FACULTY OF ENGINEERING - UNIVERSITY OF PORTO

Mathematical Modelling of Human Thermoregulation in Hot Environments

A thesis submitted for the degree of

PhilosophiæDoctor (PhD) in Occupational Health and Safety

18th January 2016

Author:
J. C. GUEDES

Supervisor:
Prof. Dr. J. SANTOS BAPTISTA

Co- Supervisor:
Prof. Dr. J. TORRES COSTA

Abstract

Modelling human thermal response is not a new research study field. There are many studies concerning with the human thermophysiological responses but only a few integrate transient and three-dimensional whole-body responses to heat exposure. The main goal of this work was to integrate several mathematical modelling approaches based on the best current rational models available and presents a new model implemented in a modern programming language that could support a sustainable development. Bibliographic review allowed gathering information about the best rational approaches to define the mathematical model. The search for programming languages elects Python as the new generation language used in scientific simulation. Implementation was based on the best numerical systems, such as numpy, scipy and fipy, as commonly used by scientific communities. The code was implemented considering a three dimensional diffusion. Last model version is presented in form of annotated listings. The final structure followed an object oriented approach, allowing a high level of parametrization. It is based on 15 segments (head, neck, trunk, upper arms, forearm, hands , thighs, legs and feet). Right limbs are distinguished from the left. Initial conditions can be settled to each type of biological tissue at each segment of the body. Boundary values change over the time at each segment of the mesh. Current implementation gives the output responses of a mean male subject at thermoneutral basal conditions. However, the final passive system implementation is prepared to respond to simulations of three-dimensional heat diffusion in transient environments. Results are given under the form of '.TSV' text files, bi-dimensional plots or animation of 4-dimensional plots. Active system mathematical model approaches are presented and exemplification of how it should be implemented in presented and suggestions for future updates based on current development are also defined. The passive system was tested to find faults and parametrization difficulties, before connect the passive to the active system.

Key-Words: Whole-body thermal modelling, Annotated listings.

Resumo

A modelação da termorregulação humana não é um novo campo de investigação. Contudo apenas alguns modelos integram numa abordagem tridimensional de corpo inteiro, os estados transitórios da exposição ao calor. Assim, o principal objetivo do trabalho consiste em integrar diversas abordagens de modelação matemática, com base nos melhores modelos disponíveis, apresentando um modelo integrado, implementado numa linguagem que pudesse apoiar o seu desenvolvimento sustentável. O modelo matemático foi definido a partir das melhores soluções encontradas na revisão bibliográfica. O Python foi escolhido por ser a linguagem de programação de nova geração usada em simulação científica. A implementação foi baseada em ferramentas de cálculo numérico, como por exemplo numpy, scipy e fipy, utilizados habitualmente pela comunidade científica . A implementação considerou uma difusão tridimensional do calor. O modelo é apresentado na forma de itens anotados. A estrutura final seguiu uma abordagem orientada a objetos, utilizando dezenas de parâmetros para cada um dos 15 segmentos (cabeça, pescoço, tronco, braços, antebraços, mãos, coxas, pernas e pés). Os membros direitos são diferenciados dos esquerdos. As condições iniciais podem ser descritas para cada tipo de tecido biológico em cada um dos segmentos do corpo. Os valores limite mudam ao longo do tempo em cada segmento da malha permitindo observação dos regimes transitórios. A implementação atual dá a resposta de um indivíduo do sexo masculino médio, em condições basais termoneutrais. Contudo, a implementação final do sistema passivo já está preparada para responder à difusão de calor tridimensional em ambientes transitórios. Os resultados são apresentados sob a forma de arquivos ".TSV" de texto, gráficos bidimensionais e animação com 4-dimensões. É apresentada uma abordagem matemática do modelo do sistema ativo e é exemplificado o modo como deve ser implementado, com sugestões para futuras atualizações com base no conhecimento atual. O sistema passivo foi testado no sentido de procurar eventuais falhas e dificuldades de parametrização, previamente à implementação do sistema ativo de termorregulação.

Palavras-Chave: Modelação térmica de corpo inteiro, código anotado.

“Whoever gives reverence to those worthy of reverence,
whether the awakened or their disciples,
those who have overcome the army
and crossed the river of sorrow,
whoever gives reverence to such as have found deliverance
and are free of fear,
their merit cannot be measured by anyone.”

Lord Buddha

*T*o my Family,
with Love and Faith,

J.C. Guedes

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Human thermal modelling starts many years ago with the definition of the bioheat equation in 1948 (Pennes, 1948). Since then, many studies have been developed trying to simulate biological heat transfer and physiological response. Several whole-body models were developed that include physiological parameters and predict sweat rate, heart rate, core and shell temperatures.

Different research guidelines were created and followed in whole-body and local thermal modelling. Some try to map temperatures all over the body (Wissler, 1964; Fiala et al., 1999; Ferreira and Yanagihara, 2009; Albuquerque-Neto and Yanagihara, 2009), others use global heat balance principles to predict the amount of body heat gain/loss to predict other physiological parameters (Givoni and Goldman, 1972; Pandolf et al., 1986; Fanger and Toftum, 2002).

Human thermal models are quite important for research and practice in health and safety at work. When complete and integrated with a thermoregulatory system models can be used to evaluate:

- Critical exposure to high or low temperatures;
- Long term exposure periods;
- Unpleasant submersions in cold or hot fluids;
- Whole-body discomfort, local discomfort or pre-burning stage exposure;
- Exposure time period limits;

- Sweat rate and dehydration level;
- Physiological indicators of excess of thermoregulation;
- Overheat process and individual heat tolerance limits;
- and so on.

The biggest advantage of this kind of research is to avoid the need to submit people to unpleasant and/or unsafe conditions to analyse individual physiological responses. Models allow to test quickly different exposure conditions. Tests with models are low cost comparatively to experimental research. Assuring a correct and rigorous validation process, results can be as reliable as those obtained with the most advanced experimental techniques. In practice, models can be a competitive management tool that can help engineers and managers in planning and testing work conditions, even before implementing solutions, without submit anyone at risk or under an unpleasant situation.

Exmples of Successful Models

Givoni and Goldman (1972) and Pandolf et al. (1986) developed and applied predictive models that combine statistical black-box modelling with parameter related throughout a phenomenological basis. Models were specifically developed, tested and validated for military enforcement. The final adjusts to real data were accurate enough and simple, what make them easy to be used in the field as important tactical decision making tool. It relates physiological parameters (core temperature, heart rate and sweat rate) with the time, which allow to build a response profile over the time.

Pandolf et al. (1986) model was tested and validated with a large number of samples and data from many critical conditions collected in controlled environment and in the operational field. Givoni and Goldman (1972) model was validated by comparison with experimental data collected in controlled environment only.

In spite of both models had been an application success, other scientific areas required more sophisticated, accurate and even more reliable models. The inclusion of individual aspects and health condition help to predict the decrease in physiological tolerance that is, in the case of health and safety at work, important for special populations during dangerous works, like firefighters, mineworkers, divers, pilots,soldiers or even astronauts. But, even to accomplish the increase demand for production and quality, the development of models capable to respond to changes in time and space increase the potential for product design, building comfort environment and energy efficiency, or even assess outdoor thermal environments.

The development of whole-body thermal models to predict thermal and physiological responses to heat, are not new in this field of research. In fact, in the time line, rational model were the first ones to appear, and still most suitable for appliance in a wide range of environmental exposures and give the most reliable predictions of thermophysiological responses.

Although there is a large quantity of models already developed there is no free program code published or available, with the exception of Stolwijk (1971), Gagge et al. (1972) (whose complete Fortran annotated listings were published in final report version to NASA and scientific paper respectively) and Fiala et al. (1999), that have is UTCI-model available for public release in 2008 (beta version) being the final version presented in the subsequent year.

Wissler (1964) model was and it still is, with the upgrades made in 1985, one of the most complete thermal models, that include the simulations of oxygen and carbon dioxide mass exchange, lactate production and distribution along the body. Regardless of that fact, disadvantages come from unavailable information to allow a complete reproduction of the model.

Easy access to full code of some models, easier to implement, and the difficulty to recreate the others, due to the complexity and lack of information, lead most of subsequent researchers to use simpler versions re-adapted, instead of others sophisticated approaches. This is probably the reason that took more complex models, with higher level of mathematical description associated to human heat exchange phenomena, being put aside.

Even so, none of the mentioned models were able to join three-dimensionality to transient, non-homogeneous environments, as the present study did with BioHeatSIM model proposal.

Chapter 2

State of Art

Human thermal modelling is not a new research field. Since the last seven decades that many models have been developed and implemented in practice. From simpler empirical models – as effective temperature (ET), wet bulb globe temperature (WBGT) and heat strain index (HSI) (Belding and Hatch, 1955) – to complex rational ones - Stolwijk and Hardy 25-node model, Gagge and Nishi 2-node model or Wissler model – a large path has been trilled in order to get the most accurate, reliable and effective model.

Simple indexes have been developed based on environmental and basic physiologic parameters. Soon it was noticed that as more basic it was the index/model more difficult it was to adapt it to the different circumstances. Until the present data, in particular indexes are divided according to the type of environment. Heat exposure, cold exposure and thermal comfort are still the three main areas of modelling. Even now, the performance of modern rational models are not similar to all kind of environments or physical efforts.

With computer science advances thermal modelling also increase the potential to developed more complex models. With complexity, increased enforcement potential, so, there was a natural tendency to follow rational modelling in the last 10 years.

Currently, to fit specific requirements for individual differences and escape to the average response prediction, black box modelling techniques are being added to rational models, creating modern actual hybrid models. According to Gonzalez (2004), research has shown that data representing a random or stochastic physical phenomenon cannot be wholly described by an explicit mathematical relationship structured around the average response criteria because each observation of the phenomenon will be unique. So, the inclusions of probabilistic models or neural networks are being used to increase the predictive power of the older phenomenological approaches.

The next sections intend to provide a historical perspective of whole body thermal modelling research for predicting thermal and physiological responses of human body in steady or transient states with a particular attention to heat stress exposure. It is intended to describe the most important models in each research approach: from the most simplest to the most sophisticated; models based on different concepts regarding different types of field enforcement; and finally give a holistic perspectives of what could be done in this research field in order to justify current modelling options.

2.1 Empirical Models

Empirical models can be developed by exposing human subjects to a range of thermal environments and "fitting" mathematical models to the human response data collected (Parsons, 2014). These models are considered operational in the sense that they predict a series of physiologic responses from empirically derived equations calculated assuming that the exposure to an environmental challenge is for a finite interval of time. Mathematical equations, of rational models, can be seen as an extension of thermal balance equations or use the classical formulation of heat diffusion. The main goal is to maintain thermal neutrality between acceptable ranges. The highest quality models that have been developed in this area that allow a time dependent prediction in steady and transient states will be presented.

From the pioneers Givoni and Goldman (1972) to the latest revision of evaporative required sweat rate by Malchaire et al. (2001) and purpose of predicted heat strain (PHS), it is shown that this practical tools are easy to implement. To reproduce and implement empirical models it is not needed expert skills in physics, thermodynamics, mathematics or computational. Of course that the potential for practical application in specific conditions is limited, but when accuracy is not a priority and generic evaluations fit the requirements, these models can be useful.

2.1.1 Givoni and Goldman Model (1972-1973)

Givoni and Goldman (1972, 1973b) were pioneers at Ergonomic Division of the US Army Institute of Environmental Medicine. They establish an empirical mathematical model, easy to implement, with a mix of phenomenological base and statistical adjustments. Model consists in distinct equations to predict core temperature applied considering if the person was exercising, resting or recovering. It is necessary to have information about the task energy expenditure and clothing insulation. Core temperature at zero seconds is used as initial condition. Final result is the core temperatures over the time.

The application of the model is quite restrictive. As other empirical models, it only follows the tendency inside the range to which was designed.

2.1.2 Pandolf Model (1986)

Pandolf et al. (1986) specifically gather information from research team of Ergonomic Division of the US Army Institute of Environmental Medicine, where he was director at the time. He published the mathematical equation for predicting rectal temperature, heart rate and sweat loss as function of work intensity and clothing ensemble. Model was based on Givoni and Goldman (1972, 1973b) and Shapiro et al. (1982). He also include further specific information from research team that include solar heat load (Breckenridge and Goldman, 1971), influence of acclimation to heat (Givoni and Goldman, 1973a) and energy expenditure (Pandolf et al., 1977).

2.1.3 Malchaire Model (2000)

Malchaire et al. (2001) predicted heat strain model (PHS) it was developed by large research team composed by Belgian, German, Swedish, Netherlander, British, and Italian elements. Important researchers like K. Parsons and G. Havenith, whose models and extensive work in human thermal physiologic responses are largely known, have participated. PHS was a product of the revised theoretical concepts and modern scientific knowledge of international standard ISO 7933:1989 – required sweat rate index.

Model was developed and validated considering a database with 909 laboratory and field experiments collected from partners. The problem it was that part of the data were used to develop PHS and the other part was used to validate it.

2.2 Rational Models

Rational models most of the times can be seen as an extension of the heat balance equations used to describe rational thermal indexes. However the expression is nowadays commonly used to describe dynamic mathematical simulation of the human thermal and physiological responses. Usually consists in two systems modelled in separate, the controlled or passive system, and the active or control system. The complexity of mathematical simulations should be, somehow, related to technological developments, however here is going to be seen that the most complex mathematical models were described in the beginning of 60's and 70's, and their phenomenological basis are followed by recent researchers.

2.2.1 Wissler Model (1961-1985)

From Department of Chemical Engineering of the University of Texas, Wissler is one of most important researchers in thermophysiological modelling. He starts earlier studying deeply the physiological concepts of human thermoregulation and applying them in a revolutionary way. The base of the described model is completely phenomenological, with a high level of detail, particularly for the time. Model development was supported by the office of Surgeon General of the united States Army (Wissler, 1964).

Before him, the developed mathematical models were based on simple equations, using a core-shell approach. According to the author, the most sophisticated previous works have been developed by: (1) C.H. Wyndham and A. R. Atkins (1960), that represent the human body by a series of concentric cylinders; (2) after R.J. Crosbie, J. D. Hardy, and E. Fessenden (1961), that have adopted a very similar approach using an infinite slab rather than a cylinder; (3) meanwhile, in 1964 Wyndham and Atkins were trying to adapt their model to include regional variations by using a physical system similar to the one discussed here. The biggest difficulty, at the time, it was to solve heat conduction equation with endogenous heat production for transient state with limited computational resources (Wissler, 1964).

Wissler's firsts publications date from 1961, focused on find the mathematical solution for both steady (Wissler, 1961) and transient states (Wissler, 1963) for an human model represented by six cylinders. In 1964 he presented the final model purpose implementing the solution of bioheat equation by using finite difference technique.

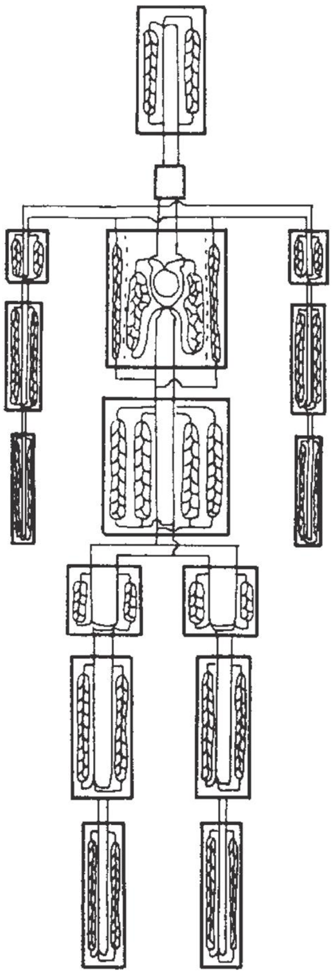


Figure 2.1: Passive or controlled system of Wissler models. [Adapted from Wissler (1964)].

This version, that was the base for the final model (presented in 1985) , describing a human geometrical model with cylinders, for arms, legs, trunk and head. Each longitudinal segment divided into 4 layers of tissue (bone, muscle, fat and skin). Arteries and veins cross the 15 segments carrying the blood from one element to the other and a network of capillaries supplying blood to the tissue layers of each element. Heat is generated, stored and carried away from an element through blood circulation, or conducted to the outside of skin surface (see figure 2.1). The equation of heat diffusion in biological tissues was published by Pennes (1948), and is a simple statement of the first law of thermodynamics (equation 2.1 [1]).

$$\rho C_i \frac{\partial T_i}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(k_i r \frac{T_i}{dr}\right) + h_{mi} + Q_{ci}\left(T_{ai} - T_i\right) + H_{ai}\left(T_{ai} - T_i\right) + H_{vi}\left(T_{vi} - T_i\right) \tag{2.1}$$

Where:

- $T_i(t,r)$ is the instantaneous temperature of the tissue a the radial distance r from the axis of the i^{th} element,
- $\rho_i(r)$ is the tissue density,
- $C_i(r)$ is specific heat of the tissue,
- $k_i(r)$ is thermal conductivity of the tissue;
- $h_{mi}(t,r)$ is metabolic heat generated by unit of volume;
- $Q_{ci}(t,r)$ is the quantity of heat, result from the product of mass flow rate and specific heat of blood per unit of volume;
- $H_{ai}(t,r)$ is the heat transfer coefficient between the arteries and tissue per unit of volume;
- $H_{vi}(t,r)$ is the heat transfer coefficient between the veins and tissues per unit of volume;
- $T_{ai}(t)$ is arterial temperature of blood in the i^{th} element;
- $T_{vi}(t)$ is venous temperature of blood in the i^{th} element.

The equation 2.1 is used to determine tissue temperature changing in radial direction. It depends on time and space domains. Also considers endogenous heat production of the tissues, where can be included the estimation of metabolic heat generated

[1]The units were not available in the original publication.

by work (h_{mi}); the amount of heat exchange with arterial and venous reservoirs/pools (large vessels heat exchange); and the heat changed between tissues and capillaries (small vessels).

Variables that depend on space (r) can be seen as constants if the correct values are associated to the type of tissue. However, variables that depend on time (t) need to be modelled through another equations, or assumptions and simplifications must be defined.

As blood circulation is the most important mechanism to transport the heat and it is stated that blood temperatures change over the time, the inclusion of two extra mathematical expressions to predict the changes in arterial and venous reservoirs are necessary in each segment. Temperature changes in arterial pool [2] of a segment is described by equation 2.2 [3]:

$$(MC)_{ai}\frac{dT_{ai}}{dt} = Q_{ai}(T_{am} - T_{ai}) + 2\pi L_i \int_0^{a_i} H_{ai}(T_i - T_{ai})rdr + H_{avi}(T_{vi} - T_{ai}) \tag{2.2}$$

in which:

- M_{ai} the mass of blood contained in the arterial pool of the i^{th} segment;
- C_{ai} is the specific heat of the arterial blood;
- $Q_{ai}(t)$ the product of the mas flow rate and specific heat for blood entering the arterial pool;
- $T_{am}(t)$ is the temperature of the blood entering the arterial pool;
- L_i the length of the i^{th} element;
- H_{avi} heat transfer coefficient for direct transfer between large arteries and veins.

The analogous generic equation 2.3 [4], predicts temperature changes of venous

[2]Is the name given to an arterial reservoir that represents a large vessel containing the arterial blood in a segment.
[3]The units of the variables were not available in the original publication.
[4]The units of the variables were not available in the original publication.

blood in the venous pool for each segment.

$$(MC)_{vi}\frac{dT_{vi}}{dt} = Q_{vi}(T_{vn} - T_{vi}) + 2\pi L_i \int_0^{a_i} (Q_{ci} + H_{vi})(T_i - T_{vi}) r dr + H_{avi}(T_{ai} - T_{vi}) \tag{2.3}$$

Being:

- M_{vi} the mass of blood contained in the venous pool of the i^{th} segment;
- C_{vi} the specific heat of the venous blood;
- Q_{vi} the product of mass flow rate and specific heat for venous blood entering on the i^{th} segment venous pool coming from the n^{th} segment;
- T_{vn} the temperature of the blood that come from the n^{th} segment.

These three equations are the principle of passive system modelling. Of course that in the case of thorax element, or pelvis, the equations must be slightly different. They need to incorporate more than one branch of veins to determine final temperature in venous pool, and to consider the heat exchange along the respiratory tract.

This equations describe the controlled or passive system of the human body, as it stills being used nowadays. The gap of such detailed descriptive equations, is the need of a large set of information that cannot be directly measured or evaluated. That is the case of the heat transfer coefficients (H_{ai}, H_{vi}, H_{avi}), thermal properties of the biological tissues and mass flow rates.

As a physician, mathematician, biophysic and physiologist, Wissler described a mathematical model with several doors opened to the future. He was the first researcher developing a multi-segmental model of the human body to predict physiologic responses to a wide range of environments.

This is only a small part of his brilliant work. Heat transfer form outer skin to environment was also presented by him as a sequence of equations describing the heat transfer coefficients of radiant, convective and evaporative heat transfer as depending on: environmental conditions (temperature of the air, relative humidity and radiant temperature), body wetness and outer skin temperature. The same was stated for heat exchange due to respiration, that depend on breath rate, temperature and humidity of the inspired air. In the "final version" of the model published by Wissler (1985), it is perceived, by the equations 2.1, 2.2 and 2.3, the 15 profiles of temperature in each element (225 node model), inclusion of oxygen uptake, production of carbon dioxide and lactate concentration of the blood, that final goal increases the accuracy of results by adjusting metabolic heat production and the change, over the time, of thermal properties. This is the proof that the whole model was built from the beginning thinking about future developments.

This final version was not extensively reproduced or tested by other researchers in the field, not only due to its increased complexity, but also due to the lack of information available in published contents (Haslam, 1989).

2.2.2 Stolwijk and Hardy 25-node Model (1971)

The model proposed by Stolwijk (1971) came from investigation work in partnership with Hardy years before (Stolwijk and Hardy, 1966). Although their experience in the area, have by reference the remarkable and innovate work developed by Wissler (1964).

This mathematical model of physiological temperature regulation in man, fully published in 1971 in report format with annotated listing of Fortran , was sponsored by National Aeronautics and Space Administration (NASA), and performed by John B. Pierce Foundation Laboratory, from Department of Epidemiology and Public Health of Yale University School of Medicine.

Comparatively to Wissler's models (225 nodes), this model is quite simpler and much easier to implement. The passive system is constituted by 5 cylinder, with size closed to the average dimensions of a real man, representing the trunk, arms, hands, legs and feet and a sphere for the head. Each body segment was divided in 4 concentric layers: core, muscle, fat and skin. The six elements divided in 4 compartments make 24 nodes. The 25^{th} is the central blood pool that represents large arteries and veins. Figure 2.2 shows in a) the scheme of the controlled system, and in b) the scheme of heat diffusion through the tissues and respective connection to the central blood pool. This representation of the circulatory system works as the principal mean of heat dissipation. Without him temperatures rise to irrational values.

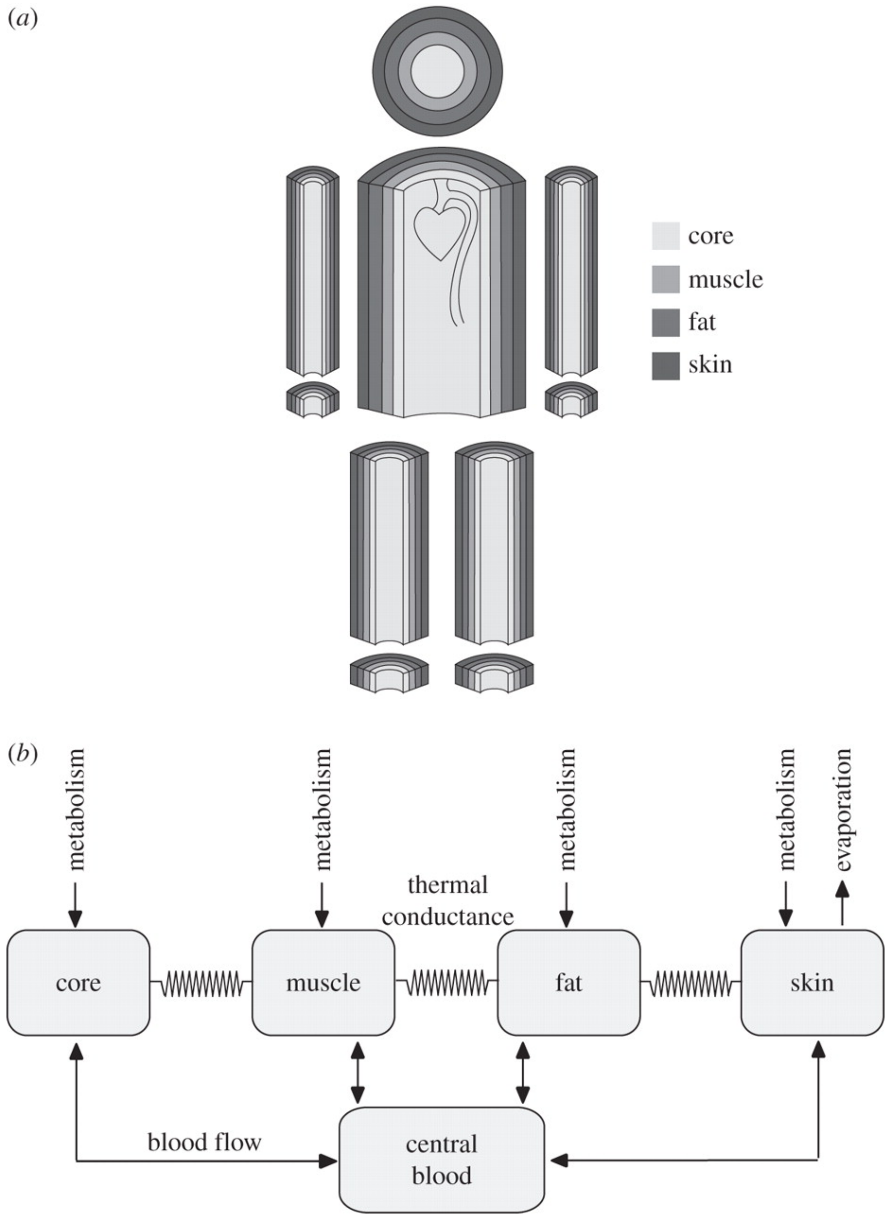


Figure 2.2: a) Representation of the passive system of Stolwijk model, b) Heat echange scheme between compartments. [From: van Beek et al. (2011)]

2.2.3 Nishi and Gagge 2-node Model (1972)

It is important to refer that Gagge worked with Hardy and Stolwijk at John B. Pierce Foundation Laboratory, so the version of Nishi and Gagge 2-node model, also known as Pierce 2-node model, in fact, it was a simplification of the Stolwijk model for practical applications. It follows the same concept but in a simplified core-shell approach with a layer of cloth (see figure 2.3)

With only two compartments it is much simpler to implement heavy complex

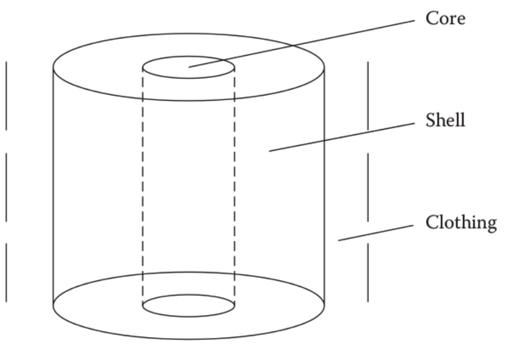


Figure 2.3: Representation of Gagge and Nishi 2-node model. [From: Haslam (1989); Parsons (2014)]

parametrization features. In spite of following the same *school*, they stretch the application of thermal models to other fields. Being members of the America Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE), lead the models and SET scales to improve energy performance of building and lately in occupational health and safety, based on the analysis of human thermal comfort.

Their final goal it was to build an environmental temperature scale based on their current knowledge or, by other words, a practical operational tool, based on the computed results of a simple thermophysiological model (Gagge et al., 1972).

Nishi and Gagge (1977) present the SET (Standard Effective Temperature scale) for application in hyperbaric and hypobaric environments. The application of SET concept require measurement of prevailing environmental conditions, the correct evaluations of metabolic heat loss from skin surface by energy metabolism, estimation clothing insulation worn, measure of exposure time, and using this inputs to predict the mean body temperature.

2.2.4 Haslam's and Parsons' Models

Haslam and Parsons (1987) developed a set of 4 models based on ISO 7933:1989, Givoni & Goldman, Stolwijk & Hardy, and Nishi & Gagge , respectively named LUTISO, LUTtre, LUT25node and LUT2node. They develop this study particularly to answer to British army, that required reliable tools to evaluate the thermal stress in a wide range of environmental conditions. The criteria for chose the models seem to be the different types of models, rational and empirical, more complex or simpler versions.

Modifications to the original models consist in making them useful to provide predictions for work/rest cycles and people wearing cloths, considering both dry and vapour heat transfer. What means that there were no significant improvements. Remarkable it was the independent process of validation and comparison between the models. This was the first time that the same validation procedure and data was used to compare the performance of different types of models to a large set of environmental conditions (see table 2.1).

Table 2.1: Accuracy of Haslam's and Parsons' models over thermal conditions. [adapted from:(Parsons, 2014)].

Thermal Conditions	$t_a \leq 5°C$		$5°C < t_a \leq 15°C$		$15°C < t_a \leq 25°C$		$25°C < t_a \leq 35°C$		$35°C < t_a$	
	T_{cr}	T_{sk}	T_{cr}	T_{sk}	T_{cr}	T_{sk}	T_{cr}	T_{sk}	T_{cr}	T_{sk}
Nude, rest, no wind	none	lut2	lut2	lut25 lut2	luttre lut2 lut25	lut2 lut25	lut25 lut2 luttre	lut2 lut25	lut25 lut2 luttre	lut25 lut2
Nude, rest, wind	-	-	lut2	lut25	luttre lut2	lut25	-	-	-	-
Nude, work, no wind	-	-	-	-	lut25 luttre	lut2 lut25	lut25 luttre lut2	lut25? lut2?	lut25 lut2	lut2 lut25
Nude, work, wind	-	-	-	-	-	-	lut2 lut25 luttre	-	lut25	lut2 lut25
Clothed, rest, no wind	none	lut2 lut25	lut2	lut2 lut25	-	-	-	-	-	-
Clothed, rest, wind	none	none	none	lut2	-	-	-	-	-	-
Clothed, work, no wind	lut25	lut25	lut2 lut25	lut25	-	-	lut2	lut2 lut25	none	lut2 lut25
Clothed, work, wind	none	none	-	-	-	-	lut25	-	-	-

Note: Are presented the most accurate models in descending order that accomplish the maximum rmsds (the average root mean square standard deviation) of 0.5 $°C$ for core temperature and 1.6 $°C$ for mean skin temperature.
'none' – prediction values out of the range for all the models.
t_a - air temperature in Celsius degrees.
T_{cr} - core temperature in Celsius degrees.
T_{sk} - mean skin temperature in Celsius degrees.
Dash means that there is no data available for comparison.

It is important to underline the fact that empirical model loses in terms of accuracy and range of environmental application comparing with rational models. In particular, for clothed situations his level of accuracy did not fit the maximum *rmsds (root mean square standard deviation)* criteria. As expected and described by other authors the numbers of nodes is important in cold but when thinking about hot temperatures when temperatures over the tissues become more homogeneous, level of detail depends on the application goal, because not always offer higher predictive power. The model with best predictive power for core and skin temperature above $25°C$, at work, clothed, with no-wind it was the LUT2-node model (i.e. $0.3°C$ for t_{cr} and $0.8°C$ for t_{sk})(Parsons, 2014).

2.2.5 Kraning & Gonzalez Model (1997)

This modelling proposal was developed by the research team from Military Ergonomic Division of the US Army Institute of environmental medicine Kraning II and Gonzalez (1997). Gonzalez was co-author in the Pandolf predictive model and both worked Givoni and Goldman in parallel projects to collect data and general and individual physiological responses to heat exposures.

SCENARIO is a soldier-based physiological/thermoregulatory model that reliably incorporates novel schemes for calculating changes in blood flow to muscle, visceral areas and skin, and changes in stroke volume, heart rate and cardiac output along with previous depictions for control of core temperature and sweating rate (Kraning II and Gonzalez, 1997). Passive system is modelled as a single cylinder divided into 5 layers (core, muscle, fat, vascular skin and outer skin), plus a central blood compartment. Predicted results are average values for an individual with $W = 70\ kg$ of weight and surface area $A_D = 1.8m^2$.

The rates of metabolic heat in core, fat and avascular skin, are assumed to be fixed percentages of total resting metabolic rate. Endogenous heat production in muscle layer (H_{mu}) depends on the total energy expenditure (M_{tot}) and external work performed (W).

The heat flows through radial direction by conduction along adjacent compartments (figure 2.4). Between central blood compartment and the first 4 layers (core, muscle, fat and vascular skin) heat is exchanged by controlled forced convection. From avascular skin heat is lost to the air by sweat evaporation, radiation and convection.

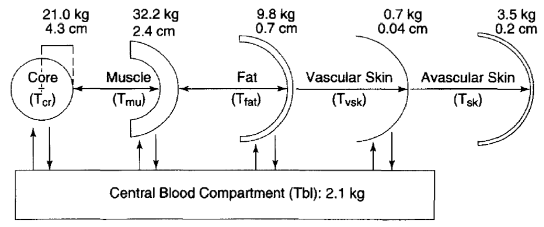


Figure 2.4: Cross-section of cylindrical model containing five concentric annular tissue compartments.

The heat lost due to respirations is the result of evaporation and convection. As input variables to simulate individual characteristics are given: body weight (W), stature (H) and percentage of body fat. These inputs allow calculating metabolic heat production, size of compartments and thermal conductances between adjacent layers ($K_{(n-1).n}$ or $K_{n.(n+1)}$). The heat flow that vary between compartments is described by equation 2.4.

$$\begin{aligned}\frac{d}{dt}Q_n(t) = H_n(t) &+ \left(K_{(n-1).n} \cdot [T_{n-1}(t) - T_n(t)]\right) \\ &- \left(K_{n.(n+1)} \cdot [T_n(t) - T_{n+1}(t)]\right) \\ &- \left(BF_n(t) \cdot \rho_{bl} \cdot c_{bl} \left[T_n(t) - T_{bl}(t)\right]\right)\end{aligned} \tag{2.4}$$

Variable Q_n is the heat content, T_n is the temperature, H_n is the rate of heat production and BF_n is the rate of blood flow through compartment n. Temperature of blood is defined by T_{bl}. The variables T_{n-1} and T_{n+1} are the temperatures in the adjacent compartments to n. Core layer does not have compartment $n-1$ and the outer skin layer does not have heat exchange with the central blood compartment and the outer compartment $n+1$ is replaced by $E+R+C$.

$$\begin{aligned}\frac{d}{dt}Q_{bl} = \rho_{bl}c_{bl} \cdot (BF_{cr} \cdot (T_{cr} - T_{bl}) + BF_{mu} \cdot (T_{mu} - T_{bl}) + BF_{fat} \cdot (T_{fat} - T_{bl}) \\ + BF_{ask} \cdot (T_{ask} - T_{bl})) - (C_{res} + E_{res})\end{aligned} \tag{2.5}$$

Equation 2.5 represents the heat exchanged in central blood compartment. It integrates the heat lost by breathing and the heat exchanged with the four considered layers (being the subscripts representing *bl* the blood, *cr* the core, *mu* the muscle, fat the fat, *ask* the non-vascularized skin layer).

Controlling/active system: integrates a vascular model with distinct blood distribution along the tissue layers; modulates the sweat rate secretion; adjusts level of conductance between vascular and avascular skin layers; and occasionally, corrects heat balance by shivering.

Input variables for control system include blood temperature, average skin temperature, oxygen uptake, heat production and body weight decrement in percentage.

All algorithms are updated in intervals of Δt. As shorter the interval better would be the response for transient states.

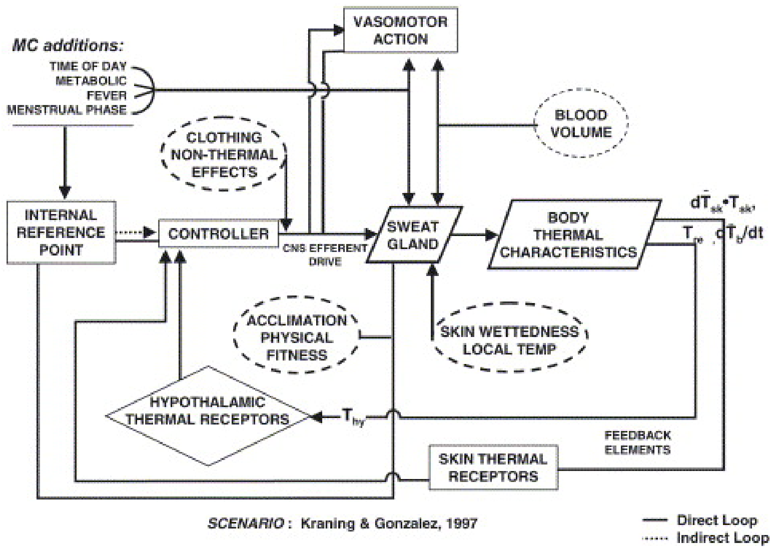


Figure 2.5: Scenario - MC block diagram Gonzalez (2004).

After some years of research, Gonzalez (2004) due to limitation of mathematical modelling of physiological human thermal responses, proposed an improvement by using a probabilistic application of Bayesian statistical paradigms joined with Monte Carlo's methods. The current control diagram showing the passive and active components of the model with the key feedback loops, effector and non-thermal properties along the *"MC additions"* to the input design (figure 2.5).

The innovations are to use statistical predictive adjustments to set point values in the control system, and act directly on the vasomotion and sweating capacity of the individuals. This way by integrating the probability of cause effect due to circadian cycle, malignant hyperthermia and hormonal cycle, the sensitiveness of the model to individual characteristics improved significantly.

2.2.6 Fiala Models (1999-2012)

In the past three decades, several multi-segmental mathematical models of human thermoregulation have been developed (Wissler, 1985; Stolwijk, 1971) and have become valuable tools in the contribute of a deeper understanding of regulatory processes.

At the time Fiala et al. (1999) considered that the reasons why they did not became widespread, include lack of confidence in their predictive abilities, limited range of applicability, and poor modelling of the heat exchange with the environment. In his words, Fiala et al. (1999) intend to contribute to research efforts to formulate a more precise, flexible and universal model of the human thermoregulatory system.

In order to overcome limitations of previous models Fiala (1998); Fiala et al. (1999) present a new multi-segmental model (DTS - dynamic thermal sensation). Model includes metabolic heat produced within the body, respective heat distributed over body regions by blood circulation and heat conducted to the body surface. Insulated by clothing, the heat is lost from body surface to the surroundings by convection, radiation, and evaporation. Heat exchange by respiratory tract is also included. To achieve more realistic results all the phenomena of heat diffusion counts with geometric and anatomic characteristics of the human body and considers the thermophysical and the basal physiological properties of tissue materials.

The body was idealized as 15 spherical or cylindrical body elements: head, face, neck, shoulders, arms, hands, thorax, abdomen, legs, and feet. The multilayer model consists of annular concentric tissue layers and uses seven different tissue materials: brain, lung, bone, muscle, viscera, fat and skin. The criteria for division consist in separate layers whenever a significant change in tissue properties occurs. As shown in figure 2.6 the model is composed by 19 compartments and 342 nodes, and it is able to consider asymmetry and heat transfer direction for each compartment.

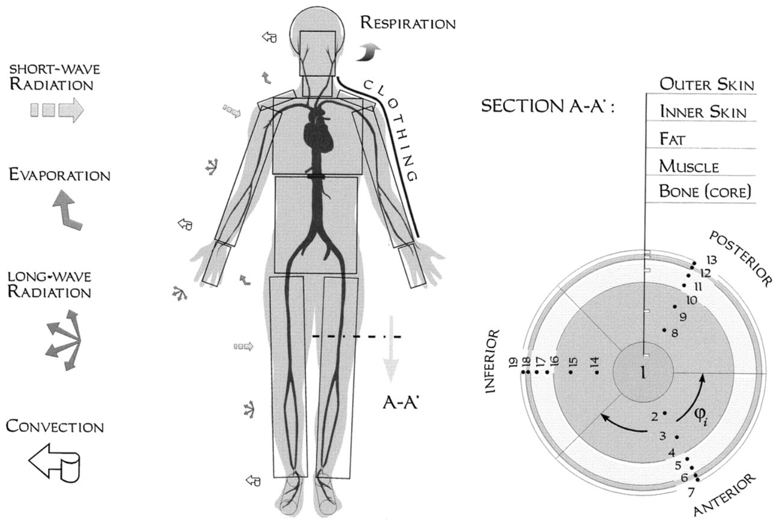


Figure 2.6: Sheme of the passive system of Fiala's DTS model. [From: Fiala et al. (1999)].

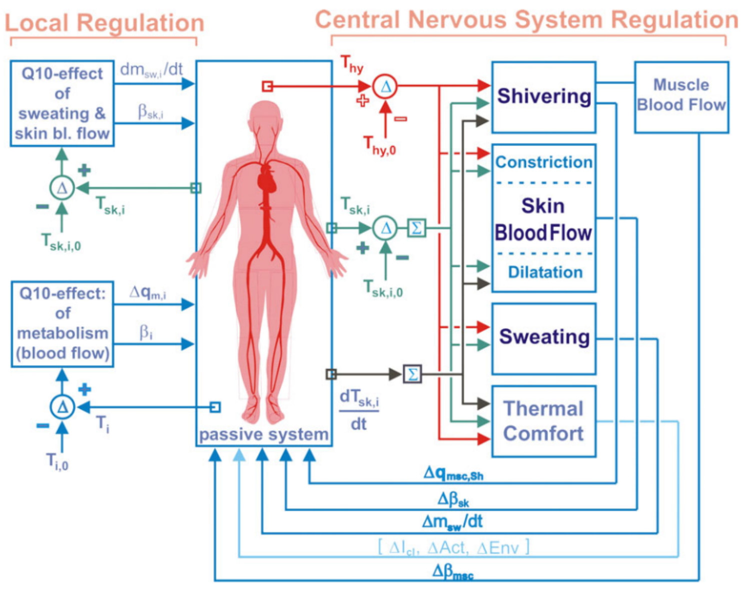


Figure 2.7: Fiala's DTS and UTCI block diagram of active system. [From: Fiala et al. (2012)].

The first version of the model purposed by Fiala et al. (1999) (DTS version), was initially developed to be applied in thermal comfort assessment in buildings, however, the huge potential of the model was quickly discovered. Several generic and specific applications were object of validations studies regarding: human and thermal regulatory behaviours; transient indoor climate conditions in cars; asymmetric radiation scenarios and exposures to high intensity sources; anaesthesia and clinical trials (Fiala et al., 2012).

The active system modelling proposal consist in reacting loops. Central nervous system physiological reactions (shivering, skin blood flow) are continuously adjusted according the values of hypothalamic temperature and mean skin temperature. Local adaptation, due to Q10-effect, is modelled using the temperatures of the skin and tissues of the element of the body adjusting the sweating, heat production and blood flow. The block diagram that describes Fiala's modelling of the active system can be described in figure 2.7.

The need to simplify the DTS models lead Fiala et al. (2012) to develop a new version called UTCI model (universal thermal climate index) . Simplification consists in assuming body symmetry, reducing the 19 compartments to 12, what means 187-node model. These final UTCI model, was the base (target) for other developments, including a purpose to incorporate a clothing system and, most important, an independent study of validations in a wide range of environmental contexts and working conditions (Parsons, 2014). It is one of the models that came a free calculator available on the internet.

2.2.7 Other important models

After pioneer models, potential for innovations is based on the integration of other perspectives or other pioneers models in order to improve the phenomenological approach or to find through experiments more accurate values for thermal properties of environment or biological tissues, to better describe mathematically the interface between man and environment.

Fu Model (1995)

Fu (1995), in his Ph. D. thesis presents a three-dimensional model that responds to transient environmental conditions. Part of his research it was dedicated to experimental work in order to find threshold values for active system, in particular relating the arterial radii with the Cardiac Output.

Tanabe (2002)

Application of whole-body thermal models has gaining widespread integration with new systems and new ways to apply it became research advances. Tanabe et al. (2002) present a 65-node model, similar to Stolwijk and many others, with passive and active systems. But this time, he compare the heat dynamics between man and environment by integrating the human model in a computer fluid mechanics (the well-known CFD) system.

This application got very important developments in particular for indoor applications.

Salloum (2007)

Salloum et al. (2007) described a new transient bioheat model with a lot of potential. Once again passive system model followed closely the Stolwijk (1971) version. Body was divided into 15 body segments composed by four layers. However instead a singe blood pool they divide circulatory system in arterial blood and venous blood. The challenge it was the adoption of arterial model of Avolio (1980) to simulate blood flow. Blood circulation and flow rates were based real physiological data, real dimensions and positions of arterial branches in the human body. The best advantage of Avolio's model is that, with the impedances calculated along the artery branches,with a pulsating signal of CO at the entry of aortic branch, is able to estimate blood velocity and pressure at any point of the arterial tree (Avolio, 1980).

With a realistic blood flow and cardiac output distribution, with a good representation of the change in stroke volume and central blood volume, the potential for predicting heat illnesses, in particular heat syncope, heat exhaustion and heat stroke, is significantly higher.

Active system approach is also based on the work presented by Fu (1995).

Ferreira and Yanagihara (2009)

Based on the studies of Penne and Wissler, Ferreira and Yanagihara (2009) presented the mathematical solution for three-dimensional thermal diffusive model, in particular for cylindrical and elliptical geometries. It also gives important contributions in the calculation of thermal coefficients between arteries and veins H_{av}, and the settled of other parameters of passive system.

Neto (2010)

Behind the pioneer models of Wissler (1964, 1985) and Stolwijk (1971), Albuquerque-Neto and Yanagihara (2009) present a mathematical model of the human thermal and respiratory systems to predict temperature distributions and oxygen and carbon dioxide concentrations along the body from environmental parameters and metabolic physical effort. Despite being a bi-dimensional heat diffusion model, the upgrade done to circulatory system, based on the principle established byWissler (1964), that depends on the concentration of the respiratory gases in the body, temperature and metabolic rate, make from this model one of the more complex models built since Wissler (1964). This mode have shown the ability to predict the complex interaction between thermal and respiratory model, as the decrease of body temperature due to the increase of breath rate (that depends on O_2 and CO_2 concentrations) and the reduction in partial pressure of the gases in the end parts of body segments due to the effect of temperature in the ability of blood to carrying them.

2.3 Closing remarks

Traditional whole-body thermal models, born in America, linked to US army medical (Wissler, 1961), aeronautical (Stolwijk, 1971) and environmental ergonomic departments (Givoni and Goldman, 1972; Pandolf et al., 1977).

In spite of this two branches of American research teams (empirical and rational research), they were aware of each other and integrate the knowledge in order to get the best solution for their practical application. The researcher's from NASA invest in high complex models to develop safer condition with aeronautic application, and for military enforcement develop practical portable tools that support decision making at the field.

It is important to state that many models did not suffer an independent validation with clear criteria design for the effect. Most of them are validated by the authors, usually not physiologists or physicians. This means that in spite of suffering comparison with real data, a non-systematized process of validation and different criteria goals and data types make difficult to elect the best option to fit the best practical application.

Haslam and Parsons (1987) work was a step in comparing the potential for practical use of very distinct models, from simple empirical indexes to more complex whole-body phenomenological models. For the first time it could be compared the accuracy level of such distinct models. Once again this was a study made for a project for UK army. It also became clear that complex models are the ones who have the ability to adapt to the different kind of environment. To model thermal response to hot environments the rational complex models had the best behaviour.

To achieve a good level of accuracy in heat stress prediction a powerful control system is necessary.

The most important enforcement field of research was focused on military field, but generic civil enforcements gain position with European researchers (Fiala et al., 1999; Malchaire et al., 2001), from environmental ergonomics, to building energy efficiency, passing by the development of new tools or products, all have been motive for the use whole-body thermophysiological models.

Curiously, considering the more than 50 year of research, there are no significant tools available to evaluate exposures to hot environments, being the UTCI calculator [5] from Dunsan Fiala , and a program for risk assessment using PHS model [6], two of the tools recognized by the scientific community that are available for free on the internet. The availability of the source codes of the programs are even more rare. With the exception of the programs published by Gagge and Nishi in scientific article (Gagge et al., 1972), and previously by annotated listings by Stolwijk in Nasa's report (Stolwijk, 1971), both in Fortran code, there are published for full, most of the models do not allow reliable reproduction for further studies.

[5]Calculator for the Universal Thermal Climate Index (UTCI) - `www.utci.org`

[6]Program for risk assessment using PHS model - `http://www.deparisnet.be/chaleur/Chaleur.htm`

Chapter 3

Problem State and Study Goals

3.1 Introduction

The literature review presented in the state of art have shown that there are a few rational models that can be considered the origin of whole-body thermal modelling. Wissler, Stolwijk and Hardy, Nishi and Gagge can be considered the fathers of new rational modelling approaches. However, from the three approaches, Wissler's mathematical description of the models presents a significant lack of information that do not allow a faithful reproduction, as the other models do, with a simple and available Fortran code. This is probably the reason that most of the recent developments of rational models are based on Stolwijk 25-node model and Gagge 2-node model.

None of these models was able, in their original formulation, to simulate heat diffusion in three dimensions or respond to transient and non-homogeneous environments, which are necessary requirements to accomplish reliable simulations of real life conditions. The recent researches based on these works were able to respond to one or two of those requirements, but not to all at the same time. One reason for this gap can be justified by the increased complexity of the model that would need a higher level of detail and result in non reasonable simulation time.

Besides, with the exception of Gonzalez work, whole-body models try to simulate the average human response instead of following the trend of individual aspects for accurate and representative results.

All of these aspects allied to the fact of models had been implemented in older programming schemes and did not use the new potentialities of recent high performance, top level languages, leads to a new research opportunity that can be seen as:

The design of new mathematical model based on the best practices of previous models, and solve the differences between them in order to find the most suitable prediction of thermophysiological responses to hot environments, and implement it through an high performance language that allow to simulate complex real life conditions of exposure.

3.2 Study Goals

Develop a whole-body bioheat model, based on phenomenological concepts, whose system is able to:

- simulate heat diffusion in three dimensions;
- respond to steady and transitory environments;
- allow infinite combinations for settling boundary and initial conditions;
- detailed mapping of temperatures profile.

From actual scientific knowledge must be found informations to:

- built the physical model -
  - number of segments;
  - number of tissue layers ;
  - size and shape of the segments.
- define mathematical model of bioheat diffusion -
  - recognize significant bioheat exchanges inside the organism;
  - establish mathematical equations that describe those exchanges;
  - found values to describe thermal properties and thermal behaviour of biological tissue;
  - propose a suitable method to solve mathematical model numerically.

To allow a sustainable development of the program, software tools should:

- be freeware, multiplatform, made by scientific community, validated and approved by them;
- programming language must be easy readable and easy to learn;
- have appropriate tools for large dataset handling namely - definition, parametrization and visualization;
- have the ability to plot in real time the simulated results.

3.3 Technical and Scientific Contributions

Every year people die due to episodes of critical heat exposure. Even at work the assessment through the traditional indexes (ET, ETC, WBGT or PHS) do not assure a safe environment due to a limitation of their applicability. As heat stress is typically linked to uncontrolled environmental conditions (unstable temperatures, air turbulence and asymmetric radiance), and individual aspects, is important to build a model that can reply to it. The mathematical modelling proposal presents features that the previous models were not able to join in a single model (three dimensional heat diffusion to transient and non-homogeneous boundaries). So, a program that is able to predict the distribution of temperature along several body parts, and respective arterial and venous temperatures, as function of time and skin temperature in any type of environment considered out of control or unhealthy.

The release of the source code of the program give an important technical and scientific contribution, because it makes clear the interpretation of the mathematical equations and the relation between variables that allow go forward from the traditional models to the one that is able to include all previously mentioned features for future developments by the community.

Chapter 4

Model Conception and Development

4.1 Introduction

This chapter is dedicated to present the materials used, its features and the sequence of steps required in model conception and development processes. It is divided in two sections: one where are detailed the computational materials used, its advantages/disadvantages and the reasons that support the options and decisions; and other where model conception and development is detailed step by step according the chronology of the model construction process.
This second section includes information about:

- mathematical equations;
- numerical methods used to solve the equations;
- computational implementation materials;
- behavioural test of the program modules.

The detailed description of the model and the respective conception and test intends to contribute to the construction of a *Free Software* that can serve the social community of potential users that will be able to use a tool to assess the thermal stress for free, and more than that, that can be adapted to specific situations by adding and/or changing the source code.

4.2 Materials

The choice of the software and programming language was based on three main issues:

1. create a multiplatform tool for universal usage;
2. a tool that allow a sustainable development of the present model;
3. open source code to allow how numeric solving is implemented.

Multi-platform opensource software, easily integrable with other scientific software packages and library routines (namely 3D Scientific Data Visualization and Plotting, graphical user interface packages (GUI) and numerical computation) where chosen. Questions such as quality and quantity of the technical and scientific information available in internet official sites, as well as the opinion of the scientific user's community (checked in official blogs, forum discussions and so on) was also considered.

4.2.1 Python Language

The Python language was created by the German programmer Guido van Rossum in the late 80s, during their collaboration in the Centrum Voor Wiskunde in Informatics (CWI) [1]. Rossum still representing an important role, being known among the members of Python community as BDFL (Benelovant Dictator For Life). He continues to take the main decisions on the development direction of this language.
All the versions of Python are open source which means, among other features that the program, or produced program modules, can easily be programmed, tested and distributed free of charge. Although interpreted, Python is an high performance language, simple, fast and powerful, with different application fields:

- Web and Internet development;
- scientific and numerical computing;
- educational tool in introductory and advanced programming courses;

[1]History and License of Python - official documents of the version 2.7 - `http://docs.python.org/release/2.7/license.html` - acessed February 2nd of 2014.

