Scientific Research Publishing

Advances in Linear Algebra & Matrix Theory, 2022, 12, 24-38
https://www.scirp.org/journal/alamt
ISSN Online: 2165-3348
ISSN Print: 2165-333X

A Compact Heart Iteration for Large Eigenvalues Problems

Achiya Dax

Hydrological Service of Israel, Jerusalem, Israel
Email: dax20@water.gov.il

How to cite this paper: Dax, A. (2022) A Compact Heart Iteration for Large Eigenvalues Problems. *Advances in Linear Algebra & Matrix Theory*, **12**, 24-38.
https://doi.org/10.4236/alamt.2022.121002

Received: February 7, 2022
Accepted: March 11, 2022
Published: March 14, 2022

Abstract

In this paper, we present a compact version of the Heart iteration. One that requires less matrix-vector products per iteration and attains faster convergence. The Heart iteration is a new type of Restarted Krylov methods for calculating peripheral eigenvalues of symmetric matrices. The new framework avoids the Lanczos tridiagonalization process and the use of implicit restarts. This simplifies the restarting mechanism and allows the introduction of several modifications. Convergence is assured by a monotonicity property that pushes the computed Ritz values toward their limits. Numerical experiments illustrate the usefulness of the proposed approach.

Keywords

Large Sparse Matrices, Restarted Krylov Methods, Exterior Eigenvalues, Symmetric Matrices, Monotonicity, Starting Vectors

1. Introduction

The Heart iteration is a new type of Restarted Krylov methods. Given a symmetric matrix $G \in \mathbb{R}^{n\times n}$, the method is aimed at calculating a cluster of k exterior eigenvalues of G. As other Krylov methods it is best suited for handling large sparse matrices in which a matrix-vector product needs only $0(n)$ flops. Another underlying assumption is that the number of computed eigenvalues, k, is much smaller than n. The use of restarted Krylov methods for solving such problems was considered by several authors, e.g., [1]-[22]. Most of these methods are based on a Lanczos tridiagonalization algorithm in which the starting vector is determined by an implicit restart process. The Heart iteration is not using these tools. It is based on Gram-Schmidt orthogonalization and a simple intuitive choice of the starting vector. This results in a simple iteration that allows several

modifications. Convergence is assured by a monotonicity property that pushes the computed eigenvalues toward their limits.

The main idea behind the new method is clarified by inspecting its basic iteration. Below we concentrate on the largest eigenvalues, but the algorithm can compute any cluster of k exterior eigenvalues. Let the eigenvalues of G be sorted to satisfy

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n. \tag{1.1}$$

Then the term "exterior eigenvalues" refers to the k largest eigenvalues, the k smallest eigenvalues, or any set of k eigenvalues that is combined from a number of the largest eigenvalues plus a number of the smallest ones. Other names for such eigenvalues are "peripheral eigenvalues" and "extreme eigenvalues".

Note that although the above definitions refer to clusters of eigenvalues, the algorithm is carried out by computing the corresponding k eigenvectors of G. The subspace that is spanned by these eigenvectors is called the **target space**.

The basic Heart iteration

The qth iteration, $q = 0,1,2,\cdots$, is composed of the following five steps. The first step starts with a matrix $V_q \in \mathbb{R}^{n\times k}$ that contains "old" information on the target space, a matrix $Y_q \in \mathbb{R}^{n\times \ell}$ that contains "new" information, and a matrix $X_q = \left[V_q, Y_q\right] \in \mathbb{R}^{n\times(k+\ell)}$ that includes all the known information. The matrix X_q has $p = k + \ell$ orthonormal columns. That is

$$X_q^{\mathrm{T}} X_q = I \in \mathbb{R}^{p\times p}.$$

(Typical values for ℓ lie between k to $2k$.)

Step 1: Eigenvalues extraction. Given the Rayleigh quotient matrix

$$S_q = X_q^{\mathrm{T}} G X_q,$$

compute the k largest eigenvalues of S_q. The corresponding k eigenvectors of S_q are assembled in a matrix

$$U_q \in \mathbb{R}^{p\times k}, \quad U_q^{\mathrm{T}} U_q = I \in \mathbb{R}^{k\times k},$$

which is used to compute the related matrix of Ritz vectors,

$$V_{q+1} = X_q U_q.$$

Note that both X_q and U_q have orthonormal columns and V_{q+1} inherits this property.

Step 2: Collect new information. Compute a Krylov matrix $B_q \in \mathbb{R}^{n\times \ell}$ that contains new information on the target space.

Step 3: Orthogonalize the columns of B_q against the columns of V_{q+1}. There are several ways to achieve this task. In exact arithmetic, the resulting matrix, Z_q, satisfies the Gram-Schmidt formula

$$Z_q = B_q - V_{q+1}\left(V_{q+1}^{\mathrm{T}} B_q\right).$$

Step 4: Build an orthonormal basis of Range (Z_q). Compute a matrix,

$$Y_{q+1} \in \mathbb{R}^{n\times \ell}, \quad Y_{q+1}^{\mathrm{T}} Y_{q+1} = I \in \mathbb{R}^{\ell\times \ell},$$

whose columns form an orthonormal basis of Range (Z_q). This can be done by a QR factorization of Z_q. (If rank (Z_q) is smaller than ℓ, then ℓ is temporarily reduced to be rank (Z_q).)

