Numerical Diffusion Modelling of Light Propagation in Turbid Media for Medical Diagnostics

Diploma paper
by
Charlotta Lindquist

Lund Reports on Atomic Physics, LRAP-157
Lund, March 1994

Abstract

A computer code that simulates light transport through a biological tissue slab by solving the diffusion equation numerically has been used. The sensitivity of the method to changes in boundary-values, model dimensions etc. was investigated. To the code was added the possibility to simulate experiments with an in-phase/anti-phase laser array and to study the Fourier transform of the resulting time dispersion curve. The sources were modulated at a frequency of 200 MHz. The diffusing waves originating from the out-of-phase sources give an amplitude null and a sharp phase transition in the mid-plane. Intensity and phase data were acquired as a highly absorbing rod was translated inside the medium, and the results were compared with experimental results.

Contents

1. Introduction

This is a Diploma work made at the Department of Physics at the Lund Institute of Technology. It is a part of a project of the medical laser physics group at the Atomic Physics Division, developing a method for early tumour detection using optical transillumination of tissue.

Since the development of laser light sources the use of light in therapeutic and diagnostic medicine has increased rapidly. This has created a need to understand the interaction of light with biological media. A field of considerable interest is finding a method to use optical radiation instead of X-rays to study living tissue properties. One area is aiming at finding an alternative to X-ray mammography. Breast cancer is the most common cancer disease among women today, concerning about 25% of the total number of female cancer patients. Early discovery of the tumours is of vital importance to reduce the mortality. However, with X-ray mammography, which is the standard breast diagnosis method, it is not always possible to see small tumours. Furthermore there might be a risk of inducing cancer, especially in young women, since ionising radiation is used. It has been found the last few years that 1% of the population are heterozygotes for the so called AT-gene (Ataxia-Telangiectasia)[1,2]. These persons are unusually sensitive to ionising radiation. Cancer rates are significantly higher in this group, especially that for breast cancer in women (5.1 times higher than for non-carriers). AT homozygote is occurring with an incidence of about 1/40 000. In this group cancers develop at a rate approximately 100 times higher than for non-carriers. For AT-gene carriers the risk of radiation-induced cancer due to mammography may exceed the benefits of early detection. If we could replace X-ray mammography with an optical transillumination method, regular screening for breast cancer would be feasible, even for these patients.

When performing an optical transillumination for tissue diagnostics, a laser light pulse is sent through the tissue, and the transmitted light is detected[3,4]. Light in the near-IR region is used, since the tissue constituents absorb less in this wavelength region than for other wavelengths. However, in this wavelength region, biological tissue is a highly scattering medium, and most of the photons are scattered several times in different directions inside the sample before they exit. Unfortunately, it is the light that has travelled a straight path without being scattered that carries the most information about the tissue located along the straight path between the source and detector. The scattering blurs out the "shadows" from bones, tumours or other irregularities inside the tissue. As can be seen in figure 1, the image blurring due to multiple scattering in the tissue can be reduced by using time-gated viewing with detection of only the first emerging photons, who have been scattered very few times or not at all.

Light transport through a turbid (i.e. highly scattering) medium such as biological tissue can be described mathematically by the diffusion equation. This can be solved analytically for certain geometries, but for more complex geometries it has to be solved numerically. A computer code solving the diffusion equation numerically has been developed at the department, and the purpose of this paper is to study the program dependence of certain parameters, such as boundary conditions and optical constants and to add certain features to the program. One method of making optical transillumination is by using an in-phase/out-of phase laser array. This consists of two (or four, six etc.) laser light sources. They send out one laser pulse each, but with a time delay between them that corresponds to a 180° phase shift. Destructive interference occurs in the mid-plane between the sources. If there is an irregularity (e.g. a tumour) nearby, this plane of destructive interference will be perturbed in some way.

Hopefully, this method can detect very small tumours in the female breast. In this work, the computer code has been adjusted so that experiments of this kind can be simulated. Comparisons with experimental values have been made. This paper will briefly discuss malignant tumours and methods used today for their diagnosis and treatment. It will further present how light interaction with tissue can be modelled with special emphasis on light transillumination diagnosis of tumours in the female breast. Then the function of the computer algorithm is presented and results obtained shown.

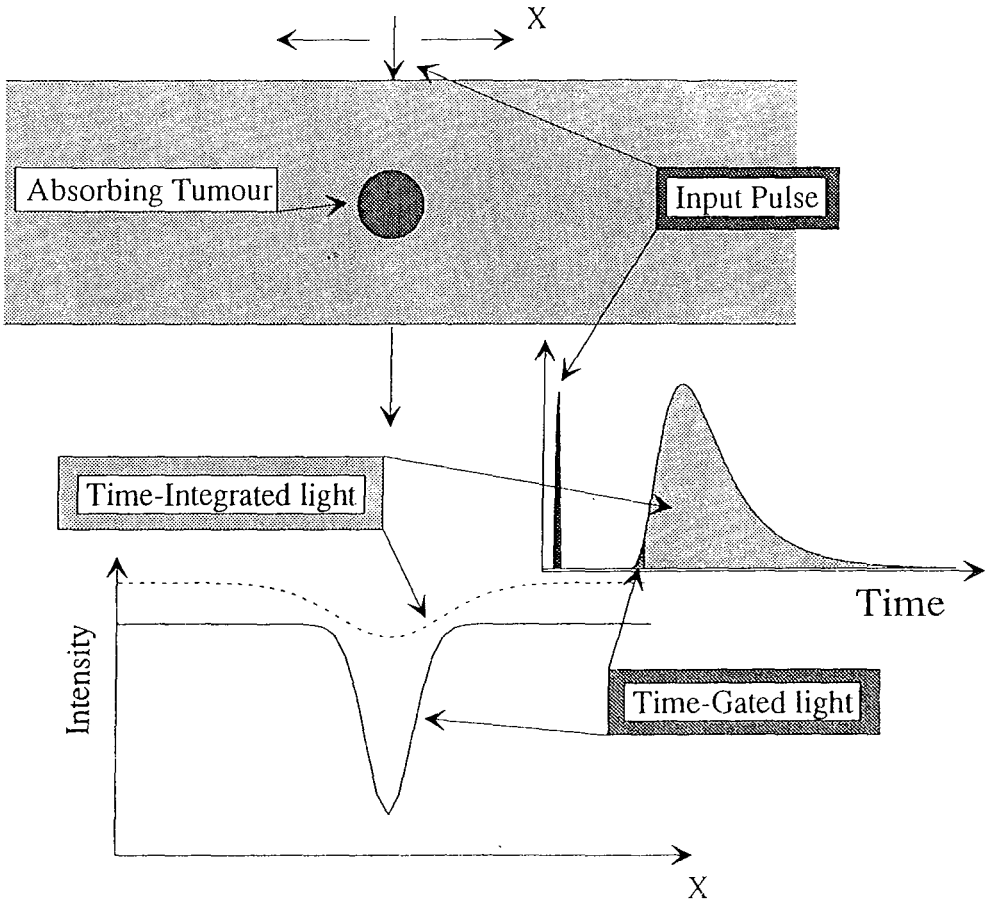

Fig. 1. Optical transillumination: The difference between time-integrated and time-gated viewing. (Ref. 4)

2. Tumours

A tumour, neoplasm, is an abnormal mass of tissue with uncontrolled growth of cells. Tumours can displace and sometimes even destroy surrounding tissues. Furthermore, toxic substances are produced that generally influence the whole organism. There are two groups of tumours, benign tumours and malignant tumours (cancers). The most common sites of cancer are the lung and prostate in males and the breast in females[5].

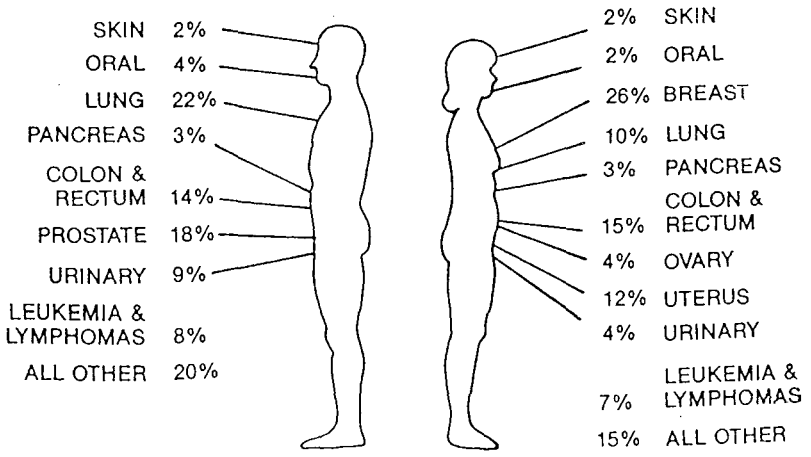

Fig. 2. The most common sites of cancer in men and women. (Ref. 5)

2.1. Benign and malignant tumours

As mentioned above, there are two different kinds of tumours, benign tumours and malignant tumours[6]:

- Benign tumours: A benign tumour is life-threatening only if it grows in a vital organ such as the brain. It grows expansively but is clearly defined. If a microscopic investigation is made of the tumour tissue, no cell splitting is observed, whereas the same investigation of a malignant tumour tissue will reveal a lot of cells under splitting.
- Malignant tumours: These are often called cancers. If a malignant tumour is left untreated it is always life-threatening, even if it is situated in a non-vital organ. Most malignant tumours grow more rapidly than benign tumours. However, some cancers grow slowly for years and then enter a phase of rapid growth. Malignant tumours are invasive, infiltrating and destroying normal tissues surrounding them. Furthermore, malignant tumours have the capacity to metastasise, i.e. invade the lymphatic vessels, blood vessels or body cavities in order to spread secondary tumour cells.

2.2. The formation of tumours

There is a variety of environmental factors that may induce the development of cancers[7]. Although a lot of research work remains to be done before we know the exact influence that different environmental factors may have, we know for sure that certain chemicals can cause cancer, as well as different types of radiation and some viruses. Tumour formation is a complicated and slow multi-step process. It can roughly be divided into two stages: *initiation* and *promotion*, both of them caused by environmental factors. In the initiation stage, the DNA of the cell is hurt but there is no increased cell splitting. Further influence on the cell, referred to as the promotion, causes increased cell splitting and a tumour is formed. The promotion in itself does not affect the DNA of the cell. A tumour is formed due to DNA changes in just one cell. The daughter cells inherit the uncontrolled cell splitting.

· Chemical carcinogens: Examples of carcinogenic chemicals are aromatic hydrocarbons (present in tobacco smoke), asbestos and various fats in our foodstuffs. Some of the carcinogen-induced cell changes can be repaired by cellular enzymes. Thus, a tumour formation can generally not develop unless a subsequent exposure to a promoter follows the initiation step.
· Radiation carcinogens: Radiant energy in the form of ultraviolet (UV) rays and different types of ionising radiation can cause cancer. UV radiation from the sun is more dangerous to fair-skinned people who live in places where they receive a great deal of sunlight. When it comes to ionising radiation, both particulate (alpha particles, neutrons) and electromagnetic radiation (X-rays, gamma-rays) is carcinogenic, since they have the ability to induce mutations. Particulate radiation is generally more carcinogenic than electromagnetic radiation.

2.3. Tumour detection

To successfully treat a tumour it is of vital importance that it is discovered as early as possible. The most important methods to detect tumours are visual examination and with radiological methods, i.e. X-rays and radioactive isotopes (scintigraphy), as well as nuclear magnetic resonance (NMR) and ultrasound. The final diagnosis is generally made by histopathological examination of a tissue sample.

- *X-rays* can be used either with the analogue technique on a photographic plate or digitally with *CT* (Computer Tomography) where cross-sections of the body can be studied.

- When a chemical substance is "labelled" with a *radioactive isotope*, the organ that contains the substance can be studied with a *gammacamera.* ^{99m}Tc is a useful isotope that can be used for examination of the thyroid gland, the liver, the skeleton, the lungs etc. By detecting the amount of radioactive radiation originating from different parts of an organ, the function of the organ can be studied. A three-dimensional technique called *emission tomography* can also be used. This technique resembles CT.

- *Nuclear magnetic resonance (NMR)* can be used to examine the soft tissues of the body. The process has three steps: First the patient is placed in a strong magnetic field. The atomic nuclei align themselves and rotate around the direction of the field. Then radio frequency pulses are added. The nuclei take up the energy (are excited). After the pulses have stopped, the energy is emitted like a weak radio signal.

- *Ultrasound* diagnostics are made with sound waves of frequencies between 0.5 and 20 MHz. A sound pulse that is sent into the body is reflected against the boundaries between different anatomic structures. The echoes are detected, and an image of the cross-section can be made. The principle is the same as for radar imaging.

2.4. Tumour treatment

In principal, there are three major ways to treat a tumour: with surgery, cytostatica or radiation therapy. Other methods of treatment include hyperthermia, PDT (photodynamic tumour therapy) and immunological methods.

- *Surgery*: When performing surgery on a tumour you either remove only the tumour and the very closest surrounding tissue (conservative surgery), or you also remove a large piece of surrounding tissue, especially the lymphatic glands that receive lymph from the tumour region (radical surgery).

- *Cytostatica*: These are substances that have the ability to damage tumour cells more than normal cells. They can be given to the whole body or locally. However, they have several undesirable secondary effects, such as damage of the bone-marrow, loss of hair and indisposition in connection with the treatments. The treatment has to be repeated subsequently.

- *Radiation therapy*: Radiation therapy is based upon the fact that a high radiation dose kills cancer cells, while a low dose might cause cancer. The reason is that the radiation damages the DNA of the cell. Thus a low radiation dose delivered to a healthy tissue can cause a DNA damage that is relatively small, but large enough to possibly cause a mutation. In those cases a tumour cell can develop and grow. On the other hand, a higher dose might cause a damage so severe that cell splitting cannot continue. This phenomenon, together with the fact that tumour cells are more sensitive to the radiation than the normal cells, makes radiation therapy an important complement to surgery.[8]

2.5. Breast cancer

Breast cancer rarely develops before the age of 25. The risk is greater in women with a family history of breast carcinoma. Although the cause remains unknown, there are certain factors that have shown to influence the development of breast cancer:

- Genetic factors: As mentioned above, heredity plays an important role. It has been shown that the risk increases in proportion to the number of first-degree relatives with breast cancer.

