Numer Algor (2016) 71:383–394
DOI 10.1007/s11075-015-9997-2

ORIGINAL PAPER

Algorithms for the CMRH method for dense linear systems

Sébastien Duminil[1] · Mohammed Heyouni[2] · Philippe Marion[1] · Hassane Sadok[1]

Received: 19 November 2013 / Accepted: 14 April 2015 / Published online: 20 May 2015

Abstract The CMRH (Changing Minimal Residual method based on the Hessenberg process) method is a Krylov subspace method for solving large linear systems with non-symmetric coefficient matrices. CMRH generates a (non orthogonal) basis of the Krylov subspace through the Hessenberg process, and minimizes a quasi-residual norm. On dense matrices, the CMRH method is less expensive and requires less storage than other Krylov methods. In this work, we describe Matlab codes for the best of these implementations. Fortran codes for sequential and parallel implementations are also presented.

Keywords Linear systems · Krylov method · Hessenberg process · Dense matrix · CMRH method

✉ Hassane Sadok
sadok@lmpa.univ-littoral.fr

Sébastien Duminil
duminil@lmpa.univ-littoral.fr

Mohammed Heyouni
mohammed.heyouni@gmail.com

Philippe Marion
marion@lmpa.univ-littoral.fr

[1] Laboratoire de Mathématiques Pures et Appliquées, Université du Littoral Côte d'Opale, Centre Universitaire de la Mi-Voix, Batiment H. Poincaré, 50 Rue F. Buisson, BP 699, 62228 Calais cedex, France

[2] ENSAH, Ecole Nationale des Sciences Appliquées d'Al-Hoceima, Université Mohammed Premier, Oujda, Maroc

1 Introduction

Let A be an $n \times n$ non-symmetric matrix, let $b,\ x \in \mathbb{R}^n$, and consider the linear system of equations

$$A\,x = b. \tag{1}$$

Given an initial guess x_0 for the exact solution of (1), the CMRH method [12] constructs approximate solutions $\{x_k\}_{k=1,\ldots,m}$ of the form

$$x_k = x_0 + w_k, \tag{2}$$

where $w_k \in K_k(A, r_0)$, $r_0 = b - A\,x_0$, is the initial residual and $K_k(A, r_0)$ is the Krylov subspace defined by

$$K_k(A, r_0) = span\{r_0, Ar_0, \ldots, A^{k-1}r_0\}.$$

The CMRH algorithm uses the Hessenberg process to compute a basis $\{l_1, \ldots, l_k\}$ of $K_k(A, r_0)$. Furthermore this basis is such that the matrix $L_k = [l_1, \ldots, l_k]$ is unit lower trapezoidal. Note also that the CMRH method shares many of the computational properties of the well-known GMRES method; see [10, 11] for details on GMRES. Some of the properties shared by these two methods include the fact that the matrix A is only needed as an operator for matrix-vector products. Moreover, only one matrix-vector product evaluation is needed at each iteration and the kth approximation is easily updated by solving a small least squares problem.

Heyouni and Sadok [7] proposed an implementation which minimizes memory requirements. At each iteration of the Hessenberg process, the matrix-vector product $A\,l_k$ is needed. But, as the $k-1$ first components of l_k are zero, we do not need access to the $k-1$ first columns of the matrix A to perform this matrix-vector product. Thus, this property allowed to provide an implementation of the CMRH method in which the $k-1$ first Krylov basis vectors of L_k are stored in the $k-1$ first columns of the matrix A.

In [6], Heyouni introduced two new methods for solving linear systems with several right-hand sides. These methods are the global Hessenberg and global CMRH methods. The analysis of the performance of the CMRH algorithm combined with the boundary element method when applied to acoustic problems is considered in [1]. Recently, in [3], Duminil presented an implementation for parallel architectures and an implementation of the left-preconditioned CMRH method.

This paper is organized as follows. In Section 2, we recall the CMRH implementation proposed in [7]. Section 3 reviews some convergence results which were given in [12, 13]. Finally, the proposed software is briefly described in Section 4.

