

Mathematics Internal Assessment

Research Question: To what extent do the computational algorithms written in C++ programming language differ in time complexity when computing the peak wavelength of the blackbody radiation spectrum?

Table of Contents

Introduction and Rationale	1
Development of the Arithmetic Algorithm	3
Development of the Gradient Descent Algorithm	5
Assessment of the Algorithms' Time Complexities	8
Evaluation and Conclusion	11
Works Cited	12

Introduction and Rationale

A blackbody is an idealized object that absorbs all incoming electromagnetic radiation (Grauer). Blackbodies are used by astronomers to model the radiation emitted by stars and planets (Jones). Using the blackbody radiation spectrum, the temperatures and properties of these celestial objects can be estimated (European Space Agency). By employing Planck's radiation law which was developed in 1900, this spectrum can be described mathematically (Wyatt).

Planck's law demonstrates how the spectral power density $B(\lambda, T)$ (in Watts per square meter per steradian per meter/ $\text{Wm}^{-2}\text{sr}^{-1}\text{m}^{-1}$) depends on the wavelength λ (in meters/ m) for a blackbody at a constant temperature T (in Kelvin/ K):

$$B(\lambda, T) = \frac{2\pi hc^2}{\lambda^5} \cdot \frac{1}{e^{\frac{hc}{\lambda kT}} - 1}$$

Formula 1: Planck's law (Planck)

The physical constants in this equation are the following:

Letter	Name	Value	Unit of measurement
h	Planck's constant	$6.62607015 \cdot 10^{-34}$	Joule-seconds/ J·s
k	Boltzmann constant	$1.380649 \cdot 10^{-23}$	Joules per Kelvin/ JK^{-1}
c	Light speed	299729458	Meters per second/ ms^{-1}

Table 1: Physical constants in Planck's Law

The graph that illustrates the Planck's law is provided below:

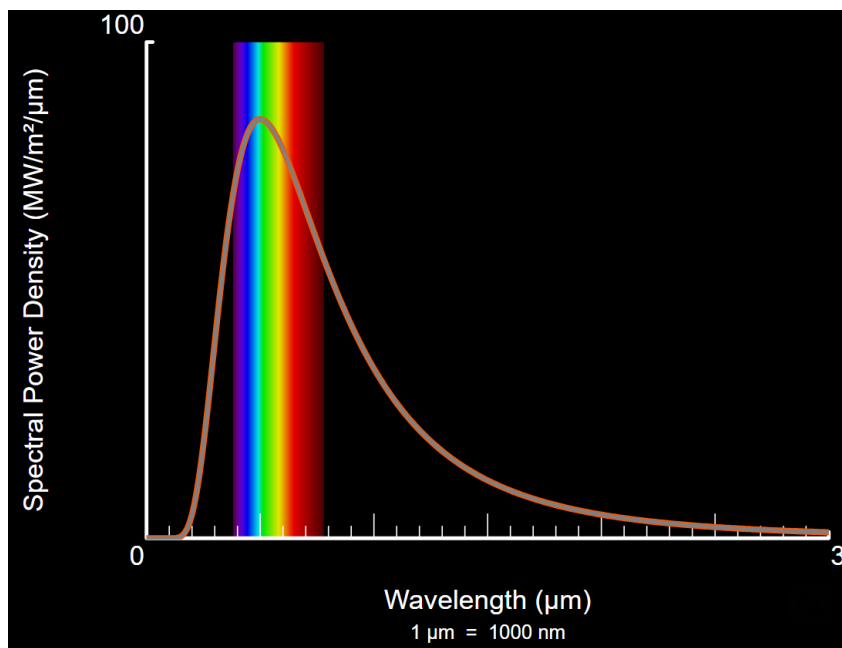


Figure 1: Planck's Radiation Law ("Blackbody Spectrum")

The global maximum of the spectral power density function occurs at the peak wavelength (λ_{max}). At this wavelength, the blackbody spectrum is most intense (PennState College). By computing it, astronomers can find the most likely wavelength at which radiation is emitted. This provides significant insights into the celestial body's temperature and properties, which plays a crucial role in characterizing and classifying objects in the universe (LibreTexts Physics).

There are many ways to compute the peak wavelength. However, astronomers cannot select a random algorithm to achieve their goal. They have to use the algorithm with lowest time complexity. The time complexity of an algorithm describes how the amount of time taken by it to run changes when the input size increases (Great Learning). The consideration of time complexity of an algorithm can be helpful for astronomers because it may give them an insight into each algorithm's time requirements when using them to handle large datasets.

Therefore, the purpose of this internal assessment is to compare algorithms that can be used to compute the peak wavelength based on their time complexity. This will enable astronomers to make a rational choice when selecting an algorithm for the peak wavelength computation. The algorithms will be written in C++ programming language which is very popular among scientists due to its high performance (Rassokhin). The research question will be formulated in the following way: *“To what extent do the computational algorithms written in C++ programming language differ in time complexity when computing the peak wavelength of the blackbody radiation spectrum?”*

Aim and Methodology

In this internal assessment, two computational algorithms will be applied to the computation of the peak wavelength of the blackbody radiation spectrum:

- 1) Arithmetic algorithm
- 2) Gradient descent algorithm

The algorithms will be written in C++ programming language in Microsoft Visual Studio 2022 IDE (Integrated Development Environment).

