A New Software and Hardware Parallelized Floating Random-Walk Algorithm for the Modified Helmholtz Equation Subject to Neumann and Mixed Boundary Conditions

Kausik Chatterjee[1], McCullen Sandora[1], Christopher Mitchell[1], Deian Stefan[1], Dave Nummey[1], and Jonathan Poggie[2]

[1]Department of Electrical Engineering
The Cooper Union for the Advancement of Science and Art, New York, NY 10003-7185, USA
chatte@cooper.edu, sandor@cooper.edu, mitche2@cooper.edu,
stefan@cooper.edu, nummey@cooper.edu

[2]The Air Force Research Laboratory, AFRL/RBAC,
Wright-Patterson Air Force Base, OH 45433-7512, USA
jonathan.poggie@wpafb.af.mil

Abstract– A new floating random-walk algorithm for the one-dimensional modified Helmholtz equation subject to Neumann and mixed boundary conditions problems is developed in this paper. Traditional floating random-walk algorithms for Neumann and mixed boundary condition problems have involved “reflecting boundaries” resulting in relatively large computational times. In a recent paper, we proposed the elimination of the use of reflecting boundaries through the use of novel Green’s functions that mimic the boundary conditions of the problem of interest. The methodology was validated by a solution of the one-dimensional Laplace’s equation. In this paper, we extend the methodology to the floating random-walk solution of the one-dimensional modified Helmholtz equation, and excellent agreement has been obtained between an analytical solution and floating random-walk results. The algorithm has been parallelized and a near linear rate of parallelization has been obtained with as many as thirty-two processors. These results have previously been published in [1]. In addition, a GPU implementation employing 4096 simultaneous threads displayed a similar near-linear parallelization gain and a one to two orders of magnitude improvement over the CPU implementation. An immediate application of this research is in the numerical solution of the electromagnetic diffusion equation in magnetically permeable and electrically conducting objects with applications in dielectrometry and magnetometry sensors that have the ability to detect sub-surface objects such as landmines. The ultimate goal of this research is the application of this methodology to the solution of aerodynamical flow problems.

Index Terms– Floating random-walk, Monte Carlo, modified Helmholtz equation, parallelizable algorithm, CUDA, GPU.

I. INTRODUCTION

The floating random-walk (FRW) method [2] is a statistical technique for the numerical solution of deterministic boundary value problems. It is a generalization of the Monte Carlo integration method [3], which is a statistical approach to estimating integrals, which, unlike many other techniques, is well-suited to evaluating multi-dimensional integrals. We will discuss one such method, “Sample Mean Monte Carlo” [3], and then demonstrate how the technique is modified to form the basis for the FRW method.

Consider a function $f(x)$ defined over the interval $a \le x \le b$. Our problem is to estimate the integral

$$\mathbf{I} = \int_a^b dx\, f(x). \tag{1}$$

In the event that the integral is improper, absolute convergence is assumed. We select an arbitrary probability density function $p(x)$, with a corresponding random variable ξ. We define another random variable η as

$$\eta = \frac{f(\xi)}{p(\xi)}. \tag{2}$$

The expectation value of the random variable η, written as $E(\eta)$, is equal to the integral $\mathbf{I}$, which can be expressed as

$$\mathbf{I} = E(\eta) = \int_a^b dx \left[\frac{f(x)}{p(x)} \right] p(x). \tag{3}$$

This integral can be evaluated by sampling the integrand with the help of a random-number generator, and averaging over a statistically significant number of samples. This approach is particularly suited to evaluating higher dimensional integrals, because the computational work of sampling the integrand does not increase substantially with the dimensionality of the integral. We will now describe how this Monte Carlo integration method can be generalized into the FRW method for the solution of boundary value problems.

