algorithms

MDPI

Article

GPU Adding-Doubling Algorithm for Analysis of Optical Spectral Images

Matija Milanic [1,2] and Rok Hren [1,3,4,*]

[1] Faculty of Mathematics and Physics, University of Ljubljana, SI-1000 Ljubljana, Slovenia
[2] Jozef Stefan Institute, SI-1000 Ljubljana, Slovenia
[3] Institute of Mathematics, Physics, and Mechanics, SI-1000 Ljubljana, Slovenia
[4] Syreon Research Institute, 1145 Budapest, Hungary
* Correspondence: rok.hren@fmf.uni-lj.si

Abstract: The Adding-Doubling (AD) algorithm is a general analytical solution of the radiative transfer equation (RTE). AD offers a favorable balance between accuracy and computational efficiency, surpassing other RTE solutions, such as Monte Carlo (MC) simulations, in terms of speed while outperforming approximate solutions like the Diffusion Approximation method in accuracy. While AD algorithms have traditionally been implemented on central processing units (CPUs), this study focuses on leveraging the capabilities of graphics processing units (GPUs) to achieve enhanced computational speed. In terms of processing speed, the GPU AD algorithm showed an improvement by a factor of about 5000 to 40,000 compared to the GPU MC method. The optimal number of threads for this algorithm was found to be approximately 3000. To illustrate the utility of the GPU AD algorithm, the Levenberg–Marquardt inverse solution was used to extract object parameters from optical spectral data of human skin under various hemodynamic conditions. With regards to computational efficiency, it took approximately 5 min to process a 220 × 100 × 61 image (x-axis × y-axis × spectral-axis). The development of the GPU AD algorithm presents an advancement in determining tissue properties compared to other RTE solutions. Moreover, the GPU AD method itself holds the potential to expedite machine learning techniques in the analysis of spectral images.

Keywords: adding-doubling method; GPU implementation; optical spectral images; tissue properties; computational speed-up and accuracy

Citation: Milanic, M.; Hren, R. GPU Adding-Doubling Algorithm for Analysis of Optical Spectral Images. *Algorithms* **2024**, *17*, 74. https://doi.org/10.3390/a17020074

Academic Editor: Francesc Pozo

Received: 26 December 2023
Revised: 31 January 2024
Accepted: 4 February 2024
Published: 7 February 2024

1. Introduction

Building on Stokes' method to calculate the transmission of light through a pile of plates [1], van de Hulst [2] in 1962 proposed the Adding-Doubling (AD) algorithm, an efficient technique to solve the radiative transfer equation (RTE) within complex media, such as clouds, aerosols, and biological tissues. Numerous studies have adopted the AD algorithm to advance our understanding of atmospheric processes and remote sensing applications [3–5], luminescence in solar cells [6], biomedical optics [7,8], and photovoltaic and nuclear reactor research [9–11].

Briefly, the AD algorithm consists of two essential components: the Doubling method and the Adding method [12,13]. The Doubling method relies on prior knowledge of the reflection and transmission properties of a single thin homogeneous layer. By combining two identical slabs and summing their individual contributions, the reflection and transmission of a slab that is twice as thick can be determined. This doubling process is then iteratively applied, progressively increasing the slab's thickness until the desired magnitude is achieved. The Adding method extends the Doubling method by allowing for the simulation of media that consist of diverse layers and involve instances of internal reflection at boundaries. By incorporating dissimilar slabs into the calculations, we can achieve a more comprehensive representation of complex media structures.

Understanding the interaction of photons with biological tissues holds immense importance in various fields, including medical imaging, diagnostics, and laser therapy. To model this intricate process, two widely used approaches are the Monte Carlo (MC) method [14] and the AD algorithm [13]. The MC method is a statistical technique that simulates photon transport by creating random photon paths within the tissue. It uses probabilistic calculations to model absorption, scattering, and reflection/refraction events based on the tissue's optical properties. Starting from an initial position and direction, photons are randomly sampled and tracked as they interact with the tissue. At each interaction, the path length and probability of scattering or absorption are considered, allowing the simulation to replicate the complex behavior of photons in a stochastic manner. The MC method offers several advantages, including its flexibility to simulate complex geometries and heterogeneous tissue structures. It is particularly adept at capturing intricate molecular and cellular photon interactions, making it valuable for biological and medical research. However, due to its statistical nature, the MC method requires an extremely large number of photons to achieve accurate results, leading to computationally intensive simulations. Our preliminary unpublished research revealed that analyzing a hyperspectral image of dimension 223 $\times$ 452 using the GPU MC method would entail approximately 9 min per spectrum, or roughly 1.7 years per single image, which would render the MC method impractical. Similar computational times when using the MC method were also reported in the literature [15].

The AD algorithm enables the calculation of light propagation in layered turbid media in a slab geometry. The algorithm breaks down the biological tissue into multiple layers, each with its specific properties, such as absorption and scattering coefficients. Using an iterative process, it determines the reflected and transmitted light at each layer interface. By aggregating the contributions from each layer, the AD algorithm provides an estimation of the overall photon distribution within the tissue. The AD algorithm accounts for both the multiple scattering events within the tissue and the interface effects, yielding accurate predictions of light transport. This capability is particularly valuable in applications such as skin imaging, where the multiple layers of the epidermis and dermis need to be accounted for. The AD algorithm is particularly well-suited for multi-layer homogeneous tissue models with known optical properties, providing a quick and efficient solution, which may be, in turn, practical for real-time simulations, clinical applications, and large-scale studies.

In recent years, the integration of graphics processing units (GPUs) has proven to be highly effective in reducing execution time and has revolutionized computational efficiency and accelerated processing, particularly in the field of medical imaging [16–18]. GPU platforms provide means to accelerate specific computational tasks and algorithms, surpassing the performance of central processing units (CPUs) while retaining a desirable level of flexibility [19]. In our study, we focus on implementing AD on GPUs to achieve enhanced computational speed; additionally, we augment GPU AD with the GPU Levenberg–Marquardt (LM) algorithm to extract object parameters from optical spectral data. The overall goal of our study was to address the practical challenges associated with analyzing hyperspectral images. We aimed to develop an algorithm that could deliver both computational speed and accuracy while yielding results comparable to the MC method. With this pragmatic focus, we sought to make the analysis of hyperspectral images more manageable.

2. Materials and Methods

2.1. Preliminaries of Adding Doubling (AD) Algorithm

The AD algorithm is specifically designed for layered geometries with uniform irradiation, requiring each layer to possess homogeneous optical properties. These assumptions align well with the requirements of layered or homogenous samples used in hyperspectral imaging (HSI), making the AD algorithm a highly practical and appealing method for fast analysis of hyperspectral images. In this subsection, the details of the AD algorithm are presented, following the formalism of Prahl et al. [13]. We have divided the AD algorithm into seven distinct steps, outlined in Sections 2.1.1–2.1.7.

2.1.1. Division of Incident and Reflected Light into Fluxes

Within the framework of the AD algorithm, incoming and outcoming light is divided into m conical fluxes [13], as shown in Figure 1. A cone of light incident on any flat surface will transmit and reflect different amounts of light depending on the angle of departure. To capture this, we introduce the notation for the cosine of an arbitrary angle $\nu_c = \cos(\theta)$. The function $R(\nu', \nu)$ denotes the reflectance in the direction ν for light incident from the direction ν'. The incident light flux, denoted as $I^+(\nu')$, and the reflected light flux, denoted as $I^-(\nu)$, can be expressed in the matrix form as:

$$\begin{bmatrix} I^-(\nu_1) \\ \vdots \\ I^-(\nu_m) \end{bmatrix} = \begin{bmatrix} R(\nu_1', \nu_1) & \cdots & R(\nu_1', \nu_m) \\ \vdots & & \vdots \\ R(\nu_m', \nu_1) & \cdots & R(\nu_m', \nu_m) \end{bmatrix} \times \begin{bmatrix} I^+(\nu_\prime') \\ \vdots \\ I^+(\nu_m') \end{bmatrix} \tag{1}$$

Analytically, the reflected flux can also be calculated as an integral:

$$I^-(\nu) = \int_{\nu_1}^{\nu_2} I^+(\nu') R(\nu', \nu)\, 2\nu'\, \mathrm{d}\nu' \tag{2}$$

where ν_1 in ν_2 are the boundary angles of the reflected flux $I^-(\nu)$. The total reflected flux for the normal incidence R_c can be determined by the Equation:

$$R_c = \int_0^1 \int_0^1 \frac{\delta(1-\nu')}{2\nu'} R(\nu', \nu)\, 2\nu'\, \mathrm{d}\nu'\, 2\nu\, \mathrm{d}\nu = \int_0^1 R(1, \nu)\, 2\nu\, \mathrm{d}\nu \tag{3}$$

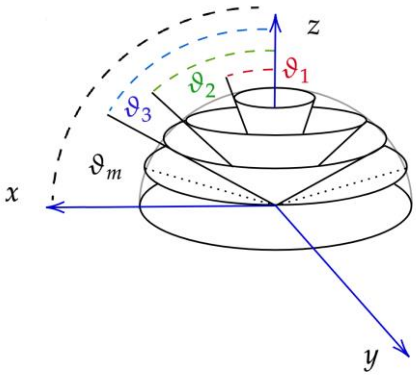

Figure 1. Division of incident and reflected light into m conical segments.

