Implementation of Airy function using Graphics Processing Unit (GPU)

Yugesh C. Keluskar[1], Megha M. Navada[2], Chaitanya S. Jage[3] and Navin G. Singhaniya[4]
Department of Electronics Engineering,
Ramrao Adik Institute of Technology,
Nerul, Navi Mumbai.
Email : yugeshkeluskar.yk@gmail.com[1], meghanavada@gmail.com[2], chaitanya.jage@rait.ac.in[3], navin.singhaniya@rait.ac.in[4]

***Abstract*—Special mathematical functions are an integral part of Fractional Calculus, one of them is the Airy function. But it's a gruelling task for the processor as well as system that is constructed around the function when it comes to evaluating the special mathematical functions on an ordinary Central Processing Unit (CPU). The Parallel processing capabilities of a Graphics processing Unit (GPU) hence is used. In this paper GPU is used to get a speedup in time required, with respect to CPU time for evaluating the Airy function on its real domain. The objective of this paper is to provide a platform for computing the special functions which will accelerate the time required for obtaining the result and thus comparing the performance of numerical solution of Airy function using CPU and GPU.**

Keywords- GPU, CPU, Airy function, Speedup.

I. INTRODUCTION

Fractional calculus (FC) is a field of mathematics that deals with derivatives and integrals of arbitrary non-integer order. It is being used in engineering domain extensively due to its ability to model real-world systems more accurately than its integer order variant. It is a topic which is around 320 years old!. The inception of this field can be attributed to a letter written to Guillaume de l'Hopital by Gottfried Leibniz in the year 1695 in which he mentions about fractional order derivatives, see [26], one can also find the foundational work of Fractional calculus in the early papers of Niels Henrik Abel [27]. But due to its mathematically intricate and cumbersome nature it remained mostly unexplored for many years until 1974 when the "First Conference on Fractional Calculus and its Applications" was held at the University of New Haven,Connecticut [28]. In recent years applications of this are growing nowadays in a rapid manner as there is a community of researchers that have came together to revive this field with the help of when advanced computing technology [1],[2]. As mentioned before using this mathematical technique, the scope of describing a real object is more accurate as compared to the classical integer-order methods. The replacement of integer orders with fractional orders has resulted into various realistic and compact model in the field of various physical, engineering, automation, biomedical, chemical engineering according to the mention made in this book [13], The function can be called a "Special Function" only when it has the ability to be useful in some applications and satisfies certain special properties. Some commonly used special functions are Gamma, Gauss Hypergeometric, Airy, Bessel,etc [2].

Airy function is the solution to Airy differential equation,which has the following form [22]:

$$\frac{d^2y}{dx^2} = xy \tag{1}$$

It has its applications in classical physics and mainly quantum physics. The Airy function also appears as the solution to many other differential equations related to elasticity, heat equation, the Orr-Sommerfield equation, etc in case of classical physics. While in the case of Quantum physics,the most well-known application of this equation is while solving WKB approximation problem [3],[24]. Taking a few steps we can arrive at the Airy Differential equation as given below. The following is the One-Dimensional Time Independent Schrödinger's equation

$$\frac{-\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\psi(x) = En\psi(x) \tag{2}$$

Where $\psi(x)$ is the wave-function, $\hbar$ is the reduced Planck's constant (Dirac constant), m being mass of the particle, $V(x)$ is the potential function and E is the total energy. This equation basically says that Kinetic Energy + Potential Energy = Total Energy, and that is what each term in this equation signifies. When we consider a triangular potential well i.e $V(x) = q\varepsilon x$ (q is charge of particle and ε is the electric field strength), we arrive at the following equation

For $x > 0$, $n = 1, 2, ..$

$$\frac{-\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + q\varepsilon x\psi(x) = En\psi(x) \tag{3}$$

The above equation is the same as equation (1) with some constant terms mugging up the equation of which the Airy function is the solution. Hence we can say that for the Schrödinger equation with triangular potential well's case the solution is Airy function. However, to our knowledge, the computational complexity of such Special Mathematical Functions included in the fractional calculus increases when long time vectors are considered while implementing. It is interesting to make a platform which will allow us to speedup the performance rate and thus use it in various hardware implementations of the functions. This process cannot be automated as the definition of the function is supposed to be

written from scratch by a human, hence human intervention is required. For this purpose the concept of parallel computing plays a crucial role. In parallel computing, many arithmetic and logical operations are carried out simultaneously using multi-core processors. Traditionally, parallel computing has been carried out using the CPU with a handful of cores. But, we are using the capabilities of multi-core graphics processors for parallel computing.

