Università di Pisa
Dipartimento di Informatica

Technical Report: TR-01-02

Heuristic Spectral Techniques for the Reduction of Bandwidth and Work-Bound of Sparse Matrices

Gianna M. Del Corso   Francesco Romani

January 11, 2001

ADDRESS: Corso Italia 40, 56125 Pisa, Italy.   TEL: +39 050 887111   FAX: +39 050 887226

Heuristic Spectral Techniques for the Reduction of Bandwidth and Work-Bound of Sparse Matrices

Gianna M. Del Corso Francesco Romani*

January 11, 2001

Abstract

In this paper we present heuristic techniques for the reduction of the bandwidth of a sparse matrix as well as for the reduction of the cost of the associated Cholesky factorization. Our algorithms are inspired by the Spectral Method of Barnard, Pothen and Simon [3] which derives a permutation for reducing the envelope-size of a sparse matrix by computing the second eigenvector of the associated Laplacian matrix. Two main modifications of that method are proposed and tested. The first is based on the experimental observation that it is often preferable to perform only few iterations of an iterative method converging to the second eigenvector; the second is the introduction of a weighted Laplacian. These simple ideas allows us to obtain a family of spectral methods that have been carefully tested on a set of matrices whose size range from few hundred to one-million.

1 Introduction

The problem of reordering a sparse symmetric matrix to reduce its bandwidth has always played a central role in the solution of sparse linear systems. The reduction of the bandwidth allows to attenuate the phenomenon of the fill-in and yields advantages both in memory requirements and computational time. Two other important parameters of sparse matrices are the envelope-size that is the sum of the row-widths and the work-bound that measures the cost of the Cholesky factorization when an envelope storage scheme is applied.

Recently the reordering of sparse matrices has found application in several other contexts: for example, as a preprocessing for computing preconditioners based on incomplete factorizations [7][8], in parallel sparse matrix manipulation [1], and in information retrieval [4].

Many heuristic techniques have been proposed for reducing these quantities. However, those algorithms were designed to reduce only the bandwidth or the envelope-size. In this paper we propose a class of algorithms based on the Spectral Method (recently proposed by Barnard, Pothen and Simon [3]) which derives a reordering of the matrix by computing an eigenvector of a discrete Laplacian matrix associated with the given matrix. This method is therefore deeply different from the other reordering schemes such as the reverse Cuthill-McKee (`RCM` [5]), Gibbs-Pole-Stockmayer (`GPS`[13][14]), Gibbs-King (`GK` [18]) and the more recent `WBRA` [9][10]. The Spectral Method is especially designed to reduce the work-bound of the matrix and the bandwidth of the reordered matrix can be rather

*Dipartimento di Informatica, Università di Pisa, Corso Italia 40, 56100 Pisa, Italy. E-mail:{delcorso, romani}@di.unipi.it

high. The modifications we propose to the Spectral Method aim to produce a reordered matrix with a bandwidth and work-bound "sufficiently" close to the values achieved by heuristic techniques designed to reduce only one of these parameters.

We propose and test two main ideas. The first one is based on the observation that it does not always pay to compute with high precision the eigenvector required by the Spectral Method. Experimentally we have seen that we get a smaller bandwidth performing only few iterations of an iterative method converging to that eigenvector. The second idea is that, once we have reordered the matrix, we can apply again the Spectral Method to the reordered matrix with a *modified Laplacian* which penalizes the entries far away from the main diagonal. These two simple ideas are combined to get an iterative scheme in which at each step we first update the Laplacian and then we compute few iterations for approximating the second eigenvector of the modified Laplacian. We devised several strategies for combining the updating of the Laplacian and the computation of the eigenvector. Such strategies lead to a family of spectral methods that we have been extensively tested on the Sparse Matrix Collection [6] of matrices whose size ranges from few hundred to one-million.

The paper is organized as follows. In Section 2 we formally define the problem and the various measures associated with the bandwidth. In Section 3 we describe the main features of the Spectral Method. In Section 4 we describe the Modified Spectral Methods and we discuss the alternatives for the choices of the weighted Laplacian, the iteration method and the stopping criteria. Time and space complexity are studied as well. Section 5 presents the set of matrices and the performance measures used in the tests. In Section 6 the results of the numerical experiments are presented and discussed.

2 Definition of the Problem

Let $A = (a_{ij})$ be a sparse symmetric $n \times n$ matrix with $2k$ non-diagonal nonzeros. We briefly define different parameters associated with the matrix. Let $\mathrm{row}(i)$ be the column indexes of the nonzero entries in the lower triangular part of the i-th row, that is $\mathrm{row}(i) = \{j : a_{ij} \neq 0, 1 \leq j \leq i\}$. Denote by r_i the row-width of the i-th row of A that is

$$r_i = \max\{(i-j) : j \in \mathrm{row}(i)\}.$$

The *bandwidth* and the *envelope* of A are defined as

$$\begin{aligned} bw(A) &= \max\{r_i(A) : i = 1, 2, \ldots, n\}, \\ Env(A) &= \{(i,j) : i - r_i \leq j < i, i = 1, \ldots, n\}. \end{aligned} \tag{1}$$

The size of the envelope is denoted by $Esize(A) = |Env(A)| = \sum_{i=1}^{n} r_i$ and is the sum of the width of every row of A. The cost of solving a linear system (see [17]) by using the Cholesky factorization method based on a envelope storage scheme can be bounded from above by $(1/2)\sum_{i=2}^{n} r_i(r_i + 3)$. We associate to A the quantity $wb(A) = \sum_{i=1}^{n} r_i^2$ that bounds the *work* in such a factorization.

Bandwidth, envelope-size and work-bound depend on the ordering of rows and columns of A and, from a computational point of view, it is often profitable to permute the rows and columns of A in order to reduce the values of these parameters. In general, the bandwidth, the envelope-size and the work-bound are not minimized by the same permutation, but it is obvious that a matrix with a small bandwidth has also a small envelope-size and work-bound. For this reason it is interesting to find a permutation that induces an ordering in the rows and columns of A which leads to "good" values of bandwidth, work-bound and envelope-size.

Some other problems are related to the bandwidth and envelope minimization problem. Let $p, 1 \leq p < \infty$ be a real number, we define the p-sum as

$$\sigma_p(A) = \left(\sum_{i=1}^{n} \sum_{j \in \text{row}(i)} (i-j)^p \right)^{1/p}.$$

Note that $Esize(A)$ and $wb(A)$ can be rewritten as

$$\begin{aligned} Esize(A) &= \sum_{i=1}^{n} \max_{j \in \text{row}(i)} (i-j) \\ wb(A) &= \sum_{i=1}^{n} \max_{j \in \text{row}(i)} (i-j)^2 \end{aligned} \tag{2}$$

showing their similarity with

$$\sigma_1(A) = \sum_{i=1}^{n} \sum_{j \in \text{row}(i)} (i-j)$$

and

$$\sigma_2^2 = \sum_{i=1}^{n} \sum_{j \in \text{row}(i)} (i-j)^2, \tag{3}$$

respectively. In [12] the authors prove some relations between σ_1, σ_2 and *Esize* and *wb* showing that a permutation that minimizes σ_1 or σ_2 produce a reordered matrix with *Esize* or *wb* close to the minimum.

