Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Failure detection for transport processes on networks

Supervisor:
Prof. Armando Bazzani

Submitted by:
Edoardo Rolando

Academic Year 2020/2021

Abstract

Diffusion on networks is a convenient framework to describe transport systems of different nature (from biological transport systems to urban mobility). The mathematical models are based on master equations that describe the diffusion processes by means of the weighted Laplacian matrix that connects the nodes. The link weight represent the coupling strength between the nodes. In this thesis we cope with the problem of localizing a single-edge failure that occurs in the network. An edge failure is meant to be as a sudden decrease of its transport capacities. An incomplete observation of the dynamical state of the network is available. An optimal clustering procedure based on the correlation properties among the node states is proposed. The network dimensionality is then reduced introducing representative nodes for each cluster, whose dynamical state is observed. We check the efficiency of the failure localization for our clustering method in comparison with more traditional techniques, using different graph configurations.

Contents

Introduction

A graph (also called network) is a collection of entities (nodes) whose interaction structure is described by the edges (or links) connecting them. Therefore a network is a simplified and abstract representation of a system topology. This simple yet powerful definition allow the description of a rich variety of structures. In fact, nowadays network science has became ubiquitous, finding applications in a broaden range of fields ranging from physical, biological, social and transport systems. For instance, networks can be suited for the study of the interaction among biological entities, like genes or species. In social sciences, the nodes might represent agents that interact and influence each other [26]. Concerning transport systems, network theory has found fruitful applications for traffic models and water distribution systems (WDS) [12]. Therefore, the study of networks is highly interdisciplinary, borrowing tools and ideas stemming from various area of knowledge, like mathematics, physics and engineering. The success of network science has also prompted results in mathematical fields, like Spectral Graph Theory [10], that studies the characteristics of a graph in relation to the spectral properties of matrices associated to it, like the Adjacency matrix and the Laplacian matrix. The former encompass the topological relations of the network, whereas the latter can be seen as a difference operator on the network nodes. The Laplacian matrix has found countless applications in the study of networks. For instance, the spectral decomposition of the Laplacian matrix is used to define a graph analogue of the Fourier transform. The overlap between traditional signal processing and graph theory has lead to a new emerging field called Graph Signal Processing (GSP), that has gained a lot of attention recently [49].

Networks are a suitable mathematical framework also for the study of dynamical processes occurring on discrete domains, such as traffic flow, the spreading of a disease or an opinion in a social network, or the diffusion of a physical quantity, like heat, electric current or water [4]. According to the particular system under study, one might focus solely on the topology, on the dynamics taking place on the network, or both. Nevertheless, as intuitive, there is an interplay between the network topology and the eventual dynamics occurring on the graph. The Laplacian matrix establishes a relation between the continuous world processes and the discrete analogue on a network. Indeed, it naturally describes diffusion process and consensus dynamics occurring on networks.

This thesis work focuses on transport networks that can be modeled by means of a diffusion equation, with the adding on an external input driving the system. A typical problem of a transport system is the study of its resilience and stability under the failure of one or more of its components. Even a single-edge failure can lead to a cascade effect, putting at risk the transport capabilities of the whole structure. This kind of problem is particularly important for power grid networks, for which a maximum voltage capacity is defined for each transmission line. In the literature, several studies focus on the creation and planning of optimal transport networks, such that the resilience and stability under one or multiples failures of the network is maximized.

However, in other situations the topology of the network is considered to be given and fixed, because pre-existing or independent from human will. Therefore, one is concerned with a different but related problem, that is the installation of an efficient sensor system, to monitor and eventually control the state of the network. In fact, monitoring all the nodes of the transport system may be unfeasible and expensive. Consequently, the aim becomes to find a sensor placement criterion, such that the observability and localization of the failure is optimal. Several algorithms has been devised with this scope, ranging from engineering tools to ideas stemming from network science. The traditional clustering algorithms used to partition a graph might be exploited in order to group nodes according to some characteristics [16]. Among these methods, a prominent importance is given to the spectral clustering technique, which is based on the Laplacian matrix spectral properties. In fact, the eigenvectors of the Laplacian matrix, provide a natural way to embed the network nodes in an euclidean space. This procedure helps revealing the structural features of the network. The idea of spectral clustering can be justified with the physical analogy of a network with a spring-mass system, in which the nodes weights are the elastic constants of the springs [35].

The set of equations governing the whole system and its reduced (or *coarse-grained*) version find an analogue counterpart in linear time-invariant system, for which the concept of observability and controllability has long been studied. In particular, a system is observable if the observability matrix has full rank [9]. Concerning the problem we are dealing with, observing the state of the network only through a subset of nodes (sensors), the system does not satisfy the observability condition. However, once a partition of the network is established, the idea is to set a single sensor node for each cluster, such that the failure localization is possible with good accuracy.

Outline of the thesis

The first chapter of this thesis work is a brief introduction to network science, giving a special focus on the Laplacian matrix and its properties. In particular, it is shown how the evolution of diffusion processes taking place on a network is governed by the Laplacian matrix. The equation describing such processes turn out to be the discrete analogue of the heat equation for continuous bodies. The Laplacian matrix is also used to describe a spring-mass system, in which the nodes are the masses and the edge weights correspond to the stiffness of the springs. Generally, it is assumed an heterogeneous distribution of the spring constants. This analogy will be particularly fruitful for the subsequent chapters.

In Chapter 2, the main graph community detection and clustering techniques are described. Traditionally, they are based solely the network topology. Particular attention is given to the spectral clustering technique, since it exploits the Laplacian matrix to partition the network in such a way that densely connected nodes belongs to the same cluster. The analogy with a mass-spring system allow a physical interpretation of the spectral clustering results. A successful definition of community is a group of nodes whose internal connections are denser than the expected structure of a random graph. This definition lead to the concept of modularity function, which has the unique privilege of being at the same time a clustering technique and a benchmark function to test the performance of other algorithms.

Chapter 3 contains the original part of this thesis work. In the first place, it is explained the importance of clustering techniques based not solely on the structure of the network but also on the dynamical processes occurring on top of it. The idea is to find a coarse-grained description of the dynamics, such that nodes belonging to the same cluster behave in a similar fashion. In particular, the spectral clustering admits also an interpretation based on the timescales of a diffusion process. Secondly, we propose an original network reduction scheme based on the correlation properties of the nodes under a perturbation of their states. We considered two kind of perturbations. First, we add a white noise to the external forcing of the system. The correlation properties found depend on the spectrum of the Laplacian matrix governing th dynamics, but not on the forcing itself. Secondly, we study a perturbation of the network weights, leading to a direct perturbation of the Laplacian matrix. The found correlation properties depend on the spectrum of the Laplacian matrix and on the fluxes of the transport system. In particular, more importance is given to those eigenvectors that are able to distinguish edges with higher flux.

In Chapter 4 we have exploited to correlation matrix obtained in order to generate a clustering procedure for the network nodes. Afterwards, we compare the efficiency under a single-edge failure event of the proposed clustering algorithm with the traditional structure-based ones described in Chapter 2. The used failure identification function is based on a Bayesian approach. The network architectures used for the comparison are the Erods-Renyii random model, lattice network and Barabasi-Albert graph. It is shown how the proposed clustering algorithm is more sensitive to failure of edges with higher fluxes, that are the most important ones in a transport system.

Chapter 1

Network theory

In this Chapter, we provide the basic definitions and mathematical tools necessary for the study of network systems. Particular emphasis is put on the representation of a graph given by the so called Laplacian matrix, whose properties and applications are fundamental for the development of this thesis work.

1.1 Basic definitions

A *network* or *graph*, as usually denoted in the mathematical literature (the two terms will be used interchangeably) is an abstraction of a system defined by entities and the relations (or couplings, depending on the context) between them. More formally,

Definition 1 (Network) *A network is defined by a pair of sets $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, in which the elements of $\mathcal{V}(\mathcal{G}) = \{v_1, \dots, v_n\}$ are called nodes or vertices and the set $\mathcal{E}(\mathcal{G}) = \{e_{ij}\}$ consists in all the edges or links between them. The size of the vertex set is usually denoted by the letter $N = |\mathcal{V}|$, whereas the size of the edges set by $M = |\mathcal{E}|$.*

Moreover, two nodes i and j are said to be neighbors (or adjacent) if there is and edge (ij) connecting them.

Its is possible to defined several networks categories according to the kind of edges allowed in its structure. For instance, *multiedges* networks features nodes with more than one edge between two nodes. Otherwise, one might want to consider graphs with edges connecting a single node to itself: these kind of connections are called *self-edges*. Several network models feature directed connections. These kind of graphs are called *directed networks*. A *simple network* is defined as a graph with neither self-edges nor multiedges [40]. Throughout the rest of this thesis work we will consider only undirected simple networks, for which $w_{ij} = w_{ji}$.

To describe faithfully some networks, it is useful to define edges as having a strength or weight. These kind of networks are called *weighted graphs*, and they are characterized by a real-valued function [1] associated to their connections

$$w : \mathcal{V}(\mathcal{G}) \times \mathcal{V}(\mathcal{G}) \to \mathbb{R}^+$$

Edges weights might have several interpretations, according to the kind of system the network represents. For instance, they might measure the strength of connection or similarity between the nodes. In transport systems, the edge weight might represent its carrying capacity. In power grid networks, the edge weights are usually the conductance of the wires. Finally, in spatial graphs (for instance road or airplane networks), the edge weights might be regarded as a distance between the nodes [2].

The fundamental topological information of a network is encoded by the so called *Adjacency matrix*, defined as a $N \times N$ matrix whose entries in the case of a weighted undirected graph are defined as

$$A_{ij} := \begin{cases} w_{ij} & \text{if } (i,j) \in \mathcal{E}(\mathcal{G}) \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

It is worth noting that a network with no self-edges has diagonal matrix elements equal to zero. Secondly, if the network is undirected, the adjacency matrix is symmetric, hence $A_{ij} = A_{ji}$.

The first quantity that is used to describe a node is its *degree*. For an undirected graph, the degree of a node is defined as the number of edges connected to it, and can be computed through the unweighted adjacency matrix

$$k_i = \sum_{j=1}^{N} A_{ij}. \tag{1.2}$$

Since every edge is connected to two nodes, we have the following relation between the total number of edges M and the degree distribution of the nodes

$$2M = \sum_{i=1}^{N} k_i = \sum_{ij} A_{ij}. \tag{1.3}$$

The mean degree of a node in an undirected network is given by

$$\langle k \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i \tag{1.4}$$

[1]Recently, networks with both positive and negative weights, called *signed graphs*, have gained attention, especially in social sciences [34].

[2]Considering the edge weight as a distance can be misleading. In fact, since the weights usually describe the strength of interactions between the nodes, one should in principle consider the edge weight to be the reciprocal of a distance. This is analogue to using the conductance instead of the resistance for power grid networks.

and combining the two equations above we get to

$$\langle k \rangle = \frac{2M}{N} \tag{1.5}$$

that relates the average degree of the network with its total number of nodes and edges.

For weighted networks, it is natural to define another useful quantity called *node strength*, that measures the sum of all weights connected to a given node. therefore, the node strength is still given by the equation 1.2, using the weighted adjacency matrix.

The following definitions will be useful throughout the text.

Definition 2 (Walk) *A walk in a network is defined as a sequence of nodes such that every consecutive pair of nodes in the sequence is connected by and edge.*

A *shortest path* (or *geodesic path*) is the shortest walk between a pair of nodes.

Definition 3 (Component) *A component of a network is a subset of nodes such that there exists at least one path connecting each member of that subset to any other member.*

A network in which all nodes belongs to the same component is said to be *connected*. It is straightforward that the adjacency matrix of a disconnected network can be written in block diagonal form.

To capture the relative importance of nodes and edges in a network, several measures have been devised in the scientific literature [40]. A lot of them are centrality based, meaning that the importance of a node or edge is related to its centrality measure. There are many possible definitions. For instance, the simplest one is the *degree centrality* that is just the degree of a node.

Definition 4 (Betweenness centrality) *Betweenness centrality measures the extent to which a node lies on paths between other nodes and it is given by*

$$x_i = \sum_{st} \frac{n^i_{st}}{g_{st}} \tag{1.6}$$

where n^i_{st} is the number of shortest paths from node s to node t that pass through node i, and the normalization factor g_{st} is the total number of shortest paths between s and t.

Nodes with high betweenness centrality have a influence within the network, since they have a role of control of the information passing through them. Several generalization of definition 4 are possible [17].

Finally, we mention that it might be useful to define real-valued functions on the nodes, that is

$$\mathbf{f} : \nu \to \Re, \quad \mathbf{f} \in \Re^N$$

assigning a real number to each node of the network. This real-valued functions might encode information about a spatial-temporal processes taking place on the network, in which case we might define a time series on each node. Alternatively, it may represent nodes physical positions, or eigenfunctions of suitable operators defined on the network. We will encounter all these situations in the following sections.

1.2 Laplacian matrix

Another useful representation of a network is given by the so called *graph Laplacian* (also called *discrete Laplacian*, or *Kirchoff matrix*). For an undirected and weighted network, the Laplacian matrix is a $N \times N$ matrix defined as follows

$$L_{ij} = \begin{cases} -w_{ij} & \text{if } i \neq j \text{ and } i \text{ is adjacent to } j, \\ d_i & \text{if } i = j, \\ 0 & \text{otherwise} \end{cases} \tag{1.7}$$

where w_{ij} is the weight from node i to node j and $d_i = \sum_j w_{ij}$ is the degree of node i. More compactly it can be written as

$$L_{ij} = \delta_{ij} d_j - w_{ij} \tag{1.8}$$

or in matrix-vector notation $L = D - A$, where D is the diagonal degree matrix, whose i^{th} entry is the degree of node i. For undirected networks, the Laplacian matrix is symmetric $L_{ij} = L_{ji}$. The Laplacian matrix has a lot of application in graph theory, due to its particular properties. First of all, it can be seen as a difference operator on the graph. In fact, applying L to a generic real-valued function f defined on the nodes we get

$$\begin{aligned} (Lf)(i) &= \sum_j L_{ij} f_j = \\ &= d_i f_i - \sum_j w_{ij} f_j \\ &= \sum_j w_{ij}(f_i - f_j) \end{aligned} \tag{1.9}$$

where the sum is implicitly performed over the neighbors of node i, since for all other nodes $w_{ij} = 0$. Being a symmetric operator, there is a natural quadratic form associated with it

$$\begin{aligned}\vec{f}^T L \vec{f} &= \sum_{ij} L_{ij} f_i f_j \\ &= \frac{1}{2} \sum_i d_i f_i^2 - 2 \sum_{ij} w_{ij} f_i f_j + \sum_j d_j f_j^2 = \\ &= \frac{1}{2} \sum_{ij} w_{ij} \left(f_i^2 - 2 f_i f_j + f_j^2 \right) \\ &= \frac{1}{2} \sum_{ij} w_{ij} \left(f_i - f_j \right)^2 \end{aligned} \tag{1.10}$$

where the factor 1/2 is due to the double counting of the nodes in the sum. In Graph Signal Processing [49], the Laplacian quadratic form is related to the smoothness of the function (signal) on the manifold (network). In fact, the quadratic form is null if and only if the signal is constant over all vertices. In general, $\vec{f}^T L \vec{f}$ takes low values when the signal has similar components on nodes connected by edges with high weights, i.e the smoother the signal on the graph, the lower is the value of 1.10. Being related to the discrete analogue of the spatial derivative, the Laplacian matrix allows a natural connection between discrete representations, such as networks, and continuous representations, such as vector spaces and manifolds [38] [10].

Since the Laplacian matrix is symmetric, it has a complete set of orthonormal eigenvectors, which we denote by $\{\vec{v}_k\}_{k=0,1,...,N-1}$

$$L\vec{v}_k = \lambda_k \vec{v}_k \quad k = 0, 1, ..., N-1. \tag{1.11}$$

with the orthogonality condition

$$\vec{v}_k \vec{v}_h = \delta_{kh}. \tag{1.12}$$

It is easy to note that every row and column sum of the Laplacian matrix vanishes

$$\sum_j L_{ij} = \sum_j (k_i \delta_{ij} - A_{ij}) = k_i - k_i = 0$$

and therefore L is a singular matrix, i.e there is at least one eigenvalue equal to 0 corresponding to the uniform eigenvector $\vec{v}_0 = \frac{1}{\sqrt{N}}(1, ..., 1)$. Being a singular matrix, L has no inverse [3].

From the condition 1.12 and the knowledge of the kernel eigenvector $\vec{v}_0 = \frac{1}{\sqrt{N}}(1, ..., 1)$, we get that for any eigenvector with $\lambda \neq 0$

$$\sum_{i=1}^{N} v_i^{\lambda} = 0 \tag{1.13}$$

[3]However, it is possible to defined a pseudoinverse of the Laplacian matrix, for example the Moore-Penrose one [6]. The Laplacian pseudoinverse finds several application in network theory [22].

and, since we are taking them to be normalized, each component is bounded by $-1 < v_i^\lambda < 1$.

Interpreting the Laplacian eigenvectors as eigenfunctions on the vertices, we can use them as in 1.10, getting to

$$\vec{v}_\lambda^T L \vec{v}_\lambda = \frac{1}{2} \sum_{ij} w_{ij} \left(v_i^\lambda - v_j^\lambda \right)^2 .$$

Making use of equation 1.11 and the orthogonality condition 1.12 we also get

$$\vec{v}_\lambda^T L \vec{v}_\lambda = \lambda \vec{v}_\lambda^T \vec{v}_\lambda = \lambda$$

that combined with the equation above give

$$\lambda = \frac{1}{2} \sum_{ij} w_{ij} \left(v_i^\lambda - v_j^\lambda \right)^2 \geq 0 \tag{1.14}$$

and therefore, since the weights are defined to be greater than zero, all the Laplacian eigenvalues are non-negative (thus L is a positive semi-definite matrix).

The multiplicity of the null eigenvalue of the Laplacian matrix provides important information about the network connectivity. In fact, the number of connected components in the graph equals the dimension of the nullspace of the Laplacian. In this case L can be written in a block diagonal form, and each individual block is the Laplacian matrix of the corresponding connected component. Hence, if there are k connected components, each of dimension N_k, the k eigenvectors corresponding to $\lambda = 0$ will be of the form

$$\vec{v}_i = \frac{1}{\sqrt{N_i}} (\underbrace{1, \ldots, 1}_{N_i}, \underbrace{0, \ldots, 0}_{N-N_i}) \quad \text{for} \quad i = 0, \ldots, k-1$$

with non-vanishing components only on the nodes belonging to a component. Therefore, those eigenvectors are indicators of the connected components.

Since for a connected network there is only one zero eigenvalue, we can order the spectrum in increasing order as follows:

$$0 = \lambda_0 < \lambda_1 \leq \lambda_2 \ldots \leq \lambda_{N-1} := \lambda_{\max} \tag{1.15}$$

From now on we will refer only to connected networks.

The second smallest eigenvalue of L deserves a special mention: λ_1 is called *Fiedler value* of the network and, as seen above, if it equals to zero the graph is not connected. λ_1 is also called *algebraic connectivity*, since the further it is from zero, the more the network is connected. Its associated eigenvector $\vec{v}_1$ is called Fiedler vector, and it find applications in graph spectral partitioning. The Fiedler value and vector are associated with several network topological information [10].

1.3 Graph Fourier Transform

The network Laplacian matrix allows a generalization of the classical (continuous) Fourier decomposition of a signal on the discrete domain of a graph. In fact, in the classical case, the Fourier transform consists of the expansion of a signal in terms of complex exponentials, which are the eigenfunctions of the one-dimensional Laplace operator

$$\hat{f}(\xi) := \left\langle f, e^{2\pi i \xi t} \right\rangle = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt$$

$$-\Delta \left(e^{2\pi i \xi t} \right) = -\frac{\partial^2}{\partial t^2} e^{2\pi i \xi t} = (2\pi\xi)^2 e^{2\pi i \xi t}$$

Analogously to the continuous case, writing a signal in the graph spectral domain means projecting it onto the Laplacian eigenfunctions, hence its name. Thus the *graph Fourier transform* of a signal f_i on the vertices of a network is defined as [49]

$$\hat{f}\left(\lambda_l\right) := \left\langle \vec{f}, \vec{u}_l \right\rangle = \sum_{i=1}^{N} f(i) u_l^*(i). \tag{1.16}$$

Since L is symmetric, its eigenvectors are orthogonal and therefore we can define the *inverse graph Fourier transform* as

$$f(i) = \sum_{l=0}^{N-1} \hat{f}\left(\lambda_l\right) u_l(i) \tag{1.17}$$

Given the correspondence of the two transformation, we can interpret the Laplacian eigenvalues as the frequencies of the continuous case

$$\lambda_l \leftrightarrow \left\{ (2\pi\xi)^2 \right\}_{\xi \in \mathbb{R}}$$

Thus Laplacian eigenvectors corresponding to high eigenvalues oscillates more rapidly on the graph domain, that is they are likely to take different signs on vertices that are connected by high weights. The higher the eigenvalue the more its correspondent eigenvector is discontinuous on the graph domain. However, we should pay more attention in making this analogy. As the graph grows in complexity, we cannot see anymore the eigenvalues as a simple monotonic function like the frequency. In particular, it becomes difficult to order and organize the eigenmodes like for the continuous case [46].

Nevertheless, the analogy between the spectral properties of L and the normal eigenmodes in the Fourier sense can be better formalized through the *Courant-Fisher Theorem*, that provides a characterization of the eigenvalues and eigenvectors of a symmetric matrix through a constrained optimization problem [50]. The Laplacian eigenvalues and eigenvector are therefore defined through

$$
\begin{aligned}
\lambda_0 &= \min_{\substack{\vec{f}\in\Re^N,\\ \|\vec{f}\|_2=1}} \vec{f}^T L \vec{f} \\
\lambda_l &= \min_{\substack{\vec{f}\in\Re^N,\\ \|\vec{f}\|_2=1\\ \vec{f}\perp\{\vec{v}_0,\ldots,\vec{v}_{l-1}\}}} \vec{f}^T L \vec{f}
\end{aligned}
\tag{1.18}
$$

that is an optimization problem having as objective function the Laplacian quadratic form. Since, as already mentioned, $\vec{f}^T L \vec{f}$ is related to the smoothness of a function on the vertices, we deduce that the eigenvectors with lower eigenvalue show less variation. In particular, from 1.18 we again see that the lowest eigenvalue is $\lambda_0 = 0$ and corresponds to the constant eigenvector $\vec{v}_0 = \frac{1}{\sqrt{N}}(1, ..., 1)$.

1.4 Laplacian matrix applications

As seen in the previous section, the Laplacian matrix allows the generalization of ideas and tools typical of the continuous domain to the discrete one. For this reason, it is not surprising that it pops up in a great variety of applications. For instance, it can be used to model resistor networks, random walkers, synchronization systems and lastly diffusion processes, that will be the main focus of this thesis work.

1.5 Diffusion model

Suppose some quantity is circulating in a network system. We denote with $x_i(t)$ the quantity present on node i at time t. The time evolution of the system is given by an equation of the form

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = f_i\left(x_i\right) + \sum_j w_{ij} g_{ij}\left(x_i, x_j\right)$$

where w_{ij} are the edge weights, representing some parameter of interest for the system under consideration, $f_i(x_i)$ is a function that specifies the intrinsic dynamics of each node, and $g_{ij}\left(x_i, x_j\right)$ is a generic coupling function of the system state variables. If it is assumed that the function f is the same for each node and g is the same for each couple of nodes, the above equation simplifies to

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = f\left(x_i\right) + \sum_j w_{ij} g\left(x_i, x_j\right). \tag{1.19}$$

This kind of equations are ubiquitous in network theory. Indeed, they have been used to describe the spreading of a huge diversity of physical or abstract quantities, like information, viruses, heat, fluid, over the network nodes.

The simplest diffusion process one can think of is obtained setting $f(x_i) = 0$ and taking the coupling $g(x_i, x_j)$ to be equal to $x_i - x_j$, thus obtaining

$$\frac{\mathrm{d}x_i}{\mathrm{d}t} = \sum_j w_{ij}(x_i - x_j) \tag{1.20}$$

that gives the time change of the concentration on each node due to the diffusive transportation process taking place. The choice $g(x_i, x_j) = x_i - x_j$ implies that the substance flow from node j to node i is proportional to the concentration difference $(x_i - x_j)$, that is commonly known under the name of *Fick's law*. Equation 1.20 can be easily recast as follows

$$\begin{aligned}
\frac{dx_i}{dt} &= -\sum_j w_{ij}(x_i - x_j) \\
&= -x_i \sum_j w_{ij} + \sum_j w_{ij} x_j \\
&= -x_i d_i + \sum_j w_{ij} x_j \\
&= -\sum_j (\delta_{ij} d_i - w_{ij}) x_j \\
&= -\sum_j L_{ij} x_j
\end{aligned} \tag{1.21}$$

where again we see that the Laplacian matrix plays the role of a difference operator, like explained for the equation 1.9. We recognize the analogy of 1.21 with the heat equation for a continuous medium

$$\frac{\partial u(\vec{x})}{\partial t} = \nabla^2 \alpha(\vec{x}) u(\vec{x})$$

where

$$\nabla^2 = \nabla \cdot \nabla = \sum_{i=1}^{n} \frac{\partial^2}{\partial x_i^2}$$

is the Laplacian operator in Cartesian coordinates, u describe a temperature distribution and α is called thermal diffusivity. This is again a consequence of the Laplacian matrix being a natural link between discrete representations, such as graphs, and continuous domains.

The particular diffusion model described by the equation 1.21 is usually called *fluid model* [2]. The reason for this name is that this kind of equation might describe the *linear* flowing of some fluid in a pipe network: in this case the state variable x_i is the pressure on node i (that is the junction of two of more pipes) and w_{ij} encode information about the ease fluid flows through the pipes.

Equation 1.20 is also called in the literature *edge-centric continuous time random walk* [36]. In fact, it can be used to describe a random walker moving on the network. The probability to find the random walker on node i at time t is given by $p_i(t)$ and the transition rates are given by the weights w_{ij}. Thus the higher the (generalized) degree of a node, the lower the probability that the random walker is found on that specific node. The reason is that the transition rates do not sum to one $\sum_j w_{ij} \neq 1$. Consequently, the random walker leaves nodes with high degree quicker than nodes with small degree.

Expanding the Laplacian matrix in its eigenbasis

$$L = \sum_{\lambda \neq 0}^{N} \lambda \vec{v}_\lambda \vec{v}_\lambda^T$$

and recalling that all λ are greater than 0 (excluding the kernel) the full solution of 1.21 is given by

$$\vec{x}(t) = \sum_{\lambda \neq 0} c_\lambda e^{-\lambda t} \vec{v}_\lambda \tag{1.22}$$

with initial conditions

$$\vec{x}(0) = \sum_{\lambda \neq 0} c_\lambda \vec{x}_\lambda. \tag{1.23}$$

When the system reach equilibrium, we have the stationary condition

$$L\vec{x} = 0$$

that is equivalent to the detailed balance condition, that reads

$$\sum_i w_{ji} x_i^s = \sum_j w_{ij} x_j^s \tag{1.24}$$

that can be rewritten as

$$x_i^s = \frac{\sum_j w_{ij} x_j^s}{d_i} \tag{1.25}$$

where d_i is the degree of node i. The last condition is called *self-averaging property* and the function $\vec{x}$ satisfying it is called *harmonic function*[27]. As seen, if the network is connected the Laplacian matrix has only one eigenvalue equal to zero and corresponding to the uniform eigenvector $\vec{v_0} = \frac{1}{\sqrt{N}}(1, \ldots, 1)$. Therefore, at equilibrium the states of the system is uniform. If the process taking place is a heat diffusion, it means a uniform temperature over the graph.

1.6 Graphs as spring networks

In order to understand the properties of the Laplacian matrix, the analogy of a graph with a mass-spring system is particualrly useful. In this case the nodes are the masses and the edges are the ideal rubber bands between them. The adjacency matrix gives the relation between the masses, i.e which ones are connected and the edges weights w_{ij} indicate the springs constants k_{ij}. The real-valued function x_i defined on the nodes represents the one dimensional displacement of the masses. These displacements are meant to be in the direction orthogonal to the embedding space of the network (see the following section 1.7). Therefore, if the network is planar [4], the displacement is in the axis orthogonal to the plane. From *Hooke's law*

$$F_{ij} = -w_{ij}(x_i - x_j)$$

denotes the force that node j exerts on node i. Analogously to 1.21, we can write the total force on node i as

$$F_i = \sum_j F_{ij} = -\sum_j w_{ij}(x_i - x_j) = -\sum_j L_{ij}x_j \tag{1.26}$$

and the Lagrangian of the system reads

$$\begin{aligned} \mathcal{L} &= \frac{1}{2}\sum_i m_i \dot{x}_i^2 - \frac{1}{4}\sum_{ij} w_{ij}\left(x_i - x_j\right)^2 \\ &= \frac{1}{2}\sum_i m_i \dot{x}_i^2 - \frac{1}{2}\sum_{ij} x_i L_{ij} x_j. \end{aligned} \tag{1.27}$$

where the factor 1/2 for the potential energy takes care of the double counting of nodes pairs. Therefore the equations of motion are

$$m_k \ddot{x}_k = -\sum_j L_{kj}x_j \tag{1.28}$$

or in vector notation

$$\ddot{\vec{x}} = -K\vec{x} \tag{1.29}$$

where $K = M^{-1}L$ is called *stiffness matrix* and M is the diagonal matrix of the nodes masses. The solution is well known and it is given by the eigenvectors of the matrix K, that give the displacement of the masses for each harmonic mode. The corresponding

[4] A planar network is a graph that can be embedded in the plane, i.e. it can be drawn on a flat surface in such a way that its edges do not cross each other. For instance, on first approximation road networks can be considered to be planar.

eigenvalues are the energies required to sustain the oscillations and their square roots are thus the frequencies of the fundamental oscillations.

Taking all masses to be equal to one $m_i = 1 \quad \forall i$, the stiffness matrix reduces to the unnormalized Laplacian matrix. We might also make a different choice, taking the masses to be equal to the degree of the node $m_i = \sum_j w_{ij}$. In this case the stiffness matrix reduces to the normalized Laplacian of the graph. In both cases, nodes connected by strong springs will oscillate together for low frequencies (low eigenvalues of the Laplacian matrix). This physical analogy reinforce the use of Laplacian eigenmodes as normal modes of oscillations, like shown in section 1.3.