Even though the original purpose of GPU is for graphics rendering, in recent years the GPU is increasingly being used for scientific computations. This practice of using GPU for general purpose computations is called General Purpose Computing on Graphics Processing Unit (GPGPU) Technology. In paper [4] a parallel GPU solution of the Caputo fractional reaction-diffusion equation in one spatial dimension with explicit finite difference approximation has been provided. The optimized GPU solution obtained is faster than the optimized parallel CPU solution. So the power of parallel computing on GPU for solving fractional applications can be recognized. The paper, see [5] gives a description of the numerical approach that has been done for the evolution of algorithms which can do multiple operations in a given time, in order to solve the fractional differential equations. The procedure accepted is to fit the existing numerical schemes and to develop model parallel algorithms by utilising Parallel Computing toolbox of MATLAB. Unlike CPU, Graphics Processing Unit has parallel arrays of hundreds of smaller processors which function in parallel. We have utilised this parallel architecture of GPU to accelerate the execution of the above mentioned special function. The solution methods for fractional differential equations are not present. This absence is the main reason behind using the models of integer-order. In recent years, the field Fractional Calculus and its applications have drawn the interest of many authors.

The rest of the paper is organized as follows: In section 2, the basic concept of Airy function has been presented briefly. In section 3, a detailed explanation of GPU architecture has been written which also includes the tools for MATLAB computing with GPU. Section 4 represents the implementation steps which were performed while designing the algorithm. Finally all the results are given in section 5 and the conclusions drawn from this work are presented in section 6.

II. Basic Concept of Airy function

The Airy function is named after Sir George Bidell Airy [3].He stumbled upon this following integral while calculating the intensity of light in the neighbourhood of a caustic. For this purpose, he introduced the function defined by the integral given below

$$W(m) = \int_0^\infty cos(\frac{\pi}{2}(w^2 - mw))dw \tag{4}$$

which is now called the Airy function. $W(m)$ is the solution of the differential equation

$$W'' = -\frac{\pi^2}{12}mW$$

The numerical calculation of Airy function was very tricky back in those days and even nowadays it is so. But in 1838, the values of W for m varying from -4.0 to 4.0 were given by Airy and in the year it was given for values ranging from -5.6 to 5.6. The problem which arose was that the series converges slowly as m increases [3]. In 1982, a scientist named Jeffrey's introduced the notation $Ai(x)$ which is used nowadays as

$$Ai(x) = \frac{1}{\pi}\int_0^\infty cos(\frac{t^3}{2} + xt)dt \tag{5}$$

Two linearly independent solutions of equation (1) which are real when x is real are $Ai(x)$ and $Bi(x)$. For the function $Bi(x)$, the integral representation is,

$$Bi(x) = \frac{1}{\pi}\int_0^\infty e^{\frac{-t^3}{3+xt}} + sin(\frac{t^3}{3} + xt)dt \tag{6}$$

The solutions $Ai(x)$ and $Bi(x)$ satisfy the initial conditions

$$Ai(0) = \frac{3^{\frac{2}{3}}}{\Gamma(\frac{2}{3})} \tag{7}$$

$$Bi(0) = \frac{3^{\frac{-1}{6}}}{\Gamma(\frac{2}{3})} \tag{8}$$

Also the behaviour of the first kind of Airy function which is $Ai(x)$ can be described as follows:

For negative values, it has an oscillatory behaviour while for positive values, it exponentially decays. It is also observed that there is an interesting and unique smooth transition between the oscillatory behaviour and exponential decay without any abrupt change in the transition.

For the 2nd kind of Airy function $Bi(x)$ the behaviour for negative values is the same as the 1st kind but it is with a phase shift of $\frac{\pi}{2}$ and for positive values it grows exponentially. Airy function is related to Bessel Functions [25] of non integer order of first kind $J_\alpha(x)$ and to non integer order of modified Bessel function $I_\alpha(x)$ and $K_\alpha(x)$ of order $\frac{1}{3}$ which is implemented with change of the variable $\zeta = \frac{2}{3}x^{\frac{3}{2}}$. The relation between Airy functions and Bessel functions for $x > 0$ is given as

$$Ai(x) = \frac{1}{\pi}\sqrt{\frac{x}{3}}K_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) \tag{9}$$

$$Bi(x) = \sqrt{\frac{x}{3}}(I_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) + I_{-\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}})) \tag{10}$$

