Optimization of CYGNUS Readout Emulation Program

Azhar Ikhtiarudin[1], Dr. Lindsey Bignell[2]

[1] Department of Engineering Physics, Faculty of Industrial Technology, Institut Teknologi Bandung, Bandung, West Java 40132, Indonesia.
[2] Departmen of Nucelar Physics, Research School of Physics, Australian National University
azharikhtiarudin@gmail.com lindsey.bignell@anu.edu.au

Abstract. Over the past ten years, significant advancements have been made in particle detector technology, greatly improving our ability to detect dark matter—a mysterious substance that constitutes a substantial portion of the universe's mass. However, a major challenge in this pursuit is the presence of what scientists refer to as the "Neutrino Fog", which represents a significant background signal caused by neutrinos. The CYGNUS-Oz collaboration project is currently conducting studies on directional detector technology with the goal of constructing a large detector capable of inferring directional particle information and searching for dark matter amidst the neutrino fog. This paper presents improvements to the computer program used in the CYGNUS Readout Emulation Project, with the goal of boosting its speed, memory efficiency, and adaptability for various levels of simulation accuracy. These enhancements are important as we continue our mission to effectively utilize the simulation program.

1. Introduction

In the ever-evolving field of particle physics, advancements in detector technology have provided opportunities to explore the mysterious concept of dark matter, a substance believed to make up a significant part of the universe's mass. However, a significant challenge in this endeavour is the presence of a persistent background signal known as the "Neutrino Fog," primarily coming from cosmic neutrinos. This background signal makes it difficult to distinguish real dark matter signals. Given that the sun is the primary source of low-energy neutrinos, while dark matter arrives from the direction of the Cygnus constellation, a detector capable of deducing the directional information of incoming particles can be employed to search for dark matter amidst the backdrop of the neutrino fog. Furthermore, even if dark matter signals are detected above the neutrino fog, directional detection remains a critical tool for verifying such signals as originating from dark matter. This approach also enables "dark matter astronomy," facilitating the exploration of the properties of the Milky Way's dark matter halo.

The CYGNUS-Oz collaboration project is currently conducting studies on directional detector technology with the goal of constructing a large detector capable of inferring directional particle information and searching for dark matter amidst the neutrino fog. The development of the prototype (CYGNUS-1) was led by a research group from the Australian National University. In this report, we explain the methodology used and present the results of our optimization efforts.

2. Methodology

2.1. Overview of the Prior Program

The fundamental objective of this tool is to replicate the readout process for simulated events within Time Projection Chambers (TPCs). Rather than being a rigid monolithic emulator, the tool is designed as a versatile toolkit, adaptable to various ways in which it can be applied. For example, the toolkit can be employed to model the complete sequence of processes, including the drift, amplification, and readout of Geant4 primary events.

2.2. Event Processing Stages

The primary unit of operation for this tool is an 'event,' which undergoes several distinct stages:

1. PrimaryEvt
   At this stage, the tool initially processes the energy deposits within the TPC. These energy deposits are represented as a Pandas dataframe and include essential information such as the deposit locations (x, y, z, Edep). While additional data such as vertex location and particle type may be included, they are not strictly required. In cases where such data is absent, the toolkit employs Poisson sampling to determine the number of initial charge carriers based on the W-value of the gas at each energy deposit location.

2. DriftedEvt
   After primary processing, the event advances to the drifted stage, typically when it reaches the avalanche region, assumed to be a plane along the x-y coordinates with a fixed z position. During this stage, the tool applies Gaussian smearing based on parameters such as transverse and longitudinal diffusion coefficients, carrier mobility, and drift field strength. Carriers are tracked individually at this point, and their status is recorded in a dataframe containing information like the primary event index, position (x, y), and time interval (dt). It's worth noting the use of dt rather than z for tracking.

3. ReadoutEvt
   As the event progresses further, particularly when avalanche gain is introduced, storing individual carriers becomes impractical due to the large multiplication factor, often reaching around a million-fold increase. Instead, for each drifted carrier, the tool samples the gain distribution, typically assuming it follows an exponential distribution if no other information is provided. Based on this gain distribution, a point spread function is scaled according to the number of generated electrons and subsequently added to a binned readout, simplifying the storage and processing of data at this stage.

