Computación y Sistemas
ISSN: 1405-5546
computacion-y-sistemas@cic.ipn.mx
Instituto Politécnico Nacional
México

Cardoso-Nungaray, Victor E.; Vargas-Felix, Miguel; Botello-Rionda, Salvador
Parallel Processing Strategy for Solving the Thermal-Mechanical Coupled Problem Applied to a 4D System using the Finite Element Method
Computación y Sistemas, vol. 17, núm. 3, julio-septiembre, 2013, pp. 289-298
Instituto Politécnico Nacional
Distrito Federal, México

Parallel Processing Strategy for Solving the Thermal-Mechanical Coupled Problem Applied to a 4D System using the Finite Element Method

Victor E. Cardoso-Nungaray, Miguel Vargas-Félix, and Salvador Botello-Rionda

Computational Sciences Department,
Centro de Investigación en Matemáticas A.C.,
Jalisco S/N, Col. Valenciana, 36240, Guanajuato, Gto.,
Mexico

{victorc, miguelvargas, botello}@cimat.mx

Abstract. We propose a high performance computing strategy (HPC) to simulate the deformation of a solid body through time as a consequence of the internal forces provoked by its temperature change, using the Finite Element Method (FEM). The program finds a solution of a multi-physics problem, solving the heat diffusion problem and the linear strain problem for homogeneous solids at each time step, exchanging information between both solutions to simulate the material distortion. The HPC strategy approach parallelizes vector and matrix operations as well as system equation solvers. The tests were realized over a model simulating a car braking system (a rotating disk velocity decreased by friction). Then we performed a quantitative analysis of stress, strain and temperature in some points of the geometry, and a qualitative analysis to show some visualizations of the simulation.

Keywords. Parallel computing, HPC, simulation, FEM, finite element, thermal-mechanical coupled problem, dynamic analysis, heat distortion.

Estrategia de procesamiento paralelo para la solución del problema térmico-mecánico acoplado aplicado a un sistema 4D utilizando el método de elemento finito

Resumen. Utilizando el Método de Elemento Finito (FEM), se propone una estrategia de cómputo de alto rendimiento (HPC) para simular dinámicamente la deformación causada por las fuerzas internas de un cuerpo sólido, como concecuencia del cambio de su temperatura. El programa resuelve un problema de multifísica, ya que da solución al problema de difusión de calor y al problema de deformación lineal de sólidos homogéneos para cada instánte de tiempo, intercambiando información entre ambas soluciones para simular la distorción del material. La estrategia de HPC consiste en paralelizar las operaciones matriciales y los algoritmos de solución de sistemas de ecuaciones. Las pruebas se realizaron en un modelo computarizado del sistema de frenado de un vehículo moderno (disminuir la velocidad de rotación de un disco a través de la fricción de un dispositivo de frenado). Después se realizó un análisis cuantitativo del estrés, de la deformación y de la temperatura en algunos puntos de la geometría, y un análisis cualitativo para mostrar las visualizaciones más ilustrativas del fenómeno.

Palabras clave. Cómputo paralelo, simulación, MEF, elemento finito, problema térmico/mecánico acoplado, analísis dinámico, calor.

1 Introduction

The analysis of physical phenomena caused by heat-induced stress is of special interest for the metallurgic, building, automotive and electronic industries. In this work, using numerical methods and computing power, we successfully simulate this process solving two differential equations, the first to model the heat diffusion over the medium and the second to model the deformation produced by the internal forces. This paper proposes a parallel strategy to solve the heat diffusion problem as shown in [3] and the problem of linear deformation of homogeneous solids as in [2] using FEM. On the other hand, we use the Finite Differences Method (FDM) to implement the dynamic analysis with a scheme explained in [7].

The authors of [1] describe a non-transitive solution scheme based on FEM to approach the thermal distortion problem of strictly cylindrical-shaped geometries. The parallel strategy proposed here works for two dimensions (2D) and three dimensions (3D), plus time (4D) over any coherent geometry.

The implementation design includes cache-oblivious algorithms parallelized with OpenMP in order to exploit the X86 computer architecture, increasing the execution speed almost to its double using four processors.

2 Solution of the Heat Diffusion Problem

To determine the temperature of any point in the solid body, we want to integrate the following differential equation over the domain (the body):

$$\frac{\partial}{\partial x}\left(k_x\frac{\partial\phi}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y\frac{\partial\phi}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_z\frac{\partial\phi}{\partial z}\right)+Q=\rho c_p\frac{\partial\phi}{\partial t}, \quad (1)$$

where $\phi=\phi(x,y,z)$ is the temperature, k_x, k_y and k_z are the material thermal conductivity for each component, c_p is the specific heat, Q is the source term, and ρ is the density of the body.