If no energy is present in the system, the solution of

$$\ddot{\vec{x}} = -L\vec{x} \tag{1.30}$$

is the kernel of L, since in this case $L\vec{x} = 0$. This means that all masses collapses to a single point, as intuitive. An interesting situation one can think of is nailing some vertices of the system, say the first k of them, and letting the rest settle. To find the equilibrium position, we minimize the potential energy with boundary conditions (nodes at fixed positions)

$$\frac{\partial}{\partial x_v}\left(\sum_{ij} x_i L_{ij} x_j + 2\sum_{i=1}^{k} b_i x_i\right) = 0$$

where b_i with $i = 1, \ldots, k$ are Lagrange multipliers that enforce the k coinstaints. Thus we get

$$2\sum_j L_{vj} x_j + 2\sum_{i=1}^{k} b_i \delta_{v,i} = 0 \quad \text{for} \quad v = 1, \ldots, N$$

and summing over all nodes

$$2\sum_j x_j \sum_v L_{vj} + 2\sum_{i=1}^{k}\sum_v b_i \delta_{v,i} = 0 \quad \text{for} \quad v = 1, \ldots, N$$

since $\sum_v L_{vj}$=0, we get to the following contraint

$$\sum_{i=1}^{k} b_i = 0-$$

Therefore the equilibrium positions are obtained by solving the linear system

$$L\vec{x} = \vec{b} \quad \text{with} \quad \sum_i b_i = 0. \tag{1.31}$$

Notice that for all free vertices we have the *self-averaging property*

$$x_v = \frac{1}{d_v} \sum_i w_{vi} x_i \tag{1.32}$$

already encountered in the previous section.

The physical analogy between a network and a spring-mass system happens to be useful also for the understanding of the spectral clustering technique, to be discussed in chapter 2. The idea is that cluster of nodes with stiff connections among them oscillates togheter as a chunk.

1.7 Spectral embedding

Structural properties of a network might be revealed from an embedding of its nodes in a continuous metric space, called *embedded space* or *latent-space.* For instance, we might choose to map the nodes on a one-dimensional line, in such a way that nodes strongly connected are close to each other. Embedding a network is also useful for visualization purposes, if for example the embedding space is chosen to be the two-dimensional euclidean space. Generally the dimensionality of the embedded space is chosen lower than the one of the initial feature space (i.e. the numbers of nodes in the network), such that a dimensionality reduction can be performed.

This problem can be formalized writing an objective function of the nodes position and minimize it. An obvious choice is the sum of the squared distances of the nodes. Considering the one dimensional case (i.e. the mapping of a weighted network onto a line) the objective function reads

$$\vec{x}^T L \vec{x} = \frac{1}{2} \sum_{ij} w_{ij} \left(x_i - x_j\right)^2$$

that is the quadratic form associated with the graph Laplacian, with the nodes positions as real-valued function. As already noted in the case of a spring-mass system, the minimization of this function is achieved when all the nodes lie in the same position, that is not very insightful. This problem might be avoided imposing constraints on the nodes positions, such that their overall distribution remains spread out. So we fix the overall centre of mass with

$$\sum_i x_i = 0$$

and to avoid them ending up in the same position (given the constraint above, the origin), we fix also the variances of their positions

$$\sum_i x_i^2 = 1$$

This constraint optimization can be performed by use of the Lagrange multipliers method, analogously to the previous section. Alternatively, we can note that is the same optimization problem 1.18. Therefore, the constraints above are nothing but conditions in order not to choose the Kernel of L and normalize the eigenvectors.

In either way, the conclusion is that the positions of the nodes in the optimal embedding are given by the eigenvectors of the Laplacian matrix. Moreover

$$\vec{x}^T L \vec{x} = \vec{v}_\lambda^T L \vec{v}_\lambda = \lambda \vec{v}_\lambda^T \vec{v}_\lambda = \lambda \tag{1.33}$$

since the eigenvectors are normalized to one. Therefore the overall squared displacement is minimized by the eigenvectors with the lowest eigenvalue. The lowest eigenvalues is $\lambda_0 = 0$, but the corresponding eigenvector $\vec{v_0} = \frac{1}{\sqrt{N}}(1, \dots, 1)$ fails to satisfy the constraints above. Hence, the best one dimensional embedding is given by the components of the second smallest eigenvector, the Fiedler vector.

Notice that spectral embedding using small eigenvalues is equivalent to a self-averaging property for the positions, since

$$\vec{x}_i = \frac{1}{d_i - \lambda} \sum_j w_{ij} \vec{x}_j.$$

More generally, the embedding of a graph in a k-dimensional euclidean space, is given by a $N \times k$ matrix $X = [\vec{x_1}, \vec{x_2}, ..., \vec{x_k}]$ whose i-th row are the coordinates of the i-th node. Therefore, the quadratic norm to be minimized is

$$\vec{x}^T L \vec{x} = \frac{1}{2} \sum_{ij} w_{ij} \|\vec{x_i} - \vec{x_j}\|^2 \tag{1.34}$$

Thus, for each node, the k coordinates needed to perform the embedding are given by the components on that node of the eigenvectors corresponding to the k smallest non-zero eigenvalues, i.e $\lambda_1, \dots, \lambda_k$, that we can organize in the following embedding matrix

$$U = \begin{bmatrix} \vec{v}_{\lambda_2,1} & \dots & \vec{v}_{\lambda_{k+1},1} \\ \vdots & & \vdots \\ \vec{v}_{\lambda_2,N} & \dots & \vec{v}_{\lambda_{k+1},N} \end{bmatrix}$$

whose i-th row provide the embedding for the node number i.

Chapter 2

Network partitioning

Networks display a huge variety of possible configurations, and they range from totally regular graphs, like lattices, to ones with random connections, like the Erdős–Rényi and Barabasi-Albert models described in the previous chapter.

Real world networks are generally not random, since they show a certain level of organization and structure [21]. The distributions of edges is both globally inhomogeneous, meaning that the degree distributions are broad, and locally inhomogeneous, with areas displaying high density of internal edges and low concentrations between them. These regions of densely internally connected nodes suggests that the network has certain natural divisions within it and they are called *clusters*[1] or *communities*, and the problem of identifying them has became one of the most fundamental problem in network science since the seminal work [21]. For instance, in protein interaction networks, communities might identify proteins with similar roles within the cell, and in citation networks papers form communities according to the research topic [16].

The concept of community is not rigorously defined, but it depends on the algorithm that is used to find them [45]. Thus several possible criteria might be chosen that produce different outcomes. One usually defines communities as being set of nodes that share more edges within their own community than with the rest of the network. This definition is used both for non-weighted networks and weighted ones, for which the density or sparsity of connections takes care also of the edges weights (more on this in section 2.1). If the graph is not sparse and, for weighted networks, the edges weights are not heterogeneously distributed, the density based definition of clusters looses sense. In this situation, traditional data clustering techniques might come in help [25]. This data-driven approach to communities identification is particularly useful for clustering based on eventual dynamics running on a network, that is the topic of chapter 3. Thus they will be mentioned in more details in chapter 4).

In most real-world networks there is no ground truth regarding the eventual com-

[1]Not to be confused with the clustering coefficient [55]

munity structure. Thus it is difficult to asses the efficiency of a particular clustering algorithm. To address this problem, different benchmark were developed. One of this is based on the idea *modularity*, a function introduced by [41] that has the particular advantage of being at the same time a clustering method and a benchmark function. Modularity will be discussed in sections 2.3.

Communities detection is essential not only to the understanding of systems' structural features, but also how a networked system is generated form its basic constituents and how the community structure influence the dynamics acting on top of the network [47]. Furthermore, nodes belonging to the same community might have different roles within the cluster, and they can be classified accordingly. There might be nodes with important roles of control and stability within the group [16].

A kind of different but related problem to the one of communities detection is *network partitioning.* A network partition is a division of the whole graph into a predefined number of clusters, such that every vertex belongs to one of them and the edges between these groups is minimal. The main difference from communities detection algorithms is that the number of clusters, and sometimes also their sizes, is predefined, i.e. it must be know *a priori* for the partitioning algorithm to work. Generally, in a community detection problem one does not have this information, since it is exactly what we want the algorithm to produce as its output. Consequently, it is not guaranteed that partitioning algorithms produce good clusters in the community sense specified above.

In the following section we introduce the problem of graph partitioning, showing how it is related to the spectral properties of L, leading to the so called *spectral clustering* [35]. This method is particularly important for this thesis work, since the partitioning method that we will propose in chapter 3 is related to the spectral embedding of section 1.7, and thus ultimately to the spectral clustering.

2.1 Graph cuts

To mathematically formalize the network partitioning problem, it is necessary to introduce the following definition. Given two nodes subsets A and B in a network, the sum of the edges weights between them is called *cut size*

$$W(A,B) = \sum_{i \in A, j \in B} w_{ij}. \tag{2.1}$$

In a network partitioning problem one wants to divide the graph into k clusters $A_1, A_2, \dots, A_k$ such that the cut size among all of them is minimized

$$\operatorname{cut}(A_1, \dots, A_k) := \frac{1}{2} \sum_{i=1}^{k} W\left(A_i, \bar{A}_i\right) \tag{2.2}$$

where $\bar{A}_i$ denotes the complement of cluster A_i. This kind of problem is called *mincut* problem. Minimizing directly 2.2 might lead to a trivial partitioning, since it simply separates individual vertices from the rest of the network. Thus one needs to slightly modify the optimization problem such that the size of the clusters is automatically taken into account. The two main optimization functions used are called *RatioCut* and *Ncut* (where "N" stands for normalized).

The *RatioCut* of a partitioning $A_1, A_2, \ldots, A_k$ is defined as

$$\mathrm{RatioCut}\left(A_1, \ldots, A_k\right) := \frac{1}{2}\sum_{i=1}^{k} \frac{W\left(A_i, \bar{A}_i\right)}{|A_i|} = \sum_{i=1}^{k} \frac{\mathrm{cut}\left(A_i, \bar{A}_i\right)}{|A_i|} \tag{2.3}$$

where $|A_i|$ is the number of vertices in A_i. It was first used by [56] for the problem of circuit partitioning [2], in comparison with the traditional heuristic algorithm of *Kernighan–Lin* [31] for iterative bisectioning.

The *Ncut* of a partitioning $A_1, A_2, \ldots, A_k$ is defined as

$$\mathrm{Ncut}\left(A_1, \ldots, A_k\right) := \frac{1}{2}\sum_{i=1}^{k} \frac{W\left(A_i, \bar{A}_i\right)}{\mathrm{vol}\left(A_i\right)} = \sum_{i=1}^{k} \frac{\mathrm{cut}\left(A_i, \bar{A}_i\right)}{\mathrm{vol}\left(A_i\right)} \tag{2.4}$$

where $\mathrm{vol}\left(A_i\right)$ is the number of edges in A_i. It was first proposed by [48] for the problem of image segmentation [3].

Both normalization choices ensure that the clusters are not too small, avoiding the problem of cut minimization. In particular RatioCut balances the partitions with respect to the number of nodes, and thus it looks for an equipartition of the graph, independently of the intrinsic structure of the clusters. Ncut balances the partitions with respect to the number of edges internal to the clusters and between them. Thus it measures not only the dissimilarity between different clusters but also the similarity within groups (measured precisely by the internal density of edges).

Both RatioCut and Ncut are NP hard problem. As was first noted by [24], a relaxed version of the RatioCut minimization can be solved using spectral properties of the unnormalized (combinatorial) Laplacian matrix. Similarly, as stated by the original authors [48], a relaxed version of the Ncut problem can be solved by te normalized Laplacian matrix via the so called Spectral Clustering, that is the topic of the next section.

[2]The problem of circuit partitioning consists in placing electronic circuits on a board such that the elements on different boards are connected with the lowest number of links.

[3]Image segmentation is the problem of partitioning a digital image into multiple segments such that the pixels in each region share some characteristics.

2.2 Spectral clustering

In section 1.7 we saw that the Laplacian eigenvectors provide a natural method to draw a graph (i.e. embed the graph on a 2D manifold). Nodes that are strongly connected between them will tend to be closer in an optimal embedding of the network. This idea was motivated by the analogy with a spring-mass system 1.6 and by the Courant-Fisher theorem REF O CIT. In this section the ideas of sections 1.6 will be presented from a different perspective, showing how they can be exploited to partition a network in densely connected communities. This partitioning method is called *spectral clustering.*

As shown in section 1.2, if a graph of N vertices consists of k connected components, the first k eigenvalues are degenerate $\{\lambda_i\}_{i=0,\dots,k-1} = 0$. The Laplacian matrix can thus be written in block-diagonal form, with each block matching one connected component. Therefore, the k eigenvectors $\{\vec{v}_{\lambda_i}\}_{i=0,\dots,k-1}$ corresponding to the degenerate eigenvalues $\{\lambda_i\}_{i=0,\dots,k-1} = 0$ have uniform components along their block, and vanishing component elsewhere. $\{\vec{v}_{\lambda_i}\}_{i=0,\dots,k-1}$ are thus indicators of each component, that is

$$\vec{v}_{\lambda_i} = \frac{1}{\sqrt{N_i}} \left(\underbrace{1, 1, 1}_{N_i}, 0, \dots, 0 \right) \quad \text{for} \quad i = 0, \dots, k-1$$

where N_i is the number of nodes in the connected component number i. Thus let's consider the $N \times k$ matrix with the first k eigenvectors as columns. The i-th row is a k-dimensional vector that represent the i-th node. Given the consideration above, the vectors representing nodes belonging to the same connected component coincide. They are of the form $(0, 0, \dots, 1, \dots, 0, 0)$, and so they lie on the axes of a k-dimensional system of coordinates. The euclidean embedding performed with the Laplacian eigenvectors is thus able to detect the k components.

From a graph clustering perspective, this is an ideal case, in which the partitioning of the network is trivial. Nevertheless, it provides the general idea behind Laplacian spectral clustering, that can be justified by a perturbation theory approach [35]. In fact, if the graph is connected, but there are k subgraphs weakly linked to each other, the Laplacian cannot be put anymore in block diagonal form. However, the off-diagonal elements will be sparser than the diagonals blocks. Thus the lowest k non vanishing eigenvalues are close to zero $\{\lambda_i\}_{i=0,\dots,k-1} \approx 0$. Differently from before, the embedding vectors (the rows of the first k eigenvectors) of nodes belonging to the same community do not coincide. Nevertheless, they are close to each other (several possible metrics might be used), and thus the corresponding eigenvectors should still be able to distinguish the k communities, if used as a k-dimensional embedding. Finally, once we have a spectral embedding, traditional data clustering methods like k-means and k-medoids [25] can be used to retrieve the partitions of the nodes.

A drawback of this method is that the number of expected communities has to be provided beforehand, rather than being produced as output by the algorithm itself [4]. Thus one is forced to make assumptions that are often arbitrary. This problem might be partially overcome observing the spectrum of the Laplacian matrix. As noted previously, if the k communities are weakly connected to each other, one expects the first k eigenvalues $\{\lambda_i\}_{i=0,\dots,k-1}$ being close to zero, while there is a bigger gap between λ_k and λ_{k+1}. Thus, the number of clusters might be derived heuristically checking whether there is a number k corresponding to a relevant step in the spectrum. On the other hand, if the network is highly homogeneous, without a well pronounced community structure, this kind of reasoning might be of little help. In these cases the number of eigenvalues to be taken is highly arbitrary.

2.2.1 Ratiocut and Ncut approximations

The idea of spectral clustering might be used to solve relaxed versions of the Ratiocut and Ncut problems of section 2.1. The general idea for the Ratiocut and $k = 2$ is the following. First, it can be shown [35] that the measure to minimize 2.3 can be expressed in a matrix form through the Laplacian matrix as

$$\mathrm{RatioCut}(A, \bar{A}) = \frac{1}{|V|} f' L f \tag{2.5}$$

where A and $\bar{A}$ are generic partitions of the graph into two complementary subsets, and f is an indicator matrix such that

$$f_i = \begin{cases} \left(\frac{|\bar{A}|}{|A|}\right)^{\frac{1}{2}} & \text{if } v_i \in A \\ -\left(\frac{|A|}{|\bar{A}|}\right)^{\frac{1}{2}} & \text{if } v_i \in \bar{A} \end{cases} . \tag{2.6}$$

The minimization of the Ratiocut is thus expressed in an equivalent discrete optimization problem that involves the Laplacian matrix quadratic form with partition indicator vectors. The relaxation consists in allowing the entries of the indicators vector to take also real values, turning the NP hard problem into a tractable optimization problem. The solutions of 2.5 are given by the Laplacian eigenvectors. Since we are considering the case $k = 2$, the solution is given by the Fiedler vector $\vec{v}_1$. Its components are thus indicators of the two partitions. Analogously to the spectral clustering, from the components of the Fiedler vector it is possible to retrieve the two partitions simply applying the sign function or, better, employing the k-means algorithm.

More generally, spectral clustering includes all the kind of techniques that allow to partition the network into communities by mean of the eigenvectors of matrices.

[4]This problem is common to the most used clustering technique, cut-based partitioning included. Hierarchical clustering is an exception [25]

Traditionally it is used as a data clustering technique. Given a set of N objects, a pairwise similarity matrix S is defined, such that $s_{ij} \geq 0$. The similarity matrix can be interpreted as the adjacency matrix of a graph over the data set. Computing the corresponding Laplacian matrix (unormalized or normalized), the general idea is to transform the initial objects into a set of points in a metric space, whose coordinates are the components of the Laplacian eigenvectors. The change of representation induces by the eigenvectors makes the cluster structure of the system more evident.

2.3 Modularity

A network with random connection, like the Erdős–Rényi model, is not expected to have a community structure. Thus one might define network communities as groups of nodes with a denser connectivity than the one expected from a pure random structure. This is the idea on which the *modularity* function introduced by [41] is based.

First, we need the number of edges running between nodes belonging to the same community C_i, that is given by

$$\sum_{\text{edges}(i,j)} \delta_{C_i C_j} = \frac{1}{2}\sum_{ij} A_{ij}\delta_{C_i C_j} \tag{2.7}$$

where the factor 1/2 is due to the double counting of nodes pairs. To find the expected number of edges within a groups of nodes if the connections were random, we need first to define a null model. The mostly used one consists in making the connections random but keeping the nodes degree distribution. Thus, consider a particular edge attached to a node i with degree k_i. The probability that the other end of this edge is a node with degree k_j is $\frac{k_j}{2M-1} \approx \frac{k_j}{2M}$ for large networks, where M is the total number of edges in the network. Considering all the k_i connection of node i, the expected number of edges between nodes i and j is $\frac{k_i k_j}{2M}$. since the edges are places independently one from each other. The expected number of links between nodes of the same community is

$$\frac{1}{2}\sum_{ij} \frac{k_i k_j}{2m}\delta_{C_i C_j}. \tag{2.8}$$

Subtracting 2.8 from 2.7 and normalizing with the total number fo edges we get to the definition of modularity

$$Q = \frac{1}{2M}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2M}\right)\delta_{C_i C_j}. \tag{2.9}$$

Modularity equals to 0 if the whole graph is in a single community, since the two contributions in 2.9 is equal and opposite. Conversely, if each node is in its own cluster, $Q < 0$, since the first term vanishes. Otherwise, modularity is always smaller than 1.

Since higher values of modularity imply a better clustering of the network, several algorithms have been proposed to retrieve a network partitioning optimizing the modularity value. However, the number of possible configurations is huge, and it has been proven than modularity optimization is a NP hard problem [8]. Several strategies have been proposed, among which the algorithm [11], that will be used in chapter 4. Modularity maximization also admit a spectral optimization analogue [39]. Similarly to the spectral clustering of section 2.2, the eigenvectors of the modularity matrix, defined as $B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$, are used to solve a relaxed version of the modularity optimization problem.

Finally, it is worth mentioning that modularity is also used as a simple heuristics to benchmark the efficiency of other clustering methods [16].

Chapter 3

Dynamic based clustering

In the previous chapter we have introduced the main traditional clustering methods for undirected networks. As seen, they basically rely on the edges density in order to partition a network, given the desired number of partitions. Thus, they exclusively take into account the structure of the network, its composition as represented by the adjacency matrix, disregarding eventual processes running on top of them. The topology of a graph greatly influences the characteristics of the dynamical process, but alone it is not able to describe it [47].

Instead of finding communities of nodes based on their connectivity, one might be interested in finding a *coarse-grained* description of the network based on the dynamical properties of the process occurring on top of it. The idea is that structural communities of chapter 2 are replaced by group of nodes that evolve in a similar fashion, forming building blocks for the dynamics description. Finding a dynamically relevant coarse-grained description of a graph has gained a lot of interest recently in network science [16] [45]. Of course, the particular cluster definition to be used depends on the dynamical features of the system. For instance, nodes communities might be defined as groups of nodes in which a random walker remains trapped for a relatively long time [44] (more on this in the following section). Analogously, for oscillating systems (like the Kuramoto model for neuronal activity [32]) communities might be identified as groups of nodes that synchronize first [3]. In both cases, one quantifies nodes similarities by variables that depends both the dynamics and the underlying graph topology.

Using the above mentioned point of view, it is implicitly assumed that the network connections are fixed in time. Other than studying the network structure and the dynamics occurring, a third situation might arise, that is a graph whose structure has its own dynamical properties. In other words, one might in principle consider evolving networks (also called *temporal networks*), in which the graph is subject to a structural dynamics over time. For instance, these kind of networks are used to describe social interactions, that are subject to time evolution. The evolving nature of the graph makes the study of the dynamical processes much harder than the static counterpart [29].

The evolution of the graph might also reflect the dynamics taking place on it. For instance, in neural networks model the weights of the links (synapses) change accordingly to the activity of the nodes (neurons) connecting them. This is the mechanism used to describe neural plasticity and learning [30]. Therefore, this situation the system is also characterized by an influence of the dynamics over the structure, and not only the other way round.

For the rest of this script, the focus is on time fixed networks on which a diffusion process is taking place. Given a particular graph, we look for partitions that are aligned with the dynamical process. The present chapter contains the original results of this thesis work, i.e. a novel method for partitioning a network on which a diffusion dynamics is taking place. We start by reviewing the spectral clustering method of section 2.2 from a dynamical point of view, further explaining how it can be exploited to effectively reduce the dimensionality of the network. Afterwards, in section 3.3, we briefly describe a generalization of the diffusion processes on networks introduced in 1.5, that will be used for all the subsequent sections.

3.1 Spectral clustering and diffusion processes

In section 1.5 the Laplacian matrix was used to describe diffusion processes on networks, given its correspondence with the continuous Laplacian operator. The solution 1.22 that we rewrite

$$\vec{p}(t) = \sum_{\lambda \neq 0} c_\lambda e^{-\lambda t} \vec{v}_\lambda + \vec{p^s} \tag{3.1}$$

depends of the spectrum of the Laplacian matrix. From 3.1 we see that each eigenmode of the Laplacian matrix decay with its own characteristic time scale that equals to

$$\tau_k = \frac{1}{\lambda_k} \quad \text{for} \quad k = 1, \dots, N \tag{3.2}$$

excluding the kernel component that of course it is not involved in the diffusion process and remains constant throughout the transient. Hence, if there are big gaps between some eigenvalues, the system will show a time scale separation between the relaxation times of the corresponding eigenmodes.

The traditional spectral clustering technique relies on the fact that the Laplacian matrix spectrum holds topological information about the network. As explained in section 2.2, spectral clustering can be justified by means of a perturbation theory approach. First, one starts with a graph with k disconnected components, for which $\{\lambda_i\}_{i=0,\dots,k} = 0$ and the corresponding eigenvectors are each a indicator of one connected component. Then the network is perturbed adding weak connections between the components. The kernel of L is not anymore degenerate, but still the first k eigenvectors can be used to retrieve k distinguishable partitions of the graph, whereas eigenvectors with higher energies correspond to normal modes inside each of the k communities found.

Since the method itself do not provide the optimal number k of eigenvectors to be used, one usually uses a heuristic approach and look for the presence of a gap in the spectrum of L. In the particular case in which the Fiedler value λ_1 is big, the network can be naturally partitioned into two communities of densely connected nodes.

From equation 3.1 it is possible to justify the spectral clustering method also from a dynamical point of view. Consider again a graph featuring a big gap in the spectrum of L between the eigenvalues λ_k and λ_{k+1}. From 3.20 we see that for a diffusion process (or a random walker) the eigenmodes greater than λ_k, that correspond approximately to normal modes inside the communities, reach equilibrium first. After $t = 1/\lambda_{k+1}$ also the eigenmodes between the communities start decaying, leading the system to a uniform solution over the whole network. The slower the relaxation of the eigenmodes $\{\lambda_1 \dots \lambda_k\}$ is, with respect to the higher ones (i.e. the higher is the gap $\lambda_{k+1} - \lambda_k$), the more the k communities are well separated one from the other. In this case, the eigenmodes $\{\lambda_{k+1}, \dots, \lambda_n\}$ are negligible for $t > 1/\lambda_{k+1}$, and the system can be described in a coarse-grained manner by the first k eigenmodes. Therefore, $\{\lambda_1 \dots \lambda_k\}$ form an invariant subspace valid after the time scale $1/\lambda_{k+1}$, i.e. after the eigenmodes internal to the clusters have decayed. This lower dimensional dynamical description is thus strictly associated with localized substructures of the graph.

This is an example about how the topology of the network influence the dynamics taking place on it. In this particular case, the connections is due to the double role of L, that holds information both about the topology of the graph and the diffusion dynamics taking place. We mention that L plays a similar role also for synchronization processes [37]. In the following section, the idea of an effective lower dimensional description of the diffusion process is formalized using perturbation theory.

3.2 Coarse-grained network

As mentioned in section 1.6, the eigenvalues of the Laplacian matrix can be interpreted as the energies of the normal modes of a spring-mass system. In this analogy, the nodes of the network are the masses and the edges corresponds to the springs, with elastic constant equal to the edges weights w_{ij}. If the masses are all equal, the system dynamics is governed by the unnormalized Laplacian matrix, with equation 1.28. Lower eigenmodes represent oscillations that spread through the whole network, like the Fiedler vector. Higher eigenmodes corresponds to oscillations of high energy, and they are localized on the edges with higher weights [28]. These are the only eigenmodes energetic enough to generate a dipole change on those strongly connected nodes, i.e. they are able to distinguish them, moving them away one from the other. Moreover, according to the discussion of the previous section, these are the eigenmodes that relax faster in a diffusion process. On the other hand, lower eigenmodes perceive those strongly connected nodes effectively as a single entity.

Thus one can think of a procedure in which two strongly connected nodes are glued together. This means that the weight of their connection w_{kh} diverges and the state of the two node, as seen by the lower eigenmodes, is exactly the same. Moreover, the corresponding eigenvalue should tend to infinity, implying that more and more energy is required to move those nodes apart. In this way, the dimensionality of the network is reduced from N to $N-1$ effective degrees of freedom.

Iterating this procedure, one can further reduce the dimensionality of the network. Generally, the collapsing procedure means to perform a limit $\lambda \to \infty$ of the eigenvalues that are associated to the eigenvectors that distinguish the nodes in a small spatially confined community. The idea is that monitoring only the reduced network one could get relevant information on the state of the whole graph, since each monitored node should represents the other nodes that were collapsed to it, keeping the dynamics under control. As already mentioned, this reduced version of the graph is also called *coarse-grained graph* [1].

This reduction procedure is therefore a large scale map from a graph to a smaller one, and it helps to unveil properties of the graph making its study easier, specially if it preserves the eventual dynamics defined on the original graph. That is, one usually looks for a coarse-grained graph that form a reduced model of the dynamics taking place on the original network, in which blocks of nodes are aggregated to single nodes whose dynamical function is similar.

However, it is necessary to study what happens to the other eigenvectors and eigenvalues, i.e. we need to check if (part of) the spectrum of the Laplacian matrix is preserved under this coarse-graining procedure. The more they are dissimilar and the more the procedure is losing information about the original network. This is what we are going to do using a perturbation approach [5].

