Asociación Argentina
de Mecánica Computacional

Mecánica Computacional Vol XXIX, págs. 2969-2984 (artículo completo)
Eduardo Dvorkin, Marcela Goldschmit, Mario Storti (Eds.)
Buenos Aires, Argentina, 15-18 Noviembre 2010

ACCELERATING THE RESOLUTION OF AN INVERSE THERMAL PROBLEM VIA POSIX THREAD API AND OPTIMIZATION FLAGS

Jonas L. Ansoni [a], Analice C. Brandi [a], Paulo Seleghim Jr. [a]

[a]*Thermal and Fluids Engineering Laboratory, EESC, University of São Paulo, Av. Trabalhador São-carlense 400, 13566-970, São Paulo, Brazil, jonasansoni@sc.usp.br,analice@sc.usp.br, seleghim@sc.usp.br, http://netef.blogspot.com*

Keywords: Parallel processing, POSIX Thread, Optimization flags, Inverse thermal problem.

Abstract. A major difficulty for the resolution of an inverse thermal problem is its high computational cost. Therefore, thermal tomography has found a limited application to industrial processes where fast execution is required, e.g., mapping imperfections in the process of manufacturing castings. In an attempt to improve the computational performance of the resolution of an inverse thermal problem developed, the computational power of multi-core CPUs was used by programming tool parallel architectures shared memory POSIX Threads (Pthreads). Tests were performed on processors with different microarchitectures (Intel® Core 2 Duo, Intel® Core 2 Quad and Intel® Core i7) and different amounts of threads. The parallel implementation of our algorithm exploiting the processing power of multi-core CPUs with the aid of optimization flags arrived to get a performance of up to four times faster than the serial version of the code on the same CPU representing a significant gain for simulations involving the resolution of inverse thermal problems.

1 INTRODUCTION

Industrial process tomography is a technique used to determine the internal properties of an object through nonintrusive external measurements. The main application of thermal tomography is to map imperfections (fouling, cracks, voids, etc.) in solid materials. Another important application is in the measurement of thermophysical properties in processes involving fluids. It is the case of flowmeters of thermal principle, in which the velocity of the flow is determined by the measurement of the convection coefficient. The principle of thermal tomography operation is usually based on the application of a heat flux at the external boundary of the process or object and on the measurement of the response temperatures at that same boundary.

The construction of an image from signals with the aid of tomography techniques generally involves solving an inverse problem. A thermal problem is described by a differential equation that governs the temperature inside the domain, and by adequate boundary conditions, which simulate the excitation and measurement process. In the inverse problem studied in this paper, we assume that no access is granted to the surface where the convection coefficient distribution is to be determined. This lack of information, whose minimization corresponds to the sought solution (Borcea, 2003, Özisik and Orlande, 2000) is defined.

Among the main advantages of thermal tomography are its low cost and robustness, which make it extremely interesting for industrial applications. However, the problem of thermal tomography is highly expensive, in terms of computational cost, requiring considerable CPU time, particularly to calculate and regularize the Hessian matrix of Newton's optimization method used to solve the problem. The Hessian matrix is singular due to the intrinsic ill-conditioned nature of the inverse problem (Brandi et al., 2009), being extremely sensitive to the presence of rounding errors.

Table 1 shows the number of times the Preconditioned Bi-Conjugate Gradient (PBCG) method is called and percentage of time spent on the calculation and assembly of the Hessian matrix in an iteration of Newton's method.

Newton's method	Calculation and assembling of Hessian matrix	Percentage of time (s)
3869	3813	98%

Table 1: Number of times the PBCG method is called and percentage of times spent on the calculation and assembly of the Hessian matrix.

The computational power of modern shared memory multiprocessor architectures (Symmetric Multiprocessing - SMP) was used to reduce the computational cost of the calculation and assembling of matrix Hessian of the inverse thermal problem.

In our problem the PBCG method solver is called several times independently in the assembly of the Hessian matrix of the problem, characterizing a case of Single Instruction, Multiple Data (SIMD) and allowing its parallelization. A tool for parallel programming in shared memory architectures POSIX Thread (Pthreads) was used to manage the threads in multi-cores processors. In another study we use the processing power of GPUs to accelerate the process of calculating and assembling the Hessian matrix (Ansoni et al., 2010).

2 POSIX THREADS

The POSIX Threads library is an IEEE POSIX (Potable Operating System, Interface for UNIX) for thread, which defines a standard API to create and manipulate threads in C/C++ languages. The libraries implemented are also called Pthread, they are widespread in the Unix universe and other operating systems, like Linux and Solaris, but there are implementations for Windows (Barney, 2010, Butenhof, 1997). This library provides us with effective ways to expand into new process running concurrent processes, which can run more efficiently on computer systems with multiple processors and/or processors with multiple cores.

In general, for a program to take advantage of Pthreads, it should be organized into discrete, independent tasks that can be performed simultaneously. Pthreads are defined as a set of C language types and procedure calls, implemented with a header (header/include) pthread.h. There are about 100 procedures in Pthreads, all with the prefix "pthread_". The operations with threads include creation, termination synchronization, scheduling and signaling. All threads have access to the same global shared memory, and the programmer is responsible for synchronizing access (protection) globally shared data.

As multiple threads can concurrently access the same variables, there may be cases in which multiple threads access and modify the same data at the same time causing "race conditions[1]" as seen in Figure 1.