- graphical user interfaces programming;
- software development.

The potential of Python can be easily expanded using small modules in C and C + +.The huge amount of existing libraries and packages, enables an endless programming and calculation alternatives. Python language has a structure and a Code simple to write, directed to a modular programming by objects. All these features and capabilities justify the choice by this programming language. It allows the construction of modular mathematical models, able of undergoing evolution in different fields, either computation or interface. It is recalled that the ultimate aim is, rather than presenting a model, it is to present, a possible tool able to evolve and adapt.
Python language also has the advantage of being cross-platform, meaning it is compatible with different operating systems such as Windows and Linux. These were the main reasons why Python language proved to be the most suitable for developing this kind of software and make it a tool for universal use.

Python Libraries

The proposed computation model is integrated in scientific and numerical computing. To this end, there is a community of users / developers of libraries, with routines that can be integrated with python to support this kind of programming. Among the different available packages were found some that have proven to be the most relevant for this calculation approach, they are:

- *NumPy* - is the fundamental package for scientific and numeric computation. It contains useful linear algebra, Fourier transform, random number capabilities, N-dimensional array objects and tools for integrating C/C++ and Fortran code.
- *SciPy* Library - a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics and more.
- *Matplotlib* - a package of data visualization that provides publication-quality 2D and simpler 3D plotting.
- *IPython* - an interactive interface that allows process data and test ideas quickly.

- *Fipy* - a tool for numerical problems resolution, based on Finite Volume Methods, for solving partial differential equations.

- *Mayavi* - provide easy and interactive visualization of 3D data. Additional installing features allow simple and clean scripting interface in Python, including one-liners, or an object-oriented programming interface. Mayavi integrates easily with NumPy and SciPy for 3D plotting and can even be interactively used in IPython, similarly to Matplotlib. Additionally Mayavi is a reusable tool that can be embedded in applications in different ways or combined with the Envisage application-building framework to assemble domain-specific tools.

Although the availability of many "open source" tools with an easy integration with Python, the option to program a graphical interface in *Qt*, through *PyQt*, proved to be an option simple to implement. However, more complex and/or complete interfaces can be performed in *Eclipse*, by using *PyDev*, *wxGlade* through *wxPython*, *Tkinter* or with its integrative tools.

4.2.2 Mesh Creator

To solve partial differential equations that are part of the model, it was needed to define the physical volume that should be used to make the calculations. Fipy is a numeric solver package that handles with the data integrating them in a single cell variable. To create it, Fipy needs the mesh that is interpreted as the physical volume were heat diffusion take place. Cell variable relates the instant temperature at each cell to the position in the mesh. This is the kind of variable that is used to supply the information that is needed to numerically solve the mathematical expressions. Mesh creator was defined based on the file extension accepted as an input by the Fipy cell variable. Gmsh was pointed by Fipy creator as an open source mesh creator with potential and easy integration with Fipy routines, what mean that fills the minimal purposes to be included as tool for mesh creation. It allows a very simple and easy programming of simple geometrical grids and structures. More sophisticated non geometric shapes can also be created and implemented with using Gmsh interface.
There is the awareness that the quality of the mesh is very important to the accuracy

of the results and Gmsh is a good solution for creating regular and irregular meshes, and does not limit the upgrade potential of the model.

4.2.3 *U*buntu Operative System

Python is, as mentioned above, a multiplatform programming language, however it is easier to install and manage the package dependencies of Python routines in Linux based Operative System, than Windows Operative System. That is why Ubuntu was chosen to be the platform to implement the model and develop the program software.

4.2.4 Python IDE

There are many free Python editors that are Windows and Linux compatible. As Linux operative system is the best choice to solve package dependencies, the option based on the most user friendly Python Editor. A simple editor, like VIM or Gedit, have colouring options and special functions that help correct syntax errors. However, Integrated Development Environments (IDE), can join these features to object inspection, variable explorer, file explorer, Python run shells and debug features.
Spyder is a IDE for Python commonly used, and the Graphical User Interface (GUI) interface is quite similar to Matlab. As, in engineering, Matlab is a popular tool used very often, and being interfaces similar, Spyder becomes easy to learn.

4.3 Methodology

Simulation is a very important tool in science. These input and output variables and conditions are the guide lines for the model design. When thinking in complex simulation models, only the exercise of define precisely the inputs, such as variables boundary conditions, initial conditions, constants, among others, can be an hard work. The same can happens with the output data, specially because output always depend on the input conditions.

So, methodology definition when modelling and implementing the model is not always an easy process, but there are some techniques that helps to organize the development of complex and extent models.

4.3.1 Design Approaches

Modelling techniques can be essentially divided into three main design approaches, there are: *Top-Down Design*, *Bottom-Up Design* and *Spiral Design*.

Top-Down Design

This method involves a hierarchical or tree-like structure for a system as illustrated by figure4.1. Only in some words, in *top-down design*, a complex problem is expressed

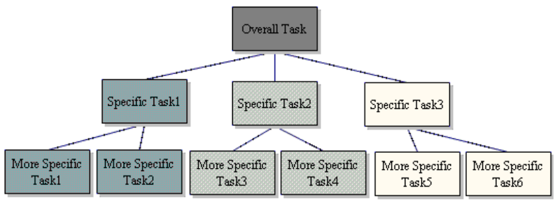


Figure 4.1: Top-down Design diagram.

as a solution in terms of smaller, simpler problems. These smaller problems are then solved by expressing even more smaller and simpler questions. This process continues until the modular programs are completely trivial solutions. Finally all the small parts are put back together and and thus achieves the prototype model. Here is defined the top-level module, structuring all down in a tree structure, creating routines to the smaller tasks .

Bottom-Up Design

The *bottom-up design* can be considered the opposite design process. As the name suggests, the design process begins by defining the most basic routines of the tree chart's bottom (figure 4.1) till the top of it, in the reverse order of the tasks process.

Spiral Design

This design process is known by a systematic iterative process that assures the cyclic development. At each step it is included a new feature from the bottom-level until the top-level. The process stops when there are no more components to include.
In spite of modelling process could result in better or worse models, can not be said that there is such thing as good or bad design techniques. Usually, in complex programs, all the designs are mixed and that is it which can adapt the design process to the complexity of the model. *"Good design is as much creative process as science, and as such, there are no hard and fast rules"* Langtangen (2006).The advice is to practice and test several possibilities until get, even if not the best, one suitable option or result.

4.3.2 Model Design

Obviously, in a complex modelling process such as whole-body human thermal modelling, mixing the techniques is the suitable choice.
The process starts by aligning the phenomenological and mathematical concepts of the model with the program architecture and potential features. According to it:

1. To forecast the diffusion of heat in the three directions of a three-dimensional physical element;
2. Mathematical model must take into account with the importance of a phenomenological interpretation;
3. Should respond not only to steady-state conditions but also to transitory-state situations;
4. Which parametrization should be as detailed as possible to include individual characteristics, all possibilities of boundary and initial conditions.

4.4 Whole-Body Thermoregulation

Firstly some techniques of top-down design to describe the model and its features were applied. A complete whole-body human thermal model can be divided in two large systems - passive and active - being each section is subdivided in smaller functions. The structure adopted to the present model can be consulted in figure 4.2.

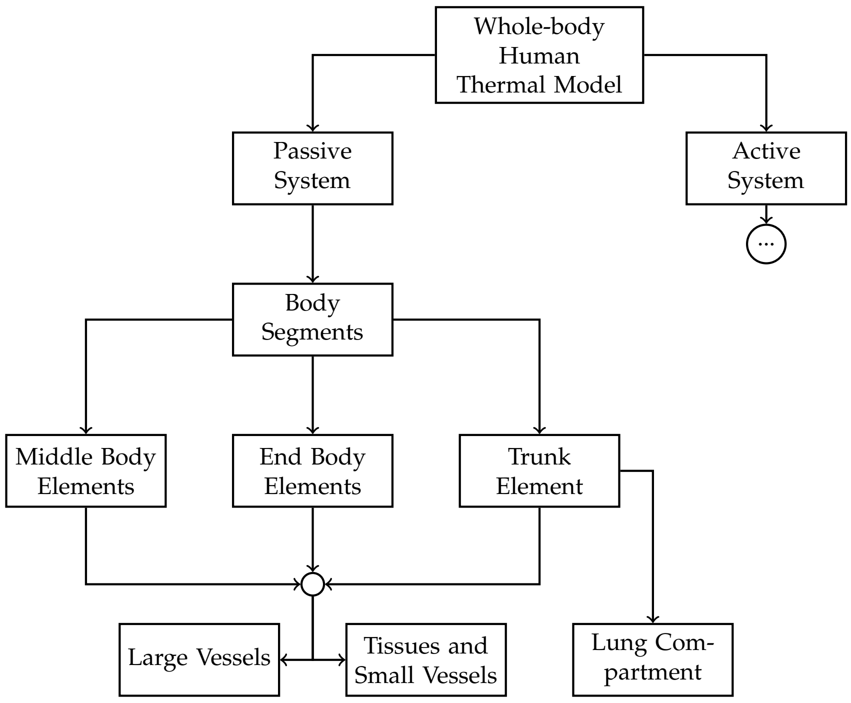


Figure 4.2: Model design.

In the passive/controlled system is simulated the heat diffusion throughout the biological tissues. This part gathers the circulatory system with several tissue layers that compose the “solid” physical structure. It predicts heat diffusion and heat dissipation inside the human body according to initial conditions, tissues thermal properties and boundary conditions.

Heat transfer in biological tissues occurs due to the important role of blood flow. Blood is life. It transports oxygen and all other important nutrients to the cells, all over the organism, as well as it carries the heat from the inside out, or the opposite, to assure temperatures compatible with life. Blood circulation is the main mechanism used to warm up or cool down the body. Blood flows from the inner core layer to the outer external layers through a complex system of arterial and venous vessels that change in size and in number according to the kind of biological tissues or organ. The inner vessels are bigger and are the responsible for the massive blood transport to the numerous body segments. So, there are a few number of huge vessels responsible for this massive distributions in the core layers of the body. But, as the vessels get to extremities and external layers of the organs, they increase dramatically in number and decrease in size. Here, small vessels and capillary beds mix within the tissue layers. The vessels and capillary densities are so high that they almost become an "homogeneous structure". It is in this structures that is found the highest heat transfer rate between blood and tissues.

Big arteries and veins, also exchange heat with surrounding tissues. However, the amount of heat exchanged occur by conduction, specially in radial direction, throughout the tick tissue of the vessels. The most significant heat transfer occurs between the large countercurrent vessels. As blood goes from the heart through arteries to the tissues and return trough veins, heat is swapped along the path between the main vessels (arteries and veins) that cross the body side by side (figure 4.3).

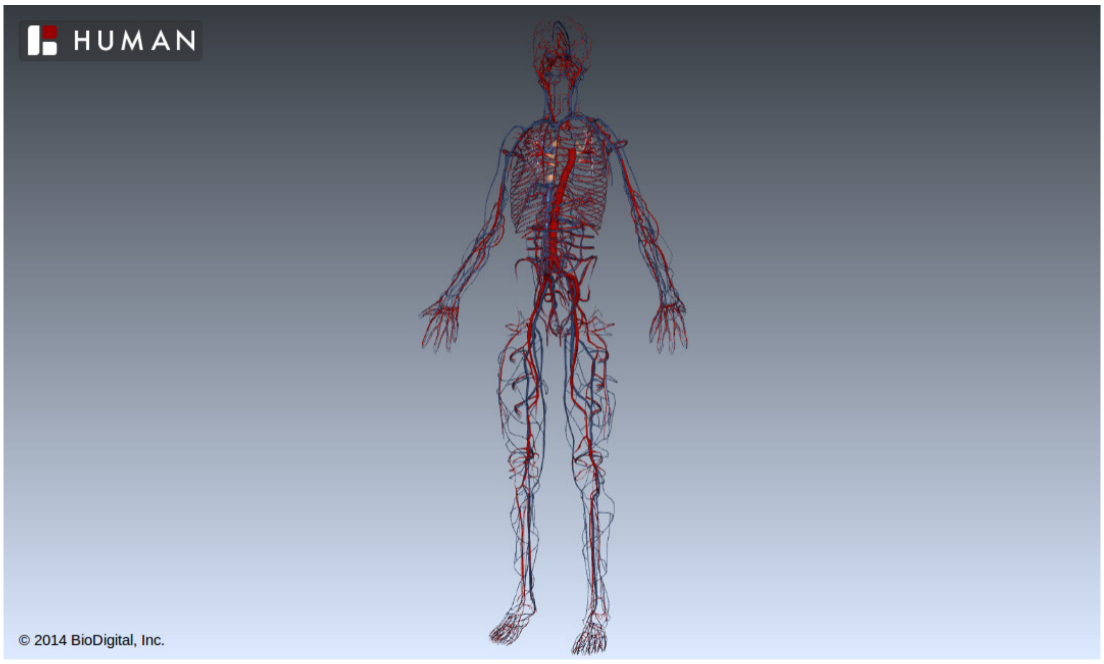


Figure 4.3: Circulatory system. The countercurrent vessels, artery and vein pair.

Remember that the blood function is fundamental to assure homoeostasis. Blood carries nutrients, oxygen, metabolic wastes and transports heat from all biological structures of the body. Of course there are several parameters used to compute the heat transfer in the biological tissues that are not constant and are constrained by the active system response. Moreover the importance of the passive system modelling is to establish the relationship between all those variables and simulate the heat transfer to each values that the variables can assume.

The active system should include the basic thermoregulation mechanisms such as shivering, vasocontriction, vasodilation and sweating and set biological response according to the needs. These decisions are made at the central nervous system, in particular area known as hypothalamus. This part of the brain feels the most slightly change in core temperature or anywhere inside and outside the body. Several nerves act in the whole body, as thermal-receptors and sent special information about temperature changes and levels. In the skin, these thermal-receptors are the responsible for thermal sensation. So, according to the information, if hypothalamus decodes cold sensation reduces radiant heat losses through skin by reducing the blood flow with vasoconstriction and increases metabolic heat production by shivering. If it feels hot, increase heat losses allowing vasodilation and a high level of blood flow at skin what increases radiant and convective heat losses and sweating (that is the most effective mechanism of heat dissipation). But this is a non-linear process. On one hand exists metabolism that is the life-base mechanism. Metabolism is known for creating permanent disturbs producing mechanical energy (when exercising) and sustaining life. On other hand, homeostatic processes are responsible for keeping normal biological conditions. Here, some clues about the complexity of the active system modelling can be found.

A simple example is the exercise in hot environments. Exercising produces a huge amount of endogenous heat that needs to be eliminated. Vasodilation and sweating are the most used mechanisms that help to keep core temperature at reasonable values. However, the excess of regulation leads the organism to fail and overheat (developing pathological heat conditions). Vasodilation and the increase of blood flow is not only used to cool down the body but also to assure arterial blood pressure and a good level of oxygen supply to the muscles in exercise. For an efficient sweating process, high water levels in the body are essential. If the body sweats to much to maintain core temperature, blood pressure will decrease due to blood level depletion and might also begin an electrolyte imbalance combined with dehydration. These conditions can result in serious health problems that could even lead to death.

In brief, active system acts in the passive system to keep temperature under control. Together they build a closed loop mode process where environment and physical activity can become a disturbance. Thermal sensors in passive system gather information about body elements and hypothalamus decides the reaction mechanism. This cycle of measuring/acting is the main basis of the thermal homoeostatic process. The active system, continuously analyses the passive one acting together to keep the necessary body balances. This positive interaction is only broken if the biological break point is reached.This is another important feature that should not be forgotten. Predict the individual break point is a challenge because it depends on many individual, lifestyle and instant factors. But it is important for the success of the simulation to critical environmental exposures that the limits have been established, otherwise heat tolerance could never be predicted.

4.5 Model design

Modelling process began by defining the main modules of the program. To do so, were applied top down design techniques. Human thermal heat response were divided in:

1. physical body components as physical dimension subjected to heat diffusion processes (the called passive system);

2. central nervous system, that reacts to differences in temperature perceived by the thermal receptors (located in the skin and hypothalamus) cooling or heating the body through thermoregulation mechanisms.

To describe the passive system the body was considered as a whole. Different parts of the body has different characteristics and the heat transfer is done by different ways. For example, the limbs have similar structure, with the same kind of biological tissues and disposition. However, trunk include extra features, because of the respiratory tract. When modelling the breathing process is important to include the mass exchange. Heat exchange in the head is also affected by respiration, specially in critical exposures. The first step was to segment the body into pieces. The division consists in get body elements small enough to represent the different body positions, contacts to set respective boundary conditions. Limbs can be divided into three pieces each according to the main joints. Hands and feet are considered similar because they are all extremities and have almost the same anatomical structure and circulatory model approach. The same happens in the intermediate structures of the limbs.

In spite of the differences between structure and circulatory process, each body element is composed by large blood vessels reservoirs (which intend to model heat transfer between the two pipes containing liquid at different temperatures) and biological tissues (that gathers the heat diffusion through tissues and includes small vessel impact by considering blood irrigation). In short, the base of the passive system model is represented through:

1. Heat diffusion throughout tissues and small vessels;

2. Heat transfer between countercurrent large vessels.

The active system is more complex to model. Although the passive system modelling is multi-factorial, the active system involves a larger number of variables which impact is, usually, impossible to predict. In this context this modelling is trying to find common guidelines from which will can begin to have sufficient resources to adapt a general forecast to any particular individual. Mainly, the active system here is defined through :

1. Complete cardiovascular modelling ;
2. Local sweating response.

4.5.1 Body Element Design

Body segments are the key of the passive system modelling. Gathering all the segments is obtained a whole body thermal model. By programming the interaction between some variables (such as endogenous heat production and blood flow) it is settled the thermoregulatory system. As mentioned above, the model of human thermoregulatory system is composed by active system, that includes the physiological mechanisms of thermoregulation, and the passive system that models the thermal behaviour of the biological tissues, its geometry and distribution.

The passive system is modelled through thermodynamic laws of heat transfer. The general heat transfer equation in space and time can be written as shown in equation 4.1:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u \tag{4.1}$$

Being $u(x,y,z,t)$ function of space and time and α the thermal diffusivity. Considering the kind of geometry, Laplacian Operator (∇^2) can assume Cartesian, cylindrical and sphere coordinates in one, two or three dimensions. If a volume generate, itself, heat the equation 4.1 becomes:

$$\frac{\partial u}{\partial t} = \alpha \nabla^2 u + q \tag{4.2}$$

where q represents a known function that depends on both, time and space.

The bioheat equation developed by Pennes in 1948, was based on the heat transfer equation that comes from the general conservation equation. It gathers in a single equation the heat diffusion in space over the time including the endogenous heat production and tissues blood irrigation.

Tissue and Small Vessels Heat Transfer Modelling

The reference to the heat transfer in biological tissues presents the first important development in 1948 with Pennes' bioheat equation . One of the biggest challenges of the bioheat modelling it was how to adapt the basic principles of thermodynamic to the heat transfer in biological tissues once it depends a lot on tissues blood flow.

Is this characteristic that makes difficult modelling heat transfer in living tissues (Chen and Holmes, 1980). Modelling heat transfer considering the full anatomical model of the human-being was, at the time, impracticable. In order to overcome this difficulty, new modelling techniques were investigated about how to include the contribute of small vessels and capillaries in the mathematical approach. The bioheat equation, as presented by Pennes (1948), assume the original form of equation 4.3.

$$cp\frac{\partial^2\theta}{\partial r^2} = -K\nabla^2\theta + h_m + h_b \tag{4.3}$$

In 1984 become clear that the bioheat equation should be applied to body elements with multiple tissues by dividing it, at least in three layers (bone, muscle and skin) being adipose tissue considered inside the skin layer. The micro blood supply of the tissues vary with tissues' deepness in the organism (Weinbaum et al., 1984) and with the kind of biological tissue (Kuznetz, 1979). This differences do not influence significantly the way as heat is transferred through tissues and is diffused to outside the organism. The model suggested by Weinbaum et al. (1984) define a way to include the change of density, size and velocity of the countercurrent blood flow of artery-venous function of deepness (relatively to skin surface), direction of blood perfusion in transversal vessels' layer and the siding of the blood to the cutaneous layer (Jiji et al., 1984).

So, based on these two studies, Weinbaum and Jiji (1985) published an article with a new, improved version of the bioheat transfer equation (known as the W-J equation).

From here it is concluded that the term considering the isotropic blood perfusion, in Pennes' bioheat equation, is negligible due to microvascular organization, and that the primary mechanism for energy exchange between blood and tissue is the incomplete exchange in thermally significant microvessels (Weinbaum and Jiji, 1985). In their equation, Weinbaum and Jiji (1985) include the effect of tissue's conductivity tensor as function of local vascular geometry and the blood flow velocity in thermally significant countercurrent vessels. These studies become very important in the ascertainment of heat transfer in the human body and are very important modelling proposals in the whole-body human thermal phenomenological mathematical modelling.

Following this research guidelines Song, in partnership with Weinbaum and Jiji present a new implementation solution of the W-J equation to model the periperal tissue, equation that can be adapted to model heat transfer in limbs or whole-body (Song et al., 1987).

In spite of the numerous criticism about Pennes' bioheat equation, Wissler (1998) proved in his study that the assumptions made by Penne in his model are reliable and that the criticism about his study has no serious fundamentals. From Pennes' study it becomes clear that the heat exchange between large blood vessels and surrounding tissue affects tissue temperature profiles. Further studies proved that even the heat transfer that occurs between smaller vessels that supply the capillary beds affects the blood temperature entering those beds. That's why the arterial temperature, T_ar in Pennes' equation is not a constant but a time function.

It is truth that both, Pennes (1948) and Wissler (1998), ignore many factors that can influence the accuracy and reliability of results but they were aware of it. It is known that the geometry and the inhomogeneity of the examples structures influences the accuracy and reliability of the results, however, with new and more efficient numerical methods, nowadays, with computers technology, this problem became relatively simple to solve. Heat transfer between thermal significant vessels are more difficult to resolve (Wissler, 1998).

Nevertheless, the need for models based on microscale modelling technique in opposed to the macroscale mentioned above (Wang, 2000). The micro-scale modelling focuses its efforts in reproducing the exact phenomena that occur in biological tissues by taking into account the exact structure and biology of the element. In other words, it starts to model the heat transfer from a microscale perspective, at an almost cellular level. This approach based itself in Porous Media, while macroscale models based their theories in Continuum Mechanics.
In their study, Wang and Fan (2011) extensively reviewed the equations and the simplification methods, in order to maximize the predictive power of the models. According to , models that come from the Porous Media are more accurate, and have a powerful predictive ability than those that come from Continuum Mechanics. On the other hand , they are too complex and difficult to implement on macroscale. That is why this technique is widely used in areas where accuracy of the results are vital, and where there is no need for extensive macro scale process.

The reliability of the results produced by Pennes' bioheat equation and the simplicity of its implementation make of this model an example of success in the whole-body thermal modelling and it stills been applied nowadays (Wissler, 1998; Fiala et al., 1999; Ferreira and Yanagihara, 2009; Albuquerque-Neto and Yanagihara, 2009).

To solve the equation, thermal properties of the biological tissues need to be defined. In the present model, solution domain consider 8 types of human tissues. Limbs and neck are constituted by four layers of tissue skin, fat, muscle and bone. The head and trunk are the most complex structures that, beyond these layers, include respectively brain and lung, heart and viscera tissues. Table 4.1 gathers thermal properties for each type of tissue, such specific weight (ρ_t), specific heat (c_t) and thermal conductivity (k_t), assuming that for each type of tissue there are no significant differences between the molecular composition of the body elements. This assumption allows to simplify the modelling process, however it is possible, by gathering further information, to change the method of calculus and program the spatial variation of thermal properties inside each body element (Guyer et al., 2009).

As discussed above, not only thermal properties are needed to achieve the solution of Penne's equation. Basal values of blood flow, endogenous heat and arterial

Table 4.1: Thermal properties according to the type of tissue [adapted from:(Albuquerque-Neto and Yanagihara, 2009)].

Tissue	Property					
	Volume (cm^3)	Basal Blood Flow ($cm^3/m^3.s$)	Specific Weight (kg/m^3)	Specific Heat ($J/kg.°C$)	Thermal Conductivity ($W/m.°C$)	Endogenous Heat (W/m^3)
Skin	3541	362	1085	3680	0,47	368
Fat	10657	3,6	920	2300	0,21	4
Muscle	26330	543	1085	3800	0,51	684
Bone	7576	0	1357	1700	0,75	0
Brain	1514	9000	1080	3850	0,49	9472
Lung	2481	834	560	3520	0,28	339
Heart	298	14400	1080	2550	0,47	24128
Viscera	10301	5800	1080	3504	0,49	3852

blood temperature are necessary as an input. For resting response at thermoneutral conditions, basal values of blood flow and endogenous heat can be settled as constant without compromise accuracy and efficiency of the model. Of course that individual differences and sickness result in changes of basal values and might produce non constant, abnormal responses. As in all modelling processes specific situations must be simplified in order to get the best balance between number of program features and produced accuracy and efficiency. For exercise/work or non-thermoneutral states both, blood flow and endogenous heat production, change over the time. This changes are modelled by other mathematical functions. Blood flow depends on the cardiac output and cardiovascular impedance modelling that are dependent on exercise. Endogenous heat depends on the active elements of the body and the type of exercise. The input values also need to be predicted by other functions and depend mainly on the $\dot{V}_{O_2}$ uptake. Arterial blood temperature is not a constant vale such as was suggested by Pennes (1948) and Wissler (1998), but another variable modelled by a differential equation. Further information can be consulted in section .

Model must show sensitiveness but a structure easy to change and adapt that is able to easily produce clear outputs is an important goal to achieve.

Counter Current Large Vessels Modelling

The named, conventional models, are the base of all the modern and accurate new bioheat models. New models can be evaluated as upgrade, a mixture of the principles of the oldest models. As all pioneer studies, their principles and decisions have been widely discussed and severely criticised. The new mathematical models (such as dual-fase-lag bioheat model, porous media based in bioheat models, the discrete vascular bioheat model) used that criticisms to improve the model in order to reach more realistic results.

One of the main critics in all continuum models, is that they are based in the Fourier's law of heat conduction that assumes that a perturbation in the thermal state of a structure is felt, at the same time, in all regions of that structure. One problem is that if the generalized form of Fourier's law is applied to non homogeneous structures it gives non realistic results.

To correct this miss purpose in the heat conduction equation it is included phase-lag arising due to inertia (phase-lag of heat flux which is the relaxation time due to fast transient effects or small scale effect of heat transport in time) and micro-structural interaction (phase-lag of temperature gradient observed at the micro scale heat transfer). These corrections are particularly efficient and important when modelling local heating in hyperthermia treatments. In their study Askarizadeh and Ahmadikia (2014) shown that when heat flux relaxation time and temperature gradient relaxation time are equal, dual-phase lag equation is reduced to Penne's bioheat equation. The differences remain when the relaxation times parameters are different. In that case the results of Pennes' bioheat equation over estimate the temperature and damage profiles in skin tissues. Porous media modelling based on the detail of the biological tissues where velocity vector, local porosity which is the ratio of blood sub-volume to total volume, thermal conductivity, permeability tension, specific surface area, inter-facial convection heat transfer coefficient, blood perfusion rate, are mixed in a simple mathematical model. However the mathematical implementation of these models depend on detailed informations about human local anatomy and are much difficult to solve, specially in macro-scale whole-body models.

It is clear that the discussion of large vessels modelling falls into bioheat modelling of thermal diffusion adopted for modelling tissues and small vessels. In this context it was a big step for whole-body thermal modelling to associate more than one mathematical approach in a single model. This was already tried by Neto (2010) and final results found good agreement with experimental data found in literature.

$$C_{bl,a} \cdot \frac{dT_{bl,a}}{dt} = -\left[\sum_{\substack{arteries \\ arterioles}} h(t) \cdot A_{artery} \right] n \cdot (T_{bl,a} - T_{cr}) + \dot{m} \cdot c_{bl} \cdot (T_{bl,a,adjacent} - T_{bl,a}) \quad (4.4)$$

Here is intended to follow similar approach but this time by modelling the circulatory system as a parallel system where are determined the values of arterial and venous temperatures and blood flows. For modelling the heat transfer due to large vessels it was assumed that at each segment it is found an arterial and a venous node. So heat transfer is calculated based on the equations for arterial end venous blood (Salloum et al., 2007). Here arteries and veins are considered two cylinders positioned side by side (Weinbaum et al., 1984; Ferreira and Yanagihara, 2009; Neto, 2010). From one to the other there are a heat exchange that depends on the volume of blood flow and respective temperature over the time. Convective heat transfer is distinguished from one to another due to the pulsating effect of the blood flow. So, while the convective heat transfer from arteries to veins depends on time.

Salloum et al. (2007) in his thermal transient model defines the total heat exchange from the core to the arterial blood as equation 4.4. Not only blood flow changes pulsation over the time, as discussed next section, but is also depends on the length of the arteries.

The implemented model approach simplifies, somehow, this approach by using the equations 4.5 and4.6 proposed by Wissler (1964) as followed by Ferreira and Yanagihara (2009) and Neto (2010).

$$V_{ar}\rho_{bl}c_{bl}\frac{dT_{ar}}{dt} = \dot{V}_{ar}\rho_{bl}c_{bl} \cdot (T_{ar,in} - T_{ar}) + H_{av} \cdot (T_{ve} - T_{ar}) \quad (4.5)$$

Table 4.2: Heat transfer coefficients between arteries and veins for each segment, respective length and blood volume per reservoir. [adapted from:(Ferreira and Yanagihara, 2009)].

	Property					
Tissue	Arterial Vol. (cm^3)	Venous Vol. (cm^3)	Length (cm)	Pairs Num.	h_{av} ($W.K^{-1}.pair^{-1}$)	H_{av} ($W.K^{-1}.pair^{-1}$)
Head	40	180	19	-	0	0,00
Neck	15	66	8	16	0,097	1,55
Trunk	446	1484	58	-	0	0,00
Arm	24	107	31	7	0,377	2,55
Forearm	13	60	28	4	0,341	1,43
Hand	7	30	19	3	0,231	0,72
Thigh	78	349	44	15	0,535	8,26
Leg	35	155	40	8	0,487	3,67
Foot	15	66	26	5	0,316	1,55

$$V_{ve}\rho_{bl}c_{bl}\frac{dT_{ve}}{dt} = \rho_{bl}c_{bl} \cdot \left(\sum \dot{V}_{sv_i}\bar{T}_{ve_i} + \dot{V}_{ve,in}T_{ve,in} - \dot{V}_{ve}T_{ve}\right) + H_{av} \cdot (T_{ar} - T_{ve}) \tag{4.6}$$

The biggest difference from Wissler's original formulation to Ferreira and Yanagihara (2009) and Neto (2010), is on the use of a single convective heat transfer coefficient between arterial an venous reservoirs H_{av}. These values were calculated using shape factor for two-dimensional conduction between two cylinders immersed in an infinite medium. It was assumed that arteries and veins got the same radii and the distance between them was two times their diameter in basal thermoneutral conditions. Resultant values (h_{av}) was multiplied by the number of arterio-venous pairs at the segment and H_{av} was found. Volumes of the reservoirs were based on vascular volume data of Chen and Holmes (1980), and the blood distributions followed normal adult male of 67 kg with approximately 5.5 L of blood. Detailed cardiovascular data used in the dissolution of arterio-venous heat exchange coefficients are shown in table 4.2.

The implemented model, BioHeatSIM, followed the research lines of Ferreira and Yanagihara (2009). Large arteries are seen as two parallel pipe reservoirs that exchange heat with each other but, with surrounds heat exchange is modelled by the blood flow through tissues and small vessels.

$$V_{ar}\rho_{bl}c_{bl}\frac{dT_{ar}}{dt} = \dot{V}_{ar}\rho_{bl}c_{bl}\cdot(T_L - T_{ar}) + H_{av}\cdot(T_{ve} - T_{ar}) \tag{4.7}$$

$$V_{ve}\rho_{bl}c_{bl}\frac{dT_{ve}}{dt} = \rho_{bl}c_{bl}\cdot\left(\sum\dot{V}_{sv_i}\bar{T}_{ve_i} + \sum\dot{V}_{ve,in}T_{ve,in} - \dot{V}_{ve}T_{ve}\right) + H_{av}\cdot(T_{ar} - T_{ve}) \tag{4.8}$$

The two segments that do not consider the exchange between large vessels is the head and the trunk ($H_{av} = 0$). In the head due to the significant quantities of blood spread throughout all brain tissue and other layers do not have significant large vessels to consider. In the trunk, probably due to the close connection between reservoirs, that occurs through the lungs that represent the higher heat exchange rate. All blood is received by the limbs and trunk tissues (eq. 4.8) and sent to the lungs, becoming arterial blood that gets in the trunk arterial reservoir (eq. 4.7). This way arterial temperature of the trunk depends on temperature of the blood that come from the lung (T_L).

Body Segment Structure

Heat diffused through tissues and small vessels is done by using the bioheat partial differential equation and differential equation to large vessels . However to be solved they follow some structured rules according to human anatomy and physiology.

Figure 4.4 resumes the process of heat diffusion within one intermediate segment of the body. Layers represent the different types of tissues (skin,fat, muscle and bone). Blood flows from a anterior element into segment with temperature (T_{ar}) and flow ($\dot{V}_{ar}$). Part is diffused among tissue layers ($\dot{V}_{sv}$) and the rest goes to posterior element ($\dot{V}_{ar,out}$) (eq. 4.9).

$$\dot{V}_{ar,out} = \dot{V}_{ar} - \sum\dot{V}_{sv_i} \tag{4.9}$$

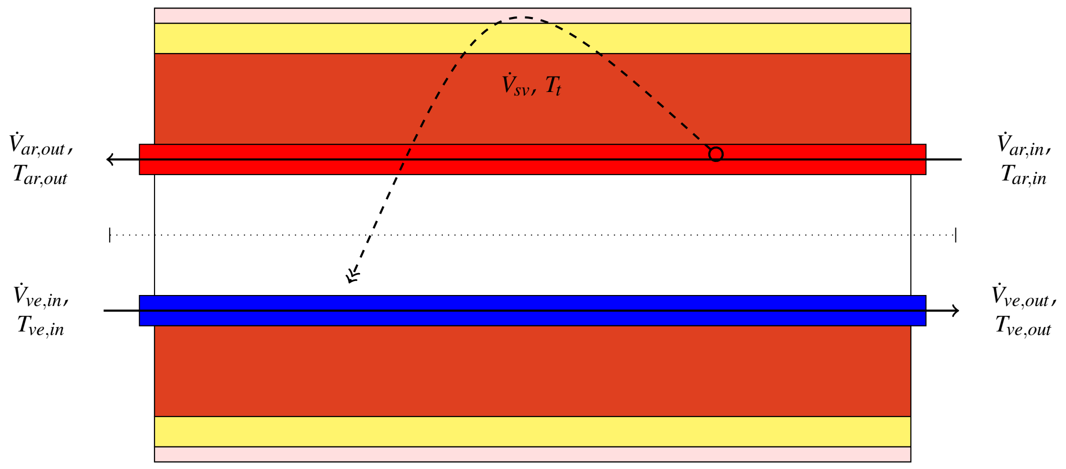

$\dot{V}_{sv}$, T_t
$\dot{V}_{ar,out}$, $T_{ar,out}$
$\dot{V}_{ar,in}$, $T_{ar,in}$
$\dot{V}_{ve,in}$, $T_{ve,in}$
$\dot{V}_{ve,out}$, $T_{ve,out}$

Figure 4.4: Heat exchange in middle elements.

Through venous reservoir receives the blood from the tissues ($\sum \dot{V}_{sv}$) and posterior elements ($\dot{V}_{ve}$). So, venous blood that flows out of the segment is given by the sum of all the flows that come from the i^{th} volume of tissue layer plus the blood flow from posterior element (V_{ve}) (eq. 4.10).

$$\dot{V}_{ve,out} = \sum \dot{V}_{sv_i} + \dot{V}_{ve} \tag{4.10}$$

In opposite to the $\dot{V}_{sv}$ in bioheat equations, that uses the flow per unit of volume, the tissue blood flow here already considers the volumes of the i^{th} element of the tissue (eq. 4.11).

$$\dot{V}_{sv_i} = \int \dot{V}_{sv} dV \tag{4.11}$$

In body segment the calculus needs; first the blood flows that get in the segment ($\dot{V}_{ar}$ and $\dot{V}_{ve}$); and second the tissue blood flow ($\dot{V}_{sv_i}$) characteristic from that segment. The temperatures are calculated based on the sequence of blood circulation. First arterial blood temperature, then tissue temperature and finally venous temperature. Notice that final temperature at venous reservoir needs the final temperature of arterial reservoir. And the arterial temperature that gets in the tissues at that instant is the initial temperature of arterial reservoir. And at the same time the blood that gets in the venous reservoir is based on initial temperatures of the tissue.

All the segments with exception of the ends and trunk follow this approach. End segments have shunts between arteries an veins called arteriovenous anastomoses. In physiology the explanations and modulations of these shunts can be a difficult part of hypothermia modelling Karaki et al. (2013).

But in heat stress the typical modelling allow model the ends with the same two blood reservoirs linked, where the output and input blood that crosses the segment can be approximated by equation 4.12 and, as the input blood flow is based on segment tissues' blood requirements, $\dot{V}_{ar,out} \approx 0$.

$$\dot{V}_{ve,out} = \sum \dot{V}_{sv_i} + \dot{V}_{ar,out} \tag{4.12}$$

The success to the implementation of these principles and equations is a carefully synchronization and parametrization.

4.5.2 Integrating Segments - Whole Body Structure

Basically the final integration of the segment is done through the trunk segment. As suggested and presented by several authors (Wissler, 1985; Stolwijk, 1971; Fiala, 1998; Tanabe et al., 2002; Ferreira and Yanagihara, 2009; Albuquerque-Neto and Yanagihara, 2009), body physical structure can be approximated through geometrical segments, with size and dimensions, and composition similar to the real humans.

Here the pieces are connected to each other through arteries and veins that carry the blood among segments. The connection resumes to sequential blood flows and respective temperature that represent the heat that crosses from one segment to another.

As shown in figure 4.5, hands and feet segments are represented by parallelepipedal geometries and the rest of the elements are based on cylindrical geometries. Geometries are represented at real size volumes. All geometries are represented by concentric layers of different types of tissues as the axial cut in figure 4.4 represents.

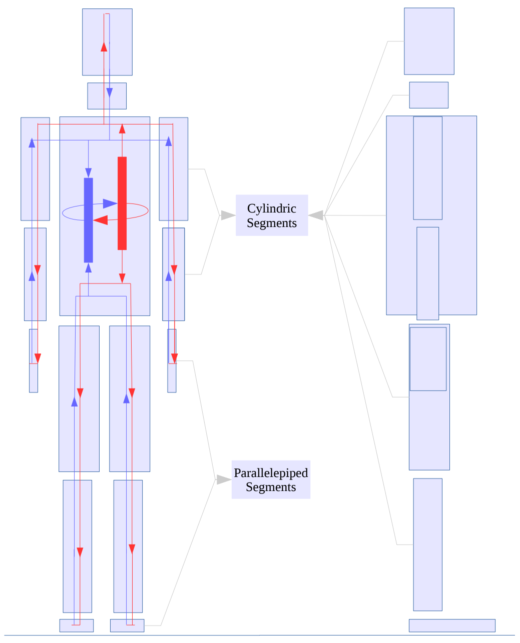


Figure 4.5: Structure of model's segments and blood flow scheme.

The only exception is the trunk structure that is composed by 3 cylindrical structures (one first end at the beginning of the heart layer, the other at the end of heart, lung and bone layer and the rest including the viscera at the abdomen level).

At the trunk, calculations need to consider the several connections of core venous pool to the 5 posterior parts (the limbs and the head), so that the venous final temperatures and flows depend on them. The linkage between arterial and venous reservoirs is done as occurs in small circulation, through the lungs. This way lungs receive blood

Table 4.3: Geometrical data of body elements currently used. [adapted from:(Ferreira and Yanagihara, 2009)].

Element	Volume (cm^3)	Surface Area (cm^2)	Length (cm)	Diameter (cm)
Head	3.542	1.135	20	14,6
Neck	850	294	8	11,
Trunk	34.758	5.985	58	26,0
Arm	1.766	831	31	9,0
Forearm	988	601	28	7,4
Hand	500	450	19	-
Thigh	5.224	1.701	44	13,4
Leg	2.317	1.080	40	8,6
Foot	980	630	26	-

equal to CO (eq. 4.13) and that would be approximated equal to the sum of the venous blood that come from all segments (right arm, left arm, right leg, left leg and head), plus the amount that comes from trunk tissues (with the exception of lungs layer) (eq. 4.13).

$$\dot{V}_{ve_{lungs}} = CO \approx \dot{V}_{ve,out} \tag{4.13}$$

$$\dot{V}_{ve,out} = \sum \dot{V}_{sv_i} + \dot{V}_{ve,out_{arm_r}} + \dot{V}_{ve,out_{arm_l}} + \dot{V}_{ve,out_{thigh_r}} + \dot{V}_{ve,out_{thigh_l}} + \dot{V}_{ve,out_{neck}} \tag{4.14}$$

The elements used to build the physical structure of the human body are presented in table 4.3.

Chapter 5

Program Implementation, Development and Test

As mentioned before, in spite of modelling process design began from the top level, the implementation process, due to complexity and the need for systematic testing use a spiral design technique. So, the program starts with the basic routine for the simulation of heat diffusion in tissues and small vessels, in the simplest form (unidimensional transfer in a 4 layer cylindrical object divided into 4 cells across the ray) and the development was assured by adding continuously new features to the program. The improvements done to the previous routines are only done after they were fully tested and the results produced validated.

The program began with a single small routine that gathered all tasks (meshing, plotting, visualization tasks and others). When this file become more complex and too long it was changed to a modular design, to allow independent development of each task and make it faster. In the modular version all functions were rewritten into smaller and simpler files that were divided according to its functionality and saved in folders. Those functions could be classified as meshing, plotting, viewing and heat diffusion routines. They were saved in folders with the name of the respective classification. The following section details the content of the routines and the development process.

5.1 Implementation

The program consists in a group of routines that interact each other in order to produce the required result. They are organized in folders according to their type or functionality. So, the main folder contains the main routine to run the program and the sub-folders contain the routines responsible for meshing, plotting, viewing and those which relate the basic routines for body segments, the middle elements routine, the extremity elements routine, trunk routines and so on. This results into a modular structure that allow adding separate development features

5.1.1 Tissue and Small Vessels

The bioheat equation

The heat transfer through tissues and small vessels is properly reproduced by the classical equations of heat diffusion that consider the properties and geometry of the tissues from body elements.
The internal heat through tissues follows the energy conservations equations, being the presented bioheat equation, by Pennes (1948), an evolution of the general heat conduction.
So, the equation 5.1 establishes the heat transfer between the different biological tissue layers (bone, muscle, fat and skin).

$$\rho_t c_t \frac{\partial T_t}{\partial t} = k_t \nabla^2 T_t + \hat{\dot{V}}_{sv} \rho_{bl} c_{bl} \left(T_{ar} - T_t\right) + \hat{\dot{q}}_t \tag{5.1}$$

being:

- $\rho_t \rightarrow$ specific weight of the tissue (Kg/m^3);
- $c_t \rightarrow$ specific heat of the tissue ($J/Kg.°C$);
- $T_t \rightarrow$ temperature of the tissue ($°C$);
- $k_t \rightarrow$ thermal conductivity of the tissue ($W/m.°C$);

- $\nabla^2 T_t \rightarrow$ temperature gradient of the tissue ($^\circ C/m$);
- $\hat{\dot{V}}_{sv} \rightarrow$ arterial blood flow of the body segment ($m^3/m^3.s$);
- $\rho_{bl} \rightarrow$ specific weight of the blood (Kg/m^3);
- $c_{bl} \rightarrow$ specific heat of the blood ($J/Kg.^\circ C$);
- $T_{ar} \rightarrow$ arterial blood temperature ($^\circ C$);
- $\hat{q}_t \rightarrow$ endogenous heat production of the tissue (W/m^3).

The scheme of tissues' temperature gradient depends on the geometry of the volume that is being modelled (cylinder or parallelepiped) and the number of dimensions for which heat transfer is modelled (uni-dimensional, bi-dimensional or three-dimensional transfer).

The gradient for a three-dimensional heat transfer in a parallelepiped, adopting orthogonal coordinates, assume the form of 5.2:

$$\frac{\partial^2 T_t}{\partial x^2} + \frac{\partial^2 T_t}{\partial y^2} + \frac{\partial^2 T_t}{\partial z^2} \tag{5.2}$$

where:
$x, y, z \rightarrow$ orthogonal coordinates (m).

The gradient above can be adapted to the other uni or bi-dimensional diffusions. To do so it is enough to delete the element(s) correspondent to the coordinate(s) for which the heat conduction is nearly null or null.

In the case of cylindrical volumes the implementation of cylindrical coordinates is more convenient. In these cases the gradient that corresponds to a three-dimensional heat transfer presents itself in (5.3):

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T_t}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 T_t}{\partial \theta^2} + \frac{\partial^2 T_t}{\partial z^2} \tag{5.3}$$

being:

- $r, z \rightarrow$ cylindrical coordinates (m);
- $\theta \rightarrow$ cylindrical coordinate (rad).

Similarly to what happens with the gradient in polar coordinates, so the gradient in cylindrical coordinates can be resized. According the elements that reach null values result different situations of heat transfer. When the gradient in θ component reach zero, the heat transfer resumes to a bi-dimensional heat transfer of a orthogonal matrix, whit the same characteristics of a bi-dimensional gradient in a parallelepiped. In the case of the gradient be along the z component it is obtained a polar bi-dimensional heat diffusion. There is even the case of unidimensional heat diffusion that in the case of cylindrical geometrical volumes eliminate, typically, the θ and z coordinates. This situations are, in everything, similar to the unidimensional heat diffusion in tissues in orthogonal coordinates.

Form and Terms of the Bioheat Equation

Partial Differential Equation (PDE) are commonly used to mathematically describe physical, chemical and biological phenomena. Processes involving changes in space and time are only solved through PDE. Pennes' bioheat equation come from the energy conservation equation in three dimensions. The energy conservation equation can simulate different kind of energy transference.
The energy flow due to heat conduction is a particular case (Neto, 2010; Pennes, 1948). So, it is possible to characterize bioheat equation, considering the homologous terms, in the energy conservation equation (5.1):

1. $\rho_t c_t \frac{\partial T_t}{\partial t}$ - Represents the transient term of the equation, in other words, the change in temperatures over the time;
2. $k_t \nabla^2 T_t$ - Represents the spacial diffusion term, that shows temperature variation over the physical geometry;
3. $\hat{V}_{sv} \rho_{bl} c_{bl} (T_{ar} - T_t) + \hat{q}_t$ - Represents the source term, responsible for the endogenous heat production and the quantity of heat provided to the body segment.

Table 5.1: Summary of the classification and behaviour of general forms of the equation of heat conduction [adapted from: (Versteeg and Malalasekera, 2007).]

Problem Type	Equation Type	Equation Prototype	Conditions	Solution Domain	Solution Stability
Balance problems	Elliptical	$div grad \phi = 0$	Boundary Conditions	Closed Domain	Stable Solution
Dynamical problems with dissipation	Parabolic	$\frac{\partial \phi}{\partial t} = \alpha \, div \, grad \, \phi$	Initial and Boundary	Open Domain	Stable Solution

The general conservation equation can be solved by applying the Finite Volume Methods (FVM) to any combination of the following terms (Guyer et al., 2009):

- Transient;
- Convective;
- Diffusive;
- Source.

In the particular case of solving the bioheat equation, that models the heat transfer through tissues and small vessels, convective term does not exist. In turn, the transient term assume a null value when lets to depend on time. That happens when is reached the called steady-state. Heat conduction over the tissue is represented in steady-state by an Elliptical equation ($\frac{\partial \phi}{\partial t} = 0$) or, in transient state by a Parabolic equation ($\frac{\partial \phi}{\partial t} \neq 0$) (see Table 5.1).

The physical meaning, classification and term definition, as well as the form that bioheat equation assumes, are critical to choose the suitable method for numeric solving.

5.1.2 Solving the Bioheat Equation

The bioheat equation is a partial differential equation that can assume, in the present work Parabolic and Elliptical forms. To solve partial differential equation there three methods that fit frequently used:

- Finite Element Methods - FEM;

- Finite Difference Methods - FDM;
- Finite Volume Methods - FVM.

FEM has been the most used among the choices. The method consists in using a simple approach to unknown variables to transform partial differential equations into algebraic equations, called finite elements, and uses variational methods to solve and minimize the error (Dhatt et al., 2012).

FDM is based on discretization method, in the opposite to variational method of FEM, to solve partial differential equations by approximating them with difference equations, in which finite differences approximate the derivatives.

FVM are closely related to FDM, and a FVM can often be interpreted directly as a finite difference approximation to the differential equations. Both use discretization methods to describe the problem. However, finite volume methods are derived on the basis of the integral form of the conservation law, a starting point that turns out to have many advantages (LeVeque, 2002). There were a few methods commonly used to solve the model. FEM and FVM were pointed as the most used due to simplicity and accuracy. In spite of some specialists defend that the accuracy level of the FEM method is higher but, usually, solution produced with FVM algorithms are more stable. The truth is that both methods has advantages and disadvantages. Depending on the problem formulations results can be accurate, precise and quite fast using any of these methods.

In his research Wang et al. (2010) tested the application of FVM to transient heat transfer and found that the success of the precision was the method for selecting Control Volume. By comparing the results that come out for FVM and FEM they found that the results calculated by FVM where more similar to the theoretical solution.

Finite Volume Methods

To apply the FVM the solution domain must first be defined by a mesh. A mesh consists in groups of cells, faces and vertexes, that divide the solution domain into non-overlapping polyhedral elements. A mesh can assume uni-dimensional, bi-dimensional or three-dimensional shape. So cells can be reduced to areas or extended to volumes.

There are several numeric techniques of FVM. Most of them depend on the way the CV is selected. Basically there are two processes of grid definition. Cell centred (CC-FMV) CV , where cells match CV; and vertex centred (VC-FMV) CV each cell is subdivided into sub-control volumes. In VC-FVM the variable values are "stored in the vertexes, CC-FVM "stores" it in the centre of the cells (Figure 5.1). But the big difference is that VC-FMV need to construct their own faces within the cells instead of use the face cells. So, face fluxes use all the vertex values from the cell to calculate interpolations instead of approximate them by using the variable values in the two adjacent cells surrounding the face. That is why the VC-FVM is not so fast and requires more storage do execute the calculus, in spite of being more suitable to any kind of topology (Guyer et al., 2009).

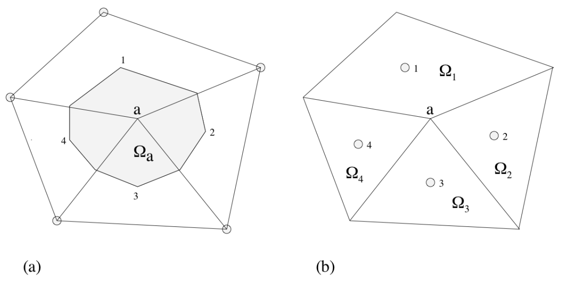


Figure 5.1: Control Volume structure for an unstructured mesh. (a) Ω_a represents a vertex centred control volume and (b) Ω_1, Ω_2, Ω_3 and Ω_4 represent cell centred control volumes.

Then the discretization process consists in integrate all the terms of the partial differential equation in the CV defined by the mesh. In transient problems the calculations are repeated for small steps of Δt being the initial values of the mesh the ones that came from the previous calculus step. The final result is obtained when t reaches the $t_i + \Delta t$.

FiPy Tool - Calculation Method

The tool used to solve partial differential equation is, as mentioned before, Fipy. The differential equations can be analysed as a combination of terms (transient, convection, diffusion and source), each of them representing an influence on the unit-volume basis. Together they describe a balance or a conservation.

The dependent variable ϕ of general differential equation 5.4, presented below, is usually specific property (such as mass fraction, velocity and specific enthalpy), or, in other words, quantities expressed on a unit-mass basis (Patankar, 1980).

$$\frac{\partial}{\partial t}(\rho\phi) + div(\rho\mathbf{u}\phi) = div(\Gamma_\phi grad\phi) + S_\phi \tag{5.4}$$

Translating it into words it means that: the Rate of increase of ϕ of element, plus the Net rate of flow of ϕ out of the element, is equal to the Rate of increase ϕ due to diffusion, plus the Rate of increase of ϕ due to sources (Versteeg and Malalasekera, 2007).
This equation can be transformed to adapt to the physical meaning of the problem that need to be solved. The transient coefficient ρ, the convection coefficient $\mathbf{u}$, the diffusion coefficient Γ_ϕ and the source term S_ϕ have an appropriate meaning according to the property in study. The dependent variable can be a function of three space coordinates and time (Equation 5.5).

$$\phi = \phi(x,y,z,t) \tag{5.5}$$

The independent variables (x, y, z and t) help to define the dimension of the solution variable in time and space. So, in its extensive form, can be a three-dimensional unsteady problem.

This general formulation of partial differential equations is the key of the Fipy tool. Being the numeric algorithm based on this general assumption.