3. Results and Discussion

3.1. Speed Improvement

To significantly improve the program's speed, a fundamental approach involves making use of Python's vectorization technique. Python vectorization, at its core, is a method for optimizing Python programs by leveraging the built-in vectorized operations provided by libraries (NumPy, Pandas, etc). When dealing with extensive datasets containing numerous elements, rather than individually processing each element using iterative loops, vectorization directly applies operations to the entire array of elements. This change in computational approach can lead to a substantial enhancement in the program's overall performance.

The essence of NumPy vectorization lies in its ability to efficiently transfer data from Python (which tends to be relatively slower) to NumPy and subsequently to C (known for its higher speed). The data is then returned to Python. This intricate process enables us to achieve a level of speed comparable to that of C programming, all without the need to write code in the C programming language itself.

The results of the speed improvement are shown in the comparison table below.

Events	Previous (µs)	New (µs)	Gain
Primary Events	6550000	5230000	1.252390057
Carrier Events	110	14.6	7.534246575
Drifted Events	5380	55.3	97.2875226
Gain & Readout Events	30700	21000	1.461904762

Figure 1. Speed comparison between the previous and improved programs.

The speed benchmark measurements from the above results were obtained using Jupyter Notebook's `%%timeit` magic command, which repeatedly runs a piece of code and calculates the average time it takes to execute. We can observe that the most significant improvement is in the Drifted Events, which are now 97 times faster than before.

3.2. Solving the Memory Usage Issues

In the previous method, the process began by creating a grid that covered all N carrier points in DriftedEvt. Following this, N 3D Multivariate normal distributions were generated to represent Nel (Number of electrons at each point). Each of these 3D multivariate distributions was centered at the coordinates of a carrier and had a size that encompassed all N points. These 3D multivariate distributions were then accumulated, representing the sum of all the numbers of electrons contributed by each carrier point. After that, a minimum threshold value for Nel was applied, and values lower than the threshold were set to zero.

For instance, in Figure 2 (on the right), there are two points, P1 and P2. This method produced a 3D matrix that encompassed P1 and P2 in three-dimensional space. One drawback of this approach is that it resulted in significant memory usage, especially when the distance between points was much larger than the grid size. If the distance between P1 and P2 was substantial compared to the grid size, it would lead to excessive memory consumption. However, in such cases, the grid information far away from the points was not actually needed because a threshold was applied, causing most of the values in this 3D matrix to be zero. This concept is also known as a "Sparse Matrix."

In contrast, the improved method capitalizes on the concept of a sparse matrix and selectively utilizes essential information, which, in this context, pertains to the vicinity of each point, such as P1 and P2. The improved method involves creating local grid coordinates for P1 and P2, with the same grid size as the global grid coordinates. Within each local coordinate system, a 3D multivariate normal distribution is applied to represent Nel (Number of Electrons in each grid). Then, a lower threshold for Nel is applied. This threshold needs to be set in such a way that we can ensure positions far enough from the central point are assumed to be zero. Subsequently, the information is transformed from each local coordinate system to the global coordinate system and merged. In this approach, information distant from the points is not retained, resulting in memory savings. The extent of memory savings is greater when the matrix contains more sparse elements (more zero points). You can visualize the two methods illustrated in Figure 1.

