Can Numba and CUDA Python Development Kits be used on a GPU to Speed Up 2D Gamma Index Calculations used in Radiotherapy?

Sun, Edward[1]

[1]Torrey Pines High School, San Diego, California March 10, 2023

1 Introduction

1.1 Radiotherapy Background

Cancer is the second leading cause of death following heart disease. In 2020 alone, cancer alone was responsible for the deaths of 602,350 deaths in the United states (*An update on cancer deaths in the United States* [2]). However, with the aid of modern radiotherapy procedures, this figure can be significantly reduced. Starting with the invention of the X-ray by German scientist Conrad Roentgen, radiotherapy has made great strides in developing more and more precise ways of targeting cancer cells and sparing health tissue. Beginning with crude methods involving radium and low-voltage diagnostic X-ray machines, radiotherapy has evolved into a complicated procedure that involving various imaging modes and treatment delivery modes that ensure the highest accuracy. This procedure of radiotherapy is defined to include: patient consultation, simulation, treatment planning, and treatment delivery (*Radiation therapy process* [20]). In the patient consultation step, patients are initially diagnosed and informed of their condition. Following the consultation, the patients are taken to imaging facilities to capture images of their tumor(s) where contrast agents or chemicals are used to exaggerate the location of the tumor(s). Computer simulations are then used to define the location of the tumor and configurations of the treatment. Following this is the treatment planning step, where radiation oncologists, physicists, and dosimeters are involved to plan the radiotherapy for the patient. The goal of this step is to deliver high doses to the tumor while limiting the damage to the surrounding health tissues, thereby reducing the side effects of radiotherapy.

1.2 Quality Assurance

As modern radiotherapy is a complex process that involves many steps as mentioned above, quality assurance (QA) methods are needed to reduce errors and ensure the quality of results before moving on to delivering the doses to the patients. One of the most important QA steps is ensuring the doses of radiation being delivered to the patient is the same as the doses that was planned to be delivered by the treatment planning system. In planning the treatment based on a patient's body, a commercial treatment planning system is often used to calculate treatment plans or how much radiation to send to a

specific location (Gardner, Kim, and Chetty [8]). Due to the sensitivity of radiotherapy, these treatment planning systems routinely require the comparison of measured and calculated dose distributions (how the doses are delivered across a part of the patient's body). Given the tumor information of the patient, the treatment planning system will generate radiation delivery plans. Thus, the calculated doses are defined to be the distribution of how much radiation to give at certain points on the body calculated by the treatment planner system. The measrued doses are defined as the dose of radiation that is actually being delivered to the patient (measured by a radiation detector).

Figure 1: Sun Nuclear MapCHECK3 Radiation QA Matrix

However, to ensure the accuracy of dose delivery to the patient and minimize accidental radiation exposure, QA is essential to treatment planning. Techniques involved in this quality assurance steps often involve the comparison of dose distributions of a measured dose distribution, obtained from running the treatment plan on a radiation detection matrix (Figure 6) that is sensitive to radiation and will measure the specific doses delivered at specific coordinates (*MapCHECK® 3* [18]).

An example of the measured dose distribution versus the planned can be seen in Figure 2. On the top left is the calculated or planned dose distribution while the top right is the measured dose distribution using the MapCHECK radiation detection matrix. This measured dose distribution will be compared against the intended/planned distribution with various mathematical comparison methods to judge the accuracy of the treatment planning system and make adjustments accordingly to ensure precision of treatment (Low et al. [16]).

2 Literature Review

2.1 QA Methods

Over the past decades, many metrics have been developed to improve QA. These methods are mathematical formulas/algorithms that often searches through and compares the calculated and measured dose distributions (dose images) in 2D or 3D to give a value that shows the resemblance of the two images. This represents how accurate the treatment planner is since the closer the measured doses are to the intended doses, the more accurate the treatment planner is. For instance, Cheng *et.al*, used a pass-fail criteria for the dose