Minimizing the bandwidth, the envelope-size, the 1-sum, the 2-sum are known to be NP-complete problems [12][19] and minimizing the work-bound is likely to be intractable as well. For this reason usually one has to settle for heuristics to reduce these quantities.

3 The Spectral Method

Recently, Barnard, Pothen and Simon [3] proposed the *Spectral Method*, i.e. an heuristic procedure for reducing the envelope-size of a sparse symmetric irreducible matrix. The method is based on spectral proprieties of the Laplacian matrix $L(A) = (l_{ij})$ that can be associated to the $n \times n$ matrix A as follows:

$$l_{ij} = \begin{cases} -1 & \text{if } i \neq j \text{ and } a_{ij} \neq 0 \\ 0 & \text{if } i \neq j \text{ and } a_{ij} = 0 \\ -\sum_{\substack{j=1 \\ j \neq i}}^{n} l_{ij} & \text{if } i = j. \end{cases} \qquad i, j = 1, 2, \ldots, n \tag{4}$$

The matrix $L(A)$ is an M-matrix, hence its eigenvalues $\lambda_1 \leq \lambda_2 \leq \cdots \lambda_n$ are non-negative. Moreover, if A is irreducible, then $L(A)$ has an unique zero eigenvalue $\lambda_1 = 0$ and the corresponding eigenvector is the constant vector $\mathbf{c} = (c, c, \ldots, c)^T$. The importance of the second eigenvalue λ_2 and its eigenvector $\mathbf{x}_2$ for the solution of partitioning and ordering problems has been noted since the 70's [11]. The second eigenvalue of the Laplacian matrix has then been used to establish upper and lower bounds on the p-sum [15] as well as to reduce bandwidth, 1 and 2-sum [16].

Figure 1: The Spectral Method returns a permutation induced by a second eigenvector of the Laplacian matrix. The matrix reordered accordingly with this vector has usually a smaller value of wb than the original matrix. The matrix J is the permutation matrix with ones on the antidiagonal.

`Spectral Method`

1. Construct the Laplacian matrix $L(A)$.
2. Compute the second eigenvector $\mathbf{x}_2$ of $L(A)$.
3. Construct the permutation vector $\mathbf{p}$ induced by $\mathbf{x}_2$
4. Reorder matrix A accordingly with $\mathbf{p}$ and $\mathbf{p}_R = J\mathbf{p}$,
   let wb_1 and wb_2 be the work-bound obtained.
5. Return $\mathbf{p}$ if $wb_1 \leq wb_2$, return $\mathbf{p}_R$ otherwise.

These considerations motivate the Spectral Method reported in the algorithm in Figure 1. A theoretical justification of this method is given in [3] and [12] where it is shown that the second Laplacian eigenvector $\mathbf{x}_2$ solves a continuous relaxation of the 2-sum minimization problem. Namely, $\mathbf{x}_2$ minimizes the quadratic form

$$\frac{1}{2\|\mathbf{y}\|_2} \min_{\mathbf{y}\mathbf{u}^T=0} \left(\sum_{i=1}^{n} \sum_{j \in \mathrm{row}(i)} (y_i - y_j)^2 \right). \tag{5}$$

Moreover, it is possible to show that the permutation vector $\mathbf{p}$ induced by $\mathbf{x}_2$, by the rule $p_i \leq p_j$ if and only if $x_i \leq x_j$, is a closest (in the 2-norm) permutation vector to $\mathbf{x}_2$.

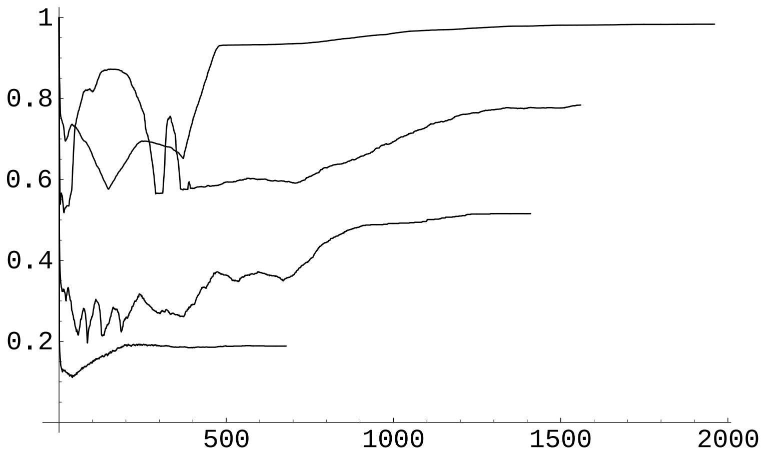
It is important to observe that the authors used the algorithm in Figure 1 as a method for reducing *Esize*, so at step 5 the checking condition was done on the envelope-size rather than on the work-bound. However, by comparing (5) with (2) and (3), it is more natural to see the Spectral Method as an algorithm for reducing wb. For this reason, we are going to compare the methods proposed in this paper with the values of wb returned by the Spectral Method rather than with the envelope-size.

Remark. In the algorithm in Figure 1 both the permutation induced by $\mathbf{x}_2$ and $-\mathbf{x}_2$ are tested since one can get different values of wb (as well as *Esize*) reordering the initial matrix accordingly with the two induced permutations. In fact, in general $wb(A)$ differs from $wb(JAJ)$.

As it is fairly reported in [3], because of the computation of the second eigenvector in step 2, the Spectral method requires much more time than other heuristics for reducing the bandwidth as `RCM`, `GK`, `GPS` methods. One of the less expensive methods for the computation of few eigenvalues is the Lanczos algorithm, eventually with selective orthogonalization against the constant vector [20]. The drawback is that, when the matrix is large and ill-conditioned, Lanczos method can be prohibitive for the quantity of memory it requires. In fact, since we are looking for an eigenvector, after each Lanczos's step we need to record a vector of size n, so that if s steps are performed we need to store ns real entries. Moreover, in ill-conditioned cases [20], $s = O(n)$.

Barnard and Simon [2] proposed a multilevel recursive method for the computation of the second eigenvalue of the Laplacian matrix. That method has some similarity with the multigrid methods. In fact, it recursively constructs a series of successively smaller matrices each of which approximates its larger predecessor. When sufficiently small dimension is reached, the second eigenvector is computed

Figure 2: Evolution of the bandwidth of 4 different matrices. The permutations induced by the vectors produced after each step of the power method for the computation of the second eigenvector are computed. Then, the matrices are reordered in accordance with that permutations and the corresponding bandwidth are computed. An automatic termination criterion has been used to stop the iterative method. On x axis is the number of iterations, on y axes is the bandwidth scaled by the initial bandwidth value.



by the Lanczos method, and the vector is interpolated into the next higher level, providing an approximation for the second eigenvector of the matrix at a finer level. The interpolated vector is then refined using the Rayleigh Quotient Iteration algorithm [20] to get a more accurate solution.