We consider a differential equation, with a differential operator *L*,

$$L[U(\mathbf{r})] = f(\mathbf{r}), \tag{4}$$

where the solution *U*(**r**) is a function of the three-dimensional position vector **r**. The function *f*(**r**) is a source term. The Green's functions for (4) are the solutions of the differential equation

$$L[G(\mathbf{r} \,|\, \mathbf{r_o})] = \delta(\mathbf{r} - \mathbf{r_o}), \tag{5}$$

subject to specified boundary conditions. We assume that the operator *L* is of the Sturm-Liouville [4] form:

$$L = \nabla \cdot [p(\mathbf{r})\nabla] + q(\mathbf{r}), \tag{6}$$

where $p(\mathbf{r})$ and $q(\mathbf{r})$ are known functions of $\mathbf{r}$. Using Green's integral representation [4] *U*(**r**) can be written as

$$\begin{aligned} U(\mathbf{r_o}) &= \iiint_V dv G(\mathbf{r} \,|\, \mathbf{r_o}) f(\mathbf{r}) \\ &- \oint\!\!\oint_S [d\mathbf{s} \cdot \nabla_{\mathbf{r}} U(\mathbf{r})] p(\mathbf{r}) G(\mathbf{r} \,|\, \mathbf{r_o}) \\ &+ \oint\!\!\oint_S [d\mathbf{s} \cdot \nabla_{\mathbf{r}} G(\mathbf{r} \,|\, \mathbf{r_o})] p(\mathbf{r}) U(\mathbf{r}). \end{aligned} \tag{7}$$

The first term on the right hand side of (7) is a volume integral involving the source term in the entire volume *V* of interest. The second and third terms are vector surface integrals over the surface *S* enclosing *V*, where *d***s** is a vector whose magnitude is equal to that of an infinitesimally small area unit on the surface *S* and directed normally outward from the center of the area unit. The term *G*(**r**|**r₀**) is often referred to as the volumetric Green's function and the term $\nabla_{\mathbf{r}} G(\mathbf{r} \,|\, \mathbf{r_o})$ is called the surface Green's function. The second term corresponds to the Neumann [4] boundary condition, whereas the third term corresponds to the Dirichlet boundary condition [4]. In traditional FRW algorithms, homogeneous Dirichlet boundary conditions are imposed on the Green's function given by (5). As a result, the second integral in the right hand side of (7) goes to zero. To evaluate the solution to (4) at a particular point in the domain of interest, we consider [2] maximal spheres, cubes, or any geometrical object for which the solution to (5) is known. We then make random hops to the surface of that geometrical object based on any predefined probability density. The weights for such random hops are determined by sampling the remaining two integrands in (7). For example, in the case of a Dirichlet problem with no source term [i.e., $f(\mathbf{r}) = 0$], the contribution of the volume integral also goes to zero and the problem reduces to a Monte Carlo integration of an infinite-dimensional integral, as given by:

$$\begin{aligned} U(\mathbf{r_0}) &= \oint_{S_1} ds_1 K(\mathbf{r_0} \,|\, \mathbf{r_1}) \ldots \oint_{S_n} ds_n K(\mathbf{r_{n-1}} \,|\, \mathbf{r_n}) U(\mathbf{r_n}), \\ K(\mathbf{r_{n-1}} \,|\, \mathbf{r_n}) &= \left| \nabla_{\mathbf{r_n}} G(\mathbf{r_{n-1}} \,|\, \mathbf{r_n}) \right| \cos(\gamma_{n-1,n}), \end{aligned} \tag{8}$$

where $\gamma_{n-1,n}$ is the angle between $\nabla_{\mathbf{r_n}} G(\mathbf{r_{n-1}} \,|\, \mathbf{r_n})$ and $d\mathbf{s_n}$. The successive surface integrals in (8) relate to successive random hops across the problem domain and the weight factors of the form $K(\mathbf{r_{n-1}} \,|\, \mathbf{r_n})$ are derived from the third integral term on the right hand side of (7) that corresponds to the Dirichlet boundary condition. A particular random walk is terminated at the boundary, where the solution is known, and the samples of successive weight factors multiplied by the solution at the boundary yield a particular sample of the solution. A numerical solution of (4) is obtained by averaging over a statistically significant number of such samples.

The termination of the random walk becomes a problem for Neumann and mixed boundary condition problems where the solution is not known at all points of the domain boundary. In traditional random walk literature [5], these boundary conditions are formulated as partially "reflecting" as the random-walker has a chance of either being absorbed in the problem boundary or being thrown back into the problem domain. In a recent paper [6], we formulated a FRW algorithm for this particular problem where the reflection at problem boundaries was eliminated through the development of a Green's function whose boundary conditions mimicked the boundary conditions of the problem of interest. In this paper, we extend the methodology to the solution of the one-dimensional modified Helmholtz equation, subject to mixed boundary conditions.