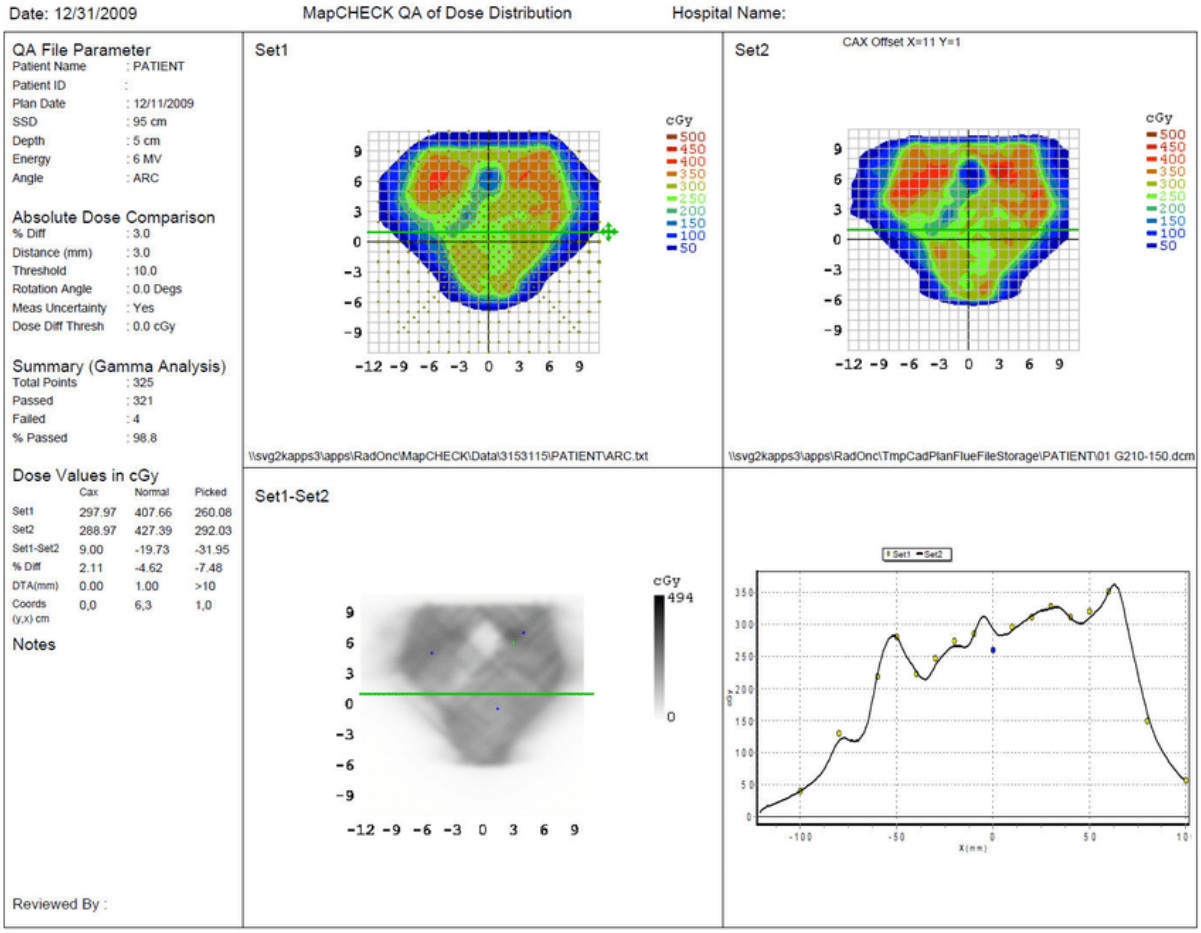

Figure 2: Example MapCHECK QA (on left) and planned dose distribution (on right)

difference (DD), which is the difference between the measured and calculated dose on a pixel of the dose distribution image, and distance to agreement (DTA), which is the distance/radius in which the algorithm searches on the dose distribution (Cheng et al. [6]). Each point within this search circle defined by the DTA is analyzed (Cheng). Similar metrics have been defined like the Global Gamma evaluation, maximum allowed dose difference (MADD) (Jiang), divide and conquer (D&C) (Stojadinovic et al. [23]), and more. However, one metric has been defined as the standard of QA metrics in the field of radiotherapy, that is the gamma index (Low et al. [16]).

