Journal of Biomedical Optics

SPIEDigitalLibrary.org/jbo

Parallel multigrid solver of radiative transfer equation for photon transport via graphics processing unit

Hao Gao
Lan Phan
Yuting Lin

SPIE

Parallel multigrid solver of radiative transfer equation for photon transport via graphics processing unit

Hao Gao,[a,b] Lan Phan,[c] and Yuting Lin[d]
[a]Emory University, Department of Mathematics and Computer Science, Atlanta, Georgia 30322
[b]Emory University, Department of Radiology and Imaging Sciences, Atlanta, Georgia 30322
[c]Virtium Technology, Inc., Rancho Santa Margarita, California 92688
[d]University of California, Tu and Yuen Center for Functional Onco-Imaging, Department of Radiological Sciences, Irvine, California 92697

Abstract. A graphics processing unit–based parallel multigrid solver for a radiative transfer equation with vacuum boundary condition or reflection boundary condition is presented for heterogeneous media with complex geometry based on two-dimensional triangular meshes or three-dimensional tetrahedral meshes. The computational complexity of this parallel solver is linearly proportional to the degrees of freedom in both angular and spatial variables, while the full multigrid method is utilized to minimize the number of iterations. The overall gain of speed is roughly 30 to 300 fold with respect to our prior multigrid solver, which depends on the underlying regime and the parallelization. The numerical validations are presented with the MATLAB codes at https://sites.google.com/site/rtefastsolver/. [DOI: 10.1117/1.JBO.17.9.096004]

Keywords: radiative transfer equation; Boltzmann equation; graphics processing unit; multigrid.

Paper 12232 received Apr. 16, 2012; revised manuscript received Aug. 1, 2012; accepted for publication Aug. 3, 2012; published online Sep. 6, 2012.

1 Introduction

The accurate modeling of photon migration is a prerequisite for quantitative optical imaging. The general approach is through either the Monte Carlo simulation (MC)[1] or solving its continuous form, so-called radiative transfer equation (RTE), as an integro-differential equation.[2] In practice, the diffusion approximation (DA) of RTE is often used for its simplicity. However, the accuracy of DA for modeling light propagation is considerably degraded in nondiffusive media,[3] particularly for small animal imaging. Several groups studied RTE based image reconstructions, which demonstrated the significant improvement over those based on DA.[4–9]

Although the numerical methods for solving RTE have been extensively studied,[2,10] the development of an efficient RTE solver is nontrivial mainly due to its high dimensionality, e.g., three spatial dimensions plus two angular dimensions for steady-state or frequency-domain RTE in three-dimensional (3-D). An advantage pertinent to this continuous form as RTE is that the solution efficiency can be improved through general techniques for solving numerical partial differential equations.[2,10] In our prior work,[11] we used the angular finite element method and the spatial Discontinuous Galerkin (DG) method to obtain a second-order scheme,[12] which means that the solution accuracy improves quadratically with respect to the mesh refinement. In other words, the same solution accuracy can be met with only the square root of the number of variables in each dimension that would be required for a first-order scheme with the linear accuracy. In addition, to deal with the slow solution convergence in the diffusive regime, where the scattering is considerably larger than the absorption, we use the multigrid method to accelerate the solution convergence. Despite of the acceleration through the high-order scheme and the multigrid method, the speed still needs to be improved in order for this RTE solver to be routinely used.

Graphics processing unit (GPU) is a highly parallel multithreaded, manycore processor.[13] It is well-suited to addressing problems that can be expressed as data-parallel computations. In addition, GPU costs much less than parallel computers. It has been employed in various biomedical applications and achieved ten-fold to hundred-fold gains in speed. In particular, GPU algorithms for solving RTE were recently developed based on the standard angular discrete ordinates and the source-iteration approach, respectively for optical imaging[14] and in the field of atmospheric radiative transfer.[15] Since our RTE solver is also highly parallelizable, the parallelization via GPU is attractive considering its efficiency, cost, and complexity.

In this work, we are going to develop a GPU algorithm to realize the extra gain in speed based on our multigrid solver.[11] Here, the parallelization is considerably different from existing works,[14,15] such as the spatial parallelization, in order to optimize the speed gain with respect to the current multigrid solver. The major contribution of this work is to use the GPU to accelerate the source-iteration steps, while the proposed parallel algorithms do not degrade the performance of the multigrid method in terms of the number of multigrid iterations. As a result, the execution time per iteration and the number of iterations will be presented separately in the result section.

