




Article

SoftNet: A Package for the Analysis of Complex Networks

Caterina Fenu ^{1,†} , Lothar Reichel ^{2,†}  and Giuseppe Rodriguez ^{1,*,†} 

¹ Department of Mathematics and Computer Science, University of Cagliari, Via Ospedale, 72, 09124 Cagliari, Italy

² Department of Mathematical Sciences, Kent State University, Kent, OH 44242, USA

* Correspondence: rodriguez@unica.it; Tel.: +39-070-675-5617

† These authors contributed equally to this work.

Abstract: Identifying the most important nodes according to specific centrality indices is an important issue in network analysis. Node metrics based on the computation of functions of the adjacency matrix of a network were defined by Estrada and his collaborators in various papers. This paper describes a MATLAB toolbox for computing such centrality indices using efficient numerical algorithms based on the connection between the Lanczos method and Gauss-type quadrature rules.

Keywords: complex network analysis; centrality measure; matrix function; Lanczos algorithm

1. Introduction

Let G be a connected, undirected, unweighted graph with a large number of nodes n and significantly fewer than n^2 edges. We assume there are no self-loops or multiple edges in G . Networks represented by such kinds of graph are found in many applications, such as epidemiology, genetics, telecommunications, and energy distribution; see [1–4]. It is usual to associate to the graph G a symmetric adjacency matrix $A = [A_{ij}] \in \mathbb{R}^{n \times n}$ with entries $A_{ij} = 1$, if nodes i and j are connected by an edge, and $A_{ij} = 0$, otherwise.

It is often meaningful to extract from a large graph numerical values describing global properties of the graph, such as the ease of traveling between vertices, or the importance of a chosen node. A walk in a network is an ordered list of nodes such that successive entries of the list are connected. A well-known fact in graph theory is that the number of walks of length $m \geq 1$ starting at node i and ending at node j is given by $[A^m]_{ij}$, i.e., the entry (i, j) of the m -th power of the adjacency matrix. Let us assume that the coefficients c_m in the matrix-valued function

$$f(A) = \sum_{m=0}^{\infty} c_m A^m \quad (1)$$

are nonnegative and decay fast enough to ensure convergence of the series. Then, the ease of traveling between the nodes i and j can be measured by $[f(A)]_{ij}$, with $i \neq j$, while the importance of node i can be quantified by $[f(A)]_{ii}$.

A common choice (see [2,3,5,6]) is to set the coefficients c_m in (1) to be nonincreasing positive functions of m , with the aim of attributing less importance to long walks than to short ones. For example, $c_m = 1/m!$ [7] yields the matrix exponential

$$f(A) = \exp(A), \quad (2)$$

while setting $c_m = \alpha^m$, with $0 < \alpha < (\rho(A))^{-1}$, where $\rho(A)$ denotes the spectral radius of A , leads to the resolvent

$$f(A) = (I - \alpha A)^{-1}. \quad (3)$$

Let $\mathbf{e} = [1, 1, \dots, 1]^T \in \mathbb{R}^n$ and let $\mathbf{e}_i = [0, \dots, 0, 1, 0, \dots, 0]^T \in \mathbb{R}^n$ be the axis vector with the i th component equal to 1. As usual, the superscript T denotes transposition. The



Citation: Fenu, C.; Reichel, L.; Rodriguez, G. SoftNet: A Package for the Analysis of Complex Networks. *Algorithms* **2022**, *15*, 296. <https://doi.org/10.3390/a15090296>

Academic Editors: Serge Gaspers and Frank Werner

Received: 29 June 2022

Accepted: 21 August 2022

Published: 23 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

following definitions, which are discussed in [2,3,5,6,8,9], are motivated by the discussion above:

- the *degree* of node i , given by $[A\mathbf{e}]_i = \mathbf{e}_i^T A \mathbf{e}$, provides a measure of the importance of node i ;
- the *f-subgraph centrality* of node i , defined by

$$[f(A)]_{ii} = \mathbf{e}_i^T f(A) \mathbf{e}_i, \tag{4}$$

furnishes a more sophisticated measure of the importance of node i than its degree;

- the *f-communicability* between nodes i and j ,

$$[f(A)]_{ij} = \mathbf{e}_i^T f(A) \mathbf{e}_j, \tag{5}$$

quantifies the ease of traveling between nodes i and j ;

- the *f-starting convenience* of node i , given by

$$\frac{\mathbf{e}_i^T f(A) \mathbf{e}}{\mathbf{e}^T f(A) \mathbf{e}}, \tag{6}$$

quantifies the ease of traveling from node i to anywhere in the network. This is the sum of the communicabilities from node i to all other nodes, scaled so that the sum of the quantity over all nodes is one.

Please note that all the centrality measures (4)–(6) are of the form

$$\mathbf{u}^T f(A) \mathbf{v} \tag{7}$$

for specific vectors \mathbf{u} and \mathbf{v} . The purpose of this paper is to present a software package that makes it easy to compute the above defined centrality measures, whose use and methods for their computation have received considerable attention in the literature; see [2,3,5–22] as well as many other references. In these references, many real applications are discussed.

When the adjacency matrix A is large, i.e., when the graph G has many nodes, direct evaluation of $f(A)$ generally is not feasible. Benzi and Boito [11] applied pairs of Gauss and Gauss–Radau rules to compute upper and lower bounds for selected entries of $f(A)$. This work is based on the connection between the symmetric Lanczos process, orthogonal polynomials, and Gauss-type quadrature, explored by Golub and his collaborators in many publications; see Golub and Meurant [23,24] for details and references. A brief review of this technique is provided in Section 2. An application of pairs of block Gauss-type quadrature rules to simultaneously determine approximate upper and lower bounds of expressions of the form (7) when \mathbf{u} and \mathbf{v} are “block vectors”, i.e., matrixes with many rows and very few columns, is described in [9].

The main drawback of quadrature-based methods is that the computational effort is proportional to the number of desired bounds. Therefore, these methods may be expensive to use when bounds for many expressions of the form (7) are to be evaluated. This situation arises, for instance, when we would like to determine one or a few nodes with the largest f -subgraph centrality in a large graph, because this requires the computation of upper and lower bounds for all diagonal entries of $f(A)$.

A method to produce upper and lower bounds for quantities of the form (7) was proposed in [20]. It is based on that knowledge of a few leading eigenvalue-eigenvector pairs gives bounds for every entry of $f(A)$, with little computational effort in addition to computing the eigenvalue-eigenvector pairs. For example, determining the m most important nodes of a graph, with m much smaller than the number of nodes n , amounts to finding the m nodes with the largest f -subgraph centrality. It is possible to quickly evaluate bounds for all entries $[f(A)]_{ij}$ if a partial spectral factorization of A is available. Using these bounds, we can determine a set of $\ell \geq m$ nodes containing the m nodes of

interest, and compute tighter bounds for the nodes in this set, if necessary, by employing Gauss-type quadrature rules. When $\ell \ll n$, the complexity of this hybrid algorithm is much smaller than computing upper and lower bounds for all entries $[f(A)]_{ii}, i = 1, \dots, n$, by Gauss quadrature.