- Hormone imbalances: An excess of oestrogen or other hormones appears to increase the risk of breast carcinoma.

- Environmental influences: High dietary fat and/or moderate alcohol consumption are associated with increased risk, as well as ionising radiation e.g. from X-ray mammography.

3. Light propagation in tissue

Most of the methods to detect tumours are based upon the penetration of electromagnetic radiation in tissue. By detecting the back-reflected or transmitted radiation, conclusions can be drawn on what is hidden inside the tissue. For transmission measurement, it is necessary that the amount of transmitted radiation is large enough to allow the detection of a reliable signal. It is thus important that the radiation has a good capability to penetrate the tissue, a feature characterised by the penetration depth of the radiation. When light propagates in a medium it loses intensity due to absorption and scattering, and the penetration depth is the distance the light has to travel before its intensity has dropped by a factor $e^{-1}= 1/2.7\approx 1/3$. The penetration depth depends on the material and on the radiation wavelength. X-rays have a short wavelength ($\approx 10^{-10}$ m) and a large penetration depth in tissue. The absorption and scattering coefficients for visible light ($\lambda= 400*10^{-9}$-$700*10^{-9}$ m) are strongly dependent on the wavelength. In the so called therapeutic window, containing wavelengths from 650-1300 nm, the absorption coefficient is small and penetration depth large. Penetration is deepest in tissues with low scattering and low absorption coefficients. For example, lung tissues are characterised by strong scattering, pigmented tissues by strong absorption, and the aorta wall by both high scattering and absorption. A laser type that is often used in medicine, the Nd-YAG (Neodymium-doped Yttrium-Aluminium-Garnet) emits radiation at 1064 nm, which is a wavelength at which the penetration into all kinds of biotissues is deep and ranges between 1.4 mm (pigmented melanoma) and 8.8 mm (brain of a new-born).

3.1. Light interaction with matter

Incident light can undergo three different types of interaction with the medium: reflection, absorption and scattering.

3.1.1. Reflection:

Reflection takes place when the light passes from one medium to another if the refraction indices of (and hence the speed of light in) the two media are different. We define the reflectance R to be the ratio of the reflected power to the incident power. If the light passes from a medium with refraction index n_i to another with refraction index n_t, the reflectance for the light of parallel and perpendicular polarisation will be

$$R_{\parallel} = \left(\frac{n_t \cos\theta_i - n_i \cos\theta_t}{n_i \cos\theta_t + n_t \cos\theta_i} \right)^2 \quad \text{and} \quad R_{\perp} = \left(\frac{n_i \cos\theta_i - n_t \cos\theta_t}{n_i \cos\theta_i + n_t \cos\theta_t} \right)^2 \text{, respectively and}$$

$$n_i \sin\theta_i = n_t \sin\theta_t \quad ; \quad \theta_r = \theta_t$$

Note that for normal incidence, i.e. when $\theta_i = \theta_t = 0$, we have:

$$R = R_{\parallel} = R_{\perp} = \left(\frac{n_t - n_i}{n_t + n_i} \right)^2$$

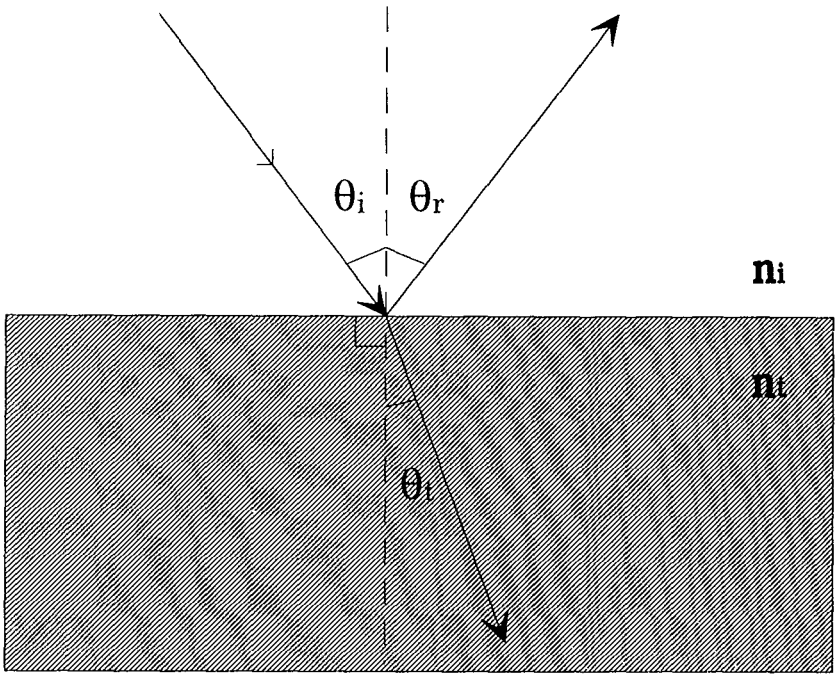

Fig. 3. Reflection and transmission of light in a boundary between two different media.

3.1.2. Absorption:

Absorption represents actual disappearance of the light during passage through matter. If the incoming photon energy (E= hv) matches that of one of the excited states in the molecule, the molecule will absorb the photon, making a quantum jump to that higher energy level. It is most likely that the excitation energy will be transferred, via collisions, to heat motion of the molecules in the absorbing material. A substance is said to show *general absorption* if it reduces the intensity of all wavelengths by nearly the same amount. (No material is known which absorbs all wavelengths exactly equally.) The transmitted light as seen by the eye, shows no change in colour, but a decrease in intensity. If light of original intensity I_0 travels a distance x in a material of absorption coefficient μ_a, the intensity will be:

$I(x)= I_0 e^{-\mu_a x}$ (Beer-Lambert law)

By *selective absorption* is meant the absorption of certain wavelengths in preference to others. Practically all coloured materials owe their colours to selective absorption. For instance, a piece of green glass absorbs well in the red and blue ends of the visible region and thus the transmitted and reflected light is in the green part of the spectrum.

3.1.3. Scattering:

In contrast to the absorption process, the nonresonant scattering process occurs when the incoming light energy is other than the energies for the resonant frequencies[9,10]. Imagine an atom interacting with an incoming photon whose energy is too small to excite it. The electromagnetic field of the light can be supposed to cause an oscillation of the electron cloud of the atom. The electron cloud will oscillate with the frequency of the incident light. The atom behaves like an oscillating dipole and hence it will immediately begin to radiate at that same

frequency. The resulting scattered light consists of a photon that sails off in some direction, carrying the same amount of energy as did the incoming photon - the scattering is elastic. This process is called *Rayleigh scattering*. It increases towards shorter wavelengths, with an approximate $1/\lambda^4$ dependence. In the non-elastic case, there is a coupling between the applied oscillation, caused by the incoming light, and the internal oscillation. The internal oscillation is caused by variations in the polarizability as the molecule rotates or vibrates. The molecule emits or absorbs an amount of energy as it interacts with the photon, depending on whether the molecule changes to a lower or a higher vibrational (or rotational) level. The photon is then shifted to either a higher or a lower energy (called Stokes and Anti-Stokes transitions, respectively). This phenomenon is called *Raman scattering*.

These scattering processes occur when the wavelength of the incoming light is larger than the diameter of the molecule. If light falls on larger particles, whose diameter is much larger than the wavelength, a type of elastic scattering called *Mie scattering* occurs. The intensity of Mie scattering increases towards shorter wavelengths, with an approximate $1/\lambda^2$ dependence.

3.2. Optical transillumination of tissue

Optical transillumination is a phenomenon that most of us have studied by putting a torch in the palm of the hand[11]. This little experiment gives us some interesting results: The light *can* pass through a tissue slab as thin as a hand, but the outcoming light is no longer white, but red. The red colour is due to absorption of the shorter wavelengths. For optical transillumination of tissue, it is important to use radiation that penetrates as deeply as possible into the tissue. The penetration depth is large for wavelengths in the so called therapeutic window between 650 and 1300 nm in the red and near-infrared spectral region. Thus, light in this wavelength region is used. Another interesting result of the torch-experiment is that no shadows of the bones can be seen. We do not actually see straight *through* the hand. What we see is light that has "bounced" back and forth in the hand several times in random directions before exiting on the other side (so called multiple scattering). When optical transillumination is performed the scattering causes a blurring of the image. There are several different techniques to reduce this blurring effect, all of them based on time-resolved detection.

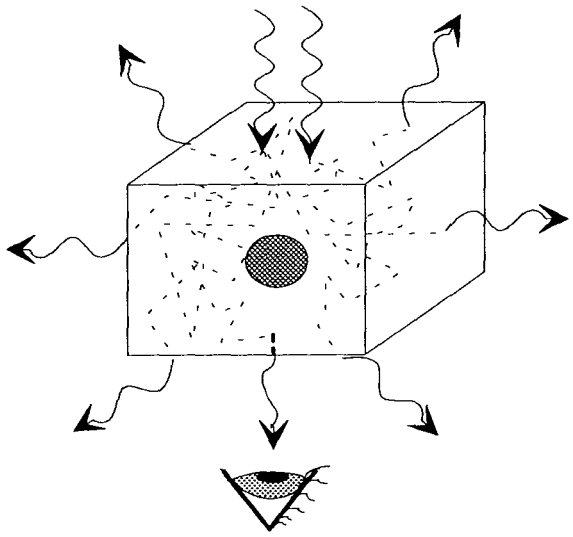

Fig. 4. No shadow of the irregularity can be seen just by looking at the transmitted light.

3.2.1. Tissue optics

As mentioned above, light can interact with matter either by reflection, absorption or scattering. When the matter is biological tissue, how do these three processes show and how important are they?

Reflection: Light is reflected when it passes a surface between two media with different optical properties. Biological tissues have higher refraction indices than air, typically around 1.4, and for light transfer from air to tissue, the reflection is about 2-4%.

Absorption: To obtain good penetration of the light into biological tissue, it is important to choose a wavelength with low absorption in the tissue. In biological tissue the main absorbers are water, proteins, haemoglobin and melanin (see figure 5). All proteins absorb strongly in the UV and blue region. The light absorption of melanin, the most important chromophore in the epidermis, decreases for longer wavelengths λ. Water absorbs in the UV region and haemoglobin absorbs through the visible region up to about 600 nm with peaks at 420, 540 and 580 nm. In the IR region, the tissue absorption increases again, mainly due to the second absorption region of water from 1.3 µm and up. Thus, blue light is generally more absorbed than red. In the red and near-IR region, between 650 and 1300 nm, the attenuation is comparatively small, and we have a favourable wavelength region for light penetration called the therapeutic window. For optical transillumination of tissue we use light in this wavelength region.

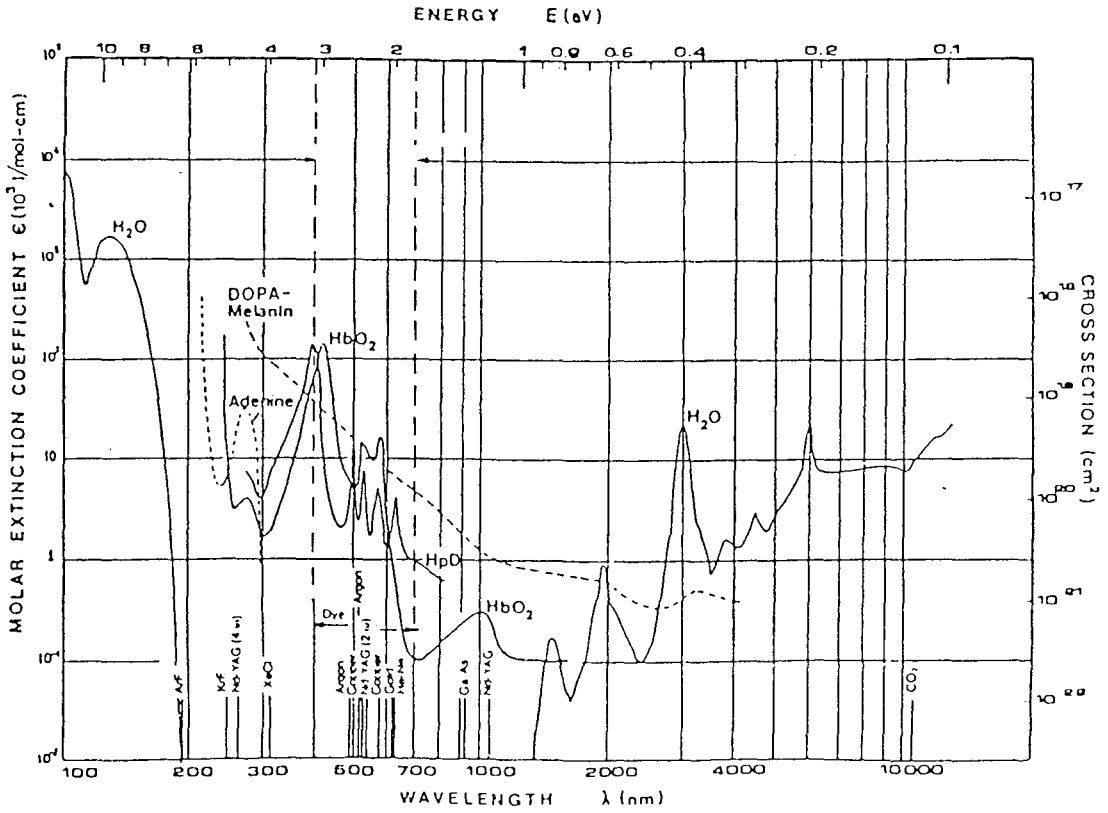

Fig. 5. Absorption curves for different tissue constituents[12].