2.2 γ index

The gamma index, originally proposed by Low et. al (Low et al. [16]), provides a representative and numerical metric that simultaneously incorporates the dose and distance criteria as mentioned above. It combines the ac curacies of previously proposed metrics into one γ index. This formula is mathematically defined by the following formulas:

$$\Gamma = \frac{|dose_{planned} - dose_{actual}|}{dose_{threshold}} \tag{1}$$

(2)

This original formula for the simple gamma index is robust and reliable. The calculation is also relatively simple and thus has been adopted and accepted widely. Since then, researchers have been making modifications to this algorithm by adding more metrics for greater accuracy or changing the formula for greater efficiency when running on computers' CPU or GPU (Gu, Jia, and Jiang [11]).

One popular modification is to add DTA as a factor into the calculation of the index. Essentially, this modified algorithm is a iterative process which passes over every pixel on a dose distribution image. For the sake of easy understanding, we will use a 2D example, but the same logic can be applied for 3D dose distributions as well. To begin the search, we will start on one reference pixel (calculated dose distribution) and test pixel (measured dose distribution), where dose distributions in this case are defined as 2D gray scale images. Then, the iteration begins with defining a reference position in real units based on a pixel on the reference image and then iterating through the circle on the test image. Test pixel within this search radius is then evaluated with the formulas above along with a percentage passing criteria, which means that this pixel passes the γ index test if it is within a certain percentage relative to the reference pixel. During this process, as the iteration uses real units (mm), it may generate a position that does not land on a pixel. Here, interpolation is used to estimate the value at that position within the search radius. This search radius for each reference pixel is actually the DTA and the percentage is the DTA criterion.

$$l = \sqrt{\frac{r^2(r_m, r)}{\Delta d_M^2} + \frac{\delta^2(r_m, r)}{\Delta D_M^2}} \tag{3}$$

$$r(r_m, r) = |r - r_m| \tag{4}$$

$$\delta(r_m, r) = D(r) - D_m(r_m) \tag{5}$$

$$\gamma(r_m) = min\{\Gamma(r_m, r_c)\forall(r_c)\} \tag{6}$$

Mathmatically, as seen in the equations above, Eq. (1) describes the intermediate value l calculated for each pixel, where $r^2(r_m, r)$ defines the square of the distance between the reference and the test pixel. The parameter r_m of $r^2(r_m, r)$ is the real position of the measured pixel and r defines the real position of the test pixel, as pointed out by Eq. (2). $\delta^2(r_m, r)$ defines the square of the difference in pixel value (in real terms, pixel value is radiation dose intensity) where r_m represents the pixel value of the measured pixel and r represents the pixel value of the test pixel, as pointed out by Eq. (3). $r^2(r_m, r)$ is then divded by Δd_M^2 which is the DTA or search radius. $\delta^2(r_m, r)$ is then divded by ΔD_M^2 which is the DTA criterion or acceptable passing percentage. This whole term is then square-rooted to produce the intermediate gamma value. This process is then repeated for all values within that radius for that refernce pixel. The final gamma index for that reference value is taken as the minimum value of all these previously calculated intermediate values. Then, this is repeated for all reference pixels from the reference image to produce one gamma value for each reference pixel, thereby creating a gamma image.

2.3 Challenges to the γ Index

With such an iterative and complex system involving intensive interpolation and calculations, Gu et. al (Gu, Jia, and Jiang [11]) highlighted the concerns of speed with the γ index calculation, where clinically relevant dose distribution sizes is very time-consuming to compare. Many researchers have attempted to improve the accuracy and efficiency of the algorithm (Bakai, Alber, and Nüsslin [3], Depuydt, Van Esch, and Huyskens [7], Stock, Kroupa, and Georg [22], Jiang et al. [13], Spezi and Lewis [21]) while others focused on speed (Ju et al. [14], Chen et al. [5], Wendling et al. [24]).

