COMBINATORIAL OPTIMIZATION STRATEGIES FOR THE INEXACT NEWTON-KRYLOV METHOD

Marcelo Carrion, Brenno Lugon
Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, 514 - Goiabeiras - 29075910 - Vitória/ES - Brazil
{marcelo.tperc,brennolugon}@gmail.com

Maria Cristina Rangel, Lucia Catabriga, Maria Claudia Silva Boeres
Universidade Federal do Espírito Santo (UFES)
Av. Fernando Ferrari, 514 - Goiabeiras - 29075910 - Vitória/ES - Brazil
{crangel,luciac,boeres}@inf.ufes.br

ABSTRACT

Combinatorial Scientific Computing is an important interdisciplinary field combining issues from Combinatorial Optimization to solve efficiently Scientific Computing problems. In this work, we solve a 2D and a 3D nonlinear problem using a Newton-type method that requires, at each step, the evaluation of a sparse matrix and the solution of a linear system. The matrix evaluation is optimized using graph coloring techniques and a matrix reordering scheme is used to accelerate the convergence of the preconditioned iterative GMRES solver.

KEYWORDS. jacobian evaluation, graph coloring, matrix reordering, ILU preconditioner

OC - Combinatorial Optimization

1 Introduction

This work focuses on applying combinatorial techniques to efficiently solve a benchmark nonlinear heat transfer problem, considering two and three-dimensional examples. To obtain an approximate solution for the problem, the Newton's method is utilized. This iterative method requires, at each step, the evaluation of a Jacobian matrix and the solution of a linear system. We employ the finite differences method to estimate the Jacobian matrix, and the non-stationary iterative method called GMRES to solve the linear system.

Our goal is to take advantage of known optimization techniques to reduce the computational effort of both processes. As shown in Fig. 1, the Jacobian evaluation is optimized using a graph coloring algorithm and a matrix reordering scheme is used to accelerate the convergence of the preconditioned iterative GMRES solver.

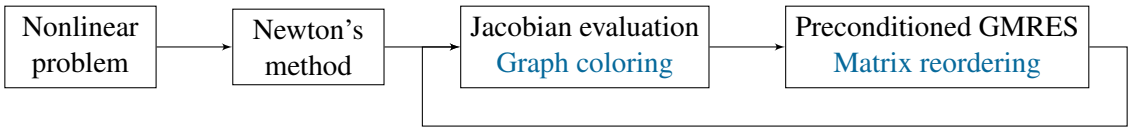


Figure 1: Problem flow and optimizations (in blue) scheme.

The problem of optimizing the evaluation time of a sparse Jacobian matrix leads to the problem of partitioning its columns into the fewest groups, each consisting of structurally orthogonal columns (i.e., columns that do not have a nonzero in the same row). Curtis, Powell, and Reid (1974) were the first to exploit sparsity in this way, followed by Coleman and Moré (1983), that modeled the matrix partitioning problem as a graph coloring problem. Later, Gebremedhin et al. (2005) proposed a different graph coloring formulation. We can find some techniques to obtain the partitioning, such as stencil-based ones, that reach the optimum number of partitions for specific problems (Goldfarb and Toint, 1984) (Lülfesmann and Kawarabayashi, 2014).

The matrix reordering problem can be treated as a graph labeling problem. Algorithms such as RCM (Cuthill and McKee, 1969), GPS (Gibbs, Poole, and Stockmeyer, 1976) and Sloan (Sloan, 1986) are based on search strategies in graphs and provide good solutions. Ghidetti et al. (2011), Ghidetti et al. (2010a), Ghidetti et al. (2010b) and Lugon and Catabriga (2013) compared, besides those already mentioned, a set of reordering algorithms such as Spectral (Barnard et al., 1993), AMD (Davis et al., 1994) and Nested Dissection (George, 1973), evaluating their impact on CPU time when applying an ILU(p) preconditioned GMRES solver.

This paper has been structured in seven distinct sections. Section 2 provides some background information about the Inexact Newton-Krylov Method. In Section 3, the Jacobian evaluation is briefly explained and its correspondent optimization detailed. Section 4 presents the preconditioned GMRES followed by its optimization technique. Section 5 defines the problem used for the numerical experiments, whose results are shown and discussed in Section 6. Finally, in Section 7 we share our conclusions and point to future work in the field.

2 Inexact Newton-Krylov Method

The nonlinear system of equations defined by:

$$\mathbf{F}(u_1, u_2, \ldots, u_N) = \begin{bmatrix} f_1(u_1, u_2, \ldots, u_N) \\ \vdots \\ f_N(u_1, u_2, \ldots, u_N) \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix} \tag{1}$$

can be solved by Newton's method, that is an iterative method for nonlinear equations that approximate the function $\mathbf{F}$ at a given point $\mathbf{u} = (u_1, u_2, \ldots, u_N)^t$ by a linear function. The Jacobian

matrix $\mathbf{J}$ represents the variation of the function $\mathbf{F}$ with respect of $\mathbf{u}$. Each iteration of the Newton's method is given by

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{s}^k \,, \tag{2}$$

where $\mathbf{s}^k$ is calculated by the solution of the linear system:

$$\mathbf{J}(\mathbf{u}^k)\mathbf{s}^k = -\mathbf{F}(\mathbf{u}^k) \,. \tag{3}$$

We may terminate the iteration when the relative nonlinear residual $\|\mathbf{F}(\mathbf{u}^k)\|_2/\|\mathbf{F}(\mathbf{u}^0)\|_2$ is small, i.e., when $\|\mathbf{F}(\mathbf{u}^k)\|_2 < \tau_{res}\|\mathbf{F}(\mathbf{u}^0)\|_2$ for a given tolerance τ_{res}.