In this work, we present a MATLAB package for the identification of the m most important nodes according to the centrality/communicability indices discussed above, based on two matrix functions, namely the exponential (2) and the resolvent (3). Either the f -subgraph centrality, the f -communicability, or the f -starting convenience can be computed. The computation can be performed using one of three different methods: Gauss quadrature, partial spectral factorization, or the hybrid method; the latter two algorithm have been introduced in [20].

This paper is organized as follows. Section 2 recalls how upper and lower bounds for quantities of the form (7) can be determined via Gauss quadrature. Approximation via partial spectral factorization of A is discussed in Section 3 and the hybrid method is summarized in Section 4. Section 5 presents the SoftNet package as well as a graphical user interface (GUI) that simplifies its use. A brief description of the code and its use also is provided. Section 6 describes some numerical experiments and Section 7 contains concluding remarks.

2. Approximation by Gauss Quadrature

Let A be a symmetric matrix of order n and suppose that we are interested in computing bounds for bilinear forms

$$F_{\mathbf{u},\mathbf{v}}(A) := \mathbf{u}^T f(A) \mathbf{v}, \tag{8}$$

where \mathbf{u} and \mathbf{v} are given vectors and f is a smooth function defined on an interval $[a, b] \subset \mathbb{R}$ that contains the spectrum of A . Since

$$F_{\mathbf{u},\mathbf{v}}(A) = \frac{1}{4}(F_{\mathbf{u}+\mathbf{v},\mathbf{u}+\mathbf{v}}(A) - F_{\mathbf{u}-\mathbf{v},\mathbf{u}-\mathbf{v}}(A)),$$

we can focus on the case $\mathbf{u} = \mathbf{v}$.

The matrix A has the spectral decomposition $A = Q\Lambda Q^T$. Then we can write

$$F_{\mathbf{u},\mathbf{u}}(A) = \mathbf{u}^T Q f(\Lambda) Q^T \mathbf{u} = \boldsymbol{\mu}^T f(\Lambda) \boldsymbol{\mu} = \sum_{i=1}^n f(\lambda_i) \mu_i^2 = \int_a^b f(\lambda) d\mu(\lambda), \tag{9}$$

i.e., we may regard $F_{\mathbf{u},\mathbf{u}}(A)$ as a Stieltjes integral; see [20,24] for further details. We approximate this integral by Gauss-type quadrature rules as follows. Let \mathbf{u} be of unit Euclidean norm. Application of k steps of the Lanczos algorithm to A with initial vector \mathbf{u} gives the $k \times k$ symmetric tridiagonal matrix T_k . It can be shown that $\mathbf{e}_1^T f(T_k) \mathbf{e}_1$ is a k -node Gauss quadrature rule \mathcal{G}_k for the Stieltjes integral (9). A $(k + 1)$ -node Gauss–Radau quadrature formula $\hat{\mathcal{G}}_{k+1}$ with a fixed node at $\theta \geq \rho(A)$ for approximating the Stieltjes integral also can be defined. This discussion assumes that the Lanczos algorithm does not break down. Breakdown is very rare and allows the computations to be simplified.

Under the assumption that the derivatives of $f(x)$ have constant sign on the convex hull of the support of the measure, which is met by the functions (2) and (3), and the Radau node θ is suitably chosen, pairs of Gauss and Gauss–Radau rules furnish lower and upper bounds of increasing accuracy for the quadratic form (9). For the functions (2) and (3), and the Radau node θ chosen as described, we have

$$\mathcal{G}_k \leq \mathcal{G}_{k+1} \leq F_{\mathbf{u},\mathbf{u}}(A) \leq \hat{\mathcal{G}}_{k+2} \leq \hat{\mathcal{G}}_{k+1}.$$

For a user-chosen accuracy τ , we terminate the iterations with the Lanczos algorithm when

$$\frac{|\mathcal{G}_k - \hat{\mathcal{G}}_{k+1}|}{|\mathcal{G}_k|} \leq \tau. \tag{10}$$

The default value in the code is $\tau = 10^{-3}$.

The matrix functions are applied to the tridiagonal matrixes using their spectral factorization. Thus, let $T_k = W_k \Lambda_k W_k^T$ be the spectral factorization. Then $f(T_k) = W_k f(\Lambda_k) W_k^T$. When f is the exponential function, we let μ be the largest eigenvalue of T_k and evaluate

$$e^{-\mu} e^{T_k} = e^{T_k - \mu I} = W e^{\Lambda_k - \mu I} W^T \tag{11}$$

instead of e^{T_k} to avoid overflow.

Regarding the choice of the Radau node θ , we often may let $\theta = \|A\|_\infty$. Alternatively, we can use the MATLAB function `eigs` or the function `irbleigs` described in [25,26] to determine an estimate of the largest eigenvalue λ_1 of A .

3. Bounds via Partial Spectral Factorization

This section recalls how to derive bounds for expressions of the form (8), with $\|\mathbf{u}\| = \|\mathbf{v}\| = 1$, using a partial spectral factorization of A . Introduce the spectral factorization

$$A = Q \Lambda Q^T,$$

where the eigenvector matrix $Q = [\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n] \in \mathbb{R}^{n \times n}$ is orthogonal and the eigenvalues in the diagonal matrix $\Lambda = \text{diag}[\lambda_1, \lambda_2, \dots, \lambda_n] \in \mathbb{R}^{n \times n}$ are ordered according to $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Then

$$f(A) = Q f(\Lambda) Q^T = \sum_{k=1}^n f(\lambda_k) \mathbf{q}_k \mathbf{q}_k^T,$$

so that

$$F_{\mathbf{u}, \mathbf{v}}(A) = \mathbf{u}^T f(A) \mathbf{v} = \sum_{k=1}^n f(\lambda_k) \tilde{u}_k \tilde{v}_k,$$

where $\tilde{u}_k = \mathbf{u}^T \mathbf{q}_k$ and $\tilde{v}_k = \mathbf{v}^T \mathbf{q}_k$. Let the first N eigenpairs $\{\lambda_k, \mathbf{v}_k\}_{k=1}^N$ of A be known. Then $F_{\mathbf{u}, \mathbf{v}}(A)$ can be approximated by

$$F_{\mathbf{u}, \mathbf{v}}(A) \approx F_{\mathbf{u}, \mathbf{v}}^{(N)} := \sum_{k=1}^N f(\lambda_k) \tilde{u}_k \tilde{v}_k. \tag{12}$$

The following results from [20] shows how upper and lower bounds for $F_{\mathbf{u}, \mathbf{v}}(A)$ can be determined with the aid of the first N eigenpairs of A .