Scattering: Living tissue is a so called *turbid medium*, i.e. an incident photon is scattered many times by molecules and particles. When a photon is scattered, it loses its original direction in a collision with a particle. Thus, an incident short light pulse will be broadened after passage through a turbid medium. In the wavelength region used for optical transillumination, the absorption is low, but unfortunately the scattering is high. The scattering coefficient μ_s is of the order of 10 mm^{-1}, while the absorption coefficient μ_a is of the order of 0.1 mm^{-1}. However, since each scattering event only causes a small angular deviation on an average, it is possible to detect light through 30-50 mm of breast tissue. It has been shown that the amount of light that has travelled a straight path without being scattered at all is infinitesimally small, so the photons we study have always been scattered several times before detection. The strong scattering in this region is connected with the fact that the wavelengths are comparable with the size of the cell or its constituents (nuclei, mitochondria, cytoskeleton etc.). When performing time-gated transillumination, we send in a short light pulse, and we are interested only in the early part of the exiting light. It has been shown that the early part of the light is more sensitive to the scattering than to the absorption characteristics[13].

3.3. Time-resolved technique

The multiple scattering in tissue prevents the photons from travelling a straight path through the tissue and no shadows from bones, tumours or other regions that hinder the photons can be detected. One method to circumvent this problem is to use a time-gating technique[14]. The idea is that light coming through the tissue at an earlier time has travelled a straighter and shorter path than light leaving later. Thus, this light is less scattered and contains more information about the optical properties of the tissue close to the optical axis.

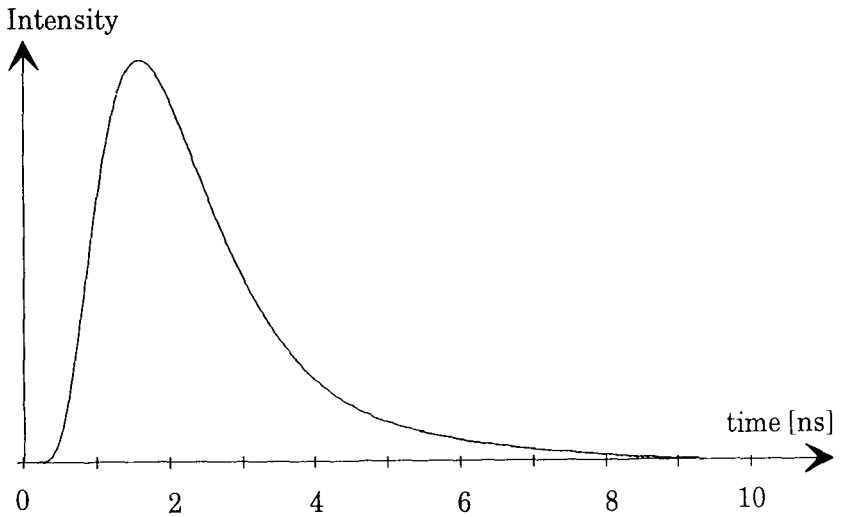

Fig. 6. A typical time dispersion curve after diffusion of a short light pulse through a slab of tissue. Result of a simulation.

3.4. Mathematical theory

A mathematical description of the propagation and scattering characteristics of light in a turbid medium can be made in different ways[15]:

- *Analytical theory*, where we start with Maxwell's equations and take into account the statistical nature of the medium and of the wave. This is the most fundamental approach, but mathematically it is very complex.

- *Transport theory*, which deals directly with the transport of power through the medium, instead of starting with Maxwell's equations. In this model coherent optical effects such as diffraction and interference are neglected as well as inelastic scattering (e.g. fluorescence).

- *Monte Carlo simulations* is a method based on tracing a large number of light rays entering the tissue. The scattering events within the tissue are determined by a probability function matching the scattering coefficient. At each scattering a part of the ray, determined by the absorption coefficient, is absorbed. The scattering angle is given by the scattering phase function.

In this paper, the transport theory has been used. The linear transport equation (Boltzmann Equation) is a balance equation describing the flow of particles (e.g. photons) in a given volume element under consideration of their velocity, location and changes due to collisions (i.e. scattering and absorption).

The optical properties of a material can be characterised by the absorption coefficient, μ_a, and the scattering coefficient μ_s. They indicate the probability for a photon to undergo absorption or scattering per unit length of the material. They are usually measured in cm^{-1} or in m^{-1}. When dealing with the mathematics of light propagation in tissue these two constants are not enough to describe the material. It is also necessary to have a function describing the angular distribution of the scattered light, since the light is not scattered isotropically. The scattering is however symmetric around the direction of the incident light wave, and this distribution can be expressed as a function of the angle between the incoming and the scattered photon. This function is usually characterised by one parameter, g, which is the mean cosine of the scattering angle, also called the *anisotropy factor*.

The transport equation in its general form is impossible to solve. However, in some cases it is possible to approximate the light transport with the diffusion equation. This approximation gives an accurate prediction of the light distribution after many scattering events provided that:

- The scattering dominates over the absorption; $\mu_a << (1\text{-}g)\,\mu_s$
- Detection is made far from the light sources (so that we can assume multiple scattering)

As for the first condition, scattering generally dominates over the absorption in soft tissues for wavelengths between 650 and 1300 nm. The second condition on multiple scattering may not be totally satisfied for the earliest light, but as we want to study qualitative rather than quantitative features of light propagation, this should not cause any problems. If we assume that the conditions are fulfilled, the diffusion approximation can be made and the transport equation is simplified to yield the *diffusion equation*:

$$\frac{1}{c}\cdot\frac{\partial}{\partial t}\phi(\mathbf{r},t) - D\nabla^2\phi(\mathbf{r},t) + \mu_a\phi(\mathbf{r},t) = q(\mathbf{r},t)$$

where $\phi(\mathbf{r},t)$ is the diffuse fluence rate, $q(\mathbf{r},t)$ is the photon source and D is the diffusion coefficient, $D = [3(\mu_a+(1-g)\mu_s)]^{-1}$. This equation can be solved analytically for some simple geometries. For a homogeneous tissue slab it has been solved by Patterson *et al.*[16]

$$T(d,t) = (4\pi Dc)^{-1/2} t^{-3/2} \exp(-\mu_a ct) \times \left\{ (d-z_0) \exp\left[-\frac{(d-z_0)^2}{4Dct} \right] - (d+z_0) \exp\left[-\frac{(d+z_0)^2}{4Dct} \right] \right.$$

$$\left. + (3d-z_0) \exp\left[-\frac{(3d-z_0)^2}{4Dct} \right] - (3d+z_0) \exp\left[-\frac{(3d+z_0)^2}{4Dct} \right] \right\}$$

where d is the slab thickness and $z_0 = [(1-g)\mu_s]^{-1}$.

Since we want to study inhomogeneous slabs of tissue, a computer code solving the diffusion equation numerically has been used. This allows us to vary the scattering and absorption in regions within the slab.

4. Materials and methods

4.1. The model

I have used a computer program to simulate light diffusion through tissue slabs with varying optical properties. The program is written in C using Borland C 3.1 and it was originally developed at the Atomic Physics Division at Lund Institute of Technology. The program simulated light diffusion through a slab of tissue, described by a 3-dimensional matrix. An infinitesimally short narrow collimated light pulse is normally incident on the surface of the tissue slab. The photon sources are assumed to be established within the medium by the initial scattering of all incident photons at a depth $z_0 = 1 / \mu'_s$. The diffusion equation was translated to a differential equation for finite steps in the x,y,z and time variables. The system is solved numerically, and the result when running the program is the time dispersion curve in points specified before running.

I adjusted the program so that one, two or four light sources could be placed at different points on the entrance surface, and so that there could be a time delay between the sources. I also added the possibility to Fourier transform the resulting time dispersion curve. In the frequency plane, the time delay between the two sources would correspond to a phase shift between them.

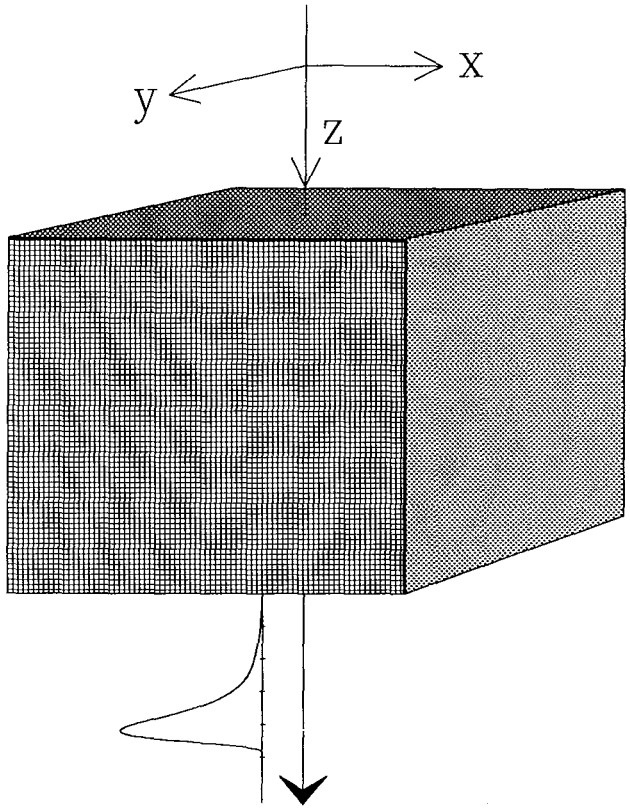

Fig. 7. The geometry of the model used for numerical calculation of the light diffusion in tissue. A short light pulse is incident along the x axis on the top surface and a time dispersion curve is calculated in different points on the bottom surface or within the specimen.

4.2. The ADI algorithm

When the diffusion equation is translated to a differential equation for finite room- and time steps, a tridiagonal system of equations is yielded. This is solved by the computer by using a generalised Crank-Nicholson algorithm in three dimensions called the ADI (alternating direction implicit) method[17]. It solves the equation for each dimension separately using a third of a time-step for the x, y and z variables. The resulting equation for the x dimension is:

$$\phi_{xyz}^{t+1/3} - \phi_{xyz}^{t} = \frac{c\Delta t}{3n\Delta^2} \{ D_{x+1/2yz}(\phi_{x+1yz}^{t+1/3} - \phi_{xyz}^{t+1/3}) - D_{x-1/2yz}(\phi_{xyz}^{t+1/3} - \phi_{x-1yz}^{t+1/3})$$

$$+ D_{xy+1/2z}(\phi_{xy+1z}^{t} - \phi_{xyz}^{t}) - D_{xy-1/2z}(\phi_{xyz}^{t} - \phi_{xy-1z}^{t})$$

$$+ D_{xyz+1/2}(\phi_{xyz+1}^{t} - \phi_{xyz}^{t}) - D_{xyz-1/2}(\phi_{xyz}^{t} - \phi_{xyz-1}^{t}) \} - \frac{c\Delta t}{6n}\mu_a (\phi_{xyz}^{t} - \phi_{xyz}^{t+1/3})$$

where ϕ_{xyz}^{t} is the fluence rate in the matrix element (x,y,z) at time t, Δt is the time step, Δ is the step size in x, y and z directions and $D_{x+1/2yz}$ is the average of the diffusion coefficient in the matrix elements (x,y,z) and (x+1,y,z).

4.3. The program

The program was written in Borland C++ 3.1. A full program listing is given in Appendix A. The parameters such as optical constants, boundary conditions, matrix dimensions and source- and detector points are specified in the file intid.txt. An example of what the infile can look like is shown below. For the exact listing of the infile, see Appendix B.

Infile: parameter	value	description
XMAX_YMAX_ZMAX	30 30 10	: 1+number of elements in each direction*)
abs_scatt_coeff.	1 1000	: absorption- and effective scattering coefficient in the material, in m^{-1}.
Regions	1	: number of regions (tumours)
Reg_abs_scatt_coeff.	20 750	: absorption- and effective scattering coefficient in the region, in m^{-1}.
Type_0=box_2=Cylind	2	: shape of the region (0=box, 1=sphere, 2=cylinder)
Cyl_X_Z_R	13 5 1	: coordinates of the region (depending on the shape)
dX	0.004	: length of an element (the same in x-, y- and z-directions), in m.
dt(psec)	20	: the time step
Timediff(psec)	2500	: time delay between the two light pulses
Light_in_X_min_max	10 10	: x-coordinates for the first light source
Light_in_Y_min_max	15 15	: y-coordinates for the first light source
Light_intensity	1	: light intensity for the first source
Light_in_X_min_max	20 20	: x-coordinates for the second light source
Light_in_Y_min_max	15 15	: y-coordinates for the second light source
Light_intensity	1	: light intensity for the second source
Bound_cond._X	1	: boundary condition in the x-direction, see ch. 7.1
Bound_cond._Y	1	: boundary condition in the y-direction
Bound_cond._Z1_in	0	: boundary condition in the z-direction, top surface
Bound_cond._Z2_out	0	: boundary condition in the z-direction, bottom surface
Point_1_X_Y_Z	16 15 9	: first detection point
Point_2_X_Y_Z	17 15 9	: second detection point
Point_3 ... etc.		

) the elements with index 0 and XMAX (or YMAX or ZMAX) represent the boundary, hence the size of the matrix is (XMAX-1)(YMAX-1)*(ZMAX-1).

Most of the simulations were made on a 486 50 MHz PC. This type of computer can handle matrices up to 25*25*25 elements. For larger matrices and to get shorter computation time, a DEC α PC was used.

5. Results

5.1. Purpose of this paper

This paper has two principal aims:
• to study the program sensitivity and dependence of certain parameters, such as the boundary conditions, matrix dimensions, optical constants, size of the time step etc.
• to add certain features to the program:
- two or four light sources placed in arbitrary positions on the top surface
-a time delay between the pulses from the sources
-the possibility to study the result of the light diffusion either time- or frequency resolved.

These features combined with the ones already given make it possible to simulate interesting experiments, such as scanning of the two sources and/or the detector over a tissue slab hiding a region of different optical properties (simulating a tumour) of either spherical, cylindrical or square shape. Another interesting investigation to make is to study a theory put forward by B. Chance *et al.*[18] This theory suggests that four light sources are placed symmetrically about the centre point of the (x,y,0) plane and that they are modulated out of phase with respect to each other. The diffusing waves originating from the out-of-phase sources give, in the mid plane, destructive interference. In the frequency space this will result in an amplitude null and a sharp phase transition.

The interesting feature of this out-of-phase method is that if a region with differing optical properties is situated near the mid-plane, the result of the Fourier transform will not be the same as for the homogeneous case. Since tumours have different optical properties than normal tissue, this method might be a tool to detect very small mammary cancer tumours. One problem might be that the method could be too sensitive to other irregularities such as blood vessels.