When an iterative method is used to solve the system in Eq. (3) the Newton's method is known as the Inexact Newton method, that is especially well suited for large-scale problems and has been used very successfully in many applications (Elias et al., 2004). Furthermore, when iterative Krylov methods are used to solve the linearized system of the Newton-type scheme, the resulting methods are known as Inexact Newton-Krylov methods (INK). They have been used to reduce the computational effort related to nonlinearities in many problems of computational fluid dynamics (Bodart et al., 2011), offering a compromise between the accuracy and the amount of effort spent per iteration. INK success depends mainly on three factors: (i) quality of initial Newton step, (ii) robustness of Jacobian evaluation and (iii) proper forcing term choice (Kelley, 1995).

3 Jacobian Evaluation

Consider $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$, a continuously differentiable vector function. The Jacobian of $\mathbf{F}$ at the point $\mathbf{u}$ is the $N \times N$ matrix $\mathbf{J}(\mathbf{u})$ of all first-order partial derivatives of $\mathbf{F}$, i.e., the matrix whose (i, j) entry $\mathbf{J}(\mathbf{u})_{ij} = \partial f_i(\mathbf{u})/\partial u_j$, where $f_1(\mathbf{u})$, $f_2(\mathbf{u})$, ..., $f_N(\mathbf{u})$ are the components of $\mathbf{F}(\mathbf{u})$.

Computationally, this matrix can be obtained in two ways: through finite difference approximation or by exact computation, using tools based on automatic differentiation (Griewank, 2000), denoted by *AD*. Employing a forward difference approximation, one can estimate the jth column of $\mathbf{J}(\mathbf{u})$ through the following formula:

$$\mathbf{J}(\mathbf{u})\mathbf{e}_j = \partial\mathbf{F}(\mathbf{u})/\partial u_j \approx [\mathbf{F}(\mathbf{u} + \varepsilon\mathbf{e}_j) - \mathbf{F}(\mathbf{u})]/\varepsilon \,, \quad 1 \leq j \leq N \,, \tag{4}$$

where $\mathbf{e}_j$ is the jth coordinate vector and $\varepsilon > 0$ is a small step size. The estimation of the entire matrix requires N function evaluations, in addition to the evaluation of $\mathbf{F}(\mathbf{u})$.

The derivatives obtained on Eq. (4) are subject to round-off error, which does not occur when they are computed analytically, as is the case with *AD*. In this technique, one sweep of the forward mode of operation is used to calculate a column of $\mathbf{J}(\mathbf{u})$, therefore N sweeps are necessary, in total, to compute the whole matrix.

The difference between the two numerical approaches is only a matter of accuracy. All the same, in both cases sparsity in the derivative matrices can be exploited to compute the nonzero entries efficiently. In the context of finite difference approximations, efficiency corresponds to reducing the number of function evaluations required. In the context of *AD*, it means reducing the total number of *AD* sweeps.

3.1 Graph Coloring Optimization

To introduce this subsection, some useful definitions follow. A group of *structurally orthogonal columns* in a matrix is a group where no two columns have nonzeros in a common row, as columns 1 and 5 of the matrix depicted in Fig. 2. A *partial distance-2 coloring* of a bipartite graph $G_b = (V_1, V_2, E)$ is the coloring of the vertices in V_2, such that any pair of vertices in this set at a distance 2 from each other receive different colors.

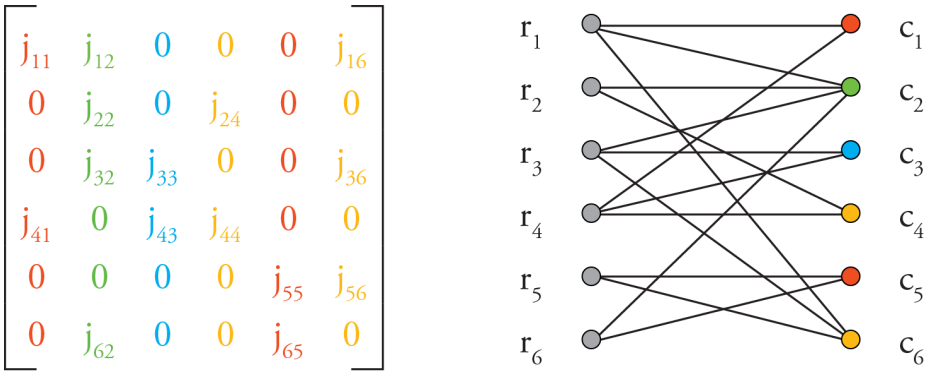


Figure 2: A partitioning of a Jacobian and its representation as a graph coloring.

