FUNCTIONS AND EIGENVECTORS OF PARTIALLY KNOWN MATRICES WITH APPLICATIONS TO NETWORK ANALYSIS

MOHAMMED AL MUGAHWI*, OMAR DE LA CRUZ CABRERA†, SILVIA NOSCHESE‡, AND LOTHAR REICHEL§

Abstract. Matrix functions play an important role in applied mathematics. In network analysis, in particular, the exponential of the adjacency matrix associated with a network provides valuable information about connectivity, as well as the relative importance or centrality of nodes. Another popular approach to rank the nodes of a network is to compute the left Perron vector of the adjacency matrix for the network. The present article addresses the problem of evaluating matrix functions, as well as computing an approximation to the left Perron vector, when only some of its columns and/or some of its rows are known. Applications to network analysis are considered, when only some sampled columns and/or rows of the adjacency matrix that defines the network are available. A sampling scheme that takes the connectivity of the network into account is described. Computed examples illustrate the performance of the methods discussed.

Key words. Matrix function, Arnoldi process, low-rank approximation, cross approximation, column subset selection, centrality measure

1. Introduction. Many problems in applied mathematics can be formulated and solved with the aid of matrix functions. This includes the solution of linear discrete ill-posed problems [7], the solution of time-dependent partial differential equations [12], and the determination of the most important node(s) of a network that is represented by a graph and its adjacency matrix [13, 15]. Usually, all entries of the adjacency matrix are assumed to be known; here we are concerned with the situation when only some columns, and/or rows, of the matrix are available. This situation arises, for instance, when one samples columns, and possibly rows, of a large matrix. We will consider applications in network analysis, where column and/or row sampling arises naturally in the process of collecting network data by accessing one node at a time and finding all the other nodes it is connected to. This is particularly important when it is too expensive or impractical to collect a full census of all the connections.

A network is represented by a graph \( G = \{V, E\} \), which consists of a set \( V = \{v_j\}_{j=1}^n \) of vertices or nodes, and a set \( E = \{e_k\}_{k=1}^m \) of edges, the latter being the links between the vertices. Edges may be directed, in which case they emerge from a node and end at a node, or undirected. Undirected edges are “two-way streets” between nodes. For notational convenience and ease of discussion, we consider simple (directed or undirected) unweighted graphs \( G \) without self-loops. Then the adjacency matrix \( A = [a_{ij}]_{i,j=1}^n \in \mathbb{R}^{n \times n} \) associated with the graph \( G \) has the entry \( a_{ij} = 1 \) if there is a directed edge emerging from vertex \( v_i \) and ending at vertex \( v_j \); if there is an undirected edge between the vertices \( v_i \) and \( v_j \), then \( a_{ij} = a_{ji} = 1 \). Other matrix entries vanish. In particular, the diagonal entries of \( A \) vanish. Typically, \( 1 \leq m \ll n^2 \), which makes the matrix \( A \) sparse. A graph is said to be undirected if all its edges are undirected, otherwise the graph is directed. The adjacency matrix for an undirected...
A graph is symmetric; for a directed graph it is nonsymmetric. Examples of networks include:

- Flight networks, with airports represented by vertices and flights by directed edges.
- Social networking services, such as Facebook and Twitter, with members or accounts represented by vertices and interactions between any two accounts by edges.

Numerous applications of networks are described in [9, 14, 27].

We are concerned with the situation when only some of the nodes and edges of a graph are known. Each node and its connections to other nodes determine one row or column of the matrix $A$. Specifically, all edges that point to node $v_i$ determine column $i$ of $A$, and all edges that emerge from this node define the $i$th row of $A$. We are interested in studying properties of networks associated with partially known adjacency matrices.

An important task in network analysis is to determine which vertices of an associated graph are the most important ones by measuring how well-connected they are to other vertices of the graph. This kind of importance measure, which often is referred to as a centrality measure, ignores intrinsic properties of the vertices but provides information about their importance within the graph just by using connectivity information.

A simple approach to measure the centrality of a vertex $v_j$ in a directed graph is to count the number of edges that point to it. This number is known as the indegree of $v_j$. Similarly, the outdegree of $v_j$ is the number of edges that emerge from this vertex. For undirected graphs, the degree of a vertex is the number of edges that “touch” it. However, this approach to measure the centrality of a vertex often is unsatisfactory, because it ignores the importance of the vertices that $v_j$ is connected to. This shortcoming has prompted the introduction of several centrality measures that are based on the evaluation of matrix functions at the adjacency matrix $A$ of $G$; see, e.g., [13] for a nice introduction.

To discuss measures determined by matrix functions, we need the notion of a walk in a graph. A walk of length $k$ is a sequence of $k + 1$ vertices $v_{i_1}, v_{i_2}, \ldots, v_{i_{k+1}}$ and a sequence of $k$ edges $e_{j_1}, e_{j_2}, \ldots, e_{j_k}$, such that $e_{j_\ell}$ points from $v_{i_\ell}$ to $v_{i_{\ell+1}}$ for $\ell = 1, 2, \ldots, k$. The vertices and edges of a walk do not have to be distinct. It is a well known fact that $[A^k]_{ij}$, i.e., the $(ij)^{th}$ entry of $A^k$, yields the number of walks of length $k$ starting at node $v_i$ and ending at node $v_j$. Thus, a matrix function evaluated at the adjacency matrix $A$, defined by a power series $\sum_{k=0}^{\infty} \alpha_k A^k$, can be interpreted as containing weighted sums of walk counts, with weights depending on the length of the walk. Unless $A$ is nilpotent (i.e., the graph is directed and contains no cycles), convergence requires that the coefficients $\alpha_k$ converge to zero; this corresponds well with the intuitively natural requirement that long walks be given less weight than short walks (which is the case in (1.1) and (1.2) below).

Commonly used matrix functions for measuring the centrality of the vertices of a graph are the exponential function $\exp(\gamma_e A)$ and the resolvent $(I - \gamma_r A)^{-1}$, where $\gamma_e$ and $\gamma_r$ are positive user-chosen scaling parameters; see, e.g., [13]. These functions can be defined by their power series expansions

\[
\exp(\gamma_e A) = I + \gamma_e A + \frac{1}{2!}(\gamma_e A)^2 + \frac{1}{3!}(\gamma_e A)^3 + \ldots, \quad (1.1)
\]

\[
(I - \gamma_r A)^{-1} = I + \gamma_r A + (\gamma_r A)^2 + (\gamma_r A)^3 + \ldots. \quad (1.2)
\]

For the resolvent, the parameter $\gamma_r$ has to be chosen small enough so that the power
series converges, which is the case when \( \gamma_r \) is strictly smaller than \( 1/\rho(A) \), where \( \rho(A) \) denotes the spectral radius of \( A \).