For the values of $x < 0$, the functions $Ai(x)$ and $Bi(x)$ are given by

$$Ai(-x) = \sqrt{\frac{x}{9}}(J_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) + J_{-\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}})) \tag{11}$$

$$Bi(-x) = \sqrt{\frac{x}{3}}(J_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) - J_{-\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}})) \tag{12}$$

While computing the time required for implementation of these functions in CPU as well as GPU, the definitions of Airy function which we used were the one with Bessel and modified Bessel function (In this experiment every function was written in a different subroutine). The reason behind doing so is simple. For calculating Airy function using the integral definition we have to depend on integration techniques like Runge–Kutta–Fehlberg method to get desired accuracy [29]. Contrary to it by using Bessel functions we eliminate the use of numerical integration as the definition of Bessel functions used for computing Airy function are in terms of a power series [14]. The Bessel function of the first kind $J_\alpha(x)$ and Modified Bessel function of non-integer order $I_\alpha(x)$ and $K_\alpha(x)$ used in Airy function are defined as

$$J_\alpha(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\alpha+1)} (\frac{x}{2})^{2m+\alpha}$$

$$I_\alpha(x) = (\frac{x}{2})^\alpha \sum_{k=0}^{\infty} \frac{(\frac{x^2}{4})^k}{k!\Gamma(k+\alpha+1)}$$

$$K_\alpha(x) = \frac{\pi}{2} \frac{I_{-\alpha}(x) - I_\alpha(x)}{\sin(\alpha\pi)}$$

Here Γ(x) is the Gamma Function. The Bessel functions have there applications in Fluid dynamics, Electrodynamics and mainly in Acoustics [25].

III. Graphics Processing Unit (GPU)

Graphic rendering is all about compute intensiveness and highly parallel computation. These are the features or specialization of Graphic Processor unit or GPU [6]. The process of generation of an image from a 2D or 3D model by means of computer programs is called rendering. This process needs a huge number of calculations to be performed in fraction of seconds. GPU has developed gradually into a highly parallel and multi threaded processor which consists of many cores. The computational horsepower of GPU is enormous and also it has high memory bandwidth. The floating point capability of GPU and CPU has inconsistency between them. Not only this but there are also differences between the architecture of GPU and traditional CPU. The CPU provides more resources to make single instruction execute faster, whereas the GPU makes use of hundreds of individual processing elements (cores) to execute single instruction on multiple data elements (elements in array) in parallel [2]. It is designed in such a way that more transistors are devoted to data processing rather than data caching and flow control.

To get a speedup in computation time, the applications consisting of large data sets make use of the platform of data parallel programming, the applications of processing of image and media which includes the post-process of image synthesis, video encoding and decoding, resizing of a digital image, stereo vision and automated recognition of patterns and regularities in data which maps the image blocks and pixels to parallel processing threads.It is quite obvious that a GPU will take less time for computing as compared to CPU, but the catch is to write a program by utilising special function/subroutines

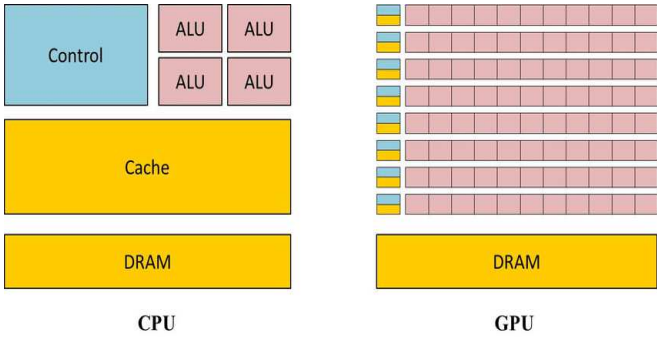

Figure 1: GPU architecture.

in such a way that it uses GPU architecture efficiently. In fact, acceleration is provided by data parallel processing for the algorithms which are present outside the field of rendering and processing of image. The GPU used in order to carry out the work represented in this paper is NVIDIA Tesla K80.

Specifications of NVIDIA Tesla K80 GPU

The following are the GPU device specifications used during computing and evaluating the Airy function.

Table I: NVIDIA Tesla K80 GPU specifications.

