Journal of Hyperstructures 9 *(3rd CSC2019)* (*2020*), *62-74.*
ISSN: 2322-1666 print/2251-8436 online

AN EFFICIENT ITERATIVE METHOD FOR SOLVING LARGE LINEAR SYSTEMS

ALI JAMALIAN, HOSSEIN AMINIKHAH

Abstract. This paper presents a new powerful iterative method for solving large and sparse linear systems. Using the idea of the Jaya method to the restarted generalized minimum residual (GMRES) method, we propose the Jaya-GMRES method. The Jaya-GMRES is an efficient solver, being based mainly on matrix-vector multiplications. Numerical results show that the Jaya-GMRES method has found more accurate solutions and converges much regular than the GMRES method.

Key Words: GMRES method, Jaya optimization algorithm, Linear systems, Iterative method, Non-symmetric matrix.
2010 Mathematics Subject Classification: Primary: 65F10; Secondary: 90B40.

1. Introduction

Some numerical computations in physics, mechanics, chemistry, engineering, economics, finance, etc., involve numerical linear algebra, i.e., computations involving matrices. Many problems in applied sciences and engineering lead to linear systems of the form

$$Ax = b \tag{1.1}$$

where A is a $n \times n$ nonsingular real matrix, $b \in \mathrm{R}^n$ the right-hand side, and the vector x is the solution of the linear system [4]. In general, the linear systems are large, sparse and nonsymmetric. These systems can

Received: 14 August 2020, Accepted: 20 August 2020. Communicated by Behrouz Fathi-Vajargah;
∗Address correspondence to Ali Jamalian; E-mail: ali.jamalian@guilan.ac.ir
© 2020 University of Mohaghegh Ardabili.

be solved by making use of direct methods and iterative methods. Direct methods are not appropriate for solving linear system of equations with large matrices, especially for linear systems with large sparse matrix because of the magnitude of calculations and also due to the limited storages and speeds of the computers. Usually, iterative methods are used for solving these types of equations. Iterative methods in comparison with direct methods are more efficient for solving linear system of equations [13, 15]. The primary use of iterative methods is for computing the solution to large, sparse systems and for finding a few eigenvalues of a large sparse matrix. Along with other problems, such systems occur in the numerical solution of partial differential equations [2, 14]. In many cases, matrices have three, four or more diagonals, are block structured, where nonzero elements exist in blocks throughout the matrix, or have little organized structure. Algorithms for dealing with large, sparse matrices are an active area of research. A Krylov subspace-based method does not access the elements of the matrix directly, but rather performs matrix-vector multiplication to obtain vectors that are projections into a lower-dimensional Krylov subspace, where a corresponding problem is solved. The solution is then converted into a solution of the original problem. These methods can give a good result after a relatively small number of iterations [5].

In recent years, swarm intelligence-based algorithms and metaheuristic optimization methods are considered effective tools for solving optimization problems. Recently, a new global optimization method called Jaya has generated growing interest because of its simplicity and efficiency. The original Jaya algorithm has been introduced by Rao in 2016 [7, 8]. It is a population-based, derivative-free and specific parameterless algorithm for global optimization problems. The basic concept of this algorithm is moving the obtained solution toward the best solution and avoiding the worst one. It does not require any algorithm-specific parameter except the general control parameters. Jaya is applied to solve several single and multi-objective optimization problems. Moreover, improved versions of Jaya algorithm called as Elitist Jaya, Quasi-Oppositional Jaya, Self-Adaptive Jaya, Self-Adaptive Multi-Population Jaya, and Self-Adaptive Multi-Population Elitist Jaya, and Multi-Objective Quasi-Oppositional Jaya are developed and applied to solve several engineering optimization problems [8].