Matrix functions \( f(A) \), such as (1.1) and (1.2), define several commonly used centrality measures:

- \( [f(A)]_{ii}, i = 1, 2, \ldots, n \). The largest entries correspond to the most important nodes, i.e., if \( [f(A)]_{ii} \geq [f(A)]_{jj} \) for all \( j \neq i \), then node \( v_i \) is considered the most important node of the network.
- \( [f(A)\mathbf{1}]_{i}, i = 1, 2, \ldots, n, \mathbf{1} = [1, 1, \ldots, 1]^T \). The largest vector entries correspond to the most important nodes, i.e., if \( [f(A)\mathbf{1}]_{i} > [f(A)\mathbf{1}]_{j} \) for all \( j \neq i \), then node \( v_i \) is considered the most important node of the network.

It may be beneficial to complement the centrality measures above by the measures \( [f(A^T)]_{ii} \) and \( [f(A^T)\mathbf{1}]_{i} \), when the graph \( G \) that defines \( A \) is directed. Here and below the superscript \( T \) denotes transposition; see, e.g., [2, 11, 13, 14] for discussions on centrality measures defined by functions of the adjacency matrix.

We are interested in computing useful approximations of the largest diagonal entries of \( f(A) \), or the largest entry of \( f(A)\mathbf{1} \) or \( f(A^T)\mathbf{1} \), when only \( 1 \leq k \ll n \) of the columns and/or rows of \( A \) are known. The need to compute such approximations arises when the entire graph \( G \) is not completely known, but only a small subset of the columns or rows of the adjacency matrix \( A \) of \( G \) are available. This happens, e.g., when not all nodes and edges of a graph are known, a situation that is common for large, complex, real-life networks. The situation we will consider is when the columns and rows of the adjacency matrix are not explicitly known, but can be sampled. It is then of considerable interest to investigate how the sampling should be carried out, as simple random sampling of columns and possibly rows of a large adjacency matrix does not give the best results. We will describe a sampling method in Section 2. A further reason for our interest in computing approximations of functions of a large matrix \( A \), that only use a few of the columns and/or rows of the matrix, is that the evaluation of these approximations typically is much cheaper than the evaluation of functions of \( A \).

Another approach to measure centrality is to compute a left eigenvector associated with the eigenvalue of largest magnitude of \( A \). Adding a small nonnegative perturbation to \( A \) secures that the eigenvector, suitably scaled, only has positive entries. This vector is commonly referred to as the left Perron vector of the (perturbed) matrix \( A \). If the \( j^{th} \) entry of this vector is the largest, then \( v_j \) is the most important vertex of the graph. Thus, the centrality of a node is given by the relative size of its associated entry of the Perron vector of a (possibly perturbed) adjacency matrix. This approach to determine centrality is known as the PageRank method; see, e.g., [3, 6, 14, 23, 27] for discussions of this method. We will discuss the application of this method to partially known adjacency matrices.

This paper is organized as follows. Section 2 discusses our sampling method for determining (partial) knowledge of the graph and its associated adjacency matrix. The evaluation of matrix functions of adjacency matrices that are only partially known is considered in Section 3, and Section 4 describes how an approximation of functions of \( A \) can be computed quite inexpensively by using low-rank approximations determined by sampling. A few computed examples are presented in Section 5, and concluding remarks can be found in Section 6.

2. Sampling adjacency matrices. Let \( \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k \geq 0 \) be the \( k \) largest singular values of a large matrix \( A \in \mathbb{R}^{n \times n} \), with \( 1 \leq k \ll n \), and let \( \mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_k \) and \( \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k \) be associated left and right singular vectors. Then the truncated
singular value decomposition (TSVD)

\[ A_k = \sum_{j=1}^{k} \sigma_j u_j v_j^T, \]  

(2.1)

c furnishes a best approximation of \( A \) of rank at most \( k \) with respect to the spectral and Frobenius matrix norms; see, e.g., [34]. However, the computation of the approximation (2.1) may be expensive when \( n \) is large and \( k \) is of moderate size. This limits the applicability of the TSVD-approximant (2.1). Moreover, the evaluation of this approximant requires that all entries of \( A \) be explicitly known. As described above, we are concerned with the situation when \( A \) is an adjacency matrix for a simple (directed or undirected) unweighted graph without self-loops and that, while the whole matrix is not known, we can sample a (relatively small) number of rows and columns. Then, approximations different from (2.1) have to be used. This section discusses methods to sample columns and/or rows of \( A \). The low-rank approximations of \( A \) determined in this manner are used in Sections 3 and 4 to compute approximations of node centralities.

In the first step, a random non-vanishing column of \( A \) is chosen; let its index be \( j_1 \), and denote the chosen column by \( c_1 \). If the columns \( c_1, \ldots, c_k \) have been chosen, corresponding to indices \( j_1, \ldots, j_k \), at the next step we pick an index \( j_{k+1} \) according to a probability distribution on \( \{1, \ldots, n\} \) proportional to \( c_1 + \cdots + c_k \). Thus, at the \((k+1)\)st step, the probability of choosing column \( i \) as the next sampled column is proportional to the number of edges in the network from node \( v_i \) to nodes \( v_{j_1}, \ldots, v_{j_k} \). At each step, if a column has already been picked, or the new column consists entirely of zeros, it is discarded and the procedure is repeated until a new, nonzero column \( c_{k+1} \) is obtained.

We remark that this scheme for selecting columns can just as easily be used in the case when the edges have positive weights (that is, the nonzero entries of \( A \) may be positive numbers other than 1). Also, if a row-sampling scheme is needed, rows of the adjacency matrix \( A \) can be selected similarly by applying the above scheme to the columns of the matrix \( A^T \); in this case we denote by \( I \) the set of row indices. The matrix \( A_{(I,\cdot)} \in \mathbb{R}^{k \times n} \) contains the selected rows of \( A \). By alternating column and row sampling, sets of columns and rows can be determined simultaneously.

