Discontinuous Galerkin Sparse Grids Method for Maxwell's Equations

Oak Ridge National Laboratory
The University of Tennessee Knoxville
香港中文大學 The Chinese University of Hong Kong
NSF
JICS Joint Institute for Computational Sciences

Student: Tianyang Wang (CUHK)

Mentors: Dr. Lin Mu, Dr. David L. Green, Dr. Ed D'Azevedo, Dr. Kwai Wong

Introduction

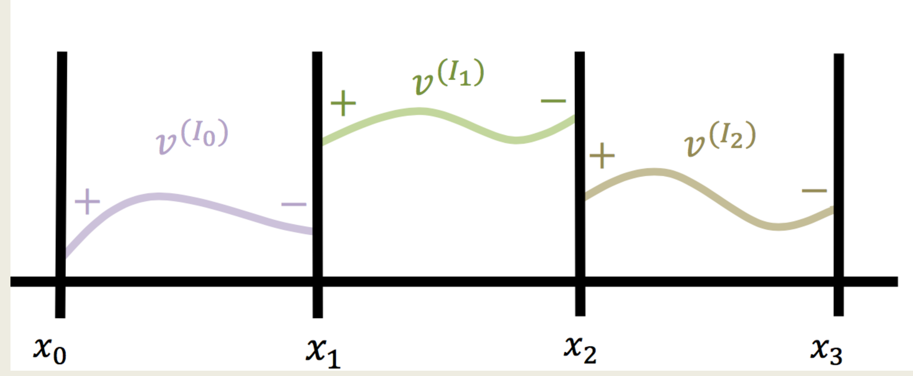
The main idea of my project is to use discontinuous Galerkin method to solve the Maxwell's equations, which allows the parallelization of computation and furthermore the implementation in HPC.

Here we use the sparse grids method in order to reduce the computational cost, especially in high dimensions. And we will test the cases for different choices of numerical fluxes, including central flux, alternating flux, and up-wind flux. The accuracy of order, L-2 error and CFL number will be the conditions for us to make the appropriate choice for numerical flux.

In addition, at first we use the explicit time advanced method—3rd order Runge Kutta method. This method will have strict condition for the choice of CFL number, for example, CFL number should be less than 0.01 for Lev 9 in up-wind scheme. This very small CFL number will have a high requirements for the high order computational cost. Hence it is important for us to develop the implicit time advanced method, in which we are able to have a much larger CFL number with acceptable error condition. But this time the computation of the inverse matrix will also be a matter. Preconditioner method is supposed to be used in this case.

DG Method

The DG method will use basis functions that are discontinuous in the boundary of each grid. And we use the set of basis functions to approximate the target function that we want. Then we are able to get the numerical solution after a series of mathematical formulation and computation. Here the jump condition in each boundary is considered to be very important, and our numerical fluxes are the choices of this jump condition.



x_j^+: start of the "left" interval I_j

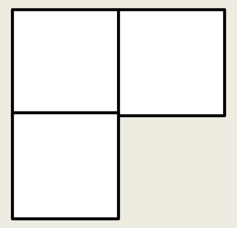
x_j^-: start of the "right" interval I_{j-1}

Sparse Grids

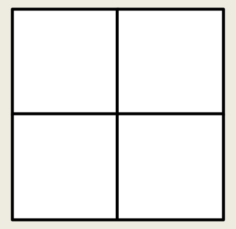
The main idea of sparse grids method is to use different levels in each dimension. Briefly we can understand in this way, full grids method requires the level in each dimension should be less than or equal to, while sparse grids method requires the sum of levels in each dimension should be less than or equal to N. Here we denote k as the degree, N as the maximum level and d as the dimension.

$$\mathbf{V}_N^k = \bigoplus_{\substack{|\mathbf{l}|_\infty \le N \\ \mathbf{l} \in \mathbb{N}_0^d}} \mathbf{W}_{\mathbf{l}}^k.$$

$$\hat{\mathbf{V}}_N^k := \bigoplus_{\substack{|\mathbf{l}|_1 \le N \\ \mathbf{l} \in \mathbb{N}_0^d}} \mathbf{W}_{\mathbf{l}}^k.$$



Lev=2 Dim=2
Sparse grids



Lev=2 Dim=2
Full grids

The degrees of freedom of sparse grid space scales as

$$\mathcal{O}((k+1)^d 2^N N^{d-1})$$

which is significantly less than that of traditional space with

$$\mathcal{O}((2^N(k+1))^d)$$

This is the key for computational savings and reduction in high dimensions.

