IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 18, Issue 6, Ver. VI (Nov.-Dec. 2016), PP 67-73
www.iosrjournals.org

Parallel Algorithm for the Simulation of Geomagnetic Fields Variations along the Equatorial Regions

[1] Tokula Umaha, [2]Esiefarienrhe Bukohwo Michael, [3]Ale Felix
[1,2]Math/Statistic/Computer Science Dept. University of Agriculture, Makurdi, Nigeria
[3]National Space Research and Development Agency, Abuja, Nigeria.

Abstract: *This work presents the development of a parallel algorithm for the computation of geomagnetic field data for stations 30 degrees North and South of the equator. The parallel computing hardware platform used involved multi-core processing unit. MATLAB 2012a's parallel computing toolbox was used to simulate solar quiet daily (Sq) variations on four(4) Intel Xeon E5410 processors, with clock speed 2.3 GHz and 16378MB RAM. A large geomagnetic dataset obtained from International Real-time Magnetic Observatory Network (INTERMAGNET), corresponding to 25 stations for year 2000 was used. The performances of both algorithm were compared using some indices such as execution time, efficiency, speed and overhead. The result from the parallel algorithm was two times faster than the corresponding sequential algorithm under same platform and workload in all of the indices.*

Keywords: *Parallel Computing, Multicore Architecture, Geomagnetic Field, Parallel Algorithm.*

I. Introduction

It is now clear that silicon based processor chips are reaching their physical limits in processing speed, as they are constrained by the speed of electricity, light, and certain thermodynamic laws. To solve this problem, multiple processors working in coordination can be connected with each other to solve this problem. There is therefore the need for parallel computing. Ananth [2] defined parallel computing as an advanced technique of computation in which large and/or complex problems broken into smaller tasks are solved concurrently. In order words, it is the simultaneous use of multiple compute resources to solve a computational problem. It has been an area of active research interest and application for decades, mainly the focus of high performance computing, but is now emerging as the prevalent computing paradigm due to the semiconductor industry's shift to multi-core processors.

The interest in parallel computing dates back to the late 1950's, with advancements surfacing in the form of supercomputers throughout the 60's and 70's. These were shared memory multiprocessors, with multiple processors working side-by-side on shared data. The supercomputer built by Caltech Concurrent Computation project for scientific applications in the mid 1980's shifted the advancements in parallel computing up a gear. This system showed that extreme performance gain could be achieved with off-the-shelf microprocessors. These massively parallel processors (MPPs) came to dominate the top end of computing, with the ASCI Red supercomputer computer in 1997 breaking the barrier of one trillion floating point operations per second. Since then, MPPs have continued to grow in size and power.

In the late 80's, clusters came into being and replaced MPPs. A cluster consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. Today, clusters are the workhorse of scientific computing and are the dominant architecture in the data centres that power the modern information age.

Today, parallel computing is becoming mainstream based on multi-core processors. Most desktop and laptop systems are now dual-core with quad-core processors readily available. Chip manufacturers have begun to increase overall processing performance by adding additional CPU cores. The reason is that increasing performance through parallel processing can be far more energy-efficient than increasing microprocessor clock frequencies.

Over the years, there has been an urgent need to compute and analyse an ever increasing volume of data and this has given rise to parallel computation. Gene [10] along with Gschwind [12] noted that advancement in multicore technology has paved way for development of efficient parallel algorithms and applications for complex and large numerical problems which are often found in science, engineering and other disciplines. Parallel programs, however, are harder to write due mainly to multiple concurrent tasks which must be synchronized.

George [11] noted that a well-designed (or configured) parallel computer, parallel algorithms and programming would ensure performance efficiency in advanced computations. Parallel computing should be scalable and robust in terms of the number of processors, fault tolerance, redundant supporting high-speed communication interfaces and inter-processor data sharing.

Eric [7] noted that the resources of parallel computing would not be used effectively unless there were available parallel algorithms, languages, and methods to evaluate the performance. Parallel programming methods and libraries are also very important in helping parallel program development by shielding users from low-level machine characteristics.

This paper is aimed at the development of an efficient parallel algorithm for the simulation of geomagnetic field variations using one-minute resolution of geomagnetic field data of locations along the equator. It has been established by Felix [8] that the activities of the sun directly impact on the variations of the geomagnetic field. The equatorial region of the earth is of interest because of the similar weather and climatic conditions experienced in this region. James [15] showed that it takes 2-3 days for solar radiations to reach the earth while Rennan [20] established that it takes 9 minutes for solar flares to reach the earth.

