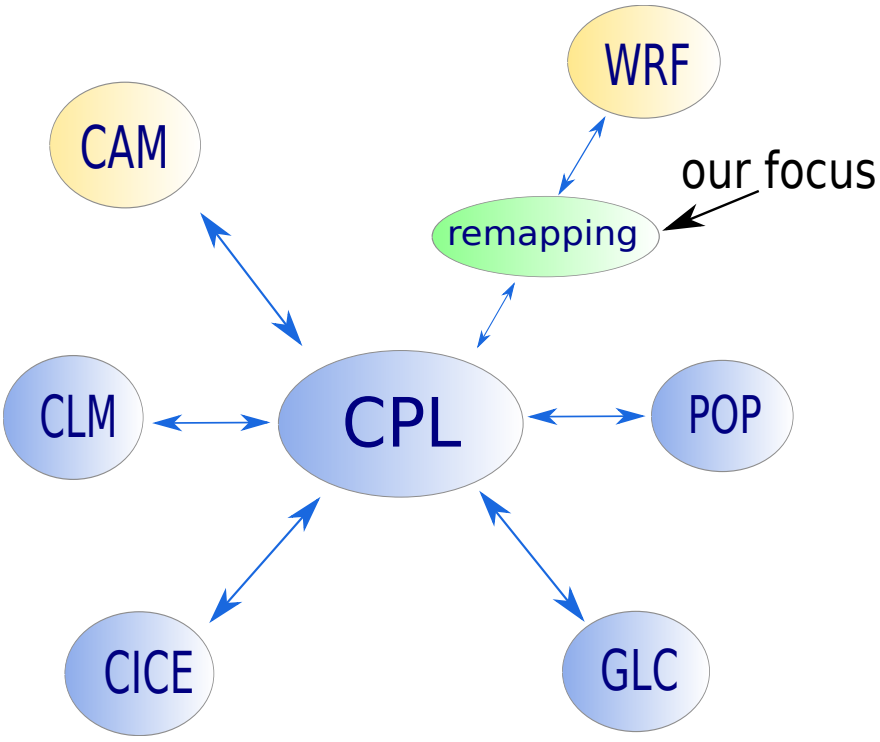

Figure 1.2: Software architecture with remapping package

1.3 Existing Methods and Their Limitations

Data transfer method can be divided into two steps in general:

1. Compute the intersection between source mesh and target mesh.

2. Transfer data based on the intersection by using certain numerical methods.

There are vast amount of existing methods for data transfer, but none of them works well. We will describe some most used existing methods in the following.

1.3.1 Intersection Algorithm

1.3.1.1 SCRIP Package

SCRIP is an abbreviation of Spherical Coordinate Remapping and Interpolation Package. This is a software package which computes addresses and weights for remapping and interpolating fields between grids in spherical coordinates. It was written originally for remapping fields to other grids in a coupled climate model. SCRIP is widely used in Atmospheric community, and its functionality has been included in the Earth System Modeling Framework and the European PRISM framework.

Since SCRIP is designed for global models, then it is not suitable for two meshes with non-matching boundaries. In addition, SCRIP computes the intersection in spherical coordinate. Therefore, the edge-to-edge intersections would be accurate for edges along latitude or longitude. If the edges are not along latitude or longitude, SCRIP just linearly approximates the edges in spherical coordinate and computes the edge-to-edge intersection. Moreover, SCRIP is not able to get accurate edge-to-edge intersection for curved edges. Furthermore, SCRIP approximates surface integral with line integral by using divergence theorem. Another issue of SCRIP is the treatment of pole. Pole is a line in latitude-longitude grid, although it is physically a point. Therefore, if the pole is not treated well, then big computational errors would be brought in.

Currently, in SCRIP, there are four numerical options which will be described in the section of numerical methods:

1. Point-wise bilinear interpolation

2. Point-wise bi-cubic interpolation

3. Point-wise distance-weighted averaging

4. Area-based conservative remapping: First- and second-order conservative remapping as described in (P. Jones, 1999).

1.3.1.2 Construct Common Refinement Mesh

Our data transfer method uses a data structure called the common refinement of two non-matching meshes, which is described in (X. Jiao and M. Heath, 2004). A common refinement of two meshes is a mesh composed of elements that subdivide the elements of both input meshes simultaneously, or, simply put, the intersections of the elements of the input meshes. It provides a one-to-one correspondence between a point in source mesh and a unique point in target mesh. In the common refinement mesh, we refer to the vertices, edges and facets as subvertices, subedges and subfacets, respectfully. For every sub-object of the common refinement, it has two corresponding parent objects, one of which in the source mesh and the other in the target mesh. A subvertex is either a vertex of the original two meshes, or an edge intersection. The subedges of the common refinement are the intervals of edges of original two meshes between the subvertices. A sub-facet is a part of corresponding facets of original two meshes. A common refinement mesh R is minimal if all sub-objects have different pair of parent objects. With common refinement, the geometric relationship between two input meshes would be very clear to us. Moreover, it enables accurate integration of functions that depend on the shape functions of the two meshes. For these reasons, the common refinement is important for our method both theoretically and practically.

However, the original common refinement method cannot handle two input meshes with curved edges or with non-matching boundaries. Therefore, we need to enhanced the original common refinement method to an extended version in our project. Before we introduce the extended version, we will briefly describe the original method in the following section. In general, the original common refinement method consists with several sub-methods:

- Looping through source mesh and target mesh.
- Face-to-face intersection
  - Project vertices of source element onto the target element.
  - Edge-to-edge intersection
- Triangulation

We will explain the above steps in the following.

Looping through the source mesh and target mesh The common refinement algorithm is accomplished by iterating element by element for both source mesh and target mesh, which can be described with following steps:

1. Choose a source element, and locate the potential target element that intersects the current source element. Add this pair to the list of seed elements.
2. For the pair of seed elements found in the last step, operate face-to-face intersection.
3. Search the neighborhood of current pair of seed elements, look for other pairs of potential seed elements and insert them into the list of seed elements.

4. Get the next pair of seed elements from the list of seed elements, and repeat from step 2 until all pairs of seed elements of that list have been operated with face-to-face intersection.

After we have the first pair of seed elements, we will operate face-to-face intersection for current pair of seed elements. In addition we create a local list of target elements that could potentially intersect with current source seed element. Target elements are added to the local list as more subvertices on the source face are found. There are two ways of finding subvertices of common-refinement: point (or vertex) projection and edge-to-edge intersection. To begin with, all the target elements that the vertices of current source seed element projected onto are added to the local list. These target elements are helpful in finding more target elements that could intersect with the current source element. If a vertex of the source element were projected onto a target vertex then all the target elements containing that target vertex are added to the local list of potential candidates. On the other hand, if a source vertex were projected onto a target edge, then both the elements (one element if it is a boundary edge) incident on that edge are potential seed target elements. If a source vertex falls inside a target element, then only that target element is added to the local list because a source vertex can just lie inside one target element at a time. Besides point (or vertex) projection, we also use edge-to-edge intersection to find subvertices of common refinement. Using the relative position of projection of source vertices found in last step, all the potential pairs of source edges and target edges for the current pair of source seed element and target seed element are marked. Then we operate edge-to-edge intersection on the pairs of edges and we never apply the edge-to-edge intersection to a pair of source and target edge

more than once in order to avoid duplicate computations. Whenever a subvertex is found we add more target elements to the list depending on whether the target parent is right at a vertex , at an edge or within an element. In this way we find all the subvertices lying on the current source element. Then we can merge the local list of potential target elements to the global list of seed elements, which has the information of both source seed elements and target seed elements.

In order to optimize the efficiency of common refinement method, we don't check for the intersection for all the possible pairs of source elements and target elements. Therefore, the list of seed elements is very important for a robust and accurate algorithm. If we fail to identify some target elements that have intersections with some source elements, then this would leads to big errors in the output. That is why the looping process plays an very important role in common-refinement method.

Face-to-face intersection In this step we traverse both the source mesh S and target mesh T element by element simultaneously. And for every pair of source element and target element, we operate face-to-face intersection, which is based on vertex projection and edge-to-edge intersection. It is applied to a pair of source and target elements while trying to find the common subvertices hosted by them. The vertices of the source element are projected onto the target element first. After finding the projections of all the source vertices, we find the projection of all the target vertices onto the source element, and mark all pairs of source and target edges that could potentially intersect. Then with the relative positions (i.e, natural coordinate) of the source vertices with respect to the target element, we can operate edge-to-edge intersection for all the pairs of source and target edges which are

marked above.

Finding the images of source vertices on target element $Elem_T$ After we locate a pair of source element $Elem_S$ and target element $Elem_T$, we begin to find the one-to-one correspondence between source vertices and points of target element $Elem_T$, and also the images of target vertices within source element $Elem_S$. This step can be referred as point projection, which is very important in common refinement method because the projections are subvertices of common refinement mesh and their pre-images are parent vertices, and we can determine whether a source edge has intersections with a target edge based on the point or vertex projection. Generally speaking, the original common refinement algorithm is suitable to any pair of surface meshes with the same boundaries. In our project, both CAM's mesh and WRF's mesh are discretized on the same Cartesian coordinate plane, so the projection of vertices is quite straightforward based on x and y coordinates. In order to improve the robustness and efficiency of point projection, we bring in a local coordinate system: boundary encoded natural coordinate system. This natural coordinate describes the location of a point compared to a 2D facet. If the facet is a triangle, then natural coordinate is just the barycentric coordinate. For a rectangular facet, we will parametrize the point along a pair of neighborhood edges of the rectangle. The natural coordinate of a vertex with respect to an element will show us whether the projection is inside the element, on the edge or right at the vertex of a specific element.

Locating subvertices produced by edge intersections This step locates the subvertices for the common-refinement mesh and also finds out their source and

target parents. Every vertex of the new common refinement mesh has at least one pre-image in source mesh and one in target mesh, respectfully. A vertex of the common refinement mesh is either a vertex of one of the two input meshes or the intersection of a pair of source and target edges, and we call a vertex of common refinement mesh as a subvertex.

This edge-to-edge sub-algorithm is to compute the intersection of a source edge $Edge_S \in S$ with a target edge $Edge_T \in T$, and reports whether the intersection is in the interior of edges, at a vertex or the empty set (i.e, the two edges don't have any intersections). The following explains this sub-algorithm when $Edge_S$ and $Edge_T$ are all line segments on the 2-D Cartesian plane. Let the source edge be $Edge_S = V_{S_0}V_{S_1}$ and the target edge be $Edge_T = V_{T_0}V_{T_1}$. We parametrize $Edge_S$ by $V_{S_0} + \alpha(V_{S_1} - V_{S_0})$, and parametrize $Edge_T$ by $V_{T_0} + \beta(V_{T_1} - V_{T_0})$. Then we have

$$V_{S_0} + \alpha(V_{S_1} - V_{S_0}) = V_{T_0} + \beta(V_{T_1} - V_{T_0}) \,, \tag{1.3.1}$$

which is a linear system with two equations in x direction and y direction respectfully. The intersection is in the interior of $Edge_S$ if $\alpha \in (0,1)$, at a vertex if $\alpha = 0$ or 1, in the exterior otherwise; similarly for β. A solution corresponds to an actual edge intersection if $\alpha \in [0,1]$ and $\beta \in [0,1]$ are both satisfied simultaneously.

Making sub-facets by triangulation After operating face-to-face intersection between a source element $Elem_S$ and a target element $Elem_T$, we get the intersection zone with a list of subvertices and their source and target parent edges. For two elements whose edges are all line segments, this intersection zone could be a polygon, a line segment, a single point or empty set. Here we only consider how to

treat the polygon, since other possible cases are relatively easier to handle. During this step, we are going to make subfacets by triangulating the intersection polygon. The sorted lists of subvertices and their parent information are utilized to form the surfacets. The source and target parent edges provide us with constraints we need to follow while making the surfacets. For a source-target element pair we find out the common subvertices between them and try to form surfacets from those common subvertices. If two subvertices (from the list of common subvertices between a source-target element pair) are on the same source edge then they should be on the same edge in the surfacet of the new common refinement mesh too. These constraints are applied to all the common subvertices of that source-target element pair. In this way starting from the first common subvertex of a source-target element pair, we form a surfacet following the constraints by traversing the subvertices in counter-clockwise order.

1.3.2 Numerical Methods

1.3.2.1 Point-wise Interpolation and Extrapolation

Interpolation (and sometimes extrapolation) is probably the most popular method for transferring data between meshes, and it is even sometimes used as a synonym for data transfer. Interpolation, or point-wise interpolation, refers to the process of determining and evaluating an interpolant to obtain values for some query points from given values at a finite number of control points, where the interpolant must be evaluated to the given values at the control points. Extrapolation is required when a query point falls outside the range of the source mesh. Point-wise interpolation can be categorized into two types. One type uses the source function as

the interpolant, and is sometimes referred to as consistent interpolation or inverse isoparametric mapping. Let $t_j = \sum_i \phi_i(t) s_i$ be a target control point, where the ϕ_i are the shape functions associated with the source mesh. Consistent interpolation assigns the value at t to be $g(t_j) = \sum_i \phi_i(t) f(s_i)$. Consistent interpolation includes bilinear interpolation and bi-cubic interpolation. The other type constructs an interpolant using high order basis functions, and will be referred to as alternative interpolation. Some popular alternative interpolation are thin-plate spline, infinite-plate splines, finite-plate splines, and multiquadric-biharmonic. The cubic spline, obtained by enforcing continuity of second derivatives, is another widely used interpolant in geometric modeling and other scientific applications. A major drawback of interpolation and extrapolation schemes is that they are not strictly conservative. When the shape functions are linear, the error in the conservation measure is of second order for consistent interpolation, but is in general non-zero and is sensitive to the second derivative of the input function. Alternative interpolation can deliver higher order accuracy but is still non-conservative. If the second derivative is large or does not exist, then the loss of information can be arbitrarily large, unless the extremes coincide with the nodes. For repeated transfers, accuracy and conservation measures may also degrade quickly over time.

1.3.2.2 Mortar Elements Methods

Mortar element methods are general techniques for projecting data at interfaces between two or more sub-domains in a non-conforming domain decomposition. In these methods, the discretized interfaces between sub domains are divided into the mortar side and the non-mortar side. Each side forms a discretization of the interfaces, and data are projected from the non-mortar side to the mortar side. The

formulations of mortar element methods are also based on a weighted residual, where the weight functions are usually chosen from the space spanned by the basis functions of the mortar side. Most investigations of mortar element methods have focused on analysis of its semi-discrete form in the context of non-conforming domain decompositions.

1.3.2.3 Specialized Methods

Point-wise interpolation and weighted residual are generic methods that are applicable to a wide range of problems. There are also methods designed for specific applications that do not fall directly into the above categories but frequently are variants or combination of them. A particular example of specialized methods is load and motion transfer in fluid–solid interaction for aero-elasticity problems. In these problems, traction from the fluid must be converted into nodal loads (equivalent to the load vectors in weighted residual methods) on the solid. Displacements of the solid are then transferred back to the fluid, typically using consistent interpolation. By taking advantage of knowledge about their specific applications, specialized methods can be more efficient than generic methods. However, they frequently must also face the same discretization issues as those of weighted residual methods and can also benefit from common-refinement-based discretization.

1.3.2.4 Area-weighted Averaging

For cell-centered data, area-weighted averaging is the simplest form of conservative data transfer. In this method, the value of a target element is the weighted average of the values of the source elements in contact, where the weights are the areas of the intersections between the source and target elements. For the discretiza-

tion of area-weighted averaging method, integrations are usually evaluated over the intersections of the source and target elements, or sometimes converted to boundary integration. Area-weighted method satisfies physical conservation strictly and it preserves positivity, but is not very accurate in some aspect, because the source basis functions and target basis functions are all constant.

