Computer modelling of the temperature distribution in high intensity focused ultrasound thermotherapy

Per Axelsson

Master's Thesis
Lund Reports on Atomic Physics, LRAP-246
Lund, March 1999

together with
Dep. of Biomedical Engineering
-
Lund University

The project was supported by
NUTEK
and
Crafoord Foundation

Abstract

This Masters thesis deals with computer modelling of temperature increase in tissue generated by high intensity focused ultrasound. At the *Dep. of Biomedical Engineering - Lund University*, work on high energy focused ultrasound thermotherapy is conducted. Equipment has been developed for cancer therapy with the aim to produce a well confined tissue necrosis in a controlled predictable way. The outcome of a treatment is highly dependent on factors such as the ultrasound power, treatment time and the thermal- and acoustical properties of the target tissue. The generated temperature is expressed by the bioheat transfer equation where the generation and dissipation of heat is governed by the properties above. In this thesis, a computer program has been developed where the bioheat transfer equation is solved numerically. The program allows parametric studies where the treatment outcome can be studied when various parameters are altered. The validity of the solutions produced by the program has been investigated and the agreement between an *in vitro* experiment and the corresponding computer solution has proved to be very good.

Contents

1 Introduction

Cancer treatment by heat can be categorized into two groups, hyperthermia and thermotherapy, depending on the generated temperature. In thermotherapy, high temperatures are employed which give rise to rapid tissue necrosis, while hyperthermia is characterized by lower temperatures and is often combined with other modalities such as chemotherapy. The outcome of a thermotherapy treatment is highly dependent on factors such as the generated tissue temperature and treatment time etc. There are a number of different ways to generate heat in tumors, such as by laser, microwave, hot water or ultrasound etc. The latter method has the advantage that tissue necrosis can be achieved in deep seated tumors in a non-invasive way.

At the *Dep. of Biomedical Engineering - Lund University*, a HIFU (high intensity focused ultrasound) system is under development. The system is mainly intended to be used for thermotherapy of breast cancer and consists of a *treatment head*, power amplifier and control electronics. The power amplifier is connected to the treatment head which generates the HIFU and each of the three currently existing treatment heads can be equipped with a diagnostic sector-scanner for monitoring purposes. During a treatment, the only feedbacks are the size of the coagulated region and the blood flow detected by the diagnostic equipment.

Since the system lacks any tissue temperature feedback, the treatment must be well planned with respect to dosimetry in order to avoid mistreatment. Consequently, this Masters thesis was outlined with the objective to develop a computer model that gives valuable information such as the temperature distribution and the coagulated volume following a thermotherapy treatment.

The computer program numerically solves the bioheat transfer equation and can handle both homogenous and layered (skin, fat, muscle) tissue models. Additionally, the models are treated as cylindrical-symmetrical and the domain, in which the equation is solved, is divided into a large number of voxels (volume elements). The time is also discretisized and in each time step, steady-state heat flow is assumed. The driving force in the bioheat transfer equation is the heat generated by the HIFU and a second computer program has been developed to calculate the heat source, which is subsequently used in the simulation program, from measured sound intensity maps.

First, a brief description of the concept of cancer is presented followed by the mechanism behind heat damage of tissue and some characteristics of HIFU. Secondly, basic heat diffusion theory is presented together with the numerical method. Finally, the validity of the models (the homogenous model in particular) is tested and compared to experimental results. Detailed manuals of the two programs are found in appendix E.

2 Cancer and heat

The human body consists of a variety of celltypes, which normally proliferate and die in a controlled, preprogrammed fashion but sometimes a cell divides incorrectly. Often the body can handle such mistakes and kills the faulty cell but if the cell survives and starts to reproduce the beginning of a tumor is a fact. If the abnormal cells are able to invade other tissue types and form metastases the tumor is malignant. Otherwise the tumor is called benign.

All cells in the body, normal as abnormal, are sensitive to heat. Therefore a widespread approach to treat tumors is to expose the tumor cells to heat. The heat can be induced in different ways, for example by laser, microwaves and HIFU. It has been shown that it is rather the "thermal history" or "thermal dose" (i.e. time vs temperature) that kills the cell than the peak temperature [1]. There are many complex reactions involved, that are not fully understood, when a cell is exposed to heat. Some of them are reversible but together they can cause irreversible damage i.e. cell death [1]. The type of reactions involved depend on the induced temperature. Temperature between 41-47°C is called "hyperthermia" and over 50°C "thermotherapy" [2]. The major drawback of hyperthermia is the difficulty to confine the heat inside the tumor during the whole treatment to prevent damage of normal tissue.

2.1 Hyperthermia

During a hyperthermic treatment the temperature in the target tissue (the tumor) is rised and maintained at a constant level between 41-47°C. Since the complex cell-killing mechanism depends on the thermal dose, the treatment time spans from a few minutes up to several hours [3].

Hyperthermia can cause or promote cell death in basically three different ways [2]: direct cell death (denaturation of proteins, deactivation of enzymes etc.), indirect cell death (destruction of the microvasculature) and finally, gained sensitivity to other treatment modalities (chemotherapy, ionizing radiation, photodynamic therapy etc.). Some cancer cells are more sensitive to heat in the range 42-44°C [2] than normal tissue. The increase of sensitivity to heat is due to many factors. One of the most significant factors is low pH which in turn arise from inadequate microcirculation [4], among other things. The low microcirculation promotes the treatment in an additional sense; since the heat convection becomes lower it makes it somewhat easier to confine the heat to the tumor.

2.2 Thermotherapy

As mentioned before, thermotherapy is associated with temperatures above 50°C. The main cell-killing mechanism at this temperatures is denaturation of proteins (coagulation) [2] which can be recognized macroscopically as whitening of the tissue. At this temperature all cells, normal as abnormal, are killed at the same high rate which makes it even more important to confine the heat within the tumor. This is possible since the heat transfer in tissue is a relatively slow process and the high temperature implies short treatment time (cf. thermal dose). Further, thermotherapy can be used as a stand-alone treatment modality to cancer [5][6][7].

If the generated temperature exceeds 100°C the water in the cells starts to boil and microbubbles of vapor are formed, which beside disrupting the tissue also affect the ultrasound penetration. The effects of the microbubbles on tissue are difficult to predict and therefore temperatures below 100°C are preferred in thermotherapy.

2.3 Thermal dose

It is commonly established that heat damage to tissue can be described by:

$$\Omega(t) = P \int_0^t e^{-\Delta E_a / RT(\tau)} \, d\tau \qquad (1)$$

where Ω [dimensionless] is the accumulated heat damage, R is the universal gas constant (=8.314 J mole^{-1} K^{-1}) and $T(\tau)$ [K] is the temperature at time τ. Equation 1 has its origin in an Arrhenius rate process equation where ΔE_a [J mole^{-1}] is the activation energy and P [s^{-1}] is the frequency factor. Henriques [1] used a similar expression to predict the time-temperature dependence of the damage to pig epidermis. The values of P and ΔE_a must be determined empirically depending on the type of tissue studied and the desired endpoint. Henriques found that his chosen endpoint, transepidermal coagulation necrosis, was reached when the value of Ω equalled 1 and P was equal to $3.1*10^{98}$ s^{-1} and ΔE_a was equal to 150 kcal mole^{-1}. The time-temperature dependence of equation 1 when Ω=1 is presented in figure 1.

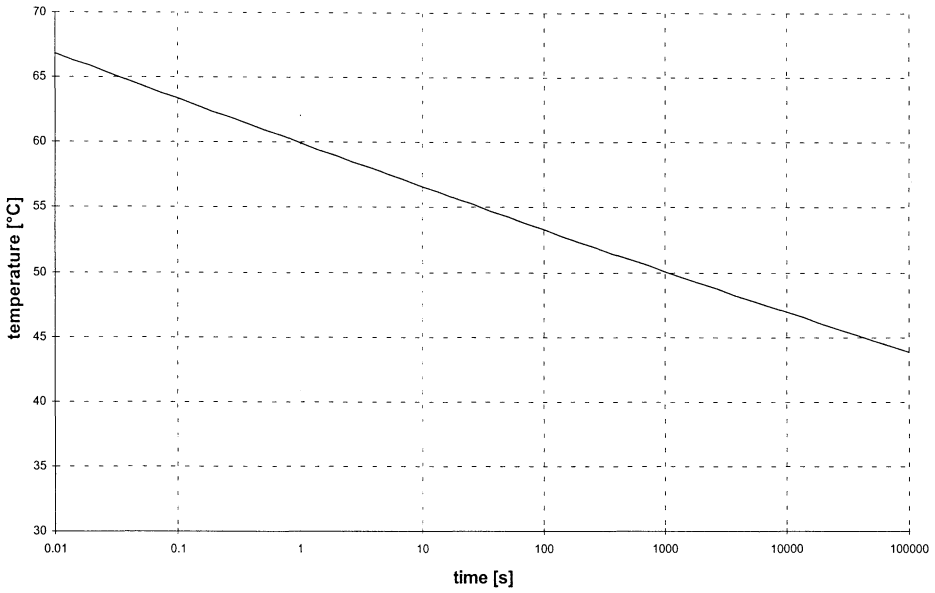

Figure 1. Time-temperature dependence of Eq. 1 when Ω=1, i.e. when transepidermal coagulation necrosis is reached according to Henriques [1].

The supply of values of P and ΔE_a is meager so the values given above will therefore be used in this thesis. Although the values don't correspond to the type of tissue studied within this thesis they have proven to give sufficiently accurate results when employed in studies of porcine muscle and human prostate [2].

3 Ultrasound

Ultrasound is defined as a mechanical wave with frequency above 20 kHz. This wave can be either longitudinal or transversal in nature. Longitudinal waves are dominating in non-solids such as gas, liquid or soft tissue.

Therapeutic US (ultrasound) can be classified as either low intensity (0.125-3.0 W cm^{-2} SATA, spatial average temporal average) or high intensity (>5.0 W cm^{-2} SATA) [8]. At low intensity non-thermal effects are dominating while thermal effects (apart from cavitation[1]) dominate at higher intensities.

3.1 Ultrasonic heat generation

When an ultrasonic wave propagates in a medium, such as soft tissue, it will be attenuated. The attenuation is due to the viscosity of the medium and the absorbed ultrasonic energy is transformed into heat. This phenomenon is used when localized heat generation in hyperthermia or thermotherapy is desired. For a plane US wave the intensity as a function of depth (z) is expressed as:

$$I(z) = I_0 e^{-2\alpha \cdot z} \qquad [\mathrm{W\ m^{-2}}] \qquad (2)$$

where I_0 is the intensity at z=0 m and α [Np m^{-1}] is the amplitude attenuation coefficient. The absorbed volumetric power is expressed as:

$$P(z) = 2\alpha \cdot I_0 e^{-2\alpha \cdot z} \qquad [\mathrm{W\ m^{-3}}] \qquad (3).$$

The amplitude attenuation coefficient α is dependent on the type of tissue, temperature and US frequency. Generally, α becomes larger at higher frequencies but is henceforth considered as dependent only on the type of tissue. Typical values of α for some tissue types are found in table 1.

[1] Cavitation is the formation of small microbubbles of gas induced by the depression in an ultrasonic wave.

Table 1. Amplitude attenuation-coefficient for some tissue types (according to [9]).

tissue	α [Np cm^{-1}]	**frequency** [MHz]
skin	0.40	1.0
fat	0.21	1.0
muscle	0.32	2.0
breast	0.06-0.13	1.76
(water	0.00064	1.6)

The frequency of therapeutic US range from 0.2 MHz to 3 MHz with an optimum in tissue penetration vs delivered energy at 1 MHz [10].

It is readily understood that for a plane wave most of the energy is absorbed in the vicinity of the surface (see Fig. 2), which makes it difficult to treat deep seated tumors without damaging the intervening tissue.

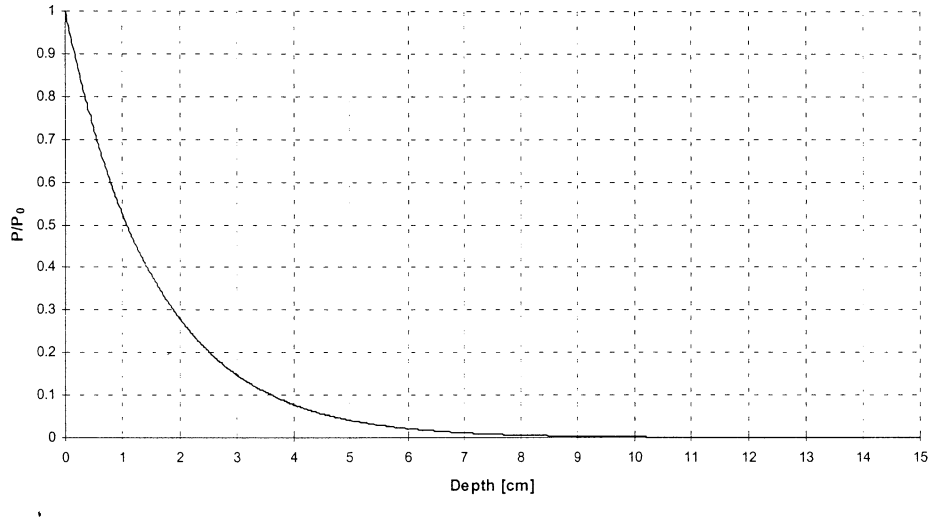

Figure 2. Absorbed power, due to a plane US-wave, as a function of depth. Amplitude attenuation-coefficient equal to 0.32 Np cm^{-1}.

In order to attain a spatially well defined hot-spot at large depths in tissue, focusing of the ultrasound is necessary. Focusing can be achieved in a number of different ways, such as with a *phased array* but the most reliable and inexpensive method is to use a spherically curved US transducer (Fig. 3).

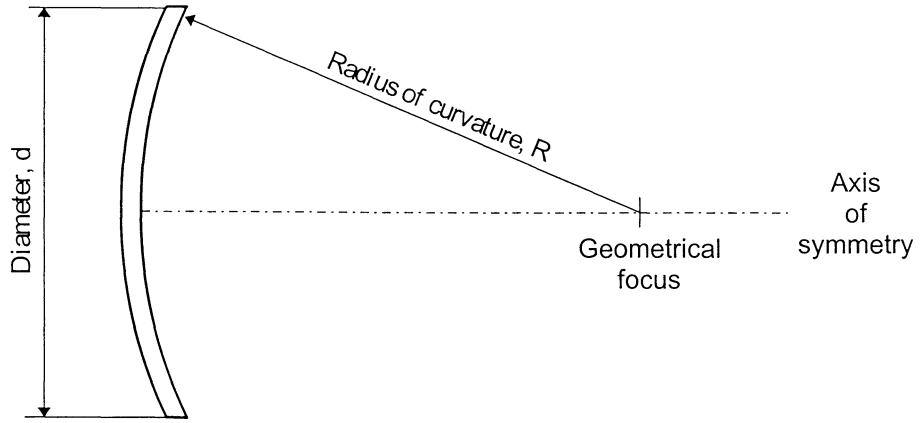

Figure 3. Schematic figure of a spherically curved transducer.

Depending on the radius of curvature, the US frequency (or the wavelength) and the diameter, the transducer can be classified as either weak, medium or strongly focused [11]. Figure 4 shows the calculated intensity distribution along the axis of symmetry of a single transducer with a diameter (d) equal to 5 cm, radius of curvature (R) equal to 7 cm and frequency (f) equal to 1.6 MHz.

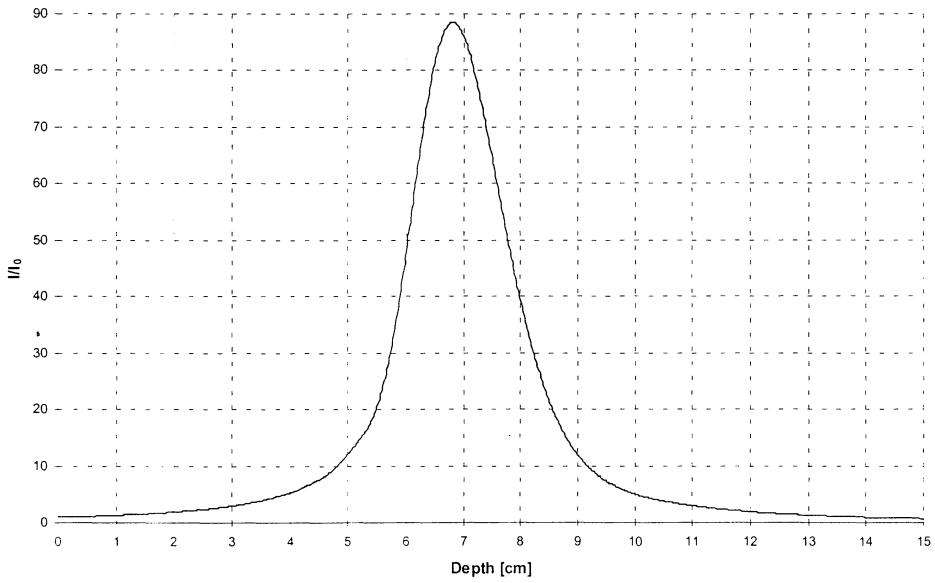

Figure 4. Calculated axial intensity distribution of a spherically curved transducer in water with cosinus-like velocity distribution over its surface. The transducer is strongly focused.

It is well-known (and not explained here) that the thermal focus and the geometrical focus of a single transducer don't coincide (Fig. 4). This deviation becomes less significant when several transducers are mounted in such a way that their geometrical foci coincide. It is then possible to attain a hot-spot only a few millimeters wide at large depths in tissue. The intensity gain of such a system, called *treatment head*, at the thermal focus relatively to the intensity at the surface of the transducers can be 800-900 [6].

A commonly used protocol in HIFU investigations on tissue coagulation (thermotherapy) is to alternately switch on and off the US with a certain burst duration and repetition period. Albu Barkman [6] has concluded that for the system developed at the department[2], a burst duration of 1.3 s and a repetition period of 10 s gives the best result.

[2] Department of Biomedical Engineering - Lund University.

4 Heat transfer in tissue

There are three fundamental ways by which heat can be transferred in tissue: conduction, convection and radiation. Common for these three ways are, according to fundamental laws of thermodynamics, that heat flows from hotter parts in the medium to cooler ones. Radiation is neglected in later parts of this thesis due to its small significance. The remaining two form the ground in the BHTE (bioheat transfer equation) where heat generation is also included.

4.1 Conduction

Thermal conduction in solids (e.g. tissue) is governed by Fourier's law [12], which states that the heat flow rate per unit area at a given point, $\boldsymbol{q}(\boldsymbol{r},t)$ [W m^{-2}], is proportional to the temperature gradient at that point:

$$\boldsymbol{q}(\boldsymbol{r},t) = -\lambda \nabla T(\boldsymbol{r},t) \qquad [\text{W m}^{-2}] \qquad (4).$$

The proportionality coefficient, λ [W m^{-1} K^{-1}], is called the thermal conductivity and is henceforth considered as constant and isotropic. The thermal conductivity indicates the ability to transfer heat i.e. if λ is large, the medium is a good heat conductor. Characteristic values of λ for some tissue types are found in table 2.

Table 2. Thermal conductivity for some tissue types (according to [9]).

tissue	λ [W m^{-1} K^{-1}]
skin	0.293
fat	0.23-0.27
muscle	0.449-0.546
breast	0.499
(water	0.60)

The heat transfer in an infinite homogenous isotropic solid due to conduction can be formulated as [12]:

$$\rho c \frac{\partial T(\boldsymbol{r},t)}{\partial t} = \nabla(\lambda \nabla T(\boldsymbol{r},t)) \overset{\lambda \text{ const.}}{=} \lambda \Delta T(\boldsymbol{r},t) \quad \Leftrightarrow$$

$$\Leftrightarrow \quad \frac{\partial T(\boldsymbol{r},t)}{\partial t} = a \Delta T(\boldsymbol{r},t) \qquad [\text{W m}^{-3}] \qquad (5)$$

where ρ [kg m^{-3}] is the local density and c [J kg^{-1} K^{-1}] is the local specific heat capacity. The quantity a ($\equiv \lambda/\rho c$) [m^2 s^{-1}] known as thermal diffusivity indicates the dynamical behavior of the heat conduction i.e. if a is large, the heat transfer is fast. The thermal diffusivity is, in conformity with λ, considered as constant and isotropic.