2.4 Code Acceleration in Python

Other than algorithmic approaches to speeding up the γ index, another approach is to use a graphics processing unit (GPU) to accelerate time consuming code. GPUs are a part of a modern computer that accelerates the rendering of computer graphics, especially in games, content creation, etc. (intel [12]). However, they have also been re-purposed for scientific computing, which takes advantage of their high calculation powers. As seen, GPUs have been used in weather prediction (Michalakes and Vachharajani [19]), mechanical structural analysis (Georgescu, Chow, and Okuda [9]), and chemical analysis (Ma, Wang, and Xie [17]). These studies typically use Nvidia CUDA acceleration toolkits in the C programming language or they are built from the ground up.

However, within recent years, Python has been a popular choice within the area of GPU acceleration. This is largely thanks to the growing development in acceleration libraries like Numba (Lam, Pitrou, and Seibert [15]). By itself, Numba is a Just-in-Time (JIT) compiler for Python, where code (in this case python code) is compiled into machine code during the execution of the python program rather than before the execution of the program. This compilation into machine code makes python code much faster as Python by itself, is interpreted during run time on a virtual machine and then executed. In recent years, Nvidia, the manufacturer of a majority of commercial graphics cards, have partnered with the developers of Numba to create GPU support for Numba, which now not only enables Numba to do JIT compilation, but also use GPUs to accelerate code.

Implementation of the gamma index in python isn't new. Many open source repositories on github have produced efficient gamma index calculation scripts[1] Similarly, Biggs et. al (Biggs et al. [4]) have proposed and maintained a community driven project named pymedphys where CPU driven gamma index codes and other medical physics related codes have been implemented. GPU acceleration with gamma index isn't new either. For instance, Gu et. al had proposed gamma index acceleration with a GPU in 2005 (Gu, Jia, and Jiang [11]). There have also been GPU acceleration toolkits released by MATLAB and researchers have implemented similar algorithms of the gamma index in MATLAB (*GPU acceleration* [10]). However, as seen, there is GPU acceleration in MATLAB, but not implemented with Numba and CUDA in python. Similarly, there have been gamma index implemented in python, however, there has not been GPU accelerated gamma index calculations.

Python is one of the most popular data analysis and processing tool in the software engineering world. With the popularity of this language, more tools should be made available for use in research. Thus, the reasoning behind the choice of this project is due to the high popularity of python, the large community support for GPU acceleration

[1] github.com/christopherpoole/pygamma

in python, and the prevalent use of the gamma index in radiotherapy. This study aims to fill this gap by creating a GPU accelerated gamma index framework for 1D, 2D, and 3D dose distributions in python via Numba and python CUDA. This will be achieved with a Nvidia GTX 1080 GPU ("@NVIDIAGeForce" [1]) running on PopOS 22.05 LTS with Nvidia CUDA support built in. For the algorithm, this study will be based off of Low et. al (Low et al. [16])'s original gamma index paper with no modifications made to the algorithm. This study aims to test the feasibility of implementing the gamma index with Numba and CUDA while also benchmarking this against other frameworks within other languages (MATLAB, C, etc) to analyze if Python Numba and CUDA accelerated gamma index will be competitive compared to these other languages' frameworks. As this study will be benchmarking and testing a framework with a limited set of testing data, it will be suitable for an experimental design methodology. In this next section, the specific methodologies will be more clearly defined.

3 Methods

3.1 Choice of Language

In traditional GPU-acceleration of the γ index (Gu, Jia, and Jiang [11]) programming languages like C/C++ or MATLAB are often used. C/C++ similarly has CUDA development support from NVidia, but the syntax is often more complicated than that used by Python. MATLAB has its own GPU Acceleration toolkit as well, but MATLAB is a expensive paid service which may make it unfeasible for some researchers who may not have the resources to purchase such a service. Thus, Python is a better language of choice for this task due to is easy syntax to implement, which may prove beneficial for researchers that want to understand or improve the source code of this project, and it is free and open-source for anyone to download and use. But the main downfall of Python as oppose to C/C++ or MATLAB is its slow speed due to Python being an interpreted language as explained earlier. Thus, it is important to also investigate whether or not the GPU acceleration through Numba and CUDA will be significant enough to overcome these performance differences based on their inherit language designs.