The discretization process is done by integrating each term of the equation 5.4 over the CV by using the CC-FVM approach. Then are made the appropriate approximations for fluxes across the boundary of each CV. Figure 5.2 shows an interpretation of the control volume in three dimensions using orthogonal coordinates. For the Transient Term $\frac{\partial(\rho\phi)}{\partial t}$ the discretization over the control volume V for the P node is given by

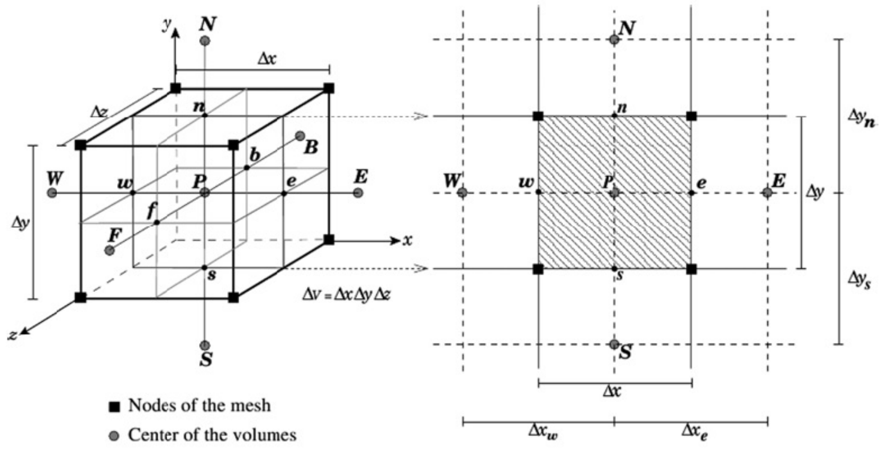


Figure 5.2: Control volume. Black squares represent the vertexes and dots represent the nodes in the middle of the control volumes. .

equation 5.6.

$$\int V \frac{\partial(\rho\phi)}{\partial t}\,\mathrm{d}V \simeq \frac{(\rho_P\phi_P - \rho_P^{old}\phi_P^{old})V_P}{\delta_t} \tag{5.6}$$

The ϕ_P represents the average value of ϕ in the centred point P and the superscript "old" indicates the previous time-step value. Values V_P and δ_t are respectively: the volume of the control volume with the centre in P and the time step size. The Fipy function used to set the transient term is defined by the following code:

```
>>> TransientTerm(coeff=rho)
```

Once again, the discretization for the convection term - $\nabla \cdot (\vec{u}\phi)$ - use the divergence theorem to transform the integral over the volume into and integral over the surface of the CV (Guyer et al., 2009). The solution of the integral is approximately equal to the sum over all the faces of the control volume (Equation 5.7).

$$\int_V \nabla \cdot (\vec{u}\phi)\,\mathrm{d}V = \int_S (\vec{n}\cdot\vec{u})\phi\,\mathrm{d}S \simeq \sum_i (\vec{n}\cdot\vec{u})_i \phi_i A_i \tag{5.7}$$

The vector $\vec{n}$ is a normal vector to the face of the CV that points out to the node of the adjacent cell. The vector $\vec{u}$ is the one which represents the velocity of the fluid in magnitude and direction. The underscript i assumes the form of the faces (f, b, w, e, s) as shown in Figure 5.2. The variable of interest ϕ_i is calculated taking into account, in a first order approximation, the value in the centre of the cell - the node P - and the values of the adjacent cell - $I = F, B, W, E, S$ or N. Its general form can be represented by Equation 5.8.

$$\phi_i = \alpha_i \phi_P + (1 - \alpha_i)\phi_I \tag{5.8}$$

The α_i factor depends on the convection scheme and can be defined by sorting the respective convection term function in FiPy. The selection can vary between the central differencing scheme, the upwind, exponential, hybrid, power law schemes by using the following functions respectively:

- CentralDifferenceConvectionTerm,
- UpWindConvectionTerm,
- ExponentialConvectionTerm,
- HybridConvectionTerm,
- PowerLawConvectionTerm.

The formulation code into Python becomes:

```
>>> <ConvectionTerm_function>(coeff=u)
```

where u is the velocity coefficient that must be represented numerically, in programming language by a list, a tuple or a FiPy FaceVariable.

Also the Diffusion Term is discretized by integrating it over the volume and can be simplified by transforming it into a surface integral which is similar to the flows over each face of the cell (Equation 5.9).

$$\int_V \nabla(\Gamma \nabla \phi)\, \mathrm{d}V = \int_S \Gamma(\vec{n} \cdot \nabla \phi)\, \mathrm{d}S \simeq \sum_i \Gamma_i (\vec{n} \cdot \nabla \phi)_i A_i \tag{5.9}$$

This represents a simple diffusion, whose code can be written as:

```
>>> DiffusionTerm(coeff=Gamma)
```

But higher order diffusion expressions can be defined and solved by specifying the various Γ so that the number of coefficients provided determines the order of the term. The flux over the faces is estimated making the difference between the ϕ variable of the successive cells and dividing it by the distance of the respective nodes (equation 5.10).

$$(\vec{n} \cdot \nabla\phi)_f \simeq \frac{\phi_I - \phi_P}{\mathrm{d}_A P} \tag{5.10}$$

This assumptions relies on the orthogonality of the mesh, what means that accuracy level decreases as the non-orthogonality increases. For last but not the least, any term of the mathematical expressions that do not belong the previous forms are considered part of the Source S_ϕ. The discretization for the Source Term, analogously to the other terms, is done by integrating it over the volume.

$$\int_V S_\phi \, \mathrm{d}V \simeq S_\phi V_P \tag{5.11}$$

The equation (5.11) does not include the negative dependence of S_ϕ on ϕ. This is done by adding a source S_0 which is independent on ϕ and a coefficient S_1 linearly dependent on ϕ, so that Equation 5.11 become Equation 5.12.

$$V_P(S_0 + S_1\phi_P) \tag{5.12}$$

Other explicit function could be chosen, however implicit method are always preferred for modelling, explicit functions are only used for illustrative purposes.

The Source Term can be codded in Fipy by:

```
>>> S_0 + ImplicitSourceTerm(coeff=S_1)
```

So, by adding the equations (5.6), (5.7), (5.9) and (5.12) it is obtained the general equations discretized for each CV (Equation 5.13).

$$\frac{(\rho_P\phi_P - \rho_P^{old}\phi_P^{old})V_P}{\delta_t} + \sum_i (\vec{n}\cdot\vec{u})_i \phi_i A_i = \sum_i \Gamma_i(\vec{n}\cdot\nabla\phi)_i A_i + V_P(S_0 + S_1\phi_P) \tag{5.13}$$

The goal of this discretization is to achieve a set of discrete linear equations used to calculate the value of the dependent variable at each CV, that results into sparse linear systems easily solved by iterative schemes (in FiPy encapsulated by PySparse and PyTrilinos solvers). The general equation 5.13 is in the form of a set of linear combinations between the control volume with the centre on P and their neighbours control volumes and can be rewritten as:

$$a_P\phi_P = \sum_f a_I\phi_I + b_P \tag{5.14}$$

where coefficients a_A, a_P and b_P are given by Equations 5.15, 5.16 and 5.17:

$$a_P = \frac{\rho_P V_P}{\delta_t} + \sum_f (a_I + F_i) - V_P S_1 \tag{5.15}$$

$$a_I = D_i - (1-\alpha_i)F_i \tag{5.16}$$

$$b_P = V_P S_0 + \frac{\rho_P V_P \phi old P}{\delta_t} \tag{5.17}$$

and the convective strength and diffusive conductance, F_i and D_i, are calculated trough Equations 5.18 and 5.19 respectively.

$$F_i = A_f(\vec{u}\cdot\vec{n})_f \tag{5.18}$$

$$D_i = \frac{A_i \Gamma_i}{d_A P} \tag{5.19}$$

The term definition, is not the only success key for the generic application of the FiPy solver. Object oriented design divides the Fipy in three fundamental classes: the Mesh, Variable and Term. The names of the classes indicate the type of objects that can be created. Mesh objects define the domain of interest. Variable object define the type of variable. And the term object allow to formulate a bigger term that defines the mathematical model of interest. Associated to the main classes there are other classes, and it is based on their relation that FiPy works (Figure 5.3).

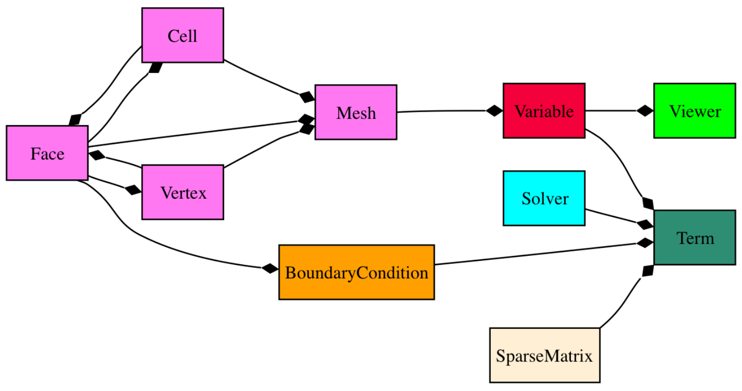


Figure 5.3: Objects and respective relation in FiPy.

As there are many types of functions to build the terms and the same happens with Mesh. Different types of mesh can be build and all of them have their own parameters that need to be settled. But in the end all of them end being object composed by cells, each cell defined by faces and boundary vertices. The Variable object needs the Mesh object associated to be defined and solved. The Term objects interprets the *Boundary Condition* and encapsulates the contribution to the Sparse Matrix. After applying a linear solver and get the solution, at each step of time it can be displayed in a Viewer object.

The high performance language that Python is, made Fipy an easy tool to operate using a few commands and lines of code to build up complex models. A simple model codded in FiPy resumes to:

- build the mesh;
- define the variable;
- specify the equation;
- and set a viewer.

5.2 Development Process

5.2.1 Unidimensional Diffusion - Annotated listing

The implementation of the program starts by modelling the heat diffusion through small vessels and tissues considering a 4 layer cylinder over the radial direction. This simple code was only used to test the stability of the solution over the 4 cell all with different thermal properties, boundaries and initial conditions.

In this case the mesh that represents the solution is created by a predefined function of FiPy. The following code lines (listing 5.1) create the cylindrical structure and take into account the thickness of each tissue layer. The `CylindricalGrid1D` function uses the number of cells over the x axis (`nx`) and their respective size (`dx`) to build the structure.

```
1 nx = 4 #number of cells of the mesh - correspond to the tissue layers
2 dx = array([2.68, 2.72, 0.31, 0.11]) # 2.68, 5.40, 5.71, 5.82 end of each layer in (cm)
3 dx = dx*10**-2
4
5 #Adaptation to neck cylindrical mesh# http://www.ctcms.nist.gov/fipy/fipy/generated/meshes.numMesh.html
6
7 fipy.meshes.numMesh.cylindricalGrid1D
```

Listing 5.1: Code lines that create the cylindrical mesh divided in 4 cells.

After defining the solution domain it is created the dependent variable `T_t`, that represents temperature of the tissue in each cell of the mesh. The variable `T_t` consists in a `CellVariable` object that works as a storage variable for later use. It integrates the initial conditions and is able to represent the field of values of any variable on a mesh.

```
1 T_t = CellVariable(name="Neck Temperatures Distribution",
2                    mesh=mesh,
3                    value=(36.5,36.,34.,33.))
```

Listing 5.2: Dependent variable definition.

The listing 5.2 shows the parameters that were settled. The `mesh` requires the grid data that divides the solution domain in control volumes as defined in the line 7 of the listing 5.1. The `value` parameter accept single values, vector or array variables according to the kind of initial conditions. So, by using arrays it is possible to set the value of a particular cell or group of cells according to the needs. This case specifies different temperatures to each one of the four cells of the mesh (listing 5.2, ln 3).

Each kind of tissue have its own values of specific heat (`c_t` or `c_bl`), specific mass (`rho_t`) and thermal conductivity (`k_t`). These constant values define the thermal properties of the tissues and are specific of the organic material and are saved in array lists (listing 5.3, ln 2-3, ln 11, ln 16-20). By the opposite, small vessels blood flow (`V_sv`) and internal heat production (`q_t`) depends not only on the kind of tissue, but also vary according to metabolic expenditure of the individuals and the body element. So the same organic tissue, i.e. in leg or in trunk, have different values **in time** and **over the time**. These key variables are the focus of the improvement of the model. The arterial blood temperature (`T_ar`) itself is settled as a constant, but in reality is a variable that depends on both, time and body element.

```
 1 #Defining the coefficient of the transient term - specific weight and
       specific heat of the tissues
 2 rho_t = array( [1357.,1086., 920., 1085.] ) # 1357 1085 920 1085 (Kg/m3)
 3 c_t = array([1700., 3800., 2300., 3680.]) # 1700 3800 2300 3680 (J/(Kg.C))
 4
 5 transcoeff = CellVariable(name="Transient Term Coefficient",
 6                          mesh=mesh,
 7                          value=rho_t*c_t)
 8 print transcoeff
 9
10 #Definition of the diffusion term coefficient - thermal conductivity of the
       tissues
11 k_t = CellVariable(name="Difusion Term Coefficient",
12                    mesh=mesh,
13                    value=(0.75, 0.51, 0.21, 0.47)) # 0.75 0.51 0.21 0.47 (W
       /(m.C))
14
15 #Defining Source Term - Tissue Blood Flow, specific weight and specific heat
        of the blood
16 V_sv = array([0., 483., 398.7, 362.]) # 0. 483. 398.7 362. (cm3/(m3.s))
17 rho_bl = 1059. # 1059 de Ferreira e Yanihara, original from Werner and Buse(
       Kg/m3)
18 c_bl = 3850. # 3850 the same source of rho_bl variable (J/(Kg.C))
19 T_ar = 37. # Arterial blood temperature (C)
20 q_t = array([0., 501., 4., 368.]) # 0. 501. 4. 368 (W/m3) - Endogenous heat
       production of each tissue layer
21
22 sorcoeff=CellVariable(name="Source Term Coefficient",
23                        mesh=mesh,
24                        value=V_sv*rho_bl*c_bl*10**-6)
25 print sorcoeff
```

```
26
27 indsourceterm=CellVariable(name="Internal Heat Production",
28                            mesh=mesh,
29                            value=V_sv*rho_bl*c_bl*T_ar*10**-6 + q_t)
30 print indsourceterm
```

Listing 5.3: Term coefficients and respective constants and variables.

The relations between variables are established according to equation 5.1. The constants `rho_t` and `c_t` are used to calculate transient term coefficient `transcoeff` (listing 5.3, ln 5-7) and the diffusion coefficient is `k_t` (listing 5.3, ln 11-13). The source coefficient is divided in; dependent source term, `sourcecoeff` (listing 5.3, ln 22-24), calculated by `rho_bl`, `c_bl` and `V_sv` (the source term that depends on the dependent variable); and the independent source term `indsourceterm` (listing 5.3, ln 27-29) that includes the additional effect of `T_ar` and `q_t` on the equation. All source terms defined are `CellVariable` objects. This is because equation that governs the heat diffusion change their coefficients all over the mesh, depending on the thermal properties of the physical volumes. Once again, by using `CellVariable` object, the values can be settled cell by cell to fit the needs of the simulation process. In this first module is simple to understand how does it works. The mesh is composed by 4 cells, each one corresponding to a different type of organic tissue, so, each value matching the coefficient that need to be used in a specific control volume (CV).

All those objects have the same size and form of the solution domain because they are build using the `mesh` variable (listing 5.1, ln 7) when setting the `mesh` parameter (listing 5.3, Ln 6, 12, 23, 28).

The equation is stored in a variable `eqX`, that relates the several terms of the equation and respective coefficients, and is used with an attribute function `solve` inside a iterative cycle (listing 5.5, ln 27-36) that calculates the solution (listing 5.5, ln 31-32) over a predefined time step (listing 5.5, ln 5). However, as discussed in previous chapter, it is needed to define boundary conditions in order to reach a stable solution.

```
valueRight = array([35., 36., 37., 37., 37., 37., 37., 37., 37., 37., 37.,
    37.,
                    37., 37., 37., 37., 38., 38., 38., 38., 38., 38., 38.,
    38., 38., 38., 38., 38., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38.,
    38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    37., 37., 37., 37., 39., 39., 40., 40.])# this represents skin temperature changing over the time - 5s
```

Listing 5.4: Boundary condition values.

This is done by using attribute function `constrain` applied to the `T_t` variable, setting the value at the respective place in the mesh. This program allows to set constant or variable boundary conditions over the time. The example (listing 5.4) shows an array list of values, named `valueRight`, that intends to simulate the transient state boundary conditions that constrain the `T_t` variable. Each value can be interpreted as an instant value of temperature that represents the temperature over the time step. In brief, it is one value for each calculation step (180 steps in the example mean 180 values of temperature in the array). That is why this is done during the iteration, inside the `for ... in ... :` cycle (listing 5.5, ln 27-36).

```
 1  eqX = TransientTerm(coeff=transcoeff) == DiffusionTerm(coeff=k_t)
 2        - ImplicitSourceTerm(coeff=sorcoeff) + indsourceterm
 3
 4  #timeStepDuration = 0.9 * 5.82**2 / (2 * (k_t)) -  5.82 is the ray of the cylinder
 5  timeStepDuration=5.  #s
 6  steps=180  #3600s/20s
 7
 8  T_tss = CellVariable(name="SS Neck Temperatures Distribution",
 9                     mesh=mesh,
10                     value=(36.5,36.,34.,33.))
11
12  T_tss.constrain(valueRight[0], (mesh.exteriorFaces & mesh.facesRight))
13  #T_tss.constrain(valueLeft, mesh.facesLeft)
14
15
16  if __name__ == '__main__':
17       viewer = Viewer(vars=(T_t,T_tss),
18                       datamin=32., datamax=40.)
19       viewer.axes.set_xlabel('r (m)')
20       viewer.axes.set_ylabel('Temp. ($\circ$C)')
```

```

    viewer.plot()

if __name__ == '__main__':
    raw_input("Viewer created. Press <enter> to proceed...")

for step in range(steps):

    T_t.constrain(valueRight[step], (mesh.exteriorFaces & mesh.facesRight))

    eqX.solve(var=T_t,
              dt=timeStepDuration)

    if __name__ == '__main__':
        viewer.plot()
        print T_t

if __name__ == '__main__':
    raw_input("Transient Diffusion. Press <enter> to proceed...")

eqxss= 0 == DiffusionTerm(coeff=k_t) - ImplicitSourceTerm(coeff=sorcoeff) +
    indsourceterm
eqxss.solve(var=T_tss)

figname = raw_input("Insert Figure's name...")

viewer.plot(filename = figname)
```

Listing 5.5: Equation storage, solving and plotting.

Total simulation period can be calculated through the `timeStep Duration` and the `step` variable. Time of the step is a very important parameter for numerically solving the equations. Use explicit formulation might lead to inaccurate results that suffer unrealistic oscillations. The implicit formulation used in this example is unconditionally stable for any size of time step, however too large steps still compromise the accuracy of the results (Versteeg and Malalasekera, 2007). So, small time steps still be preferable. To define a small time steps it is used the condition imposed by the explicit formulation of the discretization in one dimension. For a constant values of thermal coefficient and uniform grid spacing this resumes to the equation 5.20.

$$\Delta t < \rho c \frac{(\Delta x)^2}{2k} \tag{5.20}$$

The minimum value of Δt is reached for high values of k and small values of Δx, ρ and c corresponds to the maximum value accepted in explicit formulation. In this case the value should not exceed 20 seconds. For accurate results a 5 seconds time step was used. For 180 steps simulates about 15 minutes of heat diffusion (listing 5.5, ln 5-6). Iterative process of calculus can be divided in three stages: 1 - establish the boundary conditions (listing 5.5, ln 29); 2 - solve the equation (listing 5.5, ln 31-32); and 3 - plot and print the results (listing 5.5, ln 35-36). Every cycle of the iteration consists in updating boundary condition values, solve the equation for the new values (initial and boundary), and update the viewer by plotting the results.

Present example also solves the bioheat equation in steady state formulation (`eqxss`). To plot both results it was also created another tissue temperature cell variable (`T_tss`). As the steady state diffusion does not depends on time, it does not need to be calculated by an iterative process and that is why the equation is defined and solved after getting the solution of the heat diffusion at transient state in transient environment (listing 5.5, ln 41-42).

Extra code is used to print values in the prompt / python shell, or pause the program in order to give extra informations about the stage of calculus, or even ask file name and save the plot figures. So, they are not directly linked to the main goal of the simulation. This simple code intend to assure the correct implementation of Penne's Bioheat equation for several layers of tissue and considering transient conditions and test the scheme of the steps of the program.

5.2.2 Three-dimensional Heat Diffusion - Annotated listing

The code developed for 3D diffusion was based on unidimensional diffusion. The biggest improvement was adapt the code assuring that the coefficients of the bioheat equation change according to the thermal properties of tissue layers. At first, code for constant boundary conditions was tested, only then upgraded for responding to transient-states.

```
from fipy import *
from numpy import array
from viewers.update import update3D
from viewers.viewer3D import my_grid, view
```

Listing 5.6: Importing routines/function for 3D heat diffusion code.

The first few lines import the routines that are necessary to execute the program. Some of these routines were developed specifically to manage the visualization, plotting and data update of the variables (listing 5.6, ln 3-4). So, modularization started here, due to the need for particular structure of the data that allowed to create animated views of the data over calculation period. First it was created `my_grid` function, that belong to the viewer3D file system dedicated to reorganize the data of the `T_t` into a readable matrix to plot data using Mayavi functions. The `view` create the viewer, where is defined window size, quality of the printing image, data ranges, etc., and read the data returned by `my_grid`.

```python
# Import Mesh
mesh = Gmsh3D('geometries/neck.geo')
```

Listing 5.7: Creating variable mesh by importing a file with the geometry.

A very important step it was creating a three-dimensional solution domain by defining a mesh with the same characteristics of the individual that is being simulated. As mentioned in paragraph 5.2.3 these structures were created by program Gmsh and are programmed allowing easy change in size and cell dimensions. This occurs by the same order stated in the unidimensional diffusion. The `mesh` used now is created by the function `Gmsh3D`, a 3D mesh creator from Fipy, by importing a geometric structure (`neck.geo`) programmed in Gmsh , is similar to other geometric element that compose the body in final BioHeatSIM program (listing 5.7).

```python
# Define Tissue variable
T_t = CellVariable(name="Neck Temperatures Distribution",
                    mesh=mesh,
                    value=0.)

# Set initial Condition for each layer
T_t.setValue(36.5, where=mesh.physicalCells['Bone'])
T_t.setValue(36., where=mesh.physicalCells['Muscle'])
T_t.setValue(34., where=mesh.physicalCells['Fat'])
T_t.setValue(33., where=mesh.physicalCells['Skin'])

# Set boundary conditions

sidecells = ~(mesh.facesBack | mesh.facesFront) & mesh.exteriorFaces

T_t.constrain(40., where=sidecells)
```

Listing 5.8: Creating cell variable for temperatures distribution.

Variable `T_t` that relates the tissue temperature with the spatial position is created with `CellVariable` function but, in opposite to 1D code, initial conditions are not settled within the function fields in separate code. Initial conditions and boundary conditions are settled by attribute function of `CellVariable`, respectively `setValue` and `constrain`. In both functions it is needed to set the numeric value of temperature and the spatial position where those temperature values must be allocated.

Listing 5.27 shows in `setValue` function how useful can be the use of physical volumes in mesh construction. When building the mesh, the definition of the physical volumes allow here to manipulate that particular volume, or group of cells, independently. In the present example different values of temperature are settled to each tissue layer.

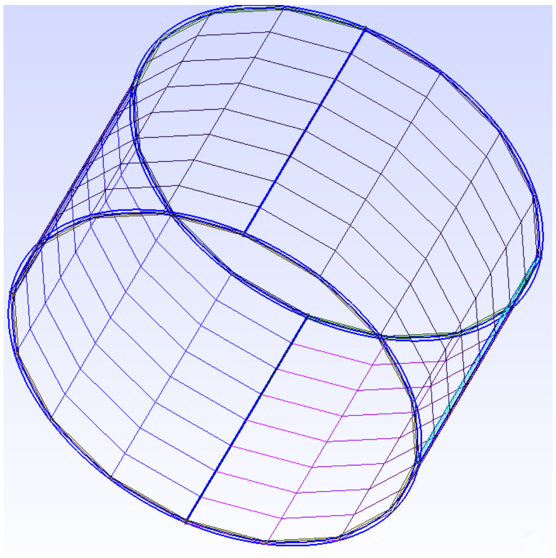


Figure 5.4: Faces of the mesh where is applied the boundary condition.

It is important to underline the fact that variable handling, as FiPy tool enables, is a strength of this modelling process. There are a wide range of combination that assure infinite ways to set de desired boundary or initials. For example, set values according to distance along one axis, set values to specific group of cells, relate it with the cylinder radius, or combine any of this preferences. The same happens with boundaries defined with `constrain`. The `where` field in `constrain` uses the same scheme as `where` in `setValue` what mean that the structure and combination used in one can be used in the other. For example, different surfaces can be submitted to different temperatures values or even to patterns of temperatures (by using matrices). These constrictions can be made inside, outside, or in any other part of the structure.

In listing 5.27 (ln 14) a variable `sidecells` define the exterior faces of the cylinder as the place to be constricted. The code excludes the top and the bottom surfaces from the exterior faces of the cylindrical mesh leaving exterior faces of the skin layer as the boundary (figure 5.4) and assure that the cell faces are at a constant temperature of $40\circ^{C}$ (listing 5.27, ln 16). As shown in 1D heat diffusion, the updating of boundaries at each step of calculus, is a reliable approach to simulate transient states.

```
# Define terms for the equation and set the coefficients to each layer

#Set the values for specific heat and mass of each tissue layer to define
    transient term coefficient

rho_t=array([1357., 1085., 920., 1085.])# 1357 1085 920 1085 (Kg/m3)
c_t=array([1700., 3800., 2300., 3680.]) # 1700 3800 2300 3680 (J/(Kg.oC))

transcoeff=CellVariable(mesh=mesh, value=0.)

# Set values for thermal conductivity of tissues to define diffusion
    coefficient term

k_t =array([0.75, 0.51, 0.21, 0.47]) # 0.75 0.51 0.21 0.47 (W/(m.oC))

diffcoeff=CellVariable(mesh=mesh, value=0.)

#Set values of tissue blood flow, specific mass and heat of the blood
V_sv=array([0., 483., 398.7, 362.]) # 0. 483. 398.7 362. (cm3/(m3.s))

rho_bl = 1059. # 1059 (Kg/m3)
c_bl = 3850. # 3850 (J/(Kg.oC))

T_ar= 37. # Arterial blood temperature values (oC)

q_t = array([0., 501., 4., 368.]) # 0. 501. 4. 368 (W/m3)/ basal heat
    production for tissue layers

sourcecoeff=CellVariable(mesh=mesh, value=0.)

indsourceterm = CellVariable(mesh=mesh, value=0.)

for i in range(4):
    if i==0 : string='Bone'
    elif i==1 : string='Muscle'
    elif i==2 : string='Fat'
```

```
34     elif i==3 : string='Skin'
35
36     transcoeff.setValue(rho_t[i]*c_t[i], where=mesh.physicalCells[string])
37
38     diffcoeff.setValue(k_t[i] , where=mesh.physicalCells[string])
39
40     sourcecoeff.setValue(V_sv[i]*rho_bl*c_bl*10**-6, where=mesh.
       physicalCells[string])
41
42     indsourceterm.setValue(V_sv[i]*rho_bl*c_bl*T_ar*10**-6 + q_t[i], where=
       mesh.physicalCells[string])
```

Listing 5.9: Defining bioheat equation terms and set values according to tissue layers.

The type of coefficients being used in 3D were defined as `CellVariable`. This type variable recognizes the space domain and let allocate the values of thermal properties at the correct place. Once again, by specifying the physical volumes using a iterative cycle, it is possible to place thermal conductivity, specific weight and specific heat of the layers bone, muscle, fat and skin at their exact place (listing 5.9, ln 30-42). One of the advantages implementing the bioheat equation with this tool is that it can be created a continuum heat flux through the tissue layers with complete different properties in a single step of calculus (listing 5.10, ln 12-19).

```
 1 # Bioheat transfer equation
 2 eqX = TransientTerm(coeff=transcoeff) == DiffusionTerm(coeff=diffcoeff) -
       ImplicitSourceTerm(coeff=sourcecoeff) + indsourceterm
 3
 4 #Solution obtained for step by step
 5 timeStepDuration=20.
 6 steps=180
 7
 8 dataset=my_grid(T_t)
 9 viewer=view(dataset)
10
11 for step in range(steps):
12
13      eqX.solve(var=T_t,
14                dt=timeStepDuration)
15
16      t = timeStepDuration*step + 10
17      update3D(dataset, T_t)
18      print T_t, 't=', t,'s'
```

Listing 5.10: Calculation and visualization of the data.

After defining `eqX` equation the same procedure of 1D diffusion is followed with the slightly differences of the new variable created t, and the management of three-dimensional dataset. So, first the definition of time step, then the number of steps, finally it is created the viewer. In the `for...in:` cycle new temperature values are calculated and updated in viewer at each step. The result is visualization of the heat diffusion process along the layers of the neck.

The algorithm also displays in prompt the matrices of tissue temperatures and respective time of simulation. Uni-dimensional code helped in testing the command sequence and helped structured the data management over calculation and plotting by using Fipy, NumPy, and SciPy tools. It also demonstrates some potential and limitation of the predefined functions of Fipy when integrate Mayavi for 3D plotting. To overtake this were developed special function that were tested in the 3D code. This implement the bioheat diffusion in a 4 layer structure with different thermal properties, allowing dataset management and visualization with step by step updating.

At this stage program changed to modular structure in order to provide flexibility, improve performance and make it easier to develop.

5.2.3 *BioHeatSIM* Program Modules - Annotated Listings

As mentioned above the *BioHeatSIM* program is organized by folders according to their role. Each folder have one or more files where can be found specific functions used to fulfil those roles. Figure 5.5 shows a flowchart of information across the folders. To integrate all the features there is a **main** file in the root of the program folder that is used to manage the modules and run the BioHeatSIM from a terminal or python/ipython shell.

In brief, the code from main file picks information from *geometries*, create the meshes and then they are used to declare variables using *variables* folder. After variables being declared, initial conditions and boundary conditions are settled using functions inside the respective folders. The most important folder of all is the *solvers* folder. Functions programmed in each file of *solvers* dedicate themselves to implement the mathematical functions that characterize the BioHeatSIM model. Data is saved in `data` folder and then visualized throughout the functions inside `view` folder. The **main.py** file calls solvers, save outputs and calls viewers functions to recreate and update the dataset to plot the results. So, sequence of calculus is assured by *main* being the inputs and

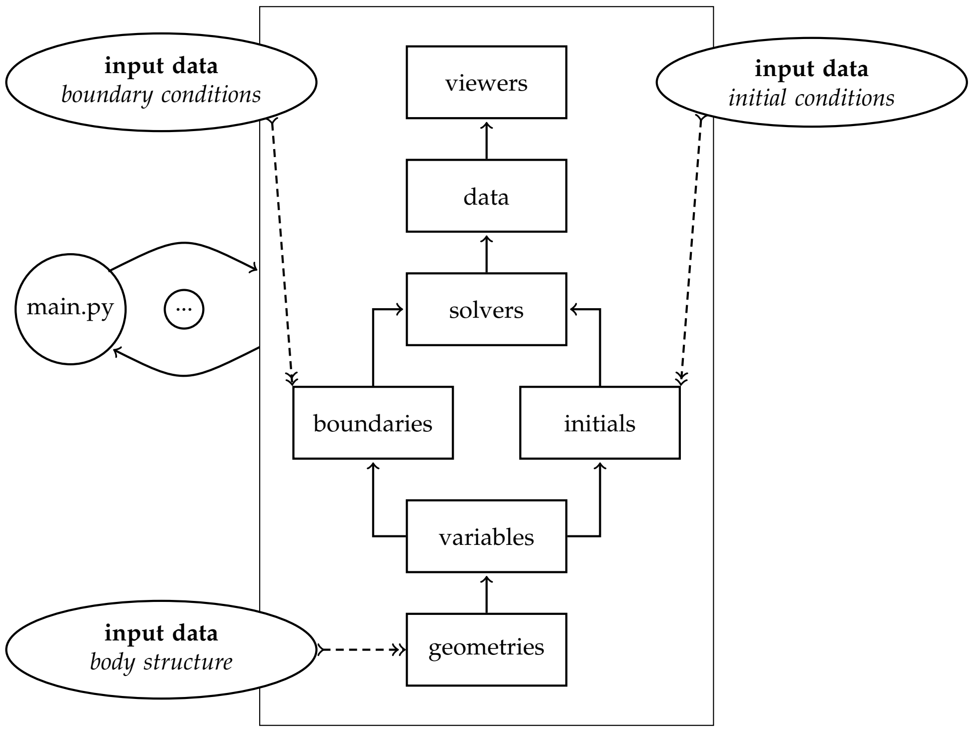


Figure 5.5: BioHeatSIM folder organization and information flow.

outputs of every stage conducted by it. Details about the code and implementation of the calculus procedures are presented in the following subsections.

Geometries

The physical structure of the human body that represents the domain of interest was drawing nearer simple geometric elements. This was not because program limitations of GMSH, but due to simplicity to building and manage the mesh parameters during the solving and plotting processes. Meshing process based itself in dividing the body into segments which geometry is reproducible by a single kind of geometry. Taking into account the accuracy level of the results, both parameters, the dimension and shape of the mesh, tried to reproduce the very own size and shape of the real body but using, whenever possible, regular structures to reach orthogonal meshes. This point in particular is important because the FiPy's accuracy depends on having a regular mesh because discretized process of Fipy is based on the orthogonality of the mesh.

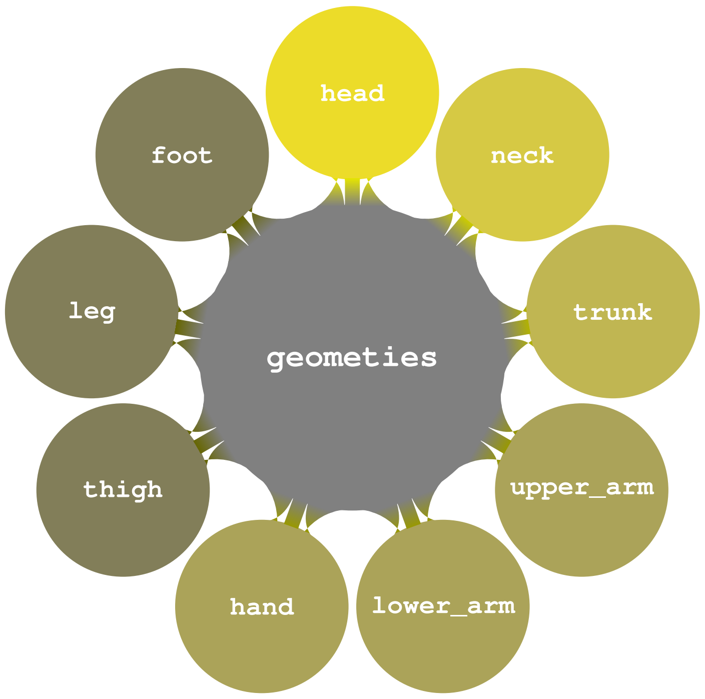


Figure 5.6: Geometries folder content - geometries that constitute the solution domain.

To attend individual characteristics of the body, meshing is based on variables that allow to change body segment length, width or radius and even the size of the control volumes (listing 5.11, ln 3-8). This feature is important not only because allow to adapt the simulation to particular physical structures, in order to get feasible results, but also because the density of the mesh (dictated by cells size) is an important way to adjust the compromise of accuracy and performance when simulating. Mesh parameters can be adjusted individually or all at once without compromising the speed of calculus.

```
 1 // ======================================================================//
 2 //                               Mesh Data                               //
 3 // ======================================================================//
 4 
 5 Size = .01;
 6 
 7 rbn  = 0.0268; // Bone (m)
 8 rmsc = 0.0540; // Muscle (m)
 9 rf   = 0.0571; // Fat (m)
10 rsk  = 0.0582; // Skin (m)
11 
12 z    = 0.080; // Neck length (m)
13 
14 // Divisions of the mesh
15 
```

```
NDc    = 5;      // Div. circ.

NDlbn  = 4;      // Div. Lines Bone
NDlmsc = 5;      // Div. Lines Muscle
NDlf   = 3;      // Div. Lines Fat
NDlsk  = 3;      // Div. Lines Skin

NL = 8; // Number of Layers
```

Listing 5.11: Code lines that declare variables to define the mesh.

All the geometric segments are written in separate folders (figure 5.6). Each folder contain: a header where variables are defined; a body where come the code for building the structure; and a footer where the physical volumes are defined. The following code is an example of the cylindrical mesh that describes the neck segment. The header describes the parameter of the mesh, that are linked to individual characteristics of the individual, where the main adjustments are done by changing the element length and the ray, or thickness, of tissue layers.

```
// ======================================================================//
//                                 Points                                //
// ======================================================================//

Point(1)={0., 0., 0., Size}; // Centre of cylinder base

// Points Circumference Bone rbn
Point(2)={0.,rbn,0.,Size};
Point(3)={rbn,0.,0.,Size};
Point(4)={0.,-rbn,0.,Size};
Point(5)={-rbn,0.,0.,Size};

// Points Circumference Fat rmsc
Point(6)={0.,rmsc,0.,Size};
Point(7)={rmsc,0.,0.,Size};
Point(8)={0.,-rmsc,0.,Size};
Point(9)={-rmsc,0.,0.,Size};

// Points Circumference Muscle rf
Point(10)={0.,rf,0.,Size};
Point(11)={rf,0.,0.,Size};
Point(12)={0.,-rf,0.,Size};
Point(13)={-rf,0.,0.,Size};

// Points Circumference Skin rsk
Point(14)={0.,rsk,0.,Size};
Point(15)={rsk,0.,0.,Size};
Point(16)={0.,-rsk,0.,Size};
Point(17)={-rsk,0.,0.,Size};
```

Listing 5.12: Settling the points to create arcs and lines.

To built the grid the steps are almost the same from peace to peace. First are defined the points (listing 5.12) that are going to be the vertices of surfaces to extrude. The lines between the points designing the edges of the surfaces (listing 5.13, **??**).

Cell size is defined but this is only used when Gmsh needs to built the structured grid using internal routines. However tetrahedral grids, built by Gmsh, are not the best option to model heat diffusion through body. In order to achieve the desired orthogonal grid, with regular elements, it was necessary to set the number of divisions for each segment and define the grid construction on the program.

```
// ======================================================================//
//                           Circumferences                              //
// ======================================================================//

//Circumference Arcs Bone
Circle( 1)={2,1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3,1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4,1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5,1,2}; Transfinite Line{ 4}=NDc;

//Circumference Arcs  Muscle
Circle( 5)={6,1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7,1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8,1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9,1,6}; Transfinite Line{ 8}=NDc;

//Circumference Arcs Fat
Circle( 9)={10,1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11,1,12}; Transfinite Line{10}=NDc;
Circle(11)={12,1,13}; Transfinite Line{11}=NDc;
Circle(12)={13,1,10}; Transfinite Line{12}=NDc;

//Circumference Arcs Skin
Circle(13)={14,1,15}; Transfinite Line{13}=NDc;
Circle(14)={15,1,16}; Transfinite Line{14}=NDc;
Circle(15)={16,1,17}; Transfinite Line{15}=NDc;
Circle(16)={17,1,14}; Transfinite Line{16}=NDc;
```

Listing 5.13: Code lines that create the arcs of the mesh.

```
// ======================================================================//
//                    Dividing Circumferences in 4                       //
// ======================================================================//

// Lines Bone
Line(21)={1,2}; Transfinite Line{21}=NDlbn Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDlbn Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDlbn Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDlbn Using Progression 0.5 ;

```

```
//Lines Muscle
Line(25)={2,6}; Transfinite Line{25}=NDlmsc;
Line(26)={3,7}; Transfinite Line{26}=NDlmsc;
Line(27)={4,8}; Transfinite Line{27}=NDlmsc;
Line(28)={5,9}; Transfinite Line{28}=NDlmsc;

// Lines Fat
Line(29)={6,10}; Transfinite Line{29}=NDlf Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlf Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlf Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlf Using Progression 0.8 ;

//Lines Skin
Line(33)={10,14}; Transfinite Line{33}=NDlsk;
Line(34)={11,15}; Transfinite Line{34}=NDlsk;
Line(35)={12,16}; Transfinite Line{35}=NDlsk;
Line(36)={13,17}; Transfinite Line{36}=NDlsk;
```

Listing 5.14: Creating the lines that divide the concentric circumferences in quarters.

When lines are created the number of divisions can be programmed by using `Transfinite Line` command. `Progression` can be used to divide the segments using a gradient. Final results are having shorter layers at the bounds of physical layers.

```
// ====================================================================//
//                     1/4 Circumf. Construction                       //
// ====================================================================//

//Close the Loops Bone
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Close the Loops Muscle
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Close the Loops Fat
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};

//Close the Loops Skin
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};
```

Listing 5.15: Closing lines as loops to built surfaces

```
// ====================================================================//
//                    1/4 Circle Surfaces Construction                  //
// ====================================================================//

// Defining Surfaces Bone
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Defining Surfaces Muscle
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Defining Surfaces Fat
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Defining Surfaces Skin
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};
```

Listing 5.16: Creating the surfaces and divisions for the mesh.

By closing lines with `Line Loop`, perimeter of the surfaces are created and plains are delimited by that perimeter (listing 5.15). To assume the transfinite division of the lines and create the respective grid inside the perimeter, `Transfinite Surface` is created (listing 5.16). Finally `Recombine Surface` joins the grid to plain surface. This allow to extrude the plain and respective grid building, at the same time, the three dimensional geometry and grid structure (listing 5.17).

```
// ======================================================================//
//                           Volume Construction                         //
// ======================================================================//

Extrude{0.0,0.0,z} {Surface{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}; Layers{
    NL}; Recombine;}
```

Listing 5.17: Surface extrusion to built volumes.

```
// ======================================================================//
//                       Physical  Volume Construction                   //
// ======================================================================//

Physical Volume("Bone") = {1,2,3,4};
Physical Volume("Muscle") = {5,6,7,8};
Physical Volume("Fat") = {9,10,11,12};
Physical Volume("Skin") = {13,14,15,16};

Show ''*''
```

Listing 5.18: Definitions of physical volumes.

The last step after extrusion it is the creation of physical volumes, defined according to the type of tissue that they represent (listing 5.18).

Initials

This folder contain the values of initial state of the system (`T_t_init, T_ve_init, T_ar_init`) (listing 5.19), constants of thermophysical properties of the physical system (`rho_t, c_t, k_t, rho_bl, c_bl, H_av`)(listing 5.20), and variables at thermoneutral basal state variables (`CO, V_sv` (listing 5.21), `q_t` (listing 5.22), `Vol_ar, Vol_ve` (listing 5.23)). Values are saved in array variables and exported to main file through simple with no argument functions.

```
def import_init_temp():
    T_ar_init = array([37.,37., #hands r-l
                        37.,37., #lower arms r-l
                        37.,37., #upper arms r-l
                        37.,37., #feet r-l
                        37.,37., #legs r-l
                        37.,37., #thighs r-l
                        37.,  # trunk
                        37.,  # neck
                        37.]) # head
    T_ve_init = array([37.,37., #hands r-l
                    37.,37., #lower arms r-l
                    37.,37., #upper arms r-l
                    37.,37., #feet r-l
                    37.,37., #legs r-l
                    37.,37., #thighs r-l
                    37.,  # trunk
                    37.,  # neck
                    37.]) # head
    T_t_init = array([[36.5, 36., 34., 33.],[36.5, 36., 34., 33.], #hands r-l
                    [36.5, 36., 34., 33.],[36.5, 36., 34., 33.], #lower arms r-l
                    [36.5, 36., 34., 33.],[36.5, 36., 34., 33.], #upper arms r-l
                    [36.5, 36., 34., 33.],[36.5, 36., 34., 33.], #feet r-l
                    [36.5, 36., 34., 33.],[36.5, 36., 34., 33.], #legs r-l
                    [36.5, 36., 34., 33.],[36.5, 36., 34., 33.], #thighs r-l
                    [36.5, 36., 34., 33., 36., 37., 37.,37], # trunk
                    [36.5, 36., 34., 33.], # neck
                    [37., 37., 35., 33., 37.]]) # head

    return T_ar_init, T_ve_init, T_t_init
```

Listing 5.19: Exports the initial temperatures values.

```
def import_constants():
    #Thermal Properties of the tissues
    rho_t = array([[1357., 1085., 920., 1085.],[1357., 1085., 920., 1085.], #hands r-l
                [1357., 1085., 920., 1085.],[1357., 1085., 920., 1085.], #lower arms r-l
                [1357., 1085., 920., 1085.],[1357., 1085., 920., 1085.], #upper arms r-l
                [1357., 1085., 920., 1085.],[1357., 1085., 920., 1085.], #feet r-l
                [1357., 1085., 920., 1085.],[1357., 1085., 920., 1085.], #legs r-l
                [1357., 1085., 920., 1085.],[1357., 1085., 920., 1085.], #thighs r-l
                [1357., 1085., 920., 1085., 1080., 1080., 560.], # trunk
                [1357., 1085., 920., 1085.], # neck
                [1357., 1085., 920., 1085., 1080.]]) # head
    c_t = array([[1700.,3800.,2300.,3680.],[1700.,3800.,2300.,3680.], #hands r-l
                [1700.,3800.,2300.,3680.],[1700.,3800.,2300.,3680.], #lower arms r-l
                [1700.,3800.,2300.,3680.],[1700.,3800.,2300.,3680.], #upper arms r-l
                [1700.,3800.,2300.,3680.],[1700.,3800.,2300.,3680.], #feet r-l
                [1700.,3800.,2300.,3680.],[1700.,3800.,2300.,3680.], #legs r-l
                [1700.,3800.,2300.,3680.],[1700.,3800.,2300.,3680.], #thighs r-l
                [1700.,3800.,2300.,3680., 3550.,3504.,  3520.], # trunk
                [1700.,3800.,2300.,3680.], # neck
                [1700.,3800.,2300.,3680., 3580.]]) # head
    k_t = array([[0.75,0.51,0.21,0.47],[0.75,0.51,0.21,0.47], #hands r-l
                [0.75,0.51,0.21,0.47],[0.75,0.51,0.21,0.47], #lower arms r-l
                [0.75,0.51,0.21,0.47],[0.75,0.51,0.21,0.47], #upper arms r-l
                [0.75,0.51,0.21,0.47],[0.75,0.51,0.21,0.47], #feet r-l
                [0.75,0.51,0.21,0.47],[0.75,0.51,0.21,0.47], #legs r-l
                [0.75,0.51,0.21,0.47],[0.75,0.51,0.21,0.47], #thighs r-l
                [0.75,0.51,0.21,0.47,0.47,0.49,0.28], # trunk
                [0.75,0.51,0.21,0.47], # neck
                [0.75,0.51,0.21,0.47,0.49]]) # head
    # Coeficientes for arterio-venous heat transfer
    H_av = array([0.72,0.72, #hands r-l
                1.43,1.43, #lower arms r-l
                2.55,2.55, #upper arms r-l
                1.55,1.55, #feet r-l
                3.67,3.67, #legs r-l
                8.26,8.26, #thighs r-l
                0.00,  # trunk
                1.55,  # neck
                0.00]) # head
    rho_bl = 1059. # 1059 (Kg/m3)
    c_bl = 3850. # 3850 (J/(Kg.oC))
    return rho_t, c_t, k_t, H_av, rho_bl, c_bl
```

Listing 5.20: Imports the constants.

```
def import_V_basal():
    CO = 80.56 #cm3.s-1
    Vol_lung = 2481.e-6 #m3
    V_sv_lung = CO/Vol_lung


    # Bn,msk,ft,sk,ln,hr,vsc,brn (cm3/(m3.s)) - basal state
    V_sv = array([[0., 483., 77., 362.,0.,0.,0.],[0., 483., 77.,
    362.,0.,0.,0.], #hands r-l
                 [0., 483., 77., 362.,0.,0.,0.],[0., 483., 77.,
    362.,0.,0.,0.], #lower arms r-l
                 [0., 483., 77., 362.,0.,0.,0.],[0., 483., 77.,
    362.,0.,0.,0.], #upper arms r-l
                 [0., 483., 77., 362.,0.,0.,0.],[0., 483., 77.,
    362.,0.,0.,0.], #feet r-l
                 [0., 483., 77., 362.,0.,0.,0.],[0., 483., 77.,
    362.,0.,0.,0.], #legs r-l
                 [0., 483., 77., 362.,0.,0.,0.],[0., 483., 77.,
    362.,0.,0.,0.], #thighs r-l
                 [0., 483., 77., 362., 14400., 8925., V_sv_lung], # trunk
                 [0., 483., 77., 362.,0.,0.,0.], # neck
                 [0., 483., 77., 362., 9000.,0.,0.]])*10**-6 # head
    return CO, V_sv
```

Listing 5.21: Calculated basal blood flow to the lung based on basal cardiac output and return tissues blood flow at basal thermoneutral state.

```
def import_q_t():
    # Bn,msk,ft,sk,ln,hr,vsc,brn (W/m3) - basal heat production of the
    tissues
    q_t = array([[0., 501., 4., 368.],[0., 501., 4., 368.], #hands r-l
                 [0., 501., 4., 368.],[0., 501., 4., 368.], #lower arms r-l
                 [0., 501., 4., 368.],[0., 501., 4., 368.], #upper arms r-l
                 [0., 501., 4., 368.],[0., 501., 4., 368.], #feet r-l
                 [0., 501., 4., 368.],[0., 501., 4., 368.], #legs r-l
                 [0., 501., 4., 368.],[0., 501., 4., 368.], #thighs r-l
                 [0., 501., 4., 368., 24128., 3852.,339.], # trunk
                 [0., 501., 4., 368.], # neck
                 [0., 501., 4., 368., 9472.]]) # head
    return q_t
```

Listing 5.22: Exports tissue heat production in basal state at thermoneutral conditions.

```
def import_vol_ar_ve():
    #Arterial and venous volumes of blood reservoirs at basal thermoneutras
    conditions
    Vol_ar = array([7.  ,7.  , #hands r-l
                    13.,13., #lower arms r-l
                    24.,24., #upper arms r-l
                    15.,15., #feet r-l
                    35.,35., #legs r-l
                    78.,78., #thighs r-l
                    446.,  # trunk
                    15.,  # neck
                    40.])*10**-6 # head
    Vol_ve = array([30.,30. , #hands r-l
                    60.,60., #lower arms r-l
                    107.,107., #upper arms r-l
                    66.,66., #feet r-l
                    155.,155., #legs r-l
                    349.,379., #thighs r-l
                    1484.,  # trunk
                    66.,  # neck
                    180.])*10**-6 # head
    return Vol_ar, Vol_ve
```

Listing 5.23: Exports arterial and venous volumes of the reservoirs at each segment.

Structure of the arrays consist is 15 lines composed by vector as long as the different types of tissues that compose the body segment. Different initial conditions can be settled to hands, feet, arms and legs elements in order to respond to asymmetry even during transient exposures.

Files of present folder also have functions to set the initial values to variables created. The `set_values_T_t` function sets the initial values of tissue temperature variables (listing 5.24).

```
def set_values_T_t(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
                  T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
                  T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
                  T_t_trunk,T_t_neck, T_t_head, T_t_init):

    # Initial conditions Right Arm
    Tthandr = T_t_init[0]
    Ttlowerarmr = T_t_init[2]
    Ttupperarmr = T_t_init[4]

    # Initial Conditions of the left Arm
    Tthandl = T_t_init[1]
    Ttlowerarml = T_t_init[3]
    Ttupperarml = T_t_init[5]

    #Initial Conditions of the Right Leg
    Ttfootr = T_t_init[6]
    Ttlegr = T_t_init[8]
    Ttthighr = T_t_init[10]

    #Initial Conditions of the Left Leg
    Ttfootl = T_t_init[7]
    Ttlegl = T_t_init[9]
    Ttthighl = T_t_init[11]

    #Initial Conditions of the Trunk
    Tttrunk = T_t_init[12]

    #Initial Conditions of the Neck
    Ttneck = T_t_init[13]

    # Intial Conditions Of the Head
    Tthead = T_t_init[14]

    for i in range(4):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'
        # Arms
        T_t_handr.setValue(Tthandr[i],
                           where=T_t_handr.mesh.physicalCells[string])
        T_t_lowerarmr.setValue(Ttlowerarmr[i],
                               where=T_t_lowerarmr.mesh.physicalCells[string
    ])
        T_t_upperarmr.setValue(Ttupperarmr[i],
                               where=T_t_upperarmr.mesh.physicalCells[string
    ])
        T_t_handl.setValue(Tthandl[i],
                           where=T_t_handl.mesh.physicalCells[string])
        T_t_lowerarml.setValue(Ttlowerarml[i],
                               where=T_t_lowerarml.mesh.physicalCells[string
    ])
        T_t_upperarml.setValue(Ttupperarml[i],
                               where=T_t_upperarml.mesh.physicalCells[string
    ])
        # Legs
        T_t_footr.setValue(Ttfootr[i],
                            where=T_t_footr.mesh.physicalCells[string])
        T_t_legr.setValue(Ttlegr[i],
                           where=T_t_legr.mesh.physicalCells[string])
        T_t_thighr.setValue(Ttthighr[i],
                             where=T_t_thighr.mesh.physicalCells[string])
        T_t_footl.setValue(Ttfootl[i],
                            where=T_t_footl.mesh.physicalCells[string])
        T_t_legl.setValue(Ttlegl[i],
                           where=T_t_legl.mesh.physicalCells[string])
        T_t_thighl.setValue(Ttthighl[i],
                             where=T_t_thighl.mesh.physicalCells[string])
        # Trunk
        T_t_trunk.setValue(Tttrunk[i],
                            where=T_t_trunk.mesh.physicalCells[string])
        # Neck
        T_t_neck.setValue(Ttneck[i],
                           where=T_t_neck.mesh.physicalCells[string])
        # Head
        T_t_head.setValue(Tthead[i], where=T_t_head.mesh.physicalCells[
    string])

    for i in range(3):
        if i==0 : string='Lung'
        elif i==1 : string='Heart'
        elif i==2 : string='Viscera'
        T_t_trunk.setValue(Tttrunk[i],
                            where=T_t_trunk.mesh.physicalCells[string])

    T_t_head.setValue(Tthead[4], where=T_t_head.mesh.physicalCells['Brain'])

    return (T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
    T_t_lowerarml,
            T_t_upperarml, T_t_footr, T_t_legr, T_t_thighr, T_t_footl,
            T_t_legl, T_t_thighl, T_t_trunk, T_t_neck, T_t_head)
```

Listing 5.24: Exports arterial and venous volumes of the reservoirs at each segment.