4.2 Convection

Convection of heat means that the medium itself flows from hotter parts to cooler ones. This heat transfer mode is thus appreciable only when the medium is a liquid or a gas. In this thesis, convection is governed by the presence of blood flow. There exists a number of different ways to model the heat transfer by blood flow [2]; one of these is the *heat sink model.* The heat sink approach is not the most accurate one in presence of large bloodvessels but is simple to implement and justified when the blood flow is almost isotropic i.e. in capillarized tissue. The heat sink model states that blood at arterial temperature T_a [K] enters the tissue where it interacts and exits at the local temperature T [K]. Thus, the convective heat, Q_b [W m^{-3}], either produced or consumed is expressed as:

$$Q_b = \omega_b \rho_b c_b \rho (T_a - T) \qquad [\text{W m}^{-3}] \qquad (6)$$

where ω_b [m^3 kg^{-1} s^{-1}] is the specific blood perfusion rate, ρ_b [kg m^{-3}] is the blood density, c_b [J kg^{-1} K^{-1}] is the specific heat capacity of blood and ρ [kg m^{-3}] is the local density of the surrounding tissue. If convection is taken into account, Eq. 5 can be rewritten as:

$$\rho c \frac{\partial T}{\partial t} = \lambda \Delta T + \omega_b \rho_b c_b \rho (T_a - T) \qquad [\text{W m}^{-3}] \qquad (7).$$

In the literature ω_b is often expressed in [ml $(100g)^{-1}$ min^{-1}], that is, blood volume per 100g tissue and minute and a representative value of blood flow in muscle tissue is 6 ml $(100g)^{-1}$ min^{-1} (rat muscle tissue).

4.3 Bioheat transfer equation

In order to fully model the heat transfer in tissue one must take heat generation into account. The heat generation is introduced by adding a source term, Q_s [W m^{-3}], in Eq. 7 and the result is the BHTE:

$$\rho c \frac{\partial T}{\partial t} = \lambda \Delta T + \omega_b \rho_b c_b \rho (T_a - T) + Q_s \qquad [\text{W m}^{-3}] \qquad (8).$$

The heat source term itself consists of two separable parts: Metabolic heat generation (Q_m) which represents the heat generation due to metabolism and external heat generation (Q_s) which, within the context of this thesis, represents the heat generation due to absorption of HIFU. Since the metabolic heat generation is several orders of magnitude smaller than the external counterpart, it is often dropped out [10].

5 Numerical model

An analytical solution to the BHTE (Eq. 8) is obtainable only in very special cases, such as when Q_s is a point- or linesource and the medium is infinite or semi-infinite. In any other case the BHTE must be solved numerically. The numerical method employed here is based on finite differences of the temporal- and spatial derivatives. This approach allows temporal-, spatial- and temperature dependency in any of the coefficients and terms in the BHTE. Further, the model is treated as cylindrical-symmetrical and the BHTE is thus solved using cylindrical coordinates, where the simulation domain (radial- and axial extension) is divided into small volume-elements (voxels) and the time is discretisized (Fig. 5). The bounded spatial extension of the models implies that boundary conditions also must be accounted for. The program that performs the simulation, *Tempsim*, can handle both homogenous and layered (skin, fat, muscle) models. Prior to any simulation, the heat source must be designed and build by means of the *Sourcemaker* program.

The following subsections describe how the two programs work and what approximations or assumptions are made. The source codes, in C, to the two programs are found in appendix D.

5.1 Building the heat source

The heat source term Q_s [W m^{-3}] represents, as mentioned before, the heat generation due to absorption of HIFU. Since the ultrasonic intensity distribution of a treatment head is rather complex, the building of the heat source is based on an intensity map in the radial- and axial direction. The intensity mapping was performed at the *Dep. of Electrical Measurements*[3] *- Lund Institute of Technology,* on each of the three currently existing treatment heads. Detailed features of the three treatment heads and an extract from an intensity map file are found in appendix A and B respectively.

[3] F. Nordbladh, "*Implementering av ett användargränssnitt till ett hydrofonmätsystem*", Report 07/97 (Masters thesis), Dep. of Electrical Measurements, Lund Institute of Technology, 1997.

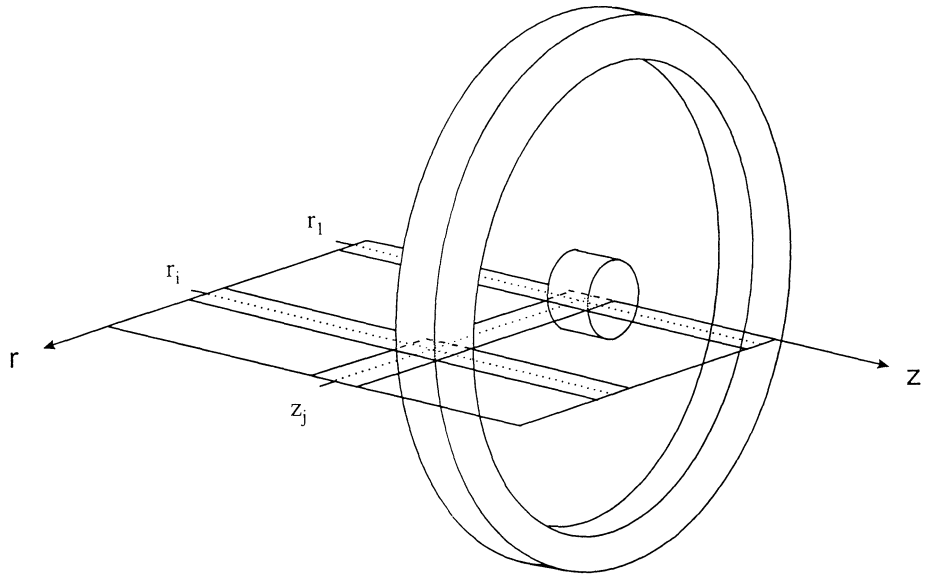

Figure 5. Schematic figure of the division of the simulation domain into volume elements, voxels.

The mapping system uses a high-resolution hydrophone to measure the relative sound intensity in water along a predefined pattern. In each measurement, the total electrical power into the treatment head was less than 500 mW intermittently. The mapped areas were oriented in such a way that the plane containing the mapped area intersects the center of one of the transducers in the treatment head (see Fig. 6). This orientation was chosen since the intensity along the axis of symmetry of the intersected transducer (cf. Fig. 4) is included in the map. The three intensity maps (saved in ASCII-format) are used, one at a time, as an input to the *Sourcemaker* program that was developed to build the heat source file.

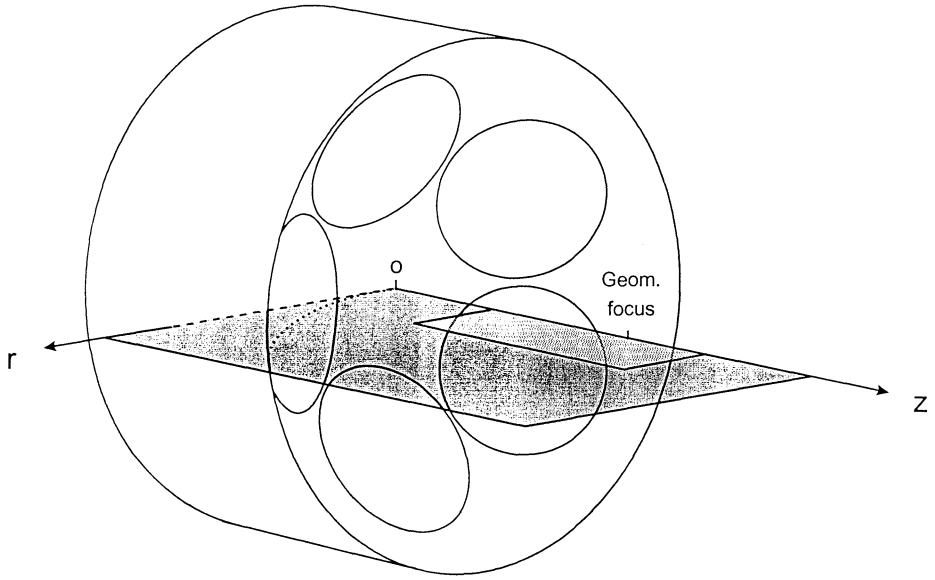

Figure 6. Schematic figure of the orientation of the plane (dark gray) containing the mapped area (light gray).

5.1.1 Sourcemaker program at a glance

The *Sourcemaker* program is designed to calculate and provide the temperature simulation program, *Tempsim*, with the heat source.

Once the program is launched, it asks for an intensity map file and an acoustical- and thermal properties file. Beside these files, the program requires further information, such as the size of the mapped area, the desired size of the simulation domain and the desired depth of the geometrical focus in the model. Prior to this, the user must choose whether the model should be homogenous or layered (see Fig. 7).

The program reads the mapped intensity from the input file and builds a matrix (called input matrix) where each element is filled in with the mapped US intensity at that coordinate. A second, empty, matrix (called output matrix) is also created. Then the program reads the tissue acoustical- and thermal properties from the other file. Two typical tissue parameter files are found in appendix B.

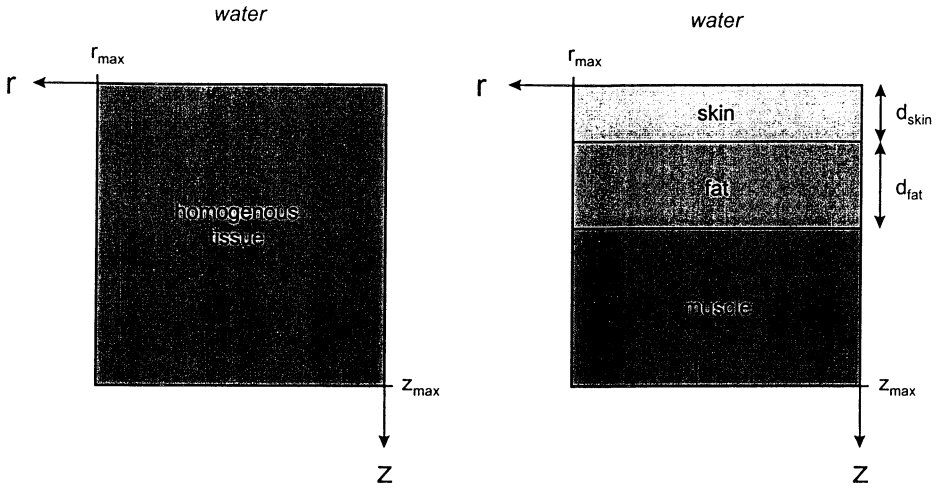

Figure 7. Schematic figure showing the two different tissue models and relevant features.

At this point, the program enters the real heat source building part. The output matrix is filled in, element by element, with the absorbed volumetric power (cf. Eq. 3) at that particular coordinate calculated via bi-linear interpolation of the sound intensity in the input matrix (see Fig. 8). The loss due to reflection in the tissue interfaces is also accounted for. An universal boundary condition at the tissue interface at z=0 m is that the tissue is in contact with water.

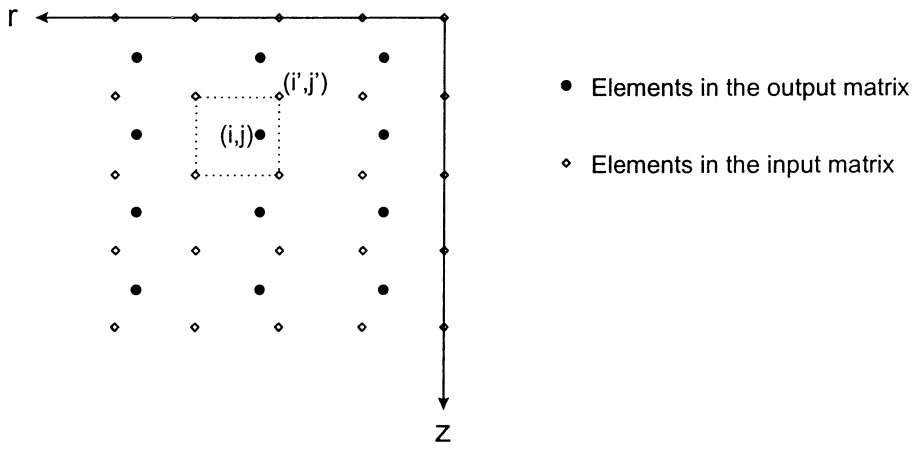

Figure 8. Schematic figure of the elements in the output matrix vs the elements in the input matrix in conjunction to their coordinates.

When the output matrix is complete, it is adjusted so that the total absorbed power in the simulation domain equals 1 W, that is, all the incident US (set to 1 W) is absorbed in the domain. This is true when the simulation domain or α is large enough. Other approximations and assumptions thus far are: The US attenuation is based on the axial depth of each voxel instead of the actual acoustic path of propagation, the intensity distribution is independent of HIFU power, no multiple reflections between the tissue layers (layered model), no scattering of the ultrasound inside the tissue, no interference phenomenon and finally, no inhomogenities, such as a tumor or a blood vessel, in the tissue.

The output matrix is finally written to a file, ready to be used in the *Tempsim* program. The file also contains some tissue properties essential to the *Tempsim* program and an extract from a heat source file is found in appendix B.

5.2 Temperature simulation

The *Tempsim* program offers wide possibilities of making parametric studies since many of the simulation parameters can be varied: the size and shape of the heat source (via *Sourcemaker*), the power of the heat source, the initial tissue temperature and the water temperature (assumed constant), the blood flow and blood temperature, the thermal- and acoustical properties of the tissue (via *Sourcemaker*), the burst duration and the repetition period and finally, the US stop time and the simulation stop time. The connections between the latter four parameters are visualized in Fig. 9.

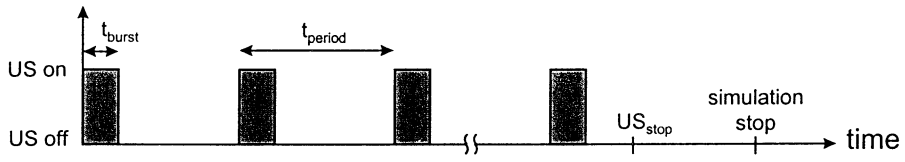

Figure 9. Schematic figure showing the temporal simulation parameters.

When the simulation is finished, the program gives the following output in text file format: the temperature distribution in the simulation domain after the simulation stop time and the excessive heat in the tissue, the coagulation distribution and the coagulated volume, the distribution of the maximum temperature in each voxel during the simulation and if the user wants, the temperature in some user defined points recorded during the simulation.

5.2.1 The Tempsim program

The general simulation idea is to calculate the net heat flow rate (as if steady-state prevails) into each voxel at time t and then increment the time by the time step Δt and update the voxel temperatures.

Once the *Tempsim* program is launched, the user is prompted to type the name of the heat source file and the desired names of the output files. The user must also provide the program with the rest of the simulation parameters (via the keyboard). If desired, a number of imaginary temperature probes can be placed anywhere within the simulation domain.

First, the source file is opened and the values are stored in a matrix denoted *source*. The program continues with some initializations where, for instance the volume of each voxel is calculated and stored in a vector denoted *vol*. When this is done, the axial- and radial heat conductances are calculated. The axial heat conductance between voxel (i,j) and (i,j-1) is calculated (with Eq. 4 in mind) with:

$$condz[i] = \frac{\lambda \cdot \pi\left((r_i + \frac{\Delta r}{2})^2 - (r_i - \frac{\Delta r}{2})^2\right)}{\Delta z} \qquad [\mathrm{W\,K^{-1}}] \qquad (9)$$

where r_i is the voxel center radius, Δr and Δz is the radial- and axial voxel size. In the case of the layered model, the above equation holds only when the voxel is completely surrounded by voxels with similar thermal conductivity. Otherwise, if the voxel is situated at any tissue interface, equation 9 is replaced by:

$$condz[i][j] = \frac{2\lambda_{j-1}\lambda_j}{\lambda_{j-1} + \lambda_j} \cdot \frac{\pi\left((r_i + \frac{\Delta r}{2})^2 - (r_i - \frac{\Delta r}{2})^2\right)}{\Delta z} \qquad [\mathrm{W\,K^{-1}}] \qquad (10)$$

where λ_{j-1} is the thermal conductivity of voxel (i,j-1) and λ_j is the thermal conductivity of voxel (i,j). When it comes to the radial heat conductance, the expression needs some mathematical manipulation since the radial part of Eq. 4 is identified as:

$$q_r(\boldsymbol{r},t) = -2\pi\lambda \cdot r \frac{\partial T(\boldsymbol{r},t)}{\partial r} \qquad [\text{W m}^{-1}] \qquad (11).$$

Equation 11 can be solved by separation of variables since q_r is independent of r:

$$q_r \frac{\partial r}{r} = -2\pi\lambda \, \partial T \qquad [\text{W m}^{-1}] \qquad (11b).$$

The solution is found when integrating Eq. 11b from r_{i-1} to r_i:

$$q_r = -\frac{2\pi\lambda \cdot \left(T(r_i) - T(r_{i-1})\right)}{\ln(\frac{r_i}{r_{i-1}})} \qquad [\text{W m}^{-1}] \qquad (12).$$

The radial heat conductance between voxel (i,j) and (i-1,j) is then identified as:

$$condr[i] = \frac{2\pi\lambda \cdot \Delta z}{\ln(\frac{r_i}{r_{i-1}})} \qquad [\text{W K}^{-1}] \qquad (13).$$

By substituting λ for λ_j in the equation above, the equation also holds for the layered model.

When the axial- and radial heat conductances are calculated, the program continues with calculating the time step, Δt, to increment the time. The approach when calculating the time step is to calculate the maximum time step allowed without letting the outward heat flow from each voxel exceed its total energy content. Expressed in mathematical terms, this is equal to:

$$\Delta t\Big(condr[i][j]\cdot(T_{i,j}-T_{i-1,j})+condr[i+1][j]\cdot(T_{i,j}-T_{i+1,j})+condz[i][j]\cdot$$

$$\cdot(T_{i,j}-T_{i,j-1})+condz[i][j+1]\cdot(T_{i,j}-T_{i,j+1})\Big)<\rho c\cdot vol[i][j]\cdot T_{i,j}+$$

$$+source[i][j]\cdot vol[i][j]\cdot\Delta t \qquad [J] \qquad (14)$$

where *vol[i][j]* is the volume [m^3] of voxel (i,j) and *source[i][j]* is the heat source [W m^{-3}] in the same voxel. Rearranging Eq. 14 yields:

$$T_{i,j}\Big(\Delta t\,(condr[i][j]+condr[i+1][j]+condz[i][j]+condz[i][j+1])-\rho c\cdot$$

$$\cdot vol[i][j]\Big)<\Delta t\,(source[i][j]\cdot vol[i][j]+condr[i][j]\cdot T_{i-1,j}+condr[i+1][j]\cdot$$

$$\cdot T_{i+1,j}+condz[i][j]\cdot T_{i,j-1}+condz[i][j+1]\cdot T_{i,j+1}) \qquad [J] \qquad (15).$$

The right-hand side of Eq. 15 is under all circumstances positive and an underestimation is done by replacing it for zero. Solving for Δt then yields:

$$\Delta t<\frac{\rho c\cdot vol[i][j]}{condr[i][j]+condr[i+1][j]+condz[i][j]+condz[i][j+1]} \qquad (16).$$

The minimum value of Δt when Eq. 16 is solved for each voxel is then used as the simulation time step. The user also can enter a desired time step that will be used as the simulation time step provided that it is smaller than the calculated one.

Every procedure described in *Tempsim* so far is executed once when the program is launched. The steps described below are repeated until the simulation time equals the simulation stop time.