Theorem 1. *Let the function f be nondecreasing and nonnegative on the convex hull of the spectrum of A and let $F_{\mathbf{u}, \mathbf{v}}^{(N)}$ be defined by (12). Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N$ be the N largest eigenvalues of A and let $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_N$ be associated orthonormal eigenvectors. Then we have*

$$L_{\mathbf{u}, \mathbf{v}}^{(N)} \leq F_{\mathbf{u}, \mathbf{v}}(A) \leq U_{\mathbf{u}, \mathbf{v}}^{(N)}, \tag{13}$$

where

$$L_{\mathbf{u}, \mathbf{v}}^{(N)} := F_{\mathbf{u}, \mathbf{v}}^{(N)} - f(\lambda_N) \left(1 - \sum_{k=1}^N \tilde{u}_k^2\right)^{1/2} \left(1 - \sum_{k=1}^N \tilde{v}_k^2\right)^{1/2},$$

$$U_{\mathbf{u}, \mathbf{v}}^{(N)} := F_{\mathbf{u}, \mathbf{v}}^{(N)} + f(\lambda_N) \left(1 - \sum_{k=1}^N \tilde{u}_k^2\right)^{1/2} \left(1 - \sum_{k=1}^N \tilde{v}_k^2\right)^{1/2}.$$

When $\mathbf{u} = \mathbf{v}$, we have

$$F_{\mathbf{u}, \mathbf{u}}^{(N)} \leq F_{\mathbf{u}, \mathbf{u}}(A) \leq U_{\mathbf{u}, \mathbf{u}}^{(N)} \tag{14}$$

and

$$F_{\mathbf{u},\mathbf{u}}^{(N)} \leq F_{\mathbf{u},\mathbf{u}}^{(N+1)}, \quad U_{\mathbf{u},\mathbf{u}}^{(N)} \geq U_{\mathbf{u},\mathbf{u}}^{(N+1)}, \quad 1 \leq N < n. \tag{15}$$

To determine which nodes have the largest f -subgraph centrality (4), we use the inequalities (14) and (15). The N leading eigenpairs $\{\lambda_k, \mathbf{q}_k\}_{k=1}^N$ of A and the bounds (13) and (14) can be used to determine a subset of nodes that contains the vertices with the largest value of the node metric we are considering.

Let $L_{\mathbf{u},\mathbf{v}}^{(N)}$ and $U_{\mathbf{u},\mathbf{v}}^{(N)}$ be the lower and upper bounds defined in Theorem 1. Since we seek an approximation of the centrality value for all the nodes of the network, we will either set $\mathbf{u} = \mathbf{v} = \mathbf{e}_i$, as in (4), or $\mathbf{u} = \mathbf{e}$ and $\mathbf{v} = \mathbf{e}_i$, as in (6). So, $F_{\mathbf{u},\mathbf{v}}^{(N)}$ will be a quantity depending on an index $i = 1, \dots, n$. We will write $F_i^{(N)} = F_{\mathbf{u},\mathbf{v}}^{(N)}$ to simplify the notation when we are computing either $F_{\mathbf{e}_i,\mathbf{e}_i}^{(N)}$ or $F_{\mathbf{e},\mathbf{e}_i}^{(N)}$. When approximating (5), we will fix a value of j and consider $F_i^{(N)} = F_{\mathbf{e}_j,\mathbf{e}_i}^{(N)}$ for $i = 1, \dots, n$.

Let $\mathcal{F}_m^{(N)}$ denote the m th largest value of the vector $(F_i^{(N)})_{i=1}^n$. Then the index sets

$$S_m^{(N)} = \left\{ i : U_{\mathbf{u},\mathbf{v}}^{(N)} \geq \mathcal{F}_m^{(N)} \right\}, \quad N = 1, 2, \dots, n. \tag{16}$$

contains the indices of the nodes that can be considered important with respect to the desired centrality index.

A computational difficulty to overcome is that we do not know in advance how the dimension N of the leading invariant subspace $\text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_N\}$ of A should be chosen in order to obtain useful bounds (13) or (14). We use the restarted block Lanczos method `irbleigs` described in [25,26], which computes the leading invariant subspace $\{\lambda_k, \mathbf{q}_k\}_{k=1}^\ell$ of A for a user-chosen dimension ℓ , and allows the extension of such subspace by successively increasing the value of ℓ . Using `irbleigs`, we compute more and more eigenpairs of A until N is such that

$$|S_m^{(N)}| = m, \tag{17}$$

where $|S|$ denotes the number of elements of the set S . This stopping criterion is referred to as the *strong convergence condition*. As shown in [20], the set $S_m^{(N)}$ contains the indices of the m nodes with the largest f -subgraph centrality.

The criterion (17) for choosing N is useful if the required value of N is not too large. The *weak convergence criterion* has been introduced to be used for problems for which a large value of N is required in order to satisfy (17), and this makes it impractical to compute the associated bounds (13). The weak convergence criterion is also well suited for applications in the hybrid algorithm described in Section 4. This criterion is designed to stop increasing N when the values $F_{\mathbf{u},\mathbf{v}}^{(N)}$ do not increase significantly with N . Specifically, we stop increasing N when the average increment of the values in the vector $F_{\mathbf{u},\mathbf{v}}^{(N)}$ is small when the N th eigenpair $\{\lambda_N, \mathbf{q}_N\}$ is included in the bounds. The average contribution of this eigenpair to $F_{\mathbf{u},\mathbf{v}}^{(N)}$, $1 \leq i \leq n$, is

$$F_i = f(\lambda_N)[\tilde{\mathbf{u}}_N]_i[\tilde{\mathbf{v}}_N]_i$$

see (12), and we stop increasing N when

$$\frac{1}{n} \sum_{i=1}^n F_i < \tau \cdot \mathcal{F}_m^{(N)} \tag{18}$$

for a user-specified tolerance τ , whose default value in the code is 10^{-3} . Please note that when this criterion is satisfied, but not (17), the nodes with index in $S_m^{(N)}$ and with the

largest value $F_{u,v}^{(N)}$ are not guaranteed to be the nodes with the largest index value we are searching for.

Furthermore, the weak convergence criterion (18) may yield a set $S_m^{(N)}$ with many more than m indices. In particular, we may not be willing to compute accurate bounds for a specific node metric by applying the approach of Section 2 to all nodes with index in $S_m^{(N)}$. We therefore describe how to determine a smaller index set \mathcal{J} , which is likely to contain the indices of the m most important nodes. We discard from the set $S_m^{(N)}$ indices for which $F_{u,v}^{(N)}$ is much smaller than $\mathcal{F}_m^{(N)}$. Thus, for a user-chosen parameter $\sigma > 0$, we include in the set \mathcal{J} all indices $i \in S_m^{(N)}$ such that

$$\mathcal{F}_m^{(N)} - F_{u,v}^{(N)} < \sigma \cdot \mathcal{F}_m^{(N)}.$$

The default value for σ in the software is 10^{-3} .