The flows over each component are denoted as

$$\begin{aligned} q_x(x,y,z) &= -k_x\frac{\partial\phi}{\partial x}, \\ q_y(x,y,z) &= -k_y\frac{\partial\phi}{\partial y}, \\ q_z(x,y,z) &= -k_z\frac{\partial\phi}{\partial z}. \end{aligned} \quad (2)$$

To solve the equation we must define at least one boundary condition, this can be

— Dirichlet: Set the temperature to a subdomain ($\phi(\hat{x},\hat{y},\hat{z})=\hat{\phi}$);

— Neumann: Set the flow to subdomain ($\Delta\phi(\hat{x},\hat{y},\hat{z})=\hat{q}$).

where $\hat{x}$ denotes a specific value for the variable x, and in the same way we use the variables $\hat{y}$, $\hat{z}$ and $\hat{q}$.

Then we must discretize the domain (this process is called meshing), and the goal is to divide a given geometry into polygons (triangles for 2D) or polyhedrons (tetrahedrons for 3D). Each of these divisions is known as an element, each element is connected with the other elements through its vertices or nodes as they are called in FEM. In [10] it is explained how to use the Delauney triangulation to generate a mesh.

There exist several types of elements according to their geometry and the integration approximation degree over the domain (using Gaussian quadrature integration), Botello [3] presents a detailed explanation of almost all of them. The implementation developed in this work uses tetrahedral elements with linear approximation.

For each element type, depending on its geometry, a shape function $N_i(\xi,\eta,\zeta)$ is defined for each node i in the normalized space and its respective partial derivatives $\frac{\partial N_i}{\partial\xi}$, $\frac{\partial N_i}{\partial\eta}$ and $\frac{\partial N_i}{\partial\zeta}$. Also, depending on the approximation degree used, each type of elements has its own integration points known as Gauss points. The authors of [3] present a table of values for each element configuration of Gauss points.

In the case of the tetrahedral element with linear approximation, there is only one integration point ($n_p=1$), in which $w_p=1$, and its normalized coordinates are

$$\xi=\frac{1}{4}, \qquad \eta=\frac{1}{4} \qquad \text{and} \qquad \zeta=\frac{1}{4}, \quad (3)$$

then, the shape functions are

$$\begin{aligned} N_1(\xi,\eta,\zeta)&=1-\xi-\eta-\zeta, & N_3(\xi,\eta,\zeta)&=\eta, \\ N_2(\xi,\eta,\zeta)&=\xi, & N_4(\xi,\eta,\zeta)&=\zeta. \end{aligned} \quad (4)$$

The following step is to create the stiffness matrix $\mathbf{K}^{(e)}\in\mathbb{R}^{n_e\times n_e}$, the mass matrix $\mathbf{M}^{(e)}\in\mathbb{R}^{n_e\times n_e}$, and the flow vector $f^{(e)}\in\mathbb{R}^{n_e}$ for each element:

$$\mathbf{K}^{(e)}=\sum_{p=1}^{n_p}\mathbf{B}^T\mathbf{D}\mathbf{B}|\mathbf{J}^{(e)}|w_p, \quad (5)$$

$$\mathbf{M}^{(e)}=\sum_{p=1}^{n_p}\rho\mathbf{N}^T\mathbf{N}|\mathbf{J}^{(e)}|w_p, \quad (6)$$

$$f^{(e)}=\sum_{p=1}^{n_p}\mathbf{N}^TQ|\mathbf{J}^{(e)}|w_p, \quad (7)$$

where $\mathbf{B}$ is the strain matrix of the element, $\mathbf{D}$ is the constitutive matrix, $\mathbf{N}$ is the shape function matrix of the element evaluated at the point p, $\mathbf{J}^{(e)}$ is the Jacobian matrix of the element evaluated at the point p, and w_p is the multiplication of Gaussian

integration weights. These matrices are given by

$$\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_{\mathbf{n_e}}], \tag{8}$$

$$\mathbf{D} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix}, \tag{9}$$

$$\mathbf{N} = [N_1, N_2, ..., N_{n_e}], \tag{10}$$

$$\mathbf{J}^{(e)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix}, \tag{11}$$