First, the heat flow rates into each voxel within the simulation domain is calculated via:

$$flowr[i][j] = condr[i][j] \cdot (T_{i-1,j} - T_{i,j}) \qquad [\mathrm{W}] \qquad (17a)$$

and

$$flowz[i][j] = condz[i][j] \cdot (T_{i,j-1} - T_{i,j}) \qquad [\mathrm{W}] \qquad (17b).$$

$T_{i,j-1}$ is substituted for T_{water} at the tissue/water interface (j=1). Further, the flows over the internal boundaries are set to zero. This is true when the simulation domain is large or when λ and the simulation time are small. The flow over the boundary at r=0 m is anyhow equal to zero due to symmetry.

Then, the new voxel temperatures are calculated. The temperature increase of voxel (i,j) is calculated from the net inward heat flow rate, originating from its four nearest neighbors, multiplied by the timestep. That is, if heat generation and blood flow are included:

$$\Delta T_{i,j} = \frac{1}{\rho c \cdot vol[i][j]}\Big(flowr[i][j] - flowr[i+1][j] + flowz[i][j] -$$

$$- flowz[i][j+1] + (source[i][j] + \omega_b \rho_b c_b \rho (T_a - T[i][j])) \cdot vol[i][j]\Big)\Delta t$$

$$[\mathrm{K}] \qquad (18)$$

where ρ and c is the density and the heat capacity of voxel (i,j). The source is either "on" or "off" according to the temporal treatment parameters (cf. Fig. 9). The coagulation at time t of the tissue is also calculated; but the integral in equation 1 is approximated by a summation:

$$\Omega_{i,j}(t) \approx \sum_{k=0}^{n_{step}} P \cdot e^{-\Delta E_a / R T_{i,j}^k} \cdot \Delta t \qquad (19)$$

where n_{step} is the number of time steps taken up to time t and $T_{i,j}^k$ is the temperature of voxel (i,j) at the time $t = k \cdot \Delta t$.

At this point, the program loops back and starts to recalculate the heat flow rates, based on the new voxel temperatures, and so forth. This procedure will be

repeated until the simulation time reaches the simulation stop time. When the simulation is finished, the program writes the results to the output files mentioned in the previous section.

Approximations and assumptions made in *Tempsim*, apart from those already mentioned, are: The thermal- and acoustical properties of the tissue are constant in every sense, no US power loss due to higher harmonics, no increase of US absorption (α) in coagulated voxels, no tumors or blood vessels, but still a blood flow, no metabolism and no heat transfer via radiation.

5.3 Solution validity

To test the accuracy of the simulation program, some problems with an analytical solution were solved using the program and the simulated results were compared with the exact solutions. The thermal medium used in all the problems described below was water (ρ=1000 kg m^{-3}, c=4.18 kJ kg^{-1} K^{-1}, λ=0.6 W m^{-1} K^{-1}) and the initial temperature was set to 0°C. Note, the model used to test the solution accuracy was the homogenous model since an analytical solution is hard to find in an inhomogenous medium such as the layered model. Tests have been performed (although not presented here) on the layered model with identical parameters in all the layers which gave the same results as the homogenous model.

5.3.1 Constant surface temperature

By applying a constant temperature to the whole surface of a semi-infinite slab with uniform initial temperature, the heat is forced to flow in the axial direction only. Thus, the solution accuracy in the axial direction can be studied by means of assigning the water a temperature equal to 1°C. The temperature is measured by a number of probes placed at different depths (0.1-19.9 mm) in the simulation domain which record the temperature at different times (5-300 s) and the simulated solution is showed in Fig. 10a.

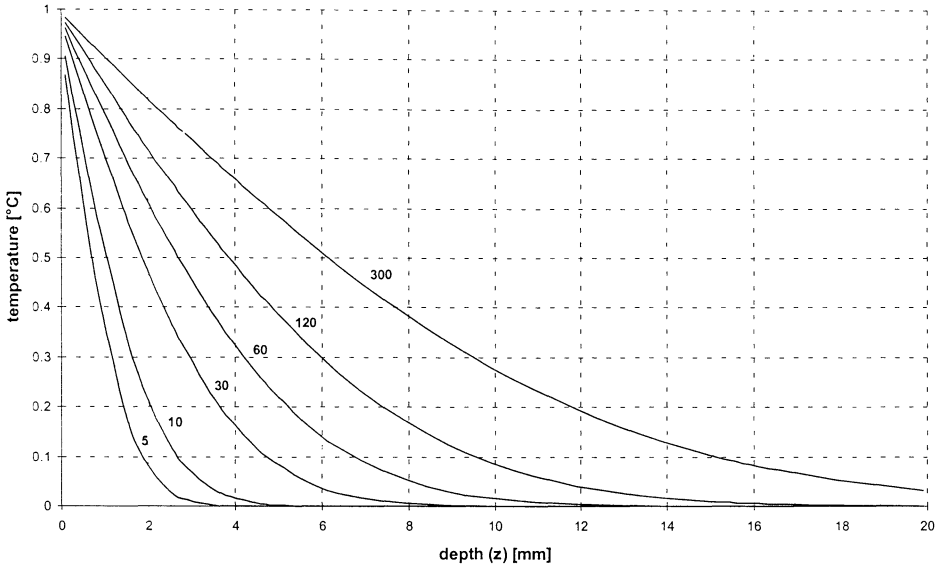

Figure 10a. The simulated solution to the problem *constant surface temperature* at different times (5-300 s), surface temperature is equal to 1°C.

The analytical solution to this one-dimensional problem is simply [12]:

$$T(z,t) = 1 - erf\left(\frac{z}{2}\sqrt{\frac{\rho c}{\lambda \cdot t}}\right) \qquad \text{[K]} \qquad (20)$$

where *erf* is the *error function* defined as:

$$erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\psi^2} d\psi \qquad (21)$$

The corresponding analytical solution is showed in Fig. 10b.

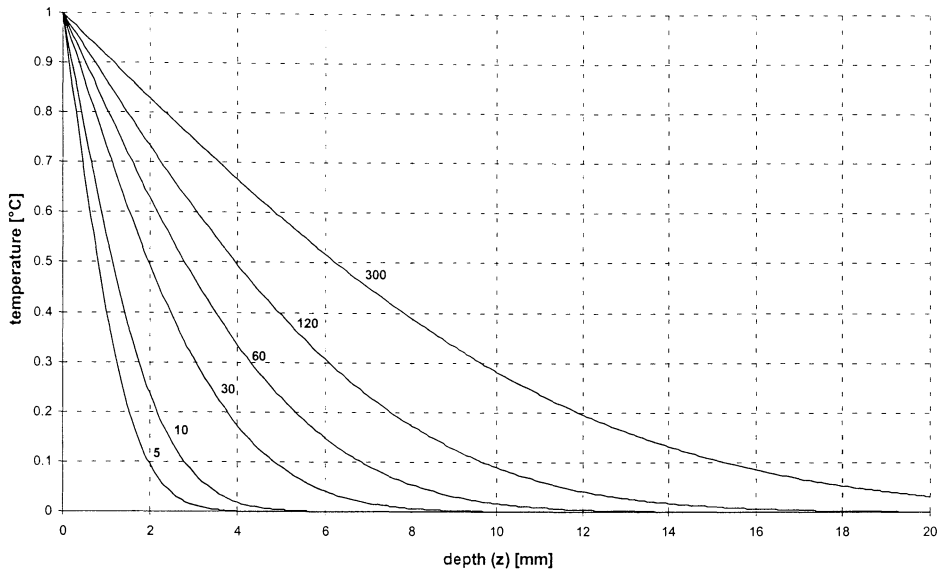

Figure 10b. The analytical solution to the problem *constant surface temperature* at different times (5-300 s), surface temperature is equal to 1°C.

5.3.2 Continuos line source

The simplest geometry to study the radial heat flow is to apply a continuos line source at r =0 m. In this way, the solution accuracy in the radial direction can be studied. The continuos line source is modelled by filling in the first column in the heat source matrix with ones and letting the rest of the elements equal zero. The input US power requested by the program is then set to a value that makes the line source produce 1 W per unit length. Temperature probes are placed in the radial direction (0.1-9.9 mm) which monitor the temperature at different times (5-300 s) and the simulated solution is showed in Fig. 11a.

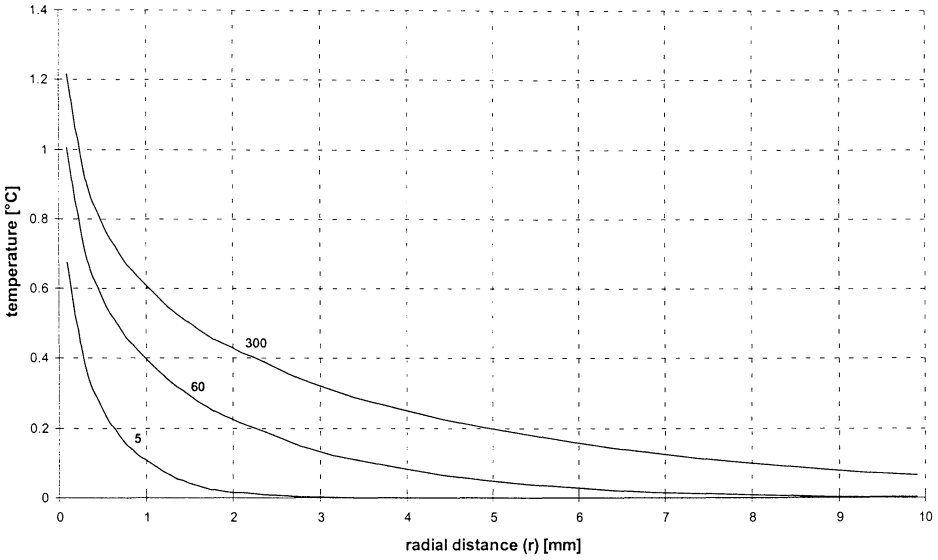

Figure 11a. The simulated solution to the problem *continuos line source* at different times (5-300 s), source power is equal to 1 W m^{-1}.

The analytical solution to this problem is simply [12]:

$$T(r,t) = -\frac{q}{4\pi \cdot \lambda} Ei\left(-\frac{r^2 \cdot \rho c}{4 \cdot \lambda \cdot t}\right) \qquad [K] \qquad (22)$$

where q [W m^{-1}] is the source power per unit length and *Ei* is the *Exponential integral* defined as:

$$Ei(-x) = -\int_x^{\infty} \frac{e^{-\psi}}{\psi} d\psi \qquad (23).$$

The corresponding analytical solution is showed in Fig. 11b.

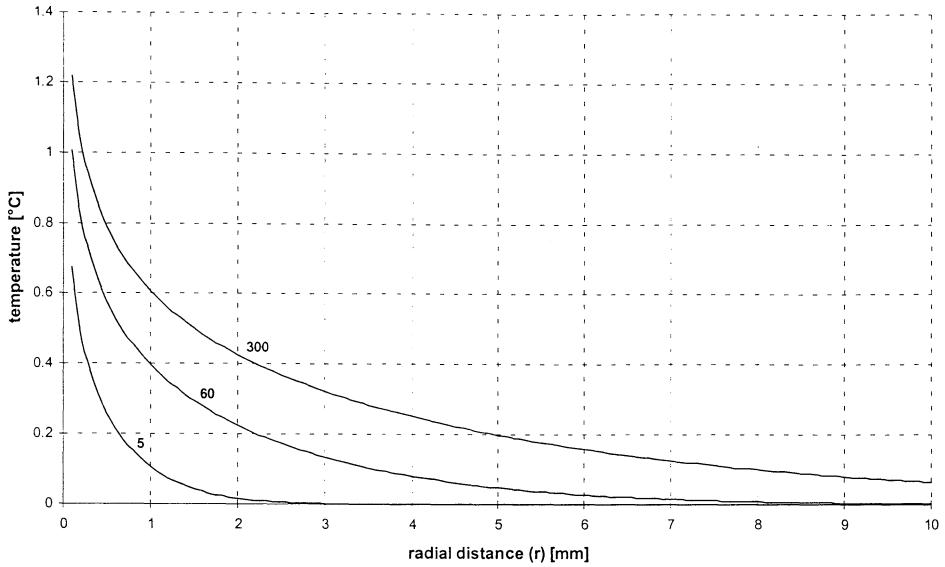

Figure 11b. The analytical solution to the problem *continuos line source* at different times (5-300 s), source power is equal to 1 W m^{-1}.

5.3.3 Pulsed point source

The previous two problems have indicated that the program can produce solutions that are accurate in both temporal- and spatial aspects, however the problems don't connect to the actual situation with a pulsed heat source. Thus, the pulsed point source tests the solution accuracy under more realistic conditions (the concept *point source* is anyhow idealized).

The pulsed point source is modelled by filling in one voxel in the first column in the heat source matrix with a one and letting the rest of the elements equal zero. The point source is placed at such a depth that the influence of the upper boundary is negligible. Further, the input US power is set to a value that makes the source produce 1 W intermittently. Accordingly, the burst duration and the repetition period are set to 1.3 s and 10 s. The simulation stop time is 100 s and the temperature at a distance of 3 mm to the point source is continuously measured during the simulation and the result is showed in Fig. 12 (note, both the simulated- and the analytical solutions are plotted in the figure).

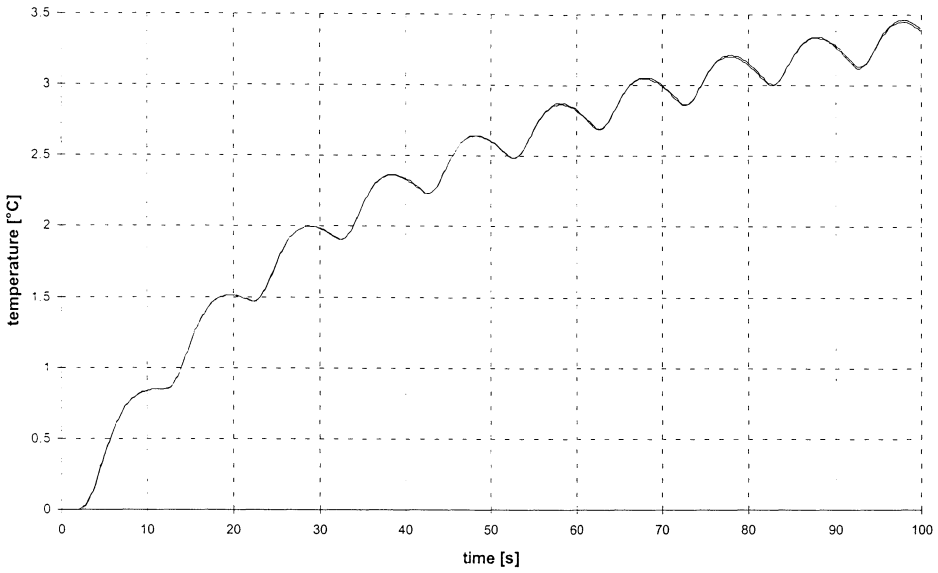

Figure 12. The simulated- and the analytical solution to the problem *pulsed point source* at a distance of 3 mm from the source (the two curves almost coincide). The source produces 1 W intermittently with the burst time equal to 1.3 s and the repetition period equal to 10 s.

The analytical solution to this problem needs some mathematical explanation. The temperature due to an instantaneous point source in an infinite homogenous isotropic medium is completely described by [12]:

$$T(r,t) = \frac{q}{8 \cdot \rho c}\left(\frac{\rho c}{\pi \cdot \lambda \cdot t}\right)^{3/2} \cdot e^{-r^2 \cdot \rho c / 4 \cdot \lambda \cdot t} \qquad [\mathrm{K}] \qquad (24)$$

where q [J] is the liberated heat. To attain the corresponding expression due to a temporally extended point source, equation 24 must be integrated with respect to time, t. Eventually, two equations emerge [13]:

$$T(r,t) = \frac{q}{4\pi \cdot \lambda \cdot r} erfc\left(\frac{r}{2}\sqrt{\frac{\rho c}{\lambda \cdot t}}\right) \quad ; \quad t \le t_{burst} \qquad [\mathrm{K}] \qquad (25a)$$

and

$$T(r,t) = \frac{q}{4\pi \cdot \lambda \cdot r}\left(erfc\left(\frac{r}{2}\sqrt{\frac{\rho c}{\lambda \cdot t}}\right) - erfc\left(\frac{r}{2}\sqrt{\frac{\rho c}{\lambda \cdot (t - t_{burst})}}\right)\right) \quad ; \quad t > t_{burst}$$

[K] (25b)

where q [W] is the source power and *erfc* is the *complementary error function* defined as *1-erf*. Equation 25a is valid when t is smaller than t_{burst}, that is, during the burst. Consequently, Eq. 25b is valid when t is larger than t_{burst}, that is, after the burst. Equations 25a and 25b describe the temperature due to one burst but when several bursts have been applied, the contribution of each burst must be added. Thus, the temperature after N bursts is calculated with [13]:

$$T(r,t) = \frac{q}{4\pi \cdot \lambda \cdot r} \cdot \sum_{k=0}^{N-1}\left(erfc\left(\frac{r}{2}\sqrt{\frac{\rho c}{\lambda \cdot (t - k \cdot t_{period})}}\right) - \right.$$

$$\left. - erfc\left(\frac{r}{2}\sqrt{\frac{\rho c}{\lambda \cdot (t - k \cdot t_{period} - t_{burst})}}\right)\right) \qquad \text{[K]} \qquad (26)$$

where t_{period} is the burst repetition period. The analytical solution is showed in Fig. 12.

6 Experimental method

An *in vitro* experiment has been performed on porcine spinal muscle with the objective to test if the simulation program can reproduce the experimental results.

In the experiment, a sample of porcine spinal muscle was submerged in a container filled with degassed water. The treatment head, with the HIFU transducers facing upwards, formed the bottom surface of the container. By using a positioning system, the tissue sample was maneuvered so that the geometrical focus was situated inside the tissue sample. Before this, three thermocouples, connected to a measurement system, were inserted in the tissue. Then, the treatment head was connected to a power amplifier and the HIFU was turned on together with the measurement system. After five minutes, the experiment was terminated and the recorded tissue temperatures were stored to disc.

A simulation of the experiment was then conducted. The simulation parameters were identical to the ones used in the experiments and the other parameters, such as the thermal conductivity etc., were adopted from [9].

The results were evaluated by comparison of the experimental temperatures and the simulated temperatures and for this purpose, *Microsoft Excel* was used.

6.1 Experimental setup

The experimental setup consists of four parts (see Fig. 13): the treatment head, the control electronics, the measurement system and the positioning system.

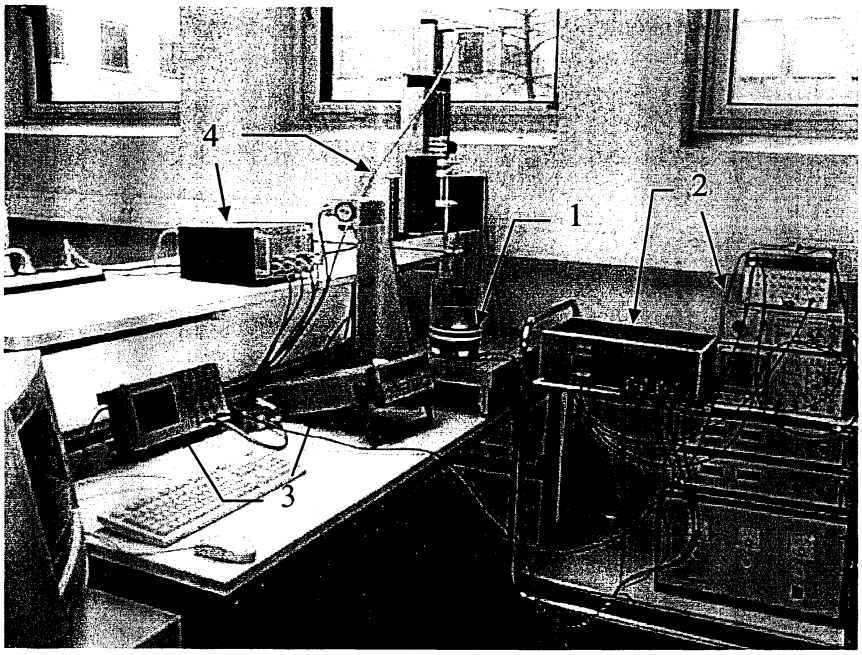

Figure 13. Photograph taken in the laboratory. The photo shows the treatment head (1) with the Plexiglass tube mounted, the control electronics (2), the measurement system (3) and the positioning system (4).