The paper is structured as follows. In the next section, we describe briefly the techniques employed in this work. Section 3 shows the parallel algorithm developed. Section 4 shows the evaluation of the parallel algorithm. In section 5, we conclude the paper.

II. Performance metrics for parallel systems

Amdahl [1] and Gustafson [13] propounded laws that defined theories and suitable metrics for benchmarking parallel algorithms performance. Amdahl's law [1] specifically uncovers the limits of parallel computing while Gustafson's law [13] sought to redress limitation encountered on fixed workload computation. Amdahl's [1] law states that the speedup achieved through parallelization of a program is limited by the percentage of its workload that is inherently serial. We can get no more than a maximum speedup equal to

$$1/(s + p\ /\ N) \qquad (1)$$

Where
s = the length of the serial part of the problem
p = the length of the parallel part of the problem
N = number of processors used

Gustafson's [13] law states that, with increasing data size, the speedup obtained through parallelization increases, because the parallel work increases (scales) with data size.

Both laws are in fact different perspective over the same truth – one sees data size as fixed and the other sees the relation as a function of data size. In this work, however, the multicore specifications used performed well under Amdahl's [1] assumptions. The following metrics were used in evaluating the performance of the parallel algorithm: efficiency, speedup, execution time and cost.

The speedup, S_p, determines how many times the parallel program is faster than the fastest sequential program using identical problem and hardware specifications. This could be measured in any increment of time. It is the time it takes a program to execute in serial (with one processor) divided by the time it takes to execute in parallel (with many processors). The formula for speedup is:

$$S_p = \frac{T_s}{T_p} \qquad (2)$$

Where $\boldsymbol{T_s}$ and $\boldsymbol{T_p}$ are the execution time for fastest sequential and parallel programs respectively.

Speed of sequential algorithm subjectively depends largely on the methodology of array or matrices manipulation and programming structure. Thus, serial runtime, $\boldsymbol{T_s}$, must be developed to be the fastest algorithm for a particular problem, on this premise Amdahl's law could be proven right and validated. Efficiency is a measure of parallel performance that is closely related to speed up and is often also presented in a description of the performance of a parallel program. Efficiency (λ) of a parallel system is given as:

$$\lambda = \frac{\frac{S_p}{p}}{p} * 100\% \qquad (3)$$

Where $\boldsymbol{S_p}$ = speedup
P = number of processors

The core elements of parallel processing are CPUs. Based on the number of instructions and data streams that can be processed simultaneously [9]. The SIMD architecture was used for this research.
Rabiu [19], Osborne [18], Onwumechili [17], Okeke [16], Chapman [5], Chapman and Rajarao [6], Campbell and Shiffmacher [3], Campbell and Honore [4] have all done extensive work on the analysis of the geomagnetic field but most of these works were based on the hourly dataset. Not much has been done using the minute dataset. This is because the dataset is very large and would require more computing power to achieve results in good time.

Table 3.1 Work breakdown structure for year 2000

S/N	Problem	Size(Matrix)	Case(n)	Task
1.	25 stations with 3 geomagnetic field components (H, D, Z)	25 x 3	75	Data post-acquisition processing
2.	Year 2000=366 days 1 Day = 24hrs * 60 = 1440 minutes	$(3 * 25 * 366, 1440)$ $\{27,450 \times 1440\}$	39,528,000	Baseline and non-cyclic correction

III. Methodology

The geomagnetic data for this research consists of minute values obtained from International Real-Time Magnetic Observatory Network (INTERMAGNET) for year 2000. INTERMAGNET is a global network of observatories, monitoring the Earth's magnetic field. INTERMAGNET has stations round the world but only data from stations on or very close to the equator were used for this work. Table 3.1 shows the work break down structure of the dataset for year 2000

