Journal of Computations & Modelling, vol.7, no.2, 2017, 29-50
ISSN: 1792-7625 (print), 1792-8850 (online)
Scienpress Ltd, 2017

An Efficient Acceleration of Solving Heat and Mass Transfer Equations with the Second Kind of Boundary Conditions in Solid and Hollow Cylinder Using Programmable Graphics Hardware

Hira Narang[1], Fan Wu[2], Abisoye Ogunniyan[3] and Abdul Rafae Mohammed[4]

Abstract

Recently, heat and mass transfer simulation is more and more important in various engineering fields. In order to analyse how heat and mass transfer in a thermal environment, heat and mass transfer simulation is needed. However, it is too much time-consuming to obtain numerical solutions to heat and mass transfer equations. Therefore, in this paper, one of acceleration techniques developed in the graphics community that exploits a graphics processing unit (GPU) is applied to the numerical solutions of heat and mass transfer equations. The nVidia Compute

[1] Computer Science Department, Tuskegee University, Tuskegee, Alabama, USA.
E-mail: narang@mytu.tuskegee.edu
[2] Computer Science Department, Tuskegee University, Tuskegee, Alabama, USA.
E-mail: wuf@mytu.tuskegee.edu
[3] Computer Science Department, Tuskegee University, Tuskegee, Alabama, USA.
E-mail: aogunniyan9354@mytu.tuskegee.edu
[4] Computer Science Department, Tuskegee University, Tuskegee, Alabama, USA.
E-mail: amohammed4505@mytu.tuskegee.edu

Article Info: *Received* : February 13, 2017. *Revised* : March 19, 2017.
Published online : April 30, 2017.

Unified Device Architecture (CUDA) programming model provides a straightforward means of describing inherently parallel computations. This paper improves the performance of solving heat and mass transfer equations over solid and hollow capillary porous cylinder with the second kind of boundary conditions numerically running on GPU. Heat and mass transfer simulation using the novel CUDA platform on nVidia Quadro FX 4800 is implemented. Our experimental results clearly show that GPU can accurately perform heat and mass transfer simulation. GPU can significantly accelerate the performance with the maximum observed speedups 10 times. Therefore, the GPU is a good approach to accelerate the heat and mass transfer simulation.

Mathematics Subject Classification: 68U20; 65Y05
Keywords: Numerical Solution; Heat and Mass Transfer; General Purpose Graphics Processing Unit; CUDA

1 Introduction

During the last half century, many scientists and engineers working in Heat and Mass Transfer processes have put lots of efforts in finding solutions both analytically/numerically, and experimentally. To precisely analyze physical behaviors of heat and mass environment, to simulate several heat and mass transfer phenomena such as heat conduction, convection, and radiation are very important. A heat transfer simulation is accomplished by utilizing parallel computer resources to simulate such heat and mass transfer phenomena. With the helps from computer, initially the sequential solutions were found, and later when high-end computers became available, fast solutions were obtained to heat and mass transfer problems. However, the heat and mass transfer simulation requires much more computing resources than the other simulations. Therefore,

acceleration of this simulation is very essential to implement a practical big data size heat and mass transfer simulation.

This paper utilizes the parallel computing power of GPUs to speed up the heat and mass transfer simulation. GPUs are very efficient considering theoretical peak floating-point operation rates [1]. Therefore, comparing with super-computer, GPUs is a powerful co-processor on a common PC which is ready to simulate a large-scale heat and mass transfer at a less resources. The GPU has several advantages over CPU architectures, such as highly parallel, computation intensive workloads, including higher bandwidth, higher floating-point throughput. The GPU can be an attractive alternative to clusters or super-computer in high performance computing areas.

CUDA [2] by nVidia already proved its effort to develop both programming and memory models. CUDA is a new parallel, C-like language programming Application program interface (API), which bypasses the rendering interface and avoids the difficulties from using GPGPU. Parallel computations are expressed as general-purpose, C-like language kernels operating in parallel over all the points in an application.

This paper develops the numerical solutions to Two-point Initial-Boundary Value Problems (TIBVP) of Heat and Mass with the second kind of boundary conditions in solid and hollow capillary porous cylinder. These problems can be found some applications in drying processes, space science, absorption of nutrients, transpiration cooling of space vehicles at re-entry phase, and many other scientific and engineering problems. Although some traditional approaches of parallel processing to the solutions of some of these problems have been investigated, no one seems to have explored the high-performance computing solutions to heat and mass transfer problems with compact multi-processing capabilities of GPU, which integrates multi-processors on a chip. With the advantages of this compact technology, we developed algorithms to find the solution of TIBVP with the second kind of boundary conditions and compare with

some existing solutions to the same problems. All of our experimental results show significant performance speedups. The maximum observed speedups are about 10 times.

The rest of the paper is organized as follow: Section II briefly introduces some closely related work; Section III describes the basic information on GPU and CUDA; Section IV presents the mathematical model of heat and mass transfer and numerical solutions to heat and mass transfer equations; Section V presents our experimental results; And Section VI concludes this paper and give some possible future work directions.

2 Related Work

The simulation of heat and mass transfer has been a very hot topic for many years. And there is lots of work related to this field, such as fluid and air flow simulation. We just refer to some most recent work close to this field here.