2 Multigrid Solver of RTE

Our approach for solving RTE is based on the full multigrid method, which consists of "*V*" multigrid cycles [Fig. 1(a)] with the purpose of reducing the number of iterations or so-called relaxation steps, especially in nontransport or diffusive regimes. The *V* cycles consist of the relaxation, the defect

Address all correspondence to: Hao Gao, Emory University, Department of Mathematics and Computer Science, Atlanta, Georgia 30322. E-mail: haog@mathcs.emory.edu

0091-3286/2012/$25.00 © 2012 SPIE

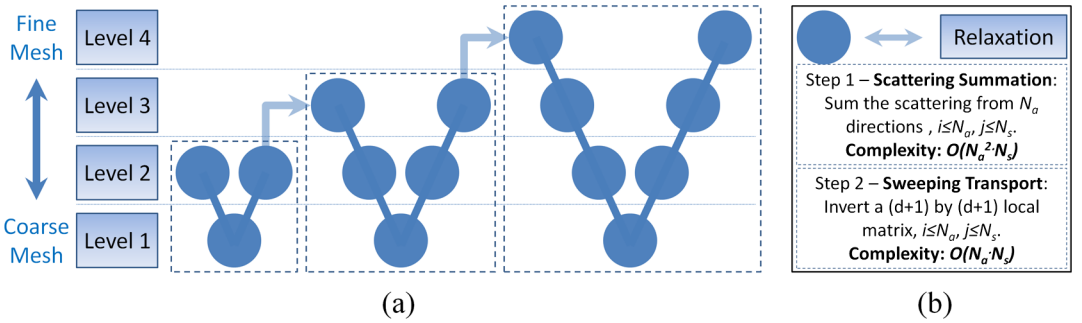

Fig. 1 Full multigrid method for solving RTE.

(residual), and the interpolation steps with respect to different levels of meshes in terms of either angular or spatial variables. During each V cycle, n_1 relaxation steps are performed on the current level of mesh, respectively, before the computation on the next coarser level of mesh. Following n_1 relaxations, the residual or defect is computed on the current level and interpolated to the next coarser level of mesh; this is repeated until the coarsest level of mesh. From the coarsest level, the solution due to the residual is projected back to the finer level of mesh and added to the finer solution, and then another n_2 relaxation steps are performed; this is repeated until the finest level of mesh. Here, $n_1 = n_2 = 3$; and the V cycles that do not include the finest level of mesh serve as to provide a good initial guess for the next finer level. As an example with four levels of meshes shown in Fig. 1, a V cycle including Levels 1 and 2 is performed to provide a good initial guess for Level 3; a V cycle including Levels 1 through 3 is performed to provide a good initial guess for Level 4; then a few V cycles including Levels 1 through 4 are performed until the stopping criterion is satisfied. The interested readers may refer to our prior work[11] for the efficiency of the multigrid method in reducing the number of iterations.

The computationally dominant steps occur from the relaxation and the defect, since the computation time for interpolations between meshes is much smaller than that for relaxations. On the other hand, the computation for defect and relaxation shares the same structure, which can be divided into two steps–scattering summation and sweeping transport [Fig. 1(b)]. As illustrated in Fig. 1(b), the order of computational complexity is $O(N_a^2 \cdot N_s)$ for the scattering step and $O(N_a \cdot N_s)$ for the sweeping step, where N_a and N_s are the degrees of freedom in angle and space, respectively. The details are available in Ref. 11. Therefore, in the next section, we are going to consider the parallel algorithms mainly for these two steps.

3 Parallelization of Scattering Summation

When solving RTE, the scattering integral in RTE is discretized as a summation with respect to angular variables: the scattering contribution from all angles is summed for each spatial node along each angular direction. Overall, this scattering summation is the most computationally dominant step. The goal of the parallelization of scattering summation is to reduce the complexity so that its computational cost is linearly, rather than quadratically, proportional to the angular degrees of freedom. For example, the computational time increases by twice, rather than four times, when the angular degrees of freedom double.