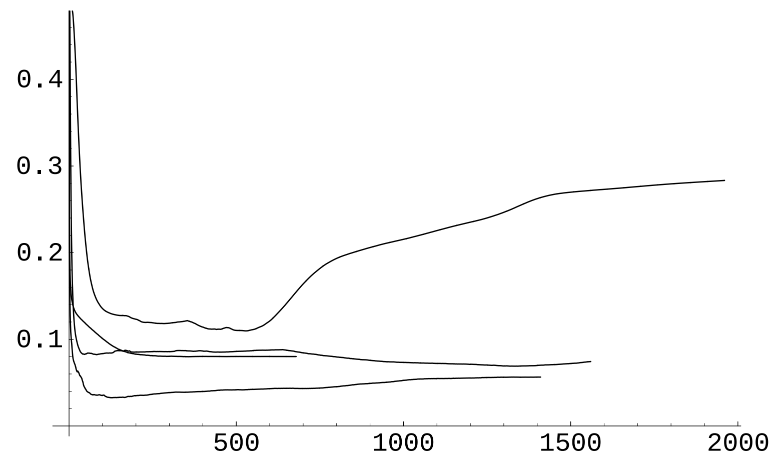
4 Modified Spectral Methods

The Spectral Method described in the previous section is an intriguing algorithm since it allows to relate parameters as bandwidth, envelope-size and work-bound with spectral properties of the Laplacian matrix. However, it has some drawbacks. In fact, the computation of an eigenvector requires lot of time, and moreover, the method has been devised to reduce the envelope-size or work-bound rather than the more interesting bandwidth. In this section we consider some modifications of the Spectral Method that somehow take into account the above considerations.

Some preliminary numerical experiments on the Spectral Method have shown two facts which gave us some ideas on how modify the original Spectral Method to obtain a reorder matrix with a smaller bandwidth and a good work-bound. In particular, we observed that

- Often it is preferable to perform only a few steps of the method for computing the second eigenvector rather than to exactly evaluate it. An example of this behaviour is shown in Figure 2 and 3 were is described the evolution of the induced bandwidth and wb during the computation of the second eigenvector with the power method. In particular, we note that the minimum of the bandwidth as well as the minimum of the wb is achieved reordering the matrix in accordance with a vector obtained before reaching convergence.

Figure 3: Evolution of the of the induced work-bound during the computation of the second eigenvector with the power method with an automatic stopping criterion. The results of the same 4 matrices of Figure 2 are presented. On x axis is the number of iterations, on y axes is the work-bound divided by the initial work-bound value.



- Often, in the matrix obtained by the Spectral Method few nonzero entries far away from the main diagonal give a little contribution to the wb but heavily influence the bandwidth.

These observations suggested us a class of algorithms which often leads to reordered matrices with lower bandwidth and a work-bound sufficiently close to that returned by the Spectral Method. The novelty of our approach consists in introducing a weighted Laplacian with larger weights for the outer nonzero entries and in performing only a few iterations of a method for the computation of the second eigenvector. The Modified Spectral Method (the algorithm in Figure 4) consists of a repetition of the core routine `Cycle` that computes the modified Laplacian, applies few steps of an iterative method for the approximation of the second eigenvector and reorders the matrix accordingly with the induced permutation. The values of $bw(A)$ and $wb(A)$ are computed and are used to check whether a termination condition is verified. An auxiliary integer vector $\mathbf{q}$ retains the permutation of the original matrix A which achieved the lower bandwidth value during the process. This vector represents the output of the Modified Spectral Method. Note that if one is interested in reducing wb instead of bw it has only to slightly change step 7 of `Cycle`.

This general scheme can produce different algorithms according to the various ways we define the weighted Laplacian $WLap(A)$, the function `DoSteps` and the termination condition. Let us discuss in more details the possibilities that we have analyzed.

Figure 4: The Modified Spectral Algorithm consists of the repetition of the procedure `Cycle` which first updates the Laplacian and then applies the procedure `DoSteps`.

`Program MSM`

1. Input matrix A.
2. $\mathbf{e} \leftarrow (1, 2, \ldots, n)^T$
3. $\mathbf{q} \leftarrow \mathbf{e}$
4. Repeat the procedure `Cycle` until a termination condition is satisfied.
5. Return $\mathbf{q}$.

`Procedure Cycle`

1. Choose the starting vector $\mathbf{v}_0 \leftarrow \left(\mathbf{e} - \frac{\mathbf{e}^T\mathbf{u}}{\mathbf{u}^T\mathbf{u}}\mathbf{u}\right)$.
2. Construct a weighted Laplacian matrix $WLap(A)$.
3. $\mathbf{v} = \mathtt{DoSteps}(\mathbf{v}_0)$, where the function `DoSteps` implements few steps of an iterative method for the approximation of the second eigenvector of the matrix $WLap(A)$.
4. Construct the corresponding permutation vector $\mathbf{p}$ induced by $\mathbf{v}$.
5. Reorder matrix A accordingly with $\mathbf{p}$.
6. Compute the values of $bw(A)$ and of $wb(A)$.
7. If the $bw(A)$ is the best value achieved till now then reorder vector $\mathbf{q}$ accordingly with $\mathbf{p}$ or $\mathbf{p}_R$.

4.1 Choice of $WLap(A)$

We considered weight matrices in which the values of non diagonal entries are simple functions of the distance from the main diagonal, namely $WLap(A) = (l_{ij})$, where

$$l_{ij} = \begin{cases} -f(|i-j|) & \text{if } i \neq j \text{ and } a_{ij} \neq 0 \\ 0 & \text{if } i \neq j \text{ and } a_{ij} = 0 \\ -\sum_{\substack{j=1 \\ j \neq i}}^{n} l_{ij} & \text{if } i = j, \end{cases} \tag{6}$$

and $f(t)$ is a non decreasing function that can be chosen in different ways. With this modification the associated quadratic form becomes

$$\frac{1}{2\,\|\mathbf{y}\|_2} \min_{\mathbf{y}\mathbf{u}^T=0} \left(\sum_{i=1}^{n} \sum_{j \in \mathrm{row}(i)} f(|i-j|)\,(y_i - y_j)^2 \right). \tag{7}$$

Note the similarity of this quadratic form with the one minimized by the Spectral Method in (5). Since $f(t)$ is a non decreasing function, the minimization of (7) over all the possible permutations will penalize the permutations which lead to reordered matrices with an high bandwidth. The choice of the function $f(t)$ is a crucial point since the weight on the elements should be heavier on the entries far away from the main diagonal to enforce these entries to be posed at smaller distance from the diagonal. On the other hand, the weight function should not grow to much because this can produce a bad reordering of the entries that before the reordering were in a "good" position.

In our tests we used two different types of weights.