$$= \begin{bmatrix} \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \xi} x_i & \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \xi} y_i & \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \xi} z_i \\ \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \eta} x_i & \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \eta} y_i & \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \eta} z_i \\ \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \zeta} x_i & \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \zeta} y_i & \sum_{i=1}^{n_e} \frac{\partial Ni}{\partial \zeta} z_i \end{bmatrix},$$

where n_e is the elemental number of nodes, x_i, y_i and z_i are the i-nodal coordinates, and $\mathbf{B_i}$ is the strain matrix which is defined by

$$\mathbf{B_i} = \begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial z} \end{bmatrix} = (\mathbf{J}^{(e)})^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \\ \frac{\partial N_i}{\partial \zeta} \end{bmatrix}. \tag{12}$$

When $\mathbf{K}^{(e)}, \mathbf{M}^{(e)}$ and $f^{(e)}$ are computed, they must be assembled inside the global stiffness matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$, the global mass matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ and the global flow vector $f \in \mathbb{R}^n$, respectively (n is the number of nodes in the mesh). This process is repeated over all the elements.

The assembly process of the elemental matrix inside the global matrix is as simple as to add to the global matrix the values of the elemental matrix in the positions of the global matrix which refer to the nodes that conform this particular element. For example, if the element is formed by the nodes 3, 5, 8 and 9, then the elemental matrices are of 4×4, and the value that is at $\mathbf{K}^{(e)}_{1,1}$ must be added to the value of $\mathbf{K}_{3,3}$, while the $\mathbf{K}^{(e)}_{2,4}$ must be added to the $\mathbf{K}_{5,9}$. In the same way $\mathbf{M}$ and f are assembled.

Summarizing, we use the spatial information (node coordinates) and the connectivity matrix (whose nodes form each element) to assembly the stiffness matrix $\mathbf{K}$, the mass matrix $\mathbf{M}$ and the flow vectors f to build the complete system which gives solution to the differential equation 1; then, we use FDM to integrate them over t as explained in [7]:

$$(\mathbf{M} + \alpha \Delta t \mathbf{K})\phi^{t+1} = (\mathbf{M} - (1-\alpha)\Delta t \mathbf{K})\phi^t + \Delta t(\alpha f^{t+1} + (1-\alpha) f^t), \tag{13}$$

if f is constant through time, the system is reduced to

$$(\mathbf{M} + \alpha \Delta t \mathbf{K})\phi^{t+1} = (\mathbf{M} - (1-\alpha)\Delta t \mathbf{K})\phi^t + (\Delta t) f, \tag{14}$$

where $\phi \in \mathbb{R}^n$ is the vector with the node's temperatures, Δt is the step size, and $\alpha \in [0, 1]$ is a parameter to adjust the scheme. Afterwards, α must be selected considering the following facts:

If $\alpha = 0$, the system is a completely explicit scheme and conditionally stable.

If $\alpha = 1$, the system is a completely implicit scheme and unconditionally stable.

If $\alpha = 0.5$, the system is a semi-implicit scheme.

3 Solution of the Linear Strain Problem for Homogenous Solids

We want to determine the displacement field at any domain point, this field is defined as

$$\mathbf{u} = \begin{bmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{bmatrix}. \tag{15}$$

The stress $\varepsilon_p^{(e)}$ at the Gauss point p of the element is computed from the displacement field using the following differential operator:

$$\varepsilon_p^{(e)} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 \\ 0 & \frac{\partial}{\partial y} & 0 \\ 0 & 0 & \frac{\partial}{\partial z} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial z} & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathbf{B}\mathbf{U}^{(e)}, \tag{16}$$

where $\mathbf{U}^{(e)} \in \mathbb{R}^{3n_e}$ is the elemental displacement vector, and $\mathbf{B} = [\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_{n_e}]$ is the elemental strain matrix, that are respectively defined as

$$\mathbf{U}^{(e)} = \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \vdots \\ \mathbf{u}_{n_e} \end{bmatrix}, \quad \text{where } \mathbf{u}_i = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}, \tag{17}$$

$$\mathbf{B_j} = \begin{bmatrix} \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial x} & 0 & 0 \\ 0 & \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial y} & 0 \\ 0 & 0 & \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial z} \\ \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial y} & \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial z} & \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial y} \\ \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial z} & 0 & \sum_{i=1}^{n_e} \frac{\partial N_i}{\partial x} \end{bmatrix}. \tag{18}$$