2.1.2. Approximating Reflected Flux Integral Using Quadrature

In the context of the AD algorithm, it is necessary to discretize the integral (Equation (3)) to convert it into matrix form, for which quadrature should be implemented. The integral of the function $f(x)$ on the interval (a, b) with a weight $g(x)$ can be approximated by summing over m points as:

$$\int_a^b f(x)\, g(x)\, \mathrm{d}x \approx \sum_{i=1}^{m} f(x_i)\, w_i \tag{4}$$

The points x_i and associated weights w_i can be chosen to be either uniformly distributed over the interval or as specific points, such as the zeros of the Legendre polynomials. Using the relation from Equation (4), the integral from Equation (3) can be expressed as:

$$\int_0^1 R(1, \nu)\, 2\nu\, \mathrm{d}\nu \approx \sum_{i=1}^{m} 2\nu_i\, w_i\, R_{i,m} \tag{5}$$

where $R_{i,m}$ is the reflectance matrix. AD algorithm further divides the integral (expression for the total reflected flux for the normal incidence R_c in Equation (3)) into two parts: the

first part on the interval $(0, \nu_c)$, and the second part on the interval $(\nu_c, 1]$, where ν_c is the cosine of the critical angle:

$$R_c = \int_0^{\nu_c} R(1,\nu)\, 2\nu \, \mathrm{d}\nu + \int_{\nu_c}^{1} R(1,\nu)\, 2\nu \, \mathrm{d}\nu \tag{6}$$

For the first integral, we use the Gaussian quadrature, and the Radau quadrature is used for the second one. The distinction between these two quadrature methods lies in the consideration of endpoints. In Gaussian quadrature, both endpoints are not included, whereas Radau quadrature accounts for only one endpoint. By adopting this approach, we can avoid the potential singularity at $\nu = 0$ in the Gaussian quadrature, while $\nu = 1$ is taken as a quadrature point in the Radau quadrature. In total, we need to find m roots, one for each conical flux: Gaussian and Radau quadratures obtain each half of the quadrature points $n = m/2$.

In the Radau quadrature, we thus need to find n roots of the following Equation:

$$R_{n-1}(x_i) + \frac{x_i - 1}{n} P_{n-1}{}'(x_i) = 0 \tag{7}$$

where $P_{n-1}(x_i)$ is the $(n-1)$th Legendre polynomial of order zero and $P'_{n-1}(x_i)$ denotes the first derivative of the $(n-1)$th Legendre polynomial. Since these roots correspond to the integration range from -1 to 1, the appropriate adjustments need to be made for integration angles ν_c and weights w_i when performing integration from ν_c to 1:

$$\begin{aligned} \nu_i &= \tfrac{1+\nu_c}{2} - \tfrac{1-\nu_c}{2} x_i \\ w_i &= \tfrac{1-\nu_c}{2(1-x_i) P_{n-1}{}'(x_i)^2} \end{aligned} \tag{8}$$

Similarly, for the Gaussian quadrature, we need to find n roots of the following Equation:

$$P_n(x_i) = 0 \tag{9}$$

with the adjustments made to accommodate the integration range from 0 to ν_c:

$$\begin{aligned} \nu_i &= \tfrac{\nu_c}{2}(1 - x_i) \\ w_i &= \tfrac{\nu_c}{(1-x_i^2)\, P_n{}'(x_i)^2} = \tfrac{\nu_c \left(1-x_i{}^2\right)}{(n+1)^2\, P_{n+1}(x_i)^2} \end{aligned} \tag{10}$$

2.1.3. Calculating Scattering Phase Function

Two matrices are constructed, one for forward scattering, denoted as $\mathbf{P^+}$, and one for backward scattering, denoted as $\mathbf{P^-}$. The elements of scattering matrix P_{ij} determine the fraction of light scattered from an incident cone with angle ν_i into a cone with angle ν_j and are calculated by averaging the phase function $p(\nu_i, \nu_j)$ over all possible azimuthal angles for fixed angles ν_i and ν_j:

$$P_{i,j} = \frac{1}{2\pi} \int_0^{2\pi} p\left(\nu_i \nu_j + \sqrt{1-\nu_i{}^2} \sqrt{1-\nu_j{}^2} \cos(\phi) \right) \mathrm{d}\phi \tag{11}$$

The choice between obtaining the forward scattering matrix $\mathbf{P^+}$ or the backward scattering matrix $\mathbf{P^-}$ depends on the signs of angles ν_i and ν_j. If both angles are either positive or negative, the forward scattering matrix $\mathbf{P^+}$ is obtained; otherwise, the backward scattering matrix $\mathbf{P^-}$ is obtained. Furthermore, it is required that the quadrature weighted sum over outgoing angles i for a given incoming angle j adds to 2:

$$\sum_{i=1}^{m} \left(P_{i,j}{}^{+} + P_{i,j}{}^{-} \right) w_i = 2 \tag{12}$$

Matrices $\mathbf{P^+}$ and $\mathbf{P^-}$ can be calculated analytically if we utilize the Hanyey–Greenstein scattering phase function (SPF), which is defined as:

$$p(\nu, g) = \frac{1}{2} \frac{1 - g^2}{(1 + g^2 + 2g\nu)^{\frac{3}{2}}} \tag{13}$$

where $-1 < g < 1$ is known as the asymmetry factor. The scattering matrix can be calculated from the following expression:

$$\begin{aligned} p(\nu, g) &= \tfrac{1-g^2}{\pi w_- \sqrt{w_+}} E_0\left(\tfrac{4gB}{w_+}\right) \\ w_\pm &= 1 + g^2 - 2g(A \pm B) \end{aligned} \tag{14}$$

where $A = \nu_i\, \nu_j$ and $B = \sqrt{1 - \nu_i{}^2}\sqrt{1 - \nu_j{}^2}$. and $E_0\,(x)$ denotes an elliptic integral of the second kind.

Alternatively, matrices $\mathbf{P^+}$ and $\mathbf{P^-}$ can be calculated through numerical integration:

$$P_{i,j}{}^{\pm} = \frac{1}{2\pi} \int_0^{2\pi} p(A + B\, \cos(z))\, \mathrm{d}z \tag{15}$$

with A and B as defined above. In this case, we employ the Gegenbauer scattering phase function (SPF), given by:

$$p(\nu, g, \alpha) = 2\alpha g\, \frac{\left(1 - g^2\right)^{2\alpha}}{(1 + g)^{2\alpha} - (1 - g)^{2\alpha}} \frac{1}{(1 + g^2 - 2g\nu)^{1+\alpha}} \tag{16}$$

It has been demonstrated by Engler [20] that when using the trapezoidal rule for numerical integration, an exponential convergence to the accurate scattering matrix elements can be achieved.

2.1.4. Initializing Starting Layer

A very thin layer must be initially simulated with optical thickness $\mathrm{d}\tau$, albedo a, and scattering matrices $\mathbf{P^+}$ and $\mathbf{P^-}$. The optical thickness is calculated as $\tau = (\mu_a + \mu_s) \cdot d$ and albedo as $a = \mu_s/(\mu_a + \mu_s)$, where d represents the thickness of the entire layer, and μ_s and μ_a are the scattering and absorption coefficients of the layer. Different approaches for initializing the layer exist, with diamond initialization being derived directly from the time-independent, one-dimensional, azimuthally averaged RTE; the details of derivation can be found elsewhere [15]. The initialization involves determining initial matrices using the following matrix relations:

$$\begin{aligned} \mathbf{c} &= \mathrm{diag}(w) \\ \mathbf{M} &= \mathrm{diag}\left(\tfrac{1}{\nu}\right) \\ \mathbf{A} &= \mathbf{M}\left(\mathbf{I} - \tfrac{a}{2}\mathbf{P^+c}\right)\tfrac{\mathrm{d}\tau}{2} \\ \mathbf{B} &= \tfrac{a}{2}\mathbf{M}\mathbf{P^-}\mathbf{c}\tfrac{d\tau}{2} \\ \mathbf{G} &= \left(\mathbf{I + A} - \mathbf{B}(\mathbf{I + A})^{-1}\mathbf{B}\right)^{-1} \\ \mathbf{R}_0 &= \mathbf{2GB}(\mathbf{I + A})^{-1} \\ \mathbf{T}_0 &= \mathbf{2G} - \mathbf{I} \end{aligned} \tag{17}$$

2.1.5. Constructing the Entire Layer

Once the reflectance and transmittance matrices of the initial thin layer are obtained, we can proceed to construct the entire layer. However, even before performing any calculations for the initial layer matrices, we must select an appropriate value for the initial

layer optical depth. In the literature, two criteria have been proposed for this selection: $d\tau < 0.01$ and $d\tau < \nu_{c-}$ For example, when using criterion $d\tau < 0.01$, we can calculate number of doubling steps required:

$$n_d = \text{floor}\left(\log\left(\frac{d}{0.01|\nu_1|}\right)/\log(2) + 1\right) \tag{18}$$

The initial optical depth is then calculated as $d\tau = \tau/(2)^n{}_d$. with n_d defined by Equation (18). To obtain the reflectance and transmittance matrices for the entire layer, the following iterative procedure is performed:

$$\begin{aligned} \mathbf{G}_k &= \left(\mathbf{I} - \mathbf{R}_k{}^2\right)^{-1} \\ \mathbf{R}_{k+1} &= \mathbf{T}_k\mathbf{G}_k\mathbf{R}_k\mathbf{T}_k + \mathbf{R}_k \\ \mathbf{T}_{k+1} &= \mathbf{T}_k\mathbf{G}_k\mathbf{T}_k \end{aligned} \tag{19}$$

where k is the number of steps from 0 to $n_d - 1$. The matrices calculated under Equation (17) obtain an index of 0. After that, with each doubling of the layer, the index increases by 1.

