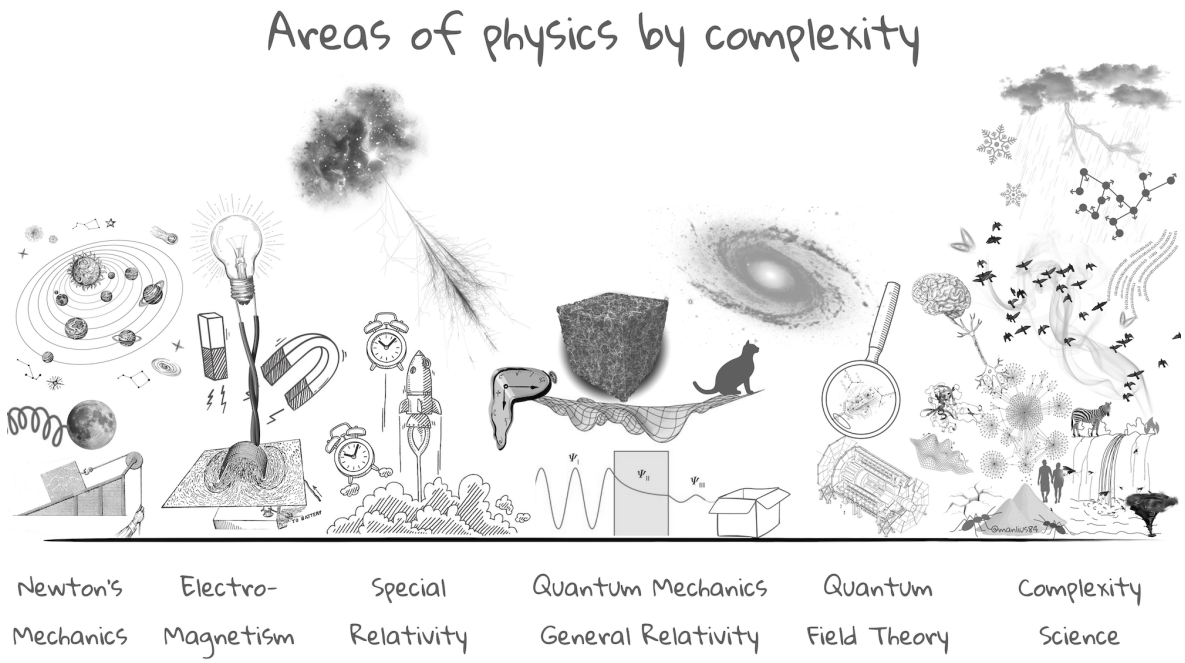
Università degli Studi di Padova



Physics of Complex Networks: Structure and Dynamics

Final Report

Student: Zara Miriam,

Last update: July 9, 2024

Contents

1 | Task 18: Turing Patterns

1.1 | Task Description

The goal of this task is to analyze a Turing activator-inhibitor dynamics on networks, with specific suggestion to replicate the findings of [1]. A system governed by a reaction-diffusion mechanism can exibit pattern emergence when perturbed from an initial linearly stable equilibrium state. The conditions for pattern initiation are found through linear stability analysis. The subsequent evolution of the pattern is non-linear and eventually results in a steady state (that can be either stationary or time dependent).
A note: The authors of [1] only state the conditions that must hold for pattern formation on the network. In fact, they can be derived with only a few minor changes in the same way as one does for pattern formation in a continuous medium. However, if the reader is not familiar with the traditional Turing patterns in continuous space (like me before this work), those relations are not at all evident. I wanted to get a true understanding of what I was going to simulate. That is why I studied in detail the case of the continuous medium and then I derived the conditions stated by the authors of [1]. However, this meant that, to keep the workload manageable, I was able to focus only on the analysis of the **pattern initiation** stage and disregard the anaylis of the multistability and histeresis effects done in the second part of the article.

1.2 | Mathematical Model

1.2.1 | Definitions

A network-organized activator-inhibitor system can be defined in very close analogy to what one does for continuous media. In the network case, equations for the system are:

$$\begin{cases} \dot{u}_i(t) = f(u_i\, v_i) - \epsilon\, [L\, u(t)]_i \\ \dot{v}_i(t) = g(u_i\, v_i) - \epsilon\, \sigma [L\, v(t)]_i \end{cases}$$

Here, $\mathbf{u} = (u_1, \cdots u_N)$ and $\mathbf{v} = (v_1, \cdots v_N)$ are, respectively, the concentrations of the activator substance and the inhibitor substance on the N nodes of the network. The reactions take place locally, on each node, and are encoded in the reaction terms $f(u_i\, v_i)$ and $f(u_i\, v_i)$. L is the laplacian matrix, defined as $L = D - A$. The diffusivity of the activator species is ϵ, while that of the inhibitor is $\epsilon\, \sigma$, so that σ is the ratio between them.

So far, the above are just the general equations of a reaction-diffusion system.

In order to have a Turing mechanism, the reaction term need to satisfy the following basic requirements:

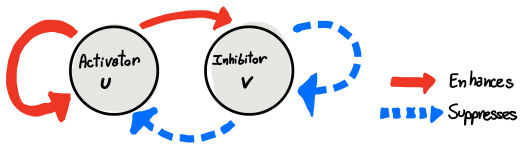
1. Existence of a homogeneous equilibrium $(\overline{u}, \overline{v})$ which is linearly stable in absence of diffusion (indeed, the key idea of the Turing model is that the instability is driven by diffusion, and appears only above a certain threshold function of the diffusion parameters) :

$$\begin{pmatrix} u_i(t) \\ v_i(t) \end{pmatrix} \equiv \begin{pmatrix} \overline{u} \\ \overline{v} \end{pmatrix} \quad \forall i \quad \text{where} \quad f(\overline{u}\,, \overline{v}) = g(\overline{u}\,, \overline{v}) = 0 \quad \text{and, given that}$$

$$J(\overline{u}\,, \overline{v}) := \begin{pmatrix} f_u & f_v \\ g_u & g_v \end{pmatrix}, \quad \begin{cases} \mathrm{tr}(J) = f_u + g_v < 0 \\ \det(J) = f_u \cdot g_v f_v \cdot g_u > 0 \end{cases} \quad \text{(see 1.3)}$$

2. Correct qualitative behaviour of reactions in the neighborhood of the fixed point $(\overline{u}, \overline{v})$:
the activator u is supposed to enhance its own production and the production of the inhibitor v. Viceversa, the inhibitor v is supposed to suppress the production of both the activator u and itself. The functions f, v need to reflect this behaviour, at least in a neighbourhood of the equilibrium $(\overline{u}, \overline{v})$. Mathematically:

$$\begin{cases} f_u > 0, & f_v < 0 \\ g_u > 0, & g_v < 0 \end{cases}$$



These requirements are just the same as in the case for the continuous medium.

1.2.2 | Conditions for diffusion-driven instability

Diffusion driven instability is investigated by means of linear stability analysis. A small random perturbation $\delta\, u_i$, $\delta\, v_i$ is added at each node, starting from the homogeneous equilibrium state. A linearized system of equations is obtained for the evolution of the perturbation:

$$\begin{cases} u_i(t) = \overline{u} + \delta\, u_i(t) \\ v_i(t) = \overline{v} + \delta\, v_i(t) \end{cases} \rightarrow \begin{cases} \delta\, \dot{u}_i(t) \simeq & f_u\, \delta u_i(t) + f_v\, \delta v_i(t) - \epsilon\, [L \cdot (\overline{u} + \delta u(t))]_i \\ \delta\, \dot{v}_i(t) \simeq & g_u\, \delta u_i(t) + g_v\, \delta v_i(t) - \epsilon\, \sigma\, [L(\overline{v} + \delta v(t))]_i \end{cases} \tag{1.1}$$

But $L\,\overline{u} = L\,\overline{v} = 0$, since the constant vectors u and v are eigenvectors of the laplacian with eigenvalue zero. In the case of a continuous medium, the perturbation is expanded as a Fourier series of plane waves. The rationale of this is that it makes the PDE turn into an eigenvalue problem. In the network case, we instead write the perturbation as:

$$\begin{pmatrix} \delta u_i(t) \\ \delta v_i(t) \end{pmatrix} = \begin{pmatrix} 1 \\ B_n \end{pmatrix} \cdot \sum_{n=1}^{N} c_n\, \Phi_i^{(n)}\, e^{\lambda_n\, t}$$