5.2. Terminology

In the following chapters the influence of different input parameters on the time dispersion curve has been investigated. In many cases, the best input value is already known, and the investigation is made merely to see how much the curve shape is altered when other input values are used. One way to measure the curve shape is by calculating the ratio I_{win} / I_{tot}, where I_{win} is the intensity exiting within a certain time window and I_{tot} is the total intensity. The ratio I_{win} / I_{tot} is plotted for different input values. Hence, the correct value of I_{win} / I_{tot} is not known, the ratio is just used as a means of comparison.

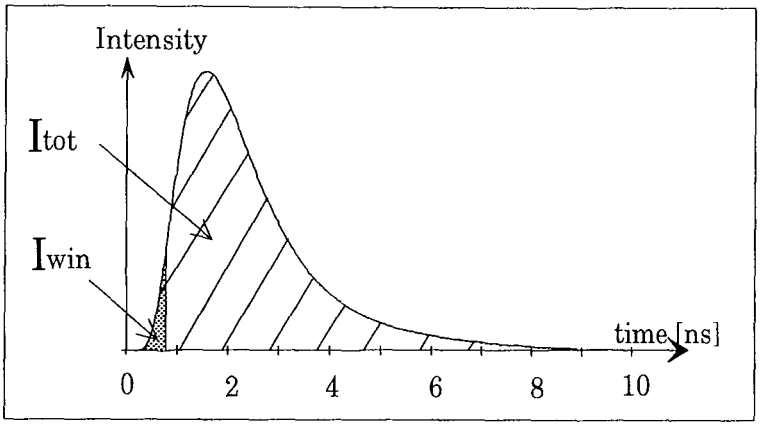

Fig. 8. An illustration of how the ratio I_{win}/I_{tot} is formed.

5.3. Boundary conditions

Before solving the equation system it is necessary to define the boundary conditions in the x-, y-, and z-directions. Since the numerical calculation of an element at a certain time is based upon the values of this element and its neighbour elements one time-step back, all the elements, even the ones next to the boundaries, need to have neighbours in all directions. Thus, we have one additional pixel outside each boundary in the x- and y-direction. They represent the boundary conditions. In the experiments they were given the same value as the neighbour element. This has previously shown to give the best result. As for the z-direction, it is possible to choose different boundary conditions for the top- and bottom- surfaces, noted Z1 and Z2 respectively. These conditions are defined by the variables `BoundCondZ1` and `BoundCondZ2` in the data infile. A boundary element is given the value of `BoundCond` multiplied by the value of its nearest neighbour inside the matrix.

Example:

```
mx[X][Y][0][TimeNow]=BoundCondZ1*mx[X][Y][1][TimeNow];
```

Series of simulations were made with different z-boundary conditions but otherwise with the same matrix- and simulation parameters. It was already known that `BoundCondZ1` and `BoundCondZ2` = 0 gives the best results[19], but it is still interesting to see if a different boundary condition would alter the shape of the time dispersion curve and in that case how much. From figure 9 the conclusion seems to be that the curve shape *is* altered when the boundary conditions change, and that the system is quite sensitive even to small changes.

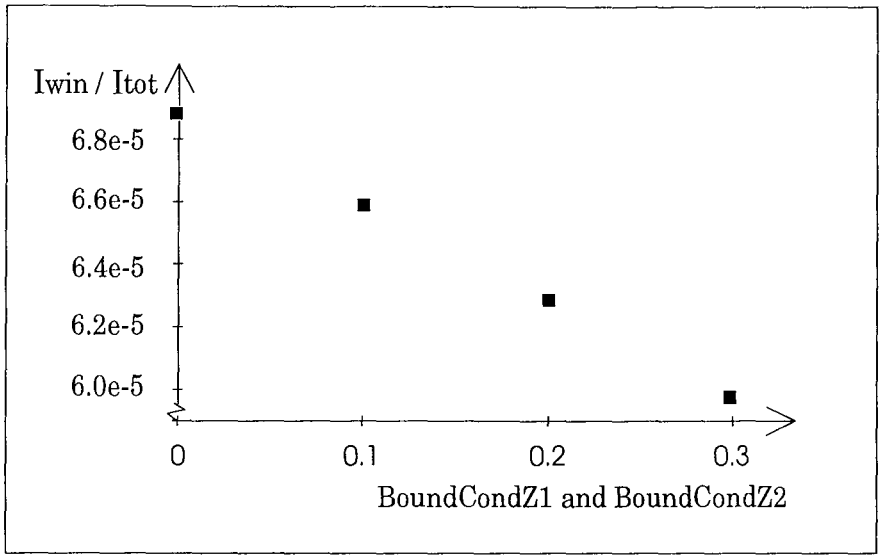

Fig. 9. The x-values represent boundary values in the z direction. The y-values represent $I_{400}/ I_{tot,}$, i e the intensity during the first 400 ps divided by the total intensity. This is a measure of the curve shape.

5.4. Matrix dimensions

The matrix in the computer code is a model of the medium through which the light is sent, in this case the tissue slab. The dimensions of the matrix are defined in the infile. An element is defined by its location (x,y,z). The matrix parameters we can choose are XMAX, YMAX, ZMAX, denoting the number of matrix elements in each direction, and dX, denoting the side of each element (all elements are cubic). To be exact, the elements in location 0, XMAX etc. always represent the boundary, and hence the number of actual elements in each direction is (XMAX-1), (YMAX-1) and (ZMAX-1) respectively. To determine the importance of the matrix dimensions, I kept the size of the matrix constant while varying the coarseness. The size of the "slab" was always 6*6*6 centimetres.

The variation of the I_{win}/I_{tot} ratio and hence the variation in curve shape for different matrix dimensions is illustrated in figure 10, where the time window is 1.2 ns. The window always starts at a time t_0 after the pulse, where t_0 is the time for light transfer straight through the slab when no scattering occurs. We can draw the conclusion that the curve shape varies drastically for coarse matrixes but seems to remain constant when there are 25 or more elements in each direction. This is an important observation, as we can save both time and computer memory by using less elements. As the program is an ordinary DOS-application, the matrix dimensions are limited to 25*25*25 elements.

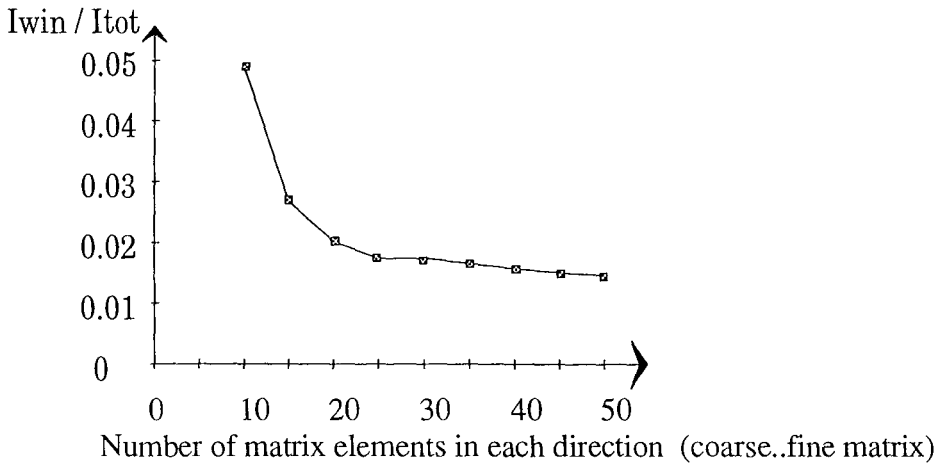

*Fig. 10. The ratio Iwin/Itot is plotted for different element sizes, where the window is 1.2 ns. The X-axis values denote the number of matrix elements in each direction, i.e. a higher number means a finer matrix. The sample size was held constant at 6*6*6 cm.*

5.5. Time step

The length of the time step, `dt`, and the number of time steps, No_of_iter (called "`Niter`" in the program) can also be chosen in the infile. They determine the resolution in the time dispersion curve and hence the resolution in frequency space after our Fourier transform. From figure 11, which is formed as described in figure 8, it is clear that the shape of the dispersion curve is practically insensitive to changes of the time step length.

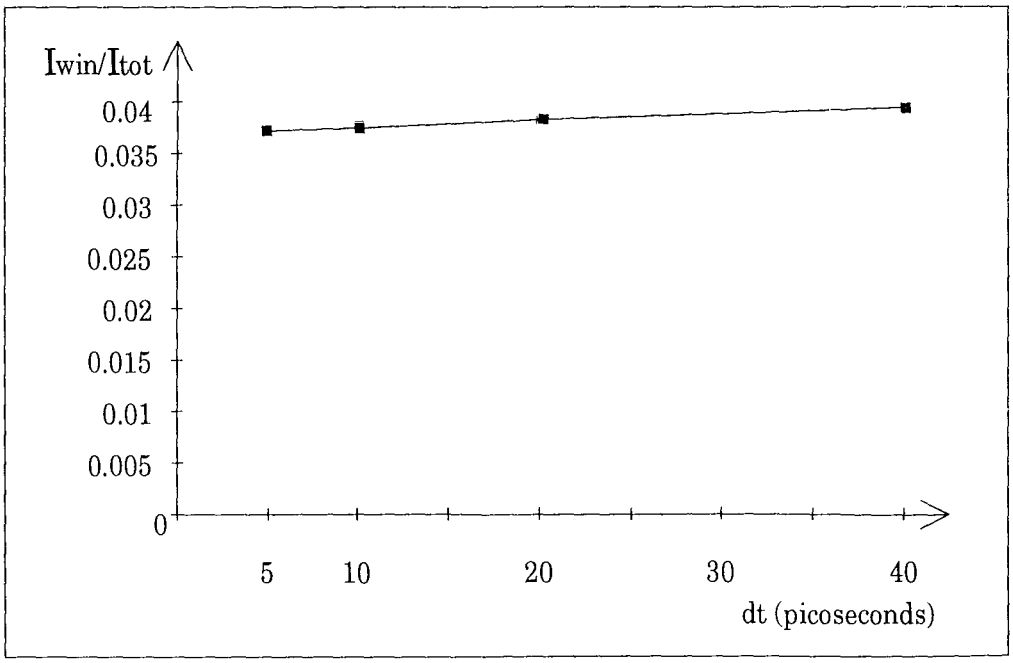

Fig. 11. The curve shape as a function of the time step `dt`. The time window is here 800 picoseconds.

5.6. Scattering coefficient

A material is characterised by a set of three optical parameters: μ_s, μ_a and the anisotropy factor, g. The scattering coefficient, μ_s, is measured in m^{-1}. It denotes the number of scattering events per unit of length. The anisotropy factor, g, is the mean cosine of the scattering angle in the diffusion approximation. If the scattering is anisotropic, i e non-symmetric in space, the scattering coefficient can be modified to $\mu_s' = (1-g)\mu_s$. By doing this modification the anisotropic scattering is transformed into isotropic scattering. The new set of parameters is:

$$\mu_a' = \mu_a \qquad \mu_s' = (1-g)\,\mu_s \qquad g' = 0$$

One would suspect that a higher μ_s' would result in a time dispersion curve with less early light, since the photons undergo more scattering. Figure 12 confirms this suspicion. Note that the y-axis has a logarithmic scale.

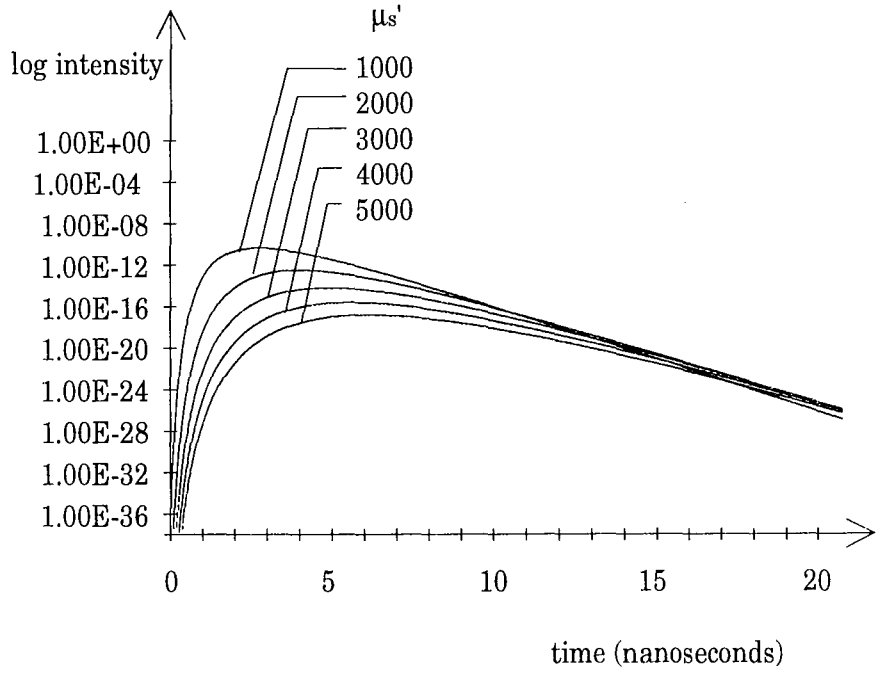

Fig. 12. A higher scattering coefficient results in a lower light intensity. Note that the y-axis has a logarithmic scale.

These pictures show that a higher scattering coefficient seriously affects the pulse shape and the amount of light transmitted through the specimen.