2.1.6. Calculating Reflectance and Transmittance

For normal irradiation, the resulting approximation for the reflectance and transmittance is given by:

$$\begin{aligned} R_n &= \int_0^1 R(1,\nu)\, 2\nu\, d\nu = \tfrac{m^2}{2} \sum_{i=1}^{m} 2\nu_i w_i R_{i,m} \\ T_n &= \int_0^1 T(1,\nu)\, 2\nu\, d\nu = \tfrac{m^2}{2} \sum_{i=1}^{m} 2\nu_i w_i T_{i,m} \end{aligned} \tag{20}$$

while for isotropic irradiation, it is expressed by:

$$\begin{aligned} R_t &= \int_0^1 2\nu\, d\nu \int_0^1 R(\nu,\nu')\, 2\nu'\, d\nu' = \left(\tfrac{m^2}{2}\right)^2 \sum_{j=1}^{m} 2\nu_j w_j \sum_{i=1}^{m} 2\nu_i w_i R_{i,j} \\ T_t &= \int_0^1 2\nu\, d\nu \int_0^1 T(\nu,\nu')\, 2\nu'\, d\nu' = \left(\tfrac{m^2}{2}\right)^2 \sum_{j=1}^{m} 2\nu_j w_j \sum_{i=1}^{m} 2\nu_i w_i T_{i,j} \end{aligned} \tag{21}$$

2.1.7. Adding Boundary Layers

Finally, boundary conditions are included as an additional layer when there is a refractive index mismatch between the layer and the surrounding medium. Since the boundary is flat, the Fresnel equations can be used:

$$\begin{aligned} r_s &= \left(\tfrac{n_1\nu_1 - n_2\nu_2}{n_1\nu_1 + n_2\nu_2}\right)^2 \\ r_p &= \left(\tfrac{n_1\nu_2 - n_2\nu_1}{n_1\nu_2 + n_2\nu_1}\right) \\ r &= \tfrac{r_s + r_p}{2} \\ t &= 1 - r \end{aligned} \tag{22}$$

As we are dealing with nonpolarized light, we simply average the reflectivities for the s and p polarizations. Before calculating the reflectance and transmittance values, it is necessary to determine the critical angle where total reflection starts:

$$\nu_c = \sqrt{1 - \left(\frac{n_1}{n_2}\right)^2} \tag{23}$$

For angles smaller than the critical angle, the cosines outside layer ν_1 are calculated from the cosines inside layer ν_2 using Snells's refraction law:

$$\nu_1 = \sqrt{\left(1 - \left(\frac{n_2}{n_1}\sqrt{1-\nu_2{}^2}\right)^2\right)} \tag{24}$$

The boundary matrices are then formulated as follows:

$$\begin{aligned} R^{01}_{i,j} &= \begin{cases} r(\nu_2)\delta_{i,j}, & \nu_2 > \nu_c \\ \delta_{i,j}, & \nu_2 \leq \nu_c \end{cases} \\ \mathbf{R}^{01} &= \mathbf{R}^{10} \\ \mathbf{T}^{01} &= \mathbf{1} - \mathbf{R}^{01} \\ \mathbf{T}^{10} &= \mathbf{T}^{01} \end{aligned} \tag{25}$$

The reflection $r(\nu_i)$ is the unpolarized Fresnel reflection function, while the Kronecker delta implies matrices $\mathbf{R}^{01}$, $\mathbf{R}^{10}$, $\mathbf{T}^{10}$, and $\mathbf{T}^{01}$ are diagonal, which means that light is specularly reflected at an angle equal to the incident angle [12]. Figure 2 depicts a boundary between medium 0 and 1 with different refraction indices.

Figure 2. Scheme of a boundary between medium 0 and 1 with different refraction indices. The reflectances R and transmittances T for both media are included.

The upper boundary is added to the layer, and the corresponding reflectance and transmittance matrices are calculated in the following way:

$$\begin{aligned} \boldsymbol{\gamma}_1 &= \left(\mathbf{I} - \mathbf{R}^{10}\mathbf{R}_t\right)^{-1} \\ \mathbf{T}^{02} &= \mathbf{T}_t\boldsymbol{\gamma}_1\mathbf{T}^{01} \\ \mathbf{R}^{20} &= \mathbf{T}_t\boldsymbol{\gamma}_1\mathbf{R}^{10}\mathbf{T}_t + \mathbf{R}_t \\ \boldsymbol{\gamma}_2 &= \left(\mathbf{I} - \mathbf{R}_t\mathbf{R}^{10}\right)^{-1} \\ \mathbf{T}^{20} &= \mathbf{T}^{10}\boldsymbol{\gamma}_2\mathbf{T}_t \\ \mathbf{R}^{02} &= \mathbf{T}^{10}\boldsymbol{\gamma}_2\mathbf{R}_t\mathbf{T}^{01} \end{aligned} \tag{26}$$

To incorporate the bottom boundary layer, different cases can be specified. In the first case, when the bottom layer is specular, the reflectance matrix $\mathbf{R}_L$ equals:

$$\mathbf{R}_L = \begin{bmatrix} R_0 & 0 & \dots & 0 \\ 0 & R_0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & R_0 \end{bmatrix} \tag{27}$$

where $\mathbf{R}_0$ stands for the surface reflectance. In this scenario, only the reflectance matrix at the sample surface is relevant, while the other matrices are 0.

In the second case, when the bottom layer is Lambertian, the reflectance matrix is calculated as follows:

$$\mathbf{R}_L = 2R_0 \begin{bmatrix} \nu_1 w_1 & \nu_2 w_2 & \cdots \\ \vdots & \ddots & \vdots \\ \nu_1 w_1 & \cdots & \nu_m w_m \end{bmatrix} \tag{28}$$

For the third case, when the bottom layer is assumed to be the same as the external medium, we can employ previously calculated reflectance and transmittance matrices from Equation (25). The reflectance matrices $\mathbf{R^{30}}$ and $\mathbf{R^{03}}$, as well as the transmittance matrices $\mathbf{T^{30}}$ and $\mathbf{T^{03}}$, are determined as follows:

$$\begin{aligned} \boldsymbol{\gamma}_1 &= \left(\mathbf{I} - \mathbf{R}^{20}\mathbf{R}^{23}\right)^{-1} \\ \mathbf{T}^{03} &= \mathbf{T}^{23}\boldsymbol{\gamma}_1\mathbf{T}^{02} \\ \mathbf{R}^{30} &= \mathbf{T}^{23}\boldsymbol{\gamma}_1\mathbf{R}^{20}\mathbf{T}^{32} + \mathbf{R}^{32} \\ \boldsymbol{\gamma}_2 &= \left(\mathbf{I} - \mathbf{R}^{23}\mathbf{R}^{20}\right)^{-1} \\ \mathbf{T}^{30} &= \mathbf{T}^{20}\boldsymbol{\gamma}_2\mathbf{T}^{32} \\ \mathbf{R}^{03} &= \mathbf{T}^{20}\boldsymbol{\gamma}_2\mathbf{R}^{23}\mathbf{T}^{02} + \mathbf{R}^{02} \end{aligned} \tag{29}$$

In the situation where the bottom layer is the same as the external medium, the boundary conditions are identical on the upper and bottom boundaries; thus, we have $\mathbf{R}^{01} = \mathbf{R}^{32}$, $\mathbf{R}^{10} = \mathbf{R}^{23}$, $\mathbf{T}^{01} = \mathbf{T}^{32}$, and $\mathbf{T}^{10} = \mathbf{T}^{23}$. If the bottom layer is specular or Lambertian, then $\mathbf{R}^{23} = \mathbf{R}_L$, and only matrix $\mathbf{R}^{03}$ is relevant, as all other matrices $\mathbf{R}^{30}$, $\mathbf{T}^{03}$, and $\mathbf{T}^{30}$ are 0. Once we have obtained all the reflectance and transmittance matrices denoted generically as $\mathbf{M}$, it is necessary to normalize them to ensure they are symmetrical:

$$M_{i,j} = \frac{M_{i,j}}{2\nu_j w_j} \tag{30}$$

After normalization, we can calculate the reflectance and transmittance for normal incidence:

$$\begin{aligned} R_n^{03} &= \sum_{i=1}^{m} 2\nu_i w_i R_{i,m}^{03} \\ T_n^{03} &= \sum_{i=1}^{m} 2\nu_i w_i T_{i,m}^{03} \end{aligned} \tag{31}$$

For isotropic irradiance, we can calculate corresponding values of reflectance and transmittance by following Equation (21). Finally, the specular reflectance is determined as follows:

$$R_n^{sp} = \sum_{i=1}^{m} 2\nu_i w_i R_{i,m}^{01} \tag{32}$$

2.2. Implementation of the Adding-Doubling (AD) Algorithm

A GPU-accelerated AD algorithm for single-layer tissue was developed using MATLAB's parallel toolbox. We utilized the library supporting batched functions, which enabled us to perform parallel calculations of several smaller matrices. Initialization of the CUDA library and running the algorithm on GPU required about 3 s; however, this initialization only needed to be performed once. Subsequent memory allocations would have posed an obstacle, potentially leading to time delays. To tackle this issue, we undertook a comprehensive rearrangement of the code to eliminate the need for for loops and enable the application of the code to larger arrays, specifically optimized for the operations with the previously mentioned library. In this way, we avoided the problem of performance degradation, which would have happened when calculating small matrices on GPU, as the matrix operation times would be comparable to data transfer times between GPU and

CPU. Figure 3 shows the flowchart of the CUDA implementation of the GPU-accelerated AD algorithm.

