Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXVII, págs. 1705-1718 (artículo completo)
Alberto Cardona, Mario Storti, Carlos Zuppa. (Eds.)
San Luis, Argentina, 10-13 Noviembre 2008

# NUMERICAL SIMULATION OF THE HEAT TRANSFER IN THREE DIMENSIONAL GEOMETRIES

**María V. Santos[a,b], Noemí Zaritzky[a,c], Alicia Califano[a] and Victoria Vampa[b]**

[a]*Centro de Investigación y Desarrollo en Criotecnología de Alimentos (CIDCA), CCT-La Plata, CONICET; Facultad de Ciencias Exactas, UNLP –. 47 y 116, La Plata (1900). Argentina*
[b] *Área Dep. Ciencias Básicas, Facultad de Ingeniería, UNLP*
[c] *Área Dep. Ingeniería Química, Facultad de Ingeniería, UNLP*

**Keywords**: three dimensional geometries, Heat Transfer, Finite Element Method, Mesh Processing.

**Abstract**. A finite element procedure for transient three dimensional (3D) heat transfer problems was developed and implemented. The domain was divided into linear tetrahedral elements using a three dimensional mesh generator software. A pre-processing program was developed in order to the mesh information to be compatible with finite element program. All the numerical algorithms have been implemented using Matlab 6.5. Results were validated by comparing with analytical solutions of heat transfer in a finite cylinder and a sphere, and with the numerical solution generated by commercial software for heating an irregular piece of meat. A post-processing code was implemented in order to obtain further information from the results, such as the temperature prediction at an arbitrary point, and the average temperature. The code can also be used to determine concentration profiles in mass transfer problems (3D domains) and to simulate heat transfer problems in food processing with convective boundary conditions. The open source program can be easily applied with the important advantage that it can be coupled with macroscopic balances, microbial inactivation rates, or with different objective functions that optimize the process (e.g. quality attributes).

## INTRODUCTION

Many engineers today are required to routinely solve complex problems in heat and mass transfer, structural mechanics, fluid dynamics, vibrations, and acoustics, using computational tools such as solid modelers, computer-aided design and finite element simulation software packages. The proficiency in using such systems enables engineers to model complex engineering design and to analyze problems efficiently. Commercial software based on finite element analysis is often used as a "black box" program, where the user is not allowed to see its inner code. Even though many physical problems can be simulated with a software package, simulations involving safety and quality of foods in biological systems have not yet been incorporated (Martins, 2004). Some of the advantages of working with an open computer program are, for example, the ability to combine the microbial inactivation kinetics, food quality, and manufacturing cost equations, which are then used for optimization techniques in the food industry (Erdogdu et al., 2005; Martins, 2006, Santos et al., 2008). In commercial software packages the ability to couple macroscopic heat balances is generally not possible, even though it is useful for the evaluation and prediction of the actual industrial conditions. Besides commercial softwares are very expensive for small scale industry.

For three dimensional (3D) problems the finite element method is often more difficult to implement, in contrast with one or two dimensional problems. An important issue when trying to generate an open source program in three dimensions is that the mesh data produced by external mesh generators are hard to integrate with other codes, especially because there is a lack of information about the assignment of the node points and elements numbering. Therefore, the ability to understand and to use the mesh information is valuable to create a three dimensional finite element program. Preprocessing of the mesh data must be implemented in order to be compatible with the finite element code, as well as the postprocessing of the results.

Many food engineering processes involve heat transfer with convective boundary conditions. For regular shapes a finite difference method gives accurate predictions, however the finite element method is more suited when dealing with non-conventional shapes (Arce et al., 1983, Ngadi et al., 1996), especially in three dimensions.

The goals of this work are:

- to develop a three dimensional finite element program to solve heat or mass transfer equations in transient state with convective boundary conditions.
- to develop a preprocessing program that combines the mesh information from 3D geometries obtained from an external mesh generator in order to be compatible with the main program.
- to generate a postprocessing program that calculates the dependent variable (temperature or concentration) in any given point of the domain and also integrates these variables on the surface or volume of the irregular object.
- to validate the model comparing the output with analytical solutions and commercial software simulations of the three dimensional problems.