4. The Hybrid Method

We summarize here the algorithm corresponding to the hybrid method. The first step is to compute a partial spectral factorization of the adjacency matrix A . Such a partial factorization makes it possible to determine a set of candidate nodes that contains the most important nodes according to a chosen criterion, e.g., the f -subgraph centrality. The accuracy of upper and lower bounds for the candidate nodes is then improved by a suitable application of Gauss and Gauss–Radau quadrature rules.

5. The SoftNet Software Package

The package SoftNet for MATLAB is available at the web page <http://bugs.unica.it/cana/software> (accessed on 20 August 2022) as a compressed archive. Uncompressing it, a directory named SoftNet will be created; in order to use the package the user should add its name to the search path. The package SoftNet consists of 14 MATLAB routines for the identification of the m most important nodes in a network according to different centrality indices. The package also includes the function `irbleigs` from [25,26], and the following 5 adjacency matrixes of real-world networks that can be used to test the software

- karate (34 nodes, 78 edges): represents the social relationships among the 34 individuals of a university karate club [27];
- yeast (2114 nodes, 4480 edges): describes the protein interaction network for yeast [28–30];
- power (4941 nodes, 13,188 edges): undirected representation of the topology of the western states power grid of the United States [27,31];
- internet (22,963 nodes, 96,872 edges): snapshot of the structure of the Internet at the level of autonomous systems from data for 22 July 2006 [27];
- collaborations (40,421 nodes, 351,304 edges): collaboration network of scientists who posted preprints at www.arxiv.org (accessed on 20 August 2022) between 1 January 1995 and 31 March 2005 [27,32];
- facebook (63,731 nodes, 1,545,686 edges): user-to-user links (*friendship*) from the Facebook New Orleans network, studied in [33] and available at [34].

The package Contest by Taylor and Higham [35] contains different kinds of synthetic networks and can be used to generate further numerical tests. We provide a convenient interface to this package.

Table 1 lists the 14 MATLAB routines with a description of their purpose. The first group, “Computational Routines,” includes the functions for computing different centralities (subgraph centrality (4), communicability (5), and starting convenience (6)) with respect to two different matrix functions, the exponential (2) and the resolvent (3). The computations can be performed with three different methods, namely the Gauss quadrature method recalled in Section 2, the low-rank approximation presented in Section 3 and the

hybrid method described in Section 4. The section “Auxiliary Routines for the Graphical User Interface” lists some routines required to start and use the graphical user interface.

Table 1. Routines and GUI.

Computational Routines	
commgauss	Identifies the m most important nodes according to f -communicability of node i through Gauss quadrature; see Section 2.
commhyb	Identifies the m most important nodes according to f -communicability of node i through partial singular value decomposition (SVD) and Gauss quadrature; see Section 4.
commlr	Identifies the m most important nodes according to f -communicability of node i through partial SVD; see Section 3.
gaussexp	Computes the bilinear form $\mathbf{u}^T f(A)\mathbf{v}$, with $f(A) = \exp(A)$ through Gauss quadrature; see Section 2.
gaussres	Computes the bilinear form $\mathbf{u}^T f(A)\mathbf{v}$, with $f(A) = (I - \alpha A)^{-1}$ through Gauss quadrature; see Section 2.
sgcengauss	Identifies the m most important nodes according to f -subgraph centrality through Gauss quadrature; see Section 2.
sgcenhyb	Identifies the m most important nodes according to f -subgraph centrality through partial SVD and Gauss quadrature; see Section 4.
sgcenlr	Identifies the m most important nodes according to f -subgraph centrality through partial SVD; see Section 3.
stconvgauss	Identifies the m most important nodes according to f -starting convenience through Gauss quadrature; see Section 2.
stconvhyb	Identifies the m most important nodes according to f -starting convenience through partial SVD and Gauss quadrature; see Section 4.
stconvlr	Identifies the m most important nodes according to f -starting convenience through partial SVD; see Section 3.
Auxiliary Routines for the Graphical User Interface	
vipnodes	Starts the graphical user interface.
compute	Performs the computations according to the chosen parameters.
choose_contest	Allow selection of a synthetic network from the Contest Package [35].
initialize_gui	Initializes the default settings for the graphical user interface.

The computational routines are totally independent of the graphical user interface and can be used by the user from the MATLAB command line. For example, the command

```
[vip, vipsgc] = sgcenlr(A, 'exp', 10);
```

identifies the 10 most important nodes according to the subgraph centrality when the low-rank approximation is used for the computation. vip and vipsgc are vectors containing the indices of the nodes that are candidates to being the most important nodes and the values of their subgraph centrality, respectively.

Identifying the five most important nodes of a network whose adjacency matrix is A with respect to the starting convenience can be done by the following lines of code

```
func = 'exp'; % the function to be used
nnodes = 5; % the number of nodes to be identified
theta = eigs(double(A), 1, 'LA'); % estimation of the largest eigenvalue
opts = struct('gausstolq', 1e-5, 'gaussmaxn', 150, 'gaussmu', theta, 'show', 1)
[vip, vipsgc, info, iters, allstconv] = stconvgauss(A, func, nnodes, opts);
```

The third line computes the largest eigenvalue, since its estimation is needed for the computation of the Gauss–Radau rule. The struct opts is initialized on the fifth line, where the tolerance (10) for the convergence of Gauss quadrature is chosen, as well as the maximum number of iterations, and the value μ used for the spectrum shift (11). Setting the show variable to 1 displays a waitbar during the computations.

The output values are:

- `vip`: indices for the most important nodes;
- `vipsgc`: values of starting convenience for the identified nodes;
- `info`: a vector containing a flag that indicates convergence and shows the number of matrix-vector products;
- `iters`: the number of iterations performed for each node;
- `allstconv`: the values of the starting convenience for each node.

Table 2 reports a subset of the options used for tuning the performance of the package; all the options have a default value. Refer to the second column of the table and to the description of the algorithms in [20] for their meaning. The available options are described in the various functions.

Table 2. Problem definition and options.

Problem Definition	
<code>func</code>	function to be used ((2) or (3))
<code>nnodes</code>	number of nodes to be identified
Fields for the <code>opts</code> variable	
<code>gausstolq</code>	tolerance for Gauss quadrature
<code>gaussmaxn</code>	maximum number of iterations
<code>gaussmu</code>	approximation of the largest eigenvalue for the spectrum shift
<code>alpha</code>	constant for the resolvent
<code>show</code>	if $\neq 0$ shows some information during the computation
<code>sgcmaxva</code>	maximum number of stored eigenvectors
<code>sgctoll</code>	tolerance for weak convergence
<code>sgcshift</code>	if $\neq 0$ applies spectrum shift

All the functions can be used interactively with the `vipnodes` graphical user interface, located in the main directory of the package. The GUI starts by typing the command `vipnodes` in the MATLAB Command Window; see Figure 1.

