**Josh Bevan**

*Development of a Parallelizable Implementation of the Vorticity Transport Model for Vortex Dominated Flows*

**Abstract:**

There are many physical systems that are dominated or strongly influenced by vortex flows such as helicopters or wind turbine farms. It is important to resolve the effects the vortex flows have on the system, but it is quite challenging computationally. In order to realize accurate but practical simulation tools a combination of vortex advection modeling with practical parallelization is a key motivator.

The Vorticity Transport Model (VTM) modifies the Navier-Stokes equation and solves for the unsteady transport equation for vorticity. It then discretizes and solves this equation along with the Biot-Savart relationship to conservatively advect vorticity throughout a domain. A significant component of the algorithm involves domain decomposition using an octree structure. The octree provides for recursive grid refinement as well as a useful context for implementation of a Fast Multipole Method (FMM) for the purposes of calculating the Biot-Savart relationship. It is proposed that the octree domain structuring provides a ready-made framework for the implementation of a parallelizable computational approach that should provide significant scalability on cluster and grid computational platforms.

**Introduction:**

The Vorticity Transport Model (VTM) [1] recasts the Navier-Stokes equation into vorticity-velocity form

 (1)

Where the velocity field is determined from the vorticity field following the Biot-Savart law [2]

 (2)

In the VTM Eqn. 2 is solved using a Cartesian fast multipole method (FMM) [2]. The FMM solves the n-body problem posed by Eqn. 2 much faster than a naïve direct solution by grouping multiple long-range interactions into one approximated bulk interaction [4]. The VTM is performed on a structured Cartesian grid that is adaptively refined to higher spatial definition as needed using an octree [2].

The VTM has been successfully used to simulate real-world phenomena [8]. However, it has faced computational challenges in satisfactorily performing adequate spatial refinement. In simulations of helicopter vortex wake by Brown and Kelly [9] the finite computational resources limited the minimum computational cell size leading to lacking spatial resolution and over-prediction of the size of the tip vortices.

The structure and solution methodology inherent in the VTM lends itself quite nicely to solution via parallel means on a cluster or grid computing platform. The octree domain decomposition provides a powerful heuristic for parallelization [7]. Additionally, work has been done with the FMM and parallelization there as well [6].

The proposed research seeks to leverage the previous work done in parallelization of the various methods employed in the VTM and modify them to work within its framework. The expectation is that this will enable significant scalability on scientific computing platforms and enable greater available fidelity in the resulting simulations.

**Technical Approach:**

VTM typically approaches defining the domain as including the entirety of the vorticity produced and advected. This avoids dealing with boundary conditions on the domain. As the vorticity evolves the domain could either be adaptively expanded, or an a priori estimate could be used to specify a static domain size that will satisfactorily enclose the system behavior for the scope of the time simulated. In areas where vorticity needs to be resolved the octree is refined, while in areas free of vorticity the octree is left unrefined; see Figure 1 as an example.

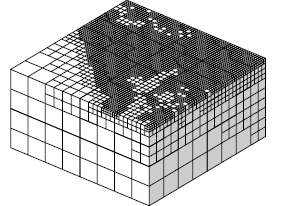


Figure 1: Adaptive grid refinement using an octree structure. [7]

An important component of efficient parallelism is domain decomposition that minimizes inter-process communication. In order to realize a FMM method that optimizes this the domain must be decomposed, and then partially reconstructed with ghost octants for non-local interactions during an upward traversal of the octree during calculation [5]; see Figure 2. A critical part of the implementation of the FMM for the VTM will require processor balancing to prevent local resource bottlenecks from slowing global execution. Some work has been done addressing this [6] by assigning computation weights to particular leaves and then repartioning to equalize the predicted load. A robust implementation will seek to both minimize communication while optimizing load balancing.

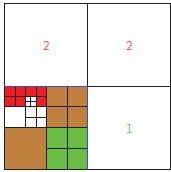


Figure 2: Domain decomposition and inter-process communication amongst "ghost" octants. [5]

The project primarily consists of three phases. First, basic algorithmic development needs to occur to create potential parallel solutions to the individual pieces of the VTM. Prototyping of un-optimized code will occur. In the second phase the prototype code will be re-written and optimized for the target language/platform. In the third phase, performance testing of the final code will occur. While performance may have had small or even mid-scale testing for development purposes in phases one and two, phase three will seek to test at full-scale.