As input it need all the tissue temperature cell variable from the 15 body segments, and the array with initial temperatures. One limitation of the routine, as presented here, is that only allow to set one temperature per layer of tissue in one body segment. Different types of initial conditions must be programmed apart.

Other two functions of the program allow settling and creating vectors to store venous and arterial temperatures of reservoirs at each segments (listing 5.24).

```
def set_values_Tar(T_ar_init):
    tar_handr     = [T_ar_init[0]]
    tar_lowerarmr = [T_ar_init[1]]
    tar_upperarmr = [T_ar_init[2]]
    tar_handl     = [T_ar_init[3]]
    tar_lowerarml = [T_ar_init[4]]
    tar_upperarml = [T_ar_init[5]]
    tar_footr     = [T_ar_init[6]]
    tar_legr      = [T_ar_init[7]]
    tar_thighr    = [T_ar_init[8]]
    tar_footl     = [T_ar_init[9]]
    tar_legl      = [T_ar_init[10]]
    tar_thighl    = [T_ar_init[11]]
    tar_trunk     = [T_ar_init[12]]
    tar_neck      = [T_ar_init[13]]
    tar_head      = [T_ar_init[14]]
    return (tar_handr, tar_lowerarmr, tar_upperarmr, tar_handl,
            tar_lowerarml, tar_upperarml , tar_footr, tar_legr,
            tar_thighr, tar_footl, tar_legl,tar_thighl,tar_trunk,
            tar_neck, tar_head)

def set_values_Tve(T_ve_init):
    tve_handr     = [T_ve_init[0]]
    tve_lowerarmr = [T_ve_init[1]]
    tve_upperarmr = [T_ve_init[2]]
    tve_handl     = [T_ve_init[3]]
    tve_lowerarml = [T_ve_init[4]]
    tve_upperarml = [T_ve_init[5]]
    tve_footr     = [T_ve_init[6]]
    tve_legr      = [T_ve_init[7]]
    tve_thighr    = [T_ve_init[8]]
    tve_footl     = [T_ve_init[9]]
    tve_legl      = [T_ve_init[10]]
    tve_thighl    = [T_ve_init[11]]
    tve_trunk     = [T_ve_init[12]]
    tve_neck      = [T_ve_init[13]]
    tve_head      = [T_ve_init[14]]
    return (tve_handr, tve_lowerarmr, tve_upperarmr,
            tve_handl,tve_lowerarml,tve_upperarml, tve_footr, tve_legr,
            tve_thighr, tve_footl,tve_legl, tve_thighl, tve_trunk,
            tve_neck, tve_head)
```

Listing 5.25: Arterial and venous temperature arrays with settled initial conditions.

Variables

Variables folder have function to create tissue temperature variables and meshes.

Function `importmeshs()` exports meshes created with `Gmsh3D` function from Fipy, by using the respective Gmsh geometric code. Function has no inputs and as outputs gives meshes from the 9 different elements of the body. Notice that, in spite of initial values consider the 15 segments, distinguishing right from left elements, meshes based themselves in the human body symmetry, so that left limbs of the body are equal to right ones, reducing the number of meshes required (listing 5.26).

```
"""
The present code intend to setle the initial conditions of tissues
to the entire body.

Have the import mesh function, variable definition and the set value
    function.
"""


def set_values_T_t(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
                   T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
                   T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
                   T_t_trunk,T_t_neck, T_t_head, T_t_init):

    # Initial conditions Right Arm
    Tthandr = T_t_init[0]
    Ttlowerarmr = T_t_init[2]
    Ttupperarmr = T_t_init[4]

    # Initial Conditions of the left Arm
    Tthandl = T_t_init[1]
    Ttlowerarml = T_t_init[3]
    Ttupperarml = T_t_init[5]

    #Initial Conditions of the Right Leg
    Ttfootr = T_t_init[6]
    Ttlegr = T_t_init[8]
    Ttthighr = T_t_init[10]

    #Initial Conditions of the Left Leg
    Ttfootl = T_t_init[7]
    Ttlegl = T_t_init[9]
    Ttthighl = T_t_init[11]

    #Initial Conditions of the Trunk
    Tttrunk = T_t_init[12]

    #Initial Conditions of the Neck
    Ttneck = T_t_init[13]

    # Intial Conditions Of the Head
    Tthead = T_t_init[14]

    for i in range(4):
```

Listing 5.26: Definition of tissue temperature variables.

Tissue temperature variables are defined with function `vars_Tt()`. The function simply creates cell variables using Fipy functions. As input uses the meshes that describe the physical body, and as output returns the 15 variables that are going to be used to solve the equations of tissues and small vessels at each segment.

Solvers

As mentioned, this is the most important folder. The routines programmed here are dedicated to solve the mathematical expressions presented in the previous chapter. Figure 5.7 shows the files and functions inside this folder. The functions can be divided in 3 levels, being the higher level routines the ones that use or integrate lower level ones. This model consider three core or basic routines, two 2^{nd} level, and one top level, respectively:

- `T_t` - calculate the spatial distributions of temperatures;
- `func_t_ar` - estimates temperature of arterial blood pool in a segment;
- `func_t_ve` - gives the temperature of venous blood pool in a segment;
- `midd_body_elem` - coordinate the calculus process of a body segment with two connection (anterior and posterior);
- `vazao` - routine that gathers a set of functions to estimate the blood flow across large vessels of all body segments based on the required basal tissues blood flow;
- `head` - functions that coordinate the heat exchange between large vessels and small vessels and tissues of the head;
- `trunk` - manage core functions for the complex geometric structure and integrate results from 5 connections.

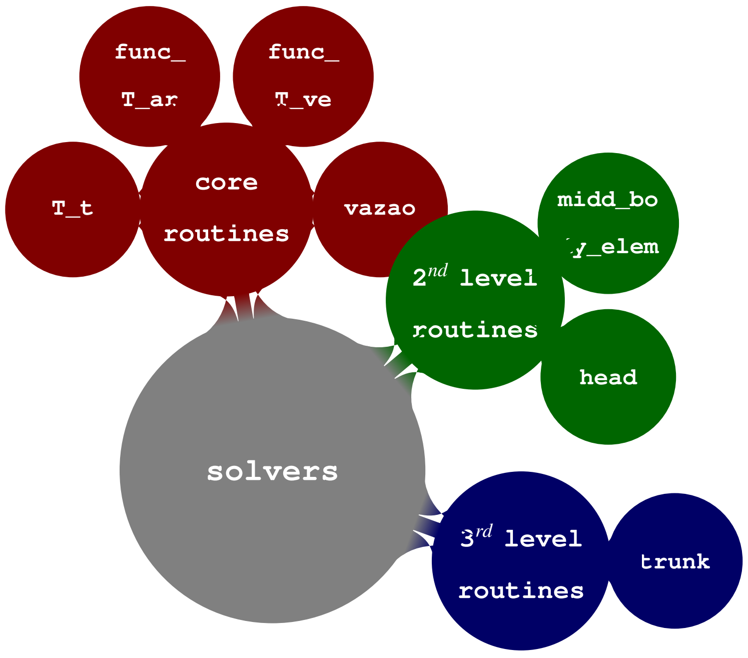


Figure 5.7: Passive system solvers.

They have a typical structure of a function, that admits a group of variable as an input, and return another group of variables, or the same variables "transformed".

Function `T_t` models tissue temperature following the structure of calculus that was previously tested. In other words, it solves Pennes' bioheat equation and returns the temperature distributed along the four tissue layers. Basically it uses the mesh of the body element, builds the terms (listing 5.27, ln 17, 22, 27, 29), sets the thermal properties of each tissue layer to the respective area of the mesh (listing 5.27, ln 33-45), built equation (listing 5.27, ln 47) , solves the bioheat equation according to the time step (listing 5.27, ln 49) and returns the matrices with the results (listing 5.27, ln 51).

```
 1 def T_t(mesh, T_t_init, T_ar, V_sv, q_t, rho_t,
 2         c_t, k_t, rho_bl, c_bl, timeStep):
 3 
 4     transcoeff=CellVariable(mesh=mesh, value=0.)
 5 
 6     diffcoeff=CellVariable(mesh=mesh, value=0.)
 7 
 8     sourcecoeff=CellVariable(mesh=mesh, value=0.)
 9 
10     indsourceterm = CellVariable(mesh=mesh, value=0.)
11 
12     T__t = T_t_init
13 
```

```
    for i in range(4):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'

        transcoeff.setValue(rho_t[i]*c_t[i], where=mesh.physicalCells[string
    ])

        diffcoeff.setValue(k_t[i] , where=mesh.physicalCells[string])

        sourcecoeff.setValue(V_sv[i]*rho_bl*c_bl,
                             where=mesh.physicalCells[string])

        indsourceterm.setValue(V_sv[i]*rho_bl*c_bl*T_ar + q_t[i],
                               where=mesh.physicalCells[string])

    eqX = TransientTerm(coeff=transcoeff) == (DiffusionTerm(coeff=diffcoeff)
    - ImplicitSourceTerm(coeff=sourcecoeff) + indsourceterm)

    eqX.solve(var=T__t, dt=timeStep)

    return T__t
```

Listing 5.27: Code of temperature diffusion through tissues and small vessels.

This function might be called and used as any other function of python. Programming structure allows to access the files and import respective functions, what confers to it a modular object oriented structure. To use it, it is only necessary to fulfil the variables as required in the code. To execute `T_t()`, is needed a mesh (from a body element), a cell variable with initial values of temperature in the tissue, arterial blood temperature at the moment of the perfusion, small vessels blood flow, endogenous heat production of each type of tissue, and a time step.

```
def T_t_head(mesh, T_t_init, T_ar, V_sv, q_t, rho_t,
             c_t, k_t, rho_bl, c_bl, timeStep):

    transcoeff=CellVariable(mesh=mesh, value=0.)

    diffcoeff=CellVariable(mesh=mesh, value=0.)

    sourcecoeff=CellVariable(mesh=mesh, value=0.)

    indsourceterm = CellVariable(mesh=mesh, value=0.)

    T__t = T_t_init

    for i in range(5):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'
        elif i==4 : string='Brain'

        transcoeff.setValue(rho_t[i]*c_t[i], where=mesh.physicalCells[string])

        diffcoeff.setValue(k_t[i] , where=mesh.physicalCells[string])

        sourcecoeff.setValue(V_sv[i]*rho_bl*c_bl,
                             where=mesh.physicalCells[string])

        indsourceterm.setValue(V_sv[i]*rho_bl*c_bl*T_ar + q_t[i],
                               where=mesh.physicalCells[string])

    eqX = TransientTerm(coeff=transcoeff) == (DiffusionTerm(coeff=diffcoeff)
    - ImplicitSourceTerm(coeff=sourcecoeff) + indsourceterm)

    eqX.solve(var=T__t, dt=timeStep)

    return T__t
```

Listing 5.28: Code of temperature diffusion through tissues and small vessels of the head.

As `head` and `trunk` have different tissue composition, two extra function were created. Function `T_t_head` is composed by 5 different types of tissues but variables remain the same from layer to layer and the structure of `T_t` function remains (listing 5.28.

In `T_t_trunk` (listing 5.29) it is necessary to redefine equation terms , in particular the `indsourceterm` (listing 5.29, ln 27), where tissue temperature depends on venous temperature (`T_ve`), instead of arterial temperature (`T_ar`), and respective venous flow to the lungs (that is stored at `V_ve` as CO per unit of volume).

```
 1 def T_t_trunk(mesh, T_t_init, T_ar, T_ve, V_sv, q_t, rho_t,
 2               c_t, k_t, rho_bl, c_bl, timeStep):
 3 
 4     transcoeff=CellVariable(mesh=mesh, value=0.)
 5 
 6     diffcoeff=CellVariable(mesh=mesh, value=0.)
 7 
 8     sourcecoeff=CellVariable(mesh=mesh, value=0.)
 9 
10     indsourceterm = CellVariable(mesh=mesh, value=0.)
11 
12     T__t = T_t_init
13 
14     for i in range(6):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'
        elif i==4 : string='Heart'
        elif i==5 : string='Viscera'

        transcoeff.setValue(rho_t[i]*c_t[i], where=mesh.physicalCells[string])

        diffcoeff.setValue(k_t[i] , where=mesh.physicalCells[string])

        sourcecoeff.setValue(V_sv[i]*rho_bl*c_bl,
                             where=mesh.physicalCells[string])

        indsourceterm.setValue(V_sv[i]*rho_bl*c_bl*T_ar + q_t[i],
                               where=mesh.physicalCells[string])

    transcoeff.setValue(rho_t[6]*c_t[6], where=mesh.physicalCells['Lung'])

    diffcoeff.setValue(k_t[6] , where=mesh.physicalCells['Lung'])

    sourcecoeff.setValue(V_sv[6]*rho_bl*c_bl, where=mesh.physicalCells['Lung'])

    indsourceterm.setValue(V_sv[6]*rho_bl*c_bl*T_ve + q_t[6],
                           where=mesh.physicalCells['Lung'])

    eqX = TransientTerm(coeff=transcoeff) == (DiffusionTerm(coeff=diffcoeff) -
                        ImplicitSourceTerm(coeff=sourcecoeff) + indsourceterm)

    eqX.solve(var=T__t, dt=timeStep)

    return T__t
```

Listing 5.29: Code of temperature diffusion through tissues and small vessels.

By looking at the variable used in each function, it becomes clear the order or calculus and function's dependency.

On the other hand, `T_ar` is dedicated to predict arterial temperature over the time. It needs the actual mean values of the arterial and venous reservoirs of the body segment, the quantity of blood that comes in, the volume of the arterial blood pool at the moment, a theoretical coefficient of convective heat exchange between arterial and venous reservoirs and finally, once again, the time step for the calculus.

```
def T_ar(T_ar_init, T_ar_in, T_ve_init, H_av, V_ar,
         Vol_ar, rho_bl, c_bl, timeStep):
```

```
 3
 4     def derivtar(T__ar,t): # return derivatives of the array T_ar
 5         return (V_ar*rho_bl*c_bl*(T_ar_in-T__ar[0])+
 6                 H_av*(T_ve_init-T__ar[0]))/(Vol_ar*rho_bl*c_bl)
 7
 8     time = linspace(0.0,timeStep,2)
 9
10     T__ar = odeint(derivtar,T_ar_init,time)
11
12     return T__ar[-1]
```

Listing 5.30: Function that predicts mean arterial temperature of a segments' pool.

This simple code creates a derivative function of arterial temperature (listing 5.30, ln 6-7) and uses a SciPy solver routine (`odeint`) to integrate the function in order to time, using initial conditions to achieve a single result (listing 5.30, ln 11). To allow synchronization, time is defined as a vector of size 2, beginning at `0` and ending at `timestep` (listing 5.30, ln 19). Outcome solution is defined by `T__ar` in pair with the vector `t`.

It is important to underline the fact that a complete human thermal model, as described in *model design* chapter, uses non-steady `H_av`, `V_ar` and `Vol_ar` variables. The artery volume `Vol_ar` comes from the length of the arterial branch (that depends on individual characteristics) and the radii. Arteries radii change over the time depending on *Cardiac Output* and biologic active response to the environment, as stated by Fu (1995), and implemented by Salloum et al. (2007) and Karaki et al. (2013). Cardiac output is calculated based on skin and core temperatures and the final values allow to calculate arteries radii. This belongs to cardiovascular response of large vessels modelling. Blood supply that goes from one element to another (represented by `V_ar`), depends directly on the cardiac output distribution and large artery radii that are input values in Avolio (1980) cardiovascular model that after impedances calculated allow to determine the velocity of blood along the arterial branches of the vascular net. Blood supply decrease or increase in arterio-venous anastomoses constriction and dilations, as skin blood flow, is predicted though the Avolio (1980) cardiovascular model, including the adaptations imposed by heat and exercise loads described by Rowell et al. (1965). However this part is included in small vessels blood supply defined in `T_t` function as `V_sv` variable.

The arterial temperature at the trunk does not depends on the same variable that the ones at limbs and head. To estimate this value `T_ar_trunk` was created(listing 5.31).

```
def T_ar_trunk(T_ar_init, T_t_trunk, Vazao_sv_lung,
               Vol_ar, rho_bl, c_bl, timeStep):

    T_lung_V_lung = sum(T_t_trunk*Vazao_sv_lung)
    V_ar = sum(Vazao_sv_lung)

    def derivtar(T__ar,t): # return derivatives of the array T_ar
        return (rho_bl*c_bl*(T_lung_V_lung-V_ar*T__ar[0]))/(Vol_ar*rho_bl*
    c_bl)

    time = linspace(0.0,timeStep,2)

    T__ar = odeint(derivtar,T_ar_init,time)

    return T__ar[-1]
```

Listing 5.31: Function that predicts mean arterial temperature of a trunk's pool.

At the trunk, blood flows from central venous reservoir to the lungs, became arterial blood and gets in arterial reservoir. This implies that final temperature of the pool depends on initial temperature, and temperature of the lung and respective blood flow. This works as temperature of the tissues. The array `vazao_sv_lung`, has the same size and shape of `T_t_trunk`, but all cell flows which do not belong to lungs are null. With the blood flow at each cell multiplied by the mean temperate at that cell volume, results a weighed temperature of all cells of the lung volume that depends on the size of the cell. As described by Ferreira and Yanagihara (2009) heat exchange between large vessels is not considered at the trunk nor the head.

Function `T_ve` (listing 5.33) predicts venous temperature. In its structure it is more complex than `func_T_ar`. It not only depends on initial value of venous blood temperature `T_ve_init`, venous blood temperature that comes in from the posterior element `T_ve_in`, volume of venous reservoir`Vol_ve`, venous blood flow that comes from the posterior element `V_ve_in` and heat transfer coefficient between arteries and veins for a segment `H_av`, but also depends on temperature (`T_t`) and flow of venous blood that come from the 4 layers of tissue (`Val_sv`). Remember that final temperature in the venous reservoir of a segment is a weighted result of the several amounts of blood and respective temperatures that come into the reservoir.

```python
def T_ve(T_ve_init, T_t, T_ar, T_ve_in, Vol_ve, V_ve_in, H_av, Vazao_sv,
         rho_bl, c_bl, timeStep):

    sum_Vsv_Tmt = sum(Vazao_sv*T_t.value)

    V_ve = V_ve_in + sum(Vazao_sv)

    def derivtve(T__ve,t): # return derivatives of the array T_ve
            return (rho_bl*c_bl*(sum_Vsv_Tmt+V_ve_in*T_ve_in-V_ve*T__ve[0])+
                    H_av*(T_ar-T__ve[0]))/(Vol_ve*rho_bl*c_bl)

    time = linspace(0.0,timeStep,2)

    T__ve = odeint(derivtve,T_ve_init,time)

    return T__ve[-1]
```

Listing 5.32: Function that predicts mean venous temperature of a segments' pool.

First it is imported the volume of the cells from the cell variable `T_t` (listing 5.33, ln 3). The calculation of the amount of blood is given by the product of the predicted tissue blood flow (`V_sv`) and the volumes of cells (`Cell_vol`). The created variable (`Vazao_sv`) is a matrix (with the same dimension and position code of the tissue temperature variable `T_t`) of values with the amount of blood that goes into venous reservoir from each cell compartment (listing 5.33, ln 17). Contribution of each cell to the final venous temperature `T__ve`, is given though the product of the amount of blood that come from each cell , by the respective mean temperature value of the cell. The relation between venous blood flow `V_ve` that exits a compartment, and venous blood flow that come in from a posterior segment `V_ve_in`, is necessary to solve the differential equation (4.6), and is here defined in listing 5.33, line 24. With constants and variables defined, the next step consists in solving the diferential equation by defining a function of the derivative of array `T__ve` (listing 5.33, ln 29-30), create the time vector for the needed time results (listing 5.33, ln 32), and solve the equation with `odeint` function, by giving the initial conditions (listing 5.33, ln 34). Delivered values are the time and temperature vectors, plus the output blood flow (listing 5.33, ln 36).

These are the main function of the model but they cannot be applied to the trunk. As pools in trunk are considered core nodes, they are modelled inside of the element trunk. In the trunk, the distribution of the blood to the lungs, trunk tissues, head and limbs, and the recover from all of those elements, justifies the creation of new object to manage all the elements.

```
def T_ve_trunk(T_ve_init, T_t, T_ve_in_armr, T_ve_in_arml, T_ve_in_legr,
               T_ve_in_legl, T_ve_in_head, Vol_ve, V_ve_in_armr,
    V_ve_in_arml,
               V_ve_in_legr, V_ve_in_legl, V_ve_in_head, Vazao_sv, rho_bl,
               c_bl, timeStep):

    sum_Vsv_Tmt = sum(Vazao_sv*T_t.value)
    V_ve = (sum(V_ve_in_armr) + sum(V_ve_in_arml) + sum(V_ve_in_legr) +
            sum(V_ve_in_legl) +
            sum(V_ve_in_head) + sum(Vazao_sv))

    def derivtve(T__ve,t): # return derivatives of the array T_ve
            return (rho_bl*c_bl*(sum_Vsv_Tmt + V_ve_in_armr*T_ve_in_armr +
                    V_ve_in_arml*T_ve_in_arml + V_ve_in_legr*T_ve_in_legr +
                    V_ve_in_legl*T_ve_in_legl + V_ve_in_head*T_ve_in_head -
                    V_ve*T__ve[0]))/(Vol_ve*rho_bl*c_bl)

    time = linspace(0.0,timeStep,2)

    T__ve = odeint(derivtve,T_ve_init,time)

    return T__ve[-1]
```

Listing 5.33: Function that predicts mean venous temperature of a trunk's pool.

Function `T_ve_trunk` (listing 5.33), responds to the specific need of trunk's venous pool. It includes the venous flows from the 5 connections, heat, arms and legs, plus the venous blood that come from the trunk tissues, with the exception of the lungs. To the venous core reservoir at the trunk, lungs are seen as the single volume receptor that links to arterial core pool.

Second level functions integrate basic functions. The information they use as input, is the same that they need to give to basic functions. Their role is mainly dedicated to organize the sequence of calculus.

```
def body_elem(T_ar_init, mesh, T_t_init, V_sv, q_t, rho_t, c_t, k_t, T_ar_in
    ,
              T_ve_init, H_av, V_ar_in, Vol_ar, rho_bl, c_bl, T_ve_in,
    Vol_ve,
              V_ve_in, Vazao_sv, timeStep):

    Tar = func_T_ar.T_ar(T_ar_init, T_ar_in, T_ve_init, H_av, V_ar_in,
    Vol_ar,
                         rho_bl, c_bl, timeStep)

    Tt = T_t.T_t(mesh, T_t_init, Tar, V_sv, q_t, rho_t, c_t, k_t, rho_bl,
                   c_bl, timeStep)

    Tve = func_T_ve.T_ve(T_ve_init, Tt, Tar, T_ve_in, Vol_ve, V_ve_in, H_av,
                         Vazao_sv, rho_bl, c_bl, timeStep)

    return Tar, Tt, Tve
```

Listing 5.34: Function that models the body segments with the exception of trunk and head.

Beginning with `body_elem`, this functions is used to simulate the thermal behaviour of middle elements at limbs and neck. Starts by calculating the arterial blood temperature in the segment, that temperature is used as an input to the function that calculates the tissue temperature, and the respective results are used to calculate the contribution of cell tissue temperature to the final temperature of venous blood.

The only difference to use of `body_elem` routine to model an end segment, is that at the extremities, *approximately* all arterial blood goes to the tissues, and all the venous blood come from the tissues. In other words, the link between artery and vein is done through tissues and small vessels. In practice the `V_ve_in` variable was settled as null value, respective venous temperature `T_ve_in` too. As output, are the same temperature variables, but they only depend on arterial blood at entry, venous blood that goes out, tissues and respective boundaries.

```
def Vazao_sv(mesh, V_sv):
    # Cell Volume
    Cell_vol = mesh.cellVolumes

    # Calculation of the volumes of flow
    Val_sv = CellVariable(mesh=mesh, value=0.)

    # Flow Matrix per unit of volume
    for i in range(4):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'
        Val_sv.setValue(V_sv[i] , where=mesh.physicalCells[string])

    # Tissues Blood Flow
    Vazao_sv = Val_sv.value * Cell_vol # Final Flow of each compartment of
    each cell in m^3/s

    return Vazao_sv

def Vazao_sv_lung(mesh, V_sv):
    # Cell Volume
    Cell_vol = mesh.cellVolumes

    # Calculation of the volumes of flow
    Val_sv = CellVariable(mesh=mesh, value=0.)

    # Flow Matrix per unit of volume
    Val_sv.setValue(V_sv[6] , where=mesh.physicalCells['Lung'])

    # Tissues Blood Flow
    Vazao_sv = Val_sv.value * Cell_vol # Final Flow of each compartment of
   each cell in m^3/s

    return Vazao_sv

def Vazao_sv_trunk(mesh, V_sv):
    # Cell Volume
    Cell_vol = mesh.cellVolumes

    # Calculation of the volumes of flow
    Val_sv = CellVariable(mesh=mesh, value=0.)

    # Flow Matrix per unit of volume
    for i in range(4):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'
        elif i==4 : string='Heart'
        elif i==5 : string='Viscera'
        Val_sv.setValue(V_sv[i] , where=mesh.physicalCells[string])

    # Tissues Blood Flow
    Vazao_sv = Val_sv.value * Cell_vol # Final Flow of each compartment of
   each cell in m^3/s

    return Vazao_sv


def Vazao_sv_head(mesh, V_sv):
    # Cell Volume
    Cell_vol = mesh.cellVolumes

    # Calculation of the volumes of flow
    Val_sv = CellVariable(mesh=mesh, value=0.)

    # Flow Matrix per unit of volume
    for i in range(4):
        if i==0 : string='Bone'
        elif i==1 : string='Muscle'
        elif i==2 : string='Fat'
        elif i==3 : string='Skin'
        elif i==4 : string='Brain'
        Val_sv.setValue(V_sv[i] , where=mesh.physicalCells[string])

    # Tissues Blood Flow
    Vazao_sv = Val_sv.value * Cell_vol # Final Flow of each compartment of
   each cell in m^3/s

    return Vazao_sv
```

Listing 5.35: Functions to found amount of blood that flows through out a cell.

For last, the routine `vazao` is composed by 6 functions. The amount of blood that flows from one element to another is considered to be dependent on the tissue/body local blood supply requirements. Initial values found in literature give the amount of blood required per type of tissue per unit of volume. The first 4 functions `Vazao_sv`, `Vazao_sv_lung`, `Vazao_sv_head`, `Vazao_sv_trunk` give the amount of blood that flows per cell. To found the total amount of blood that comes from each cell, the flow per unit of volume was multiplied by the cell volume. The result are matrices with the same size and shape of body meshes. Considering, once again, the differences in structure of trunk and head, specific function were made for them. As lungs are not seen as equal part of the trunk, extra variable was defined to lungs blood flow.

```
def Vazao_sv_seg(hand, lowerarm, upperarm, foot, leg, thigh, trunk, neck,
   head,
                 V_sv):
    vazao_sv_handr     = Vazao_sv(hand, V_sv[0])
    vazao_sv_lowerarmr = Vazao_sv(lowerarm, V_sv[2])
    vazao_sv_upperarmr = Vazao_sv(upperarm, V_sv[4])
    vazao_sv_handl     = Vazao_sv(hand, V_sv[1])
    vazao_sv_lowerarml = Vazao_sv(lowerarm, V_sv[3])
    vazao_sv_upperarml = Vazao_sv(upperarm, V_sv[5])
    vazao_sv_footr     = Vazao_sv(foot, V_sv[6])
    vazao_sv_legr      = Vazao_sv(leg, V_sv[8])
    vazao_sv_thighr    = Vazao_sv(thigh, V_sv[10])
    vazao_sv_footl     = Vazao_sv(foot, V_sv[7])
    vazao_sv_legl      = Vazao_sv(leg, V_sv[9])
    vazao_sv_thighl    = Vazao_sv(thigh, V_sv[11])
    vazao_sv_trunk     = Vazao_sv_trunk(trunk, V_sv[12])
    vazao_sv_lung      = Vazao_sv_lung(trunk, V_sv[12])
    vazao_sv_neck      = Vazao_sv(neck, V_sv[13])
    vazao_sv_head      = Vazao_sv_head(head, V_sv[14])
    return (vazao_sv_upperarmr, vazao_sv_lowerarmr, vazao_sv_handr,
            vazao_sv_upperarml, vazao_sv_lowerarml, vazao_sv_handl,
            vazao_sv_thighr, vazao_sv_legr, vazao_sv_footr, vazao_sv_thighl,
            vazao_sv_legl, vazao_sv_footl, vazao_sv_trunk, vazao_sv_neck,
            vazao_sv_lung, vazao_sv_head)

def V_ar_ve_in(Vazao_sv_upperarmr, Vazao_sv_lowerarmr, Vazao_sv_handr,
               Vazao_sv_upperarml, Vazao_sv_lowerarml, Vazao_sv_handl,
               Vazao_sv_thighr, Vazao_sv_legr, Vazao_sv_footr,
   Vazao_sv_thighl,
                Vazao_sv_legl, Vazao_sv_footl, Vazao_sv_trunk, Vazao_sv_neck,
                Vazao_sv_lung, Vazao_sv_head):
    V_ve = [0.,0.,
            sum(Vazao_sv_handr),sum(Vazao_sv_handl),
            sum(Vazao_sv_lowerarmr)+sum(Vazao_sv_handr),
            sum(Vazao_sv_lowerarml)+sum(Vazao_sv_handl),
            0.,0.,sum(Vazao_sv_footr),sum(Vazao_sv_footl),
            sum(Vazao_sv_legr)+sum(Vazao_sv_footr),
            sum(Vazao_sv_legl)+sum(Vazao_sv_footl),
            [sum(Vazao_sv_upperarmr)+sum(Vazao_sv_lowerarmr)+sum(
    Vazao_sv_handr),
            sum(Vazao_sv_upperarml)+sum(Vazao_sv_upperarml),
            sum(Vazao_sv_thighr)+sum(Vazao_sv_legr)+sum(Vazao_sv_footr),
            sum(Vazao_sv_thighl)+sum(Vazao_sv_legl)+sum(Vazao_sv_footl),
            sum(Vazao_sv_neck)+sum(Vazao_sv_head)],
            sum(Vazao_sv_head), 0.]
    V_ar = [sum(Vazao_sv_handr), sum(Vazao_sv_handl),
            sum(Vazao_sv_lowerarmr)+sum(Vazao_sv_handr),
            sum(Vazao_sv_lowerarml)+sum(Vazao_sv_handl),
            sum(Vazao_sv_upperarmr)+sum(Vazao_sv_lowerarmr)+sum(
    Vazao_sv_handr),
            sum(Vazao_sv_upperarml)+sum(Vazao_sv_lowerarml)+sum(
    Vazao_sv_handl),
            sum(Vazao_sv_footr),sum(Vazao_sv_footl),
            sum(Vazao_sv_legr)+sum(Vazao_sv_footr),
            sum(Vazao_sv_legl)+sum(Vazao_sv_footl),
            sum(Vazao_sv_thighr)+sum(Vazao_sv_legr)+sum(Vazao_sv_footr),
            sum(Vazao_sv_thighl)+sum(Vazao_sv_legl)+sum(Vazao_sv_footl),
            sum(Vazao_sv_lung),sum(Vazao_sv_neck)+sum(Vazao_sv_head),
            sum(Vazao_sv_head)]
    return V_ar, V_ve
```

Listing 5.36: Second level functions to found flows matrices per segment and arterial an venous flow that get in a segment.

With the 4 basic blood flow functions are built the flow matrices for each body element with the function `Vazao_sv_seg`. The output matrices that come from this function are used as input in `V_ar_ve` to found the specific amount of blood that goes from one segment to the other. The output matrices represent the venous and arterial flows at an entry of a segment. That is why `V_ve` at hands and feet is settled as null.

Data

Data folder is where data is saved and where routines for saving tissue temperature files are placed. `Save` file is composed by two functions `save_T_t_init` and `save_T_t`. One to save initial temperature values at $'0 seconds'$ and other to save files inside the cycle of calculus. Functions `save_T_t` is the generic version exemplified here in listing 5.37,it receives tissues temperature cell variables, from all 15 segments, as input, and saves them in '.tsv' format using a Fipy function named `TSVViewer`. The name of the file is composed by the name of the segment and the time of simulation calculated inside the cycle as `savestep` variable.

```
def save_T_t(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
             T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
             T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
             T_t_trunk, T_t_neck, T_t_head, savestep):
    str = repr(savestep) + 's'
    TSVViewer(vars=T_t_handr).plot(filename=T_t_handr.name + str)
    TSVViewer(vars=T_t_lowerarmr).plot(filename=T_t_lowerarmr.name + str)
    TSVViewer(vars=T_t_upperarmr).plot(filename=T_t_upperarmr.name + str)
    TSVViewer(vars=T_t_handl).plot(filename=T_t_handl.name + str)
    TSVViewer(vars=T_t_lowerarml).plot(filename=T_t_lowerarml.name + str)
    TSVViewer(vars=T_t_upperarml).plot(filename=T_t_upperarml.name + str)
    TSVViewer(vars=T_t_footr).plot(filename=T_t_footr.name + str)
    TSVViewer(vars=T_t_legr).plot(filename=T_t_legr.name + str)
    TSVViewer(vars=T_t_thighr).plot(filename=T_t_thighr.name + str)
    TSVViewer(vars=T_t_footl).plot(filename=T_t_footl.name + str)
    TSVViewer(vars=T_t_legl).plot(filename=T_t_legl.name + str)
    TSVViewer(vars=T_t_thighl).plot(filename=T_t_thighl.name + str)
    TSVViewer(vars=T_t_trunk).plot(filename=T_t_trunk.name + str)
    TSVViewer(vars=T_t_neck).plot(filename=T_t_neck.name + str)
    TSVViewer(vars=T_t_head).plot(filename=T_t_head.name + str)
```

Listing 5.37: Function used to save tissue temperature variables.

Viewers

Viewers are routines that allow to manage the data and display it. `CellVariable` objects could be plot directly using routines from `fipy` but, when scale up was done, it become more difficult to manage the results as desired. So here, the solution was to built functions to view and update the data during the simulation process.

The most complex function was developed to convert the data from `CellVariable` object to a readable format in Mayavi it calls `my_grid` (listing 5.38).

```
def my_grid(T_t):

    cell_num = int(T_t.mesh.numberOfCells)
    points=T_t.mesh.vertexCoords

    vertex_numb = int(size(points)/len(points))

    vertex_points = [[row[i] for row in points] for i in range(vertex_numb)]

    del points

    cells=T_t.mesh._orderedCellVertexIDs.data

    vertex_ids = [[row[i] for row in cells] for i in range(int(cells.size/
    len(cells)))]

    del cells, row

    [vertex_ids[i].insert(0,len(vertex_ids[i])-vertex_ids[i].count(-1)) for
    i in range(cell_num)];


    num=[]
    [num.append(vertex_ids[i][0]) for i in range(cell_num)];


    vertex_ids = [numb for elem in vertex_ids for numb in elem] # flatten
    the list
    count = vertex_ids.count(-1)
    [vertex_ids.remove(-1) for i in range(count)]; # remove -1

    del elem


    offset = []
    c=0
    for i in range(cell_num):
        offset.append(c)
        c=c+num[i]+1

    hex_type = tvtk.Hexahedron().cell_type
    pen_type = tvtk.Wedge().cell_type
    cell_types=[]

    for i in range(cell_num):
        if num[i] == 8: type = hex_type
        elif num[i] == 6: type = pen_type
        cell_types.append(type)
    #create cell_array data, firts number is numper of vertices of the cell
    #and the next point give the vertex ids
    cell_array = tvtk.CellArray()
    cell_array.set_cells(cell_num, vertex_ids)
    # Now create the UG.
    ug = tvtk.UnstructuredGrid(points=vertex_points)
    # Now just set the cell types and reuse the ug locations and cells.
    ug.set_cells(cell_types, offset, cell_array)
    scalars = T_t.value
    ug.cell_data.scalars = scalars
    ug.cell_data.scalars.name = 'Temperature Distribution'

    return ug
```

Listing 5.38: Data conversion into *tvtk* unstructured grid type.

Basically this function only needs a cell variable as input. First 25 lines of code are dedicated to pick cell vertex identifiers and reorganize the matrix values to match the requirements of `Cell_Array` function of tvtk library. Then the two for... in... cycles are used to create the arrays `offset` (the offsets for the cells, i.e. the indices where the cells start), and `cell_types` (array that defines the type of cells in the grid). This function was created to deal specifically with hexahedrons and wedges cell types only. The `ug` variable is an unstructured grid created to be defined through `vertex_points`. The object properties allow to set the cell types, and reuse `ug` locations to set the scalars and respective name.

Much simpler its the functions used to visualize the Mayavi scene and display the dataset in it (listing 5.39).

```
def view(dataset):
    """ Open up a mayavi scene and display the dataset in it.
    """
    fig = mlab.figure(bgcolor=(1, 1, 1), fgcolor=(0, 0, 0),
                      figure=dataset.cell_data.scalars.name)
    surf = mlab.pipeline.surface(dataset, opacity=1, colormap='jet',
                                 vmax=38., vmin= 33.)
    mlab.colorbar(surf, title='Temperature',nb_labels=5)

    return fig
```

Listing 5.39: Dataset displayer for three dimensional data.

The sequence of commands (listing 5.39) creates the figure, defines the surface to plot and finally settles the colorbar legend. Several parameters can be redefined such background colour, figure colour, opacity of the plotted grid, colour map used to scale temperature in the grid, boundaries of values to which temperature is plotted, and so on.

Finally the other type of functions considered in the viewers are the updaters. Function `update3D` is used to, during the simulation process, update the values of tissue temperatures(listing 5.40).

```
def update3D(dataset, filename , arr):

    dataset.cell_data.scalars=arr
    dataset.cell_data.scalars.name = filename
    dataset.modified()
```

Listing 5.40: Updater of tissues temperatures values of a segment.

On other hand, the updater `update_tar_tve` function repeats a sequence of code used to actualize the data vectors `tar` and `tve` of all body elements, by adding the last calculated value (listing 5.41).

```
def update_tar_tve(timestep, t, T_ar_handr, T_ar_handl, T_ar_lowerarmr,
                   T_ar_lowerarml, T_ar_upperarmr, T_ar_upperarml,
                   T_ar_footr,T_ar_footl, T_ar_legr, T_ar_legl,
                   T_ar_thighr, T_ar_thighl, T_ar_trunk, T_ar_neck,
                   T_ar_head, T_ve_handr, T_ve_handl, T_ve_lowerarmr,
                   T_ve_lowerarml, T_ve_upperarmr, T_ve_upperarml,
                   T_ve_footr,T_ve_footl, T_ve_legr,T_ve_legl,
                   T_ve_thighr,T_ve_thighl, T_ve_trunk, T_ve_neck,
                   T_ve_head, tar_handr, tar_lowerarmr, tar_upperarmr,
                   tar_handl, tar_lowerarml, tar_upperarml , tar_footr,
                   tar_legr, tar_thighr, tar_footl, tar_legl , tar_thighl,
                   tar_trunk, tar_neck, tar_head, tve_handr, tve_lowerarmr,
                   tve_upperarmr, tve_handl, tve_lowerarml, tve_upperarml,
                   tve_footr, tve_legr, tve_thighr, tve_footl, tve_legl,
                   tve_thighl, tve_trunk, tve_neck, tve_head):

    tar_handr.append(T_ar_handr)
    tar_lowerarmr.append(T_ar_lowerarmr)
    tar_upperarmr.append(T_ar_upperarmr)
    tar_handl.append(T_ar_handl)
    tar_lowerarml.append(T_ar_lowerarml)
    tar_upperarml.append(T_ar_upperarml)
    tar_footr.append(T_ar_footr)
    tar_legr.append(T_ar_legr)
    tar_thighr.append(T_ar_thighr)
    tar_footl.append(T_ar_footl)
    tar_legl.append(T_ar_legl)
    tar_thighl.append(T_ar_thighl)
    tar_trunk.append(T_ar_trunk)
    tar_neck.append(T_ar_neck)
    tar_head.append(T_ar_head)
    tve_handr.append(T_ve_handr)
    tve_lowerarmr.append(T_ve_lowerarmr)
    tve_upperarmr.append(T_ve_upperarmr)
    tve_handl.append(T_ve_handl)
    tve_lowerarml.append(T_ve_lowerarml)
    tve_upperarml.append(T_ve_upperarml)
    tve_footr.append(T_ve_footr)
    tve_legr.append(T_ve_legr)
    tve_thighr.append(T_ve_thighr)
    tve_footl.append(T_ve_footl)
    tve_legl.append(T_ve_legl)
    tve_thighl.append(T_ve_thighl)
    tve_trunk.append(T_ve_trunk)
    tve_neck.append(T_ve_neck)
    tve_head.append(T_ve_head)
    t.append(t[-1]+timestep)
```

Listing 5.41: Updater of the venous and arterial temperature of segments.

Data vectors created by `update_tar_tve` function plotted and saved by `fig_bl` function of `viewer2D` routine. It uses a single couple of `tar`, `tve` and respective description `name`, to create a plot (listing 5.42).

```
def fig_bl(t, tar, tve, name):

    figure(num="Blood Temperatures", figsize=(7, 5), dpi=100, facecolor='w',
     edgecolor='w')
    plot(t,tar, 'r-',label=name + '$\, Arterial \, Blood \, Temp$')
    plot(t,tve,'b-', label=name + '$\, Venous \, Blood \, Temp$')
    legend(fontsize='large',loc=4)
    xlabel(r'$t\,(s)$', fontsize=15)
    ylabel(r'${Temp.}\,(\,^{\circ}{C}\,)}$', fontsize=14)
    ylim(35.,38.)
    xlim(0,250)
    savefig('tartve.png')
```

Listing 5.42: Function that creates and save the plots of arterial and venous temperatures over the time.

As the data set generated during the simulation grows, the process of visualization needs to be settled apart from the calculation. Final version of the program was then divided the simulation in two stages. First stage was dedicated to the calculation or simulation process. Results were generated and saved in separate folder called `data`.

Other stage is optional and can be done in separate moment, only to visualize the saved data, it is data animation. Listing 5.43 is basic 3D routine dedicated to create the animated view of the temperature change in a body segment over the time.

```
def anim_T_t(T_t, steps):

    dataset = my_grid(T_t)
    view(dataset)

    for step in range(steps):

        savestep = step*20+20

        strn= T_t.name +repr(savestep)+'s'

        foo = open(str, 'r')
        arr = numpy.loadtxt(foo, skiprows = 2, dtype = 'f8', usecols = [3] )
        foo.close()

        update3D(dataset, strn, arr)

        sleep(1)
        print strn

if __name__ == '__main__':

    name = raw_input("Input segment name...")
    time = raw_input("Input simulation time...")

    T_t = CellVariable(name=name,
                       mesh=Gmsh3D('geometries/'+ name +'.geo'), value=0.)
    anim_T_t(T_t, time)
```

Listing 5.43: Function that creates and save the plots of arterial and venous temperatures over the time.

First two lines of code use `my_grid` and `view` function to create the figure and plot the initial values, then the `for...in...` loop opens the saved data and updates the viewer. Simulation time value is also printed. If file runs as main code, it asks information about segment to visualize and the period to see (listing 5.43).

Main File

Main file is the file, as mentioned, is the responsible to manage all the information. Constants and initial values are defined here as array variables, whose elements are used as inputs in solver functions. All the constants, initial values and boundaries are imported from the respective folders.

```

from fipy import *
from viewers.update import update_tar_tve
from viewers.viewer3D import my_grid, view
from matplotlib.pyplot import *
from pylab import *
from solvers.midd_body_elem import body_elem
from solvers.trunk import trunk
from solvers.head import head
from solvers.vazao import Vazao_sv_seg, V_ar_ve_in
from meshes.meshes import importmeshs
from variables.vars_T_t import vars_Tt
from initials.set_values import set_values_T_t, set_values_Tar,
    set_values_Tve
from initials.initial_cond import (import_init_temp, import_V_basal,
                                   import_constants, import_vol_ar_ve,
                                    import_q_t)
from boundaries.b_values import bvalues
from boundaries.bound_cond import frontsidecells, sidecells, topsidecells
from boundaries.set_bounds import set_bounds
```

Listing 5.44: Main file 'header' with an itemization of used routines.

Using the typical structure of python code, first lines are dedicated to import libraries and routines used in main code (listing 5.44). As in the simpler versions, the steps to proceed the calculus are import/set initial conditions, import meshes and define variables (listing 5.45).

```
#=========================================================================
# Initial Conditions -  initial values can, and should,
#  be settled to each element at a time

T_ar_init, T_ve_init, T_t_init = import_init_temp()

#=========================================================================
# Import Meshes and set and save initial values
(hand, lowerarm, upperarm, foot,
leg, thigh, trunk, neck, head) = importmeshs()

# Create Variables
(T_t_handr, T_t_lowerarmr,
T_t_upperarmr, T_t_handl,
T_t_lowerarml, T_t_upperarml,
T_t_footr, T_t_legr, T_t_thighr,
T_t_footl, T_t_legl, T_t_thighl,
T_t_trunk, T_t_neck, T_t_head) = vars_Tt(hand, lowerarm, upperarm, foot,
                                        leg, thigh, trunk, neck, head)

# Set initial conditions to tissues
set_values_T_t(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
               T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
               T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
               T_t_trunk, T_t_neck, T_t_head, T_t_init)

# Save the data
save_T_t_init(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
              T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
              T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
```

Listing 5.45: Code to import/set initial conditions to the mesh.

The main change of the large program code is that blood flow is calculated through a double function routine based on blood need of each segment, this is, according to segment composition, size and order in the body structure (listing 5.46).

```
#==========================================================================
#Import Bloof flow values in basal thermoneutras conditions
CO, V_sv = import_V_basal()

#==========================================================================
# Calculate the amount of blood that cross the tissues (m3.s-1)
(vazao_sv_upperarmr, vazao_sv_lowerarmr,
vazao_sv_handr, vazao_sv_upperarml,
vazao_sv_lowerarml, vazao_sv_handl,
vazao_sv_thighr, vazao_sv_legr,
vazao_sv_footr, vazao_sv_thighl,
vazao_sv_legl, vazao_sv_footl,
vazao_sv_trunk, vazao_sv_neck,
vazao_sv_lung, vazao_sv_head) = Vazao_sv_seg(hand,lowerarm,upperarm,foot,leg
    ,
                                             thigh, trunk, neck, head, V_sv)

# Calculate the arterial and venous blood flow that goes in into each
    segment
V_ar, V_ve = V_ar_ve_in(vazao_sv_upperarmr, vazao_sv_lowerarmr,
    vazao_sv_handr,
                        vazao_sv_upperarml, vazao_sv_lowerarml,
    vazao_sv_handl,
                        vazao_sv_thighr, vazao_sv_legr, vazao_sv_footr,
                        vazao_sv_thighl, vazao_sv_legl, vazao_sv_footl,
                        vazao_sv_trunk, vazao_sv_neck, vazao_sv_lung,
                        vazao_sv_head)
```

Listing 5.46: Calculation of basal blood flows in large veins and arteries based on tissue blood requirements.

Basal values of small vessels and tissue blood flow are the base of blood flow at thermoneutral basal state. However exercise and hot temperatures influence tissue requirements of blood flow. This adaptation must be considered during the incorporation of active thermoregulation mechanisms.

In practice result that impedances along the arterial branches are calculated at the beginning of the calculation cycle, and are based on the arterial radii of the segments Avolio (1980). This radii is estimated by skin and hypothalami temperature, through relation stated by Fu (1995) in his experiments. All these will lead to+ update the variables `Vol_ve` and `Vol_ar`, here imported as constants, along with tissues heat production `q_t` (listing 5.47).

```
#==========================================================================
# Import Constants
rho_t, c_t, k_t, H_av, rho_bl, c_bl = import_constants()

```

```
q_t = import_q_t()

Vol_ar, Vol_ve = import_vol_ar_ve()
```

Listing 5.47: Importing constants and basal values of passive system.

Mean cardiac output, in future, should also to be updated at each cycle. This is done though simplifications of CO as suggested by Karaki et al. (2013), depending only on core (estimated by hypothalamus), or mean skin temperature (estimated skin temperate at all segments and surface area). If depends on exercise, the proposal of Salloum et al. (2007) should be followed, that present a direct relation between final mean cardiac output and metabolic effort. This changes should be kept up with change of local heat production, such for example the increase in muscle heat production during exercise, or the active mechanism shivering, and the redistribution of blood from visceral tissue to skin and muscles.

Parametrization of boundaries are a key for the success of response of present model to transient conditions. Listing 5.48 shows how functions of boundaries were used to define the places of the mesh segment that exchange heat with surroundings. Here the mesh `where` presents one condition per body segment, but more that one boundary can be set per segment at a time. Notice that model present array is used to say “where” are the bound. The condition value is settled in the loop step by step, and this is what makes the model able to respond to transients.

```
# Set Boundary conditions - one diferent type per body segment

where = [frontsidecells(hand), frontsidecells(hand), #hands r-l
        sidecells(lowerarm), sidecells(lowerarm), #lower arms r-l
        sidecells(upperarm), sidecells(upperarm), #upper arms r-l
        frontsidecells(foot), frontsidecells(foot), #feet r-l
        sidecells(leg), sidecells(leg), #legs r-l
        sidecells(thigh), sidecells(thigh), #thighs r-l
        sidecells(trunk),  # trunk
        sidecells(neck),  # neck
        topsidecells(head)] # head
```

Listing 5.48: Defining variable to settle the place of the bounds where condition occurs.

After settle the bounds, are created vector variables to save time (`t`), arterial (`tar`) and venous (`tve`) temperatures for each segment.

The time step (`timeStep`) of calculus and number of steps (`steps`) are necessary to define the number of times that loop needs to be repeated and time used to solve equations. Here is defined a time step of 5 second and 20 steps, what means 100 seconds of simulation. As mentioned before this values must be settled according to the kind of simulation. Severe an constant changes in boundaries, and high values require small step to get accurate results, more stale conditions can be approximated through larger time steps. In the future this values are going to be asked, in prompt, to the user. For now default values used to exemplify and test routines (listing 5.49).

```
# Defining the steps for calculate the simulated values

t = [0]

(tar_handr, tar_lowerarmr, tar_upperarmr, tar_handl,
 tar_lowerarml, tar_upperarml , tar_footr, tar_legr,
 tar_thighr, tar_footl, tar_legl,tar_thighl,tar_trunk,
 tar_neck, tar_head) = set_values_Tar(T_ar_init)

(tve_handr, tve_lowerarmr, tve_upperarmr,
 tve_handl,tve_lowerarml,tve_upperarml, tve_footr, tve_legr,
 tve_thighr, tve_footl,tve_legl, tve_thighl, tve_trunk,
 tve_neck, tve_head) = set_values_Tve(T_ve_init)

timeStep=5. #time step in seconds

steps=20 # number of steps per simulation
```

Listing 5.49: Defining arterial and venous temperature values and settling time step and steps of simulation.