Then, from the stress and the constitutive equation $\mathbf{D}$ we can compute the strain at the same elemental Gauss point p.

$$\sigma_p^{(e)} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{zx} \end{bmatrix} = \mathbf{D}\varepsilon^{(e)} \tag{19}$$

$$= \underbrace{\frac{E}{d}\begin{bmatrix} (1-\nu) & \nu & \nu & & & \\ \nu & (1-\nu) & \nu & & & \\ \nu & \nu & (1-\nu) & & & \\ & & & (\frac{1}{2}-2\nu) & & \\ & & & & (\frac{1}{2}-2\nu) & \\ & & & & & (\frac{1}{2}-2\nu) \end{bmatrix}}_{\mathbf{D}} \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{zx} \end{bmatrix}$$

where $d = (1+\nu)(1-2\nu)$, and E and ν are the Young and Poisson moduli of the material, respectively.

The stress and strain of the element are obtained by numerical integration using the Gauss quadrature:

$$\varepsilon^{(e)} = \sum_{p=1}^{n_p} \varepsilon_p^{(e)} |\mathbf{J}| w_p, \qquad \sigma^{(e)} = \sum_{p=1}^{n_p} \sigma_p^{(e)} |\mathbf{J}| w_p. \tag{20}$$

The elemental stiffness matrix $\mathbf{K}^{(e)} \in \mathbb{R}^{3n_e \times 3n_e}$ is computed in the same way as in the heat diffusion problem, considering that the elemental strain matrix $\mathbf{B}$ and the constitutive matrix $\mathbf{D}$ are different.

The elemental load vector $\mathbf{f}^{(e)}$ is defined as

$$\mathbf{f}^{(e)} = \sum_{p=1}^{n_p} [\mathbf{B}^T((\sigma^{(e)})_0 - \mathbf{D}(\varepsilon^{(e)})_0) - \mathbf{Hb}] |\mathbf{J}| w_p, \tag{21}$$

where $(\sigma^{(e)})_0$ and $(\varepsilon^{(e)})_0$ are the initial strain and stress of the element, $\mathbf{H} \in \mathbb{R}^{3n_e}$ is a matrix which contains the shape functions in its diagonal and $\mathbf{b} \in \mathbb{R}^{3n_e}$ is the force vector denoted as

$$\mathbf{b} = \begin{bmatrix} (b_x)_1 \\ (b_y)_1 \\ (b_z)_1 \\ (b_x)_2 \\ (b_y)_2 \\ (b_z)_2 \\ \vdots \\ (b_x)_{n_e} \\ (b_y)_{n_e} \\ (b_z)_{n_e} \end{bmatrix}. \tag{22}$$

The assembly of the global stiffness matrix $\mathbf{K} \in \mathbb{R}^{3n \times 3n}$, the global displacement vector $\mathbf{U} \in \mathbb{R}^{3n}$, and the global load vector $\mathbf{f} \in \mathbb{R}^{3n}$ follows the same assembly process explained in the previous section (heat diffusion problem), considering that there are three variables per node instead of one; the detailed steps are shown in [2].

Finally, we solve the system:

$$\mathbf{Ku} = \mathbf{f} \tag{23}$$

to determine the displacements at each node and then compute ε and σ at the elemental Gauss points.

4 Internal Forces Produced by Thermal Variation

If the temperature of a structure increases, the material of such structure reacts with displacements as it is explained in [6]. This explanation is based on experimental evidence,

and the research community has adopted the following model to describe the caused forces:

$$\varepsilon^{(e)} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{xz} \\ \gamma_{yz} \end{bmatrix} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{24}$$

where ω_x, ω_y and ω_z are the thermal distortion constants for each component and dependent of the material.

The forces are going to be recomputed at each time step, serving as input to the software module that solves the solid strain problem; this module will compute the displacements, which will be used to update the mesh geometry as the input to the software module which solves the diffusion heat problem in the next time step.

The program must end when all the time steps (predefined before the start of the simulation) are completed, see Fig 1.

```
read_mesh();
t = 0;
while(t < time_steps){
   solve_diffusion_heat_problem();
   compute_forces();
   solve_strain_solid_problem();
   update_mesh();
   write_results();
   t = t+1;
}
```

Fig. 1. Pseudo-code

In the previous pseudo-code we give a detailed description of the program, where the bold lines are parallel implementations.