6.1.1 The treatment head

The treatment head, developed at the *Dep. of Biomedical Engineering - Lund University*, consists of five identical spherically curved piezo-electric (PZ-27) transducers (Ferroperm) mounted in a PVC holder in such a way that their geometrical foci coincide. Each transducer has a diameter of 5 cm, a focal distance of 7 cm, a resonant frequency of 1.6 MHz and an electrical-to-ultrasound power efficiency (η) of ~80%[4]. The impedance of each transducer is 50 Ω (resistive) at the resonant frequency. At the time of the experiment, the treatment head was equipped with a large Plexiglass tube and degassed water was poured into it (see Fig. 14).

[4] Measurements performed at the *Dep. of Electrical Measurements - Lund Institute of Technology.*

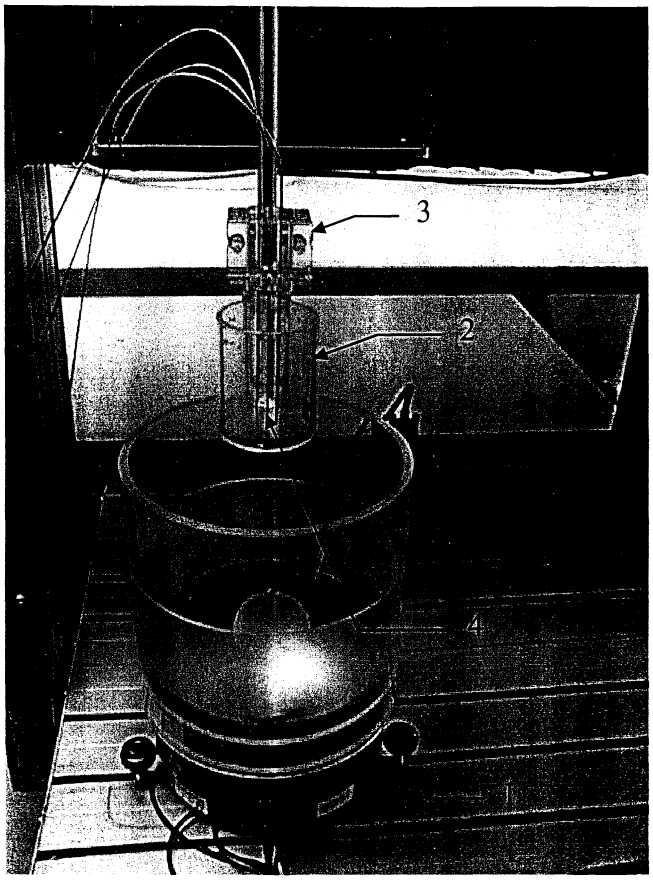

Figure 14. Close-up photo of the treatment head (1) with the Plexiglass tube, the tissue holder (2), the thermocouple guide/holder (3) and the three thermocouples (4).

6.1.2 Control electronics

The central part of the control electronics is the power amplifier[5], developed at the department, to which the treatment head is connected (see Fig. 15). The high-power parts of the amplifier are fed with DC supplied by two power supplies (SM7020-D, Delta Elektronika). Low-power 5- and 12 V_{DC} are supplied by a power supply build at the department.

[5] M. Berggren, T. Svensson, "*Effektgenerator för terapeutiskt ultraljud*", Report 7/93 (Masters thesis), Dep. of Electrical Measurements, Lund Institute of Technology.

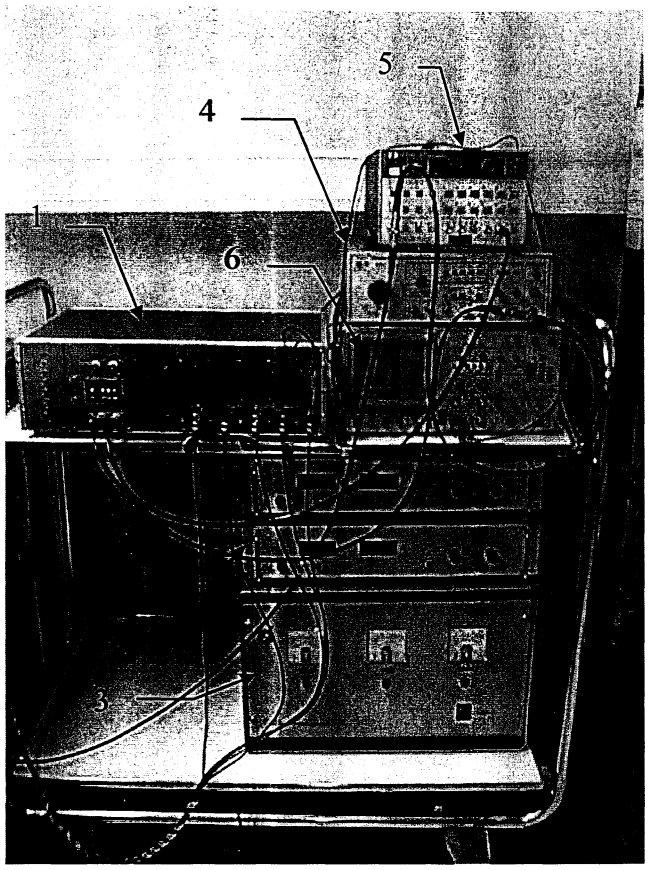

Figure 15. Photo of the control electronics. The power amplifier (1), high-power DC supplies (2), low-power 5- and 12 V_{DC} supply (3), burst start signal generator (4), burst stop signal generator (5) and the oscilloscope (6).

To create the bursted output, the power amplifier needs two signals, the burst start signal and the burst stop signal. The burst start signal is generated by a function generator (LFG-1310, Leader Electronics). The burst stop signal is triggered by the negative flank of the start signal and is generated by another function generator (3314A, Hewlett Packard). Additionally, an oscilloscope (2245A, Tektronix) is used as a monitoring tool.

6.1.3 The measurement system

To monitor the temperature progress generated by the HIFU, there is a need of a measuring system (see Fig. 16). The measurement system consists of three thermocouples (type K), a data acquisition unit (34970A with a 20ch multiplexor module 34901A, Hewlett Packard) and an oscilloscope (TDS 210, Tektronix) with a pre-amplifier (10×) build at the department.

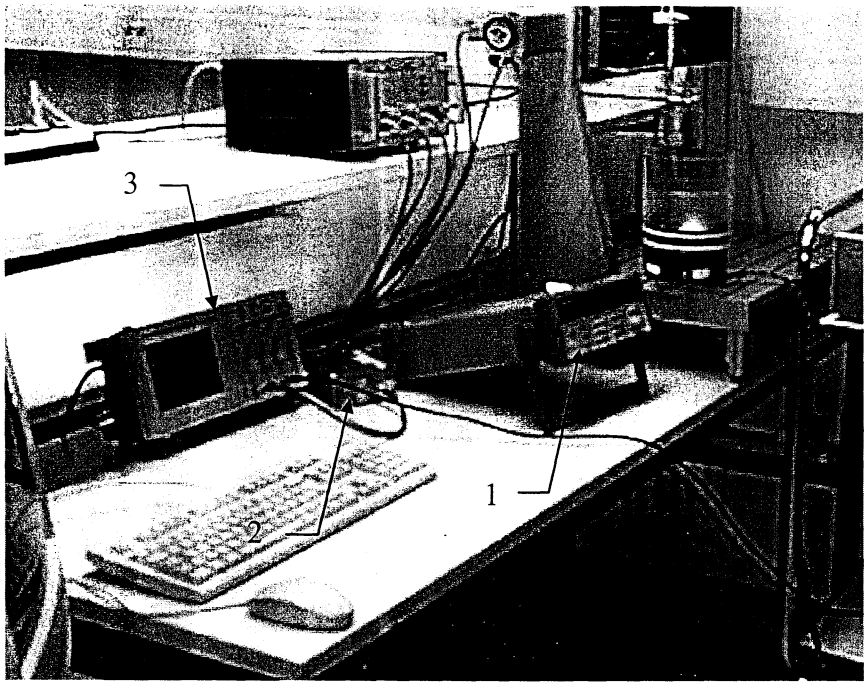

Figure 16. Photo showing the measurement system. The data acquisition unit (1), the pre-amplifier (2) and the oscilloscope (3).

The thermocouples are connected to the data acquisition unit which has an internal compensation of *the cold junction*. Further, one of the thermocouples is also connected to the pre-amplifier which amplifies the thermo-voltage before it is read by the oscilloscope. This constellation allows high-resolution measurements by the oscilloscope additional to the general measurements made by the data acquisition unit.

Both the oscilloscope and the data acquisition unit are controlled by a computer via GPIB. The software running on the computer is a LabVIEW application. It continuously reads the temperatures from the data acquisition unit 5-6 times per second at the same time as it reads the temperature measured by the oscilloscope. The oscilloscope is only triggered when a burst start signal is sensed.

Since the pre-amplifier lacks a cold junction compensation, the setup and the LabVIEW application must be calibrated prior to any experiment. This is done by alternately placing the thermocouple (the one that is connected to both the oscilloscope and the data acquisition unit) in an ice-bath or boiling water and adjusting an "offset" and a "slope" (implemented in the program) so that the temperature read by the oscilloscope is 0°C and 100°C respectively.

When a temperature measurement is complete, the program saves the recorded temperature data to disc as a text file.

6.1.4 The positioning system

Exact control and positioning of the tissue sample is accomplished by a tissue holder mounted on a positioning table (Solectro). The tissue holder is developed at the department and is made of a Plexiglass cylinder (∅ 45 mm × 60 mm) and a thermocouple guide/holder mounted above the cylinder (see Fig. 14). The positioning table allows positioning with a tolerance of 0.02 mm in any of the x-, y- and z directions. It is under control (via RS-232) by the same computer controlling the measurement system but by another LabVIEW application.

6.2 *In vitro* experiment

The first measurement, after proper calibration, was done by placing the thermocouples in an ice-bath and recording the temperatures. The recorded temperatures should thus be 0°C but the internal cold junction compensation of the data acquisition unit is not perfect which resulted in temperatures different from 0°C. This difference was afterwards subtracted from the experimental *in vitro* temperatures.

When the ice-bath recording was done, the Plexiglass tube was filled with approximately 1.5 liters of degassed water at a temperature of 10°C and a sample of tissue, with an uniform temperature of 24°C, was pushed into the Plexiglass cylinder of the tissue holder. The tissue sample was pushed into the cylinder in a way that the lower tissue surface leveled the rim of the Plexiglass cylinder. This made it easier to insert the thermocouples correctly and to position the tissue sample.

Three hollow-needles were then inserted to the desired position guided by the thermocouple guide/holder. Subsequently, the three thermocouples were inserted, through the needles, and the needles were then carefully withdrawn approximately 1.5 cm. The thermocouples were placed in such a way that the one connected both to the data acquisition unit and the oscilloscope was situated at a distance of 15(±1) mm to the lower surface. The other two were placed diametrically opposed on each side of the first one at a distance of 10(±1) mm to the lower surface (see Fig. 17). The spacing between the thermocouples was 5(±0.5) mm.

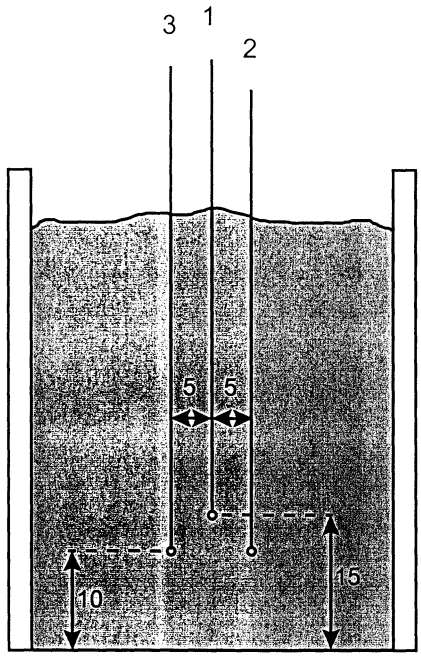

Figure 17. Schematic figure showing the tissue sample inside the Plexiglass holder with the thermocouples (denoted 1, 2 and 3) inserted and their corresponding position.

Then, the tissue sample was submerged into the water so that the geometrical focus of the treatment head coincided with the middle thermocouple (thermocouple number 1 in Fig. 17).

When everything was set, the measurement system was started and the HIFU was turned on immediately after. The total electrical power into the treatment head was set to 45 W (15 V_{DC} on the power supplies). This would only lead to a minimal coagulation of the tissue, according to a pilot simulation, which else would have caused undesirable shrinkage of the coagulated region and displacing the thermocouples. The burst start signal generator was adjusted so that the burst time was 1.38 s and the repetition period was 10.12 s while the burst stop signal generator was programmed to send a short pulse (60 μs) when trigged by the negative flank of the start signal.

Eventually, the treatment time reached 5 min and the HIFU was turned off while the measurement system was terminated 30 s later. A control measurement of the total electrical power showed that the actual power into the treatment head had been 48.7 W. No further examination of the tissue sample was done.

6.3 Simulation

A number of simulations were performed with the object to reproduce the *in vitro* experiment with respect to the measured temperatures. The input HIFU power was changed in each simulation in order to produce the best match. Since the transducers have an efficiency of about 80%, the input power was set to values ranging from 77% to 83% with respect to efficiency. Additionally, a number of imaginary probes were "inserted" to cover the area of uncertainty associated with each experimental thermocouple. The procedure described below is the simulation that produced the best temperature match.

Prior to the simulation, the parameters needed to complete the simulation were adopted from the literature and were retrieved from pig muscle data found in [9] (see table 3). A parameter file denoted *pig.par* was thus created.

Table 3. Tissue parameters, adopted from [9], used in the simulation.

parameter	value
λ	0.50 W m^{-1} K^{-1}
ρ	1050 kg m^{-3}
c	3.80 J kg^{-1} K^{-1}
v	1600 m s^{-1}
α	0.09 Np cm^{-1} @ 1 MHz

First, the heat source was build, using *Sourcemaker*, and the size of the simulation domain was chosen to be 30×45 mm in radial- and axial extension, with $\Delta r=\Delta z=0.2$ mm. The difference between the size of the simulation domain and the actual size of the tissue sample is insignificant since the temperatures are measured at spots sufficiently far from the boundaries. The intensity map associated with the treatment head used in the experiment is named *scan1.asc* and the geometrical focus was placed at the desired depth of 15 mm in accordance with the experiment. The resulting heat source file was thus build and was denoted *source.prz*.

The *Tempsim* program was then launched and the name and searchpath of the heat source file was entered. When the total input power into the tissue was asked for, the value was set to 38 W (which corresponds to an efficiency equal to 78%). Then, a number of probes were arranged in a manner described earlier. The program continued with asking for the rest of the simulation parameters, such as the initial temperature and the simulation time etc., and the values typed are found in table 4.

Table 4. Simulation parameters, apart from HIFU power and probe coordinates, used in the temperature simulation.

parameter	value
treatment time (US_{stop})	300 s
investigation time (simulation stop)	330 s
burst time (t_{burst})	1.38 s
repetition period (t_{period})	10.12 s
initial tissue temperature	24°C
water temperature	10°C
blood flow rate	0 ml $(100g)^{-1}$ min^{-1}
arterial blood temperature	0 °C
time step	0.07 s (system calculated)

When the last parameter was entered, the simulation began and carried on for about 3.5 minutes. The output data (cf. section 5.2) were then automatically stored as text files with names entered earlier in the program.

7 Results

The temperature data from the experiment were first adjusted in accordance with the procedure described earlier, that is, the mean ice-bath temperature of each thermocouple was subtracted from the *in vitro* experimental data. Further, the time abscissa was adjusted so that the first recorded burst starts at t=0 s. The experimental results after these manipulations are showed in Fig. 18.

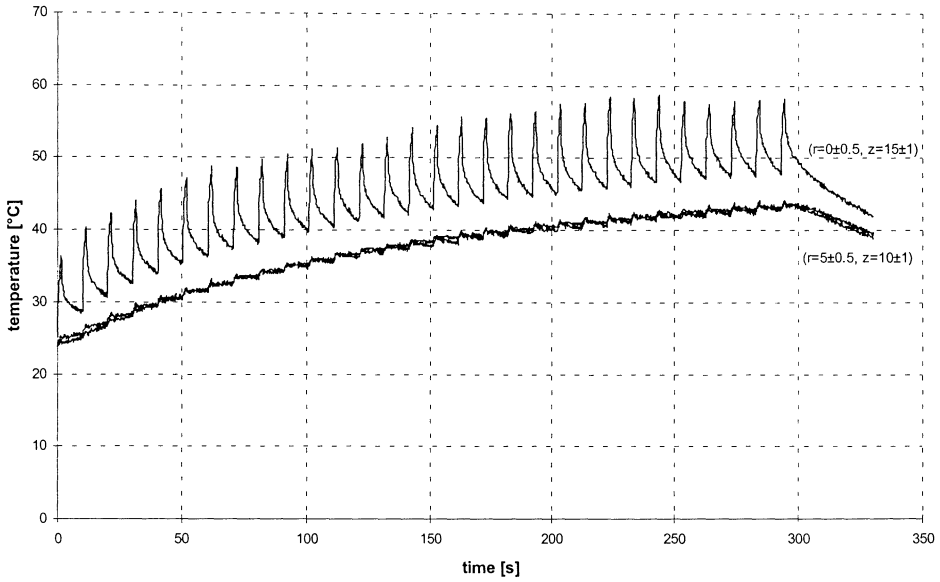

Figure 18. Experimental *in vitro* temperatures measured, at points as in the figure, with thermocouples in porcine spinal muscle. The geometrical focus was placed at z=15±1 mm, total electrical power into the treatment head was equal to 48.7 W, t_{burst}=1.38 s, t_{period}=10.12 s, $T_{initial}$=24°C and T_{water}=10°C. The HIFU was turned off after 300 s.

The best match when fitting the simulated temperatures to the experimental temperatures was achieved by the probes situated at (r=0.1, z=15.9) and (r=4.5, z=9.9). These coordinates are located within the area of uncertainty of each thermocouple. The simulated results are showed in Fig. 19.

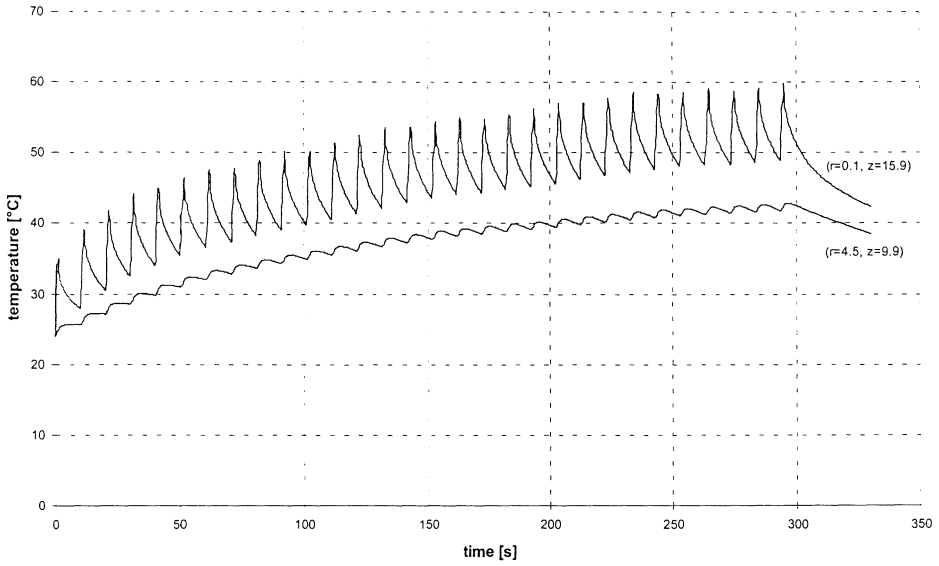

Figure 19. Simulated temperatures obtained when reproducing the *in vitro* experiment. The simulation parameters were identical to those in the experiment with exception of the HIFU power, which was set to 38 W (η=78%), and the position of the temperature probes.