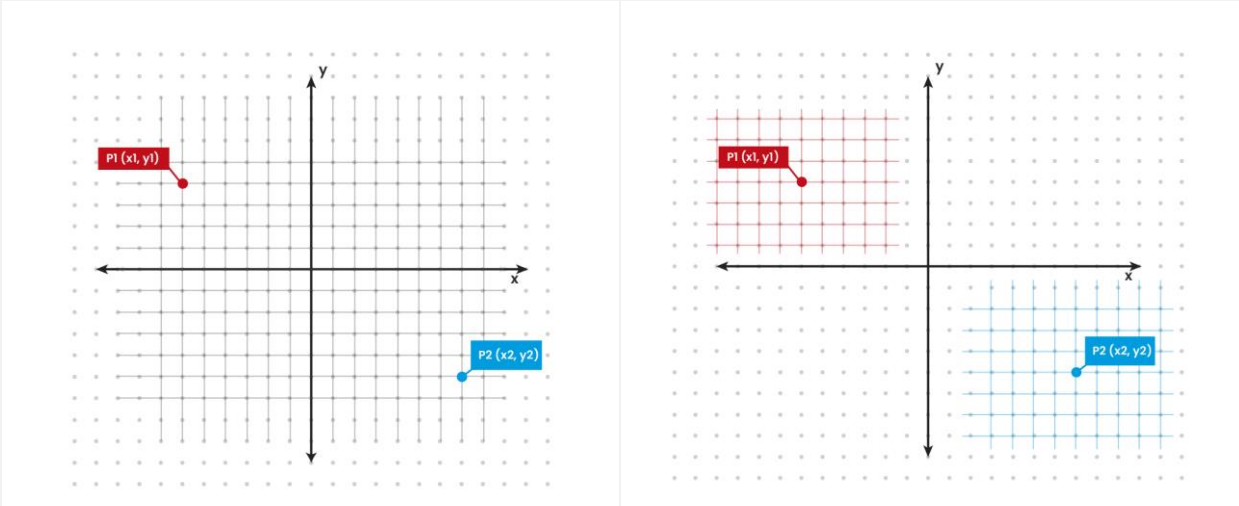

Figure 2. Coordinate system comparison between previous and improved methods

In the image above, the black grid in the centre represents the global or main coordinate, while the blue and red grids in the right image represent the local coordinates for each carrier charge (P1 and P2). Here is an example where we apply both methods to a sample set of points: [[1, 1, 1], [5, 5, 5], [10, 10, 10]]. Both methods work well in this case, and there are no memory issues because the coordinate spans are relatively small.

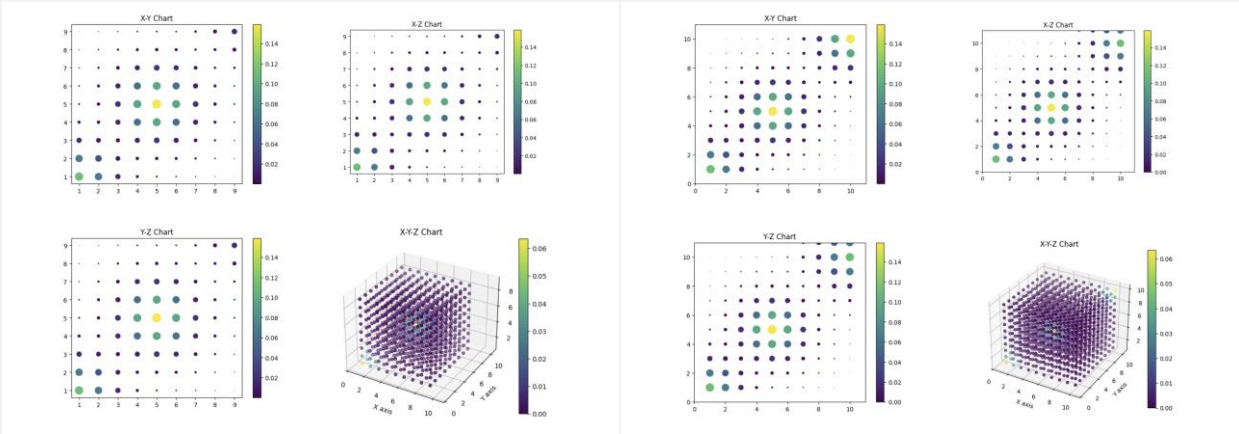

Figure 3. Results comparison between the previous and improved program using sample point 1

Moving on to the next sample point, which is [[1, 1, 1], [5000, 5000, 5000], [10000, 10000, 10000]]. This example illustrates cases where the distance between points is relatively large, while the grid size remains the same as in the previous example. In this example, the previous method encounters a memory error because the memory required increases significantly, necessitating the allocation of 21.8 TiB of memory.

```
MemoryError: Unable to allocate 21.8 TiB for an array with shape (3, 9999, 9999, 9999)
```

However, the improved method performs well because it avoids the need to create a massive 3D multivariate distribution and intelligently conserves computational resources.