As mentioned above, the process of calculus occurs inside the `for..in..:` loop. First by picking the boundary values for that step, then by calculating the temperatures (arterial, venous and tissues) for all body segments. Once calculated the values for that step, a redefinition of variables is carried out to settle the new initial values to the next step of calculus (listing 5.50).

```
for step in range(steps):
    set_bounds(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
               T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
               T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
               T_t_trunk, T_t_neck, T_t_head, T_t_init, where, bvalues(step)
    )

    (Tar_handr, Tt_handr,
Tve_handr) = body_elem(T_ar_init[0], hand, T_t_handr, V_sv[0],
                       q_t[0], rho_t[0], c_t[0], k_t[0],
                        T_ar_init[2], T_ve_init[0],
                        H_av[0], V_ar[0], Vol_ar[0],
                        rho_bl, c_bl, 0., Vol_ve[0],
                        V_ve[0], vazao_sv_handr,
                        timeStep)

(Tar_handl, Tt_handl,
Tve_handl) = body_elem(T_ar_init[1],hand, T_t_handr, V_sv[1],
                                           q_t[1], rho_t[1], c_t[1],k_t
[1],
                                           T_ar_init[3], T_ve_init[1],
                                           H_av[1], V_ar[1], Vol_ar[1],
                                           rho_bl, c_bl, 0., Vol_ve[1],
                                           V_ve[1], vazao_sv_handl,
                                           timeStep)

(Tar_footr, Tt_footr,
Tve_footr) = body_elem(T_ar_init[6],foot, T_t_handr, V_sv[6],
                                           q_t[6], rho_t[6], c_t[6],k_t
[6],
                                           T_ar_init[8], T_ve_init[6],
                                           H_av[6], V_ar[6], Vol_ar[6],
                                           rho_bl, c_bl, 0., Vol_ve[6],
                                           V_ve[6],vazao_sv_footr,
timeStep)

(Tar_footl, Tt_footl,
Tve_footl) = body_elem(T_ar_init[7],foot, T_t_handr, V_sv[7],
                                           q_t[7], rho_t[7], c_t[7],k_t
[7],
                                           T_ar_init[9], T_ve_init[7],
                                           H_av[7], V_ar[7], Vol_ar[7],
                                           rho_bl, c_bl, 0., Vol_ve[7],
                                           V_ve[7],vazao_sv_footl,
timeStep)

(Tar_lowerarmr, Tt_lowerarmr,
Tve_lowerarmr) = body_elem(T_ar_init[2],lowerarm, T_t_lowerarmr,
                           V_sv[2], q_t[2],
                           rho_t[2], c_t[2], k_t[2], T_ar_init[4],
                           T_ve_init[2], H_av[2], V_ar[2], Vol_ar[2],
                           rho_bl, c_bl, T_ve_init[0], Vol_ve[2], V_ve
[2],
                           vazao_sv_lowerarmr, timeStep)

(Tar_lowerarml, Tt_lowerarml,
Tve_lowerarml) = body_elem(T_ar_init[3],lowerarm, T_t_lowerarml,
                          V_sv[3], q_t[3],
                          rho_t[3], c_t[3], k_t[3], T_ar_init[5],
                          T_ve_init[3], H_av[3], V_ar[3], Vol_ar[3],
                          rho_bl, c_bl, T_ve_init[1], Vol_ve[3], V_ve
[3],
                          vazao_sv_lowerarml, timeStep)
```

```
    (Tar_upperarmr, Tt_upperarmr,
    Tve_upperarmr) = body_elem(T_ar_init[4],upperarm, T_t_upperarmr, V_sv
    [4],
                             q_t[4], rho_t[4],
                             c_t[4], k_t[4], T_ar_init[12], T_ve_init[4],
                             H_av[4], V_ar[4], Vol_ar[4], rho_bl, c_bl,
                             T_ve_init[2], Vol_ve[4], V_ve[4],
                             vazao_sv_upperarmr, timeStep)

    (Tar_upperarml, Tt_upperarml,
    Tve_upperarml) = body_elem(T_ar_init[5],upperarm, T_t_upperarml, V_sv
    [5],
                             q_t[5], rho_t[5],
                             c_t[5], k_t[5], T_ar_init[12], T_ve_init[5],
                             H_av[5], V_ar[5], Vol_ar[5], rho_bl, c_bl,
                             T_ve_init[3], Vol_ve[5], V_ve[5],
                             vazao_sv_upperarml, timeStep)

    (Tar_legr, Tt_legr,
     Tve_legr) = body_elem(T_ar_init[8],leg, T_t_legr,
                                             V_sv[8], q_t[8],
                                             rho_t[8], c_t[8], k_t[8],
                                             T_ar_init[10], T_ve_init[8],
                                             H_av[8], V_ar[8], Vol_ar[8],
                                             rho_bl, c_bl, T_ve_init[6],
                                             Vol_ve[8], V_ve[8],
    vazao_sv_legr,
                                             timeStep)

    Tar_legl, Tt_legl, Tve_legl = body_elem(T_ar_init[9],leg, T_t_legl,
                                             V_sv[9], q_t[9],
                                             rho_t[9], c_t[9], k_t[9],
                                             T_ar_init[11], T_ve_init[9],
                                             H_av[9], V_ar[9], Vol_ar[9],
                                             rho_bl, c_bl, T_ve_init[7],
                                             Vol_ve[9], V_ve[9],
    vazao_sv_legl,
                                             timeStep)

    Tar_thighr, Tt_thighr, Tve_thighr = body_elem(T_ar_init[10],thigh,
                                                    T_t_thighr, V_sv[10],
                                                    q_t[10], rho_t[10], c_t
    [10],
                                                    k_t[10], T_ar_init[12],
                                                    T_ve_init[10], H_av[10],
                                                    V_ar[10], Vol_ar[10],
    rho_bl,
                                                    c_bl, T_ve_init[8],
                                                    Vol_ve[10], V_ve[10],
                                                    vazao_sv_thighr, timeStep)

    Tar_thighl, Tt_thighl, Tve_thighl = body_elem(T_ar_init[11],thigh,
                                                    T_t_thighl, V_sv[11],
                                                    q_t[11], rho_t[11], c_t
    [11],
                                                    k_t[11], T_ar_init[12],
                                                    T_ve_init[11], H_av[11],
                                                    V_ar[11], Vol_ar[11], rho_bl,
                                                    c_bl, T_ve_init[9],
                                                    Vol_ve[11], V_ve[11],
                                                    vazao_sv_thighl, timeStep)

    Tar_neck, Tt_neck, Tve_neck = body_elem(T_ar_init[13],neck, T_t_neck,
                                            V_sv[13],q_t[13],rho_t[13],c_t[13],
                                            k_t[13], T_ar_init[12],
                                              T_ve_init[13], H_av[13], V_ar[13],
                                              Vol_ar[13], rho_bl, c_bl,
                                              T_ve_init[14], Vol_ve[13],V_ve[13],
                                              vazao_sv_neck, timeStep)

    Tar_head, Tt_head, Tve_head = head(T_ar_init[14],head, T_t_head, V_sv[14],
                                       q_t[14],rho_t[14],c_t[14],k_t[14],
                                       T_ar_init[13],T_ve_init[14],
                                       V_ar[14], Vol_ar[14],
                                       rho_bl, c_bl, Vol_ve[14], vazao_sv_head,
                                       timeStep)

    Tar_trunk, Tt_trunk, Tve_trunk = trunk(T_ar_init[12],T_ve_init[12],
                                           vazao_sv_lung,Vol_ar[12], trunk,
                                           T_t_trunk, V_sv[12], q_t[12],
                                           T_ve_init[4], T_ve_init[5],
                                           T_ve_init[10], T_ve_init[11],
                                           T_ve_init[13], Vol_ve[12], V_ve[4],
                                           V_ve[5], V_ve[10], V_ve[11],
                                           V_ve[13], vazao_sv_trunk, rho_bl,
                                           c_bl, timeStep)

    T_ar_init = array([Tar_handr,Tar_handl,
                    Tar_lowerarmr,Tar_lowerarml,
                    Tar_upperarmr,Tar_upperarml,
                    Tar_footr,Tar_footl,
                    Tar_legr,Tar_legl,
                    Tar_thighr,Tar_thighl,
                    Tar_trunk,
                    Tar_neck,
                    Tar_head])
    T_ve_init = array([Tve_handr,Tve_handl,
                    Tve_lowerarmr,Tve_lowerarml,
                    Tve_upperarmr,Tve_upperarml,
                    Tve_footr,Tve_footl,
                    Tve_legr,Tve_legl,
                    Tve_thighr,Tve_thighl,
                    Tve_trunk,
                    Tve_neck,
                    Tve_head])

    T_t_handr      = Tt_handr
    T_t_lowerarmr = Tt_lowerarmr
    T_t_upperarmr = Tt_upperarmr
    T_t_handl      = Tt_handl
    T_t_lowerarml = Tt_lowerarml
    T_t_upperarml = Tt_upperarml
    T_t_footr      = Tt_footr
    T_t_legr       = Tt_legr
    T_t_thighr     = Tt_thighr
    T_t_footl      = Tt_footl
    T_t_legl       = Tt_legl
    T_t_thighl     = Tt_thighl
    T_t_trunk      = Tt_trunk
    T_t_neck       = Tt_neck
    T_t_head       = Tt_head

    update_tar_tve(timestep, t, T_ar_handr, T_ar_handl, T_ar_lowerarmr,
                   T_ar_lowerarml,T_ar_upperarmr, T_ar_upperarml, T_ar_footr
    ,
                   T_ar_footl, T_ar_legr, T_ar_legl,T_ar_thighr, T_ar_thighl
    ,
                   T_ar_trunk,  T_ar_neck, T_ar_head, T_ve_handr,T_ve_handl,
                   T_ve_lowerarmr, T_ve_lowerarml, T_ve_upperarmr,
                   T_ve_upperarml, T_ve_footr,T_ve_footl, T_ve_legr,
    T_ve_legl,
                   T_ve_thighr,T_ve_thighl, T_ve_trunk, T_ve_neck, T_ve_head
    ,
                   tar_handr, tar_lowerarmr, tar_upperarmr, tar_handl,
                   tar_lowerarml, tar_upperarml , tar_footr, tar_legr,
                   tar_thighr, tar_footl, tar_legl,tar_thighl,tar_trunk,
                   tar_neck, tar_head, tve_handr, tve_lowerarmr,
    tve_upperarmr,
                   tve_handl,tve_lowerarml,tve_upperarml, tve_footr,
    tve_legr,
                   tve_thighr, tve_footl,tve_legl, tve_thighl, tve_trunk,
                   tve_neck, tve_head)
```

Listing 5.50: Calculating and redefining tissue, arterial and venous temperatures.

Arterial and venous blood temperatures are updated as vectors, but temperatures of tissues are saved in files, one per body segment, in steps of 10 seconds. Of course this value is to be settle for each kind of simulation according to user needs. Here, to exemplify, in 100 seconds of simulation leads 10 files per body segment plus the initial (zero second) condition (listing 5.51).

```
    savestep=step*5+5

    if (savestep/10. == int(savestep/10.)):
        save_T_t(T_t_handr, T_t_lowerarmr, T_t_upperarmr, T_t_handl,
                 T_t_lowerarml, T_t_upperarml, T_t_footr, T_t_legr,
                 T_t_thighr, T_t_footl, T_t_legl, T_t_thighl,
                 T_t_trunk, T_t_neck, T_t_head, savestep)
        print savestep +'s'
```

Listing 5.51: Save tissue temperatures `if..:` cycle.

Last few lines of code are dedicated to visualize the data. As the three-dimensional data used to represent body segments require a large amount of memory, the incorporation in real time of a visualization process reveals to compromise the processing time. The alternative was split the program into visualization and processing/simulation parts.

Visualization routine can be run apart or, if required by the used, after the simulation with a 'Y' in the prompt. As animation of tissue temperature variable can be seen one at a time, loop cycle presents the tissue temperatures data one after the other according to the preferences of the user (listing 5.52).

```
ans = raw_input("Visualize the data T_ar/T_ve? 'Y'/'N'...")

if ans=='Y' or 'y':
    fig_bl(tar_handr, tve_handr, 'R. Hand')
    fig_bl(tar_lowerarmr, tve_lowerarmr,'R. Forearm')
    fig_bl(tar_upperarmr, tve_upperarmr, 'R. Arm')
    fig_bl(tar_handl, tve_handl, 'L. Hand')
    fig_bl(tar_lowerarml, tve_lowerarml, 'L. Forearm')
    fig_bl(tar_upperarml , tve_upperarml, 'L. Upperarm')
    fig_bl(tar_footr, tve_footr, 'R. Foot')
    fig_bl(tar_legr, tve_legr, 'R. Leg')
    fig_bl(tar_thighr, tve_thighr, 'R. thigh')
    fig_bl(tar_footl, tve_footl, 'L. Foot')
    fig_bl(tar_legl , tve_legl, 'L. Leg')
    fig_bl(tar_thighl, tve_thighl, 'L. thigh')
    fig_bl(tar_trunk, tve_trunk, 'Trunk')
    fig_bl(tar_neck, tve_neck, 'Neck')
    fig_bl(tar_head, tve_head, 'Head')

ans2 = raw_input("Visualize the data T_t? 'Y'/'N'...")

while ans2 =='Y' or 'y':
    name = raw_input("Input segment name...")
    time = raw_input("Input simulation time...")
    anim_T_t(T_t, steps)

    ans2 = raw_input("Visualize the data? 'Y'/'N'...")

raw_input("Press ENTER to end...")
```

Listing 5.52: Data visualizations data cycles with prompt interaction.

As present model admits distinguished values to each part of the body and new boundaries at each stage of calculus, new features will provide adapted responses to asymmetrical exposures, whether they are environmental or the exercise, during transient states of exposure.

5.3 Tests and Results

As mentioned, Penne's Bioheat equation it was built to be a model based on a single equation. However as it is known today, the thermal properties of the several tissue layers are significantly different from each other, so it was important to consider that differences by using different coefficients for each tissue layer at the different areas of the body.

To test the results it is not only important to observe if the model respond to the different boundary and initial conditions, but also if it considers the correct values according to the spacial distribution of the tissues.

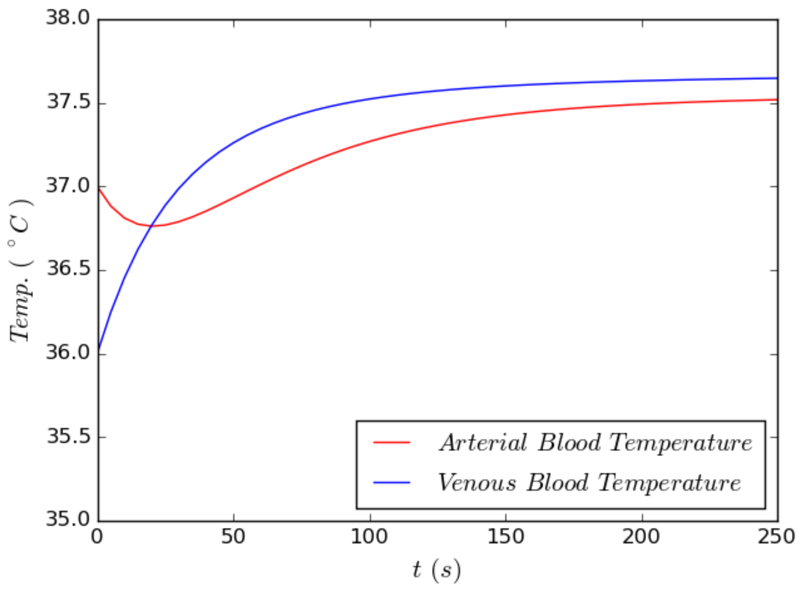
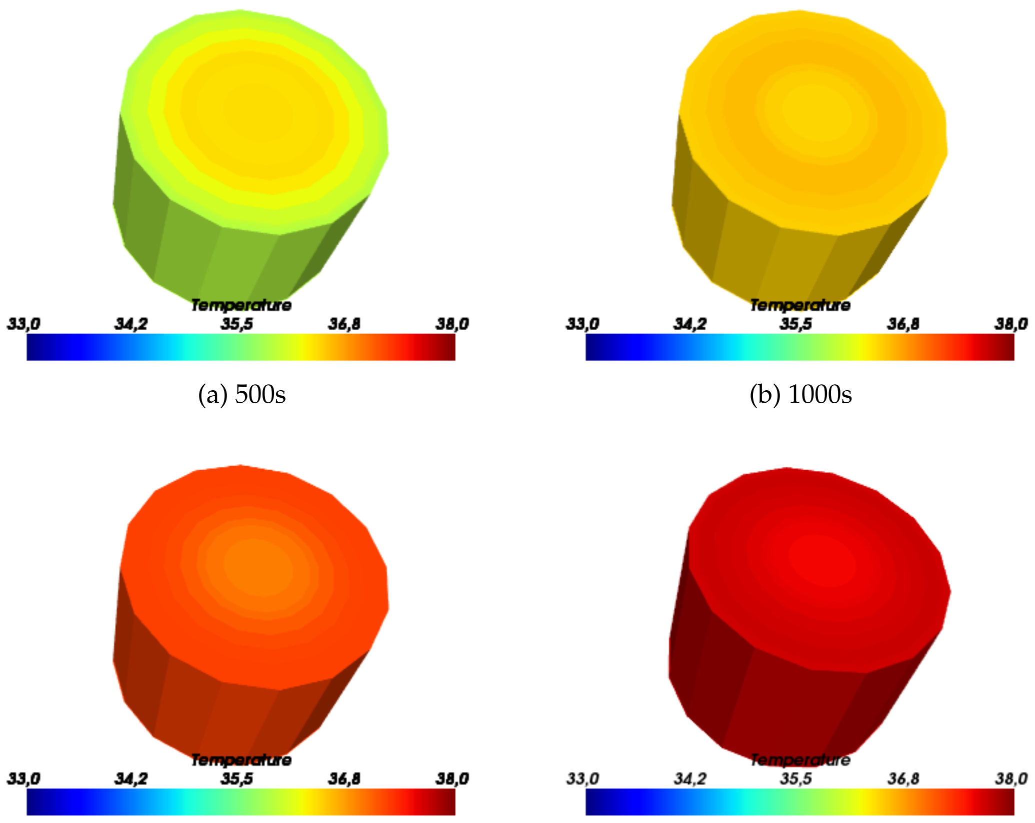


Figure 5.8: Arterial and venous temperatures at a 500s simulation.

To test the sensibility of the model values of each variable were changed, one at a time, remaining the other variables constant. The process was repeated to all variables in order to assure that the model responds according the laws of physics of heat diffusion.

For example, to test sensitivity to the thermal properties (such as specific weight, specific heat and thermal conductivity) values were changed, one at the time, for each layer of tissue remaining the other variables constants.

Following examples presented by figures 4.5 and figure 5.8 show the outputs of the neck mesh of the model. With no boundaries , the equivalent to adiabatic frontier shell around the segment, temperature of the tissues all only collied by blood temperature that crosses the element. At the end blood temperatures at the reservoirs tend to stabilize. The temperatures of the tissues are cooler at outside and then start to heat leaving the cooler area inside the cylinder.



(a) 500s

(b) 1000s

(c) 2500s

(d) 5000s

Figure 5.9: Temperature distribution over 5000 seconds of simulation considering adiabatic frontier around the element.

These were the tests performed to check the behaviour of the individual body elements before scaling up to the final whole-body thermal model.

5.3.1 BioHeatSIM's Tests Description

A sequence of tests were chosen to check the behaviour and the ability to respond to the main goals, specifically a model that was able to respond to three-dimension heat diffusion in non homogeneous transient environments. Performance tests can be divided, considering model features, in:

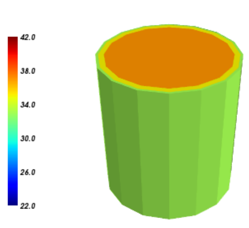
1. stability test at thermoneutral steady-state, homogeneous conditions;
2. response to steady-state, hot and cold homogeneous environments;
3. response to steady-state, non-homogeneous conditions;
4. heat diffusion response to transient, homogeneous environments;
5. model response to transient, non-homogeneous conditions.

At first, a thermoneutral steady state conditions is settled. This first neutral condition allow to check the stability of the model at conditions were thermoregulation is not needed and gives the basal values for the comparison with the other performance tests results. Secondly, the tests to the changes in boundaries along the time. In the opposite to previous test, here values in space remain constant at each time instant, but they change over the time. Thirdly, a test to verify the effect of local heating. Here boundaries stay the same along the time, but values change in space. Fourtly a test that checks responses to homogeneous changes of environments, such as moving from a cold place to a hot place. Finally, the test for boundaries changing in time and space at the same time. This was a final purpose of the model, to respond dynamically to environmental changes and local discomfort problems. This are the mainly tests that should assure if the model works properly. However, test and accuracy levels tests, should also be considered before upgrade the BioHeatSIM version.

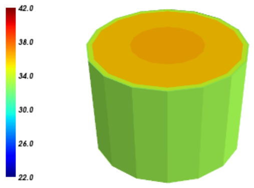
5.3.2 Summary of Main Results

All the tests were performed using the same initial conditions. All blood reservoirs start calculus from $37°C$. Distinct temperatures were settled for all layers of the body as shown the volumes and the respective cuts in figures 5.10 and 5.11. Values ranging from $33°C$ of the skin to $37°C$ at the inner bones and tissues.

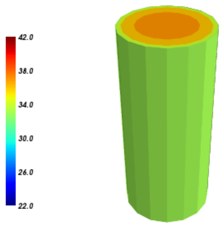
The places where boundary conditions are applied are constant and tend to represent the real situation, for example, hand and feet leave the smaller tops without bounds, corresponding to the wrists and ankles joints. Forearm, upper arm, thigh, leg



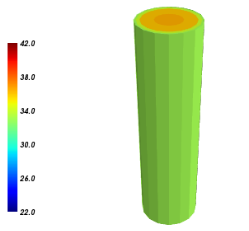
(a) Head



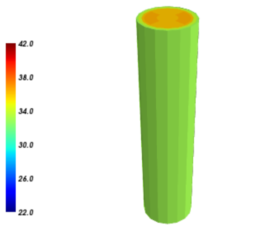
(b) Neck



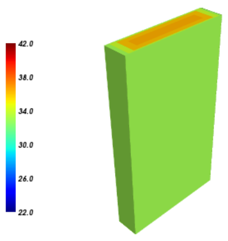
(c) Trunk



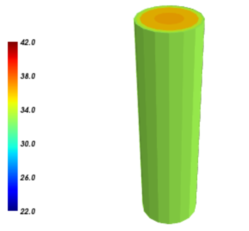
(d) Right Arm



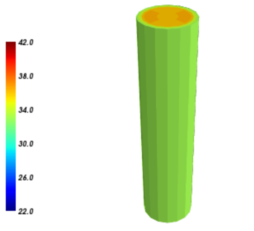
(e) Right Forearm



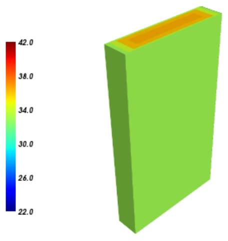
(f) Right Hand



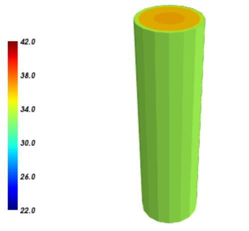
(g) Left Arm



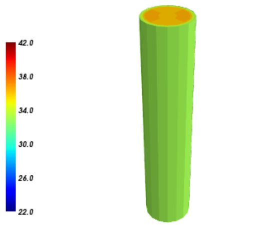
(h) Left Forearm



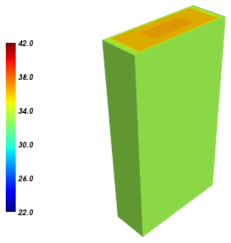
(i) Left Hand



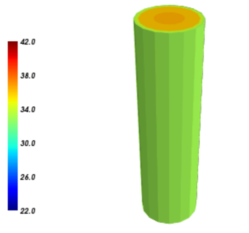
(j) Right Thigh



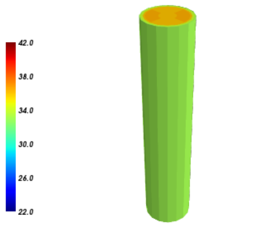
(k) Right Leg



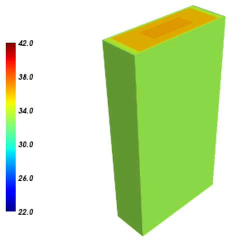
(l) Righ Foot



(m) Left Thigh



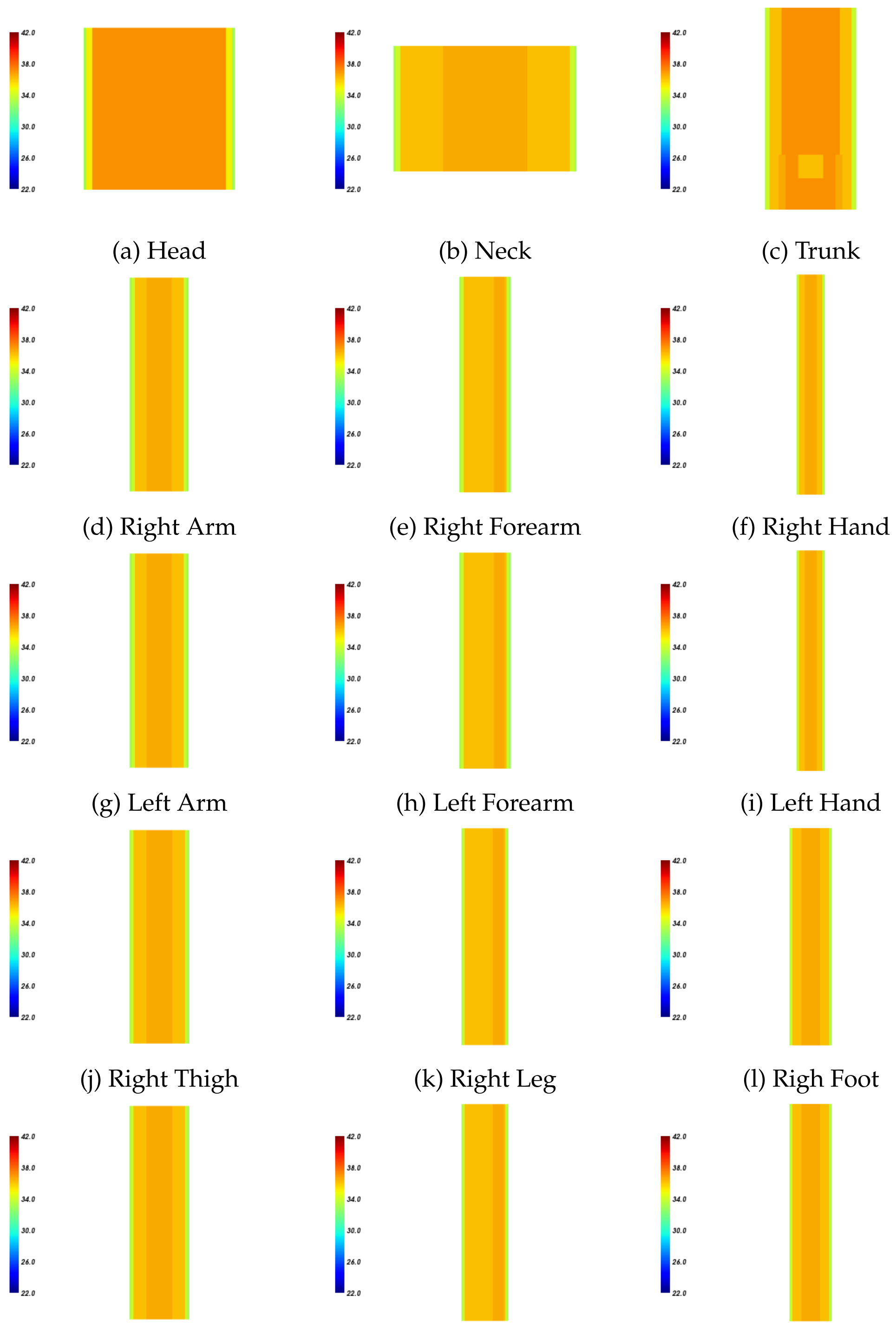
(n) Left Leg



(o) Left Foot

Figure 5.10: Initial conditions of body segments at t=0s. Isometric perspective.

and trunk are cylinders with lateral side bounded. The head is the only segment that allow heat exchange all around it. Section where bounds are not applied are modelled as adiabatic frontiers.



(a) Head (b) Neck (c) Trunk

(d) Right Arm (e) Right Forearm (f) Right Hand

(g) Left Arm (h) Left Forearm (i) Left Hand

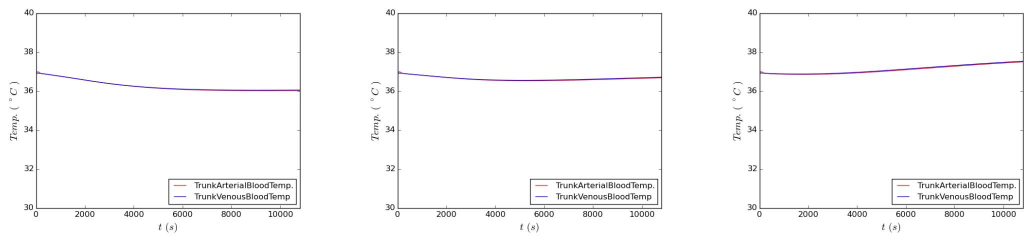
(j) Right Thigh (k) Right Leg (l) Righ Foot

(m) Left Thigh (n) Left Leg (o) Left Foot

Figure 5.11: Initial conditions of body segments at t=0s. Volume sections at z=0 plane.

Stability Test

General conditions of the model have been being presented along the deign and model implementation. To test the behaviour of the model at thermoneutrality several temperatures were tested. The passive system is considered stable when the core temperature, estimated from trunk's arterial reservoir (considered equivalent to the gold standard temperature *pulmonary artery blood*), remain stable at $37,0°C$.



(a) $28°C$ (b) $29°C$ (c) $30°C$

Figure 5.12: Temperature of arterial and venous reservoirs of the trunk for different constant, homogeneous boundaries.

Stability was found for a nude man, immersed in an high conductible infinity medium at $29°C$ (Figure 5.12. According to the model, core temperature rise up to $40°C$ in about two hours when boundaries are kept at $35°C$. But the results imply immersed head and static values for constant tissue blood perfusion for all layers. Fiala et al. (1999) already used a planed matrix with distinct values of blood perfusion at the skin in the different segments even at thermoneutral conditions. Present model use constant values for all elements, but it is already prepared to accept distinct values for the different segments, for all thermal properties and blood perfusion rates.

The blood in the limbs present a temperature gradient from the trunk to extremities. Lower temperatures are observed at the hands and the feet. This segments are the smallest and have large area of contact with the exterior boundaries, so, are the firsts reaching stability. Along all the limbs higher values of arterial blood are found next to the trunk. Arterial temperature decrease along the limbs in particularly due to the heat exchange between arterial and venous reservoirs. Common to the limbs is also the fact that venous reservoirs have lower temperatures than the respective arterial pairs. This occurs due to the the cooled blood received from the tissues.

The head and the neck must be considered separately from the limbs because of the anatomy itself. The brain have peculiar specifications comparatively to the rest of the biologic tissues. It produces huge amounts of heat and blood perfusion is the highest found in the human body. Dissipation of the heat generated inside the head is done,

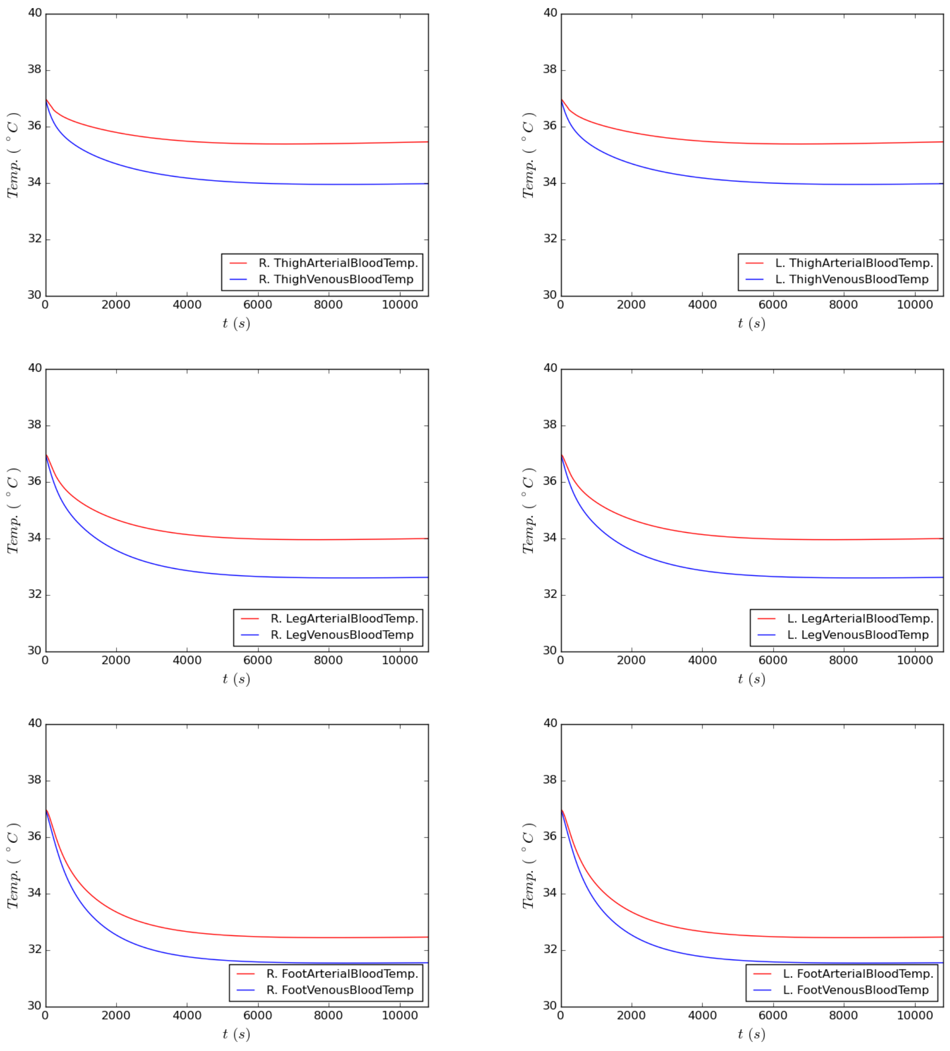
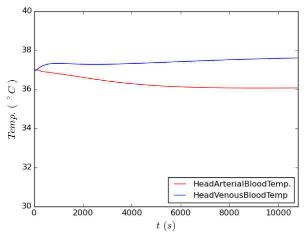
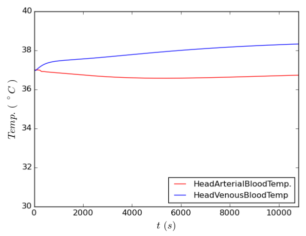


Figure 5.13: Temperature of arterial and venous reservoirs of the legs for constant, homogeneous boundaries of 29°C.

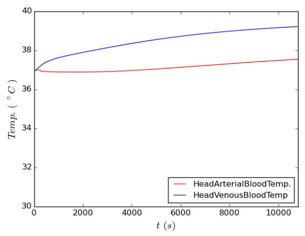
mainly, by the blood. Considering the absolute values, in thermoneutral environments the brain must remain between normothermic values. As suggested by Raimundo and Figueiredo (2009) hypothalamus admits normal thermal regulations between the 34°C and 39°C degrees. Behind this occurs hyperthermia with introversion and violent sweating , heat stroke (stop sweating and fainting) when temperature exceeds the 41°C, permanent brain damage after 42°C and reaching the 44°C degrees or higher death. To check the reliability of the available methods to access core temperature Fulbrook (1997) found that tympanic temperature can be until $1,2$°C above and $1,3$°C below the pulmonary artery blood temperature, considered the gold standard. This differences also rise the hypothesis that core blood have different temperature of the brain. Here, for 29°C the temperature of the brain rises till 38,32°C degrees, what is slightly above of the expected, but still considered normothermic value.



(a) 28°C



(b) 29°C



(c) 30°C

Figure 5.14: Temperature of arterial and venous reservoirs of the head for different constant, homogeneous boundaries.

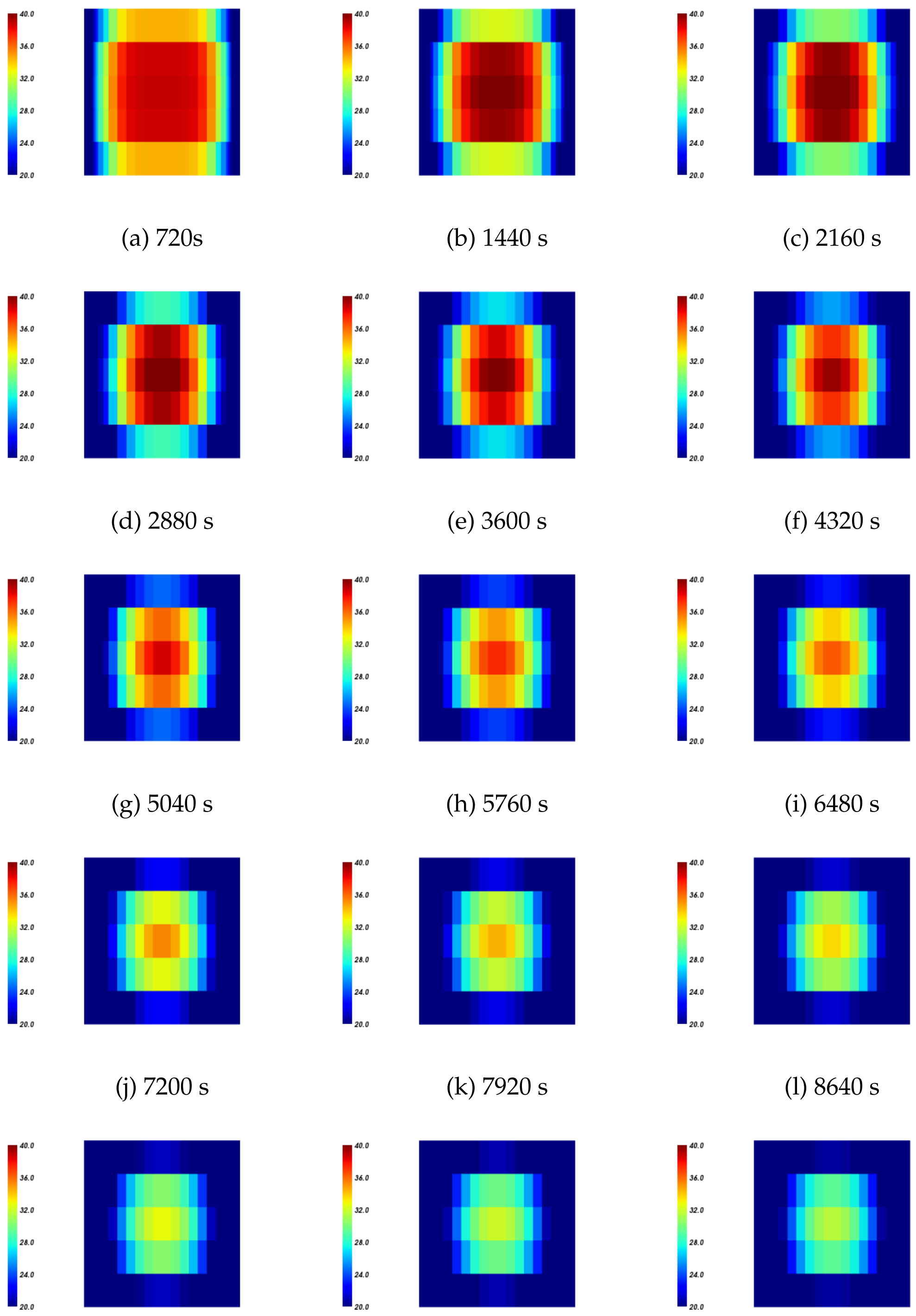
Finally, in the opposite to the limbs, venous blood temperature is significantly higher comparatively to the arterial blood. And as the neck is the segment that connects the trunk to the head, it follows the same temperature profile of the head considering slightly lower values.

Constant Exposures

Steady-state homogeneous environments were tested for cooling and heating. For cooling test, boundaries were settled at 10°C for all segments for the 3 hour period.

The cooling temperature profiles of the head show the huge amount of heat produced by the brain. Without considering the active system most of body segments reach values below 22°C in the first hour of exposure however, due to the large quantity of heat produced by the brain, the head takes more than 3 hours (figure 5.15). By observing the interior temperature matrix of the head we can see that heat is diffused in radial direction faster than diffused in axial direction. This is because of the physical structure of the head that only considers the shell of bone, muscle, fat and skin around the lateral of the cylindrical geometry. Core of brain, in the middle of the cylinder is the hottest point of the entire mesh and is the last to cool down.

Three-dimensional heat dissipation, in hands and feet, can be verified. Temperature gradient along the limbs, as in the stability test for symmetric initials and bound-



(a) 720s (b) 1440 s (c) 2160 s

(d) 2880 s (e) 3600 s (f) 4320 s

(g) 5040 s (h) 5760 s (i) 6480 s

(j) 7200 s (k) 7920 s (l) 8640 s

(m) 9360 s (n) 10080 s (o) 10800 s

Figure 5.15: Temperature decrease of the head over 3 hours of exposure at $10°C$.

aries, is symmetrical. Extremities are the first to cool down and warm up. Trunk's furthest segments have arterial and venous temperatures lower than the closest segments.

The same test was repeated for different boundaries, respectively $22°C$, and $40°C$. Results heating and cooling profiles were similar in all the tests. The comparison of core temperature for each test shows that: the temperature gradient is as much higher ,as further are the boundaries from the thermoneutral value of the model (figure 5.19).

Local Heating

Local heating intend to test the model no non-homogeneous mediums. Here initial conditions remain the standard, presented in the previous tests. Boundaries were changed for right arm and leg, settling their temperature to $40°C$. The rest of the body was kept at $22°C$. In a total of 3 hours of simulation could be seen that the limbs respond to the respective bounds. Influence of the heated limbs can be seen in the final arterial and venous temperatures of the trunk and, once again, body extremities such feet and hands are the first reaching stable results.

Figures 5.17 and 5.18 show the gradual heating over the period of 3 hours. The rest of the body reacts to the constant $22°C$. The difference from the constant homogeneous exposure at $22°C$ can be check from figure 5.19 where the body core temperature (equivalent to arterial temperatures of the trunk), reacts to local heating by slight increase in final values.

Transient Environments

Transient test checked model response to changes in boundaries. It was simulated 3 hours considering 3 different homogeneous conditions, one per hour. During the first hour the body was kept at $35°C$, then heated at $40°C$ and cooled at $30°C$.

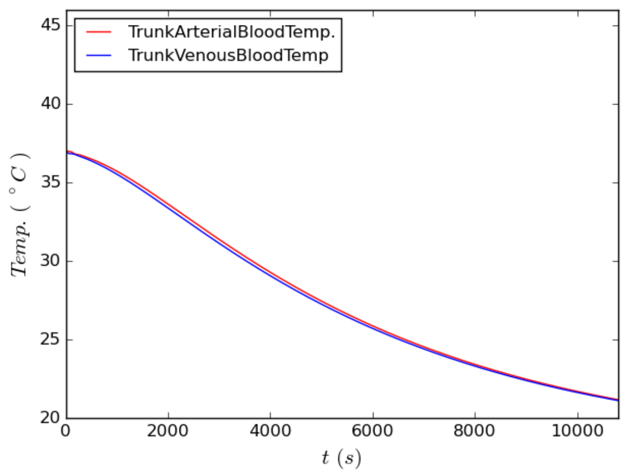
The test intend to show the different profiles of heating and the cooling (figure 5.20). All the segment heat slower in the first hour, at $35°C$ temperature, than at the second hour, at $40°C$. The cooling of 1 hour at $30°C$ it was not enough to cool core temperature at the initial values.

From the blood temperature reservoirs it is possible to check the precise moment of the change in boundary values in all segments. Temperature profiles next to the trunk show softened differences comparatively to the extremities. Once again the head presents the higher temperatures.

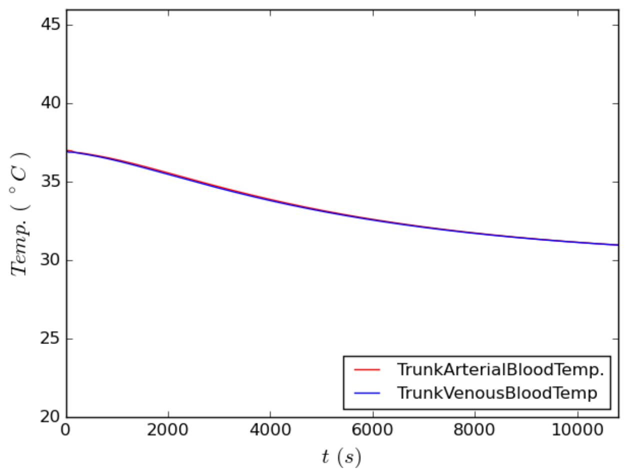
Transient and non-homogeneous Environments

For this tests body segments were alternately exposed at temperatures changing between $22°C$ and $40°C$. Changes were settled at the end of each hour to enhance the possibility to compare the results with the other results. So while the right arm and leg were at $40°C$ the left arm, leg, trunk and head reamain at $22°C$. After one hour of exposure the boundaries were changed so right arm and leg would be at $22°C$ and the left arm, leg, trunk and head raised to $40°C$. After another hour the boundaries switch again.

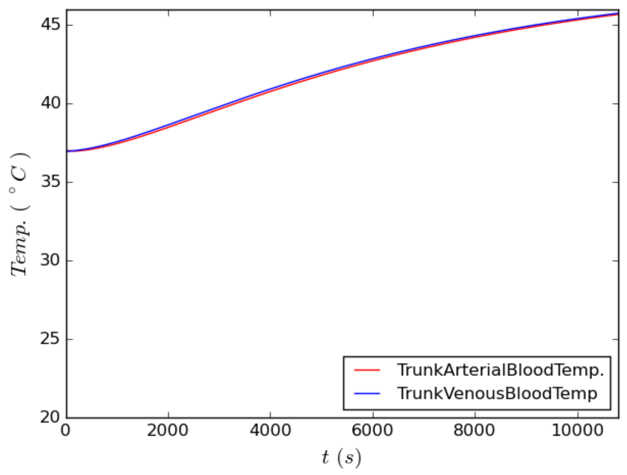
The results show that all body segments respond individually to the boundary conditions imposed (figure 5.21). The rising curves show the increase of temperature and the downward curves revels the decrease in temperatures. Final core body temperature response shows that is more sensitive to changes in the bigger and the most vascularized segments.



(a) $10°C$

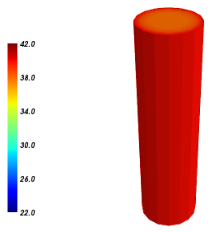


(b) $22°C$

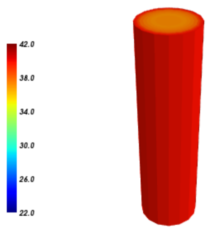


(c) $40°C$

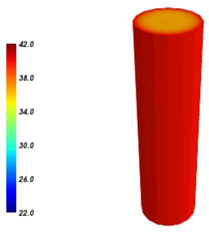
Figure 5.16: Arterial and venous temperatures of the trunk considering different types of boundaries.



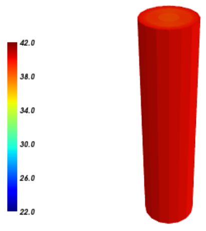
(a) Arm 3600s



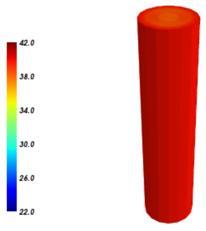
(b) Arm 7200s



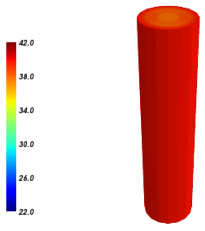
(c) Arm 10800s



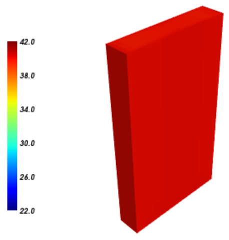
(d) Forearm3600s



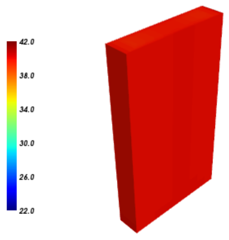
(e) Forearm 7200s



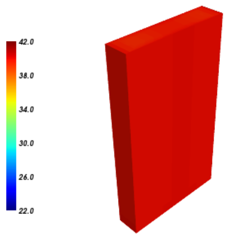
(f) Forearm 10800s



(g) Hand 3600s

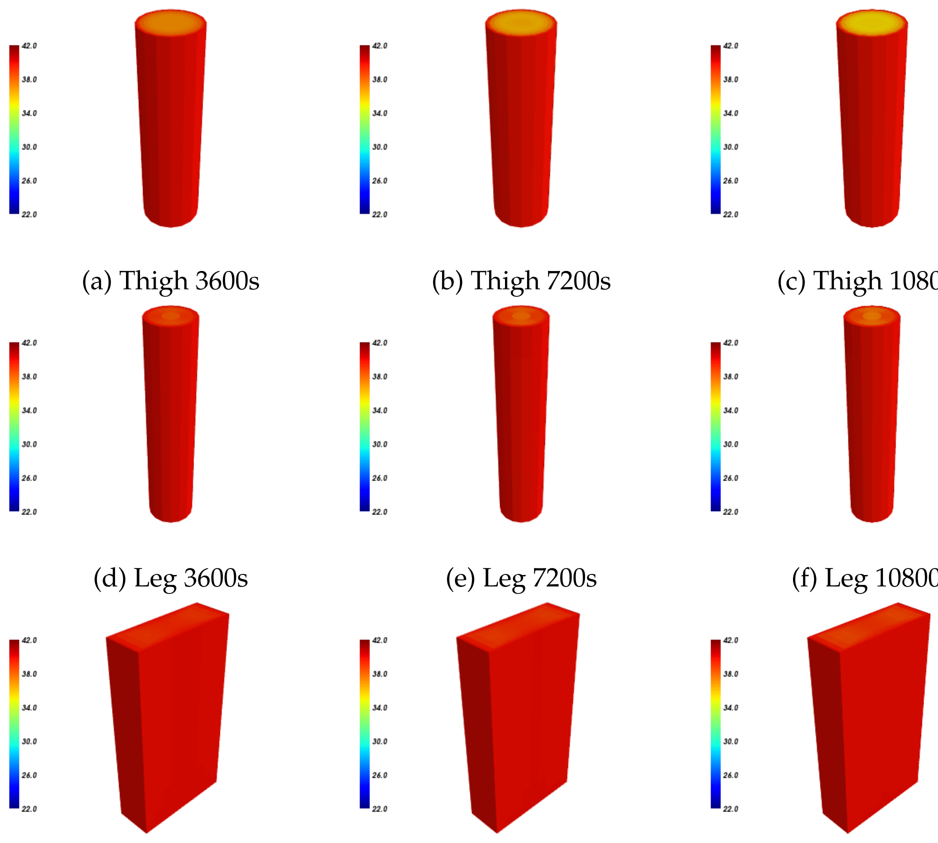


(h) Hand 7200s



(i) Hand 10800s

Figure 5.17: Right Arm heating over 3 hours of simulation being heated at $40°C$.

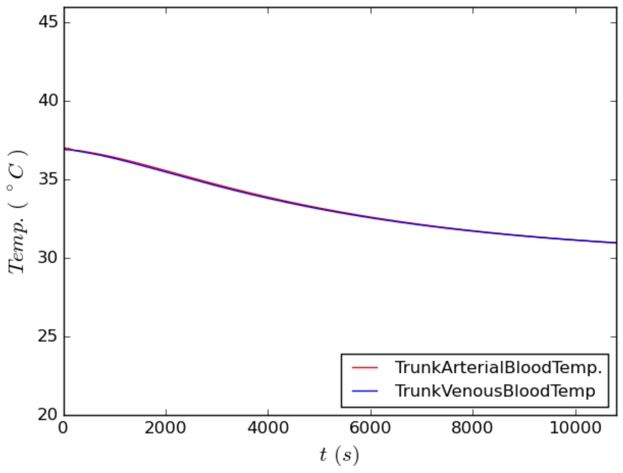


(a) Thigh 3600s (b) Thigh 7200s (c) Thigh 10800s

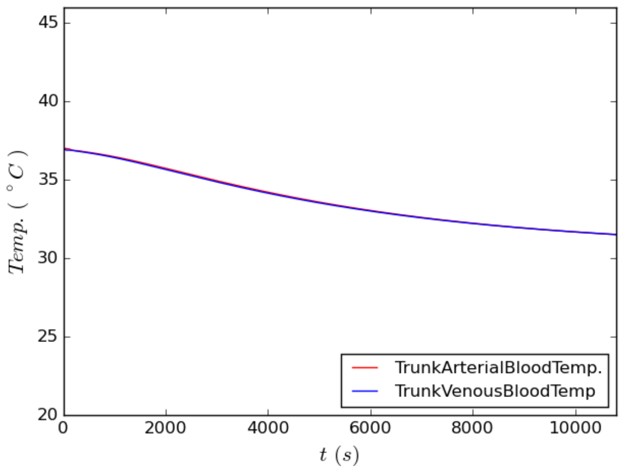
(d) Leg 3600s (e) Leg 7200s (f) Leg 10800s

(g) Foot 3600s (h) Foot 7200s (i) Foot 10800s

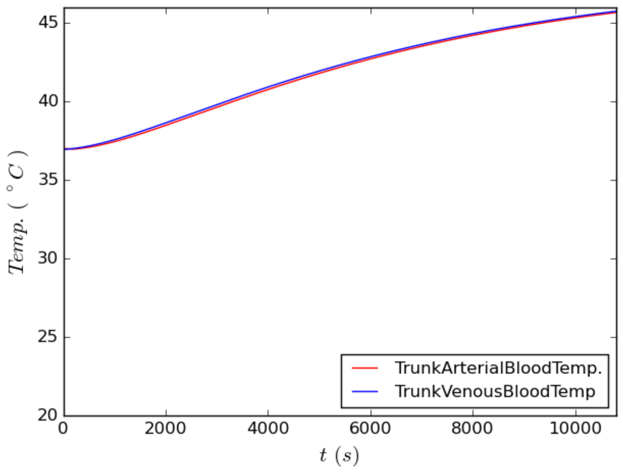
Figure 5.18: Right Leg heating over 3 hours of simulation at 40°C.



(a) All segments with the bounds constrained at 22°*C*



(b) Right Arm and Leg at 40°*C*, other segments at 22°*C*



(c) All segments with bounds constrained at 40°*C*

Figure 5.19: Arterial and venous temperatures of the trunk considering different types of boundaries.