$\mathtt{WLap1}_\alpha$ Continuous weight,

$$f(t) = t^\alpha. \tag{8}$$

$\mathtt{WLap2}_\alpha$ Step-wise weight,

$$f(t) = \begin{cases} -1 & \text{if } t > 0.9\, bw(A) \\ -\alpha & \text{if } t \leq 0.9\, bw(A). \end{cases} \tag{9}$$

Note that $\mathtt{WLap1}_\alpha$ and $\mathtt{WLap2}_\alpha$ reduce to the Laplacian of the Spectral Method $L(A)$ by setting $\alpha = 0$.

The keynote of our method is in the combination of the introduction of the weighted Laplacian and in the performance of only few steps of the iterative process for the approximation of the second eigenvector. In fact, as numerical test show, computing the second eigenvector of $WLap(A)$ and reordering the matrix accordingly, does not always leads to better values of bandwidth and workbound.

Remark. The function used in $\mathtt{WLap1}_\alpha$ does not depend on the matrix A, and can be precomputed for the n possible values of the argument at the expenses of an additional vector of size n for storing these values.

4.2 Choice of the iterative method

Let us now discuss the second ingredient of our methods. Since our primary goal is to apply our bandwidth reduction algorithm also to very large matrices, we need algorithms with low memory requirements. Let us briefly review two possibilities.

Figure 5: The function `DoStepsPower` implements $2\,nstep$ iterations of the power method.

`Function DoStepsPower`(x)

1. $\mu \leftarrow \| WLap(A)\|\infty$.
2. Repeat $nstep$ times.
3. $\quad\mathbf{y} \leftarrow (\mu I - WLap(A))\mathbf{x}$.
4. $\quad\mathbf{x} \leftarrow (\mu I - WLap(A))\mathbf{y}$.
5. $\quad\mathbf{x} \leftarrow \left(\mathbf{x} - \frac{\mathbf{x}^T\mathbf{u}}{\mathbf{u}^T\mathbf{u}}\mathbf{u}\right)$.
6. Return $\mathbf{x}/\|\mathbf{x}\|_\infty$.

Figure 6: The function `DoStepsLanczos` implements $nstep$ iterations of the Lanczos method.

`Function DoStepsLanczos`(x)

1. Let $\mathbf{v}^{(1)} = \mathbf{x}$.
2. Compute $\mathbf{v}^{(i)}$ for $i = 2, \ldots$ nstep accordingly to Lanczos method.
3. Compute the second eigenvector **t** of the resulting tridiagonal matrix of size $nstep$.
4. Return $\sum_{j=1}^{nstep} t_j \mathbf{v}^{(j)}$.

Power method. The first proposal is a nearly straightforward application of the power method for the computation of the second eigenvector which takes advantage from the fact that the dominant eigenvector has constant entries. It is well known that the power method applied to the matrix $||A||_\infty I - A$ converges to the smallest eigenpair of A. Moreover, choosing the initial vector orthogonal to the eigenspace corresponding to the smallest eigenvector, the method converges to the second smallest eigenpair. Hence, the function `DoStepsPower` in Figure 5 consists of $2\,nstep$ basic power method steps with opportune orthogonalizations against the unit vector $\mathbf{u} = (1, 1, \ldots, 1)^T$ and normalizations in the infinity norm. The orthogonalization against the vector $\mathbf{u}$ is done every 2 power steps at stage 5.

Lanczos method. Taking advantage from the knowledge of the eigenvector corresponding to the eigenvalue 0, we can apply also the well known Lanczos procedure. The Lanczos method has in fact better convergence proprieties, in particular it guarantees the convergence at the external eigenvalues in at most n step [21]. Since only few iterations of Lanczos method are planned, the memory requirements of procedure `DoStepsLanczos` in Figure 6 are not prohibitive. For the same reason it is not necessary selective reortogonalization against the unit vector $\mathbf{u}$ which is guaranteed from the choice of the starting vector $\mathbf{v}_0$ in procedure `Cycle` of the algorithm in Figure 4.

Note that, both the iterative methods described above satisfy the following property

$$\texttt{DoStep}(J\mathbf{x}) = J\,(\texttt{DoStep}(\mathbf{x}))\,,$$

where J is the permutation matrix with ones on the anti-diagonal. This propriety shows that it is not necessary to apply the permutations induced by $\mathbf{v}$ and $-\mathbf{v}$ in step 4 of `Cycle` to get the minimal

wb between the two possible values. In fact, this test can be done only once on the vector $\mathbf{q}$ returned by `MSM`.

4.3 Stopping Criteria

For the practical implementation of the Modified Spectral Method, we have to establish the value of the parameter *nstep* as well as the number of times the procedure `Cycle` is executed. We denote by *ncycle* the number of repetition of the procedure `Cycle`. As we can see from the algorithm in Figure 4 the value of *ncycle* results from the application of a termination criterion which we are going to discuss in this section.

The termination criterion can be independent from the behaviour of the algorithm during the reduction process that is the value of *ncycle* depends only on "static" properties of the matrix (e.g. the size). In this case we have a *non-adaptive* algorithm.

Another possibility is to let the algorithm determinate the stopping criteria "dynamically", during the execution, by considering the resulting sequence of values of the bandwidth and work-bound obtained. In this case we have an *adaptive* algorithm.

In our experiments we tested both non-adaptive and adaptive algorithms and, after many tests we decided to use, in the non-adaptive scheme, a value of *ncycle* proportional to $\log n$, where n is the size of A.

In the preliminary numerical tests we experienced that the behaviour of wb, in most cases, is far more regular than that of bw (compare for example Figure 2 and 3). Hence we based the adaptive implementation on the observation of the mean on q consecutive values of the parameter wb at the end of each execution of `Cycle`. Let w_i be the current value of wb after the i-th execution of `Cycle`, and

$$w_i(q) = \frac{1}{q} \sum_{j=i-q+1}^{i} w_j, \quad i \geq q.$$

We stop the adaptive algorithms after the i-th execution of `Cycle` if

$$w_i(q) > 0.99\, w_{i-1}(q). \tag{10}$$

Then, our adaptive termination criterion is based on the decreasing rate of wb. The mean operator is introduced in order to further improve the smoothness of the function under observation.

Note that, while the non-adaptive algorithm always returns the minimum value of bandwidth after its fixed number of cycles, the goodness of our adaptive termination criterion relies on the hypothesis that the trend of wb is regular enough. So, if the value of *ncycle* is high, the non-adaptive scheme produces an expensive algorithm but also better results.

4.4 Spectral Method with early termination

As we already observed at the beginning of Section 4, we proposed two improvements of the Spectral Method: the introduction of a weighted Laplacian and the early termination of the iteration process. It is possible to devise an hybrid method (`SMET`) combining the original Laplacian with the termination conditions described above (see the algorithm in Figure 7). The main difference with the algorithm in Figure 1 is that in order to implement adaptive termination conditions we check the values of bandwidth and work-bound every *nstep* iterations. The results reported in Figure 9 and Table 3 show that this hybrid method can be better than the original Spectral algorithm.