II. THE NEW FORMULATION

Consider the equation

$$\frac{d^2U}{dx^2} = 0, \tag{9}$$

where U is the dependent variable of interest defined in the problem domain $0 \le x \le L.$ The boundary conditions imposed on this problem are $U(0) = \alpha$ and $U(L) = \beta.$ A traditional FRW algorithm for this problem will be based on a Green's function given by

$$\frac{d^2G}{dx^2} = \delta(x - x_0), \tag{10}$$

defined on a problem domain $-h \le x \le h,$ with homogeneous Dirichlet boundary conditions $G(-h \mid x_0) = 0$ and $G(+h \mid x_0) = 0.$ The solution to (10) in a zero-centered notation (i.e., $x_0 = 0$) is given by

$$G(x \mid 0) = \left\{ \begin{array}{l} \frac{1}{2}(x-h),\ x \ge 0 \\ -\frac{1}{2}(x+h),\ x \le 0 \end{array} \right\}. \tag{11}$$

Based on the 1D version of the Green's integral representation (7), the solution to (9) at the center of the one-dimensional problem domain can be written as

$$\begin{aligned} U(0) = &\left[U\frac{dG}{dx} - G\frac{dU}{dx} \right]_{x=h} \\ &- \left[U\frac{dG}{dx} - G\frac{dU}{dx} \right]_{x=-h}, \end{aligned} \tag{12}$$

where no specific boundary conditions have been imposed on the Green's function. Using the Green's function given by (11), (12) can be reduced to

$$U(0) = \frac{1}{2}U(h) + \frac{1}{2}U(-h). \tag{13}$$

Thus, the solution to (9) at the center of the problem domain $-h \le x \le h$ can be expressed in terms of the solution at the two end-points. In a traditional FRW algorithm, (13) is used to generate the random walks. The random walker either hops to the left or to the right with equal probability (without any restriction on the hop size) until it is absorbed at one of the boundaries. An estimate of the solution at any given point $x = x^*$ within the problem domain $0 \le x \le L$ is given by

$$U(x^*) = \frac{N_\alpha \alpha + N_\beta \beta}{N_\alpha + N_\beta}, \tag{14}$$

where the number of times the random walker hits the $x = 0$ and the $x = L$ boundary are N_α and N_β respectively. Now let us consider the solution of (9) defined on the problem domain $0 \le x \le L,$ but with the boundary conditions $U(0) = \alpha$ and $\left\{ dU/dx \right\}_{x=L} = \beta.$ It is obvious that a FRW scheme based on (13) will not find a reward at the $x = L$ boundary. The termination at this boundary is based on a finite-difference based representation of the Neumann boundary condition [5] and the random-walker is either absorbed or reflected back into the problem domain. If the random walker is reflected back in the problem domain, once again random walks are generated based on (13). This partial reflection at the boundary increases the computational time and as a result, Neumann and mixed boundary condition problems are considered difficult to be handled with the FRW method.

In a recent paper [6], we proposed a philosophically different approach for Neumann and mixed boundary condition problems and applied it to the problem given in (9). In our approach, the boundary conditions imposed on the Green's function mimic those of the problem of interest and as a result, the reflecting boundaries are converted to absorbing boundaries. In this paper, we use this approach to develop a FRW algorithm for the one-dimensional modified Helmholtz equation given by

$$\frac{d^2U}{dx^2} - k^2U = 0, \tag{15}$$

where U is the dependent variable of interest defined in the problem domain $0 \le x \le L,$ and k is a real number independent of x. The boundary conditions imposed are $U(0) = \chi$ and $\left\{ dU \middle/ dx \right\}_{x=L} = \delta.$ Our approach is motivated by the one-dimensional version of Green's integral representation given by (12) and is based on a Green's function $G(x \mid x_0)$ of (15) given by