Weak Formulation

$$\int_K \partial_t \boldsymbol{B}_h \cdot \boldsymbol{V}\, d\boldsymbol{x} = -\int_K \boldsymbol{E}_h \cdot \nabla \times \boldsymbol{V}\, d\boldsymbol{x} - \int_{\partial K} \widehat{\boldsymbol{n} \times \boldsymbol{E}_h} \cdot \boldsymbol{V}\, ds$$

$$\int_K \epsilon\mu \partial_t \boldsymbol{E}_h \cdot \boldsymbol{V}\, d\boldsymbol{x} = \int_K \boldsymbol{B}_h \cdot \nabla \times \boldsymbol{U}\, d\boldsymbol{x} + \int_{\partial K} \widehat{\boldsymbol{n} \times \boldsymbol{B}_h} \cdot \boldsymbol{U}\, ds - \int_K \mu \boldsymbol{J} \cdot \boldsymbol{U}\, d\boldsymbol{x}$$

Numerical Flux

$$central\ flux: \quad \hat{\boldsymbol{E}}_h = \frac{1}{2}(\boldsymbol{E}_h{}^+ + \boldsymbol{E}_h{}^-),\ \hat{\boldsymbol{B}}_h = \frac{1}{2}(\boldsymbol{B}_h{}^+ + \boldsymbol{B}_h{}^-)$$

$$alternating\ flux: \quad \hat{\boldsymbol{E}}_h = \boldsymbol{E}_h{}^+,\ \hat{\boldsymbol{B}}_h = \boldsymbol{B}_h{}^-\ \ or\ \ \hat{\boldsymbol{E}}_h = \boldsymbol{E}_h{}^-,\ \hat{\boldsymbol{B}}_h = \boldsymbol{B}_h{}^+$$

$$upwind\ flux: \quad \hat{\boldsymbol{E}}_h = \{\boldsymbol{E}_h\} + \frac{1}{2}[\boldsymbol{B}_h]_\tau,\ \hat{\boldsymbol{B}}_h = \{\boldsymbol{B}_h\} - \frac{1}{2}[\boldsymbol{E}_h]_\tau$$

Explicit Scheme

Table 1: The accuracy of order for central flux

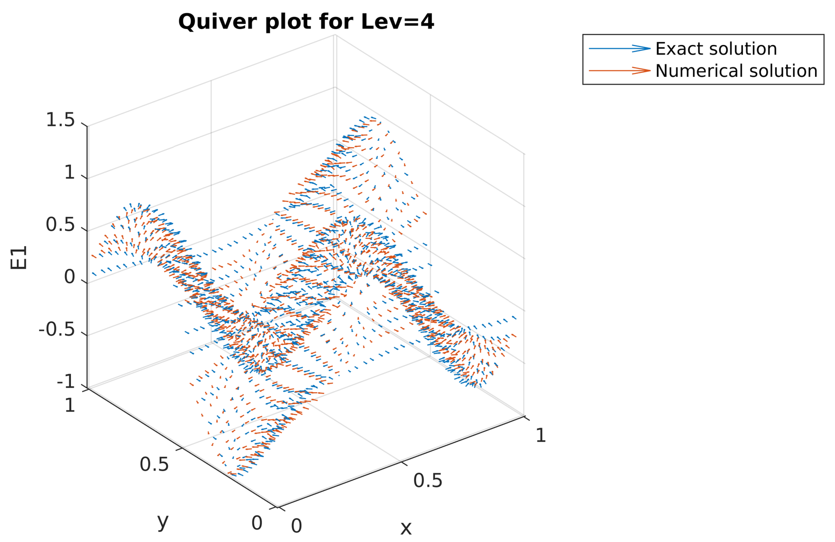
deg	level	$l-\infty$ error	order	$l-2$ error	order
2	3	3.477E-03		9.847E-03	
2	4	1.999E-03	0.799	9.020E-03	0.127
2	5	1.336E-03	0.581	7.767E-03	0.216
2	6	5.509E-04	1.279	5.017E-03	0.631
2	7	2.882E-04	0.935	2.781E-03	0.851
2	8	1.116E-04	1.369	1.463E-03	0.927
2	9	4.034E-05	1.468	7.478E-04	0.968