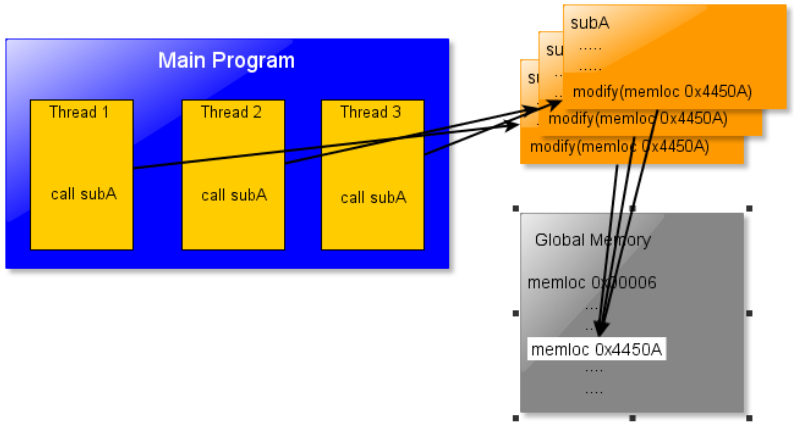

Figure 1: Race Condition (Barney, 2010).

The original Pthreads API was defined in the ANSI/IEEE POSIX 1003.1 – 1995 standard. The POSIX standard has continued to evolve and undergo revisions, including the Pthreads specifications. The subroutines which comprise the Pthreads can be informally grouped into four major groups (Barney, 2010):

- Thread management: Routines that work directly on threads – creating, detaching, joining, etc. (e.g.: pthread_create).

[1] Occurs when two threads modify a variable at the same time, there is a race condition, except that in this race, the last thread that modifies the values is having its stored value.

- Mutexes: Routines that deal with synchronization, called mutex, which is an abbreviation for mutual exclusion (e.g.: pthread_mutex_lock).
- Condition variables: Routines that address communications between threads that share a mutex (e.g.: pthread_cond_init).
- Synchronization: Routines that manage read/write locks and barriers (e.g.: pthread_join).

In the example we can understand the execution of multiple threads: the program starts with the main thread, which creates other threads (pthread_creat) and expects them to end their work (pthread_join). Each thread performs work at the same time. After pthread_join execution, the main thread ends the execution of the serial program.

Figure 2 we shows the possible execution flow of the example.

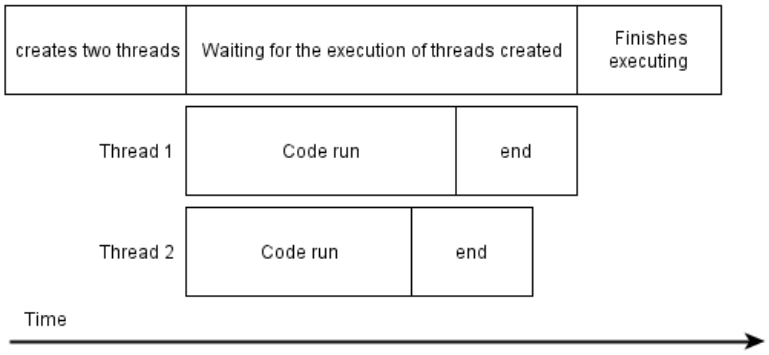

Figure 2: Possible flow of execution.

Some of the main advantages of using Pthreads are:

- Uses of the full potential of the new multi-core processors, which is becoming increasingly common today;
- Portability of written applications using Pthreads library for any operating system that supports the POSIX standard;
- The cost of exchange contexts between threads is lower than with processes, due to the fact the threads are lighter;
- System overhead reduction;

Disadvantages:

- Program complexity increases due to the use of various resources for management and synchronization between threads;
- Occurrence of race conditions and deadlocks.

Currently, every application requires a parallel version, as the trend is the use of multi-core processors. The POSIX thread is an option that allows maximizing use of these processors. Due to these advantages, Pthreads was used to improve the computational time of the resolution of the inverse thermal problem.

3 STATEMENT OF THE PROBLEM

Let θ and T be thermal potentials and k the thermal conductivity of the medium. The ambient temperature is T_∞ and, thus, $\theta = T - T_\infty$. Denoting the domain of the problem by Ω, the heat conduction equation is written as

$$\vec{\nabla}.(-k\vec{\nabla}\theta) = 0 \text{ in } \Omega. \tag{1}$$

The interaction between Ω and the exterior occurs through the boundary $\partial\Omega$ and is defined by relationships between thermal excitation and response. Consider the following mixed boundary condition, i.e., the coupling between conduction and heat convection

$$-k\frac{\partial\theta}{\partial n} = h\theta + q \text{ in } \partial\Omega, \tag{2}$$

where n is the normal vector, h is the convection coefficient and q is the heat flux imposed at the boundary. If all the parameters of equations (1) – (2) were known, a mathematically well-posed problem would be constituted, which could be solved by conventional numerical techniques. However, this is not the case in thermal tomography: one or more parameters are unknown in parts of the boundary. The strategy for the solution consists in palliating this information deficiency through the measurement of redundant boundary conditions at the accessible parts of the boundary.