The cost of the full orthogonalization of the Krylov subspace is an important factor in convergence. To avoid this disadvantage, we can restart the

GMRES method, or we apply the incomplete orthogonalization[9, 10]. The convergence properties of the incomplete orthogonalization are not well understood. A difficulty with the restarted GMRES, at the time of the restart, is that some information is lost. This can be slow down the convergence. Morgan improved the restarted GMRES by reducing the ill effect of restarting [6].
The aim of this paper is to combine the GMRES algorithm with the Jaya optimization algorithm and to try to overcome local convergence with high accuracy. The rest of the paper is organized as follows: Section 2 presents a short overview of the Krylov subspaces and Arnoldi method, while an overview of the GMRES method for solving linear systems is presented in Sections 3. A brief description of the Jaya optimization method is given in Section 4. The proposed hybrid algorithm is described in Section 5, and in Section 6, numerical examples are used to illustrate the properties of the proposed method. Finally, we make some concluding remarks.

2. Krylov subspaces and Arnoldi Iteration

An iterative method for solving a linear system $Ax = b$, is an algorithm that starts with an initial guess x_0 for the solution and successively modifies that guess in an attempt to obtain improved approximate solutions $x_1, x_2, \ldots$. The residual at step m of an iterative method for solving $Ax = b$ is the vector $r_m = b - Ax_m$, where x_m is the approximate solution generated at step m. The initial residual is $r_0 = b - Ax_0$, where x_0 is the initial guess for the solution. The error at step m is the difference between the true solution $A^{-1}b$ and the approximate solution x_m i.e., $e_m = A^{-1}b - x_m$.
A Krylov space is a space of the form span $\left\{q, Aq, \ldots, A^{m-1}q\right\}$, where A is an $n \times n$ matrix and q is an $n \times 1$ vector. This space will be denoted as $\mathrm{K}_m(A, q)$. The Arnoldi iteration is a method for constructing an orthonormal basis for a Krylov space that requires saving all of the basis vectors and orthogonalizing against them at each step. This method was developed by Arnoldi [1] in 1951. We, know A complete reduction of A to Hessenberg form by an orthogonal similarity transformation might be written $A = QHQ^T$, or $AQ = QH$. However, in dealing with iterative methods we take the view that n is huge or infinite, so that computing the full reduction is out of the question. Instead, we consider the first m columns of $AQ = QH$. Let Q_m be the $n \times m$ matrix whose columns

are the first m columns of Q

$$Q_m = \begin{pmatrix} q_1 & q_2 & \dots & q_m \end{pmatrix},$$

and $\bar{H}_m$ be the $(m+1) \times m$ upper-left section of H, which is also a Hessenberg matrix

$$\bar{H}_m = \begin{pmatrix} h_{11} & & \cdots & h_{1m} \\ h_{21} & h_{22} & & \\ & \ddots & \ddots & \vdots \\ & & h_{m,m-1} & h_{mm} \\ & & & h_{m+1,m} \end{pmatrix} = \begin{pmatrix} H_m \\ h_{m+1,m} e_m^T \end{pmatrix}.$$

Then we have

$$AQ_m = Q_{m+1}\bar{H}_m. \tag{2.1}$$

The mth column of this equation can be written as follows:

$$Aq_m = h_{1m}q_1 + \cdots + h_{mm}q_m + h_{m+1,m}q_{m+1}. \tag{2.2}$$

In words, q_{m+1} satisfies an $(m+1)-$term recurrence relation involving itself and the previous Krylov vectors. The Arnoldi iteration is simply the modified Gram-Schmidt iteration that implements the equation (2.2). Figure 1 depicts the decomposition. Algorithm 1 specifies the Arnoldi process that generates an orthonormal basis $\{q_1, q_2, \dots, q_{m+1}\}$ for the subspace spanned by $K_{m+1}(A, x_1)$ and builds the matrix $\tilde{H}_m$.

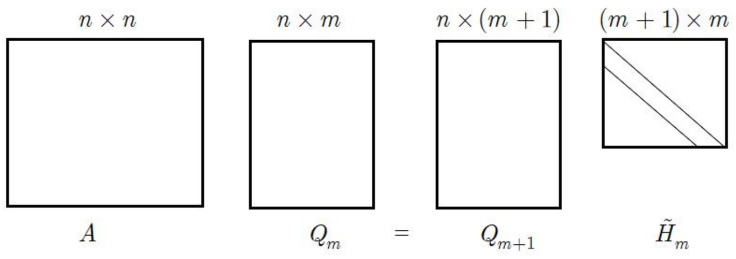

Figure 1. Arnoldi decomposition form 2

Using the Arnoldi modified Gram-Schmidt algorithm (MGS), we have the following theorems.