1.3.2.5 2nd Order Conservative Method

This method is described in (P. Jones, 1999). This method is developed based on area-weighted method by utilizing information of gradient of source function. In practical use, the gradient is approximated as linear functions. Therefore, the source basis functions are linear while the target basis functions remains constant. 2nd order conservative method strictly satisfies physical conservation and it preserves positivity, but it cannot achieve high order accuracy.

Because of the disadvantages of existing data transfer methods, we want to implement a new method based on the extended common refinement method.

1.4 Contributions

There are most three important contributions of this thesis:

- Transfer data between meshes with curved edges.
- Extend the original common-refinement method for meshes with non-matching boundaries.
- Develop a monotonic numerical method, in order to preserve the positiveness of target function.

For the original common-refinement method which is described in (Jiao and Heath, 2004), there are face-to-face intersection and edge-to-edge intersection sub-algorithm. The face-to-face intersection is suitable for a pair of elements which might be triangles or rectangles, based on the edge-to-edge intersection that computes the intersection between two line segments. However, in the model coupling of CAM and WRF, CAM's elements becomes sectors after map projection method, for which the original face-to-face intersection is not able to handle. Therefore we implement a new intersection algorithm of circular arcs with line segments to replace the original edge-to-edge intersection. In this way, the original face-to-face intersection is extended to sector-to-rectangle intersection, which is essential for conservative and accurate data remapping between CAM and WRF.

In addition, CAM's boundary is not consistent with WRF's boundary, because CAM is a global model and WRF is a regional model. In practical implementation, CAM's domain is always bigger than WRF's domain. Therefore, it would be very tricky to transfer data from WRF to CAM, because it would be highly possible to cause big jumpings and inconsistencies around the WRF's boundary. In order to remap the data smoothly from WRF to CAM, we implement a specific upscaling method to handle this issue. Such method would be suitable for any two meshes with non-matching boundaries, which would bring more convenience and more possibilities for different model couplings beside our current project.

Moreover, we have also tried to preserve the positivity of target function, because it is very important for climate models. Although L_2 method works well to preserve positivity of target function most of time, but it does not guarantee positivity. We have considered several numerical methods such as monotonic cubic spline interpolation (Wolberg, 1999), area weighted method (Jones, 1999) and 2nd order

conservative method (Jones, 1999). Because physical conservation is also significant for climate models, we choose the later two methods and integrate them to our framework, so that users is able to use corresponding numerical methods for different purposes.

Chapter 2

Extended Common Refinement Method

In order to transfer data accurately and conservatively between CAM and WRF, we introduce L_2 norm error minimization method as the key algorithm for data remapping. L_2 norm error minimization method is to minimize the $E = \sqrt{(g-f)^2}$ which is the error in L_2 norm. This method belongs to the family of Rayleigh–Ritz–Galerkin methods. The value E is also known as energy of mesh, so L_2 norm error minimization method is also consistent with minimum energy principle. For convenient purpose, we will call L_2 norm error minimization method as L_2 minimization method or L_2 method in the following. In order to use L_2 minimization method, it needs to be discretized over the remapping area. In addition, in order to minimize the numerical errors, the points of source mesh and target mesh have to be corespondent to each other well. Therefore, we use common-refinement method to prepare data structure for L_2 minimization method. Sobolev minimization method uses Sobolev norm which is a general form of L_2 norm. When the source function changes abruptly

(i.e, there are big gaps among data values in source grid), then we need to replace L_2 minimization method with Sobolev minimization method.

The whole process of common-refinement based remapping method can be divided into two phases, which will be explained in the following section:

(i). Based on the input two parent meshes, construct a common-refinement mesh.

(ii). Using L_2 minimization or Sobolev minimization to operate data transfer.

In this chapter, we will describe the extended common refinement method in details.

2.1 Algorithm Overview

Common-refinement based L_2 minimization method provides a general ideas for accurate and conservative data remapping between two meshes. However there are still many issues that need to be solved in real implementation, such as nonlinear issue and boundary issue. Common-refinement method is suitable for almost all pair of surface meshes whose edges are line segments, so how to construct common-refinement accurately when one parent mesh contains edges that are now line segments would be a big challenge. At the same time, original face-to-face intersection algorithm also needs to be extended. In addition, the original common-refinement method only handles pairs of meshes having the same boundaries, therefore we need to figure out how to smooth the results and avoid big jumpings along the boundary if two parent meshes have non-matching boundaries. Moreover, the iteration method through the parent meshes needs to be modified based on the type of those two meshes, in order to achieve the most efficiency.

In our project, CAM's mesh is a regular rectangular mesh in spherical coordinate, while WRF's mesh is a regular rectangular mesh in Cartesian coordinate. After project CAM's mesh onto Cartesian plane with Lambert projection, CAM's elements become sectors whose center is the cone center of Lambert projection, but origin is different from the cone center. Since both CAM's mesh and WRF's mesh are regular meshes, then we optimize the face by face (or element by element) iteration method. Additionally, CAM's mesh and WRF's mesh have non-matching boundaries, because one's elements are sectors and the others are rectangles. In practical use, CAM's domain is always larger than WRF's domain. Therefore, it is easy to transfer data from CAM to WRF which is referred as downscaling, but very tricky from WRF to CAM which is referred as upscaling. For this issue, we introduce a new upscaling algorithm to make it possible for smooth remapping. Furthermore, we implement an algorithm to handle the face-to-face intersection for a pair of a sector and a rectangle, based on edge-to-edge intersection algorithm between circular arcs and line segments. And for the edge-to-edge intersection, we utilize the polar coordinate for sectors for which the origin is the cone center, therefore we have to move the origin of WRF's Cartesian coordinate system to make the two coordinate systems consistent. Besides, the triangularization process also needs to be considered carefully. After all, the area integrals is computed within Cartesian coordinate, so we need to transform the area to that of spherical coordinate which would be the closest to real area on the Earth surface. In the following sections, we will explain the details of implementation of common-refinement based L_2 minimization method, and provides the resolutions for the above questions.

2.2 Main Process of Our Method in Implementation

Our method contains two phrases, the first one is to construct the new common-refinement mesh, and the second one is to remap data by using L_2 minimization method. And the numerical methods used in the second phrases can be replaced by many existing numerical methods, such as area-weighted method (P.Jones, 1999) , 2nd order monotonic method (P. Jones, 1999) and interpolation methods. The whole process includes several steps which are shown in the following figure.

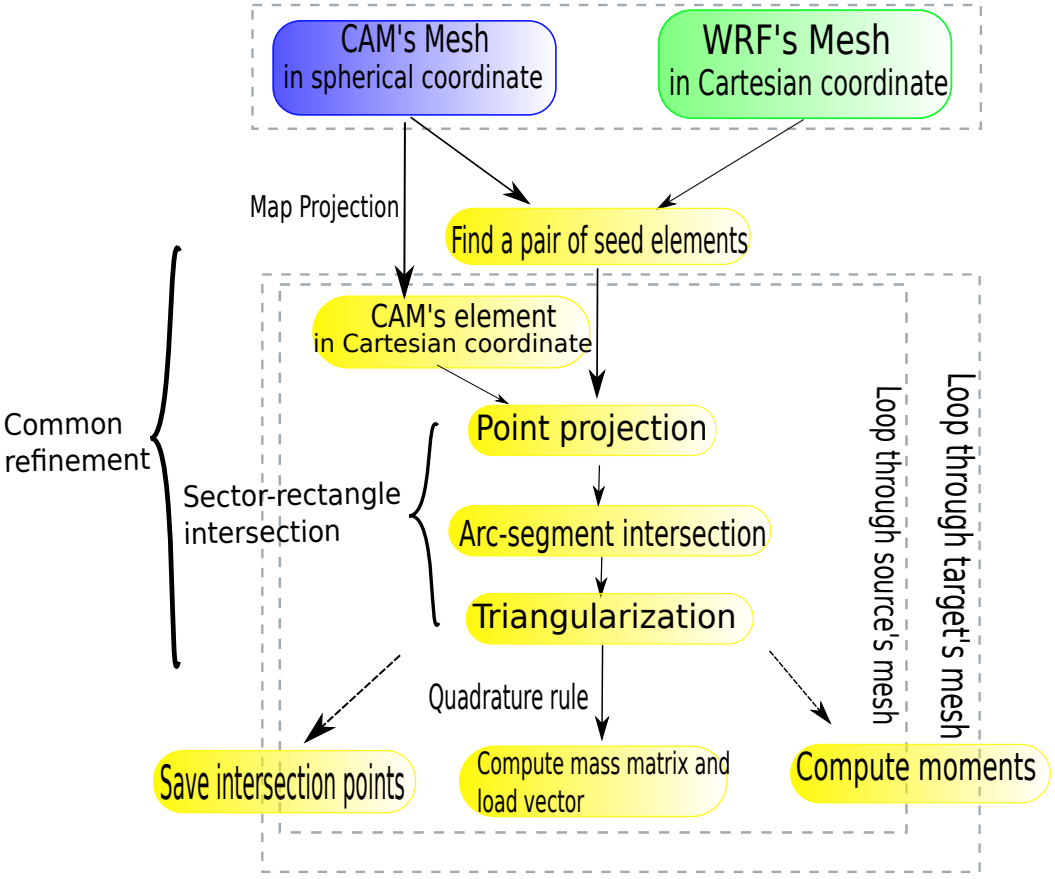

Figure 2.1: Whole process of common-refinement based L_2 minimization method

- First, input CAM's mesh and WRF's mesh. Cam's mesh is in spherical coordinate, while WRF's mesh is in Cartesian coordinate.
- Second, find a pair of a CAM's element and a WRF's element that could potentially intersect each other. This pair of elements will be set as seed elements.
- Third, compute the intersection between seed elements that are found in the second step.
  - Transform CAM's element from spherical coordinate to Cartesian coordinate.
  - Find the projection of vertices of one element to the other.
  - Operate edge-to-edge intersection.
- Fourth, triangularize the intersection zone, and compute the entries of mass matrix and load matrix within each small triangles.
- Fifth, search the neighborhood of seed elements to find next pair of elements having potential intersections. And repeat from the third step to fifth step until all the CAM's elements and WRF's elements have been touched.

2.3 Find a Pair of Seed Elements

Before the main process of common-refinement, we need to find the first pair of seed elements. There are many possible ways to locate the seed pair. In our method, we implement a 'fix-loop' method depending on the properties of CAM's mesh and

WRF's mesh. Because a CAM's element is much bigger than a WRF's element, and number of CAM's elements is much smaller than that of WRF, then we choose two central elements of CAM and WRF as starting elements. If the pair of starting elements doesn't have any intersections, we fix up WRF's starting element, and loop through the neighborhood elements of CAM's starting element until we locate a CAM's element that has intersections with WRF's starting element.

2.4 Resolution of Different Coordinates

After get a pair of seed elements, we need to consider the issue of different coordinate systems used by CAM and WRF. Because CAM is a global climate model, the Earth surface is discretized along latitude and longitude notated as which would be convenient to simulate and employ numerical methods. Contrarily, WRF is a regional weather model. Since for a small domain, the Earth surface is more like a plane instead of a sphere, then WRF model simulates a small fraction of the Earth surface within Cartesian coordinate plane by using certain types of map projection methods. Among numbers of map projection method, Lambert conformal projection is most suitable for North America, so it is widely used in WRF's model when simulating the weather forecast for the America. Therefore our project focuses on Lambert projection.

Map projection methods is a set of methods that represent the surface of a sphere onto a plane, i.e, map projection methods are used to create maps. There is no limit to the number of possible map projection methods. Every map projection method will deform the earth surface in some fashion. Therefore different map projection methods are used to preserve different kinds of properties depending on the purpose

of the map. There are map projection methods being used in WRF model, including Lambert conformal projection, Mercator projection, polar stereographic projection and cylindrical equidistant projection. Our method will focus on Lambert conformal projection, which is the most suitable for North America.

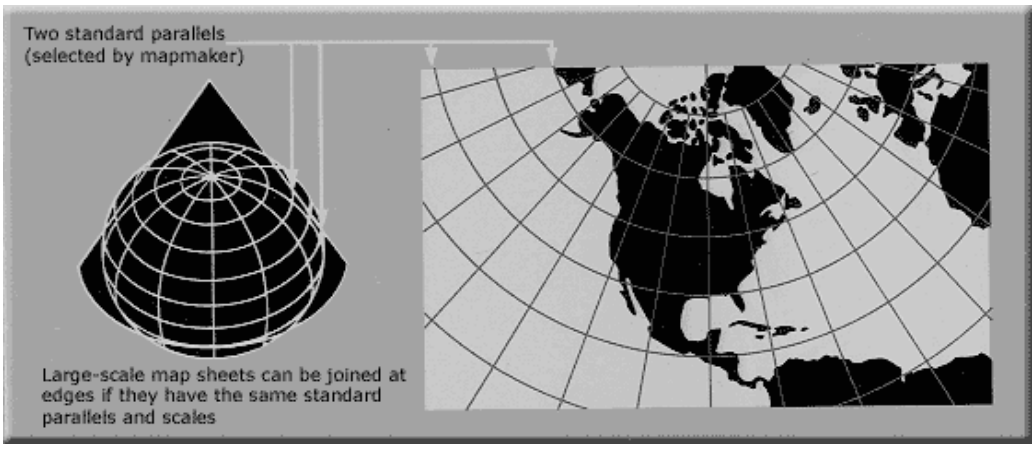

Figure 2.2: Lambert conformal conic projection

Lambert conformal projection The Lambert conformal conic map projection is developed by Johann Heinrich Lambert, who was an 18th century Swiss mathematician, physicist, philosopher, and astronomer. Lambert conformal projection is suitable for Mid-latitude area. Substantially, this projection superposes a cone over the Earth, and this cone will secant the globe at two reference latitudes. This projection is conformal, in other words, it preserves the angle and shape. There is no distortion along two reference parallels, but distortion increases further from the chosen parallels. This projection has several nice features. Besides preserving the angle and shape, a straight line on this projection approximates a great-circle route between endpoints. That is why this projection is widely used in the field of map-drawing.

With Lambert conformal projection, CAM's mesh will be projected onto Cartesian plane. CAM's mesh is rectangular in spherical coordinate, but its rectangular elements will become sectors after Lambert projection. Since CAM's elements are sectors and WRF' elements are rectangles, in order to compute the face-to-face intersection we further transform the CAM's mesh into polar coordinates based on the already know Cartesian coordinates. The origin of the polar coordinate system is the cone center of Lambert projection, while the origin of the Cartesian coordinate system produced by Lambert projection is defined by input parameters. Therefore, we have to move the origin of WRF's Cartesian plane to the cone center also.