The finite element method was chosen to discretize the problem equations from the following residual equation based on a convenient weight function (v) (Becker et al., 1981)

$$\int_\Omega k\vec{\nabla}\theta.\vec{\nabla}v\ dxdy + \int_{\partial\Omega_2}[h\theta + q]v\ ds = 0. \tag{3}$$

Our approach to solve the inverse problem associated with the reconstruction of the convection coefficient from boundary data is based on an error functional assessing the difference between measured and model temperatures.

Assuming a prospective distribution of h (h_{num}), the set of equations (1)-(2) is used to simulate the problem, returning T_{num}. In this work, instead of taking the experimental measurements, they are simulated through the same set of equations (1)-(2), however solved for a reference convection coefficient h_{actual}, and having T_{actual} as the result.

Thus, two different models were considered: one corresponding to the experimental assembly itself (actual or analogical model) and the other corresponding to the numerical implementation of equations (1)-(2) on a digital computer (mathematical-numerical model) (ROLNIK and Seleghim, 2006). The error functional can be written as follows

$$e(h_{num}) = \left\| T_{actual} - T_{num} \right\|, \tag{4}$$

where temperatures T_{actual} and T_{num} refer only to the accessible region of the boundary where measurements can be accomplished. In Eq. (4), the Euclidian norm is usually adopted, that is,

$$e(h_{num}) = \sqrt{\sum (T_{actual} - T_{num})^2}. \tag{5}$$

The correct convection coefficient distribution can be found by an optimization method that produces successive corrections d_k to prospective solutions h_k minimizing the error function given by Eq. (5). In mathematical terms, this can be expressed as (Colaço et al., 2006, Nocedal and Wright, 1999)

$$h_{k+1} = h_k + \lambda_k d_k, \tag{6}$$

where λ_k, is the search step size, d_k is the direction of descent and k is the number of iterations. The graphs of h_k or its statistical moments over the iterations correspond to the here called convergence trajectories and reveal extremely important aspects about the ill-conditioned nature of the problem.

The majority of the deterministic optimization methods can be described by Eq. (6), and they differ from each other by the form of calculating the descent direction. For instance, in the Steepest Descent method (Nocedal and Wright, 1999), which is a gradient-based method, d_k are corrections along the gradient of the error function which corresponds to downward movements along the steepest descent. Therefore, the direction of the descent is given by

$$d_k = -\vec{\nabla} e(h_k), \tag{7}$$

where $\vec{\nabla} e$ corresponds to the gradient vector associated with the error surface and calculated at the $k-th$ iteration.

The main characteristic of this method is the capacity to converge for the solution even if the initial guess is distant from the global minimum. An important disadvantage is the easiness which convergence trajectory is trapped by local minima.

A optimization technique is the Newton method (Nocedal and Wright, 1999), which in addition to using first-order derivative of the error functional, such as the Steepest Descent and the Conjugate Gradient methods, also uses information of the second derivative in order to achieve a faster convergence rate. Therefore, in this method the corrections are given by

$$d_k = -H(h_k)^{-1} \vec{\nabla} e(h_k), \tag{8}$$

where H is the Hessian matrix containing the second-order derivatives of the error functional in relation to the local values of the convection coefficient

$$H_{i,j} = \frac{\partial^2 e}{\partial h_i \partial h_j}. \tag{9}$$

Although the convergence rate of the Newton method is quadratic, the calculation of the Hessian matrix is computationally expensive. As a result, other methods have been developed that approximate the Hessian matrix with simpler and faster computing forms. In this work, the finite difference method was adopted to calculate the Hessian matrix, with derivatives approximated by standard second-order central differences.

In brief, the sequence that hypothetically leads to the solution (global minimum) is the following:

$$h_{k+1} = h_k - \lambda . \Delta h_k = h_k - \lambda . H(h_k)^{-1} \overrightarrow{\nabla e(h_k)}, \tag{10}$$

where Δh are corrections in vector h.

This method is very powerful and straightforward if the Hessian matrix associated with the optimization problem is well behaved. Since the problem we are dealing with is inverse and, therefore, intrinsically ill-conditioned, H is expected to be problematic in some sense. More precisely, corrections Δh_k in Eq.(10), which are obtained by solving the Hessian-gradient problem, may lead to an erratic convergence trajectory.

Although the Newton's method obtains a quadratic convergence rate, it may fail to converge depending on whether the initial guess is outside of the convergence attraction region (region near the maximum/minimum) or due to the ill-conditioning of the Hessian matrix.

Considering that the error functional is not regularized by the application of some specific techniques, as the Tikhonov method for instance, some equivalent procedure must be applied to the Hessian matrix. The Singular Value Decomposition was adopted in this work to obtain a pseudo-inverse of the Hessian matrix with a better conditioning number: the eigenvectors corresponding to singular or near singular eigenvalues of H are truncated (TSVD) (Brandi et al., 2009). In rough, this corresponds to throwing away sets of equations that are nearly linear-dependent or corrupted by round off errors and, thus, solving Eq. (10) in an average sense. These problematic equations attract the solution to the null space associated with H and, consequently, result in increasingly larger corrections d_k towards infinity.

The strategy for solving the inverse problem can be seen in Figure 3. The optimization cycle starts with the arbitration of a prospective distribution coefficient of convection.

This completes the boundary conditions and, consequently, the equations of heat conduction can be solved, producing a prospective distribution of temperatures in the problem domain. Then, the temperatures of the accessible parts of the boundary are compared with their respective values measured experimentally, generating an error functional that quantifies this difference. An optimization algorithm based on this functional error can produce corrections to be applied to the initial distribution of the convection coefficient, and the process is iterated from this point.