This goal can be achieved on GPU through the standard GPU tricks by using shared memory,[13] since the scattering summation is essentially equivalent to the multiplication of the angular scattering weight matrix (1 in Fig. 2) and the photon flux matrix (2 in Fig. 2). That is, we perform the matrix multiplication of block submatrices with the size BLOCKSIZE × BLOCKSIZE, so that the memory communication is minimal on the GPU device [Fig. 2(a)]. When the matrix size exceeds the device memory, since the spatial degrees of freedom N_s is often much larger than the angular degrees of freedom N_a, it is natural to divide with respect to spatial degrees of freedom, and then sequentially send the partial photon flux matrix with the size $N_s' \times N_a$ to the GPU device for parallel computation [Fig. 2(b)]. For convenience and efficiency, the matrix multiplication functions from CUBLAS library[16] are called in implementation.

Please note that, unlike the sweeping transport, the scattering summation is completely independent in both angle and space, and therefore fully parallelizable in theory, which allows the GPU device to reach its capacity in practice.

4 Parallelization of Sweeping Transport

After the scattering contribution is computed through the scattering summation in each relaxation or defect step, the photon flux or the solution residual is computed through "sweeping" over the spatial mesh for each angular direction. The details are available in Ref. 11.

Although the parallelization of sweeping transport is still (almost) parallelizable in angular variables (which will be discussed next), the sweeping order can be a restriction for the parallelization of sweeping transport with respect to spatial variables. That is, along the sweeping characteristics, the transport computation is supposed to be performed in certain orders on the spatial mesh for transport efficiency. However, this is not the case when the medium is no longer transport-dominant. That

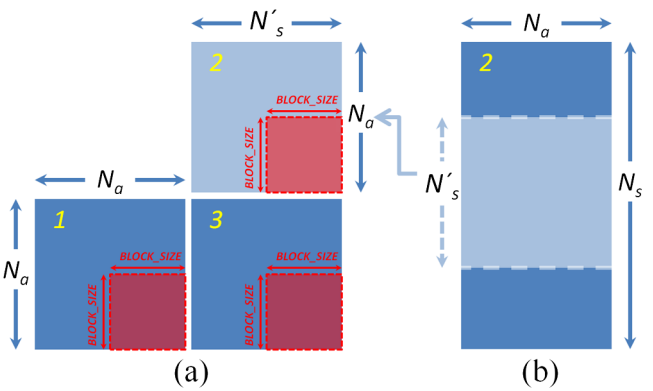

Fig. 2 Parallelization of scattering summation.

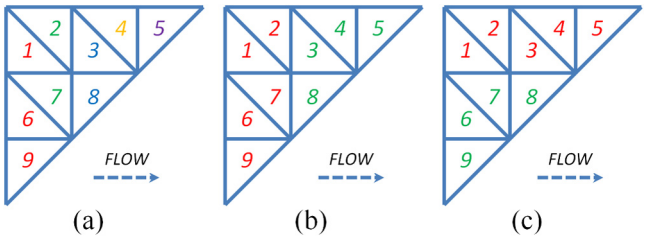

Fig. 3 Parallelization of sweeping transport.

is, the transport order along any angular direction is no longer crucial due to the scattering.

When the sweeping ordering is considered, the spatial mesh is divided into different causality groups so that the nodes in each group are independent from the nodes in other groups. Therefore, the nodal sweeping in groups can be done in parallel [Fig. 3(a)]. For example, in Fig. 3(a), according to the flow direction, Triangles 1 to 5 form a causality group, Triangles 6 through 8 form a causality group, and Triangle 9 forms a causality group. During the sweeping transport, only the nodes within the same group are dependent. For example, the computation in the group with Triangles 1 through 5 should go sequentially from Triangle 1 to Triangle 5. Therefore, in parallelization, independent computations can be carried out simultaneously. That is the parallelization can be done on the nodes with the same color, i.e., on Triangle 1, 6, and 9 first, Triangle 2 and 7 next, then Triangle 3 and 8, then Triangle 4, and last Triangle 5.

However, as indicated by Fig. 3(a), the size of parallelizable groups may vary dramatically, which is not ideal for parallelization. This strict compliance has to be compromised in implementation. That is, instead of strictly following the causality, we use distance metrics to order the spatial nodes and parallel the spatial mesh accordingly [Fig. 3(b)], which provides the uniform group size for parallelization while conforming to the causality. As an example shown in Fig. 3(b), according to the flow direction, the nodes are naturally divided into two groups according to the distance function: Triangles 1 to 2, 6 to 7 and 9 form a parallelizable group; Triangles 3 to 5 and 8 form a parallelizable group. In this way, the efficiency of the parallelization is improved while the transport ordering is approximately maintained. However, since the spatial parallelization groups are different from angle to angle, the GPU computational speed is not optimal due to this complexity in accessing the memory.