Soviet Union was in the fore-front for exploring the coupled Heat and Mass Transfer in Porous media, and major advances were made at Heat and Mass Transfer Institute at Minsk, BSSR. Later England and India took the lead and made further contributions for analytical and numerical solutions to certain problems. Narang [4-9] explored the wavelet solutions to heat and mass transfer equations and Ambethkar [10] explored the numerical solutions to some of these problems.

Krüger et al. [11] computed the basic linear algebra problems with the feathers of programmability of fragments on GPU, and further computed the 2D wavelets equations and NSEs on GPU. Bolz et al. [12] matched the sparse matrix into textures on GPU, and utilized the multigrid method to solve the fluid problem. In the meantime, Goodnight et al. [13] used the multigrid method to solve the boundary value problems on GPU. Harris [14, 15] solved the PDEs of dynamic fluid motion to get cloud animation.

GPU is also used to solve other kinds of PDEs by other researchers. Kim et al. [16] solved the crystal formation equations on GPU. Lefohn et al. [17] matched the level-set iso surface data into a dynamic sparse texture format. Another creative usage was to pack the information of the next active tiles into a vector message, which was used to control the vertices and texture coordinates needed to send from CPU to GPU. To learn more applications about general-purpose computations GPU, more information can be found from here [18].

2.1 An Overview of Cuda Architecture

The GPU that we have used in our implementations is nVidia's Quadro FX 4800, which is DirectX 10 compliant. It is one of nVidia's fastest processors that support the CUDA API and as such all implementations using this API are forward compatible with newer CUDA compliant devices. All CUDA compatible devices support 32-bit integer processing. An important consideration for GPU performance is its level of occupancy. Occupancy refers to the number of threads available for execution at any one time.

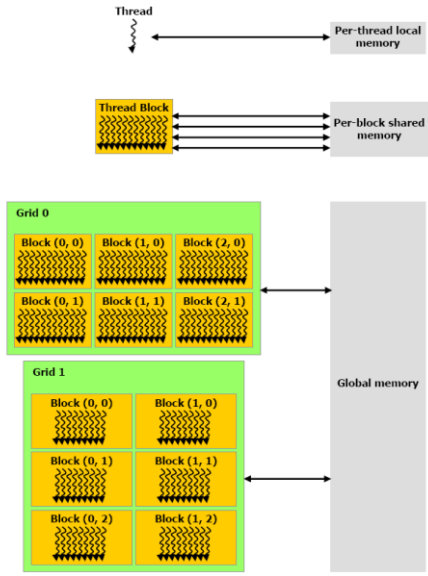

Figure 1: GPU Memory Architecture [2]

It is normally desirable to have a high level of occupancy as it facilitates the hiding of memory latency. The GPU memory architecture is shown in Figure 1.

2.2 Mathematical Model and Numerical Solutions of Heat and Mass Transfer

2.2.1 Mathematical Model

Consider the Heat and Mass Transfer through a porous cylinder with boundary conditions of the second kind. Let the z-axis be directed upward along the cylinder and the r-axis radius of the cylinder. Let u and v be the velocity components along the z- and r- axes respectively. Then the heat and mass transfer equations in the Boussinesq's approximation, are:

$$\frac{\partial T}{\partial t} = \mathrm{k}_1 \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2}\right) + \mathrm{k}_2 \left(\frac{\partial C}{\partial t}\right) \tag{1}$$

$$\frac{\partial C}{\partial t} = \mathrm{k}_3 \left(\frac{\partial^2 C}{\partial r^2} + \frac{1}{r}\frac{\partial C}{\partial r} + \frac{\partial^2 C}{\partial z^2}\right) + \mathrm{k}_4 \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2}\right) \tag{2}$$

$0 < z < \infty, a < r < b^*, t > 0 \quad {}^*a = 0, b = \infty$ for solid cylinder.

Initial Conditions

$$\begin{aligned} T(r,z,o) &= 1 \\ C(r,z,o) &= 0 \end{aligned} \tag{3}$$

Boundary Conditions

$$\begin{aligned} T(r,o,t) &= T_0 = 1 \\ C(r,z,o) &= C_0 = 0 \end{aligned} \tag{4}$$

$$\frac{\partial T(a,z,t)}{\partial X} = T_a = 0.001 \qquad \frac{\partial C(a,z,t)}{\partial X} = C_a = 0.001 \tag{5a}$$

$$\frac{\partial T(b,z,t)}{\partial X} = T_b = 0.00015 \qquad \frac{\partial C(b,z,t)}{\partial X} = C_b = 0.00015 \tag{5b}$$

$$\begin{aligned} T(r,\infty,t) &= T_\infty = 0 \\ C(r,\infty,o) &= C_\infty = 1 \end{aligned} \tag{6}$$

Since the cylinder is assumed to be porous, μ_1 is the velocity of the fluid, T_p the temperature of the fluid near the cylinder, T_∞ the temperature of the fluid far away from the cylinder, C_p the concentration near the cylinder, C_∞ the concentration far away from the cylinder, g the acceleration due to gravity, β the coefficient of volume expansion for heat transfer, β' the coefficient of volume expansion for concentration, ν the kinematic viscosity, σ the scalar electrical conductivity, ω the frequency of oscillation, k the thermal conductivity.