3.2 Program Design

As there are many modified algorithms out in the research community on how to calculate the γ index, I chose to write accelerated code for the version based on only the dose differences, dose threshold, and DTA (Eqn. 1 and 2) and the modified version of Low et al. [16]. (Eqn. 3 -6). I decided to select the first version of the calculation based off its simplicity and popularity and the second I selected based off of its relative accuracy as well as its popularity since many other researchers have used these formulas in their works (Gu, Jia, and Jiang [11]).

First, for the only dose difference based calculation that only uses matrix subtraction and not taking into account of DTA, the operation of finding the dose difference was achieved through the following simple matrix subtraction, where each matrix is the input 2D dose distribution. Then, the DTA criteria is implemented via the following formula:

$$A = doseplanned(nxnmatrix) \tag{7}$$
$$B = dosemeasured(nxnmatrix) \tag{8}$$
$$d = dosethreshold \tag{9}$$
$$\Gamma = |A - B| * 1/(dosethreshold) \tag{10}$$
$$\tag{11}$$

Then, this is combined with the regular gamma index formula through only expressing the minimum value between this calculated DTA and the gamma index.

Second, I implemented the modified Γ index that takes into consideration not only the dose planned, dose measured, and dose threshold, but also the DTA as well. This algorithm was implemented using equations 3-5. Translated into code, the code performs a search for every pixel. Using the pixel value from both the measured and calculated dose distribution, it finds the difference between them. Then, it divides the square of those difference by the square of the DTA of that pixel. Then the entire expression is taken the square root of, this becomes the intermediate value which we search within a radius

3.3 Bench Marking

For comparing the feasibility of this accelerated gamma index calculation, I compared multiple open-source projects all attempting to achieve the same purpose of calculating the γ index. Specifically, I used projects written in Python and MATLAB primarily to compare to my codes. For Python, I used pygamma by christopherpoole [1], gamma-index by janpipek [2], gamma-py also by janpipek [3], pymedphys [5] and also my own versions of non-accelerated loop-based γ index calculation (as detailed in the γ index section of this paper) and a matrix version the calculation. For MATLAB, I used mwgeurts' CalcGamma [5]. In total, there are 7 comparison codes that I ran and timed as shown in Table 1. These are used as a benchmark to later compare with the accelerated of the calculation.

4 Results

For the sake of consistency, I tried to pick repositories and algorithms that use similar algorithms (comparing modified to modified and matrix to matrix) and used the same 10 pixel by 10 pixel test image to benchmark the code to ensure fairness of comparison such that the differences in speed of running the codes will not be as of a result of using different algorithms or that of using smaller or bigger sized images. The test and reference input images are shown below. To ensure easy understanding, the images are made to

[1] github.com/christopherpoole/pygamma
[2] github.com/janpipek/gamma_index
[3] gist.github.com/janpipek/334c2533b87cd75c3f59
[5] https://github.com/pymedphys/pymedphys
[5] https://github.com/mwgeurts/gamma

be simple gray scale images with the reference and test having one pixel shifted over for easy understanding of how the gamma index worked.

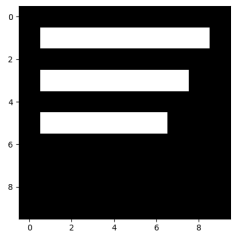

Figure 3: Measured (Reference) Dose Distribution

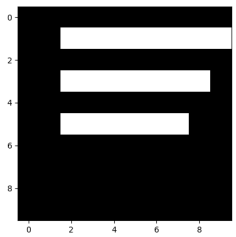

Figure 4: Planned Dose Distribution

4.1 Base Matrix Acceleration

With the two methods of accelerating, my first code that utilizes the simple γ index calculation using just matrix subtraction acceleration was compared against its base non-accelerated speed and repositories from pygamma that uses the same algorithm. Running the non-accelerated version of my code yielded a speed of around 0.0201921184 seconds for the 10x10 test image. The accelerated version of the matrix gamma index implemented in MATLAB GPU Accelerated with a single thread yielded results of around 0.00214124125112 seconds. Similarly to the MATLAB accelerated version with a single thread, my python accelerated version with a single thread yielded a time of around 0.0013515949249267578 seconds.

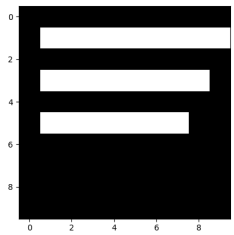

Figure 5: Gamma Index Map

Above is the resultant gamma index map that the codes yielded.