The simulated temperature distribution at the simulation stop time, the coagulation distribution and the distribution of the maximum temperature recorded in each voxel during the simulation are all found in appendix C.

8 Discussion

It is clear from Figs 18 and 19 that there is a good agreement between the experimental and the simulated results. However, there are some deviations that need to be remarked. Firstly, the most noticeable deviation is the diminution of the temperature peaks recorded by the middle thermocouple (the upper curve in Fig. 18) occurring after 250 s. This is probably due to a coagulation event in front of the thermocouple (cf. Fig. 2C in appendix C). The event of coagulation will cause the US to be highly absorbed in the coagulated region since a coagulation is associated with higher attenuation (α). This prevents the HIFU to reach the area behind the coagulated region, in which the middle thermocouple is situated[6]. The other two thermocouples are unaffected by the coagulation since they are situated beside the coagulated region. However, if the experiment would have proceeded, temperature changes would have been noticeable for those too.

Secondly, the temperature decay after each burst seems to be faster in the experiment compared to the simulation. Beside the possibility that the tabulated thermal conductivity (λ) could be wrong, it may be due to an anisotropic thermal conductivity which is not unlikely in fibrous tissues as muscle. Generally, the thermal conductivity is higher perpendicular to the fibers. The second thing that might affect the temperature decay is the fact that the thermal conductivity and the density etc. of the tissue are dependent on the temperature and not assumed constant as in the model. Finally, heat dissipation through the thermocouple wires may contribute to the faster decay.

The results from the solution validity part (section 5.3) needs no further attention since it is elucidated, by the figures, that the program accurately can calculate the heat transfer in tissue.

8.1 Sources of errors

There are sources of errors associated both with the *in vitro* experiment and the simulation that can affect the outcome.

In the experiment, the most significant source of error is the uncertainty of positioning the thermocouples and the geometrical focus. Even a small deviation from the desired position would have great impact on the result (cf. Fig. 1C and 2C in appendix C). The finite size of the thermocouples could lead to errors since the measured temperature is averaged over a volume and not measured in a point. There are also errors associated with visual reading of voltages displayed on the monitoring oscilloscope, from which the total power is calculated and finally, the impedance of the transducers might differ from 50 Ω.

[6] This phenomenon has been thoroughly investigated by Albu-Barkman [6].

The most significant sources of errors in the simulation are the tabulated tissue acoustical- and thermal properties adopted from [9]. In the calculation procedure, errors may arise from the assumption that the tissue properties are constant in every sense. Finally, the discretization of the time may lead to errors when the heat damage (coagulation) is calculated (cf. Eq. 19).

8.2 Future improvements

The future improvements on the simulation program, *Tempsim*, would be to allow inhomogenities to be introduced and to take the increase of absorption in coagulated voxels into account. If doing this, the question of how to correctly calculate the attenuation associated with each voxel inside the inhomogenity emerges. At present, the attenuation is calculated from the axial depth of each voxel and not from the actual path of propagation, which is rather complex due to contributions from several transducers. As a first-order solution, this could also be applicable in presence of tumors or coagulated tissue. Further, if the increase of attenuation should be accounted for, the whole heat source must be recalculated after every coagulation event. Complete knowledge of the magnitude of the change in the attenuation coefficient is also required. Changes in other parameters, such as the thermal conductivity, due to coagulation or temperature could also be accounted for and no heat source recalculation is required.

Another improvement would be to improve the coagulation calculation by means of employing a better integral calculation formula, such as the *trapezoid method*.

Finally, the possibility of placing the imaginary temperature probes arbitrarily, not constrained to the center of the nearest voxel, could be desirable. This could be implemented with bi-linear interpolation of the temperatures in the four nearest voxels.

The only improvement in the *Sourcemaker* program would be to allow tumors to be defined, in accordance with the *Tempsim* program.

9 Acknowledgements

I would like to thank all of my supervisors:

At the *Dep. of Physics, Atomic Physics - Lund Institute of Technology*: Associate Professor Stefan Andersson-Engels and Ph. D. Christian Sturesson.

At the *Dep. of Biomedical Engineering - Lund University*: Professor Nils-Gunnar Holmer, Associate Professor Tomas Kirkhorn, Med. Lic. Cristina Albu-Barkman and Med. Lic. Lars-Olof Almquist.

Finally, I would like to thank Assistant Professor Hans W Persson, Dr. Monica Almqvist and Dr. Tomas Jansson at the *Dep. of Electrical Measurements - Lund Institute of Technology* for letting me use their hydrophone equipment.

10 References

[1] F.C. Henriques, "*Studies of thermal injury*", Arch. Pathol., 43: 489-502, 1947

[2] C. Sturesson, "*Medical laser-induced thermotherapy - models and applications*", LRAP-235 (Doctoral thesis), Dep. of Physics, Lund Institute of Technology, 1998

[3] C. Sturesson, "*Theoretical modelling of the temperature distribution in laser-induced hyperthermia*", LRAP-159 (Masters thesis), Dep. of Physics, Lund Institute of Technology, 1994

[4] L.E. Gerweck, "*Hyperthermia in cancer therapy: The biological basis and unresolved questions*", Cancer Research, 45:3408-3414, 1985

[5] C.R. Hill, G.R. TERHAAR, "*Review article: High intensity focused ultrasound - potential for cancer treatment*", The British Journal of Radiology, 68, 1296-1303, 1995

[6] C. Albu Barkman, "*High intensity focused ultrasound in medicine - thermotherapy*", Report 1/1998 (Licentiate thesis), Dep. of Biomedical engineering, Lund University, 1998

[7] F.W. Kremkau, "*Cancer therapy with ultrasound: A historical review*", Journal of Clinical Ultrasound, vol. 7, 287-300, 1979

[8] C.R. Hill, "*Physical principles of medical ultrasonics*" Ellis Horwood limited Chichester, England, 1986

[9] F.A. Duck, "*Physical properties of tissue*", Academic Press limited London, England, 1990

[10] R.K. Jain, "*Bioheat transfer: Mathematical models of thermal systems*", Hyperthermia in cancer therapy, Hall Medical Publishing Boston, Ma, 1983

[11] G. Kossoff, "*Analysis of focusing action of spherically curved transducers*", Ultrasound in Medicine & Biology, vol.5, 359-365, 1979

[12] H.S. Carslaw, J.C. Jaeger, "*Conduction of heat in solids*", 2:nd edition, Clarendon Press Oxford, England, 1959

[13] A.J. Welch, M.J.C. van Gemert, "*Optical-thermal response of laser-irradiated tissue*", Plenum Press New York, 1995

Appendices

Appendix A Existing treatment heads

At the *Dep. of Biomedical Engineering - Lund University*, there exists three treatment heads intended to be used in thermotherapy. Characteristic data, intensity maps and the features of the scanned intensity maps for each treatment head are found below.

Table 1A. Characteristic data for the three treatment heads.

head no.	no. of transducers	resonant frequency	focal distance	transducer diameter	η	Z
1	5	1.6 MHz	7 cm	5 cm	~80%	50 Ω
2	6	1.6 MHz	10 cm	5 cm	~80%	50 Ω
3	6	0.5 MHz	10 cm	5 cm	~80%	50 Ω

Table 2A. Features of the mapped areas. Input to the *Sourcemaker* program.

head no.	axial scan distance	radial scan distance	grid size $\Delta r=\Delta z$	position of geom. focus in map	file name
1	65 mm	25.2 mm	0.33 mm	40 mm	scan1.asc
2	65 mm	25.2 mm	0.33 mm	42 mm	scan2.asc
3	65 mm	25.2 mm	0.33 mm	40 mm	scan3.asc

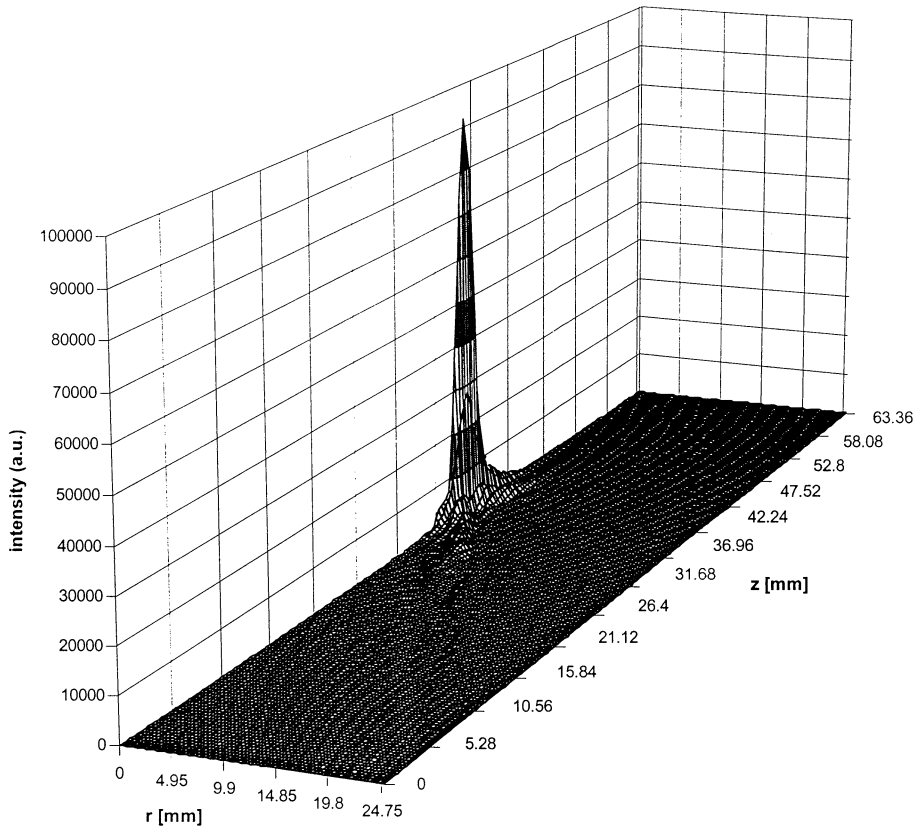

Figure 1A. The intensity distribution of treatment head no. 1. The head should be imagined located down to the left and radiating up to the right.

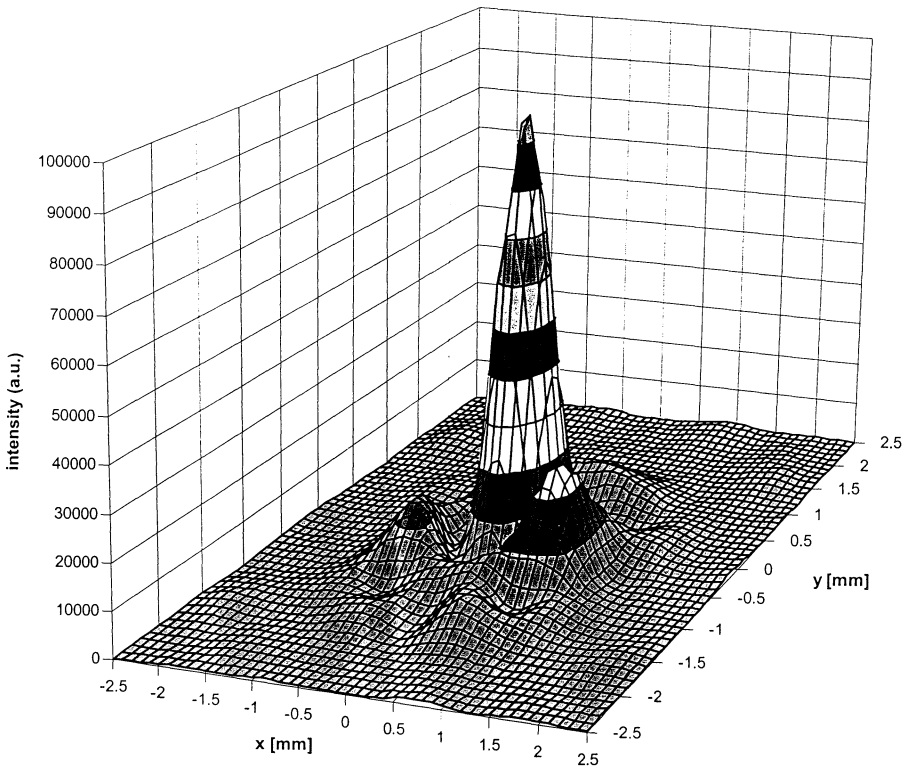

Figure 2A. The intensity distribution in the focal plane of treatment head no. 1.

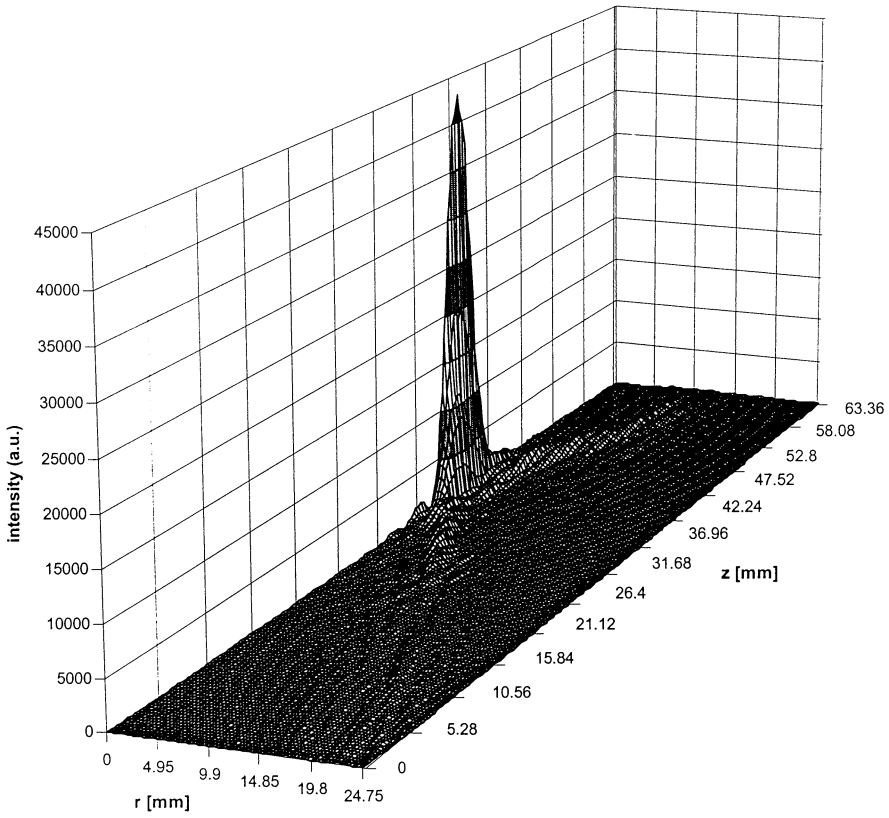

Figure 3A. The intensity distribution of treatment head no. 2. The head should be imagined located down to the left and radiating up to the right.

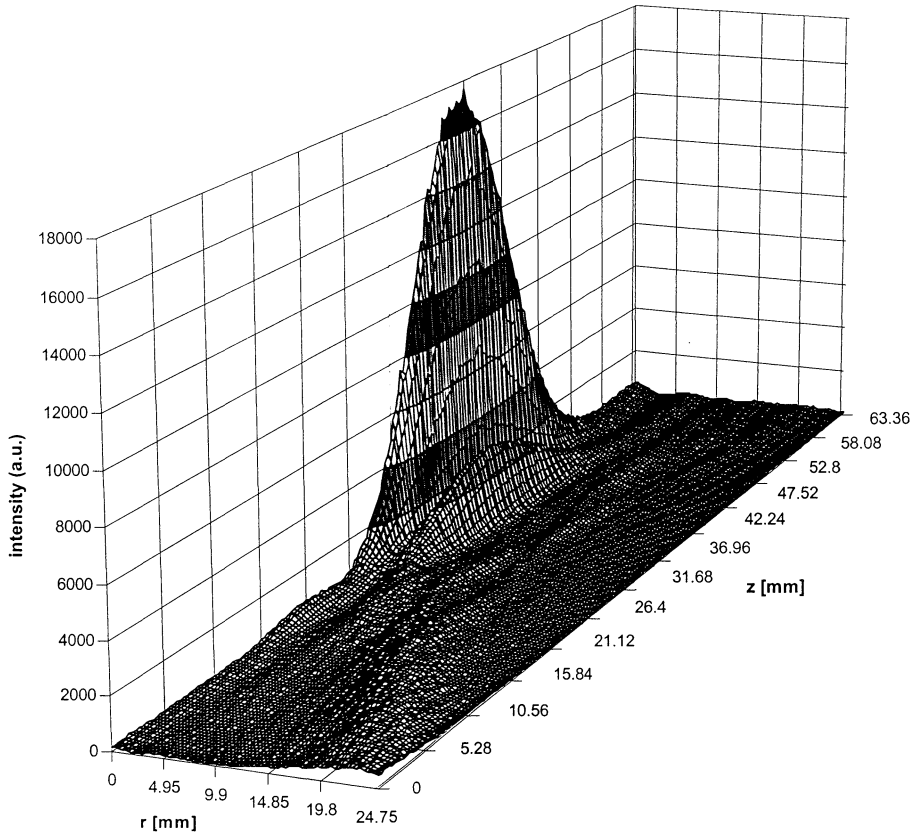

Figure 4A. The intensity distribution of treatment head no. 3. The head should be imagined located down to the left and radiating up to the right.

Appendix B File examples

Note, "_", "↵" and "⇨" should be interpreted as *space, carriage return* and *tabulator* respectively.