Where $\Phi^{(n)}$ is the $n-th$ eigenvector of the laplacian matrix L and its corresponding eigenvalue is Λ_n $(0 = \Lambda_1 \leq \Lambda_2 \leq \cdots \Lambda_N)$. The coefficients $\{c_n\}$ are determined by

the initial conditions ($t = 0$). By doing this, the system of equations is turned into a $\Lambda-$dependent eigenvalue problem. In fact, if we plug the ansaltz

$$\begin{pmatrix} \delta u_i(t) \\ \delta v_i(t) \end{pmatrix} = \begin{pmatrix} 1 \\ B_n \end{pmatrix} \cdot c_n \, \Phi_i^{(n)} \, e^{\lambda_n \, t}$$

into the linearized system of equations, we get:

$$\begin{pmatrix} \delta \dot{u}_i \\ \delta \dot{v}_i \end{pmatrix} = \begin{pmatrix} f_u - \epsilon \, \Lambda_n & f_v \\ g_u & g_v - \epsilon \, \sigma \, \Lambda_n \end{pmatrix} \cdot \begin{pmatrix} \delta u_i \\ \delta v_i \end{pmatrix}$$

$$\rightarrow \lambda_n \cdot \begin{pmatrix} 1 \\ B_n \end{pmatrix} = \begin{pmatrix} f_u - \epsilon \, \Lambda_n & f_v \\ g_u & g_v - \epsilon \, \sigma \, \Lambda_n \end{pmatrix} \cdot \begin{pmatrix} 1 \\ B_n \end{pmatrix}$$

Or, in compact form:

$$\lambda_n \, \mathbf{v}_n = M(\Lambda_n, \, \sigma, \, \epsilon) \, \mathbf{v}_n$$

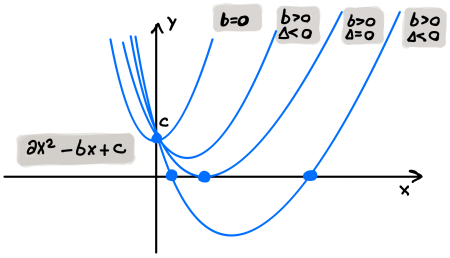
The differences between the network case and the continuous medium case end here. From now on, the steps are exactly the same. We are looking for instability, thus we want to find the range of parameters $(\sigma, \, \epsilon)$ that produce $\mathcal{R}e\{\lambda_{\pm}(\Lambda)\} > 0$ for a positive range of Λ's.
First, we must have that at least one of the following holds

$$\begin{cases} \det[M(\Lambda, \, \sigma, \, \epsilon)] < 0 \\ \mathrm{tr}[M(\Lambda, \, \sigma, \, \epsilon)] > 0 \end{cases}$$

But $\mathrm{tr}[M(\Lambda, \, \sigma, \, \epsilon)] = \mathrm{tr}[J] - \epsilon \, \Lambda \, (1 + \sigma) < 0$ because the homogeneous state is a stable equilibrium, then the only possibility is that $\det[M(\Lambda, \, \sigma, \, \epsilon)] < 0$.
Some quick boring algebra now:

$$\det[M(\Lambda, \, \sigma, \, \epsilon)] = \epsilon^2 \, \sigma \, \Lambda^4 - \epsilon \, (g_v + \sigma \, f_u) \Lambda + \det[J] \overset{!}{\leq} 0 \quad \text{for some} \, \Lambda > 0$$

This is a parable of kind $y = a \, x^2 - b \, x + c$ with $(a, \, b, \, c > 0)$, so the minimum is reached at coordinates $(x_{min}, \, y_{min}) = (\frac{b}{2 \, a} \, , c - \frac{b^2}{4 \, a})$. The parable touches the x axis $(y_{min} \leq 0) \iff \Delta^2 = b^2 - 4 \, a \, c \geq 0$.



We require:

$$\begin{cases} x_{min}(\epsilon, \, \sigma) > 0 \\ y_{min}(\epsilon, \, \sigma) \leq 0 \end{cases} \qquad \begin{cases} \epsilon \, (g_v + \sigma \, f_u) > 0 \\ [\epsilon \, (g_v + \sigma \, f_u)]^2 \geq 4 \, \epsilon^2 \, \sigma \, \det[J] \end{cases}$$

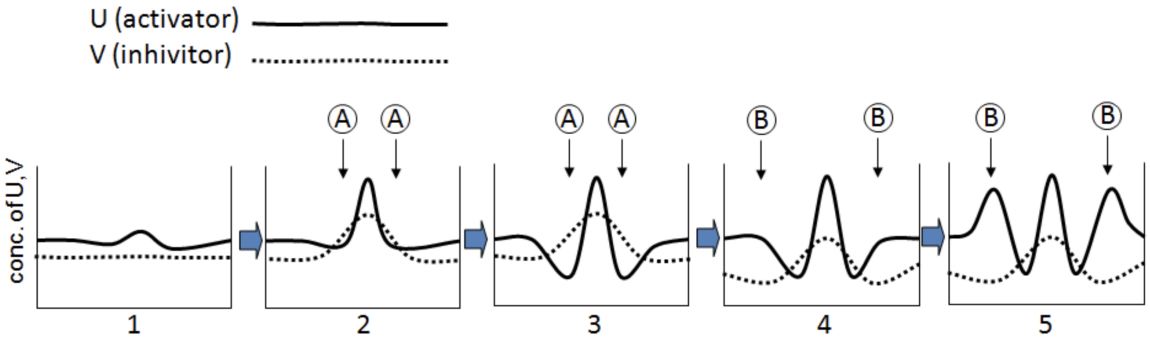


Figure 1.1: Graph 1 shows the initial condition of the system. Suppose that the concentration of the activator is relatively higher than in other regions by random fluctuation. By the self-enhancing property of the activator, the concentration of activator increases at the center region (graph2), followed by the increase of inhibitor at the neighboring region A. As the diffusion rate of inhibitor is much larger than that of the activator, substantial amounts of inhibitor move toward the lateral regions. This depresses the activator function, resulting in the decrease of the activator concentration there (graph3). Decrease of activator causes the decrease of inhibitor in the wider region (graph4). At the region B, as inhibitor concentration is gotten lower, activator becomes relatively dominant than inhibitor. This situation is enough to start the local self activation at region B (graph5). [Figure taken from 2, Supplementary Information].

From the first one, we notice that there exists a minimum value threshold for $\sigma > \sigma_{min} = \frac{|g_v|}{f_u}$. Also, since $\mathrm{tr}\,[J] = (-|g_v| + f_u) < 0$, then $\sigma_{min} > 1$: a necessary (but still not sufficient) condition for pattern initiation is that the inhibitor diffuses faster than the activator. Indeed, this difference between the diffusion constants of the two species is key to have patterns induced by diffusion (see Figure 1.1.)

$$\begin{cases} \sigma > \sigma_{min} > 1 \\ f_u^2\,\sigma^2 + 2\,(f_u\,g_v - 2\det[J])\,\sigma + g_v^2 \geq 0 \end{cases} \qquad \begin{cases} \sigma > \sigma_{min} > 1 \\ \sigma < \sigma_-(< \sigma_{min}) \quad \text{or} \quad \sigma > \sigma_+ \end{cases}$$

Where $\sigma_\pm$ are the roots of the quadratic equation:

$$\sigma_\pm = \frac{(f_u\,g_v - 2\,f_v\,g_u) \pm 2\sqrt{f_v\,g_u\,(f_v\,g_u - f_u\,g_v)}}{f_u^2}$$

The function $y_{min}(\sigma)$ reaches its maximum for $\sigma = \sigma_{min}$ and goes to $-\infty$ for both $\sigma \to 0^+$ and $\sigma \to +\infty$. Also, the lower-branch root σ_- is below the threshold value σ_{min}. Then, both our requirements are satisfied if and only if $\sigma > \sigma_+$. In summary, the necessary and sufficient condition for instability is

$$\sigma > \sigma_c := \frac{(f_u\,g_v - 2\,f_v\,g_u) + 2\sqrt{f_v\,g_u\,(f_v\,g_u - f_u\,g_v)}}{f_u^2}$$

Critical eigenvalue and graph finite size effect
The critical eigenvalue Λ is determined for give ϵ:

$$\Lambda_c(\epsilon) := \Lambda(\sigma = \sigma_c,\,\epsilon) = \frac{g_v + \sigma_c\,f_u}{2\,\epsilon\,\sigma_c} = (\cdots) = \frac{1}{\epsilon}\sqrt{\frac{\det[J]}{\sigma_c}} \tag{1.2}$$

For σ above the critical threshold, a finite range $[\Lambda_1\, \Lambda_2]$ centered around Λ_c appears where the corresponding mode is unstable. However, the eigenvalue spectrum of the laplacian matrix is not continuous. Instability will appear if there are allowed modes that fall inside this range. In particular, the eigenvalue spectrum of the laplacian is bounded above for a graph of finite size. If the mobility ϵ is small enough, the critical eigenvalue could lay beyond the largest eigenvalue of the graph, and instability would not occur. This finite-size effect appears also in the case of a continuous medium. There, instability cannot occur when the spatial domain is too small and the largest allowed wavelength is less than the critical wavelength.