Figure 7: Spectral Method with Early Termination.

```
Program SMET
```

1. Input matrix A.
2. Construct the Laplacian matrix
   $WLap(A) \leftarrow L(A)$
3. $\mathbf{e} \leftarrow (1, 2, \dots, n)^T$
4. $\mathbf{q} \leftarrow \mathbf{e}$
5. Choose the starting vector
   $\mathbf{v} \leftarrow \left(\mathbf{e} - \frac{\mathbf{e}^T\mathbf{u}}{\mathbf{u}^T\mathbf{u}}\mathbf{u}\right)$.
6. Repeat the procedure `Cycle1` until a termination
   condition is satisfied.
7. Return $\mathbf{q}$.

```
Procedure Cycle1
```

1. $\mathbf{v} = \texttt{DoSteps}(\mathbf{v})$.
2. Construct the corresponding permutation vector
   $\mathbf{p}$ induced by $\mathbf{v}$.
3. Reorder matrix A accordingly with $\mathbf{p}$.
4. Compute the values of $bw(A)$ and of $wb(A)$.
5. If the $bw(A)$ is the best value achieved till now
   then reorder vector $\mathbf{q}$ accordingly with $\mathbf{p}$.

4.5 Complexity of the MSM scheme

Memory requirements. As we already underlined speaking of the main disadvantages of the original Spectral Method, our objective was to apply the method also to very large matrices for which the method proposed by Barnard, Pothen and Simon has a prohibitive cost in terms of memory requirements. Let us now discuss the main choices we made in order to keep small the amount of memory necessary to our algorithms.

A simple and a convenient way to store the matrix A is to keep in memory the list of nonzeros indexes in the lower triangular part, this means that A is represented as $\{(i,j) \mid a_{ij} \neq 0,\, i > j\}$; this needs $2k$ integer locations.

On the other hand, it is not necessary to store the modified Laplacian $WLap(A)$. In fact that matrix will be used only to compute matrix-vector products and the i-th component of the product $WLap(A)\,\mathbf{x}$ reduces to the expression

$$\sum_{j \in \mathrm{row}(i)} f(|i-j|)\,(x_i - x_j)\,.$$

Similarly, it is not necessary to store the vectors $\mathbf{e}, \mathbf{u}$ and $\mathbf{v}_0$ in Program `MSM` since they have simple closed forms.

For what concerns the routines `MSM` and `Cycle`, the vector $\mathbf{v}$ needs n real memory location and the task of reordering can be done by means of two integer vectors of size n.

The memory cost of `DoStep` is different accordingly to the choice between Power and Lanczos method.

The memory requirements of `DoStepsPower` are the minimal possible since only an additional real vector $\mathbf{y}$ of size n is used. On the other hand `DoStepLanczos` requires *nstep* additional real vectors of size n, and nearly $3\,nstep$ real locations for the computation of the tridiagonal eigenvector.

The total memory requirements are then the following.

- $2k + 2n$ integer locations and $2n$ real locations for the Power method;
- $2k + 2n$ integer locations and $n(nstep + 1) + 3\,nstep$ real locations for the Lanczos method.

Moreover, if $\texttt{WLap1}_\alpha$ is used, an additional vector of size n is needed for storing the values that the function f achieves on $\{1, 2, \ldots, n\}$.

Time complexity.

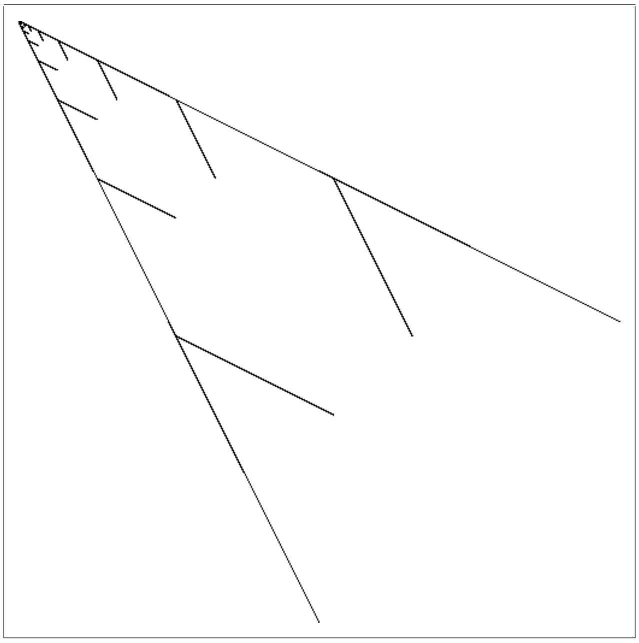
The time complexity can be expressed in terms of the quantities n, k, *nstep* and *ncycle* and depends as well on the choice of the iterative method for the approximation of eigenvectors.

The time cost of the program `MSM` is *ncycle* the cost of the procedure `Cycle`, so we examine step by step the time cost of that program. The choice of the starting vector in step 1 can be done in $O(n)$ operations, while step 2 requires no operations since the weighted Laplacian matrix is not pre-computed. The cost of step 3 depends on the choice of the iterative method. Step 4 requires $O(n \log n)$ operations using a standard sorting algorithm to reorder the vector $\mathbf{v}$. The computation of step 5 and 6 involves only the nonzero entries of A and $O(k)$ operations are then sufficient. Finally, step 7 requires $O(n)$ operations since the permutation vector $\mathbf{p}$ induced by $\mathbf{v}$ has already been computed.

To complete the analysis of the cost of the program `MSM` we have to examine the time required by procedures `DoSteps`. It is straightforward to see that time cost of the program `DoStepsPower` is

$$O(nstep(k+n))\,,$$

Figure 8: Pattern of the one-million size matrix TREE.



while program DoStepsLanczos can be executed in

$$O\Big(nstep(k+n) + nstep^3\Big)$$

operations.

Setting $nstep = O(\log n)$ the overall time complexity becomes

$$O(ncycle\,((n+k)\log n))$$

for both algorithms.

5 Test Matrices and Performance Measures

Numerical experiments have been carried out on a set of sparse matrices taken from T. Davis, University of Florida "Sparse Matrix Collection" [6]. Of these matrices only the pattern has been considered, and the main diagonal, which is not involved in the computation, has been removed. In some matrices there are rows containing only one isolated main diagonal nonzero entry; in this case, removing the main diagonal, we get void rows and columns. We have then removed these rows and columns decreasing the size of the matrix. Note that some matrices originally reducible, with this simple trick become irreducible so that the Spectral Method can be applied to these smaller matrices.

One more symmetric matrix, named TREE featuring a size $n = 10^6$ and an initial band of 5×10^5 has been added to the list. The pattern of TREE is shown in Figure 8.

In order to acquire reference values of bw and wb we applied to the test matrices the original Spectral Method (SM in the following) and the well known bandwidth reduction algorithm RCM [5].

The computation of these quantities has been carried out with MATLAB using the standard sparse matrix eigenvalue routines to get the permutation returned by `SM` and the routine `symrcm` which implements `RCM`. The `SM` method could not be applied to reducible matrices and to the 6 larger ones.