Example 1B: An intensity map file (extract from *scan1.asc*):

```
[0,_0,_0]↵
5.50E+01↵
5.05E+01↵
4.62E+01↵
5.50E+01↵
4.62E+01↵
⋮
(...continued...)
⋮
9.78E+01↵
9.78E+01↵
8.60E+01↵
8.60E+01↵
9.78E+01↵
↵
[1,_0,_0]↵
5.05E+01↵
5.97E+01↵
5.05E+01↵
5.05E+01↵
5.05E+01↵
5.50E+01↵
⋮
```

Example 2B: A heat source file (extract from *source.prz*):

```
dr[mm]=_0.200↵
Imax=_150↵
dz[mm]=_0.200↵
Jmax=_225↵
vhcap[J/m3K]=_3.990e+006↵
thermcond[W/mK]=_0.500↵
dens[kg/m3]=_1050.0↵
2.35E-05⇨    2.28E-05⇨    1.72E-05⇨    1.18E-05⇨ ⋯
2.35E-05⇨    2.27E-05⇨    1.72E-05⇨    1.17E-05⇨
2.30E-05⇨    2.15E-05⇨    1.64E-05⇨    1.15E-05⇨
2.21E-05⇨    1.89E-05⇨    1.47E-05⇨    1.09E-05⇨
1.84E-05⇨    1.58E-05⇨    1.34E-05⇨    1.10E-05⇨
1.48E-05⇨    1.33E-05⇨    1.27E-05⇨    1.17E-05⇨
⋮                                              ⋱
```

Example 3B: A homogenous parameter file:

```
tissuedensity[kg/m3]=_1050↵
tissuesoundvel[m/s]=_1580↵
tissuealfa[1/cm]=_0.32↵
tissuehcap[J/kgK]=_3.7E+3↵
tissuecond[W/mK]=_0.51↵
```

Example 4B: A layered parameter file:

```
skindensity[kg/m3]=_1100↵
fatdensity[kg/m3]=_900↵
muscledensity[kg/m3]=_1050↵
skinsoundvel[m/s]=_1537↵
fatsoundvel[m/s]=_1476↵
musclesoundvel[m/s]=_1580↵
skinalfa[1/cm]=_0.4↵
fatalfa[1/cm]=_0.21↵
musclealfa[1/cm]=_0.32↵
skinhcap[J/kgK]=_3.53E+3↵
fathcap[J/kgK]=_2.3E+3↵
musclehcap[J/kgK]=_3.7E+3↵
skincond[W/mK]=_0.293↵
fatcond[W/mK]=_0.25↵
musclecond[W/mK]=_0.51↵
dskin[mm]=_1.0↵
dfat[mm]=_3.0↵
```

Appendix C Figures from the simulation

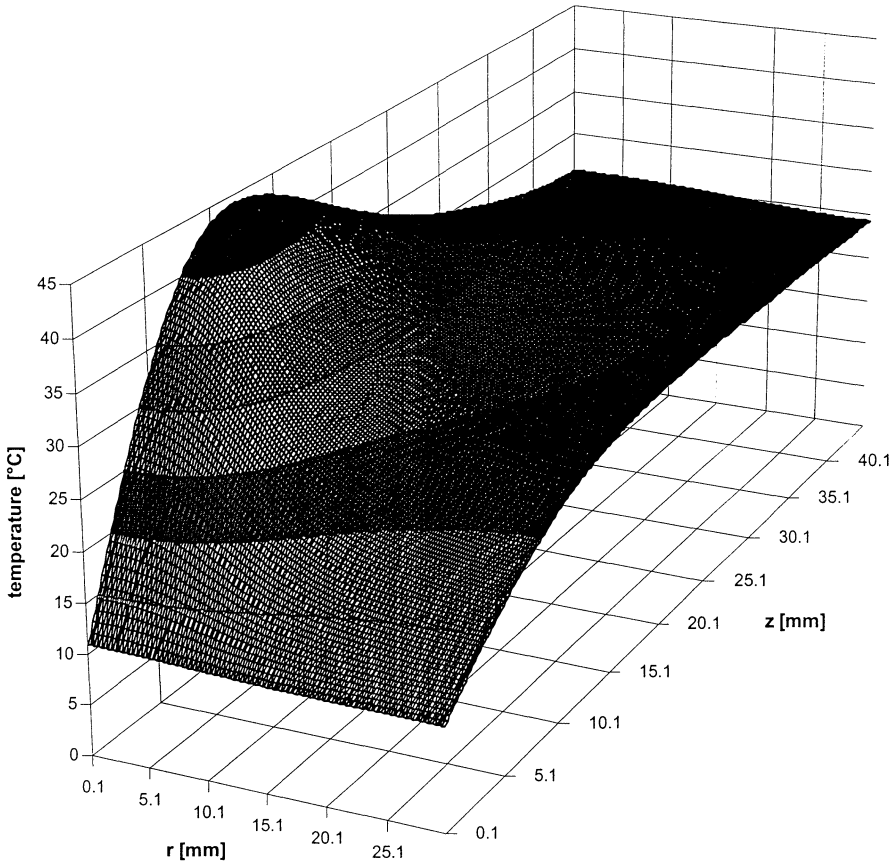

Figure 1C. The simulated temperature distribution at the time of the simulation stop time (330 s). Note, the distribution is radially symmetric. The excessive heat in the tissue is also calculated and was equal to 105 J.

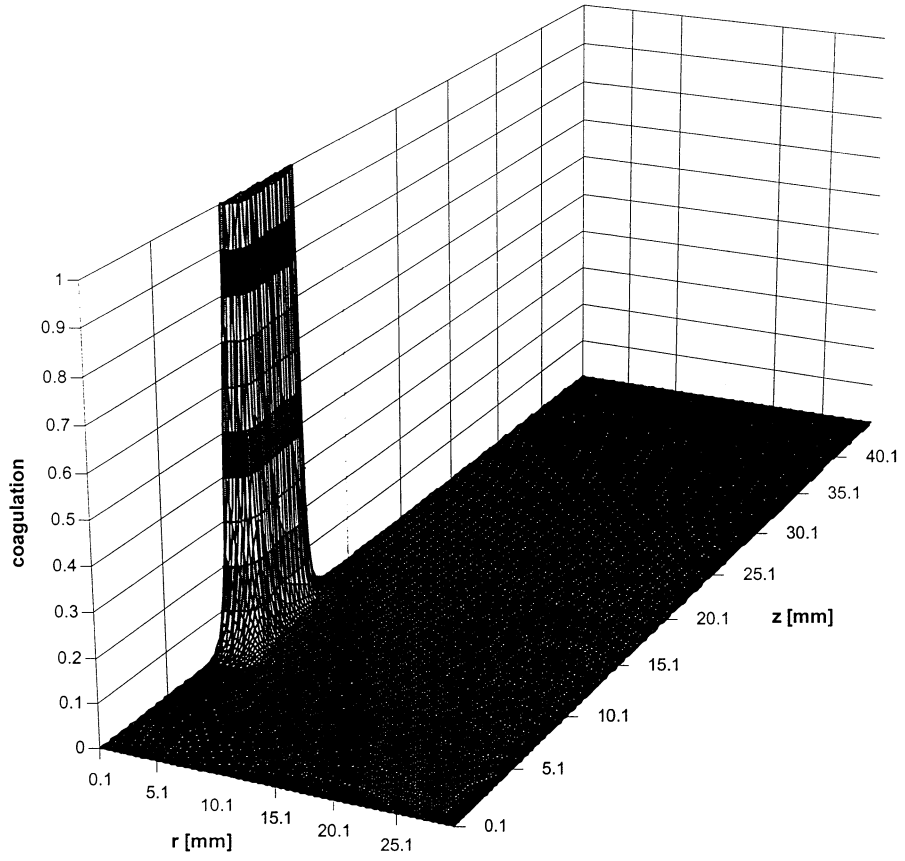

Figure 2C. The simulated coagulation distribution at the simulation stop time. Voxels equal to 1 are considered coagulated. The coagulated volume is also calculated and was equal to 0.026 cm^3.

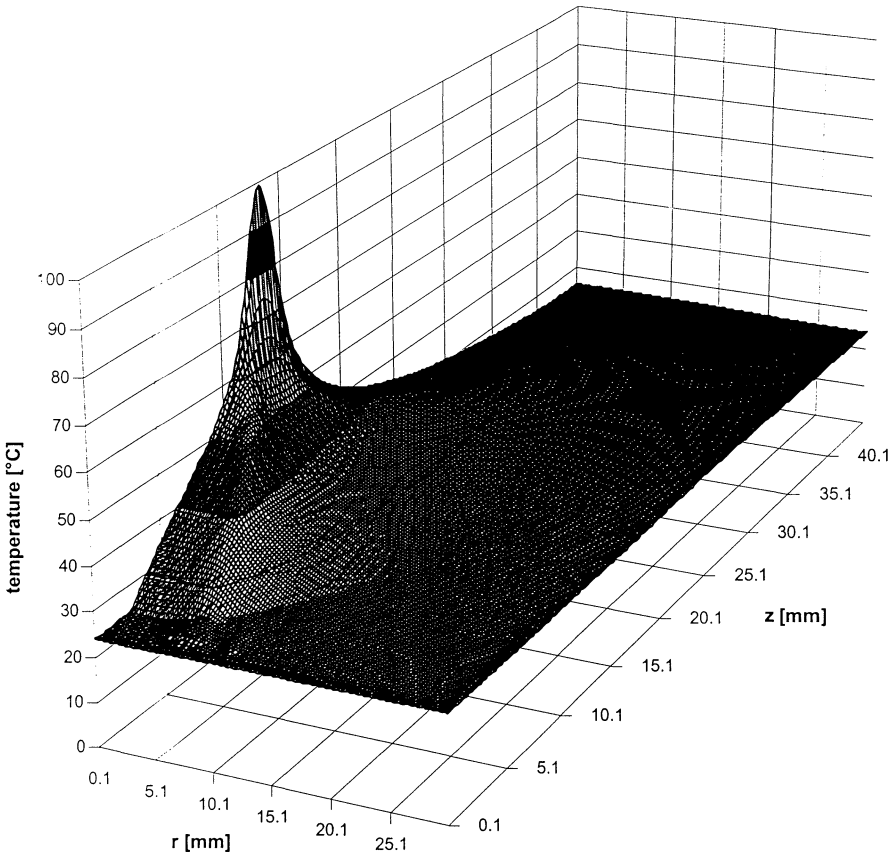

Figure 3C. The maximum temperature distribution recorded in each voxel during the simulation.

Appendix D Source codes

D.1 Sourcemaker

```
/*-------------------------------------------------------------------------
Sourcemaker program

This version requires output from the hydrophone-scanning system
currently found at the Dep. of Electrical Measurements, Lund Institute of Technology
-------------------------------------------------------------------------*/
#include <malloc.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
/*************************************************************************
* Some mathematical and physical definitions                             *
*************************************************************************/
#define PI 3.1415926535
#define sqr(a) ((a)*(a))
#define DENSW 1000.0                    /*Water density [kg/m3]    */
#define VW 1480.0                       /*Water sound velocity [m s]*/
/*************************************************************************/

float *AllocVector(short nl, short nh);
float **AllocMatrix(short ncl,short nch, short nrl,short nrh);
void FreeVector(float *v,short nl);
void FreeMatrix(float **m,short ncl,short nch,short nrl);
void nrerror(char error_text[]);
void release_memory(void);
void readinput(void);
void communicate(void);
void build(void);
void writeprz(void);
void regulate (void);

float focdist,fdepth,simr,simz,dr,dz,axdist,radist,ds;      /*Global variables are used throughout the program*/
float denss,densf,dens,vs,vf,v,alfas,alfaf,alfa,vhcaps,vhcapf,vhcap;
float thermconds,thermcondf,thermcond,dskin,dfat,adjust;
short simumodel;
short Imax,Jmax,Jskin,Jfat,Iin,Jin;
char outputfile[40];
float **inmat;
float **source;
float *vol;

void main(void)                         /*The main program begins*/
{
    readinput();                        /*Reads the output from the scanning-system*/
    communicate();                      /*Lets the user set some important parameters*/
    build();
    regulate();
    writeprz();
    release_memory();                   /*"Garbage-collector"*/
    printf("\nThe sourcemaker program has created a \nsource file with searchpath: %s\n",outputfile);
    printf("\nPress a key to continue...");
    getch();
    exit(0);
}/*end main*/

void readinput(void)
{
    FILE *testfile;
    register short i,j;
    char dumpstr[25],infile[40];
```

```
    printf("_____\
n\n");
    printf("                    Sourcemaker program\n");
    printf("_____\
n\n");
    printf("Type of simulationmodel   1) layered tissue, skin-fat-muscle");
    printf("\n                          2) homogenous tissue: ");
    scanf("%d",&simumodel);                          /*Reads the type of model*
    if((simumodel!=1)&&(simumodel!=2))
    {
        printf("\nThe choice must be either 1 or 2! Press a key to continue...");
        getch();
        exit(1);
    }
    printf("\nEnter searchpath plus name of file from hydrophone-scanning system.\n");
    printf("ex C:\\Program\\simulib\\scan.asc\n>");
    scanf("%s",infile);                              /*Reads the name of the inputfile*/
    testfile=fopen(infile."r");                      /*Opens the inputfile*/
    if (testfile != NULL)
    {
        printf("\nRadial scan-distance [mm]: ");       /*Reads some basic scanning-parameters*
        scanf("%f",&radist);
        printf("\nAxial scan-distance [mm]: ");
        scanf("%f",&axdist);
        printf("\nStepsize [mm]: ");
        scanf("%f",&ds);
        Jin=(short) floor((double) axdist/ds)+1;     /*Calculates the size of the scanned area*/
        Iin=(short) floor((double) radist/ds)+1;
        ds /=1000.0;                                 /*Converts to SI-units [m]*
        axdist=(Jin-1)*ds;
        radist=(Iin-1)*ds;
        inmat=AllocMatrix(1,Iin,1,Jin);
        fscanf(testfile,"%s",dumpstr);               /*Reads some unimportant text*/
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%s",dumpstr);
        for(i=Iin;i>=1;i--)
        {
            for(j=Jin;j>=1;j--)
                fscanf(testfile,"%f",&inmat[i][j]);  /*The values are directly proportional to the intensity
[W/m2]*/
            fscanf(testfile,"%s",dumpstr);           /*Reads some unimportant text*/
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%s",dumpstr);
        }
        fclose(testfile);
    }
    else
    {
        printf("\nFile %s not found! Press a key to continue... ",infile);
        getch();
        exit(1);
    }
}/*end readinput*/

void communicate(void)
{
  FILE *testfile;
    char dumpstr[25],layerfile[40],homfile[40];

    printf("\nGeometrical-focus axial distance in scan [mm]: ");   /*Another basic scanning-parameter*/
    scanf("%f",&focdist);
    printf("\nGeometrical-focus depth in model [mm] <= %3.3f mm: ",focdist);
    scanf("%f",&fdepth);                             /*Lets the user define the simulation domain*/
    printf("\nRadial extension in model [mm]: ");
    scanf("%f",&simr);
    printf("\nRadial resolution [mm]: ");
    scanf("%f",&dr);
    printf("\nAxial extension in model [mm]: ");
    scanf("%f",&simz);
    if(fdepth>simz)                                  /*Fdepth must be < axial extension*/
```

```
}
    printf("\nAxial extension must be > %3.3f mm! Press a key to continue...",fdepth);
    getch();
    exit(1);
}
printf("\nAxial resolution [mm]: ");
scanf("%f",&dz);
focdist /=1000.0;                                   /*Converts to SI-units [m]*/
fdepth /=1000.0;
simr /=1000.0;
dr /=1000.0;
simz /=1000.0;
dz /=1000.0;
if(simumodel==1)                                    /*Layered tissue*/
{
    printf("\nEnter searchpath plus name of layered-tissue parameter file.\n");
    printf("ex C:\\Program\\simulib\\layer.par\n>");
    scanf("%s",layerfile);                          *Reads the name of the layered-tissue parameter file*/
    testfile=fopen(layerfile,"r");
    if(testfile!=NULL)
    {
        fscanf(testfile,"%s",dumpstr);              /*Reads the unimportant text*/
        fscanf(testfile,"%f",&denss);               /*Reads the tissue properties*/
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&densf);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&dens);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&vs);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&vf);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&v);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&alfas);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&alfaf);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&alfa);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&vhcaps);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&vhcapf);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&vhcap);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&thermconds);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&thermcondf);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&thermcond);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&dskin);
        fscanf(testfile,"%s",dumpstr);
        fscanf(testfile,"%f",&dfat);
        fclose(testfile);
        alfas *=100.0;                              /*Converts to SI-units*/
        alfaf *=100.0;
        alfa *=100.0;
        dskin /=1000.0;
        dfat /=1000.0;
    }
    else
    {
        printf("\nFile %s not found! Press a key to continue... ",layerfile);
        getch();
        exit(1);
    }
}
else                                                /*Homogenous tissue*/
```

```
    {
        printf("\nEnter searchpath plus name of homogenous-tissue parameter file.\n");
        printf("ex C:\\Program\\simulib\\homogenous.par\n>");
        scanf("%s",homfile);
        testfile=fopen(homfile,"r");
        if(testfile!=NULL)
        {
            fscanf(testfile,"%s",dumpstr);              /*Reads the unimportant text*/
            fscanf(testfile,"%f",&dens);                /*Reads the tissue properties*/
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&v);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&alfa);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&vhcap);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&thermcond);
            alfa *=100.0;                               /*Converts to SI-units [1/m]*/
            fclose(testfile);
        }
        else
        {
            printf("\nFile %s not found! Press a key to continue... ",homfile);
            getch();
            exit(1);
        }
    }
    printf("\nSearchpath plus name of output-file, with extension .prz\n>");
    scanf("%s",outputfile);                             /*Reads the name of outputfile*/
}/*end communicate*/

void build(void)                                        /*Builds the defined sourcematrix based on the scanned data*/
{
    float Tskin,Tfat,Tmuscle,T,z,r,factor;
    float A,B,C,D,znew;
    register short i,j;

    Imax=(short) ceil((double) simr/dr);
    Jmax=(short) ceil((double) simz/dz);
    source=AllocMatrix(1,Imax,1,Jmax);
    vol=AllocVector(1,Imax);
    if(simumodel==1)                                    /*Layered tissue*/
    {
        Tskin=(4.0*DENSW*VW*denss*vs)/sqr(DENSW*VW+denss*vs);   /*Calculates the transmission
coefficients*/
        Tfat=(4.0*denss*vs*densf*vf)/sqr(denss*vs+densf*vf);
        Tmuscle=(4.0*densf*vf*dens*v)/sqr(densf*vf+dens*v);
        if((dskin+dfat)<=(Jmax*dz))
        {
            if(dskin==0.0)
                Jskin=1;
            else
                Jskin=(short) ceil((double) dskin/dz);
            if(dfat==0.0)
                Jfat=Jskin+1;
            else
                Jfat=(short) ceil((double) dfat/dz)+Jskin;
        }
        else
        {
            printf("\ndskin + dfat must be <=  %3.4f mm.\n",Jmax*dz*1000.0);
            printf("Press a key to continue...");
            getch();
            exit(1);
        }
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
            {
                z=(j-0.5)*dz;
                r=(i-0.5)*dr;
```

```
                if(j<=Jskin)                        /*Calculates the absorption*attenuation [1/m]*/
                    factor=2.0*alfas*Tskin*exp((double) -2.0*alfas*z);
                else if(j<=Jfat)
                    factor=2.0*alfaf*Tskin*Tfat*exp((double) -2.0*(alfas-alfaf)*Jskin*dz-2.0*alfaf*z);
                else
                    factor=2.0*alfa*Tskin*Tfat*Tmuscle*exp((double) -2.0*(alfas-alfa)*Jskin*dz-2.0*(alfaf-
alfa)*(Jfat-Jskin)*dz-2.0*alfa*z);
                if((r<=radist)&&((z+focdist-fdepth)<=axdist))   /*Coordinates within scanned area?*/
                {
                    znew=z+focdist-fdepth;
                    Iin=(short) ceil((double) r/ds);        /*Bi-linear interpolation*/
                    Jin=(short) ceil((double) znew/ds);
                    A=(1.0-(r-(Iin-1)*ds)/ds)*(1.0-(znew-(Jin-1)*ds)/ds);
                    B=(r-(Iin-1)*ds)/ds*(1.0-(znew-(Jin-1)*ds)/ds);
                    C=(znew-(Jin-1)*ds)/ds*(1.0-(r-(Iin-1)*ds)/ds);
                    D=(r-(Iin-1)*ds)*(znew-(Jin-1)*ds)/sqr(ds);

    source[i][j]=factor*(inmat[Iin][Jin]*A+inmat[Iin+1][Jin]*B+inmat[Iin][Jin+1]*C+inmat[Iin+1][Jin+1]*D);/
*[W/m3]*/
                }
                else                                    /*Outside scanned area*/
                    source[i][j]=0.0;
                if(i==1)
                    vol[i]=PI*sqr(dr)*dz;               /*Calculates the voxel volume [m3]*/
                else
                    vol[i]=2.0*PI*(i-0.5)*sqr(dr)*dz;
                adjust +=1.0E+9*vol[i]*source[i][j];    /*Calculates the total input power [W]*/
            }
    }
    else                                                /*Homogenous tissue*/
    {
        T=(4.0*DENSW*VW*dens*v)/sqr(DENSW*VW+dens*v);
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
            {
                z=(j-0.5)*dz;
                r=(i-0.5)*dr;
                factor=2.0*alfa*T*exp((double) -2.0*alfa*z);/*Calculates the absorption*attenuation [1/m]*/
                if((r<=radist)&&((z+focdist-fdepth)<=axdist))   /*Coordinates within scanned area?*/
                {
                    znew=z+focdist-fdepth;
                    Iin=(short) ceil((double) r/ds);        /*Bi-linear interpolation*/
                    Jin=(short) ceil((double) znew/ds);
                    A=(1.0-(r-(Iin-1)*ds)/ds)*(1.0-(znew-(Jin-1)*ds)/ds);
                    B=(r-(Iin-1)*ds)/ds*(1.0-(znew-(Jin-1)*ds)/ds);
                    C=(znew-(Jin-1)*ds)/ds*(1.0-(r-(Iin-1)*ds)/ds);
                    D=(r-(Iin-1)*ds)*(znew-(Jin-1)*ds)/sqr(ds);

    source[i][j]=factor*(inmat[Iin][Jin]*A+inmat[Iin+1][Jin]*B+inmat[Iin][Jin+1]*C+inmat[Iin+1][Jin+1]*D);/
*[W/m3]*/
                }
                else
                    source[i][j]=0.0;                   /*Outside scanned area*/
                if(i==1)
                    vol[i]=PI*sqr(dr)*dz;               /*Calculates the voxel volume [m3]*/
                else
                    vol[i]=2.0*PI*(i-0.5)*sqr(dr)*dz;
                adjust +=1.0E+9*vol[i]*source[i][j];    /*Calculates the total input power [W]*/
            }
    }
}/*end build*/

void regulate(void)                                 /*Regulates the source so that total input power=1W*/
{
    register short i,j;

    for(i=1;i<=Imax;i++)
        for(j=1;j<=Jmax;j++)
            source[i][j] /=adjust;                  /*[W/mm3]*/
}/*end regulate*/
```

```
void writeprz(void)
{
    FILE *testfile;
    register short i,j;

    testfile=fopen(outputfile,"w");                    /*Opens the outputfile*/
    if(simumodel==1)                                   /*Layered tissue*/
    {
        fprintf(testfile,"dr[mm]= %3.3f\n",dr*1000.0); /*Writes the data to outputfile*/
        fprintf(testfile,"Imax= %d\n",Imax);
        fprintf(testfile,"dz[mm]= %3.3f\n",dz*1000.0);
        fprintf(testfile,"Jmax= %d\n",Jmax);
        fprintf(testfile,"Jskin= %d\n",Jskin);
        fprintf(testfile,"Jfat= %d\n",Jfat);
        fprintf(testfile,"vhcaps[J/m3K]= %2.3e\n",vhcaps*denss);
        fprintf(testfile,"vhcapf[J/m3K]= %2.3e\n",vhcapf*densf);
        fprintf(testfile,"vhcap[J/m3K]= %2.3e\n",vhcap*dens);
        fprintf(testfile,"thermconds[W/mK]= %4.3f\n",thermconds);
        fprintf(testfile,"thermcondf[W/mK]= %4.3f\n",thermcondf);
        fprintf(testfile,"thermcond[W/mK]= %4.3f\n",thermcond);
        fprintf(testfile,"denss[kg/m3]= %4.1f\n",denss);
        fprintf(testfile,"densf[kg/m3]= %4.1f\n",densf);
        fprintf(testfile,"dens[kg/m3]= %4.1f\n",dens);
    }
    else                                               /*Homogenous tissue*/
    {
        fprintf(testfile,"dr[mm]= %3.3f\n",dr*1000.0); /*Writes the data to outputfile*/
        fprintf(testfile,"Imax= %d\n",Imax);
        fprintf(testfile,"dz[mm]= %3.3f\n",dz*1000.0);
        fprintf(testfile,"Jmax= %d\n",Jmax);
        fprintf(testfile,"vhcap[J/m3K]= %2.3e\n",vhcap*dens);
        fprintf(testfile,"thermcond[W/mK]= %4.3f\n",thermcond);
        fprintf(testfile,"dens[kg/m3]= %4.1f\n",dens);
    }
    for(j=1;j<=Jmax;j++)
    {
        for(i=1;i<=Imax;i++)
            fprintf(testfile,"%e\t",source[i][j]);
        fprintf(testfile,"\n");
    }
    fclose(testfile);
}/*end writeprz*/

void release_memory(void)                              /*Disengages the used memory*/
{
    FreeVector(vol,1);
    FreeMatrix(source,1,Imax,1);
    FreeMatrix(inmat,1,Iin,1);
}/*end release_memory*/

/*************************************************************
 * Report error message to stderr, then exit the program
 * with signal 1.
 *************************/
void nrerror(char error_text[])
{
    fprintf(stderr,"%s\n",error_text);
    fprintf(stderr,"...now exiting to system...\n");
    getch();
    exit(1);
}/*end nrerror*/

/*************************************************************
 * Allocate an array with index from nl to nh inclusive.
 *
 * Original matrix and vector from Numerical Recipes in C
 * don't initialize the elements to zero. This will
 * be accomplished by the following functions.
```

```
************************.
float *AllocVector(short nl, short nh)
{
    float *v;
    register short i;

    v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
    if (!v)
        nrerror("allocation failure in vector()");
    v -= nl;
    for(i=nl;i<=nh;i++)
        v[i] = 0.0;                                    /*Init*/
    return v;
}/*end AllocVector*/

/************************************************************
 * Allocate a matrix with row index from nrl to nrh
 * inclusive, and column index from ncl to nch
 * inclusive.
 ************************/
float **AllocMatrix(short ncl,short nch,short nrl,short nrh)
{
    register short i,j;
    float **m;

    m=(float **) malloc((unsigned) (nch-ncl+1)*sizeof(float *));
    if (!m)
        nrerror("allocation failure 1 in matrix()");
    m -= ncl;
    for(i=ncl;i<=nch;i++)
    {
        m[i]=(float *) malloc((unsigned) (nrh-nrl+1)*sizeof(float));
        if (!m[i])
            nrerror("allocation failure 2 in matrix()");
        m[i] -= nrl;
    }
    for(i=ncl;i<=nch;i++)
        for(j=nrl;j<=nrh;j++)
            m[i][j] = 0.0;
    return m;
}/*end AllocMatrix*/

/************************************************************
 * Release the memory.
 ************************/
void FreeVector(float *v,short nl)
{
    free((char *) (v+nl));
}/*end Free Vector*/

/************************************************************
 * Release the memory.
 ************************/
void FreeMatrix(float **m,short ncl,short nch,short nrl)
{
    register short i;

    for(i=nch;i>=ncl;i--)
        free((char *) (m[i]+nrl));
    free((char *) (m+ncl));
}/*end FreeMatrix*/


/*----------------------------END SOURCEMAKERPROGRAM----------------------------*/
```