## 1 MATHEMATICAL MODEL

The governing differential equations for transient state heat conduction in a region $\Omega$ with convective boundary conditions are the following (Carslaw and Jaeger, 1959):

$$\rho C_p \frac{\partial T}{\partial \tau} = \nabla^t \cdot (-k \nabla T) \qquad \text{in } \Omega \qquad (1)$$

$$-\mathrm{k} \nabla T \cdot \boldsymbol{n} = h(T - T_\infty) \qquad \text{in } \delta\Omega \qquad (2)$$

The variable T is the temperature scalar function $T(x, y, z, t)$, $T_\infty$ is the external fluid temperature, $k$ is the thermal conductivity (isotropic), $Cp$ the specific heat, $\rho$ the density, **n** the outward normal unit vector to the boundary surface, h the surface heat transfer coefficient, and $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})^t$ where $x$, $y$, and $z$ are the cartesian coordinates. The initial condition is $T=T_0$ at $t= 0$ in $\Omega$.

Representing the temperature using a finite dimensional space $V_h$ with shape functions $\boldsymbol{H}$, $\tilde{T} = \boldsymbol{H}(x,y,z) \cdot \hat{\boldsymbol{T}}$ (Galerkin Method) (Zienkiewicz and Taylor, 1994a; Zienkiewicz and Taylor, 1994b; Bathe, 1996) and applying the divergence theorem the following equation is obtained:

$$\underbrace{\int_\Omega (\boldsymbol{H}^t \rho C_p \boldsymbol{H} d(\Omega))}_{\boldsymbol{CG}} \overset{*}{\boldsymbol{T}} - [\underbrace{\int_{\delta\Omega} (\boldsymbol{H}^t h \boldsymbol{H} d(\delta\Omega)) + \int_\Omega (\nabla \boldsymbol{H}^t \cdot \nabla \boldsymbol{H}) \cdot k d(\Omega)}_{\boldsymbol{KG}}] \hat{\boldsymbol{T}} + \underbrace{\int_{\delta\Omega} \boldsymbol{H}^t h T_\infty d(\delta\Omega)}_{\boldsymbol{FG}} = 0 \qquad (3)$$

Where ***CG*** is the global capacitance matrix, ***KG*** is the global conductance matrix, and ***FG*** the global force vector. $\hat{\boldsymbol{T}}$ is the vector that represents the temperature values at the node points, and $\overset{*}{\boldsymbol{T}}$ represents the $\frac{\partial \hat{\boldsymbol{T}}}{\partial t}$. This semi discrete problem (equation (3)) is a system of stiff ordinary differential equations. For the time discretization we considered the classical backward Euler method (Johnson, 1986).

### 1.1 Mesh generation and preprocessing

The spatial discretization of the domain was done by means of a mesh generator using linear tetrahedral elements. The mesh information given by the program is generally transferred using three important matrices; the "***p***", point matrix, "***tm***" tetrahedral matrix, and "***e***" boundary matrix. The point matrix represents the $x$, $y$, and $z$ coordinates of the node points given in three columns, where each row represents the node number. The "***tm***" matrix gives the element connectivity with the nodes; it is a 4 x N matrix, being N the number of elements; in each row the node numbers are given in a specific order, according to the numbering of the reference tetrahedral described in Figure 1.

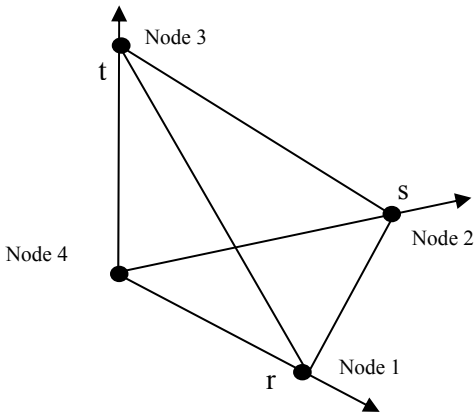



Figure1: Reference tetrahedral

The linear shape functions are the following:

*h1= r*
*h2= s*
*h3= t*
*h4= 1-r-s-t*

The boundary matrix "***e***" contains information of the node points (three) that are the vertices of the surface triangle in contact with the interface $\delta\Omega$. However there is no information of the element number to which this surface triangle belongs. This is an important issue when computing the element matrices, since the finite element code computes a "for" loop by elements.

The triangle surface in contact with the interface contains three of the four vertices. These three node points are together in a particular order in the element matrix "***tm***". As four is the number of node points of the tetrahedral, and three is the number of node points in the triangle surface the program must include as many "if" loops as the total combinations that the three node points can be found in the "***tm***" matrix which amounts to 24 (six for each of the four surface triangles).

