REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE Technical Report	3. DATES COVERED (From - To) -

4. TITLE AND SUBTITLE Progress Report 5	5a. CONTRACT NUMBER
	5b. GRANT NUMBER W911NF-11-C-0243
	5c. PROGRAM ELEMENT NUMBER 665502
6. AUTHORS Tian Xiao, EE Boost Inc.	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAMES AND ADDRESSES EE Boost Inc EE Boost Inc 618 Powers Ferry RD Cary, NC 27519 -	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Army Research Office P.O. Box 12211 Research Triangle Park, NC 27709-2211	10. SPONSOR/MONITOR'S ACRONYM(S) ARO 11. SPONSOR/MONITOR'S REPORT NUMBER(S) 60619-CS-ST1.5

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
We propose to develop a hybrid wave-based method combining two efficient methods: the ECT and PSTD, for acoustic simulation in large urban environments. In this report, we have developed an efficient scheme for our simulation method to treat arbitrarily complex geometries. Our method can correctly capture the complex shapes to produce accurate results. The complex geometries are usually represented by some CAD formats. We have developed a mesh generation to convert the STL format to the mesh that our method can directly use. We have also

15. SUBJECT TERMS
STL, Mesh Generation, Vectorization, Embedded Hybridization

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Tian Xiao 19b. TELEPHONE NUMBER 919-439-4847
UU	UU	UU	UU		

Report Title

Progress Report 5

ABSTRACT

We propose to develop a hybrid wave-based method combining two efficient methods: the ECT and PSTD, for acoustic simulation in large urban environments. In this report, we have developed an efficient scheme for our simulation method to treat arbitrarily complex geometries. Our method can correctly capture the complex shapes to produce accurate results. The complex geometries are usually represented by some CAD formats. We have developed a mesh generation to convert the STL format to the mesh that our method can directly use. We have also explored some techniques to improve our computational performance. The use of vectorization makes our code almost 3 times faster even with the use of one core. The use of blocked range of Intel TBB gives another speedup of 2.5 to 3 times with the use of 4 cores. We also show an efficient hybridization strategy to significantly improve the efficiency of our hybrid method. In the conventional strategy, the total computational domain is partitioned into boundary subdomains with ECT methods and internal subdomains with PSTD methods. This strategy requires a lot of interfacing for any pairs of adjacent subdomains. In the new strategy, we regard the whole domain as just a single PSTD domain, but with embedded boundaries inside. Boundary techniques, such as ECT or others, can be then applied locally to correct the field near the boundaries.

A Fast Wave-Based Hybrid Method for Interactive Acoustic Simulation in Large and Complex Environments

Progress Report 5

EE Boost Inc.

01/24/2012

Contractor #: W911NF-11-C-0243

CLIN: 0001AE

Period of Performance: 12/25/2011 – 01/24/2012

Table of Contents

1. Introduction

We propose to develop a new hybrid wave-based method for fast and accurate acoustic simulation in large and complex urban environments. The hybrid method combines two efficient methods: the enlarged cell technique (ECT) and the pseudo-spectral time-domain (PSTD) method.

In the previous progress report, we have implemented the Chebyshev PSTD method with the use of FFT. The use of FFT significantly improves the computational efficiency by reducing the operation complexity from $O(N^2)$ to $O(N \log N)$. Moreover, the FFT is easy to parallel. We have explored several hardware acceleration techniques for FFT. Our preliminary results show that the code can be accelerated by several hundreds to thousands times for typical problems. We have also developed a interfacing technique to communicate adjacent sub-domains. A wave-based technique, the Riemann solver interfacing is applied.

In this report, we will show our progress in the fifth reporting period in developing, improving, and testing the hybrid wave-based method. We have completed the following work:

1. **Acceptance of Arbitrarily Complex Geometries**. A real acoustic environment can be very complex. The objects inside can have arbitrary shapes. Our simulation method can correctly capture the complex shapes to produce accurate results. This requires our software to have the ability to accept the user's arbitrary geometries. STL is a simple and popular format to represent complex geometries in CAD community. In **Section 2**, we will show our method to accept this format from users and convert it to our required mesh.