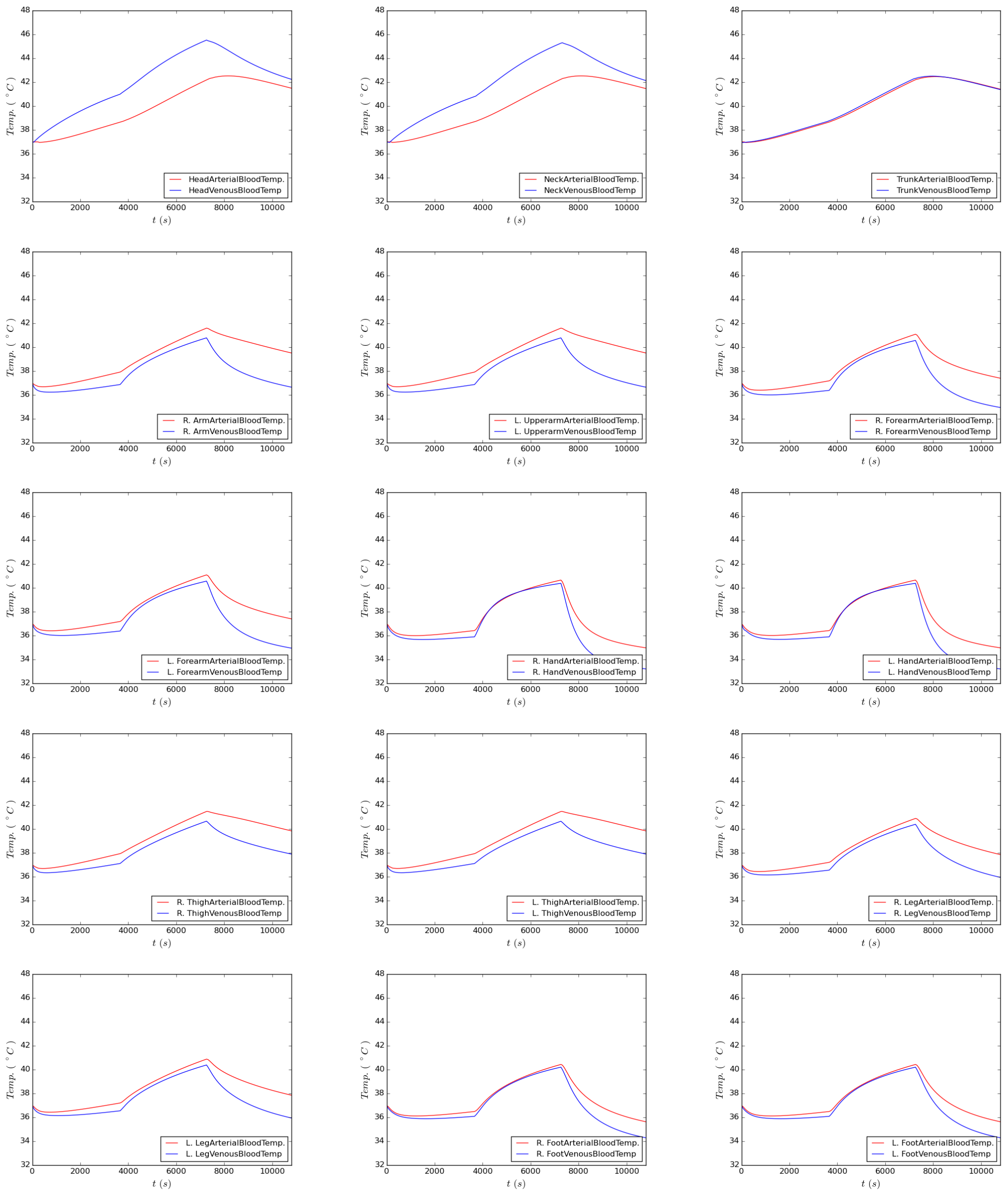


Figure 5.20: Arterial and venous temperatures for all body segments during a transient test. All segment constrained the boundaries at 35°C during first hour, at 40°C over the second hour and at 10°C in the last hour.

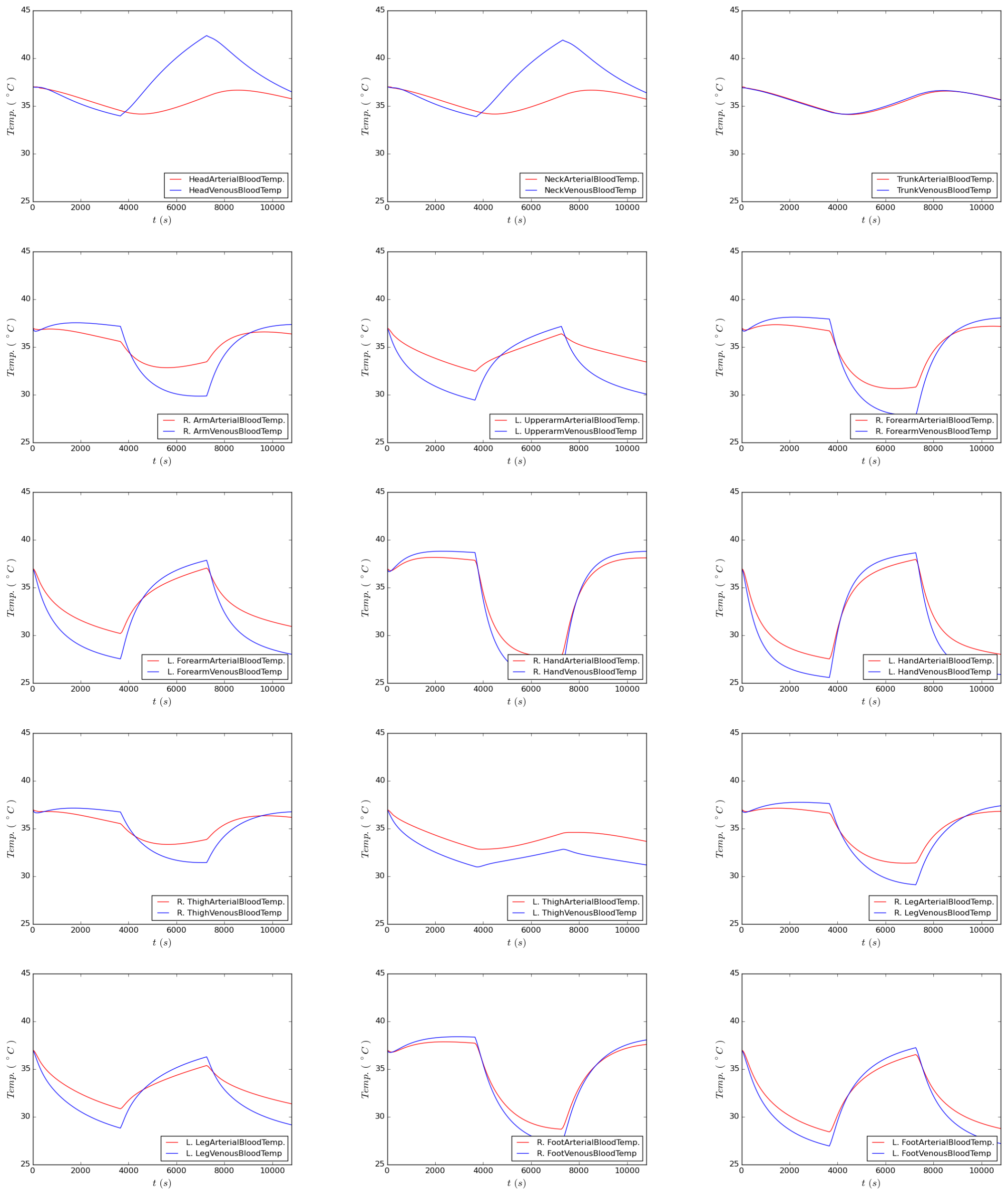


Figure 5.21: Arterial and venous temperatures for all body segments during a transient test at non-homogeneous conditions. Boundary conditions switch every hour from $22°C$ to $40°C$ or contrariwise.

Chapter 6

Conclusions

Final tests only allow to see if responds to changes in variables, but agreement with real data in terms core temperatures at thermoneutral conditions must be considered. the boundary that satisfies thermoneutrality at resting is 29ºC. Resting thermoneutrality tests in water immersion, round the $33,0°C$ (Reilly et al., 2003) to $35,0°C$ (Nakanishi et al., 1999). But even water immersion condition cannot be considered similar to the simulated perfect constant boundary imposed in the model. All experiments relate non significant physiological changes and assure core temperature to remain constant around $37°C$, but there are some adaptations, such as water loss, slightly vascular reactions and heart rate decrease, that still occur. In particular the heat exchanges between skin and surrounding environment should be included when crossing the results with the real data. Values from 29 to $33°C$ are considered thermoneutral under dynamic exercise in water immersion (Nakanishi et al., 1999) what is very closed to the values found by the model. New hypothesis could be raised that could justify that differences. The values found in literature and used in the model can be partially inadequate, or outdated. Another possibility is the time lag effect imposed by the connective tissue that, as suggested in local bioheat modelling, could be considered in modelling process. Even so, it is possible to state that BioHeatSIM is a multi-segment model prepared to respond to transient and asymmetric conditions.

Actual version does not allow interactivity in the parametrization of: physical body segments, time, type of exposure or boundaries. This can be considered as a disadvantage, however, all elements are settled in specific routines, are easy to found and all can be replace by the desired values.

Implementation of program in Python revels to increase the complexity of the model and parametrization dealing with complex cell variable objects in a very simple and organized way that allow to increase dramatically the level of detail comparatively to previous models. BioHeatSim was tested considering more than 6000 thousands of nodes (6186) what is much different from the 225 nodes of Wissler's models and even from the 386 nodes of Fialas' first model . The memory requirements were not compatible to real time animation of temperatures in physical structure, but this was perfectly overcome by separating the visualization of the data from the simulation.

Actual version of the program allow to make 3 hours of simulation, considering a 60 seconds step in less than 1 hour in a Pentium i7 running at 3.6 GHz using 4Gb of RAM in a 64bit version of Python2.7 using Ubuntu operative system. It must be said that the presented version of BioHeatSIM did not reach the limit of potential of Python capabilities. This time can be shorten and model complexity can be increased. It is proved that new language such as Python bring new possibilities to the to increase predictive power of the biological heat simulation.

Chapter 7

Currently Developed Work

Passive system model was designed thinking in future features and upgrades. So, programming and mathematical modelling decisions were made based on future implementation of the following suggestions for active system modelling. Innovation is here is focused on detailed cardiovascular modelling and the implementation of feed-forward loops to activate thermoregulation mechanisms.

7.1 Active System Modelling

In general active system can be seen as the central nervous system of the model that act in negative feedback according to the physical thermal state. As in human biology HBTM model try, with limited capabilities, to maintain homoeothermy of the system. Sweating and vascular responses are the most important thermoregulator mechanisms in hot environments.

Sweat response tried to incorporate a new approach that is being explored recently, that include a double response of the effector for sweating not only a negative feedback response (to T_{cr}) but also including a feedforward reaction to environment (considering T_sk). Modelling decision and predictability is still being refined but results are more interesting

To enable future high quality upgrades, in particular for modelling response to prolonged and critical heat exposures during mild and heavy exertion, a complete cardiovascular system is suggested. With the wide informations about cardiopathophisiology, it can also be adjusted to predict responses for populations with special needs. To fight against hot environments during exercise, cardiovascular modelling is crucial for many reasons. One is because the first response to heat consists in vasomotor reaction, this is, significant increase in skin blood flow due vasodilation. However this new adapted distribution have many cardiac implications such as the need for redistribute the blood and adjust mean arterial temperature.

This features not only allow to have others heat stress indicators to monitor, as enable to adjust a wide range of new variables to increase sensibility and adapt to new situations.

7.1.1 Circulation as input of Large and Small Vessels Blood Flow

Once the heat exchange in the passive system is complete, it is important to assure accurate values for the input variables. As the main structure of all heat exchange is the quantity of blood that cross the tissues, it is proposed by the present work an integration with a parallel model of the circulatory system. This consists in predict from trivial parameters (such as heart beat rate, and mean cardiac output) the pulsating velo-

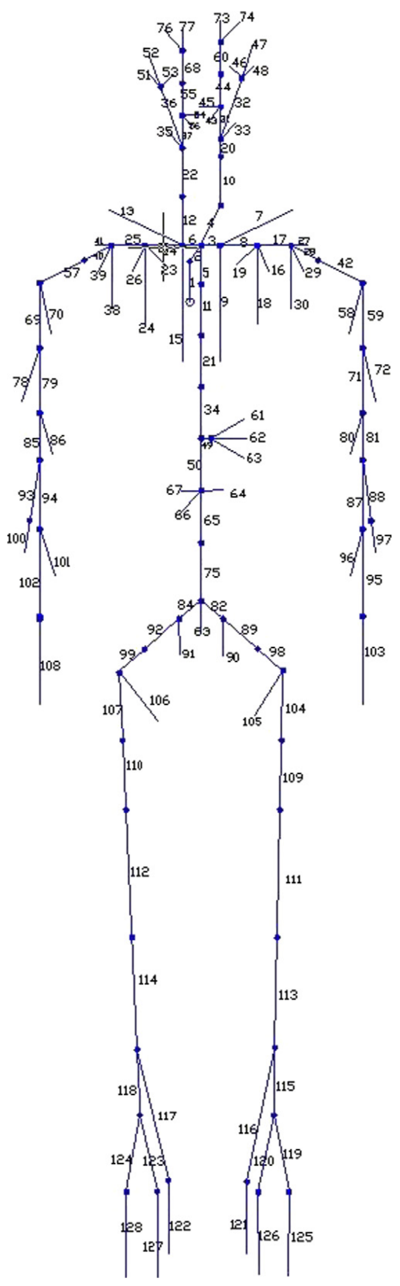


Figure 7.1: Human arterial tree system representation.

city of the blood going through arteries across the fifteen body segments. Venous blood does not suffer the effect of the heart pump. Blood flow in the veins are determined by the mean velocity of the blood flow in the corresponding artery (Salloum et al., 2007; Karaki et al., 2013).

The arterial multi-branched model, from Avolio (1980), was one of the most important representations of the circulatory system in terms of accurate physic properties. He implemented a routine to calculate the impedance along a set of arterial branches. The outcome impedances allow to calculate velocities, pressures and find cardiac outputs at each part/segment of the human body considering the realistic pulsating property of blood delivery.

The Avolio's model divide the arterial tree in 128 vessels with symmetrical distribution on right and left body parts. Figure 7.1 shows the distribution of the arterial tree as designed by Avolio (1980) and as implement to adjust the inputs along the body by linking all the elements by predicting the radii of the arteries and veins, and estimating the velocity of blood in each segment Salloum et al. (2007). These subdivision, based on real anatomy of the human body, was selected to keep simplicity, however assuring good agreement with real results.

7.1.2 Mean Cardiac Output Prediction

There are two main influences between basal Cardiac Output and the one reached in real conditions. The CO does not only depends on the Heart Rate but also from the thermal load of the environment itself. Karaki et al. (2013) relate the CO values, in agreement with Fu (1995), as depending on the human thermal sensors in the skin and core nodes. The authors define threshold mean values, stated experimentally, that allowed to establish equations 7.1 and 7.2, as depending on skin (T_{sk}) and core(T_{cr})

temperatures.

$$CO_{con} = \begin{cases} 290 \times 10^3 & if\, T_{sk} \geq 33,7°C \\ (T_{sk} - 10,7) \cdot \dfrac{290 \times 10^3 - 270 \times 10^3}{33,7 - 10,7} + 270 \times 10^3 & if\, 10,7°C < T_{sk} < 33,7°C \\ 270 \times 10^3 & if\, T_{sk} < 10,7°C \end{cases} \tag{7.1}$$

$$CO_{dil} = \begin{cases} 427,5 \times 10^3 & if\, T_{cr} \geq 41°C \\ (T_{cr} - 37,2) \cdot \dfrac{508 \times 10^3 - 400 \times 10^3}{41 - 37,2} + 400 \times 10^3 & if\, 37,2°C \geq T_{cr} < 41°C \\ (T_{cr} - 36,8) \cdot \dfrac{427,5 \times 10^3 - 290 \times 10^3}{37,2 - 36,8} + 290 \times 10^3 & if\, 36,8°C < T_{cr} < 37,2°C \\ 290 \times 10^3 & if\, T_{cr} \leq 36,8°C \end{cases} \tag{7.2}$$

This way it is possible relate CO with the thermal load of the body, as it happens physiologically in healthy human beings. Final cardiac output can be found by the product of dilated cardiac output (CO_{dil}) and constricted cardiac output (CO_{con}) divided by the basal cardiac output of $290 \times 10^3\, cm^3.h^{-1}$ (eq. 7.3) (Fu, 1995; Karaki et al., 2013).

$$CO = \frac{CO_{dil} \times CO_{con}}{290 \times 10^3} \tag{7.3}$$

The values 290×10^3 can be used as the mean cardiac outup ($\bar{Q}$) to normalize the cardiac ejection waveform in rest.

The CO value achieved by this mathematical formulation includes the physiological response to thermal load induced by the environment, but does not includes the effect of exercise. This allow the CO values to ranged from a minimum of $270 \times 10^3\, cm^3.h^{-1}$, during maximum vasoconstriction, to a maximum of $427,5 \times 10^3\, cm^3.h^{-1}$, over the maximum vasodilation, being the $290 \times 10^3\, cm^3.h^{-1}$ the CO for the condition of thermoneutrality (Karaki et al., 2013).

A generic model, also based on Fu (1995), is adopted by Salloum et al. (2007) that integrates the ratio of metabolic expenditure as referred in equation 7.4.

$$CO = \frac{M_{total}}{M_{basal}} \cdot \frac{CO_{dil} \cdot CO_{con}}{CO_{basal}} \tag{7.4}$$

The actual metabolic rate (M_{total}) divided by the metabolic rate at thermoneutral resting conditions (M_{basal}) create a proportion of increase or decrease in the CO. In spite of the accuracy of this modelling proposal, present work consider constant Stroke Volume, what is a false purpose specially when modelling response during exercise. In health toyota yaris hybridhy subjects the increase in CO during exercise is the result of a combined increase in heart rate and SV (Salmasi, 1993). According to Salmasi (1993) there studies that prove that exercise have significant impact on cardiac output, ranging values from $6,6\,l.min^{-1}$, at rest, to $14,3\,l.min^{-1}$, during exercise, in supine position, and from $5,3\,l.min^{-1}$ to $13,8\,l.min^{-1}$. This differences are due to the enhanced contraction, and increase in the left ventricular end-diastolic pressure combined with a decrease in the left ventricular end-systolic pressure, that results in an increase of SV.

Brynjolf et al. (1983) found an increase of 14 % in the end-diastolic volume (EDV) and a decrease 14% in the end-systolic volume (ESV) pumped into the heart at each beat. Being the SV mathematically expressed by equation 7.5.

$$SV = EDV - ESV \tag{7.5}$$

In prolonged aerobic exercise human adaptation allows the increase in stroke volume, due to the reduction of resting heart rate. Reduced heart rate prolongs ventricular diastole(filling) increasing EDV.

Also, during heat stress conditions, even in euhydrated subjects, stroke volume may suffer a significant reduction. This reduction might occur due to a decrease in central blood volume and cardiac filling pressure that results from the rise in skin blood flow and volume (González-Alonso et al., 2000).

The definition of SV, as the quantity of blood pumped from the heart at each beat, results into the division of CO by the heart rate.

The values of stroke volume and and cardiac output also depend on individual characteristics. Variables such as hight, body weight, skin area and body composition and distribution influence the need for nutrients and oxygen under basal or work, thermoneutral or stress conditions. To compare performances between individuals cardiac index and stroke index are commonly used as cardiovascular indexes and are particularly important during heat stress exposure.

It is well known that acclimatization to heat brings important physiological adaptation that improve performance at work/exercise under heat exposures. An earlier onset of sweating, an increase of sweat rate and evaporative cooling, decreases the heart rate at the same time that reduces skin and core temperatures. Also special adaptation occurs at stroke volume level,which increases at the same proportion of the heart rate decreases, remaining a constant cardiac output (Rowell, 1974). According Rowell (1974) there are studies that prove the volume (before and after) acclimation process is quite similar. So, the major cardiovascular adjustments during heat acclimatization are reciprocal changes in heart rate and SV while CO and blood pressure remain essentially unaltered.

Physical shape, evaluated in terms of maximum oxygen uptake ($\dot{V}O_{2max}$), can also influence the distribution of blood flow and its relation. In cases where CO remain constant by heat stress during exercise, the raises of skin blood flow must came from the redistribution of CO in a way that the higher amount goes to the skin.

7.1.3 Cardiac Output Waveform

Modelling circulatory system is very important to accurately predict physiological response to heat stress. Circulation is the main responsible for heat dissipation and one of the systems that is more affected when individuals overheat and get dehydrated.

$$Q_1(t,n) = sin^n(\omega t) \tag{7.6}$$

$$Q_2(t,\phi) = cos(\omega t - \phi) \tag{7.7}$$

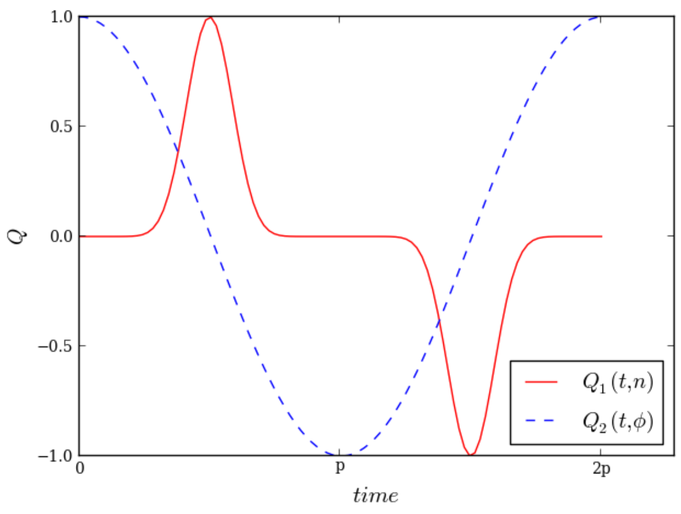
To predict the blood ejection to all body parts and parameters such as blood pressure and arterial impedance an input waveform mus be given. This wave form is defined in literature (Stevens et al., 2003) as a combination of overlap sine and cosine that approach the real wave accurately (equation 7.6, 7.7).

The Q represents blood flow, n an integer odd value, ω the heart pulse frequency, defined as π over the heart rate (b), and ϕ being a phase angle. The functions Q_1 and Q_2, plotted in Figure 7.2a, are the envelope and internal functions of the preliminary function $Q_3 = Q_1 \cdot Q_2$ (Figure 7.2b), whose outcome, after normalization and calibration results into the final waveform (equation 7.8) that is characteristic for each individual in any exertion state.

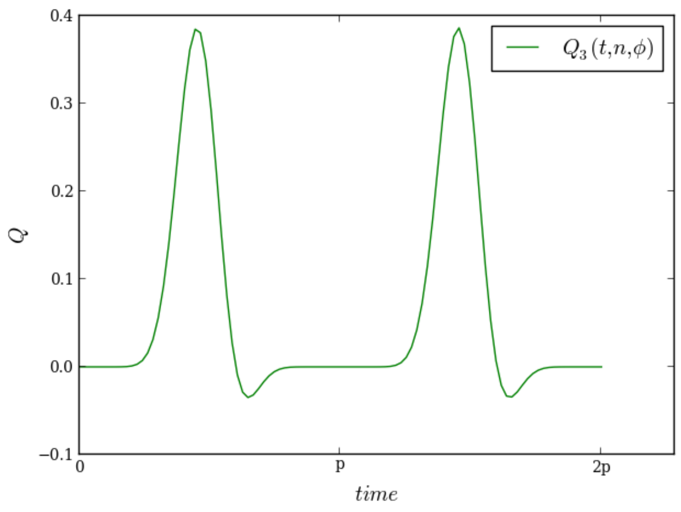
$$Q(t,n,\phi) = \frac{v}{A(n,\phi)} \cdot cos(\omega t - \phi) \cdot sin^n(\omega t - \phi) \tag{7.8}$$

Calibration and normalization consist in adjust frequency and peak values so the final waveform is able to represent a defined condition. In practice by settling the $\omega = \frac{\pi}{p}$, it can reached the desired frequency that depends on heart beat, once $p = \frac{1}{b}$. The outcome model $Q(t,n,\phi)$ is calculated from Q_3 with ω adjusted, by normalizing the function so that the outflow per period matches the output per stroke (v) that is equal to $\frac{\bar{Q}}{b}$. Calibration is made by adjusting mean cardiac output ($\bar{Q}$), heart rate beat (b) and peak to mean blood flow ratio (σ) obtained from the $\frac{Q_{peak}}{\bar{Q}}$.

Values like $\bar{Q}$ and Q_{peak} to determine σ, can be found in literature (Nichols et al., 1977; Murgo et al., 1980) for resting healthy people and people with cardiovascular



(a) Envelope Q_1 and interiorQ_2 functions of cardiac waveform for $n = 13$ and $\phi = 0$.



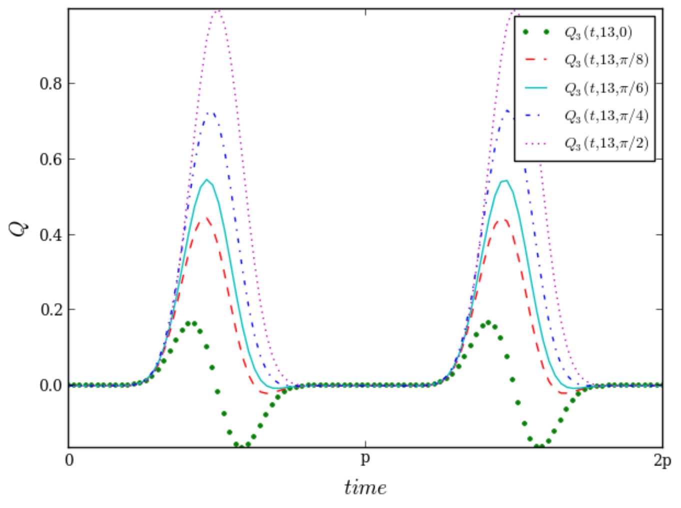
(b) Preliminary cardiac flow waveform Q_3 for $n = 13$ and $\phi = pi/10$

Figure 7.2: Functions to model the cardiac output waveform.

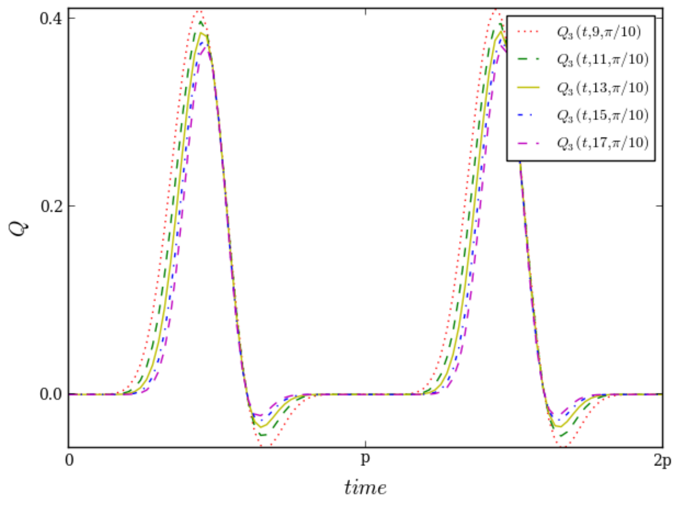
disease. However peak values Q_{peak} of aortic blood flow are more difficult to get for exercising (Mohiaddin et al., 1995), and it could not be found for environmental stress situations.

Figure 7.3 shows the differences in shape that results from phase ϕ and from chosen n.

The variable n is related to the relation of systole and diastole period in the cardiac cycle. According Stevens et al. (2003), standard texts in physiology agree that the systole periods approximately $\frac{1}{3}$ of the cardiac cycle, what means that $n = 13$. As the



(a) Shapes for different values of φ.



(b) Shapes for different values of n.

Figure 7.3: Influence of Q_3 variables in function's shape.

n value increases, smaller is the systole period. The variable φ is more difficult to define once it depends on the peak-to-mean flow ratio σ and, as mentioned above, common values are found for resting states. Equation 7.9 presents the definition of σ transformed into the equation that must be solved to found φ. The t^* is the time to peak, between 0 and p, i. e. the time to reach the maximum value of Q in one cardiac cycle. The value of t^* can be found from the differentiation of the Q function in order to time (equation 7.10), in other words the maximum and minimum values of CO function. The algebric expression that result from isolating t after differentiation (equation 7.10), is divided in four parts, being the final result selected considering that $Q > 0$ and $\frac{d^2Q}{dt^2} < 0$.

$$\sigma = \frac{Q(t^*, n, \phi)}{\bar{Q}} \tag{7.9}$$

$$\frac{dQ}{dt} = 0 \tag{7.10}$$

The value of t^* to which ϕ need to be calculated, greatly depend on heart rate h, what means that in spite of the considerations of Stevens et al. (2003), that said that the final value of ϕ depends on n but not on h, but in fact are related to each other in order to make $\bar{Q}$, ϕ does depend on it indirectly.

7.1.4 Cardiac Output Distribution

Cardiac output distribution along the body depends on several factors. Here were considered the two most important variables: the *thermal stress* and *metabolic stress*. Cardiovascular adjustments, during heat stress, assure thermal homoeostasis. The increase in skin blood flow allows an improved radiant heat loss, however to perform work oxygen and nutrients must be supplied to the body, and during exercise muscular blood supply is a priority.

Even at rest there are several physiological adaptations that occur in organism in order to maintain homoeostasis. It is vital that minimum volume of splanchnic blood is ensured. For example, when arterial mean pressure decreases below 60 $mmHg$, it is difficult to assume minimum oxygen values for vital organs, such as liver, which is usually the first main organ suffering from ischemia. However during heat exposure, most blood flow reduction is due to splanchnic vascular resistance and not for the reduction in the arterial mean pressure.

A linear regression was found by Rowell et al. (1965) describing a linear relation between the percentage of $\dot{V}O_{2max}$ and the percentage of resting splanchnic blood flow

(SBF) at different dry bulb environmental temperatures and low levels of humidity, that can be directly applied to the present model.

7.1.5 Sweat Rate Modelling

The sweat rate modelling is not a new research field, however the usual models predict amounts of sweat loss over a time period (usually the models based on heat balance equations) and do not relate it with the physiological phenomena of active thermoregulation and exercise. It is known that sweating is triggered by central nervous system when body temperature rises, but disturbs of nervous system (such as stress, anxiety or hormone fluctuations) might also stimulate sweat glands. There are two types of sweat glands: apocrine and eccrine. They perform different tasks but, when body heats, both are stimulated. Their type and concentration depend on body part. Apocrine glands are mainly located in underarms and groin areas. So eccrine are the ones that are found all over the body, in higher quantities, and play a major role in thermal homeostasis. Curiously local sweat rates are not linked to known local density of sweat glands, nor to local skin temperature (Kanosue et al., 2010).

The perspiration, in some mammals, is the most important and effective thermoregulatory mechanism when sweat evaporates. The evaporative heat loss allows to loose huge quantities of heat. From 1 gram of dripping sweat only result a few joules lost, but the same amount of evaporated sweat can dissipate about 2400 J of heat (Ashton and Gill, 2000). There are no other mechanisms that have the same effective cooling power. As others homoeostatic processes, thermoregulation is seen as a negative feedback circuit which is the basis for phenomenological modelling. The intensity of sweat rate depends on body core temperature and mean skin temperature. But sweating response generated by an increase of temperature perceived by hypothalamus is more effective than the one perceived by thermal sensors in the skin.

There are some several approaches when considering sweating rate prediction. One of the best models use heat balance equation in order to achieve the required value of sweat needed to compensate the disturbance caused by environmental thermal load.

Sweat rate modelling is a useful tool to achieve accurate results in terms of human body thermal load, but the results of the amount of water loss and hydrations balance, are crucial indicators of the heat tolerance limits. Many modelling approaches also take into account with a variable for water and heat loss by breathing. Present model does not include this feature, however for extreme dry heat and cold thermal simulations it is necessary to consider a breathing model integrated in parallel. The adjustments of water and heat loss are significant and improve tolerance/performance even by the natural induced tachycardia.

Chapter 8

Future Development Perspectives

8.1 Further elements to include

It is evident that, in spite of the potentiality of the present modelling proposal, a considerable set of upgrades must be implemented before scaling it into practice. Present level of detail would not be necessary if features like blood circulation, respiratory system, metabolic expenditure, active thermoregulatory mechanisms, were not included. Detailed local results from human body passive system also help to increase the level of detail of heat exchange with the environment.

- The first improvement should be to model the interface human environment by calculating the amount of heat lost by convection, radiation and evaporation. There are already several studies that predict the maximal evaporative heat loss linking this variable to the environmental water vapour pressure. Heat loss by radiation is linked to skin blood flow. However with arterial blood circulation, cardiac output distribution and arterial radii it is possible to achieve values of local skin blood flow close to real values. New researches conducted with thermal mannequins also bring new information about convective heat dissipation considering, more than the air velocity and humidity, the angle of incidence of the air at the skin.

- Include a detailed respiratory model to predict heat exchange and gas exchange would help to predict heat loss by respiration and metabolic expenditure. Since

Wissler (1985) that thermoregulatory response is linked to oxygen uptake, carbon dioxide exchange and lactate production. Changes in thermal properties of the blood are induced by levels of oxygen and carbon dioxide, as well as hydration level. Body management of metabolic wastes in exercise also affect it. Oxygen uptake is particularly linked to muscle endogenous heat production that can be predicted through metabolic rate of a particular exercise.

- It is known and documented in physiology that global homoeostasis come from a subgroup of homoeostatic processes that depend on each other. Water and ion concentration, blood sugar, blood pressure etc., all are affected when the organism needs to cool down or warm up. All of these homoeostatic variables are regulated by effector and response of effector feeds back to influence the magnitude of the stimulus and return them to homoeostatic balance. Typically all the models use the concept of threshold limit values to perform the control using vasomotion, shivering and sweating. However threshold values and decision mechanisms used in literature to perform this decisions rarely integrate the several processes of decision making. Research in this area would allow to increase the the potential for predicting tolerance limits in critical conditions, for healthy and non healthy people.

- The model proposed by Avolio (1980) have many distinct applications that goes from mean arterial pressure prediction to cardiovascular response due to central volume depletion. In particular, this last feature is very important to predict the tolerance limits in dry heat due to dehydration.

State of Art (chapter 2) gathers several references that can be used to easily develop and implement this feature into the present model.

8.2 Methods for future updates

There are several areas where update are needed and justified, to improve accuracy and sensitiveness.

One of the problems of this model, that can be also seen as it biggest advantage, is the level of detail of the model. It can predict the thermal diffusion and water loss by relating these values to body location. Independently on the body element size, that can be higher or lower according to programmer specifications, each place of the body has his own values of temperature, wetness and influence on global thermal load by local heating or cooling process can be achieved.

Validation procedure should be divided in three stages :

1. comparison of results with other models;
2. compare results with data collected from other researchers and laboratories;
3. validate results with experiments designed to the effect.

First stage should use one or more models from the three main guidelines of research mentioned in chapter 2. Compare results from reproduced models of Givonni, Pandolf and Stolwijk, PHS and compare with the published data of Salloum and Fiala, could cover the different types of whole body models (from empirical to rational, covering the different types of models). BioHeatSIM is designed to respond accurately to transient states what means that transient profile responses should be analysed. Comparison to real data should consider all types of environments and possible occupational demands. Should be tested three kinds of extreme heat exposure: extreme dry heat, extreme humid heat and mild hot and humid temperatures. And three types of workloads should also be considered: extreme heavy exercise, mild prolonged effort, and resting.

8.3 Practice in Occupational Health and Safety

The return of the investment done in this kind of research could cover many times the developing costs. The applicability of this kind of model, after a careful validation process, is almost illimitable, as discussed in previous section and all along the thesis report.

In terms of Safety and Hygiene can help directly to better plan production, higher rates of production in safer conditions what mean profit increase. By doing individual analysis of the work conditions, testing individual characteristics, personal protective clothing and so on can be found with the model. Indirectly safer conditions allow to reduce the number of accidents.

The kind of updates that become important for *Practice* are slightly different from those pointed in research field. Here create an user interface that linked user to all variables that need be settled is the most important improvement to consider. This way, any end user without programming skills, would be able to use this tool adapting it to his own needs. The set of variables to be adjusted and potential for changing boundary and initial conditions automatically should be explored in order to provide the suitable solution. As a multi-platform programming language, Python runs in all operative systems. However when dealing with a large set of packages, its versatility decreases. An alternative to management of the files for windows installer is needed. Incompatibility and package dependency problems are much difficult, almost impossible, to solve in Windows. In theory the program should easily run in Windows. But a problem with the dependency of Gmsh and an impossibility to integrate it with Fipy through python make the test in windows operative system to fail.

In occupational field they can range from industry to military, sports or medicine life threatening applications to thermal comfort or development of special protective equipment. Basically all environments that deal with hot or particular types of exposure can benefit form the model proposal.

With it potential for applications and development, present model intend to be the basis of future sustainable work inside academic community. Freeware, free of license is going to be distributed online for free for common use, with the hope that improvement provided by others, create a friendly user tool to improve occupational safety conditions.

Bibliography

Cyro Albuquerque-Neto and Jurandir Itizo Yanagihara. A passive model of the heat, oxygen and carbon dioxide transport in the human body. In *ASME 2009 International Mechanical Engineering Congress and Exposition*, pages 155–166. American Society of Mechanical Engineers, 2009.

Indira Ashton and Frank Gill. *Monitoring for health hazards at work*. Wiley-Blackwell, 2000.

Hossein Askarizadeh and Hossein Ahmadikia. Analytical analysis of the dual-phase-lag model of bioheat transfer equation during transient heating of skin tissue. *Heat and Mass Transfer*, 50(12):1673–1684, 2014. ISSN 0947-7411. doi: 10.1007/s00231-014-1373-6. URL `http://dx.doi.org/10.1007/s00231-014-1373-6`.

A.P. Avolio. Multi-branched model of the human arterial system. *Medical and Biological Engineering and Computing*, 18(6):709–718, 1980. ISSN 0140-0118. doi: 10.1007/BF02441895. URL `http://dx.doi.org/10.1007/BF02441895`.

H Belding and T. Hatch. Index for evaluating heat stress in terms of resulting physiological strain. *Heat Pip Air Cond*, 27(8):129 – 136, 1955.

J R Breckenridge and R F Goldman. Solar heat load in man. *Journal of Applied Physiology*, 31(5):659–663, 1971. ISSN 8750-7587. URL `http://jap.physiology.org/content/31/5/659`.

Ingelise Brynjolf, Jesper Qvist, Thorkild Mygind, Henrik Jordening, Sven Dorph, and Ole Munck. Measurement of right and left ventricular ejection fraction in dogs.

Clinical Physiology, 3(4):335–348, 1983. ISSN 1365-2281. doi: 10.1111/j.1475-097X.1983.tb00716.x. URL http://dx.doi.org/10.1111/j.1475-097X.1983.tb00716.x.

Michael M. Chen and Kenneth R. Holmes. Microvascular contributions in tissue heat transfer. *Annals of the New York Academy of Sciences*, 335(1):137–150, 1980. ISSN 1749-6632. doi: 10.1111/j.1749-6632.1980.tb50742.x. URL http://dx.doi.org/10.1111/j.1749-6632.1980.tb50742.x.

Gouri Dhatt, Emmanuel Lefrançois, and Gilbert Touzot. *Finite element method*. John Wiley & Sons, 2012.

P. Ole Fanger and JÃ¸rn Toftum. Extension of the {PMV} model to non-air-conditioned buildings in warm climates. *Energy and Buildings*, 34(6):533 – 536, 2002. ISSN 0378-7788. doi: http://dx.doi.org/10.1016/S0378-7788(02)00003-8. URL http://www.sciencedirect.com/science/article/pii/S0378778802000038. Special Issue on Thermal Comfort Standards.

M.S. Ferreira and J.I. Yanagihara. A transient three-dimensional heat transfer model of the human body. *International Communications in Heat and Mass Transfer*, 36(7):718 – 724, 2009. ISSN 0735-1933. doi: 10.1016/j.icheatmasstransfer.2009.03.010. URL http://www.sciencedirect.com/science/article/pii/S0735193309000773.

Dusan Fiala. *Dynamic Simulation of Human Heat Transfer and Thermal Comfort*. PhD thesis, Institute of Energy and Sustainable Development - DE MONTFORT UNIVERSITY LEICESTER, June 1998. URL https://www.google.pt/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB8QFjAA&url=https%3A%2F%2Fdspace.lboro.ac.uk%2Fdspace-jspui%2Fbitstream%2F2134%2F9318%2F5%2FFiala.pdf&ei=DTxKVaHwNImuUaPWgdgK&usg=AFQjCNEmDXLf6uu8q1d6HYI1bWCrjBjYWQ.

Dusan Fiala, Kevin J. Lomas, and Martin Stohrer. A computer model of human thermoregulation for a wide range of environmental conditions: the passive system. *Journal of Applied Physiology*, 87(5):1957–1972, 1999. URL http://jap.physiology.org/content/87/5/1957.abstract.

Dusan Fiala, George Havenith, Peter Bröde, Bernhard Kampmann, and Gerd Jendritzky. Utci-fiala multi-node model of human heat transfer and temperature regulation. *International Journal of Biometeorology*, 56(3):429–441, 2012. ISSN 0020-7128. doi: 10.1007/s00484-011-0424-7. URL `http://dx.doi.org/10.1007/s00484-011-0424-7`.

G. Fu. *A Transient, 3-D Mathematical Thermal Model for the Clothed Human*. PhD thesis, Kansas State University, 1995. URL `http://books.google.pt/books?id=QjpDOAAACAAJ`.

Paul Fulbrook. Core body temperature measurement: a comparison of axilla, tympanic membrane and pulmonary artery blood temperature. *Intensive and Critical Care Nursing*, 13(5):266 – 272, 1997. ISSN 0964-3397. doi: http://dx.doi.org/10.1016/S0964-3397(97)80425-9. URL `http://www.sciencedirect.com/science/article/pii/S0964339797804259`.

AP Gagge, JAJ Stolwijk, and Ysaunobu Nishi. An effective temperature scale based on a simple model of human physiological regulatiry response. *Memoirs of the Faculty of Engineering, Hokkaido University*, 13(Suppl):21–36, 1972. URL `http://eprints.lib.hokudai.ac.jp/dspace/bitstream/2115/37901/1/13Suppl_21-36.pdf`.

B Givoni and R F Goldman. Predicting rectal temperature response to work, environment, and clothing. *Journal of Applied Physiology*, 32(6):812–22, 1972. URL `http://jap.physiology.org/content/32/6/812.short`.

B Givoni and R F Goldman. Predicting effects of heat acclimatization on heart rate and rectal temperature. *Journal of Applied Physiology*, 35 (6):875–879, 1973a. ISSN 8750-7587. URL `@Article{rohtua,Title={Predictingeffectsofheatacclimatizationonheartrateandrectaltemperature.},Author={Givoni,BandGoldman,RF},Journal={JournalofAppliedPhysiology},Year={1973},Number={6},Pages={875--879},Volume={35},ISBN={1522-1601},ISSN={8750-7587},Publisher={AmericanPhysiologicalSociety}}`.

B. Givoni and R.F. Goldman. Predicting heart rate response to work, environment, and clothing. *Journal of applied physiology*, 34(2):201–204, 1973b.

URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-0015579695&partnerID=40&md5=fab9a14ba5976fbf6bb8ceb86d70204f`. cited By 27.

R.R. Gonzalez. Scenario revisited: Comparisons of operational and rational models in predicting human responses to the environment. *Journal of Thermal Biology*, 29(7-8 SPEC. ISS.):515–527, 2004. doi: 10.1016/j.jtherbio.2004.08.021. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-4744353340&partnerID=40&md5=24968808f7af6d24bf26f448a52bed79`. cited By 8.

José González-Alonso, Ricardo Mora-Rodríguez, and Edward F. Coyle. Stroke volume during exercise: interaction of environment and hydration. *American Journal of Physiology - Heart and Circulatory Physiology*, 278(2):H321–H330, 2000. ISSN 0363-6135.

Jonathan E. Guyer, Daniel Wheeler, and James A. Warren. FiPy: Partial differential equations with Python. *Computing in Science & Engineering*, 11(3):6–15, 2009. doi: 10.1109/MCSE.2009.52. URL `http://www.ctcms.nist.gov/fipy`.

R. A. Haslam and K. C. Parsons. A comparison of models for predicting human response to hot and cold environments. *Ergonomics*, 30(11):1599–1614, 1987. doi: 10.1080/00140138708966050. URL `http://dx.doi.org/10.1080/00140138708966050`. PMID: 3443087.

Roger Haslam. *An evaluation of models of human response to hot and cold environments*. PhD thesis, © RA Haslam, 1989. URL `https://dspace.lboro.ac.uk/2134/7027`.

L. M. Jiji, S. Weinbaum, and D. E. Lemons. Theory and experiment for the effect of vascular microstructure on surface tissue heat transfer—part ii: Model formulation and solution. *Journal of Biomechanical Engineering*, 106(4):331–341, 1984. doi: 10.1115/1.3138502. URL `http://link.aip.org/link/?JBY/106/331/1`.

Kazuyuki Kanosue, LarryI. Crawshaw, Kei Nagashima, and Tamae Yoda. Concepts to utilize in describing thermoregulation and neurophysiological evidence for how the system works. *European Journal of Applied Physiology*, 109(1):5–11, 2010. ISSN

1439-6319. doi: 10.1007/s00421-009-1256-6. URL `http://dx.doi.org/10.1007/s00421-009-1256-6`.

W.a Karaki, N.a Ghaddar, K.a Ghali, K.b Kuklane, I.b HolmÃ©r, and L.c Vanggaard. Human thermal response with improved ava modeling of the digits. *International Journal of Thermal Sciences*, 67:41–52, 2013. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-84874658580&partnerID=40&md5=bf08cb9ea971ff8db982f5c758f05fcf`. cited By 3.

K.K. Kraning II and R.R. Gonzalez. A mechanistic computer simulation of human work in heat that accounts for physical and physiological effects of clothing, aerobic fitness, and progressive dehydration. *Journal of Thermal Biology*, 22(4-5):331–342, 1997. doi: 10.1016/S0306-4565(97)00031-4. URL `http://www.scopus.com/inward/record.url?eid=2-s2.0-0031455661&partnerID=40&md5=af1abd12d7d3e8a20c3b8f28754b0fa3`. cited By 39.

L. H. Kuznetz. A two-dimensional transient mathematical model of human thermoregulation. *American Journal of Physiology - Regulatory, Integrative and Comparative Physiology*, 237(5):R266–R277, 1979. URL `http://ajpregu.physiology.org/content/237/5/R266.abstract`.

Hans Petter Langtangen. *Python scripting for computational science*, volume 3. Springer, 2006.

Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

Jacques Malchaire, Alain Piette, B Kampmann, P Mehnert, HJ Gebhardt, G Havenith, E Den Hartog, I Holmer, K Parsons, G Alfano, et al. Development and validation of the predicted heat strain model. *Annals of Occupational Hygiene*, 45(2):123–135, 2001. URL `http://annhyg.oxfordjournals.org/content/45/2/123.full.pdf`.

Raad H. Mohiaddin, Peter D. Gatehouse, and David N. Firmin. Exercise-related changes in aortic flow measured with spiral echo-planar mr velocity mapping.

Journal of Magnetic Resonance Imaging, 5(2):159–163, 1995. ISSN 1522-2586. doi: 10.1002/jmri.1880050209. URL http://dx.doi.org/10.1002/jmri.1880050209.

JOSEPH P Murgo, NICO Westerhof, JOHN P Giolma, and STEPHEN A Altobelli. Aortic input impedance in normal man: relationship to pressure wave forms. *Circulation*, 62(1):105–116, 1980.

Yasuto Nakanishi, Tetsuya Kimura, and Yoshinori Yokoo. Maximal physiological responses to deep water running. *Applied Human Science*, 18(2):31–35, 1999. doi: 10.2114/jpa.18.31.

Cyro Albuquerque Neto. *Modelo integrado dos sistemas térmico e respiratório do corpo humano*. Tese de doutorado em engenharia mecânica de energia de fluidos, Escola Politécnica, Universidade de São Paulo, São Paulo, 2010. URL <http://www.teses.usp.br/teses/disponiveis/3/3150/tde-28022011-124824/>.

WILMER W Nichols, C Richard Conti, WILLIAM E Walker, and WILLIAM R Milnor. Input impedance of the systemic circulation in man. *Circulation Research*, 40(5):451–458, 1977.

Y Nishi and AP Gagge. Effective temperature scale useful for hypo-and hyperbaric environments. *Aviation, space, and environmental medicine*, 48(2):97–107, 1977. URL http://www.cbe.berkeley.edu/research/other-papers/Nishi%20-%20Gagge%201977%20Effective%20temperature%20scale%20useful%20for%20hypo-%20and%20hyperbaric%20environments.pdf.

K. B. Pandolf, R. L. Burse, and R. F. Goldman. Role of physical fitness in heat acclimatisation, decay and reinduction. *Ergonomics*, 20(4):399–408, 1977. doi: 10.1080/00140137708931642. URL http://dx.doi.org/10.1080/00140137708931642. PMID: 908323.

Kent B. Pandolf, Leander A. Stroschein, Lawrence L. Drolet, Richard R. Gonzalez, and Michael N. Sawka. Prediction modeling of physiological responses and human performance in the heat. *Computers in Biology and Medicine*, 16(5):319 – 329, 1986.

ISSN 0010-4825. doi: http://dx.doi.org/10.1016/0010-4825(86)90001-6. URL `http://www.sciencedirect.com/science/article/pii/0010482586900016`.

Ken Parsons. *Human thermal environments: the effects of hot, moderate, and cold environments on human health, comfort, and performance.* Crc Press, 3rd edition, 2014.

Suhas Patankar. *Numerical heat transfer and fluid flow.* CRC Press, 1980.

Harry H. Pennes. Analysis of tissue and arterial blood temperatures in the resting human forearm. *Journal of Applied Physiology*, 1(2):93–122, 1948. URL `http://jap.physiology.org/content/1/2/93.short`.

António M. Raimundo and António R. Figueiredo. Personal protective clothing and safety of firefighters near a high intensity fire front. *Fire Safety Journal*, 44(4):514 – 521, 2009. ISSN 0379-7112. doi: http://dx.doi.org/10.1016/j.firesaf.2008.10.007. URL `http://www.sciencedirect.com/science/article/pii/S0379711208001458`.

Thomas Reilly, Clare N Dowzer, and NT Cable. The physiology of deep-water running. *Journal of Sports Sciences*, 21(12):959–972, 2003. doi: 10.1080/02640410310001641368. URL `http://dx.doi.org/10.1080/02640410310001641368`. PMID: 14748454.

L B Rowell. Human cardiovascular adjustments to exercise and thermal stress. *Physiological Reviews*, 54(1):75–159, 1974. URL `http://physrev.physiology.org/content/54/1/75`.

Loring B. Rowell, John R. Blackmon, Richard H. Martin, John A. Mazzarella, and Robert A. Bruce. Hepatic clearance of indocyanine green in man under thermal and exercise stresses. *Journal of Applied Physiology*, 20(3):384–394, 1965. ISSN 8750-7587. URL `http://jap.physiology.org/content/20/3/384`.

M. Salloum, N. Ghaddar, and K. Ghali. A new transient bioheat model of the human body and its integration to clothing models. *International Journal of Thermal Sciences*, 46(4):371 – 384, 2007. ISSN 1290-0729. doi: http://dx.doi.org/10.1016/j.ijthermalsci.2006.06.017. URL `http://www.sciencedirect.com/science/article/pii/S129007290600175X`.

Abdul-Majeed Salmasi. Cardiac output during exercise. In Abdul-Majeed Salmasi and AbdulmassihS. Iskandrian, editors, *Cardiac Output and Regional Flow in Health and Disease*, volume 138 of *Developments in Cardiovascular Medicine*, pages 91–96. Springer Netherlands, 1993. ISBN 978-94-010-4816-3. doi: 10.1007/978-94-011-1848-4_9. URL http://dx.doi.org/10.1007/978-94-011-1848-4_9.

Y. Shapiro, K.B. Pandolf, and R.F. Goldman. Predicting sweat loss response to exercise, environment and clothing. *European Journal of Applied Physiology and Occupational Physiology*, 48(1):83–96, 1982. doi: 10.1007/BF00421168. URL http://www.scopus.com/inward/record.url?eid=2-s2.0-0020078180&partnerID=40&md5=c383526991c1b841252408dfc0596286. cited By 59.

Wei Jie Song, Sheldon Weinbaum, and Latif M. Jiji. A theoretical model for peripheral tissue heat transfer using the bioheat equation of weinbaum and jiji. *Journal of Biomechanical Engineering*, 109(1):72–78, February 1987. ISSN 0148-0731. URL http://dx.doi.org/10.1115/1.3138646.

Scott A. Stevens, William D. Lakin, and Wolfgang Goetz. A differentiable, periodic function for pulsatile cardiac output based on heart rate and stroke volume. *Mathematical Biosciences*, 182(2):201 – 211, 2003. ISSN 0025-5564. doi: http://dx.doi.org/10.1016/S0025-5564(02)00200-6. URL http://www.sciencedirect.com/science/article/pii/S0025556402002006.

JAJ Stolwijk. A mathematical model of physiological temperature regulation in man. Technical report, 1971. URL http://scholar.google.pt/scholar_url?url=http%3A%2F%2Fwww.ntrs.nasa.gov%2Farchive%2Fnasa%2Fcasi.ntrs.nasa.gov%2F19710023925_1971023925.pdf&hl=en&sa=T&oi=ggp&ct=res&cd=5&ei=IfJIVei5AYOVqAHx44G4BQ&scisig=AAGBfm3X2SgCUTaQjw0Vm4ayyjCBsXTURQ&nossl=1&ws=996x955.

J.A.J. Stolwijk and J.D. Hardy. Temperature regulation in man — a theoretical study. *Pflüger's Archiv für die gesamte Physiologie des Menschen und der Tiere*, 291:129–162,

1966. ISSN 0365-267x. doi: 10.1007/BF00412787. URL http://dx.doi.org/10.1007/BF00412787.

Shin-ichi Tanabe, K.Kozo Kobayashi, Junta Nakano, Yoshiichi Ozeki, and Masaaki Konishi. Evaluation of thermal comfort using combined multi-node thermoregulation (65mn) and radiation models and computational fluid dynamics (cfd). *Energy and Buildings*, 34(6):637 – 646, 2002. ISSN 0378-7788. doi: http://dx.doi.org/10.1016/S0378-7788(02)00014-2. URL http://www.sciencedirect.com/science/article/pii/S0378778802000142. Special Issue on Thermal Comfort Standards.