Dispersion relation $\lambda(\Lambda;\, \sigma,\, \epsilon)$
Finally, the growing rate of the unstable mode, $\lambda_n = \lambda(\Lambda_n)$ is calculated from the caractheristic polynomial of $M(\Lambda_n, \sigma,\, \epsilon)$:

$$p(\lambda) = (\lambda - \lambda_1) \cdot (\lambda - \lambda_2) = \lambda^2 - \mathrm{tr}[M(\Lambda,\, \sigma,\, \epsilon)] \cdot \lambda + \det[M(\Lambda,\, \sigma,\, \epsilon)]$$

$$\begin{aligned}\lambda_\pm &= \frac{1}{2}\left[\mathrm{tr}[M] \pm \sqrt{\mathrm{tr}[M]^2 - 4\det[M]}\right] = \frac{1}{2}\left[-|\mathrm{tr}[M]| \pm \sqrt{\mathrm{tr}[M]^2 - 4\det[M]}\right] \\ &= \frac{1}{2}\left[[f_u + g_v - (1+\sigma)\,\epsilon\,\Lambda] + \sqrt{4\,f_v\,g_u + [f_u - g_v - (1-\sigma)\,\epsilon\,\Lambda]^2}\right]\end{aligned}$$

In the instability region, $\mathrm{tr}[M] < 0$ and $\det[M] < 0$, then eigenvalues are real and distinct. Also, the lower branch is always negative so we are not interested in it. By comparison with the graph of $\det[M(\Lambda,\, ;\epsilon,\, \sigma]$, we get the qualitative dependence of λ from Λ, commonly called "dispersion relation".

1.3 | Numerical Simulations

As in [1], the reaction terms are set as

$$\begin{cases} f(u,v) = \left[\frac{a+b\,u-u^2}{c} - v\right] u \\ g(u,v) = \left[u - (1+d\,v)\right] \cdot v \end{cases} \tag{1.3}$$

with parameters $a = 35$, $b = 16$, $c = 9$, $d = 2/5$, helding a linearly stable fixed point $(\overline{u}, \overline{v}) = (5, 10)$ and a critical diffusion ratio $\sigma_c \simeq 15.5$.

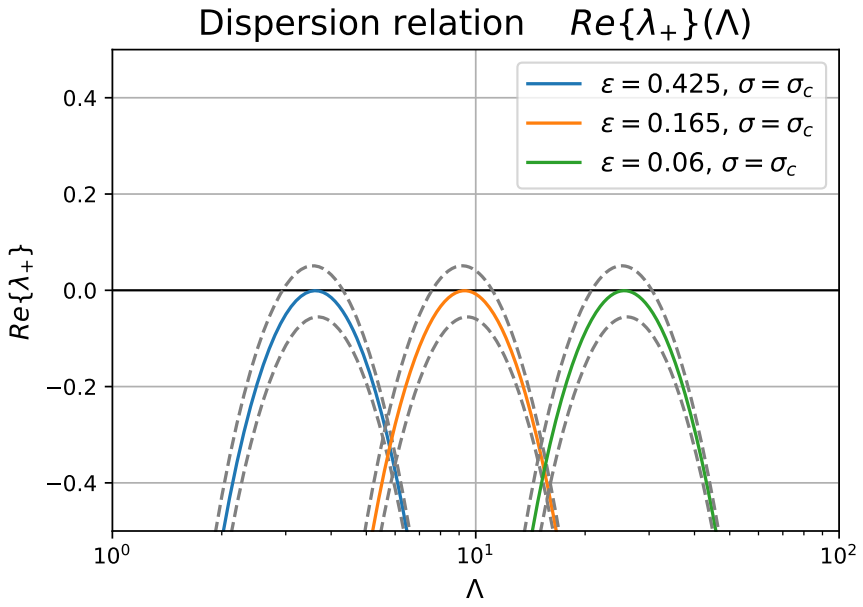
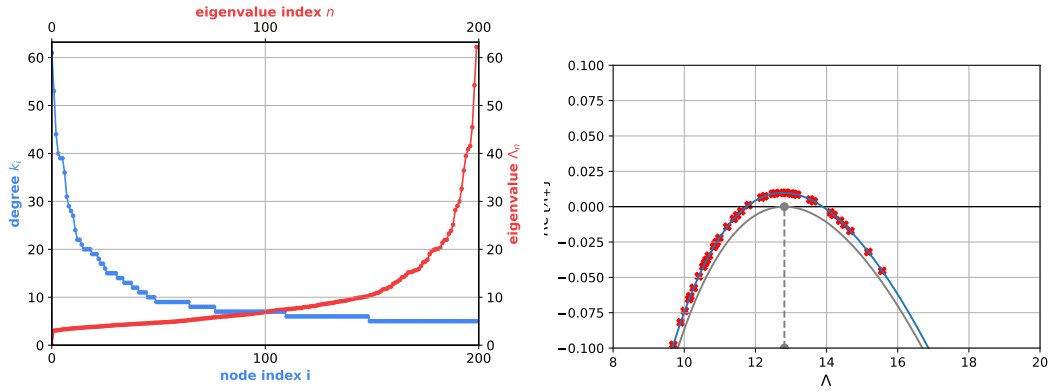


Figure 1.2: Dispersion relation for the chosen reaction kinetics [Eq. 1.3], at different values of the activator diffusivity ϵ. The dashed grey lines are the dispersion curves slightly above and below the critical diffusivity ratio σ_c. As stated in [Eq. 1.2], the critical eigenvalue Λ_c is inversely proportional to the activator diffusivity ϵ.

Following the choice of the authors, I simulated the dynamics on Barabasi-Albert scale free networks. I chose parameters $N = 200$ for the number of nodes and $k = 10$ for the mean degree. My activator diffusivity was $\epsilon = 0.12$, and my diffusivity ratio was $\sigma = 15.6 > \sigma_c = 15.5$. The reasoning for chosing a diffusivity ratio that is only slightly above the critical threshold is to keep the number of different allowed modes low.



(a) (b)

Figure 1.3: Most relevant properties of the network used in the simulation. Subfigure (a) shows the degree spectrum and the laplacian eigenvalue spectrum of the chosen graph. Network nodes are indexed by decreasing degree. Subfigure (b) shows the dispersion relation curve. Critical eigenvalue is $\Lambda_c \simeq 12.8141$. The eigenvalue closest to Λ_c is $\Lambda_n \simeq 12.64$, of index $n = 157$. However, as one can see from the graph, the network eigenvalue spectrum comprehends several other allowed modes (red crosses) in the unstable range, and they are expected to contribute to pattern initiation as well.

The initial homogeneous state $(\overline{u}, \overline{v})$ was perturbed with a random uniform perturbation at each node of amplitude 0.05. [Figure 1.4] show snapshots of the activator concentration on nodes at different times. Also, the GithHub repository [3] contains a .mp4 video of the whole evolution.

In the early stage, the pattern is expected to grow proportionally to the critical eigenvector, which is is the mode of largest growing rate: $\delta u(t)$, $\delta v(t) \propto \Phi^{(n)}$. In fact, when the initial uniform perturbation dies out and the pattern starts to grow, the activator substance distribution is located similarly to the critical eigenvector (see snapshot $t = 25.06$). Later on, the evolution becomes strongly non-linear and the pattern is reshaped (see snapshot at $t = 100.25$), till it settles into a stationary state (see snapshot at $t = 150.38$) where nodes are split into one activator-rich group and one activator-poor group. The authors found that this peculiar behaviour is well described in the framework of a mean field theory [1].

An interesting thing one can notice in the initial stage pattern [Figure 1.4, snapshot at $t = 25.06$] is that the significant variations of the activator level are localized on a subset of nodes of close degree ($k \sim 25 - 75$). That is, only a specific subset of the network undergoes differentiation. This effect is was not a fortuity but is due to the fact that the laplacian eigenvectors in a large graph with a broad degree distribution tend to be localized around a specific degree $\overline{k}(\Lambda)$. Also, as authors report, a simple relation $\overline{k}(\Lambda) \simeq \Lambda$ seems to hold. I checked wether I could find the same behaviour for my graph (see [Figure 1.5] and [Figure 1.6].