Therefore, we consider the sweeping transport without the sweeping ordering [Fig. 3(c)]. When the medium is diffusive or not under the transport regime, as mentioned above, the sweeping ordering is not essential. In this case, the parallelization is straightforward in space [Fig. 3(c)]. In Fig. 3(c), we simply parallel the spatial nodes according to its original ordering. That is Triangles 1 to 5 form a parallelizable group for all angular directions, and Triangles 6 to 9 form a parallelizable group for all angular directions. Unlike in Fig. 3(a) and 3(b), the parallelization of spatial nodes is the same for all angular directions in Fig. 3(c). In this way, the memory access is efficient and the GPU speed is optimal. However, when the medium is under the transport regime, the efficiency of the overall multigrid method could be affected, i.e., the number of multigrid cycles could increase due to the violation of the transport causality or the delay of the upwind flux.

Next, we briefly comment on the angular parallelization of the sweeping transport step, which is fully parallelizable under the vacuum boundary condition. However, under the reflection boundary condition, the mismatch of refraction index at the boundary causes the boundary scattering. That is, the boundary flux along the sweeping direction is also related to the boundary flux from other directions. Therefore, the sweeping transport is no longer independent among angles under the reflection boundary condition. However, we can still precompute the boundary scattering contributions in parallel and then fully parallelize in angle. In this way, although the boundary scattering lags behind, it does not severely degrade the overall performance.

Again, when the matrix size exceeds the device memory, we can sequentially perform the parallelization on GPU with respect to the angular variables, and even spatial variables if necessary.

5 Parallelized Multigrid Solver of RTE

The parallelization of the interpolation steps is straightforward since the interpolation consists of component-wise operations that are totally independent from each other and therefore fully parallelizable.

There is a subtle change in algorithm due to the parallelization of the relaxation. Previously, the relaxation was equivalent to the Gauss-Seidel iterative method in both angular and spatial variables. After the parallelization, the scattering step is separated from the transport step, and therefore the relaxation is equivalent to a mixed iterative method, which is like the Gauss-Seidel method in spatial variables but like the Jacobi method in angular variables. Due to this change, the number of multigrid iterations may increase in the diffusive regime, since the Jacobi method is not as efficient as the Gauss-Seidel method when solving elliptic type of differential equations with the multigrid method. However, the Jacobi method for RTE does allow the scattering summation to be separately computed from the sweeping transport, which could offer an order-of-magnitude gain in speed even on CPU (See Tables 1 through 6).

Overall, with the above parallelized relaxation, defect, and interpolation steps, the efficiency of the multigrid method is still roughly maintained so that the number of multigrid iterations stays roughly at the same level despite of some performance variations in different regimes.

6 Results

We consider a two-dimensional (2-D) circular domain centered at the origin with a diameter of 60 mm (millimeters). An isotropic point source is put at (−27 mm, 0) with the modulation frequency 100 MHz. The Henyey-Greenstein phase function with the anisotropic factor $g = 0.9$ is employed as the scattering kernel. The refraction index of the environment is set to be 1; the refraction index of the medium is set to be 1 under the vacuum boundary condition, and 1.33 under the reflection boundary condition. The homogenous background is assumed with the absorption coefficient μ_a and the scattering coefficient μ_s. Typical scattering-to-absorption ratios are selected to represent the transport regime ($\mu_a = 0.01$ and $\mu_s = 0.1$; ∼0.6 reduced optical depths), the intermediate regime ($\mu_a = 0.01$ and $\mu_s = 1$; ∼6 reduced optical depths), and the diffusive regime ($\mu_a = 0.01$ and $\mu_s = 10$; ∼60 reduced optical depths) in Tables 1 through 3, respectively.