4.2 Modified Matrix Acceleration

For the second method of accelerating, my code that utilizes the modified γ index that implements the DTA inside of it. For this method, the gamma index was more complex, yielding a speed of around 1.012313124214 seconds for the same 10x10 matrix. However, when accelerated in MATLAB, the code too around 0.0121421412512 seconds. For the python Numba version, it took around 0.07121453213 seconds.

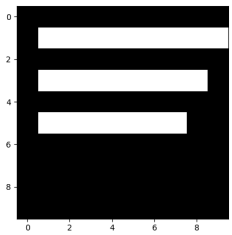

Figure 6: Gamma Index Map

(STILL MAKING THE GRAPHS WITH EXCEL)

5 Analysis

As seen in the data, it can be seen that the GPU accelerated gamma index is Numba and CUDA is comparable to that of GPU acceleration performed in MATLAB. Although some of the data for instance for that of the modified algorithm using the DTA method, there is a small discrepancy within the times taken from the MATLAB version and the Numba version, the differences are around the same order of magnitude. According to Nvidia's CUDA, the speed of GPU acceleration can differ based on the configuration of the GPU threads. Since MATLAB GPU acceleration toolkit is built in, this faster speed may be due to better customization of the GPU code. Since I have relatively less development compared to the MATLAB GPU acceleration toolkit, this discrepancy could be due to this. Furthermore, since the speed follows a non-linear pattern, the order of the magnitude in which the GPU acceleration speeds up to is more important to that of the specific value of the digit. This means that the fact the accelerations are in the same digit of value (e.g. in the thousandth's digit), it means that the two methods are comparable. This demonstrates that the Numba and CUDA method is comparable and feasible as an alternative to that of MATLAB methods.

6 Conclusion

In the modern day of GPU and high performance computing, the recent developments can be used for uses in radiotherapy. Starting with the paper from Gu et. al, it's been proven that the acceleration is feasible and very beneficial for applications in specific the γ index. Today, there are many developments and pathways that a radiotherapy researchers can take to speed up their γ index calculation. However, as many are proprietary requiring a high price subscription, some are hard to understand without prior and advance computer science knowledge, it is important to find a way that is easy for other researchers with no advanced computer science knowledge to use and yet still be comparable to other

methods like MATLAB. With advancements in Python's Numba and CUDA, it is an answer to this problem. Python is a high level language that is easy to understand and implement, however, the feasiblity of using this for radiotherapy has not be explored. This paper explores weather Numba and CUDA can be implemented for radiotherapy. Since the acceleration are in the same orders of magnitude, this paper has shown that it is feasible, even with different algorithms being tested.

However, although they are in the same orders of magnitudes, showing its possible feasiblity, it must be noted that I've only tested 2 algorithms, one with only matrices, and one of only the DTA modification. Thus, in the future, it may be better if similar algorithms and a more wide range of algorithms are tested to further test the feasiblity of using Numba and CUDA as an alternative.

References

[1] "@NVIDIAGeForce". *Nvidia GeForce 10 series graphics cards*. URL: https://www.nvidia.com/en-us/geforce/10-series/.

[2] *An update on cancer deaths in the United States*. 2022. URL: https://www.cdc.gov/cancer/dcpc/research/update-on-cancer-deaths/index.htm.

[3] Annemarie Bakai, Markus Alber, and Fridtjof Nüsslin. "A revision of the evaluation concept for the comparison of dose distributions". In: *Physics in Medicine and Biology* 48.21 (2003), pp. 3543–3553. DOI: 10.1088/0031-9155/48/21/006.

[4] Simon Biggs et al. "PyMedPhys: A community effort to develop an open, python based Standard Library for Medical Physics Applications". In: *Journal of Open Source Software* 7.78 (2022), p. 4555. DOI: 10.21105/joss.04555.

[5] Mingli Chen et al. "Efficient gamma index calculation using fast euclidean distance transform". In: *Physics in Medicine and Biology* 54.7 (2009), pp. 2037–2047. DOI: 10.1088/0031-9155/54/7/012.