Consider an edge, say (kh) and increase its weight w_{kh}

$$w_{kh} \longrightarrow w'_{kh} = w_{kh} + \gamma \quad \text{with} \quad \gamma > 1. \tag{3.3}$$

The corresponding Laplacian matrix change reads

$$L \longrightarrow L' = L + \gamma \delta L$$

with

$$\delta L_{ij} = \begin{cases} \frac{1}{2} & \text{for } i = j = k \text{ or } i = j = h \\ -\frac{1}{2} & \text{for } i = k, j = h \text{ or } i = h, j = k. \end{cases}$$

[1] If in the reduced network there are nodes belonging to the original one, this procedure is also called network *down-sampling* [49]

Thus, δL is a Laplacian matrix with eigenvectors

$$\lambda = 1 \implies \vec{v} = \left(0, \ldots, 0, \underbrace{1/\sqrt{2}}_{k}, 0, \ldots, 0, \underbrace{-1/\sqrt{2}}_{h}, 0, \ldots, 0\right)^T$$

and for the invariant subspace with $\lambda = 0$, a possible basis choice might be

$$\lambda = 0 \implies \vec{u}_\lambda = \begin{cases} (1, 0, \ldots, 0)^T \\ (0, 1, \ldots, 0)^T \\ \quad \vdots \\ \left(0, \ldots, 0, \underbrace{1/\sqrt{2}}_{k}, 0, \ldots, 0, \underbrace{1/\sqrt{2}}_{h}, 0, \ldots, 0\right)^T \end{cases} . \tag{3.4}$$

We will call $\vec{v}$ *dipole*, since it represents the above mentioned high energy eigenmode that is able to distinguish the two strongly connected nodes. Notice that $\delta L_{ij} \equiv \Pi^{\|}_{ij}$ is a projector on the subspace generated by the dipole vector $\vec{v}$. Defining

$$\varepsilon = \frac{1}{\gamma}$$

the equation for the Laplacian perturbation becomes

$$L'_{ij} = L_{ij} + \varepsilon^{-1}\delta L_{ij} \quad \text{with} \quad \varepsilon \to 0. \tag{3.5}$$

As explained above, the idea is to study how the procedure $\gamma \to \infty$, ($\varepsilon \to 0$), changes the network properties. Of course, we cannot use a perturbative approach on the Laplacian matrix L. Thus, since $\varepsilon \to 0$, we might perform a perturbative approach considering δL_{ij} as the unperturbed part and L_{ij} as the perturbation. Therefore, the eigenvalues equation for the Laplacian perturbation δL is given by

$$\left(\varepsilon L_{ij} + \Pi^{\|}_{ij}\right)(v_j + \varepsilon \delta v_j) = (1 + \varepsilon\delta\lambda)(v_i + \varepsilon\delta v_i) \tag{3.6}$$

where we have written explicitly the perturbation parameter ε. Recalling that $\Pi^{\|}_{ij} v_j = \delta L v_j = v_j$, and that $v_i = 0$ for $i \neq k, h$ we get to

$$L_{ij}(v_j + \varepsilon\delta v_j) + \Pi^{\|}_{ij}\delta v_j = \delta\lambda v_i + (1 + \varepsilon\delta\lambda)\delta v_i.$$

Projecting the above equation to the $N - 1$ dimensional subspace of the eigenvectors with $\lambda = 0$, i.e. the subspace orthogonal to the dipole, we get

$$\Pi^{\perp} L_{ij}(v_j + \varepsilon\delta v_j) + \underbrace{\Pi^{\perp}\Pi^{\|}_{ij}}_{=0}\delta v_j = \underbrace{\delta\lambda\Pi^{\perp} v_i}_{=0} + (1 + \varepsilon\delta\lambda)\Pi^{\perp}\delta v_i$$

and thus

$$\Pi^{\perp} L_{ij}\left(v_j + \varepsilon \delta v_j\right) = (1+\varepsilon\delta\lambda)\Pi^{\perp}\delta v_i. \tag{3.7}$$

Since $\Pi^{\|}_{ij} v_j = v_i$, projecting the same equation in the subspace parallel to the dipole and setting $\Pi^{\|}\delta v_i = 0$, we obtain

$$\delta\lambda v_i = \Pi^{\|} L_{ij} v_j.$$

Performing the scalar product with v_i, since it is normalized to 1, we get to

$$\delta\lambda = v_i \Pi^{\|} L_{ij} v_j.$$

From equation 3.7, one easily gets

$$\delta v_i = \frac{\Pi^{\perp} L_{ij}}{1+\varepsilon\delta\lambda}\left(v_j + \varepsilon\delta v_j\right) \tag{3.8}$$

that defines iteratively the perturbation δv_i of the dipole eigenvector. Since the subspace with $\lambda = 1$ is one-dimensional, the perturbation to the dipole must be perpendicular to the dipole itself. If ε is small compared to the norm of L we have that the solutions exists and it is given by

$$\lim_{\varepsilon\to 0} \delta v_i = \Pi^{\perp} L_{ij} v_j \tag{3.9}$$

The limit $\varepsilon \to 0$ means that the edge weight $w_{kh} \to \infty$, i.e. the two nodes k and h becomes infinitely coupled. Equation 3.9 tells us that the perturbation of the dipole eigenvector, in the limit of an infinite elastic constant w_{kh} is given by the projection on the dipole space of $L_{ij}v_j$, i.e. the image of the dipole eigenvector through the Laplacian matrix L_{ij}, here considered to be a perturbation to the dipole. Equation 3.9 is therefore a residual coupling, that has no effect on the dynamics, between the the two nodes k and h and the rest of the network,

Considering the eigenvalues equation for the other eigenvectors of δL_{ij} (equation 3.4 with $\lambda = 0$). We can choose $N-1$ independent arbitrary vectors belonging to that subspace. We have that

$$\left(\varepsilon L_{ij} + \Pi^{\|}_{ij}\right)\left(u_j + \varepsilon\delta u_j\right) = \varepsilon\mu\left(u_i + \varepsilon\delta u_i\right)$$

and, since the subspace with $\lambda = 0$ is $N-1$ degenerate, we take $\delta u_j = \alpha v_j$, thus getting to

$$\left(\varepsilon L_{ij} + \Pi^{\|}_{ij}\right)\left(u_j + \alpha\varepsilon v_j\right) = \varepsilon\mu\left(u_i + \alpha\varepsilon v_i\right)$$

and recalling that $\Pi^{\|}_{ij} u_j = 0$ and $\Pi^{\|}_{ij} v_j = v_i$ we obtain

$$L_{ij} u_j + \alpha\left(v_i + \varepsilon L_{ij} v_j\right) = \mu u_i + \varepsilon\alpha\mu v_i. \tag{3.10}$$

Like before, we project this equation on the subspace perpendicular and parallel to the dipole vector. In the first case we have

$$\Pi^{\perp} L_{ij} u_j + \underbrace{\alpha \Pi^{\perp} v_i}_{=0} + \varepsilon \alpha \Pi^{\perp} L_{ij} v_j = \underbrace{\mu \Pi^{\perp} u_i}_{=\mu u_i} + \underbrace{\varepsilon \alpha \mu \Pi^{\perp} v_i}_{=0}$$

from which using $(\Pi^{\perp})^2 = \mathbb{I}$ and $\Pi^{\perp} u_i = u_i$ we get to

$$\Pi^{\perp} L_{ij} \Pi^{\perp} u_j = \mu u_i - \varepsilon \alpha \Pi^{\perp} L_{ij} v_j. \tag{3.11}$$

For the component parallel to the dipole we have

$$\Pi^{\|} L_{ij} u_j + \underbrace{\alpha \Pi^{\|} v_i}_{=\alpha v_i} + \varepsilon \alpha \Pi^{\|} L_{ij} v_j = \underbrace{\mu \Pi^{\|} u_i}_{=0} + \underbrace{\varepsilon \alpha \mu \Pi^{\|} v_i}_{=\varepsilon \alpha \mu v_i}$$

and finally performing the scalar product, recalling that $\sum_i v_i^2 = 1$, we get

$$\alpha(1 - \varepsilon\mu) = -v_i L_{ij} \left(u_j + \varepsilon \alpha v_j\right) \tag{3.12}$$

The system of equations 3.12 and 3.11

$$\begin{cases} \Pi^{\perp} L_{ij} \Pi^{\perp} u_j & = \mu u_i - \varepsilon \alpha \Pi^{\perp} L_{ij} v_j \\ \alpha(1 - \varepsilon\mu) & = -v_i L_{ij} \left(u_j + \varepsilon \alpha v_j\right) \end{cases} \tag{3.13}$$

defines how the Laplacian properties of the reduced network are related to the original one. The second equation determines the parameter α. The first equation defines the Laplacian matrix of the reduced graph, that is therefore given by

$$L_{ij} \to \Pi^{\perp} L_{ij} \Pi^{\perp}. \tag{3.14}$$

We use the letter σ to defined the eigenvalues of $\Pi^{\perp} L_{ij} \Pi^{\perp}$ (in the subspace perpendicular to the dipole), and they are given by

$$\sigma := \mu + \varepsilon \delta\sigma$$

where the correction $\varepsilon\delta\sigma$ is due to the term $\varepsilon\alpha\Pi^{\perp} L_{ij} v_j$. The eigenvectors of $\Pi^{\perp} L_{ij} \Pi^{\perp}$ are u^{σ} with $u^{\sigma} := u + \varepsilon \delta u^{\sigma}$ where the correction δu^{σ} lies in the subspace perpendicular to the dipole $\delta u^{\sigma} \perp u$. Considering the case $\sigma = 0$, in the limit $\varepsilon \to 0$ we recover the new stationary solution, that is orthogonal to the dipole vector $\vec{v}$.

If during the limit $\varepsilon \to 0$ (that is the weight $w_{kh} \to \infty$) two eigenvalues of $\Pi^{\perp} L_{ij} \Pi^{\perp}$ are perturbed enough becoming a single degenerate eigenvalue, the perturbation approach might break down. In this case in fact, a two dimensional invariant subspace is created, and any couple of independent vectors belonging to this subspace is an eigenvector base for the corresponding eigenvalue. Therefore the perturbation approach diverges (the denominator is $\lambda - \mu$), signaling that it loses control about the eigenvector in that subspace. Therefore, in the limit $\varepsilon \to 0$, the spectrum of $\Pi^{\perp} L_{ij} \Pi^{\perp}$ may encounter bifurcations, i.e. two eigenvalues that become degenerate and then cross each other. In this case the spectral properties of the reduced graph Laplacian $\Pi^{\perp} L_{ij} \Pi^{\perp}$ may be different from the ones of the original matrix L. Thus one need to keep the bifurcations under control, that means taking to infinity only those weights that are high enough.

Notice that the projection operator $\Pi^{\perp}$ can be represented by a $N-1 \times N$ matrix whose rows are the eigenvectors 3.4. Therefore, it acts on a vector leaving unchanged its components different from k and h, while it mixes the sub-spaces of nodes k and h, summing them. This corresponds to the single entity generated by taking the weight w_{kh} to infinity, i.e. a node that behaves effectively like the sum of the two original nodes k and h. The analogy with a spring-mass system (explained in section 1.6), can give further physical insights. In fact, the reduced Laplacian matrix $\Pi^{\perp} L_{ij} \Pi^{\perp}$, that is $N-1 \times N-1$ dimensional, represents a network similar to the original one, with the only difference that the nodes k and h are replaced by a single node, with double the mass of the original nodes (that were taken to be the same, since we used the unormalized Laplacian matrix), and whose connections (springs) with the rest of the network are the sum of the original connections of nodes k and h.

Once this procedure has been performed, i.e. a partitioning of the network is established, one might be interested in applying a control or observation procedure on the graph, in which all the nodes of the reduced network have to be monitored. A local perturbation can be identified if it changes the state of one or multiple nodes of the reduced network.

We mention that if the reduced network is generated looking for *external equitable partition*, it is also called *quotient graph*. In this particular case, the spectral properties of the Laplacian matrix are preserved [42].

3.3 Diffusion model with forcing

Consider the following master equation

$$\dot{p}_i = \sum_j (w_{ij} p_j - w_{ji} p_i) + s_i = -\sum_j L_{ij} p_j + s_i \tag{3.15}$$

that is a generalization of 1.20 with the adding a forcing term s_i. Like previously, we will call $w_{ij} \geq 0$ weights or transition rates for the link (i, j) and p_i will be called potential (or signal) and it quantifies the concentration of the diffusing substance on node i. According to the system under study, the physical interpretation of $\vec{s}$ may vary. If equation 3.15 describes a heat diffusion process, the forcing represents the generation of internal heat. For a spring-mass system, we already encountered this equation in section 1.6, since it was obtained enforcing the position of some masses. For an electrical network, $\vec{s}$ may be interpreted as an imposed current. Finally, for social sciences, the above equation may describe an opinion dynamics, and the forcing is related to stubborn agents [20].

As already noted in section 1.5, $w_{ij}p_j$ represents the transition rate from node j to node i. Thus

$$Q_{ij} = w_{ij}p_j - w_{ji}p_i = w_{ij}(p_j - p_i) \tag{3.16}$$

is the net flux on the edge (i, j). If $Q_{ij} > 0$ we have a flux from j to i, if less than zero the flux direction is the opposite.

$$Q_i := \sum_j Q_{ij} = -\sum_j L_{ij}p_j$$

is the net flux on node i due to the diffusion exchange with adjacent nodes. Q_i is greater than zero if there is a net incoming flux to node i, whereas is less than zero if it outgoing. The forcing s_i thus represents the presence of local sources and sinks that define the boundary conditions and the constraints of the transport system. $s_i > 0$ means that the node acts as if it is a source, i.e. it produces the quantity that flows in the network, $s_i < 0$ as a sink of the system, i.e. it consumes. We will call the s_i *demands* or *loads* of the network.

Therefore, equation 3.15 says that the time variation of p_i is given by the net flux on node i, due both to network exchanges and the external load: if the net flux is positive, the pressure will increase, conversely it will decrease if it is negative.

In order for the system 3.15 to relax to a stationary condition, we require that the forcing belongs to the invariant subspace of the matrix L, given by $\mathrm{Span}\{\vec{v_1}, \dots, \vec{v_N}\}$. As seen in 1.2, for a connected graph ($\lambda_2 > 0$) the image of L consists of all vectors that are orthogonal to the vector $\vec{1}$), i.e. we require that

$$\sum_i s_i = 0. \tag{3.17}$$

The physical meaning of 3.17 is that we do not want a total net incoming or outgoing flux in our system, i.e. we are considering a micro-canonical ensemble If $\sum_i s_i > 0$ we would indefinitely load our system, leading to an infinite solution. If $\sum_i s_i < 0$ we would eventually completely unload our system, leading to a zero solution. Thus s must not have components in the kernel of L. According to the physical interpretation that we give to the forcing s_i, we can either intend the condition 3.17 as a property of the forcing or as if the system is closed. It is possible a decomposition of the forcing into dipole couples $s\vec{e}_{ij}$ where the vectors $\vec{e}_{ij}$ are the canonical base of the subspace (3.17

$$\vec{e}_{ij} = (0, .., 1_i, .., -1_j, ...0). \tag{3.18}$$

From linear systems theory, we know that the complete solution of 3.15 is given by the exponential decay (since all the eigenvalues of L are positive). Therefore using the spectral decomposition of the Laplacian matrix

$$L = \sum_{\lambda \neq 0}^{n} \lambda \vec{v}_\lambda \vec{v}_\lambda^\top \tag{3.19}$$

the solution reads

$$\vec{p}(t) = \sum_{\lambda \neq 0} c_\lambda e^{-\lambda t} \vec{v}_\lambda + \vec{p^s} \tag{3.20}$$

where $\vec{v}^\lambda$ are the Laplacian eigenvector $\vec{p^s}$ is the average stationary solution and the coefficients c_λ are determined by the initial conditions

$$\vec{p}(0) = \sum_{\lambda \neq 0} c_\lambda \vec{v}_\lambda + \vec{p^s}. \tag{3.21}$$

For $t \to \infty$ the first term decay to zero $\forall \lambda$, and for any initial condition the system relaxes to the stationary solution $\vec{p^s}$, that is given by

$$L\vec{p^s} = \vec{s} \tag{3.22}$$

from which we recover the condition 3.17

$$\sum_i s_i = \sum_i \sum_j L_{ij} p^s_j = \sum_j p^s_j \sum_i L_{ij} = 0 \tag{3.23}$$

The stationary condition 3.25 is solved by the *pseudoinverse* (or *generalized inverse*) L^+[6] of the Laplacian matrix [2]

$$\vec{p^s} = L^+ \vec{s} \tag{3.24}$$

or more explicitly, from the knowledge of the spectral properties of the matrix L, we write the solution in the form

$$p^s_i = \sum_{\lambda \neq 0} \frac{s^\lambda_i}{\lambda} + p^0_i \tag{3.25}$$

where s^λ_i are the components of the forcing on the eigenvector v^λ and $v^0 = \frac{1}{N}(1, \ldots, 1)$ is the normalized null eigenvector of L. From 3.25 we note that without the forcing term we recover the standard diffusion solution of section 1.5, given by the kernel of L.Thus, taking s satisfying 3.17 the problem is well posed, but the solution is not unique. We can add an arbitrary kernel component (i.e. uniform vector) to $\vec{p^s}$ and 3.25 is still satisfied. This situation is analogue to the ground choice for the voltages of an electrical system.

[2]The pseudoinverse of the Laplacian matrix has found applications in theoretical chemistry and in resistor networks to define the effective resistance between two nodes, when each edge is a resistor of unit resistance [22]. It can be shown that the effective resistance is a distance metric on the graph and a robustness metric for diffusion processes [54]

Thus one might add a further constraint, like $\sum_i (p_i^s)^2 = 1$ or $\sum_i p_i^s = 0$, to get a unique solution. With the latter choice, the stationary solution lies in the same subspace of the forcing s_i and thus can be decomposed with the same basis 3.18.

Equation 3.25 shows as the stationary solution is insensitive to the changes of the forcing s_i along the eigenvectors with $\lambda \gg 1$. Indeed, as explained for 3.2, the eigenvalue λ^{-1} defines the reaction time scales of the system to the external changes. Therefore when $\lambda \gg 1$ the system is insensitive to small changes in the forcing component along the corresponding eigenvalue.

From 3.15, we see that when the system reaches the stationary state

$$L\vec{p^s} = \vec{s}$$

the following condition is satisfied

$$\sum_j (w_{ij}p_j - w_{ji}p_i) + s_i = 0$$

that can be rewritten as

$$p_i^s = \frac{s_i + \sum_j w_{ij}p_j^s}{d_i} \tag{3.26}$$

which is similar to the equation 1.25, but with the adding of the forcing term s_i. Nevertheless, 3.26 generalizes the harmonic condition 1.25 to the case of an external forcing. Equation 3.26 can also be interpreted as a measure of the influence of the state j on the state i, and it depends on the fluxes between the nodes. Then one expects that p_i^s increases when s_i increases, according to the sum of the incoming flows. The connectivity d_i at denominator plays the role of an inertial term. Therefore the state of highly connected nodes is more robust.

From 3.26, we observe that when one of the flux is dominant with respect to the others, say $w_{ik}p_k$, we have a causality relation between the nodes i and k. Therefore, a change in node i implies that a change occurred in the node k. The idea of the proposed clustering method is to point out these causality relations and exploit them to monitor the network state only through a subset of nodes. Therefore, we want to achieve a coarse-grained graph, like the one explained in section 3.2, but taking into consideration also the dynamics of the network, that in this case is governed not only by the Laplacian matrix but also by the forcing s_i.

To find these causality relation, our approach is to define a stochastic process that influence the network state $\vec{p}$, making it fluctuate $\delta\vec{p}$, and then computing a similarity function between the nodes fluctuation. Since the system is linear, we expect these relations to be linear as well. Therefore, the similarity function used will be the correlation coefficient, that as known captures the linear relation between the variables.

The correlation matrix is then used to define the communities among the node. Since the nodes in a single cluster are correlated, the knowledge of a single node state gives information on the state of the others. Therefore, we propos the following procedure to observe the state of the network through a subset of nodes. First, we locate the most correlated nodes (in particular couples of correlated nodes). Secondly, we perform a clustering procedure where we join together correlated nodes and put a single sensor for each cluster (according to some criterion) and on the isolated node (not belonging to any cluster). There are several ways to extract a network partitioning starting from a similarity matrix, like the correlation matrix. They will be discuss in the next chapter. Nevertheless, the clustering procedure can also be performed simply by using a ranking in the correlation matrix, therefore ordering the nodes from the most correlated ones.

Once the network is partitioned and the sensors are placed, we want to study if a single-edge failure occurring wherever in the network can be localized by the sensor signals. Assuming that edge-failures have mainly local effect, the state change of a single sensor (whereas the others are not affected) would suggest that the perturbation is localized in the cluster that sensor node represents. Conversely a change that involves more clusters would suggest the the perturbation is located at the border area, called *edge boundary*, that is the set of edges that cross two clusters.

To apply this procedure, we first need to find the mentioned correlation matrix of nodes fluctuations. In order to do so, one can think of different stochastic processes. In particular, we consider two kind of processes. Firstly, we obtain a correlation matrix perturbing the system with an external noise, i.e. a noise applied to the forcing s_i. Secondly, we will consider a noise intrinsic to the structure of the network, i.e. a perturbation directly applied to the Laplacian matrix $L + \delta L$. In the following section we start describing the results of the first method.

3.4 External forcing perturbation

First of all, we compute how the system react if to the time-constant forcing $\vec{s}$ of the system we add a fluctuation term $\vec{\xi}(t)$, which represents the effects of environmental fluctuations. Equation 3.15 thus becomes

$$\dot{p}_i = \sum_j L_{ij} p_j + s_i + \xi_i(t). \tag{3.27}$$

According to the idea explained in the previous section, our aim is now to compute the correlation matrix for the fluctuations of the system variable p_i with respect to its stationary solution 3.25. This correlation function will not properly depend on the dynamics of the system, but nevertheless its encompass the general idea underlying the proposed method of section 3.5).

In order to satisfy the solvability condition 3.17, we need to add the following constraint

$$\sum_i \xi_i(t) = 0 \quad \forall\, t.$$

This kind of noise can be simulated considering N normalized independent variables $\eta_i(t)$ and subtracting from each of them their arithmetic average for each time

$$\xi_i(t) = \frac{\eta_i(t)}{\sqrt{\Delta t}} - \frac{1}{N}\sum_k \frac{\eta_k(t)}{\sqrt{\Delta t}}$$

where $\Delta t \equiv t' - t$ is the evolution time step. Further, we require each $\eta_i(t)$ to have zero average and be statistically independent form the others

$$E[\eta_k(t)] = 0 \qquad E[\eta_k(t')\eta_h(t)] = \delta_{kh}\delta_{t't}$$

thus the mean and covariance of the noise ξ_i follow in a straightforward manner

$$\begin{aligned} E[\xi_i(t)\xi_j(t')] = \frac{1}{\Delta t}\Big(E[\eta_i(t)\eta_j(t')] - \frac{1}{n}\sum_k E[\eta_i(t)\eta_k(t')] \\ - \frac{1}{n}\sum_k E[\eta_j(t')\eta_k(t)] + \frac{1}{n^2}\sum_{kh} E[\eta_k(t)\eta_h(t')]\Big) \end{aligned}$$

and computing

$$\begin{aligned} \sum_k E[\eta_i(t)\eta_k(t')] =& \delta_{tt'}\sum_k \delta_{ik} = \delta_{tt'} \\ \sum_{kh} E[\eta_k(t)\eta_h(t')] =& \delta_{tt'}\sum_{kh} \delta_{kh} = N\delta_{tt'} \end{aligned}$$

we get to the following statistical properties for the noise variables $\xi_i(t)$ (for $\Delta t \to 0$)

$$E[\xi_i(t)] = 0 \quad C_{ij} := E[\xi_i(t')\xi_j(t)] = \left(\delta_{ij} - \frac{1_{ij}}{N}\right)\delta(t'-t). \tag{3.28}$$

We remark that the covariance matrix 3.28 satisfies

$$\sum_j C_{ij} = 0$$

and thus it has the same invariant subspace 3.17 of the Laplacian matrix L. C_{ij} is in fact a projector to that subspace, as can be simply verified by

$$\sum_i \left(\sum_j C_{ij}v_j\right) = \sum_i \left(v_i - \frac{1}{n}\sum_j v_j\right) = 0.$$

Restricted to the subspace 3.17, the covariance matrix 3.28 reduces to the identity matrix $C_{ij} = \delta_{ij}$. This is a consequence of having choose the fluctuations ξ_i to have vanishing kernel component. Since the fluctuations of the forcing due to the external noise produce corresponding fluctuations of the system state, we also expect the fluctuations δp_i to have no components in the kernel of L. Thus expanding perturbatively around the stationary state

$$p_i(t) = p_i^s + \delta p_i(t) \tag{3.29}$$

the master equation (3.27) reduces to

$$\delta \dot{p}_i = -\sum_j L_{ij} \delta p_j + \xi_i(t). \tag{3.30}$$

Being $\xi_i(t)$ a white noise, the equation above describes a multidimensional *Ornstein-Uhlenbeck process* [53]. Writing 3.30 in the eigenbasis of L

$$\delta \vec{p} = \sum_\lambda \delta p_\lambda \vec{v}_\lambda \quad \text{and} \quad \vec{\xi} = \sum_\lambda \xi_\lambda \vec{v}_\lambda$$

we get

$$\begin{aligned} &\delta \dot{p}_\lambda = -\lambda \delta p_\lambda + \xi_\lambda && \text{for} \quad \lambda \neq 0 \\ &\delta \dot{p}_0 = \xi_0 = 0 && \text{for} \quad \lambda = \lambda_0 = 0 \end{aligned}$$

hence, since the noise ξ has no component in the kernel of the matrix L, also the fluctuations δp_i belong to the invariant subspace 3.17, like stated previously. If we had $\xi_0 \neq 0$, the time evolution of δp_0 would be given by a *Wiener process* [19]: thus we would not have a bounded stationary state.

The formal solution of 3.30 can be written in the form

$$\delta p_i(t) = \int_0^t e^{-L_{ij}(t-u)} \xi_j(u) du \tag{3.31}$$

where we want to perform the limit $t \to \infty$. To study the causality relations among the nodes we need to compute the stationary covariance matrix restricted to the subspace (3.17), where L is invertible (fixing $t < s$)

$$\begin{aligned} E[\delta p_i(t) \delta p_j(s)] &= \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} C_{kh} \delta(u-u') \\ &= \int_0^t dt e^{-L_{ik}(t-u)} e^{-L^T_{kj}(t-u)} \end{aligned} \tag{3.32}$$

and taking the limit $t \to \infty$ we get

$$E[\delta p_i \delta p_j] := \lim_{t \to \infty} E[\delta p_i(t) \delta p_j(t)] = \int_0^\infty dt e^{-L_{ik}t} e^{-L^T_{kj}t}. \tag{3.33}$$

Using Dirac notation, we decompose L is its eigenbasis

$$\hat{L} = \sum_{\lambda \neq 0} |v^\lambda\rangle \, \lambda \, \langle v^\lambda|$$

and correspondingly

$$\hat{L}^T = \sum_{\lambda \neq 0} |\bar{v}^\lambda\rangle \, \lambda \, \langle \bar{v}^\lambda|$$

where $\bar{v}_\lambda$ denotes the dual base with respect the eigenvector base v_λ. Thus we have that

$$e^{\hat{L}} = \sum_{\lambda \neq 0} |v^\lambda\rangle \, e^\lambda \, \langle v^\lambda|$$

and equation 3.33 becomes

$$\begin{aligned} E[\delta p^2] &= \sum_{\lambda,\mu \neq 0} |v^\lambda\rangle \, \langle v^\lambda | \bar{v}^\mu\rangle \, \langle \bar{v}^\mu| \int_0^\infty e^{-(\lambda+\mu)t} \\ &= \sum_{\lambda,\mu \neq 0} \frac{1}{\lambda+\mu} |v^\lambda\rangle \, \langle v^\lambda | \bar{v}^\mu\rangle \, \langle \bar{v}^\mu| \, . \end{aligned}$$

In the general case $\langle v^\lambda | \bar{v}^\mu\rangle = G^{\lambda\mu}$ is the inverse of the metric matrix of the eigenvector base, whereas in the case of a symmetric matrix (undirected network) we have that $\langle v^\lambda | \bar{v}^\mu\rangle = \delta_{\lambda\mu}$ and the expression reduces to

$$E[\delta p_i \delta p_j] = \sum_{\lambda \neq 0} \frac{1}{2\lambda} |v^\lambda\rangle_{i\,j} \langle v^\lambda|. \tag{3.34}$$

Hence, in the symmetric case, the covariance between two nodes i and j increases if the eigenvector v_λ do not distinguish between the two nodes (i.e. it has the same sign and similar values on both of them), whereas we have a negative contribution if the two eigenvectors distinguish the nodes. This result is similar to the idea of spectral clustering explained in section 2.2, with the physical intuition of a spring-mass system of section 1.6. Lower eigenvalues, corresponding to less energetic oscillations of the network, have components with the same sign on large sections graphs, corresponding to the large scale structure of the network. Higher eigenvalues corresponds to energetic oscillations, that are able to move in opposite directions (creating a "dipole") also nodes connected by strong edges. Because of the factor λ^{-1}, higher eigenvectors contribute less to the covariance matrix, i.e. it is largely determined by small eigenvalues, emphasizing the mesoscopic scale of the network. The correlation matrix reads

$$C[\delta p_i \delta p_j] = \frac{E[\delta p_i \delta p_j]}{\sqrt{E[\delta p_i^2] E[\delta p_j^2]}}$$

and it thus defines the linear dependence among the nodes, i.e. how much the knowledge of one node allows to forecast the behavior of another. We remark that whatever multiplicative constant in $\vec{\xi}$ would have canceled out in the correlation matrix, playing no role in finding the dependencies among nodes.

Before explaining of we perform a partition of a network using 3.33 (as explained in more details in chapter 4), we turn to a different process through which we can compute correlations among the nodes states.

3.5 Laplacian perturbation

The perturbation to the system studied in the previous section was an external noise, meaning that it was not intrinsic to structure of the network. More importantly, the resulting correlation function depends only on the spectral properties of the Laplacian matrix, and it has no information about the forcing s_i, that influence greatly the dynamics.

The Laplacian matrix represents the links weights that define the fluxes among the nodes. A change in the weights may represent a corresponding change in the graph structure, but also mimics the effect of different dynamical conditions that alter the transportation efficiency (i.e. congestion formation in traffic).

In order to obtain a result that carries more information about the dynamics taking place on the network, a different kind of noise is needed. For this purpose, we now introduce a perturbation to the Laplacian, as follows

$$\dot{\vec{p}} = -(L + \delta L)\vec{p} + \vec{s} \tag{3.35}$$

in which we require that the perturbation preserves the Laplacian properties

$$\sum_i \delta L_{ij} = 0 \quad \text{and} \quad \delta L_{ij} = \delta L_{ji}$$

consequently in δL_{ij} there are $n(n-1)/2$ independent random variables. Recalling that the Laplacian is defined as $L_{ij} = \delta_{ij} d_j - w_{ij}$, the perturbation translates directly in fluctuations of the single weights, that we call δw_{ij}. Thus we can write

$$\delta L_{ij}(t) = \begin{cases} w_{ij}\delta w_{ij}(t) & i \neq j \\ -\sum_{k \neq i} w_{ki}\delta w_{ki}(t) & i = j \end{cases} \tag{3.36}$$

or more compactly

$$\delta L_{ij}(t) = \left[w_{ij}\delta w_{ij}(t) - \delta_{ij} \sum_k w_{ki}\delta w_{ki}(t) \right].$$

Notice that each noise variable δL_{ij} is proportional to the weights δw_{ij}, so that there are no fluctuations to the entries of L_{ij} that do not correspond to any edge in the original network. We further require that the average and covariance of the variable weights δw_{ij} satisfy

$$E[\delta w_{ij}(t)] = 0 \quad \forall t$$
$$E[\delta w_{ij}(t)\delta w_{kl}(t')] = (\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk})\delta(t - t')$$

where $\delta(t - t')$ is a Dirac delta. These properties make the Laplacian fluctuations δL a *white noise*

$$E[\delta L_{kl}(t)\delta L_{hm}(t')] = C_{klhm}\delta(t - t') \tag{3.37}$$

where C_{klhm} is the correlation tensor among the entries of δL_{kl}: it takes care of its Laplacian structure and contains the weights w_{ij}.

Perturbing directly the Laplacian, we see from the equation 3.35 that we get a noise that is coupled with the system state $\vec{p}$. Thus it is a *multiplicative noise* which is more difficult to treat analytically. In order to solve the equation above, it would be necessary to use results from *random matrix theory*[52]. These tools are necessary, for instance, in case one wants to obtain the properties of the spectrum of $L + \delta L$. Since we are just interested in 3.37 to compute $E[\delta p_i \delta p_j]$, we can proceed in a more straightforward way as follows.