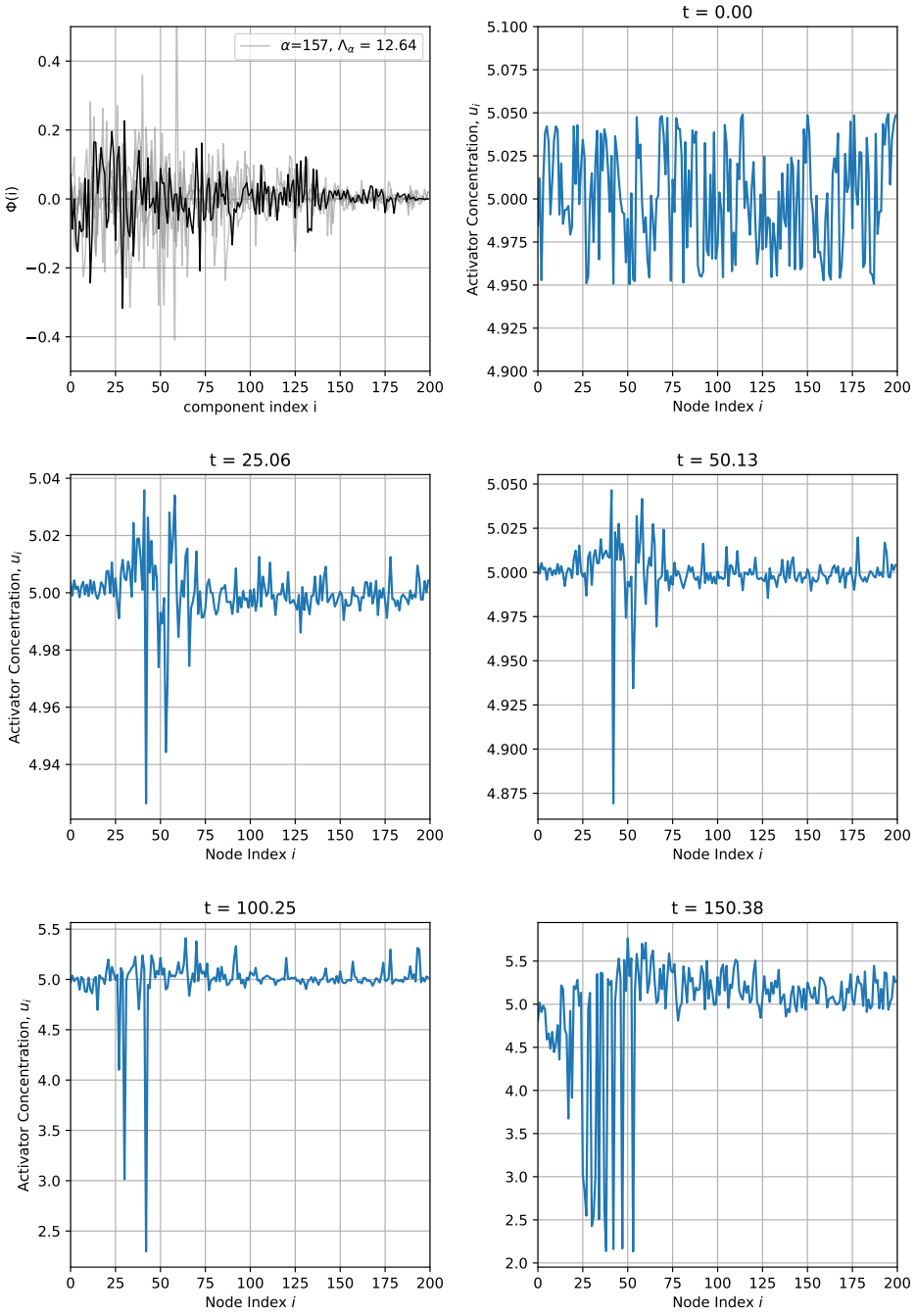


Figure 1.4: Time evolution of the activator concentration. The subfigure at top left corner shows the components of the critical eigenvector (in black) and its closest neighbours in the unstable range (in grey).

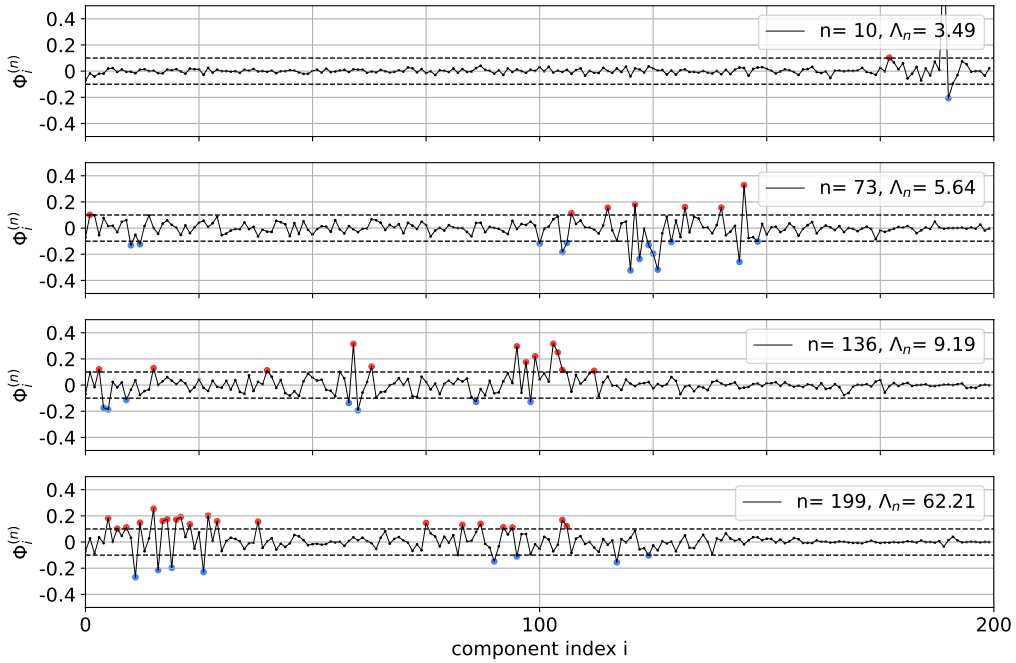
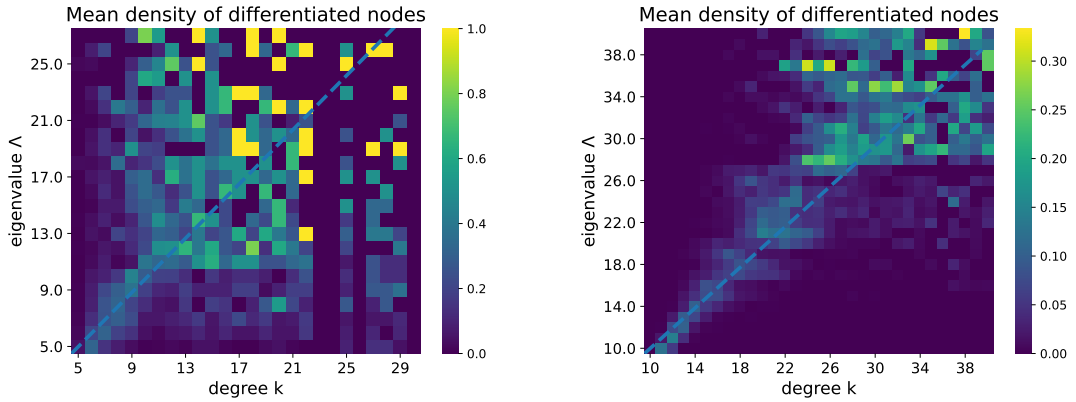


Figure 1.5: Localization of laplacian eigenvectors in a BA with $N = 200$ and $k = 10$. Nodes nodes are ranked by their degree. With incrementing eigenvalue Λ, the characteristic degree becomes higher.



(a) (b)

Figure 1.6: Localization of laplacian eigenvectors in a BA graph with $N = 200$ nodes and $k = 10$ (a) and in a BA graph with $N = 1000$ nodes and $k = 20$ (b). Eigenvalues have been grouped into bins of unitary width. The population of nodes inside each degree group, N_k, was calculated. The heatmap represents the mean density of differentiated nodes for each degree group $z = N_k^{\text{diff}}(\Lambda)/N_k$. A node i of degree was considered to be differentiated with respect to eigenvalue Λ_n if its eigenvector component satisfied, $|\Phi_i^{(n)}| > 0.1$. I found, like authors [1] report, that approximately $\overline{k}(\Lambda) \propto \Lambda$ (dashed line) and that effect is more pronounced in the larger graph.

Appendix

Linear Stability of a 2x2 autonomous system Say $(\overline{x}, \overline{y}) \in R^2$ is a fixed point (or equilibrium) of the autonomous system

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \end{pmatrix} = \begin{pmatrix} f[x(t),\, y(t)] \\ g[x(t),\, y(t)] \end{pmatrix}$$

which, namely, means that $f(\overline{x}, \overline{y}) = g(\overline{x}, \overline{y}) = 0$. Now we want to study how the system evolves if a small perturbation $(\delta x, \delta y)$ is added to the equilibrium. We can linearize the system:

$$\begin{pmatrix} \delta\, \dot{x}(t) \\ \delta\, \dot{y}(t) \end{pmatrix} = \begin{pmatrix} \frac{\delta f}{\delta x}|_{(\overline{x},\overline{y})} & \frac{\delta f}{\delta y}|_{(\overline{x},\overline{y})} \\ \frac{\delta g}{\delta x}|_{(\overline{x},\overline{y})} & \frac{\delta g}{\delta y}|_{(\overline{x},\overline{y})} \end{pmatrix} \cdot \begin{pmatrix} \delta\, x(t) \\ \delta\, y(t) \end{pmatrix} + o(||(\delta\, x, \delta\, y)||)$$