2 CMRH method

In this section, we briefly review the Hessenberg process and the CMRH implementation for dense matrices. The Hessenberg reduction process (with pivoting strategy)

computes a unit trapezoidal matrix L_m by using the following formulas. It begins by computing

$$l_1 = \frac{r_0}{(r_0)_{p_1}},$$

where $p_1 \in \{1, \ldots, n\}$ is such that $|(r_0)_{p_1}| = \|r_0\|_\infty$. Then, at each step k, it proceeds by computing l_{k+1} which satisfies

$$h_{k+1,k}\, l_{k+1} = A\, l_k - \sum_{j=1}^{k} h_{j,k}\, l_j, \quad \text{for } k = 1, \ldots, m.$$

The parameters $h_{j,k}$ are determined such that

$$l_{k+1} \perp e_{p_1}, \ldots, e_{p_k} \text{ and } (l_{k+1})_{p_{k+1}} = 1$$

where $p_j \in \{1, 2, \ldots, n\}$ and e_i is the ith vector of the canonical basis. It is not hard to see that the Hessenberg process takes fewer operations and requires less storage than the Arnoldi process. Note also that the columns of L_m form a non-orthogonal basis of $K_m(A, r_0)$.

By letting L_k and $\bar{H}_k$ be the matrices generated by the Hessenberg process, the correction w_k given in (2) can be written in the form $w_k = L_k d_k$, where $d_k \in \mathbb{R}^k$ is the solution of the following least squares problem

$$\min_{d \in \mathbb{R}^{k+1}} \|\beta\, e_1 - \bar{H}_k d\|, \qquad \text{where } \beta = (r_0)_{p_1}.$$

Note that the CMRH and QMR methods use similar ideas in updating the approximate solutions. More precisely, it is shown, in [5], that the QMR and CMRH iterates are updated by imposing a minimizing semi-norm condition. The CMRH method with over-storage and pivoting strategy is described in Algorithm 1. For more details, see [7].

3 Convergence results

We recall some results on the convergence of the CMRH method. Proofs of these results can be found in [12, 13]. The result given in the following theorem shows that CMRH cannot break down and that it gives the solution of the system (1) at the same iteration as GMRES.

Theorem 1 *If the degree of the minimal polynomial of the matrix A for the vector r_0 is k, then the iterates x_j in the CMRH method are well defined for $j = 1, \ldots, k$ and x_k is the exact solution of (1).*

We now give a result which relates the norm of the residual vector r_k^C for CMRH to the residual r_k^G for GMRES.

Algorithm 1 CMRH method

Inputs : $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $x \in \mathbb{R}^n$ an initial guess, *tol* a tolerance ;
Output : x_k the kh approximate solution. (x_k is stored in b) ;
Compute $b = b - A\,x$;
% Hessenberg process:
Let $p = [1, 2, \ldots, n]^T$;
Determine i_0 such that $|b_{i_0}| = \|b\|_\infty$;
$\beta = b_{i_0},\ b = b/\beta$;
$p_1 \leftrightarrow p_{i_0},\ b_1 \leftrightarrow b_{i_0}\ A_{1,:} \leftrightarrow A_{i_0,:},\ A_{:,1} \leftrightarrow A_{:,i_0}$;
for $k = 1, \ldots,$ *until convergence* **do**
  $u = A_{:,k} + A_{:,k+1:n}\, b_{k+1:n}$;
  $A_{k+1:n,k} = b_{k+1:n}$;
  for $j = 1, \ldots, k$ **do**
    $A_{j,k} = u_j,\ u_j = 0,\ u_{j+1:n} = u_{j+1:n} - A_{j,k}\, A_{j+1:n,j}$;
  end
  Determine $i_0 \in \{k+1, \ldots, n\}$ such that $\left|u_{p_{i_0}}\right| = \left\|u_{p_{k+1}:p_n}\right\|_\infty$;
  $h = u_{p_{i_0}},\ b = u/h$;
  $p_{k+1} \leftrightarrow p_{i_0},\ b_{k+1} \leftrightarrow b_{i_0}$;
  $A_{k+1,:} \leftrightarrow A_{i_0,:},\ A_{:,k+1} \leftrightarrow A_{:,i_0}$;
  % More details on the two next steps can be found in [7]
  Update the QR factorization of $\bar{H}_k$;
  Apply previous rotations to $\bar{H}_k$ and $\beta\, e_1$;
end
Solve $H\, d = \beta\, e_1$; % ($H = H_k = triu(A_{1:k,1:k})$)
Update $x = x + L\, d$; % ($L = L_k = diag(ones(k,1)) + tril(A_{:,1:k}, -1)$)
% Reorder the components of x
for $i = 1, \ldots, n$ **do**
  $b_{p_i} = x_i$;
end