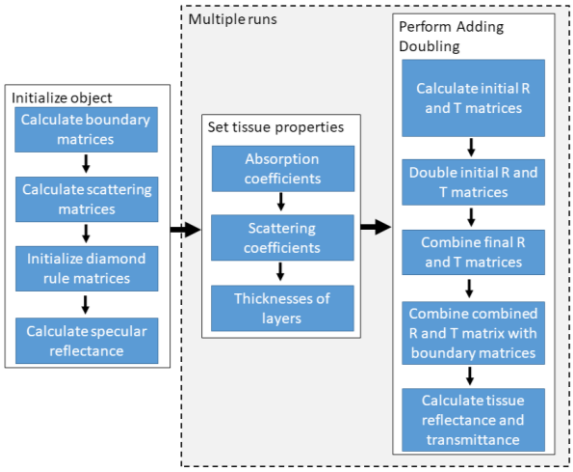

Figure 3. Flowchart of the proposed GPU-accelerated Adding-Doubling algorithm, which runs exclusively on a graphics processing unit (GPU). The first step, "Initialize object", a computationally intensive process, is executed only once during the algorithm initialization. The "Set tissue properties" and "Perform Adding Doubling" steps are run whenever there is a change in tissue properties. A detailed description of each algorithm step can be found in Prahl [12].

In the following, we will describe the individual parts of the AD1L class. The algorithm is implemented in a custom class AD1L. The class is initialized by calling the constructor **AD1L**

obj = AD1L(lam, [dtype]);

where *lam* represents an array of wavelengths with *Nl* rows. The *dtype* parameter is optional and specifies the precision used in the algorithm. There are two options: *single* and *double*. By default, the precision is set to *single* as it provides faster computation. An example of the class initialization with double precision is as follows:

A = AD1L(lam, 'dtype', 'double');

First, the algorithm is initialized by specifying the properties which remain constant throughout the computation. This includes calculating tissue matrices, such as scattering and reflectance/refraction matrices. The algorithm is initialized by calling the function **initPR** with the following syntax:

obj = obj.initPR(g, ['alpha', alpha], ['SPFtype', SPFtype], ['R0L',
R0L]['nE', nE], ['nS', nS], ['m', m], ['Nsam', Nsam])

where *g* is a matrix of tissue anisotropies at the specified wavelengths *lam* and samples; the parameters in [] are optional. These optional parameters include:

1. *alpha*—the Gegenbauer scattering phase function (SPF) exponent at the specified wavelengths *lam* for each sample. It is empty by default;
2. *SPFtype*—the type of SPF to be used. Currently implemented options are Henyey–Greenstein (HG) SPF and Gegenbauer (GB) SPF. To select HG SPF, the parameter value is 0, and to select GB SPF, the parameter value is 1. If GB SPF is selected, the *alpha* parameter must also be specified;
3. *R0L*—the reflectance of the bottom substrate, ranging between 0 and 1, if it is different from the upper environmental medium. This parameter is used in cases where a sample is placed on top of another thick substrate, such as a support plate. *R0L* is

a matrix with *Nl* rows corresponding to *lam* and *Ns* columns corresponding to the number of samples;

4. *nE*—a column vector of environmental medium refraction indices with *Nl* values corresponding to *lam*. By default, the value is set to 1, representing air;
5. *nS*—a column vector of substrate refraction indices with *Nl* values corresponding to *lam*. By default, the value is set to 1.4.+;
6. *m*—the number of fluxes (even integer >= 2) used in the algorithm. The default value is 20, as it has been found to provide adequate accuracy for most combinations of optical parameters;
7. *Nsam*—the number of different spectra being calculated. By default, it is set to 1, meaning that only one set of different wavelengths is considered.

An example of initializing the algorithm, providing *g* for Gegenbauer SPF and 20 fluxes, can be performed as follows:

```
A.initPR(g,'SPFtype',1,'alpha',alpha,'m',20);
```

where *g* is a column vector of *Nl* anisotropy factors, *alpha* is a column vector of *Nl* GB SPF exponents, and *m* is the number of fluxes.

The variable properties of the tissues are set by calling the **setOP** method. The calling syntax is

```
obj = obj.setOP(muA, muS, d)
```

where *muA* and *muS* represent the matrices of absorption and scattering coefficients in units of [1/cm], and *d* is the layer thickness in units of [cm]. The dimensions of *muA* and *muS* are *Nl* rows and *Ns* columns, while *d* has *Ns* values, one for each sample.

An example of calling this method is

```
A.setOP(muAar,muSar,d);
```

Finally, the **solveAD** is called to calculate the reflectance and transmittance values at the specified wavelengths. The calling syntax is

```
obj = obj.solveAD()
```

Although this method does not directly return any values, it stores the results in the class properties. Currently, only normal incidence of the illumination and complete diffuse reflectance and transmittance (e.g., all radiation in the exit hemispheres, two solid angles) are supported.

The results can be retrieved through the following properties:

- *obj.Rn*—2 diffuse reflectance;
- *obj.Tn*—2 transmittance;
- *obj.Rsp*—specular reflectance.

An example of retrieving all these parameters in a MATLAB environment is:

```
[A.Rn, A.Tn, A.Rsp]
```

2.3. Single-Layer AD Model Simulations

To evaluate the accuracy and computational speed of the AD1L algorithm, a comparison was made with layered MC results. The MC simulation was conducted using CUDA MCML software [21], which provided sufficient computational speed for the calculations. To achieve acceptable accuracy, 10^7 photons were simulated. The simulated single layer had fixed properties, including a thickness *d* of 1 cm, a sample refraction index *nS* of 1.4, and an environmental medium refraction index *nE* of 1, corresponding to air. The environmental medium was present on both sides of the layer.

To cover a range of optical properties similar to the SiliGlass tissue phantoms prepared in our lab, various absorption coefficients *muA* (0.1, 0.5, and 1.0/cm) and scattering coefficients *muS* (25, 50, 100, and 150/cm) were used. The computation was carried out using a single GeForce GTX TITAN X graphic card (Nvidia, Santa Clara, CA, USA).

2.4. Inverse Adding-Doubling (IAD) Solution

In addition to the forward problem, the GPU AD algorithm demonstrated utility in solving the inverse scattering problem. The human skin model used for this purpose consisted of two layers: epidermis and dermis (Figure 4). The epidermis and dermis are characterized by various optical parameters, which are discussed in more detail in Appendix A. The diamond initialization was used to simulate an initial thin skin layer, and the incident and reflected light were divided into 20 fluxes to ensure accurate results. To perform non-linear least-squares fitting, the Levenberg–Marquardt (LM) algorithm (Appendix B) was adopted on the GPU; the maximum number of iterations for the LM algorithm was limited to 200. A large batch of HSI reflectance spectra in the range of 430–750 nm with a step of 5 nm was fitted at once, enabling a rapid solution of the inverse scattering problem. The total number of fitting parameters was 11; however, only 6 were considered free variables: (a) volume fraction of melanin, f_m, (b) volume fraction of deoxyhemoglobin, f_{Hb}, (c) volume fraction of oxyhemoglobin, f_{HbO2}, (d) scattering factor a, where $a = \mu_s$ (λ = 500 nm), (e) scattering power, b, and (f) fraction of Rayleigh scattering, f_{Ray}. Among these free parameters, (a) and (d–f) were fitted for the epidermis, and (b–f) were fitted for the dermis. Several fixed parameters were also defined: (a) volume fraction of bilirubin, f_{brub}, was set to 0.01, (b) volume fraction of reduced cytochrome oxidase, f_{CO}, was fixed to 0.01; (c) volume fraction of oxidized cytochrome oxidase, f_{COO2}, was set to 0.01, (d) epidermis thickness, d_{epi}, was fixed at 0.2 mm, and (e) dermis thickness, d_{der}, was set to 1 cm. While the LM algorithm allowed the fitting of all parameters, the above parameters were fixed to prevent crosstalk between different sets of parameters and, therefore, improve the robustness of the inverse problem-solving process.

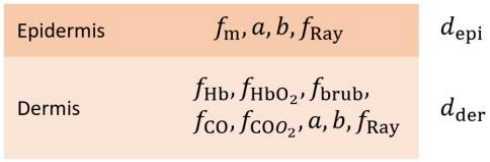

Figure 4. The two-layer skin model consists of the epidermis and dermis and corresponding optical parameters. f_m—volume fraction of melanin; a—scattering factor, where $a = \mu_s$ (λ = 500 nm); b—scattering power; f_{Ray}—fraction of Rayleigh scattering; f_{Hb}—volume fraction of deoxyhemoglobin; f_{HbO2}—volume fraction of oxyhemoglobin; f_{brub}—volume fraction of bilirubin; f_{CO}—volume fraction of reduced cytochrome oxidase; f_{COO2}—volume fraction of oxidized cytochrome oxidase; d_{epi}—epidermis thickness; d_{der}—dermis thickness.

To test the reliability and robustness of the GPU LM algorithm, a healthy participant's forearm was imaged with the HSI-integrated imaging system [22] before, during, and after applying a blood pressure cuff. Prior to cuff application, a steady-state image was acquired. The cuff was then inflated to about 200 mmHg to obstruct the blood flow completely, and the image was recorded three minutes after the initial inflation. Subsequently, the cuff was deflated, and one minute later, another image was captured. Hyperspectral images were acquired and processed as described in detail in our previous publications [22–24].

3. Results

3.1. Comparison between the Performance of GPU-Accelerated AD and CPU AD Algorithms

In the preliminary step, we compared GPU-accelerated AD and CPU AD algorithms. By repeatedly executing each algorithm 1000 times, we derived average times of solveAD as representative values, thereby minimizing the impact of external factors. For fixed values

of absorption coefficient *muA* = 0.1/cm, scattering coefficient *muS* = 25/cm, and a number of fluxes m = 20, we found that the GPU-accelerated AD algorithm was roughly seven times faster than the CPU AD algorithm (time per sample of 1.5×10^{-4} s vs. 1.1×10^{-3} s) (Figure 5). As shown in Figure 5, time per sample stabilized at around 1000 samples for GPU, while time per sample for CPU changed only marginally with the number of samples.