2. **Performance Speedup by Some Techniques**. To improve the computational performance, some techniques are explored for our simulation method. The use

of vectorization makes our code faster even with the use of one core. For a problem of $100 \times 100 \times 100$ cells, it takes less than 1 second to run 1000 time steps with ECT using just a single core of i7. Moreover, we explored the blocked range in Intel TBB. A speedup of 2.5 to 3 times can be obtained with the use of 4 cores. These speedups with the previous discussed GPU acceleration open the possibility to perform real-time or near real-time simulations for large-scale problems, such as the one in an urban environment of hundreds of meters wide at typical frequencies. This part of work is shown in **Section 3**.

3. **Embedded Hybridization**. We happened to find that our method can be significantly improved by changing our hybridization strategy. Before, we split the total computational domain into boundary subdomains with ECT methods and cuboid subdomains with PSTD methods. This strategy requires a lot of interfacing for any pairs of adjacent subdomains. In fact, we can regard the whole domain as just a single PSTD domain, but with embedded boundaries inside. Boundary techniques, such as ECT or others, can be then applied locally to correct the field near the boundaries. This new strategy is briefly discussed in **Section 4**.

In the end, a summary is made, which includes a brief plan for the next reporting period.

2. Acceptance of Arbitrarily Complex Geometries

A real acoustic environment can be very complex. The complexity comes from the fact that the objects inside can have arbitrary curved shapes. These complex boundaries of objects are really a challenge to not only the computation accuracy but also the computation speed, which has been observed and investigated by many researchers. Our simulation method can correctly capture the positions of the complex geometries to produce accurate results. However, the complex geometries are usually represented by some CAD formats, among which the STL (stereolithography) format is simple and widely-used. This requires our software to be able to accept, such as the popular STL format, and convert it to the mesh that we can directly use in our method. In this section, we will discuss the work on mesh generation of arbitrarily complex geometries of STL format.

2.1 STL Format

We choose STL format because it is simple and widely used. It is also standard for the rapid prototyping industry and almost all the CAD software accepts it and can produce it according to users' requirements. The STL format describes only the surface geometry of a 3D object. The object surface is triangulated into a number of triangles. Each triangle is represented by its vertices ordered by the right-hand rule and its outer unit normal in a 3D Cartesian coordinate system. The STL format has both ASCII and binary representations. The ASCII representation is easier to read, but the binary representation is more common, since they are more compact.

2.2 Mesh Generation

The STL format can represent arbitrarily complex geometries, but it cannot be directly used in our method. Our method uses a Cartesian grid, which is made up of three kinds of grid lines along x, y, and z directions, respectively. The boundaries of the objects are captured by the positions of the intersections of the objects with the grid lines, which is illustrated in **Figure 1**.

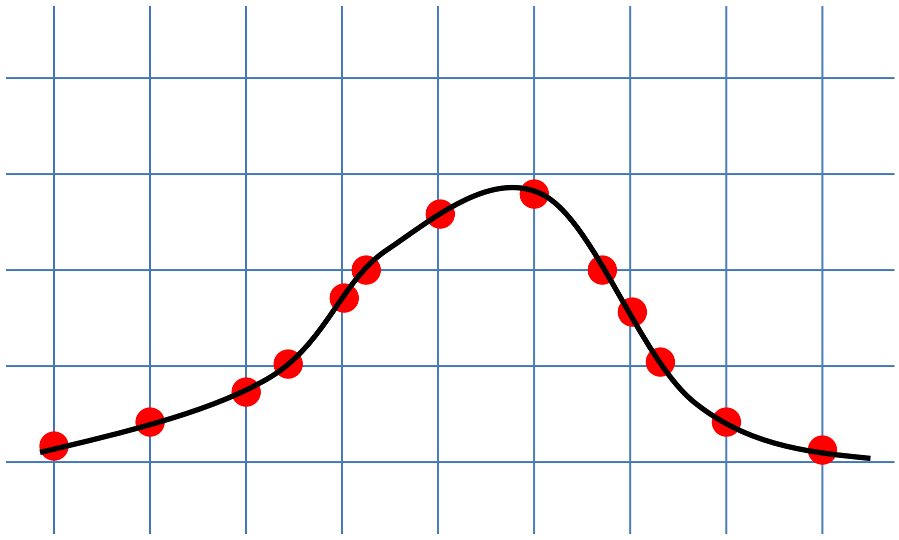

Figure 1: Illustration of the capture of complex geometries of objects in our method. A Cartesian grid is used in our method. The grid lines are denoted by the blue straight lines. The shape of a curved object is denoted by the black curved line. The intersections of the objects and the grid lines are correctly recorded, as shown by the red dots in the figure.