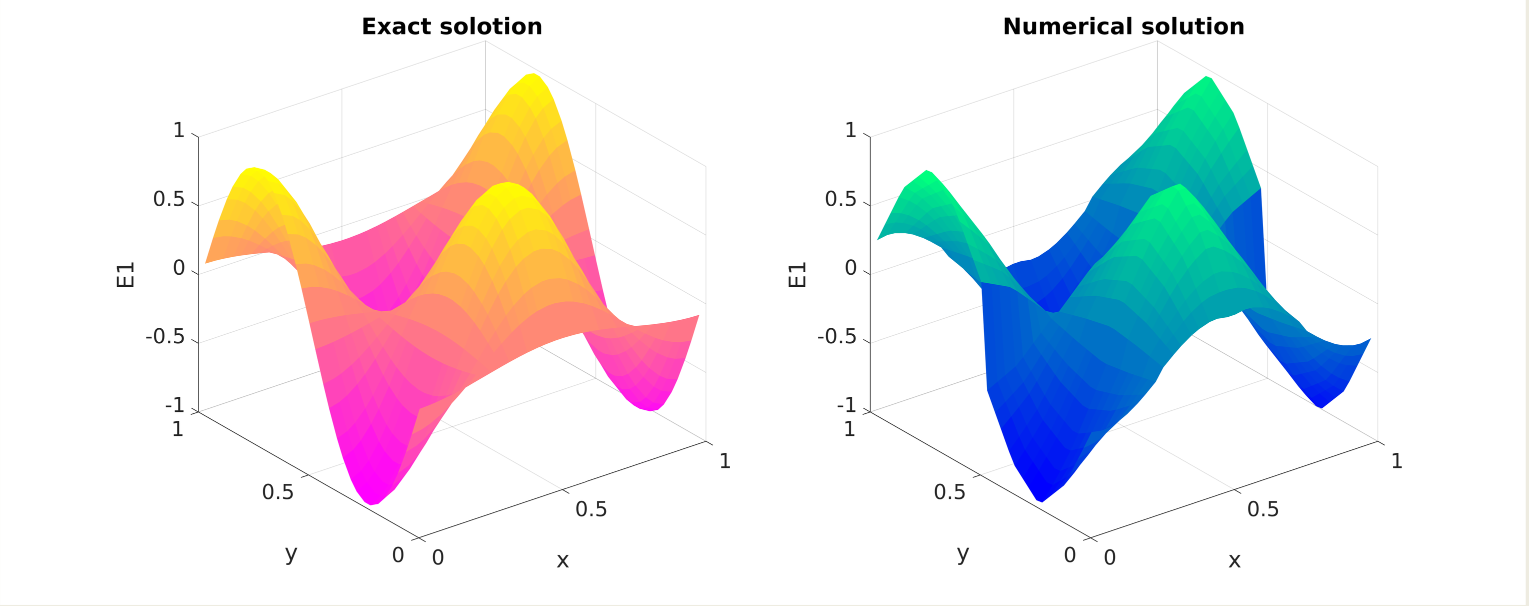
Table 2: The accuracy of order for alternating flux

deg	level	$l-\infty$ error	order	$l-2$ error	order
2	3	4.948E-03		2.195E-02	
2	4	3.293E-03	0.587	1.529E-02	0.521
2	5	1.332E-03	1.306	9.257E-03	0.724
2	6	7.036E-04	0.921	5.265E-03	0.814
2	7	2.626E-04	1.422	2.623E-03	1.006
2	8	7.931E-05	1.727	1.182E-03	1.150
2	9	1.864E-05	2.089	4.334E-04	1.448

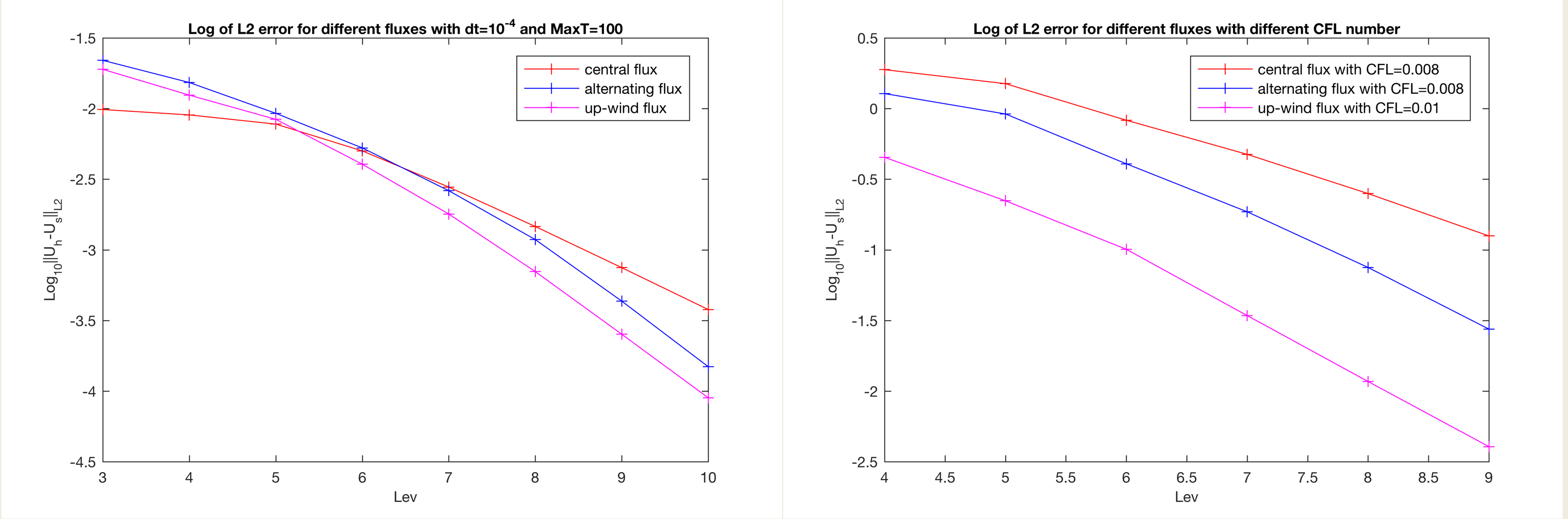
Table 3: The accuracy of order for upwind flux