In total, there are three four kinds of different coordinate systems in our algorithm:

(a) spherical coordinate, notated as (lat, lon)

(b) Cartesian coordinate produced by Lambert projection, notated as (x_L, y_L)

(c) Cartesian coordinate with the origin of cone center, notated as (x_m, y_m)

(d) polar coordinate with the origin of cone center, notated as (r, θ)

The formula to transfer (lat, lon) to (x_L, y_L) is as follows:

$$x_L = \rho \cdot sin\theta \tag{2.4.1}$$

$$y_L = \rho_0 - \rho \cdot cos\theta \tag{2.4.2}$$

where

$$\rho = RF/tan^n(\pi/4 + lat/2) \tag{2.4.3}$$

$$\theta = n(lon - lon_0) \tag{2.4.4}$$

$$\rho_0 = RF/tan^n(\pi/4 + lat_0/2) \tag{2.4.5}$$

$$F = cos(lat_1)tan^n(\pi/4 + lat_1/2)/n \tag{2.4.6}$$

$$n = ln(cos(lat_1)/cos(lat_2))/ln(tan(\pi/4 + lat_2/2)/tan(\pi/4 + lat_1/2)) \tag{2.4.7}$$

R is the radius of the Earth, lat_1 and lat_2 are two standard parallels of Lambert projection, and lat_0 and lon_0 are the latitude and longitude of the origin of Cartesian coordinate system (b). To transform Cartesian coordinate (b) to Cartesian coordinate (c), we need to know the coordinate of cone center (x_c, y_c). The x-axis and y-axis of coordinate system (b) are parallel to that of system (c), and having the same direction. Then the formulas from (x_L, y_L) to (x_m, y_m) are :

$$x_m = x_L - x_c \tag{2.4.8}$$

$$y_m = y_L - y_c \tag{2.4.9}$$

For the polar coordinate system, the set of points $(r = 0, \theta = 2k\pi)$, for $k = 0, \pm1, \pm2, \cdots$ coincides with the origin of coordinate system (c), i.e, the cone center of Lambert map projection. And $\theta = 2k\pi$, for $k = 0, \pm1, \pm2, \cdots$ are consistent with positive x-axis. Then the formulas from (x_m, y_m) to (r, θ) are :

$$r = \sqrt{x_m^2 + y_m^2} \tag{2.4.10}$$

$$\theta = acos(x_m/r) \tag{2.4.11}$$

2.5 Intersection Between Sectors and Rectangles

When we have a pair of seed elements, we need to compute the intersection between them. The original face-to-face intersection only handles linear elements whose edges are line segments. But in our project, CAM's elements are sectors in Cartesian coordinate system, so we need to extend the original face-to-face intersection sub-algorithm to sector-rectangle intersection. First of all, we transform both CAM's element and WRF's element onto the Cartesian coordinate system whose origin is the cone center of Lambert projection. Then based on the Cartesian coordinate, we further transform CAM's element into polar coordinate. This sector-rectangle intersection contains two parts:

(a). Point projection

(b). Circular arc-to-segment intersection

(c). Prepare the intersection polygon

2.5.1 Point Projection

Similar to the original face-to-face intersection, we also need to get the images of source vertices on target element and vise versa. Every vertex or edge of a element has a local id. After the point (vertex) projections of source vertices are computed, their relative positions with respect to the current target seed element is found. After that, the point (vertex) projections of target vertices will also be calculated. For every pair of source and target elements, two 4×4 matrices are maintained, which keeps track of which combination of source edge and target edge should be operated with the edge-to-edge intersection. We call these 4×4 matrices as vertex projection matrices. Entry at the position (i, j) of the source vertex projection matrix shows

the relationship between source vertex i and target edge j , where i and j are the local ids of the vertex and the edges within the elements, respectfully. If the (i, j) entry of source vertex projection matrix is positive, then the ith source vertex lies on the left of the jth target edge. Otherwise, if it is negative, then the ith source vertex lies on the right of the jth target edge. If it is zero, then it means that the ith source vertex locates right on the jth target edge. This definition is the same for target vertex projection matrix. Therefore if ith source vertex lies on the right of the jth target edge, and the $(i+1)th$ source vertex lies on the left of the jth target edge, then the ith source edge would intersect with the jth target edge. We call these two matrices as vertex projection matrices, which are necessary for extended edge-to-edge intersection in next step.

In our project, it is relatively easy to find the projection of CAM's vertices with respect to WRF's element. Because WRF's element is a rectangle in Cartesian coordinate, then we parametrize the Cartesian coordinate of a CAM's vertex along the edges of WRF's element and get the boundary encoded natural coordinate (ξ_{cam}, η_{cam}). With the natural coordinate, we can locate the relative position of that CAM's vertex comparing to WRF's edges, and determine the values for corresponding entries of one vertex projection matrix. On the other hand, when we want to project WRF's vertices onto a CAM's element, it is more reasonable use the spherical coordinate instead of Cartesian coordinate because CAM's elements are rectangular in spherical coordinate. Then we can compute the natural coordinate (ξ_{wrf}, η_{wrf}) for a WRF's vertex with respect to CAM's element, and construct the other vertex projection matrix. With the help of two vertex projection matrices, we will know a set of pairs of CAM's edge and WRF's edge that would potentially have intersections.

2.5.2 Arc-to-Segment Intersection

For several pairs of CAM's edge and WRF's edge we found in the last step, some CAM's edges would be circular arcs, while the others are line segments. For a pair of a CAM's edge and a WRF's edge, if the CAM's edge is a line segment, we can just use original edge-to-edge intersection to compute the intersection. If the CAM's edge is a circular arc, the original edge-to-edge intersection sub-algorithm would be useless, therefore we implement a arc-to-segment intersection sub-algorithm to compute the intersection between circular arcs and line segments. Assume the circular arcs is from $v_1^{arc} = (r, \theta_1)$ to $v_2^{arc} = (r, \theta_2)$ that is part of a circle C with radius r, and two end points of the line segment are $v_1^{seg} = (x_1, y_1)$ and $v_2^{seg} = (x_2, y_2)$ which is a part of straight line L, where the circular arc is in polar coordinate and line segment is in Cartesian coordinate whose origin is the cone center of Lambert projection. And the definitions of polar coordinate and Cartesian coordinate are the same as we described in last section. First, we will get the distances of v_1^{seg} and v_2^{seg} to the origin, and let the distance of v_1^{seg} to the origin as l_1 and the distance of v_2^{seg} to the origin as l_2. If $l_1 < r$, then v_1^{seg} is located inside of the circle C ; if $l_1 = r$, then v_1^{seg} coincide with a point of the circle C ; otherwise, v_1^{seg} is outside the circle C. We will also compare l_2 and r similarly. Only compare the distances of v_1^{seg} and v_2^{seg} to the origin is still not enough, because it is possible that both v_1^{seg} and v_2^{seg} are outside circle C, but this segment still has two intersection points with the circle. Therefore, we orthogonally project the origin onto the line L, and compare the location of projection point v_{proj} with two end points of line segments by $t = (v_{proj} - v_{end})/(v_{start} - v_{end})$. In addition, we also calculate the distance between projection point and the origin l_{proj}. In order to make it clear, we

list all the possible cases in the following:

Table 2.1: All possible cases for intersection between circle and line segment

Condition	Number of intersections
$l_1 < r$ and $l_2 < r$	no intersection
$l_1 = r$ and $l_2 = r$	two intersections: v_1^{seg} and v_2^{seg}
$l_1 = r$ and $l_2 < r$	one intersection: v_1^{seg}
$l_1 = r$ and $l_2 > r$ and $t \leqq 0$	one intersection: v_1^{seg}
$l_1 = r$ and $l_2 > r$ and $0 \leqq t \leqq 1$	two intersections
$l_2 = r$ and $l_1 > r$ and $t \geqq 1$	one intersection: v_2^{seg}
$l_2 = r$ and $l_1 > r$ and $0 \leqq t \leqq 1$	two intersections
$l_1 > r$ and $l_2 > r$ and $(t > 1$ or $t < 0)$ and $l_{proj} > r$	no intersection
$l_1 > r$ and $l_2 > r$ and $0 \leqq t \leqq 1$ and $l_{proj} < r$	two intersections
$l_1 > r$ and $l_2 > r$ and $0 \leqq t \leqq 1$ and $l_{proj} = r$	one intersection: v_{proj}
Otherwise	error

It is easy to get the intersection point if there is only one intersection point, because that intersection would be one of v_1^{seg}, v_2^{seg} and v_{proj}. For the cases that having two intersection points, we can the intersections by using parametrization along the line segment. For example, in the case where $l_1 > r$ and $l_2 > r$ and $0 \leqq t \leqq 1$ and $l_{proj} < r$, there will be two intersection points u_1^{inter} and u_2^{inter}. In addition, the distance from v_{proj} to either u_1^{inter} or u_2^{inter} will be $l_{proj-inter} = \sqrt{r^2 - l_{proj}^2}$. Then the parametrization of two intersection points would be t_1 and t_2, where $t_1 = (v_1^{seg} - v_{proj} - l_{proj-center})/(v_1^{seg} - v_2^{seg})$ and $t_2 = (v_1^{seg} - v_{proj} + l_{proj-center})/(v_1^{seg} - v_2^{seg})$. Based on t_1 and t_2, two intersection points are $u_1^{inter} = v_1^{seg} + (v_1^{seg} - v_2^{seg}) * t_1$ and $u_2^{seg} = v_1^{seg} + (v_1^{seg} - v_2^{seg}) * t_2$. In this way, we can get more accurate result than

Algorithm 1 Step I: Compute the intersection between whole circle and line segment

get $d\_center\_pt\_1$ {distance between circle's center and first end point of line segment}
get $d\_center\_pt\_2$ {distance between circle's center and second end point}
get d_center_proj {distance between circle's center and its projection on straight line}

if $d\_center\_pt\_1 < r$ and $d\_center\_pt\_2 < r$ **then**
  case(a): no intersection
else if ($d\_center\_pt\_1 > r$ and $d\_center\_pt\_2 < r$) or ($d\_center\_pt\_1 < r$ and $d\_center\_pt\_2 > r$) **then**
  case(b): one intersection
else if $d\_center\_pt\_1 > r$ and $d\_center\_pt\_2 > r$ **then**
  case(c): needs further judgement
  if $d\_center\_proj < r$ **then**
    case(c-i): two intersections
  else if $d\_center\_proj == r$ **then**
    case(c-ii): one intersection
  else {$d\_center\_proj > r$}
    case(c-iii): no intersection
  end if
else if $d\_center\_pt\_1 == r$ or $d\_center\_pt\_2 == r$ **then**
  case(d): needs further judgement
  if $d\_center\_pt\_1 == r$ and $d\_center\_pt\_2 == r$ **then**
    case(d-i): two intersections
  else if ($d\_center\_pt\_1 == r$ and $d\_center\_pt\_2 > r$) or ($d\_center\_pt\_1 > r$ and $d\_center\_pt\_2 == r$) **then**
    if circle center's projection is outside line segment **then**
      case(d-ii): one intersection
    else {circle center's projection is inside line segment}
      case(d-iii): two intersections
    end if
  else {($d\_center\_pt\_1 == r$ and $d\_center\_pt\_2 < r$) or ($d\_center\_pt\_1 < r$ and $d\_center\_pt\_2 == r$)}
    case(d-iv): one intersection
  end if
else
  error
end if

Algorithm 2 Step II: Check whehter the intersections are within the circular arc

get u^i {intersection point}
if u^i is within circular arc **then**
  u^i is the intersection point between circular arc and line segment
else {u^i is outside circular arc}
  u^i is NOT the intersection point between circular arc and line segment
end if

simply solving quadratic equations which would lead to rounding errors. After we get the intersection points of circle C and line segments, we can transform the intersection points to polar coordinate $u^{inter} = (r, \theta^{inter})$. If θ^{inter} is within the interior of $[\theta_1 + 2k\pi, \theta_2 + 2k\pi]$, for $k = 0, \pm1, \pm2, \cdots$, then u^{inter} is the intersection point of circular arc and line segment.

2.5.3 Prepare the Intersection Polygon

After finishing the above two steps, we get all subvertices of common-refinement. Besides the Cartesian coordinate of subvertices, we also output the parent information for the subvertices. With these information, we can construct an intersection polygon for current sector and rectangle. In addition, some edges of the polygon would be circular arcs, while others are line segments.

2.6 Common refinement Searching Neighborhood

After computing the intersection for a pair of seed elements, it is time to search neighborhood of current seed elements finding potential seed elements. Common-refinement method would be suitable for most pairs of surface meshes in general, for which we can use the method looping algorithm described in section 3.1. For

our project, CAM's mesh and WRF's mesh are all regular meshes, so we optimize the looping algorithm to make it more efficient. First of all, we need to analyze the properties of CAM's mesh and WRF's mesh. As a global climate model, the elements of CAM's mesh have much bigger size than WRF's element. In normal, every CAM's element may include nearly 100 WRF's elements. In practical remapping, we don't need to use global CAM's domain, and instead, we just use certain size of CAM's domain such that it will totally cover WRF's domain that we are interested in. Therefore for the local CAM's domain, the number of elements would be limited, while WRF may have relatively much bigger number of elements. In this way, we mark all the surrounding CAM's elements (at most eight elements) as potential CAM's seed elements being accompanied with current WRF's seed element. For WRF, we remain to use the original method explained in section 3.1, which is to utilize the information of all subvertices of the common-refinement. For example, if a subvertex is located on WRF's edge, then the WRF's element on the other side of that edge will be added into the list of seed elements. If a subvertex coincides with a WRF's vertex, that means this WRF's vertex is located within CAM's element, then the other three, one (vertex is in the boundary) or zero (vertex is in the corner) WRF's elements would intersect with current CAM's seed element. In this way, we can get all potential pairs of seed elements for CAM and WRF.

2.7 Triangulation Method

For the intersection polygon we get in the sector-rectangle intersection, we need to triangulate it so that L_2 minimization method can be used within these small triangles. Although L_2 minimization method can also be applied to the polygon itself,

it would be more accurate to be applied to small triangles. The general process of triangulation is to choose a vertex as the fan center first, and then add diagonals to all the other vertices which are not the neighborhood of the fan center. In order to avoid making ill-conditioned triangles, we choose the vertex as the fan center that has largest angles. For that, we compute *cosine* value for all the vertices, and the smaller the *consine* is, the larger the corresponding angle would be. In our project, CAM's mesh has sectors as its elements and WRF's mesh is a rectangular mesh, so the intersection polygon of sector with rectangle must be a simple connected polygon. Therefore the triangulation method mentioned in the above would be very efficient and robust enough in our project.

2.8 Output

With the small triangles we get in last step, there are several optional outputs of common refinement process. First, we can save sub-vertices to construct the new common refinement mesh whose elements are triangles. The advantages of outputting sub-vertices are very obvious. It is very flexible for further use, and it is very efficiency and straightforward. However, the set of sub-vertices involves vertices from two parent meshes and edge intersection points, so it may require relative big memory for storage if those parent meshes are large.

Instead of storing sub-vertices, we can save the value of moments for every intersection zone of two parent meshes: $\int xdxdy$, $\int ydxdy$, $\int xydxdy$, $\int x^2dxdy$ and $\int y^2dxdy$. In general , second order moments are enough for numerical usage. The intersection zones are polygons, and some edges may be circular arcs. The required storage space of 2nd order moments would be much less than sub-vertices,

and it is easy to get the values of many functions by utilizing moments.

In addition to the above two options, we can also save the entries of mass matrix and load matrix for multiple numerical methods, such as interpolation method, area-weighted method , L2 minimization method and so on. Take L2 minimization method as example, with the entries of mass matrix and load matrix, we can get the value of target functions by solving linear systems $Mg = Lf$, where M is the mass matrix, L is the load matrix, g is the target function and f is the source function. With this kind of output, it would be very easy to integrate multiple numerical method into our algorithm and also very efficient to get the result. Besides, because the entries of mass matrix and load matrix are the products of shape functions, mass matrix and load matrix are sparse matrices which would be very efficient in memory use. That is why we decide to use this output in our real implementation, which will be explained in more details during the following section.