Please note that for simplicity, the conventional methods with Cartesian grids usually approximate the curved geometries at grid points, which will introduce large errors and thus require a denser grid, leading to a more expensive computation. Our method records the curved geometries at the actual positions even they are not at grid points. Thus it is not only more accurate bust also more efficient. However, to accurately record the positions, our method requires a more powerful mesh generation to correctly compute the intersections of grid lines with arbitrarily complex objects represented by such as STL format.

The STL format decomposes the surface of a complex geometry into a number of triangles. To get the intersection of objects and grid lines, we need to determine whether a ray of grid line intersects a triangle. If it is, we need to calculate the intersection positions. This problem is well studied in the community of computational graphics. We will adopt the fast, minimum storage ray-triangle intersection algorithm developed by Tomas Moller and Ben Trymbor [1] and modify it to our particular situations, where the rays are along x, y, or z directions.

Let us consider the calculation of the intersection of a triangle with a grid line along z direction. As shown in **Figure 2**, the grid line along z direction can be denoted by its x and y coordinates, i.e., $\mathbf{v} = (x, y)$. A triangle can be denoted by its 3 vertices: $\mathbf{u}_0 = (x_0, y_0, z_0)$, $\mathbf{u}_1 = (x_1, y_1, z_1)$, and $\mathbf{u}_2 = (x_2, y_2, z_2)$. The triangle is first projected to x-y plane. The projection is a triangle in a x-y plane with vertices: $\mathbf{v}_0 = (x_0, y_0)$, $\mathbf{v}_1 = (x_1, y_1)$, and $\mathbf{v}_2 = (x_2, y_2)$. We will firstly find the three simplex coordinates of the grid line or ray at (x, y) in the projection triangle. They are

$$\lambda_0 = S_0/S \tag{1}$$

$$\lambda_1 = S_1/S \tag{2}$$

$$\lambda_2 = S_2/S \tag{3}$$

where S_0 , S_1, and S_2 are triangle areas, as shown in **Figure 2**, and S is the sum of S_0 , S_1, and S_2. The triangle areas can be easily calculated by

$$S_0 = (\mathbf{v}_1 - \mathbf{v}) \times (\mathbf{v}_2 - \mathbf{v})/2 \tag{4}$$

$$S_1 = (\mathbf{v}_2 - \mathbf{v}) \times (\mathbf{v}_0 - \mathbf{v})/2 \quad (5)$$

$$S_2 = (\mathbf{v}_0 - \mathbf{v}) \times (\mathbf{v}_1 - \mathbf{v})/2 \quad (6)$$