Parameters	Specifications
Peak Double Precision Floating point precision	2.91 Tflops
Peak Single Precision Floating point precision	8.74 Tflops
Memory Bandwidth(ECC Off)	480GB/sec
Memory size and Type	24GB GDDR5
CUDA corecores	4992
Thermal solution	Passive heat sink

Computation of GPU using MATLAB

The Matrix Laboratory, MATLAB is a programming platform constructed especially for engineers and scientists. It makes use of the MATLAB language, a matrix based language, granting the most logical expression of computational mathematics. MATLAB is used by millions of engineers worldwide. The reason behind its usage is to analyse and design the systems and products for revolutionizing our world. As MATLAB's operation is on whole matrices and arrays, it is one of the efficient ways to express computational mathematics. For envision and to obtain the insights from data, built in graphics are utilised which furthermore makes it an easy job. The desktop environment (DE) gives an opportunity to experiment, analyse and discover. These tools of MATLAB and its effectiveness are all precisely tested and constructed to work together [7]. Parallel Computing Toolbox present in MATLAB lets one to solve computationally and data-intensive problems using multi core processors like GPUs. Some of the High level constructs like parallel for loops and parallelized numerical algorithms enables us to parallelize MATLAB applications. The toolbox allows one to use the full processing power of multi core desktops by executing applications on workers that run locally [8]. The Parallel Computing Toolbox (PCT)

requires the NVIDIA CUDA enabled device with compute capability of 1.3 or greater [2].

IV. Design Of implementation

Some of the functions used while computing in GPU are as follows:

- gpuArray: This function is used to transfer the input data from MATLAB workspace to GPU. This function converts the array in the MATLAB workspace into a gpuArray object [8].
- gather: For transferring the data from GPU to MATLAB workspace this function is used. The gather function is used after all the operations on the data are performed by GPU.
- gpuDevice: For displaying the properties of the currently selected GPU device, the command gpuDevice is used.

Following are the steps to be followed while running a MATLAB code on GPU:

1) Start with the correct code which gives appropriate results.
2) Perform Vectorization of the code.
3) Profile the code using Run and Time function.
4) Variables should be transferred to GPU memory using gpuArray() function.
5) After the execution of code is done the output variables must be transferred to CPU memory by using gather() function.

The definitions of Airy function used while implementing it are as follows:

Ai(x) =

$$\begin{cases} \sqrt{\frac{x}{9}}(J_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) + J_{-\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}})) & , x < 0 \\ \frac{3^{\frac{2}{3}}}{\Gamma(\frac{2}{3})} & , x = 0 \\ \frac{1}{\pi}\sqrt{\frac{x}{3}}K_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) & , x > 0 \end{cases}$$

Bi(x) =

$$\begin{cases} \sqrt{\frac{x}{3}}(J_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) - J_{-\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}})) & , x < 0 \\ \frac{3^{-\frac{1}{6}}}{\Gamma(\frac{2}{3})} & , x = 0 \\ \sqrt{\frac{x}{3}}(I_{\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}}) + I_{-\frac{1}{3}}(\frac{2}{3}x^{\frac{3}{2}})) & , x > 0 \end{cases}$$

The following algorithm represents steps of implementation of Airy function:

1) Start with the definition of Airy function used.
2) Write a pseudo-code for the function on paper.
3) Write the actual code for the function in MATLAB and obtain the output.
4) Do the calculations of Airy function using Bessel function manually.
5) Compare the output thus obtained with that of MATLAB's result.
6) If the answer is not of the desired precision, make the necessary changes in the code.
7) Prepare a vectorized code for the same (for GPU).
8) Dump the code in GPU and thus calculate the time required for implementation.
9) On calculating the time required for the same code in CPU, obtain the Speedup parameter by using the following formula:

$$Speedup = \frac{CPUTime(T_c)}{GPUTime(T_g)} \tag{13}$$

10) Compile and analyse the results.

The Airy function is implemented on CPU and similarly on GPU by following the steps given in the above algorithm. Vectorization of code is important while implementing the function in GPU because it eliminates loops which are sequentially executed. All the operations performed on the data transferred to GPU memory are executed on GPU in parallel by using MATLAB's PCT. The above algorithm can be generalized for any function.

V. Results and Analysis

In this section a rigorous comparative study of computation time required for evaluating the Airy function on NVIDIA Tesla K80 CPU and GPU is presented. It should be taken into account that both functions (Airy function of first kind and second kind) were evaluated on total 9 test input in ascending order as shown in table 2 and 3 for 10000 times on CPU as well as GPU, and the computational time was noted down for both platforms. After that a mean was calculated for each input and listed in tables below. The reason behind doing so is to eliminate any statistical anomaly or error that might have occurred unknowingly while calculating the computation time. At last the Speedup parameter is calculated by the formula given in the algorithm described in section 4.