For uniformity, all the data was converted to the HDZ orientation using the relationship established by Vestine [21]. Sequential and parallel MATLAB m-file programs were developed to solve for Sq mean values on monthly representations. The parallel algorithm was developed from the serial algorithm. The parallel algorithm was designed to run on 4 processors. Data dependencies and recursive code segments were resolved to ensure parallelism. Critical variables such as local time (LT) that run 1440 iterations repeatedly were sliced and indexed logically so as to enable parallelization. Load balancing was achieved as follows

matlabpool open config p
spmd {
A = zeros(n, m);
B = codistributed(S);
#with the use of 4 processors (p}
a=m/p; z=O; z1=1;
For p=1:4
z = z+a; computeNode p: This node stores B(,z1 :z);
z1=z+ 1;
end;
}
matlabpool close

The sequential algorithm is shown in figure 3.1 while the parallel algorithm is shown in figure 3.2

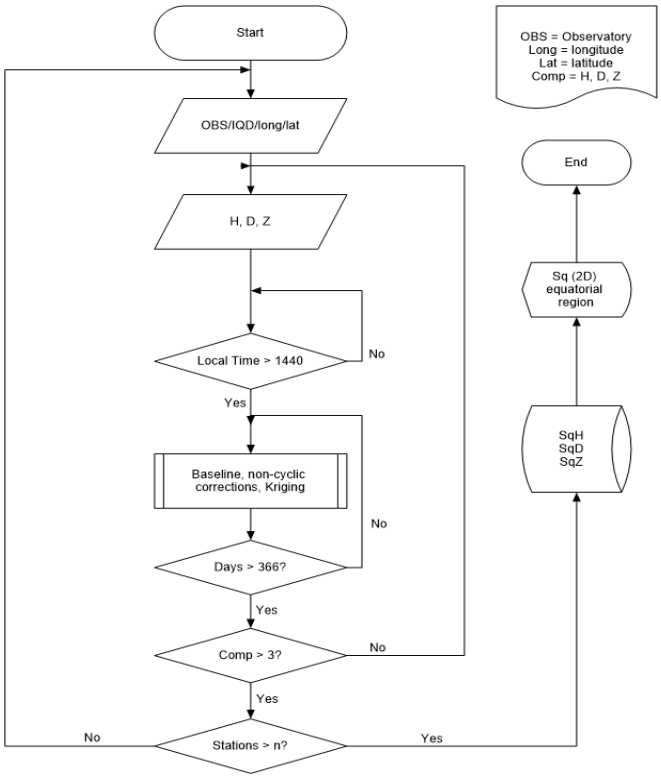

Figure 3.1: Sequential algorithm for serial computation

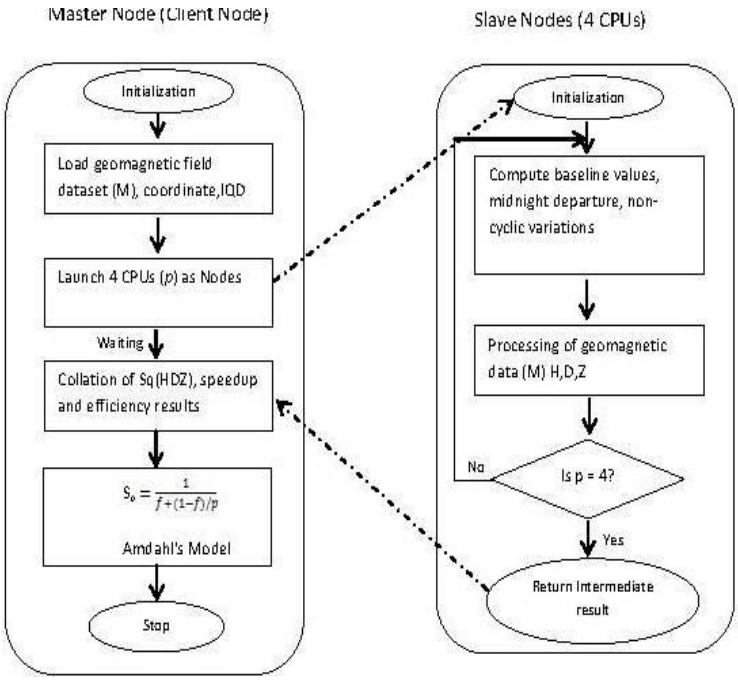

Figure 3.2 Sq Parallel Algorithm

IV. Algorithm Evaluation

Table 4.1 shows the details of the execution time of the developed algorithm.