Expanding perturbatively the state variable $\vec{p}$, the equation above can be linearized, turning the multiplicative noise into an additive noise, that is easier to deal with. Thus, analogously to the previous section, we perform a perturbation expansion near the stationary state $\vec{p^s}$

$$\vec{p} = \vec{p^s} + \delta\vec{p}$$

and recalling that $L\vec{p^s} = \vec{s}$ we get

$$\delta\dot{p}_i = -L_{ij}\delta p_j - \delta L_{ij}(p^s_j + \delta p_j). \tag{3.38}$$

that is a master equation for the perturbations. Neglecting the higher order term $O(\delta L\delta\vec{p})$ we get

$$\delta\dot{p}_i \approx -L_{ij}\delta p_j - \delta L_{ij}p^s_j. \tag{3.39}$$

that, since δL is a white noise, is again a multidimensional Ornstein-Uhlenbeck process. The justification of the approximation 3.39 will be gievn in the following sections. Since now the perturbation of the Laplacian acts on the system variable as an additive noise, similarly to the external noise of the previous section

$$\xi_i \to -\delta L_{ij}p^s_j. \tag{3.40}$$

analogously to 3.31, equation 3.39 solution is

$$\delta p_i(t) = -\int_0^t e^{-L_{ik}(t-u)}\delta L_{kl}(u)p^s_l du. \tag{3.41}$$

To compute the covariance, we proceed as follows. Fixing $t < s$ and making use of 3.37 we get

$$\begin{aligned} E[\delta p_i(t)\delta p_j(s)] &= \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(t-u')} E\left[\delta L_{kl}(u)\delta L_{hm}(u')\right] p_l^s p_m^s \\ &= \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(t-u')} C_{klhm}\delta(u-u') p_l^s p_m^s \\ &= \int_0^t du e^{-L_{ik}(t-u)} e^{-L_{jh}(t-u)} C_{klhm} p_l^s p_m^s \end{aligned}$$

and taking the limit for $t \to \infty$ we write

$$E[\delta p_i \delta p_j] := \lim_{t\to\infty} E[\delta p_i(t)\delta p_j(t)] = \int_0^\infty e^{-L_{ik}t} e^{-L_{jh}t} C_{klhm} p_l^s p_m^s dt. \tag{3.42}$$

To compute 3.42, we need the contraction of the tensor C_{klhm} with the stationary solution

$$C_{kh} := C_{klhm} p_l^s p_m^s = E[\delta L_{kl}(t)\delta L_{hm}(t)] p_l^s p_m^s \tag{3.43}$$

that is computed explicitly considering each entry of the Laplacian perturbation 3.36 as follows. If $k \neq h$, we have that

$$C_{kh} = \begin{cases} -w_{kh}^2 p_k^2 & \text{for } l = m = k, \\ -w_{kh}^2 p_h^2 & \text{for } l = m = h, \\ w_{kh}^2 p_k p_h & \text{for } (l = h \text{ and } m = k) \text{ or } (l = k \text{ and } m = h), \\ 0 & \text{otherwise} \end{cases}$$

and summing we get to $w_{kh}^2(2p_k p_h - p_k^2 - p_h^2) = -[w_{kh}(p_k - p_h)]^2$. Considering the diagonal elements $k = h$ of 3.36, we have

$$C_{kk} = \begin{cases} \sum_{i\neq k}(w_{ki}p_i)^2 & \text{for } l = m \neq k, \\ p_k^2 \sum_{i\neq k} w_{ki}^2 & \text{for } l = m = k, \\ -p_k \sum_{i\neq k} w_{ki}^2 p_i & (l = k \text{ and } m \neq k) \text{ or } (m = k \text{ and } l \neq k), \\ 0 & \text{otherwise} \end{cases}$$

summing we get to $\sum_{i\neq k} w_{ki}^2(p_k^2 - 2p_k p_i + p_i^2) = \sum_i [w_{ki}(p_k - p_i)]^2$. Thus the whole tensor is given by the following expression

$$C_{kh} = \left\{ \begin{array}{ll} -[w_{kh}(p_k - p_h)]^2 & \text{for} \quad i \neq j \\ \sum_i [w_{ki}(p_k - p_i)]^2 & \text{for} \quad i = j \end{array} \right. \tag{3.44}$$

that has the Laplacian structure

$$\sum_k C_{hk} = 0 \quad \text{and} \quad C_{hk} = C_{hk} \tag{3.45}$$

as it could be noted directly from its definition 3.43, being the contraction of a tensor with the Laplacian properties and a symmetric tensors. We remark that the terms in 3.44 are the fluxes 3.16 through the edges. Containing terms that are the differences between the stationary solution on adjacent nodes, the tensor C_{kh} holds information about the smoothness variations of the signal throughout the network.

To compute the covariance of the fluctuations

$$E[\delta p_i \delta p_j] = \int_0^\infty e^{-L_{ik}t} e^{-L_{jh}t} C_{kh} dt.$$

like in section 3.4, it is useful to expand the Laplacian in its eigenbasis

$$\hat{L} = \sum_{\lambda \neq 0} |v^\lambda\rangle \, \lambda \, \langle v^\lambda| \quad \hat{L}^T = \sum_{\lambda \neq 0} |\bar{v}^\lambda\rangle \, \lambda \, \langle \bar{v}^\lambda|$$

and therefore we get

$$E[\delta p_i \delta p_j] = \int_0^\infty \sum_{\lambda,\mu \neq 0} |v^\lambda\rangle_i \, e^{-\lambda t} \, {}_k\langle v^\lambda | C_{kh} | \bar{v}^\mu \rangle_h e^{-\mu t} \, {}_j\langle \bar{v}^\mu| \tag{3.46}$$

where we can solve directly the time-dependent part with

$$\int_0^\infty e^{-(\lambda+\mu)t} = \frac{1}{\lambda + \mu}$$

getting to the final expression

$$E[\delta p_i \delta p_j] = \sum_{\lambda,\mu \neq 0} \frac{1}{\lambda + \mu} |v^\lambda\rangle_i \left[\sum_{k,h} {}_k\langle v^\lambda | C_{kh} | v^\mu \rangle_h \right] {}_j\langle v^\mu| \tag{3.47}$$

where now we have written explicitly the sum over the vector indices k, h and we have exploited the symmetry of the Laplacian. The term in the squared brackets

$$S^{\lambda\mu} := \sum_{k,h} {}_k\langle v^\lambda | C_{kh} | v^\mu \rangle_h$$

is the scalar product between the eigenvector v_λ and v_μ, with a metric matrix given by C_{kh}. Note that if the fluxes are all equal

$$w_{ij}(p_i - p_j) = \frac{1}{\sqrt{N}} \quad \forall (i,j) \tag{3.48}$$

and the network is complete (there is an edge between all couples of nodes (i,j)) we have that

$$C_{kh} = \begin{cases} \sum_i [w_{ki}(p_k - p_i)]^2 = 1 - \frac{1}{N} & \text{if } k = h \\ -[w_{kh}(p_k - p_h)]^2 = -\frac{1}{N} & \text{if } k \neq h \end{cases} \tag{3.49}$$

thus we recover the metric matrix

$$C_{kh} = \left(\delta_{kh} - \frac{1_{kh}}{N} \right) \tag{3.50}$$

and consequently

$$\begin{aligned} S^{\lambda\mu} &:= \sum_{k,h} {}_k\langle v^\lambda | C_{kh} | v^\mu \rangle_h \\ &= \sum_{k,h} {}_k\langle v^\lambda | \delta_{kh} | v^\mu \rangle_h - \frac{1}{n} \sum_{k,h} {}_k\langle v^\lambda | 1_{kh} | v^\mu \rangle_h \\ &= \delta^{\lambda\mu} \end{aligned} \tag{3.51}$$

since the eigenvectors are orthogonal and their components sum to 1 (we are not considering the kernel here). Thus from 3.47 we recover the result of the previous section

$$E(\delta p_i \delta p_j) = \sum_{\lambda \neq 0} \frac{1}{2\lambda} \, |v^\lambda\rangle_i \, {}_j\langle v^\lambda|.$$

This is reasonable, since, as already noted, having linearized the multiplicative Laplacian noise, we get an additive white noise like in the first method. The main difference is that using the entries of L as white noise (i.e. the edges) and not the nodes like before, we are now taking care of both the topology of the network and its dynamics. In the previous method the structure of the network was present only implicitly through the spectrum of L, not explicitly like now, and the dynamics (even if stationary) was totally absent.

3.6 Higher order terms

In the previous section, we have studied the correlations of the state variable $\vec{p}$ induced by the Laplacian perturbation δL, taken to be a white noise. Neglecting higher order terms $\delta L \delta p$ in the master equation 3.38, we have obtained 3.39, that corresponds to a Ornstein-Uhlenbeck process. This way, we were able to compute the covariance matrix of the fluctuations 3.47.

However, we must justify the approximation 3.39. The whole expression 3.38, that we rewrite for reference, is

$$\dot{\delta p}_i = -L_{ij} \delta p_j - \delta L_{ij} (p_j^s + \delta p_j). \tag{3.52}$$

and does not describe anymore an Ornstein-Uhlenbeck process. Thus we do not know a priori if the variance of the fluctuations δp is finite. The formal solution is obtained treating the term $-\delta L_{ij}(p_j^s + \delta p_j)$ like a forcing to an homogeneous equation, getting to the following

$$\begin{aligned} \delta p_i(t) = & - \int_0^t e^{-L_{ij}(t-u)} \delta L_{jk}(u) \delta p_k(u) du \\ & - \int_0^t e^{-L_{ij}(t-u)} \delta L_{jk}(u) p_k^s du \end{aligned} \tag{3.53}$$

that defines the perturbations δp_i around the stationary solution iteratively through the first integral, whereas the second part represents the stationary part of the process, already studied in the previous section (equation 3.41).

Since δL is a white noise, its variance diverges, like in 3.37. Being unbounded, it makes the terms in L_{ij} (i.e. the weights w_{ij}) to become negative. This nonphysical situation induces severe problems for the diffusion process studied. In fact, the Laplacian eigenvalues might become negative, corresponding to unstable directions and turning the exponential decay 3.20 that leads to the stationary solution 3.25 into an exponential increase. Consequently, in this scenario the fluctuations δp_i grow with time, higher order term in 3.52 cannot be neglected, and the perturbation expansion would break down.

Thus, it is necessary to consider a process that describes limited perturbations δL_{ij}. However, if the chosen stochastic process underneath δL_{ij} is limited, it cannot be delta-correlated, i.e. we cannot choose δL_{ij} such that

$$E[\delta L_{ij}(t) \delta L_{kh}(t')] = \delta_{tt'} \tag{3.54}$$

otherwise the effect on the dynamics would be null and we wouldn't be able to measure any correlation of δp_i. Thus we consider δL_{ij} to follow the already-mentioned Ornstein-Uhlenbeck process

$$\frac{d}{dt} \delta L_{ij} = \frac{1}{\varepsilon} \delta L_{ij} + \frac{\sigma_{ijkh}}{\varepsilon^{1/2}} \eta_{kh}(t) \tag{3.55}$$

where η_{kh} contains $n(n-1)/2$ independent white noise variables ($\eta_{kh} = \eta_{hk}$ and $\eta_{kk} = 0$), and σ_{ijkh} contains information about the weights of the network (that give the variance of the process). Their product is given by

$$\sigma_{ijkh} \eta_{kh} = \sigma_{ijkh} \eta_{hk} = \left[w_{ij} \eta_{ij} - \delta_{ij} \sum_k w_{ki} \eta_{ki} \right]$$

and has a Laplacian structure. The perturbation has the following form

$$\delta L_{ij}(t) = \delta w_{ij}(t) - \delta_{ij} \sum_k \delta w_{ki}(t) \tag{3.56}$$

where $\delta w_{ij}(t)$ are stochastic variables whose statistical properties are now (for $i \neq j$ and $k \neq h$)

$$E[\delta w_{ij}(t)] = 0 \quad \forall t$$
$$E[\delta w_{ij}(t)\delta w_{kh}(t')] = (\delta_{ik}\delta_{jh} + \delta_{ih}\delta_{jk})e^{-(t-t')/\varepsilon}.$$

We remark that now δw_{ij} have bounded values to satisfy the constraint $w_{ij} + \delta w_{ij}(t) \geq 0$. Consequently, the Laplacian fluctuations δL have zero average

$$E[\delta L_{ij}(t)] = 0 \quad \forall t$$

and, given the covariance of a Ornstein-Uhlenbeck process [53], it correlates like (for sufficiently large t)

$$E[\delta L_{ij}(t)\delta L_{kh}(t')] = \frac{C_{ijkh}}{2}e^{-(t-t')/\varepsilon} \tag{3.57}$$

where now $C_{ijkh} := \sigma_{ijlm}\sigma^T_{mlkh}$ is the correlation tensor among the entries of δL_{kl}: it takes care of its Laplacian structure and the weights and, as we will see in section 3.7, it corresponds to the same tensor 3.37.

3.6.1 Contraction condition

The existence of a stationary solution to 3.53 depends on the statistical properties of the stochastic process δL. The stationary solution to 3.53 exists if the integral operator I in the first part

$$f_i(t) = (If_k)(t) := \int_0^t e^{-L_{ij}(t-u)}\delta L_{jk}(u)f_k(u)du \tag{3.58}$$

satisfies a contraction principle, i.e. if we define

$$\|I\|_{\text{op}}(t) = \sup\left\{\frac{\|(If)(t)\|}{\|f(t)\|}, f \neq 0\right\}$$
$$\|f\|(t) = \sup_{u \leq t}\{|f(u)|\}$$

the contraction condition reads

$$\|I\|_{\text{op}}(t) < 1 \quad \forall t.$$

Since the used norm for real-valued functions is the sup norm (i.e f belongs to L^∞), the contraction condition implies that the sup of the image through I of the function in the interval $[0,t]$ (that given the definition of the operator, is the value of the function in t) is less than the sup of the function in the same interval $[0,t]$. Thus we need to evaluate

$$\begin{aligned} |f_i(t)| &= \left|\int_0^t e^{-L_{ij}(t-u)}\delta L_{jk}(u)f_k(u)du\right| \\ &\leq \int_0^t \left|e^{-L_{ij}(t-u)}\delta L_{jk}(u)f_k(u)\right| du \quad \forall i = 1,\dots,n-1 \end{aligned} \tag{3.59}$$

where $f(t)$ is a regular test function. It is convenient use the eigenvector base $\{v_\lambda\}_{\lambda=1,\dots,n-1}$ of the Laplacian matrix L

$$\Lambda_{\lambda\mu} := U^T_{\lambda i} L_{ij} U_{j\mu}$$
$$\delta L_{\lambda\mu} := U^T_{\lambda i} \delta L_{ij} U_{j\mu}$$
$$f_\lambda := U^T_{\lambda i} f_i$$

where U is the matrix having as columns the eigenvectors of L. Hence we have

$$\begin{aligned} |f_\lambda(t)| &\leq \int_0^t e^{-\lambda(t-u)} \, |\delta L_{\lambda\mu}(u) f_\mu(u)| \, du \\ &\leq \int_0^t e^{-\lambda(t-u))} \|\delta L(u)\|_\lambda \|f(u)\| \quad \forall \lambda = 1, \dots, n-1 \end{aligned}$$

with

$$\|f(u)\| = \sqrt{\sum_\mu f^2_\mu(u)}$$
$$\|\delta L(u)\|_\lambda = \sqrt{\sum_\mu \delta L^2_{\lambda\mu}(u)}.$$

where we have used the Cauchy-Schwarz inequality and the euclidean norm, both for the function f and the λ^{th} row of δL. Noting that δL is symmetric and that we are in the eigenbasis of L (thus the eigenvectors are of the form $v_\lambda = (0, \dots, 1, \dots, 0)$), we have that

$$\|\delta L(u)\|_\lambda = \sqrt{\sum_\mu \delta L^2_{\lambda\mu}(u)} = \|\delta L(u) v_\lambda\| \quad \forall \lambda. \tag{3.60}$$

To take the norms outside of the integral, we maximize with respect to the time variable, defining

$$\|f\|(t) = \sup_{u \leq t} \|f(u)\| = \sup_{u \leq t} \sqrt{\sum_\mu f^2_\mu(u)}$$

and correspondingly

$$\|\delta L\|_\lambda(t) = \sup_{u \leq t} \|\delta L(u)\|_\lambda = \sup_{u \leq t} \|\delta L(u) v_\lambda\| \tag{3.61}$$

i.e. $\|\delta L\|_\lambda(t)$ measures the maximum *deformation* that δL causes on the eigenvector v_λ of L in the time interval $[0, t]$. Then we can evaluate the time integral in a straightforward manner

$$\begin{aligned} |f_\lambda(t)| &\leq \|\delta L\|_\lambda(t) \|f\|(t) \int_0^t e^{-\lambda(t-u)} du \\ &= \frac{\|\delta L\|_\lambda(t)}{\lambda} \|f\|(t) \end{aligned}$$

for sufficiently large t. Defining the norm of the Laplacian perturbation as

$$\|\delta L\|(t) := \sup_{\lambda} \|\delta L\|_{\lambda}(t)$$

that of course implies

$$\|\delta L\|_{\lambda}(t) \leq \|\delta L\|(t) \quad \forall \lambda$$

we can take the euclidean norm of both sides and write the inequality above as

$$\|f(t)\| \leq \sqrt{\sum_{\lambda} \frac{\|\delta L\|_{\lambda}^{2}(t)}{\lambda^2}} \|f\|(t) \leq R \|\delta L\|(t) \|f\|(t) \tag{3.62}$$

with R defined as

$$R := \sqrt{\sum_{\lambda} \frac{1}{\lambda^2}}. \tag{3.63}$$

Finally,the contraction conditions reads

$$R\|\delta L\|(t) < 1. \tag{3.64}$$

If this condition is satisfied, the operator 3.58 behaves like a contraction, namely the norm $\|f(t)\|$ of the function f evaluated at time t is less than $\|f\|(t)$, the sup norm of f in the interval $[0, t]$. Note that 3.62 can be extended $\forall t$, so that we can study the asymptotic solution of (3.52).

The contraction condition 3.64 depends on the spectral properties of the network through the average value R and the perturbation strength $\|\delta L\|$. We remark that the condition $\|\delta L\|(t) < R^{-1}$ is meant in the basis where L is diagonal: thus, as already noted, it translates directly in a condition on the *deformation* that the perturbation can make to the L eigenvectors to have contraction in 3.58. This condition in fact could not be imposed if δL was a white noise, like in the previous section, since in that case its elements have infinite variance.

How much strict the condition 3.64 is on δL, it depends on the factor R. Since R is related to the sum of the Laplacian eigenvalues, we are interested in the limit $N \to \infty$, in which the number of elements in the sum grows, and also $\lambda_{\max}$ grows accordingly. Depending on how fast λ_{max} grows with N, we have that R might be given by a convergent or divergent series. In the former case the condition on $\|\delta L\|$ does not depend on the network dimension. In the latter case it will, and in particular with N larger it will get stricter (i.e. the less $\|\delta L\|(t)$ will need to be to have a contraction in 3.53).

From [13], under general conditions on the probability distribution the Laplacian elements are drawn from, among which it is required to have finite variance, we know that

$$\lim_{N\to\infty} \inf \frac{\lambda_{\max}\left(L^{(N)}\right)}{\sqrt{N \log N}} = \sqrt{2} \quad \text{and} \quad \lim_{N\to\infty} \sup \frac{\lambda_{\max}\left(L^{(N)}\right)}{\sqrt{N \log N}} = 2 \tag{3.65}$$

thus as the network gets bigger

$$\lambda_{\max} \overset{N\to\infty}{\sim} \sqrt{N\log N}.$$

Given the definition of R, we thus need to study the convergence of the series $\sum_N \frac{1}{N\log N}$. With the integral test, it is easy to see that

$$R \sim \sqrt{\sum_N \frac{1}{N\log N}} \sim \sqrt{\log(\log N)}$$

hence the series diverges for large networks, even if very weakly. Consequently, the greater is the network we are considering and the smaller have to be the fluctuations of the Laplacian in order to have a contraction condition in 3.53.

Since the dependence on N is very weak, it would be enough just a slight bigger dependence than $\frac{1}{N\log N}$ to have convergence, but we are not aware of any result stricter or different than the one found in [13].

One can argue that for $N \to \infty$ we might have that $\lambda_F \to 0$ (i.e. the network disconnects), leading to a divergent series from below. Thus above we have implicitly considered ensembles of random networks that remains strongly connected as $N \to \infty$. For example in the case of Erdős–Rényi networks imposing that the link probability p satisfies $p > \frac{\log N}{N}$ [14], it can be shown that there is a giant component spanning the whole network for $N \to \infty$, guaranteeing that the Fiedler value does not go to zero.

We remark that since we are overestimating $\|f(t)\|$ in 3.62, actually it might not depend on N. However, since the dependence on N found is very weak, it is of little practical interest. Besides, lacking a different theoretical result, a computer program would find whether there is a dependence on N or not only simulating very big networks, out of the present scope.

Finally, we mention that there are several results in the literature connecting sum of powers of the Laplacian eigenvalues to specific properties of the network [7]. However, we are not aware of any known physical meaning concerning the sum of the inverse squared eigenvalues like in 3.63.

3.6.2 Relative fluctuations

Once shown that 3.53 tends to the stationary solution, i.e. that $R\|\delta L\|(t) < 1$, we might be interested in checking the effects of this condition on the stationary solution itself

$$\delta p_\lambda(t) = -\int_0^t e^{-L_{\lambda\mu}(t-u)}\delta L_{\mu\sigma}(u)p^s_\sigma du$$

written in the eigenbasis of L. Thus evaluating the norm in a similar fashion to the inequalities above, we get

$$
\begin{aligned}
|\delta p_\lambda(t)| &\leq \int_0^t |e^{-L_{\lambda\mu}(t-u)}\delta L_{\mu\sigma}(u)p^s_\sigma| du \\
&\leq \int_0^t \|e^{-L_{\lambda\mu}(t-u)}\| \|\delta L_{\mu\sigma}(u)\| \|p^s_\sigma\| du \\
&\leq \|p^s_\sigma\| \int_0^t e^{-\lambda_\lambda(t-u)} \| \|\delta L(u)\|_\mu du \\
&\leq \|p^s_\sigma\| \|\delta L\|_\mu(t) \int_0^t e^{-\lambda_\lambda(t-u)} du \\
&= \|p^s\| \|\delta L\|(t) \frac{1-e^{-\lambda_\lambda t}}{\lambda_\lambda} du
\end{aligned}
$$

where in the last line we have the norm of the vector p^s and the corresponding matrix norm of δL. In the stationary limit $t \to \infty$ and taking the euclidean norm with respect to the λ index we get

$$
\sqrt{\sum_\lambda \delta p^2_\lambda(t)} \leq \sqrt{\sum_\lambda \frac{1}{\lambda^2_\lambda}} \|p^s\| \|\delta L\|(t)
$$

hence

$$
\frac{\|\delta \vec{p}(t)\|}{\|\vec{p^s}\|} \leq R \|\delta L\|(t) < 1. \tag{3.66}
$$

Recalling that $\vec{p^s} = E[\vec{p}(t)]$ for large t, we get that the contraction condition implies the relative fluctuations with respect to the stationary solution are smaller than 1. Note that the norm of p is taken at time t, whereas the norm of δL is the supremum over its history.

3.7 Rescaled process

In the previous section we have set a condition 3.64 for the convergence of our process 3.53 to its stationary solution. We chose the Ornstein-Uhlenbeck process 3.55 to describe the variables δL_{ij}, since if δL_{ij} is a white noise, having infinite variance, it cannot satisfy that condition. Nevertheless, we now want to show that under certain conditions, in the limit $\varepsilon \to 0$ it is possible both to recover a white noise limit from the Ornstein-Uhlenbeck process 3.55 and to show that higher order terms in 3.38 remain negligible, justifying the approximation 3.39 made in section 3.5, even with a white noise.

We remark that simply taking the limit $\varepsilon \to 0$ in the process 3.55, we get

$$
E[\delta L_{ij}(t)\delta L_{kh}(t')] = \frac{C_{ijkh}}{2} e^{-(t-t')/\varepsilon} \underset{\varepsilon \to 0}{\longrightarrow} \frac{C_{ijkh}}{2} \delta_{t,t'}
$$

and thus, being a bounded and delta-correlated stochastic variable, the process

$$\delta p_i(t) = \int_0^t e^{-L_{ij}(t-u)} \delta L_{jk}(u) p_k^s du$$

has zero variance, so that $\delta\vec{p}$ is equivalent to the null process. Starting from the master equation 3.35, that we rewrite for convenience

$$\dot{\vec{p}} = -(L + \delta L)\,\vec{p} + \vec{s} \tag{3.67}$$

we perform the following trick

$$L + \varepsilon^{1/2} \frac{\delta L(t)}{\varepsilon^{1/2}} \tag{3.68}$$

and defining

$$\delta\hat{L} := \frac{\delta L}{\varepsilon^{1/2}} \quad \text{with} \quad \varepsilon \to 0.$$

we rewrite the master equation as

$$\dot{\vec{p}} = -\left(L + \varepsilon^{1/2}\delta\hat{L}\right)\vec{p} + \vec{s}. \tag{3.69}$$

Correspondingly, the master equation for the fluctuations of the network state now becomes

$$\dot{\vec{\delta p}} = -L\vec{\delta p} + \varepsilon^{1/2}\delta\hat{L}\left(\vec{p^s} + \vec{\delta p}\right). \tag{3.70}$$

The idea behind the scaling is to perform a perturbative expansion and a white noise limit at the same time, taking $\varepsilon \to 0$. We are thus now considering a rescaled version of the Ornstein-Uhlenbeck process 3.55. In fact, the corresponding master equation now becomes

$$\frac{d}{dt}\delta\hat{L}_{ij} = \frac{1}{\varepsilon}\delta\hat{L}_{ij} + \frac{\sigma_{ijkh}}{\varepsilon}\eta_{kh}(t) \tag{3.71}$$

where the parameters of the stochastic variable $\delta\hat{L}_{ij}$ and of the white noise have now the same strength, of $1/\varepsilon$. Thus the statistical properties read

$$E[\delta\hat{L}_{ij}(t)] = 0 \quad \forall t$$
$$E[\delta\hat{L}_{ij}(t)\delta\hat{L}_{kh}(t')] = \frac{C_{ijkh}}{2\varepsilon}e^{-(t-t')/\varepsilon} \underset{\varepsilon\to 0}{\longrightarrow} C_{ijkh}\delta(t-t')$$

where $\delta(t-t')$ is a Dirac's delta. Therefore the variables $\delta\hat{L}$ for $\varepsilon \to 0$ tends to infinity, with the statistical properties of a white noise. Thus dividing the master equation 3.70 with $\varepsilon^{1/2}$ we have that

$$\frac{\delta\dot{p}_i}{\varepsilon^{1/2}} = -L_{ij}\frac{\delta p_i}{\varepsilon^{1/2}} - \delta\hat{L}_{ij}\left(p_j^s + \delta p_j\right)$$

and defining

$$\delta\hat{p} := \lim_{\varepsilon\to 0} \frac{\delta p}{\varepsilon^{1/2}} \tag{3.72}$$

we get to the following

$$\dot{\delta\hat{p}_i} = -L_{ij}\delta\hat{p}_j - \delta\hat{L}_{ij}\left(p_j^s + \varepsilon^{1/2}\delta\hat{p}_j\right). \tag{3.73}$$

whose formal solution is

$$\begin{aligned}\delta\hat{p}_i(t) =& -\varepsilon^{1/2}\int_0^t e^{-L_{ij}(t-u)}\delta\hat{L}_{jk}(u)\delta\hat{p}_k(u)du \\ &-\int_0^t e^{-L_{ij}(t-u)}\delta\hat{L}_{jk}(u)p_k^s du\end{aligned} \tag{3.74}$$

where $\varepsilon \to 0$ stands both for a *white noise limit* and and a *perturbation parameter* for higher order terms. Our aim is now to prove that in the limit $\varepsilon \to 0$, in which $\hat{L}$ tends to a white noise, the rescaled process satisfies

$$\delta\hat{p}(t) \to -\int_0^t e^{-L_{ij}(t-u)}\delta\hat{L}_{jk}(u)p_k^s du$$

in a weak sense, i.e. the distribution of $\delta\hat{p}(t)$ tends to the one of the stochastic process 3.41, treated in section 3.5. We remark that it is not possible to apply directly the contraction principle 3.64 to the process defined by 3.74, since the elements of $\delta\hat{L}$ becomes infinite in the limit $\varepsilon \to 0$. Thus the convergence cannot be pointwise but only in a distribution sense.

In Appendix A, we find an equation for the second moment of 3.74 in which higher order corrections (given by the first term) vanish for $\varepsilon \to 0$. Thus the second moments of 3.74 and of 3.41 converge. Since for *Wick's theorem* [57] (also called *Isserlis' theorem*), any Gaussian process with zero average is described completely by the second moment, we have that the considered rescaled process converges in distribution to the white noise process 3.41. Therefore for the rescaled process we recover the covariance matrix 3.47

$$E[\delta p_i \delta p_j] = \sum_{\lambda,\mu\neq 0} \frac{1}{\lambda+\mu} |v^\lambda\rangle_i \left[\sum_{k,h} {}_k\langle v^\lambda|C_{kh}|v^\mu\rangle_h\right]_j \langle v^\mu|$$

justifying its validity despite the approximation 3.39.

3.8 Observability

The diffusion equation 3.15 can also be interpreted in terms of *linear time-invariant (LTI) systems*, a general class of systems studied in system analysis and in control theory that produces an output starting from an input in a linear manner. The governing equation of a LTI system is given by

$$\begin{aligned}\dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)\\ \mathbf{y}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)\end{aligned} \tag{3.75}$$

where $\mathbf{x}$ is the system state variable and $\mathbf{u}$ is a set of exogenous inputs. The matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ governs the dynamics of the system. $\mathbf{B} \in \mathbb{R}^{n\times p}$ is the matrix that allow the inputs $\mathbf{u}$ to be applied to the state variable $\mathbf{x}$. In this kind of system we might not be able to determine the whole system state, and this is captured by the matrices $\mathbf{C} \in \mathbb{R}^{q\times n}$ and $\mathbf{D} \in \mathbb{R}^{q\times p}$. The output of the system $\mathbf{y}$ is therefore a linear transformation of the system state $\mathbf{x}$. All the matrices are time constant.

A system like 3.75 is said to be *observable* if for any unknown initial state $\mathbf{x}(0)$, there is a finite time $t_1 > 0$ such that the knowledge of the variables $\mathbf{u}$ and $\mathbf{y}$ over the time interval $[0, t_1]$ is sufficient to determine uniquely the initial state $\mathbf{x}(0)$.

One can determine if a system is observable or not looking at the pair of matrices $\mathbf{A}$ and $\mathbf{C}$. If the *observability* matrix, defined as

$$\mathcal{O} = \begin{bmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{bmatrix}$$

has rank n, the system is observable. Equivalently, if the *observability Gramian*

$$\mathbf{W}_o(t) = \int_0^t e^{\mathbf{A}^T\tau}\mathbf{C}^T\mathbf{C}e^{\mathbf{A}\tau}d\tau$$

is nonsingular for any $t > 0$, the system is observable. [9]

Reducing the network as explained above, from local measurements on sensor nodes we cannot infer the new solution of the non observed nodes. Therefore the system is not observable. Still, we want to be able to understand the location of the link failure using the general idea of section 3.2. In the next chapter we will exploit 3.47 to find an optimal clustering method. so that the network dimensionality is reduced, still keeping a good failure localization efficiency. The proposed clustering method will be compared with more traditional techniques.

Chapter 4

Simulations results

4.1 Problem definition

In the present chapter, we compare the efficiency of different clustering methods under a single-edge failure occurring in the network. With a single-edge failure we mean the sudden decrease of an edge weight, say w_{kh}, such that

$$w_{kh} \longrightarrow w'_{kh} = \alpha w_{kh} \quad \text{with} \quad 0 < \alpha < 1 \tag{4.1}$$

and thus the corresponding change of the Laplacian matrix reads

$$L \longrightarrow L' = L + \Delta L \tag{4.2}$$

with

$$\Delta L_{ij} = \begin{cases} -(1-\alpha)w_{kh} & \text{for } i=j=k \text{ or } i=j=h \\ (1-\alpha)w_{kh} & \text{for } i=k, j=h \text{ or } i=h, j=k. \end{cases} \tag{4.3}$$

Phisically, this implies that the edge failure is meant to be as a sudden decrease in the *conductance* of that link, not as a proper link breaking. Since we are dealing with transport networks, in the former case there is a decrease of the transport capacity of a single link, whereas the latter case would eventually lead to an empty network, if the sources do not compensate explicitly for the lost material. For instance, in urban networks the condition 4.1 means a congestion of a road, whereas for water distribution networks it may signal the obstruction a pipe, instead of a proper leak.