D.2 Tempsim

```
/*-------------------------------------------------------------------------
Tempsim program

Temperature-distribution simulation
in radial (r) and axial (z) direction

This version accepts only output from the Sourcemaker program

-------------------------------------------------------------------------*/
#include <malloc.h>
#include <string.h>
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <conio.h>
/*************************************************************************
* Some mathematical and physical definitions                             *
*************************************************************************/
#define PI 3.1415926535
#define sqr(a) ((a)*(a))
#define WATERCOND 0.60              /*Water heat conductivity [W/mK]     */
#define VHCAPB 4.05E+6              /*Blood volumetric heatcapacity [J/m3K]  */
/*************************************************************************/

float *AllocVector(short nl, short nh);
float **AllocMatrix(short ncl,short nch, short nrl,short nrh);
void initial_conditions(void);
void FreeVector(float *v,short nl);
void FreeMatrix(float **m,short ncl,short nch,short nrl);
void nrerror(char error_text[]);
void select_model(void);
void communicate(void);
void readinput(void);
void init_allocation(void);
void heat_conductances(void);
void inverted_heatcapacities(void);
void stability_timestep(void);
void heat_flows(void);
void new_temperatures(void);
void tempwrite(void);
void dosewrite(void);
void maxwrite(void);
void release_memory(void);
void initwriteinterval(void);
void intervalwrite(void);

short Jmax,Jskin,Jfat,Imax,Timeinterval,stepnbr;        /*Global variables are used throughout the program*/
short countz[40],countr[40],nrprobes,choice,simumodel;
float dz,dr,rmax,zmax,arttemp,denss,densf,dens;
float ownstep,maxtime,sourcepower,vhcaps,vhcapf,vhcap;
float thermconds,thermcondf,thermcond;
float bursttime,period,treattime,inittemp,watertemp,bloodflow;
float time,timestep,tinterval,factor,maxtinv;
char temp_file[40],max_file[40],dose_file[40],probe_file[40];
float **source;     /*Heat source matrix                          */
float **temp;       /*Temperature matrix                        */
float **flowr;      /*Radial heat flow rate matrix                */
float **flowz;      /*Axial heat flow rate matrix                 */
float **dose;       /*Matrix for calculation of celldeath         */
float **maxtemp;/*Matrix for the highest temp in the voxels  */
float **lhcapinv;   /*Inverted heat capacity matrix (layermodel)*/
float **lcondr;     /*Radial conductivity matrix (layermodel)   */
float **lcondz;     /*Axial conductivity matrix (layermodel)    */
float *hcapinv;     /*Inverted heat capacity vector             */
float *condr;       /*Radial conductivity vector                  */
float *condz;       /*Axial conductivity vector                   */
float *vol;         /*Voxel volume vector                         */
```

```
float *ldensity;        /*Tissue density vector (layermodel)        */
div_t testint;

void main(void)                                          /*The main program begins*/
{
    select_model();                                      /*Lets the user choose simulation model*/
    readinput();                                         /*Lets the user set some important parameters*/
    communicate();
    init_allocation();                                   /*Some initialization*/
    initial_conditions();
    heat_conductances();
    inverted_heatcapacities();
    stability_timestep();
    printf("\nAll input is ok. Press a key to start the simulation");
    getch();
    printf("\n");
    if(choice==2)                                        /*If the user chose '2' in communicate()*/
    {
        stepnbr=0;                                       /*Some more initialization*/
        initwriteinterval();
        do
        {
            time += timestep;
            stepnbr++;                                   /*Counts the number of timesteps*/
            heat_flows();                                /*Calculates the heatflow rates*/
            new_temperatures();                          /*Calculates the temperatures*/
            testint=div(stepnbr,Timeinterval);
            if(testint.rem == 0)                         /*Time to write to probefile?*/
            {
                intervalwrite();
                printf("\nSimulated time: %3.3f s\n",time);
            }
        }
        while (time <= (maxtime-timestep/2.0));          /*Continues until time = investigationtime*/
    }
    else                                                 /*If the user chose '1' i communicate()*/
    {
        printf("\nSimulating, please wait...\n");
        do
        {
            time += timestep;
            heat_flows();                                /*Calculates the heatflow rates*/
            new_temperatures();                          /*Calculates the temperatures*/
        }
        while (time <= (maxtime-timestep/2.0));          /*Continues until time = investigationtime*/
    }
    tempwrite();                                         /*Writes the result*/
    dosewrite();
    maxwrite();
    release_memory();                                    /*"Garbage-collector"*/
    printf("\nSimulation completed! Press a key to continue...");
    getch();
    exit(0);
}/*end main*/


void select_model(void)
{
    printf("_____\
n\n");
    printf("                Temperature simulation\n");
    printf("_____\
n\n");
    printf("Type of simulationmodel:  1) layered tissue, skin-fat-muscle\n");
    printf("                          2) homogenous tissue: ");
    scanf("%d",&simumodel);                              /*Reads the simulation model*/
    if((simumodel!=1)&&(simumodel!=2))
    {
        printf("\nThe choice must be either 1 or 2! Press a key to continue...");
        getch();
```

```
        exit(1);
    }
}/*end select_model*/


void readinput(void)
{
    FILE *testfile;
    char dumpstr[25].infile[40];
    register short i,j;


    printf("\nEnter searchpath plus name of file from Sourcemaker program.\n");
    printf("ex  C:\\Program\\simulib\\source.prz\n>");
    scanf("%s",infile);                                  /*Reads the name of the inputfile*/
    testfile=fopen(infile,"r");                          /*Opens the inputfile*/
    if (testfile != NULL)
    {
        if(simumodel==1)                                 /*Layered tissue*/
        {
            fscanf(testfile,"%s",dumpstr);               /*Reads the unimportant text*/
            fscanf(testfile,"%f",&dr);                   /*Units in mm*/
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%d",&Imax);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&dz);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%d",&Jmax);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%d",&Jskin);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%d",&Jfat);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&vhcaps);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&vhcapf);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&vhcap);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&thermconds);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&thermcondf);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&thermcond);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&denss);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&densf);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&dens);
        }
        else                                             /*Homogenous tissue*/
        {
            fscanf(testfile,"%s",dumpstr);               /*Reads the unimportant text*/
            fscanf(testfile,"%f",&dr);                   /*Units in mm*/
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%d",&Imax);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&dz);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%d",&Jmax);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&vhcap);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&thermcond);
            fscanf(testfile,"%s",dumpstr);
            fscanf(testfile,"%f",&dens);
        }
        rmax=dr*(Imax-0.5);
        zmax=dz*(Jmax-0.5);
```

```
        printf("\nDimensions of voxels:  dr=%3.4f dz=%3.4f [mm]\n",dr,dz);
        printf("The area defined in the Sourcemaker program is: rmax=%3.4f zmax=%3.4f [mm]\n"
            ,rmax,zmax);
        dr /=1000.0;                                    /*Converts to SI-units [m]*/
        rmax /=1000.0;
        dz /=1000.0;
        zmax /=1000.0;
        source=AllocMatrix(1,Imax,1,Jmax);              /*Allocate memory for the source*/
        printf("\nSourcepower inside tissue [W]: ");
        scanf("%f",&sourcepower);                       /*Reads the power of the source*/
        for (j=1;j<=Jmax;j++)
            for (i=1;i<=Imax;i++)
            {
                fscanf(testfile,"%e",&source[i][j]);    /*Units in W/mm3*/
                source[i][j] *=(1.0E+9*sourcepower);    /*Converts to SI-units [W/m3]*/
            }
        fclose(testfile);
    }
    else
    {
        printf("\nFile %s not found! Press a key to continue...",infile);
        getch();
        exit(1);
    }
}/*end readinput*/


void communicate()
{
    float rcoord,zcoord;
    register short i;

    printf("\nSearchpath plus name of temperature-resultfile:\n>");
    scanf("%s",temp_file);                              /*Lets the user choose name and directory*/
    printf("\nSearchpath plus name of dose-resultfile:\n>");
    scanf("%s",dose_file);
    printf("\nSearchpath plus name of maxtemp-resultfile:\n>");
    scanf("%s",max_file);
    printf("\nOutput data    1) temperature after investigation time");
    printf("\n              2) points at a certain time interval and temperature after");
    printf("\n                 investigation time: ");
    scanf("%d",&choice);                                /*Reads the choice*/
    if((choice!=1)&&(choice!=2))
    {
        printf("\nThe choice must be either 1 or 2! Press a key to continue...");
        getch();
        exit(1);
    }
    if(choice==2)
    {
        printf("\nSearchpath plus name of probe-resultfile:\n>");
        scanf("%s",probe_file);
        printf("\nNumber of probes (max. 40): ");     /*Lets the user define a number of probes*/
        scanf("%d",&nrprobes);
        for(i=1;i<=nrprobes;i++)
        {
            printf("\nr-coordinate of probe %d [mm]: ".i);
            scanf("%f".&rcoord);                        /*Reads the r-coordinate of the probes*/
            if (rcoord == 0.0)
                countr[i-1]=1;
            else if (rcoord<=(rmax*1000.0))
                countr[i-1]=((short) ceil((double) (rcoord/(dr*1000.0))));
            else
                countr[i-1]=Imax;                       /*If rcoord > rmax*/
            printf("z-coordinate of point %d [mm]: ",i);
            scanf("%f".&zcoord);                        /*Reads the z-coordinate of the probes*/
            if (zcoord == 0.0)
                countz[i-1]=1;
            else if (zcoord<=(zmax*1000.0))
                countz[i-1]=((short) ceil((double) (zcoord/(dz*1000.0))));
```

```
            else
                countz[i-1]=Jmax;                    /*If zcoord > zmax*/
        }
        printf("\nTime interval between probe read-outs [s]: ");
        scanf("%f",&tinterval);
    }
    printf("\nTreatment time [s]: ");                 /*Reads some important simulation parameters*/
    scanf("%f",&treattime);
    printf("\nInvestigation time [s]  >= %4.3f s: ",treattime);
    scanf("%f",&maxtime);
    printf("\nBurst length [s]: ");
    scanf("%f",&bursttime);
    printf("\nTime interval between bursts [s]: ");
    scanf("%f",&period);
    printf("\nInitial temperature in tissue [deg. C]: ");
    scanf("%f",&inittemp);
    printf("\nWater temperature [deg. C]: ");
    scanf("%f",&watertemp);
    printf("\nBloodflowrate in tissue [ml/100 g/min]: ");
    scanf("%f",&bloodflow);
    bloodflow /=6.0E+6;                              /*Converts to [m3/kg/s]*/
    printf("\nArterial bloodtemperature [deg. C]: ");
    scanf("%f",&arttemp);
}/*end communicate*/


void init_allocation(void)                           /*Allocate big matrices*/
{
    if(simumodel==1)                                 /*Layered tissue*/
    {
        lhcapinv=AllocMatrix(1,Imax,1,Jmax);
        lcondr=AllocMatrix(1,Imax+1,1,Jmax);
        lcondz=AllocMatrix(1,Imax,1,Jmax+1);
        ldensity=AllocVector(1,Jmax);
    }
    else                                             /*Homogenous tissue*/
    {
        hcapinv=AllocVector(1,Imax);
        condr=AllocVector(1,Imax+1);
        condz=AllocVector(1,Imax);
    }
    vol=AllocVector(1,Imax);
    flowr=AllocMatrix(1,Imax+1,1,Jmax);
    flowz=AllocMatrix(1,Imax,1,Jmax+1);
    temp=AllocMatrix(1,Imax,1,Jmax);
    dose=AllocMatrix(1,Imax,1,Jmax);
    maxtemp=AllocMatrix(1,Imax,1,Jmax);
}/* end init_allocation*/


void initial_conditions(void)
{
    register short j,i;

    time = 0.0;
    for (i=1;i<=Imax;i++)
        for (j=1;j<=Jmax;j++)
        {
            if (i==1)
                vol[i]=PI*sqr(dr)*dz;                /*Volume [m3] for the voxels closest to the z-axis*/
            else
                vol[i]=2.0*PI*(i-0.5)*sqr(dr)*dz;    /*Volume calculation for the other voxels */
            if(simumodel==1)                         /*Layered tissue*/
                if(j<=Jskin)
                    ldensity[j]=denss;               /*Initiates the density vector*/
                else if (j<=Jfat)
                    ldensity[j]=densf;
                else
                    ldensity[j]=dens;
            temp[i][j] = inittemp;                   /*Initiates some matrices*/
```

```
            maxtemp[i][j]=inittemp;
            source[i][j]*=vol[i];                    /*[W]*/
        }
}/*end initial conditions*/


void heat_conductances(void)                         /*Calculates the heat conductances [W/K]*/
{
    register short i,j;

    if(simumodel==1)                                 /*Layered tissue*/
    {
        for(i=1;i<=(Imax+1);i++)
            for(j=1;j<=(Jmax+1);j++)
            {
                if(i<=Imax)
                    if(j==1)

    lcondz[i][j]=2.0*WATERCOND*thermconds*vol[i]/(sqr(dz)*(WATERCOND+thermconds));
                    else if(j<=Jskin)
                        lcondz[i][j]=thermconds*vol[i]/sqr(dz);
                    else if(j==(Jskin+1))
                        lcondz[i][j]=2.0*thermconds*thermcondf*vol[i]/(sqr(dz)*(thermconds+thermcondf));
                    else if(j<=Jfat)
                        lcondz[i][j]=thermcondf*vol[i]/sqr(dz);
                    else if(j==(Jfat+1))
                        lcondz[i][j]=2.0*thermcondf*thermcond*vol[i]/(sqr(dz)*(thermcondf+thermcond));
                    else
                        lcondz[i][j]=thermcond*vol[i]/sqr(dz);
                if(j<=Jmax)
                    if(i==1)
                        lcondr[i][j]=0.0;            /*Symmetry reason*/
                    else if(j<=Jskin)
                        lcondr[i][j]=2.0*PI*dz*thermconds/(log((double) ((i-0.5)/(i-1.5))));
                    else if(j<=Jfat)
                        lcondr[i][j]=2.0*PI*dz*thermcondf/(log((double) ((i-0.5)/(i-1.5))));
                    else
                        lcondr[i][j]=2.0*PI*dz*thermcond/(log((double) ((i-0.5)/(i-1.5))));
            }
    }
    else                                             /*Homogenous tissue*/
        for(i=1;i<=(Imax+1);i++)
        {
            if (i==1)
                condr[i]=0.0;                        /*Symmetry reason*/
            else
                condr[i]= 2.0*PI*dz*thermcond/(log((double) ((i-0.5)/(i-1.5))));
            if(i<=Imax)
                condz[i] = thermcond*vol[i]/sqr(dz);/*The fact thermcond <> WATERCOND is disregarded*/
        }
}/*end heat_conductances*/


void inverted_heatcapacities(void)                   /*Calculates the hcapinv [K/J] once and for all*/
{
    register short i,j;

    if(simumodel==1)                                 /*Layered tissue*/
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
                if(j<=Jskin)
                    lhcapinv[i][j]=1.0/(vhcaps*vol[i]);
                else if(j<=Jfat)
                    lhcapinv[i][j]=1.0/(vhcapf*vol[i]);
                else
                    lhcapinv[i][j]=1.0/(vhcap*vol[i]);
    else                                             /*Homogenous tissue*/
        for(i=1;i<=Imax;i++)
            hcapinv[i] = 1.0/(vhcap*vol[i]);
}/*end inverted_heatcapacities*/
```

```
void stability_timestep(void)
{
    float tinv;
    register short i,j;

    maxtinv = 0.0;
    if(simumodel==1)                                    /*Layered tissue*/
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
            {
                tinv=lhcapinv[i][j]*(lcondr[i][j]+lcondr[i+1][j]+lcondz[i][j]+lcondz[i][j+1]);
                if(tinv>maxtinv)
                    maxtinv=tinv;                       /*Gives the maximum value of tinv [1/s]*/
            }
    else                                                /*Homogenous tissue*/
        for(i=1;i<=Imax;i++)
            {
                tinv = hcapinv[i]*(condr[i]+condr[i+1]+2.0*condz[i]);
                if (tinv>maxtinv)
                    maxtinv=tinv;                       /*Gives the maximum value of tinv [1/s]*/
            }
    timestep = 0.99/maxtinv;                            /*Some safety margin*/
    printf("\nTimestep [s] (type -1 to use system calculated timestep= %2.5f s): ",timestep);
    scanf("%f",&ownstep);
    if ((ownstep<timestep)&&(ownstep>0.0))
        timestep=ownstep;
    printf("\nSimulation timestep = %2.6f  sec.\n",timestep);
    if (choice==2)
        if(tinterval>timestep)
            if (((fmod( tinterval,timestep))/timestep)>=0.5)
                Timeinterval=((short) ceil((double) (tinterval/ timestep)));     /*Calculates the number of
timesteps between read-outs*/
            else
                Timeinterval=((short) floor((double) (tinterval/ timestep)));
        else
            Timeinterval=1;
}/*end stability_timestep*/


void heat_flows(void)                                   /*Calculates the heatflow rates [W]*/
{
    register short i,j;

    if(simumodel==1)                                    /*Layered tissue*/
        for(i=1;i<=(Imax+1);i++)
            for(j=1;j<=(Jmax+1);j++)
            {
                if(j<=Jmax)                             /*The r-direction*/
                    if(i==1)
                        flowr[i][j]=0.0;                /*Symmetry reason*/
                    else if(i<=Imax)
                        flowr[i][j]=lcondr[i][j]*(temp[i-1][j]-temp[i][j]);
                    else
                        flowr[i][j]=0.0;                /*Very little flow assumed*/
                if(i<=Imax)                             /*The z-direction*/
                    if(j==1)
                        flowz[i][j]=lcondz[i][j]*(watertemp-temp[i][j]);
                    else if(j<=Jmax)
                        flowz[i][j]=lcondz[i][j]*(temp[i][j-1]-temp[i][j]);
                    else
                        flowz[i][j]=0.0;                /*Very little flow assumed*/
            }
    else                                                /*Homogenous tissue*/
        for(i=1;i<=(Imax+1);i++)
            for(j=1;j<=(Jmax+1);j++)
            {
                if(j<=Jmax)                             /*The r-direction*/
                    if(i==1)
```

```
                    flowr[i][j]=0.0;                    /*Symmetry reason*/
                else if(i<=Imax)
                    flowr[i][j]=condr[i]*(temp[i-1][j]-temp[i][j]);
                else
                    flowr[i][j]=0.0;                    /*Very little flow assumed*/
            if(i<=Imax)                                 /*The z-direction*/
                if(j==1)
                    flowz[i][j]=condz[i]*(watertemp-temp[i][j]);
                else if(j<=Jmax)
                    flowz[i][j]=condz[i]*(temp[i][j-1]-temp[i][j]);
                else
                    flowz[i][j]=0.0;                    /*Very little flow assumed*/
        }
}/*end heat_flows*/


void new_temperatures(void)                             /*Calculates the new temperature in the voxels*/
{
    register short i,j;

    if (time>treattime)                                 /*Decides whether the source is on or off*/
        factor=0.0;
    else if ((fmod(time,period)>=(timestep/2.0))&&(fmod(time,period)<=(bursttime+timestep/2.0)))
        factor=1.0;
    else
        factor=0.0;
    if(simumodel==1)                                    /*Layered tissue*/
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
            {
                temp[i][j] +=
                lhcapinv[i][j]*(flowr[i][j]-flowr[i-1][j]+flowz[i][j]-flowz[i][j+1]+
                factor*source[i][j]+bloodflow*VHCAPB*ldensity[j]*vol[i]*(arttemp-temp[i][j]))*timestep;
                if (dose[i][j]<=2.0)
                    dose[i][j] +=timestep*3.10*exp(225.65333911-75537.64734183/
                        (temp[i][j]+273.15));           /*Coagulation calculation, a rate process*/
                if (temp[i][j]>maxtemp[i][j])
                    maxtemp[i][j]=temp[i][j];           /*Records the highest temp in the voxels*/
            }
    else                                                /*Homogenous tissue*/
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
            {
                temp[i][j] +=
                hcapinv[i]*(flowr[i][j]-flowr[i+1][j]-flowz[i][j]-flowz[i][j+1]+
                factor*source[i][j]+bloodflow*VHCAPB*dens*vol[i]*(arttemp-temp[i][j]))*timestep;
                if (dose[i][j]<=2.0)
                    dose[i][j] +=timestep*3.10*exp(225.65333911-75537.64734183/
                        (temp[i][j]+273.15));           /*Coagulation calculation, a rate process*/
                if (temp[i][j]>maxtemp[i][j])
                    maxtemp[i][j]=temp[i][j];           /*Records the highest temp in the voxels*/
            }
}/*end new_temperatures*/


void tempwrite(void)
{
    FILE *testfile;
    register short i,j;
    float dist,energy;

    if(simumodel==1)                                    /*Layered tissue*/
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
                energy +=(temp[i][j]-inittemp)/lhcapinv[i][j];   /*Calculates the excessive heat [J]*/
    else                                                /*Homogenous tissue*/
        for(i=1;i<=Imax;i++)
            for(j=1;j<=Jmax;j++)
                energy +=(temp[i][j]-inittemp)/hcapinv[i];  /*Calculates the excessive heat [J]*/
    testfile=fopen(temp_file,"w");                      /*Opens the the temperature file*/
```

```
    fprintf(testfile,"Temperature file\nThe first row and column gives the middle\ncoordinates of the voxel in
[mm]\nThe matrix gives the actual temperature in deg. C\n");
    fprintf(testfile,"Excessive heat in tissue= %4.4f [J]\n",energy);
    fprintf(testfile,"\n0");
    for(i=1;i<=Imax;i++)
    {
        dist=(i-0.5)*dr*1000.0;
        fprintf(testfile,"\t%3.4f",dist);                    /*Writes the first row, the r-axis*/
    }
    fprintf(testfile,"\n");
    for(j=1;j<=Jmax;j++)
    {
        dist=(j-0.5)*dz*1000.0;
        fprintf(testfile,"%3.4f",dist);                      /*Writes the current z-coordinate...*/
        for(i=1;i<=Imax;i++)
            fprintf(testfile,"\t%e",temp[i][j]);             /*... followed by the simulated temperature in the voxels*/
        fprintf(testfile,"\n");
    }
    fclose(testfile);
}/*end tempwrite*/

void dosewrite(void)
{
    FILE *testfile;
    register short i,j;
    float dist,killvol;

    for(i=1;i<=Imax;i++)
        for(j=1;j<=Jmax;j++)
            if (dose[i][j]>=1.0)
                killvol +=vol[i]*1.0E+6;                     /*Calculates the coagulated volume [cm3]*/
    testfile=fopen(dose_file,"w");                           /*Opens the the dose file*/
    fprintf(testfile,"Dose file\nThe first row and column gives the middle\ncoordinates of the voxel in [mm]\n");
    fprintf(testfile,"Coagulated volume= %4.4f [cm3]\n",killvol);
    fprintf(testfile,"\n0");
    for(i=1;i<=Imax;i++)
    {
        dist=(i-0.5)*dr*1000.0;
        fprintf(testfile,"\t%3.4f",dist);                    /*Writes the first row, the r-axis*/
    }
    fprintf(testfile,"\n");
    for(j=1;j<=Jmax;j++)
    {
        dist=(j-0.5)*dz*1000.0;
        fprintf(testfile,"%3.4f",dist);                      /*Writes the current z-coordinate...*/
        for(i=1;i<=Imax;i++)
            if (dose[i][j]<=1.0)
                fprintf(testfile,"\t%3.4e",dose[i][j]);      /*... followed by the data in the dose-matrix*/
            else
                fprintf(testfile,"\t%3.4e",1.0);
        fprintf(testfile,"\n");
    }
    fclose(testfile);
}/*end dosewrite*/

void maxwrite(void)
{
FILE *testfile;
    register short i,j;
    float dist;

    testfile=fopen(max_file,"w");                            /*Opens the the maximum-temperature file*/
    fprintf(testfile,"Maximum temperature file\nThe first row and column gives the middle\ncoordinates of the
voxel in [mm]\nThe matrix gives the temperature in deg. C\n");
    fprintf(testfile,"\n0");
    for(i=1;i<=Imax;i++)
    {
        dist=(i-0.5)*dr*1000.0;
        fprintf(testfile,"\t%3.4f",dist);                    /*Writes the first row, the r-axis*/
    }
```

```
    fprintf(testfile,"\n");
    for(j=1;j<=Jmax;j++)
    {
        dist=(j-0.5)*dz*1000.0;
        fprintf(testfile,"%3.4f",dist);                    /*Writes the current z-coordinate...*/
        for(i=1;i<=Imax;i++)
            fprintf(testfile,"\t%e",maxtemp[i][j]);/*... followed by the maximum-temperature in the voxels*/
        fprintf(testfile,"\n");
    }
    fclose(testfile);
}/*end maxwrite*/

void initwriteinterval(void)                               /*Prepares the probe_file for interval writing*/
{
    FILE *testfile;
    register short i;

    testfile=fopen(probe_file,"w");                        /*Opens the the probe_file*/
    fprintf(testfile,"Probe file\nTemperature [deg. C] sensed at\n\n");
    fprintf(testfile,"Time");
    for(i=1;i<=nrprobes;i++)                               /*Writes the coordinates for the given probes*/

        fprintf(testfile,"\t[r=%3.4f ,z=%3.4f]",(countr[i-1]-0.5)*dr*1000.0,(countz[i-1]-0.5)*dz*1000.0);
    fprintf(testfile,"\n0");
    for(i=1;i<=nrprobes;i++)
        fprintf(testfile,"\t%3.3f",inittemp);              /*The first row is the init. temperature*/
    fprintf(testfile,"\n");
    fclose(testfile);
}/*end initwriteinterval*/


void intervalwrite(void)
{
    FILE *testfile;
    register short i;

    testfile=fopen(probe_file,"a");                        /*Opens the probe_file to append the probe data*/
    fprintf(testfile,"%3.5f",time);
    for(i=1;i<=nrprobes;i++)                               /*Writes the temperature sensed by the given probes*/
        fprintf(testfile,"\t%3.3f",temp[countr[i-1]][countz[i-1]]);
    fprintf(testfile,"\n");
    fclose(testfile);
}/*end intervalwrite*


void release_memory(void)                                  /*Disengages the used memory*/
{
    if(simumodel==1)                                       /*Layered tissue*/
    {
        FreeMatrix(lhcapinv,1,Imax,1);
        FreeMatrix(lcondr,1,Imax+1,1);
        FreeMatrix(lcondz,1,Imax,1);
        FreeVector(ldensity,1);
    }
    else                                                   /*Homogenous tissue*/
    {
        FreeVector(hcapinv,1);
        FreeVector(condr,1);
        FreeVector(condz,1);
    }
    FreeVector(vol,1);
    FreeMatrix(flowr,1,Imax+1,1);
    FreeMatrix(flowz,1,Imax,1);
    FreeMatrix(temp,1,Imax,1);
    FreeMatrix(source,1,Imax,1);
    FreeMatrix(maxtemp,1,Imax,1);
    FreeMatrix(dose,1,Imax,1);
}/*end release_memory*/
```

```
/*************************************************************
 * Report error message to stderr, then exit the program
 * with signal 1.
 *************************/
void nrerror(char error_text[])
{
    fprintf(stderr,"%s\n",error_text);
    fprintf(stderr,"...now exiting to system...\n");
    getch();
    exit(1);
}/*end nrerror*/

/*************************************************************
 * Allocate an array with index from nl to nh inclusive.
 *
 * Original matrix and vector from Numerical Recipes in C
 * don't initialize the elements to zero. This will
 * be accomplished by the following functions.
 *************************/
float *AllocVector(short nl, short nh)
{
    float *v;
    register short i;

    v=(float *)malloc((unsigned) (nh-nl+1)*sizeof(float));
    if (!v)
        nrerror("allocation failure in vector()");
    v -= nl;
    for(i=nl;i<=nh;i++)
        v[i] = 0.0;                                    /*Init*/
    return v;
}/*end AllocVector*/

/*************************************************************
 * Allocate a matrix with row index from nrl to nrh
 * inclusive, and column index from ncl to nch
 * inclusive.
 *************************/
float **AllocMatrix(short ncl,short nch,short nrl,short nrh)
{
    register short i,j;
    float **m;

    m=(float **) malloc((unsigned) (nch-ncl+1)*sizeof(float *));
    if (!m)
        nrerror("allocation failure 1 in matrix()");
    m -= ncl;
    for(i=ncl;i<=nch;i++)
    {
        m[i]=(float *) malloc((unsigned) (nrh-nrl+1)*sizeof(float));
        if (!m[i])
            nrerror("allocation failure 2 in matrix()");
        m[i] -= nrl;
    }
    for(i=ncl;i<=nch;i++)
        for(j=nrl;j<=nrh;j++)
            m[i][j] = 0.0;
    return m;
}/*end AllocMatrix*/

/*************************************************************
 * Release the memory.
 ************************/
void FreeVector(float *v,short nl)
{
    free((char *) (v+nl));
}/*end Free Vector*/

/*************************************************************
 * Release the memory.
```

```
**********/
void FreeMatrix(float **m,short ncl,short nch,short nrl)
{
    register short i;

    for(i=nch;i>=ncl;i--)
        free((char *) (m[i]+nrl));
    free((char *) (m+ncl));
}/*end FreeMatrix*/


/*-----------------------------END TEMPSIM-----------------------------*/
```