2.8.1 Implementation of L_2 Minimization Method

Within every small triangles, we compute a fraction of entries for mass matrix and load matrix. As we explained before, L_2 minimization method is to minimize the error in L_2 norm: $E = \sqrt{\int_\Omega (g-f)^2 dx}$, according to principle of minimum energy. Minimizing E will lead to solving a linear system as follows:

$$\begin{cases} \int_\Omega (\psi_1\psi_1 g_1 + \psi_1\psi_2 g_2 + \cdots + \psi_1\psi_n g_n) ds = & \int_\Omega \psi_1 (\sum_{i=1}^m \phi_i f_i) ds \\ \ldots\ldots & \ldots \\ \int_\Omega (\psi_n\psi_1 g_1 + \psi_n\psi_2 g_2 + \cdots + \psi_n\psi_n g_n) ds = & \int_\Omega \psi_n (\sum_{i=1}^m \phi_i f_i) ds \end{cases} \tag{2.8.1}$$

or in matrix form:

$$\begin{bmatrix} \int_\Omega \psi_1\psi_1 dx & \int_\Omega \psi_1\psi_2 dx & \cdots & \int_\Omega \psi_1\psi_n dx \\ \int_\Omega \psi_2\psi_1 dx & \int_\Omega \psi_2\psi_2 dx & \cdots & \int_\Omega \psi_2\psi_n dx \\ \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_n\psi_1 dx & \int_\Omega \psi_n\psi_2 dx & \cdots & \int_\Omega \psi_n\psi_n dx \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = \begin{bmatrix} \int_\Omega \psi_1(\sum_{i=1}^m \phi_i f_i) dx \\ \int_\Omega \psi_2(\sum_{i=1}^m \phi_i f_i) dx \\ \vdots \\ \int_\Omega \psi_n(\sum_{i=1}^m \phi_i f_i) dx \end{bmatrix} \tag{2.8.2}$$

The above linear system can be referred as $Mg = L$, where M is the mass matrix, L is the load vector, g is the target function and $f = [f_1, f_2, \cdots, f_m]^T$ is the source function. In order to make our algorithm more convenient and the interface more clear, we divide the load vector L into:

$$\begin{bmatrix} \int_\Omega \psi_1(\sum_{i=1}^m \phi_i f_i) dx \\ \int_\Omega \psi_2(\sum_{i=1}^m \phi_i f_i) dx \\ \vdots \\ \int_\Omega \psi_n(\sum_{i=1}^m \phi_i f_i) dx \end{bmatrix} = \begin{bmatrix} \int_\Omega \psi_1\phi_1 dx & \int_\Omega \psi_1\phi_2 dx & \cdots & \int_\Omega \psi_1\phi_m dx \\ \int_\Omega \psi_2\phi_1 dx & \int_\Omega \psi_2\phi_2 dx & \cdots & \int_\Omega \psi_2\phi_m dx \\ \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_n\phi_1 dx & \int_\Omega \psi_n\phi_2 dx & \cdots & \int_\Omega \psi_n\phi_m dx \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix} \tag{2.8.3}$$

Therefore $Mg = L = Nf$, where N is the load matrix and f is the source function.

2.8.2 Isoparametric Method and Shape Functions

Let's look at the linear system $Mg = Nf$, the entry of mass matrix is in the form $\int_\Omega \psi_i\psi_j ds$, where ψ_i and ψ_j are the basis (or shape) functions at the $i-th$ and $j-th$ vertices of target mesh. This is essentially making use of isoparametric method. Isoparametric method belongs to the family of finite element methods. As other

finite element methods, isoparametric method discretize the whole mesh into small elements, and in every element, the geometry and displacement of some data fields (i.e, source function or target function) are defined by local natural coordinate. The geometry of elements is parametrized with the natural coordinate, while the displacement of data field is parametrized with shape functions. If the elements are super parametric, then natural coordinate coincident to shape functions. The shape functions can be defined at vertices, edge's middle points and center of element. The more shape functions of one element, the higher order of this element will be , and hence the data transfer will be more accurate. Isoparametric method is a coordinate transformation method which transforms between physical coordinate and local natural coordinate.

A complete set of shape functions are essentially the basis of a subspace, so shape functions are also known as basis functions. Certain shape functions of source mesh and target mesh span the subspace of source mesh and target mesh, respectfully. Therefore shape functions can be defined arbitrarily, as long as they can span the specific subspace. For the isoparametric method, shape functions for a quadrilateral element are defined below:

$$\begin{cases} N_1 = & (1-\xi)(1-\eta) \\ N_2 = & \xi(1-\eta) \\ N_3 = & \xi\eta \\ N_4 = & (1-\xi)\eta \end{cases} \tag{2.8.4}$$

where ξ and η are natural coordinate of a certain point with respect to quadrilateral elements, and N_k is the basis function with respect to $k-th$ vertex within that

element. Due to the properties of natural coordinate, basis function $N_1 = 1$ at first vertex but equal to zero at all the other vertices, which is similar for N_2 , N_3 and N_4. The product of N_k and N_l will only have nonzero value within that quadrilateral element if $k \neq l$, and have nonzero value within the surrounding element of the $k-th$ vertex if $k = l$. Furthermore, the product of two basis functions from different elements would be zero if those two elements are not neighborhood to each other. That is why most of entries of mass matrix are equal to zero. In our project, for WRF's element, ξ and η are computed in Cartesian coordinate. While for CAM's element, the natural coordinates are computed in spherical coordinate. That's because CAM's element is rectangle in spherical coordinate, but it will be a sector in Cartesian coordinate. Thus computing natural coordinates in spherical coordinate will be more reasonable and also more accurate for CAM's element than in Cartesian coordinate. Since natural coordinate is local coordinate system within each element, so after we get the natural coordinate, it will be independent from either Cartesian coordinate or spherical coordinate. That's why CAM's mesh can be viewed as quadrilateral mesh.

2.8.3 Discretize Mass Matrix and Load Matrix into Small Triangles

Although the entries of the mass matrix $\int_\Omega \psi_i \psi_j ds$ and the entries of load matrix $\int_\Omega \psi_i \phi_j ds$ are computed over whole target domain Ω, they can be discretized into every small triangles due to isoparametric method and the properties of basis functions. To take the entries of mass matrix as an example, according to properties of basis functions, the domain Ω can be replaced with $p_{i_1} + p_{i_2} + \cdots + p_{i_h}$, where

$p_{i_k}(k = 1, \cdots, h)$ is an intersection polygon of certain pair of a CAM's element and a WRF's element within which the product of basis functions ψ_i and ψ_j is nonzero. Assume the above set of intersection polygon can be divided into a set of small triangles: $t_{i_1} + t_{i_2} + \cdots + t_{i_q}$. Then we have

$$\int_{\Omega} \psi_i \psi_j ds = \int_{\sum_{k=1}^{h} p_{i_k}} \psi_i \psi_j ds = \int_{\sum_{k=1}^{q} t_{i_k}} \psi_i \psi_j ds = \sum_{k=1}^{q} \int_{t_{i_k}} \psi_i \psi_j ds \tag{2.8.5}$$

So the entry of mass matrix $\int_{\Omega} \psi_i \psi_j ds$ can be computed within every small triangle $t_{i_k}(k = 1, \cdots q)$ now. For the entries of load matrix $\int_{\Omega} \psi_i \phi_j ds$, we also discretize it into small triangles similarly. Therefore, using quadrature rule whose order high enough, the entries of both mass matrix and load matrix can be computed exactly over each small triangle.

2.8.4 Computation Over Each Triangle

For a small triangle t_i, we only need to compute following 10 integrals for up-triangle of sub mass matrix due to symmetry of sparse matrix:

$$\begin{bmatrix} \int_{t_i} \psi_{i_1} \psi_{i_1} ds & \int_{t_i} \psi_{i_1} \psi_{i_2} ds & \int_{t_i} \psi_{i_1} \psi_{i_3} ds & \int_{t_i} \psi_{i_1} \psi_{i_4} ds \\ \int_{t_i} \psi_{i_1} \psi_{i_2} ds & \int_{t_i} \psi_{i_2} \psi_{i_2} ds & \int_{t_i} \psi_{i_2} \psi_{i_3} ds & \int_{t_i} \psi_{i_2} \psi_{i_4} ds \\ \int_{t_i} \psi_{i_1} \psi_{i_3} ds & \int_{t_i} \psi_{i_2} \psi_{i_3} ds & \int_{t_i} \psi_{i_3} \psi_{i_3} ds & \int_{t_i} \psi_{i_3} \psi_{i_4} ds \\ \int_{t_i} \psi_{i_1} \psi_{i_4} ds & \int_{t_i} \psi_{i_2} \psi_{i_4} ds & \int_{t_i} \psi_{i_3} \psi_{i_4} ds & \int_{t_i} \psi_{i_4} \psi_{i_4} ds \end{bmatrix} \tag{2.8.6}$$

and also corresponding 16 integrals for sub load matrix:

$$\begin{bmatrix} \int_{t_i} \psi_{i_1}\phi_{j_1}ds & \int_{t_i} \psi_{i_1}\phi_{j_2}ds & \int_{t_i} \psi_{i_1}\phi_{j_3}ds & \int_{t_i} \psi_{i_1}\phi_{j_4}ds \\ \int_{t_i} \psi_{i_2}\phi_{j_1}ds & \int_{t_i} \psi_{i_2}\phi_{j_2}ds & \int_{t_i} \psi_{i_2}\phi_{j_3}ds & \int_{t_i} \psi_{i_2}\phi_{j_4}ds \\ \int_{t_i} \psi_{i_3}\phi_{j_1}ds & \int_{t_i} \psi_{i_3}\phi_{j_2}ds & \int_{t_i} \psi_{i_3}\phi_{j_3}ds & \int_{t_i} \psi_{i_3}\phi_{j_4}ds \\ \int_{t_i} \psi_{i_4}\phi_{j_1}ds & \int_{t_i} \psi_{i_4}\phi_{j_2}ds & \int_{t_i} \psi_{i_4}\phi_{j_3}ds & \int_{t_i} \psi_{i_4}\phi_{j_4}ds \end{bmatrix} \tag{2.8.7}$$

These integrals are computed by using quadrature rule within the triangle t_i. Where ψ_{i_k} (for $k = 1, \cdots, 4$) and ϕ_{j_k} (for $k = 1, \cdots, 4$) are shape functions of parent target element and source element, respectfully. After that we insert the sub mass matrix and sub load matrix into mass matrix M and load matrix N.

2.8.5 Quadrature Rule

In order to compute integral $\int_{t_i} \psi_i \psi_j ds$ within a triangle t_i, we will utilize quadrature rule within that triangle. In our implementation, we use 6-points quadrature rule and we have:

$$\int_{t_i} \psi_i \psi_j ds = \int_{t_i} \psi_i \psi_j |J| d\xi_1 d\xi_2 d\xi_3 \tag{2.8.8}$$

where $|J|$ is the determinant of Jacobian matrix from local coordinate to physical coordinate, and ξ_i is the $i-th$ local coordinate of that triangle. Moreover, ψ_i is also in local coordinate of that triangle. With quadrature rule,

$$\int_{t_i} \psi_i \psi_j |J| d\xi_1 d\xi_2 d\xi_3 = \sum_{k=1}^{6} w_k (\psi_i \psi_j |J|)_k \tag{2.8.9}$$

And the equation of Jacobian matrix is as follows:

$$J = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 \\ \sum_{k=1}^{6} x_k \frac{\partial N_k}{\partial \xi_1} & \sum_{k=1}^{6} x_k \frac{\partial N_k}{\partial \xi_2} & \sum_{k=1}^{6} x_k \frac{\partial N_k}{\partial \xi_3} \\ \sum_{k=1}^{6} y_k \frac{\partial N_k}{\partial \xi_1} & \sum_{k=1}^{6} y_k \frac{\partial N_k}{\partial \xi_2} & \sum_{k=1}^{6} y_k \frac{\partial N_k}{\partial \xi_3} \end{bmatrix} \tag{2.8.10}$$

In the above equation of Jacobian matrix, x_k and y_k are physical coordinates, N_k are shape functions and ξ_i are local natural coordinate. Therefore, the process of computing mass matrix and load matrix is just a process combining isoparametric method and quadrature rule, i.e, to use quadrature on function $\psi_i\psi_j|J|$ within triangle t_i.

2.8.6 Jacobian Matrix

The entries of mass matrix and load matrix are computed within the small triangles of the common-refinement. For a triangle t_i, computing $\int_{t_i} \psi_i\psi_j \cdot |J|dx$ and $\int_{t_i} \psi_i\phi_k \cdot |J|dx$ is just to calculate the area integral of function $\psi_i\psi_j \cdot |J|$ and $\psi_i\phi_k \cdot |J|$ over the small triangle t_i. In our method, we utilize isoparametric method, quadrature rule and Jacobian matrix to get the real integral value in spherical coordinate for the intersection of CAM's mesh and WRF's mesh. The above Jacobian matrix J is the transformation from local coordinate to spherical coordinate. In practice the Jacobian matrix J is divided into two Jacobian matrix J_1 and J_2, where J_1 is the Jacobian matrix which transforms local coordinate to Cartesian coordinate, and J_2 is the Jacobian matrix that transform spherical coordinate to Cartesian coordinate. Then the formulas of computing entries of mass matrix and load matrix become:

$\int_\Omega \psi_i \psi_j \cdot |J_1|/|J_2| dx$ and $\int_\Omega \psi_i \phi_k \cdot |J_1|/|J_2| dx$. Here $J_1 = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}$ while $J_2 = \begin{bmatrix} \frac{\partial x}{\partial (lon)} & \frac{\partial x}{\partial (lat)} \\ \frac{\partial y}{\partial (lon)} & \frac{\partial y}{\partial (lat)} \end{bmatrix}$, where (x,y) is Cartesian coordinate, (ξ,η) is local coordinate and (lat,lon) is spherical coordinate. And in the second Jacobian matrix J_2, the longitude is ahead of latitude. Take Lambert map projection as an example, the formula of Lambert projection is :

$$\begin{cases} x = \rho \cdot sin(n(lon - lon_0)) \\ y = \rho_0 - \rho \cdot cos(n(lon - lon_0)) \\ n = \frac{ln(sin(\pi/2 - lat1)) - ln(sin(\pi/2 - lat2))}{ln(tan(\pi/4 - lat1/2)) - ln(tan(\pi/4 - lat2/2))} \\ F = cos(lat1) \cdot tan^n(\pi/4 + lat1/2)/n \\ \rho = RF \cdot tan^n(\pi/4 - lat/2) \end{cases} \tag{2.8.11}$$

where R is the radius of the Earth, lat1 and lat2 are two standard parallels of Lambert projection, and lat0 and lon0 are the latitude and longitude of the origin. Then the entries of J_2 are as follows:

$$\begin{cases} \frac{\partial x}{\partial (lon)} = \rho \cdot n \cdot cos[n(lon - lon_0)] \\ \frac{\partial x}{\partial (lat)} = -\frac{n}{cos(lat)} \cdot \rho \cdot sin[n(lon - lon_0)] \\ \frac{\partial y}{\partial (lon)} = \rho \cdot n \cdot sin[n(lon - lon_0)] \\ \frac{\partial y}{\partial (lat)} = \frac{n}{cos(lat)} \cdot \rho \cdot cos[n(lon - lon_0)] \end{cases} \tag{2.8.12}$$

2.9 Resolution of Non-Matching Boundary

Boundary issue is very important for remapping method between multiple scales, because it is highly possible that some big jumpings or inconsistencies happen around the boundary. Especially in our project, since CAM's elements are sectors in Cartesian coordinate and at the same time WRF's elements are rectangles, then the boundary between two meshes would be very irregular. In practical use, CAM's domain will totally cover WRF's domain. We don't need to worry about the boundary issue when transfer data from CAM to WRF, because the computing domain is the target domain which is WRF's domain in downscaling. However, it would be very difficult when transfer dada back from WRF to CAM, because WRF's mesh doesn't have information for parts of CAM's domain which are not covered by WRF's domain. In order to solve this issue, we implement a specific upscaling method to handle the data remapping from WRF to CAM. For this upscaling method, the general idea is to integrate the information into source function for multiple parts of CAM's domain which are not covered by WRF's domain. For example, assume source function in WRF's domain is $f = (f_1, \cdots, f_n)^T$, and the unknown target function in CAM's domain is $g = (g_1, \cdots, g_n)^T$. In addition, $g^{out} = (g_1^{out}, \cdots, g_h^{out})$ is the information for parts of CAM's domain that are outside WRF's domain. In order to transfer data smoothly, we integrate g^{out} into source function, so that source function becomes $f^{ext} = (f_1, \cdots, f_n, g_1^{out}, \cdots, g_h^{out})$, while the target function remains the same. In real implementation, two way data remapping always starts from downscaling that is from CAM to WRF, which makes our upscaling method practical. We can stores the g^{out} during downscaling transfer first, and then integrate g^{out} into f during upscaling.