After the failure occurs, the system reaches a new stationary solution following the same equation 3.15, that is

$$\frac{d\vec{p'}}{dt} = -L'\vec{p'} + \vec{s} \tag{4.4}$$

whose transient solution is given by

$$\vec{p'}(t) = \sum_{\lambda' \neq 0} c_{\lambda'} e^{-\lambda' t} \vec{v}_{\lambda'} + \vec{p^{s'}} \tag{4.5}$$

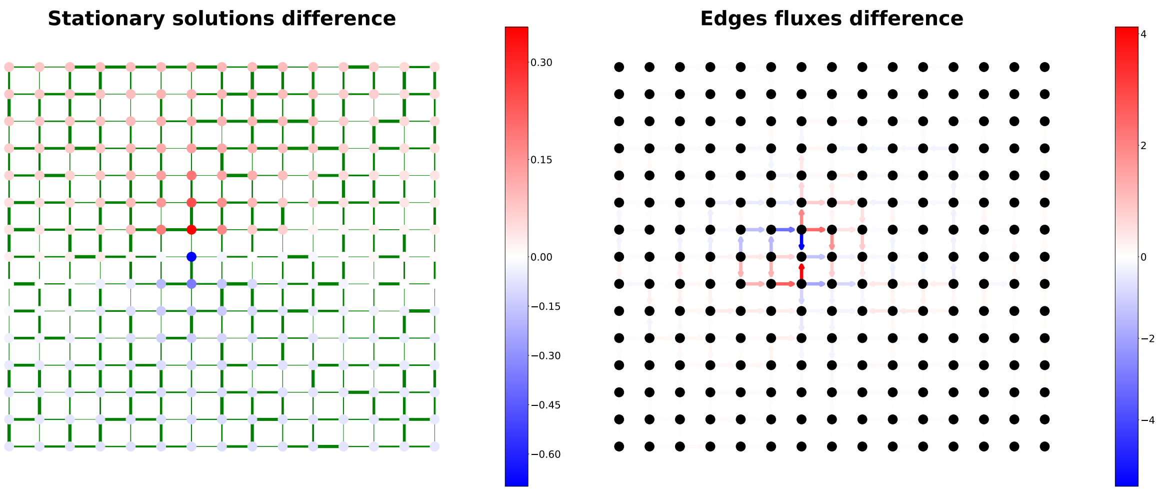


Figure 4.1: Stationary potentials and fluxes differences after a single-edge failure on a 15×15 grid network. The failed edge can be easily localized since it is the edge with the maximum flow drop, corresponding to the nodes with the maximum potential difference. Therefore, the generation of a dipole difference makes the failure localization trivial.

where λ' and $\vec{v}_{\lambda'}$ are respectively the eigenvalues and eigenvectors of the new Laplacian matrix L', and the coefficients $c_{\lambda'}$ are determine by the initial condition

$$\vec{p^s} \equiv \vec{p}(0) = \sum_{\lambda' \neq 0} c_{\lambda'} \vec{v}^{\lambda'} + \vec{p^{s'}}$$

assuming that before the edge failure the system was in its stationary state.

If the whole state $p(\vec{t})$ of the network is observable, the failure identification problem would be trivial. Figure 4.1 shows a colormap of the stationary solution variations $\Delta\vec{p} = \vec{p^{s'}} - \vec{p^s}$ and the edge flux differences $\Delta Q_{ij} = Q'_{ij} - Q_{ij}$ for a single-edge failure in a 15×15 lattice network. After the edge-failure, the flux between the two nodes decreases (negative variation in the plot). Thus, correspondingly the potential of the node located upstream will increase, whereas the potential of the node placed downstream will decrease, as shown on the left plot. Therefore, monitoring the potential for each node of the network, the failure localization would be trivial. One would just need to find the couple of nodes with the maximum potential variations, that should correspond to the edge with the maximum potential or flow drop. Those two nodes, that defined unambiguously the location of the failure, form a *dipole* varitation of the potential. In Appendix B we show the equivalent situation for an Erdős–Rényi network.

Hence, as discussed in chapter 3, the network state $\vec{p}$ is assumed to be observable only on a subset of nodes $S \in \mathcal{N}(\mathcal{G})$, that we call *sensors*. The problem is thus twofold. First, we need to define a clustering method to partition the graph, as explained in chapter 2. Secondly, a criterion has to be chosen in order to pick a single sensor node

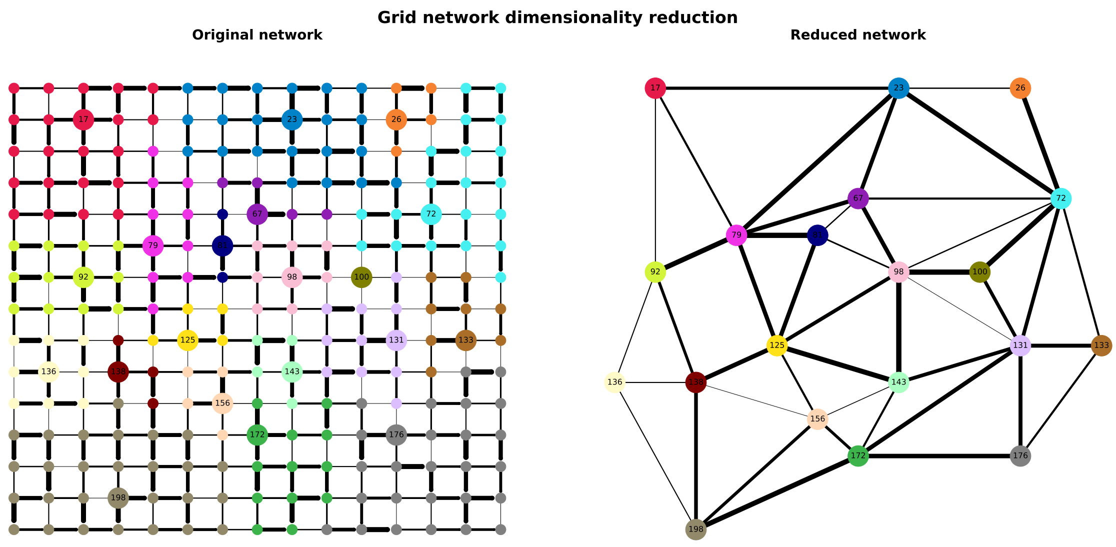


Figure 4.2: Dimensionality reduction of a 15×15 grid network, from the initial 225 nodes to 20 nodes. Each sensor node of the reduced graph represents the nodes belonging to its cluster in the original network. A single edge is set between two sensor nodes if there were edges between the nodes they represent. In the original network, each color represents a cluster and bigger nodes corresponds to the sensors. The lines thickness is proportional to the edge weights.

for each cluster, as representative of the other nodes. The state of the network is then observed only through these sensor nodes. Our aim is to find an optimal partitioning of the network and sensor placement such that the failure identification probability is maximized. One might might be interested in choosing a sensor placement method in order to have a better sensitivity for the failures involving only on a subset of edges. As we will see, the proposed clustering criterion focuses on edges with higher fluxes.

Once a partitioning is established and a sensor nodes is chosen for each cluster, a reduced network is constructed from the original one, following the ideas of section 3.2. Therefore, the sensor nodes (that are the nodes of the coarse-grained graph) are considered to be connected if there were edges between the nodes they represent in the original network. The set of edges crossing two groups of nodes, or clusters in our case, is called *edge-boundary*. The reduced network edges weights are taken to be the sum of the original edges weights belonging to the edge-boundary of the two clusters considered. This quantity is also called *cut size*. Picture 4.2 shows an example of graph reduction for a 15×15 grid network. Picture 4.3 shows the same procedure for a Erdős–Rényi random network with $N = 225$ nodes and average degree $\langle k \rangle = 3.73$.

Observing the signal only through the sensor nodes, the failure localization procedure is slightly different. In particular, we might encounter two different situations. If the

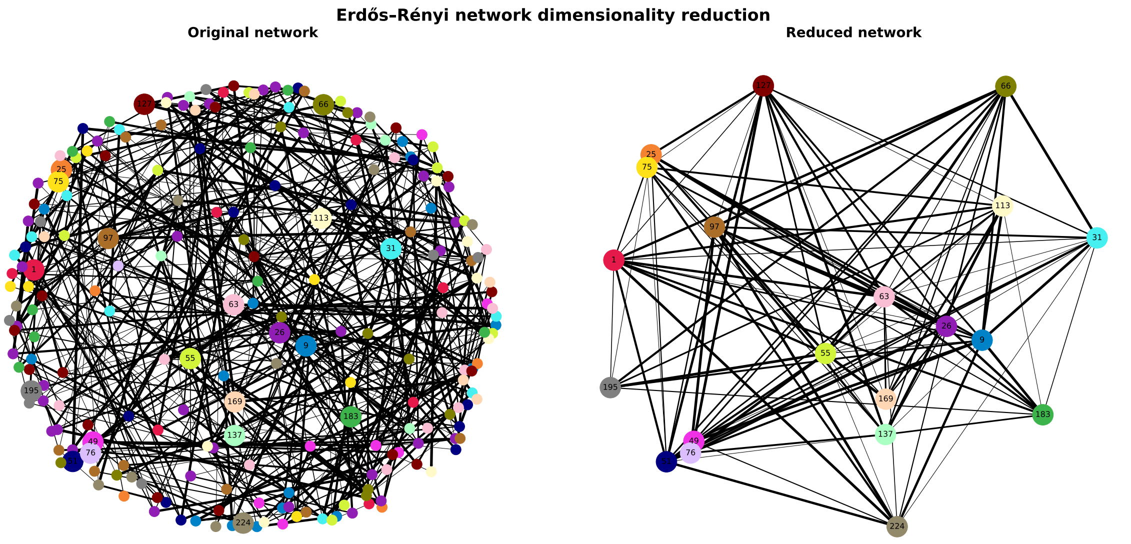


Figure 4.3: Reduced network creation for an Erdős–Rényi network with $N = 225$ and $\langle k \rangle = 3.73$. The number of clusters is 20. The network structure is greatly simplified by the coarse-grained modeling described in the main text.

edge failure happens inside a cluster, we expect to have a strong potential difference corresponding to the sensor node of that cluster, and lower responses from the rest of the network. We call the signal pattern corresponding to this situation *monopole*. However, if the failed edge belongs to the edge boundary of two clusters, we expect to see a situation analogue to figure 4.1, but restricted to the sensor nodes, i.e. we should see a dipole like the one shown in figure 4.4 on the right. In this cases, we are able to spot easily the group of nodes among which there is the failed one, and it is the edge boundary between the clusters corresponding to the observed dipole.

Actually, from this situation it is possible to restrict more the set of candidate failed edges. Knowing the direction of the stationary fluxes for all edges, we might infer which one can lead to the observed dipole and which ones are not compatible with it. In fact, as seen in picture 4.1, the potential difference is positive for the node upstream and negative for the node downstream the failed edge. Therefore, the candidate group of edges must include only those whose flux is outgoing the positive monopole and incoming to the negative monopole. This might help to better localize the failure in some situations. The basic assumption here is that the demand pattern $\vec{s}$ is known and constant over the time needed to identify the failure, so that the flux direction cannot change.

The signal patterns depicted in figure 4.4 are ideal cases. In other situations we might have a more spread potential difference, that involves the whole network, like shown in appendix B, picture B.4. Nevertheless, if there is a net boundary between the regions of sensors with a positive and negative signal difference, it should still be possible to

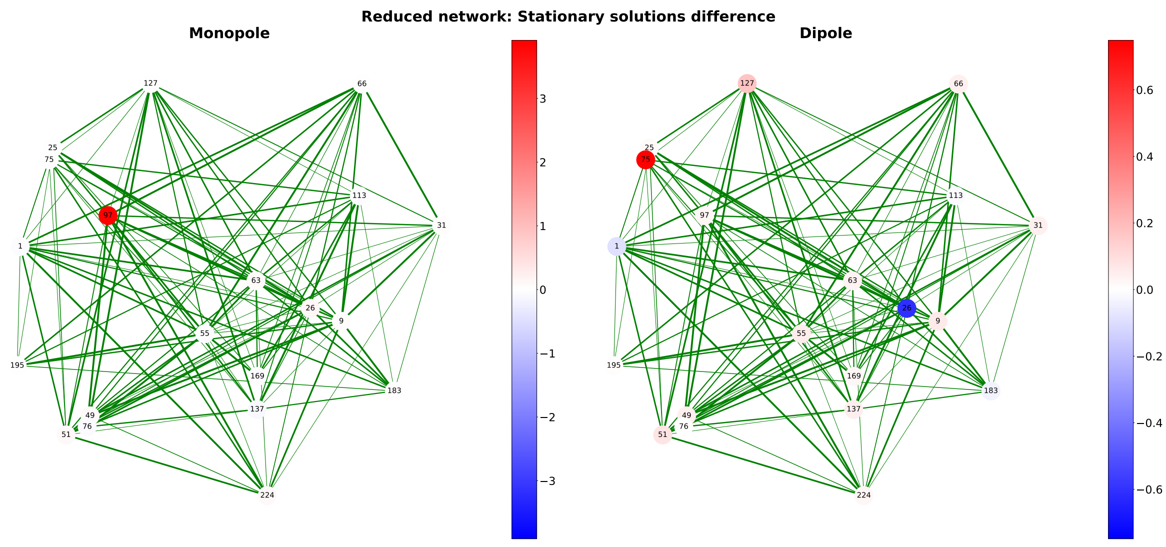


Figure 4.4: Expected behavior for an edge failing inside a single cluster (left) and between two clusters (right). In the former case we observe a monopole, in the latter case a dipole in the signal difference. In both cases, we are able to identify the group of edges the failed one belongs to.

identify the spot in which the failure happened, like explained above.

4.2 Failure identification

The aim of this chapter is to compare the efficiency of different clustering and sensor placement algorithms against the localization of a single-edge failure. Since there is no a priori preferred method, the comparison is performed through a simulation written in Python and available on GitHub [15].

In particular, the failure localization is performed using a function that takes as input a network partitioning, characterized by the clusters and the corresponding sensor choices, and the observed signal difference over the sensor nodes. The idea is to perform the localization for each failed edge and averaging the results, finding the partitioning able to localize the failures with more efficiency.

The candidate group of nodes for each edge failure is picked up exploiting the considerations of the previous section 4.1. In particular, first of all the dipole with the highest potential drop is chosen. Thus, there are four different possibilities. The failed edge might be internal to the cluster with positive potential difference (positive monopole), or to the cluster with negative potential difference (negative monopole). Alternatively, the failed edge belongs to the edge boundary of the positive and negative monopoles. Finally, the failed edge does not belong to the above three categories. A Bayesian [51] reasoning is performed to pick the most probable situation among the first three possibilities. In our case, the Bayes rule reads

$$P\left(C_i \mid E\right) = \frac{\mathcal{L}\left(E \mid C_i\right) P\left(C_i\right)}{\sum_j \mathcal{L}\left(E \mid C_j\right) P\left(C_j\right)} \quad \text{with} \quad i = +, -, \text{dipole} \tag{4.6}$$

where $C_+, C_-, C_{\text{dipole}}$ denotes, respectively, the failure of an edge belonging to the positive monopole, the negative monopole, or the dipole (edge boundary). $P\left(C_i\right)$ are the prior probabilities of these events. The evidence E represents the observed signal difference $\Delta\vec{p}$ restricted to the sensor nodes. $\mathcal{L}(E \mid C_i)$ is the likelihood that a failed edge in the group C_i caused the observed signal $\Delta\vec{p}$. Finally, $P\left(C_i \mid E\right)$ is the posterior probability, that is the probability that the failed edge is within the group C_i given the observed signal. The prior distribution is assumed to be uniform over the edges of each group. Therefore it is given by

$$P(C_i) = \frac{|\mathcal{E}_i|}{\sum_j |\mathcal{E}_j|} \quad \text{with} \quad i = +, -, \text{dipole}$$

where $\mathcal{E}_i$ is the number of edges inside the group i.

The likelihood function must depend on the particular signal that is observed (the *evidence*). Focusing only on the strongest dipole, as a variable we choose the ratio between the signal on the positive monopole and the total signal of the dipole, that is

$$r = \frac{\Delta p_+}{\Delta p_+ + |\Delta p_-|} = \frac{\Delta p_+}{\Delta p_{\text{dipole}}} \tag{4.7}$$

It is also assumed that the likelihood function depends on the ratio between the number of edges internal to the clusters and the total number of edges in the network M. We denote this ratio with m. It corresponds also to the ratio of the expected number of monopoles over all the single-edge failures of the network. Thus we write $\mathcal{L}_m(r) \equiv \mathcal{L}\left(E \mid C_i\right)$. Concerning the shape of the likelihood function, we chose $\mathcal{L}_m(r)$ such that the area between 0 and 1/2 equals to the expected number of negative monopoles, i.e. $m/2$

$$\frac{m}{2} \equiv \int_0^{\frac{1}{2}} e^{-\alpha x} dx = \frac{1}{\alpha}\left(1 - e^{\frac{\alpha}{2}}\right)$$

getting to the equation

$$\alpha = \frac{2}{m}\left(1 - e^{\frac{\alpha}{2}}\right)$$

that is solved using the *Lambert function* $\mathcal{W}_0$ [33]. Thus the likelihood function for the negative monopoles become

$$\mathcal{L}_m(r) = e^{-\alpha} \quad \text{with} \quad \alpha = -2\left(\frac{1}{m} + \mathcal{W}_0\left(\frac{-e^{\frac{1}{m}}}{m}\right)\right). \tag{4.8}$$

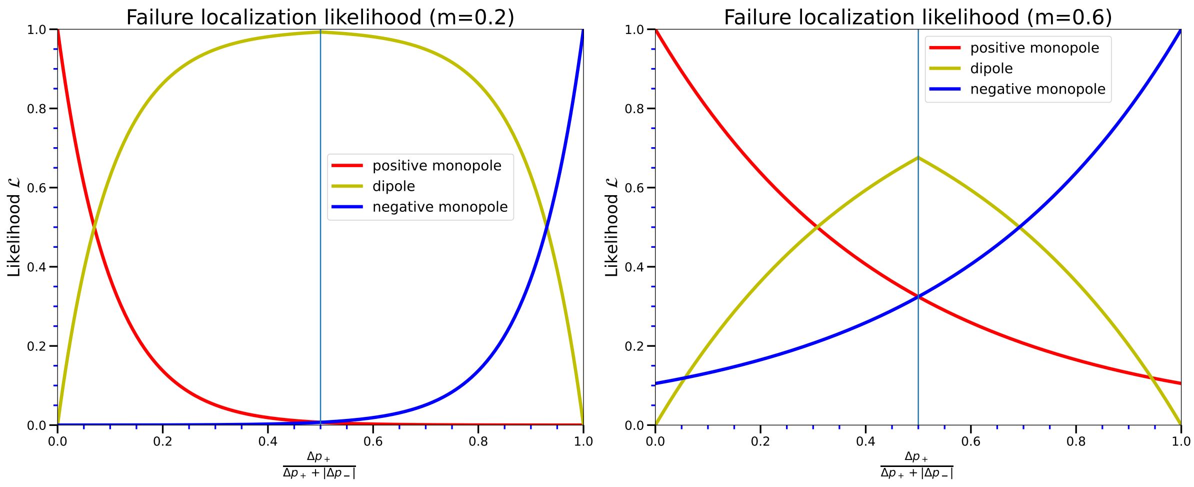


Figure 4.5: Likelihood function used in the failure identification function. On the left the number of expected dipole is greater than the number of expected monopoles (internal failure), thus $0 < m < 0.5$. Thus, more area is given to the dipoles likelihood curve. Conversely, on the right the dipole curve starts decreasing, since the expected number of internal failures outcomes the expected number of inter-cluster failures.

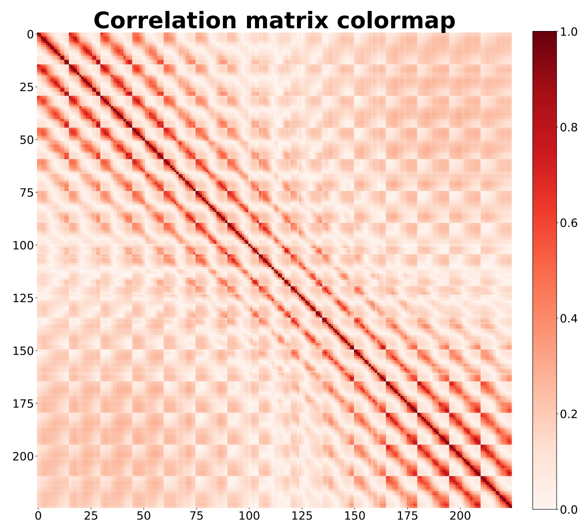
that is plotted in figure 4.5 for two different values of the monopoles ratio m. The curve for the positive monopole and for the dipole are obtained simply translating and reflecting equation 4.8, getting to

$$\mathcal{L}_{m,-}(r) = \begin{cases} \mathcal{L}_m(r) & \text{for } 0 \leq r \leq \frac{1}{2} \\ 0 & \text{for } \frac{1}{2} < r \leq 1 \end{cases}$$

$$\mathcal{L}_{m,+}(r) = \begin{cases} 0 & \text{for } 0 \leq r \leq \frac{1}{2} \\ \mathcal{L}_m(1-r) & \text{for } \frac{1}{2} < r \leq 1 \end{cases}$$

$$\mathcal{L}_{m,\text{dipole}}(r) = \begin{cases} 1 - \mathcal{L}_m(r) & \text{for } 0 \leq r \leq \frac{1}{2} \\ 1 - \mathcal{L}_m(1-r) & \text{for } \frac{1}{2} < r \leq 1 \end{cases}.$$

Finally, the prior is updated according to the Bayes rule 4.6, that gives the probability that the failure occurred in each of the three parts of the strongest dipole, given the number of edges for each possibility (in $P_{\text{prior},k}$), the total number of expected monopoles m and the observed signal r, defined as in 4.7. The candidate cluster in which the failure occurred is chosen to be the one with the highest posterior probability. If the candidate group of nodes is the correct one, its posterior probability is divided by the total number of possible edges it contains, according to the considerations about the edges fluxes above. This means that since we have no information about which of the



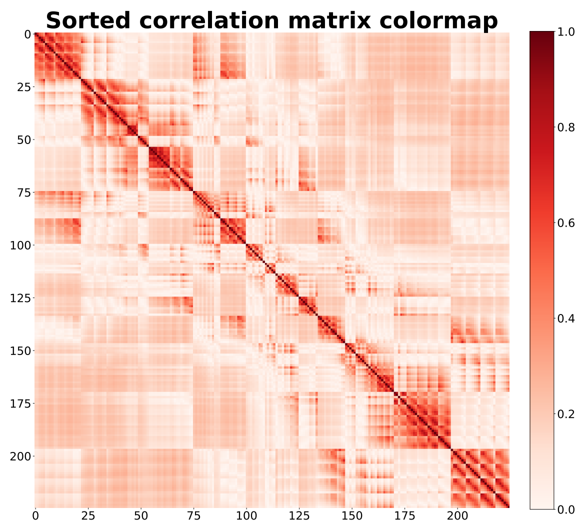


Figure 4.6: On the left, the raw correlation matrix as obtained from 4.10. The diagonal structures are due to the periodicity of the grid network used. On the right, the same correlation is displayed with nodes ordering matching the clusters as produced by the ranking algorithm.

edges might be the failed one, we associate a uniform probability inside each cluster. Therefore, we get the probability associated with a single edge, that we call *confidence*. On the other hand, if the candidate cluster is wrong, the confidence value is set to be zero. Averaging over all the possible failure, we get to average confidence value, that is the one we use as a benchmark for the various clustering methods.

4.3 Clustering methods comparison

As discussed in chapter 3, a dynamics-based clustering method should group together nodes that behave in a similar fashion under the equations that govern the dynamics taking place. Concerning the failure identification problem we are dealing with, it means that the sensor nodes should follow the behavior of the nodes they represent, when one of their edges fails. The correlation 3.47 we have obtained in the previous chapter, contain information about how each node *linearly* correlates with its neighbors under a perturbation of the network weights. This situation should mimic the edge failure described. Thus we now need to partition the network according to 3.47.

The starting point of the proposed clustering method is the covariance matrix 3.47, that we rewrite for convenience

$$E[\delta p_i \delta p_j] = \sum_{\lambda,\mu\neq 0} \frac{1}{\lambda+\mu} |v^\lambda\rangle_i \left[\sum_{k,h} {}_k\langle v^\lambda|C_{kh}|v^\mu\rangle_h\right] {}_j\langle v^\mu|. \tag{4.9}$$

We choose to normalize it, getting to the correlation matrix

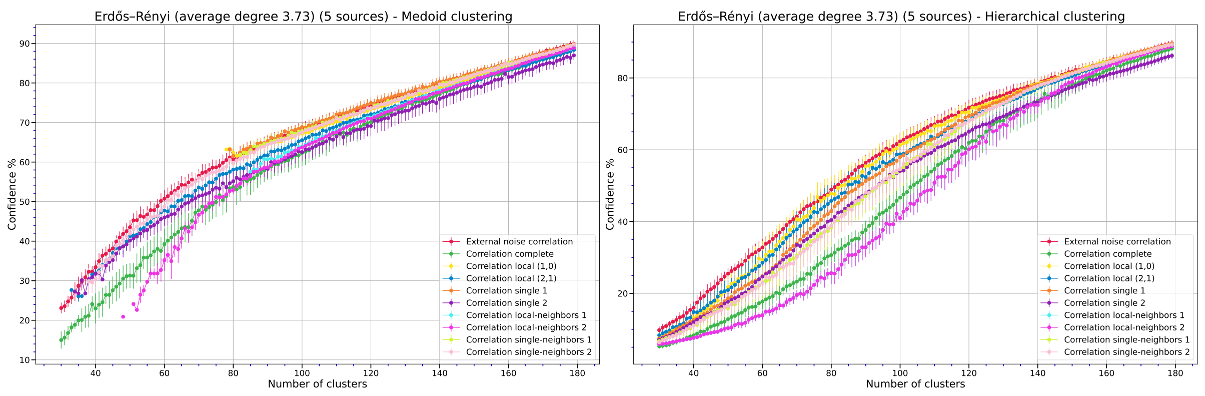
$$C[\delta p_i \delta p_j] = \frac{E[\delta p_i \delta p_j]}{\sqrt{E[\delta p_i^2] E[\delta p_j^2]}}. \tag{4.10}$$

Therefore, we need define an algorithm that takes as input 4.3 and produces clusters of nodes as its output. The idea is that the network is partitioned in such a way that maximally similar nodes are placed within the same cluster. To compute the clusters starting from the correlation matrix above, we propose a clustering algorithm, whose Python code is available on GitHub [15]. The algorithm consists in a simple ranking of the correlation values, such that nodes with strong correlations are grouped together first. The sensor in each cluster is therefore a node with the highest average correlation with the other nodes in the cluster. We will also call this node *medoid* of the cluster. Performing the clustering this way, the network dimension cannot be decreased much mor than a factor of two. Therefore, at each iteration if another node is found that is able to increase the cluster size, while keeping a good average correlation with the other nodes of the cluster, a medoid change is allowed.

It is important to note that equation 4.9 depends on the matrix C_{kh}, that is a Laplacian matrix containing all the fluxes of the system. This is a consequence of the procedure followed to obtain the above result. All the edges of the networks were made to fluctuate, in order to compute the correlation of the nodes states fluctuations $\delta\vec{p}$. However, we might be interested in computing the correlation of each node with just its first or second neighbors and under a different stochastic processes fro each couple of nodes. For example, given node pair, we might be interested in computing their correlation under the fluctuations solely of their adjacent edges. This way we neglect the influence of the rest of the network. Different possibilities were explored (described in Appendix B.2.1), leading to different failure localization efficiencies.

Picture 4.6 shows the absolute value of the correlation matrix 4.10, with the complete matrix C_{kh}, for a grid network 15×15, with edges weights taken to be integers following a uniform distribution. On the left the correlation matrix is displayed with the original nodes order. The diagonal structures are due to the periodicity of the particular lattice network used. It simply signals that neighbors nodes are more correlated than nodes far apart. On the right, the same correlation matrix is shown ordering the nodes in such a way that nearby nodes belong to the same cluster, as produced by the proposed ranking algorithm. Therefore, each block diagonal structure represents a cluster.

We wrote a second clustering method, that we call *affinity clustering*, different from the ranking medoid clustering mentioned above. In particular, it was thought to create more "spherical" clusters, that for some networks structures (like the grid network, as we will see) leads to better results. Moreover, generally it generates bigger clusters than the ranking medoid clustering, therefore allowing a more pronounced dimensionality reduction of the network. This function is described and available on [15]. It is loosely



(a) Medoid based clustering (b) Hierarchical clustering

Figure 4.7: Comparison between the custom medoid-based clustering and the traditional hierarchical clustering. They were both used for all the correlation functions described in the main text. The best results (higher values of the confidence value) are obtain for the medoid clustering algorithm.

inspired by the Affinity Propagation clustering used in data analysis [18].

Furthermore, we notice that the proposed correlation matrix can be interpreted like an effective similarity measure between the nodes. Thus one can also make use of all the traditional techniques employed in data clustering [25]. Those methods usually require as inputs a set of data points embedded in a metric space, and the definition of a distance between these data points, to be used as a dissimilarity measure. Alternatively, some methods can take as input a precomputed similarity matrix, without the need of an embedding. These are the methods of interest for our application, since our similarity function is computed from a simulated dynamics, and not from nodes similarities that emerge from an embedding of the network. For instance, one might exploit the spectral clustering of section 2.2. In this case, the similarities between nodes are interpreted as the edges weights of an effective graph. Computing its Laplacian, one can easily retrieve the clusters. Another popular method that can take a precomputed similarity matrix as its input is called *hierarchical clustering*, which seeks to build a hierarchy of cluster, hence its name [25]. Both of these similarity-based methods are used for the comparison.