deg	level	$l-\infty$ error	order	$l-2$ error	order
2	3	5.638E-03		1.896E-02	
2	4	2.859E-03	0.980	1.245E-02	6.067
2	5	1.553E-03	1.017	1.181E-03	5.691
2	6	4.444E-04	1.670	4.035E-03	1.056
2	7	1.839E-04	1.273	1.793E-03	1.171
2	8	4.926E-05	1.900	7.015E-04	1.354
2	9	1.227E-05	2.005	2.526E-04	1.474

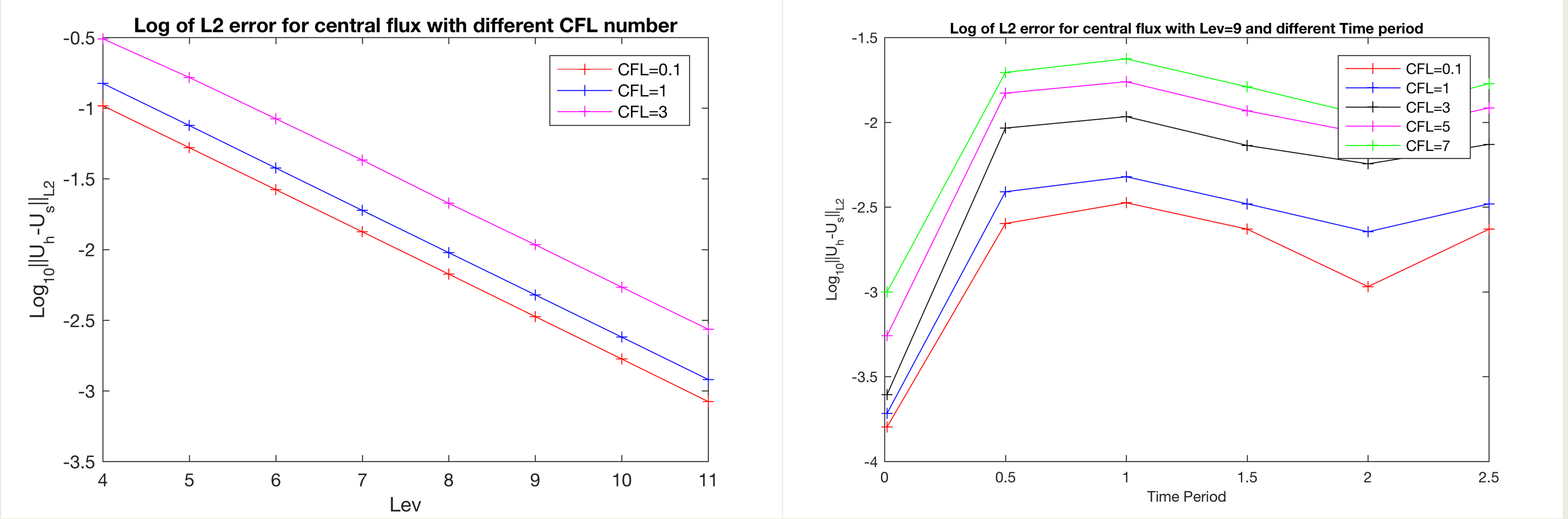




It is easy for us to see that alternating flux and up-wind flux have better convergence rate than central flux. For degree=2, the accuracy of order for central flux is a half less than those of other two fluxes.



Implicit Scheme



For 1D implicit scheme, we just use the backward Euler time method and test the central flux, then we are able to see that the CFL number can be much larger than what we need in explicit scheme, which means we are able to have a significant reduction of computational cost.

In the future, we would like to test different numerical fluxes for implicit scheme, and use preconditioner method to reduce the matrix inverse calculation. Furthermore, we also want to develop different implicit time advanced method, such as 3rd Implicit Runge Kutta method , as well as semi-implicit scheme.

Acknowledgements

This project was sponsored by the National Science Foundation through Research Experience for Undergraduates (REU) award, with additional support from the Joint Institute of Computational Sciences at University of Tennessee Knoxville. This project used allocations from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by the National Science Foundation. In addition, the computing work was also performed on technical workstations donated by the BP High Performance Computing Team.