From Equation (1) we observe From Equation (1) we observe that v_1 is independent of space co-ordinates and may be taken as constant. We define the following non-dimensional variables and parameters.

$$t = \frac{t_1 V_0^2}{4\nu}, z = \frac{V_0 z_1}{4\nu} \tag{8}$$

$$u = \frac{u_1}{V_0}, T = \frac{T_1 - T_\infty}{T_P - T_\infty}, C = \frac{C_1 - C_\infty}{C_P - C_\infty}, P_r = \frac{\nu}{k}, S_c = \frac{\nu}{D'}$$

$$M = \frac{\sigma B_0^2 \nu}{\rho V_0^2}, G_r = \frac{\nu g \beta (T_P - T_\infty)}{V_0^3} \qquad G_m = \frac{\nu g \beta' (C_P - C_\infty)}{V_0^3}, \omega = \frac{4\nu\omega_i}{V_0^2}$$

Now taking into account Equations (5), (6), (7), and (8), equations (1) and (2) reduce to the following form:

$$\frac{\partial T}{\partial t} + \frac{\partial^2 T}{\partial r^2} - 4\frac{\partial C}{\partial t} + \frac{1}{r}\frac{\partial T}{\partial r} = \frac{4}{P_r}\frac{\partial^2 T}{\partial z^2} \tag{9}$$

$$\frac{\partial C}{\partial t} + \frac{\partial^2 C}{\partial r^2} - 4\frac{\partial T}{\partial t} + \frac{1}{r}\frac{\partial C}{\partial r} = \frac{4}{P_r}\frac{\partial^2 C}{\partial z^2} \tag{10}$$

$$C(r,z,t) = 0,\ T(r,z,t) = T_0 \qquad t > 0 \tag{11}$$

$$\frac{\partial C(r,z,t)}{\partial z} + k\frac{\partial T(r,z,t)}{\partial z} = 0 \qquad t \le 0 \tag{12}$$

$$T(r,\infty,t) = 0,\ C(r,\infty,t) = 1 \qquad t > 0 \tag{13}$$

2.2.2 Numerical Solutions

Here we sought a solution by finite difference technique of implicit type namely Crank-Nicolson implicit finite difference method which is always convergent and stable. This method has been used to solve Equations (9), and (10) subject to the conditions given by (11), (12) and (13). To obtain the difference equations, the region of the heat is divided into a gird or mesh of lines parallel to *z* and *r* axes. Solutions of difference equations are obtained at the intersection of these mesh lines called nodes. The values of the dependent variables *T,* and *C* at the nodal points along the plane $x = 0$ are given by $T(0,t)$ and $C(0,t)$ hence are known from the boundary conditions.

In the figure 2, Δz, Δr are constant mesh sizes along *z* and *r* directions respectively. We need an algorithm to find single values at next time level in terms of known values at an earlier time level. A forward difference approximation for the first order partial derivatives of *T* and *C*. And a central difference approximation for the second order partial derivative of *T* and *C* are used. On introducing finite difference approximations for:

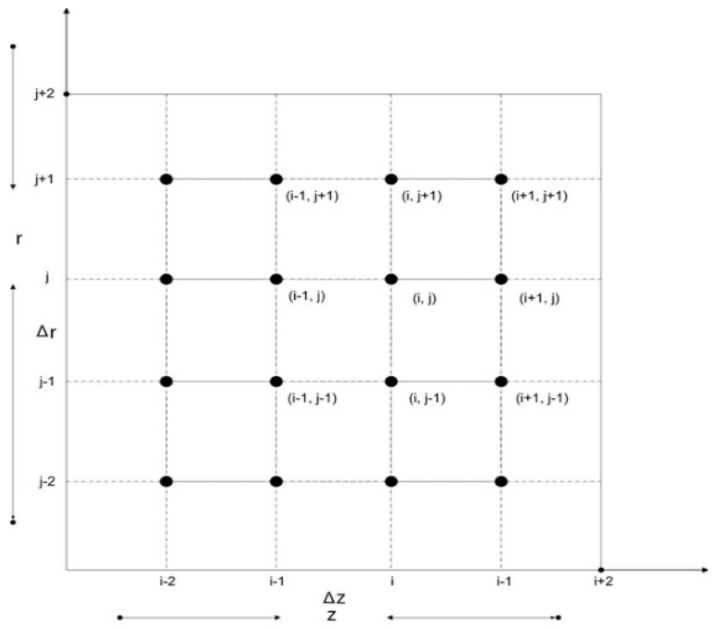

Figure 2: Finite Difference Grid for solid cylinder

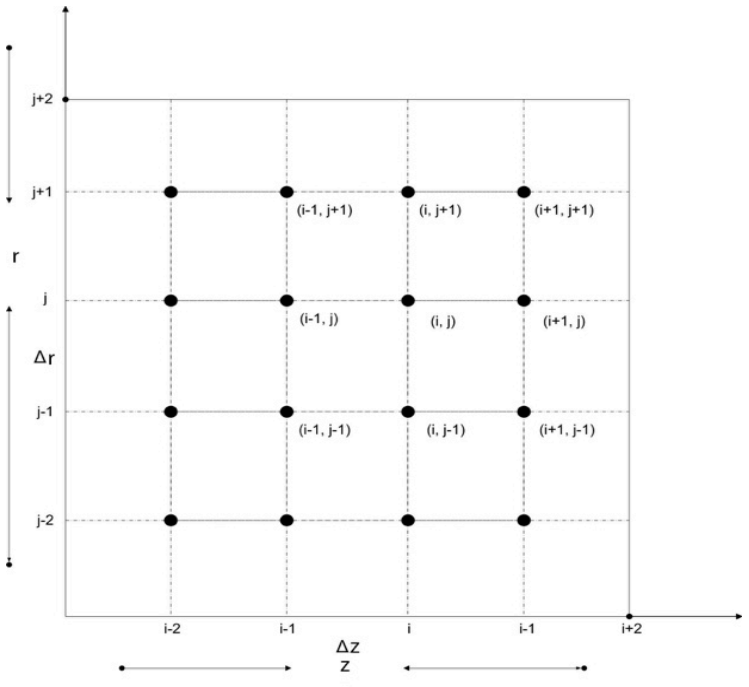

Figure 3: Finite Difference Grid for hollow cylinder

The shaded portion of the grid represents the hollow in the cylinder. For the purposes of coming up with a numerical solution for the problem, the height of the hollow is 0.1, while the radius of the cylinder is 1.0.

$$\left(\frac{\partial^2 T}{\partial z^2}\right)_{i,j} = \frac{T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1} - 2T_{i,j}}{2(\Delta z)^2} \quad \left(\frac{\partial^2 T}{\partial r^2}\right)_{i,j} = \frac{T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1} - 2T_{i,j}}{2(\Delta r)^2}$$

$$\left(\frac{\partial T}{\partial r}\right)_{i,j} = \frac{T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1}}{4(\Delta r)} \qquad \left(\frac{\partial T}{\partial t}\right)_{i,j} = \frac{T_{i,j+1} - T_{i,j}}{\Delta t}, \left(\frac{\partial C}{\partial t}\right)_{i,j} = \frac{C_{i,j+1} - C_{i,j}}{\Delta t}, \left(\frac{\partial u}{\partial t}\right)_{i,j} = \frac{u_{i,j+1} - u_{i,j}}{\Delta t}$$

$$\left(\frac{\partial C}{\partial t}\right)_{i,j} = \frac{C_{i+1,j} - C_{i-1,j} + C_{i+1,j+1} - C_{i-1,j+1}}{4(\Delta t)} \qquad \left(\frac{\partial^2 C}{\partial z^2}\right)_{i,j} = \frac{C_{i+1,j} - C_{i-1,j} + C_{i+1,j+1} - C_{i-1,j+1} - 2C_{i,j}}{2(\Delta z)^2}$$

$$\left(\frac{\partial^2 C}{\partial r^2}\right)_{i,j} = \frac{C_{i+1,j} - C_{i-1,j} + C_{i+1,j+1} - C_{i-1,j+1}}{2(\Delta r)^2} \qquad \left(\frac{\partial C}{\partial r}\right)_{i,j} = \frac{C_{i+1,j} - C_{i-1,j} + C_{i+1,j+1} - C_{i-1,j+1}}{4(\Delta r)}$$

The finite difference approximation of Equations (9) and (10) are obtained with substituting Equation (14) into Equations (9) and (10) and multiplying both sides by Δt and after simplifying, we let $\frac{\Delta t}{(\Delta z)^2} = r' = 1$ (method is always stable and convergent), under this condition the above equations can be written as:

$$\frac{\partial C}{\partial t} = \frac{1}{2}\left(\frac{U+V-2(T_{i,j}+C_{i,j})}{(\Delta r)^2} + \frac{U+V}{2r(\Delta r)} + \left(\frac{U+V-2(T_{i,j}+C_{i,j})}{(\Delta z)^2}\right)\right.$$

$$\frac{\partial T}{\partial t} = \frac{1}{2}\left(\frac{2U+V-2(2T_{i,j}+C_{i,j})}{(\Delta r)^2} + \frac{2U+V}{2r(\Delta r)} + \frac{2U+V-2(2T_{i,j}+C_{i,j})}{(\Delta z)^2}\right)$$

Let

$$\mathrm{U} = T_{i+1,j} - T_{i-1,j} + T_{i+1,j+1} - T_{i-1,j+1}$$

and

$$\mathrm{V} = C_{i+1,j} - C_{i-1,j} + C_{i+1,j+1} - C_{i-1,j+1}$$

3 Experimental Results and Discussion

3.1 Setup and Device Configuration

The experiment was executed using the CUDA Runtime Library, Quadro FX 4800 graphics card, Intel Core 2 Duo. The programming interface used was Visual Studio. The experiments were performed using a 64-bit Lenovo ThinkStation D20 with an Intel Xeon CPU E5520 with processor speed of 2.27 GHZ and physical RAM of 4.00GB. The Graphics Processing Unit (GPU) used was an NVIDIA Quadro FX 4800 with the following specifications:

CUDA Driver Version:	3.0
Total amount of global memory:	1.59 Gbytes
Number of multiprocessors:	24
Number of cores:	92
Total amount of constant memory:	65536 bytes

Total amount of shared memory per block: 16384 bytes
Total number of registers available per block: 16384
Maximum number of threads per block: 512
Bandwidth:
Host to Device Bandwidth: 3412.1 (MB/s)
Device to Host Bandwidth: 3189.4 (MB/s)
Device to Device Bandwidth: 57509.6 (MB/s)

In the experiments, we considered solving heat and mass transfer differential equations in solid and hollow capillary porous cylinder with boundary conditions of second kind using numerical methods. Our main purpose here was to obtain numerical solutions for Temperature *T*, and concentration *C* distributions across the various points in a cylinder as heat and mass are transferred from one end of the cylinder to the other. For our experiment, we compared the similarity of the CPU and GPU results. We also compared the performance of the CPU and GPU in terms of processing times of these results.