5 Parallel Processing

The HPC strategy approach parallelizes the matrix-vector operations of the program (Algorithm 1) and uses Jacobi preconditioned conjugate gradient, as it is shown in [8], to solve the equation systems (Algorithm 2). Fig. 2 shows how many times it takes for Algorithm 1 to compute a system with more than 185 000 variables as a function of the number of cores.

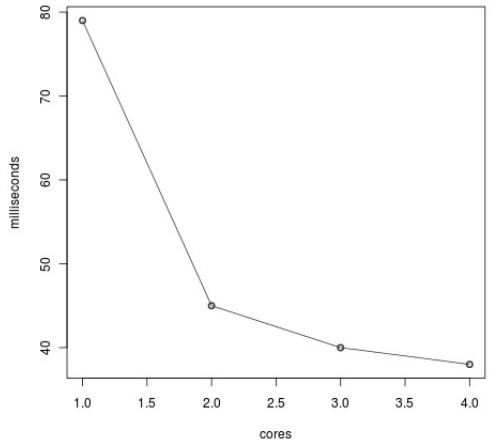

Fig. 2. Cores vs. Time processing for Algorithm 1

A core is a processing unit. A processor can have one or more cores and its own RAM memory, with intermediate cache memories which make the data traffic between the processor and the RAM easy. The management of memory and thread creation to execute several simultaneous processes is done using OpenMP which is an API for shared memory parallel computing. In [4], OpenMP is explained with examples and applications.

In order to use the least possible amount of memory, the Compression Storage by Rows format (CSR) is used to house the sparse matrix, explained in detail in [9]. The CSR format maintains an array with indices and an array of values for each row of the matrix, only the values different from zero (nonzero values) are stored. The indices of the rows correspond to the column values and must be sorted in ascending mode. For example, the matrix

$$\mathbf{A} = \begin{bmatrix} 5.2 & 2.6 & 0 & 0 & 0 & 0 \\ 2.6 & 6.1 & 1.8 & 0 & 0 & 0 \\ 0 & 1.8 & 3.3 & 8.4 & 0 & 0 \\ 0 & 0 & 8.4 & 2.7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.2 & 1.4 \\ 0 & 0 & 0 & 0 & 1.4 & 5.6 \end{bmatrix},$$

must be stored (in CRS format) as:

$$
\begin{array}{lcl}
\mathbf{A}_0^{(j)} & \leftarrow & 0,1 \\
\mathbf{A}_0^{(v)} & \leftarrow & 5.2, 2.6 \\
\mathbf{A}_1^{(j)} & \leftarrow & 0,1,2 \\
\mathbf{A}_1^{(v)} & \leftarrow & 2.6, 6.1, 1.8 \\
\mathbf{A}_2^{(j)} & \leftarrow & 1,2,3 \\
\mathbf{A}_2^{(v)} & \leftarrow & 1.8, 3.3, 8.4 \\
\mathbf{A}_3^{(j)} & \leftarrow & 2,3 \\
\mathbf{A}_3^{(v)} & \leftarrow & 8.4, 2.7 \\
\mathbf{A}_4^{(j)} & \leftarrow & 4,5 \\
\mathbf{A}_4^{(v)} & \leftarrow & 1.2, 1.4 \\
\mathbf{A}_5^{(j)} & \leftarrow & 4,5 \\
\mathbf{A}_5^{(v)} & \leftarrow & 1.4, 5.6
\end{array}
,
$$

where $\mathbf{A}_i^{(j)}$ is the value for the corresponding columns at the row i of the matrix $\mathbf{A}$, with the index numeration starting from zero (like in C/C++), and $\mathbf{A}_i^{(v)}$ being the vector of values of the i^{th} row of the same matrix.

In the previous example, we store only 38.88% of the values of $\mathbf{A}$, but this percentage is lower than 0.1% when we work with matrices generated with FEM in 3D problems with more than 100 nodes.

Algorithm 1. Matrix-Vector multiplication (C code)

```
void multiply(csr* const M,
                double* const vec,
                double* out){
    int i;
    #pragma omp parallel for private(i)
    for(i=0; i < M->rows; i++) {
       int j, col;
       out[i] = 0;
       for(j=0; j < M->rows_size[i]; j++){
          col = M->index[i][j];
          out[i] +=
             M->values[i][j]*vec[col];
       }
    }
}
```

To compile Algorithm 1, we have to tell the compiler that we use OpenMP; in the case of `gcc`, we use the flag `-fopenmp`.