The pre-processing program helps to integrate the surface information required by the main program. As an example one of the 24 "if" loops in Matlab language, to determine the element number to which the surface triangle belongs, considering that *length (**e**)* is the number of boundary elements and *length (**tm**)* equals the total number of elements;

```
for k=1:length(e)
for i=1:length(tm)

  %%%First main "if" loop of one of the 24 if loops%%%%%%%%%
 if tm(1,i)==e(1,k)
     if tm(2,i)==e(2,k)
        if tm(3,i)==e(3,k)
           element(1,k)=1;
           element(2,k)=1;
           element(3,k)=1;
```

```
            element(4,k)=0;
            element(5,k)=i;
          end
        end
      end
%%%%%end of the first main "if" loop
```

The element number that contains the surface triangle is accumulated in the matrix "***element***" in the fifth row. The value zero is assigned to the node that does not belong to the surface triangle; in this example the shape function h4 is zero, thus row 4 of the element matrix is zero. The case *h4 = 0* is named as surface triangle $S_4$, which involves local node points 1, 2, and 3. The surface triangle $S_1$ involves the local node points 2, 3 and 4 (*h1 = 0*), $S_2$ involves the local node points 1, 4, and 3 (*h2 = 0*), and $S_3$ the node points 1, 2, 4 (*h3 = 0*). Figure 2 illustrates all possible surface triangle cases and the node points that correspond to each surface type.

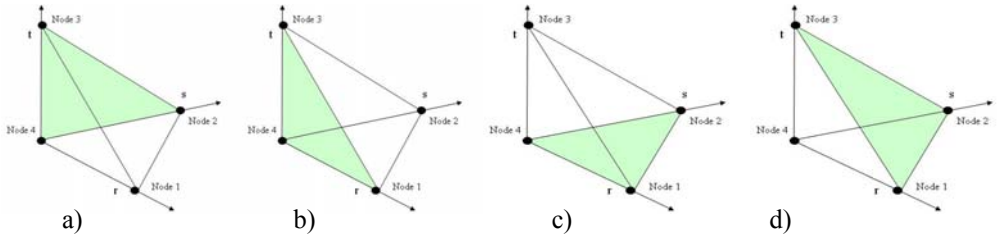



Figure 2: Surface triangles and local node points that constitute each surface type. a) $S_1$, b) $S_2$, c) $S_3$, d) $S_4$

The Figure 2 d) which is the surface type $S_4$ corresponds to the case described in the example.

## 1.2 Finite element program

The preprocessed mesh information is then used in the main program, which is written following the construction scheme presented in Becker et al., (1983). Figure 3 shows a flow chart explaining how the program works, where *det(**J**)* is the determinant of the Jacobian, aux is the module of the normal outward vector, and ***w*** is the vector that contains the weights of the quadrature integration rules.