In the experimental setup, we are given the initial temperature T_0 and concentration C_0 at point $z = 0$ on the cylinder. Also, there is a constant temperature and concentration N_0 constantly working the surface of the cylinder. The temperature at the other end of the cylinder where $z = \infty$ is assumed to be ambient temperature (assumed to be zero). Also, the concentration at the other end of the cylinder where $z = \infty$ is assumed to be negligible (≈ 0). Our initial problem was to derive the temperature T_1 and concentration C_1 associated with the initial temperature and concentration respectively. We did this by employing the finite difference technique. Hence, we obtained total initial temperature of $(T_0 + T_1)$ and total initial concentration of $(C_0 + C_1)$ at $z = 0$. These total initial conditions were then used to perform calculations.

For the purpose of implementation, we assumed a fixed length of the cylinder and varied the number of nodal points *N* to be determined in the cylinder. Since *N*

is inversely proportional to the step size Δz, increasing N decreases Δz and therefore more accurate results are obtained with larger values of N. For easy implementation in Visual Studio, we employed the Forward Euler Method (FEM) for forward calculation of the temperature and concentration distributions at each nodal point in both the CPU and GPU. For a given array of size N, the nodal points are calculated iteratively until the values of temperature and concentration become stable. In this experiment, we performed the iteration for 10 different time steps. After the tenth step, the values of the temperature and concentration became stable and are recorded. We run the tests for several different values of N and Δz and the error between the GPU and CPU calculated results were increasingly smaller as N increased. Finally, our results were normalized in both the GPU and CPU.

3.2 Experimental Results

The normalized temperature and concentration distributions at various points in the solid cylinder are depicted in Table 1 and Table 2, and hollow cylinder are depicted in Table 3 and Table 4 respectively. We can immediately see that, at each point in the cylinder, the CPU and GPU computed results are similar. In addition, the value of temperature is highest and the value of concentration is lowest at the point on the cylinder where the heat resource and mass resource are constantly applied. As we move away from this point, the values of the temperature decrease and concentration increase. At a point near the designated end of the cylinder, the values of the temperature approach zero and concentration approach one.

Table 1: Comparison of GPU and CPU Results for Solid Cylinder (Temperature)

Z	CPU RESULTS	GPU RESULTS
7.813	0.697518	0.697419
15.625	0.437492	0.437392
29.688	0.122683	0.122583

37.500	0.047844	0.047644
45.313	0.016684	0.016584
53.125	0.006823	0.006723
60.938	0.003061	0.002961
68.750	0.002110	0.002010
76.563	0.001917	0.001817
82.813	0.002074	0.001774
89.063	0.001860	0.001760
96.875	0.001856	0.001756
104.690	0.001850	0.001750
129.690	0.001657	0.001558
143.750	0.000897	0.000797
150.000	-	-

Table 2: Comparison of GPU and CPU Results for Solid Cylinder (Concentration)

Z	CPU RESULTS	GPU RESULTS
7.813	0.000470	0.000464
15.625	0.000581	0.000575
29.688	0.000645	0.000638
37.500	0.000653	0.000645
45.313	0.000657	0.000648
53.125	0.000659	0.000649
60.938	0.000656	0.000651
68.750	0.000671	0.000665
76.563	0.000732	0.000725
82.813	0.000909	0.000901
89.063	0.001629	0.001620
96.875	0.004753	0.004743
104.690	0.015492	0.015482
129.690	0.244210	0.244200
143.750	0.696986	0.696978
150.000	1.000000	1.000000

Table 3: Comparison of GPU and CPU Results for hollow Cylinder (Concentration)

Z	GPU RESULTS	CPU RESULTS
7.813	0.000487	0.000488
15.625	0.001130	0.001132
29.688	0.002084	0.002087
37.5	0.002567	0.002571
45.313	0.003079	0.003084
53.125	0.003671	0.003677
60.938	0.004812	0.004813
68.75	0.006331	0.006333
76.563	0.010615	0.010618
82.813	0.014678	0.014682
89.063	0.025268	0.025273
96.875	0.044846	0.044852
104.69	0.095074	0.095081
129.69	0.448736	0.448737
143.75	0.849127	0.849128
150	1.000000	1.000000

Table 4: Comparison of GPU and CPU Results for hollow Cylinder (Tempreture)

Z	GPU RESULTS	CPU RESULTS
7.813	0.774215	0.774216
15.625	0.504365	0.504366
29.688	0.2184	0.218401
37.500	0.132061	0.132062
45.313	0.076065	0.076066
53.125	0.034741	0.034747
60.938	0.016236	0.016237
68.750	0.011722	0.011724

76.563	0.007502	0.007503
82.813	0.006701	0.006702
89.063	0.006239	0.006240
96.875	0.006163	0.006169
104.690	0.006122	0.006130
129.690	0.004154	0.004155
143.750	0.00117	0.001171
150.000	0	0

Figure 3a and Figure 3b Shows the temperature and concentration distribution in the solid cylinder with 4 different radiuses.