The argument `*M` is a pointer to a C structure which implements the CSR format to store a sparse matrix, whose code is shown in Fig. 3.

```
typedef struct {
     int rows;
     int *rows_size;
     int **index;
     double **values;
}csr;
```

Fig. 3. Compress storage by rows (CSR) format C implementation

where `rows` is the number of the matrix rows, `*rows_size` is an array of length equal to `rows` which contains the number of columns for each row, `**index` is a double array which contains the indices of enabled columns of each row, and `**values` store its values.

Algorithm 2. Preconditioned conjugate gradient

Given the matrix $\mathbf{A}$, the initial vector $\mathbf{x}_o$ and the vector $\mathbf{b}$ of the equation system $\mathbf{Ax} = \mathbf{b}$,

Given a convergence tolerance ϵ.

$\mathbf{g}_0 \leftarrow \mathbf{A}\mathbf{x}_0 - \mathbf{b}$

$\mathbf{q}_0 \leftarrow diagonal(\mathbf{A})^{-1}\mathbf{g}_0$ (Jacobi)

$\mathbf{p}_0 \leftarrow -\mathbf{q}_0$ (Descent direction).

$k \leftarrow 0$ (Iteration counter).

While $\mathbf{g}_k > \epsilon$

$\mathbf{w}_k \leftarrow \mathbf{A}\mathbf{p}_k$ **(Parallel operation)**

$$\alpha_k \leftarrow \frac{\mathbf{g}_k^T \mathbf{q}_k}{\mathbf{p}_k^T \mathbf{w}_k}$$

$\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{p}_k$

$\mathbf{g}_{k+1} \leftarrow \mathbf{g}_k + \alpha_k \mathbf{w}_k$

$\mathbf{q}_{k+1} \leftarrow diagonal(\mathbf{A})^{-1} g_{k+1}$

$$\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{q}_{k+1}}{\mathbf{g}_k^T \mathbf{q}_k}$$

$\mathbf{p}_{k+1} \leftarrow -\mathbf{q}_{k+1} + \beta_k \mathbf{p}_k$

$k \leftarrow k + 1$

End of iterations

The solver algorithm was designed to achieve high performance on two levels. On the upper level, the CSR format allows the parallelization of the matrix-vector multiplication on several cores. On the lower level, by allocating each row of the sparse matrix in a compact way, the algorithm makes a better use of the cache memory of each core, increasing the cache hits and reducing the number of accesses to the RAM (the access time to the cache memory is at least an order of magnitude faster than the access to the RAM), see [5].

6 Application Example

We model a conventional car braking system formed by a disk which is slowed by friction when a hydraulic device (mounted over the disk) presses both sides of its geometry. To simulate friction, we impose Dirichlet boundary conditions on the equation system of the heat diffusion problem, where it is supposed to be the contact. The following function defines the temperature that evaluates the friction:

$$\phi(t) = \lambda \, log(1 + t) \tag{25}$$

where λ is a constant dependent of the material. This function is proposed only to show how the method works, but the heat produced by the friction between two materials doesn't necessarily have this behavior.

GiD is a software for pre- and post-processing of FEM results (among other numerical analyses), the user manual and the reference manual can be downloaded from `www.gidhome.com/support/manuals`.

Fig. 4 shows the real system and the computerized model created with GiD to realize the numeric simulation.

Generally, the disks of the real braking system are made of cast iron, based on [11], we choose the following parameters to approximate the same behavior for the computer model:

$$\begin{aligned} k_x &= k_y = k_z = 83.2646 \\ \omega_x &= \omega_y = \omega_z = 4.0\mathsf{E}-05 \\ c_p &= 1 \\ Q &= 460 \\ \rho &= 7870 \\ \lambda &= 50 \end{aligned}$$

The mounted device pads (braking pads) are made of high iron content, but the device is made of steel with low carbon content, and its properties are approximately

$$\begin{aligned} k_x &= k_y = k_z = 45.0810 \\ \omega_x &= \omega_y = \omega_z = 1.5\mathsf{E}-06 \\ c_p &= 1 \\ Q &= 490 \\ \rho &= 7850 \\ \lambda &= 50 \end{aligned}$$