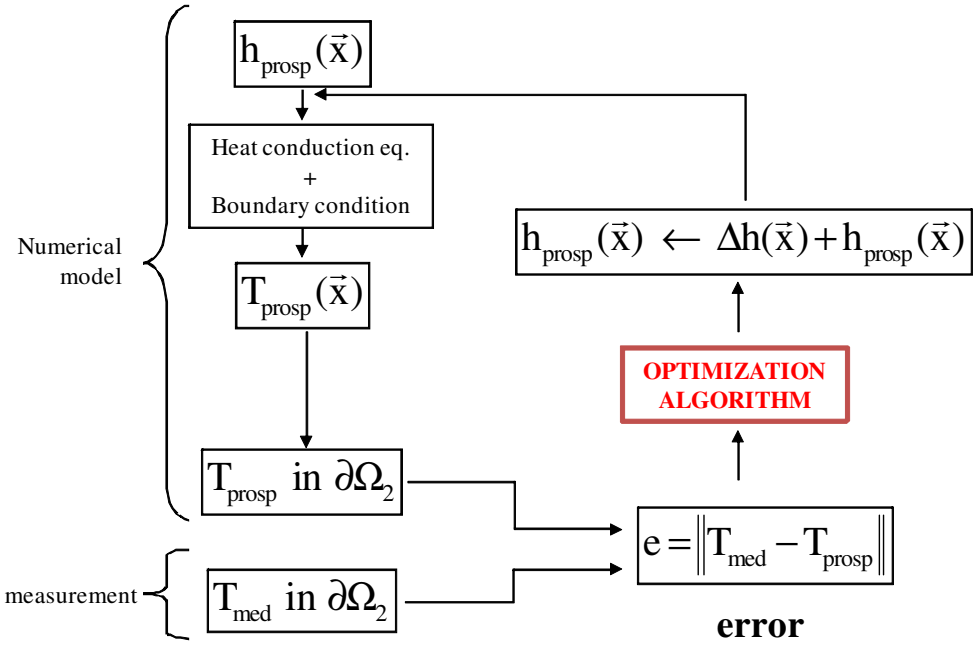

Figure 3: Solution algorithm.

4 PARALLEL APPROACH

The parallelization of the code using the API Pthread derived directly from a sequential code for solving the inverse thermal problem implemented in C language. The Pthread allows the creation of threads and for each thread created can be assigned a function, along with their arguments.

The implementation of Pthreads focuses on parallelizing the calculation and assembly of the Hessian matrix of the inverse thermal problem, which is the major bottleneck in the processing of the problem. To assemble the Hessian matrix it is necessary to solve several sparse linear systems that arise from the central difference method used to discretize the second derivatives.

Figure 4 shows the model code which was parallelized using the procedures of the Pthread. The program starts from a master thread that executes the serial program. Arriving in the function of calculating and assembling the Hessian matrix threads are created so that each one created is responsible for calculating and assembling a line of Hessian matrix. The threads run the finite element method (FEM) in parallel to generate the sparse linear systems which are settled soon afterwards by the Preconditioned Bi-Conjugate Gradient (PBCG) (Press et al., 1992), which returns a fitness value used in the calculation and assembling of the Hessian matrix.

At the end of the calculation and assembly of the Hessian matrix, the code execution proceeds in serial mode through the master thread. As the parallel programming model adopted uses shared memory, it is necessary to be careful so that race conditions do not occur in the variables used.