Therefore, the original L_2 minimization algorithm is:

$$\begin{bmatrix} \int_\Omega \psi_1\psi_1\cdot|J|dx & \int_\Omega \psi_1\psi_2\cdot|J|dx & \cdots & \int_\Omega \psi_1\psi_m\cdot|J|dx \\ \int_\Omega \psi_2\psi_1\cdot|J|dx & \int_\Omega \psi_2\psi_2\cdot|J|dx & \cdots & \int_\Omega \psi_2\psi_m\cdot|J|dx \\ \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_m\psi_1\cdot|J|dx & \int_\Omega \psi_m\psi_2\cdot|J|dx & \cdots & \int_\Omega \psi_m\psi_m\cdot|J|dx \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix} \tag{2.9.1}$$

$$= \begin{bmatrix} \int_\Omega \psi_1\phi_1\cdot|J|dx & \int_\Omega \psi_1\phi_2\cdot|J|dx & \cdots & \int_\Omega \psi_1\phi_n\cdot|J|dx \\ \int_\Omega \psi_2\phi_1\cdot|J|dx & \int_\Omega \psi_2\phi_2\cdot|J|dx & \cdots & \int_\Omega \psi_2\phi_n\cdot|J|dx \\ \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_m\phi_1\cdot|J|dx & \int_\Omega \psi_m\phi_2\cdot|J|dx & \cdots & \int_\Omega \psi_m\phi_n\cdot|J|dx \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix} \tag{2.9.2}$$

After being applied with the new upscaling method, the algorithm becomes:

$$\begin{bmatrix} \int_\Omega \psi_1\psi_1\cdot|J|dx & \int_\Omega \psi_1\psi_2\cdot|J|dx & \cdots & \int_\Omega \psi_1\psi_m\cdot|J|dx \\ \int_\Omega \psi_2\psi_1\cdot|J|dx & \int_\Omega \psi_2\psi_2\cdot|J|dx & \cdots & \int_\Omega \psi_2\psi_m\cdot|J|dx \\ \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_m\psi_1\cdot|J|dx & \int_\Omega \psi_m\psi_2\cdot|J|dx & \cdots & \int_\Omega \psi_m\psi_m\cdot|J|dx \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_m \end{bmatrix} \tag{2.9.3}$$

$$= \begin{bmatrix} \int_\Omega \psi_1\phi_1 \cdot |J|dx & \cdots & \int_\Omega \psi_1\phi_n \cdot |J|dx & \int_\Omega \psi_1\psi_1^{out} \cdot |J|dx & \cdots & \int_\Omega \psi_1\psi_h^{out} \cdot |J|dx \\ \int_\Omega \psi_2\phi_1 \cdot |J|dx & \cdots & \int_\Omega \psi_2\phi_n \cdot |J|dx & \int_\Omega \psi_2\psi_1^{out} \cdot |J|dx & \cdots & \int_\Omega \psi_2\psi_h^{out} \cdot |J|dx \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_m\phi_1 \cdot |J|dx & \cdots & \int_\Omega \psi_m\phi_n \cdot |J|dx & \int_\Omega \psi_m\psi_1^{out} \cdot |J|dx & \cdots & \int_\Omega \psi_m\psi_h^{out} \cdot |J|dx \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ g_1^{out} \\ \vdots \\ g_h^{out} \end{bmatrix} \tag{2.9.4}$$

Chapter 3

Conservative Data Transfer

In this chapter, we will describe several three numerical methods for data transfer: L_2 minimization method, Sobolev method and monotone method. All these three methods satisfy physical conservation, which is very important for data transfer between climate models. L_2 method utilizes 2nd order source and target shape functions, so that it can achieve high order accuracy besides physical conservation. Sobolev method is an extension of L_2 minimization method, and it will be used when the source functions have very big gradients at some points. L_2 minimization method doesn't guarantee positivity, although it does well in our tests. In order to strictly preserve positivity, we implement a monotone method.

3.1 L_2 Minimization Method

In the following section, we will introduce L_2 minimization method, which is accurate and physically conservative.

3.1.1 Notation and Semi-Discrete Formulation

Main idea is to minimize L_2 norm of errors of source function and target function over target domain: $E = \sqrt{\int_\Omega (g-f)^2 dx}$, where f is the source function and g is the target function. Assume source mesh has m vertices, while target mesh has n vertices, then derivation from finite element method we have $f = \sum_{i=1}^{m} \phi_i f_i$ and $g = \sum_{j=1}^{n} \psi_j g_j$, where ϕ_i and f_i are basis function and function value of the $i-th$ vertex of source mesh, respectfully. Similar case for ψ_j and g_j. According to principle of minimum energy (or Rayleigh-Ritz method), the error will be minimized if $\partial E^2/\partial g_i = 0$, which leads to $\partial \int_\Omega (g-f)^2 dx/\partial g_i = 0$, for $k = 1, \cdots, n$. Then we can have the following equations:

$$\frac{\partial \int_\Omega (g-f)^2 ds}{\partial g_k} = \frac{\partial \int_\Omega g^2 ds}{\partial g_k} - 2\frac{\int_\Omega gf ds}{\partial g_k} + \frac{\partial \int_\Omega f^2 ds}{\partial g_k} \tag{3.1.1}$$

$$= \frac{\partial \int_\Omega (\sum_{j=1}^{n} \psi_j g_j)^2 ds}{\partial g_k} - 2\frac{\int_\Omega (\sum_{j=1}^{n} \psi_j g_j)(\sum_{i=1}^{m} \phi_i f_i) ds}{\partial g_k} + \frac{\partial \int_\Omega (\sum_{i=1}^{m} \phi_i f_i)^2 ds}{\partial g_k} \tag{3.1.2}$$

$$= 2\int_\Omega \psi_k \sum_{j=1}^{n} \psi_j g_j ds - 2\int_\Omega \psi_k (\sum_{i=1}^{m} \phi_i f_i) ds \tag{3.1.3}$$

$$= 2\int_\Omega (\psi_k \psi_1 g_1 + \psi_k \psi_2 g_2 + \cdots + \psi_k \psi_n g_n) ds - 2\int_\Omega \psi_k (\sum_{i=1}^{m} \phi_i f_i) ds = 0 \tag{3.1.4}$$

The above equations are actually a linear equation system:

$$\begin{cases} \int_\Omega(\psi_1\psi_1 g_1+\psi_1\psi_2 g_2+\cdots+\psi_1\psi_n g_n)ds= & \int_\Omega \psi_1(\sum_{i=1}^m \phi_i f_i)ds \\ \ldots\ldots & \ldots \\ \int_\Omega(\psi_n\psi_1 g_1+\psi_n\psi_2 g_2+\cdots+\psi_n\psi_n g_n)ds= & \int_\Omega \psi_n(\sum_{i=1}^m \phi_i f_i)ds \end{cases} \tag{3.1.5}$$

Then transform linear system to matrix form $Mx=b$, where M is the mass matrix, b is the load vector and J is the Jacobian matrix from Cartesian coordinate to spherical coordinate:

$$\begin{bmatrix} \int_\Omega \psi_i\psi_j\cdot|J|dx \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = \begin{bmatrix} \int_\Omega \psi_i(\sum_{k=1}^m \phi_k f_k)\cdot|J|dx \end{bmatrix} \tag{3.1.6}$$

3.1.2 Proof of Physical Conservation

L_2 minimization method strictly satisfies physical conservation over the domain of target mesh. The mathematical proof of physical conservation are as follows. Based on the above linear system , we can get for $k=1,\cdots,n$:

$$\int_\Omega(\psi_k\sum_{j=1}^n \psi_j g_j)ds=\int_\Omega(\psi_k\sum_{i=1}^m \phi_i f_i)ds \tag{3.1.7}$$

Since $f=\sum_{i=1}^m \phi_i f_i$ and $g=\sum_{j=1}^n \psi_j g_j$, we get:

$$\int_\Omega \psi_k g ds=\int_\Omega \psi_k f ds \tag{3.1.8}$$

Move the right hand side to the left, we have:

$$\int \psi_k(g-f)ds = 0\,, \tag{3.1.9}$$

which means if target function g satisfies principle of minimum energy, then $(g-f)$ will be orthogonal to all basis functions ψ_k, for $k = 1, \cdots n$. Therefore $(g-f)$ will be orthogonal to the function space spanned by a set of target basis functions ψ_k , for $k = 1, \cdots n$:

$$\int_\Omega (g-f)vds = 0\,, \tag{3.1.10}$$

where v is any function within that function space. If let v be the constant function 1, then we have

$$\int_\Omega gds = \int_\Omega fds\,, \tag{3.1.11}$$

which means the value is physical conservative over the domain of target mesh. Then the target function g which we get from the output of linear system satisfies physical conservation.

3.1.3 Proof of Accuracy

Besides physical conservation, we also concern about accuracy of L_2minimization method. Here we will compare L_2 minimization method with point wise Linear Interpolation method. If we assume $g^{(1)}$ is the result of L_2 minimization method, and $g^{(2)}$ is the result of interpolation method. Because $||g^{(1)} - f||_2$ is minimized in the function space spanned target basis function, then $||g^{(1)} - f||_2 \leqq ||g^{(2)} - f||_2$. From this aspect, L_2minimization method will be more accurate then interpolation method in general, and will be as accurate as interpolation method at least.

3.1.4 Features of Mass Matrix

Mass matrix has some attracting properties, such as: symmetric, sparse and positive definite. Based on these features, solving the linear system of mass matrix would be relatively easy and efficient. In addition, mass matrix would require less storage space compared to storing the subvertices of common-refinement mesh. We will show more details of these properties in the following.

First, the mass matrix M is symmetric, which is obvious from the definition $M_{ij} = \int_\Omega \psi_i \psi_j dx$. Then $M_{ij} = M_{ji} = \int_\Omega \psi_i \psi_j dx$.

Second, basis functionsψ_i are nearly orthogonal to each other, so most of $\psi_i \psi_j$ are equal to zero. Then mass matrix is a typical sparse matrix.

In addition, because for any vector $x = (x_1, x_2, \cdots, x_n)^T$, we have

$$x^T M x = (x_1, x_2, \cdots, x_n) \begin{bmatrix} \int_\Omega \psi_1 \psi_1 ds & \int_\Omega \psi_1 \psi_2 ds & \cdots & \int_\Omega \psi_1 \psi_n ds \\ \int_\Omega \psi_2 \psi_1 ds & \int_\Omega \psi_2 \psi_2 ds & \cdots & \int_\Omega \psi_2 \psi_n ds \\ \cdots & \cdots & \cdots & \cdots \\ \int_\Omega \psi_n \psi_1 ds & \int_\Omega \psi_n \psi_2 ds & \cdots & \int_\Omega \psi_n \psi_n ds \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \tag{3.1.12}$$

$$x^T M x = \int_\Omega (\sum_{j=1}^{n} \psi_j x_j)^2 ds \geqq 0 \,, \tag{3.1.13}$$

where $x^T M x = 0$ if and only if $x = 0$. Then the mass matrix is a positive definite matrix.

Moreover, the condition number of M is only dependent on mesh quality of target mesh, so for well-shaped mesh L_2 minimization method will always be good conditioned.

3.2 Sobolev Minimization Method

L_2 minimization gives accurate and conservative solutions for many cases, but unfortunately it cannot always deliver satisfactory results. In particular, if the source function changes abruptly, L_2 minimization method may cause overshoots and undershoots at some points, a behavior similar to the Gibbs phenomenon seen in Fourier analysis. A few approaches are feasible to resolve these overshoots and undershoots. For simple applications where jumps are isolated at well-characterized geometric features such as sharp edges of a surface, the simplest and most effective approach is to split the interface into patches along these features and to transfer data separately for each patch. Another approach is still to solve for the whole interface but introduce non-linear limiters to the linear solvers. This section pursues a different avenue by altering the objective function through minimizing a different norm of error, namely the Sobolev norm, to damp out the oscillations.

Let k be a constant integer. Given a function f whose derivatives of all orders less than or equal to k exist almost everywhere, then the Sobolev norm of f is

$$||f||_{H^k} = \sqrt{\sum_{i=0}^{k} \alpha_i ||f^{(i)}||_2^2}\,, \tag{3.2.1}$$

where $f^{(i)}$ denotes the $i-th$ derivative of f, and α_i is its associated weight and is in general non-negative. Actually, the L_2 norm is a special case of the Sobolev norm with $k = 0$. Intuitively, if the derivatives of the source function are already known, a data transfer method can take advantage of them to obtain a more meaningful solution, and this section exploits this idea. For simplicity, this paper considers

only the case $k = 1$ and $\alpha_0 = 1$, i.e. minimizing

$$||g-f||_{H^1}^2 = ||g-f||_2^2 + \alpha||g^{'} - f^{'}||_2^2 \tag{3.2.2}$$

This method is referred to as Sobolev minimization. The first derivative of $f = \sum_{i=1}^{m} \phi_i f_i$ can be computed as $f^{'} = \sum_{i=1}^{m} \phi_i^{'} f_i$, obtained by differentiating f element-by-element, or as $f^{'} = \sum_{i=1}^{m} \phi_i f_i^{'}$, where $f_i^{'}$ is some derivative value at the node i supplied as input or computed using a certain approximation. The first approximation is referred to as consistent differentiation, and the second as alternative differentiation. Alternative differentiation can potentially deliver higher order of accuracy than consistent differentiation.