Step 5: Define X_{q+1} by the rule

$$X_{q+1} = \left[V_{q+1}, Y_{q+1} \right],$$

which ensures that

$$X_{q+1}^{\mathrm{T}} X_{q+1} = I \in \mathbb{R}^{p \times p}.$$

Then compute the new Rayleigh quotient matrix

$$S_{q+1} = X_{q+1}^{\mathrm{T}} G X_{q+1}.$$

This matrix will be used at the beginning of the next iteration.

At this point we are not concerned with efficiency issues, and the above description is mainly aimed at clarifying the purpose of each step. (A more effective scheme is proposed in Section 4.) The name "**Heart iteration**" comes from the similarity to the heart's systole-diastole cardiac cycle: Step 1 achieves subspace contraction (eigenvalues extraction), while in Steps 2 - 5 the subspace expands (collecting new information).

The plan of the paper is as follows. The monotonicity property that motivates the new method is established in the next section. Then, in Section 3, we describe a simple Krylov subspace process that constructs B_q. The aim of this paper is to present an efficient implementation of the Heart iteration. The new iteration combines Steps 2 - 5 into one "compact" step. This results in a simple effective algorithm that uses less matrix-vector products per iteration. The details of the new iteration are given in Section 4. The paper ends with numerical experiments that illustrate the usefulness of the proposed method.

2. The Monotonicity Property

In this section we establish a useful property of the proposed method. The proof can be found in former presentations of the Heart iteration, e.g., [5] [6] [7] [8]. Yet, in order to make this paper self-contained, we provide the proof. The main argument is based on the following well-known interlacing theorems, e.g., [11] [15] [23].

Theorem 1 (Cauchy interlace theorem) *Let* $G \in \mathbb{R}^{n \times n}$ *be a symmetric matrix with eigenvalues*

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n. \tag{2.1}$$

Let the symmetric matrix $H \in \mathbb{R}^{k \times k}$ be obtained from G by deleting $n-k$ rows and the corresponding $n-k$ columns. Let

$$\eta_1 \geq \eta_2 \geq \cdots \geq \eta_k \tag{2.2}$$

denote the eigenvalues of H. Then

$$\lambda_j \geq \eta_j \quad \text{for } j = 1, \cdots, k, \tag{2.3}$$

and

$$\eta_{k+1-i} \geq \lambda_{n+1-i} \quad \text{for } i = 1, \cdots, k. \tag{2.4}$$

In particular, for $k = n-1$ we have the interlacing relations

$$\lambda_1 \geq \eta_1 \geq \lambda_2 \geq \eta_2 \geq \lambda_3 \geq \cdots \geq \lambda_{n-1} \geq \eta_{n-1} \geq \lambda_n. \tag{2.5}$$

Corollary 2 (Poincaré separation theorem) *Let the matrix $V \in \mathbb{R}^{n\times k}$ have k orthonormal columns. That is $V^{\mathrm{T}}V = I \in \mathbb{R}^{k\times k}$. Let the matrix $H = V^{\mathrm{T}}GV$ have the eigenvalues* (2.2)*. Then the eigenvalues of H and G satisfy* (2.3) *and* (2.4)*.*

The last observation enables us to prove the following monotonicity property.

Theorem 3. *Consider the qth iteration of the new method, $q = 1, 2, 3, \cdots$. Assume that the eigenvalues of G satisfy* (2.1) *and let the eigenvalues of the matrix*

$$S_q = X_q^{\mathrm{T}} G X_q = \left[V_q, Y_q\right]^{\mathrm{T}} G \left[V_q, Y_q\right]$$

be denoted as

$$\lambda_1^{(q)} \geq \lambda_2^{(q)} \geq \cdots \geq \lambda_k^{(q)} \geq \cdots \geq \lambda_p^{(q)}.$$

Then the inequalities

$$\lambda_j \geq \lambda_j^{(q)} \geq \lambda_j^{(q-1)} \tag{2.6}$$

hold for $j = 1, \cdots, k$ and $q = 1, 2, 3, \cdots$.

Proof: The Ritz values which are computed at Step 1 are

$$\lambda_1^{(q)} \geq \lambda_2^{(q)} \cdots \geq \lambda_k^{(q)},$$

and these values are the largest eigenvalues of the matrix

$$S_q = X_q^{\mathrm{T}} G X_q.$$

Similarly,

$$\lambda_1^{(q-1)} \geq \lambda_2^{(q-1)} \geq \cdots \geq \lambda_k^{(q-1)},$$

are eigenvalues of the matrix

$$V_q^{\mathrm{T}} G V_q.$$

Therefore, since the columns of V_q are the first k columns of X_q,

$$\lambda_j^{(q)} \geq \lambda_j^{(q-1)} \quad \text{for } j = 1, \cdots, k,$$

while a further use of Corollary 2 gives

$$\lambda_j \geq \lambda_j^{(q)} \quad \text{for } j = 1, \cdots, k.$$

Hence by combining these relations we obtain (2.6). □

The treatment of other exterior clusters is done in a similar way. Assume for example that the algorithm is aimed at computing the k smallest eigenvalues of G,

$$\left\{\lambda_{n+1-k}, \cdots, \lambda_{n-1}, \lambda_n\right\}.$$

Then similar arguments show that

$$\lambda_{p+1-i}^{(q-1)} \geq \lambda_{p+1-i}^{(q)} \geq \lambda_{n+1-i} \tag{2.7}$$

for $i=1,\cdots,k$, and $q=1,2,3,\cdots$.