We also consider a 3-D cubic domain centered at (20 mm, 20 mm, 20 mm) with the side length of 40 mm each. The setting and the parameters in 3-D are the similar to those in 2-D, with an

Table 1 Performance comparison of RTE solvers in 2-D when $\mu_a = 0.01$ and $\mu_s = 0.1$ (transport regime). In the results, the first number is the computational time (in seconds) of a multigrid iteration, and the second is the number of multigrid iterations.

B.C.	Vacuum boundary condition			Reflection boundary condition		
N_a	128	256	256	128	256	256
N_s	6144	6144	24,576	6144	6144	24,576
CPU	8.2/1	45/1	356/1	8.2/1	45/1	360/1
CPU (fast)	3.7/1	12/1	49/1	3.7/2	12/1	50/1
GPU	0.33/2	0.69/1	2.9/1	0.33/2	0.69/1	2.9/1
GPU (fast)	0.14/5	0.30/4	1.2/7	0.14/6	0.30/5	1.2/9

Table 2 Performance comparison of RTE solvers in 2-D when $\mu_a = 0.01$ and $\mu_s = 1$ (intermediate regime). In the results, the first number is the computational time (in seconds) of a multigrid iteration, and the second is the number of multigrid iterations.

B.C.	Vacuum boundary condition			Reflection boundary condition		
N_a	128	256	256	128	256	256
N_s	6144	6144	24,576	6144	6144	24,576
CPU	8.2/1	44/1	358/1	8.2/1	45/2	354/1
CPU (fast)	3.7/1	12/2	49/1	3.7/2	12/2	48/1
GPU	0.33/2	0.69/2	2.9/2	0.33/3	0.69/2	2.9/2
GPU (fast)	0.14/2	0.30/2	1.2/2	0.14/2	0.30/2	1.2/2

Table 3 Performance comparison of RTE solvers in 2-D when $\mu_a = 0.01$ and $\mu_s = 10$ (diffusive regime). In the results, the first number is the computational time (in seconds) of a multigrid iteration, and the second is the number of multigrid iterations.

B.C.	Vacuum boundary condition			Reflection boundary condition		
N_a	128	256	256	128	256	256
N_s	6144	6144	24,576	6144	6144	24,576
CPU	8.2/11	44/11	344/11	8.2/11	44/11	358/12
CPU (fast)	3.7/26	12/25	50/26	3.7/27	12/26	49/26
GPU	0.33/29	0.69/28	2.9/28	0.33/29	0.69/28	2.9/28
GPU (fast)	0.14/29	0.30/28	1.2/28	0.14/30	0.30/29	1.2/28

isotropic point source at (2 mm, 2 mm, 2 mm). Again, typical scattering-to-absorption ratios are selected to represent the transport regime ($\mu_a = 0.01$ and $\mu_s = 0.1$; ~0.6 reduced optical depths), the intermediate regime ($\mu_a = 0.01$ and $\mu_s = 1$; ~6 reduced optical depths), and the diffusive regime ($\mu_a = 0.01$ and $\mu_s = 10$; ~60 reduced optical depths) in Tables 4 to 6, respectively.

In the result tables, $N_a(N_s)$ is the angular (spatial) degrees of the freedom; CPU denotes the prior multigrid RTE solver; CPU (fast) denotes the fast multigrid RTE solver, with the accelerated scattering summation via the matrix multiplication of block submatrices [Fig. 2(a)]; GPU denotes the parallelized multigrid RTE solver with transport spatial ordering [Fig. 3(b)]; GPU (fast) denotes the parallelized multigrid RTE solver with natural spatial ordering [Fig. 3(c)]. The same stopping criterion based on the residual[11] is used to make sure the solutions from various solvers have the same order of the accuracy.

The above results are based on NVIDIA Tesla C2070 (5.25 GB device memory) on a desktop with Intel Xeon CPU E5620 2.40 GHz, and the solvers are complied through x64 complier from Microsoft Visual Studio with single precision.

Table 4 Performance comparison of RTE solvers in 3-D when $\mu_a = 0.01$ and $\mu_s = 0.1$ (transport regime). In the results, the first number is the computational time (in seconds) of a multigrid iteration, and the second is the number of multigrid iterations.