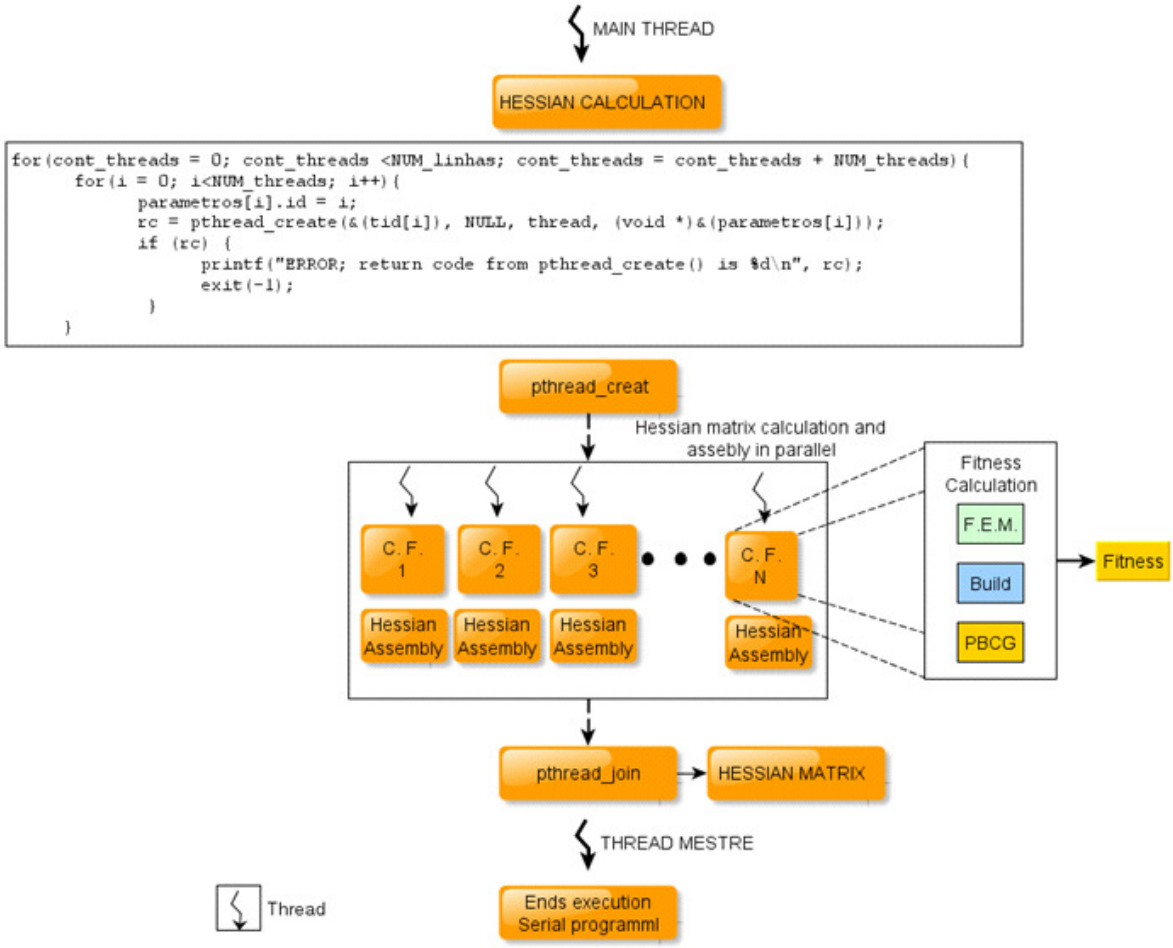

Figure 4: Implementation of the code using Pthread.

5 NUMERICAL SIMULATIONS AND RESULTS

Codes (serial and parallel version) were run on computers with different types of multi-cores CPUs and with different amounts of threads. The specifications of processor and operating systems used are described in Table 2. The compiler used for the codes developed (serial and parallel) in C programming language was compiled with *gcc 4.3* with the aid of the optimization flag –O2.

Optimization flags were used to achieve the maximum possible performance to reduce the computational cost of the inverse thermal problem. *CFLAGS* are flags of compiler C, usually options for GCC (GNU Compiler Collection) and it can be used to customize and optimize applications when it is built from source. They are commonly utilized to specify the architecture of your computer as well as the type of CPU used and any other special option that want to enable or disable. This information is important for the GCC which the flag tells exactly how to customize the assembly instructions that it creates through the source code of the application.

All times shown are the averages of 10 consecutive executions of each algorithm with 10 iterations of Newton's method. The total execution time of the program was calculated using the time function of Linux.

To analyze the convergence of the parallel algorithm were performed 1000 iterations of the Newton's method and compared with the serial version of the code. All tests used variables with double precision.

	CPU 1	CPU 2	CPU 3
Processor	**Intel Core 2 Duo (E4400)**	**Intel Core 2 Quad (Q8200)**	**Intel Core i7 975 Extreme Edition**
Number of cores	**2**	**4**	**4**
Clock Speed (GHz)	**2**	**2.5**	**3.33**
RAM memory	**1 GB DDR2**	**4 GB DDR2**	**12 GB DDR3**
Linux kernel version	**Ubuntu 9.10 Karmic Koala 32 bits**	**Debian Lenny 64 bits**	**Ubuntu 9.10 Karmic Koala 64 bits**

Table 2: Characteristic of the CPUs used.

5.1 Test Case

Consider a two-dimensional domain, that is, a square of unitary sides with $k = 1$ in the whole domain and the other boundary condition parameters according to Figure 5. The governing Eq.(1), in Cartesian coordinates, was discretized by the finite element method. The domain was discretized in a computational mesh of 31×31 points, generating 961 nodes equally spaced and 1800 linear triangular elements. Both temperature variation and thermal conductivity were assumed linear inside the elements.

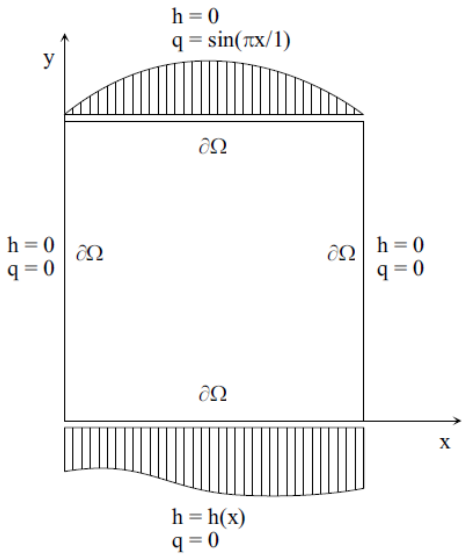

Figure 5: Configuration of the problem simulated numerically.

The reference problem, which mimics the actual experimental test, was defined by setting a reference convection coefficient distribution h_{actual} on the inferior side of the domain $(y = 0)$. More specifically, by fixing h_{actual}, reference temperatures are obtained on the accessible boundary of the domain $(y = 1)$, which would be measured in response to the application of the reference heat flux. As the error (4) decreases during the optimization process, the obtained temperatures of the numerical model converge to the reference problem temperatures and, supposedly, h_{num} converges to h_{actual}. In this work, a triangular convection coefficient was adopted as $h_{actual} = \begin{cases} 2 - 2(x/15), & x \le 15 \\ -2 + 2(x/15), & x > 15 \end{cases}$.