To compare the time complexity of the two algorithms, its function will be developed for each algorithm using Big-O notation. This mathematical notation is used to characterize the limiting behavior of a function as its argument tends towards a particular value or infinity (*“What Is Big O Notation?”*). The time complexity function is written within parentheses after a capital letter *O*. For instance, $O(n)$ means that the runtime of an algorithm increases linearly with the input size n . The algorithm where the impact of the input size on its runtime is the least is considered the most efficient in terms of time complexity. For example, an algorithm whose runtime is constant and does not depend in the input size at all is believed to have the best time complexity. It's Big-O notation is $O(1)$ (Prado). The following picture illustrates how different time complexities expressed in Big-O notation are compared:

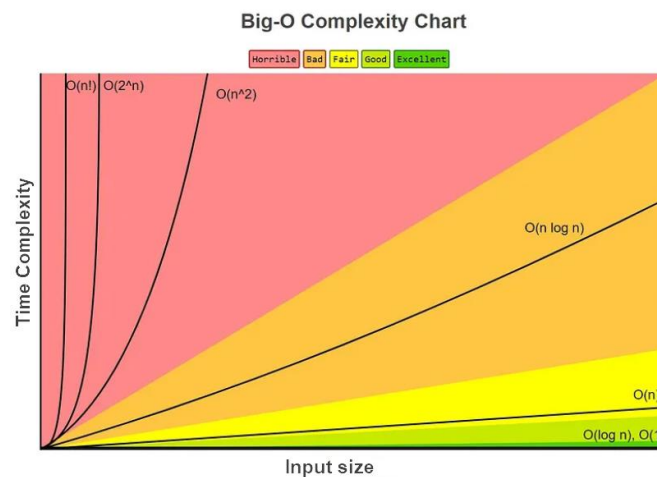


Figure 2: Comparing time complexities expressed as functions using Big-O notation (Prado)

Development of the Arithmetic Algorithm

To develop the arithmetic algorithm for the computation of the peak wavelength of the blackbody radiation spectrum, the derivative of the spectral power density function with respect to wavelength $\frac{dB}{d\lambda}$ must be analyzed. The derivative of the spectral power density function with respect to wavelength has to be equated to zero to find the peak wavelength because this is where the global maximum of this function occur. The calculation of the derivative is provided below:

- To simplify the calculation, the following constants will be defined: $\alpha = \frac{hc}{k}$, $\beta = 2\pi hc^2$. After that, the spectral power density function will look like so:

$$B(\lambda, T) = \frac{\beta}{\lambda^5} \cdot \frac{1}{e^{\frac{\alpha}{\lambda T}} - 1}$$

Formula 2: Simplified spectral power density function

- To calculate the derivative of the spectral power density function with respect to wavelength, the product rule will be applied. The derivative of the spectral power density function with respect to wavelength $\frac{dB}{d\lambda}$ will be considered as the derivative of the product of two functions:

$$u(\lambda) = \frac{\beta}{\lambda^5}$$

$$v(\lambda) = \frac{1}{e^{\frac{\alpha}{\lambda T}} - 1}$$

- To apply the product rule, the derivatives of these functions must be calculated:

$$\frac{du}{d\lambda} = -\frac{5\beta}{\lambda^6}$$

$$\frac{dv}{d\lambda} = \left(-\frac{1}{\left(e^{\frac{\alpha}{\lambda T}} - 1 \right)^2} \right) \cdot \left(-\frac{\alpha}{\lambda^2 T} \right) \cdot e^{\frac{\alpha}{\lambda T}}$$

- Thus, the derivative of the spectral power density function with respect to wavelength will be the following:

$$\begin{aligned} \frac{dB}{d\lambda} &= \frac{du}{d\lambda} v(\lambda) + u(\lambda) \frac{dv}{d\lambda} = -\frac{5\beta}{\lambda^6} \left(\frac{1}{e^{\frac{\alpha}{\lambda T}} - 1} \right) + \frac{\beta}{\lambda^5} \cdot \left(-\frac{1}{\left(e^{\frac{\alpha}{\lambda T}} - 1 \right)^2} \right) \cdot \left(-\frac{\alpha}{\lambda^2 T} \right) \cdot e^{\frac{\alpha}{\lambda T}} = \\ &= \frac{\beta}{\lambda^6 \left(e^{\frac{\alpha}{\lambda T}} - 1 \right)^2} \left(5 - 5e^{\frac{\alpha}{\lambda T}} + \frac{\alpha}{\lambda T} e^{\frac{\alpha}{\lambda T}} \right) \end{aligned}$$

Formula 3: The differentiation of the spectral power density function with respect to wavelength

After that, the derivative will be equated to zero and the equation will be solved for λ to find the peak wavelength (λ_{max}). The procedure will be written below:

$$\frac{dB}{d\lambda} = 0$$

$$\frac{\beta}{\lambda^6 \left(e^{\frac{\alpha}{\lambda T}} - 1 \right)^2} \left(5 - 5e^{\frac{\alpha}{\lambda T}} + \frac{\alpha}{\lambda T} e^{\frac{\alpha}{\lambda T}} \right) = 0$$