Curtis, Powell, and Reid (1974) showed that the problem of minimizing the number of function evaluations required to calculate a Jacobian matrix is the problem of partitioning its columns into the fewest groups of structurally orthogonal columns. This partitioning problem was later modeled as a graph coloring problem by Coleman and Moré (1983), to which Gebremedhin et al. (2005) proposed a new graph coloring formulation, declaring it to be superior to the original one. In this work, the latter graph model is applied to solve the matrix partitioning problem.

First, we explain how the referred partitioning is employed to reduce the number of function evaluations when estimating a Jacobian through finite differences. Suppose one has partitioned the columns of a $N \times N$ Jacobian matrix $\mathbf{J}$ into p groups of structurally orthogonal columns, where the partitions are defined by column vectors $\mathbf{d}_j$, $1 \leq j \leq p$, that have 1's in components corresponding to the columns in a partition and zeros in all other components. Consider a "compressed" version of $\mathbf{J}$, a $N \times p$ matrix $\mathbf{J'} = [\mathbf{J}\mathbf{d}_1\ \mathbf{J}\mathbf{d}_2\ ...\ \mathbf{J}\mathbf{d}_p]$, that continues to store all nonzero elements of $\mathbf{J}$. Each column of $\mathbf{J'}$ can be estimated using:

$$\mathbf{J'}\mathbf{d}_j = \partial\mathbf{F}(\mathbf{u})/\partial\mathbf{d}_j \approx [\mathbf{F}(\mathbf{u} + \varepsilon\mathbf{d}_j) - \mathbf{F}(\mathbf{u})]/\varepsilon\ , \quad 1 \leq j \leq p\ , \tag{5}$$

where $\varepsilon > 0$ is a small step size. Eq. (5) allows the nonzeros of all columns in a given partition to be simultaneously determined through one finite difference operation. It is easy to see that, if a good partitioning can be obtained, we can improve the whole evaluation time of the matrix compared to its regular computation. With automatic differentiation, the usage of the partitioning is analogous: only one sweep of the forward mode must be performed to compute a whole group of structurally orthogonal columns in $\mathbf{J}$.

Now we can detail how to obtain a structurally orthogonal partitioning of the columns in a matrix, using the bipartite graph representation proposed by Gebremedhin et al. (2005). The first step is to represent the sparsity structure of a Jacobian as a bipartite graph. Given a $N \times N$ matrix $\mathbf{J}$, the bipartite graph of $\mathbf{J}$ is defined as $G_b(\mathbf{J}) = (V_1, V_2, E)$, where V_1 and V_2 correspond to the rows and columns of $\mathbf{J}$, respectively, and an edge connects row r_i to column c_j whenever j_{ij} is a nonzero. The next step is to color the graph. The partial distance-2 coloring of this bipartite graph provides a partition of the column vertices into groups of structurally orthogonal columns. Fig. 2 exemplifies this matrix-to-graph equivalence.

The optimum coloring of G_b shall provide the minimum number of partitions. As the graph coloring problem is NP-hard (Lin and Skiena, 1995), an heuristic is utilized to perform the coloring, that proved to be practically effective. We must remember the focus in this work is to reduce the evaluation time, and we wish to find - in feasible time - a good structurally orthogonal partitioning to decrease the number of function evaluations from N to p, where p is the number of column partitions. It is not imperative that we achieve the chromatic number of G_b, but an acceptable $p << N$, and the simple greedy heuristic adopted, described in Algorithm 1 (where $N_1(v)$ is the

set of vertices adjacent to vertex v), has provided a good enough coloring for the problem and instances considered here.

Algorithm 1: Partial Distance-2 Coloring (Gebremedhin et al., 2005)

```
1  Input: G_b = (V_1, V_2, E)
2  Initialize forbiddenColors with some value a ∉ V_2
3  foreach v ∈ V_2 do
4      foreach w ∈ N_1(v) do
5          foreach colored vertex x ∈ N_1(w) do
6              forbiddenColors[color[x]] = v
7          end
8      end
9      color[v] = min{c > 0 : forbiddenColors[c] ≠ v}
10 end
```

4 Preconditioned GMRES

The generalized minimal residual (GMRES) method (Saad and Schultz, 1986) is a non-stationary iterative method used to determine the approximate solution of a linear system of the form $Ax = b$, as the system in Eq. (3). Being x_0 the initial solution, $\{v_j\}_{j=1,N}$ vectors in the *Krylov* basis and $y_1, y_2, ..., y_N$ terms of a linear combination, the solution of the system can be represented by:

$$x = x_0 + \sum_{j=1,N} y_j v_j. \tag{6}$$

To reduce de number of floating-point operations and control storage requirements, this work considers a restarted GMRES, also called GMRES(k). This technique, developed by Saad and Schultz (1986), generate new vectors in the *Krylov* basis with k elements for each k iterations. Thus, the new approximation can be represented by:

$$x_{i+k} = x_i + \sum_{j=1,k} y_j v_j. \tag{7}$$

Preconditioning is a good choice for accelerating iterative methods such as GMRES. The main idea is to transform the original system $Ax = b$ into the preconditioned system $M^{-1}Ax = M^{-1}b$, where M is the preconditioner matrix and must be a good approximation of A. The preconditioning technique used in this work is based on approximate factorization of $M = \tilde{L}\tilde{U}$, called incomplete LU factorization and denoted ILU(p), where p is the *level of fill* used to control the number of new elements generated during the process (Saad, 2003).