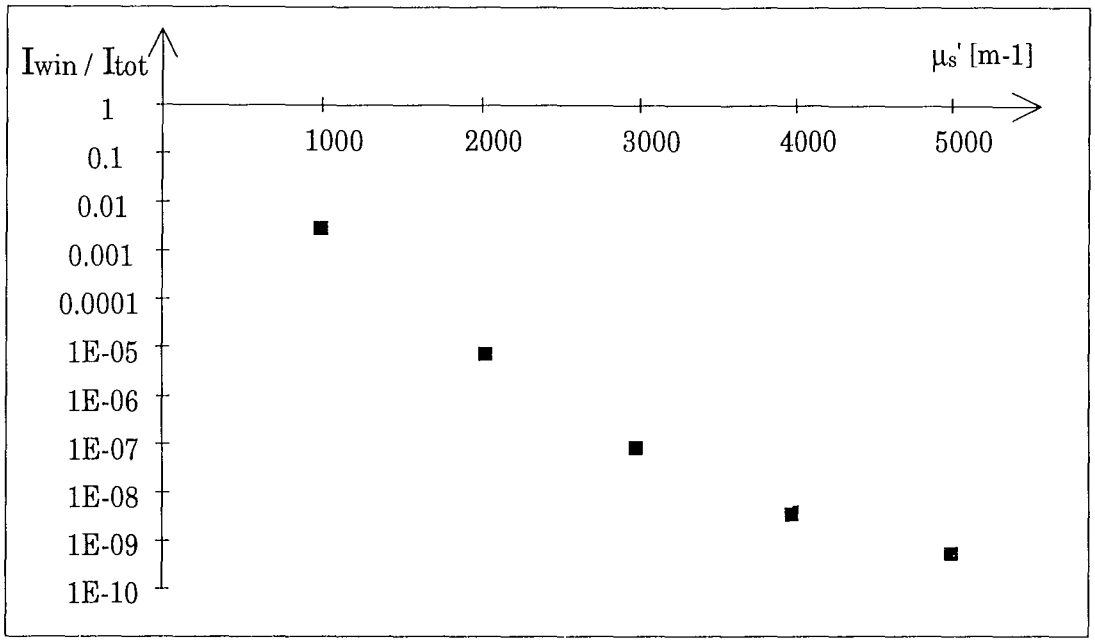

Fig. 13. I_{win} / I_{tot} *, i.e. the intensity during the first nanosecond divided by the total intensity plotted against effective scattering coefficient.*

5.7. Anti-phase measurements

When two identical light sources are placed on one surface of the tissue sample, and we modulate the sources 180° out of phase, the diffusing waves will interfere and give an amplitude null and a sharp phase transition in the mid plane between the sources. If an absorbing region, such as a tumour, is hidden inside the tissue, this null plane will bend off. This method might be a useful means to detect small breast tumours[20]. In this case we are initially working in the time-plane and the 180° phase shift is obtained by sending the two light pulses at a time separation that corresponds to half a wavelength. Note that this wavelength is not the light wavelength, but the wavelength corresponding to the chosen modulation frequency. For example, if we want to modulate our sources at the frequency 200 MHz, the corresponding period time is $1/(200*10^6)$ s = $5*10^{-9}$ s, and the time separation between the two light pulses should be $2.5*10^{-9}$ s. I added a Fourier transform algorithm to the program, to be able to convert the time-dispersion curve to the frequency plane.

5.8. Four sources

One new feature of the program is that four light sources can be placed on the top surface of the slab model. The pulses are sent two at the time, so that two of them are sent at time t=0 and two are sent after a chosen delay. I tried to simulate an experiment previously conducted by Chance *et al.* (ref. 18). As can be seen in figure 14, the sources are placed on equal distance along the x-axis, and the detector is scanned, also in the x-direction. The spacing between the sources is approximately 3.5 cm, and the detector is at 5 cm depth. Initially, the sample is homogeneous without any absorbing region. The goal of this first test is to study the phase- and amplitude values as the detector is moved along the x-axis.

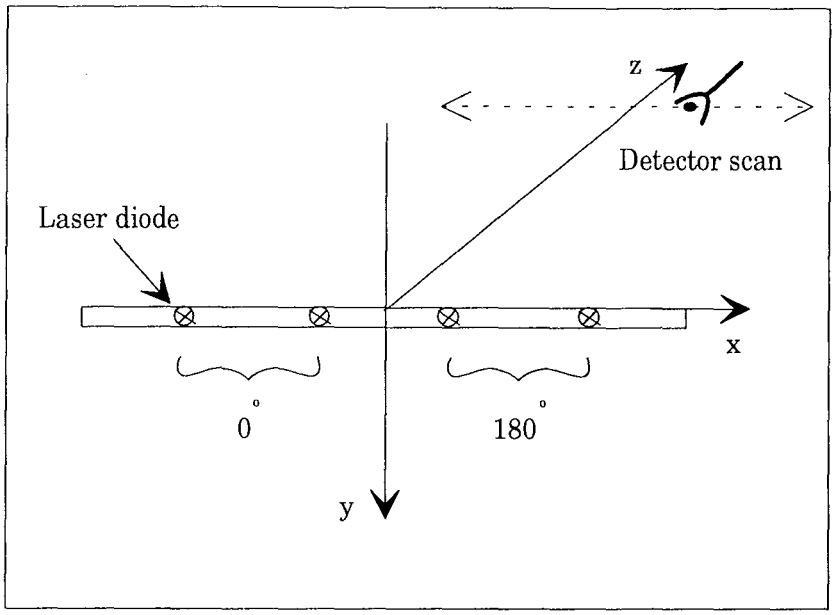

Fig. 14. The geometry for the determination of the amplitude null and sharp phase transition.

The two 0° sources are located in elements number 21 and 35, corresponding to 5.25 and 8.75 cm, respectively (dX = 0.0025 m). The two 180° sources are in elements 49 and 63, corresponding to x = 12.25 cm and x = 15.75 cm. Hence, there is an amplitude null and a sharp phase transition in x = 10.5 cm. The optical constants μ_a and μ_s' were set to 2 and 1000 m^{-1}, corresponding to 0.5% Intralipid®, as used by Chance *et al.* in the physical experiment.

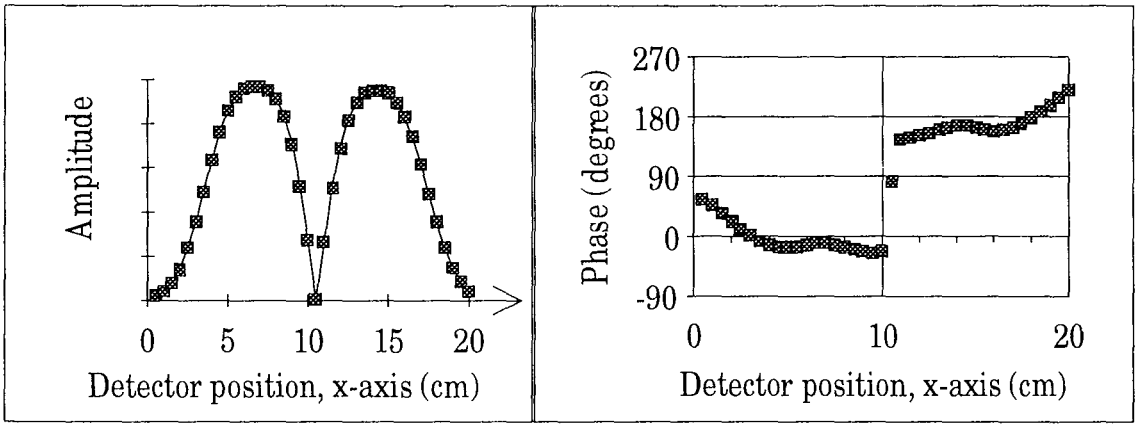

Fig. 15. Amplitude and phase values received when simulating an optical transillumination according to figure 14.

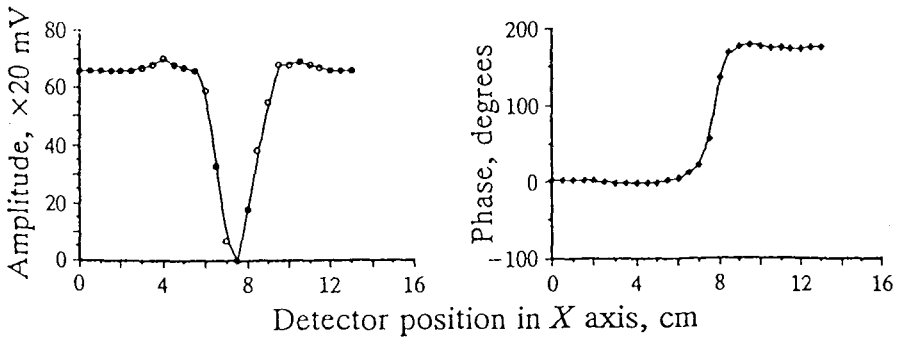

Fig. 16. Amplitude and phase values received when performing the experiment described above[18].

When comparing these values with the experimental ones, figure 16, we can see that they agree well, except for the amplitudes. The experimental amplitude values are nearly constant until the mid-line null is reached, while the simulated amplitudes have maximums straight under each pair of light sources. I cannot see the reason why the experimental curve looks like it does, and not like the simulated one.

In the second test, an absorbing region has been scanned inside the sample in the x-direction. This is equivalent to scanning sources and detector in parallel across the sample. The absorbing region is supposed to mimic a tumour. In this case, the absorber was rod-shaped with a square cross-section in the x-z plane. In the y-direction it goes straight through the sample. See figure 17.

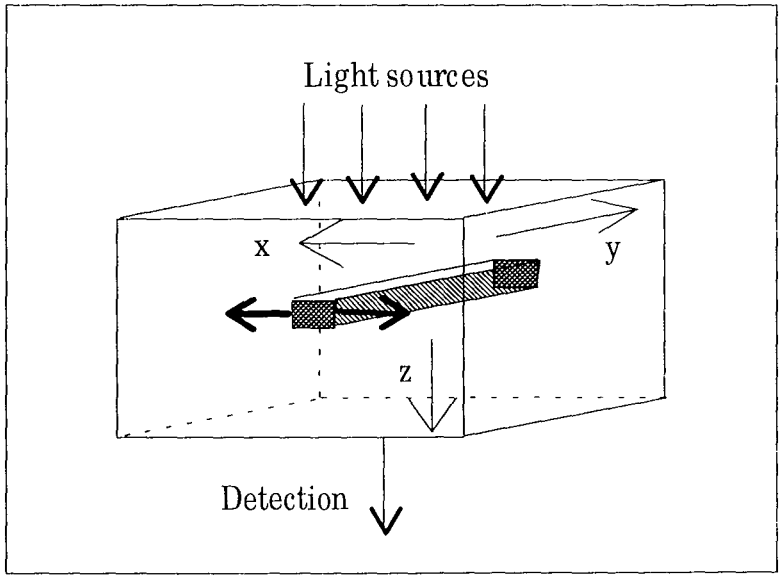

Fig. 17. Geometry for the second test. The detector is placed in the point where we got an amplitude null in the previous test (i.e. at x=10.5 cm). The black rod is translated in the x-direction, midway between sources and detector. Values of amplitude and phase are noted for each centimetre the absorber is translated.

If we now study the resulting amplitude and phase curves in figure 18, we see that their shapes are similar to those presented by Chance *et al.* in figure 19. They differ in scale, since in the simulations the absorber was rod-shaped with 2.5 mm square cross-section, while Chance *et al.* used a cylinder of 1.25 mm diameter to get their experimental values.

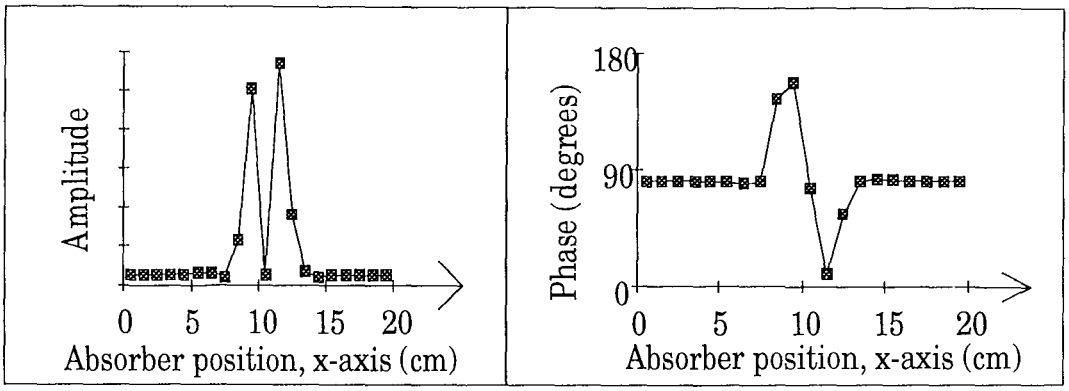

Fig. 18. Amplitude and phase plots for perturbation of the amplitude null and phase transition for the model in figure 17.

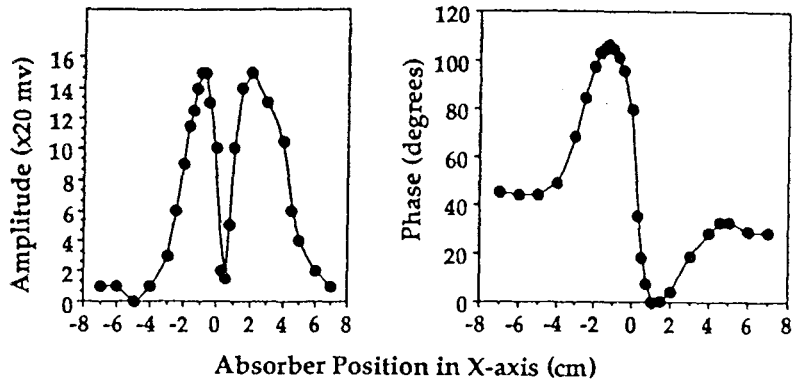

Fig. 19. Amplitude and phase values received when translating a black absorber cylinder inside the sample[18].

6. Conclusions

This work has clearly shown that the developed computer code well manages to simulate transillumination experiments. When four light sources are used to simulate an in-phase/anti-phase laser source array, and the resulting time dispersion curve is Fourier transformed, the results agree very well with those obtained experimentally by Chance *et al.* (ref.18). The model seems to be more sensitive to variations in matrix coarseness than to changes in boundary values. As an example, a slab of 6*6*6 centimetres should at least be divided in 25*25*25 elements when one light source is used. The model is also sensitive to choice of optical constants and accurate values are required to get a result that compares well with experimentally recorded data.