Phase one will require several (2-3) algorithm developers to prototype code. The developers will need to have experience both in fluid mechanics, as well as CFD; particularly implementation of parallel codes.

Two potential development paths are available to pursue. Firstly, the current available code developed by Brown et al. could be used as a base and refactored to incorporate the likely extensive changes required. This may also pose potential legal/financial challenges depending on the nature of how the code/intellectual property is handled. Alternatively, development can start from the ground-up generating an entirely separate code-base. This may be more time consuming, but it will give additional flexibility. Other than human capital, the only real resources required will be workstations for code development.

Phase two will require several general purpose coders in addition to the personnel from phase one. The additional coders should have experience in parallel programming, high-performance computing, and ideally prior experience programming for the targeted hardware platforms/architecture that the code will be optimized for. Mid-scale testing to target important areas for optimization will require either rented time on a nearby cluster, or rented time on on-demand computing platforms (e.g. Amazon EC2)

Phase three will require only several of the original developers for testing purposes. Full-scale testing will be performed on rented time at a large computational facility or perhaps more easily on a very large Amazon EC2 cluster. The computational power of the cluster is unimportant the primary variable of interest is the number of nodes in the cluster or processors in the grid. The intention is to investigate the scalability of the final code.

**Schedule**

**Cost**

-Dev/ GP coder Salaries based on numbers from [11].  
-HPC Equipment based on [12].  
-IT services are for general maintenance and backup services for phase 1, with additional configuration cost allowances for HPC system made for phase 2.  
-AWS services from example [10], with additional 20% margin for Amazon technical help.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase:** | **1** | **2** | **3** | **Total** |
| **Devs** | $300,000 | $150,000 | $50,000 | $500,000 |
| **GP Coders** | - | $120,000 | $20,000 | $140,000 |
| **Equipment** | GP Comp: $5000 | HPC: $15,000 | - | $20,000 |
| **Services** | IT: $1,000 | IT: $5,000 | AWS: $60,000 | $56,000 |
|  |  |  |  | $716,000 |

**Bibliography**

[1] Brown RE. Rotor wake modeling for flight dynamic simulation of helicopters. AIAA Journal 2000; 38:57– 63.

[2] Brown RE, Line AJ. Efficient high-resolution wake modeling using the vorticity transport equation. AIAA Journal 2005; 43:1434–1443.

[3] Toro, E. F., “A Weighted Average Flux Method for Hyperbolic Conservation Laws,” Proceedings of the Royal Society of London, Series A: Mathematical and Physical Sciences, Vol. 423, No. 1864, 1989, pp. 401–418.

[4] Greengard, L., and Rokhlin, V., “A Fast Algorithm for Particle Simulations,” Journal of Computational Physics, Vol. 73, No. 1, 1987, pp. 325–348.

[5] Lashuk, I., et al., “A massively parallel adaptive fast-multipole method on heterogeneous architectures” Communications of the ACM, Vol. 55 Issue 5, May 2012 pp. 101-109.

[6] L. Ying, G. Biros, D. Zorin. “A Kernel-independent Adaptive Fast Multipole Method in Two and Three Dimensions.” Journal of Computational Physics, 196(2), 591-626, 2004.

[7] Bielak, J., et al. Parallel Octree-Based Finite Element Method for Large-Scale Earthquake Ground Motion Simulation, CMES, vol.10, no.2, pp.99-112, 2005

[8] Fletcher, T., Brown, R. Simulation of wind turbine wake interaction using

the vorticity transport model. Wind Energy 2010; 13:587–602.

[9] Kelly, M., Brown, R. Predicting the Wake Structure of the HART II rotor using the

Vorticity Transport Model. 34th European Rotorcraft Forum, 16th - 19th Sept 2008, Liverpool, England.

[10]  [Brodkin](http://arstechnica.com/author/jon-brodkin/), Jon. “18 hours, $33K, and 156,314 cores: Amazon cloud HPC hits a petaflop”

<http://arstechnica.com/information-technology/2013/11/18-hours-33k-and-156314-cores-amazon-cloud-hpc-hits-a-petaflop/>. Retrieved Nov.19, 2013.

[11] Glassdoor.com, various pages. <http://www.glassdoor.com/Salaries/algorithm-developer-salary-SRCH_KO0,19.htm>. <http://www.glassdoor.com/Salaries/hpc-salary-SRCH_KO0,3.htm>. Retrieved Nov.19, 2013.

[12]