$$\frac{d^2G}{dx^2} - k^2G(x \mid x_0) = \delta(x - x_0), \tag{16}$$

defined in the problem domain $-h \le x \le h$ with the boundary conditions $G(-h \mid x_0) = 0$ and $\left\{ dG \middle/ dx \right\}_{x=h} = 0.$ This Green's function is explicitly given by

$$\begin{aligned} G(x \mid x_0) &= -\frac{\cosh[k(x_0 - h)]}{k\cosh[2kh]} \\ &\quad \times \sinh[k(x+h)], \quad x \le x_0, \\ G(x \mid x_0) &= -\frac{\sinh[k(x_0 + h)]}{k\cosh[2kh]} \\ &\quad \times \cosh[k(x-h)], \quad x \ge x_0. \end{aligned} \tag{17}$$

We use the boundary conditions that have been imposed on the Green's function given by (17) and the one-dimensional Green's integral representation given by (12) to obtain a representation of the solution $U(x_0)$ at a point $-h \le x_0 \le h$ given by

$$\begin{aligned} U(x_0) &= -\left[G(x \mid x_0)\frac{dU}{dx} \right]_{x=h} \\ &\quad - \left[U(x)\frac{dG}{dx} \right]_{x=-h}. \end{aligned} \tag{18}$$

We now obtain a derivative of (18) with respect to x_0 and obtain a representation of the derivative of the variable of interest U given by

$$\frac{dU}{dx_0} = -\left[\frac{dG}{dx_0}\frac{dU}{dx} \right]_{x=h} - \left[U(x)\frac{d^2G}{dxdx_0} \right]_{x=-h}. \tag{19}$$

Equations (18) and (19) are used to generate a FRW scheme that is different from the scheme based on (13). In order to estimate the solution at a given point, the random walker hops to either the left or the right with probability $1/2$ as given by (18). If the random walker moves to the left, there is a multiplicative weight factor given by $W_{LL} = [-2G_x(x \mid x_0)]_{x=-h}$ and (18) is again used to generate the random walks in the next hop. On the other hand, if the random walker moves to the right, there is a multiplicative weight factor given by $W_{LR} = [-2G(x \mid x_0)]_{x=h}$ and (19) is used to generate the random walks in the next hop. As (19) is used to generate the random walks, the random walker moves to the left or the right with probability $1/2$. If the random walker moves to the left, there is a multiplicative weight factor given by $W_{RL} = \left[-2G_{xx_0}(x \mid x_0)\right]_{x=-h}$ and (18) is used to generate the random walks in the next hop. On the other hand, if the random walker moves to the right, there is a multiplicative weight factor given by $W_{RR} = \left[-2G_{x_0}(x \mid x_0)\right]_{x=h}$ and (19) is used to generate the random walks in the next hop. A particular random walk terminates either in the left boundary with a reward χ or at the right boundary with a reward δ, and an estimate of the solution is obtained by averaging over a statistically significant number of random walks. Thus, through the use of the Green's function in (17), the partially reflecting boundary at $x = L$ is converted to an absorbing boundary and there is no reflection. The results for the problem given by (15) will now be presented.

III. SOFTWARE RESULTS

A mixed boundary condition problem for the modified Helmholtz equation given by (15) is chosen with $L = 3$, $k = 0.5$, $\chi = 1$ and $\delta = 3$. Fig. 1 plots the exact analytical solution and the FRW results and these are seen to be in excellent agreement. In the FRW simulations, 2×10^6 random walks have been carried out for each solution point and the average error between the analytical results and the FRW solution was seen to be about 0.1 percent. Fig. 2 shows the relative speed of parallelization with respect to single processor computation. It is seen that the relative speed of parallelization gets closer to unity as the number of random-walks per solution point is increased. The increased deviation from linearity with decrease in the number of random-walks can be interpreted as the percentage increase in inter-processor communication time with respect to actual computation time that occurs with decrease in the number of random-walks.