The latter is a homogeneous linear system with general solution:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \alpha_1 \mathbf{v}_1\, e^{\lambda_1\, t} + \alpha_2 \mathbf{v}_2\, e^{\lambda_2\, t}$$

where λ_1, λ_2 are the eigenvalues of the jacobian and $\mathbf{v}_1\, \mathbf{v}_2$ are the corresponding eigenvectors, and α_1, α_2 are coefficients determined by the initial condition. The eigenvalues of a $2 \cdot 2$ matrix can be both real or complex cooniugates ($\lambda_1 = \lambda$, $\lambda_2 = \overline{\lambda}$). Wether the amplitude of the perturbation dies out exponentially or explodes depends on the sign of $\mathcal{R}e\{\lambda\}$. The linear stability requirement is

$$\mathcal{R}e\{\lambda\} < 0 \quad \text{for both } \lambda\text{'s}$$

The eigenvalues λ_1, λ_2 are the roots of the characteristic polynomial of the jacobian J:

$$p_J(\lambda) = (\lambda - \lambda_1) \cdot (\lambda - \lambda_2) = \lambda^2 - \mathrm{tr}[J] \cdot \lambda + \det[J]$$

Then we have the following cases, depending on the discriminant $\Delta^2 = \mathrm{tr}[J]^2 - 4\det[J]$:

$$\begin{cases} \Delta^2 > 0: & \lambda_1,\, \lambda_2 = \frac{\mathrm{tr}[J] \pm \Delta}{2} \quad \text{real, distinct} \\ \Delta^2 = 0: & \lambda_1, = \lambda_2 = \frac{\mathrm{tr}[J]}{2} \in R \\ \Delta^2 < 0: & \lambda_1, = \lambda_2 = \alpha \pm i\beta \in C \end{cases}$$

One can easily see from the polynomial $p_J(\lambda)$ that in the case $\Delta^2 \geq 0$, the requirement that both roots are negative is satisfied if and only if $\mathrm{tr}[J] < 0$, $\det[J] > 0$. In the case $\Delta < 0$, we have obligatorily $\det[J] = \lambda \cdot \overline{\lambda} = |\lambda|^2 > 0$, and $\mathrm{tr}[J] = 2 \cdot \mathcal{R}e(\lambda)$. Then again the requirement holds if and only if $\mathrm{tr}[J] < 0$.

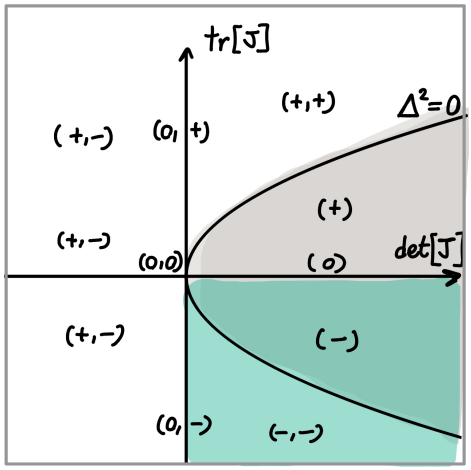


Figure 1.7: Bifurcation diagram for a $2 \cdot 2$ homogeneous system. $(\pm, \pm)$ are the signs of $\mathcal{R}e\{\lambda\}$ in the various regions of the (tr[J], det[J]) plane. Green marks the stability region and Grey marks the region where eigenvalues are complex.

2 | Task 28: Voter Model

2.1 | Task Description

The voter model is perhaps the simplest and most studied model of cooperative behaviour. While its behaviour on regular lattices of arbitrary dimention d is known in detail, it is much more difficult to get a comprehnsive general picture of its behaviour on a complex network, where there is interplay between many factors, such as the degree distribution, the effective dimentionality, the degree of disorder, the presence of correlations [6]. The aim of this task is to describe the voter model and attempt to reproduce its behaviour on some class of networks. Specifically, I chose to focus on scale free networks, for which some analitical results have been found [7].

2.2 | Model

In the voter model, each node of the network is in one of two possible states, say, spin up or spin down: $\sigma_i \in \{\pm 1\}$. The evolution starts with some random configuration of up and down spins. The update rule is defined as follows:

i) choose one node at random,
ii) choose one of its neighbors at random,
iii) ascribe to this node the current state of its neighbour.

Node-update rule

Actually, there is an alternative rule which may look equivalent:

i) choose one link at random,
ii) choose one of its ends at random,
iii) ascribe to this node the current state of the other end.

Link-update rule

Indeed the two rules are equivalent for regular lattices, but not in the case of complex networks [8]. For each time step, this update rule is applied N times if N is the network size, so that *on average* each node is updated once. The voter model defines a markovian stochastic process. There are two absorbing states: all spins up and all spins down (when it reaches one of these states, it cannot change anymore, that is why they are called absorbing). If we identify network nodes as people and spin up and down with two possible opinion, the absorbing state represent consensum.
There is always a chance that a finite size network reaches consensus. In fact, **the mean time to reach consensus in a finite network is always finite** (the mean is here intended over the set of all evolution histories and initial conditions), whether it is a regular lattice or a complex network. However, conceptually there are different scenarios when infinite size networks are considered instead.

The mean time τ to reach consensus in a finite, regular lattice of N nodes and dimention d is known to scale as:

$$\tau\left(N\right) \sim \begin{cases} N^2 & \text{if} \quad d = 1 \\ N \cdot \ln N & \text{if} \quad d = 2 \\ N & \text{if} \quad d > 2 \end{cases}$$

For networks, one could guess to find $\tau\left(N\right) \sim N$ as for high dimentional lattices. However this is not true in general. For instance, [7] found analitically that, for uncorrelated, scale-free networks with degree distribution $P(k) \sim k^{-\gamma}$ and *node-update* rule:

$$\tau\left(N\right) \sim \begin{cases} N^{\alpha},\ \alpha < 1 & \text{if} \quad \gamma < 3 \\ \frac{N}{\ln N} & \text{if} \quad \gamma = 3 \\ N & \text{if} \quad \gamma > 3 \end{cases}$$

Generically, $\tau(N)$ grows sublinearly with N; that is, high-degree nodes greatly accelerate the approach to consensus. [6] numerically found for Barabasi-Albert networks a scaling $\tau \sim N^{0.88}$, which is compatible with $\tau \sim \frac{N}{\ln N}$, and a scaling $\tau \sim N$ (like for the high dimentional lattices) when the *edge-update* rule is used instead.
A standard order parameter used to measure the ordering process of the voter dynamic is the *average interface density* ρ, defined as the density of edges connecting nodes with different states:

$$\rho(\sigma) := \frac{\sum_{i,j}\left[A_{i,j} \cdot \mathbb{1}(\sigma_i,\, \sigma_j)\right]}{\sum_{i,j} A_{i,j}}, \quad \text{where } \mathbb{1}(\sigma_i,\, \sigma_j) := \frac{1 - \sigma_i\, \sigma_j}{2} = \begin{cases} 1 & \text{if} \quad \sigma_i \neq \sigma_j \\ 0 & \text{if} \quad \sigma_i = \sigma_j \end{cases} \tag{2.1}$$

2.3 | Simulation results

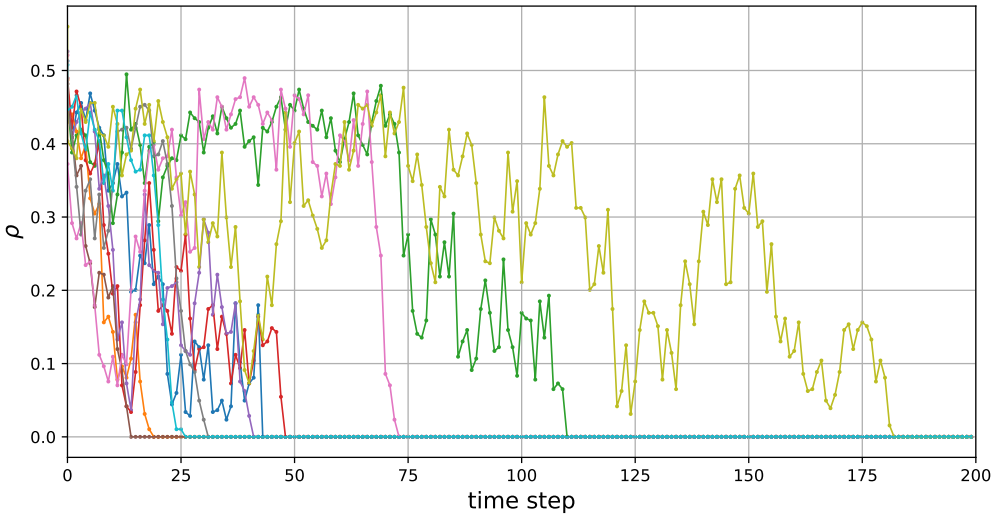


Figure 2.1: Evolution of the voter model on Barabasi-Albert networks with $N = 100$ and $k = 8$. Each line is a trajectory on a different network instance of the BA model. Initial state is drawn uniformly at random for each trajectory.