In section B.2.1 of Appendix B, we compare two proposed clustering algorithms with the hierarchical clustering (with linkage choice "*single*", as implemented by the Python scientific package *Scikit-Learn* [43] and the unnormalized spectral clustering, all applied to the different correlation matrices, as explained in the Appendix main text. Since spectral and hierarchical clustering are not expected to provide a default representative node choice for each cluster, the node with the highest betweenness centrality (equation 4 of chapter 1) is chosen. For instance, figure 4.7 shown the comparison between the custom medoid clustering and the hierarchical clustering using an Erdős–Rényi network

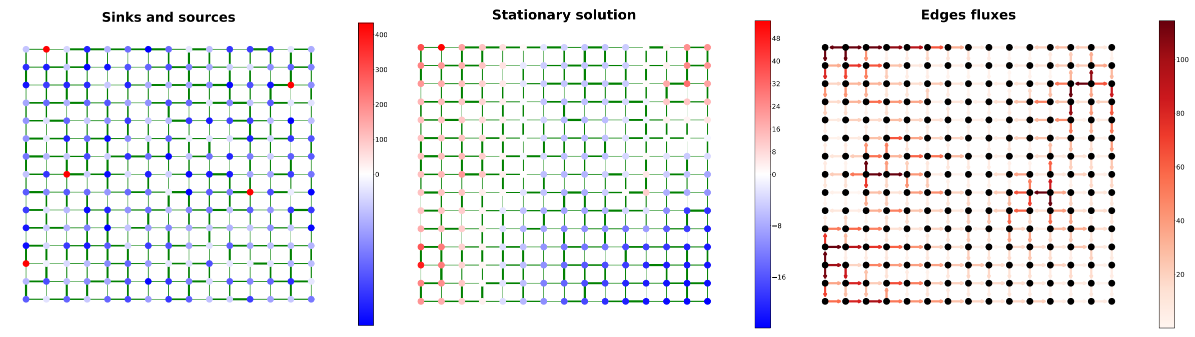


Figure 4.8: On the left, forcing distribution over a grid network. The number of sources defined is five (shown in red). The sinks values (negative, shown in blue) are here chosen to follow a uniform distribution. On the right, the correspondent stationary distribution of the signal over the nodes and stationary fluxes over the edges. For the edge fluxes plot, the arrow point to the same direction to the flux. Notice that signal and fluxes values are higher nearby the sources nodes.

model, with average degree $\langle k \rangle = 3.73$, five sources and a uniform distribution of edges weights and sinks values. We see that the confidence parameter for the custom ranking clustering is on average higher than the hierarchical clustering. This applies for all the variations of the correlation used, showing an overall better performance. Concerning the comparison of the different correlations among themselves, we see that generally the ones that takes into account also the second neighbors of each node perform generally worse than the corresponding ones that stop at first neighbors. Therefore, those are the ones that will be kept for the successive comparisons with the traditional network partitioning techniques.

Comparison with traditional clustering methods

The first traditional structure-based clustering method used for the comparison is *modularity maximization*, explained in section 2.3, with the greedy algorithm by Clauset-Newman-Moore [11], one of the most used. The second one is the divisive algorithm by Girvan and Newman [21], that detects communities by removing edges at each iteration from the original graph. The chosen edge traditionally is the one with the highest betweenness centrality (equation 4) in the network . Step by step the underlying community structure of the network emerges. For both algorithm, we used the versions implemented in *Networkx* [23], a Python package for the creation and study, both structural and dynamical, of complex networks.

As shown in figure 4.8, the presence of a forcing in the network influences the distribution of the potential and therefore also of the fluxes. Since our clustering method

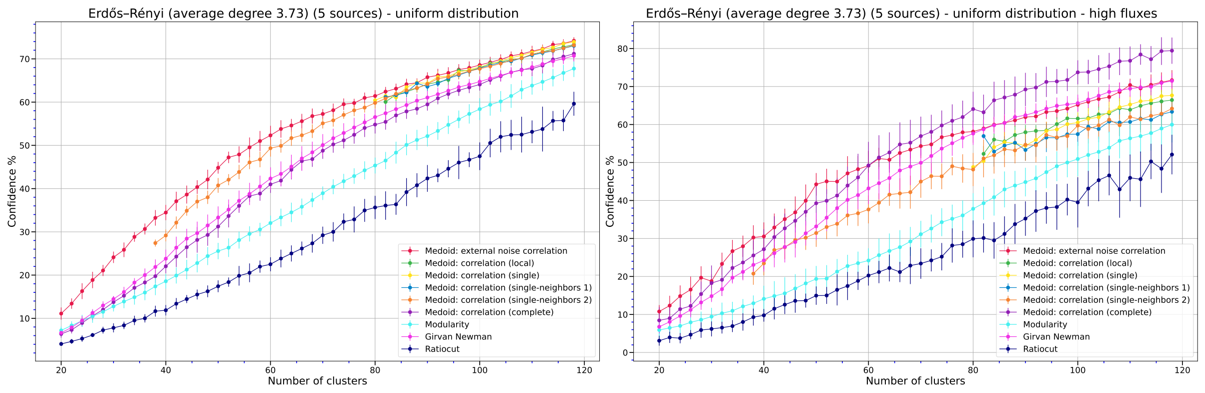


Figure 4.9: Simulations results for 10 Erdős–Rényi networks with average degree of $\langle k \rangle = 3.73$, five sources and uniform distribution of the edge weights and sinks values. On the left, the results are averaged over the all possible single-edge failure in the network. On the right, only the failure of half of the edges with higher flux is considered. In both cases, the proposed clustering performs better on average.

gives more importance to edges with higher fluxes, we expect its performance to increase when considering only the failure of those edges.

First, the comparison is performed using the same *Erdős–Rényi network model* as the one considered above, i.e. average degree of $\langle k \rangle = 3.73$, five sources and uniform distribution of the edge weights and sinks values. Figure 4.9 shows the failure localization efficiency of the compared methods as a function of the number of clusters. On the left, the results are averaged over all possible edge failures. On the right, only half of the edges with higher fluxes are considered. In both cases, our clustering method, based on different versions of the correlation matrix 4.3 performs marginally better than the Girvan Newman algorithm and considerably better than the modularity maximization and the (relaxed version of) the Ratiocut. In particular, the difference becomes more prominent when considering only the higher fluxes of the network. We notice, that the results of three versions of the correlation matrix start only at around 80 clusters. The reason is that those matrices are based only on the first neighbors correlations. Therefore, using the proposed medoid clustering algorithm, we cannot increase the size of each cluster more than its ability to exploit the information the correlation matrix provides.

In Appendix B we show the results for the same network model, but using a homogeneous distributions of the weights. The same kind of comparison is then performed changing the number of sources from 5 to 50. The results are similar to the one shown in figure 4.9, with the only exception of the case with 50 sources, uniform distribution of the weights and higher fluxes failures, for which le Girvan-Newman algorithm perform slightly better.

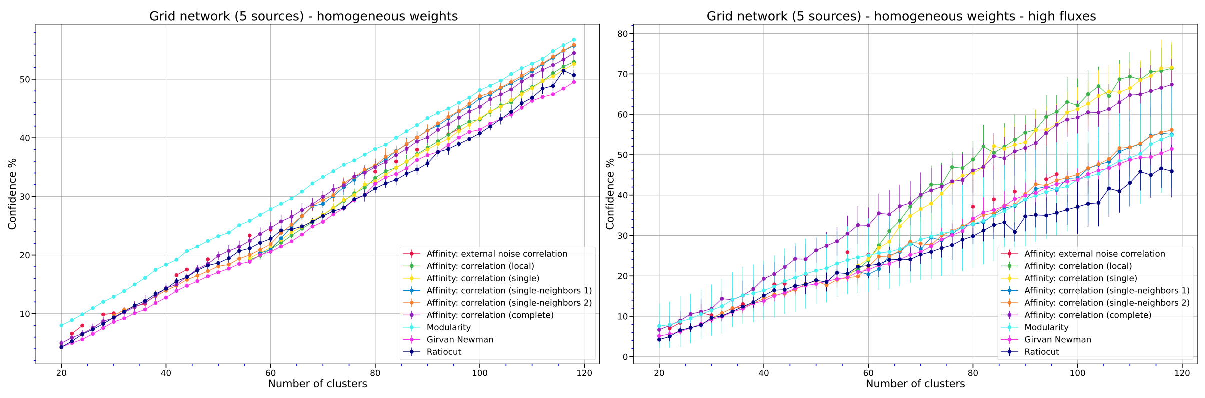


Figure 4.10: Simulation results for ten 15 × 15 grid network, with five sources, uniform distribution of the sinks values and homogeneous distributions of the weights. Notice that for this network model the difference between the results averaged over all edges and those averaged over the edges with higher flux is more evident.

The second set of comparisons is performed using a 15 × 15 *grid network*, with five sources, uniform distribution of the sinks values and homogeneous distributions of the weights. For this particular network model, we used the above mentioned Affinity clustering algorithm to partition the network starting from 4.3, in order not to have the clusters size problem explained above. From the results shown in Figure 4.10 it can be noted that the efficiency gain of our clustering method when considering only the higher fluxes is even more prominent.

In Appendix B we show the results of the same comparison using this time a uniform distributions of the weights. Analogously to the Erdős–Rényi networks, the same kind of results are presented changing the number of sources from 5 to 50. The results are similar to the one shown in figure 4.10. Concerning the configuration with 50 sources, uniform distribution of the weights and higher fluxes failures, the spread between the proposed method and the traditional clustering techniques decreases. This was the same configuration that led to the worst results for the Erdős–Rényi model comparison. Therefore, we might infer that the proposed clustering method performs better, in comparison to traditional techniques, when the number of sources in the network is lower. In this situation, in fact, we have a greater heterogeneity of fluxes in the system, and the proposed method based on 4.3 is indeed sensitive to the fluxes distribution around the network.

A third category of networks used for this comparison is *k-regular random graphs*, a particular type of structure in which each node as the same degree k. We set the degree to be equal to four $k = 4$. Analogously to the models analyzed previously, the number of sources is five, the sinks values are drawn from a uniform distribution, and the edges weights from an homogeneous distributions. As shown in figure 4.11, the qualitative behavior for this kind of network is very different from the one encountered above.

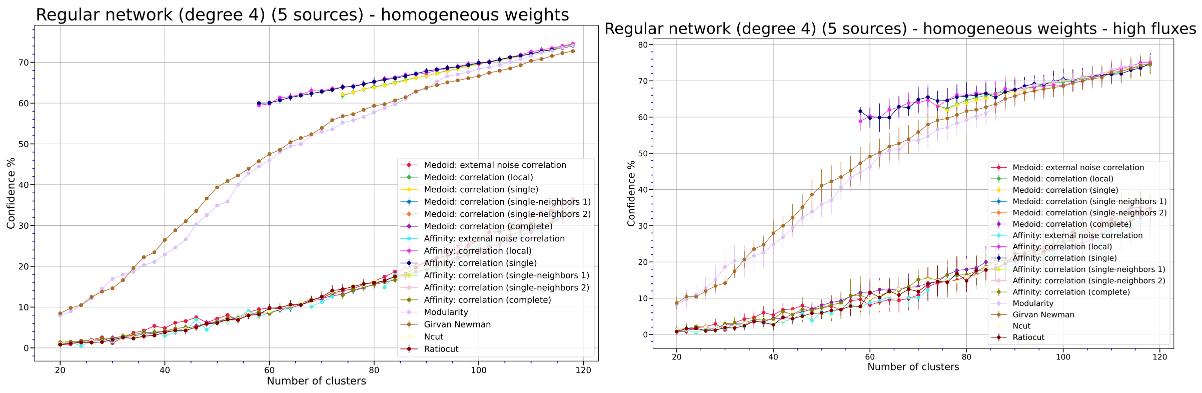
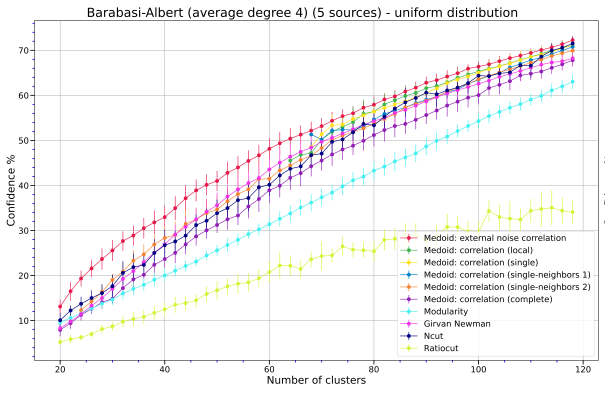


Figure 4.11: Simulation results averaged for ten regular networks with degree $k = 4$, five sources, uniform distribution of the sinks values and homogeneous distributions of the weights. For this particular network we show all the clustering methods used.

There is a clear separation between low performance clustering methods and very high performance algorithms. The higher failure localization confidences are obtained with Modularity maximization (whose performance for the other types of networks was not satisfactory), Girvan Newman algorithm, and four of our proposed clustering procedures. In particular, we notice that using the affinity clustering with the correlations *local* and *single* (as explained in Appendix B) leads to better result, even if for a limited range of clusters number. Therefore, using different clustering algorithm on the same correlation matrices might expand this range, further improving the results. We notice that the results do not change in a considerable way when restricting the edge failures only on edges with higher fluxes.

Finally, we compare the failures identification performances for a *Barabasi-Albert network* model [1]. Barabasi-Albert networks feature a scale-free structure, meaning that they have a power-law degree distribution, in particular $P(k) \sim k^{-3}$. Many real world networks might be described by this model. For this particular type of networks, Girvan Newman algorithm and the proposed clustering procedures perform in a similar fashion, whereas modularity maximization and the (relaxed version of) Ratiocut perform considerably worse. Differently form the Erdős–Rényi and grid network cases, and analogously to the 4-regular graphs, there is no clear distinction between the failure identification results of higher edges. In Appendix B we shoe the simulations results with the same current network configuration, changing only the distribution of the edges weights, take to be homogeneous. The results for the Girvan Newman algorithm and the Modularity maximization change marginally, improving in the homogeneous case.



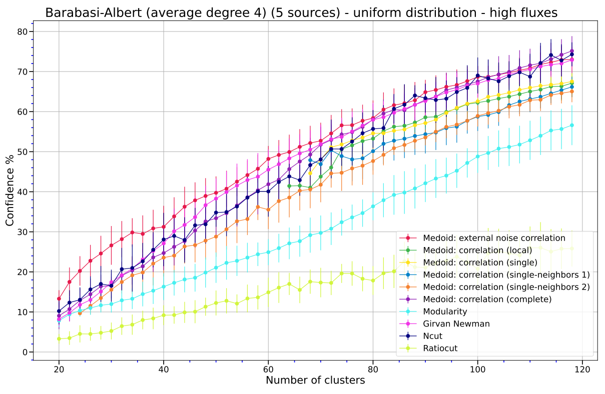


Figure 4.12: Averaged simulation results for ten Barabasi-Albert networks with average degree $k = 4$, five sources, uniform distribution of the sinks values and homogeneous distributions of the weights.

Chapter 5

Conclusions

Transport systems are ubiquitous in nature and modern society, displaying several architectures and structures. Their topology can be effectively encoded by a network structure. Diffusion process are a commonly studied linear dynamical model taking place on top of these networks. The time evolution of a diffusion process on a network is given by the Laplacian matrix of the graph, whose properties were studied in the first chapter of this thesis.

Transport systems are subject to failure of one or more components. If this event occurs, the transport capacities of the network are reduced. Observing the state of the whole system is usually unfeasible, if not impossible. Therefore, a commonly faced problem is finding an optimal sensor placement over the network, such that the observability of some failure events is enchanted. A typical approach is given by graph partitioning tools developed in network science. These techniques, like modularity maximization, are solely based on the topology of the network and they were described in the second chapter of this thesis. In particular, spectral clustering, exploiting the information contained in the Laplacian matrix spectrum, admits an interpretation based on a spring-mass system, for bidirectional networks.

We proposed a clustering procedure that considers not only the network structure, but also the dynamics taking place on it. In particular, we studied the linear dependence among the nodes states under the fluctuations of the edge weights. The correlation matrix obtained is used to partition the network in such a way that dynamically similar nodes are placed inside the same cluster. Therefore, a coarse-grained network model is generated, that captures the dynamics of the network on a lower dimensional space. The performed simulations show that the proposed clustering method is more efficient in localizing single-edge failures involving connections with higher flux. Future extensions of the simulation results, could make use of different network configurations rather than the one used, with disparate sinks and sources distributions.

Moreover, in the scientific literature, the problem of dynamics observability is usually considered only diffusion-like dynamics. Therefore, a possible extensions of the present work include taking into consideration also nonlinear dynamics. Finally, the link between coarse-grained networks and the model reduction, typically considered in control theory, might be further explored, with the aim of applying a control theory on a network structure.

Appendix A

Rescaled process moments

Fixing $t < s$, the covariance of the rescaled process reads

$$
\begin{aligned}
E\left[\delta\hat{p}_i(t)\delta\hat{p}_j(s)\right] = & \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\right] p_l^s p_m^s \qquad \text{(A.1)}\\
& + \varepsilon^{1/2} \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{p}_l(u)\right] p_m^s \\
& + \varepsilon^{1/2} \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{p}_m(u')\right] p_l^s \\
& + \varepsilon \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{p}_l(u)\delta\hat{p}_m(u')\right]
\end{aligned}
$$

where the first term represents the stationary process (already computed in section 3.5, and corresponding to the equation 3.42), and the others are higher order corrections, defined iteratively. Taking for instance the second terms and substituting

$$
\delta\hat{p}_l(u) = -\varepsilon^{1/2} \int_0^u du'' e^{-L_{lr}(u-u'')} \delta\hat{L}_{rv}(u'')\delta\hat{p}_v(u'') - \int_0^u du'' e^{-L_{lr}(u-u'')} \delta\hat{L}_{rv}(u'') p_v^s
$$

we get to the following expression

$$
\begin{aligned}
& -\varepsilon^{1/2} \int_0^t du \int_0^s du' \int_0^u du'' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} e^{-L_{lr}(u-u'')} E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{L}_{rv}(u'')\right] p_m^s p_v^s \\
& -\varepsilon \int_0^t du \int_0^s du' \int_0^u du'' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} e^{-L_{lr}(u-u'')} E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{L}_{rv}(u'')\delta\hat{p}_v(u'')\right] p_m^s
\end{aligned}
$$

in which the first term is the third moment of the rescaled noise $\delta\hat{L}$ and the second term defines the successive iterations. Since the Ornstein-Uhlenbeck process is a Gaussian process, the odd moments vanish. For the higher order even moments, we can use Isserlis theorem [57] to reduce them to products of variances. Thus we check what happens for the first higher order moment that do not vanish, i.e. the fourth one. For instance, iterating again the expression above with

$$\delta\hat{p}_v(u'') = -\varepsilon^{1/2}\int_0^{u''} du''' e^{-L_{vw}(u''-u''')}\delta\hat{L}_{wz}(u''')\delta\hat{p}_z(u''') - \int_0^{u''} du''' e^{-L_{vw}(u''-u''')}\delta\hat{L}_{wz}(u''')p^s_z$$

we get to

$$-\varepsilon^{3/2}\int_0^t du \int_0^s du' \int_0^u du'' \int_0^{u''} du''' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} e^{-L_{lr}(u-u'')} e^{-L_{vw}(u''-u''')}$$
$$E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{L}_{rv}(u'')\delta\hat{L}_{wz}(u''')\delta\hat{p}_z(u''')\right] p^s_m$$

for the generator of successive iterations term and

$$-\varepsilon\int_0^t du \int_0^s du' \int_0^u du'' \int_0^{u''} du''' e^{-L_{ik}(t-u)} e^{-L_{jh}(s-u')} e^{-L_{lr}(u-u'')} e^{-L_{vw}(u''-u''')}$$
$$E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{L}_{rv}(u'')\delta\hat{L}_{wz}(u''')\right] p^s_m p^s_z \quad \text{(A.2)}$$

for the fourth moment. Using Isserlis theorem, the last term can thus be written as

$$E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\delta\hat{L}_{rv}(u'')\delta\hat{L}_{wz}(u''')\right] = E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\right] E\left[\delta\hat{L}_{rv}(u'')\delta\hat{L}_{wz}(u''')\right] +$$
$$E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{rv}(u'')\right] E\left[\delta\hat{L}_{hm}(u')\delta\hat{L}_{wz}(u''')\right] + E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{wz}(u''')\right] E\left[\delta\hat{L}_{hm}(u')\delta\hat{L}_{rv}(u'')\right]$$

that for $\varepsilon \to 0$ becomes

$$\begin{aligned} C_{klhm}C_{rvwz}\delta(u-u')\delta(u''-u''') \quad &+ \quad C_{klrv}C_{hmwz}\delta(u-u'')\delta(u'-u''') \\ &+ \quad C_{klwz}C_{hmrv}\delta(u-u''')\delta(u'-u''). \end{aligned}$$

Substituting the last expression into A.2 we get three terms. The first one, for instance, is

$$-\varepsilon\int_0^t du \int_0^u du'' e^{-L_{ik}(t-u)} e^{-L_{jh}(t-u')} e^{-L_{lr}(u-u'')} C_{klhm}C_{rvwz}1_{vw}p^s_m p^s_z \quad \text{(A.3)}$$

that integrating in the eigenbasis of L and taking the limit $t \to \infty$ leads to

$$\propto \frac{\varepsilon}{\lambda_i + \lambda_j}.$$

that vanishes for $\varepsilon \to 0$. For all the other terms the idea is analogue. For the sixth moment, for example, we get 15 terms, each of which has an ε^2 and 6 integrals with 6 exponentials and the product of 3 Dirac's deltas: contracting the deltas, we are left with 3 integrals (one more than for the 4th moment) that should lead to a term like

$$\propto \frac{\varepsilon^2}{\lambda^2}.$$

Similarly for all the other higher moments. Thus summing all terms that we get from the iterative approach we get something like

$$\lim_{\varepsilon\to 0}\sum_{k=2}^{\infty}\left(\frac{\varepsilon}{\lambda}\right)^k$$

that since its a convergent geometric series ($\varepsilon \ll \lambda_F$) we can interchange the limit and the summation

$$\sum_{k=2}^{\infty}\lim_{\varepsilon\to 0}\left(\frac{\varepsilon}{\lambda}\right)^k = 0$$

The same procedure can be applied to the other two terms in A.1: the third one is identical to the second one (with a changing in the indices, irrelevant in the limit of $\varepsilon \to 0$) and the fourth one produces similarly higher order moments.

Thus the only term that remains is the first one in A.1 that has no ε parameter in front of the integral, thus

$$\begin{aligned}
E\left[\delta\hat{p}_i(t)\delta\hat{p}_j(s)\right] \xrightarrow[\varepsilon\to 0]{} & \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)}e^{-L_{jh}(s-u')}E\left[\delta\hat{L}_{kl}(u)\delta\hat{L}_{hm}(u')\right]p_l^s p_m^s \\
&= \int_0^t du \int_0^s du' e^{-L_{ik}(t-u)}e^{-L_{jh}(s-u')}C_{klhm}\delta(u-u')p_l^s p_m^s \\
&= \int_0^t du e^{-L_{ik}(t-u)}e^{-L_{jh}(t-u)}C_{klhm}p_l^s p_m^s
\end{aligned}$$

that equals to 3.47, as computed in section 3.5.

The same recursion idea can be applied to the mean of the rescaled process, showing that

$$E[\delta\hat{p}_i(t)] = 0 \quad \forall t. \tag{A.4}$$

i.e. as before, the mean of the rescaled process vanishes at all times.

Appendix B

Simulation plots

B.1 Problem introduction

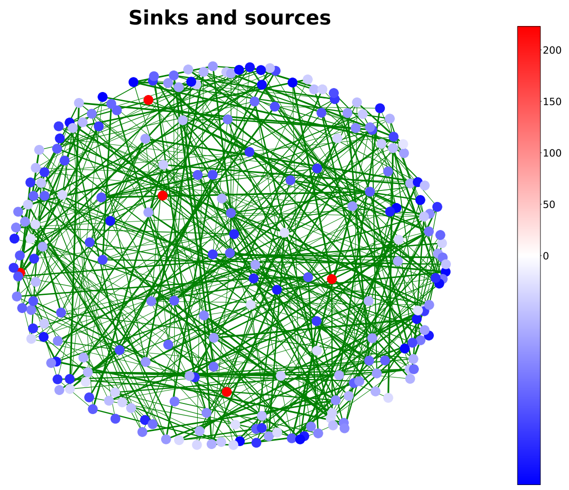


Figure B.1: Sinks and sources s_i distribution for an Erdős–Rényi network. The sinks values (displayed in blue) and the edges weights (thickness of the lines) follow a uniform distribution. Since we are considering a micro-canonical ensemble, we have that $\sum_i s_i = 0$. To enforce this condition, the values of the sources (here displayed in red) need to compensate the sinks values. We choose the sources values to be the same, i.e. to be equal to the sum of the sinks, changed sign, and divided by the number of sources.

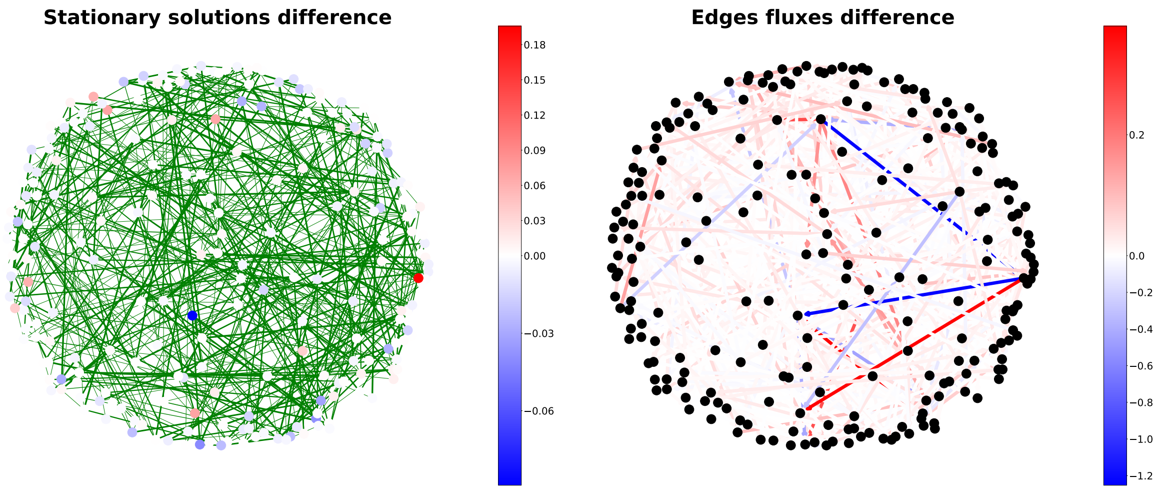


Figure B.2: Stationary potentials and fluxes differences after a single-edge failure for an Erdős–Rényi network with 225 nodes, uniform weights and forcing distributions. If the state of the whole network is available, the failure localization problem becomes trivial.

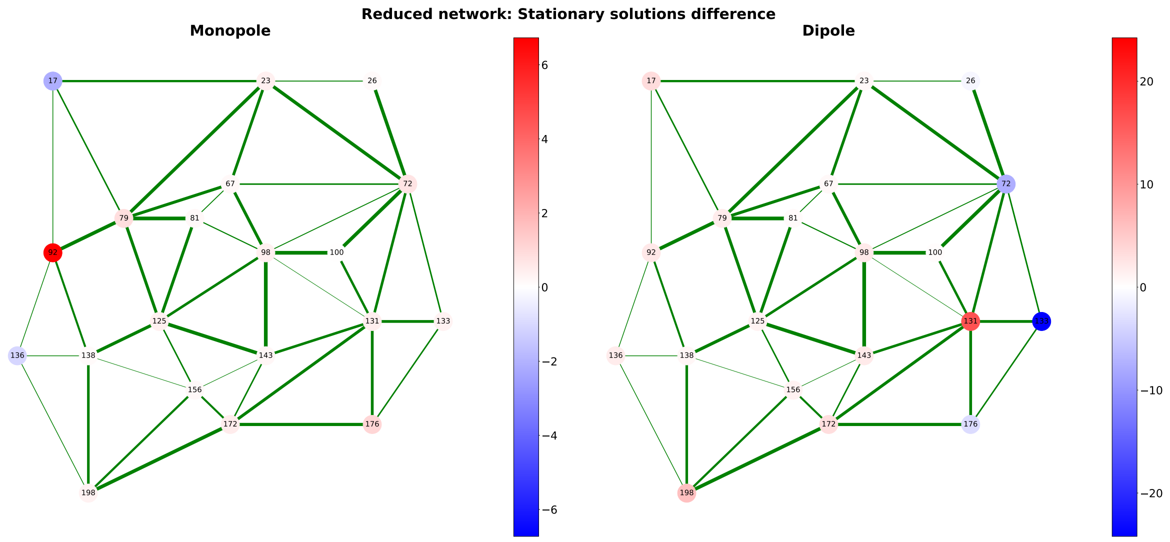


Figure B.3: Stationary potential difference (before and after the failure) for the reduced model of a 15 × 15 lattice network. We might face two possible outcomes. On the left the failed edge is internal to one of the clusters, leading to a net signal (monopole) on the sensor of that cluster. On the right, the failed edge lies between two clusters (i.e. on the edge-boundary of those clusters). An optimal clustering method should distinguish clearly those situations, reducing the dimensionality of the network but keeping failure localization capacities.

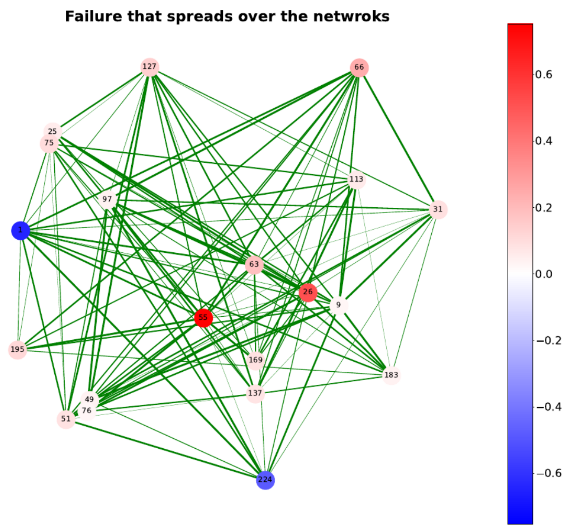


Figure B.4: Single edge-failure for the reduced version of an Erdős–Rényi graph, originally with 225 nodes. This particular edge-failure causes a spread signal over the reduced network, involving most sensor nodes and making the localization fuzzier. Nevertheless, if the boundary between positive and negative nodes is clear (i.e. there is a unique couple of nodes whose potential drop is maximum). Therefore the failure localization is still possible.

B.2 Erdős–Rényi networks

B.2.1 Comparison

In this section we show the results relative to different ways the correlation matrix 4.3 can be exploited to partition the network. First of all, *external noise correlation* refers to the first correlation matrix 3.34 obtained in section 3.4. The stochastic process considered to compute the correlations of the potential fluctuations was an external white noise added to the forcing of the system. As explained in the main text, this correlation depends solely on the Laplacian spectral properties, and not on the fluxes of the network.

All the other correlation functions refer to 4.3, with different categories of fluctuating edges each. Recall that in order not to consider the fluctuations of an edge for the computation of the correlation matrix, it is enough to omit the correspondent term in the fluxes matrix C_{kh}.