The proof of Theorem 3 emphasizes the importance of the orthonormality relations, and provides the motivation behind the basic iteration. Moreover, since orthonormality ensures monotonicity, it is not essential to construct B_q by applying the Lanczos algorithm. This consequence is used in the next sections.

3. The Basic Krylov Matrix

The basic Krylov information matrix has the form

$$B_q=[\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots,\boldsymbol{b}_\ell]\in\mathbb{R}^{n\times\ell}, \tag{3.1}$$

where the sequence $\boldsymbol{b}_1,\boldsymbol{b}_2,\cdots$, is initialized by the starting vector $\boldsymbol{b}_0$. The ability of a Krylov subspace to approximate a dominant subspace is characterized by the Kaniel-Paige-Saad bounds (See, for example, [10]: pp. 552-554; [15]: pp. 242-247; [16]: pp. 147-151; [18]: pp. 272-274), and the references therein. One consequence of these bounds regards the angle between $\boldsymbol{b}_1$ and the dominant subspace: The smaller the angle, the better approximation we get. This suggests that $\boldsymbol{b}_0$ should be defined as the sum of the current Ritz vectors. That is,

$$\boldsymbol{b}_0=V_{q+1}\boldsymbol{e} \tag{3.2}$$

where $\boldsymbol{e}=(1,1,\cdots,1)^{\mathrm{T}}\in\mathbb{R}^k$ is a vector of ones. (If some of the Ritz vectors have already converged then it is possible to remove these vectors from the sum.) Note that there is no point in setting $\boldsymbol{b}_1=V_{q+1}\boldsymbol{e}$, since in the next step B_q is orthogonalized against V_{q+1}.

The other columns of B_q are obtained by a Krylov process that resembles Lanczos' algorithm but uses direct orthogonalization. Let $\boldsymbol{r}\in\mathbb{R}^n$ be a given vector and let $\boldsymbol{q}\in\mathbb{R}^n$ be a unit length vector. That is $\|\boldsymbol{q}\|_2=1$ where $\|\cdot\|_2$ denotes the Euclidean vector norm. Then the statement "orthogonalize $\boldsymbol{r}$ against $\boldsymbol{q}$" is carried out by replacing $\boldsymbol{r}$ with $\boldsymbol{r}-(\boldsymbol{r}^{\mathrm{T}}\boldsymbol{q})\boldsymbol{q}$. Similarly, the statement "normalize $\boldsymbol{r}$" is carried out by replacing $\boldsymbol{r}$ with $\boldsymbol{r}/\|\boldsymbol{r}\|_2$. With these conventions at hand the construction of the vectors $\boldsymbol{b}_0,\boldsymbol{b}_1,\cdots,\boldsymbol{b}_\ell$, is carried out as follows.

The preparations part

1) Compute the starting vector:

$$\boldsymbol{b}_0=V_{q+1}\boldsymbol{e}\big/\|V_{q+1}\boldsymbol{e}\|_2 \tag{3.3}$$

2) Compute $\boldsymbol{b}_1$: Set $\boldsymbol{b}_1=G\boldsymbol{b}_0$.
Orthogonalize $\boldsymbol{b}_1$ against $\boldsymbol{b}_0$.
Normalize $\boldsymbol{b}_1$.

3) Compute $\boldsymbol{b}_2$: Set $\boldsymbol{b}_2=G\boldsymbol{b}_1$.
Orthogonalize $\boldsymbol{b}_2$ against $\boldsymbol{b}_0$.
Orthogonalize $\boldsymbol{b}_2$ against $\boldsymbol{b}_1$.
Normalize $\boldsymbol{b}_2$.

The iterative part

For $j=3,\cdots,\ell$, compute $\boldsymbol{b}_j$ as follows:

Set $\boldsymbol{b}_j = G\boldsymbol{b}_{j-1}$.
Orthogonalize $\boldsymbol{b}_j$ against $\boldsymbol{b}_{j-2}$.
Orthogonalize $\boldsymbol{b}_j$ against $\boldsymbol{b}_{j-1}$.
Normalize $\boldsymbol{b}_j$.