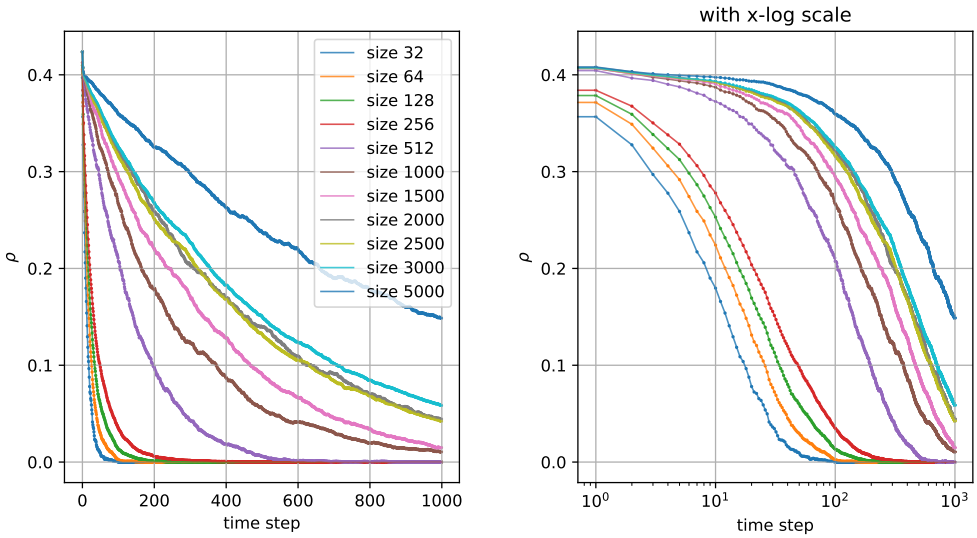


Figure 2.2: Evolution of the average interface density [Eq. **??**] in Barabasi Albert networks of different sizes (see legend) and mean degree $k = 6$. Data is averaged over 500 realizations.

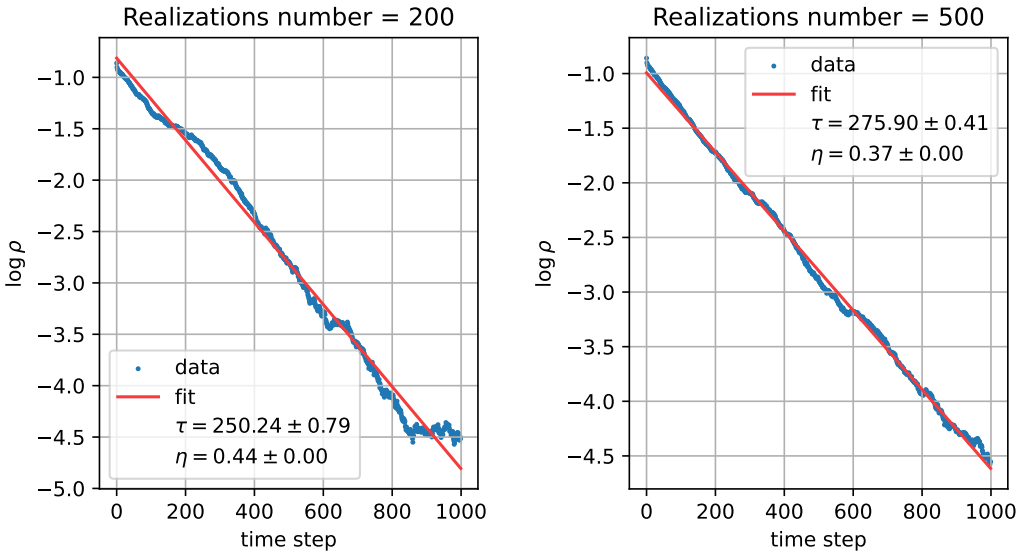


Figure 2.3: Comparison between exponential fit parameters estimated from 200 (left) and 500 (right) different realization on BA networks with same size $N = 1000$.

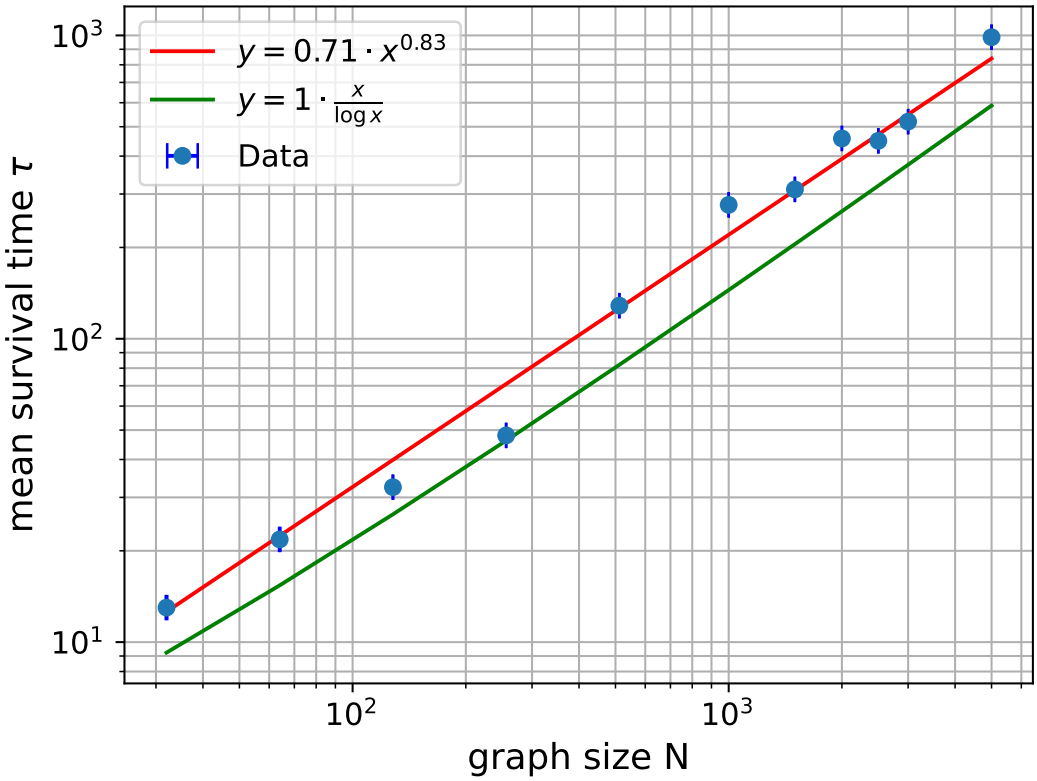


Figure 2.4: Average survival times for Barabasi Albert networks of different sizes, as results from the exponential fit of the curves in [Fig: 2.2]. Data is fitted with a power law $y = \alpha \cdot x^{\gamma}$ with free parameters α, γ with the method of least squares implemented in **scipy.optimize.curve_fit** from Phyton library Scipy [9]. The best estimation for fit parameters are $\gamma = 0.8312 + / - 0.0007$, $\alpha = 0.707 + -0.004$. The fit curve is red. For comparison, also the theoretical expectation $\tau \sim \frac{N}{\log N}$ is plotted. One can see that the slope of the two curves are very similiar. To make $x/logx$ fit the data, one simply needs to add a prefactor, which corresponds to an intercept in the log log scale of the plot presented here.

3 | Task 01: Ising Model

3.1 | Task Description

4 | Task 46: EU transportation network II

4.1 | Task Description

The aim of this project is to build a network from raw geographical data of European railways.

The data used for the analysis is sourced from the *EuroGlobalMap (EGM)* database, which is published under an open license by EuroGeographics. EuroGeographics is a not-for-profit membership association for European National Mapping, Cadastral, and Land Registration Authorities (NMCAs), in partnership with the National Geographic Institute (NGI) Belgium. The latest data is available at mapsforeurope.org. For this analysis, we refer to the EGM 2019 dataset, released in March 2019.

The requested output of Project #46 consists of two **.csv** files for each country with an ISO3 code starting from IT onwards (in alphabetical order) and for the whole of Europe:

- **country_nodes.csv**, which shall contain the columns (nodeID, nodeLabel, latitude, longitude, country_name, country_ISO3). Node identifiers must be sequential integer numbers starting from 1.
- **country_edges.csv**, containing edges list with columns (nodeID_from, nodeID_to, node_label_from, node_label_to).

The choice of what exactly the nodes of the network should represent (cities, administrative districts, etc.) is not specifically prescribed. We decided to construct networks where **nodes are identified with cities** and **two cities are connected through an edge if and only if they are consecutive stops of some railway**. After some preliminary trials (and failures!), we discovered that the quickest and most effective method to build the city/rails network involves first creating an initial "raw" network. In this raw network, each straight segment of rails is represented as a pair of nodes connected by an edge. Subsequently, the city/rails network is built as a subset of this raw network.

With this approach in mind, Section [1 - Data Extraction] is organized as follows:

- In Paragraph 1.1, we describe the method used to read the raw data from the EGM19 database and present a preliminary visualization of this data.
- In Paragraph 1.2, we outline the main steps for creating the "raw" networks.

- In Paragraph 1.3, we detail the main steps for constructing the city/rails network.

Section [2 - Results Visualization] contains graphical visualization of the results.