4.1 Matrix Reordering Optimization

Some important concepts from graph theory and matrix reordering are useful to understand the following algorithm and it is appropriate to state some basic definitions. Being A a matrix structurally symmetric, the *bandwidth* of A can be defined by:

$$bw(A) = \max_{i=1,..,N}\{b_i\} \tag{8}$$

$$b_i = (i - j) \; \forall \; a_{ij} \neq 0, \; i = 1, ..., N. \tag{9}$$

And the *envelope* of A can be defined by:

$$env(A) = \sum_{i=1}^{N} b_i. \tag{10}$$

The *level structure* rooted at a vertex e may be expressed as $L(e) = \{l_1(e), l_2(e), ..., l_h(e)\}$ where h is the total number of levels of a tree rooted on e. Considering the vertices v, w, the *distance* $d(v, w)$ is the length of the shortest path between v and w. The *eccentricity* of a vertex v is the greatest distance between v and any other vertex. The *diameter* of a graph is the maximum eccentricity of any vertex in the graph and the *pseudo-diameter* is a high eccentricity, but not necessarily the largest. Finally, the *pseudo-peripheral vertices* are vertices that achieve the pseudo-diameter.

A sparse matrix reordering scheme with the purpose of minimizing its bandwidth is implemented in this work, as alternative to speed up the convergence of the preconditioned GMRES method. The preconditioning calculations are optimized when we use a reordering, once with the reduced bandwidth, less floating-point operations are made.

For the computational tests we adopted the Sloan reordering, described in Sloan (1986). In preliminary tests, this algorithm reached good solutions for the matrices generated in our specific problem. The method aims to reduce the envelope and the bandwidth of structurally symmetric sparse matrices. The algorithm works assigning priorities and status to each vertex, maintaining a queue that is being updated with informations of the graph. Algorithm 2 describes the steps for labeling a graph with N vertices.

Algorithm 2: Sloan

1. Enter with pseudo-peripheral vertices (Sloan, 1986), initial vertex s and end vertex e.
2. Generate the level structure rooted at the end vertex, $L(e) = l_1, l_2, ..., l_{h(e)}$ and compute the distance δ_i of each vertex i from the end vertex e, using a BFS algorithm.
3. Assign each vertex in the graph G an inactive status and an initial priority P_i according to $P_i \leftarrow W_1 * \delta_i - W_2 * (d_i + 1)$ where W_1 and W_2 are integer weights and d_i is the degree of i.
4. Insert the starting vertex, s, in the queue of eligible vertices and assign it a preactive status.
5. While the queue of eligible vertices is not empty, do steps 6-9.
6. Search the queue of eligible vertices and select the vertex with the highest priority. Let it be i.
7. Delete vertex i from the queue. If i is not preactive, go to step 8. Else, examine each vertex j which is adjacent to i and set $P_j = P_j + W_2$. If vertex j is inactive, then insert it in the priority queue with a preactive status.
8. Label vertex i with its new label and assign to it a postactive status.
9. Examine each vertex j which is adjacent to vertex i. If vertex j is not preactive, take no action. Else, assign vertex j an active status, set $P_j = P_j + W_2$ and examine each vertex k which is adjacent to vertex j. If vertex k is not postactive, increment its priority according to $P_k = P_k + W_2$. If vertex k is inactive, insert it in the priority queue with a preactive status.
10. Exit with the new vertex labels.

5 Test Problems

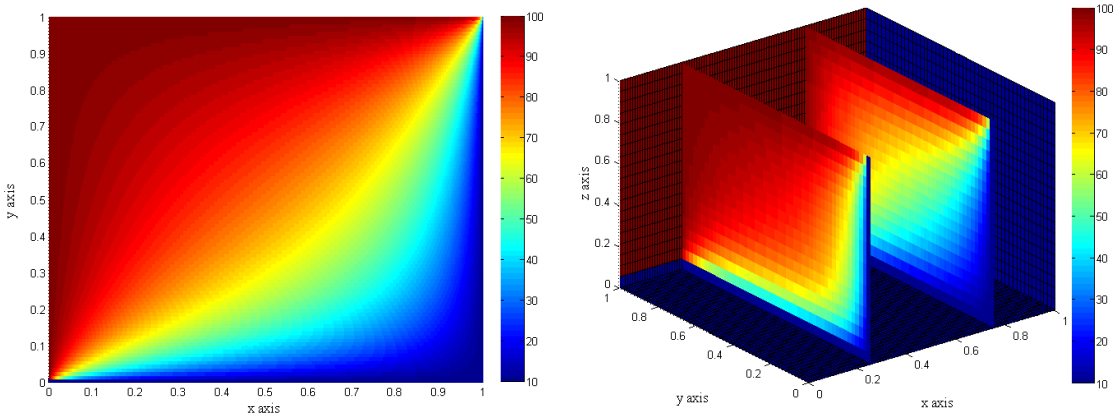
For the numerical experiments conducted in this work, we use the two-dimensional and the three-dimensional variations of the heat transfer problem defined by the following nonlinear differential equation:

$$-\nabla \cdot (K(u)\nabla u) = 0 \quad \text{in} \quad \Omega \tag{11}$$

where u is the temperature and the thermal conductivity is considered as $K(u) = 0.0000002u^2 + 0.00001u + 0.001$. The definitions, for each domain, are:

- for the two-dimensional problem, the boundary conditions are $u(x,0) = u(1,y) = 10$ and $u(x,1) = u(0,y) = 100$, with $\Omega = (0,1) \times (0,1)$ discretized into an uniform grid with $N = n \times m$ unknowns points, respectively, in the x, y directions.
- for the three-dimensional problem, the boundary conditions are $u(x,y,0) = u(x,0,z) = u(1,y,z) = 10$ and $u(x,y,1) = u(x,1,z) = u(0,y,z) = 100$, with $\Omega = (0,1) \times (0,1) \times (0,1)$ discretized into an uniform grid with $N = n \times m \times l$ unknowns points, respectively, in the x, y, z directions.

We approximate the derivatives by combining forward, backward and centered finite differences, arriving to the nonlinear system of equations $\mathbf{F}(\mathbf{u}) = 0$, where $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$ is a nonlinear vector function, $\mathbf{u} = (u_1, u_2, \ldots, u_N)^T$ is the unknown vector. For the two-dimensional case, each component of $\mathbf{F}$ depends only on the five unknowns u_{I-n}, u_{I-1}, u_I, u_{I+1}, u_{I+n} for $I = 1, 2, \ldots, N$ and the resulting matrix has a pentadiagonal structure. For the three-dimensional case, each component of $\mathbf{F}$ depends only on the seven unknowns u_{I-m*n}, u_{I-n}, u_{I-1}, u_I, u_{I+1}, u_{I+n}, u_{I+m*n} for $I = 1, 2, \ldots, N$ and the resulting matrix has a heptadiagonal structure. Fig. 3 shows the temperature distribution for a mesh with 100.000 unknowns.



(a) 2D problem solution - 1000x100 mesh

(b) 3D problem solution - 100x50x20 mesh

Figure 3: Temperature distribution for a 100.000 mesh.

6 Experimental Results

For the heat transfer problem described in Section 5, we developed programs in the C language (compiled with GCC 4.6.3). All computational tests were performed on an Intel Core i5-2400 3.10GHz $\times$ 4 machine with 4GB of RAM under Ubuntu 12.04 operating system. To store the sparse matrices derived from the problem we use an optimized storage scheme called Compressed Sparse Row (*CSR*), that places the subsequent nonzeros of the matrix rows in contiguous memory locations.

In the following subsections, eight problem instances are considered for each two and three-dimensional domains. The results in Tables 1 and 3 show, for each instance I, the size N of the problem, the p parameter to the ILU(p) preconditioner and the CPU time, in seconds, for computing the solution with no optimization (*NO*), with optimized Jacobian evaluation (*JE*), with matrix reordering (*MR*) and with both optimizations (*OP*). The CPU times for each optimized case (*JE, MR, OP*) are followed by the percentage time reduction (red) from time *NO*, given by $\text{red} = 1 - \text{time}([JE|MR|OP])/\text{time}(NO)$.

In the GMRES algorithm, the number of vectors in the Krylov basis is 100 and its stopping tolerance is 1×10^{-7} for all instances. The step size ε for the finite difference operations is 1×10^{-9}, and the tolerance in the Newton's method is $\tau_{res} = 10^{-6}$.

6.1 Two-dimensional problem

For all instances, the coloring number achieved for the graph associated with the problem matrix was 6, which means that the optimized Jacobian evaluation performed 6 finite difference operations. The experimental results and analysis for the two-dimensional heat transfer problem are shown in Tables 1 and 2. Fig. 4 shows the percentage reduction for each problem size.