6.1. Future

Now that the results from the model have shown to be in good agreement with experimental values, the next step should be to use the model to assess the potential of in-phase/anti-phase laser source array approach for optical mammography. It would be interesting to try out the optimal geometry of sources and detector and investigate differently shaped and sized tumours, as well as other tumour locations, e.g. tumours near a boundary. Another possibility is to investigate the resolution of two small tumours near each other. The sensitivity and robustness of the method are other important subjects. It might be too sensitive and react even on very small irregularities, such as blood vessels.

7. Acknowledgements

I would like to thank Stefan Andersson-Engels for guiding me through this work. Roger Berg helped me with the computer problems.

I would like to thank Johan Eker for giving me good advice and for being so patient and supportive. He also helped with the proof reading.

Annika Nilsson cheered me up when I needed it.

I would also like to thank professor Sune Svanberg and the other persons at the department who made this paper possible.

Finally, I would like to thank my family and everyone else who has helped me and encouraged me during the work with this project.

8. References

1 M. Swift, D. Morrell, R.B. Massey, C.L. Chase: "Incidence of cancer in 161 families affected by Ataxia-Telangiectasia", The New England Journal of Medicine **325**, 1831 (1991)

2 P.J. McKinnon: "Ataxia-telangiectasia: an inherited disorder of ionizing-radiation sensitivity in man", Human Genetics **75**, 97 (1987)

3 S. Svanberg: "Tissue diagnostics using lasers", Invited book chapter for "Lasers in medicine", G. Pettit and R.W. Wayant (eds.) (To appear)

4 R. Berg, S. Andersson-Engels, S. Svanberg: "Time-resolved transillumination imaging", chapter in "Medical Optical Tomography: Functional Imaging and Monitoring", G. Müller, B. Chance and others (eds.)

5 T. Andreoli, C. Carpenter, F. Plum, L. Smith: "Cecil Essentials of Medicine", 2nd ed., W.B. Saunders Company (1990)

6 R.N.M. MacSween, K. Whaley: "Muir's Textbook of Pathology", 13th ed., Edward Arnold (1992)

7 S. Robbins, R. Cotran, V. Kumar: "Pocket Companion to Robbins Pathologic Basis of Disease", W.B.Saunders Company, Harcourt Brace Jovanovich, Inc. (1991)

8 B. Jacobsson: "Medicin och teknik", 3rd ed., Studentlitteratur (1987)

9 S. Svanberg: "Atomic and molecular spectroscopy", 2nd ed., Springer-Verlag (1992)

10 E. Hecht: "Optics", 2nd ed., Addison-Wesley Publishing Company (1987)

11 K. Svanberg, S. Svanberg: "Le laser en médecine", La Recherche **24**, 686 (1993)

12 J.-L. Boulnois: "Photophysical processes in laser-tissue interactions", in R.Ginsberg (ed.) "Laser Applications in Cardiovascular Diseases", Futura (1987)

13 S. Andersson-Engels, R. Berg, S. Svanberg: "Effects of optical constants on time-gated transillumination of tissue and tissue-like media", J.Photochem. Photobiol. B: Biol. **16**, 155 (1992)

[14] S. Andersson-Engels, R. Berg, O. Jarlman, S. Svanberg: "Time-resolved transillumination for medical diagnostics", Optics Letters **15**, 1179 (1990)

[15] R. Graaff, J.G. Aarnoudse, F.F.M de Mul, H.W. Jentink: "Light propagation parameters for anisotropically scattering media based on a rigorous solution of the transport equation", Applied Optics **28**, 2273 (1989)

[16] M. S. Patterson, B. Chance, B. C. Wilson: "Time resolved reflection and transmittance for the non-invasive measurement of tissue optical properties", Applied Optics **28**, 2331 (1989)

[17] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T Vetterling: "Numerical recipes in Pascal" , Cambridge University Press, Cambridge (1990).

[18] B. Chance, K. Kang, L. He, J. Weng, E. Sevick: "Highly sensitive object location in tissue models with linear in-phase and anti-phase multi-element optical arrays in one and two dimensions", Proceedings of the National Academy of Sciences of the United States of America **19**, 3423 (1993)

[19] Roger Berg, Private communication (1993)

[20] A. Knüttel, J.M. Schmitt, R. Barnes, J.R. Knutson: "Spatial localization using interfering photon-density waves: Contrast enhancement and limitations", Proc. SPIE **1888**, 322 (1993)