Since $\beta \neq 0$, the expression in brackets must be equal to zero:

$$5 - 5e^{\frac{\alpha}{\lambda T}} + \frac{\alpha}{\lambda T} e^{\frac{\alpha}{\lambda T}} = 0$$

$$5e^{\frac{\alpha}{\lambda T}} - \frac{\alpha}{\lambda T} e^{\frac{\alpha}{\lambda T}} = 5$$

$$e^{\frac{\alpha}{\lambda T}} \left(5 - \frac{\alpha}{\lambda T} \right) = 5$$

The substitution $x = \frac{\alpha}{\lambda T}$ will be used:

$$e^x(5 - x) = 5$$

To find the general solution to this equation, the two equations $f(x) = 5$ and $f(x) = e^x(5 - x)$ will be plotted on the same graph and the points of their intersection will be found. This will be done using Desmos calculator.

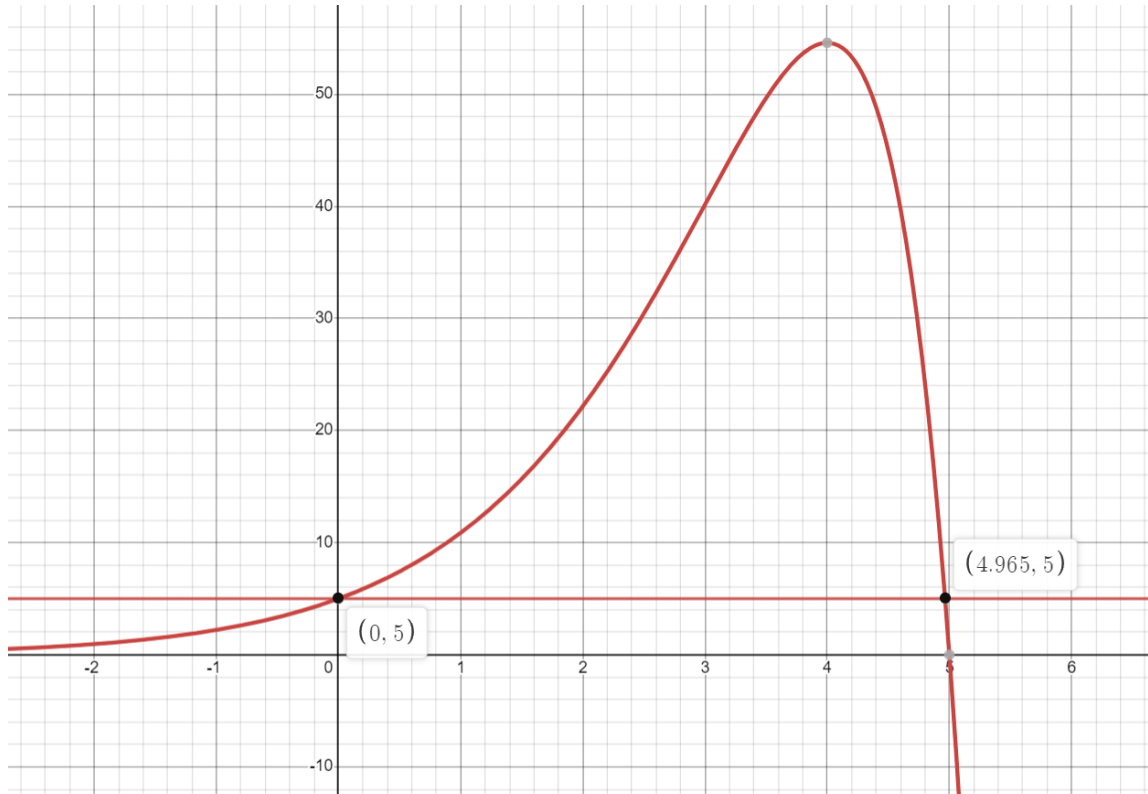


Figure 3: The solutions to the equation $e^x(5-x) = 5$

The solution $x = \frac{\alpha}{\lambda T} = \frac{hc}{\lambda kT} = 0$ is trivial and is only valid in the high-temperature limit $T \rightarrow \infty$. The non-trivial solution is $x \simeq 4.965$ (the more precise value is 4.9651142). At this point, it is possible to find the value of λ_{max} :

$$x = \frac{hc}{\lambda kT}$$

$$\lambda = \lambda_{max} = \frac{hc}{xkT} \simeq \frac{hc}{4.965kT} \simeq \frac{2.897163 \cdot 10^{-3}}{T}$$

Formula 4: The formula to compute the peak wavelength

Now, the arithmetic algorithm for the computation of the peak wavelength of the blackbody radiation spectrum can be developed in C++. The steps will be the following:

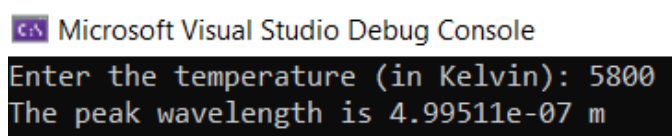
- 1) Ask the user to input the blackbody's temperature in Kelvin.
- 2) Compute the peak wavelength using the formula that was derived above (Formula 4).
- 3) Output the result.