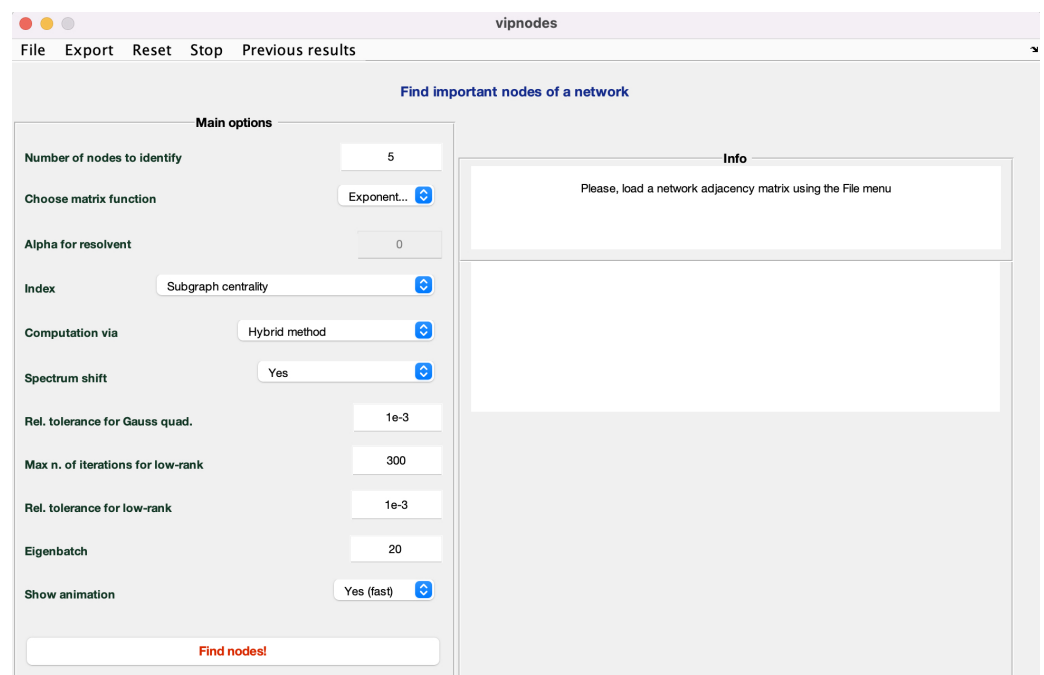


Figure 1. The graphical user interface (GUI).

The GUI consists of one input panel, on the left, and an output area, on the right. The former allows the user to set different parameters to perform the computations, the latter shows some information about the loaded network and the results, once the computations are done. A drop-down menu at the top of the window allows the user to perform different tasks as follows:

- **File.** This menu allows the user to load a network in three different ways: load it from a mat file, extract it from the workspace, and create it using the Contest package [35], if the latter is installed.
- **Export.** This menu allows the user to export the results as a mat file, as a text file, or export them to variables in the workspace.
- **Reset.** Reset options and computed results or just the results.
- **Stop.** Interrupt the computations if they take too long time.
- **Previous results.** Display a table with results of the previous computation.

The first step to complete in order to carry out the computations is to load an adjacency matrix through the “File” menu at the top left of the main window. Once this task is done, general information about the network is shown, namely the number of nodes, the number of edges and, if the network contains self-loops, the number of removed edges. The parameters are set to their default values, and can be modified by the user. By pressing the “Find nodes” button the computations start. If the user chooses to show the animation (this possibility is not given if computation via Gauss quadrature is selected), a new window “Animation” will appear. It contains a spy plot of the adjacency matrix associated to the network with the number of non-zero elements, i.e., the number of edges, shown at the bottom of the figure. Below, the spectrum of A is drawn and the graph is updated once a new set of eigenvalues is computed. On the right, an animation with the computed lower and upper bounds when each new eigenpair is added to the sum (13) is shown. If either the strong or weak convergence criteria are satisfied, then the candidate nodes are highlighted with red circles. The title of the last graph reports the number of used eigenpair and the cardinality of the set $S_m^{(N)}$ defined in (16).

Figure 2 shows a typical animation window. In this case, the computations aim to identify the five most important nodes with respect to the subgraph centrality of the power network included in the package. The computations were carried out by the low-rank method with the strong convergence condition. The number of computed eigenpairs is the minimal integer N such that the cardinality of $S_5^{(N)}$ is 5.

Figure 3 shows the same window after identifying the same number of nodes as in Figure 2 by the hybrid method. In this case, the cardinality of the set $S_5^{(N)}$ is larger than 5, and the final computation to identify the five most important nodes is performed by Gauss quadrature.

Once the computations are made, the main window shows the following information:

- the method used to perform the computation (low-rank with strong convergence, hybrid method or Gauss quadrature);
- the line of code that has to be written on the command window to perform the same computation without using the graphical user interface;
- whether the strong or weak convergence criteria are satisfied (if one of them was selected);
- the number of used eigenpairs (if either the low-rank or hybrid methods have been used for the computations);
- the number of VIP nodes identified (if either the low-rank or hybrid methods have been used);
- the elapsed time;
- a table with the index of the identified nodes and the value of the corresponding centrality index.

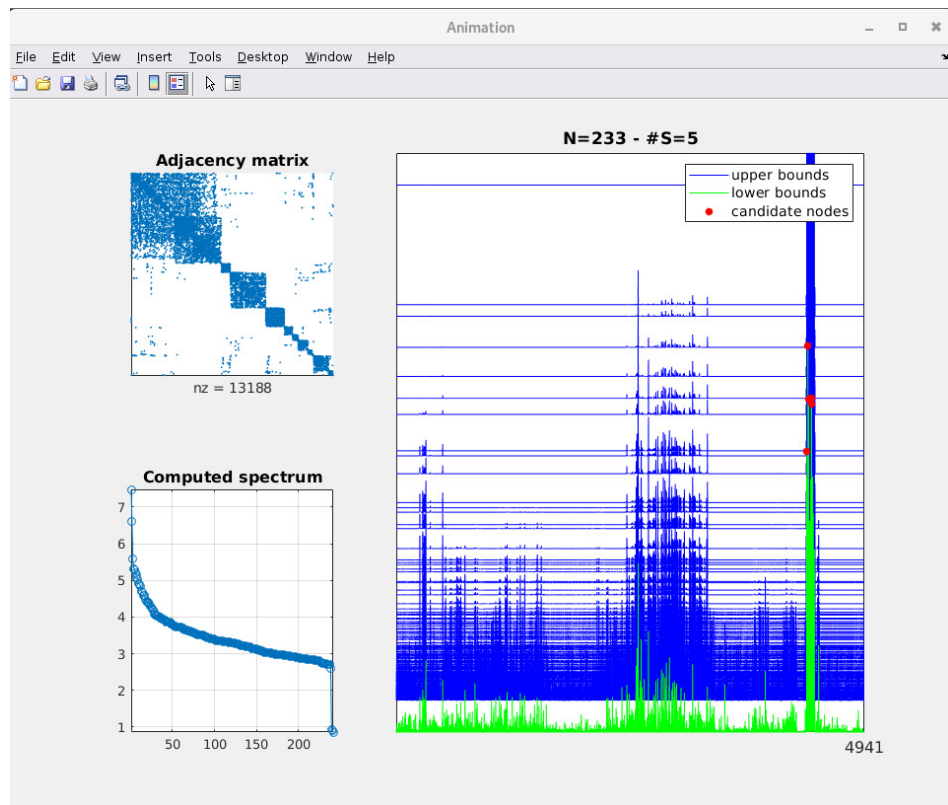


Figure 2. The animation window after the identification of the five most important nodes for the Power network by the low-rank approximation method.