The direct orthogonalization that we use differs from Lanczos' algorithm and, therefore, fails to achieve a reduction of G into a tridiagonal form (The difference lies in the term that connects $\boldsymbol{b}_j$ with $\boldsymbol{b}_{j-2}$). It is also important to note that although $\boldsymbol{b}_0$ is defined in an "explicit" way, there is a major difference between our method and former explicitly restarted Krylov methods. That is, in Steps 3 and 4 the Krylov matrix B_q is orthogonalized against V_{q+1} and the resulting matrix, Z_q, is used to construct an orthonormal extension of V_{q+1}. This important ingredient is missing in the former explicit methods.

4. A Compact Version of the Heart Iteration

One feature that characterizes the basic Heart iteration is a direct computation of the Rayleigh quotient matrix

$$S_q = X_q^{\mathrm{T}} G X_q. \tag{4.1}$$

In this section we describe a compact version of the Heart iteration that avoids this computation. Instead S_q is computed "on the fly", as a by-product of the expanding process. The main idea is that the Krylov process in Step 2 and the orthogonalization in Steps 3 - 4 can be combined into one process. Moreover, observe that the columns of S_q have the form

$$X_q^{\mathrm{T}}\left(G\boldsymbol{x}_j\right), \quad j = 1, \cdots, p, \tag{4.2}$$

where $\boldsymbol{x}_j$ denotes the jth column of X_q. Hence the vector $G\boldsymbol{x}_j$ can be used both to construct S_q and to expand the Krylov subspace.

The contraction part remains unchanged. As before, it ends by computing a Ritz vectors matrix

$$V \in \mathbb{R}^{n\times k}, \quad V^{\mathrm{T}}V = I \in \mathbb{R}^{k\times k}, \tag{4.3}$$

that satisfies

$$V^{\mathrm{T}} G V = D, \tag{4.4}$$

where D is a $k \times k$ diagonal matrix whose diagonal entries are the computed Ritz values (The iteration subscripts are removed to ease the description).

The expansion process starts with the matrices

$$X = V$$

and

$$S = D.$$

Then the Krylov sequence begins with the vector

$$\boldsymbol{z} = G\left(V\boldsymbol{e}\right),$$

and performs ℓ steps. The jth step, $j = k+1, \cdots, k+\ell$, begins with the matrix

$$X = \left[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{j-1} \right] \in \mathbb{R}^{n \times (j-1)} \tag{4.5}$$

and ends with the matrix

$$X = \left[\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{j-1}, \boldsymbol{x}_j \right] \in \mathbb{R}^{n \times j}. \tag{4.6}$$

The vector $\boldsymbol{z} = G\boldsymbol{x}_j$ serves two purposes: To start the computation of $\boldsymbol{x}_{j+1}$, and to extend the Rayleigh quotient matrix. The vector $\boldsymbol{x}_{j+1}$ is obtained by orthogonalizing $\boldsymbol{z}$ against the columns of X. When using Gram-Schmidt orthogonalization this is achieved by replacing $\boldsymbol{z}$ with the vector $\boldsymbol{z} - X\left(X^{\mathrm{T}}\boldsymbol{z}\right)$. Below we denote this operation as

$$\boldsymbol{z} = \boldsymbol{z} - X\left(X^{\mathrm{T}}\boldsymbol{z}\right). \tag{4.7}$$

In practice one use of (4.7) is not sufficient to maintain orthogonality, so we need to repeat this operation.

The building of the Rayleigh quotient matrix

$$S_{q+1} = X_{q+1}^{\mathrm{T}} G X_{q+1} \tag{4.8}$$

is based on the following observations. At the beginning of the jth step, $j = k+1, \cdots, k+\ell$, the matrix X has the form (4.5) and the matrix

$$S = X^{\mathrm{T}} G X \tag{4.9}$$

is a $(j-1)\times(j-1)$ principal submatrix of S_{q+1}. At the end of the jth step X has the form (4.6) and the matrix (4.9) is a $j \times j$ principal submatrix of S_{q+1}. The "new" entries of S are $s_{ij} = s_{ji}$, $i = 1, \cdots, j$, and these entries are obtained from the vector

$$\boldsymbol{r} = \left(r_1, \cdots, r_j\right)^{\mathrm{T}} = X^{\mathrm{T}}\left(G\boldsymbol{x}_j\right) = X^{\mathrm{T}}\boldsymbol{z}. \tag{4.10}$$

A further saving is gained by noting that $\boldsymbol{r}$ can be used in the orthogonalization of $\boldsymbol{z}$ against the columns of X. The expansion process ends with a matrix X that has $k+\ell$ orthonormal columns, and the related Rayleigh quotient matrix

$$S = X^{\mathrm{T}} G X. \tag{4.11}$$

These matrices are the input for the next iteration (As before, we omit indices to ease the notation).

The compact Heart iteration

Part I: Contraction

Compute the k largest eigenvalues of S and the corresponding eigenvectors. The eigenvalues construct a diagonal matrix, $D \in \mathbb{R}^{k \times k}$. The eigenvectors are assembled into the matrix

$$U \in \mathbb{R}^{(k+\ell)\times k}, \quad U^{\mathrm{T}}U = I, \tag{4.12}$$

which is used to compute the related matrix of Ritz vectors

$$V = XU \in \mathbb{R}^{n \times k}. \tag{4.13}$$

Since both X and U have orthonormal columns the matrix V inherits this property.