The implementation of this algorithm in C++ is demonstrated in the figure below. The comments in green describe what each line of code does.

```
1 // include the header file that is a part of the C++ standard library for input and output
2 #include <iostream>
3
4 // define the main() function where the algorithm will be written
5 int main() {
6
7     double temperature; // Temperature variable in Kelvin definition
8     double peakWavelength; // Peak wavelength variable in metres definition
9
10    // Input temperature from the user
11    std::cout << "Enter the temperature (in Kelvin): ";
12    std::cin >> temperature;
13
14    // Compute the peak wavelength
15    peakWavelength = 2.897163e-3 / temperature;
16
17    // Output the result
18    std::cout << "The peak wavelength is " << peakWavelength << " m " << std::endl;
19
20    // Exit the main() function
21    return 0;
22 }
```

Figure 4: The arithmetic algorithm for the computation of the peak wavelength of the blackbody radiation spectrum developed in C++

If the user inputs the temperature of the Sun (5800 K), the output of the program will be the following:



Microsoft Visual Studio Debug Console

```
Enter the temperature (in Kelvin): 5800
The peak wavelength is 4.99511e-07 m
```

Figure 5: The output of the program

To determine if the algorithm produces the correct result, the literature values of the peak wavelength were checked. These values corresponded to the algorithm's result. For example, in the article published by the Pennsylvania State University, it is stated that the peak wavelength of the Sun radiation spectrum is close to 500 nanometers (nm), that is $5.0 \cdot 10^{-7}$ m (PennState College).

Development of the Gradient Descent Algorithm

Gradient descent is an iterative algorithm that can be used to calculate the minimum or maximum of a function (Kwiatkowski). The steps are the following:

- 1) A point close to the maximum/minimum of the function must be selected on the graph of the function.
- 2) The gradient of the tangent at this point has to be calculated using the function's derivative.
- 3) The gradient is supposed to be scaled by the learning rate (the parameter that controls how big is the step towards the function's minimum/maximum) (Murphy).
- 4) The scaled gradient is subtracted from the current point if we move towards the function's minimum or added to it if we move towards the function's maximum.

The process described above is represented by the gradient descent formula:

$$p_{n+1} = p_n \pm \eta \nabla f(p_n)$$

Formula 5: Gradient Descent Formula (Kwiatkowski)

Where p_{n+1} is the point on the function calculated for the next iteration, p_n is the current point on the function, η is the learning rate, $\nabla f(p_n)$ is the gradient of tangent at the current point p_n .

To find the global maximum of the spectral power density function, the scaled gradient will be added to the current point because the point on the function needs to be moved towards its maximum. Therefore, the gradient descent formula needed for our calculations will look like so:

$$p_{n+1} = p_n + \eta \nabla f(p_n)$$

Formula 6: Gradient Descent Formula adjusted for the peak wavelength computation

Let us suppose that we want to compute the peak wavelength of the radiation spectrum at the Sun temperature (5800 K). From article published by the Pennsylvania State University mentioned above, we know that its value is close to $5.0 \cdot 10^{-7}$ m. Therefore, the initial point on the spectral power density function p_0 will be set to this value. Next, we will calculate the gradient of tangent at this point using the derivative of the spectral power density function (Formula 3):

$$\frac{dB}{d\lambda} \Big|_{\lambda=p_0} = \frac{dB}{d\lambda} \Big|_{\lambda=5.0 \cdot 10^{-7}} = -7.97819 \cdot 10^{17}$$

It is possible to make an observation that the gradient of tangent is negative at the initial point p_0 when $\lambda = 5.0 \cdot 10^{-7}$. This means that the spectral power density function decreases at this point. Since the spectral power density function decreases after the turning point (the global maximum that was computed to be $4.99511 \cdot 10^{-7}$ m by the arithmetic algorithm described above), the negative sign of the gradient of tangent at $\lambda = 5.0 \cdot 10^{-7}$ must be reasonable.

The learning rate η does not have to be large because if it is, the algorithm may jump over the maximum point and fail to achieve its goal. However, it does not have to be too little as well because otherwise the algorithm will not reach the maximum point. The learning rate for a particular problem can only be defined experimentally in the range between 0.0 and 1.0 (Brownlee). When I was developing the algorithm in C++, I determined that the most optimal learning rate for the peak wavelength computation is $\eta = 1.0 \cdot 10^{-28}$.

Now, it is possible to calculate the value of the current point on the function after the first iteration of the gradient descent algorithm for the peak wavelength computation:

$$p_1 = p_0 + \eta \nabla f(p_0) = 5.0 \cdot 10^{-7} + 1.0 \cdot 10^{-28} \cdot (-7.97819 \cdot 10^{17}) = 4.99920 \cdot 10^{-7}$$

At this point, the value of the gradient of tangent will be:

$$\left. \frac{dB}{d\lambda} \right|_{\lambda=p_1} = \left. \frac{dB}{d\lambda} \right|_{\lambda=4.99920 \cdot 10^{-7}} = -6.67909 \cdot 10^{17}$$

As we see, the value of the gradient of tangent increases and starts moving towards zero (where the value of the spectral power density function is maximized). In the next iterations, this continues. The values of the current point on the spectral power density function p_n and the gradient of tangent at this point $\nabla f(p_n)$ for the first four iterations are presented in the table below:

Iteration	p_n	$\nabla f(p_n)$
0	$5.00000 \cdot 10^{-7}$	$-7.97819 \cdot 10^{17}$
1	$4.99920 \cdot 10^{-7}$	$-6.67909 \cdot 10^{17}$
2	$4.99853 \cdot 10^{-7}$	$-5.59086 \cdot 10^{17}$
3	$4.99798 \cdot 10^{-7}$	$-4.67947 \cdot 10^{17}$

Table 2: The values of p_n and $\nabla f(p_n)$ for the first four iterations

It is important to keep in mind that the gradient descent loop must have the termination condition (iterations must stop when the value of the gradient of tangent at the point p_n will be very close to zero, for example, $-1.0 \cdot 10^{-10}$). It can be seen from the table above that this value must be negative because the gradient of tangent at the current point $\nabla f(p_n)$ will approach zero from the negative side of the number line:

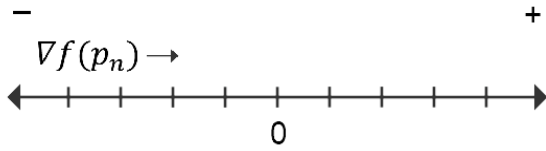


Figure 6: $\nabla f(p_n)$ approaches zero from the negative side of the number line

With the considerations described above, the gradient descent algorithm for the computation of the peak wavelength of the blackbody radiation spectrum can now be developed in C++.

```

1 // Define _USE_MATH_DEFINES to access mathematical constants
2 #define _USE_MATH_DEFINES
3
4 // Include the C++ library with mathematical functions
5 #include <cmath>
6
7 // Include the header file that is a part of C++ standard library for input and output
8 #include <iostream>
9
10 // Use namespace std that contains classes, functions, and objects provided by the C++ standard library
11 using namespace std;
12
13 // Physical constants definitions
14 const double h = 6.62607015e-34; // Planck's constant (J·s)
15 const double c = 299729458; // Speed of light (m/s)
16 const double k = 1.380649e-23; // Boltzmann constant (J/K)
17
18 // Mathematical constants definitions
19 const double pi = M_PI;
20 const double NeperNumber = M_E;
21
22 // Function to calculate the the gradient of tangent at the current point on the spectral power density function
23 // using its derivative with respect to wavelength
24 double derivative(double lambda, double T) {
25
26     return (((2*pi*h*pow(c, 2))/(pow(lambda, 6)*pow(pow(NeperNumber, h*c/(lambda*k*T))-1, 2)))
27             *(5-5*pow(NeperNumber, h*c/(lambda*k*T))+h*c/(lambda*k*T)*pow(NeperNumber, h*c/(lambda*k*T))));
28 }
29

```



```

30 int main() {
31     // Ask the user to input the initial guess for the peak wavelength in meters
32     double peakWavelength;
33     cout << "Input the initial guess for the peak wavelength" << endl;
34     cin >> peakWavelength;
35
36     // Ask the user to input the temperature of the blackbody in Kelvin
37     double temperature;
38     cout << "Input the temperature of the blackbody in Kelvin" << endl;
39     cin >> temperature;
40
41     // Learning rate for gradient descent
42     double learningRate = 1e-28;
43
44     // Maximum number of iterations
45     int maxIterations = 1000;
46
47     // Gradient descent loop
48     for (int i = 0; i < maxIterations; i++) {
49
50         // Calculate the gradient of tangent at the current point on the spectral power density function
51         // using its derivative
52         double gradient = derivative(peakWavelength, temperature);
53
54         // Update the current point on the spectral power density function using the gradient descent formula
55         peakWavelength += learningRate*gradient;
56
57         // Termination of the algorithm condition (gradient magnitude very close to zero)
58         if (gradient > (-1e-10)) {
59             break;
60         }
61     }
62
63     // Output the peak wavelength
64     cout << "Peak wavelength = " << peakWavelength << " meters" << endl;
65
66     return 0;
67 }

```

Figure 7: The gradient descent algorithm for the computation of the peak wavelength of the blackbody radiation spectrum developed in C++

When the user inputs the temperature of Sun in Kelvin and chooses the initial guess for the peak wavelength (initial point on the spectral power density function p_0) to be $5.0 \cdot 10^{-7}$, the output of the program is the following:

Microsoft Visual Studio Debug Console

```

Input the initial guess for the peak wavelength
5e-7
Input the temperature of the blackbody in Kelvin
5800
Peak wavelength = 4.99511e-07 meters

```

Figure 8: The output of the program

It can be seen that the gradient descent algorithm computed the peak wavelength to be $4.99511 \cdot 10^{-7}$ m. This value is exactly the same as the one that was calculated by the arithmetic algorithm.

Assessment of the Algorithms' Time Complexities

In order to express the time complexities of both algorithms in Big-O notation, it is necessary to consider the following rules:

1) When determining the time complexity of an algorithm, the first step is to count the number of basic operations performed by the algorithm (arithmetic operations, assignments of values to variables, comparisons) as a function of the input size. This function is denoted as $T(n)$ where n is the input size (SaturnCloud).

2) During the operations counting, it is a must to consider the number of operations the algorithm will take to complete in the worst-case scenario (Reynaga).