The adaptive cross approximation method (ACA) applied to a matrix \( A \) also samples rows and columns to obtain an approximation of the whole matrix. In ACA, one uses the fact that the rows and columns of \( A_{(I,\cdot)} \) and \( A_{(\cdot,J)} \) have common entries. These entries form the matrix \( A_{(I,J)} \in \mathbb{R}^{k \times k} \). Assuming that the latter matrix is nonsingular, the cross approximation of \( A \) is given by

\[ M_k = A_{(\cdot,J)} A_{(I,J)}^{-1} A_{(I,\cdot)}. \]  

(2.2)

Assume for the moment that the matrix (2.1) satisfies \( \|A - A_k\|_2 = \varepsilon \), where \( \| \cdot \|_2 \) denotes the spectral norm. Goreinov et al. [18] show that there is a matrix \( M_k^* \) of rank \( k \), determined by cross approximation of \( A \), such that

\[ \|A - M_k^*\|_2 = \mathcal{O}(\varepsilon \sqrt{kn}). \]  

(2.3)

However, the selection of columns and rows of \( A \) so that (2.3) holds is computationally difficult. In their analysis, Goreinov et al. [19], select sets \( I \) and \( J \) that give the submatrix \( A_{(I,J)} \) maximal “volume” (modulus of the determinant). It is difficult to
compute these index sets in a fast manner. Therefore, other methods to select the sets
I and J have been proposed; see, e.g., [16, 25]. They are related to incomplete Gaus-
sian elimination with complete pivoting. These methods work well when the matrix
A is not very sparse. The adjacency matrices of concern in the present paper typically
are quite sparse, and we found the sampling methods described in [16, 25] often to
give singular matrices $A_{(I,J)}$. This makes the use of adaptive cross approximation
difficult. We therefore will not use the expression (2.2) in subsequent sections.

3. Functions of low-rank matrix approximations. This section discusses
the approximation of functions $f$ of a large matrix $A \in \mathbb{R}^{n \times n}$ that is only partially
known. Specifically, we assume that only $1 \leq \ell \ll n$ columns of $A$ are available, and
we would like to determine an approximation of $f(A)$. We will tacitly assume that
the function $f$ and matrix $A$ are such that $f(A)$ is well defined; see, e.g., [17, 20] for
several definitions of matrix functions. For the purpose of this paper, the definition
of a matrix function by its power series expansion suffices; cf. (1.1) and (1.2). We
first will assume that the matrix $A$ is nonsymmetric. At the end of this section, we
will address the situation when $A$ is symmetric.

Let $P \in \mathbb{R}^{n \times n}$ be a permutation matrix such that the known columns of the
matrix $AP$ have index 1, 2, ..., $\ell$. Then, using that $P^T = P^{-1}$, we obtain

$$f(A) \approx Pf(A_\ell)P^T, \quad A_\ell = [c_1, \ldots, c_\ell, 0_{n,n-\ell}], \quad P^TAP = [c_1, \ldots, c_n]. \quad (3.1)$$

Hence, it suffices to consider the evaluation of $f$ at an $n \times n$ matrix, whose
$n - \ell$ last columns vanish.

The computations simplify when $f(0) = 0$. We therefore will consider the func-
tions

$$f(A_\ell) = \exp(\gamma e A_\ell) - I \quad \text{and} \quad f(A_\ell) = (I - \gamma r A_\ell)^{-1} - I. \quad (3.2)$$

The subtraction of $I$ in the above expressions generally is of no significance for the
analysis of networks, because one typically is interested in the relative sizes of the
diagonal entries of $f(A_\ell)$, or of the vector entries $f(A_\ell)1$ or $f(A_\ell^T)1$.

The power series representations of the functions in (3.2),

$$f(A_\ell) = c_1 A_\ell + c_2 A_\ell^2 + \ldots,$$

show that only the first $\ell$ columns of the matrix $f(A_\ell)$ contain nonvanishing entries.

Let $v_1$ be a random unit vector. Application of $\ell$ steps of the Arnoldi process to
$A_\ell$ with initial vector $v_1$, generically, yields the Arnoldi decomposition

$$A_\ell V_{\ell+1} = V_{\ell+1} H_{\ell+1}, \quad (3.3)$$

where $H_{\ell+1} \in \mathbb{R}^{(\ell+1) \times (\ell+1)}$ is an upper Hessenberg matrix and the matrix $V_{\ell+1} \in
\mathbb{R}^{n \times (\ell+1)}$ has orthonormal columns. The computation of the Arnoldi decomposition
(3.3) requires the evaluation of $\ell$ matrix-vector products with $A_\ell$, which is quite
inexpensive since $A_\ell$ has at most $\ell$ nonvanishing columns. We assume that the de-
composition (3.3) exists. This is the generic situation. Breakdown of the Arnoldi
process, generically, occurs at step $\ell + 1$; see Saad [31, Chapter 6] for a thorough
discussion of the Arnoldi decomposition and its computation.

Introduce the spectral factorization

$$H_{\ell+1} = S_{\ell+1} A_{\ell+1} S_{\ell+1}^{-1}, \quad (3.4)$$
which we tacitly assume to exist. This is the generic situation. Thus, the matrix $\Lambda_{\ell+1}$ is diagonal; its diagonal entries are the eigenvalues of $H_{\ell+1}$. At least one of these eigenvalues vanishes. There is a permutation matrix $\tilde{P}$ such that the last diagonal entry of the diagonal matrix $PA_{\ell+1}P^T$ in the decomposition

$$H_{\ell+1} = S_{\ell+1}\tilde{P}^T(\tilde{P}\Lambda_{\ell+1}\tilde{P}^T)\tilde{P}S_{\ell+1}^{-1}$$

vanishes. We henceforth also will refer to the permuted matrices in this decomposition as $\Lambda_{\ell+1}$, $S_{\ell+1}$, and $S_{\ell+1}^{-1}$. Thus, the last column of the permuted matrix $S_{\ell+1}$ is an eigenvector that is associated with a vanishing eigenvalue. There may be other vanishing diagonal entries, but this will not be exploited. The situation when the factorization (3.4) does not exist can be handled as described by Pozza et al. [29].

We have

$$A_\ell V_{\ell+1}S_{\ell+1} = V_{\ell+1}S_{\ell+1}\Lambda_{\ell+1}.$$  

The columns of $V_{\ell+1}S_{\ell+1}$ are eigenvectors of $A_\ell$. The last column of $V_{\ell+1}S_{\ell+1}$ is an eigenvector that is associated with a vanishing eigenvalue.

Let $w_j = V_{\ell+1}S_{\ell+1}e_j$, $j = 1, 2, \ldots, \ell$, where $e_j$ denotes the $j^{th}$ column of an identity matrix of appropriate order. Then

$$S_n = [w_1, \ldots, w_\ell, e_{\ell+1}, \ldots, e_n] \in \mathbb{R}^{n \times n}$$

is an eigenvector matrix of $A_\ell$, and

$$A_\ell S_n = S_n \begin{bmatrix} \Lambda_\ell & & \\ & \ddots & \\ & & 0 \end{bmatrix},$$

where $\Lambda_\ell$ is the $\ell \times \ell$ leading principal submatrix of $\Lambda_{\ell+1}$. Hence,

$$f(A_\ell) = S_n f \begin{bmatrix} \Lambda_\ell & & \\ & \ddots & \\ & & 0 \end{bmatrix} S_n^{-1}$$

$$= S_n \begin{bmatrix} f(\lambda_1) & & \\ & \ddots & \| \\ & & f(\lambda_{\ell}) \end{bmatrix} S_n^{-1}, \quad (3.5)$$

where we have used the fact that $f(0) = 0$.