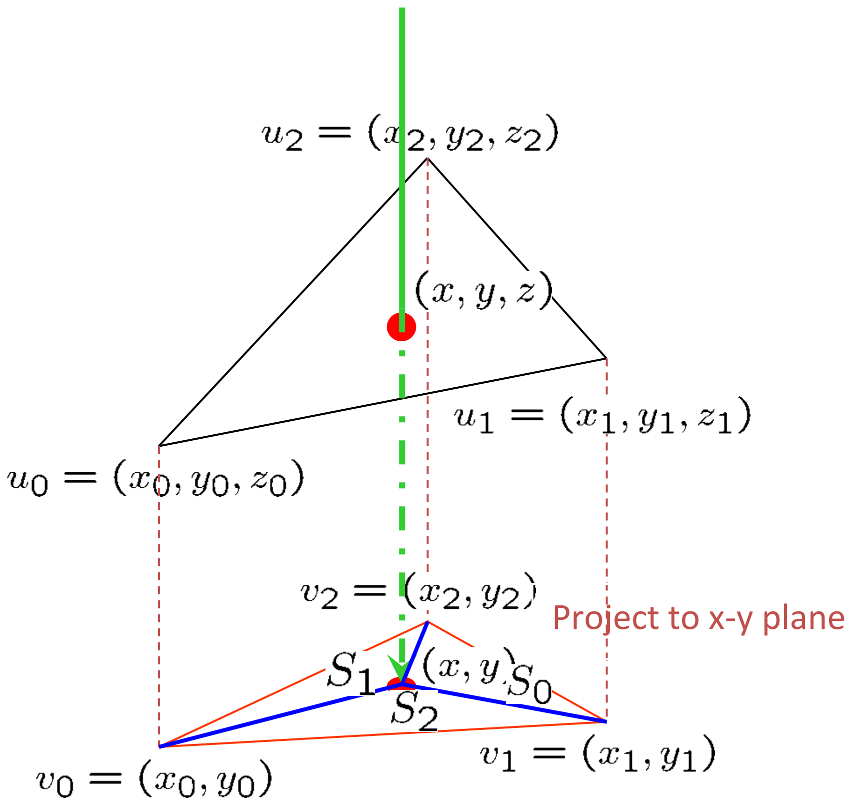

Figure 2: Ray-triangle intersection calculation. A ray along z direction at (x, y) (green line) is intersected with a triangle with vertices u_1, u_2, and u_3. Their projection on x-y plane is used to calculate the intersection position.

To determine if the grid line is intersected with the triangle, we look at the range of the three simplex coordinates. If one of them is out of the range of $[0, 1]$, then they are not intersected, otherwise they are intersected. Due to the linearity of the triangle, the position of intersection is then (x, y, z) with

$$z = \lambda_0 z_0 + \lambda_1 z_1 + \lambda_2 z_2 \qquad (7)$$

With the ray-triangle intersection algorithm, we can now make the geometry mesh that our method can directly use from the STL format. We calculate all the intersections of the grid lines with the surfaced triangles of the objects with STL format. Usually for large problems, the number of grid lines and surface triangles are huge. To find all the intersections, we may need to loop all the grid lines and also all the triangles. The complexity is the multiplication of the two large numbers: $O(M \times N)$, when M is the number of grid lines, while N is the number of surface triangles. Here we develop a very efficient scheme to calculate all the intersections by computation complexity of only $O(N)$ by taking advantage of the structure of grid lines. The scheme is shown as below:

1. Loop for all the surface triangles
2. For each triangle, find its bounding box first. Only the grid lines in the bounding box is possible to intersect with the triangle. This consideration extremely limits the number of our intersecting grid line candidates from M to just a few.
3. Determine if those candidate grid lines intersect with the triangle one by one. If it is, calculate the intersection position with the fast, minimum storage ray-triangle algorithm.

3. Performance Speedup

To improve the computational performance, some techniques are explored. They include the use of vectorization and blocked-range of Intel TBB.

3.1 Vectorization

Vectorization is a special case of parallelization. By default, a software program performs only one operation at a time on a single thread. In the vectorization, it performs multiple operations simultaneously still in a single thread by processing a vector implementation on multiple pairs of operands at once.