Algorithm 1 The Arnoldi MGS process

```
Input: $A \in \mathrm{R}^{n\times n}, x_0 \in \mathrm{R}^n$.
Output: Orthogonal basis $\{q_1, q_2, \ldots, q_{m+1}\}$ of $K_{m+1}(A, r_0)$ and an $(m+1)\times m$ matrix $\bar{H}_m$.
Initialization: $r_0 = b - Ax_0$.
$q_1 = r_0/\|r_0\|$
for $k = 1, 2, \ldots, m$ do
  $w_k = Aq_k$
  for $j = 1, \ldots, k$ do
    $h_{jk} = q_j^T w_k$
    $w_k = w_k - h_{jk}q_j$
  end for
  $h_{k+1,k} = \|w_k\|_2$
  if $h_{k+1,k} = 0$ then break
  else $q_{k+1} = \frac{w_k}{h_{k+1,k}}$
end for
```

Theorem 1. [11] The vectors $q_1, q_2, \ldots, q_m$ produced by the Arnoldi algorithm form an orthonormal basis of the subspace $\mathrm{K}_m = \mathrm{span}\left\{q_1, Aq_1, \ldots, A^{m-1}q_1\right\}$.

Theorem 2. [11] Denote by Q_ma $n \times m$ matrix with column vectors $q_1, q_2, \ldots, q_m$ and by H_m a $m \times m$ Hessenberg matrix whose nonzero entries are defined by the algorithm. Then, the following relations hold:

$$\begin{aligned}&AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^T,\\ &Q_m^T A Q_m \approx H_m.\end{aligned}$$

3. The GMRES method

Generalized Minimum Residual (GMRES) method [12] is to repeatedly approximate the solution of a linear system Eq. (1.1) by a vector $x_m \in x_0 + \mathrm{K}_m(A, r_0) = x_0 + Q_m y_m$, $y_m \in \mathrm{R}^m$ in the sequence of Krylov subspaces

$$\begin{aligned}&\mathrm{K}_m(A, b) = \mathrm{span}\left\{r_0, Ar_0, \ldots, A^{m-1}r_0\right\},\\ &r_0 = b - Ax_0,\end{aligned}$$

where the columns of Q_m are an $n-$dimensional orthogonal basis for the $\mathrm{K}_m(A, b)$ and x_0 is an initial guess for the solution of $Ax = b$. At step m, we solve a least squares problem to find the vector y_m that minimizes

$$\|r_m\|_2 = \|b - Ax_m\|_2 = \|b - A(x_0 + Q_m y_m)\|_2 = \|r_0 - AQ_m y_m\|_2 .$$

Solving this least squares problem directly in each iteration would be a numerically unstable procedure. It is therefore essential to create a new set of vectors that span the same space but have better numerical properties. This is what the Arnoldi iteration takes care of. In the Arnoldi method, we put $r_0 = b - Ax_0$. Now that we have an orthonormal basis $\{q_1, q_2, \ldots, q_m\}$ of $\mathrm{K}_m(A, r_0)$ and let Q_m be the orthogonal matrix with $q_1, q_2, \ldots, q_m$ as its columns. We can derive the minimization problem that defines GMRES. In the mth iterate, x_m can be written as $x_0+Q_m y_m$ for some $y_m \in \mathrm{R}^m$. So, for $x_m \in x_0 + \mathrm{K}_m(A, r_0)$, we have

$$\min_{x_m \in x_0 + \mathrm{K}_m(A,r_0)} \|b - Ax_m\|_2 = \min_{y \in \mathrm{R}^m} \|r_0 - AQ_m y_m\|_2 \,. \tag{3.1}$$