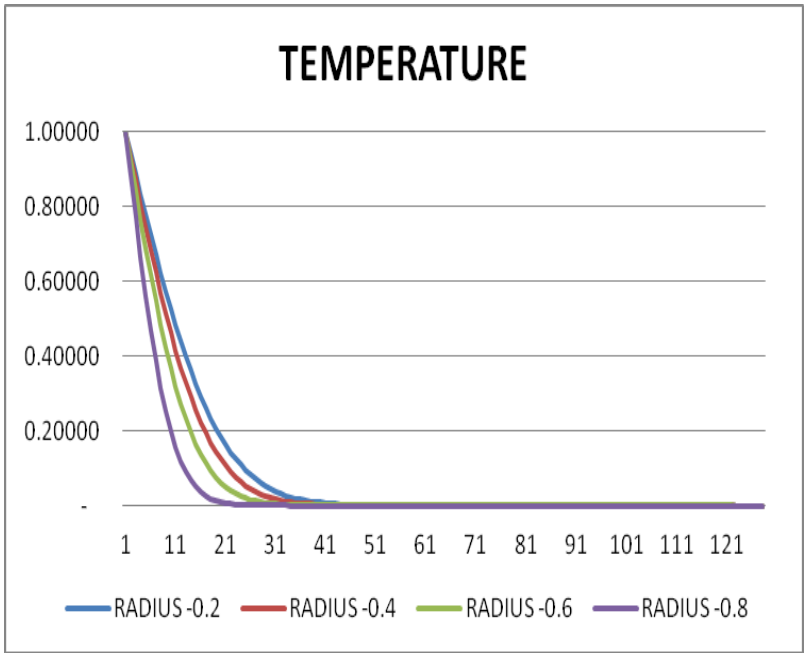

Figure 3a: Temperature for Solid Cylinder with 4 Different Radiuses

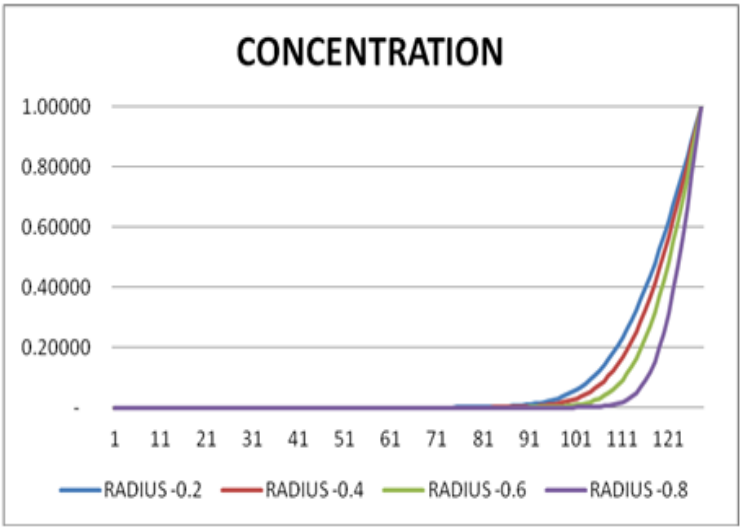

Figure 3b: Concertration for Solid Cylinder with 4 Different Radiuses

Figure 4a and Figure 4b shows the temperature and concentration distribution in the hollow cylinder with 4 different radiuses.

