R.M.J. Vooys

Numerical radiative transfer modelled by a Markov process

Bachelor Thesis, March 2011

Thesis supervisors: dr. Dion Gijswijt and prof.dr. Vincent Icke

Mathematical Institute and Leiden Observatory,
Leiden University

Numerical radiative transfer modelled by a Markov process

R.M.J. Vooys

March 11, 2011

Abstract

A model based on random processes for radiative transfer is introduced and investigated. Based on the ideas of SimpleX, for a given point distribution of an astronomical object, the points are being connected with a number of nearest neighbours. On the edges of the resulting graph, probabilities are assigned that reflect the way how radiation is being transferred throughout the object. Here only diffuse transfer is considered, i.e. all radiation is divided equally among the neighbours. The idea of adding sources and sinks is introduced and used. All of this eventually results in a large, sparse matrix, and replaces the question of solving the radiation equation to finding the stationary distribution vector, the eigenvector of the matrix corresponding to eigenvalue one. Various results from Markov theory, the Perron Frobenius theorem and numerical linear algebra are used to find this solution. As it turns out, our matrix is irreducible and aperiodic, so that the dominant eigenvalue equals one and the corresponding eigenvector is the only eigenvector that has positive entries. Hence we know that a stationary distribution vector exists and is unique. By means of the Arnoldi algorithm one can compute this eigenvector, along with other eigenvectors and eigenvalues. As an example, a given point distribution describing the collision of two small galaxies is taken. The model is investigated by experimenting with the parameters of the model, i.e., including the number of nearest neighbours, the luminosity of the sources and the location of the sinks, to see if we get any physically true results. As it turns out, taking 20 nearest neighbours and a certain allocation of the sinks gives a physcially satisfying result. Finally, one scheme for absorption by the points in the distribution is considered. The conclusion of this thesis is that this model can be used to model radiative transfer, although one needs to work more on the absorption aspect, and there is still work to be done on various other types of radiative transfer.

Keywords: radiative transfer, Markov theory, Perron-Frobenius theorem, numerical linear algebra, Arnoldi algorithm, nearest neighbours.

1 Introduction

When looking at the night sky, at cloudless weather, one can see millions of stars and if watched with a telescope galaxies or other astronomical phenomena. Somehow the light of each and every object in space found a way through their own medium, through space, through the earth's atmosphere, untill it reached the detectors of your telescope or your eye. When you look at an astronomical object, a galaxy or a star, you can only see the light that got through the matter the object consists of. How the light has travelled within the object cannot be seen, and therefore information about the inside of the object is hidden. Knowing how radiation found it's way from it's point of creation to the edge of the astronomical object gives us insight on what is going on inside the object.
To do this, we need to know what factors play a role on how radiation is being transfered. Among these factors are absorption α and emission j. These effects can be incorporated in one single equation [RL], giving the variation of the specific intensity I_ν with frequency ν:

$$\frac{dI_\nu}{ds} = -\alpha_\nu I_\nu + j_\nu$$

This differential equation looks rather simple, but it's general solution cannot be written in closed form: the factors α and j can vary vastly throughout the medium. More on this in a later chapter. The important thing is, that we either have to concentrate further on the equations of radiative transfer, or go away from it and think harder about how radiation is transfered. For this another method is found, where in stead of writing down the differential equation, one looks at the problem as a random process. When radiation arrives at a certain point, we say it has a specified probability to be either absorbed or re-emitted. The medium must be resembled by a smaller, finite, number of points, and each points needs to be connected in such a way that when radiation will be re-emitted at one point, what the probability specified will be that the emitted radiation reaches the other point. This process will then repeat itself.
What we want to find is what is called an invariant distribution: the total amount of radiation residing at the points will not change after the next iteration of the absorption and emission process. How to do this explicitly will be a topic in one of the next chapters, but the important thing to note is that we now have the following setting. Given the connections between the points, and the probabilities, we store this information in a matrix. That is, in row i and column j we store the probability that radiation will be transfered from point i to point j. If we resembled the astronomical object with N points, we will have a $N \times N$ matrix. The radiation transfer can be followed by constructing a vector with N entries that gives you the relative total amount of radiation residing at each point, so the sum of all entries will be unity. The next distribution can be calculated by multiplying this vector with the matrix. The invariant distribution will now be equal to the eigenvector of this matrix corresponding to eigenvalue 1, since multiplying that eigenvector with the matrix will return the same vector. So instead of looking at a differential equation, we now look at the problem of finding the eigenvector with eigenvalue 1 of a rather large matrix.
In this thesis all the ideas above will be explained more in depth. In Chapter 2 we start off with some mathematical background on Markov chains, where the idea of building a matrix from a graph and finding an invariant distribution will be presented. In Chapter 3 the model for radiative transfer will be explained. As it will turn out, the model that will be used will ensure that the matrix will be sparse: for any number N, we will only have a few nonzero entries at each row. This fact will be used to find the eigenvectors and eigenvalues of the matrix, which will be the topic of Chapter 4. After we have become familiar with the mathematical background and the model, a practical case will be taken and the method will be studied in Chapter 5, and the final chapter will be some discussion of the results. The chapters on mathematical background (Chapter 2 and 4) are meant to give an overview of what one should know about the topics to understand the steps and conclusions in this chapter. Many of theorems and proofs are taken from literature, and are referenced accordingly.

2 Mathematical Background

This section is devoted to giving the reader some more mathematical background about two main topics: Markov chains and numerical linear algebra. Some background in probability theory is assumed and background in linear algebra. Where needed, some basic definitions and theorems will be given, we refer the reader to textbooks for the most proofs. Only those proofs which are absolutely necessary for the remainder of this thesis will be given explicitly, or at least the idea of the proof will be given. The following definitions are from [Häggström] and [Norris], from which most notation is slightly altered for consistency throughout this thesis. There will be also two kinds of matrices in this thesis, one that is stochastic and sparse, denoted by M, and a more general matrix, denoted by A. From the context it will be clear if the theorems or other assertions will hold for the stochastic case only, or also in a more general setting.

2.1 Markov Theory

Definition 1. *Let $X_0, X_1, X_2, \dots$ be a sequence of random variables. The sequence $(X_0, X_1, \dots)$ is called a **Markov chain** if this random proces possesses the **Markov property**:*

$$\mathbf{P}\left(X_{n+1} = x | X_0 = x_0, X_1 = x_1, \dots, X_n = x_n\right) = \mathbf{P}\left(X_{n+1} = x | X_n = x_n\right)$$

Or in words: The conditional distribution of the state X_{n+1} given the states $(X_0, X_1, \dots, X_n)$ depends only on X_n.

Remark: We call X_0 the initial (starting) state, and X_n the state after n steps.

Definition 2. *Let M be a $k \times k$ matrix with elements $\{M_{i,j} : i, j = 1, 2, \dots, k\}$. A random process $(X_0, X_1, \dots)$ with finite state space $S = \{s_1, s_2, \dots, s_k\}$ is said to be a **homogeneous Markov chain with transition matrix** M, if for all n, for all $i, j \in \{1, \dots, k\}$ and for all $i_0, \dots, i_{n-1} \in \{1, \dots, k\}$ we have:*

$$\begin{aligned} \mathbf{P}\left(X_{n+1} = s_j | X_0 = s_{i_0}, X_1 = s_{i_1}, \dots, X_{n-1} = s_{i_{n-1}}, X_n = s_n\right) &= \\ \mathbf{P}\left(X_{n+1} = s_j | X_n = s_i\right) &= M_{i,j} \end{aligned}$$

Remarks: The matrix elements $M_{i,j}$ are called transition probabilities: it tells you the probability that a state s_i will go to state s_j. The term "homogeneous" in the definition above means that the probability from one state to another does not change over time. An inhomogeneous Markov chain has transition matrices $M^{(1)}, M^{(2)}, \dots$ for each timestep. We will not be considering inhomogeneous Markov chains in the rest of this thesis.
Every transition matrix is nonnegative: $M_{i,j} \geq 0$ for all $i, j \in \{1, \dots, k\}$, and each row of the matrix sums up to unity: $\sum_{j=1}^{k} M_{i,j} = 1$ for all $i \in \{1, \dots, k\}$. Such a matrix is therefore also called a **stochastic matrix**. Now we give a definition of the starting point of the Markov process, and how we denote the distribution of states after several times.