- *Correlation complete* refers to the case in which all the edges of the network are made to fluctuate, thus corresponding to the original 4.3. It worth noting that when one computes the correlation coefficient of adjacent nodes, the fluctuation of the edge between them gives a negative contribution, as intuitive. Moreover, one might be interested in knowing the correlation properties of these two nodes under a fluctuating environment. From this consideration, the following correlations were explored.

- *Correlation local* is computed as follows. Consider a single node of the network, called *center* node. Take the group of nodes distant k_{neig} step from it. The edges allowed to fluctuate are the ones further than $k_{\text{local}} > k_{\text{neig}}$ steps from the center node. This defines the environment for the center node. Then, the correlation of the center node is computed only for the nodes belonging to its ambient, that is the set of nodes distant k_{neig} step from it. One might consider different combinations of the parameters k_{neig} and k_{local}. The one presented in the plot are $(k_{\text{local}}, k_{\text{neig}}) = (1, 0)$ and $(k_{\text{local}}, k_{\text{neig}}) = (2, 1)$. Thus the former computes the correlations only with the first neighbors for each node, whereas the latter extends the computation also to the second neighbors. Also the combinations $(k_{\text{local}}, k_{\text{neig}}) = (1, 1)$ and $(k_{\text{local}}, k_{\text{neig}}) = (2, 2)$ were explored, leading to worse results.

- Starting from the considerations above, *Correlation single 1* refers to the correlations values computed keeping still only the edges connecting the two nodes. *Correlation single 2* tries to extend the above idea also to second neighbors. Therefore, the correlations with the first neighbors of each node are the same as Correlation single 1, whereas for the second neighbors all the edges belonging to the shortest path between them are kept still.

- *Correlation local-neighbors* is the same idea as correlation local, but only the edges adjacent to the ambient node are made to fluctuate. Therefore, a different stochastic process is considered for each center node and each ambient node.

- *Correlation single-neighbors 1* takes each couple of node that are neighbors and computes the correlations making when only the edges that are neighbors of the two fluctuate, keeping the edge between them still, like always. Correlation single-neighbors 2 extend this approach also to second neighbors.

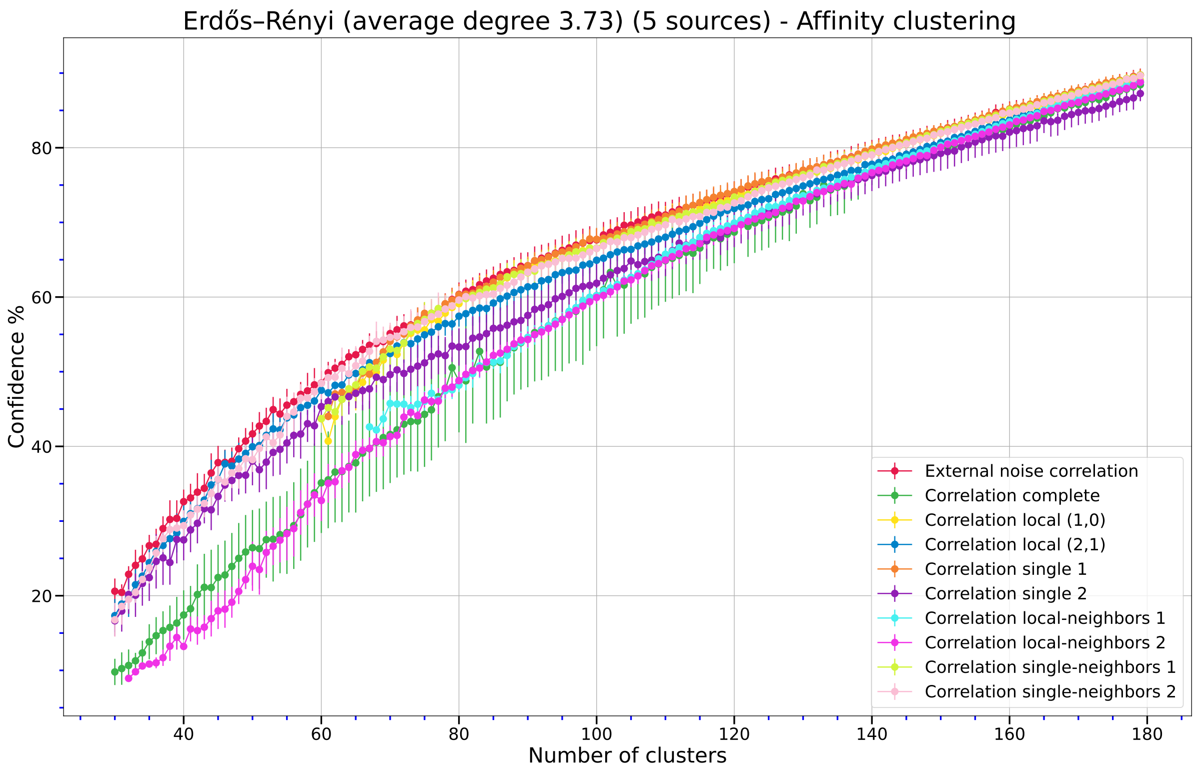


Figure B.5: Failure identification results using the correlation matrices explained in the main text and a partitioning method that we call affinity clustering.

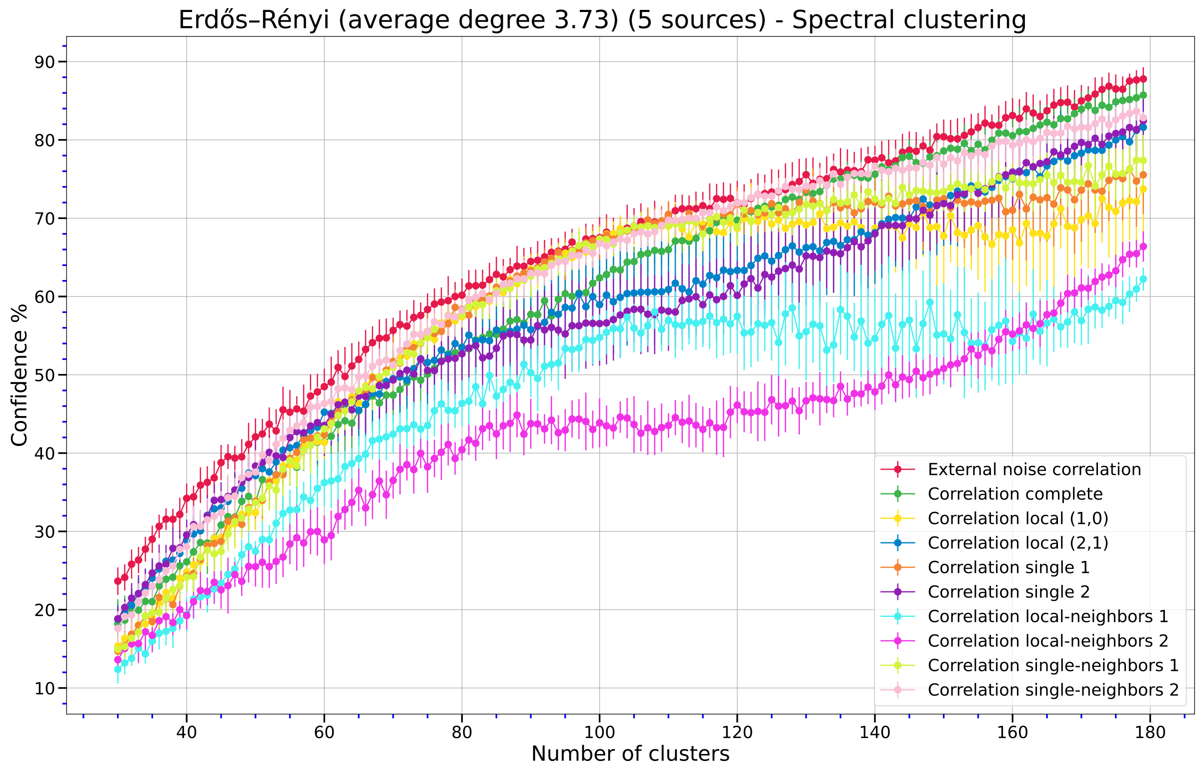
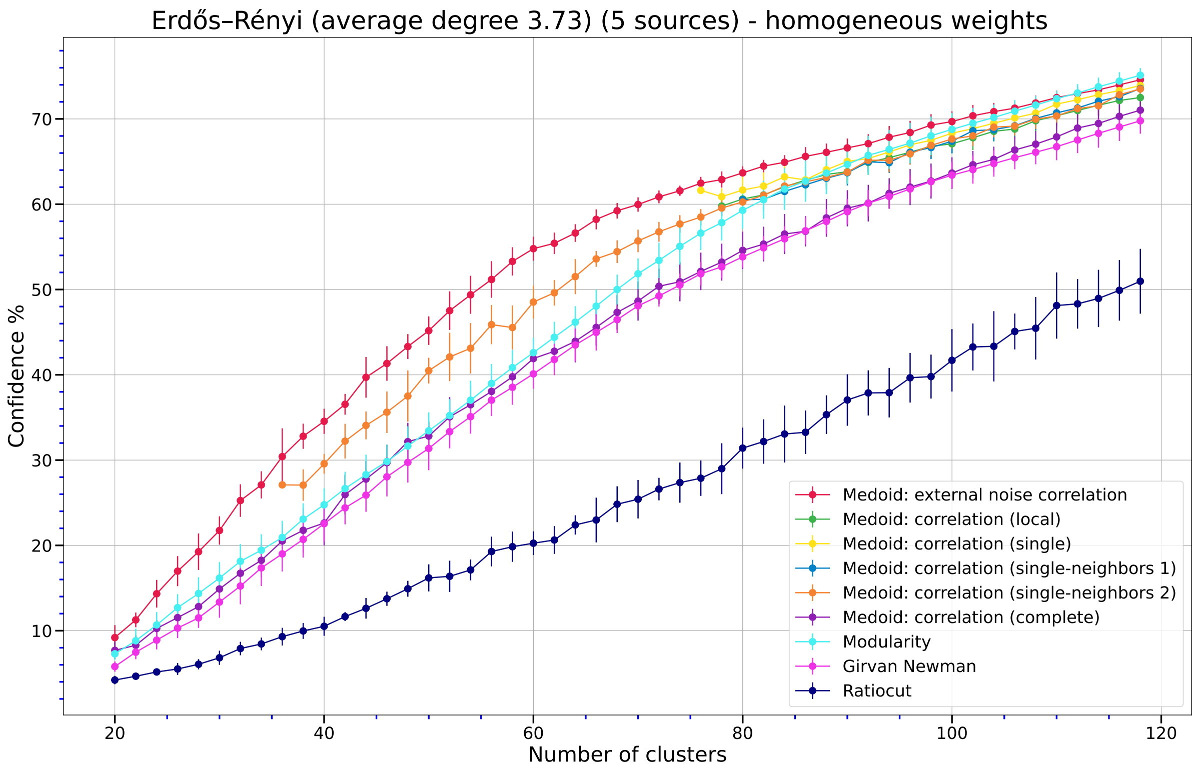
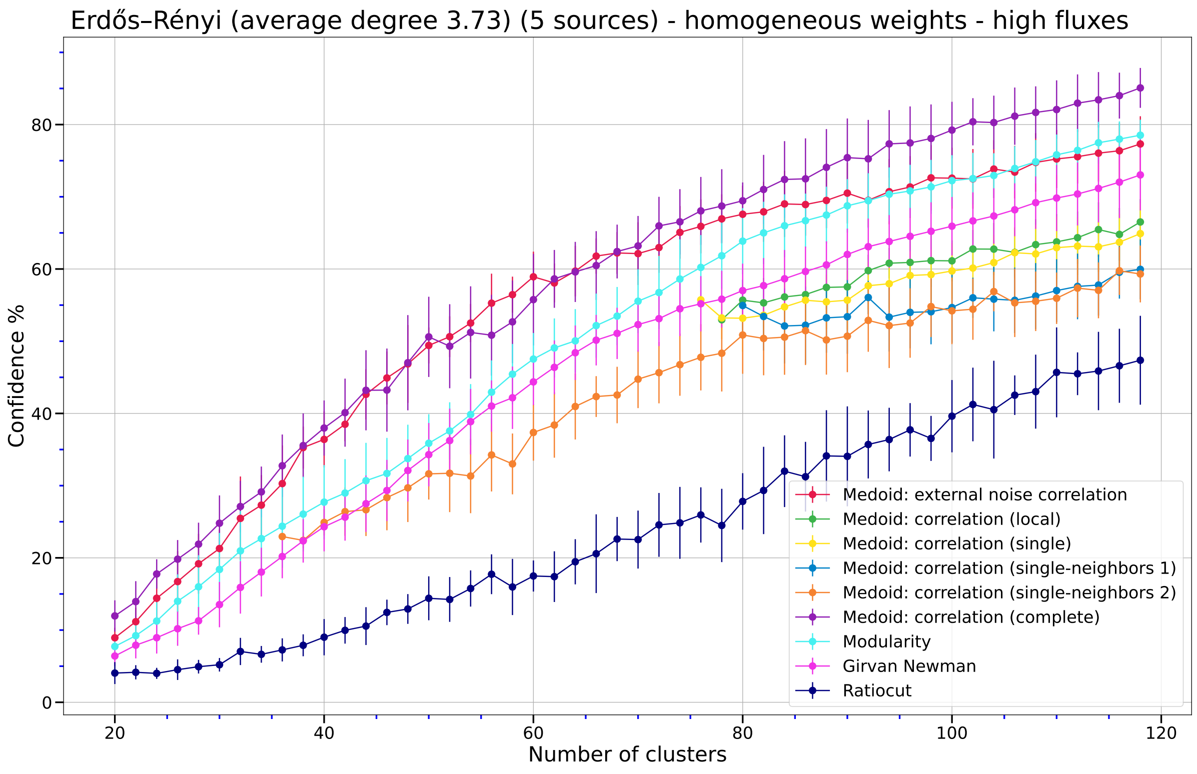
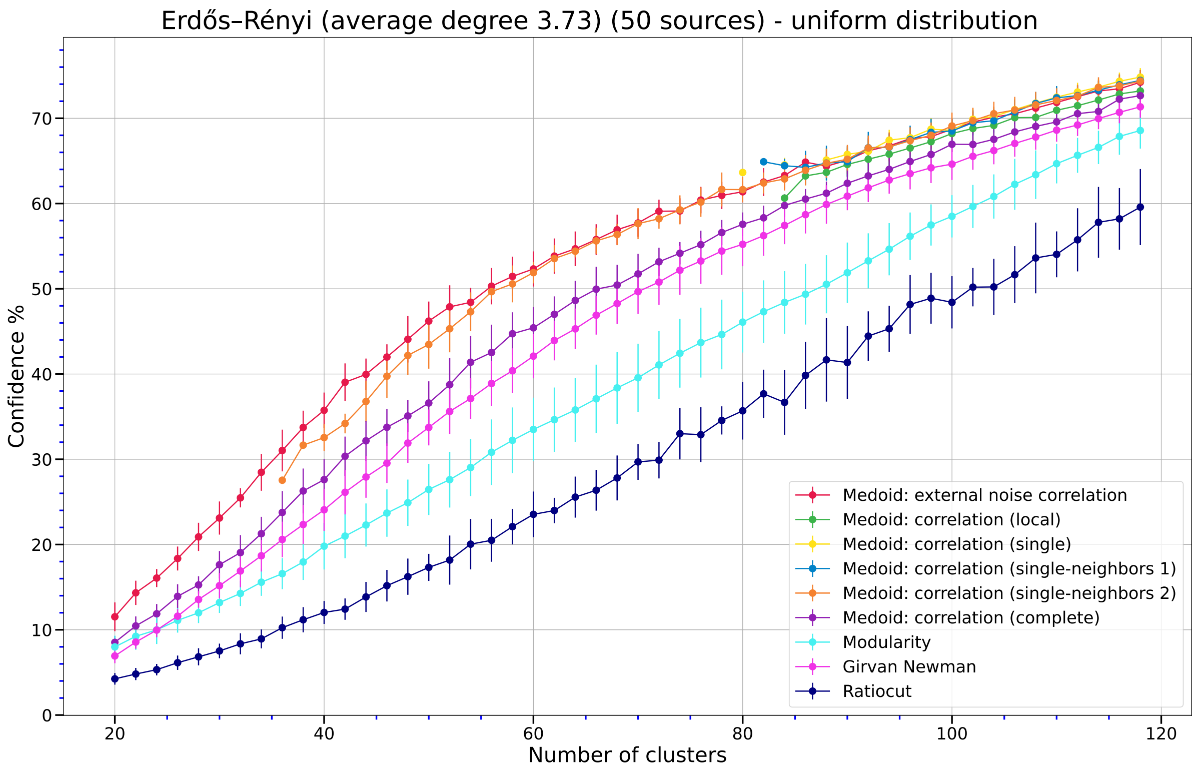
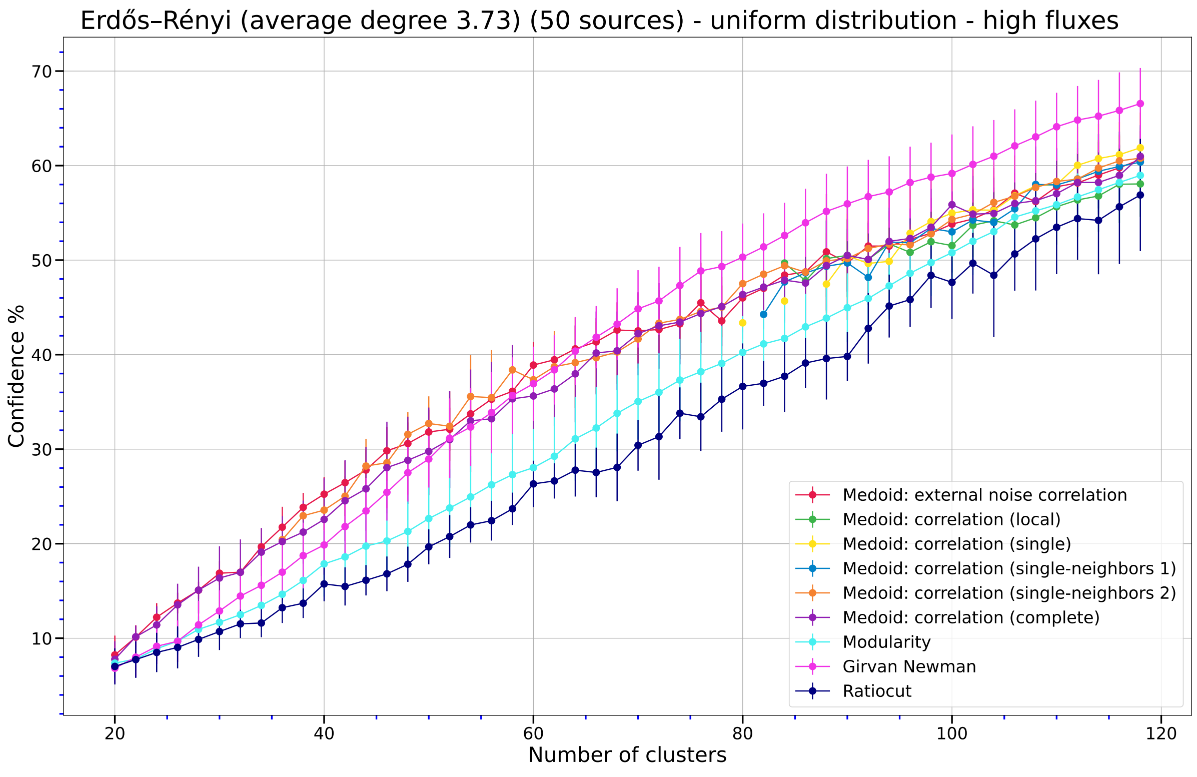


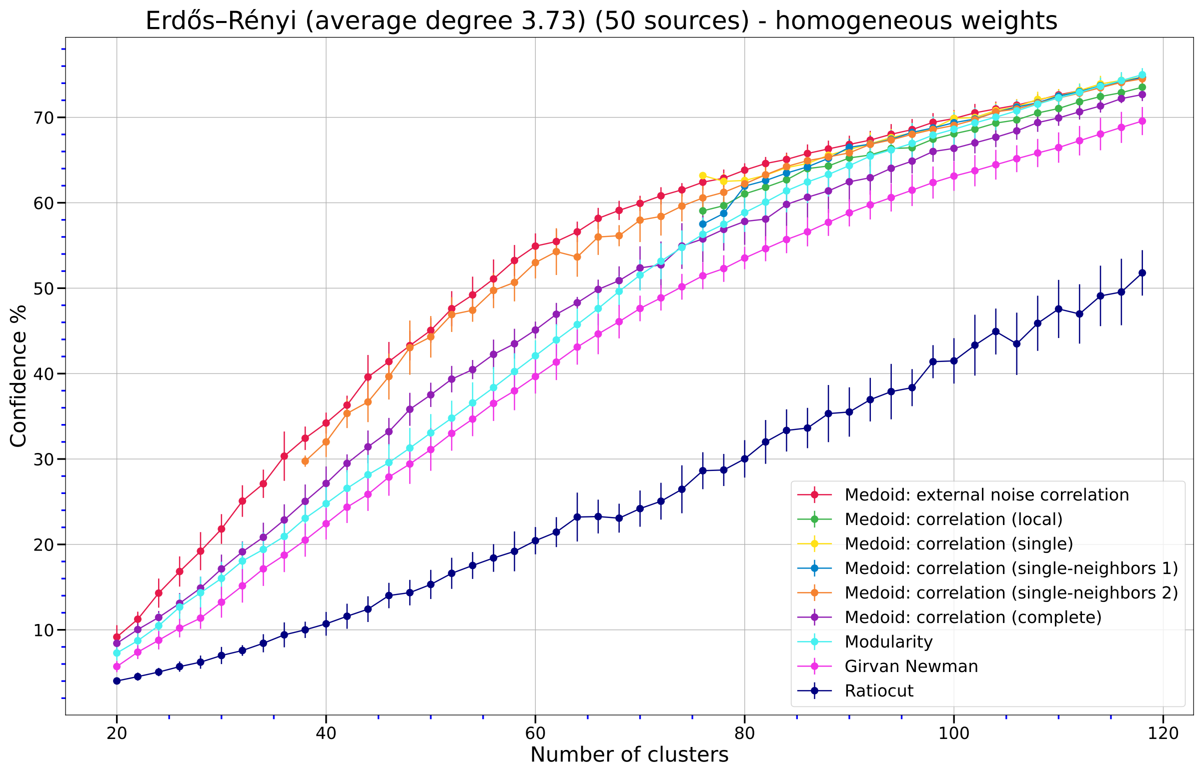
Figure B.6: In this comparison, a traditional spectral clustering was used to partition the network starting from the correlation matrices explained in the main text.

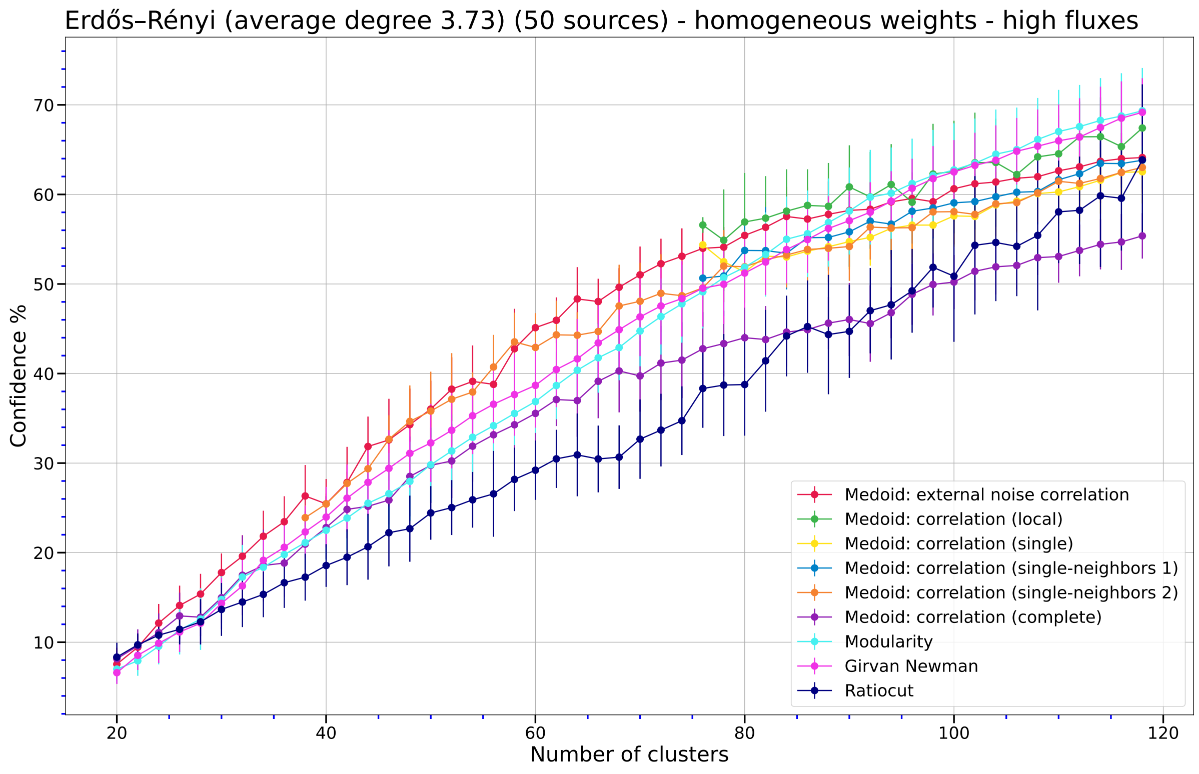

Erdős–Rényi (average degree 3.73) (5 sources) - homogeneous weights
Confidence %
Number of clusters
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut


Erdős–Rényi (average degree 3.73) (5 sources) - homogeneous weights - high fluxes
Confidence %
Number of clusters
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut


Erdős–Rényi (average degree 3.73) (50 sources) - uniform distribution
Confidence %
Number of clusters
0
10
20
30
40
50
60
70
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

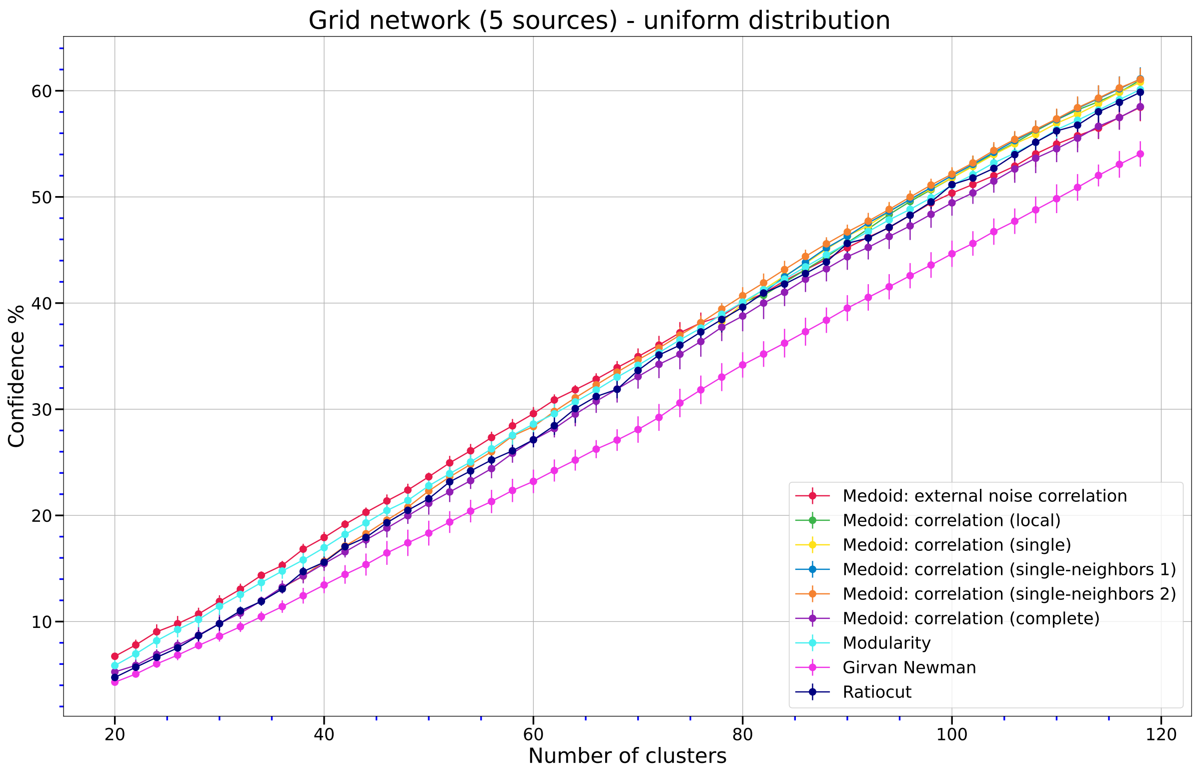

Erdős–Rényi (average degree 3.73) (50 sources) - uniform distribution - high fluxes
Confidence %
Number of clusters
10
20
30
40
50
60
70
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

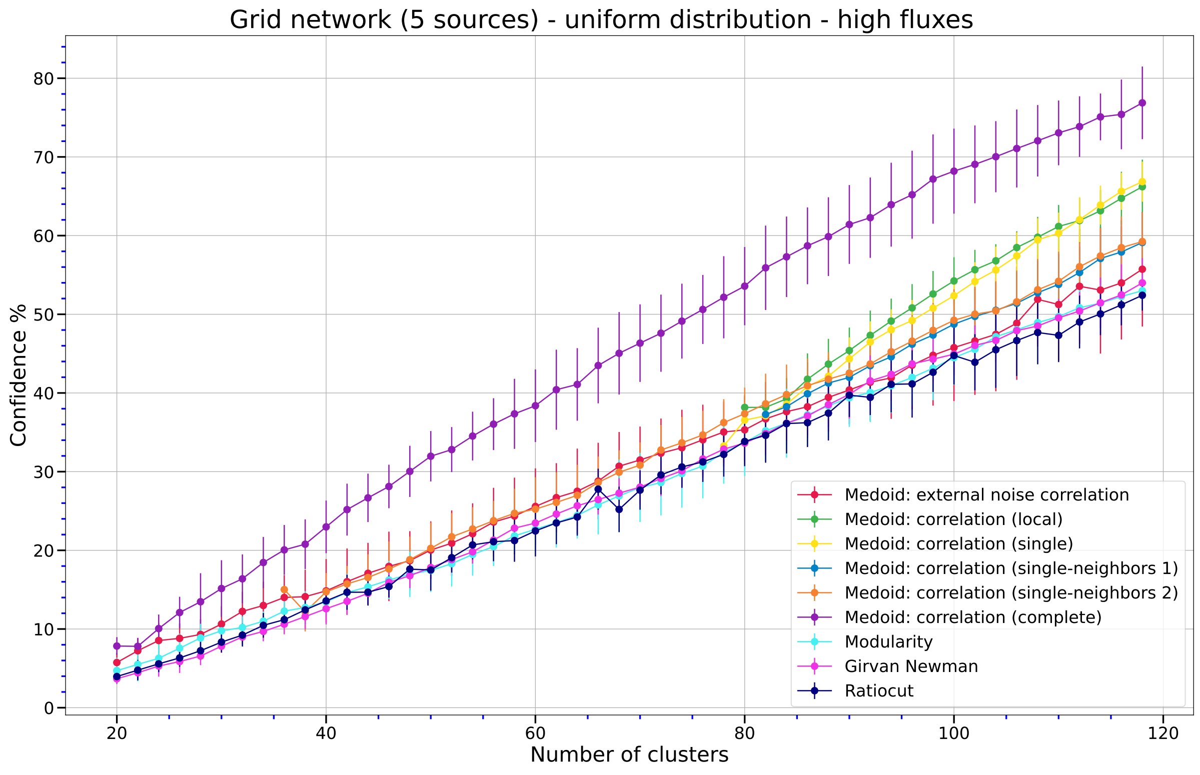

Erdős–Rényi (average degree 3.73) (50 sources) - homogeneous weights
Confidence %
Number of clusters
10
20
30
40
50
60
70
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