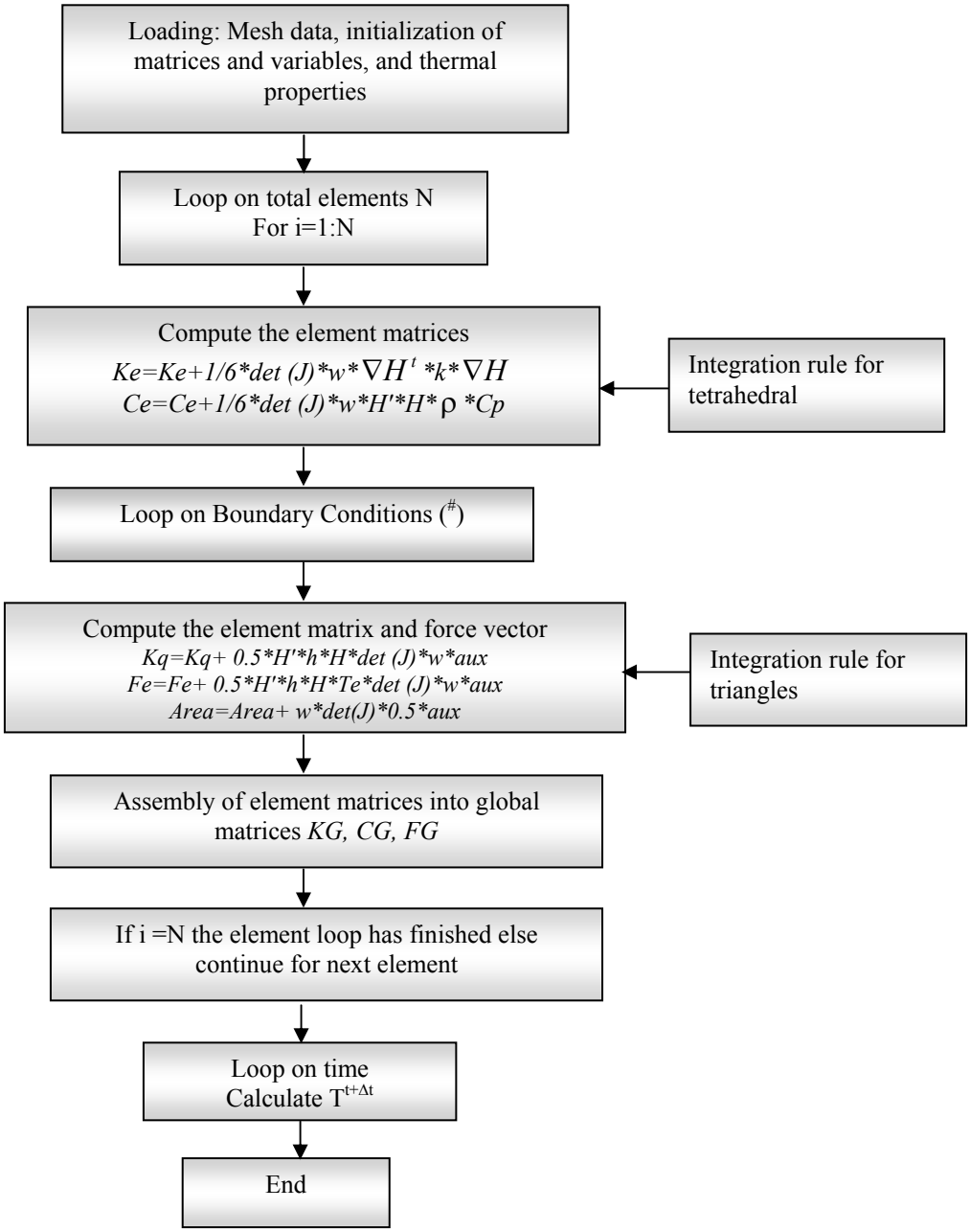



Figure 3: Flow chart of the finite element program.

(#) The calculation concerning the boundary integration is explained in detail in the following paragraphs:

For each element when the computation of the ***Ke*** and ***Ce*** element matrices is completed, the boundary integration is performed if the tetrahedral contains a triangle that is exposed to the surface. The "***element***" matrix contains information of the surface type involved: $S_1$, $S_2$, $S_3$ or $S_4$. In a three dimensional space the integration of a function over a surface is represented by the following equation, (Leithold, 1998)

$$\iint_S f\, dS = \iint_A f \left| \mathbf{n} \right| dA \tag{4}$$

As an example, if it is assumed that $S_4$ has a boundary condition, the three involved node points that represents the surface is:

$$c1(x-x1) + c2\,(y-y1) + c3\,(z-z1) = 0 \tag{5}$$

where *c1*, *c2*, and *c3* are:

$$c1 = (y3-y4)*(z2-z4)-(z3-z4)*(y2-y4) \tag{6}$$

$$c2 = (z3-z4)*(x2-x4)-(x3-x4)*(z2-z4) \tag{7}$$

$$c3 = (x3-x4)*(y2-y4)-(y3-y4)*(x2-x4) \tag{8}$$

When calculating the normal vector to the surface triangle special attention must be focused on the detection of a surface that is parallel to a coordinate plane, therefore the maximum value of *cI* with *I= 1, 2,* or *3* is identified by using the simple Matlab sentence code,

*[Value, Imax] = max (abs(c)) where* $\boldsymbol{c} = [c1\ c2\ c3]$

The next step calculates the Jacobian of the transformation *y=y(r,s), z=z(r,s),* since the surface is projected over the plane *"yz"* when *c1* is the maximum value of the vector $\boldsymbol{c}$. The Jacobian is defined as follows:

$$\boldsymbol{J} = \begin{pmatrix} \dfrac{dy}{ds} & \dfrac{dy}{dr} \\ \dfrac{dz}{ds} & \dfrac{dz}{dr} \end{pmatrix} \tag{9}$$

Finally, the evaluation of the function is done by numerical integration using the quadrature rule points for the triangle, obtaining the following expression (Bathe, 1996):

$$\iint f\,dS \cong \sum_i f(r(i),s(i))\,w(i)\,det(J)*0.5*aux \tag{10}$$

Where i denotes the corresponding quadrature integration point for the triangle, aux = $\left|\boldsymbol{n}\right|$, and w(i) are the weights of the quadrature rules (Hughes, 1987). The final computation for each element of the matrix $\boldsymbol{Kq}$, vector $\boldsymbol{Fe}$, and the area of the element triangle can be calculated, where the *f(r(i), s(i))* is a polynomial function formed by multiplications of the interpolating functions (see Figure 3).