Appendix A: Computer code listing

```
/*************************************************************************

    PROGRAM: NYTID4.C

    PURPOSE: Tissue Optics Simulation Program for Borland C++.

    Rev.  : March 1994

*************************************************************************/

# include <stdlib.h>
# include <stdio.h>
# include <malloc.h>
# include <math.h>
# include <float.h>
# include <string.h>
# include <ctype.h>
# include <conio.h>
# include <time.h>
#define MAXT 1
#define PI 3.141592654f
#define C0 (float)3E8
#define N (float)1.4
#define C (float)(C0/N)
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

float absfloat(float f,float g)
{
  return (sqrt(pow(f,2)+pow(g,2)));
}

void    goexit(char *);
void    tridag(float*,float*,float*,float*,float*,int);
void    nrerror(char *);
void    free_vector(float *,int);
void    realft(float *,int,int);
void    four1(float *,int,int);
void    Freemx(float  * * * *m,short nxl,short nxh,
                        short nyl,short nyh,
                        short nzl,short nzh,
                        short ntl);
void    Freein(char * * *m,short nxl,short nxh,
                        short nyl,short nyh,short nzl);
float   *vector(int,int);
float   *gam;
float   maxi(float value1, float value2);
```

```
float   mini(float value1, float value2);
float   Diffc(float,float,float);
float   * * * *mx;
float   * * * *Allocmx(short nxl,short nxh,
                       short nyl,short nyh,
                       short nzl,short nzh,
                       short ntl,short nth);
float   Absarray[16];
float   Diffarray[16];
float   Dcon;

int     MAXX,MAXY,MAXZ;
int     xmin,xmax,zmin,zmax;
char    * * *in;
char    * * *Allocin(short nxl,short nxh,
                       short nyl,short nyh,
                       short nzl,short nzh);


void main(void){
        float   Abscoeff,Scattcoeff1;
        float   Dump;
        float   *a;
        float   *b;
        float   *c;
        float   *r;
        float   *u;
        float   Time;
        float   dX,Intensity,BoundCondX,BoundCondY,BoundCondZ1,BoundCondZ2;
        float   dt,Constant1,Constant2,fdump,fdump2,Timediff;
        float  *data1,*data2,*data3,*data4,*data5;
        float  *data6,*data7,*data8,*data9,*data10;
        int     maxdim;
        int     LiXmin,LiXmax,LiYmin,LiYmax,Niter;
        int     LiXmin2,LiXmax2,LiYmin2,LiYmax2;
        int     X,Y,Z,TimeNow,TimeNext;
        int     Xpoint1,Ypoint1,Zpoint1,Xpoint2,Ypoint2,Zpoint2;
        int     Xpoint3,Ypoint3,Zpoint3,Xpoint4,Ypoint4,Zpoint4;
        int     Xpoint5,Ypoint5,Zpoint5,Xpoint6,Ypoint6,Zpoint6;
        int     Xpoint7,Ypoint7,Zpoint7,Xpoint8,Ypoint8,Zpoint8;
        int     Xpoint9,Ypoint9,Zpoint9,Xpoint10,Ypoint10,Zpoint10;
        int     Inpoint;
        int     i,j,k,l,m,n,o,First;
        int     Type,Regions;
        long    Number;
        long    noofbytes;
        char    string[20];
        time_t  starttime,stoptime;
        FILE   *infile;
        FILE   *outarr1;
        FILE   *ftarr;
```

```
infile=fopen("intid4.txt","rt");
if(infile==NULL){
        printf("Error: Can't find infile.txt.\n");
        exit(1);
}
ftarr=fopen("ftabs.dat","wt");

fscanf(infile,"%s %d %d %d",string,&MAXX,&MAXY,&MAXZ);
fprintf(ftarr,"%s %d %d %d\n",string,MAXX,MAXY,MAXZ);

noofbytes=(long)((long)(MAXX+1)*(long)(MAXY+1)*(long)(MAXZ+1)*2L);
printf("No Of points : %ld\n",noofbytes);

mx=Allocmx(0,MAXX,0,MAXY,0,MAXZ,0,1);
if(mx==NULL) goexit("mx alloc error.");

maxdim=(int)maxi(MAXX,MAXY);
maxdim=(int)maxi((float)maxdim,MAXZ)+1;

a=vector(1,maxdim);
b=vector(1,maxdim);
c=vector(1,maxdim);
r=vector(1,maxdim);
u=vector(1,maxdim);
gam=vector(1,maxdim);
First=1;

in=Allocin(0,MAXX,0,MAXY,0,MAXZ);
if(in==NULL) goexit("in alloc error.");

fscanf(infile,"%s %f %f",string,&Absarray[0],&Scattcoeff1);
fprintf(ftarr,"%s %f %f\n",string,Absarray[0],Scattcoeff1);
Diffarray[0]=1/(3*(Absarray[0]+Scattcoeff1));
Dcon=Diffarray[0];
printf("Diffc=%f  Abscoeff=%f\n",Dcon,Absarray[0]);


fscanf(infile,"%s %d",string,&Regions);

/* Type 0 = Box; 1 = Sphere; 2 = Cylinder */
if(Regions>0){
        for(i=1;i<=Regions;i++){
                fscanf(infile,"%s %f %f",string,&Absarray[i],&fdump);
                Diffarray[i]=1/(3*(Absarray[i]+fdump));
                fscanf(infile,"%s %d",string,&Type);
                if(Type==0){
                /* Box : j, k, l, m, n, o = xmin, ymin, zmin, xmax, ymax, zmax */
                        fscanf(infile,"%s %d %d %d %d %d
                                        %d",string,&j,&k,&l,&m,&n,&o);
                        for(X=j;X<=m;X++)
```

```
                        for(Y=k;Y<=n;Y++)
                            for(Z=l;Z<=o;Z++)
                                in[X][Y][Z]=(char)(16*i+i);
                }
                if(Type==1){
            /* Sphere : j, k, l, m = x, y, z, r */
                    fscanf(infile,"%s %d %d %d %d",string,&j,&k,&l,&m);
                    for(X=j-m;X<=j+m;X++)
                        for(Y=k-m;Y<=k+m;Y++)
                            for(Z=l-m;Z<=l+m;Z++)
                                if((int)sqrt(pow((double)(X-j),2)+pow
                                  ((double)(Y-k),2)+pow((double)(Z-l),2))<=m)
                                in[X][Y][Z]=(char)(16*i+i);
                }
                if(Type==2){
            /* Cylinder : fdump2, k, fdump = x, z, r */
                fscanf(infile,"%s %f %d %f",string,&fdump2,&k,&fdump);
                m=(int)ceil(fdump);
                j=(int)(fdump2+1);
                xmin=j-m;
                xmax=j+m;
                zmin=k-m;
                zmax=k+m;
                for(X=j-m;X<=j+m;X++)
                    for(Y=0;Y<=MAXY;Y++)
                        for(Z=k-m;Z<=k+m;Z++)
                            if(sqrt(pow((double)(X-fdump2),2)
                            +pow((double)(Z-k),2))<=fdump){
                                if(Y==3) printf("%d %d\t",X,Z);
                                    in[X][Y][Z]=(char)(16*i+i);
                            }
                }
        }
}

fscanf(infile,"%s %f",string,&dX);
fprintf(ftarr,"%s %f, ",string,dX);

Inpoint=(int)(1.5+(1/Scattcoeff1)/dX);
printf("In point=%d\n",Inpoint);

fscanf(infile,"%s %f",string,&dt);
fprintf(ftarr,"%s %f\n",string,dt);
dt*=1e-12;
printf("Delta t : %e s.\n",dt);

fscanf(infile,"%s %f",string,&Timediff);
fprintf(ftarr,"%s %f\n",string,Timediff);
Timediff*=1e-12;
printf ("TIMEDIFF = %e\n",Timediff);
```

```
/* FIRST TWO SOURCES : */
fscanf(infile,"%s %d %d",string,&LiXmin,&LiXmax);
if((LiXmin>LiXmax)||(LiXmin<0)||(LiXmax>MAXX))
        goexit("Exit8 : LightinX");

fscanf(infile,"%s %d %d",string,&LiYmin,&LiYmax);
if((LiYmin>LiYmax)||(LiYmin<0)||(LiYmax>MAXY))
        goexit("Exit9 : LightinY");

fscanf(infile,"%s %d %d",string,&LiXmin2,&LiXmax2);
if((LiXmin2>LiXmax2)||(LiXmin2<0)||(LiXmax2>MAXX)) goexit("Exit8 :
LightinX2");

fscanf(infile,"%s %d %d",string,&LiYmin2,&LiYmax2);
if((LiYmin2>LiYmax2)||(LiYmin2<0)||(LiYmax2>MAXY)) goexit("Exit9 :
LightinY2");

fscanf(infile,"%s %f",string,&Intensity);
printf("Xmin,Xmax : %d %d \n",LiXmin, LiXmax);
printf("Ymin,Ymax : %d %d \n",LiYmin, LiYmax);
printf("Xmin2,Xmax2 : %d %d \n",LiXmin2, LiXmax2);
printf("Ymin2,Ymax2 : %d %d \n",LiYmin2, LiYmax2);
printf("Intensity : %f \n",Intensity);
if (Intensity!=0){
        for (X=LiXmin;X<=LiXmax;X++)
                for (Y=LiYmin;Y<=LiYmax;Y++)
                        mx[X][Y][Inpoint][0]+=Intensity;
        printf("\n FIRST SOURCE 1 ON!\n\n");

        for (X=LiXmin2;X<=LiXmax2;X++)
                for (Y=LiYmin2;Y<=LiYmax2;Y++)
                        mx[X][Y][Inpoint][0]+=Intensity;
        printf("\n FIRST SOURCE 2 ON!\n\n");
}
/* SECOND TWO SOURCES : */
fscanf(infile,"%s %d %d",string,&LiXmin,&LiXmax);
if((LiXmin>LiXmax)||(LiXmin<0)||(LiXmax>MAXX))
        goexit("Exit8 : LightinX");
fscanf(infile,"%s %d %d",string,&LiYmin,&LiYmax);
if((LiYmin>LiYmax)||(LiYmin<0)||(LiYmax>MAXY))
        goexit("Exit9 : LightinY");
fscanf(infile,"%s %d %d",string,&LiXmin2,&LiXmax2);
if((LiXmin2>LiXmax2)||(LiXmin2<0)||(LiXmax2>MAXX))
        goexit("Exit8 : LightinX2");
fscanf(infile,"%s %d %d",string,&LiYmin2,&LiYmax2);
if((LiYmin2>LiYmax2)||(LiYmin2<0)||(LiYmax2>MAXY))
        goexit("Exit9 : LightinY2");

fscanf(infile,"%s %f",string,&Intensity);
printf("Xmin,Xmax : %d %d \n",LiXmin, LiXmax);
```

```
printf("Ymin,Ymax : %d %d \n",LiYmin, LiYmax);
printf("Xmin2,Xmax2 : %d %d \n",LiXmin2, LiXmax2);
printf("Ymin2,Ymax2 : %d %d \n",LiYmin2, LiYmax2);
printf("Intensity : %f \n",Intensity);
if ((Intensity!=0)&&(Timediff==0)){
        for (X=LiXmin;X<=LiXmax;X++)
                for (Y=LiYmin;Y<=LiYmax;Y++)
                        mx[X][Y][Inpoint][0]+=Intensity;
        printf("\nSECOND SOURCE 1 ON !!\nTIMEDIFF=0\n");
        for (X=LiXmin2;X<=LiXmax2;X++)
                for (Y=LiYmin2;Y<=LiYmax2;Y++)
                        mx[X][Y][Inpoint][0]+=Intensity;
        printf("\nSECOND SOURCE 2 ON !!\nTIMEDIFF=0\n");
        First=0;
}

fscanf(infile,"%s %f",string,&BoundCondX);
fscanf(infile,"%s %f",string,&BoundCondY);
fscanf(infile,"%s %f",string,&BoundCondZ1);
fscanf(infile,"%s %f",string,&BoundCondZ2);

fscanf(infile,"%s %d",string,&Niter);
printf("Number of iterations is %d.\n",Niter);

/* DETECTION POINTS : */
fscanf(infile,"%s %d %d %d",string,&Xpoint1,&Ypoint1,&Zpoint1);
fscanf(infile,"%s %d %d %d",string,&Xpoint2,&Ypoint2,&Zpoint2);
fscanf(infile,"%s %d %d %d",string,&Xpoint3,&Ypoint3,&Zpoint3);
fscanf(infile,"%s %d %d %d",string,&Xpoint4,&Ypoint4,&Zpoint4);
fscanf(infile,"%s %d %d %d",string,&Xpoint5,&Ypoint5,&Zpoint5);
fscanf(infile,"%s %d %d %d",string,&Xpoint6,&Ypoint6,&Zpoint6);
fscanf(infile,"%s %d %d %d",string,&Xpoint7,&Ypoint7,&Zpoint7);
fscanf(infile,"%s %d %d %d",string,&Xpoint8,&Ypoint8,&Zpoint8);
fscanf(infile,"%s %d %d %d",string,&Xpoint9,&Ypoint9,&Zpoint9);
fscanf(infile,"%s %d %d %d",string,&Xpoint10,&Ypoint10,&Zpoint10);

Constant1=(float)C*dt/(3*dX*dX);
Constant2=(float)C*dt/6;

Number=0;
Time=0;
TimeNow=0;
TimeNext=1;

outarr1=fopen("barr1.dat","wt");
fprintf(outarr1,"time\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t", Xpoint1, Ypoint1,
        Zpoint1, Xpoint2, Ypoint2, Zpoint2, Xpoint3, Ypoint3, Zpoint3);
fprintf(outarr1,"%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t", Xpoint4, Ypoint4, Zpoint4,
        Xpoint5, Ypoint5, Zpoint5, Xpoint6, Ypoint6, Zpoint6);
```

```
fprintf(outarr1,"%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\n", Xpoint7,
        Ypoint7, Zpoint7, Xpoint8, Ypoint8, Zpoint8, Xpoint9, Ypoint9, Zpoint9,
        Xpoint10, Ypoint10, Zpoint10);

fprintf(ftarr,"f\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t",
        Xpoint1, Ypoint1, Zpoint1, Xpoint2, Ypoint2, Zpoint2, Xpoint3, Ypoint3,
        Zpoint3, Xpoint4,Ypoint4,Zpoint4,Xpoint5,Ypoint5,Zpoint5);
fprintf(ftarr,"%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t%d,%d,%d\t",
        Xpoint6, Ypoint6, Zpoint6, Xpoint7, Ypoint7, Zpoint7, Xpoint8, Ypoint8,
        Zpoint8, Xpoint9, Ypoint9, Zpoint9, Xpoint10,Ypoint10, Zpoint10);
fprintf(ftarr,"%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\n", Xpoint1, Xpoint2,
        Xpoint3, Xpoint4, Xpoint5, Xpoint6, Xpoint7, Xpoint8,Xpoint9,Xpoint10);

fclose(infile);
_strtime( string );
printf( "Start time:\t\t\t\t%s\n", string );
starttime=time(NULL);
data1=malloc(Niter * sizeof(float));
data2=malloc(Niter * sizeof(float));
data3=malloc(Niter * sizeof(float));
data4=malloc(Niter * sizeof(float));
data5=malloc(Niter * sizeof(float));
data6=malloc(Niter * sizeof(float));
data7=malloc(Niter * sizeof(float));
data8=malloc(Niter * sizeof(float));
data9=malloc(Niter * sizeof(float));
data10=malloc(Niter * sizeof(float));

do{

        Number++;
        for (X=1;X<MAXX;X++)
        for (Y=1;Y<MAXY;Y++){
              mx[X][Y][0][TimeNow]=BoundCondZ1*mx[X][Y][1][TimeNow];
              mx[X][Y][MAXZ][TimeNow]=BoundCondZ2*mx[X][Y][MAXZ-1]
                                                        [TimeNow];
        }

        for (X=1;X<MAXX;X++)
              for (Z=1;Z<MAXZ;Z++){
                    mx[X][0][Z][TimeNow]=mx[X][1][Z][TimeNow];
                    mx[X][MAXY][Z][TimeNow]=mx[X][MAXY-1][Z]
                                                        [TimeNow];
         }

        for (Z=1;Z<MAXZ;Z++)
              for (Y=1;Y<MAXY;Y++){
                    mx[0][Y][Z][TimeNow]=mx[1][Y][Z][TimeNow];
                    mx[MAXX][Y][Z][TimeNow]=mx[MAXX-1][Y][Z]
                                                        [TimeNow];
         }
```

```
for (Y=1;Y<MAXY;Y++)
        for (Z=1;Z<MAXZ;Z++){
                for (X=1;X<MAXX;X++){
                        Abscoeff=Absarray[(int)(in[X][Y][Z]/16)];
                        a[X]=-Constant1*Diffc(X-0.5,Y,Z);
                        b[X]=1+Constant1*(Diffc(X-0.5,Y,Z)
                                +Diffc(X+0.5,Y,Z)) +Constant2*Abscoeff;
                        c[X]=-Constant1*Diffc(X+0.5,Y,Z);
                        Dump=mx[X][Y][Z][TimeNow];
                        r[X]=Constant1*(
                        Diffc(X,Y+0.5,Z)*(mx[X][Y+1][Z][TimeNow]-Dump)-
                        Diffc(X,Y-0.5,Z)* (Dump-mx[X][Y-1][Z][TimeNow])+
                        Diffc(X,Y,Z+0.5)*(mx[X][Y][Z+1][TimeNow]-Dump)-
                         Diffc(X,Y,Z-0.5)*(Dump-mx[X][Y][Z-1][TimeNow]))
                        +(1-Constant2*Abscoeff)*Dump;
                }
                tridag(a,b,c,r,u,MAXX-1);
                for (X=1;X<MAXX;X++)
                mx[X][Y][Z][TimeNext]=u[X];
        }

        TimeNext=(!TimeNext);
        TimeNow=(!TimeNow);

        for (X=1;X<MAXX;X++)
        for (Y=1;Y<MAXY;Y++){
        mx[X][Y][0][TimeNow]=BoundCondZ1*mx[X][Y][1][TimeNow];
        mx[X][Y][MAXZ][TimeNow]=BoundCondZ2*
                                        mx[X][Y][MAXZ-1][TimeNow];
}

for (X=1;X<MAXX;X++)
        for (Z=1;Z<MAXZ;Z++){
                mx[X][0][Z][TimeNow]=mx[X][1][Z][TimeNow];
                mx[X][MAXY][Z][TimeNow]=mx[X][MAXY-1][Z]
                                                        [TimeNow];
        }

for (Z=1;Z<MAXZ;Z++)
        for (Y=1;Y<MAXY;Y++){
                mx[0][Y][Z][TimeNow]=mx[1][Y][Z][TimeNow];
                mx[MAXX][Y][Z][TimeNow]=mx[MAXX-1][Y]
                                                [Z][TimeNow];
                }

for (X=1;X<MAXX;X++)
        for (Z=1;Z<MAXZ;Z++){
                for (Y=1;Y<MAXY;Y++){
                        Abscoeff=Absarray[(int)(in[X][Y][Z]/16)];
```

```
                    a[Y]=-Constant1*Diffc(X,Y-0.5,Z);
                    b[Y]=1+Constant1*(Diffc(X,Y-0.5,Z)
                    +Diffc(X,Y+0.5,Z))+Constant2*Abscoeff;
                     c[Y]=-Constant1
                            *Diffc((float)X,(float)Y+0.5,(float)Z);
                    Dump=mx[X][Y][Z][TimeNow];
                    r[Y]=Constant1*(
                    Diffc(X+0.5,Y,Z)*(mx[X+1][Y][Z][TimeNow]-Dump)-
                    Diffc(X-0.5,Y,Z)*(Dump-mx[X-1][Y][Z][TimeNow])+
                    Diffc(X,Y,Z+0.5)*(mx[X][Y][Z+1][TimeNow]-Dump)-
                     Diffc(X,Y,Z-0.5)*(Dump-mx[X][Y][Z-1][TimeNow]))
                    +(1-Constant2*Abscoeff)*Dump;
                }
        tridag(a,b,c,r,u,MAXY-1);
        for (Y=1;Y<MAXY;Y++)
        mx[X][Y][Z][TimeNext]=u[Y];
 }
TimeNext=(!TimeNext);
TimeNow=(!TimeNow);

for (X=1;X<MAXX;X++)
        for (Y=1;Y<MAXY;Y++){
                mx[X][Y][0][TimeNow]=BoundCondZ1*mx[X][Y][1]
                                                        [TimeNow];
                mx[X][Y][MAXZ][TimeNow]=BoundCondZ2*mx[X][Y]
                                                [MAXZ-1][TimeNow];
        }

for (X=1;X<MAXX;X++)
        for (Z=1;Z<MAXZ;Z++){
                mx[X][0][Z][TimeNow]=mx[X][1][Z][TimeNow];
                mx[X][MAXY][Z][TimeNow]=mx[X][MAXY-1][Z]
                                                        [TimeNow];
        }

for (Z=1;Z<MAXZ;Z++)
        for (Y=1;Y<MAXY;Y++){
                mx[0][Y][Z][TimeNow]=mx[1][Y][Z][TimeNow];
                mx[MAXX][Y][Z][TimeNow]=mx[MAXX-1][Y][Z]
                                                        [TimeNow];
        }


for (Y=1;Y<MAXY;Y++)
for (X=1;X<MAXX;X++){
        for (Z=1;Z<MAXZ;Z++){
                Abscoeff=Absarray[(int)(in[X][Y][Z]/16)];
                a[Z]=-Constant1*Diffc(X,Y,Z-0.5);
                b[Z]=1+Constant1*(Diffc(X,Y,Z-
                                0.5)+Diffc(X,Y,Z+0.5))+Constant2*Abscoeff;
                c[Z]=-Constant1*Diffc(X,Y,Z+0.5);
```

```
                Dump=mx[X][Y][Z][TimeNow];
                r[Z]=Constant1*(
                Diffc(X+0.5,Y,Z)*(mx[X+1][Y][Z][TimeNow]-Dump)-
                Diffc(X-0.5,Y,Z)*(Dump-mx[X-1][Y][Z][TimeNow])+
                Diffc(X,Y+0.5,Z)*(mx[X][Y+1][Z][TimeNow]-Dump)-
                Diffc(X,Y-0.5,Z)*(Dump-mx[X][Y-1][Z][TimeNow]))
                +(1-Constant2*Abscoeff)*Dump;
        }
        tridag(a,b,c,r,u,MAXZ-1);
        for (Z=1;Z<MAXZ;Z++)
                mx[X][Y][Z][TimeNext]=u[Z];
}
TimeNext=(!TimeNext);
TimeNow=(!TimeNow);

Time+=dt;

Dump=mx[Xpoint1][Ypoint1][Zpoint1][TimeNow];
data1[Number]=Dump;
fprintf(outarr1,"%g\t%e\t",Time,Dump);

Dump=mx[Xpoint2][Ypoint2][Zpoint2][TimeNow];
data2[Number]=Dump;
fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint3][Ypoint3][Zpoint3][TimeNow];
data3[Number]=Dump;
 fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint4][Ypoint4][Zpoint4][TimeNow];
data4[Number]=Dump;
fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint5][Ypoint5][Zpoint5][TimeNow];
data5[Number]=Dump;
fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint6][Ypoint6][Zpoint6][TimeNow];
data6[Number]=Dump;
fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint7][Ypoint7][Zpoint7][TimeNow];
data7[Number]=Dump;
fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint8][Ypoint8][Zpoint8][TimeNow];
data8[Number]=Dump;
fprintf(outarr1,"%e\t",Dump);

Dump=mx[Xpoint9][Ypoint9][Zpoint9][TimeNow];
 data9[Number]=Dump;
```

```
        fprintf(outarr1,"%e\t",Dump);

        Dump=mx[Xpoint10][Ypoint10][Zpoint10][TimeNow];
        data10[Number]=Dump;
        fprintf(outarr1,"%e\n",Dump);

        printf(" Time: %E sec. Iter. no. %d ready.\n",Time,Number);
        if ((Time>=Timediff) && (First==1)){
        /* LIGHT SECOND SOURCES */
                if (Intensity!=0){
                        for (X=LiXmin;X<=LiXmax;X++)
                                for (Y=LiYmin;Y<=LiYmax;Y++){
                                        mx[X][Y][Inpoint][TimeNow]+=Intensity;
                                }
                        printf(" SECOND SOURCE 1 ON\n ");
                        for (X=LiXmin2;X<=LiXmax2;X++)
                                for (Y=LiYmin2;Y<=LiYmax2;Y++)
                                        mx[X][Y][Inpoint][TimeNow]+=Intensity;
                        printf(" SECOND SOURCE 2 ON\n ");

                }
                First=0;
        }

}while(Niter>Number);

stoptime=time(NULL);
_strtime( string );
printf( "Time:\t\t\t\t%s\n", string );

realft(data1,Niter/2,1);
realft(data2,Niter/2,1);
realft(data3,Niter/2,1);
realft(data4,Niter/2,1);
realft(data5,Niter/2,1);
realft(data6,Niter/2,1);
realft(data7,Niter/2,1);
realft(data8,Niter/2,1);
realft(data9,Niter/2,1);
realft(data10,Niter/2,1);
for (i=1;i<Niter;i+=2){
        fprintf(ftarr,"%e\t%e\t%e\t",(((i-1)/2)/(Niter*dt)),absfloat(data1[i],data1[i+1]),
                absfloat(data2[i],data2[i+1]));
        fprintf(ftarr,"%e\t%e\t%e\t%e\t",absfloat(data3[i],data3[i+1]),
                absfloat(data4[i],data4[i+1]),absfloat(data5[i],data5[i+1]),
                absfloat(data6[i],data6[i+1]));
        fprintf(ftarr,"%e\t%e\t%e\t%e\t",absfloat(data7[i],data7[i+1]),
                absfloat(data8[i],data8[i+1]),absfloat(data9[i],data9[i+1]),
                absfloat(data10[i],data10[i+1]));
```

```
            fprintf(ftarr,"%e\t%e\t%e\t%e\t",atan2(data1[i+1],data1[i]),
                    atan2(data2[i+1],data2[i]),atan2(data3[i+1],data3[i]),
                    atan2(data4[i+1],data4[i]));
            fprintf(ftarr,"%e\t%e\t%e\t%e\t",atan2(data5[i+1],data5[i]),
                    atan2(data6[i+1],data6[i]),atan2(data7[i+1],data7[i]),
                    atan2(data8[i+1],data8[i]));
            fprintf(ftarr,"%e\t%e\n",atan2(data9[i+1],data9[i]),
            atan2(data10[i+1],data10[i]));
        }
        fclose(outarr1);
        fclose(ftarr);
        free(data1);
        free(data2);
        free(data3);
        free(data4);
        free(data5);
        free(data6);
        free(data7);
        free(data8);
        free(data9);
        free(data10);

        printf("No of itr. : %ld\n",Number);
        printf("Elapsed time in seconds: %f s\n",difftime(stoptime,starttime));

        Freemx(mx,0,MAXX,0,MAXY,0,MAXZ,0);
        Freein(in,0,MAXX,0,MAXY,0);
        free_vector(a,1);
        free_vector(b,1);
        free_vector(c,1);
        free_vector(u,1);
        free_vector(r,1);
        free_vector(gam,1);
        exit(1);
}
/*************************************************************************
*                                   END OF MAIN
*************************************************************************/


float Diffc(float x,float y, float z){
        float a,b;

        if(x<xmin) return(Dcon);
        if(z<zmin) return(Dcon);
        if(x>xmax) return(Dcon);
        if(z>zmax) return(Dcon);
        a=Diffarray[(int)fmod((double)in[(int)floor(x)][(int)floor(y)][(int)floor(z)],16)];
        b=Diffarray[(int)fmod((double)in[(int)ceil(x)][(int)ceil(y)][(int)ceil(z)],16)];
        return((a+b)/2);
}
```

```
void goexit(char *outbuffer){
        printf("%s\n",outbuffer);
        free(mx);
        exit(1);
}


float maxi(float value1, float value2){
        if(value1>value2) return(value1);
        return(value2);
}


float mini(float value1, float value2){
        if(value1<value2) return(value1);
                return(value2);
}


void four1(float *data,int nn,int isign)
{
        int n,mmax,m,j,istep,i;
        double wtemp,wr,wpr,wpi,wi,theta;
        float tempr,tempi;

        n=nn << 1;
        j=1;
        for (i=1;i<n;i+=2) {
                if (j > i) {
                        SWAP(data[j],data[i]);
                        SWAP(data[j+1],data[i+1]);
                }
                m=n >> 1;
                while (m >= 2 && j > m) {
                        j -= m;
                        m >>= 1;
                }
                j += m;
        }
        mmax=2;
        while (n > mmax) {
                istep=2*mmax;
                theta=6.28318530717959/(isign*mmax);
                wtemp=sin(0.5*theta);
                wpr = -2.0*wtemp*wtemp;
                wpi=sin(theta);
                wr=1.0;
                wi=0.0;
                for (m=1;m<mmax;m+=2) {
                        for (i=m;i<=n;i+=istep) {
                                j=i+mmax;
```

```
                        tempr=wr*data[j]-wi*data[j+1];
                        tempi=wr*data[j+1]+wi*data[j];
                        data[j]=data[i]-tempr;
                        data[j+1]=data[i+1]-tempi;
                        data[i] += tempr;
                        data[i+1] += tempi;
                }
                wr=(wtemp=wr)*wpr-wi*wpi+wr;
                wi=wi*wpr+wtemp*wpi+wi;
        }
        mmax=istep;
    }
}


void realft(float *data,int n,int isign){

    int i,i1,i2,i3,i4,n2p3;
    float c1=0.5,c2,h1r,h1i,h2r,h2i;
    double wr,wi,wpr,wpi,wtemp,theta;

    theta=3.141592653589793/(double) n;
    if (isign == 1) {
        c2 = -0.5;
        four1(data,n,1);
     } else {
        c2=0.5;
        theta = -theta;
    }
    wtemp=sin(0.5*theta);
    wpr = -2.0*wtemp*wtemp;
    wpi=sin(theta);
    wr=1.0+wpr;
    wi=wpi;
    n2p3=2*n+3;

    printf("FOURIER TRANSFORM PERFORMED\n");

    for (i=2;i<=n/2;i++) {
        i4=1+(i3=n2p3-(i2=1+(i1=i+i-1)));
        h1r=c1*(data[i1]+data[i3]);
        h1i=c1*(data[i2]-data[i4]);
        h2r = -c2*(data[i2]+data[i4]);
        h2i=c2*(data[i1]-data[i3]);
        data[i1]=h1r+wr*h2r-wi*h2i;
        data[i2]=h1i+wr*h2i+wi*h2r;
        data[i3]=h1r-wr*h2r+wi*h2i;
        data[i4] = -h1i+wr*h2i+wi*h2r;
        wr=(wtemp=wr)*wpr-wi*wpi+wr;
        wi=wi*wpr+wtemp*wpi+wi;
    }
```

```
        if (isign == 1) {
                data[1] = (h1r=data[1])+data[2];
                data[2] = h1r-data[2];
        } else {
                data[1]=c1*((h1r=data[1])+data[2]);
                data[2]=c1*(h1r-data[2]);
                four1(data,n,-1);
        }
}


/*************************************************************
*     Allocate a matrix with row index from  inclusive:                 *
*************************************************************/

float  * * * *Allocmx(short nxl,short nxh,
                        short nyl,short nyh,
                        short nzl,short nzh,
                        short ntl,short nth)
{
        short i,j,k,l;
        float  * * * *m;


        m=(float  * * * *) malloc((unsigned) (nxh-nxl+1)
                                *sizeof(float  * * *));
        if (!m)
                goexit("allocation failure 1 in matrix()");
        m -= nxl;

        for(i=nxl;i<=nxh;i++) {
                m[i]=(float  * * *) malloc((unsigned) (nyh-nyl+1)
                                *sizeof(float  * *));
                if (!m[i])
                        goexit("allocation failure 2 in matrix()");
                m[i] -= nyl;
        }
        for(i=nxl;i<=nxh;i++) {
                for(j=nyl;j<=nyh;j++) {
                        m[i][j]=(float  * *) malloc((unsigned) (nzh-nzl+1)
                                        *sizeof(float *));
                        if (!m[i][j]) goexit("allocation failure 3 in matrix()");
                        m[i][j] -= nzl;
                }
        }
        for(i=nxl;i<=nxh;i++) {
                for(j=nyl;j<=nyh;j++) {
                        for(k=nzl;k<=nzh;k++) {
                                m[i][j][k]=(float  *) malloc((unsigned) (nth-ntl+1)
                                                *sizeof(float));
                                if (!m[i][j][k]) goexit("allocation failure 4 in matrix()");
```

```
                        m[i][j][k] -= ntl;
                    }
                }
        }

        for(i=nxl;i<=nxh;i++)
                for(j=nyl;j<=nyh;j++)
                        for(k=nzl;k<=nzh;k++)
                                for(l=ntl;l<=nth;l++)
                                        m[i][j][k][l] = 0.0;
        return m;
}

/*************************************************************
 *     Allocate a matrix with row index from inclusive:                    *
 *************************************************************/

char  * * *Allocin(short nxl,short nxh,
                        short nyl,short nyh,
                        short nzl,short nzh)
{
        short i,j,k;
        char  * * *m;

        m=(char  * * *) malloc((unsigned) (nxh-nxl+1)
                                *sizeof(char  * *));
        if (!m)
                goexit("allocation failure 1 in matrix()");
        m -= nxl;

        for(i=nxl;i<=nxh;i++) {
                m[i]=(char  * *) malloc((unsigned) (nyh-nyl+1)
                                *sizeof(char  *));
                if (!m[i])
                        goexit("allocation failure 2 in matrix()");
                m[i] -= nyl;
        }

        for(i=nxl;i<=nxh;i++) {
                for(j=nyl;j<=nyh;j++) {
                        m[i][j]=(char  *) malloc((unsigned) (nzh-nzl+1)
                                        *sizeof(char));
                if (!m[i][j])
                        goexit("allocation failure 3 in matrix()");
                m[i][j] -= nzl;
                }
        }

         for(i=nxl;i<=nxh;i++)
                for(j=nyl;j<=nyh;j++)
                        for(k=nzl;k<=nzh;k++)
```

```
                                m[i][j][k]= 0.0;
        return m;
}


/*************************************************************
*     Release the memory:                                    *
*************************************************************/

void Freemx(float  * * *  *m,short nxl,short nxh,
                          short nyl,short nyh,
                          short nzl,short nzh,
                          short ntl)
{
        short i,j,k;

        for(i=nxl;i<=nxh;i++)
                for(j=nyl;j<=nyh;j++)
                        for(k=nzl;k<=nzh;k++)
                                free((float *)m[i][j][k]+ntl);
        for(i=nxl;i<=nxh;i++)
                for(j=nyl;j<=nyh;j++)
                        free((float **)m[i][j]+nzl);
        for(i=nxl;i<=nxh;i++)
                free((float ***)m[i]+nyl);

        free((float ****)m+nxl);
}


void Freein(char  * *  *m,short nxl,short nxh,
                          short nyl,short nyh,
                          short nzl)
{
        short i,j;

        for(i=nxl;i<=nxh;i++)
                for(j=nyl;j<=nyh;j++)
                        free((char *)m[i][j]+nzl);
        for(i=nxl;i<=nxh;i++)
                free((char **)m[i]+nyl);
        free((char ***)m+nxl);
}

void tridag(float *a,float *b,float *c,float *r,float *u,int n){
        int j;
        float bet;

        if (b[1] == 0.0) nrerror("Error 1 in TRIDAG");
        u[1]=r[1]/(bet=b[1]);
        for (j=2;j<=n;j++) {
```

```
                gam[j]=c[j-1]/bet;
                bet=b[j]-a[j]*gam[j];
                if (bet == 0.0) nrerror("Error 2 in TRIDAG");
                        u[j]=(r[j]-a[j]*u[j-1])/bet;
        }
        for (j=(n-1);j>=1;j--)
                u[j] -= gam[j+1]*u[j+1];
}


void nrerror(char *error_text){
        void exit();

        fprintf(stderr,"Numerical Recipes run-time error...\n");
        fprintf(stderr,"%s\n",error_text);
        fprintf(stderr,"...now exiting to system...\n");
        exit(1);
}


float *vector(int nl,int nh)
{
        float *v;

        v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
        if (!v) nrerror("allocation failure in vector()");
        return v-nl;
}


void free_vector(float *v,int nl)
{
        free((char*) (v+nl));
}
```