4.2 | Data extraction

4.2.1 | Reading the data

The database *EGM19* is provided with documentation files *EGM19_User_Guide.pdf* and *EGM19_DataSpecification.pdf*. We consulted the former to get a dictionary of correspondences between country IS03 codes and country names (see Annex C) and the latter to gain information about the raw data (see Annex C - *Definition of Features and Attributes*).
The raw data is provided in various formats: we opted for **.shp** shapefile format which we read using the Python library **GeoPandas**. The required imput files are:

- *RailrdC.shp*, *RailrdC.shx*, *RailrdC.dbf*, containing data of railway stations,
- *RailrdL.shp*, *RailrdL.shx*, *RailrdL.dbf*, containing data of railway lines.

The basic command to open a shapefile with GeoPandas is

```
stations = gpd.read_file("RailrdC.shp")
railways = gpd.read_file("RailrdL.shp")
```

This command creates a GeoPandaDataFrame with various columns. The column **geometry** contains the spatial information, encoded as **LineString** and **Point** objects. Coordinates are latitude and longitude, expressed in degrees. A LineString is a series of connected line segments.

```
print(type(rails.geometry[0]), rails.geometry[0])
    #<class 'shapely.geometry.linestring.LineString'>
       LINESTRING (0.5972559999997884
       43.64944749999984, 0.5974139999997874
       43.650735499999854, 0.5974145699997874
       43.66748499999854)
print(type(stations.geometry[0]), stations.geometry[0])
# <class 'shapely.geometry.point.Point'> POINT
   (14.470226499999796 47.57296499999984)
```

Other columns contain data attributes. For stations, the attributes we need are: 'ICC': the 2- character country IS03 code (es. IT, for Italy) and 'NAMA1' : the station name in first national language, written in the international alphabet (ex: "Ancona"). For railways, they are: 'ICC', same as before and 'EXS' : existence cathegory (0: 'Unknown', 5: 'Under Construction', 6: 'Abandoned/Disused', 28: 'Operational', -32768 : 'Invalid Value').

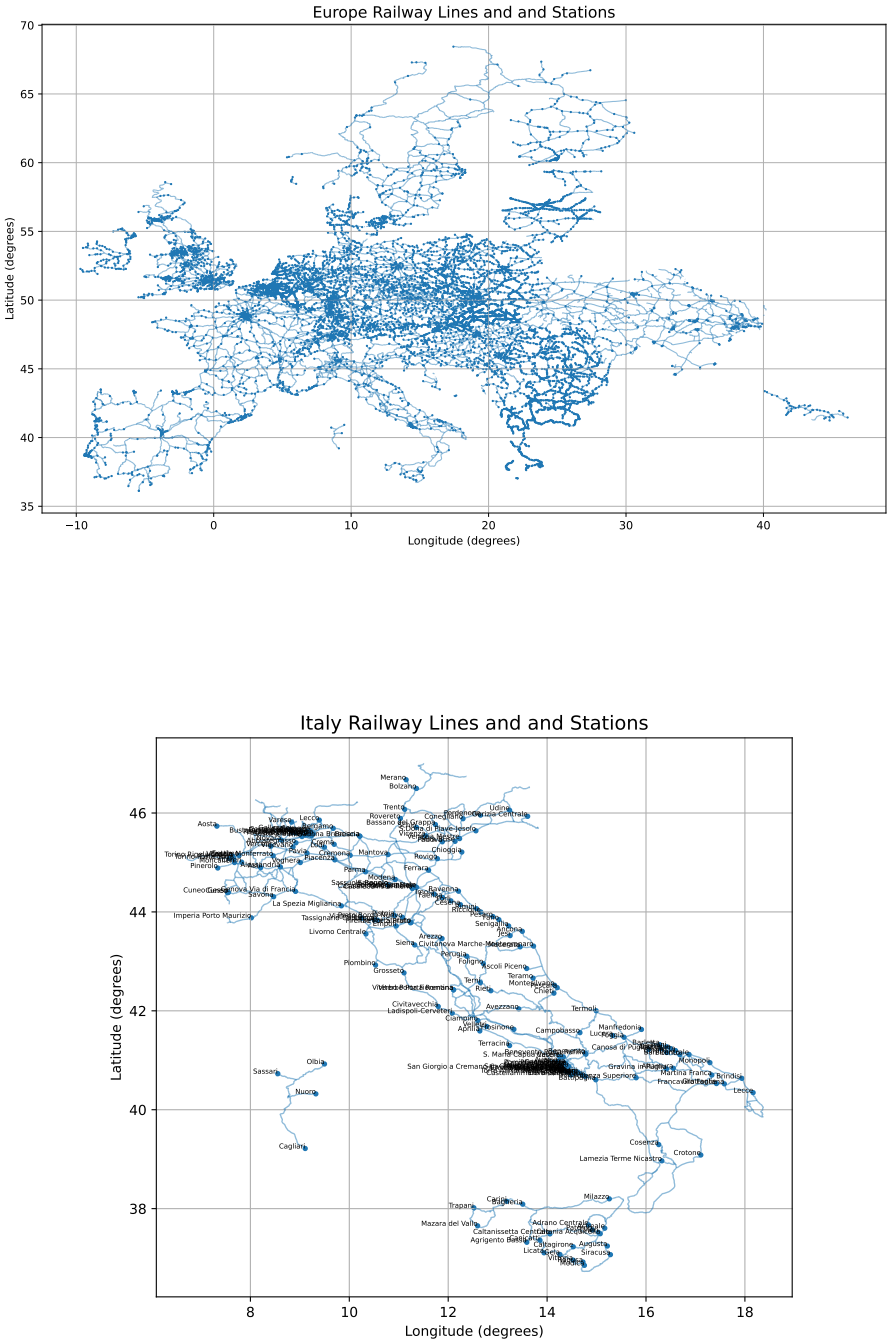


Figure 4.1: **Top**: Railways and stations of the whole Europe. Each dot (line, respectively) is an element of the dataframe stations (railways, respectively). Only operational lines (EXS code = 28) have been filtered out. **Bottom**: A zoom on Italy. We can already see by eye that data is incomplete (for instance,stations of Verona, Trieste and Roma are missing form the chart). Also, in the metropolitan area of Napoli there is a high density of stations, because also the smallest ones are reported in the database, whereas for other areas of Italy (like Veneto) only the biggest stations are reported. We cannot thus expect to recover a very accurate network of railways from this dataset.

4.3 | Creation of the "raw" networks

To create the raw graph we used Python library **NetworkX**. In the raw graph, each straight segment of rails, corresponding to two consecutive items in a LineString object, is assigned to an edge. The starting point and ending point of the segment are assigned to nodes. Identifiers of nodes are sequential integer numbers starting from 1. A helper function **add_node** is defined to handle node numbering properly and avoid adding the same node to the graph more than once (which, anyway, seems unlikely having such high precision on coordinates... but never say never!).

Finally, the graph data is stored as node list with attributes **"ICC_raw_nodes.csv"** and edge list **"ICC_raw_edges.csv"** via basic networkx and pandas methods.

4.4 | Creation of the city/railways networks

The creation of the raw network was straightforward; however, the refined network required a few specific steps:

1. Data was read from "RailrdC.shp" to create a list of city nodes**"ICC_city_nodes.csv"**. This file contains the columns: nodeID, nodeLabel, latitude, longitude, country_name, and country_ISO3.
2. The raw graph data "ICC_raw_nodes.csv" and "ICC_raw_edges.csv" is recovered. Attributes 'label' and 'is near city' are added to raw nodes. A raw node that matches a city's coordinates in "ICC_city_nodes.csv" within a specified threshold distance was assigned either "label" = "city_name" and "is_near_city" = "empty" (if it was the best match) or "label" = "empty" and "is_near_city" = "city_name".
3. The city/rails graph was derived as a subset of the current graph: for each pair of labeled nodes ('label' $\neq$ 'empty') that are within a maximum radius, the shortest paths were calculated. If a shortest path existed where all intermediate nodes were neither cities nor dangerously close to other cities, an edge was added to the final output graph.

Then, the graph is created and saved with usual routines. Final output files are **"ICC_nodes.csv"** and **"ICC_nodes.csv"**.

The introduction of the "is_near_city" field was necessary because rail bifurcations often start slightly before or after entering a city station. In reality, the train must pass through the city station even if it takes the bifurcation. Without the "is_near_city" field, non-existent edges would be erroneously created (see figure 4.2). The maximum radius parameter for checking edges only between sufficiently close cities was crucial for speeding up computations, especially in countries with a high number of stations and densely connected railways, such as Poland. Both threshold parameter for city matching in **flexible_matching** and maximum radius have to be tuned after looking at the raw data.