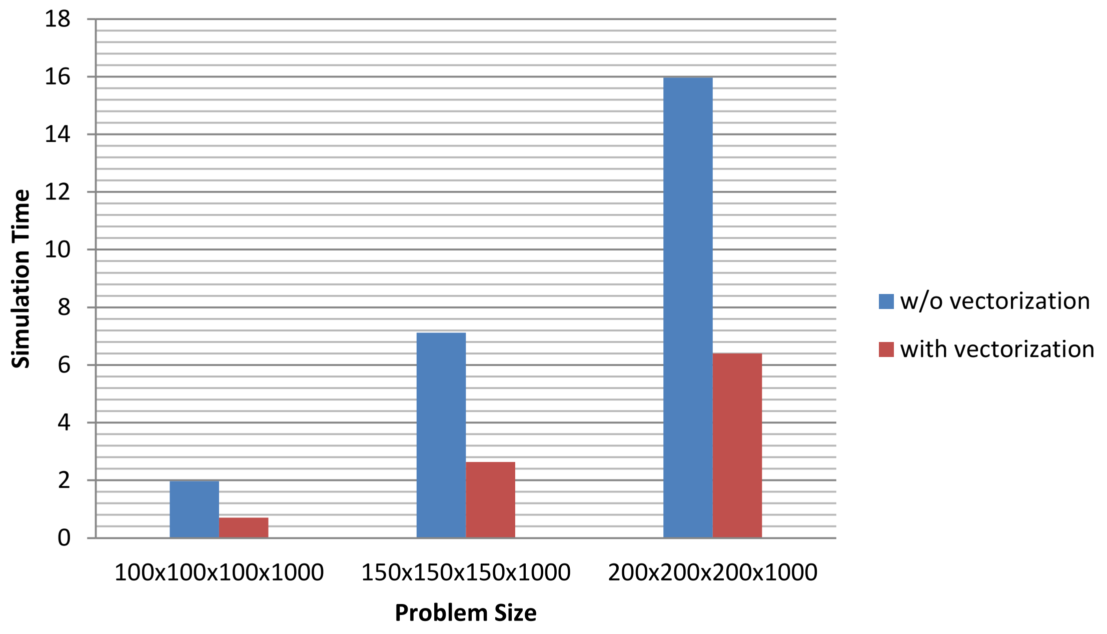

Figure 3: Computation performance comparison with using and without using vectorization for different sizes of problems with 100, 150, and 200 cells along each directions, respectively and with 1000 time steps of simulation.

We compare the performances of the computations with using and without using the vectorization. To use the vectorization, we add a compiler indicator “#pragma ivdep” before the kernel loop in our code to instruct the compiler to ignore assumed vector dependencies, which is true for our simulation method. Several sizes of problems are tested. They have 100, 150, and 200 cells, respectively, along each Cartesian direction. All are simulated with 1000 time steps. Their simulation times are shown in **Figure 3**. It is observed that the vectorization speeds up the simulation by almost 3 times. And for a problem of $100 \times 100 \times 100$ cells, it takes less than 1 second to run 1000 time steps with ECT using just a single core of i7.

3.2 Blocked Range of Intel TBB

Moreover, we explored the use of the blocked range of Intel TBB for multi-core computation. A blocked range models the range concept. It represents a half-open range that can be recursively split for parallel computation with multiple cores. In our preliminary testing, a speedup of 2.5 to 3 times can be obtained with the use of all the 4 cores of i-7.

All these speedups with parallel techniques of vectorization, blocked range of Intel TBB, and GPU acceleration (discussed in previous reports) open the possibility to perform real-time or near real-time simulations for large-scale problems, such as the one in an urban environment of hundreds of meters wide at typical frequencies.

4. Embedded Hybridization

The conventional strategy of hybrid method is to partition the whole domain space into regions or sub-domains of several categories according to their characteristics. The best proper method is then applied to each region. Interfacing schemes are employed to communicate adjacent regions. Using our hybrid method as an example, this conventional strategy is illustrated in **Figure 4**.