While the developed sequential program took 19.3 minutes to run on one processor, the parallel program, executed on one processor on a parallel platform took 14.5 minutes to run. Thus, Ts was not equal to T1. Figures 4.1, 4.2, 4.3 and 4.4 show graphically, the execution time, speedup time and efficiency with respect to the number of processors.

Table 4.1 Performance metrics for the parallel algorithm

No of CPU (p)	Execution time (s) (Tp)	Execution time (min) (Tp)	Computation time (s)	Overhead time (s) (t")	Speedup (Sp)	Efficiency (φ)
1	873.8960	14.56493	865.9430	7.953	1	100
2	525.4830	8.75805	517.1010	8.382	1.6630	83.15
3	403.2020	6.72003	393.1090	10.093	2.1673	72.243
4	352.1520	5.8692	339.0510	13.101	2.4815	62.0375

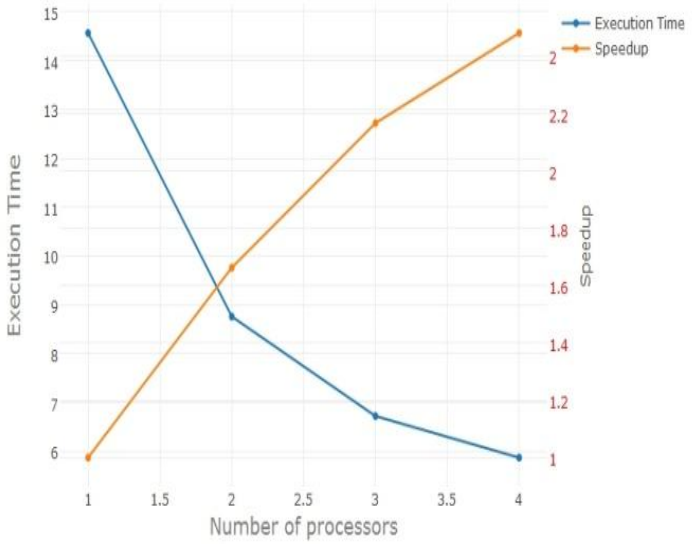

Fig 4.1: Execution time vs Speedup

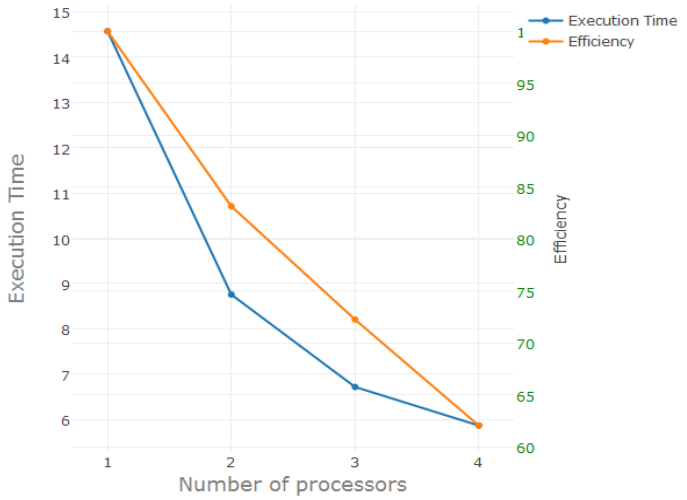

Fig 4.2: Execution Time vs Efficiency

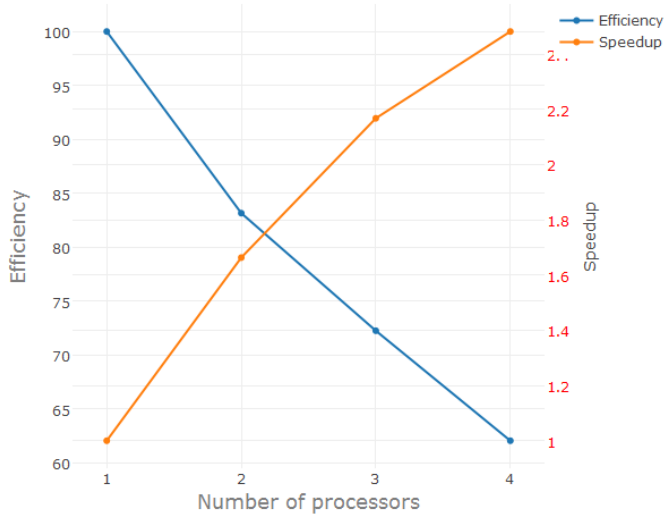

Fig 4.3: Efficiency time vs Speedup

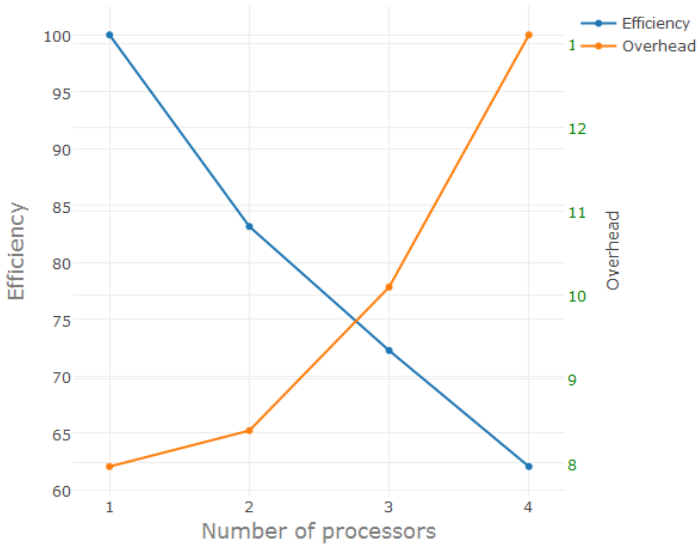

Fig 4.4: Efficiency vs Overhead Time

The performance results showed that the execution time on one processor took 14.5 minutes while the corresponding sequential program took 19.3minutes to run. Figure 4.1 shows the effect of increase in number of processors on the speedup and execution. As the number of processors increase the execution time decreases while the speedup increases. The speedup of a parallel code is how much faster it runs in parallel. Figure 4.1 for example, shows that as the number of processors was increased from 1 to 2, the program ran faster by a factor of 1.6630. Furthermore, Figure 4.2 shows that as the number of processors increased, execution time decreased but efficiency also decreased. Efficiency, which is a measure of how much of available processing power is being used decreased because more time is spent communicating between processors as their number increases.

Figure 4.3 shows the relationship between efficiency and speedup as the number of processors increased. It was seen that as number of processors increased, speedup increased but efficiency decreases. Again, speedup increases because there are more available compute resources but more time spent in inter-processor communication, leading to decrease in efficiency.

Finally, figure 4.4 represents the relationship between overhead and efficiency. As overhead time increases, efficiency decreases.

V. Conclusion

Parallelization improves the performance of programs for solving large and/or complex scientific and engineering design problems. Metrics used to describe the performance include execution time, speedup, efficiency, and cost. These results complement existing studies and demonstrate that parallel computing and multicore platforms provide better performance of space weather parameter, Sq with the use of higher time resolution magnetic data.

In this work, we analyzed performance of parallel computation of Sq using data-intensive one-minute time series magnetic data from 25 observatories close to the equator for year 2000. The algorithm and model developed showed that speedup increased at the expense of efficiency.