Johannes H. G. M. van Beek, Farahaniza Supandi, Anand K. Gavai, Albert A. de Graaf, Thomas W. Binsl, and Hannes Hettling. Simulating the physiology of athletes during endurance sports events: modelling human energy conversion and metabolism. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 369(1954):4295–4315, 2011. ISSN 1364-503X. doi: 10.1098/rsta.2011.0166.

H.K. Versteeg and W. Malalasekera. *An introduction to computational fluid dynamics: the finite volume method*. Prentice Hall, 2007.

Liqiu Wang. Flows through porous media: A theoretical development at macroscale. *Transport in Porous Media*, 39(1):1–24, 2000. ISSN 0169-3913. doi: 10.1023/A:1006647505709. URL http://dx.doi.org/10.1023/A%3A1006647505709.

Liqiu Wang and Jing Fan. Modeling bioheat transport at macroscale. *Journal of Heat Transfer*, 133(1):011010, 2011. doi: 10.1115/1.4002361. URL http://link.aip.org/link/?JHR/133/011010/1.

Yuehong Wang, Yueping Qin, and Jiuling Zhang. Application of new finite volume method (fvm) on transient heat transferring. In Rongbo Zhu, Yanchun Zhang, Baoxiang Liu, and Chunfeng Liu, editors, *Information Computing and Applications*, volume 105 of *Communications in Computer and Information Science*, pages 109–116. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-16335-7. doi: 10.1007/978-3-642-16336-4_15. URL http://dx.doi.org/10.1007/978-3-642-16336-4_15.

S. Weinbaum and L. M. Jiji. A new simplified bioheat equation for the effect of blood flow on local average tissue temperature. *Journal of Biomechanical Engineering*, 107 (2):131–139, 1985. doi: 10.1115/1.3138533. URL `http://link.aip.org/link/?JBY/107/131/1`.

S. Weinbaum, D. E. Lemons, and L. M. Jiji. Theory and experiment for the effect of vascular microstructure on surface tissue heat transfer part i: Anatomical foundation and model conceptualization. *Journal of Biomechanical Engineering*, 106(4):321–330, November 1984. ISSN 0148-0731. URL `http://dx.doi.org/10.1115/1.3138501`.

E. H. Wissler. *Heat Transfer in Medicine and Biology*, volume 1, chapter Mathematical simulation of human thermal behavior using whole body models, page 325–373. New York: Plenum, 1985.

Eugene H. Wissler. Steady-state temperature distribution in man. *Journal of Applied Physiology*, 16(4):734–740, 1961. ISSN 8750-7587. URL `http://jap.physiology.org/content/16/4/734`.

Eugene H Wissler. An analysis of factors affecting temperature levels in the nude human. *Temperature-its measurement and control in science and industry*, 3:603–612, 1963.

Eugene H. Wissler. Pennes' 1948 paper revisited. *Journal of Applied Physiology*, 85(1): 35–41, 1998. URL `http://jap.physiology.org/content/85/1/35`.

EugeneH. Wissler. A mathematical model of the human thermal system. *The bulletin of mathematical biophysics*, 26(2):147–166, 1964. ISSN 0007-4985. doi: 10.1007/BF02476835. URL `http://dx.doi.org/10.1007/BF02476835`.

Acronyms

default

$\nabla^2 T_t$

temperature gradient of the tissue ($°C/m$). 59

ρ_{bl}

specific weight of the blood (kg/m^3). 59

ρ_t

specific weight of the tissue (Kg/m^3). 58

C

c_{bl}

specific heat of the blood ($J/kg.°C$). 59

c_t

specific heat of the tissue ($J/Kg.°C$). 58

CO

Cardiac output ($\dot{Q}$) is the quantity of blood that is pumped from the heart over one minute. The units usually vary between $cm^3 \cdot h$ and $l \cdot min.$. 23, 56, 119, 150–152, 155

Continuum Mechanics

Modelling an object as a continuum assumes that the substance of the object completely fills the space it occupies. Modelling objects in this way ignores the fact that matter is made of atoms, and so is not continuous; however, on length scales much greater than that of inter-atomic distances, such models are highly accurate. Fundamental physical laws such as the conservation of mass, the conservation of momentum, and the conservation of energy may be applied to such models to derive differential equations describing the behaviour of such objects, and some information about the particular material studied is added through constitutive relations.

`http://en.wikipedia.org/wiki/Continuum_mechanics`. 47

CV

Control Volume. 62, 63, 65, 66, 68, 75

E

Eclipse

The Eclipse Platform, and all the tools needed to develop and como corre a revisão da provadebug it: Java and Plug-in Development Tooling, Git and CVS.

`http://pydev.org/screenshots.html`. 35

F

FDM

Finite Difference Methods. 62

FEM

Finite Element Methods. 61, 62

Fipy

FiPy is an object oriented, partial differential equation (PDE) solver, written in Python, based on a standard finite volume (FV) approach. The framework has been developed in the Materials Science and Engineering Division (MSED) and Center for Theoretical and Computational Materials Science (CTCMS), in the Material Measurement Laboratory (MML) at the National Institute of Standards

and Technology (NIST).
`http://www.ctcms.nist.gov/fipy/`. 35, 64, 79, 83, 96, 97, 111, 162

Free Software

Free Software is software free of charge, whose source code is completely accessible to the user. More than be able to see exactly what the program does, in *Free Software* the user can also change the program features according to his will or needs.. 31

FVM

Finite Volume Methods. 35, 61–63

G

Gedit

Gedit is the GNOME text editor. While aiming at simplicity and ease of use, gedit is a powerful general purpose text editor.
`https://wiki.gnome.org/Apps/Gedit`. 36

Gmsh

Gmsh is a 3D finite element grid generator with a build-in CAD engine and post-processor. Its design goal is to provide a fast, light and user-friendly meshing tool with parametric input and advanced visualization capabilities. Gmsh is built around four modules: geometry, mesh, solver and post-processing. The specification of any input to these modules is done either interactively using the graphical user interface or in ASCII text files using Gmsh's own scripting language.
`http://geuz.org/gmsh/`. 35, 36, 79, 96, 162

GUI

Graphical User Interface. 36

I

IDE

Integrated Development Environments. ii, 36

IPython

IPython provides a rich architecture for interactive computing with: powerful interactive shells (terminal and Qt-based); a browser-based notebook with support for code, text, mathematical expressions, in-line plots and other rich media; support for interactive data visualization and use of GUI toolkits; flexible, embeddable interpreters to load into your own projects; easy to use, high performance tools for parallel computing.

`http://ipython.org/`. 34, 35

K

k_t

thermal conductivity of the tissue ($W/m.^{\circ}C$). 58

M

Matplotlib

matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. matplotlib can be used in python scripts, the python and ipython shell (ala MATLAB® or Mathematica®), web application servers, and six graphical user interface toolkits.

`http://matplotlib.org/`. 34, 35

Mayavi

The Mayavi project includes two related packages for 3-dimensional visualization:

- Mayavi: A tool for easy and interactive visualization of data, with seamless integration with Python scientific libraries.

- TVTK: A Traits-based wrapper for the Visualization Toolkit, a popular open-source visualization library.

These libraries operate at different levels of abstraction. TVTK manipulates visualization objects, while Mayavi lets you operate on your data, and then see the results. Most users either use the Mayavi user interface or program to its scripting interface; you probably don't need to interact with TVTK unless you want to create a new Mayavi module.
`http://code.enthought.com/projects/mayavi/`. 35, 79, 111, 113

mesh

a mesh is considered the group of elements (vertices, faces and cells) that define the segmentation of the solution domain.
`http://www.ctcms.nist.gov/fipy/documentation/numerical/discret.html`. 63

N

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object; sophisticated (broadcasting) functions; tools for integrating C/C++ and Fortran code; useful linear algebra, Fourier transform, and random number capabilities.
Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
`http://www.numpy.org/`. 34, 35, 83

P

PDE

Partial Differential Equation. 60

Porous Media

A porous medium (or a porous material) is a material containing pores (voids). The skeletal portion of the material is often called the "matrix" or "frame". The pores are typically filled with a fluid (liquid or gas). The skeletal material is usually a solid, but structures like foams are often also usefully analysed using concept of porous media.
The concept of porous media is used in many areas of applied science and engineering: filtration, mechanics (acoustics, geomechanics, soil mechanics, rock mechanics), engineering (petroleum engineering, bio-remediation, construction engineering), geosciences (hydrogeology, petroleum geology, geophysics), biology and biophysics, material science, etc. Fluid flow through porous media is a subject of most common interest and has emerged a separate field of study. The study of more general behaviour of porous media involving deformation of the solid frame is called poromechanics.
`http://en.wikipedia.org/wiki/Porous_medium`. 47

PyDev

PyDev is a Python IDE for Eclipse, which may be used in Python, Jython and IronPython development.
`http://pydev.org/`. 35

PyQt

PyQt is a set of Python v2 and v3 bindings for Digia's Qt application framework and runs on all platforms supported by Qt including Windows, MacOS/X and Linux. `http://www.riverbankcomputing.co.uk/software/pyqt/intro`. 35

Q

$\hat{q}_t$

endogenous heat production of the tissue (W/m^3). 59

Qt

Qt is a cross-platform application and UI framework for developers.
`http://qt-project.org/`. 35

S

SciPy

The SciPy library is one of the core packages that make up the SciPy stack. It provides many user-friendly and efficient numerical routines such as routines for numerical integration and optimization.
`http://www.scipy.org/`. 34, 35, 70, 83, 103

Spyder

Spyder (previously known as Pydee) is a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features.
`https://code.google.com/p/spyderlib/`. 36

SV

Stroke Volume. 151, 152

T

T_{ar}

arterial blood temperature ($°C$). 59

T_t

temperature of the tissue ($°C$). 58

Tkinter

Tkinter is Python's de-facto standard GUI (Graphical User Interface) package.
`https://wiki.python.org/moin/TkInter`. 35

V

$\hat{V}_{sv}$

arterial blood flow of the body segment ($m^3/m^3.s$). 59

VIM

Vim is an advanced text editor that seeks to provide the power of the de-facto Unix editor 'Vi', with a more complete feature set.

`http://www.vim.org/about.php`. 36

W

wxGlade

wxGlade is a GUI designer written in Python with the popular GUI toolkit wxPython, that helps you create wxWidgets/wxPython user interfaces.

`http://wxglade.sourceforge.net/`. 35

wxPython

wxPython is a GUI toolkit for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a Python extension module (native code) that wraps the popular wxWidgets cross platform GUI library, which is written in C++.

`http://www.wxpython.org/`. 35

Appendix A

Source Code Listings of the Human Body Geometry

A.1 Hand Geometric Code

```
// Hand Geometry - 4 layers

// ==================================================================//
//                               Mesh Chars                          //
// ==================================================================//

Size = .01;

xsk1  = 0.0;
xsk2 = 0.1098;
ysk1   = 0.0;
ysk2  = 0.0242;
xft1 = 0.0034;
xft2 = 0.1055;
yft1 = 0.00076;
yft2 = 0.0234;
xmsc1 = 0.0093;
xmsc2 = 0.0996;
ymsc1 = 0.00205;
ymsc2 = 0.0221;
xbn1 = 0.0167;
xbn2 = 0.0922;
ybn1 = 0.00699;
ybn2 = 0.0172;

z    = 0.19; // hand length (m)

// Mesh Divisions

NDlx  = 10;     // Div. Lines Bone
```

```
NDly = 5;      // Div. Lines Muscle
NDlz   = 19;      // Div. Lines Gord

// ===================================================================//
//                               Points                               //
// ===================================================================//

// Outer Vertexes of the Prisma - Skin
Point(1)={0., 0., 0., Size};
Point(2)={0., ysk2, 0., Size};
Point(3)={xsk2, ysk2, 0., Size};
Point(4)={xsk2, 0., 0., Size};

// Outer Vertexes of the Prisma - Fat
Point(5)={xft1, yft1, 0., Size};
Point(6)={xft1, yft2, 0., Size};
Point(7)={xft2, yft2, 0., Size};
Point(8)={xft2, yft1, 0., Size};

// Outer Vertexes of the Prisma - Muscle
Point(9) ={xmsc1, ymsc1, 0., Size};
Point(10)={xmsc1, ymsc2, 0., Size};
Point(11)={xmsc2, ymsc2, 0., Size};
Point(12)={xmsc2, ymsc1, 0., Size};

//Outer Vertexes of the Prisma - Bone
Point(13)={xbn1, ybn1, 0., Size};
Point(14)={xbn1, ybn2, 0., Size};
Point(15)={xbn2, ybn2, 0., Size};
Point(16)={xbn2, ybn1, 0., Size};

// ===================================================================//
//                               Lines                                //
// ===================================================================//

//Lines SK
Line(1)={1,2}; Transfinite Line{1}=4;
Line(2)={2,3}; Transfinite Line{2}=10;
Line(3)={3,4}; Transfinite Line{3}=4;
Line(4)={4,1}; Transfinite Line{4}=10;

//Lines Ft
Line(5)={5,6}; Transfinite Line{5}=4;
Line(6)={6,7}; Transfinite Line{6}=10;
Line(7)={7,8}; Transfinite Line{7}=4;
Line(8)={8,5}; Transfinite Line{8}=10;

//Lines Msc
Line(9) ={9, 10}; Transfinite Line{9}=4;
Line(10)={10,11}; Transfinite Line{10}=10;
Line(11)={11,12}; Transfinite Line{11}=4;
Line(12)={12, 9}; Transfinite Line{12}=10;

//Lines Bone
Line(13)={13, 14}; Transfinite Line{13}=4;
Line(14)={14, 15}; Transfinite Line{14}=10;
Line(15)={15, 16}; Transfinite Line{15}=4;
Line(16)={16, 13}; Transfinite Line{16}=10;

//Diagonals
Line(17)={1,5}; Transfinite Line{17}=2;
Line(18)={2,6}; Transfinite Line{18}=2;
Line(19)={3,7}; Transfinite Line{19}=2;
Line(20)={4,8}; Transfinite Line{20}=2;

Line(21)={5, 9}; Transfinite Line{21}=2;
Line(22)={6,10}; Transfinite Line{22}=2;
Line(23)={7,11}; Transfinite Line{23}=2;
Line(24)={8,12}; Transfinite Line{24}=2;

Line(25)={ 9,13}; Transfinite Line{25}=4;
Line(26)={10,14}; Transfinite Line{26}=4;
Line(27)={11,15}; Transfinite Line{27}=4;
Line(28)={12,16}; Transfinite Line{28}=4;

// ===================================================================//
//                              Loops                                //
// ===================================================================//

//Line Loops of the Sk, Ft, Msc, Bn
Line Loop(1)={ 1, 18, -5, -17};
Line Loop(2)={ 2, 19, -6, -18};
Line Loop(3)={ 3, 20, -7, -19};
Line Loop(4)={ 4, 17, -8, -20};

Line Loop(5)={ 5, 22, -9, -21};
Line Loop(6)={ 6, 23,-10, -22};
Line Loop(7)={ 7, 24,-11, -23};
Line Loop(8)={ 8, 21,-12, -24};

Line Loop( 9)={  9, 26,-13, -25};
Line Loop(10)={ 10, 27,-14, -26};
Line Loop(11)={ 11, 28,-15, -27};
Line Loop(12)={ 12, 25,-16, -28};

Line Loop(13)={13,14,15,16};

// ===================================================================//
//                      Creating the Surfaces                        //
// ===================================================================//

Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 6, 5}; Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 2, 3, 7, 6}; Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 3, 4, 8, 7}; Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 4, 1, 5, 8}; Recombine
    Surface{4};

Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 5, 6,10, 9}; Recombine
    Surface{5};
```

```
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 6, 7,11,10}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 7, 8,12,11}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 8, 5, 9,12}; Recombine
    Surface{8};

Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 9,10,14,13}; Recombine
    Surface{9};
Plane Surface(10)={10}; Transfinite Surface{10}={10,11,15,14}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={11,12,16,15}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={12, 9,13,16}; Recombine
    Surface{12};

Plane Surface(13)={13}; Transfinite Surface{13}={13,14,15,16}; Recombine
    Surface{13};

// ==================================================================//
//                       Volumes Construction                        //
// ==================================================================//

Extrude{0.0,0.0,z} {Surface{1,2,3,4,5,6,7,8,9,10,11,12,13}; Layers{NDlz};
    Recombine;}

// ==================================================================//
//                      Defining Physical Volumes                    //
// ==================================================================//

Physical Volume("Bone")   = {13};
Physical Volume("Muscle") = { 9,10,11,12};
Physical Volume("Fat")    = { 5, 6, 7, 8};
Physical Volume("Skin")   = { 1, 2, 3, 4};

Show "*";

Show "*";
Show "*";
```

A.2 Lower Arm Geometric Code

```
// Lower Arm Geometry - multiple layers

// ==================================================================//
//                              Mesh Data                            //
// ==================================================================//

Size = .01;

rbn1  = 0.0117; // Bone (m)
rbn2  = 0.0251; // Bone (m)
rmsc = 0.0278; // Muscle (m)
```

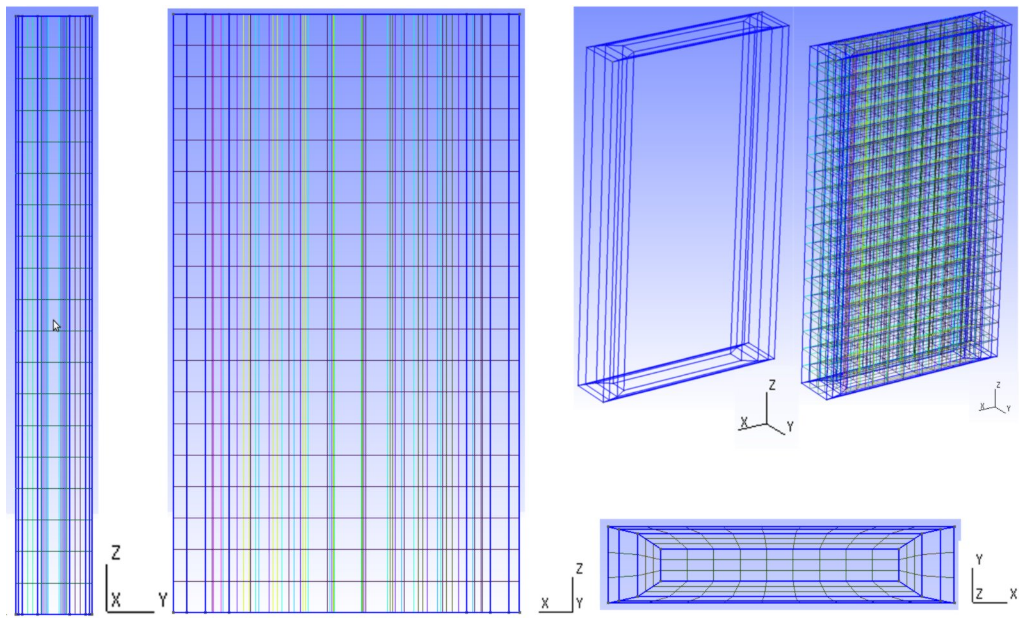
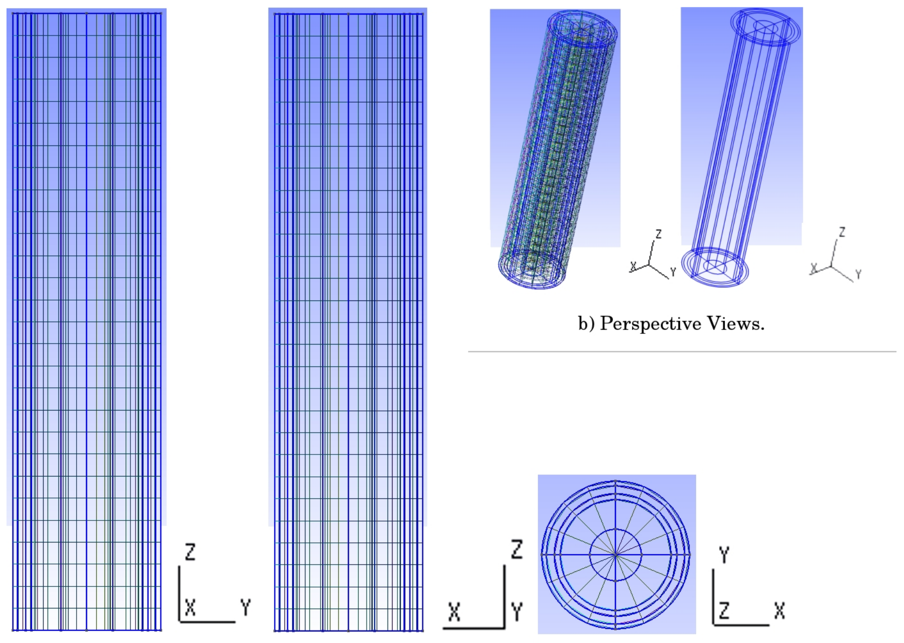


Figure A.1: Hand mesh Geometry in GMSH.

```
rf    = 0.0314; // Fat (m)
rsk   = 0.0335; // Skin (m)

z     = 0.280; // lower arm length (m)

// Mesh Divisions

NDc     = 5;      // Div. circ.

NDlbn1   = 2;      // Div. Lines  Bone
NDlbn2   = 2;      // Div. Lines  Bone
NDlmsc  = 2;      // Div. Lines  Muscle
NDlf    = 2;      // Div. Lines  Fat
NDlsk   = 2;      // Div. Lines  Skin

NL = 28; // Number of layers

Point(1)={0., 0., 0., Size}; // Centre of the Cylinder

// ===================================================================//
//                              Points                               //
// ===================================================================//

// Circumference Points Bone init rbn1
Point(2)={0.,rbn1,0.,Size};
Point(3)={rbn1,0.,0.,Size};
Point(4)={0.,-rbn1,0.,Size};
Point(5)={-rbn1,0.,0.,Size};

// Circumference Points Bone final rbn2
Point(6)={0.,rbn2,0.,Size};
Point(7)={rbn2,0.,0.,Size};
Point(8)={0.,-rbn2,0.,Size};
Point(9)={-rbn2,0.,0.,Size};

// Circumference Points Muscle rmsc
Point(10)={0.,rmsc,0.,Size};
Point(11)={rmsc,0.,0.,Size};
Point(12)={0.,-rmsc,0.,Size};
Point(13)={-rmsc,0.,0.,Size};

// Circumference Points Fat rf
Point(14)={0.,rf,0.,Size};
Point(15)={rf,0.,0.,Size};
Point(16)={0.,-rf,0.,Size};
Point(17)={-rf,0.,0.,Size};

// Circumference Points Skin rsk
Point(18)={0.,rsk,0.,Size};
Point(19)={rsk,0.,0.,Size};
Point(20)={0.,-rsk,0.,Size};
Point(21)={-rsk,0.,0.,Size};

// ====================================================================//
//                          Circumferences                             //
// ====================================================================//

//Circumf. Arcs Bone1
Circle( 1)={2,1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3,1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4,1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5,1,2}; Transfinite Line{ 4}=NDc;

//Circumf. Arcs Bone2
Circle( 5)={6,1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7,1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8,1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9,1,6}; Transfinite Line{ 8}=NDc;

//Circumf. Arcs Muscle
Circle( 9)={10,1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11,1,12}; Transfinite Line{10}=NDc;
Circle(11)={12,1,13}; Transfinite Line{11}=NDc;
Circle(12)={13,1,10}; Transfinite Line{12}=NDc;

//Circumf. Arcs Fat
Circle(13)={14,1,15}; Transfinite Line{13}=NDc;
Circle(14)={15,1,16}; Transfinite Line{14}=NDc;
Circle(15)={16,1,17}; Transfinite Line{15}=NDc;
Circle(16)={17,1,14}; Transfinite Line{16}=NDc;

//Circumf. Arcs Skin
Circle(17)={18,1,19}; Transfinite Line{17}=NDc;
Circle(18)={19,1,20}; Transfinite Line{18}=NDc;
Circle(19)={20,1,21}; Transfinite Line{19}=NDc;
Circle(20)={21,1,18}; Transfinite Line{20}=NDc;


// ====================================================================//
//                         1/4 Circle division lines                         //
// ======================================================================//

// Lines  Bone1
Line(21)={1,2}; Transfinite Line{21}=NDlbn1 Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDlbn1 Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDlbn1 Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDlbn1 Using Progression 0.5 ;

//Lines  Bone2
Line(25)={2,6}; Transfinite Line{25}=NDlbn2;
Line(26)={3,7}; Transfinite Line{26}=NDlbn2;
Line(27)={4,8}; Transfinite Line{27}=NDlbn2;
Line(28)={5,9}; Transfinite Line{28}=NDlbn2;

// Lines  Muscle
Line(29)={6,10}; Transfinite Line{29}=NDlmsc Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlmsc Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlmsc Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlmsc Using Progression 0.8 ;

//Lines  Fat
Line(33)={10,14}; Transfinite Line{33}=NDlf;
Line(34)={11,15}; Transfinite Line{34}=NDlf;
Line(35)={12,16}; Transfinite Line{35}=NDlf;
Line(36)={13,17}; Transfinite Line{36}=NDlf;

// Lines  Skin
Line(37)={14,18}; Transfinite Line{37}=NDlsk Using Progression 0.8 ;
Line(38)={15,19}; Transfinite Line{38}=NDlsk Using Progression 0.8 ;
Line(39)={16,20}; Transfinite Line{39}=NDlsk Using Progression 0.8 ;
Line(40)={17,21}; Transfinite Line{40}=NDlsk Using Progression 0.8 ;

// ======================================================================//
//                          1/4 circle Sections                          //
// ======================================================================//


//Close Loops Bone
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Close Loops Bone
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Close Loops Muscle
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};

```

```
//Close Loops Fat
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};

//Close Loops Skin
Line Loop(17)={37,17,-38,-13};
Line Loop(18)={38,18,-39,-14};
Line Loop(19)={39,19,-40,-15};
Line Loop(20)={40,20,-37,-16};

// ====================================================================//
//                  1/4 Circumf. Surface Construction                  //
// ====================================================================//

// Defining Surfaces Bone (1/4 circle)
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Defining Surfaces Bone (pipe sections)
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Defining Surfaces Muscle (pipe sections)
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Defining Surfaces Fat (pipe sections)
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};

```

```
//Defining Surfaces Skin (pipe sections)
Plane Surface(17)={17}; Transfinite Surface{17}={14,18,19,15}; Recombine
    Surface{17};
Plane Surface(18)={18}; Transfinite Surface{18}={15,19,20,16}; Recombine
    Surface{18};
Plane Surface(19)={19}; Transfinite Surface{19}={16,20,21,17}; Recombine
    Surface{19};
Plane Surface(20)={20}; Transfinite Surface{20}={17,21,18,14}; Recombine
    Surface{20};

// ===================================================================//
//                          Volume Construction                        //
// ===================================================================//

Extrude{0.0,0.0,z} {Surface
    {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}; Layers{NL};
    Recombine;}

// ===================================================================//
//                     Defining Physical Volumes                       //
// ===================================================================//

Physical Volume("Bone") = {6,8};
Physical Volume("Muscle") = {1,2,3,4,5,7,9,10,11,12};
Physical Volume("Fat") = {13,14,15,16};
Physical Volume("Skin") = {17,18,19,20};

Show "*";
```



b) Perspective Views.

a) Top views.

Figure A.2: Lower Arm mesh Geometry in GMSH.

A.3 Upper Arm Geometric Code

```
// Arm Geometry - 4 layers

//==================================================================//
//                           Mesh Data                              //
//==================================================================//

Size = .01;

rbn  = 0.0180; // Bone (m)
rmsc = 0.0354; // Muscle (m)
rf   = 0.0400; // Fat (m)
rsk  = 0.0426; // Skin (m)

z    = 0.310; // arm length (m)

// Divisions of the Mesh

NDc    = 5;     // Div. circumf.

NDlbn  = 4;     // Div. Lines Bone
NDlmsc = 5;     // Div. Lines Muscle
NDlf   = 3;     // Div. Lines Fat
NDlsk  = 3;     // Div. Lines Skin

NL = 31; // Number of Layers

Point(1)={0., 0., 0., Size}; // Centre of the Cylinder

//==================================================================//
//                           Points                                 //
//==================================================================//

// Circumference Points Bone rbn
Point(2)={0.,rbn,0.,Size};
Point(3)={rbn,0.,0.,Size};
Point(4)={0.,-rbn,0.,Size};
Point(5)={-rbn,0.,0.,Size};

// Circumference Points Fat rmsc
Point(6)={0.,rmsc,0.,Size};
Point(7)={rmsc,0.,0.,Size};
Point(8)={0.,-rmsc,0.,Size};
Point(9)={-rmsc,0.,0.,Size};

// Circumference Points Muscle rf
Point(10)={0.,rf,0.,Size};
Point(11)={rf,0.,0.,Size};
Point(12)={0.,-rf,0.,Size};
Point(13)={-rf,0.,0.,Size};

// Circumference Points Skin rsk
Point(14)={0.,rsk,0.,Size};
Point(15)={rsk,0.,0.,Size};
Point(16)={0.,-rsk,0.,Size};
Point(17)={-rsk,0.,0.,Size};

//================================================================//
//                         Circumferences                         //
//================================================================//

//Circumference Arcs Bone
Circle( 1)={2,1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3,1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4,1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5,1,2}; Transfinite Line{ 4}=NDc;

//Circumference Arcs Muscle
Circle( 5)={6,1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7,1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8,1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9,1,6}; Transfinite Line{ 8}=NDc;

//Circumference Arcs Fat
Circle( 9)={10,1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11,1,12}; Transfinite Line{10}=NDc;
Circle(11)={12,1,13}; Transfinite Line{11}=NDc;
Circle(12)={13,1,10}; Transfinite Line{12}=NDc;

//Circumference Arcs Skin
Circle(13)={14,1,15}; Transfinite Line{13}=NDc;
Circle(14)={15,1,16}; Transfinite Line{14}=NDc;
Circle(15)={16,1,17}; Transfinite Line{15}=NDc;
Circle(16)={17,1,14}; Transfinite Line{16}=NDc;

//================================================================//
//                  1/4 Circle Division Lines                     //
//================================================================//

// Lines Bone
Line(21)={1,2}; Transfinite Line{21}=NDlbn Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDlbn Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDlbn Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDlbn Using Progression 0.5 ;

//Lines Muscle
Line(25)={2,6}; Transfinite Line{25}=NDlmsc;
Line(26)={3,7}; Transfinite Line{26}=NDlmsc;
Line(27)={4,8}; Transfinite Line{27}=NDlmsc;
Line(28)={5,9}; Transfinite Line{28}=NDlmsc;

// Lines Fat
Line(29)={6,10}; Transfinite Line{29}=NDlf Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlf Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlf Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlf Using Progression 0.8 ;

//Lines Skin
Line(33)={10,14}; Transfinite Line{33}=NDlsk;
Line(34)={11,15}; Transfinite Line{34}=NDlsk;
Line(35)={12,16}; Transfinite Line{35}=NDlsk;
Line(36)={13,17}; Transfinite Line{36}=NDlsk;

//================================================================//
//                1/4 circimf. section construction               //
//================================================================//

//Close Loopss Bone
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Close Loopss Muscle
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Close Loopss Fat
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};

//Close Loopss Skin
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};

//================================================================//
//               1/4 Circle Surface Construction                  //
//================================================================//

// Defining Surfaces  Bone (1/4 circle surface)
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Defining Surfaces  Muscle (pipe sections)
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Defining Surfaces  Fat (pipe sections)
```

```
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Defining Surfaces  Skin (pipe sections)
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};

//=================================================================//
//                       Volume Construction                       //
//=================================================================//

Extrude{0.0,0.0,z} {Surface{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}; Layers{
    NL}; Recombine;}

//=================================================================//
//                  Defining Physical Volumes                      //
//=================================================================//

Physical Volume("Bone") = {1,2,3,4};
Physical Volume("Muscle") = {5,6,7,8};
Physical Volume("Fat") = {9,10,11,12};
Physical Volume("Skin") = {13,14,15,16};

Show "*";
```

A.4 Trunk Geometric Code

```
// Geometry of the Trunk

// ===================================================================//
//                             Mesh Data                              //
// ===================================================================//

Size = .01;

rh   = 0.0368; // Heart (m)
rlu  = 0.0737; // Lung (m)
rv   = 0.0866; // Viscera (m)
rbn  = 0.0941; // Bone (m)
rmsc = 0.1213; // Muscle (m)
rf   = 0.1336; // Fat (m)
```

```
rsk  = 0.1358; // Skin (m)

zh       = 0.600-0.507; // init heart layer (m)
zh_lun_bn = 0.507-0.437; // end layers (m)
z        = 0.437; // trunk length (m)

// Mesh Divisions

NDc = 5;        // Div. circ.

NDlh   = 4;     // Div. Lines Heart
NDllu  = 7;     // Div. Lines Lung
NDlv   = 8;     // Div. Lines Viscera
NDlbn  = 3;     // Div. Lines Bone
NDlmsc = 4;     // Div. Lines Muscle
NDlf   = 2;     // Div. Lines Fat
NDlsk  = 2;     // Div. Lines Skin

NL = 40; // Number of layers

Point(1)={0., 0., 0., Size}; // Centre of the Cylinder

// ====================================================================//
//                              Points                                 //
// ====================================================================//

// Circle Points Lung rlu
Point(2)={0.,rlu,0.,Size};
Point(3)={rlu,0.,0.,Size};
Point(4)={0.,-rlu,0.,Size};
Point(5)={-rlu,0.,0.,Size};

// Circle Points Bone rbn
Point(6)={0.,rbn,0.,Size};
Point(7)={rbn,0.,0.,Size};
Point(8)={0.,-rbn,0.,Size};
Point(9)={-rbn,0.,0.,Size};

// Circle Points Muscle rmsc
Point(10)={0.,rmsc,0.,Size};
Point(11)={rmsc,0.,0.,Size};
Point(12)={0.,-rmsc,0.,Size};
Point(13)={-rmsc,0.,0.,Size};

// Circle Points Fat rf
Point(14)={0.,rf,0.,Size};
Point(15)={rf,0.,0.,Size};
Point(16)={0.,-rf,0.,Size};
Point(17)={-rf,0.,0.,Size};

// Circle Points Skin rsk
Point(18)={0.,rsk,0.,Size};
Point(19)={rsk,0.,0.,Size};
Point(20)={0.,-rsk,0.,Size};
Point(21)={-rsk,0.,0.,Size};

```

```
// ====================================================================//
//                          Circumference                              //
// ====================================================================//

//Draw circle arcs Lung
Circle( 1)={2, 1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3, 1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4, 1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5, 1,2}; Transfinite Line{ 4}=NDc;

//Draw circle arcs Bone
Circle( 5)={6, 1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7, 1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8, 1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9, 1,6}; Transfinite Line{ 8}=NDc;

//Draw circle arcs Muscle
Circle( 9)={10, 1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11, 1,12}; Transfinite Line{10}=NDc;
Circle(11)={12, 1,13}; Transfinite Line{11}=NDc;
Circle(12)={13, 1,10}; Transfinite Line{12}=NDc;

//Draw circle arcs Fat
Circle(13)={14, 1,15}; Transfinite Line{13}=NDc;
Circle(14)={15, 1,16}; Transfinite Line{14}=NDc;
Circle(15)={16, 1,17}; Transfinite Line{15}=NDc;
Circle(16)={17, 1,14}; Transfinite Line{16}=NDc;

//Draw circle arcs Skin
Circle(17)={18, 1,19}; Transfinite Line{17}=NDc;
Circle(18)={19, 1,20}; Transfinite Line{18}=NDc;
Circle(19)={20, 1,21}; Transfinite Line{19}=NDc;
Circle(20)={21, 1,18}; Transfinite Line{20}=NDc;

// ====================================================================//
//             Lines that divide circumferences into 1/4               //
// ====================================================================//

// Lines Lung
Line(21)={1,2}; Transfinite Line{21}=NDllu Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDllu Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDllu Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDllu Using Progression 0.5 ;

//Lines Bone
Line(25)={2,6}; Transfinite Line{25}=NDlbn;
Line(26)={3,7}; Transfinite Line{26}=NDlbn;
Line(27)={4,8}; Transfinite Line{27}=NDlbn;
Line(28)={5,9}; Transfinite Line{28}=NDlbn;

// Lines Muscle
Line(29)={6,10}; Transfinite Line{29}=NDlmsc Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlmsc Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlmsc Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlmsc Using Progression 0.8 ;

```

```
//Lines Fat
Line(33)={10,14}; Transfinite Line{33}=NDlf;
Line(34)={11,15}; Transfinite Line{34}=NDlf;
Line(35)={12,16}; Transfinite Line{35}=NDlf;
Line(36)={13,17}; Transfinite Line{36}=NDlf;

// Lines Skin
Line(37)={14,18}; Transfinite Line{37}=NDlsk Using Progression 0.8 ;
Line(38)={15,19}; Transfinite Line{38}=NDlsk Using Progression 0.8 ;
Line(39)={16,20}; Transfinite Line{39}=NDlsk Using Progression 0.8 ;
Line(40)={17,21}; Transfinite Line{40}=NDlsk Using Progression 0.8 ;

// ===================================================================//
//                  Close loops of the 1/4 de circumf.                //
// ===================================================================//

//Close Loops Lung
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Close Loops Bone
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Close Loops Muscle
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};

//Close Loops Fat
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};

//Close Loops Skin
Line Loop(17)={37,17,-38,-13};
Line Loop(18)={38,18,-39,-14};
Line Loop(19)={39,19,-40,-15};
Line Loop(20)={40,20,-37,-16};

// ===================================================================//
//                   1/4 circle surfaces construction                 //
// ===================================================================//

// Surface Construction Lung
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
```

```
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Surface Construction Bone (pipe sections)
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Surface Construction Muscle (pipe sections)
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Surface Construction Fat (pipe sections)
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};

//Surface Construction Skin (pipe sections)
Plane Surface(17)={17}; Transfinite Surface{17}={14,18,19,15}; Recombine
    Surface{17};
Plane Surface(18)={18}; Transfinite Surface{18}={15,19,20,16}; Recombine
    Surface{18};
Plane Surface(19)={19}; Transfinite Surface{19}={16,20,21,17}; Recombine
    Surface{19};
Plane Surface(20)={20}; Transfinite Surface{20}={17,21,18,14}; Recombine
    Surface{20};

//===================================================================//

Point(22)={0., 0., zh, Size}; // Cylinder Centre

// ==================================================================//
//                                  Points                           //
// ==================================================================//

// Circle Points Heart rh
Point(23)={0.  ,rh  ,zh,Size};
Point(24)={rh ,0.   ,zh,Size};
Point(25)={0.  ,-rh ,zh,Size};
Point(26)={-rh,0.   ,zh,Size};

// Circle Points Lung rlu
Point(27)={0.  ,rlu  ,zh,Size};
Point(28)={rlu ,0.   ,zh,Size};
Point(29)={0.  ,-rlu ,zh,Size};
Point(30)={-rlu,0.   ,zh,Size};

// Circle Points Bone rbn
Point(31)={0.,rbn,zh,Size};
Point(32)={rbn,0.,zh,Size};
Point(33)={0.,-rbn,zh,Size};
Point(34)={-rbn,0.,zh,Size};

// Circle Points Muscle rmsc
Point(35)={0.,rmsc,zh,Size};
Point(36)={rmsc,0.,zh,Size};
Point(37)={0.,-rmsc,zh,Size};
Point(38)={-rmsc,0.,zh,Size};

// Circle Points Fat rf
Point(39)={0.,rf,zh,Size};
Point(40)={rf,0.,zh,Size};
Point(41)={0.,-rf,zh,Size};
Point(42)={-rf,0.,zh,Size};

// Circle Points Skin rsk
Point(43)={0.,rsk,zh,Size};
Point(44)={rsk,0.,zh,Size};
Point(45)={0.,-rsk,zh,Size};
Point(46)={-rsk,0.,zh,Size};

// ==================================================================//
//                            Circumference                         //
// ==================================================================//

//Draw circle arcs Heart
Circle( 41)={23,22,24}; Transfinite Line{ 41}=NDc;
Circle( 42)={24,22,25}; Transfinite Line{ 42}=NDc;
Circle( 43)={25,22,26}; Transfinite Line{ 43}=NDc;
Circle( 44)={26,22,23}; Transfinite Line{ 44}=NDc;

//Draw circle arcs Lung
Circle( 45)={27,22,28}; Transfinite Line{ 45}=NDc;
Circle( 46)={28,22,29}; Transfinite Line{ 46}=NDc;
Circle( 47)={29,22,30}; Transfinite Line{ 47}=NDc;
Circle( 48)={30,22,27}; Transfinite Line{ 48}=NDc;

//Draw circle arcs Bone
Circle(49)={31,22,32}; Transfinite Line{49}=NDc;
Circle(50)={32,22,33}; Transfinite Line{50}=NDc;
Circle(51)={33,22,34}; Transfinite Line{51}=NDc;
Circle(52)={34,22,31}; Transfinite Line{52}=NDc;

//Draw circle arcs Muscle
```

```
Circle(53)={35,22,36}; Transfinite Line{53}=NDc;
Circle(54)={36,22,37}; Transfinite Line{54}=NDc;
Circle(55)={37,22,38}; Transfinite Line{55}=NDc;
Circle(56)={38,22,35}; Transfinite Line{56}=NDc;

//Draw circle arcs Fat
Circle(57)={39,22,40}; Transfinite Line{57}=NDc;
Circle(58)={40,22,41}; Transfinite Line{58}=NDc;
Circle(59)={41,22,42}; Transfinite Line{59}=NDc;
Circle(60)={42,22,39}; Transfinite Line{60}=NDc;

//Draw circle arcs Skin
Circle(61)={43,22,44}; Transfinite Line{61}=NDc;
Circle(62)={44,22,45}; Transfinite Line{62}=NDc;
Circle(63)={45,22,46}; Transfinite Line{63}=NDc;
Circle(64)={46,22,43}; Transfinite Line{64}=NDc;

// ==================================================================//
//                         1/4 Circ. Division Lines                  //
// ==================================================================//

// Lines Heart
Line(65)={22,23}; Transfinite Line{65}=NDllu Using Progression 0.5 ;
Line(66)={22,24}; Transfinite Line{66}=NDllu Using Progression 0.5 ;
Line(67)={22,25}; Transfinite Line{67}=NDllu Using Progression 0.5 ;
Line(68)={22,26}; Transfinite Line{68}=NDllu Using Progression 0.5 ;

//Lines Lung
Line(69)={23,27}; Transfinite Line{69}=NDlbn;
Line(70)={24,28}; Transfinite Line{70}=NDlbn;
Line(71)={25,29}; Transfinite Line{71}=NDlbn;
Line(72)={26,30}; Transfinite Line{72}=NDlbn;

// Lines Bone
Line(73)={27,31}; Transfinite Line{73}=NDlmsc Using Progression 0.8 ;
Line(74)={28,32}; Transfinite Line{74}=NDlmsc Using Progression 0.8 ;
Line(75)={29,33}; Transfinite Line{75}=NDlmsc Using Progression 0.8 ;
Line(76)={30,34}; Transfinite Line{76}=NDlmsc Using Progression 0.8 ;

//Lines Muscle
Line(77)={31,35}; Transfinite Line{77}=NDlf;
Line(78)={32,36}; Transfinite Line{78}=NDlf;
Line(79)={33,37}; Transfinite Line{79}=NDlf;
Line(80)={34,38}; Transfinite Line{80}=NDlf;

// Lines Fat
Line(81)={35,39}; Transfinite Line{81}=NDlsk Using Progression 0.8 ;
Line(82)={36,40}; Transfinite Line{82}=NDlsk Using Progression 0.8 ;
Line(83)={37,41}; Transfinite Line{83}=NDlsk Using Progression 0.8 ;
Line(84)={38,42}; Transfinite Line{84}=NDlsk Using Progression 0.8 ;

// Lines Skin
Line(85)={39,43}; Transfinite Line{85}=NDlsk Using Progression 0.8 ;
Line(86)={40,44}; Transfinite Line{86}=NDlsk Using Progression 0.8 ;
Line(87)={41,45}; Transfinite Line{87}=NDlsk Using Progression 0.8 ;
Line(88)={42,46}; Transfinite Line{88}=NDlsk Using Progression 0.8 ;
```

```

// ==================================================================//
//                  Close loops of the 1/4 de circumf.              //
// ==================================================================//

//Close Loops Heart
Line Loop(21)={65,41,-66};
Line Loop(22)={66,42,-67};
Line Loop(23)={67,43,-68};
Line Loop(24)={68,44,-65};

//Close Loops Lung
Line Loop(25)={69,45,-70,-41};
Line Loop(26)={70,46,-71,-42};
Line Loop(27)={71,47,-72,-43};
Line Loop(28)={72,48,-69,-44};

//Close Loops Bone
Line Loop(29)={73,49,-74,-45};
Line Loop(30)={74,50,-75,-46};
Line Loop(31)={75,51,-76,-47};
Line Loop(32)={76,52,-73,-48};

//Close Loops Muscle
Line Loop(33)={77,53,-78,-49};
Line Loop(34)={78,54,-79,-50};
Line Loop(35)={79,55,-80,-51};
Line Loop(36)={80,56,-77,-52};

//Close Loops Fat
Line Loop(37)={81,57,-82,-53};
Line Loop(38)={82,58,-83,-54};
Line Loop(39)={83,59,-84,-55};
Line Loop(40)={84,60,-81,-56};

//Close Loops Skin
Line Loop(41)={85,61,-86,-57};
Line Loop(42)={86,62,-87,-58};
Line Loop(43)={87,63,-88,-59};
Line Loop(44)={88,64,-85,-60};

// ==================================================================//
//                  1/4 Circle surfaces construction                //
// ==================================================================//

// Surface Construction Heart (1/4 circle)
Plane Surface(21)={21}; Transfinite Surface{21}={22,23,24};   Recombine
    Surface{21};
Plane Surface(22)={22}; Transfinite Surface{22}={22,24,25};   Recombine
    Surface{22};
Plane Surface(23)={23}; Transfinite Surface{23}={22,25,26};   Recombine
    Surface{23};
Plane Surface(24)={24}; Transfinite Surface{24}={22,26,23};   Recombine
    Surface{24};

//Surface Construction Lung (pipe sections)
```

```
Plane Surface(25)={25}; Transfinite Surface{25}={23,27,28,24}; Recombine
    Surface{25};
Plane Surface(26)={26}; Transfinite Surface{26}={24,28,29,25}; Recombine
    Surface{26};
Plane Surface(27)={27}; Transfinite Surface{27}={25,29,30,26}; Recombine
    Surface{27};
Plane Surface(28)={28}; Transfinite Surface{28}={26,30,27,23}; Recombine
    Surface{28};

//Surface Construction Bone (pipe sections)
Plane Surface(29)={29}; Transfinite Surface{29}={27,31,32,28}; Recombine
    Surface{29};
Plane Surface(30)={30}; Transfinite Surface{30}={28,32,33,29}; Recombine
    Surface{30};
Plane Surface(31)={31}; Transfinite Surface{31}={29,33,34,30}; Recombine
    Surface{31};
Plane Surface(32)={32}; Transfinite Surface{32}={30,34,31,27}; Recombine
    Surface{32};

//Surface Construction Muscle (pipe sections)
Plane Surface(33)={33}; Transfinite Surface{33}={31,35,36,32}; Recombine
    Surface{33};
Plane Surface(34)={34}; Transfinite Surface{34}={32,36,37,33}; Recombine
    Surface{34};
Plane Surface(35)={35}; Transfinite Surface{35}={33,37,38,34}; Recombine
    Surface{35};
Plane Surface(36)={36}; Transfinite Surface{36}={34,38,35,31}; Recombine
    Surface{36};

//Surface Construction Fat (pipe sections)
Plane Surface(37)={37}; Transfinite Surface{37}={35,39,40,36}; Recombine
    Surface{37};
Plane Surface(38)={38}; Transfinite Surface{38}={36,40,41,37}; Recombine
    Surface{38};
Plane Surface(39)={39}; Transfinite Surface{39}={37,41,42,38}; Recombine
    Surface{39};
Plane Surface(40)={40}; Transfinite Surface{40}={38,42,39,35}; Recombine
    Surface{40};

//Surface Construction Skin (pipe sections)
Plane Surface(41)={41}; Transfinite Surface{41}={39,43,44,40}; Recombine
    Surface{41};
Plane Surface(42)={42}; Transfinite Surface{42}={40,44,45,41}; Recombine
    Surface{42};
Plane Surface(43)={43}; Transfinite Surface{43}={41,45,46,42}; Recombine
    Surface{43};
Plane Surface(44)={44}; Transfinite Surface{44}={42,46,43,39}; Recombine
    Surface{44};


// ==================================================================//

Point(47)={0., 0., zh+zh_lun_bn, Size}; // Cylinder Centre

// ==================================================================//
//                            Points                                 //
// ====================================================================//

// Circle Points Viscera
Point(48)={0.  ,rv  ,zh+zh_lun_bn,Size};
Point(49)={rv ,0.   ,zh+zh_lun_bn,Size};
Point(50)={0.  ,-rv ,zh+zh_lun_bn,Size};
Point(51)={-rv,0.   ,zh+zh_lun_bn,Size};

// Circle Points Muscle
Point(52)={0.  ,rmsc ,zh+zh_lun_bn,Size};
Point(53)={rmsc,0.   ,zh+zh_lun_bn,Size};
Point(54)={0.  ,-rmsc,zh+zh_lun_bn,Size};
Point(55)={-rmsc,0.  ,zh+zh_lun_bn,Size};

// Circle Points Fat
Point(56)={0.,rf ,zh+zh_lun_bn,Size};
Point(57)={rf,0. ,zh+zh_lun_bn,Size};
Point(58)={0.,-rf,zh+zh_lun_bn,Size};
Point(59)={-rf,0.,zh+zh_lun_bn,Size};

// Circle Points Skin
Point(60)={0.,rsk ,zh+zh_lun_bn,Size};
Point(61)={rsk,0. ,zh+zh_lun_bn,Size};
Point(62)={0.,-rsk,zh+zh_lun_bn,Size};
Point(63)={-rsk,0.,zh+zh_lun_bn,Size};

// ====================================================================//
//                          Circumference                              //
// ====================================================================//

//Draw circle arcs Viscera
Circle( 89)={48,47,49}; Transfinite Line{ 89}=NDc;
Circle( 90)={49,47,50}; Transfinite Line{ 90}=NDc;
Circle( 91)={50,47,51}; Transfinite Line{ 91}=NDc;
Circle( 92)={51,47,48}; Transfinite Line{ 92}=NDc;

//Draw circle arcs Muscle
Circle( 93)={52,47,53}; Transfinite Line{ 93}=NDc;
Circle( 94)={53,47,54}; Transfinite Line{ 94}=NDc;
Circle( 95)={54,47,55}; Transfinite Line{ 95}=NDc;
Circle( 96)={55,47,52}; Transfinite Line{ 96}=NDc;

//Draw circle arcs Fat
Circle( 97)={56,47,57}; Transfinite Line{ 97}=NDc;
Circle( 98)={57,47,58}; Transfinite Line{ 98}=NDc;
Circle( 99)={58,47,59}; Transfinite Line{ 99}=NDc;
Circle(100)={59,47,56}; Transfinite Line{100}=NDc;

//Draw circle arcs Skin
Circle(101)={60,47,61}; Transfinite Line{101}=NDc;
Circle(102)={61,47,62}; Transfinite Line{102}=NDc;
Circle(103)={62,47,63}; Transfinite Line{103}=NDc;
Circle(104)={63,47,60}; Transfinite Line{104}=NDc;

// ====================================================================//
//              Lines that Divide the Circ. in 1/4                     //
// ==================================================================//

// Lines Viscera
Line(105)={47,48}; Transfinite Line{105}=NDlv Using Progression 0.5 ;
Line(106)={47,49}; Transfinite Line{106}=NDlv Using Progression 0.5 ;
Line(107)={47,50}; Transfinite Line{107}=NDlv Using Progression 0.5 ;
Line(108)={47,51}; Transfinite Line{108}=NDlv Using Progression 0.5 ;

//Lines Muscle
Line(109)={48,52}; Transfinite Line{109}=NDlmsc Using Progression 0.8 ;
Line(110)={49,53}; Transfinite Line{110}=NDlmsc Using Progression 0.8 ;
Line(111)={50,54}; Transfinite Line{111}=NDlmsc Using Progression 0.8 ;
Line(112)={51,55}; Transfinite Line{112}=NDlmsc Using Progression 0.8 ;

// Lines Fat
Line(113)={52,56}; Transfinite Line{113}=NDlf Using Progression 0.8 ;
Line(114)={53,57}; Transfinite Line{114}=NDlf Using Progression 0.8 ;
Line(115)={54,58}; Transfinite Line{115}=NDlf Using Progression 0.8 ;
Line(116)={55,59}; Transfinite Line{116}=NDlf Using Progression 0.8 ;

//Lines Skin
Line(117)={56,60}; Transfinite Line{117}=NDlsk;
Line(118)={57,61}; Transfinite Line{118}=NDlsk;
Line(119)={58,62}; Transfinite Line{119}=NDlsk;
Line(120)={59,63}; Transfinite Line{120}=NDlsk;

// ==================================================================//
//                  Construction of the 1/4 Circumf.                 //
// ==================================================================//

//Close Loops Viscera
Line Loop(45)={105,89,-106};
Line Loop(46)={106,90,-107};
Line Loop(47)={107,91,-108};
Line Loop(48)={108,92,-105};

//Close Loops Muscle
Line Loop(49)={109,93,-110,-89};
Line Loop(50)={110,94,-111,-90};
Line Loop(51)={111,95,-112,-91};
Line Loop(52)={112,96,-109,-92};

//Close Loops Fat
Line Loop(53)={113, 97,-114,-93};
Line Loop(54)={114, 98,-115,-94};
Line Loop(55)={115, 99,-116,-95};
Line Loop(56)={116,100,-113,-96};

//Close Loops Skin
Line Loop(57)={117,101,-118,-97};
Line Loop(58)={118,102,-119,-98};
Line Loop(59)={119,103,-120,-99};
Line Loop(60)={120,104,-117,-100};

// ==================================================================//
//                    1/4 Circle surfaces construction               //
// ====================================================================//

// Surface Construction Viscera (1/4 circle)
Plane Surface(45)={45}; Transfinite Surface{45}={47,48,49};   Recombine
    Surface{45};
Plane Surface(46)={46}; Transfinite Surface{46}={47,49,50};   Recombine
    Surface{46};
Plane Surface(47)={47}; Transfinite Surface{47}={47,50,51};   Recombine
    Surface{47};
Plane Surface(48)={48}; Transfinite Surface{48}={47,51,48};   Recombine
    Surface{48};

//Surface Construction Muscle (pipe sections)
Plane Surface(49)={49}; Transfinite Surface{49}={48,52,53,49}; Recombine
    Surface{49};
Plane Surface(50)={50}; Transfinite Surface{50}={49,53,54,50}; Recombine
    Surface{50};
Plane Surface(51)={51}; Transfinite Surface{51}={50,54,55,51}; Recombine
    Surface{51};
Plane Surface(52)={52}; Transfinite Surface{52}={51,55,52,48}; Recombine
    Surface{52};

//Surface Construction Fat (pipe sections)
Plane Surface(53)={53}; Transfinite Surface{53}={52,56,57,53}; Recombine
    Surface{53};
Plane Surface(54)={54}; Transfinite Surface{54}={53,57,58,54}; Recombine
    Surface{54};
Plane Surface(55)={55}; Transfinite Surface{55}={54,58,59,55}; Recombine
    Surface{55};
Plane Surface(56)={56}; Transfinite Surface{56}={55,59,56,52}; Recombine
    Surface{56};

//Surface Construction Skin (pipe sections)
Plane Surface(57)={57}; Transfinite Surface{57}={56,60,61,57}; Recombine
    Surface{57};
Plane Surface(58)={58}; Transfinite Surface{58}={57,61,62,58}; Recombine
    Surface{58};
Plane Surface(59)={59}; Transfinite Surface{59}={58,62,63,59}; Recombine
    Surface{59};
Plane Surface(60)={60}; Transfinite Surface{60}={59,63,60,56}; Recombine
    Surface{60};


// ====================================================================//
//                           1st Extrusion                             //
// ====================================================================//

//Volume Construction (until the begining of the heart - 5 layers)
Extrude{0.0,0.0,zh} {Surface
    {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}; Layers{NL};
    Recombine;}

// ====================================================================//
//                           2nd Extrusion                             //
// ====================================================================//