5.2 Performance and Convergence

As previously mentioned, since the inverse problem is intrinsically ill-conditioned, the corresponding optimization problem is pathological. This can be shown by simulating the convergence trajectory on the error surfaces, whose two-dimensional versions were analyzed in (ROLNIK and Seleghim, 2006). More specifically, starting from perturbations of h_{actual} in Eq. (4), the sequence given by Newton's recurrence formula Eq.(10) should converge monotonically if the error surface were concave. As shown in (Brandi, 2010) optimization surfaces are pathological, that is, they contain structures such as narrow valleys, multiple local minima, plateaus, etc., and, consequently, the corresponding convergence trajectories are expected to be erratic.

Three convergence trajectories will be constructed starting from progressively more perturbed versions of h_{actual} given by the formula $h_{inicial} = h_{actual} + \delta$, where δ, corresponds to a uniform random perturbation vector centered around zero, and a standard deviation previously set. This ensures that the initial guess in Eq.(10) be taken progressively more distantly from

the correct solution h_{actual}, where is a triangular function and the perturbations are set according to the following formula:

$$h = \begin{Bmatrix} 2-2(x/15),\ x \le 15 \\ -2+2(x/15),\ x > 15 \end{Bmatrix} + \begin{cases} \delta \in [-2.0, +2.0], \text{ case 1;} \\ \delta \in [-1.0, +1.0], \text{ case 2;} \\ \delta \in [-0.5, +0.5], \text{ case 3.} \end{cases} \tag{11}$$

The convergence trajectory analysis showed to be satisfactory if compared with the serial version, as seen in Figures 6 – 8.

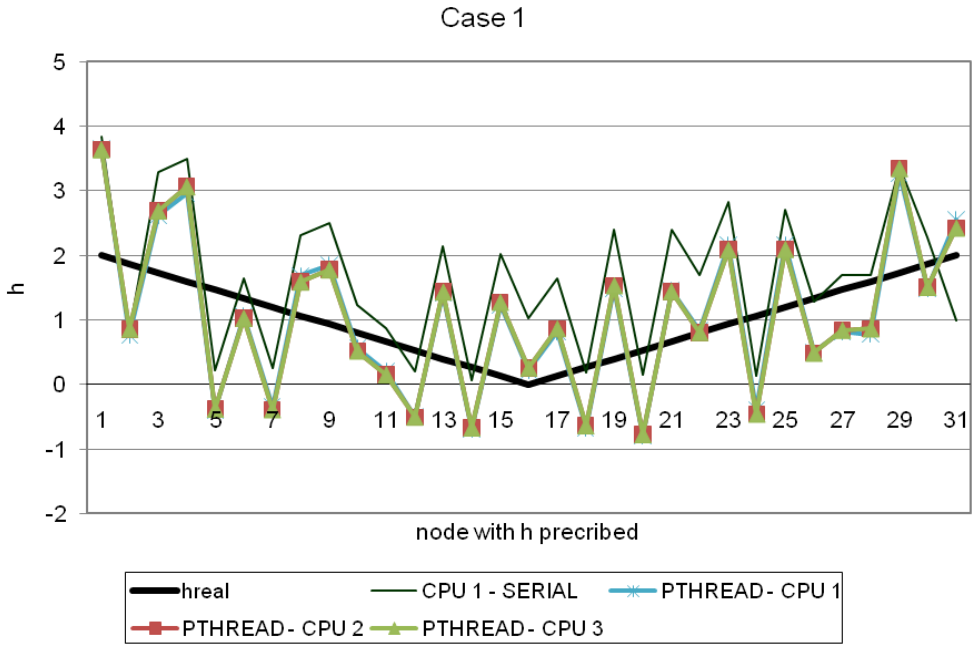

Figure 6: Initial guesses and real solution of the convection coefficient – Case 1.

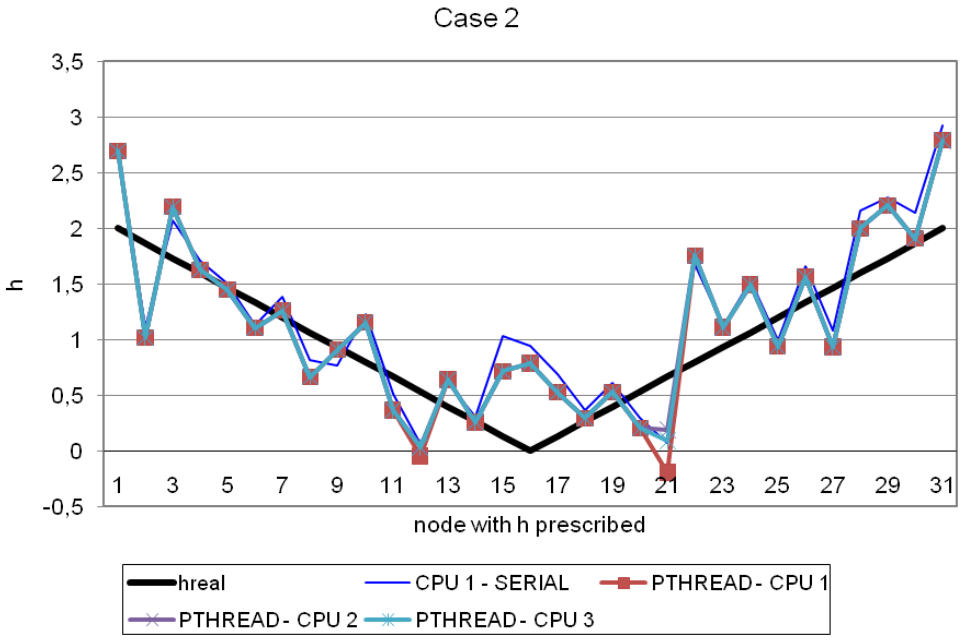

Figure 7: Initial guesses and real solution of the convection coefficient – Case 2.