Assume that consistent differentiation is used for g. The Sobolev norm is minimized if $\partial \int_\Omega (g-f)^2 + \alpha(g^{'} - f^{'})^2 dx / \partial g_k = 0$ for $k = 1, ..., n$. Now

$$\frac{\partial \int_\Omega (g-f)^2 + \alpha(g^{'} - f^{'})^2 dx}{\partial g_k} = \frac{\partial \int (\sum_{j=1}^{n} \psi_j g_j - f)^2 + (\sum_{j=1}^{n} \psi_j g_j^{'} - f^{'})^2 dx}{\partial g_k} \tag{3.2.3}$$

$$= 2\sum_{j=1}^{n} g_k \int_\Omega \psi_k \psi_j + \alpha \psi_k^{'} \psi_j^{'} dx - 2\int_\Omega \psi_k f + \alpha \psi_k^{'} f^{'} dx \,, \tag{3.2.4}$$

which again leads to an $n \times n$ linear system $Sx = b$, where $S = M + \alpha K$ is a weighted sum of the mass matrix M and the stiffness matrix K, with entries $K_{ij} = \int_\Omega \psi_i^{'} \psi_j^{'} dx$, and b is the modified load vector, with components $b_k = \int_\Omega \psi_k f + \alpha \psi_k^{'} f^{'} dx$. Same as L_2 minimization, Sobolev minimization is also conservative, because $\int_\Omega (g - f)v + (g^{'} - f^{'})v^{'} dx = 0$ for any function v in the function space spanned by the ψ_i , and constant functions are in this space. Note that this argument holds indepen-

dently of how f' is computed.

Sobolev minimization shares some properties with L_2 minimization: The matrix S is sparse and symmetric positive definite, and this scheme stabilizes the solution in that $||g||_{H^1} = ||f||_{H^1}$. However, because the stiffness matrix K is singular, the matrix S is not necessarily well conditioned for large α. In general, α should be of order $O(h_t^2)$, as a larger α would lead to ill-conditioned systems as $h_t \to 0$. Furthermore, $||f-g||_{H^1}^2$ is proportional to $O(\alpha h_t^2 + h_t^4 + h_s^4)$, and an α of order $O(h_t^2)$ ensures the same order of accuracy as L_2 minimization. To discretize this method, a common-refinement based scheme needs to be used and exact integration is preferred for optimal accuracy. When computing S_{ij}, because the degree of the polynomials $\psi_i'\psi_j'$ is lower than that of $\psi_i\psi_j$, integrating to compute K_{ij} requires a smaller number of quadrature points than for computing M_{ij}.

Sobolev minimization can effectively smooth out overshoots and undershoots, because such oscillations tend to have large derivatives, and Sobolev minimization inhibits unwanted oscillations by biasing the target function toward some prescribed derivatives. Sobolev minimization also has advantages in transferring data from a coarse mesh onto a fine mesh. Consider transferring a piecewise linear representation of a smooth function from a coarse mesh with k intervals onto a fine grid with ck intervals, where c is a positive integer. When minimizing the L_2 norm, the target function would be identical to the source function, because the source basis functions are in the space of the target basis functions. By specifying continuously varying f', Sobolev minimization can construct a smoother representation on the target mesh.

3.3 Monotone Method

In climate models, negative values are meaningless for some physical fields. Therefore, positivity preservation is very important for data transfer for coupling of climate models. In our framework, we have implemented a monotone method for positivity preservation which is described in (P. Jones, 1999) and (J. K. Dukowicz and J. W. Kodis, 1987). In this monotone method, the target shape functions are constant 1, while the source shape functions are linear. We utilize the gradient information of the source function to improve the accuracy of the data transfer.

3.3.1 Formulation

In order to describe the method more clear, we list the formulation of monotone method in the following:

$$I \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} \tag{3.3.1}$$

$$= \begin{bmatrix} l_{1k_{a1}}(\bar{f}_{k_{a_1}} + \nabla_{k_{a_1}} f(r) \cdot (r - \bar{r}_{a_1})) + \cdots + l_{1k_{b1}}(\bar{f}_{k_{b_1}} + \nabla_{k_{b_1}} f(r) \cdot (r - \bar{r}_{b_1})) \\ l_{2k_{a2}}(\bar{f}_{k_{a_2}} + \nabla_{k_{a_2}} f(r) \cdot (r - \bar{r}_{a_2})) + \cdots + l_{1k_{b2}}(\bar{f}_{k_{b_2}} + \nabla_{k_{b_2}} f(r) \cdot (r - \bar{r}_{b_2})) \\ \vdots \\ l_{1k_{an}}(\bar{f}_{k_{a_n}} + \nabla_{k_{a_n}} f(r) \cdot (r - \bar{r}_{a_n})) + \cdots + l_{1k_{bn}}(\bar{f}_{k_{b_n}} + \nabla_{k_{b_n}} f(r) \cdot (r - \bar{r}_{b_n})) \end{bmatrix} \tag{3.3.2}$$

In the above linear system $Ig = f$, I is an identity matrix, g is the cell-centered value of target function and f is the source function with gradient information. $\bar{f}_k$ is the cell-centered average value of the kth element in the source domain, and

$\nabla_k f(r)$ is the gradient value at location r in the kth element and $\bar{r}_k$ is the centroid of the kth source element. The ith row of right hand side vector f only contains the information of source elements which have intersections with the ith target element, and k_{ai} to k_{bi} are indices of those related source elements of the ith target element. In addition, $l_{ik_{ai}}$ to $l_{ik_{bi}}$ are the parameters of corresponding source element. In order to make it as consistent with L_2 method, we reorganize right hand side of the above linear system $Ig = f$ into $Ig = L_1\bar{f} + L_2\nabla f$ as follows:

$$I\begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ g_n \end{bmatrix} = \begin{bmatrix} l_{11}^1 & l_{12}^1 & \cdots & l_{1m}^1 \\ l_{21}^1 & l_{22}^1 & \cdots & l_{2m}^1 \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1}^1 & l_{n2}^1 & \cdots & l_{nm}^1 \end{bmatrix} \begin{bmatrix} \bar{f}_1 \\ \bar{f}_2 \\ \vdots \\ \bar{f}_m \end{bmatrix} + \begin{bmatrix} l_{11}^2 & l_{12}^2 & \cdots & l_{1m}^2 \\ l_{21}^2 & l_{22}^2 & \cdots & l_{2m}^2 \\ \cdots & \cdots & \cdots & \cdots \\ l_{n1}^2 & l_{n2}^2 & \cdots & l_{nm}^2 \end{bmatrix} \begin{bmatrix} \nabla_1 f \\ \nabla_2 f \\ \vdots \\ \nabla_m f \end{bmatrix} \tag{3.3.3}$$

If we assume the ith target element has positive intersection area $area_{inter}$ with the kth source element, then l_{ik}^1 and l_{ik}^2 are positive, and their values are $area_{inter}/area_i$ and $(area_{inter}/area_i) \cdot (\bar{r}_{inter} - \bar{r}_k)$ respectfully, where $area_i$ is the area of the ith target element and $\bar{r}_{inter}$ is the centroid of the intersection zone of the ith target element and the kth source element.

From the above formulation, it is obvious that the monotone will reduce to normal area-weighted method if we ignore the gradient information. The same as area-weighted method, monotone method also satisfies physical conservation, but it has one more order accuracy than area-weighted method with source gradient information. In our project, we obtain linear approximation of the gradient, therefore the gradient would be exact if the source function is linear. But if the source function has some sharp features at some points, the gradient approximation would

get large overshoots or undershoots at those points, which would cause monotone method loses the ability to preserve positivity. For such reasons, we need to introduce gradient limiting to avoid such overshoots and undershoots.

3.3.2 Gradient Limiting

The gradient limiting is to smooth the overshoots and undershoots of gradient approximation when the source function has very big gradient at some points. Currently, we use Van Leer limiting method, which is described in (J. K. Dukowicz and J. W. Kodis). The general idea of Van Leer limiting is to multiply the initial value of gradient approximation $\nabla_i f$ with a parameter α_i, where $\alpha_i = min\{1, \alpha_{max}, \alpha_{min}\}$. In addition, $\alpha_{max} = max\{0, (f_{max} - f_i)/(f_{i_{max}} - f_i)\}$, and $\alpha_{min} = max\{0, (f_{min} - f_i)/(f_{i_{min}} - f_i)\}$, where f_{max} is the maximum cell-centered value of neighborhood source elements and $f_{i_{max}}$ is the maximum value within the ith source element. Therefore, with the help of gradient limiting, monotone method can preserve the monotonicity, which is very helpful for data transfer of coupling of climate models.

Chapter 4

Results

In order to show the accuracy and physical conservation of our method, we have operated series of tests. We test convergent rate of L_2 method, and also plot the results for several analytic functions and real fields from climate models. From the results, L_2 method are more accurate than area-weighted method and monotone method. L_2 method preserves positivity very well in our test, although it doesn't guarantee positivity. We also test multiple steps transfer in order to show the ability of our method in real practice, and the results are quite encouraging. Comparing to area-weighted method and monotone method, L_2 method preserves the shape very well when transferring water vapor between CAM and WRF after 1000 steps. Therefore , in this chapter we will show the results and explain them in details.

4.1 Convergent Rate

With the result of common-refinement based L_2 minimization method, we need to test the convergent rate of error. We choose a pair of CAM's mesh and WRF's

mesh, whose sizes are 26×28 and160×160, respectfully. The two meshes are shown as follows:

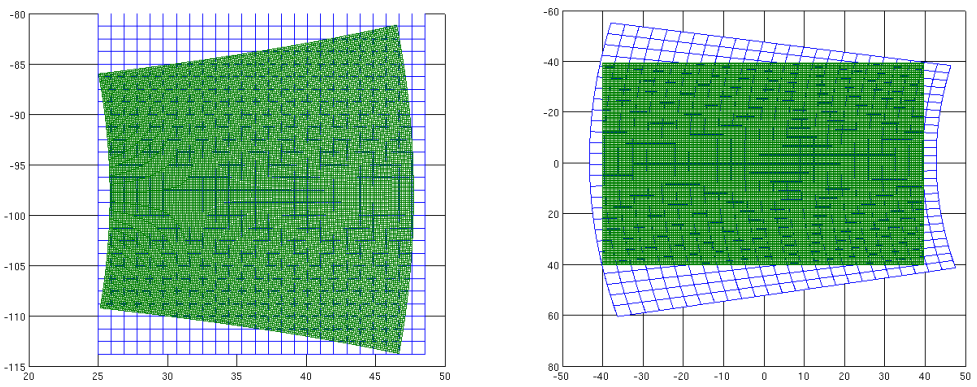

Figure 4.1: Left to right: (a). two meshes in spherical coordinate, (b). two meshes in Cartesian coordinate

Within the above figure, blue mesh is the CAM's mesh, and green mesh is WRF's mesh. For different resolutions of that pair of meshes, we test three analytic functions: $f_1 = 2 + cos^2(\theta)cos(2\phi)$, $f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$ and $f_3 = y^2$, and compare the output with analytic value of target function to get the relative error in L_2 norm by using the equation:

$$error = \sqrt{\sum_{k=1}^{n}(g_k - g_k^{analytic})^2 / \sum_{k=1}^{n}(g_k^{analytic})^2}\,, \tag{4.1.1}$$

where g_k is the output value of target function at $k-th$ vertex, and $g_k^{analytic}$ is the corresponding analytic value. For L_2 minimization method, area-weighted method and monotone method, we all calculate the relative error in L_2 norm for both downscaling (from CAM to WRF) and upscaling (from WRF to CAM). First, we show the quantitative results for L_2 minimization method. And from the table, we can see

that if the mesh is fine enought, the L_2 error can achieve the order of 10^{-5}.

Table 4.1: L_2 norm error in downscaling

Relative L_2 error in downscaling	7×7 vs40×40	13×14 vs 80×80	26×28 vs160×160
$f_1 = 2 + cos^2(\theta)cos(2\phi)$	0.0018	4.6199e-04	8.4396e-05
$f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$	0.0512	0.0129	0.0025
$f_3 = y^2$	5.9026e-04	1.4880e-04	2.9375e-05

Table 4.2: L_2 norm error in upscaling

Relative L_2 error in upscaling	7×7 vs 40×40	13×14 vs 80×80	26×28 vs160×160
$f_1 = 2 + cos^2(\theta)cos(2\phi)$	0.0010	2.5407e-04	6.2861e-05
$f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$	0.0331	0.0083	0.0020
$f_3 = y^2$	3.4685e-04	8.9371e-05	2.2225e-05

Afterwards, we also compute L_2 error for area-weighted method. For area-weighted method, the errors are at least around 10^{-3}, which are bigger than that of L_2 method.

Table 4.3: L_2 norm error in downscaling

Relative L_2 error in downscaling	7×7 vs40×40	13×14 vs 80×80	26×28 vs160×160
$f_1 = 2 + cos^2(\theta)cos(2\phi)$	0.0123	0.0062	0.0031
$f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$	0.0770	0.0382	0.0188
$f_3 = y^2$	0.0250	0.0125	0.0063

Table 4.4: L_2 norm error in upscaling

Relative L_2 error in upscaling	7×7 vs 40×40	13×14 vs 80×80	26×28 vs160×160
$f_1 = 2 + cos^2(\theta)cos(2\phi)$	0.0023	7.8190e-04	3.2218e-04
$f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$	0.0184	0.0056	0.0018
$f_3 = y^2$	0.0041	0.0015	5.7208e-04

In addition, the following are the results of monotone method. From the following tables, monotone method has more accurate results than area-weighted method, but has less accurate results than L_2 minimization method. From the above chapters, we know that area-weighted method has both constant source shape functions and constant target shape functions, while monotone method has linear source shape functions and constant target functions. In addition, L_2 minimization method has

2nd order source and target shape functions. Then the quantitative results are consistent with the order of shape funtions of those three methods.

Table 4.5: L_2 norm error in downscaling

Relative L_2 error in downscaling	7×7 vs40×40	13×14 vs 80×80	26×28 vs160×160
$f_1 = 2 + cos^2(\theta)cos(2\phi)$	3×10^{-3}	7×10^{-4}	4×10^{-4}
$f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$	0.0769	0.0382	0.0186
$f_3 = y^2$	0.0036	0.0019	6.2672e-04

Table 4.6: L_2 norm error in upscaling

Relative L_2 error in upscaling	7×7 vs 40×40	13×14 vs 80×80	26×28 vs160×160
$f_1 = 2 + cos^2(\theta)cos(2\phi)$	6×10^{-4}	1×10^{-4}	6×10^{-5}
$f_2 = 2 + sin^{16}(2\theta)cos(16\phi)$	0.0184	0.0056	0.0018
$f_3 = y^2$	1×10^{-3}	6×10^{-4}	1×10^{-4}

And the plots of convergent rate for $f_3 = y^2$ are:

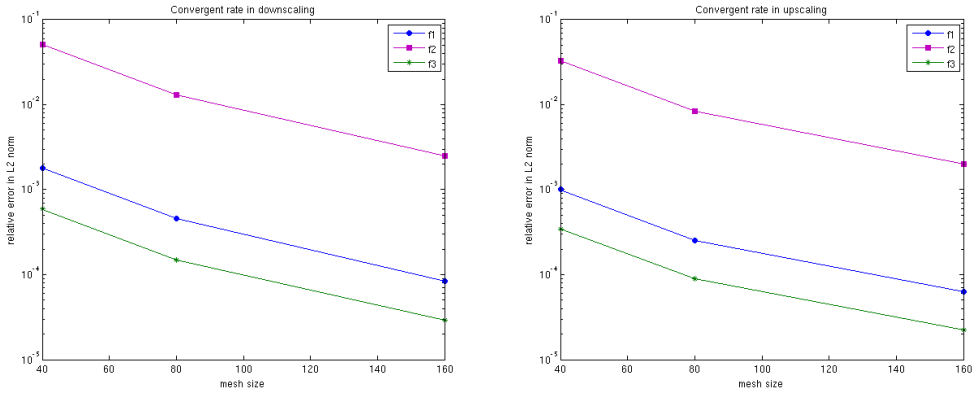

Figure 4.2: Left to right: (a). downscaling convergent rate, (b). upscaling convergent rate

4.2 Comparison with Existing Methods

There are many existing available numerical methods for data remapping, such as: interpolation method, area weighted method (P. John, 1999), monotone method (P. John, 1999) and so on. Among these numerical methods, interpolation method doesn't satisfy physical conservation, while area weighted method, monotone method and our method strictly satisfy physical conservation. Since physical conservation is very important for climate and weather models, then we only focus on the comparison among our method, area weighted method and monotone method. The comparison among interpolation method, area weighted method and monotone method can be found in (P. John, 1999). The later two methods are also implemented by P. John into the numerical package SCRIP, which is widely used in atmospheric community.