In Table 1 are listed the main properties of the 44 symmetric test matrices; in particular we denote by *Name* the matrix identifier, n the matrix size, k the number of nonzero entries in the strictly lower triangular part, c the number of connected components, bw_0 the initial bandwidth, $bw_{\mathtt{RCM}}$ and $bw_{\mathtt{SM}}$ the bandwidth after `RCM` and `SM`, respectively, $wb_{\mathtt{RCM}}$ and $wb_{\mathtt{SM}}$ the values of wb after `RCM` and `SM`, respectively. If the size is marked with an asterisk it means that the original size has been reduced by removing isolated diagonal entries. Also 9 unsymmetric matrices has been taken from the database [6]. These matrices were put in symmetric form by taking the pattern of $A + A^T$.

Since it is unlikely the same algorithm will return the best values of bw and wb for all the test matrices, it is necessary to introduce some performance indicator that can help us in rating the various possibilities.

Let $\mathcal{T} = \{T_i : T_i \in \mathbf{R}^{n_i \times n_i}\, i = 1, 2, \ldots, t\}$ be a generic set of t test matrices. Let `ALG` be the algorithm under consideration and let $bw^{\mathtt{ALG}}(T_i)$, $wb^{\mathtt{ALG}}(T_i)$ and $time^{\mathtt{ALG}}(T_i)$ be the bandwidth, the work-bound, and the execution time after the application of algorithm `ALG` to the matrix T_i. We define the following performance measures.

Relative Mean Bandwidth

$$\mathrm{RMB}(\mathtt{ALG}) = \sum_{i=1}^{t} \frac{bw^{\mathtt{ALG}}(T_i)}{t\, n_i}.$$

Relative Mean Work-Bound

$$\mathrm{RMW}(\mathtt{ALG}) = \sum_{i=1}^{t} \frac{wb^{\mathtt{ALG}}(T_i)}{t\, n_i^2}.$$

Mean Time

$$\mathrm{TIME}(\mathtt{ALG}) = \sum_{i=1}^{t} \frac{time^{\mathtt{ALG}}(T_i)}{t}.$$

Obviously, different performance measure on difference test sets, can alter the rating of the algorithms. To be fair we have chosen the "most natural" measures and we used only matrices given in the literature except the biggest one (`TREE`) introduced to reach the ambitious size of one million.

6 Numerical Experiments

All the experiments have been carried out using double precision arithmetic on a 650 MHz Pentium III computer with a 512 MByte RAM.

A large number of algorithms can be obtained by combining the different ingredients discussed in the previous sections. An extensive preliminary set of numerical experiments was carried out suggesting us the values of parameters for the final implementation of the algorithms. In particular, the values of α, *nstep*, *ncycle* and q have been chosen accordingly with the results of these experimentation. The generic algorithm is denoted with a four character identifier *xwtm* with the following

Table 1: List of symmetric test matrices.

Name	n	k	c	bw_0	$bw_{\mathtt{RCM}}$	$wb_{\mathtt{RCM}}$	$bw_{\mathtt{SM}}$	$wb_{\mathtt{SM}}$
3DTUBE	45330	1584144	1	3374	2741	1.54e11	-	-
BCSPWR08	1624	2213	1	1494	134	7.78e6	341	1.23e6
BCSPWR09	1723	2394	1	1663	133	7.33e6	241	1.41e6
BCSPWR10	5300	8271	1	5189	288	1.37e8	554	1.43e7
BCSSTK12	1473	16384	9	650	110	4.59e6	-	-
BCSSTK13	2003	40940	1	1250	546	1.76e8	1131	1.63e8
BCSSTK24	3562	78174	1	3333	251	1.00e8	285	7.42e7
BCSSTK29	13992	302748	28	1157	914	5.26e9	-	-
BCSSTK30	28924	1007284	1	16947	2512	3.82e10	5258	4.32e9
BCSSTK31	35588	572914	2	1668	1176	1.97e10	-	-
BCSSTK32	44609	985046	1	43030	2390	9.06e10	13336	2.83e10
BCSSTK33	8738	291583	1	932	749	1.93e9	1195	1.93e9
BCSSTK35	30237	709963	1	20690	1764	3.65e10	6021	1.00e10
BCSSTK36	23052	560044	1	19337	1474	1.94e10	4148	8.52e9
BCSSTK37	25503	557737	1	19193	1373	1.56e10	6864	1.45e10
BCSSTK38	8032	173714	1	6029	669	6.25e8	1000	4.52e8
BCSSTM13	691*	9970	22	280	189	6.63e6	-	-
BLCKHOLE	2121*	6370	1	1805	105	1.55e7	354	8.91e6
BUS1138	1138	1458	1	1030	131	3.72e6	272	7.37e5
BUS685	685	1282	1	550	102	1.78e6	163	2.28e5
CAN1054	1054	5571	1	1030	123	2.97e6	145	2.62e6
CAN1072	1072	5686	1	1048	178	6.07e6	312	4.08e6
CAN445	445	1682	1	403	78	1.13e6	155	9.12e5
CAN838	838	4586	1	837	149	4.19e6	139	2.89e6
CT20STIF	52329	1323067	1	49323	3187	1.94e11	-	-
DWT1007	1007	3784	1	986	38	6.27e5	81	4.63e5
DWT2680	2680	11173	1	2499	69	4.45e6	161	3.74e6
DWT918	918	3233	1	839	57	6.91e5	82	4.68e5
FINAN512	74752	261120	1	74724	1331	6.51e10	2278	6.19e9
GEARBOX	153746	4463329	6	137410	6271	1.36e12	-	-
GUPTA3	16783	4653322	1	16744	15046	3.26e11	-	-
JAGMESH1	936	2664	1	778	27	5.38e5	53	5.83e5
JAGMESH9	1349	3876	1	1024	40	1.19e6	107	9.82e5
MSC10848	10848	609465	1	10706	1349	4.00e9	1799	3.08e9
MSC23052	23052	565881	1	23046	1524	1.83e10	4066	8.00e9
NASA1824	1824	18692	1	239	302	4.72e7	448	3.13e7
NASA4704	4704	50026	1	423	419	2.42e8	408	1.36e8
PWT	36463*	144794	1	35399	339	1.02e9	923	7.51e8
PWTK	217918	5708253	1	189331	2190	2.27e11	-	-
SHUTTLEEDDY	10429	46585	1	9597	177	6.67e7	321	6.46e7
SKIRT	12595*	91961	4	1046	314	1.73e8	-	-
SSTMODEL	2730*	9702	1	123	87	6.09e6	200	4.72e6
TREE	1000000	2674627	1	500000	155652	5.88e15	-	-
VIBROBOX	12328	165250	1	12162	4764	1.06e11	4101	2.70e10