The following two tables present the computational time and the Speedup parameter for the purpose as discussed above. The table 2 is for $Ai(x)$ and table 3 is for $Bi(x)$. We can infer from table 2 that for $Ai(x)$ the Speedup parameter calculated using the device used in this experiment gives an improvement in computation time (for largest input) of around 2.667 times the computation time required for the CPU to evaluate the same function with the same input on GPU. This is the crux of using GPUs multi-core architecture instead of single or dual core CPUs. Similarly for $Bi(x)$ the table 3 shows that the calculated Speedup parameter is around 2.601 for the same system as mentioned above. The Speedup parameter depends on various factors that can decrease its value. As mentioned in section 2, for every function a different subroutine was used to evaluate the code, because of the complexity of the Airy function it calculates a total of 5 subroutines to evaluate the final $Ai(x)$ or $Bi(x)$ value. The function was not hardcoded (implementing Bessel and Airy functions into one single subroutine) as in real world application where this function might be used there would be an existing definition of the Bessel function.

Table II: Speedup parameter for Airy function of first kind.

Argument for $Ai(x)$	CPU Time (T_c in ms)	GPU Time (T_g in ms)	Speedup
1	0.494	0.186	2.655913978
2	0.475	0.184	2.581521739
3	0.474	0.183	2.590163934
4	0.473	0.182	2.598901099
5	0.473	0.181	2.613259669
10	0.467	0.181	2.580110497
50	0.455	0.177	2.570621469
100	0.447	0.174	2.568965517
1000	0.448	0.168	2.666666667

Table III: Speedup parameter for Airy function of Second kind.

Argument for $Bi(x)$	CPU Time (T_c in ms)	GPU Time (T_g in ms)	Speedup
1	0.468	0.191	2.450261783
2	0.463	0.190	2.436842105
3	0.461	0.189	2.439153439
4	0.460	0.185	2.486486486
5	0.457	0.184	2.483695652
10	0.453	0.180	2.516666667
50	0.450	0.176	2.556818182
100	0.447	0.173	2.583815029
1000	0.437	0.168	2.601190476

VI. Conclusion

The work which was carried out was the acceleration of computation of Airy function on GPU. The acceleration is obtained because of the presence of multiple processing cores in GPU. A validation for the function was set using MATLAB. The same code was transformed into vectorized version using Parallel Computing toolbox of MATLAB. From the results, it is observed that the computational time required for the execution is more in CPU as compared to GPU. Thus the Speedup parameter obtained is around 2 to 3 fold for Airy function when evaluated on systems with NVIDIA Tesla K80 GPU. From the results it may be concluded that Speedup parameter depends on the amount of data and number of iterations of an operation.

References

[1] V. Kiryakova, "The special functions of fractional calculus as generalized fractional calculus operators of some basic functions," Computers and Mathematics with Applications, vol. 59, no. 3, pp. 1128-1141, Feb. 2010

[2] Patil Parag,Singhaniya Navin,Jage Chaitanya,Vyawahare Vishwesh,Patil Mukesh and Paluri Nataraj. (2018). GPU Computing of Special Mathematical Functions used in Fractional Calculus. 10.2174/9781681085999118010011.

[3] Olivier Vallee and Manuel Soares - Airy function and applications in physics

[4] J. Liu, C. Gong, W. Bao, G. Tang, and Y. Jiang, "Solving the Caputo fractional reactiondiffusion equation on GPU", Discrete Dynamics in Nature and Society, Hindawi Publishing Corporation, vol. 2014, 2014.

[5] N. E. Banks, "Insights from the parallel implementation of efficient algorithms for the fractional calculus", University of Chester, United Kingdom, 2015

[6] Nvidia, CUDA C Programming Guide , Ver. 7.5, Sep. 2015. [Online]. Available: https://developer.nvidia.com/hpc

[7] "MATLAB Documentation" https://in.mathworks.com/help/matlab/

[8] Available:https://in.mathworks.com/help/parallel-computing/index.html'stid=CRUXlftnav

[9] "Predicting and Measuring Parallel Performance", February 1, 2012. [Online]. Available:https://software.intel.com

[10] "Parallel Computing", Wolfram Mathworld. [Online]. Available: http://mathworld.wolfram.com/ParallelComputing.html

[11] "Best Practices for MATLAB GPU Coding", Cornell University Center for Advanced Computing. [Online]. Available: https://www.cac.cornell.edu/matlab

[12] Ç. Uluişik and L. Sevgi, "A Tutorial on Bessel Functions and Numerical Evaluation of Bessel Integrals," in IEEE Antennas and Propagation Magazine, vol. 51, no. 6, pp. 222-233, Dec. 2009.