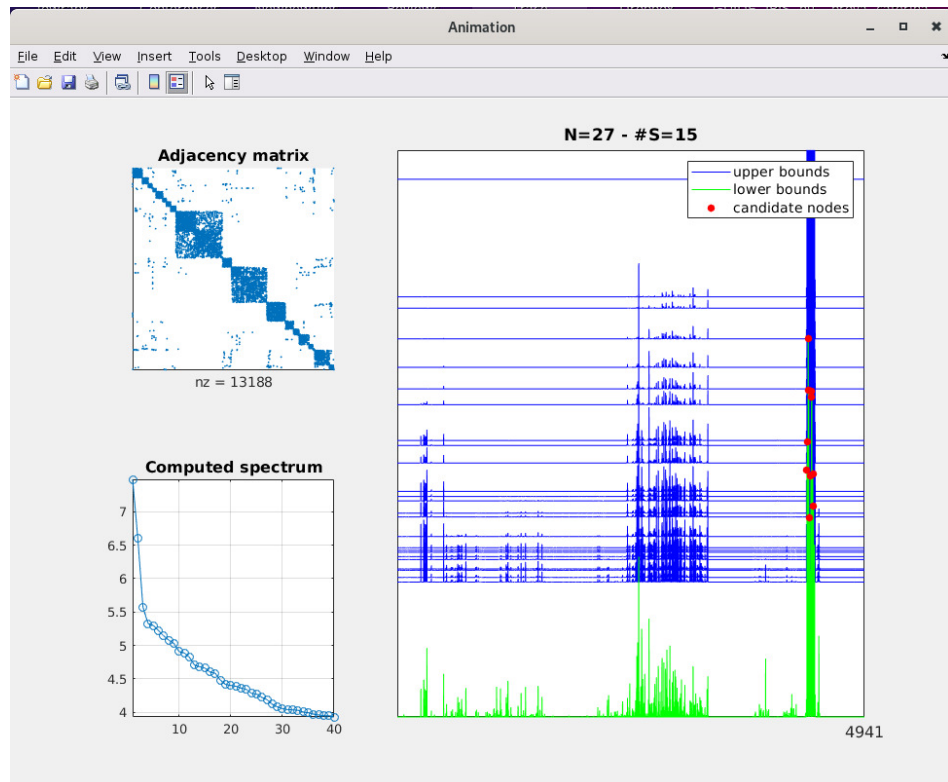


Figure 3. The animation window after the identification of the five most important nodes for the Power network by the hybrid method.

Figure 4 shows the main window once the computations related to Figure 2 have been carried out. Figure 5 shows the same window after the hybrid method has been used.

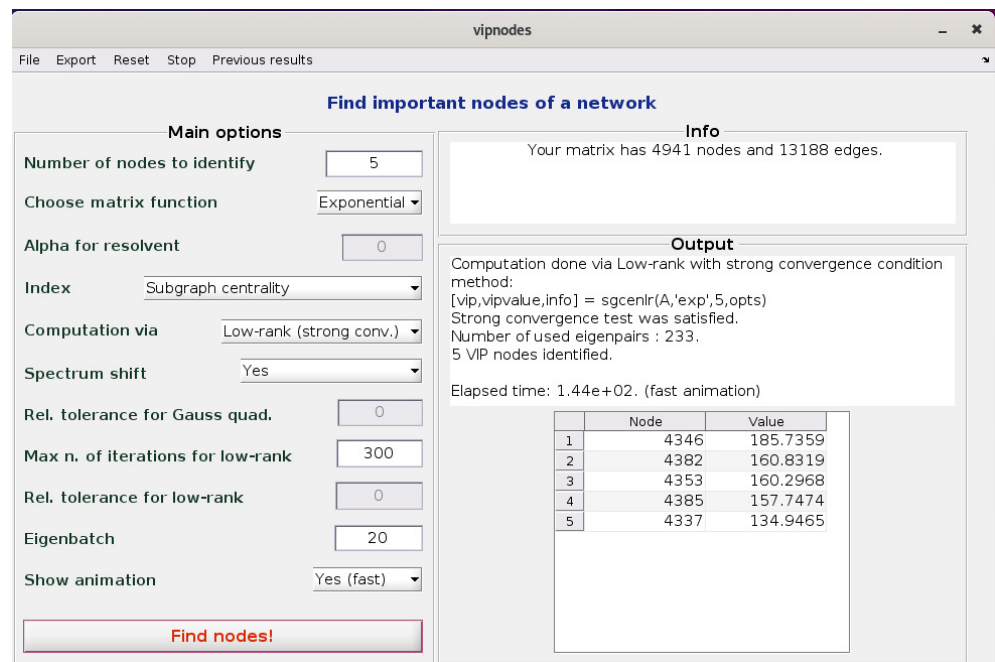


Figure 4. Main window after the identification of the five most important nodes for the Power network by the low-rank approximation method.

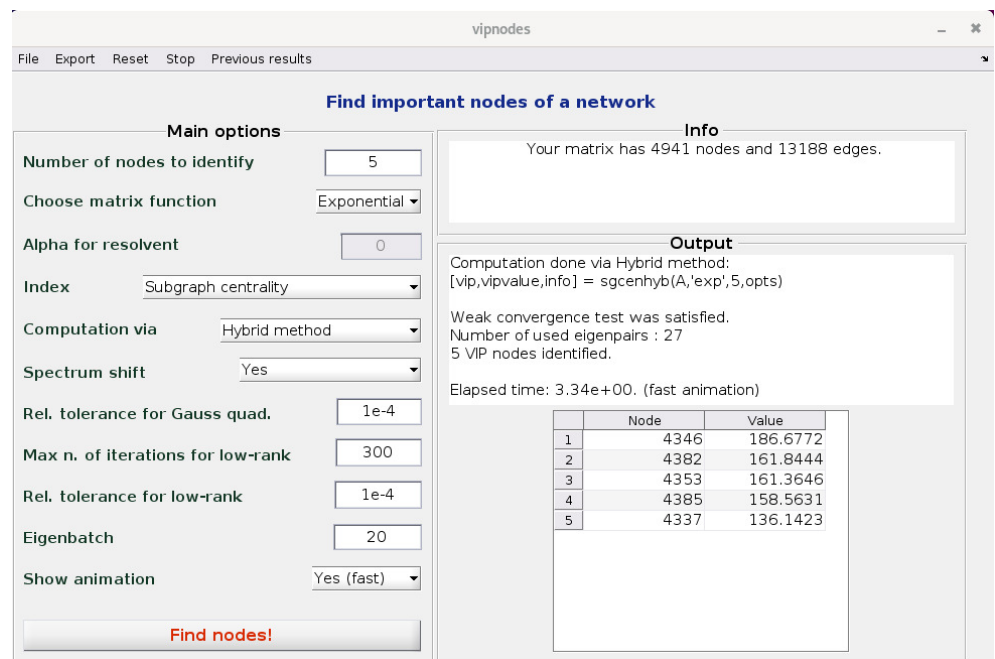


Figure 5. Main window after the identification of the five most important nodes for the Power network by the hybrid method.