The time discretization with the $\alpha$-Method (Segerlind, 1984), using $\alpha=1$ was used, therefore an implicit scheme was implemented (see equation 14). It was used a time step of 1 s ($\Delta t = 1$ s).

$$(\Delta t^{-1}\boldsymbol{CG} + \alpha\boldsymbol{KG})\tilde{\boldsymbol{T}}^{t+\Delta t} = \boldsymbol{FG} + (\Delta t^{-1}\boldsymbol{CG} - \alpha\boldsymbol{KG})\tilde{\boldsymbol{T}}^{t} \tag{11}$$

### 1.3 Postprocessing

The results can be obtained using a postprocessing program that enables the user to calculate, for example, the temperature at any given point in the domain. A subroutine or function such as the next sentence (in Matlab language) can be implemented where the input

information are (for the point of interest ***M***) the *x, y, z* coordinates of the point, and the "***p***" and "***tm***" matrices. The output information involves the node points and the interpolation functions evaluated at ***M***.

```
function [h1,h2,h3,h4,node1,node2,node3,node4]=pospro(x, y, z, p, tm)
```

The function "pospro" calculates for each element the distance from ***M*** to the geometric center of the tetrahedral, and finds the least square distance storing the element number. A few code lines are shown below as an example.

```
for i=1:length(tm)
      %% square distance
      xnode1=p(tm(1,i),1)
      ynode1=p(tm(1,i),2)
      znode1=p(tm(1,i),3)
xm=(xnode1+xnode2+xnode3+xnode4)/4
……….
dc(i)=(xm-x)^2+(ym-y)^2+(zm-z)^2
end

%%%%find minimum distance%%%%%%%%%
[dcmin , element] = min(dc)
```

Since the nodes that constitute the element are known, as well as their coordinates, the x coordinate of the point ***M*** is written as:

$$x = xnode1*h1+xnode2*h2+xnode3*h3+xnode4*h4 \tag{12}$$

Combining the interpolation functions, h4=1-h1-h2-h3, a simple system of three equations with three unknown variables h1, h2 and h3 is defined. With these data, the temperature (or concentration) at any point ***M*** is:

$$T=T(node1)*h1+T(node2)*h2+T(node3)*h3+T(node4)*h4 \tag{13}$$

When the final concentration of a substance is the object value to be found, the concentration values of the nodes are stored and the computation of the volume is required.

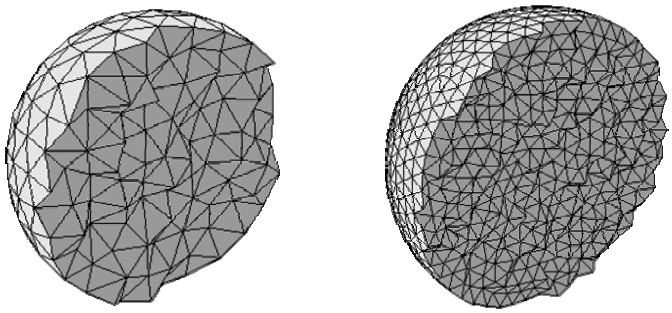
```
for n=1:length(tm)
C1=c(tm(1,n))
C2=c(tm(2,n))
C3=c(tm(3,n))
C4=c(tm(4,n))

for i=1:pp  %%%%%integration points in tetrahedral
    Cp=C1*h1+C2*h2+C3*h3+C4*h4
     Volume=Volume+ det(J)*w/6
     Cm=Cm+ Cp*det(J)*w/6
End
End
Cfinal = Cm/Volume
```

## 2 RESULTS AND DISCUSSION

### 2.1 Code validation and testing of the numerical program with analytical solutions

The analytical solution of the heat transfer with convective boundary conditions in a sphere is well known (Welty, 1974, Carslaw and Jaeger, 1959), therefore it was used to corroborate the accuracy and convergence of the numerical temperature predictions using different mesh sizes. The thermal properties of an acrylic material were used (k=0.2075 W/m ºC, Cp=1464 J/ kg ºC, $\rho$=1180 kg/m$^3$). The initial temperature of the solid was 20º C and the fluid temperature was 60.3º C, and the heat transfer coefficient was 55 W/m$^2$ ºC. The radius of the sphere (r) was 0.0152 m.