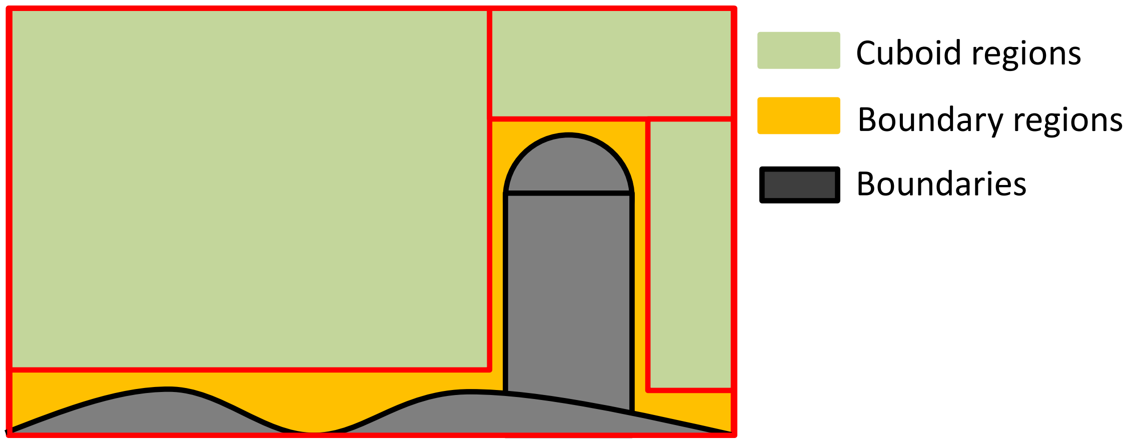

Figure 4: An illustration of the conventional hybridization strategy. The total domain is divided into cuboid regions where PSTD is used and boundary regions where ECT is used.

This strategy has some disadvantages:

1. Interfacing is required for any pair of adjacent sub-domains. If the geometry is very complex, the partition will have lots of sub-domains. Therefore, lot of interfacing is required, which will add significant burden to the computation.

2. Complex geometries may make the partition quite challenging. To find the best partition needs lots of intelligence and experience for large and complex problems.

3. Due to the geometry complexity, the partition may include too many sub-domains, which will reduce the efficiency of the numerical method.

Now we are trying to put forward a new hybridization strategy to overcome those disadvantages.

4.1 Embedding Strategy

In the new hybridization, we will embed the boundaries into a cuboid domain, as shown in **Figure 5**. This embedding technique, if it works, will significantly improve the computation efficiency, making it as fast as the computation of a single domain because the boundary layer (the yellow part in **Figure 5**) occupies only a very small portion of the volume.

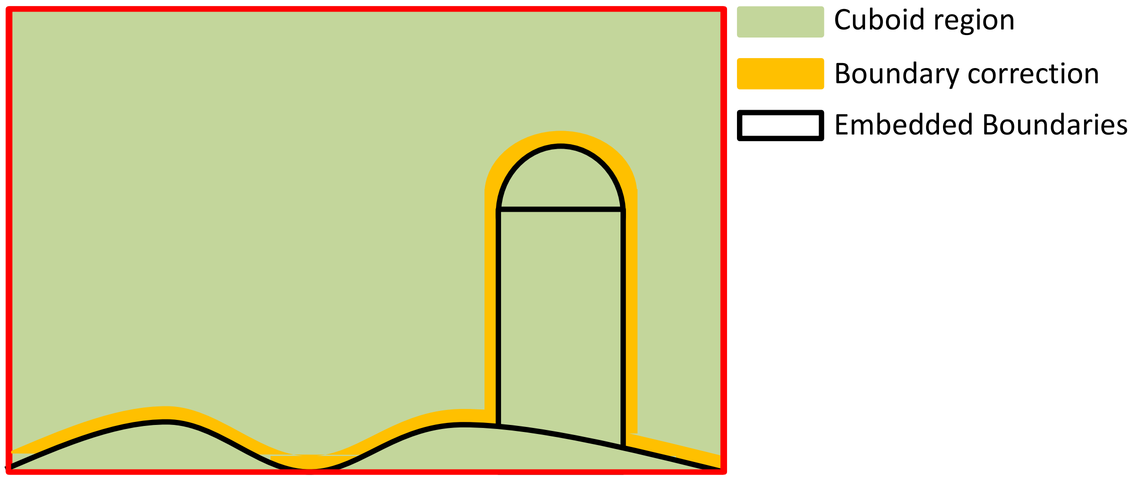

Figure 5: Boundaries are embedded in a cuboid domain. Efficient method, such as the PSTD is applied to the whole cuboid domain regardless of internal boundaries. Boundary correction techniques, such as ECT, are then applied to get correct results of fields around the boundaries.

In the embedding strategy, we regard the total computational domain as a single domain and a very efficient method is applied to it, such as the PSTD or other high-order method, regardless of internal boundaries. The ignorance of boundaries will bring large errors to the fields around the boundaries. To correct the fields around the boundaries, we then use ECT or ECT-like method, which can accurately capture the positions of the boundaries, to produce accurate results there. Please note that we only use the ECT around the boundary layers. The small volume of boundary layers makes the computational time of boundary correction negligible. Thus, the embedding hybrid method can accurately deal with the boundaries with almost the same efficiency as the one for single domain without internal boundaries. It also removes the partition challenges and interfacing burden in the conventional hybridization strategy.

