Modeling Chemical Transport with the Spectral Element Method

Samuel Loomis (NCSU) Cynthia Chan (CUHK)

Mentors: John Drake (UTK) Joshua Fu (UTK) Kwai L. Wong (UTK/ORNL)

Joint Institute for Computational Sciences

Overview

The Finite Element Method (FEM) is a popular method for solving differential equations be dividing the domain into small regions and approximating function values on those elements separately. Until recently, it has not been believed to be locally conservative.

Chemical transports are typically done using the Finite Volume Method (FVM), which is explicitly locally conservative.

However, climate models such as CESM and the HOMME equations use the FEM to simulate meteorological phenomena on the globe. To allow for a more complete model which includes atmospheric interactions with chemicals, it is desirable to also use the same method for both climate and chemical transport.

Mark Taylor[1] has shown relatively recently that the Spectral Element Method, a type of FEM, is explicitly locally conservative. It is also simpler to solve than most FEM problems, because the mass matrix is diagonal and easy to invert.

Our goal is to test the advantage of using SEM to model chemical transport in the form of the equation:

$$\frac{\partial u^{\alpha}}{\partial t} = D\nabla^2 u^{\alpha} + R^{\alpha}(u)$$

Ultimately we seek to integrate SEM chemical transport models with CESM. Our plan is to take two simultaneous paths, writing a serial code to test the 1D model and a parallel code to generalize to higher dimensions.

[1] Mark A. Taylor and Aime Fournier. A compatible and conservative spectral element method on unstructured grids. Journal of Computational Physics, 229(17):5879 – 5895, 2010.

Introduction to the Serial Code

We construct the serial code base on the chemical equation $Cl_2 \rightleftharpoons Cl + Cl$. The mathematical model on the population of [Cl_2] and [Cl] is **:** $\frac{\partial u_{\alpha}}{\partial t} = d_{\alpha}\frac{\partial^2 u_{\alpha}}{\partial x^2} + R(U)$

The serial code is construct to solve such a 1-dimensional continuous time-dependent differential equation, based on the programming language C.

Mathematical Scheme of the Serial Code

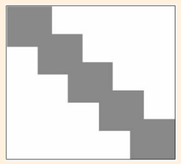
Formulate the equation

- Continuous Galerkin Method
- Spectral Element Method
- Gauss-Lobatto quadrature

Solve the equation

- Euler Backward Method
- Newton Method

Structure of the Jacobian Matrix in Newton Method



As the Jacobian Matrix can be with huge size, it is a good property that the structure of the matrix is banded. Hence, we use lapack to solve the linear algebra equation J(△x)=-F(x).

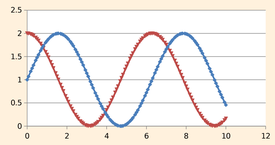
Example of Running the Serial Code to Solve the Convection-Diffusion Equation

Initialize the differential equation

Based on the chemical equation, input the differential equation to be solved as follows.

$$\frac{\partial[Cl_2]}{\partial t} = d_1\frac{\partial^2[Cl_2]}{\partial x^2} - k_1[Cl_2] + \frac{1}{2}[Cl]^2$$
$$\frac{\partial[Cl]}{\partial t} = d_2\frac{\partial^2[Cl]}{\partial x^2} + 2k_1[Cl_2] - [Cl]^2$$

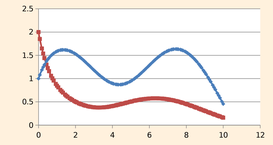
Input the initial value



Input the value of [Cl] and [Cl_2] at the nodal points at time T= 0. Here we choose [0,10] as the domain.

Output the solution

The serial code can give us the value of [Cl] and [Cl_2] on the domain [0,10] at time T.



This graph shows the value of [Cl] and [Cl2] at T=10. Moreover, we have use 40 element and 121 nodal points in this example.

Parallel Code Structure

The parallel code is a pair of Fortran modules designed to (1) store data about the mesh, fields, and equation and (2) integrate the fields forward in time. The two modules are Homer and Hesiod.

HESIOD (complete, testing)

Hesiod contains the data types *mesh*, *fields*, *fn_ptr*, and *equation*. Among other things, these types store information on the mesh structure, initial values of the fields, and the differential equation to be solved. Hesiod also calculates mesh quantities such as the mass and derivative matrices.

HOMER (in progress)

Homer contains the subroutines which will perform the time integration. Homer must complete the following tasks:

1. Send data in each element from root processor to that element's processor
2. In each processor, integrate an implicit Euler step using splitting and Newton methods
3. Return data to root processor to perform weighted sums and enforce continuity
4. Repeat process until complete

Current Results and Future Progress

Results

Though Homer is still in progress, we can test the accuracy of Hesiod by comparing the calculated mass and derivative matrices of the parallel and serial code.

Example: Diagonal Mass Matrix Values

Serial	*Parallel*	*Error (%)*
0.0104165	0.0104167	0.0019
0.0520835	0.0520833	0.0004
0.0520835	0.0520833	0.0004
0.0104165	0.0104167	0.0019

Similar accuracy is seen in the derivative matrices.

Future Progress

Homer still requires work on removing bugs in the parallel nature of the code. Once this is done, it will be progressively tested on the following:

1. Explicit Euler evolution
2. Explicit Euler with splitting
3. Implicit Euler with splitting