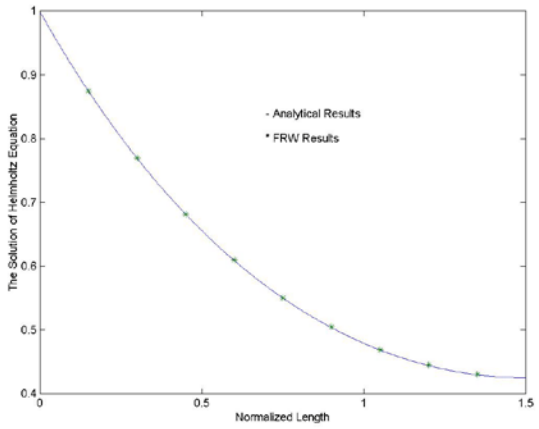

Fig. 1. Analytical and FRW results for the solution of the modified Helmholtz equation plotted against normalized length $\left(x' = kx\right)$.

IV. GPU IMPLEMENTATION

While multiprocessor environments provide exceptional throughput advantages in scientific computing, GPUs have recently emerged as an alternative highly-parallel technology for general purpose computing. GPUs contain a significantly higher core density than any commercially-available CPU, with the tradeoff

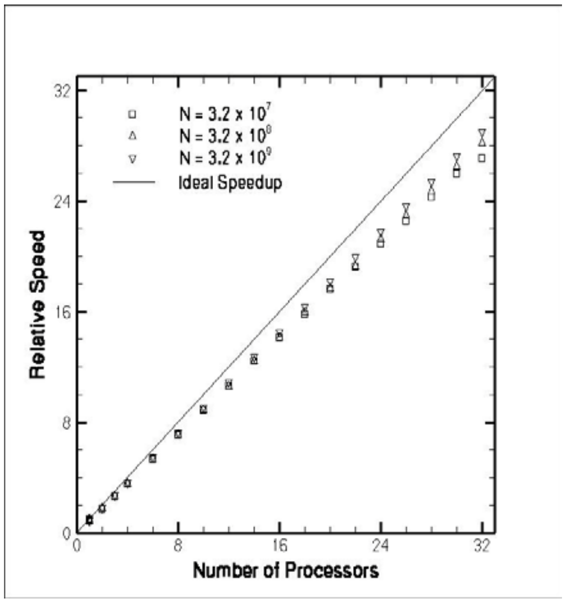

Fig. 2. Relative speed of parallelization with respect to single processor computation.

of a restricted and relatively more specific instruction set and the need for complex memory management by the designer. With the recent introduction [7] of the Compute Unified Device Architecture (CUDA) to developers on commodity NVDIA graphics cards, throughput gains of scientific applications can be increased by orders of magnitude. Using NVIDIA's CUDA application programming interface (API), we have implemented the previously-developed algorithm to explore and evaluate the merit and value of the approach. Efficient parallelization is a significant logical problem that may be solved in different ways depending on the nature of the architectural environment and the algorithm under parallelization. CUDA’s shared memory model and layout of threads into blocks, for example, must be taken into account to maximally utilize the GPU’s resources.

The Helmholtz problem under study lends itself well to parallelization. As the problem requires executing highly-repetitive code for millions of iterations per point at which the equation is solved, it can directly be parallelized by assigning each point to a thread within a thread block. Our final program was divided into three sections:

1) A serial (CPU) part that set up initial conditions.
2) A parallel version that ran 200,000 iterations of the algorithm for each point.
3) A serial finalization section that collected and displayed the results from the GPU.

The implementation has been simplified through the use of global GPU memory that is retained for the execution lifetime of the application. Additional improvements of 100x to 150x speedup can be achieved by using the shared memory, visible between the threads of a block but not shared among other blocks. This requires some overhead to load from global to shared memory at the beginning of the block and restore results to global memory at the end, but 'reads' and 'writes' from shared memory are comparable to register access time which is drastically faster than the 400-600 cycles required for global memory 'reads/writes.' The initial copy from global to shared memory is hidden by using thousands of threads that are more computationally intensive.