B.C.	Vacuum boundary condition			Reflection boundary condition		
N_a	66	258	258	66	258	258
N_s	49,152	49,152	393,216	49,152	49,152	393,216
CPU	41/1	602/1	11673/1	41/1	655/1	11412/1
CPU (fast)	11/1	95/1	784/1	12/1	95/2	790/1
GPU	0.98/1	4.6/1	191/1[a]	0.98/2	4.6/2	193/1[a]
GPU (fast)	0.52/8	3.0/7	NA	0.54/11	2.9/13	NA

[a]when $N_a = 258$ and $N_s = 393,216$, since the required memory exceeds the GPU memory, there are multiple memory passes between CPU and GPU and the GPU speed is degraded.

Table 5 Performance comparison of RTE solvers in 3-D when $\mu_a = 0.01$ and $\mu_s = 1$. In the results, the first number is the computational time (in seconds) of a multigrid iteration, and the second is the number of multigrid iterations.

B.C.	Vacuum boundary condition			Reflection boundary condition		
N_a	66	258	258	66	258	258
N_s	49,152	49,152	393,216	49,152	49,152	393,216
CPU	41/3	654/1	12048/1	41/3	658/2	11698/2
CPU (fast)	12/3	94/2	790/2	12/3	95/3	790/3
GPU	0.98/3	4.6/2	191/2[a]	1.0/4	4.6/6	195/3[a]
GPU (fast)	0.53/5	3.0/4	NA	0.54/7	2.9/9	NA

[a]when $N_a = 258$ and $N_s = 393,216$, since the required memory exceeds the GPU memory, there are multiple memory passes between CPU and GPU and the GPU speed is degraded.

Table 6 Performance comparison of RTE solvers in 3-D when $\mu_a = 0.01$ and $\mu_s = 10$ (diffusive regime). In the results, the first number is the computational time (in seconds) of a multigrid iteration, and the second is the number of multigrid iterations.

B.C.	Vacuum boundary condition			Reflection boundary condition		
N_a	66	258	258	66	258	258
N_s	49,152	49,152	393,216	49,152	49,152	393,216
CPU	41/10	641/12	11544/9	41/12	595/15	11930/11
CPU (fast)	12/27	93/27	746/23	12/30	92/32	747/27
GPU	0.98/40	4.6/41	183/23[a]	0.99/45	4.6/48	191/27[a]
GPU (fast)	0.52/43	3.0/42	NA	0.53/49	2.9/50	NA

[a]when $N_a = 258$ and $N_s = 393,216$, since the required memory exceeds the GPU memory, there are multiple memory passes between CPU and GPU and the GPU speed is degraded.

7 Discussions and Conclusions

In terms of the speed gain for a multigrid iteration, it is clear from Tables 1 to 6 that the parallelized solvers (GPU and GPU (fast)) have roughly the linear dependence on the angular degrees of freedom N_a, while the prior multigrid solver (CPU) has roughly the quadratic dependence on N_a. And the performance of the fast multigrid solver (CPU(fast)) is in between. As a result, the computational speed for a multigrid iteration is significantly improved through the parallelization on GPU. For example, the achieved gain of speed from CPU to GPU (fast) is 60 to 300 fold in 2-D (Tables 1 to 3) and 80 to 200 fold in 3-D (Tables 4 to 6). This linear dependence

on N_a is crucial, since the computational time can be drastically reduced when a significant number of angular directions are required in order to accurately simulate optically thin medium, the near-source field, the transport region, or strongly forwarded peaking scattering. This is particularly essential for the 3-D RTE computation, since the angular space in 3-D has two dimensions, which means a regular refining of the angular meshes increases N_a by four times as comparing with 2-fold increases for 2-D RTE.

Regarding the number of multigrid iterations, the effect of the parallelization on the multigrid method varies under different RTE regimes. In the transport regime (Tables 1 and 4), the violation of the transport causality causes the increase of the number of iterations for GPU (fast), and therefore the parallelized solver GPU with transport ordering [Fig. 3(b)] is recommended. In both the intermediate regime (Tables 2 and 5) and the diffusive regime (Tables 3 and 6), GPU (fast) is recommended, since the natural spatial ordering [Fig. 3(c)] is sufficient in the presence of the strong scattering.

One aspect of our future work will be on the cases where the problem size exceeds the GPU device memory. As shown in the given 3-D examples ($N_a = 258$ and $N_s = 393,216$), the speed gain is limited by the multiple passes from the host memory to the device memory. A possible solution is to consider the multi-GPU parallelization. On the other hand, the current GPU solver will allow us to efficiently study a few RTE based inverse problems,[17–20] so that we can assess the potential gain in the imaging accuracy due to RTE. In addition, we will also spend time to write user documents and develop user-friendly interfaces to share this solver with the community.