Part II: Expansion

First set

$$X = V, \tag{4.14}$$

$$S = D, \tag{4.15}$$

$$\boldsymbol{z} = G(V\boldsymbol{e}), \tag{4.16}$$

and

$$\boldsymbol{r} = X^{\mathrm{T}}\boldsymbol{z}. \tag{4.17}$$

Then for $j = k+1, \cdots, k+\ell$, do as follows.

Gram-Schmidt orthogonalization: Set $\boldsymbol{z} = \boldsymbol{z} - X\boldsymbol{r}$.

Gram-Schmidt reorthogonalization: Set $\boldsymbol{z} = \boldsymbol{z} - X\left(X^{\mathrm{T}}\boldsymbol{z}\right)$.

Normalize $\boldsymbol{z}$.

Expand X to be $[X, \boldsymbol{z}]$.

Set $\boldsymbol{z} = G\boldsymbol{z}$.

Expand S by computing the vector

$$\boldsymbol{r} = \left(r_1, \cdots, r_j\right)^{\mathrm{T}} = X^{\mathrm{T}}\boldsymbol{z}$$

and setting the new entries of S to be

$$s_{ij} = s_{ji} = r_i \quad \text{for } i = 1, \cdots, j.$$

The main feature that characterizes this scheme is its simplicity. Furthermore, now each iteration needs only $\ell+1$ matrix-vector products.

5. The Initial Orthonormal Matrix

To start the Heart iteration we need to supply an "initial" orthonormal matrix, $X_0 \in \mathbb{R}^{n\times p}$, and the corresponding Rayleigh quotient matrix $S_0 = X_0^{\mathrm{T}} G X_0$. In our experiments this was done in the following way. Define $p = k+\ell$ and let the $n \times p$ matrix

$$B_0 = \left[\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_p\right] \tag{5.1}$$

be generated as in Section 3, using the starting vector

$$\boldsymbol{b}_0 = \boldsymbol{e} / \|\boldsymbol{e}\|_2 \tag{5.2}$$

where $\boldsymbol{e} = (1,1,\cdots,1)^{\mathrm{T}} \in \mathbb{R}^n$. Then X_0 is obtained by computing an orthonormal basis of Range (B_0).

6. Numerical Experiments

In this section we describe some experiments with the proposed methods. The basic Heart iteration was used with $\ell = k+40$ (Recall that k denotes the number of desired eigenvalues). The compact Heart iteration was tested with two values of ℓ. The first one is $\ell = k+40$, as in the basic iteration. The second value of ℓ is

$$\ell = [k]_{40}^{100} \tag{6.1a}$$

This notation means that ℓ is obtained from k in the following way:

$$\text{If} \quad k \le 40 \quad \text{then} \quad \ell = 40; \tag{6.1b}$$

$$\text{if} \quad 40 \le k \le 100 \quad \text{then} \quad \ell = k; \tag{6.1c}$$

$$\text{if} \quad 100 \le k \quad \text{then} \quad \ell = 100. \tag{6.1d}$$

Note that $k + 40 \ge \lceil k \rceil_{40}^{100}$, hence the second choice increases the number of iterations, but reduces the computational effort per iteration. The first experiments concentrate on the number of iterations (number of restarts) that are needed by each method. For this purpose we have used diagonal test matrices have the form

$$D = \operatorname{diag}\{\lambda_1, \lambda_2, \cdots, \lambda_n\} \in \mathbb{R}^{n \times n} \tag{6.2}$$

where

$$\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_n \ge 0. \tag{6.3}$$

(Since we are interested in iterations there is no loss of generality in experimenting with diagonal matrices, e.g., ([9]: page 367) The diagonal matrices that we have used are displayed in **Table 1**. The eigenvalues of the "Normal distribution" matrix were generated with MATLAB's command "randn(*n*, 1)".

Table 1. Types of test matrices, $n = 200000$.

Matrix type	Matrix eigenvalues $\lambda_j, j = 1, \cdots, n$
Harmonic	$\lambda_j = 1/j$
Harmonic roots	$\lambda_j = (1/j)^{1/2}$
Geometric decay	$\lambda_j = (0.95)^j$
Moderate geometric decay	$\lambda_j = (0.99)^j$
Slow geometric decay	$\lambda_j = (0.999)^j$
Very slow geometric decay	$\lambda_j = (0.9999)^j$
Equispaced	$\lambda_j = (1001 - j)/1000$ for $j = 1, \cdots, 1000$, $\lambda_j = 1/j$ for $j = 1001, \cdots, n$
Densely equispaced	$\lambda_j = (10001 - j)/10000$ for $j = 1, \cdots, 10000$, $\lambda_j = 1/j$ for $j = 10001, \cdots, n$
Normal distribution	Normally distributed eigenvalues

All the experiments were carried out with $\boldsymbol{n = 200{,}000}$, and are aimed at computing the k largest eigenvalues. The iterative process was terminated as soon as it satisfies the stopping condition.

$$\left(\sum_{j=1}^{k} \left| \lambda_j - \lambda_j^{(q)} \right| \right) \Big/ \left(k \left| \lambda_1 \right| \right) \le 10^{-14}, \tag{6.4}$$

where, as before,

$$\lambda_1^{(q)} \geq \cdots \geq \lambda_k^{(q)}$$

denote the computed Ritz values at the qth iteration.