3) The non-dominant terms of the function $T(n)$ must be dropped. Dropping them means leaving only the dominant term. This is the term that grows the fastest as the input size increases and influences the overall time complexity most. The Big-O notation focuses just on it. Additionally, constant factors have to be removed from the function's equation because their impact on the runtime of the algorithm is considered to be negligible. For example, if $T(n) = 2n^2 + 3n + 1$, it must be simplified to $O(n^2)$ by disregarding lower-order terms and constant coefficients (Bui).

In the arithmetic algorithm for the peak wavelength computation, there are four operations performed:

- 3 input-output operations that usually take constant time and their number does not depend on the input size.
- The computation of the peak wavelength that is not dependent on the input size too because it has a fixed number of mathematical operations that cannot be modified by the value of *temperature* variable which represents the input size because this is the variable that the user enters.

Thus, $T(n) = 4$. This means that the number of operations performed by the arithmetic algorithm will always be 4, regardless of the input size n . Therefore, its time complexity will be $O(1)$ because this is how the time complexity of an algorithm is denoted in Big-O notation if its runtime is constant and does not depend on the input size.

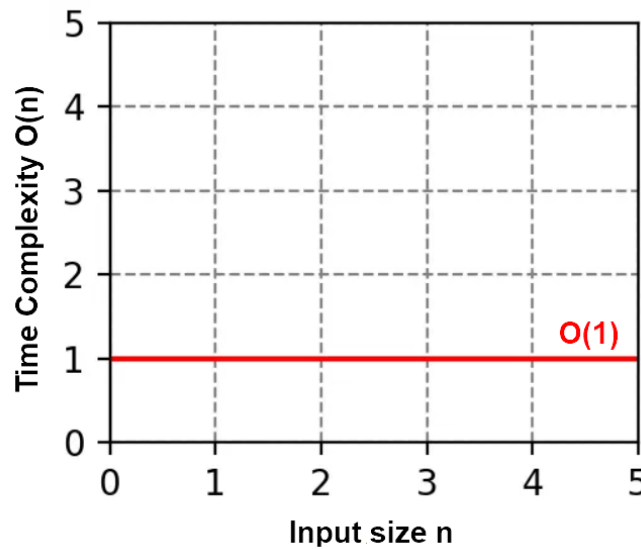


Figure 8: The time complexity for the arithmetic algorithm for the peak wavelength computation (Bui)

In the gradient descent algorithm for the peak wavelength computation, there is a number of operations:

- Inside the *derivative* function that calculates the gradient of tangent at the current point $\nabla f(p_n)$, there is a series of mathematical operations (subtraction, division, multiplication and exponentiation). The number

of subtraction, division and multiplication operations done by the function does not depend on the input size represented by the initial guess for the peak wavelength and the temperature of blackbody entered by the user. However, exponentiation in this function is performed by the `pow()` function from the `<cmath>` C++ library whose arguments include the base number and the exponent. This function repeatedly divides the exponent by 2 and squares the base until the exponent becomes zero. The time complexity of this function is logarithmic (denoted by $O(\log(n))$) because the logarithmic function represents the number of times a base has to be multiplied or divided by itself to reach the target value, which is zero in this case (Geeks for Geeks). Since $O(\log(n))$ will be the dominant term because it influences the time taken by the *derivative* function to execute most, the time complexity of the *derivative* function can be considered to be logarithmic.

- Inside the gradient descent loop that performs iterations that make the gradient of the current point $\nabla f(p_n)$ to approach zero gradually, there are some operations too. The loop will iterate until either the maximum number of iterations to reach the solution set by the programmer will be equal to 1000 or until the condition that the gradient of the current point $\nabla f(p_n)$ is very close to zero is met. According to the second Big-O notation rule described earlier, we must consider the number of operations the algorithm will take to complete in the worst-case scenario, which is 1000, is constant and does not depend on the input size entered by the user. The loop performs the execution of the *derivative* function whose time complexity was determined earlier to be $O(\log(n))$.

Apart from that, the loop does operations of updating the *peakWavelength* variable, and the check of the condition whether the gradient of the current point $\nabla f(p_n)$ is very close to zero. The maximum number of these operations is constant (1000) and does not depend on the input size. Therefore, the dominant factor that influences the time complexity of this loop is the time complexity of the *derivative* function executed by it, which is $O(\log(n))$. So the time complexity of the loop will be the same as that of the *derivative* function $O(\log(n))$.

- 10 operations are outside the gradient descent loop and the *derivative* function. There are 5 operations that assign values to variables and 5 input-output operations. These operations take constant time and their number does not depend on the input size, so their time complexity will be constant ($O(1)$).

With the above considerations, it is possible to make a conclusion that the dominant factor affecting the time complexity of the gradient descent algorithm for the peak wavelength computation will be $O(\log(n))$, so its time complexity will be logarithmic.