Using Eq. (2.2), we get $\|r_0 - AQ_m y_m\|_2 = \left\| r_0 - Q_{m+1}\tilde{H}_m y_m \right\|_2$. Since multiplying the vectors by Q_{m+1}^T does not change the norm, we get

$$\min_{y \in \mathrm{R}^m} \left\| Q_{m+1}^T (r_0 - Q_{m+1}\tilde{H}_m y_m) \right\|_2 = \min_{y \in \mathrm{R}^m} \left\| Q_{m+1}^T r_0 - \tilde{H}_m y_m \right\|_2 .$$

The product $Q_{m+1}^T r_0$ in this equation becomes $\beta = q_1^T r_0 = \frac{\|r_0\|_2^2}{\|r_0\|_2} = \|r_0\|_2$ and $q_k^T r_0 = \|r_0\| \, q_k^T q_1 = 0$ for all $k > 1$. Thus, we must minimize $\left\| \beta e_1 - \bar{H}_m y_m \right\|_2$. We use the QR decomposition approach to solving overdetermined least-squares problems. Since $\bar{H}_m$ in $\bar{H}_m y_m = \beta e_1$ is an upper Hessenberg matrix, the QR decomposition can be done by Givens rotation. Algorithm 2 presents the GMRES method.

In the practical implementation of GMRES, one estimate x_m is often not sufficient to obtain the error tolerance desired. The restarted GMRES algorithm, is defined by simply restarting GMRES using the latest iterate as the initial guess for the next GMRES iteration. Sometimes partial information from the previous GMRES cycle is retained and used after the restart.

4. Jaya optimization algorithm

Consider a special class of optimization problems with bounded variables in the form of

$$\begin{array}{ll} Min & f(x) \\ s.t. & l_k \leq x_k \leq u_k \quad k = 1, \ldots, m. \end{array} \tag{4.1}$$

$f(x)$ is the objective function to be minimized with a m dimensional variable x and l, u are lower bound and upper bound of x, respectively.

Algorithm 2 GMRES process

```
  **Input:** $A \in \mathrm{R}^{n\times n}, b \in \mathrm{R}^n, x_0 \in \mathrm{R}^n, \varepsilon > 0.$
  **Output:** Approximate solution of $Ax = b$.
  **Initialization:** $r_0 = b - Ax_0$.
  $\beta = \|r_0\|_2$
  $q_1 = r_0/\beta$
  $\xi = (\beta, 0, \dots, 0)^T$
  for $k = 1, 2, \dots$ do
    $w_k = Aq_k$
    for $l = 1, \dots, k$ do
      $h_{lk} = q_l^T w_k$
      $w_k = w_k - h_{lk} q_l$
    end for
    $h_{k+1,k} = \|w_k\|_2$
    for $i = 1, 2, \dots, k-1$ do
      $\begin{pmatrix} h_{ik} \\ h_{i+1,k} \end{pmatrix} = \begin{pmatrix} c_i & s_i \\ -s_i & c_i \end{pmatrix} \begin{pmatrix} h_{ik} \\ h_{i+1,k} \end{pmatrix}$
    end for
    $\gamma = \sqrt{h_{kk}^2 + h_{k+1,k}^2},\ c_k = h_{kk}/\gamma,\ s_k = c_k h_{k+1,k}/h_{kk}$
    $\begin{pmatrix} \xi_k \\ \xi_{k+1} \end{pmatrix} = \begin{pmatrix} c_k & s_k \\ -s_k & c_k \end{pmatrix} \begin{pmatrix} \xi_k \\ 0 \end{pmatrix}$
    $h_{kk} = \gamma,\ h_{k+1,k} = 0$
    if $|\xi_{k+1}| < \varepsilon$ then
      for $l = k, k-1, \dots, 1$ do
        $y_l = \frac{1}{h_{ll}} \left( \xi_l - \sum_{i=l+1}^{k} h_{li} y_i \right)$
      end for
      $x_k = x_0 + \sum_{i=1}^{k} y_k q_k$
      break
    else
      $q_{k+1} = \frac{w_k}{h_{k+1,k}}$
    end if
  end for
```