				NO	*JE*		*MR*		*OP*	
I	N	$(n \times m)$	p	time(sec)	time(sec)	red(%)	time(sec)	red(%)	time(sec)	red(%)
1	10.000	(200×50)	5	13,0	2,1	84,2	12,6	2,7	1,7	86,9
2	10.000	(200×50)	10	13,8	3,0	78,5	12,8	7,2	1,9	86,6
3	50.000	(500×100)	5	324,7	42,1	87,0	316,4	2,5	36,2	88,8
4	50.000	(500×100)	10	327,3	44,6	86,4	317,4	3,0	36,3	88,9
5	100.000	(1000×100)	5	1.294,4	159,0	87,7	1.263,7	2,4	137,9	89,4
6	100.000	(1000×100)	10	1.296,9	165,0	87,3	1.262,9	2,6	138,2	89,3
7	300.000	(1500×200)	5	11.625,8	1.342,8	88,4	11.447,8	1,5	1.221,5	89,5
8	300.000	(1500×200)	10	11.645,8	1.337,1	88,5	11.447,2	1,7	1.213,4	89,6

Table 1: Set of chosen parameters and computational results for the two-dimensional problem.

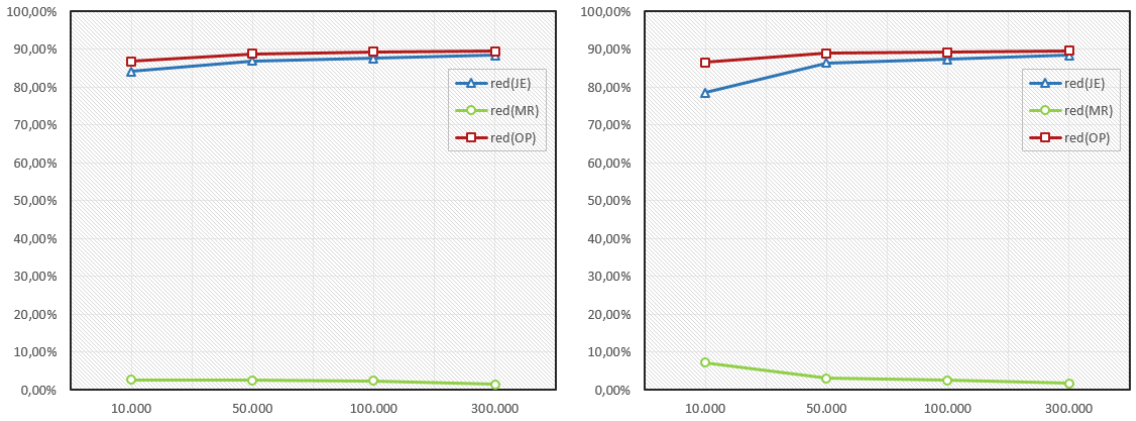
Analyzing the results in Table 1, we first observe that the combination of the Jacobian optimization and the matrix reordering scheme achieved a reduction of over 86%, for all instances considered. Moreover, the biggest contribution (by a large margin) came from the Jacobian optimization, as can be seen in Fig. 4. If we focus on the level of fill-in, $p = 5$ and $p = 10$, for each problem size its variation did not affect significantly the computation time, although the ILU(10) preconditioner produced a slightly larger reduction than the ILU(5) (see green lines on Fig. 4). However, as seen in Table 1, the ILU(10) spent a higher overall CPU time for almost all instances.

The impact of the reordering strategy in the total computation time was small for both preconditioners, although the number of GMRES iterations and the total time consumed by them do decrease when we use the reordering scheme (see *NO* and *MR* for each ILU(p) in Table 2). The overall time for solving the linear system, on the other hand, does not change much, even though the bandwidth is reduced. This happens because the time to calculate the preconditioner remains

		T_J (sec)	T_P (sec)	T_G (sec)	GMRES iter.
ILU(5)	NO	10.286,8	1.186,1	152,9	1135
	JE	0,4	1.187,0	155,2	1135
	MR	10.224,4	1.183,5	36,7	363
	OP	0,3	1.180,4	37,1	363
ILU(10)	NO	10.295,4	1.226,4	123,9	719
	JE	0,4	1.211,6	124,8	719
	MR	10.234,9	1.192,3	16,8	191
	OP	0,4	1.192,5	16,9	191

Table 2: Detailed information for each execution case (*NO, JE, MR* and *OP*) of the instance with size 300.000, displaying the total time consumed in Jacobian evaluations (T_J), preconditioner calculations (T_P), GMRES solver (T_G), and the total number of GMRES iterations.

high, showing that the operations required to obtain the L and U factors prevail over the time spent on the GMRES iterations.



(a) ILU(5) **(b)** ILU(10)

Figure 4: Percentage time reduction for each problem size.

6.2 Three-dimensional problem

For all instances of this problem case, the coloring number achieved was 13. The experimental results and analysis for the three-dimensional heat transfer problem are shown in Tables 3 and 4. Fig. 5 shows the percentage reduction for each problem size.

For all instances, the optimization techniques were able to reduce the final computation time in over 78%. Examining Table 3, we observe a higher level of fill-in ($p = 10$) induces a higher contribution of the reordering strategy on the final time reduction, while on the lower level ($p = 5$, which has better final runtime) the graph coloring strategy is responsible for the biggest improvement. Furthermore, by increasing the dimension of the problem but maintaining the same level of fill-in, the reduction achieved by the Jacobian optimization increases while the reduction achieved by the reordering decreases. Those behaviors can be clearly observed in Fig. 5.