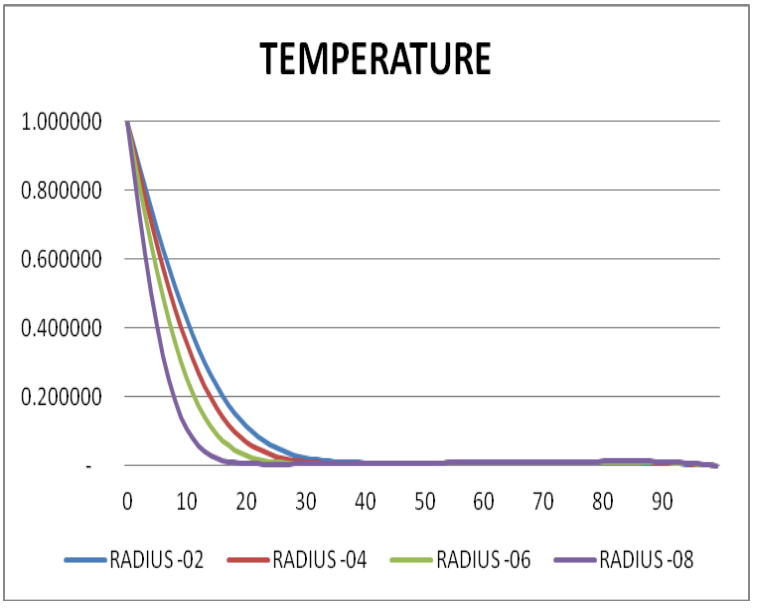

Figure 4a: Temperature for Hollow Cylinder with 4 Different Radiuses

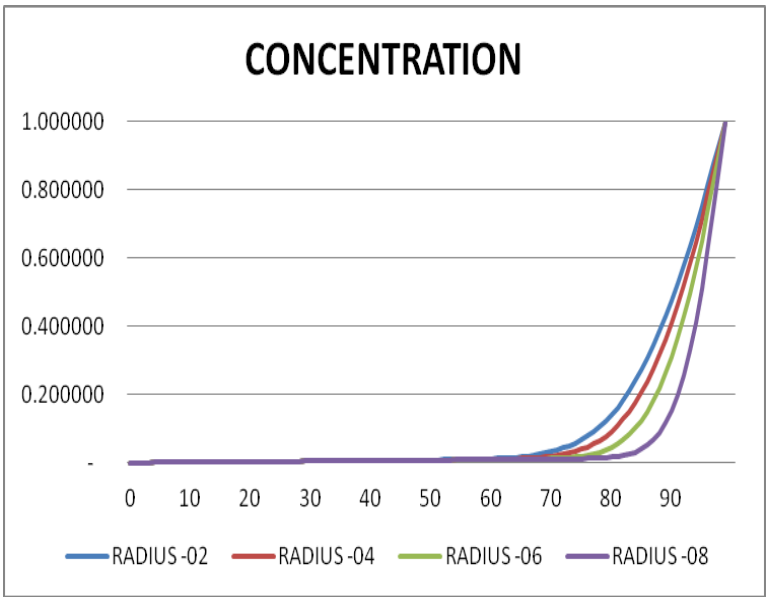

Figure 4b: Concertration for Hollow Cylinder with 4 Different Radiuses

Furthermore, we also evaluated the performance of the GPU (NVIDIA Quadro FX 4800) in terms of solving heat and mass transfer equations by comparing its execution time to that of the CPU (Intel Xeon E5520).

For the purpose of measuring the execution time, the same functions were implemented in both the device (GPU) and the host (CPU), to initialize the temperature and concentration and to compute the numerical solutions. In this case, we measured the processing time for different values of N. The graph in Figure 5 depicts the performance of the GPU versus the CPU in terms of the processing time. We run the test for N running from 15 to 11500 with increments of 30 and generally, the GPU performed the calculations a lot faster than the CPU.

- When N was smaller than 3500, the CPU performed the calculations faster than the GPU.
- For N larger than 3500 the GPU performance began to increase considerably.

Figure 5 and Figure 6 show some of our experimental results for both solid and hollow cylinder.

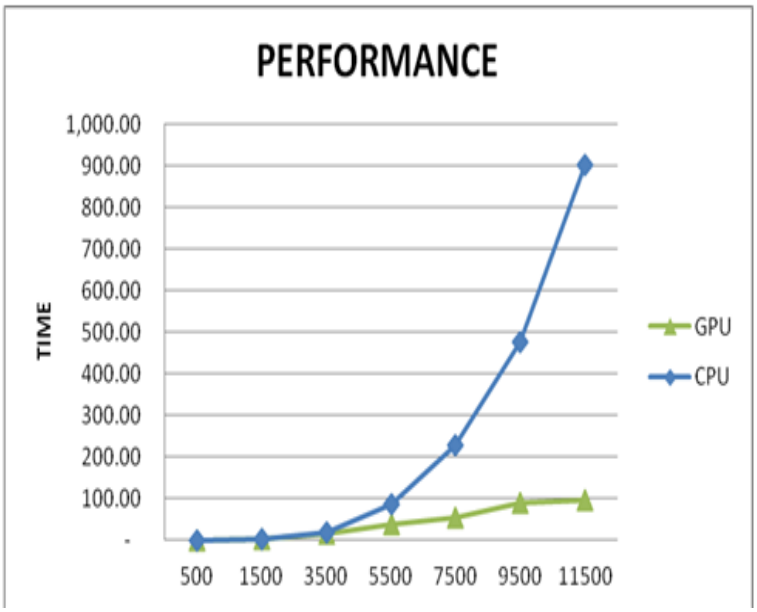

Figure 5: Performance of GPU and CPU Implementations for solid cylinder

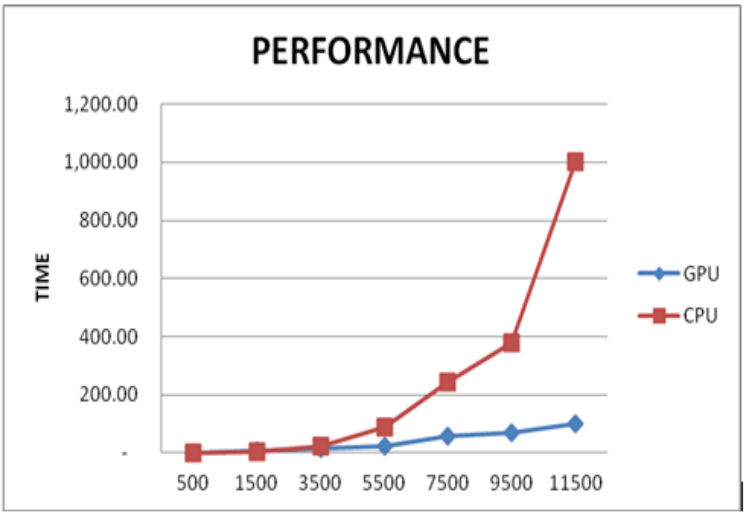

Figure 6: Performance of GPU and CPU Implementations for hollow cylinder

Finally, the accuracy of our numerical solution was dependent on the number of iterations we performed in calculating each nodal point, where more iteration means more accurate results. In our experiment, we observed that after 9 or 10

iterations, the solution to the heat and mass equation at a given point became stable. For optimal performance, and to keep the number of iterations the same for both CPU and GPU, we used 10 iterations. Both groups of experimental results for solid and hollow cylinder show about 10 times speed-up.