First, we choose a CAM's mesh with size 12×17 and a WRF's mesh with size

101×101, which are shown below:

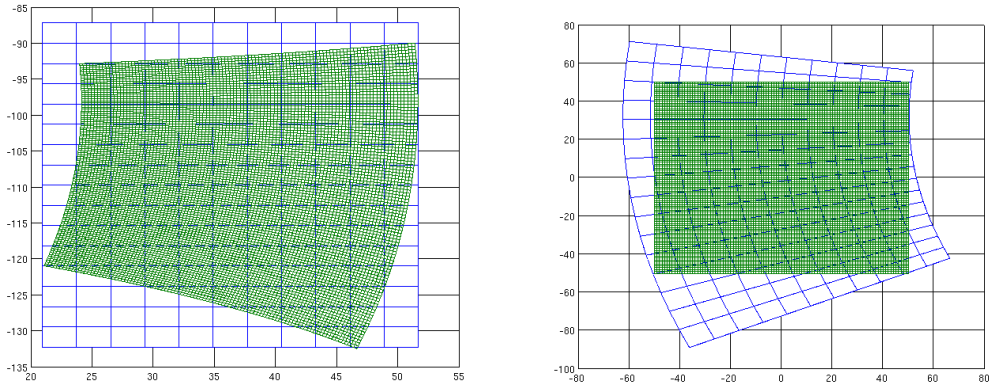

Figure 4.3: Left to right: (a). two meshes in spherical coordinate, (b). two meshes in Cartesian coordinate

For the above pair of meshes, we test several functions or real field: $f_1 = y$, $f_2 = y^{-6}$, $f_3 = 2 + cos^2(\theta)cos(2\phi)$, $f_4 = 2 + sin^{16}(2\theta)cos(16\phi)$ and $f_5 = q$, where f_5 is the water vapor from the output of CAM, and we calculate relative maximum errors and relative L_2 errors for the above analytic functions or real fields. First, we show the results of test with function $f_1 = y$:

Table 4.7: Compare the result of $f_1 = y$

	Downscaling		Upscaling	
Method	max error	L_2error	max error	L_2error
area weighted	0.0133	0.0065	0.0039	6.8558e-04
monotone	0.0035	9.9020e-04	9.5820e-04	6.9536e-05
L_2 minimization	9.8315e-07	1.2453e-08	3.9371e-09	5.0633e-10

Then the following are the results of the test with the function $f_2 = y^{-6}$:

Table 4.8: Compare the result of $f_2 = y^{-6}$

	Downscaling		Upscaling	
Method	max error	L_2error	max error	L_2error
area weighted	0.0835	0.0429	0.0226	0.0060
monotone	0.0496	0.0207	0.0149	0.0046
L_2 minimization	0.0050	0.0031	0.0041	0.0018

The following pictures shows the data transferring of $f_2 = y^{-6}$ between CAM and WRF:

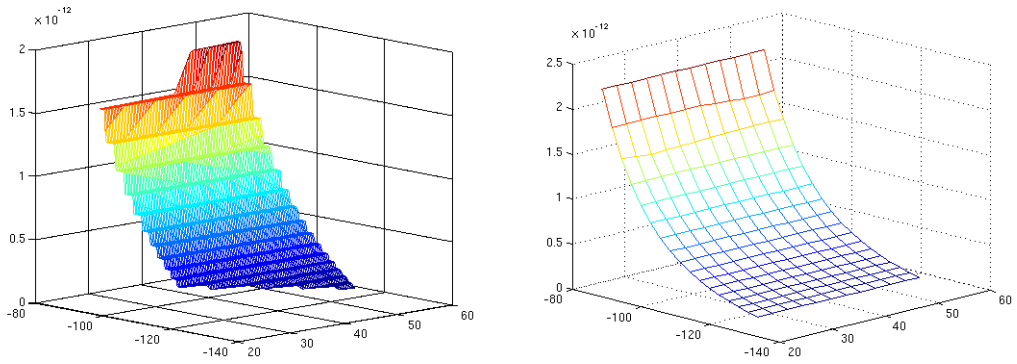

Figure 4.4: Area weighted method. Left is downscaling and right is upscaling.

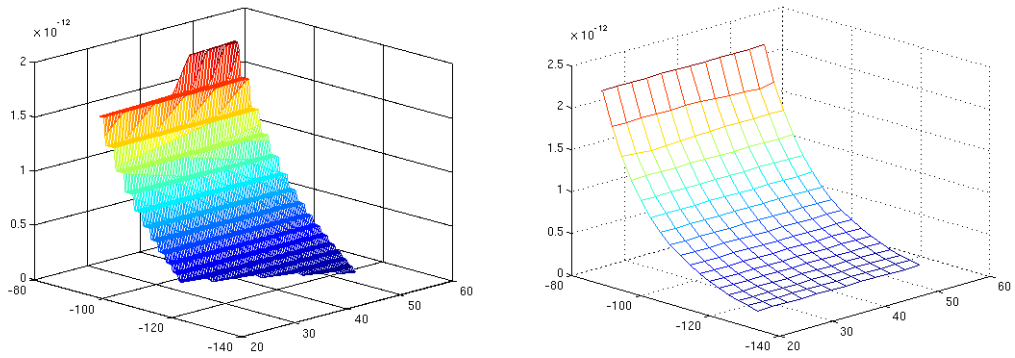

Figure 4.5: Monotone method. Left is downscaling and right is upscaling.

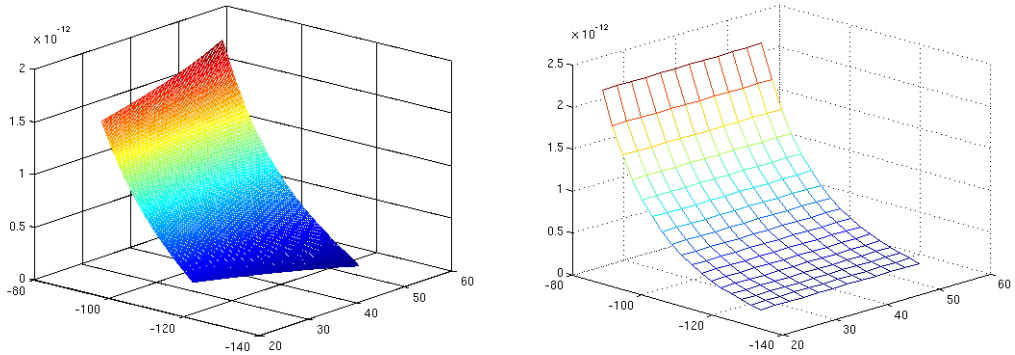

Figure 4.6: L_2 minimization method. Left is downscaling and right is upscaling.

Also, test with the function $f_3 = 2 + cos^2(\theta)cos(2\phi)$:

Table 4.9: Compare the result of $f_3 = 2 + cos^2(\theta)cos(2\phi)$

	Downscaling		Upscaling	
Method	max error	L_2error	max error	L_2error
area weighted	0.0249	0.0089	0.0044	0.0011
monotone	0.0243	0.0088	0.0040	0.0010
L_2 minimization	0.0011	3.4856e-04	9.3304e-04	2.3761e-04

Pictures of the output of function f_3 are as follows:

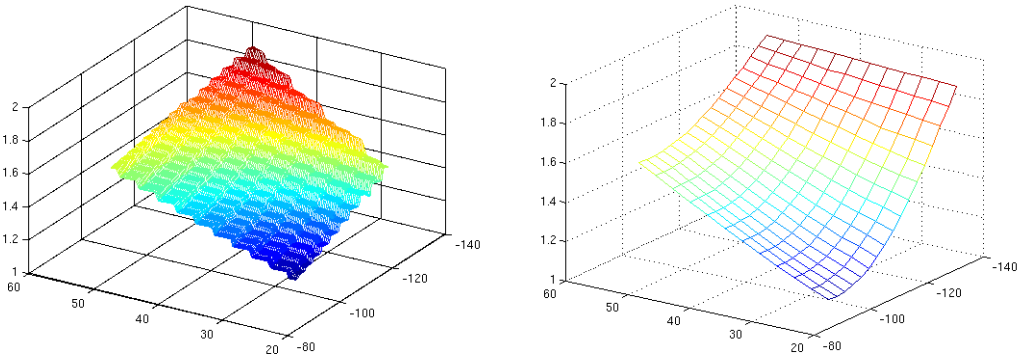

Figure 4.7: Area weighted method. Left is downscaling and right is upscaling.

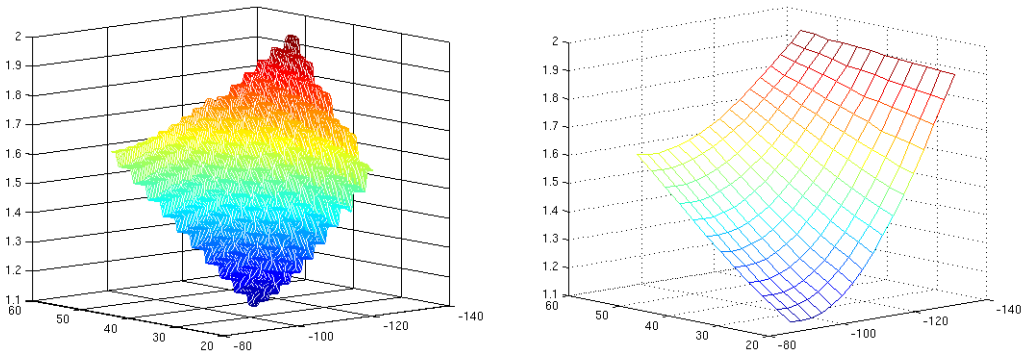

Figure 4.8: Monotone method. Left is downscaling and right is upscaling.

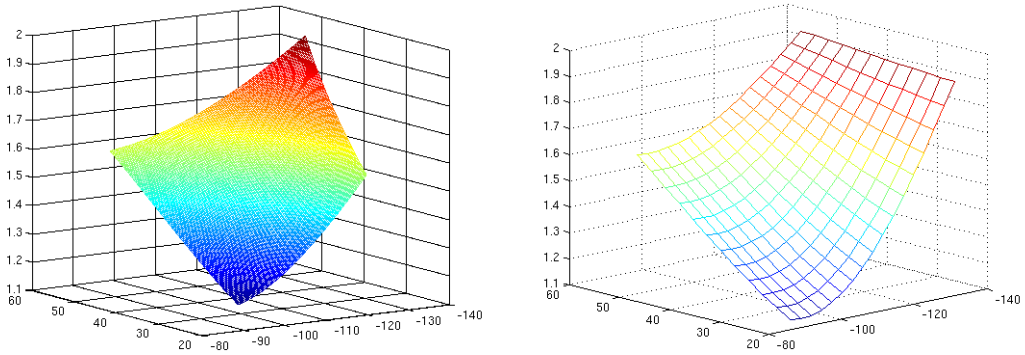

Figure 4.9: L_2 minimization method. Left is downscaling and right is upscaling.

In addition, test the function $f_4 = 2 + sin^{16}(2\theta)cos(16\phi)$:

Table 4.10: Compare the result of $f_4 = 2 + sin^{16}(2\theta)cos(16\phi)$

	Downscaling		Upscaling	
Method	max error	L_2error	max error	L_2error
area weighted	0.1953	0.0459	0.0328	0.0064
monotone	0.1608	0.0361	0.0229	0.0051
L_2 minimization	0.0810	0.0137	0.0086	0.0007

And corresponding pictures of function f_4 are:

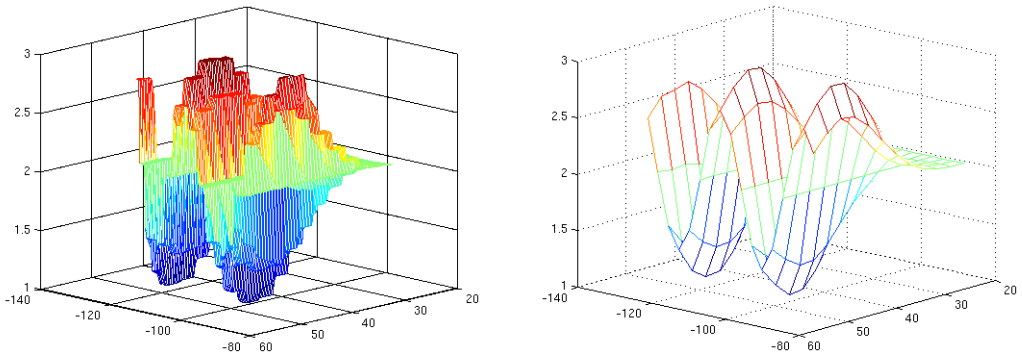

Figure 4.10: Area weighted method. Left is downscaling and right is upscaling.

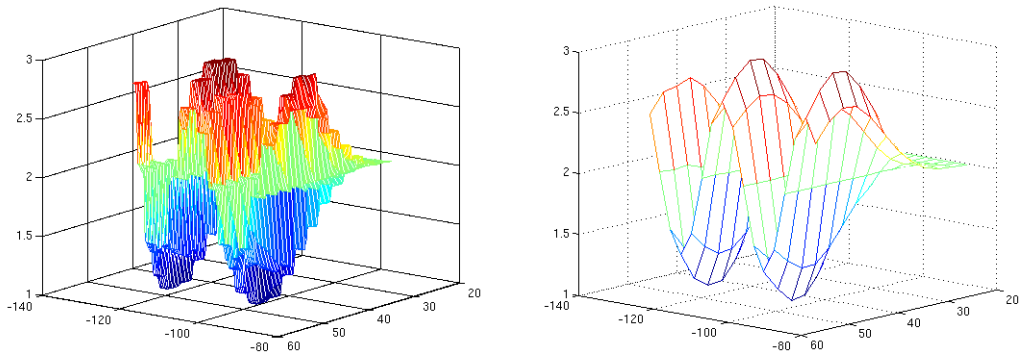

Figure 4.11: Monotone method. Left is downscaling and right is upscaling.

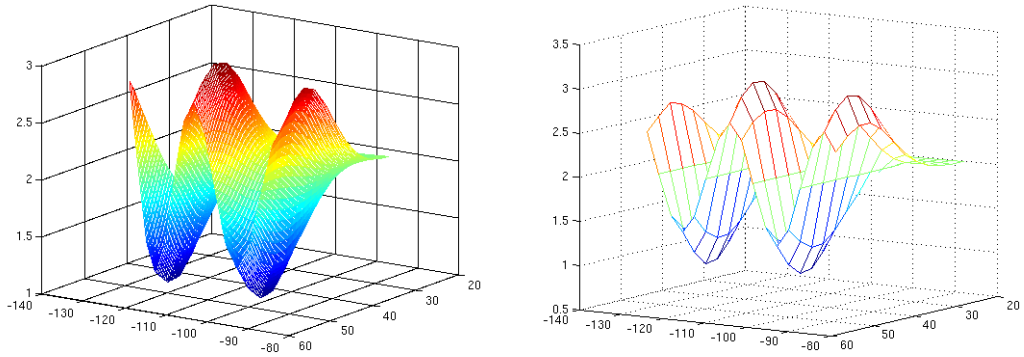

Figure 4.12: L_2 minimization method. Left is downscaling and right is upscaling.