While in the two-dimensional problem the matrix reordering did not contribute significantly on the time reduction, for the three-dimensional case the Sloan algorithm proved to be effective. Table 4 shows that the time for both preconditioner computation and GMRES iterations notably decreased from *NO* to *MR*. The improvement accomplished by each optimization technique varies, depending whether the Jacobian evaluation or the preconditioned GMRES demands the biggest computational

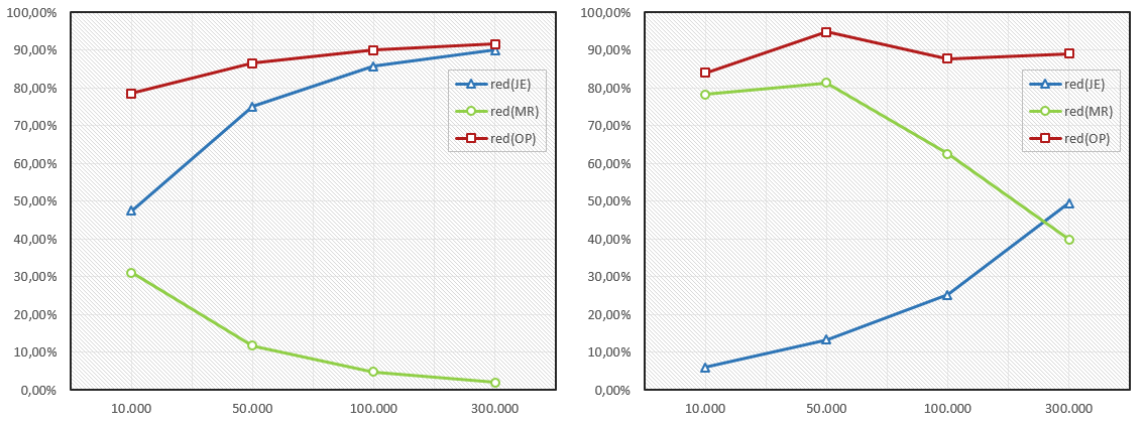
				NO	JE		MR		OP	
I	N	$(n \times m \times l)$	p	time(sec)	time(sec)	red(%)	time(sec)	red(%)	time(sec)	red(%)
1	10.000	(100×10×10)	5	23,8	12,5	47,5	16,4	31,1	5,1	78,6
2	10.000	(100×10×10)	10	196,7	184,9	6,0	42,7	78,3	31,3	84,1
3	50.000	(200×50×5)	5	340,6	84,8	75,1	300,7	11,7	45,6	86,6
4	50.000	(200×50×5)	10	1.911,9	1.655,9	13,4	354,9	81,4	99,7	94,8
5	100.000	(100×50×20)	5	1.784,0	255,4	85,7	1.698,8	4,8	179,3	90,0
6	100.000	(100×50×20)	10	6.027,5	4.508,2	25,2	2.255,8	62,6	733,5	87,8
7	300.000	(200×50×30)	5	15.991,3	1.592,7	90,0	15.679,1	2,0	1.349,5	91,6
8	300.000	(200×50×30)	10	29.148,3	14.712,9	49,5	17.522,7	39,9	3.184,2	89,1

Table 3: Set of chosen parameters and computational results for the three-dimensional problem.

		T_J (sec)	T_P (sec)	T_G (sec)	GMRES iter.
ILU(5)	NO	14.405,1	1.535,5	50,7	182
	JE	1,0	1.540,1	51,1	182
	MR	14.328,3	1.303,6	18,5	103
	OP	0,9	1.301,0	18,1	103
ILU(10)	NO	14.412,6	14.661,4	74,1	117
	JE	1,0	14.637,0	74,3	117
	MR	14.335,4	3.134,6	23,7	57
	OP	0,9	3.130,3	23,3	57

Table 4: Detailed information for each execution case (*NO, JE, MR* and *OP*) of the instance with size 300.000, displaying the total time consumed in Jacobian evaluations (T_J), preconditioner calculations (T_P), GMRES solver (T_G), and the total number of GMRES iterations.

effort. Nevertheless, the final time reduction is roughly the same, showing the strength in the use of the optimization strategies.



(a) ILU(5) **(b)** ILU(10)

Figure 5: Percentage time reduction for each problem size.

7 Conclusions and Future Work

The preliminary results obtained in this work show that the optimization strategies imposed on the Jacobian evaluation and on GMRES solver significantly reduced the final execution time for the

benchmark nonlinear heat transfer problem. Comparing the two-dimensional and three-dimensional problems, the former took less time overall, as the problem matrix associated is sparser and less operations are performed. Also, the computational gains were definitely better in the three-dimensional case. The 2D problem, for its simpler structure, has less to benefit from the reordering scheme - with ILU(5) and ILU(10) - than the 3D problem, that has a more complicated structure, with two more diagonals of nonzero elements.