Theorem 2 *Let r_k^C and r_k^G be the CMRH and GMRES residuals obtained at the kth iteration respectively and such that the initial residual satisfies $r_0^C = r_0^G = r_0$. Then*

$$\|r_k^G\| \leq \|r_k^C\| \leq \kappa(L_{k+1})\|r_k^G\|,$$

where $\kappa(L_{k+1}) = \|L_{k+1}\|\|L_{k+1}^+\|$ is the condition number of L_{k+1}.

The above result indicates that as long as the condition number of L_{k+1} does not grow too fast, the CMRH residual norm will be close to the GMRES residual norm.

4 Parallel implementation

In [3], Duminil proposed a parallel implementation of the CMRH method. In this section, we give some details on this implementation. We want to analyse the cost of one iteration of CMRH with a specific data distribution. Duminil assumed that the matrix is distributed among Np processors in which each processor holds a set of rows of the matrix. The biggest communication costs in this parallelization problems arise in matrix-vector product evaluation and the swap of rows. He showed that this distribution allows the same number of operations in each processor for matrix-vector

Table 1 Summary of the maximum costs and maximum transfers at iteration k

	CMRH		GMRES	
	Operations	Transfer	Operations	Transfer
Matrix-vector product	$2nl \times (n-k)$		$2nl \times n$	
Dot product			$2k \times nl + Np$	$k \times Np$
$u = u - h_{j,k} v_j$	$2k \times nl$	k	$2k \times nl$	
$\|u\|_2$ or $\|u\|_\infty$	nl	Np	$2nl + Np + 1$	Np
Swap column	nl			
Swap row	n	n		

product evaluations and permits a permutation of the columns of the matrix A without any data transfer; only the permutation of the rows involves transfer of data.

Table 1 gives a summary of all costs and transfers for one iteration of the CMRH and GMRES methods with the same data distribution.

5 Software

We provide Matlab and Fortran codes for the CMRH implementations described in [3, 7]. Some tests are also included. The software is available from Netlib (http://www.netlib.org/numeralgo/) as the na41 package.

It is distributed as a compressed archive. Installation details can be found in the README.txt file included in the package. These programs can be tested on matrices coming from the University of Florida Matrix collection [2], from the Matrix Market web server [9], or from [3]. All results can be stored in an output file. The package contains two directories: Fortran and Matlab. We give an overview of the subroutines and programs included in these folders.

5.1 Fortran directory

This folder consists of a number of programs to test sequential and parallel CMRH methods on some examples. The following programs are included in this folder.

- **makefile**: install and configure all fortran programs
- **initialisation.f90**: This file creates inputfile.dat

Table 2 Results for Example 1

	CMRH	GMRES
Iter.	130	125
residual norm	$8.31\ 10^{-6}$	$7.7\ 10^{-6}$
error norm	$1.22\ 10^{-5}$	$3.5\ 10^{-6}$

Table 3 Results for Example 2

	CMRH	GMRES
Iter.	11	11
residual norm	$1.17\ 10^{-10}$	$8.68\ 10^{-11}$
error norm	$7.55\ 10^{-11}$	$5.13\ 10^{-11}$

- **CMRH.f90**: CMRH program
- **PCMRH.f90**: Parallel CMRH program
- **numtest.f90**: contains all subroutines for numerical tests section

Before compiling the fortran programs, one should adjust libraries (lapack, blas) and compilers (fortran, mpi) to your computer configurations in *makefile*. Then one enters make all (to configure all programs) or make allseq (to config only sequential programs) to compile the programs. Finally one enters ./initialisation to config inputfile.dat and ./CMRH to run the CMRH program or mprun -n (nb of procs) ./PCMRH to run the PCMRH program.