To evaluate the expression (3.5), it remains to determine the first $\ell$ rows of $S_n^{-1}$. Define the matrix $W = [w_1, w_2, \ldots, w_\ell] \in \mathbb{R}^{n \times \ell}$ and introduce a reduced QR factorization of $W = QR$, where $Q \in \mathbb{R}^{n \times \ell}$ has orthonormal columns and $R \in \mathbb{R}^{\ell \times \ell}$ is upper triangular; see [34]. The matrix $R^{-1}Q^T$ represents the first $\ell$ rows of $S_n^{-1}$. Then we can evaluate

$$f(A_\ell) = QRf(\Lambda_\ell)R^{-1}Q^T. \quad (3.6)$$
Our approximation of \( f(A) \) is given by \( Pf(A)fPT \). For a large matrix \( A \), the computationally most expensive part of evaluating this approximation, when the matrix \( A_{\ell} \) is available, is the computation of the Arnoldi decomposition (3.3), which requires \( O(m^2) \) arithmetic floating point operations.

We remark that for functions such that \( f(A) = (f(A^T))^T \), which includes the functions (1.1) and (1.2), we may instead sample rows of \( A \), which are columns of \( A^T \), to determine an approximation of \( f(A) \) using the same approach as described above.

We turn to the situation when the matrix \( A \in \mathbb{R}^{n \times n} \) is symmetric, and assume that \( 1 \leq \ell \ll n \) of its columns are known. Let the permutation matrix \( P \) be the same as above. Then the first \( \ell \) rows and columns of the symmetric matrix \( A_{\ell} = P^T A P \) are available. Letting \( v_1 \) be a random unit vector and applying \( \ell \) steps of the symmetric Lanczos process to \( A_{\ell} \) with initial vector \( v_1 \) gives, generically, the Lanczos decomposition

\[
A_{\ell}v_{\ell+1} = V_{\ell+1}T_{\ell+1},
\]

where \( T_{\ell+1} \in \mathbb{R}^{(\ell+1) \times (\ell+1)} \) is a symmetric tridiagonal matrix and \( V_{\ell+1} \in \mathbb{R}^{n \times (\ell+1)} \) has orthonormal columns. The computation of the decomposition (3.7) requires the evaluation of \( \ell \) matrix-vector products with \( A_{\ell} \). We assume \( \ell \) is small enough so that the decomposition (3.7) exists. This is the generic situation. Breakdown of the symmetric Lanczos process, generically, occurs at step \( \ell + 1 \). We now can derive a representation of \( f(A_{\ell}) \) of the form (3.5), making use of the spectral factorization of \( T_{\ell+1} \).

4. The computation of an approximate left Perron vector. Let \( A \in \mathbb{R}^{n \times n} \) be an adjacency matrix of a graph, and modify its entries slightly so that they all are positive; see below. Then the modified matrix has a unique left eigenvector \( y = [y_1, y_2, \ldots, y_n]^T \in \mathbb{R}^n \) of unit length with all entries positive. This vector is referred to as the left Perron vector. The importance of node \( v_i \) is proportional to \( y_i \); see [3, 13, 23, 27] for detailed discussions of this centrality measure, which commonly is referred to as eigenvector centrality. When the matrix \( A \) is nonsymmetric, the left Perron vector measures the centrality of the nodes as receivers. It is an extension of the in-degree centrality. The entries of the right Perron vector yield the centrality of the nodes as transmitters. This is an extension of the out-degree centrality and sometimes also is of interest. The present paper focuses on the left Perron vector.

Assume for the moment that the (unmodified) adjacency matrix \( A \) is nonsymmetric. We would like to determine an approximation of the left Perron vector by using a submatrix determined by sampling columns and rows as described in Section 2. Let the set \( J \) contain the \( \ell \) indices of the sampled columns of \( A \). Thus, the matrix \( A_{(i,j)} \in \mathbb{R}^{n \times n} \) contains the sampled columns. Similarly, applying the same column sampling method to \( A^T \) gives the index set \( I \) of \( \ell \) indices, and the matrix \( A_{(i,j)} \in \mathbb{R}^{n \times n} \) containing the sampled rows. We will compute an approximation of the left Perron vector of \( A \) by applying the power method to the matrix \( M_{\ell} = A_{(i,j)}A_{(i,j)} \) (without explicitly forming \( M_{\ell} \)). Possible nonuniciy of the Perron vector can be remedied by adding the matrix \( E \in \mathbb{R}^{n \times n} \) to \( M_{\ell} \), where all entries of \( E \) are equal to a small parameter \( \varepsilon > 0 \). The computations with the power method are carried out without explicitly storing the matrix \( E \) and forming \( M_{\ell} + E \). The iterations with the power method applied to \( M_{\ell} + E \) are much cheaper than the iterations with the power method applied to \( A \), when \( \ell \ll n \). Moreover, our method does not require the whole matrix \( A \) to be explicitly known. In the computed examples reported in Section 5 we achieved fairly accurate rankings of the most important nodes without using the
matrix $E$ defined above. Moreover, we found that only a few rows and columns of $A$ were needed to quite accurately determine the most important nodes in several “real” examples.

When the adjacency matrix $A$ is symmetric, we propose to compute the Perron vector of the matrix $M_\ell = A_{(,j)}A_{(j,)}$, which can be constructed by sampling the columns of $A$ only to construct $A_{(,j)}$, since $A_{(j,)} = A^T_{(,j)}$. Notice that for symmetric matrices the right and left Perron vectors are the same.