In summary, based on our prior multigrid solver of RTE, we have developed a parallelized solver of RTE with considerably improved speed. This parallel solver has the linear computational complexity with respect to both angular and spatial degrees of freedom. As a result of the parallelization, the overall gain of speed is roughly 30 to 300 fold, which depends on the underlying regime and the parallelization. The solver is for heterogeneous media with complex geometry based on 2-D triangular meshes or 3-D tetrahedral meshes. The MATLAB codes are available at https://sites.google.com/site/rtefastsolver/.

Acknowledgments

This research is supported by NIH/NIBIB (R21 EB013387).

References

1. J. Spanier and E. Gelbard, *Monte Carlo Principles and Neutron Transport Problems*, Addison-Wesley Pub. Co., Boston, MA (1969).
2. E. E. Lewis and W. F. Miller, *Computational Methods of Neutron Transport*, Wiley-Interscience, New York (1984).
3. K. M. Case and P. F. Zweifel, *Linear Transport Theory*, Addison-Wesley Pub. Co., Boston, MA (1967).
4. A. D. Klose et al., "Optical tomography using the time-independent equation of radiative transfer- part 1: forward model," *J. Quant. Spectrosc. R.T.* **72**(5), 691–713 (2002).
5. K. Ren et al., "Algorithm for solving the equation of radiative transfer in the frequency domain," *Opt. Lett.* **29**(6), 578–580 (2004).
6. T. Tarvainen et al "Hybrid radiative-transfer-diffusion model for optical tomography," *Appl. Opt.* **44**(6), 876–886 (2005).
7. A. Joshi et al, "Radiative transport-based frequency-domain fluorescence tomography," *Phys. Med. Biol.* **53**(8), 2069–2088 (2008).
8. H. Gao and H. Zhao, "Multilevel bioluminescence tomography based on radiative transfer equation Part 1: l1 regularization," *Opt. Express* **18**(3), 1854–1871 (2010).
9. H. Gao and H. Zhao, "Multilevel bioluminescence tomography based on radiative transfer equation Part 2: total variation and l1 data fidelity," *Opt. Express* **18**(3), 2894–2912 (2010).
10. M. L. Adams and E. W. Larsen, "Fast iterative methods for discrete-ordinates particle transport calculations," *Prog. Nuc. Energ.* **40**(1), 3–159 (2002).
11. H. Gao and H. Zhao, "A fast forward solver of radiative transfer equation," *Transport Theor. Stat.* **38**(3), 149–192 (2009).
12. H. Gao and H. Zhao, "Analysis of a numerical solver for radiative transport equation," *Math. Comput.*, in press (2012).
13. NVIDIA CUDA C Programming Guide (Version 4.0) (2011).
14. K. Peng et al, "Graphics processing unit parallel accelerated solution of the discrete ordinates for photon transport in biological tissues," *Appl. Opt.* **50**(21), 3808–3823 (2011).
15. B. Huang et al., "Development of a GPU-based high-performance radiative transfer model for the infrared atmospheric sounding interferometer (IASI)," *J. Comput. Phys.* **230**(6), 2207–2221 (2011).
16. NVIDIA CUDA Toolkit 4.0 CUBLAS Library (2011).
17. Y. Lin et al, "Quantitative fluorescence tomography using a tri-modality system: in vivo validation," *J. Biomed. Opt.* **15**(4), 040503 (2010).
18. Y Lin et al, "Photo-multiplier tube based hybrid MRI and frequency domain fluorescence tomography system for small animal imaging," *Phys. Med. Biol.* **56**(15), 4731–4747 (2011).
19. Y. Lin et al., "Tumor characterization in small animals using magnetic resonance-guided dynamic contrast enhanced diffuse optical tomography," *J. Biomed. Opt.* **16**(10), 106015 (2011).
20. H. Gao, S. Osher, and H. Zhao, "Quantitative photoacoustic tomography," Chapter 5, in *Mathematical Modeling in Biomedical Imaging: II Optical, Ultrasound and Opto-Acoustic Tomographies (Lecture Notes in Mathematics)*, H. Ammari, Ed., pp. 131–158, Springer, New York (2012).