The basic concept of Jaya algorithm is moving the obtained solution toward the best solution and avoiding the worst one. It does not require any algorithm-specific parameter except the general control parameters [8]. We assume that population size of candidate solutions is n and the best candidate obtains the best value of $f(x)$ (i.e. $f(x)_{best}$) in the

Algorithm 3 restarted GMRES

```
**Input:** $A \in \mathrm{R}^{n\times n}, b \in \mathrm{R}^n, x_0 \in \mathrm{R}^n, j \in \mathrm{N}, \varepsilon > 0$.
**Output:** Approximate solution of $Ax = b$.
**Initialization:** $r_0 = b - Ax_0$.
Set $\beta = \|r_0\|_2$
while $\beta > \varepsilon$ do
    Compute the GMRES approximation $x_j$ using Algorithm 2
    Set $x_0 = x_j,\ r_0 = b - Ax_0,\ \beta = \|r_0\|_2$
end while
```

entire candidate solutions and the worst candidate obtains the worst value of $f(x)$ (i.e. $f(x)_{worst}$) in the entire candidate solutions. If $x_{j,k,i}$ is the value of jth variable for kth candidate during ith iteration, then the corresponding value of a new solution for $j = 1, \ldots, m$, $k = 1, \ldots, n$, $i = 1, \ldots, MaxIt$, is generated by

$$y_{j,k,i} = x_{j,k,i} + r_{1,j,i}(x_{j,b,i} - |x_{j,k,i}|) - r_{2,j,i}(x_{j,w,i} - |x_{j,k,i}|), \tag{4.2}$$

where, $x_{j,b,i}$ is the value of the jth variable for the best candidate during ith iteration and $x_{j,w,i}$ is the value of the jth variable for the worst candidate during ith iteration. $y_{j,k,i}$ is the updated value of $x_{j,k,i}$ and $r_{1,j,i}$ and $r_{2,j,i}$ are the two random numbers in the range $[0, 1]$. The term $r_{1,j,i}(x_{j,b,i} - |x_{j,k,i}|)$ indicates the tendency of the solution to move closer to the best solution and the term $r_{2,j,i}(x_{j,w,i} - |x_{j,k,i}|)$ indicates the tendency of the solution to avoid the worst solution. $y_{j,k,i}$ is accepted if it gives better function value. All the accepted function values at the end of each iteration are maintained and these values are used as input for the next iteration. Algorithm 4 shows the steps of the Jaya optimization method.

5. The Jaya-GMRES algorithm

A difficulty that arises when applying the restarted GMRES method to solve large linear system of equations is that the method may diverge before to get a good approximation to the solution or after a number of iterations it may finds solutions but with approximation which is not good one, because it depends on the initial vector. In Jaya-GMRES algorithm we apply Jaya optimization algorithm as an improvement procedure to find better solutions. The proposed algorithm of Jaya-GMRES is given in Algorithm 5.

Algorithm 4 Jaya optimization method

```
Input parameters: choose values of $m, n$.
Generate the initial $n$ candidate solutions with dimension of $m$, randomly.
Evaluate each candidate solution with fitness function.
Identify the best and the worst solutions.
while termination criteria not reached do
  for each candidate solution do
    Generate the new solutions using equation (4.2)
    Evaluate the new solution by the fitness function
    if the new solution is better than the old one then
      Update the solution with the new one.
    end if
  end for
  Update the best and the worst solutions.
end while
Output optimal solution and optimal objective values.
```

In this algorithm we store the solution of system, x, and its fitness value $f(x)$ of each individual. We define the fitness function as

$$f(x) = \|Ax - b\|. \tag{5.1}$$

For termination criteria of the process at iteration *iter*, we can use the following stopping criterion:

$$\textbf{if}\,(f(x_{best}) \leq \varepsilon \ \ \textbf{or} \ \ iter > MaxIter) \ \ \textbf{Then} \ \text{stop} \tag{5.2}$$

where ε is an arbitrary value. In addition, another termination criterion might be used for stopping the process when the best solution is not changed for a certain number of iterations.