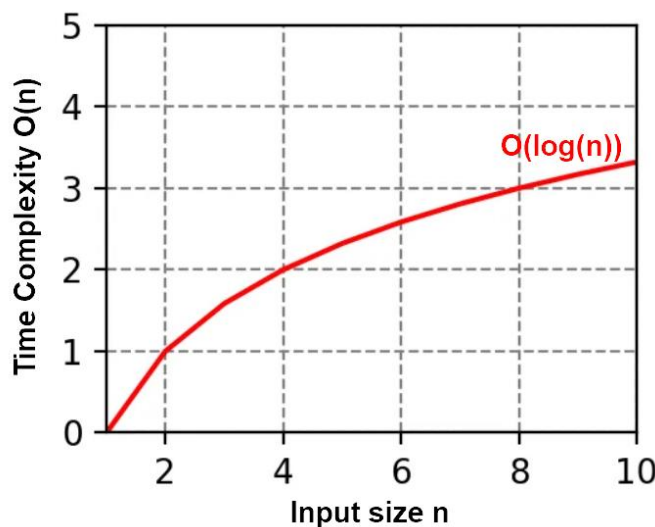


Figure 9: The time complexity for the gradient descent algorithm for the peak wavelength computation (Bui)

Evaluation and Conclusion

The arithmetic and gradient descent algorithms both have a good time complexity because the impact of the input size on their runtime is very low. In case of the arithmetic algorithm, it is absent at all. This means that both algorithms can be used time-efficiently to compute the peak wavelength of the blackbody radiation spectrum. However, it must be pointed out that the time complexity will be slightly better for the arithmetic algorithm because the input size in it influences the runtime of the gradient descent algorithm more.

The Big-O notation selected for this investigation provides the time complexity of an algorithm in the worst case (when the number of operations performed by the algorithm is maximized) (Educative). This notation was chosen because it is most commonly used and known among software developers and scientists who could potentially use this research for their purposes (Singh). On the other hand, the analysis for the time complexity could be a little bit more precise if the expression for the time complexity was not simplified (if the non-dominant terms and constant factors were not excluded). In this way, the time complexity function would provide information about the algorithms' behavior not only for the worst case.

Additionally, this internal assessment could be improved if not only the arithmetic and gradient descent algorithms would be assessed in terms of the time complexity. For example, another good algorithm suitable for the peak wavelength computation would be the Newton-Raphson algorithm. Despite the fact that it was originally designed for root-finding problems, it could be modified to compute the global maximum of a function (Ludwig Maximilian University of Munich).

Although this internal assessment provides valuable insights that could possibly help astronomers to choose an algorithm for the computation of the peak wavelength, it should be noted that the time complexity is just one of dozens of factors that need to be considered during the algorithm's selection. For example, another useful property of algorithms to assess would be their space complexity (memory usage relative to the input size) ("Time and Space Complexity Analysis").

During my mathematics internal assessment journey, I was failing to develop the gradient descent algorithm that computes the peak wavelength correctly many times. My calculations results did not correspond with the literature values at all. However, after my first successful attempt to compute the peak wavelength using the algorithm I wrote, I understood that success in mathematics requires patient perseverance. If the problem is analyzed for a long time, the solution will eventually be found. My working algorithms and completed internal assessment inspired me to dive more into the math of computer science. My research process convinced me that mathematics has many useful and exciting applications in my field of interest.

Works Cited

- “Big O Notation: A Primer for Beginning Devs.” *Educative*, 26 Dec. 2019, www.educative.io/blog/a-big-o-primer-for-beginning-devs.
- “Blackbody Radiation.” *European Space Agency*, 1 Sept. 2019, sci.esa.int/web/education/-/48986-blackbody-radiation.
- “Blackbody Radiation.” *LibreTexts Physics*, 4 Aug. 2022, [phys.libretexts.org/Bookshelves/University_Physics/Book%3A_University_Physics_\(OpenStax\)/University_Physics_III_-_Optics_and_Modern_Physics_\(OpenStax\)/06%3A_Photons_and_Matter_Waves/6.02%3A_Blackbody_Radiation](https://phys.libretexts.org/Bookshelves/University_Physics/Book%3A_University_Physics_(OpenStax)/University_Physics_III_-_Optics_and_Modern_Physics_(OpenStax)/06%3A_Photons_and_Matter_Waves/6.02%3A_Blackbody_Radiation).
- Brownlee, Jason. “Understand the Impact of Learning Rate on Neural Network Performance.” *Machine Learning Mastery*, 12 Sept. 2020, machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/#:~:text=Learning%20Rate%20and%20Gradient%20Descent,-Deep%20learning%20neural&text=Specifically%2C%20the%20learning%20rate%20is,is%20adapted%20to%20the%20problem.
- Bui, Truong. “A Simple Explanation of Big O Notation With Examples.” *Medium*, 3 July 2023, medium.com/@truongbui95/a-simple-explanation-of-big-o-notation-with-examples-e98b2c2fefa8.
- Grauer, Samuel J. “Volumetric Emission Tomography for Combustion Processes.” *Progress in Energy and Combustion Science*, vol. 94, Jan. 2023. *ScienceDirect*, www.sciencedirect.com/science/article/abs/pii/S0360128522000338.
- Jones, Barrie W., and S. Jocelyn Bell Burnell. *An Introduction to the Sun and Stars*. Edited by Simon F. Green and Mark H. Jones, Cambridge UP, 2004.
- Kwiatkowski, Robert. “Gradient Descent Algorithm — a Deep Dive.” *Towards Data Science*, 22 May 2021, towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21.
- Murphy, Kevin P. “Machine Learning: A Probabilistic Perspective.” *MIT Press*, 2012.