Appendix E Manuals

When *Sourcemaker* or *Tempsim* are launched, a Dos-window will appear on the screen and the user is asked, step-by-step, to enter information via the keyboard.

E.1 Sourcemaker

1. Create a parameter file with desired tissue properties. The design must be identical to example 3B in appendix B if the model should be homogenous or identical to example 4B if the model should be layered.

2. Start Sourcemaker.

3. Choose the desired model: 1 - layered model, 2 - homogenous model (Ex. "2↵").

4. Enter searchpath and name of intensity map file that should be used as a source (Ex. "C:\my_directory\scan1.asc↵").

5. Enter the radial scan distance associated with the particular intensity map. Data are found in appendix A, table 2A (Ex. "25.2↵").

6. Enter the axial scan distance associated with the particular intensity map. Data are found in appendix A, table 2A (Ex. "65↵").

7. Enter the step size associated with the particular intensity map. Data are found in appendix A, table 2A (Ex. "0.33↵").

8. Enter the position of the geometrical focus associated with the particular intensity map. Data are found in appendix A, table 2A (Ex. "40↵").

9. Enter the desired position of the geometrical focus to be applied in the model (Ex. "15.0↵").

10. Enter the desired radial extension of the simulation domain (Ex. "30↵").

11. Enter the desired radial voxel size (Ex. "0.2↵").

12. Enter the desired axial extension of the simulation domain (Ex. "45↵").

13. Enter the desired axial voxel size (Ex. "0.2↵").

14. Enter searchpath and name of the parameter file, created under item 1 (Ex. "C:\my_directory\homogenous.par↵").

15. Enter the desired searchpath and name of the created heat source file. Note, if the searchpath is omitted, the file will be created in the same directory as the Sourcemaker program (Ex. "source.prz↵").

16. Press a key to continue... *(this terminates Sourcemaker).*

E.2 Tempsim

1. Start Tempsim.

2. Choose the desired model: 1 - layered model, 2 - homogenous model (Ex. "2↵").

3. Enter searchpath and name of heat source file that should be used as a heat source in the simulation (Ex. "C:\my_directory\source.prz↵").

4. Enter the desired power, inside the tissue, of the heat source (Ex."38.0↵").

5. Enter the desired searchpath and name of the temperature distribution file. Note, if the searchpath is omitted, the file will be created in the same directory as the Tempsim program (Ex. "temp.txt↵").

6. Enter the desired searchpath and name of the coagulation (dose) distribution file. Note, if the searchpath is omitted, the file will be created in the same directory as the Tempsim program (Ex. "dose.txt↵").

7. Enter the desired searchpath and name of the maximum temperature distribution file. Note, if the searchpath is omitted, the file will be created in the same directory as the Tempsim program (Ex. "max.txt↵").

8. Choose whether imaginary temperature probes should be defined or not: 1 - output according to item 5-7 only, 2 - the same as (1) plus probe temperatures (Ex. "2↵"). *If choice #1 is entered ⇒ proceed with item 15.*

9. Enter the desired searchpath and name of the probe file. Note, if the searchpath is omitted, the file will be created in the same directory as the Tempsim program (Ex. "probe.txt↵").

10. Enter the desired total number of probes to be defined (Ex. "2↵").

11. Enter the desired radial coordinate of probe #1 (Ex. "0.0↵").

12. Enter the desired axial coordinate of probe #1 (Ex. "15.9↵").

13. *If more than one probe was entered ⇒ repeat step 11-12 until all the probes have been defined (...#1 should be replaced by ...#2 and so on).*

14. Enter the desired time interval between probe read-outs (Ex. "0.25↵").

15. Enter the desired treatment time, US_{stop} (Ex."300↵").

16. Enter the desired investigation time, *simulation stop* (Ex. "330↵").

17. Enter the desired burst duration (Ex. "1.38↵").

18. Enter the desired time between bursts, *repetition period* (Ex. "10.12↵").

19. Enter the initial tissue temperature (Ex. "24↵").

20. Enter the desired water temperature (Ex. "10↵").

21. Enter the desired blood flow rate (Ex. "0.0↵").

22. Enter the desired arterial blood temperature (Ex. "0.0↵").

23. Enter the desired time step to be applied provided it is smaller than the system calculated time step (Ex. "0.07↵"). Note, if the system calculated time step is to be applied, type "-1↵".

24. Press a key to start the simulation *(the simulation starts)*.

25. Simulation completed! Press a key to continue... *(this terminates Tempsim)*.