5.2 Matlab directory

This folder consists of a number of functions to test the sequential CMRH method in Matlab. We list these subroutines with a brief explanation.

- **main.m**: main program
- **CMRH.m**: CMRH function. Input parameters are :
  - A, $n \times n$ matrix,
  - b, right-hand side vector,
  - tol, tolerance of the stopping criterion (default 10^{-7}),
  - $maxiter$, maximum number of iterations (default n),
  - x_0, initial approximation (default $(0, \ldots, 0)^t$),
  - $iprint$, flag parameter to print or not some components of the approximated solution and the norm of the residual,
  - $ctol$ stopping criterion (residual seminorm or residual norm).

Table 4 Results for Example 3

	$\epsilon = 0.1$		$\epsilon = 10^{-15}$	
	CMRH	GMRES	CMRH	GMRES
Iter.	472	447	1000	1000
residual norm	$3.34\ 10^{-10}$	$3.40\ 10^{-10}$	$6.81\ 10^{-15}$	$2.68\ 10^{-15}$
error norm	$3.31\ 10^{-9}$	$2.41\ 10^{-9}$	$5.72\ 10^{-13}$	$4.33\ 10^{-13}$

Table 5 Results for Example 4

	CMRH	GMRES
Iter.	133	157
residual norm	$3.8\ 10^{-11}$	$4.59\ 10^{-12}$
error norm	$2.34\ 10^{-5}$	$2.29\ 10^{-5}$

Output parameter are :

- x, solution,
- $tnorm$, residual norm array,
- $flag$, to indicate what the stopping criterion used (tol or $maxiter$).

- **matrice.m**: To create a matrix example
- **mmread.m**: Matrix market program used to store matrix market file in A
- **example1.m**: Test matrix
- **hankel.m**: Test matrix
- **brown.m**: Test matrix
- **helsing.m**: Test matrix
- **gemat11.mtx**: A matrix market example

One has to enter the Matlab directory and run main.m to test the CMRH method.

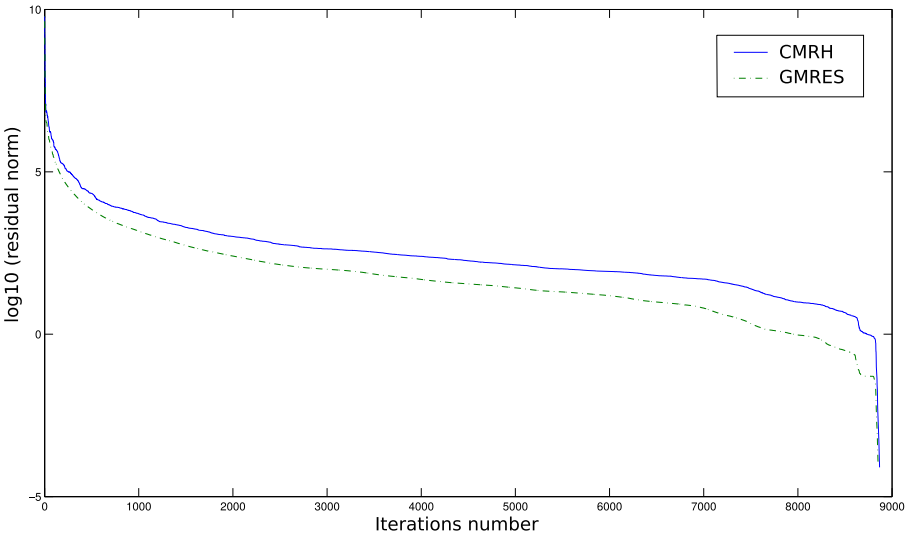

Fig. 1 Residual norm for the matrix B

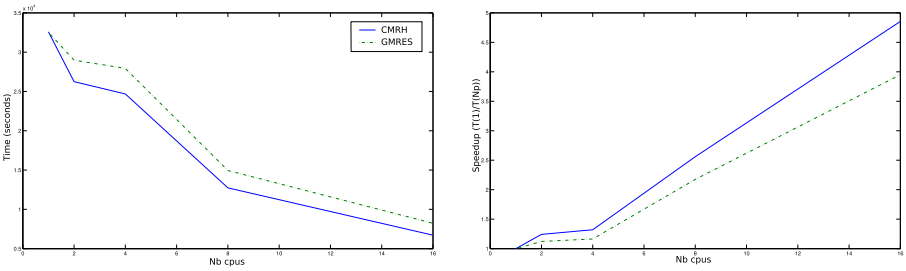

Fig. 2 CPU Time (seconds) and Speedup for the matrix B ($n = 32000$)