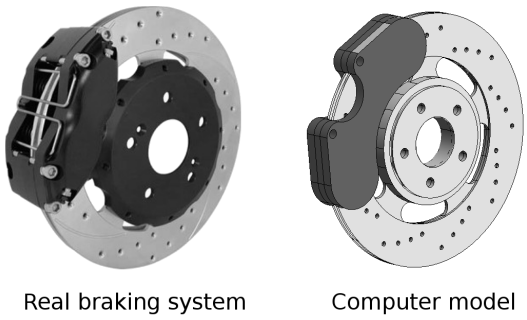

Fig. 4. Real braking system and its computer model

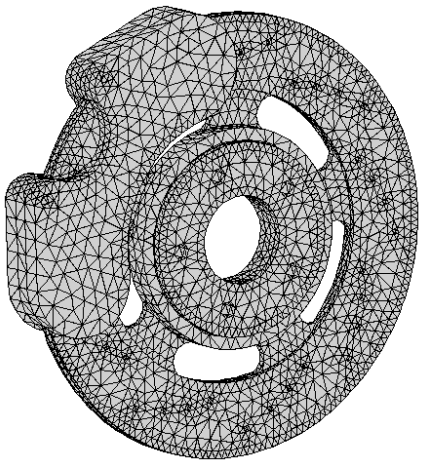

Fig. 5. Finite Element Mesh of the model

The mesh which discretizes the braking device has 51464 elements and 9612 nodes, while the mesh of the disk has only 22844 elements and 8116 nodes. The device mesh must be finer than the disk mesh in order to capture the temperature variations in short spaces. Fig. 5 shows the mesh of both geometries.

The computation was made with $\Delta t = 4.2\mathsf{E}-05$ for 1000 time steps using $\alpha = 1$ (parameter to adjust the FDM schema), and then we post-processed the results using GiD.

Fig. 6. Simulation of step 80 with transparency

Fig. 6 shows the simulation at the time step 80 (0.00336 seconds) with transparency.

If we compare the visualizations (with and without transparency) of the simulation step 1000 (0.042 seconds) in Fig. 7, we can see the increment of the temperature.

The heat spreads through the disk from the friction surface, and in the same way, the braking device starts to heat, although with a lower rate than the rotating disk, because the physical properties of heat diffusion are different for each material.

The distance (with respect to the center) displaced by the edge of the disk caused by the distortion of the solid, represents just the 4.081% of the initial distance. Fig. 8 shows the initial and the final geometry of the solid. The distortion is evident in the middle of the brake device.

Using 4 cores working in parallel, we spend 1340.12 seconds to complete the process (1.34 seconds for each time step), while the solution of the same problem using only one core takes 2380.35 seconds (2.38 seconds for each time step approx.). We achieved the goal reducing the computing time 56.29% of the serial version.

7 Conclusions

We show how to compute a solid distortion simulation caused by temperature variation using HPC techniques on two levels: a lower level using cache-oblivious algorithms (or cache-transcendent algorithms) and a high level using OpenMP to solve

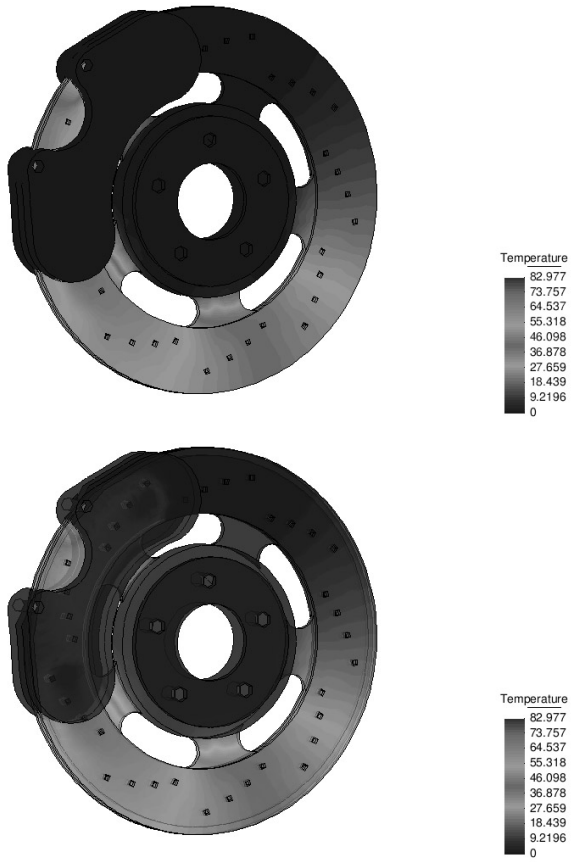

Fig. 7. Simulation of step 1000 with and without transparency

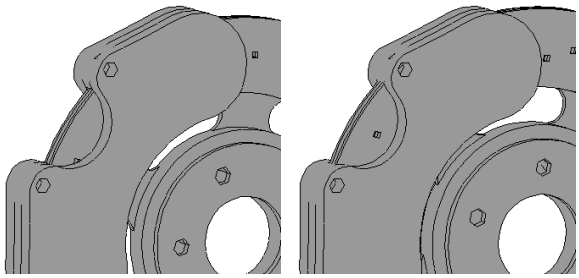

Fig. 8. Initial geometry (left) and final distortion (right)

the equation system taking advantage of several cores.