5. Computed examples. This section illustrates the performance of the methods discussed when applied to the ranking of nodes in several “real” large networks. All computations were carried out in MATLAB with about 15 significant decimal digits on a Microsoft Windows 10 computer with CPU Intel(R) Core(TM) i7-8550U @ 1.80GHz, 1992 Mhz, 4 Cores, 8 Logical Processors and 16GB of RAM.

| Rank | $A_{500}$ | $A_{1000}$ | $A_{1500}$ | $A_{2000}$ | $A_{2500}$ | $A_{3000}$ |
|------|-----------|-----------|-----------|-----------|-----------|-----------|
| 1    | 35        | 35        | 35        | 35        | 35        | 35        |
| 2    | 28        | 646       | 2         | 28        | 32        | 32        |
| 3    | 32        | 564       | 646       | 564       | 564       | 28        |
| 4    | 402       | 28        | 547       | 646       | 2         | 402       |
| 5    | 564       | 428       | 419       | 547       | 32        | 28        |
| 6    | 2         | 427       | 334       | 419       | 31        | 2         |
| 7    | 31        | 547       | 427       | 31        | 402       | 564       |
| 8    | 419       | 450       | 41       | 45        | 547       | 419       |
| 9    | 646       | 20        | 402       | 448       | 547       | 75        |
| 10   | 547       | 170       | 664       | 427       | 419       | 646       |
| 11   | 75        | 23        | 552       | 170       | 646       | 419       |
| 12   | 45        | 334       | 31        | 444       | 75        | 82        |
| 13   | 738       | 402       | 13        | 102       | 664       | 45        |
| 14   | 664       | 75        | 767       | 551       | 444       | 551       |
| 15   | 41        | 69        | 12        | 690       | 102       | 102       |
| 16   | 551       | 142       | 120       | 44        | 49        | 444       |
| 17   | 444       | 726       | 415       | 67        | 13        | 49        |
| 18   | 82        | 13        | 450       | 635       | 128       | 170       |
| 19   | 20        | 376       | 32        | 736       | 67        | 664       |
| 20   | 13        | 80        | 384       | 62        | 551       | 450       |

Fig. 5.1. soc-Epinions1: The top twenty ranked nodes using the diagonal of $f(A)$ (2nd column), and rankings determined by the diagonals of $f(A_\ell)$ for $\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}$. The columns of $A$ are sampled as described in Section 2.

5.1. soc-Epinions1. The network of this example is a “web of trust” among members of the website Epinions.com. This network describes who-trust-whom. Each user may decide to trust the reviews of other users or not. The users are represented by nodes. An edge from node $v_i$ to node $v_j$ indicates that user $i$ trusts user $j$. The network is directed with 75,888 members (nodes) and 508,837 trust connections (edges) [30, 32]. We will illustrate that one can determine a fairly accurate ranking of the nodes by only using a relatively small number of columns of the nonsymmetric
adjacency matrix $A \in \mathbb{R}^{n \times n}$ with $n = 75888$. The node centrality is determined by evaluating approximations of the diagonal entries of the matrix function $f(A) = \exp(A) - I$.

We sample $\ell \ll n$ columns of the adjacency matrix $A$ using the method described in Section 2. The first column, $c_1$, is a randomly chosen nonvanishing column of $A$; the remaining columns are chosen as described in Section 2. Once the $\ell$ columns of $A$ have been chosen, we evaluate an approximation of $f(A)$ as described in Section 3. The rankings obtained are displayed in Figure 5.1; see below for a detailed description of this figure. When instead all columns of $A$ are chosen randomly, then we obtain the rankings shown in Figure 5.2.

The exact ranking of the nodes of the network is difficult to determine due to the large size of the adjacency matrix. It is problematic to evaluate $f(A)$ both because of the large amount of computational arithmetic required, and because of the large storage demand. While the matrix $A$ is sparse, and therefore can be stored efficiently using a sparse storage format, the matrix $f(A)$ is dense. In fact, the MATLAB function `expm` cannot be applied to evaluate $\exp(A)$ on the computer used for the numerical experiments. Instead, we apply the Arnoldi process to approximate $f(A)$. Specifically, $k$ steps of the Arnoldi process applied to $A$ with a random unit initial vector generically gives the Arnoldi decomposition

$$AV_k = V_k H_k + g_k e_k^T,$$

(5.1)
where the matrix $V_k \in \mathbb{R}^{n \times k}$ has orthonormal columns, $H_k \in \mathbb{R}^{k \times k}$ is an upper Hessenberg matrix, and the vector $g_k \in \mathbb{R}^n$ is orthogonal to the columns of $V_k$. We then approximate $f(A)$ by $V_k f(H_k) V_k^T$; see, e.g., [11,12] for discussions on the approximation of matrix function using the Arnoldi process. These computations were carried out for $k = 4000$, $k = 6000$, $k = 8000$, and $k = 9000$, and rankings $\text{diag}(V_k f(H_k) V_k^T)$ for these $k$-values were determined. We found the rankings to converge as $k$ increases. The ranking obtained for $k = 9000$ therefore is considered the “exact” ranking. It is shown in the second column of Figure 5.1. Subsequent columns of this figure display rankings determined by the diagonal entries of $f(A_\ell)$ for $\ell = 500$, 1000, 1500, 2000, 2500, and 3000, when the columns of $A$ are sampled by the method of Section 2. Each column shows the top 20 ranked nodes. To make it easier for the reader to see the ranking, we use 4 colors, and 5 levels for each color. As we pick 500 columns of $A$, 9 of the top 20 ranked nodes are identified, but only the most important node (35) has the correct ranking. When $\ell = 1000$, the computed ranking improves somewhat. We are able to identify 11 out of top 20 nodes. As we sample more columns of $A$, we obtain improved rankings. For $\ell = 3000$, we are able to identify 17 of the 20 most important nodes, and the rankings get closer to the exact ranking. The figure illustrates that useful information about node centrality can be determined by sampling many fewer than $n$ columns of $A$.

Figure 5.2 differs from Figure 5.1 in that the columns of the matrix $A$ are randomly sampled. Comparing these figures shows the sampling method of Section 2 to yield rankings that are closer to the “exact ranking” of the second column for the same number of sampled columns.

5.2. ca-CondMat. This example illustrates the application of the technique of Section 3 to symmetric partially known matrices. We consider a collaboration network from e-print arXiv. The 23,133 nodes of the associated graph represent authors. If author $i$ co-authored a paper with author $j$, then the graph has an undirected edge connecting the nodes $v_i$ and $v_j$. The adjacency matrix $A$ is symmetric with 186,936 non-zero entries [24,32]. Of the entries, 58 are on the diagonal. Since we are interested in graphs without self-loops, we set the latter entries to zero. We use the node centrality measure furnished by the diagonal of $f(A) = \exp(A) - I$.

Figure 5.3 shows results when using the sampling method described in Section 2 to choose $\ell$ rows of $A$. The figure compares the ranking of the nodes using the diagonal of the matrix $f(A)$ (which is the exact ranking) with the rankings determined by the diagonal entries of $f(A_\ell)$ for $\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}$. The figure shows the top 20 ranked nodes determined by each matrix. For $\ell = 500$, a couple of the 20 most important can be identified among the first 20 nodes, but their rankings are incorrect. The most important node (5013) is in the 13th position, and the second most important node (21052) is in the 3rd position. Increasing $\ell$ to 1000 yields more accurate rankings. The most important nodes, i.e., (5013), (21052), and (18746), are ranked correctly. Increasing $\ell$ further yields rankings that are closer to the “exact” ranking of the second column. For instance, $\ell = 2000$ identifies 19 of the 20 most important nodes, and 8 of them have the correct rank. The figure suggests that we may gain valuable insight into the ranking of the nodes by using fairly few columns (and rows) of the adjacency matrix, only.