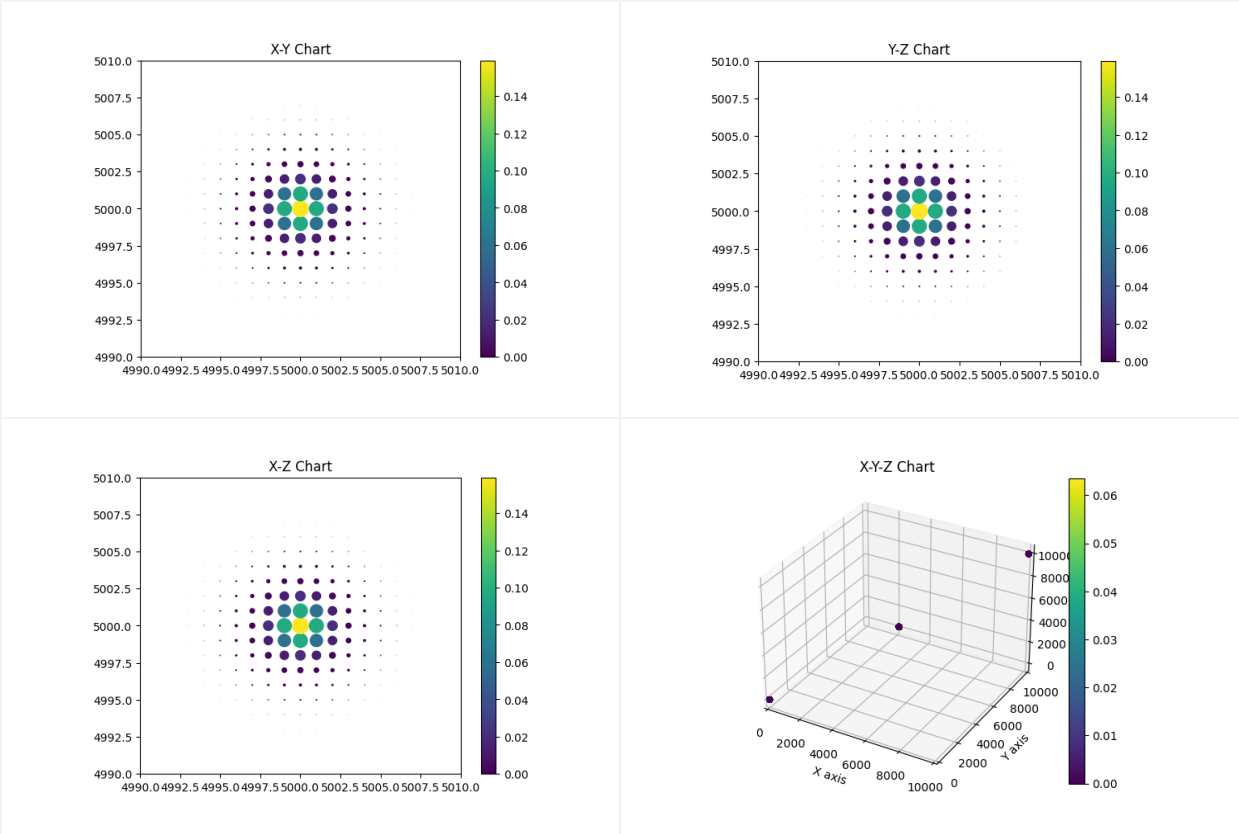

Figure 4. Results comparison between the previous and improved program using sample point 2

3.3. Alternative Methods for Advancement in Nel Calculation

The next optimization aims to improve the accuracy of the calculation of the number of electrons in a grid. The probability distribution function for multivariate normal is calculated as follows.

$$f(x) = \frac{1}{\sqrt{(2\pi)^k \det \Sigma}} \exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right),$$

where μ is the mean, Σ the covariance matrix, *k* the rank of Σ.

The previous method used the probability distribution function (PDF) from the center point and multiplied it by the area of the grid. The fundamental assumption in this method is that the probability distribution value in each point in the grid has the same value. However, this is not always the case

because if the grid size is large enough, then the value of the probability distribution function from each edge will have quite different values.

The improved method involves using the Cumulative Distribution Function (CDF). The cumulative distribution is defined as follows: it sums all the PDF values from negative infinity up to a specific point. In the 3D grid, it performs integration for each of the three coordinates, summing up from negative infinity in the x, y, and z axes up to that specific point (x, y, z).

To calculate the cumulative distribution function (CDF) within the grid, we can leverage the CDF values at each corner and carry out the necessary arithmetic operations. The two-dimensional simplification of this approach is visually depicted in the image below.