6 Numerical tests

In this section, we give results of some tests included in our package and in [3, 7, 12, 13]. We compare the CMRH with the GMRES method. We see that the convergences are similar and the parallel CMRH method is faster than the parallel GMRES method. When not specified otherwise, the right-hand side is chosen such that the exact solution is $x^* = 1$. We let $x_0 = 0$. The iteration was stopped as soon as the convergence criterion

$$\|r_k\|/\|r_0\| < 10^{-10}$$

was satisfied. In parallel tests, we use speedup to define the ratio of serial execution time to the parallel execution time. In tables and figures we give the absolute residual norm and the absolute residual error.

All following results obtained in this section have been computed on a BullNovascale 5160 (16 CPUs : Itanium II 1.5 GHz; 64 GB of RAM).

6.1 Sequential tests

Example 1 The first example is taken from [12]. We consider the $n \times n$ matrix

$$A = \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ a_1 & 1 & 1 & \cdots & 1 & 1 \\ a_1 & a_2 & 1 & \cdots & 1 & 1 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_{n-1} & 1 \end{pmatrix}$$

with $a_i = 1 + i\epsilon$. We choose $\epsilon = 10^{-2}$ and $n = 1000$. The accuracy is investigated in Table 2.

Table 6 CPU time (seconds) of the CMRH method and GMRES method with different values of n

n	iterations	CMRH	GMRES
16000	6688	941	1525
32000	12846	6713	8229
48000	18181	20873	28217

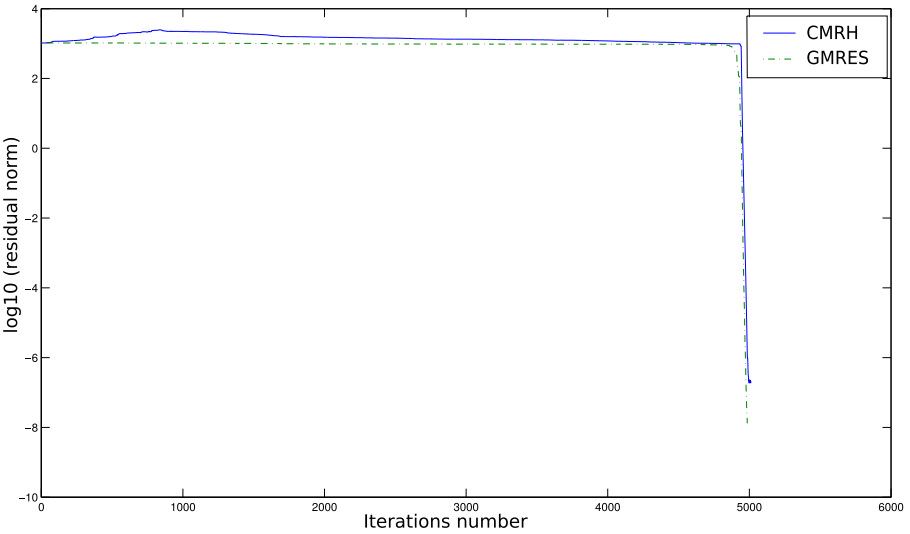

Fig. 3 Residual Norm for the TSOPF matrix

Example 2 In this example we consider the symmetric Hankel matrix with elements

$$A(i, j) = \frac{0.5}{n - i - j + 1.5},$$

see [13]. The convergence of both methods is very fast as can be seen in Table 3 with $n = 1000$.

Example 3 This test was given in [12]. We consider the $n \times n$ matrix

$$A = \begin{pmatrix} \epsilon & 1 & & \\ -1 & \epsilon & 1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & \epsilon & 1 \\ & & & -1 & \epsilon \end{pmatrix}$$

In Table 4, we give the results for two values of ϵ when $n = 1000$.