A 200,000-iteration test has been run on both a CPU and a GPU, with 16 points for the CPU and 4096 points for the GPU, the CPU completed its task in 2.57 seconds, while the GPU took 16.56 seconds. This indicates a speedup of 39.7 (0.16s/point vs. 0.004s/pt) in favor of the GPU. With additional improvements in memory handling, significant additional speedup may be possible. The GPU implementation is based on 4096 parallel threads organized into 32 blocks of 128 threads each for purposes of coalesced memory access and thread scheduling. To circumvent an 8-second execution time limit imposed by the Linux drivers for the graphics card, tests involving more than 100,000 random walks were broken down into multiple sub-steps each running at most 100,000 walks. Note that the imprecision visible in Figure 3 below is attributable both to the use of single-precision floating point calculations instead of double-precision in our tests (the GPU in question, GeForce GTS 8800, does not contain double-precision floating point units) and to the relatively low number of walks (i.e., 200000) for each point. Although only nine representative points are shown on the graph, the Helmholtz solution was calculated for a full 4096-point span for $x' = [0,1.5]$, where $x' = kx$ is the normalized length scale shown in Fig. 1.

The GPU implementation utilized the Mersenne Twister pseudorandom number generator (PRNG) written by NVIDIA [8]. It was used to generate a very large array of pseudorandom numbers before the random walk algorithm began; the algorithm then picked successive numbers out of the pre-calculated array. The random number array of about 24 million values was recalculated on the GPU in roughly 1 second using 4096 parallel threads; when the test was broken into multiple sub-steps, a new set of PRNGs was calculated with a new seed for each sub-step.

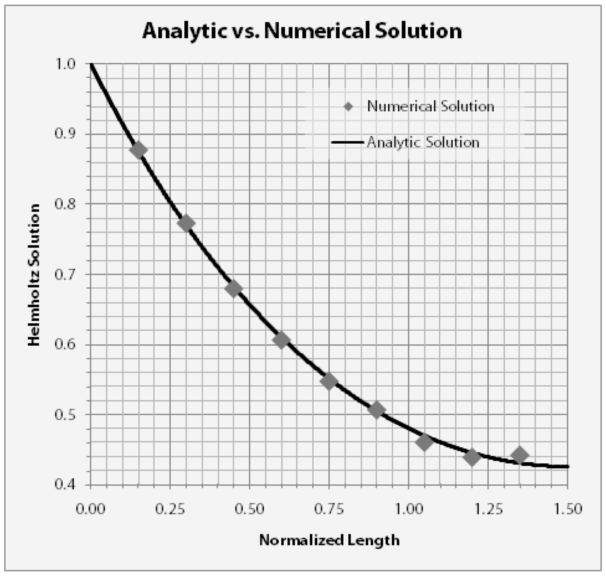

Fig. 3. GPU solution to the Helmholtz equation utilizing random walks. The solution at nine representative points from a total of 4096 is shown.

V. CONCLUSION

Summarizing, a previously-developed FRW methodology [6] for Neumann and mixed boundary problems has been extended to the solution of the 1D modified Helmholtz equation. In this methodology, reflecting boundaries are converted into absorbing boundaries through the development of Green's functions that mimic the boundary conditions of the problem of interest. The algorithm has been validated by an analytical solution and excellent agreement has been obtained between analytical and numerical results. The algorithm has been parallelized in software and a near linear rate of parallelization has been obtained for as many as thirty-two processors. On a commodity graphics card, a speedup of over two orders of magnitude over the software implementation has been obtained. Further work involving GPU implementation will therefore begin with further optimizing our current implementation and considering a method of higher parallelization by scheduling threads on the per-walk level rather than the per-point level, which while introducing more overhead will allow for similar (and therefore lower) execution time between threads. Our future work in this area will involve the extension of this methodology to other important equations and to problems in two and three dimensions. The ultimate goal of this research is the utilization of this methodology for the solution of aerodynamical problems with Neumann and mixed boundary conditions.

ACKNOWLEDGMENT

This research has been supported by the Air Force Office of Scientific Research through a grant (FA9550-06-1-0439) monitored by Dr. F. Fahroo.

REFERENCES

[1] K. Chatterjee, M. Sandora, C. W. Yu, S. Srinivasan, J. Poggie, “A New Parallelized Floating Random-Walk Algorithm for the Modified Helmholtz Equation Subject to Neumann and Mixed Boundary Conditions: Validation with a 1D Benchmark Problem,” *The 25th International Review of Progress in Applied Computational Electromagnetics*, March 2009.

[2] Y. L. Le Coz and R. B. Iverson, “A Stochastic Algorithm for High Speed Capacitance Extraction in Integrated Circuits,” *Solid-State Electronics*, Vol. 35, pp. 1005-1012, 1992.