The mesh information was obtained using a mesh generator called "DistMesh" (Perssons and Strang, 2004). This program is a simple MATLAB code that generates unstructured triangular and tetrahedral mesh. Figure 4 shows an example of two meshes used for the calculations.



a) b)

Figure 4: Different meshes used in the program, a) Mesh 2 b) Mesh 4.

The infinite norm, $\| \ \|_\infty$ of the percentage error ( $e = \frac{T_a - T_n}{T_a} \cdot 100$ ) was calculated for each time step in three points in the domain: center (r=0), middle point (r= 7.6*10$^{-3}$m) and a boundary point (r=0.0152m). The time required by the CPU to solve the numerical problem was also computed as time$_{cpu}$ given in minutes (see Table 1). The PC used for the simulations was an Intel(R) Core(TM) 2 6300 with a processor speed of 1.86 GHz and has a RAM memory of 2GB.

| | $\|e\|_{\infty,c}$ | $\|e\|_{\infty,m}$ | $\|e\|_{\infty,b}$ | $\text{time}_{\text{cpu}}$ |
|---|---|---|---|---|
| Mesh 1<br>142 node points<br>515 elements | 10.92 | 8.75 | 6.89 | 0.23 |
| Mesh 2<br>592 node points<br>2726 elements | 3.75 | 1.70 | 3.86 | 1.02 |
| Mesh 3<br>1908 node points<br>9573 elements | 0.95 | 0.73 | 0.84 | 11.92 |
| Mesh 4<br>4500 node points<br>23695 elements | 0.37 | 0.32 | 0.23 | 66.66 |

Table 1: Maximum percentage error for different meshes and computational time required.

It can be seen that there was an improvement of the solution as the number of node points increases, however the computational cost becomes higher. The optimal mesh that balanced the numerical effort in achieving an accurate solution and the execution speed of the code was Mesh 3, since the numerical error was low (less than 1%) and CPU time, acceptable.

Secondly, the analytical solution of a finite acrylic cylinder was also compared with the numerical predictions using three different meshes. The height of the cylinder used was 0.0304 m and the radius was 0.0152 m. The same thermal properties and initial conditions were used. Figure 5 shows as an example two of the meshes used. The $\|e\|_{\infty}$ was calculated in three points of the domain; center, middle point and a border point as well as the computational time required for the program to run the simulations (see Table 2).

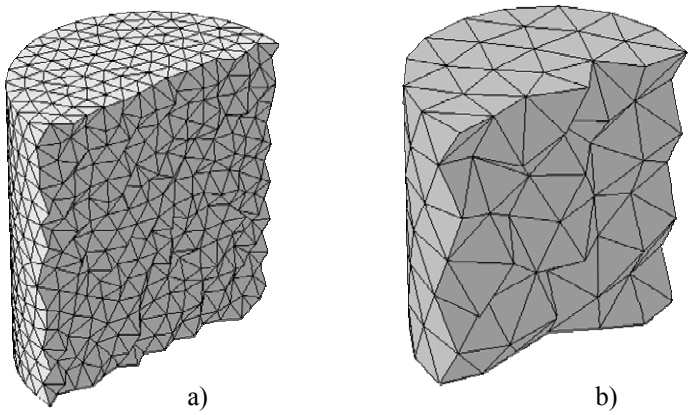



Figure 5: Different meshes for the finite cylinder used in the program, a) Mesh 3 b) Mesh 1.

| | $\|e\|_{\infty,c}$ | $\|e\|_{\infty,m}$ | $\|e\|_{\infty,b}$ | $\text{time}_{\text{cpu}}$ |
|---|---|---|---|---|
| Mesh 1<br>392 node points<br>1609 elements | 4.15 | 1.99 | 8.23 | 0.71 |
| Mesh 2<br>957 node points<br>4373 elements | 2.73 | 0.88 | 3.77 | 2.98 |
| Mesh 3<br>2960 node points<br>14795 elements | 0.73 | 0.58 | 1.86 | 33.5 |

Table 2: Maximum percentage error for different mesh and computational time required.

It can be observed that the numerical results for the finite cylinder satisfactorily agreed with the analytical solutions. The $\|e\|_{\infty}$ decreased as the number of elements increases, but the computational cost became also higher. The choice of the mesh used for the numerical program must be made balancing the loss in accuracy against savings in computational cost. In this case an optimum mesh selection could be an intermediate between Mesh 2 and Mesh 3.