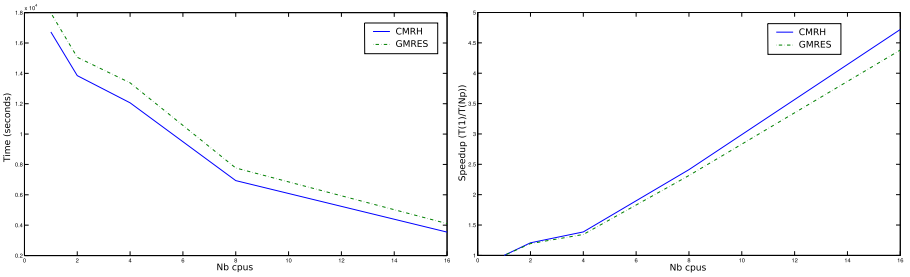

Fig. 4 CPU Time (seconds) and Speedup for the TSOPF matrix

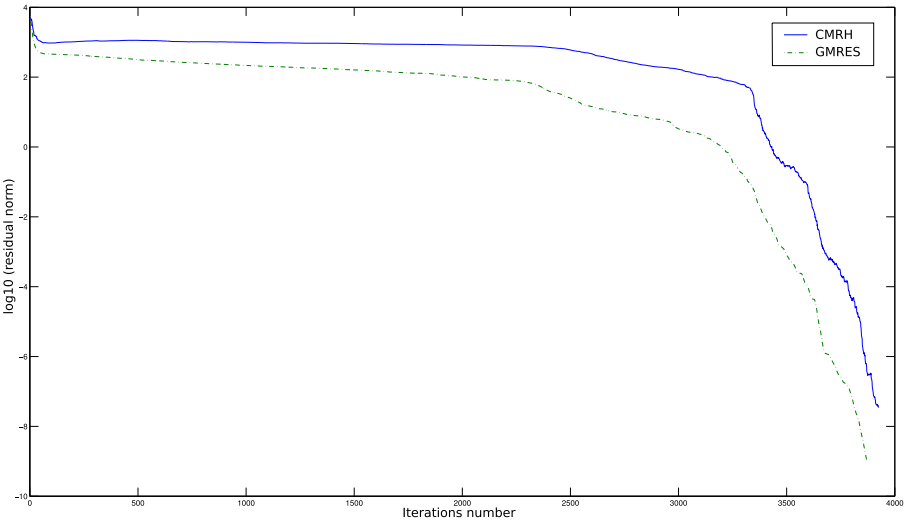

Fig. 5 Residual Norm for the matrix C

Example 4 This test was studied in [7, 13]. Consider the solution of the Fredholm integral equation of the first kind

$$\int_{6}^{6} \kappa(s,t)x(t)\,dt = y(s), \quad -6 \le s \le 6, \tag{3}$$

Its solution, kernel and right-hand side are given by

$$x(t) = \begin{cases} 1 + cos(\frac{\pi}{3}t) & if\ |t| < 3, \\ 0 & otherwise \end{cases}$$

$$\kappa(s,t) = x(s-t),$$

$$y(s) = (6 - |s|)\left(1 + \frac{1}{2}cos(\frac{\pi}{3}s)\right) + \frac{9}{2\pi}sin(\frac{\pi}{3}|s|).$$

We use the code phillips.m from [4] to discretize (3) by a Galerkin method with orthonormal box functions as test and trial functions to obtain the symmetric matrix

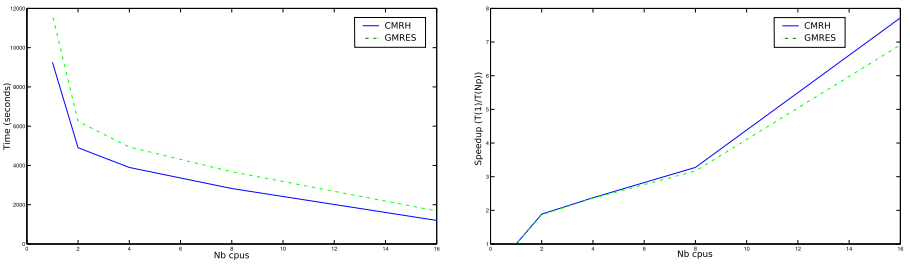

Fig. 6 CPU Time (seconds) and Speedup for the matrix C

A of order $n = 1000$. Table 5 compares the convergence of the CMRH and GMRES methods.