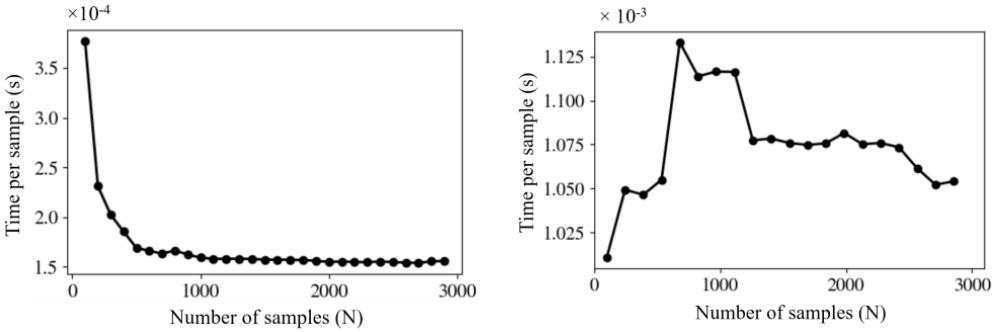

Figure 5. Dependence of time per sample on the number of samples, computed on the graphics processing unit (GPU) (**left**) and central processing unit (CPU) (**right**) at m = 20.

3.2. Comparison between Single and Double Precision for Single-Layer AD Model Simulations

We conducted a comparative analysis of reflectance and transmittance calculations using both single and double precision. The testing involved all possible combinations of absorption coefficients (μ_a = 0.1, 0.5, 1.0 cm^{-1}) and scattering coefficients (μ_s = 25, 50, 100, 150 cm^{-1}) at m = 20. The relative difference between the single and double precision results for reflectance ranged from 2×10^{-7} to 6×10^{-5}; the highest discrepancy occurred with the combination of the highest scattering coefficient and the smallest absorption coefficient. For transmittance, the differences between the single and double precision results were slightly higher, between 2×10^{-6} and 5×10^{-4}, the worst agreement corresponding again to the highest scattering and lowest absorption coefficients. However, these discrepancies are considerably smaller than the typical noise amplitude encountered in experimental spectra, which usually falls within the range of 1%.

In terms of computation speed, the single precision computations were, on average, 4.3 times faster compared to double precision when considering the above set of optical parameters. In addition to the faster computation, the advantage of using single precision is that it also has two times lower memory requirements, facilitating the upload of larger datasets to the GPU memory.

3.3. Single-Layer AD Model Simulations

We compared the AD algorithm with MC separately for the HG SPF and GB SPF. As the results were similar for both SPFs, we are presenting the results solely for the GB SPF with the exponent α_{GB} set at 1.051 and factor of anisotropy g at 0.851. Our evaluation focused on different numbers of fluxes, specifically m = 8, 12, 16, and 20.

Table 1 shows results for reflectance and transmittance, obtained using the CUDA MCML software, while Tables 2 and 3 exhibit results obtained using the AD1L algorithm with single precision. A comparison between MCML and AD1L reveals that perfect agreement was reached for m = 20, though adequate accuracy was obtained for m = 16.

To assess the computational speed of the GPU-accelerated AD1L algorithm, we considered different simultaneous numbers of threads: 100, 1000, and 3000. Table 4 presents the computation times for the MC method per single optical properties set, whereas Tables 5 and 6 present the corresponding results for the AD1L algorithm using the GB SPF for the number of examined fluxes m = 20 and m = 16, respectively. Comparing the MC method to the AD1L algorithm, we found that at m = 16 and a thread count of N = 3000, the speed-up ranged from 5333 to 40,556 times (Table 7). The computation times did not vary between the two SPFs since the number of elements in the array was the same. The optimal

number of threads for the AD1L algorithm was around 3000; larger batches of threads did not considerably enhance computational speed.

Table 1. Reflectance and transmittance for different absorption and scattering coefficients (in cm^{-1}) using the Monte Carlo method (CUDA MCML software). μ_a—absorption coefficient; μ_s—scattering coefficient.

	Reflectance				**Transmittance**			
μ_s \ μ_a	**25**	**50**	**100**	**150**	**25**	**50**	**100**	**150**
0.1	0.26	0.36	0.48	0.54	0.42	0.31	0.20	0.14
0.5	0.09	0.15	0.24	0.31	0.18	0.11	0.05	0.03
1.0	0.04	0.08	0.15	0.20	0.08	0.04	0.01	0.01

Table 2. Reflectance for different absorption and scattering coefficients (in cm^{-1}) using the AD method (AD1L algorithm) employing Gegenbauer scattering phase function; the value of exponent α_{GB} was set at 1.051 and factor of anisotropy g at 0.851; the number of examined fluxes m was 8, 12, 16, and 20. μ_a—absorption coefficient; μ_s—scattering coefficient. The values where the discrepancy with the Monte Carlo method in Table 1 is present are shown in grey.

	***m* = 8**				***m* = 12**			
μ_s \ μ_a	**25**	**50**	**100**	**150**	**25**	**50**	**100**	**150**
0.1	0.28	0.43	0.68	0.96	0.27	0.37	0.50	0.59
0.5	0.07	0.13	0.25	0.34	0.08	0.15	0.24	0.31
1.0	0.03	0.06	0.13	0.19	0.03	0.08	0.15	0.2
	***m* = 16**				***m* = 20**			
μ_a \ μ_s	**25**	**50**	**100**	**150**	**25**	**50**	**100**	**150**
0.1	0.26	0.36	0.48	0.55	0.26	0.36	0.48	0.54
0.5	0.09	0.15	0.24	0.31	0.09	0.15	0.24	0.31
1.0	0.04	0.08	0.15	0.20	0.04	0.08	0.15	0.20

Table 3. Transmittance for different absorption and scattering coefficients (in cm^{-1}) using the AD method (AD1L algorithm) employing Gegenbauer scattering phase function; the value of exponent α_{GB} was set at 1.051 and factor of anisotropy g at 0.851; the number of examined fluxes m was 8, 12, 16, and 20. μ_a—absorption coefficient; μ_s—scattering coefficient. The values where the discrepancy with the Monte Carlo method in Table 1 is present are shown in grey.

	***m* = 8**				***m* = 12**			
μ_s \ μ_a	**25**	**50**	**100**	**150**	**25**	**50**	**100**	**150**
0.1	0.59	0.50	0.47	0.55	0.45	0.34	0.23	0.17
0.5	0.28	0.17	0.10	0.07	0.20	0.11	0.05	0.03
1.0	0.14	0.07	0.03	0.02	0.09	0.04	0.02	0.01
	***m* = 16**				***m* = 20**			
μ_a \ μ_s	**25**	**50**	**100**	**150**	**25**	**50**	**100**	**150**
0.1	0.42	0.32	0.20	0.14	0.42	0.31	0.20	0.14
0.5	0.18	0.11	0.05	0.03	0.18	0.11	0.05	0.03
1.0	0.09	0.04	0.01	0.01	0.08	0.04	0.01	0.01

Table 4. Values of times (**in seconds**) per single optical properties set of the Monte Carlo (MC) simulation for different absorption and scattering coefficients (in cm^{-1}); μ_a—absorption coefficient; μ_s—scattering coefficient.

μ_a \ μ_s	**25**	**50**	**100**	**150**
0.1	1.5	1.7	3.1	7.3
0.5	0.8	1.5	2.6	3.4
1.0	0.8	1.3	2.0	2.6

Table 5. Average values of times (**in milliseconds**) per single optical properties set of the AD method (AD1L algorithm) employing Gegenbauer scattering phase function for different absorption and scattering coefficients (in cm^{-1}) and three different simultaneous number of threads; **the number of examined fluxes *m* was 20**; the value of exponent α_{GB} was set at 1.051 and factor of anisotropy g at 0.851. μ_a—absorption coefficient; μ_s—scattering coefficient.

	μ_a \ μ_s	**25**	**50**	**100**	**150**
Time per set (ms), N = 100	**0.1**	1.1	1.1	1.2	1.2
	0.5	1.1	1.1	1.2	1.2
	1.0	1.1	1.1	1.2	1.2
	μ_a \ μ_s	**25**	**50**	**100**	**150**
Time per set (ms), N = 1000	**0.1**	0.30	0.31	0.33	0.34
	0.5	0.29	0.31	0.32	0.34
	1.0	0.30	0.31	0.31	0.34
	μ_a \ μ_s	**25**	**50**	**100**	**150**
Time per set (ms), N = 3000	**0.1**	0.24	0.25	0.27	0.29
	0.5	0.24	0.25	0.27	0.29
	1.0	0.23	0.25	0.27	0.29

Table 6. Average values of times (**in milliseconds**) per single optical properties set of the AD method (AD1L algorithm) employing Gegenbauer scattering phase function for different absorption and scattering coefficients (in cm^{-1}) and three different simultaneous number of threads; **the number of examined fluxes *m* was 16**; the value of exponent α_{GB} was set at 1.051 and factor of anisotropy g at 0.851. μ_a—absorption coefficient; μ_s—scattering coefficient.

	μ_a \ μ_s	**25**	**50**	**100**	**150**
Time per set (ms), N = 100	**0.1**	0.87	0.89	0.96	1.60
	0.5	0.86	0.90	1.30	1.40
	1.0	0.86	0.90	1.30	1.20
	μ_a \ μ_s	**25**	**50**	**100**	**150**
Time per set (ms), N = 1000	**0.1**	0.21	0.22	0.23	0.24
	0.5	0.21	0.22	0.24	0.24
	1.0	0.21	0.22	0.23	0.24
	μ_a \ μ_s	**25**	**50**	**100**	**150**
Time per set (ms), N = 3000	**0.1**	0.15	0.16	0.17	0.18
	0.5	0.15	0.16	0.17	0.18
	1.0	0.15	0.16	0.17	0.18

Table 7. Computational speedup when comparing the Monte Carlo (MC) method (Table 4) and the AD1L algorithm using the GB SPF for the number of examined fluxes m = 16 and N = 3000 (Table 6). μ_a—absorption coefficient; μ_s—scattering coefficient.