References

[1]. Amdahl, G.M., "Validity of the Single-Processor Approach to Achieving Large-Scale
[2]. Ananth, G., Anshul, G., George, K., & Vipin, K. (2005). Introduction to parallel Computing,. Boston: Addison Wesley.
[3]. Campbell,W.H. & Shiffmacher, E.R. (1987), Quiet ionospheric currents and earth conductivity profile computed from quite-time geomagnetic field changes in the region of Australia. Aust. J. Phys. 40, 73-87.
[4]. Campbell,W. & Honore, (1997), Introduction to Geomagnetic Fields, Cambridge University Press.
[5]. Chapman, S., The equatorial electrojet as detected from the abnormal electric current distribution above Huancayo, Peru, and elsewhere, Arch.Meteorol. Geophys. Bioclimatol. A, 4, 368–390, 1951.
[6]. Chapman, S. and K. O. Rajarao, The H and Z variation along and near equatorial electrojet in India, Africa and the Pacific, J. Atmos. Terr. Phys., 27, 559–581, 1965.
[7]. Eric Olmedo, Jorge de la Calleja, Antonio Benitez, and Ma. Auxilio Medina. Point to point processing of digital images using parallel computing. IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 3, No 3, May 2012
[8]. Felix, A., Ibidapo-Obe, O., Fashanu, T. A., & Agboola, O. A. (2012). Computation of Variations of Geomagnetic Field Using One-Minute Time Dataset. International Journal of Engineering and Technology, Volume 2 No. 11
[9]. Flynn, M. J. (1966). Very High-Speed Computing Systems. (pp. VOL. 54, NO. 12). PROCEEDINGS OF THE IEEE.
[10]. Gene H. Golub. and James M. Ortega., (1993). Scientific Computing: An Introduction with Parallel Computing, Academic Press, Inc.
[11]. George S. Almasi and Allan Gottlieb, 1994, Highly Parallel Computing, Second Edition Benjamin/Cummings Publishers
[12]. Gschwind M., (2006), Chip Multiprocessing and the Cell Broadband Engine, ACM Computing Frontiers.
[13]. Gustafson, J.L., "Reevaluating Amdahl's Law," Comm. ACM, May 1988, pp. 532-533.
[14]. Honore, M. E., Comelo, K. D., & Cesar, M. B. (2014). Day-to-Day Variability of H Component of Geomagnetic Field in Central African Sector Provided by Yaoundé-Cameroon Amber. International Journal of Geosciences, 5, 1190 1205.
[15]. James A. Marusek, 2007, Solar Storm Threat Analysis, Impact, Bloomfield, Indiana 47424
[16]. Okeke, F. N., C. A. Onwumechili, and B. A. Rabiu, Day-to-day variability of geomagnetic hourly amplitudes at low latitudes,Geophys.J.Int.,134,484–500, 1998.
[17]. Onwumechili, C. A., Spatial and temporal distributions of ionospheric currents in sub-solar elevations, J. Atmos. Terr. Phys., 59, 1891–1899, 1997.
[18]. Osborne, D. (1964). Daily and seasonal changes of equatorial electrojet in Peru. J. atmos. terr. phys , 26, 1097-1105.
[19]. Rabiu A.B., (2001), Seasonal Variability of Solar Quiet at Middle Latitudes, Ghana Journal of Science, 41, 15 22.
[20]. Rennan Pekunlu (1999), Solar Flares, Tr. J. of Physics International Workshop: 23 (1999), 415-423
[21]. Vestine, E. H., Description of the earth's main magnetic field and its secular change, 1905-1945, Carnegie Institute of Washington Publication 578. Washington. D. C., 1947.

Umaha Isaac Tokula completed his primary school education with Benue Cement Company (BCC) Staff School Gboko, Benue in the year 1996 where he obtained his First school leaving Certificate (FSLC). He then proceeded to Federal Government College, Otobi, Benue State where he further obtained his Senior School Certificate Examination SSCE in the year 2002. A graduate of Computer Science (B.sc) from Ambrose Ali University Ekpoma in 2008, Umaha Tokula is currently undergoing his Masters Degree (M.sc) program with the Federal University of Agriculture, Makurdi, Benue State where he is studying Computer Science.

Esiefarienrhe Bukohwo Michael holds a PhD in Computer Science from Modibbo Adama University of Technology, Yola, Nigeria (2012), MSc in Computer Science from Abubakar Tafawa Balewa University, Bauchi (2004) and BSc in Computer Science from University of Benin (1996). He is an Oracle Certified Professional and a member of several professional bodies including ACM, NCS, CPN. His area of research interest spans Software Engineering (Risk Analysis), Networks, Databases and Data Mining.

Ale Felix graduated with a Doctor of Philosophy (PhD) in Systems Engineering in 2014 from the University of Lagos. He has a BSc (1998) and MSc (2005) degrees in Computer Engineering and Computer Science respectively from the Obafemi Awolowo University, Ile-Ife. Dr. Ale Felix is a Chief Engineer with the National Space Research and Development Agency. He undertakes joint research between the academia and the industry in area of High-Performance computing applications and support systems, Distributed and parallel computing. Dr. Ale Felix is also a part-time lecturer in the Department of Electrical/Electronics Engineering, University of Abuja, Nigeria. He specializes in Software, OBDH satellite subsystem and Systems engineering with broad interest in embedded systems and software design and development. He is a registered engineer with the Council for Regulation of Engineering in Nigeria (COREN) and also a member of the Nigeria Society of Engineers (NSE).