Appendix B: Infile listing

```
XMAX_YMAX_ZMAX      84 30 20
abs_scatt_coeff.(m-1)    2 1000

Regions             1
Reg_abs_scatt_coeff 10000 1000
Type:0=B_1=Sph_2=Cyl 0
Min-x_y_z_Max-x_y_z 22 1 15 22 29 15
dX(m)               0.0025
dt(psec)            20
Timediff(psec)      2500

Light_in_X_min_max  21 21
Light_in_Y_min_max  15 15
Light_in_X2_min_max 35 35
Light_in_Y2_min_max 15 15
Light_intensity     1

Light_in_X_min_max  49 49
Light_in_Y_min_max  15 15
Light_in_X2_min_max 63 63
Light_in_Y2_min_max 15 15
Light_intensity     1

Bound_cond._X       1
Bound_cond._Y       1
Bound_cond._Z1_in  0
Bound_cond._Z2_out  0
No_of_iter          2048

Point_1_X_Y_Z       32 15 19
Point_2_X_Y_Z       34 15 19
Point_3_X_Y_Z       36 15 19
Point_4_X_Y_Z       38 15 19
Point_5_X_Y_Z       40 15 19
Point_6_X_Y_Z       42 15 19
Point_7_X_Y_Z       44 15 19
Point_8_X_Y_Z       46 15 19
Point_9_X_Y_Z       48 15 19
Point_10_X_Y_Z      50 15 19
```