[6] Abel Cheng et al. "Systematic verification of a three-dimensional electron beam dose calculation algorithm". In: *Medical Physics* 23.5 (1996), pp. 685–693. DOI: 10.1118/1.597714.

[7] Tom Depuydt, Ann Van Esch, and Dominique Pierre Huyskens. "A quantitative evaluation of IMRT dose distributions: Refinement and clinical assessment of the Gamma Evaluation". In: *Radiotherapy and Oncology* 62.3 (2002), pp. 309–319. DOI: 10.1016/s0167-8140(01)00497-2.

[8] Stephen J. Gardner, Joshua Kim, and Indrin J. Chetty. "Modern radiation therapy planning and delivery". In: *Hematology/Oncology Clinics of North America* 33.6 (2019), pp. 947–962. DOI: 10.1016/j.hoc.2019.08.005.

[9] Serban Georgescu, Peter Chow, and Hiroshi Okuda. "GPU acceleration for FEM-based structural analysis". In: *Archives of Computational Methods in Engineering* 20.2 (2013), pp. 111–121. DOI: 10.1007/s11831-013-9082-8.

[10] *GPU acceleration*. URL: https://www.mathworks.com/help/wavelet/gpu-acceleration.html.

[11] Xuejun Gu, Xun Jia, and Steve B Jiang. "GPU-based Fast Gamma Index Calculation". In: *Physics in Medicine and Biology* 56.5 (2011), pp. 1431–1441. DOI: 10.1088/0031-9155/56/5/014.

[12] intel. *What is a GPU? graphics processing units defined*. URL: https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html.

[13] Steve B Jiang et al. "On dose distribution comparison". In: *Physics in Medicine and Biology* 51.4 (2006), pp. 759–776. DOI: 10.1088/0031-9155/51/4/001.

[14] Tao Ju et al. "Geometric interpretation of the γ dose distribution comparison technique: Interpolation-free calculation". In: *Medical Physics* 35.3 (2008), pp. 879–887. DOI: 10.1118/1.2836952.

[15] Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. "Numba". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC LLVM 15* (2015). DOI: 10.1145/2833157.2833162.

[16] Daniel A. Low et al. “A technique for the quantitative evaluation of dose distributions”. In: *Medical Physics* 25.5 (1998), pp. 656–661. DOI: 10.1118/1.598248.

[17] Chao Ma, Lirong Wang, and Xiang-Qun Xie. “GPU accelerated chemical similarity calculation for compound library comparison”. In: *Journal of Chemical Information and Modeling* 51.7 (2011), pp. 1521–1527. DOI: 10.1021/ci1004948.

[18] *MapCHECK® 3*. URL: https://www.sunnuclear.com/products/mapcheck-3.

[19] John Michalakes and Manish Vachharajani. “GPU acceleration of numerical weather prediction”. In: *2008 IEEE International Symposium on Parallel and Distributed Processing* (2008). DOI: 10.1109/ipdps.2008.4536351.

[20] *Radiation therapy process*. URL: https://cancer.stonybrookmedicine.edu/RadiationTherapyProcess.

[21] Emiliano Spezi and D. Geraint Lewis. “Gamma histograms for Radiotherapy Plan Evaluation”. In: *Radiotherapy and Oncology* 79.2 (2006), pp. 224–230. DOI: 10.1016/j.radonc.2006.03.020.

[22] Markus Stock, Bernhard Kroupa, and Dietmar Georg. “Interpretation and evaluation of the γ index and the γ index angle for the verification of Imrt Hybrid Plans”. In: *Physics in Medicine and Biology* 50.3 (2005), pp. 399–411. DOI: 10.1088/0031-9155/50/3/001.

[23] Strahinja Stojadinovic et al. “Breaking bad imrt qa practice”. In: *Journal of Applied Clinical Medical Physics* 16.3 (2015), pp. 154–165. DOI: 10.1120/jacmp.v16i3.5242.

[24] Markus Wendling et al. “A fast algorithm for Gamma Evaluation in 3D”. In: *Medical Physics* 34.5 (2007), pp. 1647–1654. DOI: 10.1118/1.2721657.