### 2.2 Code testing in a complex 3D geometry with finite element software

Finally, the program was used to simulate cooking of meat piece (semi-tendinousus muscle), where the cross-section was scanned and digitalized in order to create the irregular domain (Califano and Zaritzky (1993)). The mesh generated is shown in Figure 6 a) and it was obtained form the software package COMSOL, where the mesh information was exported from the program as a structure named “fem”. The mesh consisted of 3112 node points and 13735 elements. The irregular shaped meat cut had two domains which corresponded with two set of different thermal properties as shown in Figure 6 b). The thermal properties for the meat and fat were obtained by Califano and Zaritzky, (1993) and the thermal processing conditions are given in Table 3. The numerical prediction in a center point (7.65, 5.78, 3.5) and border point (3.84, 7.56, 3.5) given in centimeters were compared with the output temperatures of the software (see Figure 7).

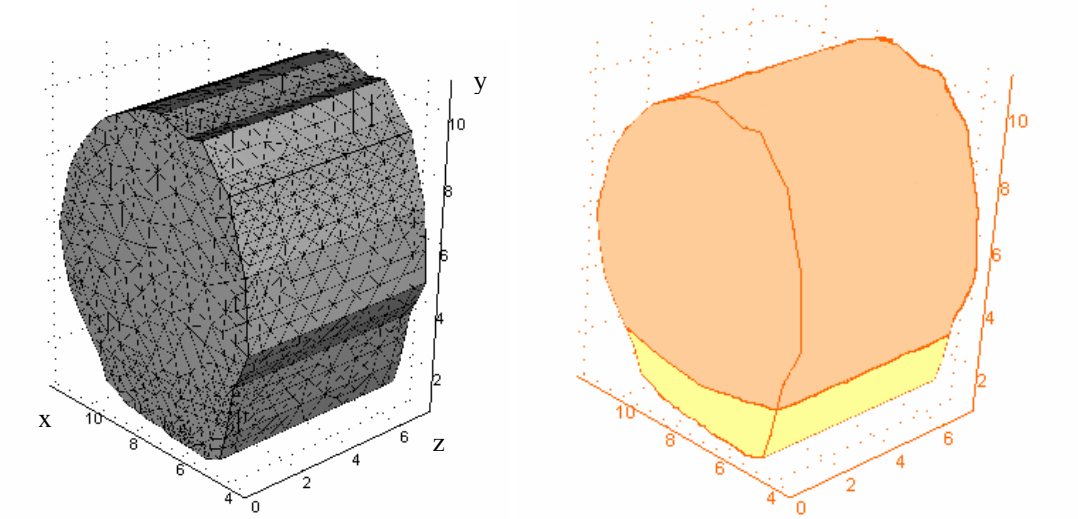



Figure 6: a) Mesh used for simulation b) Food composition: Meat (orange) and fat (yellow).

| Thermal Properties and Process Conditions | |
|---|---|
| Initial Temperature (ºC) | 14.6 |
| Fluid Temperature (ºC) | 70 |
| Surface Heat Transfer Coefficient h (W/m$^2$ ºC) | 300 |
| Meat | |
| Thermal Conductivity (W/m ºC) | 0.454 |
| Specific Heat (J/kg ºC) | 3477.8 |
| Density (kg/m$^3$) | 969.2 |
| Fat | |
| Thermal Conductivity (W/m ºC) | 0.175 |
| Specific Heat (J/kg ºC) | 4111.95 |
| Density (kg/m$^3$) | 930 |