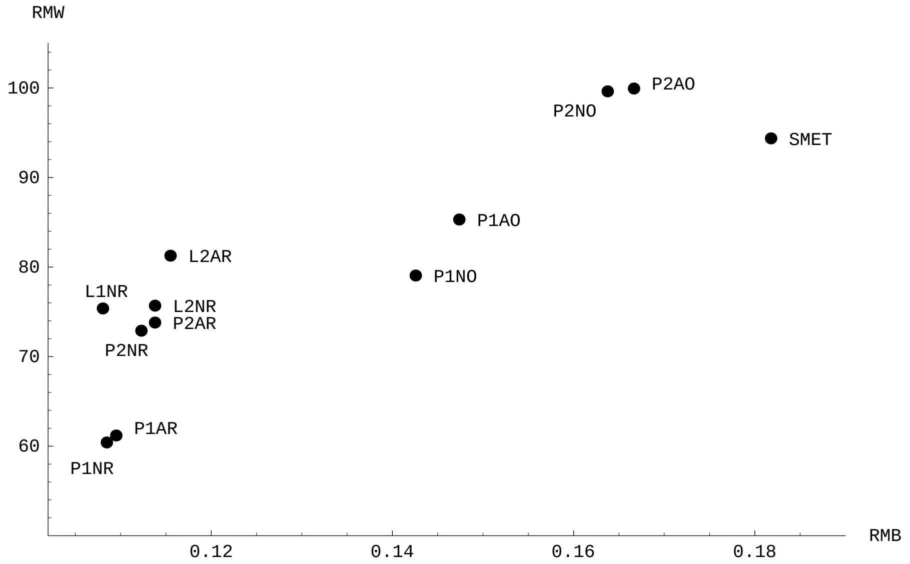
Table 2: List of unsymmetric test matrices.

Name	n	k	c	bw_0	bw_{RCM}	wb_{RCM}	bw_{SM}	wb_{SM}
ADD20	2395	7462	1	2313	1157	3.20e8	1816	9.78e7
ADD32	4960	9462	1	4029	734	3.90e8	1043	1.67e7
BARTH	6691	19748	1	6523	200	9.82e7	341	4.09e7
BARTH4	6019	17473	1	5305	215	1.07e8	872	3.23e7
BARTH5	15606	45878	1	15080	370	8.13e8	688	1.89e8
EX27	974	19904	1	136	197	6.74e6	188	6.47e6
MEMPLUS	17758	54196	1	17751	10763	2.56e11	14594	3.23e11
PRE2	659033	4792957	3	117686	53603	3.42e14	-	-
TWOTONE	120750	1029429	1	21749	30782	1.78e13	-	-

Table 3: Results of numerical experiments. The underscript 40 denotes that we have taken the mean over the 40 matrices for which the Spectral Method is applicable; the underscript ALL denotes that we have taken the mean over all the 53 test matrices.

Algorithm	RMB_{40}	RMB_{40}	TIME_{40}	RMB_{ALL}	RMW_{ALL}	TIME_{ALL}
P1NR	0.097	21.53	137.91	0.108	60.41	702.88
P1AR	0.098	21.90	44.95	0.109	61.19	258.53
L1NR	0.099	24.26	13.22	0.108	75.37	52.17
P2NR	0.101	22.00	134.45	0.112	72.89	663.80
L1AR	0.102	30.82	6.19	0.111	104.07	24.40
P2AR	0.103	22.58	49.74	0.113	73.80	225.57
L2NR	0.104	25.55	12.52	0.113	75.69	50.77
L2AR	0.106	32.14	5.60	0.115	81.26	18.58
RCM	0.115	50.77	–	0.125	214.31	–
P1NO	0.131	17.98	138.38	0.142	79.05	733.77
P1AO	0.136	18.60	71.64	0.147	85.31	305.19
P2NO	0.148	18.23	134.17	0.163	99.61	658.36
P2AO	0.151	18.60	62.01	0.166	99.93	264.80
SMET	0.173	16.15	70.73	0.181	94.37	246.17
L2NO	0.178	25.15	12.45	0.197	893.95	52.16
L2AO	0.189	33.82	9.98	0.206	900.95	21.40
L1NO	0.195	210.88	13.31	0.196	238.41	52.62
SM	0.195	36.50	–	–	–	–
L1AO	0.202	211.62	9.65	0.204	237.21	31.22

Figure 9: Performance of the best algorithms.



meaning:

$$x = \begin{cases} P & \text{The function } \texttt{DoStepsPower} \text{ is used choosing } nstep = \lfloor 4 \log n \rfloor, \\ L & \text{The function } \texttt{DoStepsLanczos} \text{ is used choosing } nstep = \lfloor 0.5 \log n \rfloor \ . \end{cases}$$

$$w = \begin{cases} 1 & \text{The continuous weight Laplacian } \texttt{WLap1}_\alpha \text{ is used choosing } \alpha = 0.5, \\ 2 & \text{The Step-wise weight Laplacian } \texttt{WLap2}_\alpha \text{ is used choosing } \alpha = 0.4. \end{cases}$$

$$t = \begin{cases} N & \text{Non adaptive termination condition with } ncycle = \lfloor 50 \log n \rfloor, \\ A & \text{Adaptive termination condition in equation(10) with } q = 10. \end{cases}$$

$$m = \begin{cases} O & \text{Algorithm applied to the original matrix,} \\ R & \text{Algorithm applied to the matrix pre-processed with } \texttt{RCM}. \end{cases}$$

So, for example, L1NR identifies the algorithm that uses RCM as a preprocessor, Lanczos method, the continuous variation of the Laplacian weight $\texttt{WLap1}_\alpha$ and a non adaptive strategy.

In Table 3 are listed the values of the performance parameters on two sets of test matrices with 16 algorithms, and with RCM, SM and the Spectral Method with early termination SMET.

The pair $(\texttt{RMB}_{\texttt{ALL}}, \texttt{RMW}_{\texttt{ALL}})$ has been used to show the performance of the 12 best algorithm in a cartesian plot (see Figure 9). In Table 4, for each matrix are presented the best result of bandwidth together with the corresponding work-bound and the identifier of the algorithm which gets that performance.

The results obtained on our test sets and with our performance measures allow us to compare our algorithms with RCM and SM. From Table 3 it is self evident that better results are achieved by

Table 4: Best results of bandwidth for the 53 test matrices, together with the corresponding work-bound and the identifier of the algorithm which achieves these values.