Figure 5.4 differs from Figure 5.3 in that the columns of the matrix $A$ are randomly sampled. Comparing these figures shows that the sampling method of Section 2 gives rankings that are closer to the “exact ranking” of the second column for the same
| Rank | \(A\)   | \(A_i\) |
|------|---------|---------|
| 1    | 5013    | 18746   |
| 2    | 21052   | 11302   |
| 3    | 18746   | 21052   |
| 4    | 11302   | 9872    |
| 5    | 9872    | 5013    |
| 6    | 13768   | 4081    |
| 7    | 5500    | 17245   |
| 8    | 20667   | 5500    |
| 9    | 17245   | 6707    |
| 10   | 4081    | 5617    |
| 11   | 5617    | 7050    |
| 12   | 6707    | 20128   |
| 13   | 9956    | 5013    |
| 14   | 18866   | 7597    |
| 15   | 16897   | 12149   |
| 16   | 7050    | 7597    |
| 17   | 7184    | 11551   |
| 18   | 18743   | 19083   |
| 19   | 11914   | 12426   |
| 20   | 20128   | 13455   |

Fig. 5.3. ca-CondMat: The top twenty nodes determined by the diagonals of \(f(A_\ell)\) for \(\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}\). The columns of \(A\) are sampled as described in Section 2.

5.3. Enron. This example illustrates the application of the method described in Section 4 to a nonsymmetric adjacency matrix. The network in this example is an e-mail exchange network, which represents e-mails (edges) sent between Enron employees (nodes). The associated graph is unweighted and directed with 69,244 nodes and 276,143 edges, including 1,535 self-loops. We removed the self-loops before running the experiment. This network has been studied in [10] and can be found at [33].

We choose \(\ell\) columns of the matrix \(A\) as described in Section 2 and put the indices of these columns in the index set \(J\). Similarly, we select \(\ell\) columns of the matrix \(A^T\). The indices of these rows make up the set \(I\). This determines the matrix \(M_\ell = A(\cdot,J)A(I,\cdot) \in \mathbb{R}^{n \times n}\) of rank at most \(\ell\). We determine an approximation of a left Perron vector of \(A\), by computing a left Perron vector of \(M_\ell\). The size of the entries of the Perron vectors determine the ranking.

The second column of Figure 5.5 shows the “exact ranking” determined by a left Perron vector of \(A\). The remaining columns show the rankings defined by Perron vectors of \(M_\ell\) for \(\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}\) with the sampling of the columns of \(A\) carried out as described in Section 2. The ranking determined by Perron vectors of \(M_\ell\) gets closer to the exact ranking in the second column as \(\ell\) increases. When \(\ell = 500\), we are able to identify 12 out of the 20 most important nodes, but...
not in the correct order. The three most important nodes have the correct ranking for $\ell \geq 1000$. When $\ell \geq 2000$, we almost can identify all the 20 important nodes, because node (60606) is actually ranked 21st.

Figure 5.5 differs from Figure 5.5 in that the columns of the matrix $A$ are randomly sampled. These figures show that sampling method of Section 2 gives rankings that are closer to the “exact ranking” of the second column for the same number of sampled columns.

5.4. Cond-mat-2005. The network in example models the collaboration network of scientists posting preprints on the condensed matter archive at www.arxiv.org. It is discussed in [28] and it can be found at [26]. We use unweighted version of the network. The associated graph is undirected and has 40,421 nodes and 351,382 edges. We use the Perron vector as a centrality measure, and compare the node ranking using the Perron vector of $A$ with ranking determined by the Perron vector for the matrices $M_\ell = A_{(\cdot,J)}A_{(J,\cdot)} \in \mathbb{R}^{n \times n}$ for several $\ell$-values. The matrix $A_{(\cdot,J)}$ is determined as described in Section 2, and $A_{(J,\cdot)}$ is just $A^T_{(\cdot,J)}$.

Figure 5.7 shows the (exact) ranking obtained with the Perron vector for $A$ (2nd column) and the rankings determined by the Perron vector for $M_\ell$, for $\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}$, when the columns of $A$ are sampled as described in Section 2. We compare the top 20 ranked nodes in these rankings. When $\ell = 500$, the two most important nodes are ranked correctly by using the Perron vector for

| Rank | $A_{500}$ | $A_{1000}$ | $A_{1500}$ | $A_{2000}$ | $A_{2500}$ | $A_{3000}$ |
|------|-----------|-----------|-----------|-----------|-----------|-----------|
| 1    | 5013      | 505       | 21833     | 7597      | 7597      | 7597      | 11302     |
| 2    | 21052     | 7997      | 505       | 11551     | 11551     | 11551     | 18746     |
| 3    | 18746     | 18840     | 14528     | 21833     | 21833     | 21833     |           |
| 4    | 11302     | 1991      | 11295     | 505       | 19371     | 19371     | 7597      |
| 5    | 9872      | 11915     | 7597      | 17033     | 13100     | 13100     | 11551     |
| 6    | 13768     | 21052     | 1991      | 14528     | 21833     | 21833     | 11551     |
| 7    | 5500      | 2719      | 18840     | 18840     | 14509     | 18840     | 5617      |
| 8    | 20667     | 5013      | 21052     | 16240     | 16240     | 7184      |           |
| 9    | 17245     | 2556      | 15351     | 2719      | 505       | 505       | 21052     |
| 10   | 4081      | 19117     | 3129      | 2099      | 2099      | 20976     | 11295     |
| 11   | 5617      | 9956      | 3444      | 11153     | 2719      | 14509     | 9667      |
| 12   | 6707      | 8532      | 17365     | 14528     | 9843      | 12053     | 5223      |
| 13   | 9956      | 18210     | 14869     | 19371     | 13303     | 16240     | 12426     |
| 14   | 18866     | 4738      | 19962     | 1991      | 12426     | 12426     | 10117     |
| 15   | 16897     | 9869      | 13696     | 21052     | 18840     | 2099      | 9872      |
| 16   | 7050      | 15351     | 4415      | 13303     | 4845      | 2719      | 13100     |
| 17   | 7184      | 14793     | 14883     | 9843      | 16228     | 9667      | 505       |
| 18   | 18743     | 20217     | 958       | 8215      | 12093     | 12426     | 5013      |
| 19   | 11914     | 21874     | 9226      | 11295     | 3173      | 9843      | 22637     |
| 20   | 20128     | 16304     | 9879      | 14869     | 2375      | 9872      | 5500      |

Fig. 5.4. ca-CondMat: The top twenty nodes determined by the diagonals of $f(A_{\ell})$ for $\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}$. The columns of $A$ are sampled randomly.
The above examples illustrate that valuable information about the ranking of nodes can be gained by sampling columns and rows of the adjacency matrix. The last two examples determine the left Perron vector. The most popular methods for computing this vector for a large adjacency matrix is the power method and enhanced variants of the power method that do not require much computer storage. These methods, of course, also can be applied to determine the left Perron vector of the matrices $M_\ell$. It is outside the scope of the present paper to compare approaches to efficiently compute the left Perron vector. Extrapolation and other techniques for accelerating the power method are described in [3, 4, 5, 8, 21, 22, 35].