Table 3: Thermal Properties of the meat piece and process conditions

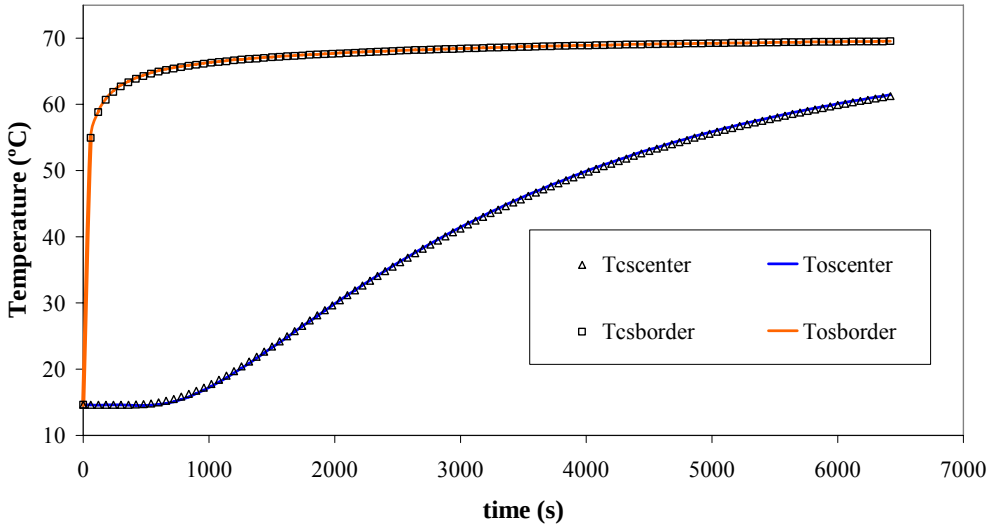



Figure 7: Numerical Predictions in center and border point of the domain using the software (Tcs) and open source code (Tos).

As can be seen numerical predictions from the commercial software agreed satisfactorily with the numerical results from the open source code.

## 3 CONCLUSIONS

A finite element code for three dimensions has been developed to solve the linear unsteady heat transfer problem with convective boundary conditions. A pre- and post- processing code was successfully implemented in order to integrate the mesh information with the main program and to calculate the temperature distribution at any given point inside the domain. The numerical code was validated by comparing the temperature predictions with the analytical solutions of a sphere and finite cylinder. The program was then used to simulate heat transfer in a meat product with irregular domain. The numerical results obtained by the open source code were also compared with the commercial software predictions resulting in high agreement. The program code can be applied to three dimensional domains using an external mesh generator with the advantage that in food processing the microbial inactivation or quality kinetics can be easily coupled to the heat transfer process.

## REFERENCES

Arce, J.A., Potluri, P.L., Schneider, K.C., Sweat, and V.E., Dutson, T.R., *Modeling Beef Carcass Cooling Using a Finite Element Technique*. Transactions of the ASAE. Paper nº 816030, 1983.

Bathe, K.J., *Finite element procedures*. Prentice Hall, New Jersey, 1996.

Becker, E., Carey, G.F. and Oden, J.T., "*Finite Elements: An Introduction*" (Vol. 1). Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

Califano, A. N., Zaritzky, N., A Numerical Method for Simulating Heat Transfer in Heterogeneous and Irregularly Shaped Foodstuffs. *Journal of Food Process Engineering*, 16: 159-171, 1993.

Carslaw, H. S., and Jaeger, J.C., *Conduction of heat in solids*. University Press, Oxford. 1959.

Erdogdu, F., Zorrilla, S. E., Singh, R. P., Effect of different objective functions on optimal decision variables: a study using modified complex method to optimize hamburger cooking. *Lebensmitel-Wissenschaft und –Technologie,* 38: 111-118, 2005.

Hughes, T. J. R., The *Finite Element Method- Linear Static and Dynamic Finite Element Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

Johnson, Claes, *Numerical Solution of Partial Differential Equations by FEM*, Cambridge University Press, 1987.

Leithold, L., *El Cálculo*, 7th ed. Oxford University Press-Harla México, México, 1998.

Martins, R. C., Simple finite volumes and finite elements procedures for food quality and safety simulations. *Journal of Food Engineering,* 73: 327-338, 2006.

Martins, R.C., Modelling temperatures abuses to frozen food and effects on quality. PhD thesis. Escola Superior de Biotecnologia, Universidade Católica Portuguesa, Porto, Portugal, 2004.

Ngadi, M.O., Watts, K.C., and Correira, L.R., Finite Element Method Modelling of Moisture Transfer in Chicken Drum During Deep-fat Frying. *Journal of Food Engineering,* 32:11-20, 1997.

Persson, P.O., Strang G., A Simple Mesh Generator in MATLAB.*SIAM Review*, 46 (2): 329-345, 2004.

Santos M. V., Zaritzky N., Califano A. N., Modeling heat transfer and inactivation of Escherichia coli O157:H7 in precooked meat products in Argentina using the finite element method. *Meat Science* 79 (3): 595-602, 2008.

Segerlind, L. J., *Applied Finite Element Analysis,* 2nd ed. John Wiley and Sons, New York, 1984.

Welty, J. R., *Engineering Heat Transfer*. John Wiley and Sons, New York, 1974.

Zienkiewicz, O.C., y Taylor, R.L., *El método de los elementos finitos*, volumen I. McGraw-Hill, Barcelona, 1994 a).

Zienkiewicz, O.C., y Taylor, R.L., *El método de los elementos finitos*, volumen II. McGraw-Hill, Barcelona, 1994 b).