Table 2. Computing k dominant eigenvalues with the basic Heart iteration, $\ell = k + 40$.

Matrix type	Number of iterations					
	$k = 6$	$k = 10$	$k = 20$	$k = 40$	$k = 100$	$k = 200$
Harmonic	0	0	0	0	0	0
Harmonic roots	0	0	0	2	3	4
Geometric	0	0	0	0	0	0
Moderate Geometric	1	1	2	2	0	0
Slow Geometric	7	7	7	9	10	7
Very slow Geometric	38	35	34	26	33	30
Equispaced	6	7	7	9	8	5
Densely Equispaced	38	35	37	29	31	29
Normal Distribution	3	6	9	17	21	32

The figures in Tables 2-4 provide the number of iterations that are needed to satisfy (6.4). Thus, for example, from Table 2 we see that only 5 iterations of basic Heart are needed to compute the largest $k = 200$ eigenvalues of the Equispaced test matrix. A comparison of Table 2 with Table 3 shows that the compact version often requires a smaller number of iterations. One reason for this gain lies in the order of the orthogonalizations. In the basic scheme rank (Y_{q+1}) can be smaller than l, while in the compact scheme rank (Y_{q+1}) is guaranteed to be l. This implies that the compact scheme is able to collect a larger amount of new information.

Table 3. Computing k dominant eigenvalues with the compact Heart iteration, $\ell = k + 40$.

Matrix type	Number of iterations					
	$k = 6$	$k = 10$	$k = 20$	$k = 40$	$k = 100$	$k = 200$
Harmonic	0	0	0	0	0	0
Harmonic roots	0	0	0	1	1	1
Geometric	0	0	0	0	0	0
Moderate Geometric	1	1	1	1	0	0
Slow Geometric	6	7	6	5	4	3
Very slow Geometric	38	36	30	23	16	12
Equispaced	6	7	6	5	4	2
Densely Equispaced	38	36	30	22	16	12
Normal Distribution	2	5	5	6	6	7

Table 4. Computing k dominant eigenvalues with the compact Heart iteration, $\ell = [k]_{40}^{100}$.

Matrix type	Number of iterations					
	$k=6$	$k=10$	$k=20$	$k=40$	$k=100$	$k=200$
Harmonic	0	0	0	1	1	2
Harmonic roots	0	0	1	2	1	3
Geometric	0	0	0	0	0	0
Moderate Geometric	2	2	2	3	1	1
Slow Geometric	9	10	11	15	6	8
Very slow Geometric	47	50	57	75	27	35
Equispaced	8	9	10	15	6	6
Densely Equispaced	47	50	56	76	25	32
Normal Distribution	3	8	8	12	9	18

The second part of our experiments provides timing results that compare the Heart iterations and MATLAB's "eigs" function. For this purpose we have used "PH" matrices that have the following form:

$$G = \left(\prod_{i=1}^{p} H_i\right) D \left(\prod_{i=1}^{p} H_i\right)^{\mathrm{T}} \in \mathbb{R}^{n\times n}, \tag{6.5}$$

where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix and $H_i, i = 1,\cdots, p$, are $n\times n$ sparse random Householder matrices. The matrix D has the form (6.2)-(6.3) with eigenvalues that achieve "slow geometric decay". That is,

$$\lambda_j = (0.999)^{j-1} \text{ for } j = 1,\cdots, n. \tag{6.6}$$

The Householder matrices, $H_i, i = 1,\cdots, p$, have the form

$$H_i = I - 2\boldsymbol{h}_i\boldsymbol{h}_i^{\mathrm{T}} / \boldsymbol{h}_i^{\mathrm{T}}\boldsymbol{h}_i, \tag{6.7}$$

where $\boldsymbol{h}_i \in \mathbb{R}^n$ is a sparse random vector. To generate this vector we have used MATLAB's command h = sprand(n, 1, density) with the density 1000/n. This yields a sparse n-vector that has about 1000 nonzero entries at random locations. Consequently, for small values of p the resulting PH matrix (6.5) is a large sparse symmetric matrix whose nonzero entries lie at random locations. The number of nonzero, ν, increases with p, see **Table 6**. Thus, for example, the $3H$ matrix has 9,071,462 nonzero entries, while the $7H$ matrix has 47,417,512 nonzeroes.

The results in **Table 5** and **Table 6** provide both the computation times (in seconds) and the number of iterations that are needed to compute the k largest eigenvalues. The eigenvalues of a PH matrix are given by (6.6). Thus, as before, the iterative process terminates as soon as (6.4) is satisfied.

Table 5. Timing results (in seconds) and number of iterations for the $3H$ matrix, $n = 200000$, $\nu = 9071462$.