[13] Amparo Gil , Javier Segura , Nico M.Temme,"Numerical methods for special functions-Society for industrial and Applied Mathematics".

[14] B.K.Agarwal, Hari Prakash 2016,Quantum Mechanics,PHI,Delhi.

[15] A. K. Ghatak, R. L. Gallawa and I. C. Goyal,"Accurate solutions to Schrodinger's equation using modified Airy function," in IEEE Journal of Quantum Electronics, vol. 28, no. 2, pp. 400-403, Feb. 1992.

[16] B. Zhang, S. Xu, F. Zhang, Y. Bi and L. Huang ,"Accelerating MATLAB code using GPU:A review of tools and strategies". Artificial Intelligence, Management Science and ElectronicCommerce(AIMSEC):Dengleng, China, Aug. 2011.

[17] Fractional-order modeling of neutron transport in a nuclear reactor , by Vishwesh A.Vyawahare, P.S.V.Nataraj.

[18] G. Chen, G. Li, S. Pei and B. Wu, "High Performance Computing via a GPU" 2009 First International Conference on Information Science and Engineering, Nanjing, 2009, pp. 238-241.

[19] P. Sebah and X. Gourdon,"Introduction to the gamma function", American Journal of Scientific Research, Feb. 2002.

[20] D. W. Lozier, "NIST digital library of mathematical functions", Annals of Mathematics and Artificial Intelligence, Springer, May 2003.[Online]. Available:https://link.springer.com/article/

[21] E. Lindholm, J. Nickolls, S. Oberman and J. Montrym, "NVIDIA Tesla: A Unified Graphics and Computing Architecture," in IEEE Micro, vol. 28, no. 2, pp. 39-55, March-April 2008.

[22] Abramowitz, Milton; Stegun, Irene Ann, eds. (1983) [June 1964]. "Chapter 10". Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables. Applied Mathematics Series. 55 (Ninth reprint with additional corrections of tenth original printing with corrections (December 1972); first ed.). Washington D.C.; New York: United States Department of Commerce, National Bureau of Standards; Dover Publications. p. 446. ISBN 978-0-486-61272-0. LCCN 64-60036. MR 0167642. LCCN 65-12253.

[23] Digital Library of Mathematical Functions, *National Institute of Standards and Technology*, available at http://dlmf.nist.gov/.

[24] Donald A. Neamen,"Semiconductor Physics and Devices",chapter 2 - Introduction to Quantum theory of solids .

[25] K. Parand, M. Nikarya,Application of Bessel functions for solving differential and integro-differential equations of the fractional order,Applied Mathematical Modelling,Volume 38, Issues 15–16,2014,Pages 4137-4147,ISSN 0307-904X,https://doi.org/10.1016/j.apm.2014.02.001.

[26] Katugampola, Udita N. (15 October 2014). "A New Approach To Generalized Fractional Derivatives" (PDF). Bulletin of Mathematical Analysis and Applications. 6 (4): 1–15. arXiv:1106.0965. Bibcode:2011arXiv1106.0965K.

[27] Niels Henrik Abel (1823). "Oplösning af et par opgaver ved hjelp af bestemte integraler (Solution de quelques problèmes à l'aide d'intégrales définies, Solution of a couple of problems by means of definite integrals)" (PDF). Magazin for Naturvidenskaberne. Kristiania (Oslo): 55–68.

[28] Tenreiro Machado, José Kiryakova, Virginia Mainardi, Francesco. (2011). Recent history of fractional calculus. Communications in Nonlinear Science and Numerical Simulation - COMMUN NONLINEAR SCI NUMER SI. 16. 1140-1153. 10.1016/j.cnsns.2010.05.027.

[29] Kreyszig, Erwin. Advanced Engineering Mathematics. New York :Wiley, 1983.

[30] NVIDIA. (2015). TESLA K80 GPU ACCELERATOR - Board specifications [PDF file]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/Tesla-K80-BoardSpec-07317-001-v05.pdf