6. Numerical experiments

In this section, we will show some numerical examples in order to compare Jaya-GMRES algorithm and restarted GMRES algorithms. The experiments tested here are from the Matrix Market Web server. We have run the algorithms using MATLAB R2018b on a machine with Intel CORE i7 2.5 GHz of CPU and with 8 GB of RAM and machine precision 2.22×10^{-16}.

In all experiments, the right-hand side is a random vector with entries uniformly distributed in the interval $[0, 1]$. The initial guess for

Algorithm 5 Jaya-GMRES method

```
Input parameters: choose values of $m, n$.
Generate the initial $n$ individual with dimension of $m$, randomly.
for each individual do
    Run restarted-GMRES()
    Update the individual
    Evaluate individual with the fitness function
end for
Identify the best and the worst individuals.
while termination criteria not met do
    for each individual do
        Generate the new solutions using equation (4.2)
        Run restarted-GMRES() for the new solution
        Update the new solution
        Evaluate the new solution by the fitness function
        if the new solution is better than the old one then
            Update the individual with the new one.
        end if
    end for
    Update the best and the worst solutions.
end while
Output best solution of $Ax = b$ with least residual.
```

restarted GMERS is $x_0 = (0, \cdots, 0)$ and initial individuals of Jaya-GMRES choosed randomly. The termination criteria include both maximum number of iterations ($MaxIter$) and accuracy level. Hence, the Jaya-GMRES continues until

$$iter \leq MaxIter_{Jaya} \ \textbf{and} \ \|Ax_{best} - b\|_2 \leq tol,$$

where, *best* stands for index of the best agent in all iterations and $MaxIter_{Jaya}$ is maximum number of iterations of the Jaya-GMRES. As well, the restarted GMRES method iterates until

$$iter \leq MaxIter_{GMRES} \ \textbf{and} \ \left\|Ax^{(iter)} - b\right\|_2 \leq tol,$$

where, $x^{(iter)}$ stands for the solution generated by GMRES method at iteration *iter* and $MaxIter_{GMRES}$ is maximum number of iterations for the GMRES method. In all experiments, the population size, the

$MaxIter_{Jaya}$ and the $MaxIter_{GMRES}$, are set as 5, 300, 30, respectively.

Example 1. *The matrix add20.* This is a 2395×2395 matrix from the HAMM group, from the independent sets and generators of the Matrix Market, with 17319 nonzero entries. The estimated condition number is 1.76×10^4. This example was tested with various values of m. The Jaya-GMRES method was always better than restarted GMRES and MATLAB operator (\). In order to illustrate this example, Table 1 gives the obtained results with $tol = 10^{-24}$.

Table 1. Comparing results of Jaya-GMRES, restarted GMRES and MATLAB for the matrix *add20*

m	Jaya-GMRES		restarted GMRES		MATLAB operator	
	Error	CPU	Error	CPU	Error	CPU
2	**1.45e-24**	1.36	3.6636e-15	0.050305	2.3754e-24	0.0079786
3	**1.59e-24**	0.0434974	3.6636e-15	0.0762023	2. 3754e -24	0.01232
4	**1.07e-24**	0.06441310	3.6636e-15	0.0562137	2. 3754e -24	0.0083116
5	**1.38e-24**	1.3657489	3.6636e-15	0.0507335	2. 3754e -24	0.0073797
10	**9.72e-25**	0.1149018	3.6636e-15	0.0699647	2. 3754e -24	0.0081065
15	**1.44e-24**	1.4879709	3.6636e-15	0.0616817	2. 3754e -24	0.0085716
20	**1.04e-24**	0.2812876	3.6636e-15	0.0626403	2. 3754e -24	0.011704

Example 2. *The matrix memplus.* This is a 17758×17758 matrix from the set HAMM, from the independent sets and generators of the Matrix Market with 126150 nonzero entries. The estimated condition number is 2.67×10^5. The results are given in Table 2, with $tol = 10^{-24}$.