The implementation solves the heat diffusion problem and the solid strain problem separately, but with feedback between them. We highlight the processing time reduction of the solution with an application example.

References

1. **Alizadeh, H.** (**2010**). *Simulation of Heat-Induced Elastic Deformation of Cylindrical-Shaped Bodies*. Master's thesis, Friedrich-Alexander-Universitat Erlangen-Nurnberg.
2. **Botello, S., Esqueda, H., Gómez, F., Moreles, M., & Oñate, E.** (**2004**). *Módulo de aplicaciones del método de los elementos finitos MEFI 1.0*, chapter Manual Teórico. CIMAT & CIMNE, 6–69.
3. **Botello, S., Moreles, M., & Oñate, E.** (**2009**). *Módulo de aplicaciones del método de los elementos finitos para resolver la ecuación de Poisson MEFIPOIS 1.0*, chapter Manual Teórico. CIMAT & CIMNE, 6–69.
4. **Chapman, B., Jost, G., & Van Der Pas, A.** (**2008**). *Using OpenMP. Portable Shared Memory Parallel Programming*. Massachusetts Institute of Technology.
5. **Drepper, U.** (**2007**). What every programmer should know about memory.
6. **Krysl, P.** (**2010**). *Thermal and Stress Analysis with the Finite Element Method*, chapter Galerkin Formulation for Elastodynamics. Pressure Cooker Press, 292–299.
7. **Lewis, R., Nithiarasu, P., & Seetharamu, K.** (**2004**). *Fundamentals of the Finite Element Method for Heat and Fluid Flow*, chapter Transient Heat Conduction Analysis. John Wiley & Sons, 150–172.
8. **Nocedal, J. & Wright, S.** (**2006**). *Numerical Optimization, Second Edition*, chapter Conjugate Gradient Methods. Springer, 118–120.
9. **Saad, Y.** (**2003**). *Iterative methods for sparse linear systems, Second Edition*, chapter Sparse Matrices. SIAM, 92–94.
10. **Siu-Wing, C., Krishna, D., & Shewchuk, J.** (**2013**). *Delaunay Mesh Generation*, chapter Three-dimensional Delaunay Triangulations. Chapman & Hall/CRC, 85–102.
11. **Wilson, J.** (**2007**). Thermal diffusivity. `http://www.electronics-cooling.com/2007/08/thermal-diffusivity/`.

Victor E. Cardoso-Nungaray graduated from the *Instituo Tecnológico y de Estudios Superiores de Monterrey* as Information Technology Engineer, he obtained a Master's degree in Information Technology from the same University and an M.Sc. in Industrial Mathematics and Computational Sciences from the *Centro de Investigación en Matemáticas A.C.* He is currently enrolled in the Ph.D. Computer Sciences program of the CIMAT. His work is related to numerical methods, discrete element, finite element, PDE's solutions, parallel programming and optimization.

Miguel Vargas-Félix graduated from the *Centro de Investigación en Matemáticas, A.C.* and obtained a M.Sc. in Industrial Mathematics and Computational Sciences. He is currently enrolled in the Ph.D. Computer Sciences CIMAT program. He is the main developer of FEMT, a library to solve large systems in parallel using FEM. His work is related to numerical linear algebra for sparse matrices, finite element, isogeometric analysis, partial differential equations and parallel computing.

Salvador Botello-Rionda graduated from *Universidad de Guanajuato* as Civil Engineer and obtained a Master's degree in Engineering from the *Instituo Tecnológico y de Estudios Superiores de Monterrey*, afterwards he received his Ph.D. from the *Universitat Politécnica de Catalunya* in the same field. His main interests are solid mechanics and flows, applications of the Finite Element Method, multiobjective optimization and image processing. He is author of 33 international journal papers, 8 national journal papers, 19 books and 15 books chapters, and is a member of the National System of Researches of Mexico (Level II). Also, he was editor of 6 congress memories.

Article received on 10/02/2013; accepted on 20/07/2013.