Definition 3. *The start of the Markov chain is called the **initial distribution**, and is represented as a row vector given by:*

$$\begin{aligned} \mu^{(0)} &= \left(\mu_1^{(0)}, \mu_2^{(0)}, \dots, \mu_k^{(0)}\right) \\ &= \left(\mathbf{P}\left(X_0 = s_1\right), \mathbf{P}\left(X_0 = s_2\right), \dots, \mathbf{P}\left(X_0 = s_k\right)\right) \end{aligned}$$

Similarly, the row vectors $\mu^{(1)}, \mu^{(2)}, \dots$ denote the Markov chain at times $1, 2, \dots$, so that for time n we have:

$$\begin{aligned} \mu^{(n)} &= \left(\mu_1^{(n)}, \mu_2^{(n)}, \dots, \mu_k^{(n)}\right) \\ &= \left(\mathbf{P}\left(X_n = s_1\right), \mathbf{P}\left(X_n = s_2\right), \dots, \mathbf{P}\left(X_n = s_k\right)\right) \end{aligned}$$

The next theorem will tell us how to compute future distributions, given an initial state.

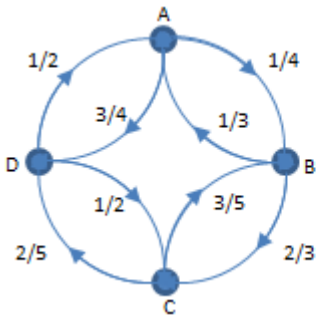
Theorem 1. *For a Markov chain $(X_0, X_1, \dots)$ with state space $\{s_1, \dots, s_k\}$, initial distribution $\mu^{(0)}$ and transition matrix M, we have for any n that the distribution $\mu^{(n)}$ at time n satisfies*

$$\mu^{(n)} = \mu^{(0)} M^n$$

Proof. The proof can be found in [Häggström]. There induction by n is used to show that $\mu^{(n+1)} = \mu^{(n)} M = \mu^{(0)} M^{n+1}$. □

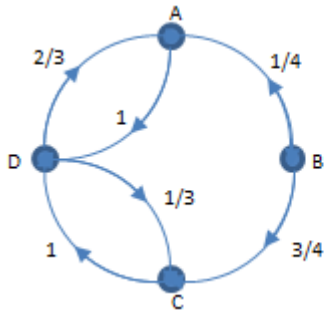
It is possible to picture a Markov chain by it so-called **transition graph**. The nodes represent the states and arrows between the nodes the transition probablities. Of course this also works the other way around, we start with a graph, with transition probabilities between nodes, and list these probabilities in a matrix. See figure below for an example.

Next we introduce two classifications for Markov chains: irreducible and periodic Markov chains.



$$\begin{pmatrix} 0 & 1/4 & 0 & 3/4 \\ 1/3 & 0 & 2/3 & 0 \\ 0 & 3/5 & 0 & 2/5 \\ 1/2 & 0 & 1/2 & 0 \end{pmatrix}$$

Figure 1: Irreducible chain



$$\begin{pmatrix} 0 & 0 & 0 & 1 \\ 1/4 & 0 & 3/4 & 0 \\ 0 & 0 & 0 & 1 \\ 2/3 & 0 & 1/3 & 0 \end{pmatrix}$$