```

```
//Volume Construction (until last layer of the heart - 6 layers)
Extrude{0.0,0.0,zh_lun_bn} {Surface
    {21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44};
     Layers{NL}; Recombine;}

// ====================================================================//
//                             3rd Extrusion                           //
// ====================================================================//

//Volume Construction (after the heart - 7 layers)
Extrude{0.0,0.0,z} {Surface
    {45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60}; Layers{NL}; Recombine
    ;}

// =====================================================================//
//                          Defining Physical Volumes                   //
// =====================================================================//

Physical Volume("Lung") = {1,2,3,4,25,26,27,28};
Physical Volume("Bone") = {5,6,7,8,29,30,31,32};
Physical Volume("Muscle") = {9,10,11,12,33,34,35,36,49,50,51,52};
Physical Volume("Fat") = {13,14,15,16,37,38,39,40,53,54,55,56};
Physical Volume("Skin") = {17,18, 19, 20,41,42,43,44,57,58,59,60};

Physical Volume("Heart") = {21,22,23,24};
Physical Volume("Viscera") = {45,46,47,48};

Show "*";
```

A.5 Thigh Geometric Code

```
// Thigh Geometry - 4 layers

// ===================================================================//
//                             Mesh Data                              //
// ===================================================================//

Size = .01; // cell size

rbn  = 0.0268; // Bone (m)
rmsc = 0.0533; // Muscle (m)
rf   = 0.0590; // Fat (m)
rsk  = 0.0615; // Skin (m)

z    = 0.440; // Thigh length (m)

// Mesh Divisions

NDc    = 5;     // Div. circ.

NDlbn  = 4;     // Div. Lines Bone
NDlmsc = 5;     // Div. Lines Muscle
NDlf   = 3;     // Div. Lines Fat
```

```
NDlsk  = 3;     // Div. Lines Skin

NL = 44; // Number of Layers

Point(1)={0., 0., 0., Size}; // Centre Cylinder

// ===================================================================//
//                              Points                                //
// ===================================================================//

// Points Circumference Bone rbn
Point(2)={0.,rbn,0.,Size};
Point(3)={rbn,0.,0.,Size};
Point(4)={0.,-rbn,0.,Size};
Point(5)={-rbn,0.,0.,Size};

// Points Circumference Fat rmsc
Point(6)={0.,rmsc,0.,Size};
Point(7)={rmsc,0.,0.,Size};
Point(8)={0.,-rmsc,0.,Size};
Point(9)={-rmsc,0.,0.,Size};

// Points Circumference Muscle rf
Point(10)={0.,rf,0.,Size};
Point(11)={rf,0.,0.,Size};
Point(12)={0.,-rf,0.,Size};
Point(13)={-rf,0.,0.,Size};

// Points Circumference Skin rsk
Point(14)={0.,rsk,0.,Size};
Point(15)={rsk,0.,0.,Size};
Point(16)={0.,-rsk,0.,Size};
Point(17)={-rsk,0.,0.,Size};


// ===================================================================//
//                         Circumferences                             //
// ===================================================================//

//Circumference Arcs Bone
Circle( 1)={2,1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3,1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4,1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5,1,2}; Transfinite Line{ 4}=NDc;

//Circumference Arcs Muscle
Circle( 5)={6,1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7,1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8,1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9,1,6}; Transfinite Line{ 8}=NDc;

//Circumference Arcs Fat
Circle( 9)={10,1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11,1,12}; Transfinite Line{10}=NDc;
Circle(11)={12,1,13}; Transfinite Line{11}=NDc;
Circle(12)={13,1,10}; Transfinite Line{12}=NDc;
```

```

//Circumference Arcs Skin
Circle(13)={14,1,15}; Transfinite Line{13}=NDc;
Circle(14)={15,1,16}; Transfinite Line{14}=NDc;
Circle(15)={16,1,17}; Transfinite Line{15}=NDc;
Circle(16)={17,1,14}; Transfinite Line{16}=NDc;

// ====================================================================//
//                         Dividing Cirfumference in 4                 //
// ====================================================================//

// Lines Bone
Line(21)={1,2}; Transfinite Line{21}=NDlbn Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDlbn Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDlbn Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDlbn Using Progression 0.5 ;

//Lines Muscle
Line(25)={2,6}; Transfinite Line{25}=NDlmsc;
Line(26)={3,7}; Transfinite Line{26}=NDlmsc;
Line(27)={4,8}; Transfinite Line{27}=NDlmsc;
Line(28)={5,9}; Transfinite Line{28}=NDlmsc;

// Lines Fat
Line(29)={6,10}; Transfinite Line{29}=NDlf Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlf Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlf Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlf Using Progression 0.8 ;

//Lines Skin
Line(33)={10,14}; Transfinite Line{33}=NDlsk;
Line(34)={11,15}; Transfinite Line{34}=NDlsk;
Line(35)={12,16}; Transfinite Line{35}=NDlsk;
Line(36)={13,17}; Transfinite Line{36}=NDlsk;

// ====================================================================//
//                         1/4 Circumf. Construction                   //
// ====================================================================//

//Close the Loops Bone
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Close the Loops Muscle
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Close the Loops Fat
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};
```

```

//Close the Loops Skin
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};

// ==================================================================//
//                    1/4 Circle Surfaces Construction                //
// ==================================================================//

// Defining Surfaces Bone (1/4 circle)
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Defining Surfaces Muscle (Pipe sections)
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Defining Surfaces Fat (Pipe sections)
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Defining Surfaces Skin (Pipe sections)
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};

// ==================================================================//
//                          Volume Construction                       //
// ==================================================================//

```

```
Extrude{0.0,0.0,z} {Surface{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}; Layers{
    NL}; Recombine;}

// ===================================================================//
//                      Physical Volume Construction                 //
// ===================================================================//

Physical Volume("Bone") = {1,2,3,4};
Physical Volume("Muscle") = {5,6,7,8};
Physical Volume("Fat") = {9,10,11,12};
Physical Volume("Skin") = {13,14,15,16};

Show "*";
```

A.6 Head Geometric Code

```
// Head Geometry

// Mesh Data

Size = .01;


rlu  = 0.0517; // Brain (m)
rbn  = 0.0615; // Bone (m)
rmsc = 0.0664; // Muscle (m)
rf   = 0.0724; // Fat (m)
rsk  = 0.0751; // Skin (m)

zh   = 0.1604; // init brain layer (m)
z    = 0.0098; // head lenght (m)
zmsc = 0.0020;

// Mesh Division

NDc = 5;        // Div. circ.

//NDlh   = 4;
NDllu  = 7;     // Div. Linhas Brain
NDlv   = 8;   // Div.
NDlbn  = 3;     // Div. Linhas Osso
NDlmsc = 4;     // Div. Linhas Musculo
NDlf   = 2;     // Div. Linhas Gordura
NDlsk  = 2;     // Div. Linhas Pele

NL = 5; // Number of layers

Point(1)={0., 0., 0., Size}; // Centro do Cilindro

//
    ================================================================================
//                                  Pontos
          //
//
    ===============================================================================


// Pontos da Circunferencia Brain rlu
Point(2)={0.,rlu,0.,Size};
Point(3)={rlu,0.,0.,Size};
Point(4)={0.,-rlu,0.,Size};
Point(5)={-rlu,0.,0.,Size};

// Pontos da Circunferencia Osso rbn
Point(6)={0.,rbn,0.,Size};
Point(7)={rbn,0.,0.,Size};
Point(8)={0.,-rbn,0.,Size};
Point(9)={-rbn,0.,0.,Size};

// Pontos da Circunferencia Musculo rmsc
Point(10)={0.,rmsc,0.,Size};
Point(11)={rmsc,0.,0.,Size};
Point(12)={0.,-rmsc,0.,Size};
Point(13)={-rmsc,0.,0.,Size};

// Pontos da Circunferencia Gordura rf
Point(14)={0.,rf,0.,Size};
Point(15)={rf,0.,0.,Size};
Point(16)={0.,-rf,0.,Size};
Point(17)={-rf,0.,0.,Size};

// Pontos da Circunferencia Pele rsk
Point(18)={0.,rsk,0.,Size};
Point(19)={rsk,0.,0.,Size};
Point(20)={0.,-rsk,0.,Size};
Point(21)={-rsk,0.,0.,Size};

//
    =============================================================================//

//                          Circunferencias
          //
//
    =============================================================================//


//Desenho dos Arcos de circunferencia Pulmao
Circle( 1)={2, 1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3, 1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4, 1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5, 1,2}; Transfinite Line{ 4}=NDc;

//Desenho dos Arcos de Circunferencia Osso
Circle( 5)={6, 1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7, 1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8, 1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9, 1,6}; Transfinite Line{ 8}=NDc;
```

```

//Desenho dos Arcos de circunferencia Musculo
Circle( 9)={10, 1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11, 1,12}; Transfinite Line{10}=NDc;
Circle(11)={12, 1,13}; Transfinite Line{11}=NDc;
Circle(12)={13, 1,10}; Transfinite Line{12}=NDc;

//Desenho dos Arcos de Circunferencia Gordura
Circle(13)={14, 1,15}; Transfinite Line{13}=NDc;
Circle(14)={15, 1,16}; Transfinite Line{14}=NDc;
Circle(15)={16, 1,17}; Transfinite Line{15}=NDc;
Circle(16)={17, 1,14}; Transfinite Line{16}=NDc;

//Desenho dos Arcos de Circunferencia Pele
Circle(17)={18, 1,19}; Transfinite Line{17}=NDc;
Circle(18)={19, 1,20}; Transfinite Line{18}=NDc;
Circle(19)={20, 1,21}; Transfinite Line{19}=NDc;
Circle(20)={21, 1,18}; Transfinite Line{20}=NDc;

//
    ====================================================================================//
//                        Linhas de divisao da Circ. em 1/4
          //
//
    ====================================================================================//

// Linhas Brain (dividir o tronco em quatro)
Line(21)={1,2}; Transfinite Line{21}=NDllu Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDllu Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDllu Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDllu Using Progression 0.5 ;

//Linhas Osso (dividir o tronco em quatro)
Line(25)={2,6}; Transfinite Line{25}=NDlbn;
Line(26)={3,7}; Transfinite Line{26}=NDlbn;
Line(27)={4,8}; Transfinite Line{27}=NDlbn;
Line(28)={5,9}; Transfinite Line{28}=NDlbn;

// Linhas Musculo (dividir o tronco em quatro)
Line(29)={6,10}; Transfinite Line{29}=NDlmsc Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlmsc Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlmsc Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlmsc Using Progression 0.8 ;

//Linhas Gordura (dividir o tronco em quatro)
Line(33)={10,14}; Transfinite Line{33}=NDlf;
Line(34)={11,15}; Transfinite Line{34}=NDlf;
Line(35)={12,16}; Transfinite Line{35}=NDlf;
Line(36)={13,17}; Transfinite Line{36}=NDlf;

// Linhas Pele (dividir o tronco em quatro)
Line(37)={14,18}; Transfinite Line{37}=NDlsk Using Progression 0.8 ;
Line(38)={15,19}; Transfinite Line{38}=NDlsk Using Progression 0.8 ;
Line(39)={16,20}; Transfinite Line{39}=NDlsk Using Progression 0.8 ;
Line(40)={17,21}; Transfinite Line{40}=NDlsk Using Progression 0.8 ;


// ===============================================================================//
//                          Construcao dos 1/4 de circunf.                          //
// ===============================================================================//


//Fecho dos Modules Cerebro
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Fecho dos Modules Osso
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Fecho dos Modules Musculo
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};

//Fecho dos Modules Gordura
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};

//Fecho dos Modules Pele
Line Loop(17)={37,17,-38,-13};
Line Loop(18)={38,18,-39,-14};
Line Loop(19)={39,19,-40,-15};
Line Loop(20)={40,20,-37,-16};

// ===============================================================================//
//                    Construcao das sup. dos 1/4 de circunf.                    //
// ===============================================================================//

// Definition das Superficies Pulmao
```

```
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Definition das Superficies Osso
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Definition das Superficies Musculo
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Definition das Superficies Gordura
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};

//Definition das Superficies Pele
Plane Surface(17)={17}; Transfinite Surface{17}={14,18,19,15}; Recombine
    Surface{17};
Plane Surface(18)={18}; Transfinite Surface{18}={15,19,20,16}; Recombine
    Surface{18};
Plane Surface(19)={19}; Transfinite Surface{19}={16,20,21,17}; Recombine
    Surface{19};
Plane Surface(20)={20}; Transfinite Surface{20}={17,21,18,14}; Recombine
    Surface{20};

//=============================================================================


Point(47)={0., 0., zh, Size}; // Centro do Cilindro

//
    ===========================================================================
//                                    Pontos
                                                   //
// ================================================================================

// Pontos da Circunferencia Visceras
Point(48)={0.   ,rbn   ,zh  ,Size};
Point(49)={rbn  ,0.    ,zh  ,Size};
Point(50)={0.   ,-rbn  ,zh  ,Size};
Point(51)={-rbn,0.    ,zh  ,Size};

// Pontos da Circunferencia Musulo
Point(52)={0.   ,rmsc  ,zh  ,Size};
Point(53)={rmsc,0.    ,zh  ,Size};
Point(54)={0.   ,-rmsc,zh  ,Size};
Point(55)={-rmsc,0.   ,zh  ,Size};

// Pontos da Circunferencia Gordura
Point(56)={0.,rf  ,zh  ,Size};
Point(57)={rf,0.  ,zh  ,Size};
Point(58)={0.,-rf,zh  ,Size};
Point(59)={-rf,0.,zh  ,Size};

// Pontos da Circunferencia Pele
Point(60)={0.,rsk  ,zh  ,Size};
Point(61)={rsk,0.  ,zh  ,Size};
Point(62)={0.,-rsk,zh  ,Size};
Point(63)={-rsk,0.,zh  ,Size};

// ==============================================================================//

//                          Circunferencias
          //
// ==============================================================================//

//Desenho dos Arcos de circunferencia Visceras
Circle( 89)={48,47,49}; Transfinite Line{ 89}=NDc;
Circle( 90)={49,47,50}; Transfinite Line{ 90}=NDc;
Circle( 91)={50,47,51}; Transfinite Line{ 91}=NDc;
Circle( 92)={51,47,48}; Transfinite Line{ 92}=NDc;

//Desenho dos Arcos de Circunferencia Musculo
Circle( 93)={52,47,53}; Transfinite Line{ 93}=NDc;
Circle( 94)={53,47,54}; Transfinite Line{ 94}=NDc;
Circle( 95)={54,47,55}; Transfinite Line{ 95}=NDc;
Circle( 96)={55,47,52}; Transfinite Line{ 96}=NDc;

//Desenho dos Arcos de circunferencia Gordura
Circle( 97)={56,47,57}; Transfinite Line{ 97}=NDc;
Circle( 98)={57,47,58}; Transfinite Line{ 98}=NDc;
Circle( 99)={58,47,59}; Transfinite Line{ 99}=NDc;
Circle(100)={59,47,56}; Transfinite Line{100}=NDc;

//Desenho dos Arcos de Circunferencia Pele
Circle(101)={60,47,61}; Transfinite Line{101}=NDc;
Circle(102)={61,47,62}; Transfinite Line{102}=NDc;
Circle(103)={62,47,63}; Transfinite Line{103}=NDc;
Circle(104)={63,47,60}; Transfinite Line{104}=NDc;

//
    ===============================================================================//
//                         Linhas de divisao da Circ. em 1/4
          //
//
    ===============================================================================//

// Linhas bone (dividir em quatro)
Line(105)={47,48}; Transfinite Line{105}=NDlv ;
Line(106)={47,49}; Transfinite Line{106}=NDlv ;
Line(107)={47,50}; Transfinite Line{107}=NDlv ;
Line(108)={47,51}; Transfinite Line{108}=NDlv ;

//Linhas Musculo (dividir em quatro)
Line(109)={48,52}; Transfinite Line{109}=NDlmsc ;
Line(110)={49,53}; Transfinite Line{110}=NDlmsc ;
Line(111)={50,54}; Transfinite Line{111}=NDlmsc ;
Line(112)={51,55}; Transfinite Line{112}=NDlmsc ;

// Linhas Gordura (dividir em quatro)
Line(113)={52,56}; Transfinite Line{113}=NDlf ;
Line(114)={53,57}; Transfinite Line{114}=NDlf ;
Line(115)={54,58}; Transfinite Line{115}=NDlf ;
Line(116)={55,59}; Transfinite Line{116}=NDlf ;

//Linhas Pele (dividir em quatro)
Line(117)={56,60}; Transfinite Line{117}=NDlsk;
Line(118)={57,61}; Transfinite Line{118}=NDlsk;
Line(119)={58,62}; Transfinite Line{119}=NDlsk;
Line(120)={59,63}; Transfinite Line{120}=NDlsk;

//
    ===============================================================================//
//                         Build dos 1/4 de circunf.
     //
//
    ===============================================================================//

//Fecho dos Modules bone
Line Loop(45)={105,89,-106};
Line Loop(46)={106,90,-107};
Line Loop(47)={107,91,-108};
Line Loop(48)={108,92,-105};

//Fecho dos Modules Musculo
Line Loop(49)={109,93,-110,-89};
Line Loop(50)={110,94,-111,-90};
Line Loop(51)={111,95,-112,-91};
Line Loop(52)={112,96,-109,-92};

//Fecho dos Modules Gordura
Line Loop(53)={113, 97,-114,-93};
Line Loop(54)={114, 98,-115,-94};
Line Loop(55)={115, 99,-116,-95};
Line Loop(56)={116,100,-113,-96};

//Fecho dos Modules Pele
Line Loop(57)={117,101,-118,-97};
Line Loop(58)={118,102,-119,-98};
Line Loop(59)={119,103,-120,-99};
Line Loop(60)={120,104,-117,-100};

//
    ==============================================================================//
//                    Build das sup. dos 1/4 de circunf.
     //
//
    ==============================================================================//

// Definition das Superfcies Bone
Plane Surface(45)={45}; Transfinite Surface{45}={47,48,49};   Recombine
    Surface{45};
Plane Surface(46)={46}; Transfinite Surface{46}={47,49,50};   Recombine
    Surface{46};
Plane Surface(47)={47}; Transfinite Surface{47}={47,50,51};   Recombine
    Surface{47};
Plane Surface(48)={48}; Transfinite Surface{48}={47,51,48};   Recombine
    Surface{48};

//Definition das Superficies Musculo
Plane Surface(49)={49}; Transfinite Surface{49}={48,52,53,49}; Recombine
    Surface{49};
Plane Surface(50)={50}; Transfinite Surface{50}={49,53,54,50}; Recombine
    Surface{50};
Plane Surface(51)={51}; Transfinite Surface{51}={50,54,55,51}; Recombine
    Surface{51};
Plane Surface(52)={52}; Transfinite Surface{52}={51,55,52,48}; Recombine
    Surface{52};

//Definition das Superficies Gordura
Plane Surface(53)={53}; Transfinite Surface{53}={52,56,57,53}; Recombine
    Surface{53};
Plane Surface(54)={54}; Transfinite Surface{54}={53,57,58,54}; Recombine
    Surface{54};
Plane Surface(55)={55}; Transfinite Surface{55}={54,58,59,55}; Recombine
    Surface{55};
```

```
Plane Surface(56)={56}; Transfinite Surface{56}={55,59,56,52}; Recombine Surface{56};

//Definition das Superficies Pele
Plane Surface(57)={57}; Transfinite Surface{57}={56,60,61,57}; Recombine Surface{57};
Plane Surface(58)={58}; Transfinite Surface{58}={57,61,62,58}; Recombine Surface{58};
Plane Surface(59)={59}; Transfinite Surface{59}={58,62,63,59}; Recombine Surface{59};
Plane Surface(60)={60}; Transfinite Surface{60}={59,63,60,56}; Recombine Surface{60};

//================================================================================//

Point(64)={0., 0., zh+z, Size}; // Cylinder centre

//================================================================================//
//                                  Points                                        //
//================================================================================//


// Pontos da Circunferencia Musulo
Point(65)={0.  ,rmsc ,zh+z,Size};
Point(66)={rmsc,0.   ,zh+z,Size};
Point(67)={0.  ,-rmsc,zh+z,Size};
Point(68)={-rmsc,0.  ,zh+z,Size};

// Pontos da Circunferencia Gordura
Point(69)={0.,rf ,zh+z,Size};
Point(70)={rf,0. ,zh+z,Size};
Point(71)={0.,-rf,zh+z,Size};
Point(72)={-rf,0.,zh+z,Size};

// Pontos da Circunferencia Pele
Point(73)={0.,rsk ,zh+z,Size};
Point(74)={rsk,0. ,zh+z,Size};
Point(75)={0.,-rsk,zh+z,Size};
Point(76)={-rsk,0.,zh+z,Size};

//================================================================================//
//                               Circumferences                                   //
//================================================================================//

//Desenho dos Arcos de Circunferencia Musculo
```

```
Circle(121)={65,64,66}; Transfinite Line{121}=NDc;
Circle(122)={66,64,67}; Transfinite Line{122}=NDc;
Circle(123)={67,64,68}; Transfinite Line{123}=NDc;
Circle(124)={68,64,65}; Transfinite Line{124}=NDc;

//Desenho dos Arcos de circunferencia Gordura
Circle(125)={69,64,70}; Transfinite Line{125}=NDc;
Circle(126)={70,64,71}; Transfinite Line{126}=NDc;
Circle(127)={71,64,72}; Transfinite Line{127}=NDc;
Circle(128)={72,64,69}; Transfinite Line{128}=NDc;

//Desenho dos Arcos de Circunferencia Pele
Circle(129)={73,64,74}; Transfinite Line{129}=NDc;
Circle(130)={74,64,75}; Transfinite Line{130}=NDc;
Circle(131)={75,64,76}; Transfinite Line{131}=NDc;
Circle(132)={76,64,73}; Transfinite Line{132}=NDc;

// ===============================================================================//
//                         Linhas de divisao da Circ. em 1/4                   //
// ===============================================================================//

//Linhas Musculo (dividir em quatro)
Line(133)={64,65}; Transfinite Line{133}=NDlmsc ;
Line(134)={64,66}; Transfinite Line{134}=NDlmsc ;
Line(135)={64,67}; Transfinite Line{135}=NDlmsc ;
Line(136)={64,68}; Transfinite Line{136}=NDlmsc ;

// Linhas Gordura (dividir em quatro)
Line(137)={65,69}; Transfinite Line{137}=NDlf ;
Line(138)={66,70}; Transfinite Line{138}=NDlf ;
Line(139)={67,71}; Transfinite Line{139}=NDlf ;
Line(140)={68,72}; Transfinite Line{140}=NDlf ;

//Linhas Pele (dividir em quatro)
Line(141)={69,73}; Transfinite Line{141}=NDlsk;
Line(142)={70,74}; Transfinite Line{142}=NDlsk;
Line(143)={71,75}; Transfinite Line{143}=NDlsk;
Line(144)={72,76}; Transfinite Line{144}=NDlsk;

// ===============================================================================//
//                         Build dos 1/4 de circunf.                         //
// ===============================================================================//

//Fecho dos Modules Musculo
Line Loop(61)={133,121,-134};
Line Loop(62)={134,122,-135};
Line Loop(63)={135,123,-136};
Line Loop(64)={136,124,-133};

//Fecho dos Modules Gordura
Line Loop(65)={137,125,-138,-121};
Line Loop(66)={138,126,-139,-122};
Line Loop(67)={139,127,-140,-123};
Line Loop(68)={140,128,-137,-124};

//Fecho dos Modules Pele
Line Loop(69)={141,129,-142,-125};
Line Loop(70)={142,130,-143,-126};
Line Loop(71)={143,131,-144,-127};
Line Loop(72)={144,132,-141,-128};

//
    ==============================================================================//

//                    Build das sup. dos 1/4 de circunf.
     //
//
    ==============================================================================//


//Definition das Superficies Musculo
Plane Surface(61)={61}; Transfinite Surface{61}={64,65,66}; Recombine
    Surface{61};
Plane Surface(62)={62}; Transfinite Surface{62}={64,66,67}; Recombine
    Surface{62};
Plane Surface(63)={63}; Transfinite Surface{63}={64,67,68}; Recombine
    Surface{63};
Plane Surface(64)={64}; Transfinite Surface{64}={64,68,65}; Recombine
    Surface{64};

//Definition das Superficies Gordura
Plane Surface(65)={65}; Transfinite Surface{65}={65,69,70,66}; Recombine
    Surface{65};
Plane Surface(66)={66}; Transfinite Surface{66}={66,70,71,67}; Recombine
    Surface{66};
Plane Surface(67)={67}; Transfinite Surface{67}={67,71,72,68}; Recombine
    Surface{67};
Plane Surface(68)={68}; Transfinite Surface{68}={68,72,69,65}; Recombine
    Surface{68};

//Definition das Superficies Pele
Plane Surface(69)={69}; Transfinite Surface{69}={69,73,74,70}; Recombine
    Surface{69};
Plane Surface(70)={70}; Transfinite Surface{70}={70,74,75,71}; Recombine
    Surface{70};
Plane Surface(71)={71}; Transfinite Surface{71}={71,75,76,72}; Recombine
    Surface{71};
Plane Surface(72)={72}; Transfinite Surface{72}={72,76,73,69}; Recombine
    Surface{72};

//=====================================================================================

// ======================================================================================//
//                                    Volumes Construction                                  //
// ======================================================================================//

//Volumes Construction  (5 Layers)
Extrude{0.0,0.0,zh} {Surface {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}; Layers{NL}; Recombine;}

// ======================================================================================//

//Volumes Construction  (4 Layers)
Extrude{0.0,0.0,z} {Surface {45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60}; Layers{1}; Recombine;}

// ======================================================================================//

//Volumes Construction (3 Layers)
Extrude{0.0,0.0,zmsc} {Surface{61,62,63,64,65,66,67,68,69,70,71,72}; Layers{1}; Recombine;}

// ======================================================================================//
//                                   Defining Physical Volumes                              //
// ======================================================================================//

Physical Volume("Brain") = {1,2,3,4};
Physical Volume("Bone") = {5,6,7,8,45,46,47,48};
Physical Volume("Muscle") = {9,10,11,12,49,50,51,52,61,62,63,64};
Physical Volume("Fat") = {13,14,15,16,53,54,55,56,65,66,67,68};
Physical Volume("Skin") = {17,18,19,20,57,58,59,60,69,70,71,72};
```

A.7 Leg Geometric Code

```
// Geometria do Perna - 4 camadas

// ==================================================================//
//                               Mesh Data                            //
// ==================================================================//

Size = .01;

rbn1  = 0.0143; // Bone (m)
rbn2  = 0.0343; // Bone (m)
rmsc = 0.0372; // Musculo (m)
rf   = 0.0412; // Fat (m)
rsk  = 0.0429; // Skin (m)

z    = 0.40; // comprimento da perna (m)

// Divisions da Rede

NDc    = 5;     // Div. circ.

NDlbn1  = 2;     // Div. Lines Bone
NDlbn2  = 2;     // Div. Lines Bone
NDlmsc = 2;     // Div. Lines Muscle
NDlf   = 2;     // Div. Lines Fat
NDlsk  = 2;     // Div. Lines Skin

NL = 8; // Number of Layers

Point(1)={0., 0., 0., Size}; // Centro do Cilindro


// ==================================================================//
//                              Points                              //
// ==================================================================//

// Circumference Points Bone init rbn1 rbn1
Point(2)={0.,rbn1,0.,Size};
Point(3)={rbn1,0.,0.,Size};
Point(4)={0.,-rbn1,0.,Size};
Point(5)={-rbn1,0.,0.,Size};

// Circumference Points Bone final rbn2
Point(6)={0.,rbn2,0.,Size};
Point(7)={rbn2,0.,0.,Size};
Point(8)={0.,-rbn2,0.,Size};
Point(9)={-rbn2,0.,0.,Size};

/ Circumference Points Muscle rmsc
Point(10)={0.,rmsc,0.,Size};
Point(11)={rmsc,0.,0.,Size};
Point(12)={0.,-rmsc,0.,Size};
Point(13)={-rmsc,0.,0.,Size};

/ Circumference Points Fat rf
Point(14)={0.,rf,0.,Size};
Point(15)={rf,0.,0.,Size};
Point(16)={0.,-rf,0.,Size};
Point(17)={-rf,0.,0.,Size};

/ Circumference Points Skin rsk
```

```
Point(18)={0.,rsk,0.,Size};
Point(19)={rsk,0.,0.,Size};
Point(20)={0.,-rsk,0.,Size};
Point(21)={-rsk,0.,0.,Size};

//
    ==================================================================================//

//                            Circumf.
    //
//
    ==================================================================================//


//Circumf. Arcs Bone1
Circle( 1)={2,1,3}; Transfinite Line{ 1}=NDc;
Circle( 2)={3,1,4}; Transfinite Line{ 2}=NDc;
Circle( 3)={4,1,5}; Transfinite Line{ 3}=NDc;
Circle( 4)={5,1,2}; Transfinite Line{ 4}=NDc;

//Circumf. Arcs Bone2
Circle( 5)={6,1,7}; Transfinite Line{ 5}=NDc;
Circle( 6)={7,1,8}; Transfinite Line{ 6}=NDc;
Circle( 7)={8,1,9}; Transfinite Line{ 7}=NDc;
Circle( 8)={9,1,6}; Transfinite Line{ 8}=NDc;

//Circumf. Arcs Muscle
Circle( 9)={10,1,11}; Transfinite Line{ 9}=NDc;
Circle(10)={11,1,12}; Transfinite Line{10}=NDc;
Circle(11)={12,1,13}; Transfinite Line{11}=NDc;
Circle(12)={13,1,10}; Transfinite Line{12}=NDc;

//Circumf. Arcs Fat
Circle(13)={14,1,15}; Transfinite Line{13}=NDc;
Circle(14)={15,1,16}; Transfinite Line{14}=NDc;
Circle(15)={16,1,17}; Transfinite Line{15}=NDc;
Circle(16)={17,1,14}; Transfinite Line{16}=NDc;

//Circumf. Arcs Skin
Circle(17)={18,1,19}; Transfinite Line{17}=NDc;
Circle(18)={19,1,20}; Transfinite Line{18}=NDc;
Circle(19)={20,1,21}; Transfinite Line{19}=NDc;
Circle(20)={21,1,18}; Transfinite Line{20}=NDc;


// ====================================================================//
//                       1/4 Circle division lines                     //
// ====================================================================//

// Lines Bone1
Line(21)={1,2}; Transfinite Line{21}=NDlbn1 Using Progression 0.5 ;
Line(22)={1,3}; Transfinite Line{22}=NDlbn1 Using Progression 0.5 ;
Line(23)={1,4}; Transfinite Line{23}=NDlbn1 Using Progression 0.5 ;
Line(24)={1,5}; Transfinite Line{24}=NDlbn1 Using Progression 0.5 ;

//Lines Bone2
```

```
Line(25)={2,6}; Transfinite Line{25}=NDlbn2;
Line(26)={3,7}; Transfinite Line{26}=NDlbn2;
Line(27)={4,8}; Transfinite Line{27}=NDlbn2;
Line(28)={5,9}; Transfinite Line{28}=NDlbn2;

// Lines Muscle
Line(29)={6,10}; Transfinite Line{29}=NDlmsc Using Progression 0.8 ;
Line(30)={7,11}; Transfinite Line{30}=NDlmsc Using Progression 0.8 ;
Line(31)={8,12}; Transfinite Line{31}=NDlmsc Using Progression 0.8 ;
Line(32)={9,13}; Transfinite Line{32}=NDlmsc Using Progression 0.8 ;

//Lines Fat
Line(33)={10,14}; Transfinite Line{33}=NDlf;
Line(34)={11,15}; Transfinite Line{34}=NDlf;
Line(35)={12,16}; Transfinite Line{35}=NDlf;
Line(36)={13,17}; Transfinite Line{36}=NDlf;

// Lines Skin
Line(37)={14,18}; Transfinite Line{37}=NDlsk Using Progression 0.8 ;
Line(38)={15,19}; Transfinite Line{38}=NDlsk Using Progression 0.8 ;
Line(39)={16,20}; Transfinite Line{39}=NDlsk Using Progression 0.8 ;
Line(40)={17,21}; Transfinite Line{40}=NDlsk Using Progression 0.8 ;

// ====================================================================//
//                        1/4 circle Sections                          //
// ====================================================================//


//Close Loops Bone
Line Loop(1)={21,1,-22};
Line Loop(2)={22,2,-23};
Line Loop(3)={23,3,-24};
Line Loop(4)={24,4,-21};

//Close Loops Bone
Line Loop(5)={25,5,-26,-1};
Line Loop(6)={26,6,-27,-2};
Line Loop(7)={27,7,-28,-3};
Line Loop(8)={28,8,-25,-4};

//Close Loops Muscle
Line Loop( 9)={29, 9,-30,-5};
Line Loop(10)={30,10,-31,-6};
Line Loop(11)={31,11,-32,-7};
Line Loop(12)={32,12,-29,-8};

//Close Loops Fat
Line Loop(13)={33,13,-34, -9};
Line Loop(14)={34,14,-35,-10};
Line Loop(15)={35,15,-36,-11};
Line Loop(16)={36,16,-33,-12};

////Close Loops Skin
Line Loop(17)={37,17,-38,-13};
Line Loop(18)={38,18,-39,-14};
Line Loop(19)={39,19,-40,-15};
Line Loop(20)={40,20,-37,-16};

// ===================================================================//
//                 1/4 Circumf. Surface Construction                  //
// ===================================================================//

// Defining Surfaces Bone (1/4 circle)
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 3};   Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 1, 3, 4};   Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 1, 4, 5};   Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 1, 5, 2};   Recombine
    Surface{4};

//Defining Surfaces Bone (pipe sections)
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 2, 6, 7, 3}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 3, 7, 8, 4}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 4, 8, 9, 5}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 5, 9, 6, 2}; Recombine
    Surface{8};

//Defining Surfaces Muscle (pipe sections)
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 6,10,11, 7}; Recombine
    Surface{ 9};
Plane Surface(10)={10}; Transfinite Surface{10}={ 7,11,12, 8}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={ 8,12,13, 9}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={ 9,13,10, 6}; Recombine
    Surface{12};

//Defining Surfaces Fat (pipe sections)
Plane Surface(13)={13}; Transfinite Surface{13}={10,14,15,11}; Recombine
    Surface{13};
Plane Surface(14)={14}; Transfinite Surface{14}={11,15,16,12}; Recombine
    Surface{14};
Plane Surface(15)={15}; Transfinite Surface{15}={12,16,17,13}; Recombine
    Surface{15};
Plane Surface(16)={16}; Transfinite Surface{16}={13,17,14,10}; Recombine
    Surface{16};

//Defining Surfaces Skin (pipe sections)
Plane Surface(17)={17}; Transfinite Surface{17}={14,18,19,15}; Recombine
    Surface{17};
Plane Surface(18)={18}; Transfinite Surface{18}={15,19,20,16}; Recombine
    Surface{18};
Plane Surface(19)={19}; Transfinite Surface{19}={16,20,21,17}; Recombine
    Surface{19};
Plane Surface(20)={20}; Transfinite Surface{20}={17,21,18,14}; Recombine
    Surface{20};

```

```
// ====================================================================//
//                          Volume Construction                         //
// ====================================================================//

Extrude{0.0,0.0,z} {Surface
    {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20}; Layers{NL};
    Recombine;}

// ====================================================================//
//                    Defining Physical Volumes                         //
// ====================================================================//

Physical Volume("Bone") = {6,8};
Physical Volume("Muscle") = {1,2,3,4,5,7,9,10,11,12};
Physical Volume("Fat") = {13,14,15,16};
Physical Volume("Skin") = {17,18,19,20};
```

A.8 Foot Geometric Code

```
// Geometria do foot - 4 layers

// Mesh Data


Size = .01;

xsk1  = 0.0; // Bone (m)
xsk2 = 0.1021; // Muscle (m)
ysk1   = 0.0; // Fat (m)
ysk2  = 0.0370; // Skin (m)
xft1 = 0.0021;
xft2 = 0.1000;
yft1 = 0.00076;
yft2 = 0.0362;
xmsc1 = 0.0068;
xmsc2 = 0.0953;
ymsc1 = 0.00246;
ymsc2 = 0.0345;
xbn1 = 0.0288;
xbn2 = 0.0733;
ybn1 = 0.0104;
ybn2 = 0.0265;



z    = 0.19; // lenght (m)

// Mesh Divisions

NDlx  = 10;     // Div. Lines Bone
NDly = 5;     // Div. Lines Muscle
NDlz   = 19;     // Div. Lines Gord

```

```
//
    ================================================================================

//                                    Pontos
            //
//
    ================================================================================


// Prisma Corners Skin
Point(1)={0., 0., 0., Size};
Point(2)={0., ysk2, 0., Size};
Point(3)={xsk2, ysk2, 0., Size};
Point(4)={xsk2, 0., 0., Size};

// Prisma Corners  Fat
Point(5)={xft1, yft1, 0., Size};
Point(6)={xft1, yft2, 0., Size};
Point(7)={xft2, yft2, 0., Size};
Point(8)={xft2, yft1, 0., Size};

// Prisma Corners Muscle
Point(9) ={xmsc1, ymsc1, 0., Size};
Point(10)={xmsc1, ymsc2, 0., Size};
Point(11)={xmsc2, ymsc2, 0., Size};
Point(12)={xmsc2, ymsc1, 0., Size};

// Prisma Corners Bone
Point(13)={xbn1, ybn1, 0., Size};
Point(14)={xbn1, ybn2, 0., Size};
Point(15)={xbn2, ybn2, 0., Size};
Point(16)={xbn2, ybn1, 0., Size};


//Lines SK
Line(1)={1,2}; Transfinite Line{1}=4;
Line(2)={2,3}; Transfinite Line{2}=10;
Line(3)={3,4}; Transfinite Line{3}=4;
Line(4)={4,1}; Transfinite Line{4}=10;

//Lines Ft
Line(5)={5,6}; Transfinite Line{5}=4;
Line(6)={6,7}; Transfinite Line{6}=10;
Line(7)={7,8}; Transfinite Line{7}=4;
Line(8)={8,5}; Transfinite Line{8}=10;

//Lines Msc
Line(9) ={9, 10}; Transfinite Line{9}=4;
Line(10)={10,11}; Transfinite Line{10}=10;
Line(11)={11,12}; Transfinite Line{11}=4;
Line(12)={12, 9}; Transfinite Line{12}=10;

//Lines Bone2
Line(13)={13, 14}; Transfinite Line{13}=4;
Line(14)={14, 15}; Transfinite Line{14}=10;
Line(15)={15, 16}; Transfinite Line{15}=4;
Line(16)={16, 13}; Transfinite Line{16}=10;

//Lines
Line(17)={1,5}; Transfinite Line{17}=2;
Line(18)={2,6}; Transfinite Line{18}=2;
Line(19)={3,7}; Transfinite Line{19}=2;
Line(20)={4,8}; Transfinite Line{20}=2;

Line(21)={5, 9}; Transfinite Line{21}=2;
Line(22)={6,10}; Transfinite Line{22}=2;
Line(23)={7,11}; Transfinite Line{23}=2;
Line(24)={8,12}; Transfinite Line{24}=2;

Line(25)={ 9,13}; Transfinite Line{25}=4;
Line(26)={10,14}; Transfinite Line{26}=4;
Line(27)={11,15}; Transfinite Line{27}=4;
Line(28)={12,16}; Transfinite Line{28}=4;

//Close Modules Sk, Ft, Msc, Bn
Line Loop(1)={ 1, 18, -5, -17};
Line Loop(2)={ 2, 19, -6, -18};
Line Loop(3)={ 3, 20, -7, -19};
Line Loop(4)={ 4, 17, -8, -20};

Line Loop(5)={ 5, 22, -9, -21};
Line Loop(6)={ 6, 23,-10, -22};
Line Loop(7)={ 7, 24,-11, -23};
Line Loop(8)={ 8, 21,-12, -24};

Line Loop( 9)={  9, 26,-13, -25};
Line Loop(10)={ 10, 27,-14, -26};
Line Loop(11)={ 11, 28,-15, -27};
Line Loop(12)={ 12, 25,-16, -28};

Line Loop(13)={13,14,15,16};


//Defining Surfaces
Plane Surface( 1)={ 1}; Transfinite Surface{ 1}={ 1, 2, 6, 5}; Recombine
    Surface{1};
Plane Surface( 2)={ 2}; Transfinite Surface{ 2}={ 2, 3, 7, 6}; Recombine
    Surface{2};
Plane Surface( 3)={ 3}; Transfinite Surface{ 3}={ 3, 4, 8, 7}; Recombine
    Surface{3};
Plane Surface( 4)={ 4}; Transfinite Surface{ 4}={ 4, 1, 5, 8}; Recombine
    Surface{4};
Plane Surface( 5)={ 5}; Transfinite Surface{ 5}={ 5, 6,10, 9}; Recombine
    Surface{5};
Plane Surface( 6)={ 6}; Transfinite Surface{ 6}={ 6, 7,11,10}; Recombine
    Surface{6};
Plane Surface( 7)={ 7}; Transfinite Surface{ 7}={ 7, 8,12,11}; Recombine
    Surface{7};
Plane Surface( 8)={ 8}; Transfinite Surface{ 8}={ 8, 5, 9,12}; Recombine
    Surface{8};
Plane Surface( 9)={ 9}; Transfinite Surface{ 9}={ 9,10,14,13}; Recombine
    Surface{9};
```

```
Plane Surface(10)={10}; Transfinite Surface{10}={10,11,15,14}; Recombine
    Surface{10};
Plane Surface(11)={11}; Transfinite Surface{11}={11,12,16,15}; Recombine
    Surface{11};
Plane Surface(12)={12}; Transfinite Surface{12}={12, 9,13,16}; Recombine
    Surface{12};

Plane Surface(13)={13}; Transfinite Surface{13}={13,14,15,16}; Recombine
    Surface{13};


//Volume Construction
Extrude{0.0,0.0,z} {Surface{1,2,3,4,5,6,7,8,9,10,11,12,13}; Layers{NDlz};
    Recombine;}

//Physical Volumes Definition
Physical Volume("Bone")   = {13};
Physical Volume("Muscle") = { 9,10,11,12};
Physical Volume("Fat")    = { 5, 6, 7, 8};
Physical Volume("Skin")   = { 1, 2, 3, 4};
```

Appendix B

Program Development Modules

B.1 Developments modules

```
# -*- coding: utf-8 -*-

"""
Created on Mon Jan 14 15:03:12 2013

@author: jccg
"""


from fipy import *
from numpy import array

nx = 4 #number of cells of the mesh - correspond to the tissue layers
dx = array([2.68, 2.72, 0.31, 0.11]) # 2.68, 5.40, 5.71, 5.82 end of each
    layer in (cm)
dx = dx*10**-2

#Adaptation to neck cylindrical mesh# http://www.ctcms.nist.gov/fipy/fipy/
    generated/meshes.numMesh.html

fipy.meshes.numMesh.cylindricalGrid1D
mesh = CylindricalGrid1D(nx=nx, dx=dx)

print mesh.getCellCenters()

if __name__ == '__main__':
     raw_input("Cell Centers matrix. Press <return> to proceed...")

T_t = CellVariable(name="Neck Temperatures Distribution",
                   mesh=mesh,
                   value=(36.5,36.,34.,33.))

# Boundary Condition Values
```

```
valueRight = array([35., 36., 37., 37., 37., 37., 37., 37., 37., 37., 37., 37.,
                    37., 37., 37., 37., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 38., 37., 37., 37., 37., 37., 37., 37., 37.,
                    37., 37., 37., 37., 39., 39., 40., 40.])# this represents skin temperature changing over the time - 5s

#T_t.constrain(valueRight[step], (mesh.exteriorFaces & mesh.facesRight))
#T_t.constrain(valueLeft, mesh.facesLeft)

#Defining the coefficient of the transient term - specific weight and specific heat of the tissues
rho_t = array( [1357.,1086., 920., 1085.] ) # 1357 1085 920 1085 (Kg/m3)
c_t = array([1700., 3800., 2300., 3680.]) # 1700 3800 2300 3680 (J/(Kg.C))

transcoeff = CellVariable(name="Transient Term Coefficient",
                         mesh=mesh,
                         value=rho_t*c_t)
print transcoeff

#Definition of the diffusion term coefficient - thermal conductivity of the tissues
k_t = CellVariable(name="Difusion Term Coefficient",
                   mesh=mesh,
                   value=(0.75, 0.51, 0.21, 0.47)) # 0.75 0.51 0.21 0.47 (W/(m.C))

#Defining Source Term - Tissue Blood Flow, specific weight and specific heat of the blood
V_sv = array([0., 483., 398.7, 362.]) # 0. 483. 398.7 362. (cm3/(m3.s))
rho_bl = 1059. # 1059 de Ferreira e Yanihara, original from Werner and Buse(Kg/m3)
c_bl = 3850. # 3850 the same source of rho_bl variable (J/(Kg.C))
T_ar = 37. # Arterial blood temperature (C)
q_t = array([0., 501., 4., 368.]) # 0. 501. 4. 368 (W/m3) - Endogenous heat production of each tissue layer

sorcoeff=CellVariable(name="Source Term Coefficient",
                       mesh=mesh,
                       value=V_sv*rho_bl*c_bl*10**-6)
print sorcoeff

```

```python
indsourceterm=CellVariable(name="Internal Heat Production",
                           mesh=mesh,
                           value=V_sv*rho_bl*c_bl*T_ar*10**-6 + q_t)
print indsourceterm

eqX = TransientTerm(coeff=transcoeff) == DiffusionTerm(coeff=k_t)
      - ImplicitSourceTerm(coeff=sorcoeff) + indsourceterm

#timeStepDuration = 0.9 * 5.82**2 / (2 * (k_t)) -  5.82 is the ray of the
    cylinder
timeStepDuration=5.  #s
steps=180  #3600s/20s

T_tss = CellVariable(name="SS Neck Temperatures Distribution",
                     mesh=mesh,
                     value=(36.5,36.,34.,33.))

T_tss.constrain(valueRight[0], (mesh.exteriorFaces & mesh.facesRight))
#T_tss.constrain(valueLeft, mesh.facesLeft)


if __name__ == '__main__':
     viewer = Viewer(vars=(T_t,T_tss),
                     datamin=32., datamax=40.)
     viewer.axes.set_xlabel('r (m)')
     viewer.axes.set_ylabel('Temp. ($\circ$C)')

     viewer.plot()

if __name__ == '__main__':
     raw_input("Viewer created. Press <enter> to proceed...")

for step in range(steps):

     T_t.constrain(valueRight[step], (mesh.exteriorFaces & mesh.facesRight))

     eqX.solve(var=T_t,
               dt=timeStepDuration)

     if __name__ == '__main__':
         viewer.plot()
         print T_t

if __name__ == '__main__':
     raw_input("Transient Diffusion. Press <enter> to proceed...")

eqxss= 0 == DiffusionTerm(coeff=k_t) - ImplicitSourceTerm(coeff=sorcoeff) +
    indsourceterm
eqxss.solve(var=T_tss)

figname = raw_input("Insert Figure's name...")

viewer.plot(filename = figname)

print T_tss

```

```
if __name__ == '__main__':
     raw_input("Steady-state Diffusion. Press <enter> to proceed...")

TSVViewer(vars=(T_t,T_tss)).plot(filename=figname)

if __name__ == '__main__':

     raw_input("Press <return> to end. Consult present directory...")
```

Listing B.1: Sorce Code of the first development module, unidimensional heat diffusion for 4 layers.

```
# -*- coding: utf-8 -*-
"""
Created on Fri Feb 22 06:31:42 2013

@author: jc
"""

from fipy import *
from numpy import array
from viewers.update import update3D
from viewers.viewer3D import my_grid, view

# Import Mesh
mesh = Gmsh3D('geometries/neck.geo')

# Define Tissue variable
T_t = CellVariable(name="Neck Temperatures Distribution",
                    mesh=mesh,
                    value=0.)

# Set initial Condition for each layer
T_t.setValue(36.5, where=mesh.physicalCells['Bone'])
T_t.setValue(36., where=mesh.physicalCells['Muscle'])
T_t.setValue(34., where=mesh.physicalCells['Fat'])
T_t.setValue(33., where=mesh.physicalCells['Skin'])

# Set boundary conditions

sidecells = ~(mesh.facesBack | mesh.facesFront) & mesh.exteriorFaces

T_t.constrain(40., where=sidecells)

# Define terms for the equation and set the coefficients to each layer

#Set the values for specific heat and mass of each tissue layer to define
    transient term coefficient

rho_t=array([1357., 1085., 920., 1085.])# 1357 1085 920 1085 (Kg/m3)
c_t=array([1700., 3800., 2300., 3680.]) # 1700 3800 2300 3680 (J/(Kg.oC))

transcoeff=CellVariable(mesh=mesh, value=0.)

# Set values for thermal conductivity of tissues to define diffusion
    coefficient term
```

```

k_t =array([0.75, 0.51, 0.21, 0.47]) # 0.75 0.51 0.21 0.47 (W/(m.oC))

diffcoeff=CellVariable(mesh=mesh, value=0.)

#Set values of tissue blood flow, specific mass and heat of the blood
V_sv=array([0., 483., 398.7, 362.]) # 0. 483. 398.7 362. (cm3/(m3.s))

rho_bl = 1059. # 1059 (Kg/m3)
c_bl = 3850. # 3850 (J/(Kg.oC))

T_ar= 37. # Arterial blood temperature values (oC)

q_t = array([0., 501., 4., 368.]) # 0. 501. 4. 368 (W/m3)/ basal heat
    production for tissue layers

sourcecoeff=CellVariable(mesh=mesh, value=0.)

indsourceterm = CellVariable(mesh=mesh, value=0.)

for i in range(4):
    if i==0 : string='Bone'
    elif i==1 : string='Muscle'
    elif i==2 : string='Fat'
    elif i==3 : string='Skin'

    transcoeff.setValue(rho_t[i]*c_t[i], where=mesh.physicalCells[string])

    diffcoeff.setValue(k_t[i] , where=mesh.physicalCells[string])

    sourcecoeff.setValue(V_sv[i]*rho_bl*c_bl*10**-6, where=mesh.
    physicalCells[string])

    indsourceterm.setValue(V_sv[i]*rho_bl*c_bl*T_ar*10**-6 + q_t[i], where=
    mesh.physicalCells[string])

# Bioheat transfer equation
eqX = TransientTerm(coeff=transcoeff) == DiffusionTerm(coeff=diffcoeff) -
    ImplicitSourceTerm(coeff=sourcecoeff) + indsourceterm

#Solution obtained for step by step
timeStepDuration=20.
steps=180

dataset=my_grid(T_t)
viewer=view(dataset)

for step in range(steps):

     eqX.solve(var=T_t,
               dt=timeStepDuration)

     t = timeStepDuration*step + 10
     update3D(dataset, T_t)
     print T_t, 't=', t,'s'

```

```

# Save Results
#figname = raw_input("Insert Figure's name...")
#
#viewer.plot(filename = figname)
#
#TSVViewer(vars=(T_t,t).plot(filename=figname)
#
if __name__ == '__main__':

     raw_input("Press <any key> to end. Consult present directory...")
```

Listing B.2: Sorce Code of second development module, three-dimensional heat diffusion.