[3] M. Sobol, *A Primer for the Monte Carlo Method*, CRC Press: Boca Raton, 1994.

[4] R. Haberman, *Elementary Applied Partial Differential Equations*, 3rd ed., Prentice-Hall: New Jersey, 1998.

[5] A. Haji-Sheikh, *Application of Monte Carlo Methods to Thermal Conduction Problems*, Ph.D. dissertation, University of Minnesota, 1965, pp. 106-108.

[6] K. Chatterjee, C. Yu, S. Srinivasan and J. Poggie, “A New Floating Random Walk Methodology for Neumann and Mixed Boundary Condition Problems Without Reflections at Boundaries: Validation with Laplace’s Equation in One Dimension,” *Far East Journal of Applied Mathematics*, Vol. 26, No. 3, pp. 705-713, 2007.

[7] NVIDIA, 2007, Compute Unified Device Architecture: http://developer.download.nvidia.com/compute/cuda/1_0/NVIDIA_CUDA_Programming_Guide_1.0.pdf.

[8] V. Podlozhnyuk, Parallel Mersenne Twister: http://developer.download.nvidia.com/compute/cuda/sdk/website/projects/MersenneTwister/doc/MersenneTwister.pdf, 2007.

Kausik Chatterjee was born in India in 1969. In 1992, he received a Bachelor of Engineering degree in Electrical Engineering from Jadavpur University, Calcutta, India. Subsequently, in 1995, he received a Master of Technology degree in Nuclear Engineering from Indian Institute of Technology, Kanpur, India, and in 2002, he received his PhD degree in Electrical Engineering from Rensselaer Polytechnic Institute, Troy, New York. In 2002, he joined the faculty at California State University, Fresno as an Assistant Professor of Electrical and Computer Engineering, a position he held till 2005. He had been a visiting scientist at MIT Laboratory for Electromagnetic and Electronic Systems and had held faculty fellowships at Air Force Research Laboratory (Wright-Patterson Air Force Base) and NASA, Langley Research Center. He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at The

Cooper Union for the Advancement of Science and Art in New York City. His current research interests include the development of random-walk algorithms for important equations in nature and a theory for high temperature superconductors.

McCullen Sandora was born on March 21, 1987. In May 2009, he received a Bachelor of Science degree in Interdisciplinary Engineering from The Cooper Union for the Advancement of Science and Art in New York City. He is currently a doctoral student in the Department of Physics at University of California, Davis and his current research focus is in the area of High Energy Theory.

Christopher Mitchell was born on March 11, 1987. In May 2009, he received his BE degree in Electrical Engineering from The Cooper Union for the Advancement of Science and Art in New York City, and expects to receive his Masters of Engineering in Electrical Engineering from the same institution in May, 2010. His current research interests include augmented reality and wearable computing hardware and applications, neural network applications in the areas of image recognition and self-training syntactical parsing and the implementation of high-functioning software applications on low-resource portable devices.

Deian Stefan received his B.E. degree in Electrical Engineering from The Cooper Union for the Advancement of Science and Art in 2009 and will receive his M.E. degree in Electrical Engineering in 2010. His current research interests include the analysis and implementation of cryptographic and coding algorithms.

David Nummey was born on April 18, 1987. In May 2009, he received his Bachelor of Engineering degree in Electrical Engineering from The Cooper Union for the Advancement of Science and Art in New York City. He is now working toward his Masters of Engineering degree in Electrical Engineering at the same institution, and expects to finish in May 2010. His current research interests include statistical signal processing methods, pattern recognition and machine learning techniques, and applications towards medical imaging, electrophysiology, neuroscience and assistive devices.

Jonathan Poggie was born in the United States in 1966. He studied mechanical engineering at the University of Rhode Island, receiving a B.S. degree in 1988. He went on to obtain a Ph.D. degree in mechanical and aerospace engineering from Princeton University in 1995, specializing in fluid mechanics. Since then, he has worked at the U.S. Air Force Research Laboratory, focusing on physical problems associated high speed flight. His work has encompassed laminar/turbulent transition in hypersonic boundary layers, unsteadiness of shock waves in separated flow, and the control of ionized gas flow by electromagnetic means.