Name	Algorithm	best bw	wb	Name	Algorithm	best bw	wb
3DTUBE	SMET	2334	1.48e11	CAN445	RCM	78	1.13e6
ADD20	L1NR	711	1.19e8	CAN838	P2NR	126	2.8e6
ADD32	P1AR	669	3.09e7	CT20STIF	RCM	3187	1.94e11
BARTH	RCM	200	9.82e7	DWT1007	L1AR	38	6.27e5
BARTH4	RCM	215	1.07e8	DWT2680	L1AR	69	4.45e6
BARTH5	L1AR	370	8.13e8	DWT918	L1NR	50	4.55e5
BCSPWR08	P1NR	131	1.10e6	EX27	L2AO	128	5.81e6
BCSPWR09	P1NR	123	1.18e6	FINAN512	L1AR	1331	6.51e10
BCSPWR10	RCM	288	1.37e8	GEARBOX	RCM	6271	1.36e12
BCSSTK12	SMET	109	4.29e6	GUPTA3	L1NO	12535	5.97e11
BCSSTK13	L1AR	546	1.76e8	JAGMESH1	L1AR	27	5.38e5
BCSSTK24	P2NO	227	7.10e7	JAGMESH9	L1AR	40	1.19e6
BCSSTK29	SMET	838	1.09e9	MEMPLUS	P1NR	5747	7.48e10
BCSSTK30	RCM	2512	3.82e10	MSC10848	L1AR	1349	4.00e9
BCSSTK31	RCM	1176	1.97e10	MSC23052	L1AR	1524	1.83e10
BCSSTK32	L1AR	2390	9.06e10	NASA1824	L1AO	205	2.68e7
BCSSTK33	RCM	749	1.93e9	NASA4704	L1NO	348	1.40e8
BCSSTK35	L1AR	1764	3.65e10	PRE2	P1NR	50367	6.58e13
BCSSTK36	L1AR	1474	1.94e10	PWT	RCM	339	1.02e9
BCSSTK37	L1AR	1373	1.56e10	PWTK	L1AR	2190	2.27e11
BCSSTK38	RCM	669	6.25e8	SHUTTLEEDDY	RCM	177	6.67e7
BCSSTM13	L1NO	171	6.5e6	SKIRT	RCM	314	1.73e8
BLCKHOLE	RCM	105	1.55e7	SSTMODEL	RCM	87	6.09e6
BUS1138	P1AR	106	5.52e5	TREE	P1AR	100045	1.20e12
BUS685	L1NR	83	2.72e5	TWOTONE	P1AO	19538	4.43e12
CAN1054	L1AR	121	2.59e6	VIBROBOX	L1NR	3961	2.72e10
CAN1072	RCM	178	6.07e6	–	–	–	–

using the RCM algorithms as a sort of preprocessing. In fact, the application of our methods after RCM slightly improves the bandwidth producing at the same time a significant improvement of the work-bound. The Spectral method, even when applicable gives poor results in terms of bandwidth. We note that non adaptive algorithms are slightly better than adaptive ones at the expenses of an huge increase of the computational cost. Figure 9 evidences that the use of the continuous modified Laplacian $\mathtt{WLap1}_\alpha$ gives better results than the discrete modified Laplacian $\mathtt{WLap2}_\alpha$ and the time cost of the two alternatives is similar. Moreover, the time cost of Lanczos algorithm is much smaller than the cost of the Power method due to the smaller number of iterations since the memory constraint suggests a low value of *nstep*. On the other hand, Lanczos method seems to be less reliable when it is not applied after RCM. As a final conclusion we can suggest that if RCM algorithm can be used as a preprocessor one can use L1NR or P1AR, otherwise, P1AO seems to be the best compromise between cost and performance for a direct application of a Modified Spectral Method.

7 Conclusions

The Modified Spectral Methods seem to return a permutation which yields a good compromise of bandwidth and work-bound. In fact, typically, the ordering obtained produces a lower bandwidth than the Spectral Method and a lower work-bound than the RCM method. We stress that applying one of the Modified Spectral Methods after RCM allows often getting reordered matrices with also better values of bandwidth.

In the paper our attention has been posed on the measures of bandwidth and work-bound rather than on the envelope-size. This has been done to have a more fair comparison with the Spectral Method which is especially designed to reduce the work-bound. Our methods result however competitive also with respect to the envelope-size measure.

References

[1] G. Alaghband. Parallel Pivoting Combined with Parallel Reduction and fill-in control. *Parallel Comput.*, 11:210–221, 1989.

[2] S. T. Barnard and H. D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6(2):101–117, 1994.

[3] S.T. Barnard, A. Pothen, and H.D. Simon. A spectral algorithm for envelope reduction of sparse matrices. *Journal Num. Lin. Alg. with Appl.*, 2:311–334, 1995.

[4] M. Berry, B. Hendrickson, and P.raghavan. Sparse matrix reordering schemes for browsing hypertext. In S. Smale J. Renegar, M. Shub, editor, *Lectures in Appl. Math. 32: the Mathematics of Numerical Analysis*, pages 99–123. AMS, Providence, RI, 1996.

[5] E. H. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proc. 24th Nat. Conf. Assoc. Comp. Mach.*, pages 157–172. ACM Publications, 1969.

[6] T. Davis. Sparse matrix collection.
http://www.cise.ufl.edu/˜davis/sparse/, ftp://ftp.cise.ufl.edu/pub/faculty/davis/matrices,

1994. NA Digest, vol. 92, no. 42, October 16, 1994, NA Digest, vol. 96, no. 28, July 23, 1996, and NA Digest, vol. 97, no. 23, June 7, 1997.

[7] E. F. D'Azevedo, P. A. Forsyth, and W. P. Tang. Ordering methods for preconditioned conjugate gradient methods applied to unstructured grid problems. *SIAM J. Matrix. Anal. Appl.*, 13:944–961, 1992.

[8] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29:635–657, 1989.

[9] A. Esposito, S. Fiorenzo Catalano, F. Malucelli, and L. Tarricone. A new matrix bandwidth reduction algorithm. *Operation Research Letters*, 1999. To appear.

[10] A. Esposito, F. Malucelli, and L. Tarricone. Bandwidth reduction of sparse symmetric and unsymmetric matrices: an experimental comparison of new heuristics. `http://dvorak.istel.ing.unipg.it`.

[11] M. Fiedler. A propriety of non-negative symmetric matrices and its application to graph theory. *Czech. Mat. J.*, 25:619–633, 1975.

[12] A. George and A. Pothen. An analysis of spectral envelope-reduction via quadratic assignment problems. *SIAM J. Matrix Anal. Appl.*, 18(3):706–732, 1997.

[13] N. Gibbs, W. Poole, and P. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Numer. Anal.*, 13:235–251, 1976.

[14] N. Gibbs, W. Poole, and P. Stockmeyer. A comparison of several bandwidth and profile reduction algorithms. *ACM Trans. on Math. Software*, 2:322–330, 1976.

[15] M. Juvan and B. Mohar. Optimal linear labelings and eigenvalues of graphs. *Discr. Appl. Math*, 36:153–168, 1992.

[16] M. Juvan and B. Mohar. Laplace eigenvalues and bandwidth-type invariants of graphs. *J. Graph Theory*, 17:393–407, 1993.

[17] J.W.H.Liu and H.Sherman. Comparative analysis of the Cuthill-Mckee and the reverse Cuthill-Mckee ordering algorithms for sparse matrices. *SIAM J. Numer. Anal.*, 13:198–213, 1976.

[18] J. G. Lewis. Implementations of the Gibbs-Poole-Stockmeyer and Gibbs-King algorithms. *ACM Trans. on Math. Soft.*, 8:180–189, 1982.

[19] C. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[20] B. N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice Hall, Englewood Cliffs, New Jersey, 1980.

[21] Y. Saad. *Numerical Methods for Large Eigenvalues Problems.* Manchester University Press, Manchester, 1992.