Please note that the lists of nodes produced by the two methods is the same, but the values of the centrality index are slightly different. This happens because the value of the subgraph centrality computed by the low-rank approximation is estimated as an average of the lower and upper bounds computed by the method, while the value computed by Gauss quadrature is more accurate.

6. Numerical Experiments

This section provides some numerical experiments to explore the performance of the centrality indices used in the software, namely the f -subgraph centrality and the f -starting convenience. In particular, we compare them to the following well-known centrality indices:

- degree: the number of edges adjacent to a node;
- betweenness: the number of shortest paths that pass through the node;
- closeness: the reciprocal of the sum of the length of the shortest paths between a node and all other nodes in the graph;
- eigenvector: a score is assigned to each node taking into account connections with nodes that have high scores;
- pagerank: a variant of the eigenvector centrality.

The computation of the centrality indices listed above has been done by the centrality function included in Matlab. An example of its usage it is the following:

```
centr = 'betweenness'; % the centrality to be used
nnodes = 5; % the number of nodes to be identified
G = graph(A); % converts the adjacency matrix A in to the graph G
values = centrality(G, centr); % computes all the centralities of graph G
[~, node_ind] = sort(values, 'descend'); % sorts all the centralities
disp(node_ind(1:nnodes)); % displays the index for the nodes with the largest centrality
```

The string `centr` can be set to `degree`, `betweenness`, `closeness`, `eigenvector`, and `pagerank`.

The first network we analyze is the famous Zachary’s karate club network [27]. The most important nodes of the network are node 1 and node 34, which stand for the instructor and the club president, respectively.

Each column of Table 3 reports the ranking of the five most important nodes obtained using the centrality indices listed above, namely degree, betweenness, closeness, eigenvector, and pagerank centralities, compared with the ranking obtained by the exp -subgraph centrality, the res -subgraph centrality, the exp -starting convenience, and the res -starting convenience. The value used for α in (3) is $0.95 \cdot (\rho(A))^{-1}$. We remark that for this example, the ranking is not very sensitive to the choice of the parameter α .

Table 3. Ranking of the five most important nodes for the karate network identified by the centrality function of Matlab, the f -subgraph centrality, and f -starting convenience.

Degree	betw	clos	eigvec	pagerank	exp-sgc	res-sgc	exp-stc	res-stc
34	1	1	34	34	34	34	34	34
1	34	3	1	1	1	1	1	1
33	33	34	3	33	33	33	3	33
3	3	32	33	3	3	3	33	3
2	32	9	2	2	2	2	2	2

It is worth noting that all the centrality indices correctly identify nodes 1 and 34 as the most important ones. The list of the five most important nodes contains the same indices except for the betweenness centrality, which includes in the list node 32, and the closeness centrality, which determines that node 32 and 9 are among the five most important nodes.

The second example we are going to consider is the Facebook network included in the package. The graph has 63,731 nodes and 1,545,686 edges. Neither the exponential nor the resolvent of the adjacency matrix A can be evaluated in a straightforward manner due to the large size of the matrix. We therefore apply the hybrid algorithm described in Section 4 to find the 10 most important nodes in the network.

Table 4 reports in each column the ranking of the 10 most important nodes according to the centrality indices described above and computed by the centrality function of Matlab.

Table 5 reports the ranking of the 10 most important nodes obtained by the *exp*-subgraph, the *res*-subgraph, the *exp*-starting convenience and the *res*-starting convenience. The value used for α in (3) is $0.95 \cdot (\rho(A))^{-1}$ and $0.1 \cdot (\rho(A))^{-1}$.

Table 4. Ranking of the 10 most important nodes for the facebook network identified by the centrality function of Matlab.

Degree	Betweenness	Closeness	Eigenvector	Pagerank
2332	554	2332	9904	554
471	471	471	2322	471
554	2332	23	5170	2332
2322	23	554	5157	23
451	451	1463	2362	451
23	280	207	3943	2208
2208	1463	280	7765	1463
9904	207	451	133	423
1463	84	1996	2332	280
3943	1996	2805	1902	207

It can be seen that the considered centrality indices generally produce different rankings. This confirms that they measure different features of the nodes in a network. It is remarkable to observe that in this example the *exp*-subgraph centrality and the *exp*-starting convenience produce the same list as the eigenvector centrality.

Table 5. Ranking of the 10 most important nodes for the facebook network identified by the *f*-subgraph centrality and *f*-starting convenience.

sgcen_exp	sgcen_res(.95)	sgcen_res(.1)	stconv_exp	stconv_res(.95)	stconv_res(.1)
9904	9904	2332	9904	9904	2332
2322	2322	471	2322	2322	471
5170	5170	451	5170	5170	451
5157	3943	2208	5157	3943	2208
2362	2362	9904	2362	2362	9904
3943	7765	3943	3943	7765	3943
7765	5157	133	7765	5157	133
133	2332	423	133	2332	423
2332	1902	7765	2332	1902	7765
1902	133	14,253	1902	133	14,253

7. Conclusions

This article introduces the SoftNet toolbox written in MATLAB, designed to compute the most important nodes of a network by some centrality indices based on the computation of matrix functions. The methods used to perform the computation were introduced in [20]. The use of the toolbox is illustrated by examples.

Author Contributions: All authors equally contributed to the research that led to this paper. All authors have read and agreed to the published version of the manuscript.