Number of eigenpairs	basic Heart		compact Heart		compact Heart		eigs
	$\ell = k + 40$		$\ell = k + 40$		$\ell = [k]_{40}^{100}$		
k	time	iter.	time	iter.	time	iter.	time
6	24.5	7	16.3	7	18.9	9	17.6
10	28.5	7	18.8	7	22.0	10	18.5
20	34.3	7	23.1	6	27.6	11	22.7
40	78.6	10	37.5	5	47.0	14	35.2
100	151.6	10	83.9	4	83.8	6	80.1
200	271.5	8	215.1	4	162.9	8	187.9
300	336.7	6	316.4	3	290.8	11	289.6
400	388.3	5	461.4	3	421.9	13	407.3
500	479.8	5	672.7	3	475.2	12	651.4
600	722.5	5	657.8	2	617.4	13	734.3

Table 6. Timing results (in seconds) and number of iterations for computing $k = 50$ dominant eigenpairs of PH matrices.

	Matrix	basic Heart		compact Heart		compact Heart		eigs
p	Number of nonzero	$\ell = k + 40 = 90$		$\ell = k + 40 = 90$		$\ell = [k]_{40}^{100} = 50$		
	ν	time	iter.	time	iter.	time	iter.	time
0	200,000	48.6	10	30.3	5	33.5	12	14.1
1	1,197,002	50.5	10	31.5	5	37.7	13	26.4
2	4,158,110	66.4	11	36.2	5	40.3	12	31.5
3	9,071,462	83.5	10	47.8	5	55.3	13	39.4
4	15,909,332	113.5	10	60.6	5	70.9	13	50.3
5	24,559,160	158.0	10	76.8	5	88.4	13	64.8
6	35,098,556	180.1	9	94.6	5	108.0	13	80.1
7	47,417,512	250.0	10	108.0	5	133.1	13	100.1
8	61,438,450	265.5	9	135.0	5	158.5	13	120.0
9	77,349,872	343.6	10	151.9	5	185.4	13	143.1

Let us turn now to conduct a brief operations count for the compact Heart iteration. The Gram-Schmidt orthogonalizations require about

$$n\left[(k+l)^2 - k^2\right] = n\left(l^2 + 2kl\right) = nl(l+2k)$$

multiplications per iteration, while the matrix-vector products require $(l+1)\nu$

multiplications per iteration. The size of the Rayleigh quotient matrix is $k+l$, and the spectral decomposition of this matrix requires a moderate multiple of $(k+l)^3$ multiplications. Thus, when $k+l$ is negligible with respect to n, the spectral decomposition of the Rayleigh-quotient matrix requires considerably less efforts than the orthogonalizations. In our experiments the spectral decomposition was carried out with MATLAB's "eig" function. In this case, for $k=100$ and $l=k+40=140$, the spectral decomposition of the related 240 × 240 matrix required, on average, about 0.01 seconds. Similarly, for $k=500$ and $l=k+40=540$, the spectral decomposition of the related 1040 × 1040 matrix required, on average, 0.1 seconds. That is, **the time spent on the spectral decomposition is negligible with respect to the overall computation time.** Recall that the Lanczos process provides a tridiagonal $(k+l)\times(k+l)$ Rayleigh quotient matrix whose spectral decomposition is faster than that of a full matrix of the same size. Yet the last observation suggests that this is not a real gain. The Ritz vectors matrix $V_{q+1}=X_qU_q$ is obtained by one matrix-matrix product. Thus, although this product achieves $n(k+l)k$ multiplications, it needs considerably less time than the time required for orthogonalizations. These considerations show that most of the computation time is spent during the expansion process, on orthogonalizations and matrix-vector products.

The experiments in Table 5 and Table 6 illustrate how these tasks affect the computation times. Table 5 concentrates on the $3H$ matrix and runs the algorithms for increasing values of k. Thus, for example, we see that for $k=200$ "eigs" required 187.9 seconds, while compact Heart with $l=[200]_{40}^{100}=100$ terminated after 8 iterations and 162.9 seconds. In $3H$ matrices the number of nonzero is moderate; hence for large k most of the computation time is spent on orthogonalizations. Table 6 concentrates on $k=50$ and runs the algorithm with increasing numbers of nonzero. Since k is fixed, the time spent on orthogonalizations is fixed, and the increase in time is mainly due to the cost of matrix-vector products.

The timing results demonstrate the ability of the compact Heart algorithm to compete with MATLAB's "eigs" program. We see that compact Heart is not much slower than "eigs", and in some cases, it is faster. When judging the timing results it is important to note that compact Heart was programmed (in MATLAB) exactly as described in Section 4. Yet, as in other Krylov subspace methods, the basic version can be improved in several ways. Such modifications may include, for example, more effective orthogonalization schemes, locking, and improved rules for the choice of l. Hence from this point of view, the new method is quite promising.

7. Concluding Remarks

Perhaps the main feature that characterizes the new iteration is its simplicity. The discarding of Lanczos' tridiagonalization and implicit restarts results in a simple iteration that retains a fast rate of convergence. As we have seen, in many

cases it requires a remarkably small number of iterations.