μ_a \ μ_s	25	50	100	150
0.1	10,000	10,625	18,235	40,556
0.5	5333	9375	15,294	18,889
1.0	5333	8125	11,765	14,444

3.4. Inverse Adding-Doubling (IAD) Solution: Arterial Occlusion Test

Results obtained with a GPU-accelerated LM algorithm are shown alongside measured data in Figure 6. Here, reflectance spectra of skin before, during, and after a cuff test are presented for a single spatial data point in a curvature-corrected hyperspectral image [25]. Upon examining the calculated spectra, we observed excellent agreement with the measured data across the entire spectral range of 430–750 nm. However, minor deviations can be spotted near 700 nm due to slight wiggles in the measured spectra. To assess the goodness of fit, we calculated the reduced χ^2 value for each spatial data point; the computed data yielded reduced χ^2 ranging from 10^{-5}–10^{-4}. In terms of computational efficiency, it took approximately 5 min to process a 220 × 100 × 61 image (x-axis × y-axis × spectral-axis).

Figure 7 shows colormaps extracted from the hyperspectral images by the inverse AD algorithm, depicting tissue oxygenation, calculated as the ratio of f_{HbO2} to the sum of f_{Hb} and f_{HbO2} multiplied by 100%, and the total hemoglobin concentration, calculated as a sum of f_{Hb} and f_{HbO2} multiplied by 100%. These colormaps capture the conditions before, during, and after a cuff test. Upon analyzing Figure 7a, we observe that tissue oxygenation is visibly reduced during the cuff test, coinciding with the obstruction of blood flow when a blood-pressure cuff is inflated to around 200 mmHg. Additionally, we can discern from Figure 7b a slight increase in the total hemoglobin concentration throughout the entire experiment. Conversely, tissue oxygenation reaches its peak approximately 3 min after deflating the blood-pressure cuff, indicating rapid tissue reoxygenation following the vascular obstruction.

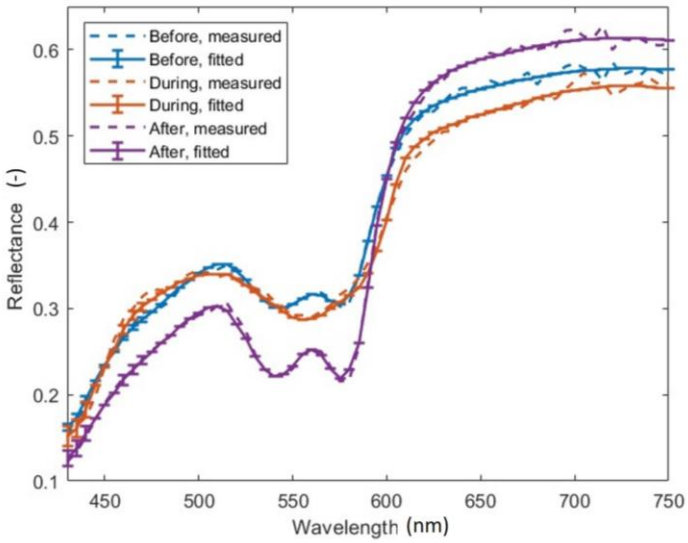

Figure 6. Measured (dashed lines) and fitted (solid lines with error bars) reflectance spectra of skin before, during, and after performing a cuff test.

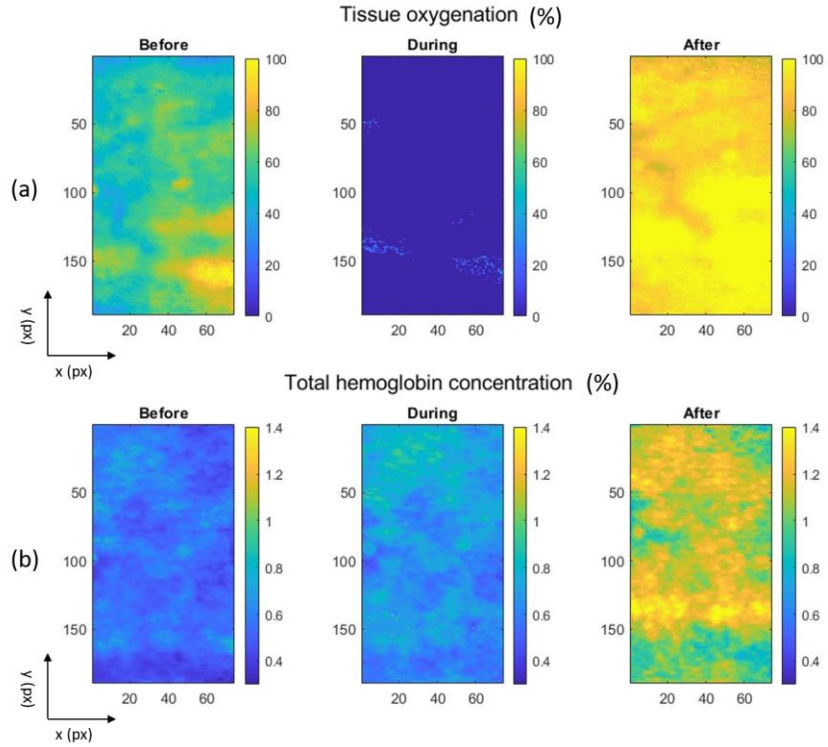

Figure 7. Colormaps of (**a**) tissue oxygenation and (**b**) the total hemoglobin concentration before, during, and after a cuff test.

To quantify our observations from the colormaps, we present mean values and standard deviations of tissue oxygenation and the total hemoglobin concentration before, during, and after the cuff test in Table 8. These numerical findings substantiate the conclusions drawn from the visual examination of the colormaps in Figure 5 and also corroborate data reported by Verdel et al. [26].

Table 8. Calculated mean values of tissue oxygenation and the total hemoglobin concentration before, during, and after the cuff test.

	Before	**During**	**After**
Tissue oxygenation (%)	58.57 ± 10.83	0.24 ± 2.11	92.91 ± 6.02
Total hemoglobin (%)	0.54 ± 0.07	0.68 ± 0.09	1.11 ± 0.12

4. Discussion

Our study has demonstrated that the AD algorithm matches the accuracy of the MC method while also being versatile and compatible with implementation on GPUs. The computational speed-up achieved by the GPU-accelerated AD algorithm when compared to the GPU MC method, ranges from about 5000 to 40,000 times faster. This remarkable improvement in computational efficiency allows for the entire hyperspectral image to be analyzed within minutes. While the MC method excels in capturing intricate photon interactions, its computational demands are often prohibitive for large-scale simulations. In contrast, the AD algorithm serves as a more efficient alternative, offering an analytical solution that strikes a balance between accuracy and computational efficiency.

Based on our study results, we can conclude that a GPU-accelerated two-layer IAD provides an acceptably accurate, fast, and reliable approach for solving inverse scattering problems and extracting important tissue properties, such as scattering properties and chromophore concentrations. Our calculated skin reflectance values align closely with the measurements taken before, during, and after the cuff test. Additionally, the obtained tissue properties using the IAD approach effectively reflect the changes in tissue physiology.

In recent years, the use of GPUs has advanced computational efficiency and accelerated processing capabilities. In our study, we specifically focused on implementing the AD algorithm on GPUs to enhance computational speed further. Additionally, we combined the AD and LM algorithms on the GPU platform to extract object parameters from optical spectral data effectively. By harnessing the power of GPUs, we aimed to achieve improvements in the computational speed of AD calculations. This integration of GPU technology with the AD algorithm holds the potential to advance our understanding of light transport in biological tissues and can pave the way for more efficient simulations and clinical applications, either in HSI [27] or related biomedical optical applications [28,29].

In our previous publication, we reported a GPU implementation of the diffuse approximation algorithm for analyzing hyperspectral images of skin [30]. Although this algorithm proved to be sufficiently fast in providing extracted tissue parameter maps in (almost) real-time, its primary limitation is that it is an approximation only, and therefore, the resulting maps differ from the true values. On the other hand, the AD algorithm is as accurate as the MC algorithm; thus, the extracted tissue properties correspond to the true values, as demonstrated in our recent publication [31]. Although real-time image processing has not been achieved yet, we believe that by reducing the number of the analyzed wavelengths, further optimizing the algorithm, and implementing it in more computationally efficient programming languages, we can attain real-time image analysis. This advancement would make HSI even more attractive for clinical applications.

Our approach has an obvious limitation in that it operates under assumptions outlined by Prahl et al. [13], including the absence of time dependence, a geometric model comprising uniform layers of finite thickness extended infinitely parallel to the surface, uniform scattering and absorbing properties within tissue layers, and uniform illumination by collimated or diffuse light. While these assumptions do not contradict the application of GPU AD to HSI, it may be necessary to resort to the MC method in more complex cases when, for example, a beam would be spreading from a finite source [32].

An obvious extension of our AD approach involves transitioning from MATLAB to Python. We have already begun exploring this direction, and preliminary results indicate an additional computational speed-up ranging from a factor of 3 to 10 when comparing GPU AD (Python) to GPU AD (MATLAB). Furthermore, the GPU AD method itself has the potential to accelerate machine learning techniques (ML) in spectral image analysis by providing data for the training process. In contrast to the MC algorithm, the AD algorithm can generate large training datasets in significantly shorter times, facilitating faster optimizations of ML tools.