Funding: C.F. and G.R. were partially supported by the Regione Autonoma della Sardegna research project “Algorithms and Models for Imaging Science (AMIS)” [RASSR57257] and the INdAM-GNCS research project “Tecniche numeriche per l’analisi delle reti complesse e lo studio dei problemi inversi”. C.F. gratefully acknowledges Regione Autonoma della Sardegna for the financial support provided under the Operational Programme P.O.R. Sardegna F.S.E. (European Social Fund 2014–2020—Axis III Education and Formation, Objective 10.5, Line of Activity 10.5.12).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data used in the numerical experiments is contained in the software package presented in the paper. Data sources are acknowledged in the text.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

GUI graphical user interface

References

1. Bini, D.A.; Del Corso, G.M.; Romani, F. Evaluating scientific products by means of citation-based models: A first analysis and validation. *Electron. Trans. Numer. Anal.* **2008**, *33*, 1–16.
2. Estrada, E. *The Structure of Complex Networks*; Oxford University Press: Oxford, UK, 2012.
3. Estrada, E.; Hatano, N. Statistical-mechanical approach to subgraph centrality in complex networks. *Chem. Phys. Lett.* **2007**, *439*, 247–251. [[CrossRef](#)]
4. Newman, M.E.J. *Networks: An Introduction*; Oxford University Press: Oxford, UK, 2010.
5. Estrada, E.; Rodríguez-Velázquez, J.A. Subgraph centrality in complex networks. *Phys. Rev. E* **2005**, *71*, 056103. [[CrossRef](#)] [[PubMed](#)]
6. Estrada, E.; Rodríguez-Velázquez, J.A. Subgraph centrality and clustering in complex hyper-networks. *Phys. A* **2006**, *364*, 581–594. [[CrossRef](#)]
7. Estrada, E.; Hatano, N.; Benzi, M. The physics of communicability in complex networks. *Phys. Rep.* **2012**, *514*, 89–119. [[CrossRef](#)]
8. Estrada, E.; Hatano, N. Communicability in complex networks. *Phys. Rev. E* **2008**, *77*, 036111. [[CrossRef](#)]
9. Fenu, C.; Martin, D.; Reichel, L.; Rodriguez, G. Block Gauss and anti-Gauss quadrature with application to networks. *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 1655–1684. [[CrossRef](#)]
10. Aprahamian, M.; Higham, D.J.; Higham, N.J. Matching exponential-based and resolvent-based centrality measures. *J. Complex Netw.* **2016**, *4*, 157–176. [[CrossRef](#)]
11. Benzi, M.; Boito, P. Quadrature rule-based bounds for functions of adjacency matrices. *Linear Algebra Its Appl.* **2010**, *433*, 637–652. [[CrossRef](#)]
12. Bini, D.A.; Del Corso, G.M.; Romani, F. A combined approach for evaluating papers, authors and scientific journals. *J. Comput. Appl. Math.* **2010**, *234*, 3104–3121. [[CrossRef](#)]
13. Benzi, M.; Klymko, C. Total communicability as a centrality measure. *J. Complex Netw.* **2013**, *1*, 124–149. [[CrossRef](#)]
14. Bonacich, P.F. Power and centrality: A family of measures. *Am. J. Sociol.* **1987**, *92*, 1170–1182. [[CrossRef](#)]
15. Crofts, J.J.; Estrada, E.; Higham, D.J.; Taylor, A. Mapping directed networks. *Electron. Trans. Numer. Anal.* **2010**, *37*, 337–350.
16. Crofts, J.J.; Higham, D.J. Googling the brain: Discovering hierarchical and asymmetric network structures, with applications in neuroscience. *Internet Math.* **2011**, *7*, 233–254. [[CrossRef](#)]
17. Del Corso, G.M.; Romani, F. Versatile weighting strategies for a citation-based research evaluation model. *Bull. Belg. Math. Soc. Simon Stevin* **2009**, *16*, 723–743. [[CrossRef](#)]
18. Estrada, E.; Higham, D.J. Network properties revealed through matrix functions. *SIAM Rev.* **2010**, *52*, 696–714. [[CrossRef](#)]
19. Henson, V.E.; Sanders, G. Locally supported eigenvectors and matrices associated with connected and unweighted power-law graphs. *Electron. Trans. Numer. Anal.* **2012**, *39*, 353–378.
20. Fenu, C.; Martin, D.; Reichel, L.; Rodriguez, G. Network analysis via partial spectral factorization and Gauss quadrature. *SIAM J. Sci. Comput.* **2013**, *35*, A2046–A2068. [[CrossRef](#)]
21. Gleich, D.F. PageRank beyond the web. *SIAM Rev.* **2015**, *57*, 321–363. [[CrossRef](#)]
22. Langville, A.N.; Meyer, C.D. *Google's Pagerank and Beyond: The Science of Search Engine Rankings*; Princeton University Press: Princeton, NJ, USA, 2006.
23. Golub, G.H.; Meurant, G. *Matrices, Moments and Quadrature in Numerical Analysis*; Griffiths, D.F., Watson, G.A., Eds.; Longman: Essex, UK, 1994; pp. 105–156.
24. Golub, G.H.; Meurant, G. *Matrices, Moments and Quadrature with Applications*; Princeton University Press: Princeton, NJ, USA, 2010.
25. Baglama, J.; Calvetti, D.; Reichel, L. IRBL: An implicitly restarted block Lanczos method for large-scale Hermitian eigenproblems. *SIAM J. Sci. Comput.* **2003**, *24*, 1650–1677. [[CrossRef](#)]
26. Baglama, J.; Calvetti, D.; Reichel, L. Algorithm 827: Irbleigs: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix. *ACM Trans. Math. Softw.* **2003**, *29*, 337–348. [[CrossRef](#)]
27. Newman, M.E.J. Newman's Web Page. Available online: <http://www-personal.umich.edu/~mejn/netdata/> (accessed on 28 June 2022).
28. Jeong, H.; Mason, S.; Barabási, A.-L.; Oltvai, Z.N. Lethality and centrality of protein networks. *Nature* **2001**, *411*, 41–42. [[CrossRef](#)] [[PubMed](#)]

29. Sun, S.; Ling, L.; Zhang, N.; Li, G.; Chen, R. Topological structure analysis of the protein–protein interaction network in budding yeast. *Nucleic Acids Res.* **2003**, *31*, 2443–2450.
30. Batagelj, V.; Mrvar, A. Pajek Datasets. 2006. Available online: <http://vlado.fmf.uni-lj.si/pub/networks/data/> (accessed on 28 June 2022).
31. Watts, D.J.; Strogatz, S.H. Collective dynamics of ‘small-world’ networks. *Nature* **1998**, *393*, 440–442. [[CrossRef](#)] [[PubMed](#)]
32. Newman, M.E.J. The structure of scientific collaboration networks. *Proc. Natl. Acad. Sci. USA* **2001**, *98*, 404–409. [[CrossRef](#)]
33. Viswanath, B.; Mislove, A.; Cha, M.; Gummadi, K.P. On the evolution of user interaction in Facebook. In Proceedings of the 2nd ACM SIGCOMM Workshop on Social Networks (WOSN’09), Barcelona, Spain, 17 August 2009.
34. The Max Plank Institute for Software Systems Web Site. Available online: <http://socialnetworks.mpi-sws.org/data-wosn2009.html> (accessed on 28 June 2022).
35. Taylor, A.; Higham, D.J. CONTEST: A controllable test matrix toolbox for MATLAB. *ACM Trans. Math. Softw.* **2009**, *35*, 26:1–26:17. [[CrossRef](#)]