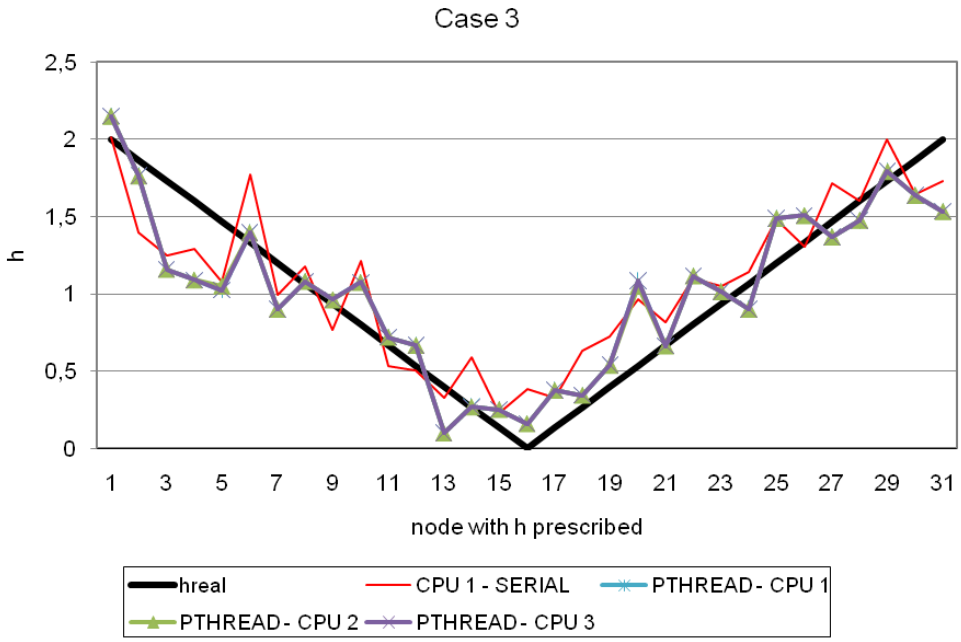

Figure 8: Initial guesses and real solution of the convection coefficient – Case3.

Figure 9 shows the speedups obtained with the parallelized code using the processing power of multi-core processors through Pthreads API. It is possible to observe that the speedups achieved are very close to the amount of physical processor cores, or with the use of the CPU 2 which has four cores showed a speedup close to four.

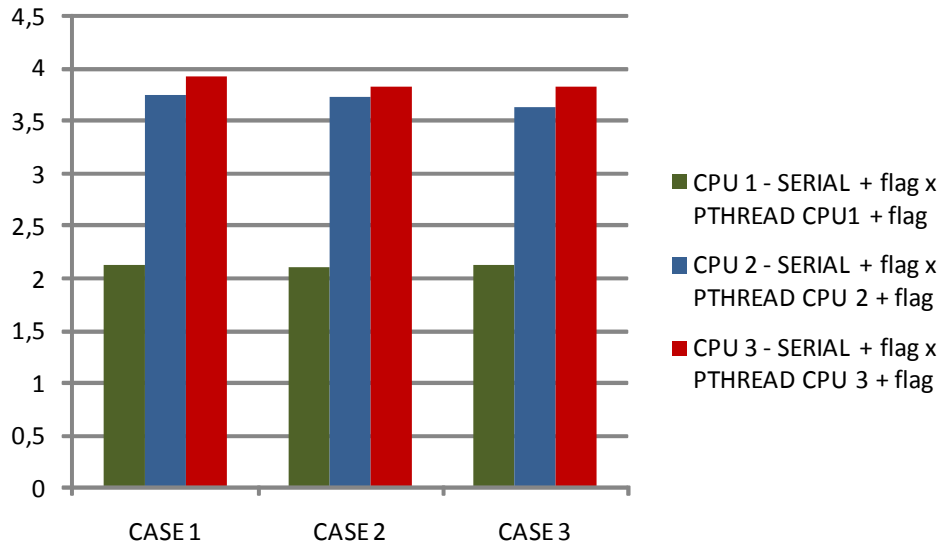

Figure 9: Speedup (Serial and Pthread).

Figure 10 shows the processing time required to perform 10 iterations of Newton's method to solve the problem. We can observe a performance gain using the optimization flag in the serial and parallelized program.