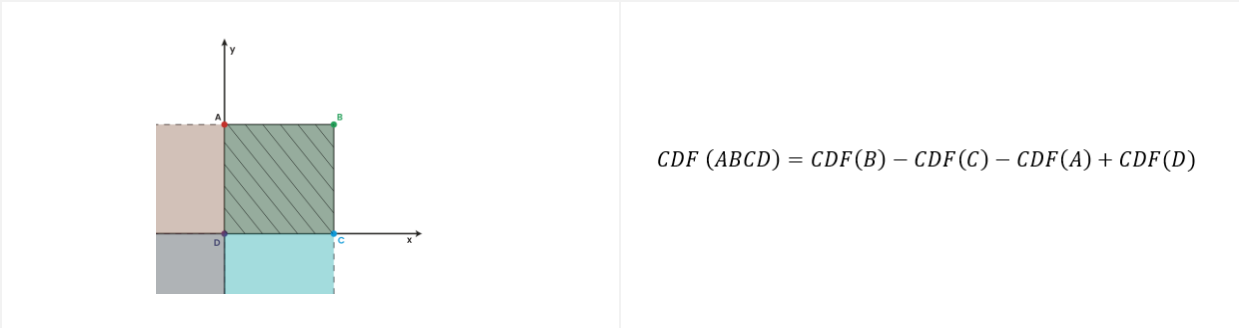

Figure 5. Geometric visualizations and equations used to calculate the Cumulative Distribution Functions

Utilizing the cumulative distribution function (CDF) offers improved accuracy in our calculations, eliminating the need for the assumption that the grid span is significantly smaller than the deviation of the probability density function (PDF) values. However, it's important to note that this method does come with a drawback: it requires additional computation time. This is because we must calculate the CDF for each corner in order to determine the number of electrons within a given range. While the accuracy gains are substantial, it's essential to weigh this against the increased computational demands when choosing the most suitable approach for the task at hand.

4. Conclusion

In summary, the project "Optimization of CYGNUS Readout Emulation Program" has delivered impressive results. The program's speed has been significantly enhanced, a fact clearly demonstrated in Figure 1. Memory-related issues have been effectively tackled through an innovative technique inspired by sparse matrices, as showcased in Figure 2, and this approach has been validated through various examples and benchmarks. Furthermore, the project has introduced alternative methods to improve Nel calculation, including the utilization of cumulative distribution functions rather than relying solely on probability density functions. These accomplishments underscore the project's unwavering commitment to making the CYGNUS Readout Emulation Program faster, more memory-efficient, and better equipped to excel in its intended applications.

5. Acknowledgements

We would like to convey our sincere appreciation to the individuals and organizations listed below for their invaluable contributions and unwavering support throughout the duration of this research. We are genuinely thankful for the exceptional guidance and constructive feedback generously provided by Dr. Lindsey Bignell, Prof. Hermawan K. Dipojono, MSEE., Ph.D., and the CYGNUS Dark Matter

research group at the Australian National University. Additionally, we extend our recognition to Jay Poria and the Team, who served as the committee for the Australian National University's Future Research Talent Awards Program. Their substantial support significantly facilitated the successful completion of this study. We also express our gratitude for the assistance provided by the Engineering Physics Department at Institut Teknologi Bandung. Finally, we want to convey our heartfelt thanks to our family and friends for their unwavering encouragement and support.

References

Bignell, L. (n.d.). *Available student project - Directional dark matter measurements with CYGNUS*. https://physics.anu.edu.au/study/projects/project.php?ProjectID=409

Garrett, K., & Duda, G. (2011). Dark matter: A primer. *Advances in Astronomy*, *2011*, 1-22.

Griffiths, D. (2020). *Introduction to elementary particles*. John Wiley & Sons.

Liu, J., Chen, X., & Ji, X. (2017). Current status of direct dark matter detection experiments. *Nature Physics*, *13*(3), 212-216.

Sauli, F. (2015). *Gaseous radiation detectors: fundamentals and applications* (p. 497). Cambridge University Press.

Sauli, F. (2016). The gas electron multiplier (GEM): Operating principles and applications. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *805*, 2-24.

Vahsen, S. E., O'Hare, C. A., & Loomba, D. (2021). Directional recoil detection. *Annual Review of Nuclear and Particle Science*, *71*, 189-224.