Through thorough development and testing, we have successfully implemented the GPU-based AD algorithm, which offers an analytical solution for the RTE. Our research has demonstrated the GPU AD algorithm's significant speedup over the GPU MC method while maintaining a high level of accuracy. We have also shown that the GPU AD algorithm can be used to extract tissue parameters from optical spectral data of human skin under various hemodynamic conditions using the Levenberg–Marquardt inverse algorithm. The GPU AD algorithm can process large spectral images within minutes, surpassing the performance of other RTE solutions. Therefore, it has the potential for real-time processing of spectral imaging in the clinical environment.

Author Contributions: Conceptualization, R.H. and M.M.; methodology, R.H. and M.M.; software, M.M.; validation, M.M.; formal analysis, R.H. and M.M.; investigation, R.H.; resources, M.M.; data curation, M.M.; writing—original draft preparation, R.H.; writing—review and editing, R.H. and M.M.; visualization, M.M.; supervision, R.H. and M.M.; project administration, R.H. and M.M.; funding acquisition, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Slovenian Research and Innovation Agency, grant numbers P1-0389 and J3-3083.

Institutional Review Board Statement: The study was conducted in accordance with the Declaration of Helsinki and approved by the Medical Ethics Committee of the Republic of Slovenia (protocol code 66/01/17 and date of approval 25 May 2017).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Data and the MATLAB implementation of the algorithms are available upon reasonable request to the corresponding author.

Acknowledgments: The authors wish to thank Anej Sterle for running the simulations.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Optical Parameters of Epidermis and Dermis

Skin, the largest organ in the human body, covers a total surface area of approximately 1.6–1.8 m^2 and weighs around 8% of the total body weight in adults [33]. It consists of two main layers—the epidermis and dermis, which rest on the subcutaneous fat layer, called the subcutis; the epidermis (approximately 100–150 µm) is thinner than the dermis (approximately 2–4 mm), which contains structural proteins such as collagen and elastin [33]. When exposed to ultraviolet (UV), visible (VIS), and near-infrared (NIR) radiation, human skin exhibits distinct photobiological responses, which strongly depend on, among others, melanin, oxygenated and deoxygenated blood particles, and water.

In the model of the epidermis and the dermis, presented in Figure 4, we considered the following six optical parameters as free variables, which we briefly outline for the epidermis and the dermis separately.

Appendix A.1. Epidermis

The volume fraction of melanin f_m measures the concentration of melanin pigment in the epidermis. Measuring f_m can provide insights into various skin conditions, such as hyperpigmentation, hypopigmentation, and melanoma.

Appendix A.2. Epidermis and Dermis

Scattering factor $a = \mu_s$ (λ = 500 nm) quantifies the scattering properties of tissue at a specific wavelength, usually 500 nm. It measures how light scatters within tissue due to interactions with cellular and subcellular structures.

Scattering power b is a parameter that characterizes the scattering intensity of the tissue. It provides insights into the size and density of scattering particles within the tissue, affecting the degree of light dispersion.

Fraction of Rayleigh scattering f_{Ray} refers to the scattering caused by small molecules or particles compared to other forms of scattering. It is significant in studying tissue composition and structure, as it provides information about the molecular and chemical composition of the tissue.

Appendix A.3. Dermis

The volume fraction of deoxyhemoglobin f_{Hb} signifies the concentration of oxygen-poor hemoglobin in tissue and serves as an indicator of tissue oxygenation. Monitoring changes in f_{Hb} is valuable for studying conditions like hypoxia, ischemia, and vascular disorders.

The volume fraction of oxyhemoglobin f_{HbO2} represents the concentration of oxygen-rich hemoglobin in tissue and provides information about tissue oxygenation. Monitoring changes in f_{HbO2} is particularly relevant in studying tissue response to interventions, such as in oncology treatments or surgical procedures.

Appendix A.4. Other Parameters

The following five other parameters (four in the dermis and one in the epidermis) were treated as fixed variables: f_{brub}—volume fraction of bilirubin; f_{CO}—volume fraction of reduced cytochrome oxidase; f_{COO2}—volume fraction of oxidized cytochrome oxidase;

d_{epi}—epidermis thickness; d_{der}—dermis thickness. The volume fraction of bilirubin f_{brub} measures the concentration of bilirubin, a yellow pigment derived from the breakdown of heme in red blood cells; elevated levels of bilirubin can indicate liver dysfunction. An increase in the volume fraction of reduced cytochrome oxidase f_{CO} cytochrome oxidase generally indicates a higher level of cellular respiration and oxygen consumption, while the presence of oxidized cytochrome oxidase f_{COO2} usually indicates a lower metabolic activity and reduced oxygen consumption.

Appendix B. Levenberg–Marquardt Algorithm

The Levenberg–Marquardt (LM) algorithm is a well-known tool for solving nonlinear least squares problems [34–37]. LM has been implemented as a MATLAB Class for multithread GPU fitting of spectral images. Matlab Class consists of multiple properties and methods, which are described below.

Appendix B.1. Initialization

The class is initialized by calling the class constructor:

LM = Fit_X_(lam,img,[dtype]);

where _X_ represents the tissue light transport method to be fitted. The necessary parameters are:

- *lam*—vector of wavelengths (***Nl*** × ***1***);
- *img*—spectral image (***Nx*** × ***Ny*** × ***Nl***).

Where ***Nl*** is the number of wavelengths, and ***Nx*** and ***Ny*** are the numbers of image pixels in x- and y-directions.

The optional parameter is as follows:

- *dtype*—the datatype used for the computation. It can be either 'single' for the single precision or 'double' for the double precision.

The initialization stores image data and wavelengths in the class properties.

Appendix B.2. Algorithm Setup

The algorithm is set by calling the InitLM method; it is the main function where all parameters of the algorithms are set:

LM.InitLM(pimg0,[options])

The necessary parameters are as follows:
*pimg*0—the initial parameters for the model function (***Nx*** × ***Ny*** × ***Npar***).
Optional parameters are as follows:

- *dp*—fractional increment of parameters *p* for numerical derivatives (***Npar*** × ***1***). Default: 0.005. Additional info:
  1. d*p(j)* > 0 central differences calculated;
  2. d*p(j)* < 0 one-sided 'backwards' differences calculated;
  3. d*p(j)* = 0 sets corresponding partials to zero, i.e., holds *p(j)* fixed;
- *p_min*—lower bounds for parameter values (***Npar*** × ***1***). Default values depend on the tissue type;
- *p_max*—upper bounds for parameter values (***Npar*** × ***1***). Default values depend on the tissue type;
- *weight*—weight matrices (***Nl*** × ***Nl***) or a scalar weight value; weight must be non-negative. For more information about weights, see [36]. Default value: 1;
- *eps1*—convergence in the gradient. Default value: 1×10^{-4};
- *eps2*—convergence in parameters. Default value: 1×10^{-4};

- *eps3*—convergence in $\boldsymbol{\chi^2}$. Default value: 1×10^{-4};
- *eps4*—$\boldsymbol{\lambda_{lm}}$ update criterion. Default value: 1×10^{-3};
- *MaxIter*—maximum number of iterations. Default value: 200;
- *L0*—initial lambda value. Default value: 1×10^{-3};
- *Lu*—lambda increase multiplicator. Default value: 1.5;
- *Ld*—lambda decrease divisor. Default value: 5;
- *Update_Type* LM lambda update method. Parameter values: 1 = Levenberg–Marquardt lambda update (default), 2 = Quadratic update, 3 = Nielsen's lambda update equations, 4 = geodesic acceleration;
- *epsJtWJ*—the minimum allowed value of the Levenberg-Marquardt damping term (Update_Type = 1). Default value: 1×10^{-3};
- *dg*—geodesic acceleration FD step. Default value: 0.1;
- *galpha*—geodesic acceleration acceptance criterion. Default value: 0.75;
- *Broyden*—Broyden's rank 1 update for Jacobian, preventing the calculation of finite differences, 1 = use, 0 = do not use rank 1 approximation. Default value: 1;
- *lambdaMax*—maximum dumping parameter lambda. Default value: 10;
- *muPenTer*—penalty term for the nearest neighbor's initial coefficient. Set to 0 if no term penalty is required. Default value: 1;
- *Nsit*—number of spectra fitted simultaneously. Default value: 10;
- *Verbose*—Display messages during execution. Parameter values: 0 = no messages, 1 = messages. Default value: 1

The constraint (i.e., muPenTer) takes care to penalize individual parameter values that are significantly different from the parameter values in the neighbor spatial points.

Appendix B.3. Running

The optimization is started by calling the RunFit method:

LM.RunFit;

The method starts an iterative process of finding the optimal parameters, minimizing the input data—model values. The final results are stored in the class properties.

Appendix B.4. Output

The output variables are stored in the class properties and can be accessed directly by calling:

LM.[variable]

The output variables are:

- *pimgOut*—fitted parameters ($\boldsymbol{Nx \times Ny \times Npar}$);
- *imgOut*—fitted image ($\boldsymbol{Nx \times Ny \times Nl}$);
- *redX2*—reduced $\boldsymbol{\chi^2}$ ($\boldsymbol{Nx \times Ny \times 1}$);
- *sigma_p*—asymptotic standard parameter errors ($\boldsymbol{Nx \times Ny \times Npar}$);
- *sigma_y*—standard error of the fit ($\boldsymbol{Nx \times Ny \times Nl}$);
- *corr_p*—parameter correlation matrix ($\boldsymbol{Nx \times Ny \times Npar \times Npar}$);
- *R_sq*—coefficient of multiple determination ($\boldsymbol{Nx \times Ny \times 1}$);
- *iterOut*—matrix of iterations needed in each image point to stop fitting.