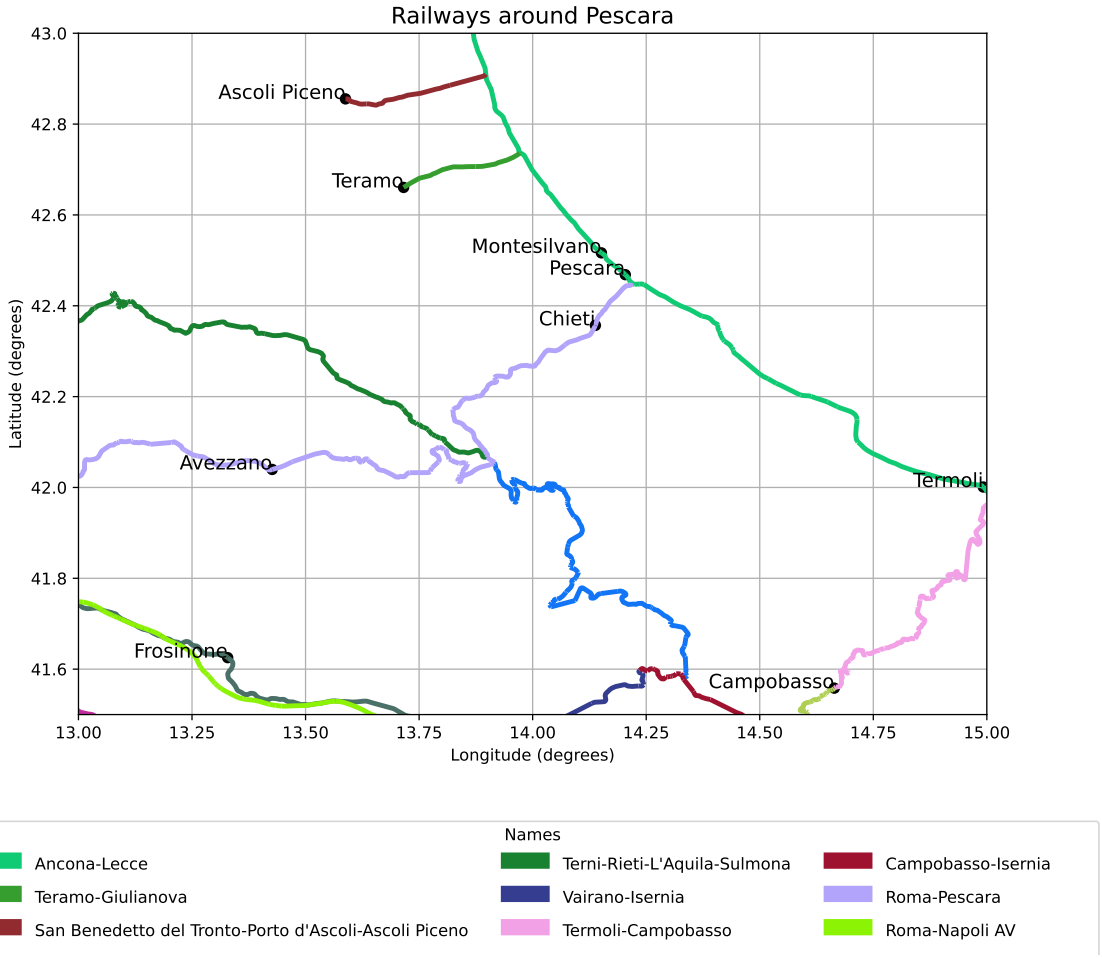


Figure 4.2: An example to justify the need for the "is near city" check: a rail bifurcation starts just before entering Pescara from south. Without the check implementation, edge "Termoli-Chieti" is created whereas in reality it does not exist: one wanting to get to Chieti must first get to Pescara, then take another train to Chieti.

4.5 | Results Visualization

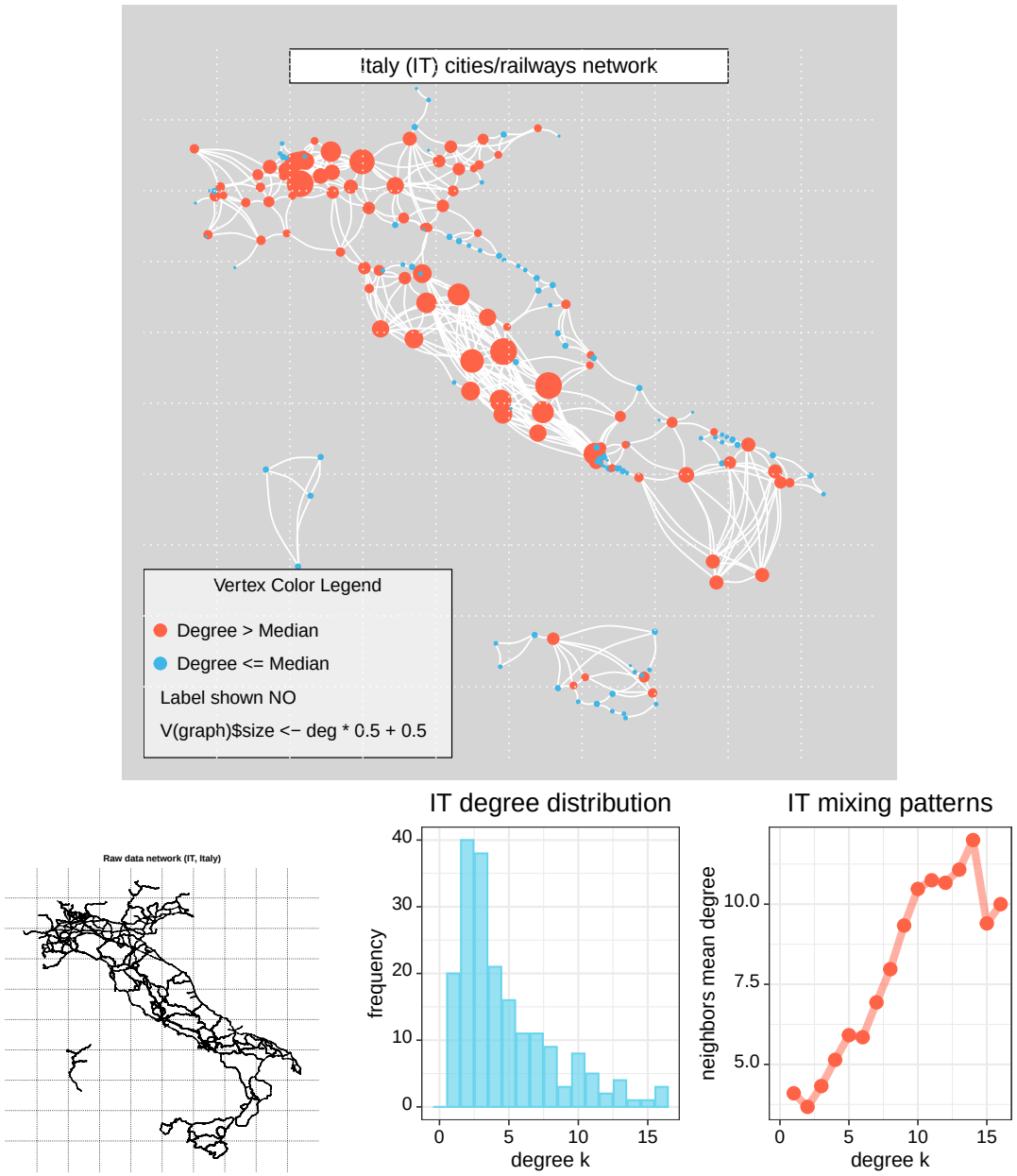


Figure 4.3

5 | Bibliography

Task 18: Turing Patterns

[1] Mikhailov A. Nakao H. "Turing patterns in network-organized activator–inhibitor systems". In: *Nature Phys* (2010). DOI: 10.1038/nphys1651.

[2] Takashi Miura Shigeru Kondo. "Reaction-Diffusion Model as a Framework for Understanding Biological Pattern Formation". In: *Science* (2010). DOI: 10.1126/science.1179047.

[3] Miriam Zara. *ComplexNet_Project*. https://github.com/miriamzara/ComplexNet_project.

[4] J. D. Murray. *Mathematical Biology II*. 3rd ed. Interdisciplinary Applied Mathematics. Springer New York, NY, 2003. DOI: 10.1007/b98869.

[5] Arianna Bianchi. *The Dynamics of Biological Systems*. Mathematics of Planet Earth. Springer Cham, 2019. DOI: 10.1007/978-3-030-22583-4.

A note to the bibliography. This assignment specifically focuses on Turing patterns in *networks*, with recommendation to replicate the findings presented in [1]. However, for someone unfamiliar with Turing patterns, a few preliminary read may be necessary. Article [2], targeted at biologists, offers an intuitive overview of the concept with minimal mathematical details. To delve deeper into the mathematical foundations, [4, *Chapter 2: Spatial Pattern Formation with Reaction Diffusion Systems*] provides a comprehensive and detailed explanation of Turing Patterns in a continuous medium. Another valuable resource is [5, *Chapter 7: The Turing Model for Biological Pattern Formation*], which, while shorter than Murray's chapter, still offers insightful observations.

Task 28: Voter Model

[6] K. et al Suchecki. "Voter model dynamics in complex networks: Role of dimensionality, disorder, and degree distribution". In: *Physical Review* (2005). DOI: 10.1103/PhysRevE.72.036132.

[8] K. Suchecki. "Conservation laws for the voter model in complex networks". In: *EPL* (2005).

[9] *Scipy: Fundamental algorithms for scientific computing in Python*. https://scipy.org.