Regarding the optimization techniques, the coloring scheme influences the task of evaluating the Jacobian matrix, by reducing the number of required finite difference operations, whereas the Sloan algorithm utilized for the matrix reordering has effects in the preconditioner calculations and in the convergence time of the GMRES method. The bandwidth in both problem cases was always significantly reduced, but only on the three-dimensional case there was an actual impact on the overall computation time. This strongly suggests the matrix reordering optimization has a better use for problems where the matrix has a non-trivial sparsity pattern.

There is further work to be done. We aim to investigate the following directions: (i) perform computational tests considering problems of a higher complexity, whose resulting matrices have non-trivial sparsity patterns; (ii) implement an algorithm that finds the minimum structurally orthogonal partitioning; (iii) apply automatic differentiation as an alternative to the finite differences method for the Jacobian evaluation; (iv) calibrate the p parameter to the ILU(p) in order to improve the matrix reordering impact on the CPU time consumed by the GMRES solver and (v) implement different matrix reordering schemes.

Acknowledgement

This work was partly supported by FAPES n^o 48511579/2009, CNPq 552630/2011-0 and CNPq 307020/2012-6.

References

S. T. Barnard, A. Pothen, and H. D. Simon. A spectral algorithm for envelope reduction of sparse matrices. In *Proceedings of the 1993 ACM/IEEE conference on Supercomputing*, Supercomputing '93, pages 493–502, New York, NY, USA, 1993. ACM. ISBN 0-8186-4340-4. doi: 10.1145/169627.169790.

N.L.O. Bodart, L. Catabriga, and A.L.G.A. Coutinho. Forcing term effects on Inexact Newton-Krylov method for solving nonlinear equations emanating from the SUPG/PSPG finite element formulation of Navier-Stokes equations. In *32th Iberian Latin American Congress on Computational Methods in Engineering*, Ouro Preto, 2011.

Thomas F. Coleman and Jorge J. Moré. Estimation of sparse Jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209, 1983.

A. R. Curtis, M. J. D. Powell, and J. K. Reid. On the estimation of sparse jacobian matrices. *IMA J Appl Math*, 13 (1):117–119, 1974.

E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, ACM '69, pages 157–172, New York, NY, USA, 1969. ACM. doi: 10.1145/800195.805928.

Timothy A. Davis, Patrick Amestoy, Iain S. Duff, Iain, and S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17:886–905, 1994.

R.N. Elias, A.L.G.A. Coutinho, and M.A.D.A. Martins. Inexact-Newton-type methods for non-linear problems arising from the SUPG/PSPG solution of steady incompressible Navier-Stokes equations. *J. Braz. Soc. Mech. Sci. & Eng.*, 26:330–339, 2004.

Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your jacobian? graph coloring for computing derivatives. *SIAM REV*, 47:629–705, 2005.

Alan George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10(2):345–363, 1973. doi: 10.2307/2156361.

K. Ghidetti, M.C.S. Boeres, and L. Catabriga. Um estudo comparativo de métodos heurísticos para reordenamento de matrizes esparsas. *XLII Simpósio Brasileiro de Pesquisa Operacional*, 1: 2066–207, 2010a.

K. Ghidetti, L. Catabriga, M.C.S. Boeres, and M.C. Rangel. A study of the influence of sparse matrices reordering algorithms on krylov-type preconditioned iterative methods. *XXXI Ibero Latin American Congress on Computational Methods in Engineering*, XXIX:2323–2343, 2010b.

K. Ghidetti, Lucia Catabriga, M.C.S. Boeres, and M.C. Rangel. A study of the influence of sparse matrices reordering algorithms for ILU(p) preconditioner on the GMRES method. *Fifth SIAM Workshop on Combinatorial Scientific Computing*, 1:125–127, 2011.

N. E. Gibbs, W. G Poole, and P. K. Stockmeyer. An algorithm for reducing the profile and bandwidth of a sparse matrix. *SIAM J. Numer. Anal.*, 13:236–250, 1976.

D. Goldfarb and Ph. L. Toint. Optimal estimation of jacobian and hessian matrices that arise in finite difference calculations. *Mathematics of Computation*, 43:69–88, 1984.

Andreas Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Number 19 in Frontiers in Appl. Math. SIAM, Philadelphia, PA, 2000. ISBN 0-89871-451-6.

C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Number 16 in Frontiers in Applied Mathematics. SIAM, 1995.

Yaw-Ling Lin and Steven Skiena. Algorithms for square roots of graphs. *SIAM J. Discrete Math.*, 8(1):99–118, 1995.

Brenno Lugon and Lucia Catabriga. Algoritmos de reordenamento de matrizes esparsas aplicados a precondicionadores ILU(p). *Simpósio Brasileiro de Pesquisa Operacional (SBPO)*, 45:2343–2355, 2013.

Michael Lülfesmann and Ken-ichi Kawarabayashi. Sub-exponential graph coloring algorithm for stencil-based jacobian computations. *J. Comput. Science*, 5(1):1–11, 2014.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003.

Youcef Saad and Martin H Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, July 1986. ISSN 0196-5204. doi: 10.1137/0907058.

S. W. Sloan. An algorithm for profile and wavefront reduction of sparse matrices. *Internacional Journal for Numerical Methods in Engineering*, 23:239–251, 1986.