6. Conclusion. In this work we have described novel methods for analyzing large networks in situations when not all of the adjacency matrix is available. This was done by evaluating matrix functions or computing approximations of the Perron

|     | A      |       | $M_\ell$ |       |       |       |
|-----|--------|-------|---------|-------|-------|-------|
| Rank|        | 500   | 1000    | 1500  | 2000  | 2500  |
| 1   | 30280  | 56169 | 30280   | 30280 | 30280 | 30280 |
| 2   | 46050  | 30280 | 46050   | 46050 | 46050 | 46050 |
| 3   | 30281  | 46050 | 30281   | 30281 | 30281 | 30281 |
| 4   | 60455  | 56039 | 30278   | 30278 | 30278 | 30278 |
| 5   | 60758  | 56183 | 60653   | 60653 | 60653 | 60653 |
| 6   | 30278  | 56038 | 57188   | 60455 | 60455 | 60455 |
| 7   | 60639  | 30281 | 60639   | 60639 | 60639 | 60639 |
| 8   | 60653  | 30278 | 60630   | 60639 | 60630 | 60639 |
| 9   | 45536  | 56168 | 56039   | 60455 | 60758 | 60630 |
| 10  | 56039  | 57188 | 60455   | 57188 | 56039 | 56039 |
| 11  | 60630  | 56176 | 56169   | 60639 | 60758 | 56039 |
| 12  | 56169  | 56173 | 56038   | 56183 | 56169 | 56169 |
| 13  | 57188  | 60653 | 45536   | 56038 | 45536 | 45536 |
| 14  | 56183  | 56146 | 60758   | 60676 | 60676 | 56038 |
| 15  | 60676  | 60426 | 60676   | 60758 | 60758 | 60676 |
| 16  | 60431  | 30279 | 60435   | 60279 | 56183 | 56183 |
| 17  | 56038  | 56035 | 60435   | 56168 | 60435 | 60435 |
| 18  | 60435  | 60630 | 60466   | 30279 | 56183 | 60435 |
| 19  | 30279  | 60606 | 60606   | 45536 | 60606 | 60606 |
| 20  | 30229  | 56104 | 60638   | 60606 | 30229 | 60606 |

**Fig. 5.5.** Enron: The top 20 ranked nodes given by the left Perron vector of $A$ and of $M_\ell = A_{(i,j)}A_{(i,j)}$ for $\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}$. The columns of $A$ are sampled as described in Section 2.
vector of partially known matrices. In the computed examples, we considered the situation when only fairly small subsets of columns, or of rows, or both, are known.

There are two distinct advantages to the approaches developed here:

1. It is much cheaper, computationally, than evaluating the matrix functions, or computing the Perron vector, of the entire matrix, especially if the matrix is large in size.

2. The methods described correspond to a compelling sampling strategy when obtaining the full adjacency information of a network is prohibitively costly.

In many realistic scenarios, the easiest way to collect information about a network is to access nodes (e.g., individuals) and interrogating them about the other nodes they are connected to. This version of sequential sampling is described in Section 6.

Finally, in order to illustrate the feasibility of our techniques, we have shown how to approximate well-known node centrality measures for large networks, obtaining quite good approximate node rankings, by using only a few columns and rows of the underlying adjacency matrix.

REFERENCES
| Rank | \( A \)       | \( M_\ell \)       |
|------|---------------|-------------------|
| 1    | 1887          | 1887              |
| 2    | 1886          | 1886              |
| 3    | 4599          | 7207              |
| 4    | 2380          | 5929              |
| 5    | 680           | 5929              |
| 6    | 7207          | 2380              |
| 7    | 5929          | 3596              |
| 8    | 4600          | 4639              |
| 9    | 3596          | 5932              |
| 10   | 769           | 5933              |
| 11   | 5932          | 4596              |
| 12   | 4596          | 5930              |
| 13   | 4601          | 5933              |
| 14   | 4639          | 5931              |
| 15   | 5933          | 4601              |
| 16   | 3668          | 680               |
| 17   | 4637          | 6341              |
| 18   | 6758          | 1131              |
| 19   | 5930          | 9609              |
| 20   | 1885          | 6340              |

**Fig. 5.7.** Cond-mat-2005: The top 20 ranked nodes determined by the Perron vectors of \( A \) and of \( M_\ell = A_{(\cdot,j)}A_{(j,\cdot)} \) for \( \ell \in \{500, 1000, 1500, 2000, 2500, 3000\} \). The columns of \( A \) are sampled as described in Section 3.