4.2 Boundary Correction

We use the ECT scheme for boundary correction in the embedding hybrid method. The whole domain is updated by the PSTD method. If the two meshes of the PSTD and ECT are the same, the boundary correction will be not hard. Fortunately, both the PSTD and ECT use the Cartesian grid mesh. Thus, there should be no difficulty to do the boundary correction with the ECT method.

4.3 Runge-Kutta Scheme

For spectral or high-order method, to reduce the dispersion error, high-order numerical integrations are used to advance field with time. The popular Runge-Kutta schemes are applied. The Runge-Kutta schmes advance fields by a time step with multiple stages or sub-steps. The general Runge-Kutta schemes require storing fields at every stage, since all the old stage values are used to update field at a new stage. To save the computational memory, we employ low-storage Runge-Kutta methods which require only two stage storage. The scheme for low storage methods with I stages is given by

$$s_0 = t_n,\ r_0 = 0,\ q_0 = u^n \tag{8}$$

$$s_i = t_n + c_i h, \qquad r_i = a_i r_i + hf(s_{i-1}, q_{i-1}), \qquad q_i = q_{i-1} + b_i r_i, \quad i = 1,2,\dots,I \tag{9}$$

$$u^{n+1} = q_I \tag{10}$$

Here residue r is introduced and a_i, b_i, and c_i are specified by specific Runge-Kutta method. For example, the 4th order, five-stage, low-storage Runge-Kutta method [2] has

$$a = \left[0, \frac{-567301805773}{1357537059087}, \frac{-2404267990393}{2016746695238}, \frac{-3550918686646}{2091501179385}, \frac{-1275806237668}{842570457699}\right] \tag{11}$$

$$b = \left[\frac{1432997174477}{9575080441755}, \frac{5161836677717}{13612068292357}, \frac{1720146321549}{2090206949498}, \frac{3134564353537}{4481467310338}, \frac{2277821191437}{14882151754819}\right] \tag{12}$$

$$c = \left[\frac{1432997174477}{9575080441755}, \frac{2526269341429}{6820363962896}, \frac{2006345519317}{3224310063776}, \frac{2802321613138}{2924317926251}, 1\right] \tag{13}$$

5. Summary

In this report, we have developed an efficient scheme for our simulation method to treat arbitrarily complex geometries. Our method can correctly capture the complex shapes to produce accurate results. The complex geometries are usually represented by some CAD formats. We have developed a mesh generation to convert the STL format to the mesh that our method can directly use. We have also explored some techniques to improve our computational performance. The use of vectorization makes our code almost 3 times faster even with the use of one core. For a problem of $100 \times 100 \times 100$ cells, it takes less than 1 second to run 1000 time steps with ECT using just a single core of i7. The use of blocked range of Intel TBB gives another speedup of 2.5 to 3 times with the use of 4 cores. We also show an efficient hybridization strategy to significantly improve the efficiency of our hybrid method. In the conventional strategy, the total computational domain is partitioned into boundary subdomains with ECT methods and internal subdomains with PSTD methods. This strategy requires a lot of interfacing for any pairs of adjacent subdomains. In the new strategy, we regard the whole domain as just a single PSTD domain, but with embedded boundaries inside. Boundary techniques, such as ECT or others, can be then applied locally to correct the field near the boundaries.

With the development of mesh generation for arbitrarily complex geometries of the popular STL format, our method is ready to simulate real applications in large and complex urban environments. In the final reporting period, we will keep improving our simulation method, test its accuracy and performance with large and real applications, analyze its feasibility for real-time or near real-time simulation in large urban environments, compare it with other methods, and point out the future work we need to do.

Reference

[1] Tomax Moller and Ben Trumbore, "Fast, minimum storage ray-triangle intersection," *Journal of Graphics Tools*, vol. 2, no. 1, pp. 21-28, 1997.

[2] J. H. Williamson, "Low-storage Runge-Kutta schemes," *J. Comput. Phys.*, vol. 35, pp. 48-56, 1980.