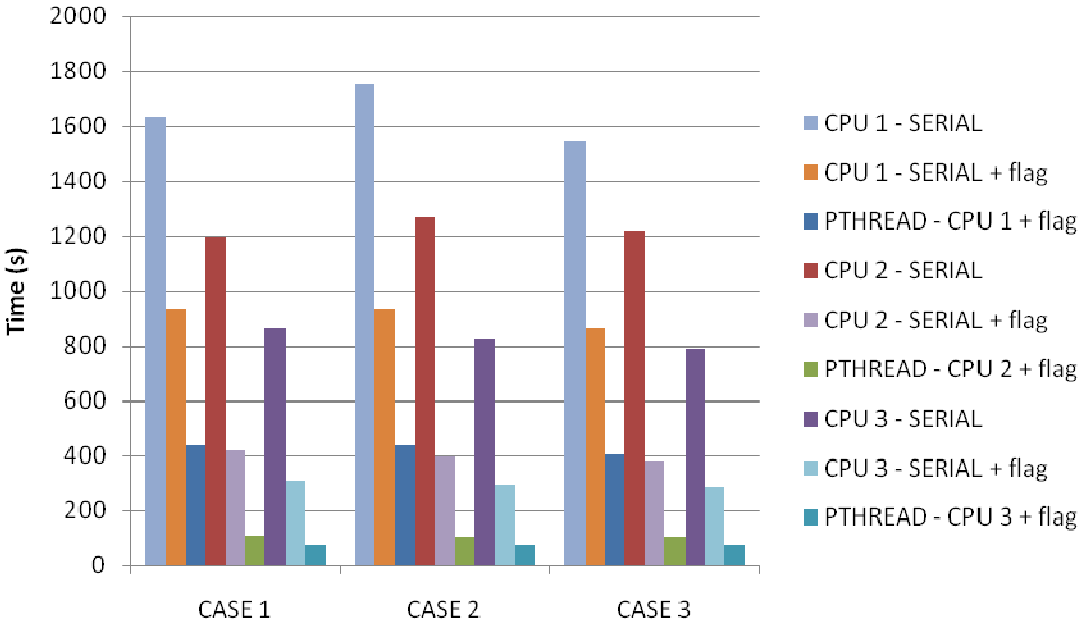

Figure 10: Processing time.

6 CONCLUSIONS

Engineering problems have become increasingly complex and computationally expensive, requiring the search for new tools that make feasible its resolution in a time habile. With the growing trend of multi-core processors has opened up a new perspective in this relentless pursuit to solve increasingly complex problems, however, it is necessary to seek planning strategies to take advantage of these processors so that problems that demand high computational cost can be solved.

This paper has presented an application of the multi-core processors through parallel programming tool for shared memory Pthread combined with optimization flags in order to accelerate the processing time for the resolution of an inverse thermal problem.

The results showed that the implementations of parallel algorithms on shared memory architectures, combined with optimization flags are a very useful tool to solve problems that require high computational time. With the aid of the optimization flags, it was possible to obtain a speedup of the parallelized code about four times faster than the serial version on the same processor. Thus, the optimized code has become a very interesting application, helping to increase the range of tests and the complexity of the problem at a low cost.

REFERENCES

ANSONI, J. L., BRANDI, A. C., MARCO, A. D., CAROSIO, G. L. C. & SELEGHIM JR, P. Year. Sparse matrix solver on the GPU: Conjugate gradient method applied to solve an inverse thermal tomography problem. *In:* Inverse Problems, Design and Optimization Symposium, 2010 João Pessoa, PB, Brazil. IPDO 2010.

BARNEY, B. 2010. *POSIX Thread programming* [Online]. Available: https://computing.llnl.gov/tutorials/pthreads/ [Accessed 06/03 2009].

BECKER, E. B., CAREY, G. F. & ODEN, J. T. 1981. *Finite Elements: An Introduction,* Englewood Cliffs, Prentice-Hall.

BORCEA, L. 2003. Electrical Impedance Tomography. *Inverse Problems,* 18**,** R99-R136.

BRANDI, A. C. 2010. *Desenvolvimento de uma Técnica não Intrusiva de Medicção do Coeficiente de Convecção - Solução do Problema Térmico Inverso.* Doutorado, Escola de Engenharia de São Carlos, Universidade de São Paulo.

BRANDI, A. C., ANSONI, J. L. & SELEGHIM JR, P. Year. Reconstruction of the convection coefficient from non-intrusive measurements: Regularization of the inverse problem by the truncated singular value decomposition method. *In:* 20th International Congress of Mechanical Engineering, November 15-20 2009 Gramado, RS, Brazil. ABCM, 1-8.

BUTENHOF, D. R. 1997. *Programming with POSIX Threads,* Boston, Addison-Wesley Longman Publishing.

COLAÇO, M. J., ORLANDE, H. R. B. & DULIKRAVICH, G. S. 2006. Inverse and optimization problems in heat transfer. *Journal of the Brazilian Soc. Mechanical Sciences and Engineering,* 28**,** 1-24.

NOCEDAL, J. & WRIGHT, S. J. 1999. *Numerical optimization,* New York, Springer.

ÖZISIK, M. N. & ORLANDE, H. R. B. 2000. *Inverse Heat Transfer: Fundamentals and Applications*, Taylor & Francis.

PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T. & FLANNERY, B. P. 1992. *Numerical Recipes in C: The Art of Science Computing,* New York, Cambridge University Press.

ROLNIK, V. P. & SELEGHIM, P. J. 2006. A Specialized Genetic Algorithm for the Eletrical Impedance Tomography of Two-Phase Flows. *Journal of the Brazilian Soc. Mechanical Sciences and Engineering,* 28**,** 378-389.