The compact Heart iteration is an elegant version of the basic Heart iteration. It uses an effective orthogonalization scheme that avoids the direct computation of the Rayleigh quotient matrix. The experiments that we have done are quite encouraging.

The Heart iteration is a useful tool for calculating low-rank approximations of large matrices. The cross-product approach that was proposed in [8] uses the basic Heart iteration and can be improved by applying the new compact version.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Baglama, J., Calvetti, D. and Reichel, L. (2003) IRBL: An Implicitly Restarted Block Lanczos Method for Large-Scale Hermitian Eigen Problems. *SIAM Journal on Scientific Computing*, **24**, 1650-1677. https://doi.org/10.1137/S1064827501397949

[2] Bai, A., Demmel, J., Dongarra, J., Ruhe, A. and van der Vorst, H. (1999) Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. Society for Industrial and Applied Mathematics, Philadelphia. https://doi.org/10.1137/1.9780898719581

[3] Calvetti, D., Reichel, L. and Sorenson, D.C. (1994) An Implicitly Restarted Lanczos Method for Large Symmetric Eigenvalue Problems. *Electronic Transactions on Numerical Analysis*, **2**, 1-21.

[4] Dax, A. (2017) The Numerical Rank of Krylov Matrices. *Linear Algebra and Its Applications*, **528**, 185-205. https://doi.org/10.1016/j.laa.2016.07.022

[5] Dax, A. (2017) A New Type of Restarted Krylov Methods. *Advances in Linear Algebra & Matrix Theory*, **7**, 18-28. https://doi.org/10.4236/alamt.2017.71003

[6] Dax, A. (2019) A Restarted Krylov Method with Inexact Inversions. *Numerical Linear Algebra with Applications*, **26**, Article No. e2213. https://doi.org/10.1002/nla.2213

[7] Dax, A. (2019) Computing the Smallest Singular Triplets of a Large Matrix. *Results in Applied Mathematic*, **3**, Article ID: 100006. https://doi.org/10.1016/j.rinam.2019.100006

[8] Dax, A. (2019) A Cross-Product Approach for Low-Rank Approximations of Large Matrices. *Journal of Computational and Applied Mathematics*, **369**, Article ID: 112576. https://doi.org/10.1016/j.cam.2019.112576

[9] Demmel, J.W. (1997) Applied Numerical Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia. https://doi.org/10.1137/1.9781611971446

[10] Golub, G.H. and Van Loan, C.F. (2013) Matrix Computations. 4th Edition, Johns Hopkins University Press, Baltimore.

[11] Horn, R.A. and Johnson, C.R. (1985) Matrix Analysis. Cambridge University Press, Cambridge. https://doi.org/10.1017/CBO9780511810817

[12] Larsen, R.M. (2001) Combining Implicit Restarts and Partial Reorthogonalization in Lanczos Bidiagonalization. Technical Report, Stanford University.

[13] Li, R., Xi, Y., Vecharynski, E., Yang, C. and Saad, Y. (2016) A Thick-Restart Lanczos

Algorithm with Polynomial Filtering for Hermitian Eigenvalue Problems. *SIAM Journal on Scientific Computing*, **38**, A2512-A2534. https://doi.org/10.1137/15M1054493

[14] Morgan, R.B. (1996) On Restarting the Arnoldi Method for Large Non-Symmetric Eigenvalues Problems. *Mathematics of Computation*, **65**, 1213-1230. https://doi.org/10.1090/S0025-5718-96-00745-4

[15] Parlett, B.N. (1980) The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs.

[16] Saad, Y. (2011) Numerical Methods for Large Eigenvalue Problems. Revised Edition, Society for Industrial and Applied Mathematics, Philadelphia. https://doi.org/10.1137/1.9781611970739

[17] Sorensen, D.C. (1992) Implicit Application of Polynomial Filters in a *k*-Step Arnoldi Method. *SIAM Journal on Matrix Analysis and Applications*, **13**, 357-385. https://doi.org/10.1137/0613025

[18] Stewart, G.W. (2001) Matrix Algorithms, Vol. II: Eigensystems. SIAM, Philadelphia. https://doi.org/10.1137/1.9780898718058

[19] Trefethen, L.N. and Bau III, D. (1997) Numerical Linear Algebra. Society for Industrial and Applied Mathematics, Philadelphia.

[20] Watkins, D.S. (2007) The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods. Society for Industrial and Applied Mathematics, Philadelphia. https://doi.org/10.1137/1.9780898717808

[21] Wu, K. and Simon, H. (2000) Thick-Restarted Lanczos Method for Large Symmetric Eigenvalue Problems. *SIAM Journal on Matrix Analysis and Applications*, **22**, 602-616. https://doi.org/10.1137/S0895479898334605

[22] Yamazaki, I., Bai, Z., Simon, H., Wang, L. and Wu, K. (2010) Adaptive Projection Subspace Dimension for the Thick-Restart Lanczos Method. *ACM Transactions on Mathematical Software*, **47**, Article No. 27. https://doi.org/10.1145/1824801.1824805

[23] Zhang, F. (1999) Matrix Theory: Basic Results and Techniques. Springer-Verlag, New York.