References

1. Stokes, G.G., IV. On the intensity of the light reflected from or transmitted through a pile of plates. *Proc. R. Soc. Lond.* **1862**, *11*, 545–556. [CrossRef]
2. van de Hulst, H.C.; Christoffel, H. *A New Look at Multiple Scattering*; NASA Institute for Space Studies: New York, NY, USA, 1962.
3. Liu, Q.; Weng, F. Advanced Doubling–Adding Method for Radiative Transfer in Planetary Atmospheres. *J. Atmos. Sci.* **2006**, *63*, 3459–3465. [CrossRef]

4. Zhang, Z.; Yang, P.; Kattawar, G.; Huang, H.-L.; Greenwald, T.; Li, J.; Baum, B.A.; Zhou, D.K.; Hu, Y. A fast infrared radiative transfer model based on the adding–doubling method for hyperspectral remote-sensing applications. *J. Quant. Spectrosc. Radiat. Transf.* **2007**, *105*, 243–263. [CrossRef]
5. Mukai, S.; Sano, I.; Nakata, M. Improved Algorithms for Remote Sensing-Based Aerosol Retrieval during Extreme Biomass Burning Events. *Atmosphere* **2021**, *12*, 403. [CrossRef]
6. Leyre, S.; Cappelle, J.; Durinck, G.; Abass, A.; Hofkens, J.; Deconinck, G.; Hanselaer, P. The use of the adding-doubling method for the optical optimization of planar luminescent down shifting layers for solar cells. *Opt. Express* **2014**, *22*, A765. [CrossRef] [PubMed]
7. Pickering, J.W.; Prahl, S.A.; Van Wieringen, N.; Beek, J.F.; Sterenborg, H.J.C.M.; Van Gemert, M.J.C. Double-integrating-sphere system for measuring the optical properties of tissue. *Appl. Opt.* **1993**, *32*, 399. [CrossRef] [PubMed]
8. Lemaillet, P.; Cooksey, C.C.; Hwang, J.; Wabnitz, H.; Grosenick, D.; Yang, L.; Allen, D.W. Correction of an adding-doubling inversion algorithm for the measurement of the optical parameters of turbid media. *Biomed. Opt. Express* **2018**, *9*, 55. [CrossRef] [PubMed]
9. Sun, B.; Gao, C.; Spurr, R. Scalar thermal radiation using the adding-doubling method. *Opt. Express* **2022**, *30*, 30075. [CrossRef] [PubMed]
10. Liu, X.; Wu, Y. Monte-Carlo optical model coupled with Inverse Adding-Doubling for Building Integrated Photovoltaic smart window design and characterisation. *Sol. Energy Mater. Sol. Cells* **2021**, *223*, 110972. [CrossRef]
11. Calvin, O.W.; Ganapol, B.D.; Borrelli, R.A. Introduction of the Adding and Doubling Method for Solving Bateman Equations for Nuclear Fuel Depletion. *Nucl. Sci. Eng.* **2023**, *197*, 558–588. [CrossRef]
12. Prahl, S.A. The Adding-Doubling Method. In *Optical-Thermal Response of Laser-Irradiated Tissue*; Welch, A.J., Van Gemert, M.J.C., Eds.; Springer: Boston, MA, USA, 1995; pp. 101–129. [CrossRef]
13. Prahl, S.A.; Van Gemert, M.J.C.; Welch, A.J. Determining the optical properties of turbid media by using the adding–doubling method. *Appl. Opt.* **1993**, *32*, 559. [CrossRef]
14. Prahl, S.A. A Monte Carlo model of light propagation in tissue. In Proceedings of the Institutes for Advanced Optical Technologies, Berlin, Germany, 10 January 1989; Mueller, G.J., Sliney, D.H., Potter, R.F., Eds.; p. 1030509. [CrossRef]
15. Wang, C.-Y.; Kao, T.-C.; Chen, Y.-F.; Su, W.-W.; Shen, H.-J.; Sung, K.-B. Validation of an Inverse Fitting Method of Diffuse Reflectance Spectroscopy to Quantify Multi-Layered Skin Optical Properties. *Photonics* **2019**, *6*, 61. [CrossRef]
16. Després, P.; Jia, X. A review of GPU-based medical image reconstruction. *Phys. Medica* **2017**, *42*, 76–92. [CrossRef] [PubMed]
17. Kalaiselvi, T.; Sriramakrishnan, P.; Somasundaram, K. Survey of using GPU CUDA programming model in medical image analysis. *Inform. Med. Unlocked* **2017**, *9*, 133–144. [CrossRef]
18. Smistad, E.; Falch, T.L.; Bozorgi, M.; Elster, A.C.; Lindseth, F. Medical image segmentation on GPUs—A comprehensive review. *Med. Image Anal.* **2015**, *20*, 1–18. [CrossRef]
19. Alcaín, E.; Fernández, P.R.; Nieto, R.; Montemayor, A.S.; Vilas, J.; Galiana-Bordera, A.; Martinez-Girones, P.M.; Prieto-De-La-Lastra, C.; Rodriguez-Vila, B.; Bonet, M.; et al. Hardware Architectures for Real-Time Medical Imaging. *Electronics* **2021**, *10*, 3118. [CrossRef]
20. Engler, H. Computation of Scattering Kernels in Radiative Transfer. *arxiv* **2015**. [CrossRef]
21. Alerstam, E.; Svensson, T.; Andersson-Engels, S. Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration. *J. Biomed. Opt.* **2008**, *13*, 060504. [CrossRef]
22. Stergar, J.; Hren, R.; Milanič, M. Design and Validation of a Custom-Made Laboratory Hyperspectral Imaging System for Biomedical Applications Using a Broadband LED Light Source. *Sensors* **2022**, *22*, 6274. [CrossRef]
23. Rogelj, L.; Simončič, U.; Tomanič, T.; Jezeršek, M.; Pavlovčič, U.; Stergar, J.; Milanič, M. Effect of curvature correction on parameters extracted from hyperspectral images. *J. Biomed. Opt.* **2021**, *26*, 096003. [CrossRef]
24. Rogelj, L.; Dolenec, R.; Tomšič, M.V.; Laistler, E.; Simončič, U.; Milanič, M.; Hren, R. Anatomically Accurate, High-Resolution Modeling of the Human Index Finger Using In Vivo Magnetic Resonance Imaging. *Tomography* **2022**, *8*, 2347–2359. [CrossRef] [PubMed]
25. Rogelj, L.; Pavlovčič, U.; Stergar, J.; Jezeršek, M.; Simončič, U.; Milanič, M. Curvature and height corrections of hyperspectral images using built-in 3D laser profilometry. *Appl. Opt.* **2019**, *58*, 9002. [CrossRef]
26. Verdel, N.; Marin, A.; Milanič, M.; Majaron, B. Physiological and structural characterization of human skin in vivo using combined photothermal radiometry and diffuse reflectance spectroscopy. *Biomed. Opt. Express* **2019**, *10*, 944. [CrossRef] [PubMed]
27. Hren, R.; Sersa, G.; Simoncic, U.; Milanic, M. Imaging perfusion changes in oncological clinical applications by hyperspectral imaging: A literature review. *Radiol. Oncol.* **2022**, *56*, 420–429. [CrossRef] [PubMed]
28. Hren, R.; Sersa, G.; Simoncic, U.; Milanic, M. Imaging microvascular changes in nonocular oncological clinical applications by optical coherence tomography angiography: A literature review. *Radiol. Oncol.* **2023**, *57*, 411–418. [CrossRef] [PubMed]
29. Marin, A.; Hren, R.; Milanič, M. Pulsed Photothermal Radiometric Depth Profiling of Bruises by 532 nm and 1064 nm Lasers. *Sensors* **2023**, *23*, 2196. [CrossRef] [PubMed]
30. Bjorgan, A.; Milanic, M.; Randeberg, L.L. Estimation of skin optical parameters for real-time hyperspectral imaging applications. *J. Biomed. Opt.* **2014**, *19*, 066003. [CrossRef] [PubMed]
31. Tomanič, T.; Rogelj, L.; Milanič, M. Robustness of diffuse reflectance spectra analysis by inverse adding doubling algorithm. *Biomed. Opt. Express* **2022**, *13*, 921. [CrossRef]

32. Klanecek, Z.; Hren, R.; Simončič, U.; Muc, B.T.; Lukač, M.; Milanič, M. Finite Element Method (FEM) Modeling of Laser-Tissue Interaction during Hair Removal. *Appl. Sci.* **2023**, *13*, 8553. [CrossRef]
33. Young, A.R. Chromophores in human skin. *Phys. Med. Biol.* **1997**, *42*, 789–802. [CrossRef]
34. Hren, R.; Stroink, G. Application of the surface harmonic expansions for modeling the human torso. *IEEE Trans. Biomed. Eng.* **1995**, *42*, 521–524. [CrossRef]
35. Hren, R.; Zhang, X.; Stroink, G. Comparison between electrocardiographic and magnetocardiographic inverse solutions using the boundary element method. *Med. Biol. Eng. Comput.* **1996**, *34*, 110–114. [CrossRef] [PubMed]
36. Gavin, H.P. The Levenberg-Marquardt Algorithm for Nonlinear Least Squares Curve-Fitting Problems. 27 November 2022. Available online: https://people.duke.edu/~hpgavin/ExperimentalSystems/lm.pdf (accessed on 3 February 2024).
37. Du, Y.-C.; Stephanus, A. Levenberg-Marquardt Neural Network Algorithm for Degree of Arteriovenous Fistula Stenosis Classification Using a Dual Optical Photoplethysmography Sensor. *Sensors* **2018**, *18*, 2322. [CrossRef] [PubMed]