4 Conclusion and Future Work

We have presented our numerical approximations to the solution of the heat and mass transfer equation with the second kind of boundary conditions for both solid and hollow cylinder using finite difference method on GPGPUs. Our conclusion shows that finite difference method is well suited for parallel programming. We implemented numerical solutions utilizing highly parallel computations capability of GPGPU on nVidia CUDA. We have demonstrated GPU can perform significantly faster than CPU in the field of numerical solution to heat and mass transfer. Both groups of experimental results for solid and hollow cylinder indicate that our GPU-based implementation shows a significant performance improvement over CPU-based implementation and the maximum observed speedups are about 10 times.

There are several avenues for future work. We would like to test our algorithm on different GPUs and explore the new performance opportunities offered by newer generations of GPUs. It would also be interesting to explore more tests with large-scale data set. Finally, further attempts will be made to explore more complicated problems both in terms of boundary and initial conditions as well as other geometry.

References

[1] J.D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krger, A.E. Lefohn and T.J. Purcell, A survey of general-purpose computation on graphics hardware, *Computer Graphics Forum*, **26**(1), (2007), 80-113.

[2] NVIDIA Corporation, NVIDIA Programming Guide 2.3. Retrieved July, 2009. www.nvidia.com.

[3] A.V. Luikov, *Heat and Mass Transfer in Capillary Porous Bodies*, Pergamon Press, 1966.

[4] Hira Narang and Rajiv Nekkanti, Wavelet-based Solution to Time-dependent Two-point Initial Boundary Value Problems with Non-Periodic Boundary Conditions, *Proceedings of the IATED International Conference Signal Processing, Pattern Recognition & Applications*, (July 3-6, 2001), Rhodes, Greece.

[5] Hira Narang and Rajiv Nekkanti, Wavelet-based Solution of Boundary Value Problems involving Hyperbolic Equations, *Proceedings from the IATED International Conference Signal Processing, Pattern Recognition & Applications*, (June 25-26, 2002).

[6] Hira Narang and Rajiv Nekkanti, Wavelet-based solutions to problems involving Parabolic Equations, *Proceedings of the IATED International Conference Signal Processing, Pattern Recognition & Applications*, (2001), Greece.

[7] Hira Narang and Rajiv Nekkanti, Wavelet-Based Solution to Elliptic Two-Point Boundary Value Problems with Non-Periodic Boundary Conditions, *Proceedings from the* WSEAS *international conference in Signal, Speech, and Image processing*, (Sept. 25-28, 2002).

[8] Hira Narang and Rajiv Nekkanti, Wavelet-Based Solution to Some Time-Dependent Two-Point Initial Boundary Value Problems with Non-Linear Non-Periodic Boundary Conditions, *International Conference on*

Scientific computation and differential equations, SCICADE 2003, Trondheim, Norway, (June 30-July 4, 2003).

[9] Hira Narang and Rajiv Nekkanti, Wavelet based Solution to Time-Dependent Two Point Initial Boundary Value Problems with Non-Periodic Boundary Conditions involving High Intensity Heat and Mass Transfer in Capillary Porous Bodies, IATED *International Conference Proceedings*, Gainesville, FL (2004).

[10] Vishwavidyalaya Ambethkar, Numerical Solutions of Heat and Mass Transfer Effects of an Unsteady MHD Free Conective Flow Past an Iffinite Vertical Plate With Constant Suction, *Journal of Naval Architecture and Marine Engineering*, (June, 2008), 28-36.

[11] Jens Krüger and Rüdiger Westermann, Linear Algebra Operators for GPU Implementation of Numerical Algorithms, *ACM Transactions on Graphics, Proceedings of* SIGGRAPH, (July, 2003), 908-916.

[12] Jeff Bolz, Ian Farmer, Eitan Grinspun and Peter Schröoder, Sparse Matrix Solvers on the GPU: Conjugate Gradients and Multigrid, *ACM Transactions on Graphics, Proceedings of* SIGGRAPH, (July, 2003), 917-924.

[13] Nolan Goodnight, Cliff Woolley, David Luebke and Greg Humphreys. A Multigrid Solver for Boundary Value Problems Using Programmable Graphics Hardware, *Proceeding of Graphics Hardware*, (July, 2003), 102-111.

[14] Mark Harris, William Baxter, Thorsten Scheuermann and Anselmo Lastra. Simulation of Cloud Dynamics on Graphics Hardware, *Proceedings of Graphics Hardware*, (July, 2003), 92-101.

[15] Mark Harris, *Real-Time Cloud Simulation and Rendering*, PhD thesis, 2003.

[16] Theodore Kim and Ming Lin, Visual Simulation of Ice Crystal Growth, *Proceedings of* SIGGRAPH/Eurographics *Symposium on Computer Amination*, (July, 2003), 86-97.

[17] Aaron Lefohn, Joe Kniss, Charles Hansen and Ross Whitaker, Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware, IEEE *Visualization*, (2003), 75-82.

[18] GPGPU website http://www.gpgpu.org.