More importantly, our method need to be verified by using real field. Here we choose water vapor, which has small values in general and has very big gradient at some points. The following pictures shows the result of remapping water vapor between CAM and WRF:

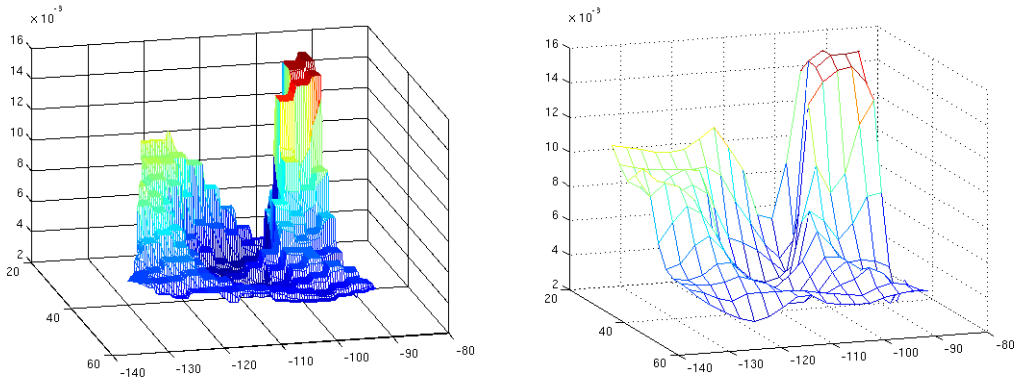

Figure 4.13: Area weighted method: Left is downscaling and right is upscaling.

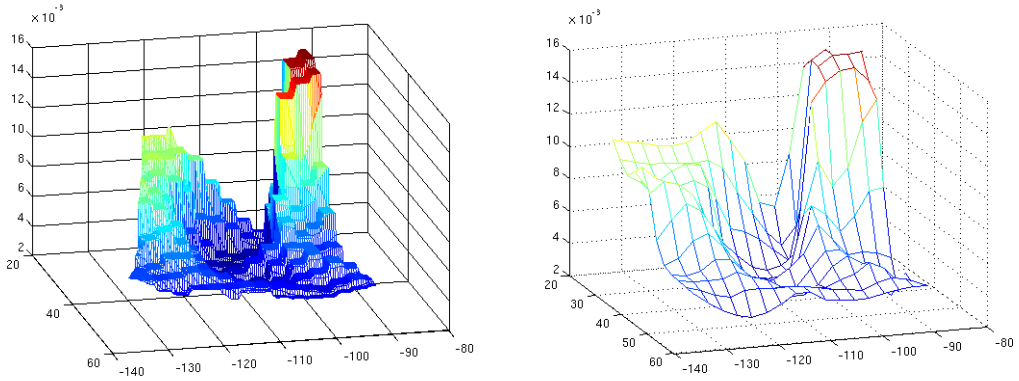

Figure 4.14: Monotone method: Left is downscaling and right is upscaling.

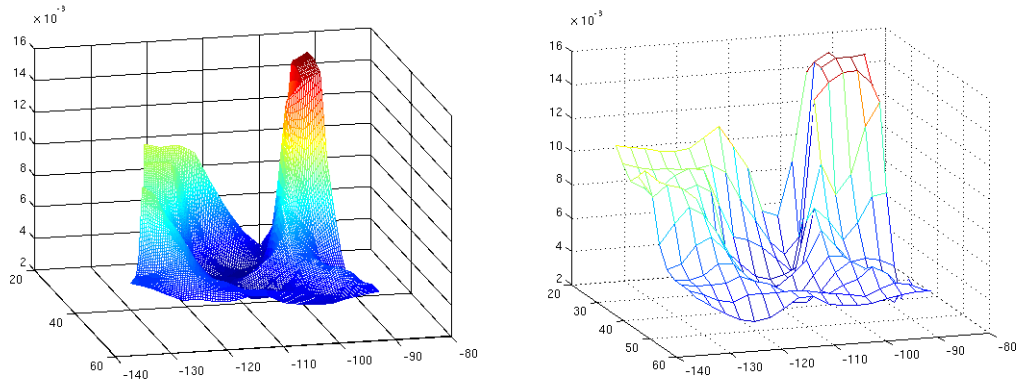

Figure 4.15: L_2 minimization method: Left is downscaling and right is upscaling.

4.3 Multiple Steps Remapping

Furthermore, we test multiple steps remapping between CAM and WRF with the above three methods. We transfer the data for 1000 steps, first of which is downscaling from CAM to WRF, and then transfer the data back from WRF to CAM, after that we will transfer data from CAM to WRF again. We have tested with function $f = 1$ and the real field water vapor. We compare the output CAM's value after 1000 steps with the original CAM's value, and get the errors as follows:

Table 4.11: Compare the result of $f = 1$

Method	Relative maximum error	Relative L_2 norm error
area weighted method	0.0130	0.0080
monotone method	0.0016	3.0162e-4
L_2 minimization method	1.7006e-09	2.6250e-10

The following pictures shows the results for water vapor after 1000 steps:

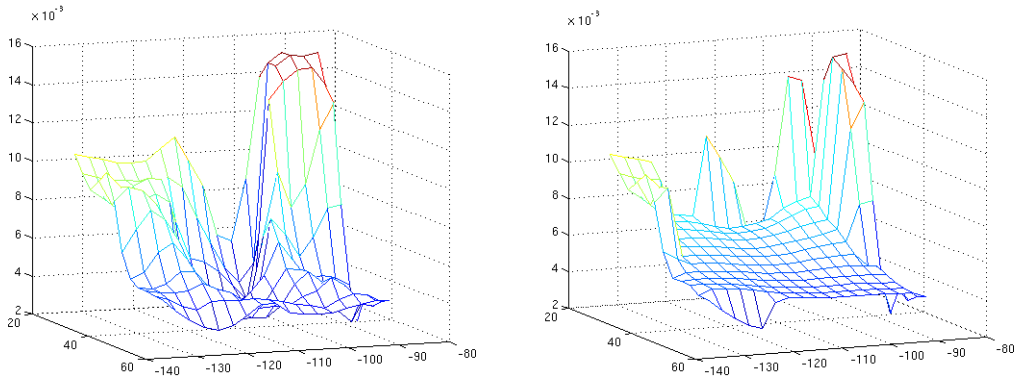

Figure 4.16: Left to right: (a). original water vapor in CAM's domain; (b). after 1000 steps transfer with area weighted method

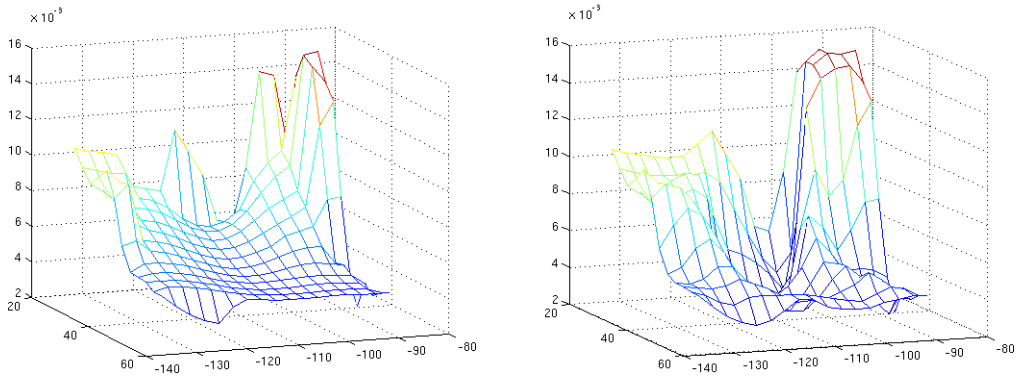

Figure 4.17: Left to right: (a). 1000 steps transfer with 2nd order monotonic method; (b). 1000 steps transfer with L_2 minimization method

4.4 Analysis of Results

According to the results of our tests, L_2 minimization method has higher order convergent rate than area weighted method and monotone method in accuracy. Because WRF's mesh is a much finer mesh than CAM's mesh, then data transfer from WRF

to CAM is more accurate than that from CAM to WRF. In multiple steps transfer, L_2 minimization method preserves the shape quite well, while area-weighted method and monotone method lose information in the middle domain.

Chapter 5

Conclusion and Future Research Directions

Among the conventional methods, high-order interpolation schemes are accurate but not strictly conservative, area weighted method and 2nd order conservative method (P. Johns, 1999) satisfy physical conservation but not that accurate. The L_2 minimization method can achieve both accuracy and physical conservation. We have compared our method with area weighted method and 2nd order conservative method, and the results are encouraging. Common-refinement based discretization have obvious advantages compared with other existing methods for repeated data transfer, because it is accurate, robust and time-efficient. Based on the original common-refinement method (X. Jiao, 2004) which is only suitable for meshes with linear elements and matching boundaries, we have implemented an extended method which can perfectly handle nonlinear issue and non-matching boundaries. In addition, we have also fixed the issue of different coordinate systems used by CAM and WRF. Moreover, we have extended the original face-to-face intersection so that it is able to compute the intersection between sectors and rectangles.

For the technical aspect, the interface of our algorithm is consistent with many

existing numerical methods, so that it is easy to integrate other available methods into our package. For the use of comparison, we have integrated area weighted method and 2nd order conservative method into our package. And further integration with more methods would be quite straightforward.

In the next step, we can implement a smoothing process from coarse mesh to fine mesh. In addition, we can further explore the issue of positivity preservation. Moreover, we can impose the physical conservation constrained by PDEs. Furthermore, there are other applications of common refinement method, so we can compare the results among these applications. Besides, we can extend current 2-D method to 3-D so that volumetric data can be transferred.

Bibliography

[1] P. B. Bochev and M. D. Gunzburger. *Least-Squares Finite Element Methods*. Springer, New York.

[2] S. F. Bockman. Generalizing the formula for areas of polygons to moments. *The American Mathematical Monthly*, 96:131–132, 1989.

[3] W. D. Collins, P. J. Rasch, B. A. Boville, J. J. Hack, J. R. McCaa, D. L. Williamson, J. T. Kiehl, B. Briegleb, C. Bitz, S.-J. Lin, M. Zhang, and Y. Dai. *Description of the NCAR Community Atmosphere Model (CAM 3.0)*. National Center For Atmospheric Research, Boulder, Colorado, June 2004.

[4] G. R. Cowper. Gaussian quadrature formulas for triangles. *International Journal for Numerical Methods in Engineering*, 7:405–408, 1973.

[5] B. Doty. *The Grid Analysis and Display System GrADS*, September 1995.

[6] J. K. Dukowicz and J. W. Kodis. Accurate conservative remapping (rezoning) for arbitrary lagrangian-eulerian computations. *SIAM Journal on Scientific and Statistical Computing*, 8(3):305–321, 1987.

[7] D. A. Dunavant. High degree efficient symmetrical gaussian quadrature rules for the triangle. *International Journal for Numerical Methods in Engineering*, 21:1129–1148, 1985.

[8] C. Farhat, M. Lesoinnea, and P. LeTallecb. Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: Momentum and energy conservation, optimal discretization and application to aeroelasticity. *Computer methods in applied mechanics and engineerlng*, 157:95–114, 1998.

[9] Y.-R. Guo and S. Chen. *Terrain and Land Use for the Fifth-Generation Penn State/NCAR Mesoscale Modeling System (MM5): Program TERRAIN*. National Center For Atmospheric Research, Boulder, Colorado, January 1994.

[10] M. T. Heath. *Scientific Computing: An Introductory Surve*. McGraw-Hill, New York, second edition, 2002.

[11] X. Jiao, H. Edelsbrunner, and M. T. Heath. Mesh association: formulation and algorithms. In *8th Int. Meshing Roundtable*, pages 75–82, South Lake Taho, CA, 1999.

[12] X. Jiao and M. T. Heath. Common-refinement-based data transfer between non-matching meshes in multiphysics simulations. *International Journal Numerical Methods in Engineering*, 61:2402–2427, 2004.

[13] X. Jiao and M. T. Heath. Overlaying surface meshes, part i: algorithms. *International Journal on Computational Geometry and Applications*, 14:379–402, 2004.

[14] X. Jiao and M. T. Heath. Overlaying surface meshes, part ii: topology preservation and feature matching. *International Journal on Computational Geometry and Applications*, 14:403–419, 2004.

[15] X. Jiao and D. Wang. Reconstructing high-order surfaces for meshing. In *19th International Meshing Roundtable*, pages 143–160, Chattanooga, Tennessee, 2010.

[16] X. Jiao and H. Zha. Consistent computation of first- and second-order differential quantities for surface meshes. In *2008 ACM symposium on Solid and physical modeling*, pages 159–170, New York, NY, 2008.

[17] X. Jiao, G. Zheng, P. Alexander, M. Campbell, O. Lawlor, J. Norris, A. Haselbacher, and M. T. Heath. A system integration framework for coupled multiphysics simulations. *Engineering with Computers*, 22, 2006.

[18] P. W. Jones. *A User's Guide for SCRIP: A Spherical Coordinate Remapping and Interpolation Package*. Los Alamos National Laboratory, 1.4 edition.

[19] P. W. Jones. First- and second-order conservative remapping schemes for grids in spherical coordinates. *Monthly Weather Review*, 127:2204–2210, 1999.

[20] P. H. Lauritzen and R. D. Nair. Monotone and conservative cascade remapping between spherical grids(cars): Regular latitude-longitude and cubed sphere grids. *Monthly Weather Review*, 136:1416–1432, 2008.

[21] J. R. McCaa, M. Rothstein, B. E. Eaton, J. M. Rosinski, E. Kluzek, and M. Vertenstein. *User's Guide to the NCAR Community Atmosphere Model (CAM 3.0)*. National Center For Atmospheric Research, Boulder, Colorado, June 2004.

[22] National Center For Atmospheric Research, Boulder, Colorado. *ARW Version 3 Modeling System User's Guide*, April 2010.

[23] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor and Francis, 1980.

[24] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, New York, 2000.

[25] J. D. Ramshaw. Conservative rezoning algorithm for generalized two-dimensional meshes. *Journal of Computational Physics*, 59:193–199, 1985.

[26] C. Sagastizabal and G. Vige. *How to interface Fortran with Matlab*, 1995.

[27] U. Schulzweida, L. Kornblueh, and R. Quast. *CDO User's Guide*, 1.4.1 edition, December 2009.

[28] W. C. Skamarock, J. B. Klemp, J. Dudhia, D. O. Gill, D. M. Barker, M. G. Duda, X.-Y. Huang, W. Wang, and J. G. Powers. *A Description of the Advanced Research WRF Version 3*. National Center For Atmospheric Research, Boulder, Colorado, June 2008.

[29] J. P. Snyder. *Map projections – A Working Manual*. United States Government Printing Office, Washington, 1987.

[30] L. White and A. Adcroft. A high-order finite volume remapping scheme for nonuniform grids: The piecewise quartic method (pqm). *Journal of Computational Physics*, 227, 2008.

[31] G. Wolberg and I. Alfy. Monotonic cubic spline interpolation. In *Computer Graphics International, 1999*, pages 188–195, Canmore, Alta., Canada, 1999.

[32] L. Zhang, T. Cui, and H. Liu. A set of symmetric quadrature rules on triangles and tetrahedra. *Journal of Computational Mathematics*, 27:89–96, 2009.