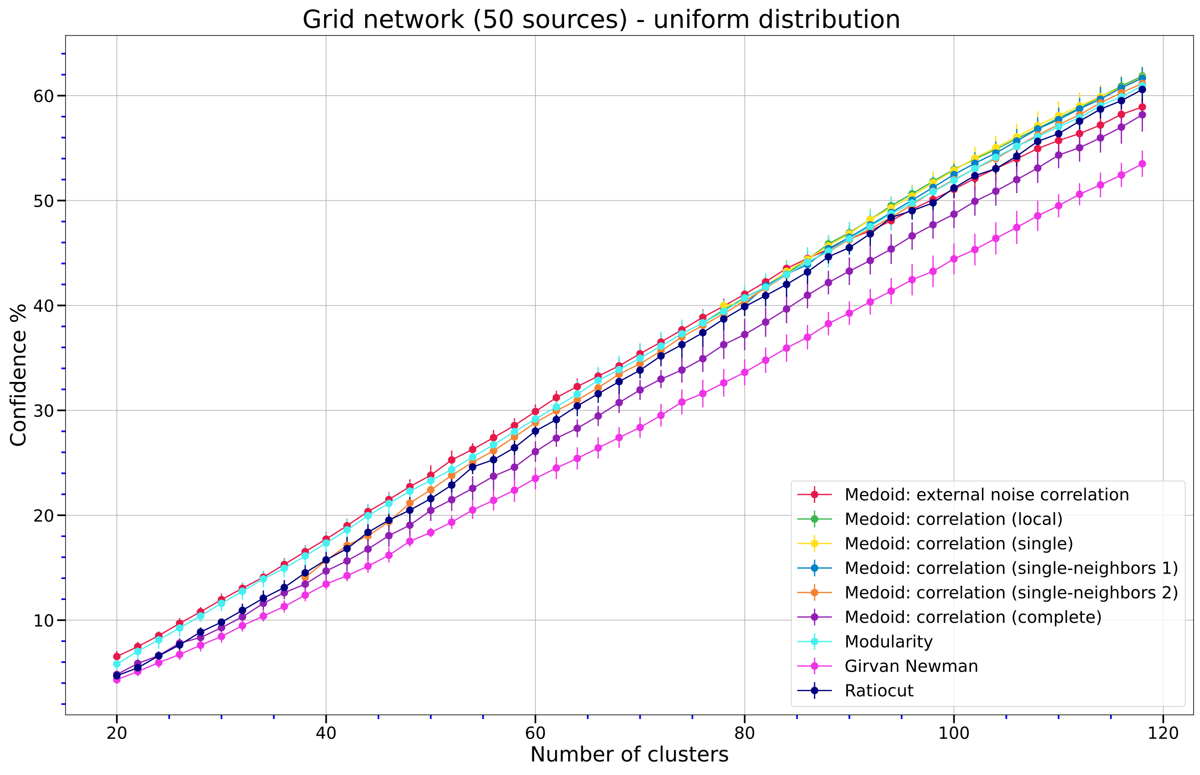

Erdős–Rényi (average degree 3.73) (50 sources) - homogeneous weights - high fluxes
Confidence %
Number of clusters
10
20
30
40
50
60
70
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

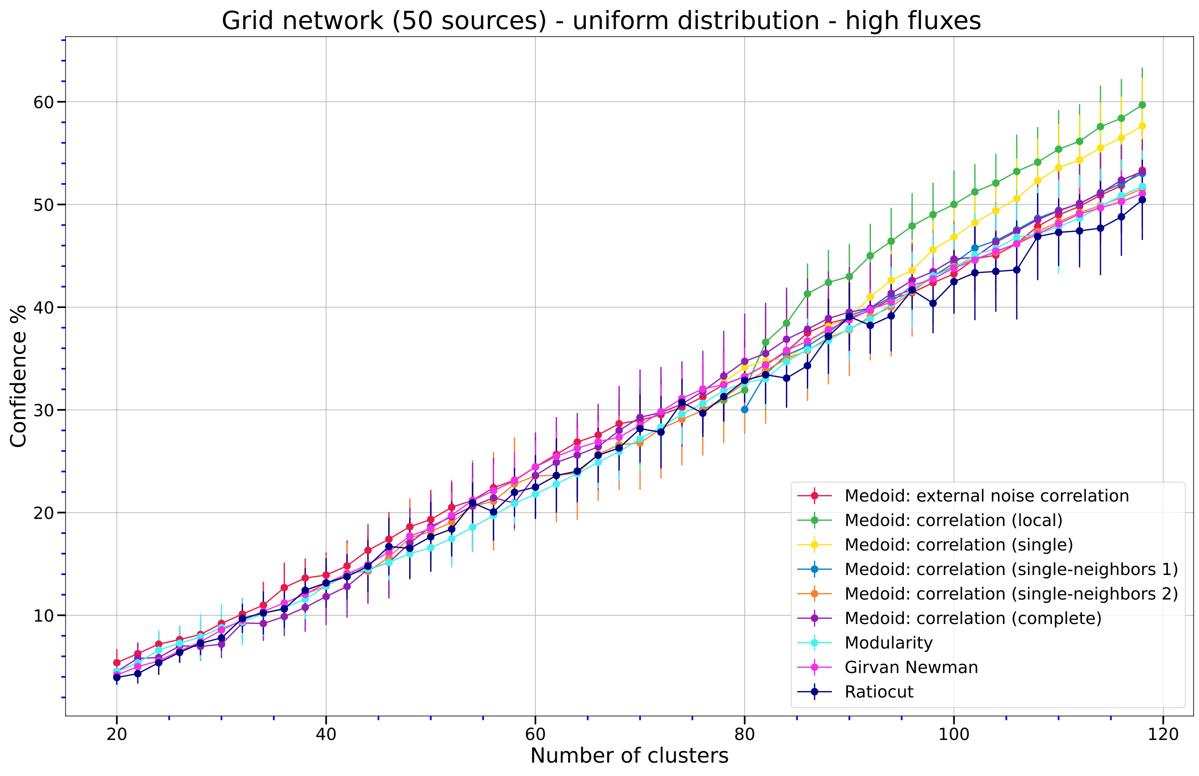
B.3 Grid networks

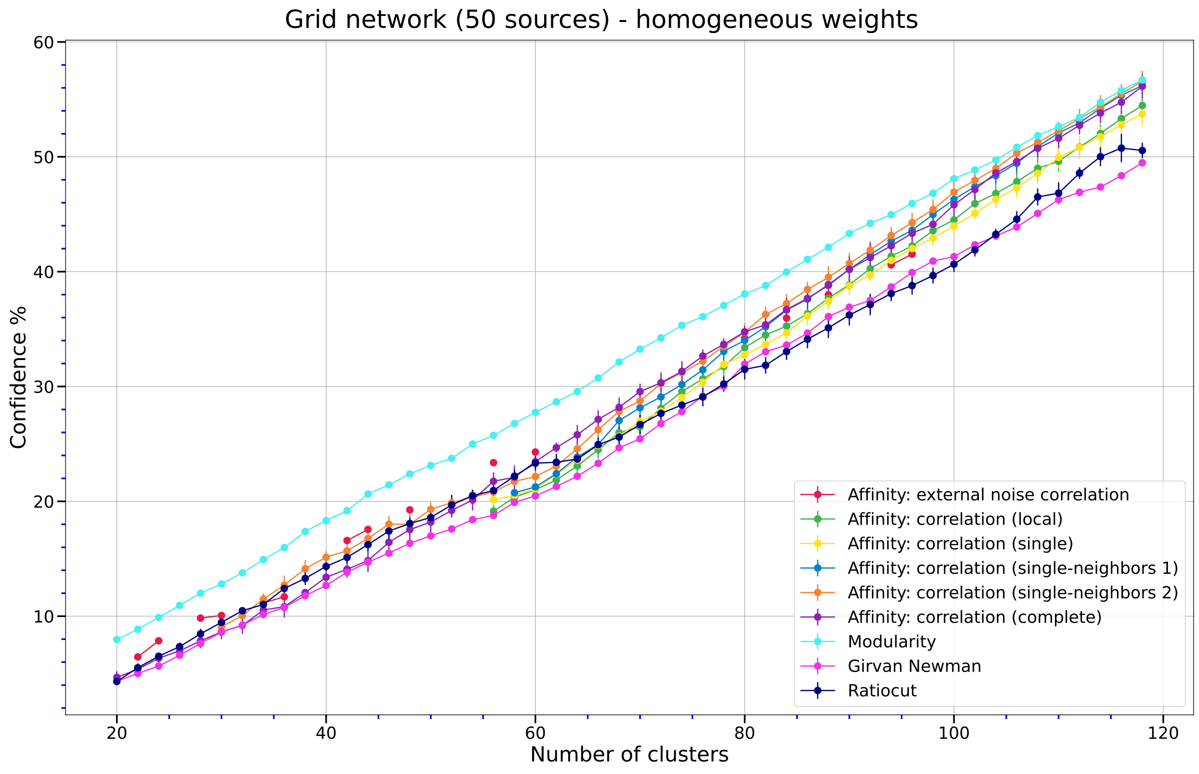
In this section, we show the simulations results for a grid network model that were not shown in chapter 4.

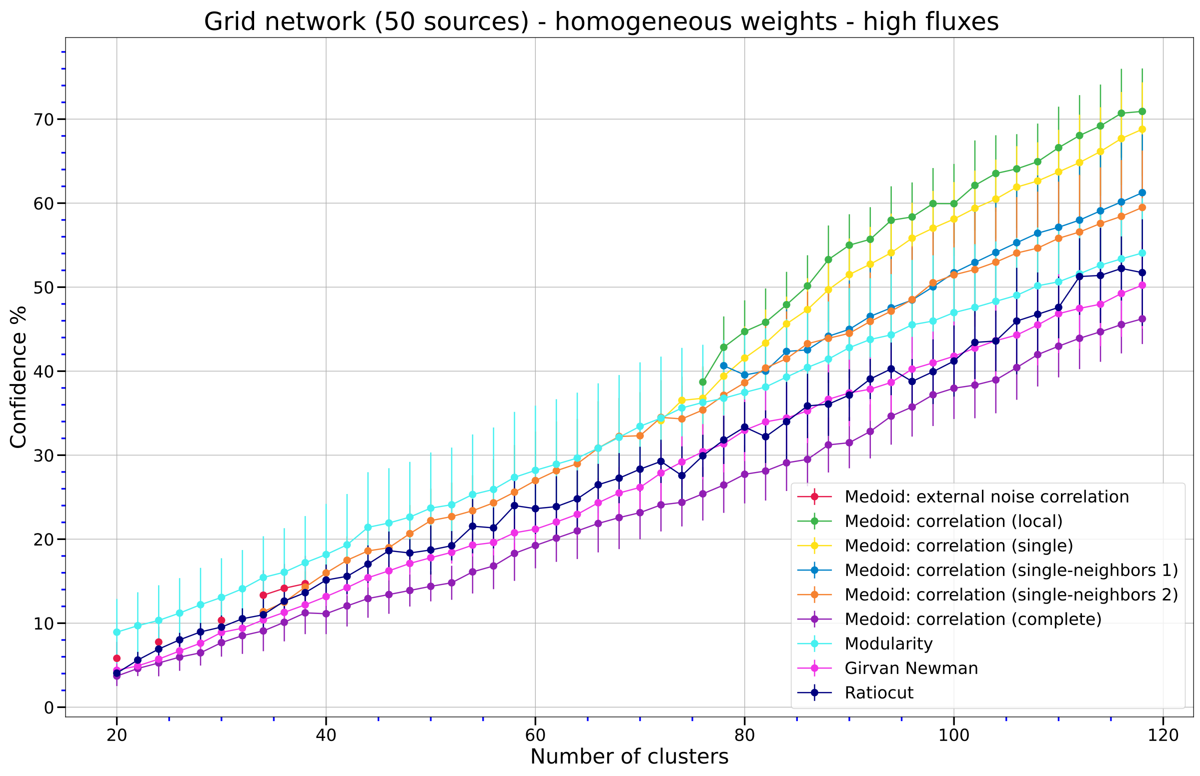



Grid network (5 sources) - uniform distribution - high fluxes
Confidence %
Number of clusters
0
10
20
30
40
50
60
70
80
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut


Grid network (50 sources) - uniform distribution
Confidence %
Number of clusters
10
20
30
40
50
60
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

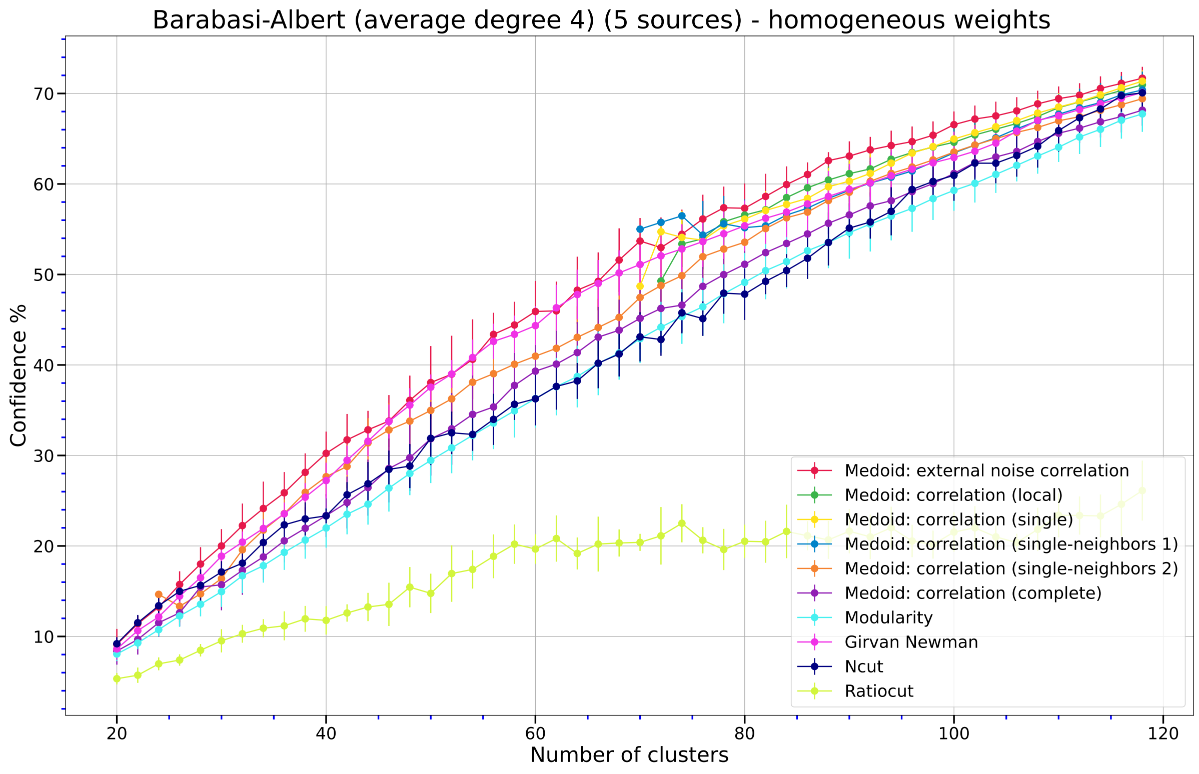

Grid network (50 sources) - uniform distribution - high fluxes
Confidence %
Number of clusters
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

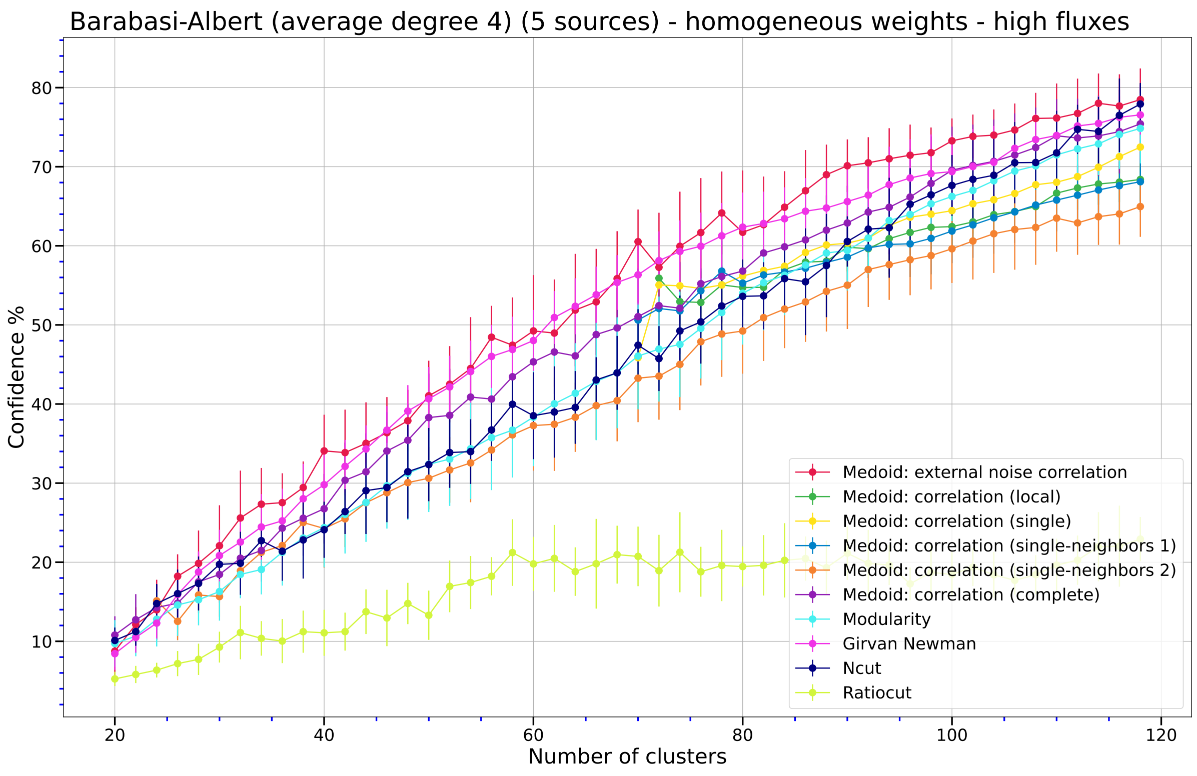

Grid network (50 sources) - homogeneous weights
Confidence %
Number of clusters
Affinity: external noise correlation
Affinity: correlation (local)
Affinity: correlation (single)
Affinity: correlation (single-neighbors 1)
Affinity: correlation (single-neighbors 2)
Affinity: correlation (complete)
Modularity
Girvan Newman
Ratiocut


Grid network (50 sources) - homogeneous weights - high fluxes
Confidence %
Number of clusters
0
10
20
30
40
50
60
70
20
40
60
80
100
120
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ratiocut

B.4 Barabasi-Albert networks

In this section, we shoe the simulation results for ten Barabasi-Albert networks with average degree equal to four, five sources, uniform distributions of the forcing and, differently form the main text, with a homogeneous distribution of the edge weights.




Barabasi-Albert (average degree 4) (5 sources) - homogeneous weights - high fluxes
Confidence %
Number of clusters
Medoid: external noise correlation
Medoid: correlation (local)
Medoid: correlation (single)
Medoid: correlation (single-neighbors 1)
Medoid: correlation (single-neighbors 2)
Medoid: correlation (complete)
Modularity
Girvan Newman
Ncut
Ratiocut

Bibliography

[1] Réka Albert and Albert-lászló Barabási. “Statistical mechanics of complex networks”. In: *Rev. Mod. Phys* (), p. 2002.

[2] D. Aldous and James Fill. “Reversible Markov Chains and Random Walks on Graphs”. In: (Jan. 2002).

[3] Alex Arenas, Albert Diaz-Guilera, and Conrad J. Perez-Vicente. “Synchronization Reveals Topological Scales in Complex Networks”. In: *Phys. Rev. Lett.* 96 (11 Mar. 2006), p. 114102. DOI: `10.1103/PhysRevLett.96.114102`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.96.114102`.

[4] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. *Dynamical Processes on Complex Networks*. Cambridge University Press, 2008. DOI: `10.1017/CBO9780511791383`.

[5] Armando Bazzani. “Linear differential equations and physical models”.

[6] Adi Ben-Israel and Thomas NE Greville. *Generalized inverses: theory and applications*. Vol. 15. Springer Science & Business Media, 2003.

[7] Burcu Bozkurt and Durmus Bozkurt. “On the sum of powers of normalized Laplacian eigenvalues of graphs”. In: *MATCH - Communications in Mathematical and in Computer Chemistry* 68 (Jan. 2012).

[8] Ulrik Brandes et al. “On Modularity - NP-Completeness and Beyond”. In: 2006.

[9] Chi-Tsong Chen. *Linear System Theory and Design*. 3rd. USA: Oxford University Press, Inc., 1998. ISBN: 0195117778.

[10] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[11] Aaron Clauset, M Newman, and Cristopher Moore. “Finding community structure in very large networks”. In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 70 (Jan. 2005), p. 066111. DOI: `10.1103/PhysRevE.70.066111`.

[12] Armando Di Nardo et al. “Sensor Placement in Water Distribution Networks based on Spectral Algorithms”. In: July 2018. DOI: `10.29007/whzr`.

[13] Xue Ding and Tiefeng Jiang. “Spectral distributions of adjacency and Laplacian matrices of random graphs”. In: *The Annals of Applied Probability* 20.6 (Dec. 2010).

[14] Paul L. Erdos and Alfréd Rényi. "On the evolution of random graphs". In: *Transactions of the American Mathematical Society* 286 (1984), pp. 257–257.

[15] *Failure identification for diffusion processes on networks.* 2022. URL: `https://github.com/EdoardoRolando/Failure-identification-for-diffusion-processes-on-networks`.

[16] Santo Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3-5 (Feb. 2010), pp. 75–174.

[17] Linton Freeman. "A Set of Measures of Centrality Based on Betweenness". In: *Sociometry* 40 (Mar. 1977), pp. 35–41. DOI: `10.2307/3033543`.

[18] Brendan J. Frey and Delbert Dueck. "Clustering by passing messages between data points". In: *Science* 315 (2007), p. 2007.

[19] C. W. Gardiner. *Handbook of stochastic methods for physics, chemistry and the natural sciences.* Vol. 13. Springer Series in Synergetics. Berlin: Springer-Verlag, 2004.

[20] Javad Ghaderi and R. Srikant. "Opinion dynamics in social networks with stubborn agents: Equilibrium and convergence rate". In: *Automatica* 50.12 (2014), pp. 3209–3215. ISSN: 0005-1098. DOI: `https://doi.org/10.1016/j.automatica.2014.10.034`. URL: `https://www.sciencedirect.com/science/article/pii/S0005109814004154`.

[21] M. Girvan and M. E. J. Newman. "Community structure in social and biological networks". In: *Proceedings of the National Academy of Sciences* 99.12 (2002), pp. 7821–7826.

[22] Ivan Gutman and Wenjun Xiao. "Generalized inverse of the Laplacian matrix and some applications". In: *Bulletin: Classe Des Sciences Mathematiques Et Natturalles* 129 (2004), pp. 15–23.

[23] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference.* Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. Pasadena, CA USA, 2008, pp. 11–15.

[24] L. Hagen and A.B. Kahng. "New spectral methods for ratio cut partitioning and clustering". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 11.9 (1992), pp. 1074–1085. DOI: `10.1109/43.159993`.

[25] David J. Hand. "Data Clustering: Theory, Algorithms, and Applications by Guojun Gan, Chaoqun Ma, Jianhong Wu". In: *International Statistical Review* 76.1 (Apr. 2008), pp. 141–141.

[26] Frank Harary and Robert Z Norman. *Graph theory as a mathematical model in social science*. 2. University of Michigan, Institute for Social Research Ann Arbor, 1953.

[27] *Harmonic Functions on Graphs*.

[28] Shigefumi Hata and Hiroya Nakao. "Erratum: Localization of Laplacian eigenvectors on random networks". In: *Scientific Reports* 7 (Dec. 2017). DOI: `10.1038/s41598-017-06298-6`.

[29] Petter Holme and Jari Saramäki. "Temporal networks". In: *Physics Reports* 519.3 (2012). Temporal Networks, pp. 97–125. ISSN: 0370-1573. DOI: `https://doi.org/10.1016/j.physrep.2012.03.001`. URL: `https://www.sciencedirect.com/science/article/pii/S0370157312000841`.

[30] John Hopfield. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities". In: *Proceedings of the National Academy of Sciences of the United States of America* 79 (May 1982), pp. 2554–8. DOI: `10.1073/pnas.79.8.2554`.

[31] B. W. Kernighan and S. Lin. "An efficient heuristic procedure for partitioning graphs". In: *The Bell System Technical Journal* 49.2 (1970), pp. 291–307. DOI: `10.1002/j.1538-7305.1970.tb01770.x`.

[32] Y. Kuramoto. *Chemical Oscillations, Waves, and Turbulence*. Dover Books on Chemistry Series. Dover Publications, 2003.

[33] *Lambert W-Function*. URL: `https://mathworld.wolfram.com/LambertW-Function.html`.

[34] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. "Signed Networks in Social Media". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Atlanta, Georgia, USA: Association for Computing Machinery, 2010, pp. 1361–1370. ISBN: 9781605589299. DOI: `10.1145/1753326.1753532`. URL: `https://doi.org/10.1145/1753326.1753532`.

[35] Ulrike von Luxburg. "A tutorial on spectral clustering". In: *Statistics and Computing* 17.4 (Dec. 2007), pp. 395–416.

[36] Naoki Masuda, Mason A. Porter, and Renaud Lambiotte. "Random walks and diffusion on networks". In: *Physics Reports* 716-717 (2017). Random walks and diffusion on networks, pp. 1–58.

[37] Patrick N. McGraw and Michael Menzinger. "Laplacian spectra as a diagnostic tool for network structure and dynamics". In: *Phys. Rev. E* 77 (3 Mar. 2008), p. 031102. DOI: `10.1103/PhysRevE.77.031102`. URL: `https://link.aps.org/doi/10.1103/PhysRevE.77.031102`.

[38] Russell Merris. “Laplacian matrices of graphs: a survey”. In: *Linear Algebra and its Applications* 197-198 (1994), pp. 143–176. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(94)90486-3. URL: https://www.sciencedirect.com/science/article/pii/0024379594904863.

[39] M. E. J. Newman. “Modularity and community structure in networks”. In: *Proceedings of the National Academy of Sciences* 103.23 (2006), pp. 8577–8582. DOI: 10.1073/pnas.0601602103. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.0601602103. URL: https://www.pnas.org/doi/abs/10.1073/pnas.0601602103.

[40] Mark Newman. *Networks*. Oxford University Press, 2018, 2018. ISBN: 0192527495.

[41] Mark Newman and Michelle Girvan. “Finding and Evaluating Community Structure in Networks”. In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 69 (Mar. 2004), p. 026113. DOI: 10.1103/PhysRevE.69.026113.

[42] Neave O’Clery et al. “Observability and coarse graining of consensus dynamics through the external equitable partition”. In: *Phys. Rev. E* 88 (4 Oct. 2013), p. 042805. DOI: 10.1103/PhysRevE.88.042805. URL: https://link.aps.org/doi/10.1103/PhysRevE.88.042805.

[43] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[44] Martin Rosvall and Carl T. Bergstrom. “Maps of random walks on complex networks reveal community structure”. In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123. DOI: 10.1073/pnas.0706851105. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.0706851105. URL: https://www.pnas.org/doi/abs/10.1073/pnas.0706851105.

[45] Martin Rosvall et al. “Different Approaches to Community Detection”. In: (2019), pp. 105–119.

[46] Naoki Saito. “How Can We Naturally Order and Organize Graph Laplacian Eigenvectors?” In: (2018), pp. 483–487. DOI: 10.1109/SSP.2018.8450808.

[47] Michael Schaub et al. “Structured networks and coarse-grained descriptions: a dynamical perspective”. In: (Apr. 2018).

[48] Jianbo Shi and J. Malik. “Normalized cuts and image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.8 (2000), pp. 888–905. DOI: 10.1109/34.868688.

[49] D. I. Shuman et al. “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains”. In: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98.

[50] Daniel A. Spielman. “Spectral and Algebraic Graph Theory Incomplete Draft”.

[51] James Stone. *Information Theory: A Tutorial Introduction.* Feb. 2015. ISBN: 978-0956372857. DOI: `10.13140/2.1.1633.8240`.

[52] T. Tao. *[PDF] topics in random matrix theory: Semantic scholar.* Jan. 1970. URL: `https://www.semanticscholar.org/paper/Topics-in-Random-Matrix-Theory-Tao/611503ed9d36bad843374774c59ab335ebf7eac4`.

[53] Pat Vatiwutipong and Nattakorn Phewchean. "Alternative way to derive the distribution of the multivariate Ornstein–Uhlenbeck process". In: *Advances in Difference Equations* 2019 (July 2019). DOI: `10.1186/s13662-019-2214-1`.

[54] Xiangrong Wang et al. "Improving robustness of complex networks via the effective graph resistance". In: *The European Physical Journal B* 87 (Sept. 2014), pp. 1–12. DOI: `10.1140/epjb/e2014-50276-0`.

[55] Duncan Watts et al. "Collective dynamics of 'small world' networks". In: Jan. 2006, pp. 301–303.

[56] Yen-Chuen Wei and Chung-Kuan Cheng. "Towards efficient hierarchical designs by ratio cut partitioning". In: (1989), pp. 298–301. DOI: `10.1109/ICCAD.1989.76957`.

[57] G. C. Wick. "The Evaluation of the Collision Matrix". In: *Phys. Rev.* 80 (2 Oct. 1950), pp. 268–272. DOI: `10.1103/PhysRev.80.268`. URL: `https://link.aps.org/doi/10.1103/PhysRev.80.268`.