“Newton-Raphson (NR) Optimization.” *Ludwig Maximilian University of Munich*, 16 Oct. 2004,

www.cup.uni-muenchen.de/ch/compchem/geom/nr.html.

PennState College of Earth and Mineral Sciences. “Blackbody Radiation.” *Dutton Institute for Teaching*

and Learning Excellence, 14 June 2020, [www.e-](http://www.e-education.psu.edu/astro801/book/export/html/1548#:~:text=To%20determine%20the%20peak%20wavelength,c%20m%20%3D%20500%20n%20m)

[education.psu.edu/astro801/book/export/html/1548#:~:text=To%20determine%20the%20peak%20wavelength,c%20m%20%3D%20500%20n%20m](http://www.e-education.psu.edu/astro801/book/export/html/1548#:~:text=To%20determine%20the%20peak%20wavelength,c%20m%20%3D%20500%20n%20m).

Planck, Max. *The Theory of Heat Radiation*. 2nd ed., P. Blakiston’s Son and Co., 1914.

Prado, Kelvin Salton do. “Understanding Time Complexity With Python Examples.” *Towards Data*

Science, 4 Mar. 2019, towardsdatascience.com/understanding-time-complexity-with-python-examples-2bda6e8158a7.

Rassokhin, Dmitrii. “The C++ Programming Language in Cheminformatics and Computational

Chemistry.” *Journal of Cheminformatics*, Jan. 2020. *BioMedCentral*,

[jcheminf.biomedcentral.com/articles/10.1186/s13321-020-0415-](https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-0415-y#:~:text=In%20scientific%20programming%2C%20C%2B%2B%20is,advantage%20of%20its%20expressive%20power)

[y#:~:text=In%20scientific%20programming%2C%20C%2B%2B%20is,advantage%20of%20its%20expressive%20power](https://jcheminf.biomedcentral.com/articles/10.1186/s13321-020-0415-y#:~:text=In%20scientific%20programming%2C%20C%2B%2B%20is,advantage%20of%20its%20expressive%20power).

Reynaga, Abril Anchondo. “A Simple Explanation of Big O and the Linear Search Algorithm.” *Medium*,

12 Jan. 2020, medium.com/@abril79reynaga/a-simple-explanation-of-big-o-and-the-linear-search-algorithm-5d718f1d80a0.

Singh, Harpal. “Practical Examples of the Big O Notation.” *Baeldung*, 12 July 2023,

[www.baeldung.com/cs/big-oh-asymptotic-](https://www.baeldung.com/cs/big-oh-asymptotic-complexity#:~:text=Big%20O%2C%20Big%20%CE%A9%2C%20and,based%20on%20the%20input%20size)

[complexity#:~:text=Big%20O%2C%20Big%20%CE%A9%2C%20and,based%20on%20the%20input%20size](https://www.baeldung.com/cs/big-oh-asymptotic-complexity#:~:text=Big%20O%2C%20Big%20%CE%A9%2C%20and,based%20on%20the%20input%20size).

“Time and Space Complexity Analysis.” *SimpliLearn*, uploaded by Soni Upadhyay,

[www.simplilearn.com/tutorials/data-structure-tutorial/time-and-space-](https://www.simplilearn.com/tutorials/data-structure-tutorial/time-and-space-complexity#:~:text=Space%20complexity%20refers%20to%20the,program%20to%20determine%20space%20complexity)

[complexity#:~:text=Space%20complexity%20refers%20to%20the,program%20to%20determine%20space%20complexity](https://www.simplilearn.com/tutorials/data-structure-tutorial/time-and-space-complexity#:~:text=Space%20complexity%20refers%20to%20the,program%20to%20determine%20space%20complexity). Accessed 31 July 2023.

“Time Complexity Calculation for My Algorithm.” *SaturnCloud*, 18 July 2023, saturncloud.io/blog/time-complexity-calculation-for-my-algorithm.

University of Colorado Boulder. “Blackbody Spectrum.” *PhET Interactive Simulations*, 16 May 2023, phet.colorado.edu/sims/html/blackbody-spectrum/latest/blackbody-spectrum_all.html.

“What Is Big O Notation?” *A.I. for Anyone*, 1 Oct. 2022, www.aiforanyone.org/glossary/big-o-notation#:~:text=input%20size%20grows.-,Big%20O%20notation%20is%20a%20mathematical%20notation%20that%20describes%20the,a%20particular%20value%20or%20infinity.

“What Is Time Complexity and Why Is It Essential?” *Great Learning*, 24 Aug. 2023, www.mygreatlearning.com/blog/why-is-time-complexity-essential/#what-are-the-different-types-of-time-complexity-notation-used.

“Write an Iterative $O(\log Y)$ Function for $\text{Pow}(X, Y)$.” *Geeks for Geeks*, www.geeksforgeeks.org/write-an-iterative-olog-y-function-for-powx-y. Accessed 12 Apr. 2023.

Wyatt, Clair L. *Radiometric Calibration: Theory and Methods*. 1978. *ScienceDirect*, www.sciencedirect.com/book/9780127661506/radiometric-calibration-theory-and-methods#book-info.