6.2 Parallel tests

– *Helsing code :* In [3], Duminil studied a matrix defined by

$$B_{i,j} = \begin{cases} -log\left|z_i - z_j\right|, & i \neq j, \\ -log\left|r_i\right| \end{cases} \tag{4}$$

where z_i are n randomly distributed points in a unit square centered at the origin in the complex plane and where each r_i is a number in $(0, d_i]$, d_i being the distance between the point z_i and its nearest neighbour. We observe that computing an off-diagonal entry B_{ij} corresponds, up to a factor of $-\frac{1}{2\pi}$, to evaluating the free space Green's function for the Laplacian in two dimensions with argument $z_i - z_j$. For more details, see [8]. We compared the convergence analysis, the computing times with different sizes of matrices and different number of processors and the speedup. Figures 1 and 2 and Table 6 give the results.

– *TSOPF matrix:* We use a matrix coming either from University of Florida Matrix collection [2] TSOPF. It is of order $n = 38120$ with $nnz = 16171169$ nonzeros entries. We treated this matrix as dense and nonsymmetric matrix. Figures 3 and 4 give the results.

– *Helmholtz code:* The last matrix comes from the integral formulation of the Helmholtz equation. The matrix is denoted as C and defined as follows

$$C = I + \kappa^2 K,$$

where κ represents the frequency of the wave oscillations and the entries of the K matrix as $K_{ij} = h^2 g(x_i - x_j)$. h is the step size of the discretized domain and g is the freespace Green's function in $\mathbb{R}^2$. The matrix K (and therefore A) is a dense non-Hermitian square matrix. We choose $n = 20736$. Figures 5 and 6 give the results.

Acknowledgments We thank the referees for their comments and questions, which helped improve our presentation.

References

1. Alia, A., Sadok, H., Souli, M.: CMRH method as iterative solver for boundary element acoustic systems. Eng. Anal. Bound. Elem. **36**, 346–350 (2012)
2. Davis, T.A., Hu, Y.: The University of Florida matrix collection. ACM T. Math. Software, 38 (2011). http://www.cise.ufl.edu/research/sparse/matrices
3. Duminil, S.: A parallel implementation of the CMRH method for dense linear systems. Numer. Alg. **63**, 127–142 (2012)
4. Hansen, P.C.: Regularization tools: a Matlab package for analysis and solution of discrete ill-posed problems. Numer. Alg. **6**, 1–35 (1994)
5. Heyouni, M., Sadok, H.: On a variable smoothing procedure for Krylov subspace methods. Linear Algebra Appl. **268**(1), 131–149 (1998)

6. Heyouni, M.: The global Hessenberg and CMRH methods for linear systems with multiple right-hand sides. Numer. Alg. **26**, 317–332 (2001)
7. Heyouni, M., Sadok, H.: A new implementation of the CMRH method for solving dense linear systems. J. Comp. Appl. Math. **213**, 387–399 (2008)
8. Helsing, J.: Approximate inverse preconditioners for some large dense random electrostatic interaction matrices. BIT Numer. Math. **46**, 307–323 (2006)
9. Matrix Market: http://math.nist.gov/MatrixMarket/data/, Test Matrices Database Maintained by NIST, Gaithersburg, MD.
10. Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comp. **7**, 856–869 (1986)
11. Saad, Y.: Iterative methods for sparse linear systems, 2nd edn. SIAM, Philadelphia (2003)
12. Sadok, H.: CMRH: a new method for solving nonsymmetric linear systems based on the Hessenberg reduction algorithm. Numer. Algorithms **20**, 303–321 (1999)
13. Sadok, H., Szyld, D.B.: A new look at CMRH and its relation to GMRES. BIT Numer. Math. **52**, 485–501 (2012)