[1] B. Beckermann and L. Reichel, Error estimation and evaluation of matrix functions via the Faber transform, SIAM J. Numer. Anal., 47 (2009), pp. 3849–3883.
[2] M. Benzi and C. Klymko, Total communicability as a centrality measure, J. Complex Networks, 1 (2013), pp. 124–149.
[3] C. Brezinski and M. Redivo-Zaglia, Rational extrapolation for the PageRank vector, Math. Comp., 77 (2008), pp. 1585–1598.
[4] C. Brezinski and M. Redivo-Zaglia, The simplified topological \( \varepsilon \)algorithms for accelerating sequences in a vector space, SIAM J. Sci. Comput., 36 (2014), pp. A2227–A2247.
[5] C. Brezinski and M. Redivo-Zaglia, The genesis and early developments of Aitkens process, Shanks transformation, the \( \varepsilon \)algorithm, and related fixed point methods, Numer. Algorithms, 80 (2019), pp. 11–133.
[6] S. Brin and L. Page, The anatomy of a large-scale hypertextual web search engine, Comput. Networks ISDN Systems, 30 (1998), pp. 107–117.
[7] D. Calvetti and L. Reichel, Lanczos-based exponential filtering for discrete ill-posed problems, Numer. Algorithms, 29 (2002), pp. 45–65.
[8] S. Cipolla, M. Redivo-Zaglia, and F. Tudisco, Shifted and extrapolated power methods for tensor \( p \)-eigenpairs, Electron. Trans. Numer. Anal., 53 (2020), pp. 1–27.
[9] J. J. Crofts, E. Estrada, D. J. Higham, and A. Taylor, Mapping directed networks, Electron. Trans. Numer. Anal., 37 (2010), pp. 337–350.
[10] A. Cruciani, D. Pasquini, G. Amati, and P. Vocca, About Graph Index Compression Techniques, Proceedings of the 10th Italian Information Retrieval Workshop (IIR-2019), Padua, Italy, September 16-18, 2019, CEUR-WS.org/Vol-2441/paper23.pdf.
| Rank | 500    | 1000   | 1500   | 2000   | 2500   | 3000   |
|------|--------|--------|--------|--------|--------|--------|
| 1    | 1887   | 788    | 1887   | 2380   | 1887   | 1887   |
| 2    | 1886   | 1169   | 1804   | 2380   | 1887   | 1887   |
| 3    | 4599   | 9198   | 1169   | 1311   | 680    | 680    |
| 4    | 2380   | 1804   | 9198   | 7207   | 1886   | 4599   |
| 5    | 680    | 37779  | 3317   | 3668   | 5930   | 4600   |
| 6    | 7207   | 10048  | 6648   | 2645   | 3237   |
| 7    | 5929   | 16559  | 7680   | 4600   | 3668   | 11250  |
| 8    | 4600   | 38962  | 34270  | 1161   | 3596   |
| 9    | 3596   | 787    | 39239  | 10596  | 5929   |
| 10   | 769    | 7680   | 8946   | 2023   | 5932   |
| 11   | 5932   | 14025  | 10048  | 3237   | 4599   |
| 12   | 4596   | 1070   | 14025  | 1885   | 5933   |
| 13   | 4601   | 7849   | 16559  | 379    | 3236   |
| 14   | 4639   | 8946   | 38962  | 5930   |
| 15   | 5933   | 39239  | 787    | 1322   | 7207   |
| 16   | 3668   | 34270  | 1070   | 4600   | 5933   |
| 17   | 4637   | 38712  | 7849   | 372    | 4596   |
| 18   | 6758   | 3317   | 37779  | 3234   |
| 19   | 5930   | 6648   | 38712  | 5053   |
| 20   | 1885   | 786    | 605    | 1886   |

Fig. 5.8. Cond-mat-2005: The top 20 ranked nodes determined by the Perron vectors of $A$ and of $M_{\ell} = A_{(\cdot, J)} A_{(J, \cdot)}$ for $\ell \in \{500, 1000, 1500, 2000, 2500, 3000\}$. The columns of $A$ are sampled randomly.

[11] O. De la Cruz Cabrera, M. Matar, and L. Reichel, Analysis of directed networks via the matrix exponential, J. Comput. Appl. Math., 355 (2019), pp. 182–192.
[12] V. Druskin, L. Knizhnerman, and M. Zaslavsky, Solution of large scale evolutionary problems using rational Krylov subspaces with optimized shifts, SIAM J. Sci. Comput., 31 (2009), pp. 3760–3780.
[13] E. Estrada and D. J. Higham, Network properties revealed through matrix functions, SIAM Rev., 52 (2010), pp. 696–714.
[14] E. Estrada, The Structure of Complex Networks, Oxford University Press, Oxford, 2012.
[15] C. Fenu, D. Martin, L. Reichel, and G. Rodriguez, Block Gauss and anti-Gauss quadrature with application to networks, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1655–1684.
[16] K. Frederix and M. Van Barel, Solving a large dense linear system by adaptive cross approximation, J. Comput. Appl. Math., 234 (2010), pp. 3181–3195.
[17] G. H. Golub and C. F. Van Loan, Matrix Computations, 4th ed., Johns Hopkins University Press, Baltimore, 2013.
[18] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, A theory of pseudo-skeleton approximation, Linear Algebra Appl., 261 (1997), pp. 1–21.
[19] S. A. Goreinov, E. E. Tyrtyshnikov, and N. L. Zamarashkin, Pseudo-skeleton approximations by matrices of maximal volume, Math. Notes, 62 (1997), pp. 515–519.
[20] N. J. Higham, Functions of Matrices: Theory and Computation, SIAM, Philadelphia, 2008.
[21] K. Jbilou and H. Sadok, LUimplementation of the modified minimal polynomial extrapolation method,IMA J. Numer. Anal., 19 (1999), pp. 549–561.
[22] K. Jbilou and H. Sadok, Vector extrapolation methods. Applications and numerical comparison,
[23] A. N. Langville and C. D. Meyer, Googles Pagerank and Beyond, Princeton University Press, Princeton, 2006.
[24] J. Leskovec, J. Kleinberg, and C. Faloutsos, Graph evaluation: Densification and shrinking diameters, ACM Trans. Knowledge Discovery from Data, 1(1) (2007), Art. 2, pp. 1–41.
[25] T. Mach, L. Reichel, M. Van Barel, and R. Vandebril, Adaptive cross approximation for ill-posed problems, J. Comput. Appl. Math., 303 (2016), pp. 206–217.
[26] M. E. J. Newman, Network Data, http://www-personal.umich.edu/~mejn/netdata/.
[27] M. E. J. Newman, Networks: An Introduction, Oxford University Press, Oxford, 2010.
[28] M. E. J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA, 98 (2001), pp. 404–409.
[29] S. Pozza, M. S. Pranić, and A. Strakoš, The Lanczos algorithm and complex Gauss quadrature, Electron. Trans. Numer. Anal., 48 (2018), pp. 362–372.
[30] M. Richardson and R. Agrawal and P. Domingos, Trust Management for the semantic web, in The Semantic Web - ISWC 2003, eds. D. Fensel, K. Sycara, and J. Mylopoulos, Lecture Notes in Computer Science, vol. 2870, Springer, Berlin, pp. 351–368.
[31] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd ed., SIAM, Philadelphia, 2003.
[32] Stanford Large Network Dataset Collection. http://snap.stanford.edu/data/index.html
[33] SuiteSparse Matrix Collection, https://sparse.tamu.edu.
[34] L. L. Trefethen, D. Bau III, Numerical Linear Algebra, SIAM, Philadelphia, 1997.
[35] G. Wu, Y. Zhang, and Y. Wei, Accelerating the Arnoldi-type algorithm for the PageRank problem and the ProteinRank problem, J. Sci. Comput., 57 (2013), pp. 74–104.