Table 2. Comparing results of Jaya-GMRES, restarted GMRES and MATLAB for the matrix *memplus*

m	Jaya-GMRES		restarted GMRES		MATLAB operator	
	Error	CPU	Error	CPU	Error	CPU
2	**1.5914e-25**	14.4467814	5.1034e-14	0.4582524	1.6155e-25	0.0834938
5	**1.5692e-25**	15.541664	5.1034e-14	0.4264089	1.6155e-25	0.0966378
10	**1.5882e-25**	876.2163249	5.1034e-14	0.4692447	1.6155e-25	0.0952062
15	**1.5202e-25**	1314.4889132	5.1034e-14	0.5116114	1.6155e-25	0.0841371
30	**1.5561e-25**	3304.7236516	5.1034e-14	0.5020445	1.6155e-25	0.0938476

7. Conclusions

In this paper, we proposed a novel and powerful method, called Jaya-GMRES, for solving large linear system of equations. The Jaya-GMRES is based on the original Jaya optimization algorithm, a population-based metaheuristic introduced recently for optimization problems and has attracted researchers' attention because of simple implementation and computational efficiency. The Jaya-GMRES employs restarted GMRES method and it enhances quality of obtained solution from GMRES, based on optimization process of the Jaya algorithm. To verify the efficiency of the Jaya-GMRES algorithm, some test problems were evaluated. The performance of the Jaya-GMRES algorithm was compared with restarted GMRES method and MATLAB operator (back-slash). Numerical results indicate that the Jaya-GMRES algorithm provides superior performance.

References

[1] W. E. Arnoldi, *The principle of minimized interations in the solution of the matrix eigenvalue problem*, Quarterly of applied mathematics, **9 (1)** (1951) 17-29.

[2] P. G. Ciarlet, *Introduction to Numerical Linear Algebra and Optimisation*, Cambridge Texts in Applied Mathematics, Cambridge University Press, Cambridge, (1989).

[3] I. S. Duff, R. G. Grimes and J. G. Lewis, *User's guide for the Harwell–Boeing sparse matrix collection (Release I)*, Technical Report TR/PA/92/86, CERFACS, Toulouse, France (1992).

[4] W. Ford, *Numerical Linear Algebra with Applications*, Academic Press, USA, (2015).

[5] L. Hogben, *Handbook of linear algebra*, Chapman & Hall/CRC, New York, USA, (2007).

[6] R. B. Morgan, *A restarted GMRES method augmented with eigenvectors*, SIAM Journal on Matrix Analysis and Applications, **16 (4)** (1995) 1154-1171.

[7] R. Rao, *Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems*, International Journal of Industrial Engineering Computations, **7 (1)** (2016) 19-34.

[8] R. V. Rao, *Jaya: an advanced optimization algorithm and its engineering applications*, Cham: SpringerInternational Publishing, (2019).

[9] Y. Saad, *Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices*, Linear algebra and its applications, **34** (1980) 269-295.

[10] Y. Saad, *Krylov subspace methods for solving large unsymmetric linear systems*, Mathematics of computation, **37 (155)** (1981) 105-126.

[11] Y. Saad, *Iteractive Method for Sparse Linear Systems*, PWS Publishing Company, a division of International Thomson Publishing Inc., USA, (1996).

[12] Y. Saad, Schultz, Martin, H. *GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems*, SIAM Journal on scientific and statistical computing, **7** (1986) 856-869.

[13] L. N. Trefethen and D. Bau, *Numerical Linear Algebra*, SIAM, (1997).

[14] D. S. Watkins, *Fundamentals of Matrix Computations*, third ed.,Wiley, Hoboken, NJ, (2010).

[15] H. Wendland, *Numerical Linear Algebra*, Cambridge Texts in Applied Mathematics. Cambridge University Press, (2018).

Ali Jamalian
Department of Computer Science, University of Guilan, P.O.Box 41335-19141, Rasht, Iran
Email: ali.jamalian@guilan.ac.ir, a.jamalian.math@gmail.com

Hossein Aminikhah
Department of Applied Mathematics and Department of Computer Science, University of Guilan, P.O.Box 41335-19141, Rasht, Iran
Center of Excellence for Mathematical Modelling, Optimization and Combinational Computing (MMOCC), University of Guilan, P.O.Box 41938-19141, Rasht, Iran.
Email: aminikhah@guilan.ac.ir