Figure 2: Reducible chain

Definition 4. *A Markov chain $(X_0, X_1, \dots)$ with state space $S = \{s_1, s_2, \dots, s_k\}$ and transition matrix M is said to be **irreducible** if for all $s_i, s_j \in S$, there is a positive probability of ever reaching state s_j when we start from s_i. Or in other words: The chain is irreducible if for any $s_i, s_j \in S$ we can find an n such that $(M^n)_{i,j} > 0$. Otherwise, the chain is called **reducible**.*

The graph in Figure 1 is irreducible. In Figure 2, an example of an reducible chain is given, since there is no way to reach point B from any other point. Loosely speaking, irreducibility is the property that every state of the Markov chain can be reached from any other states (even if it may take a while to get there).

Definition 5. *A markov chain is said to be **aperiodic** if all its states are aperiodic. A state s_i is aperiodic if its period $d(s_i)$ is 1; where*

$$d(s_i) = gcd\left\{n \geq 1 : (M^n)_{i,i} > 0\right\}.$$

*Otherwise, the chain is said to be **periodic**.*

Now we state and prove a theorem on nonnegative, irreducible matrices. A few of its assertions will be used for some specific research questions later in this thesis. Various versions of the Perron-Frobenius exist, here we adopt the one from [Meyer] and [Seneta]. As will be seen in a later chapter, the matrix considered will be aperiodic and irreducible. Therefore the assertions are restricted to these cases. The proofs can be found in [Meyer], so I omit them all but one.

Theorem 2 (Perron-Frobenius theorem for nonnegative matrices)**.** *Let $A = (a_{ij})$ be an $n \times n$ nonnegative matrix, let it be irreducible with period 1. Then the following holds:*

- *There exists a positive real number r such that for all eigenvalues $\lambda : |\lambda| \leq r$*
- *r is a simple root of the characteristic polynomial of A, hence r is an eigenvalue of A, with algebraic multiplicity one.*
- *The left and right eigenvectors v and w corresponding to r have strictly positive entries.*

- *Any nonnegative left eigenvector of A is a scalar multiple of v, and any nonnegative right eigenvector of A is a scalar multiple of w.*
- *r satisfies the inequalities* $\min_i \sum_j a_{ij} \leq r \leq \max_i \sum_j a_{ij}$

Proof. We only prove the uniqueness of the nonnegative eigenvector assiociated with r, and assume the other statements are true, in particular that the real number r is a simple root of the characteristic polynomial of A. This means that the eigenspace for r is one dimensional (dimension of the eigenspace of an eigenvalue is the number of linearily independent eigenvectors with that eigenvalue). Assume x is a positive eigenvector with eigenvalue s of A. Let v be the eigenvector with eigenvalue r such that $v < x$, with r the eigenvalue from the statement of the theorem. Then for any $n \in \mathbb{N}$, $r^n v = A^n v < A^n x = s^n x$. Since this is not possible if $s < r$, and by the first statement of the theorem, we cannot have $r < s$, so it must be that $s = r$. Hence x must have eigenvalue r, and by the above remark, it must be that x is a scalar multiple of v. □

2.2 Invariant distributions

If a Markov chain is irreducible, then the last assertion of the Perron-Frobenius theorem states that the maximal eigenvalue of the transition matrix equals 1, since the minimal and maximal rowsums are both unity. This is very good to know, since we now know that there exists an (unique) positive eigenvector x such that $x = xM$, which leads us to the following definition.

Definition 6. *Let $(X_0, X_1, \dots)$ be a Markov chain with state space $S = \{s_1, s_2, \dots, s_k\}$ and transition matrix M. A row vector $\pi = (\pi_1, \dots, \pi_k)$ is called an **invariant** (or: stationary, equilibrium) distribution for the Markov chain, if:*

- *$\pi_i \geq 0$ for $i = 1, \dots, k$ and $\sum_{i=1}^k \pi_i = 1$, and*
- *$\pi M = \pi$, meaning that $\sum_{i=1}^k \pi_i M_{ij} = \pi_j$ for $j = 1, \dots, k$*

Now we state the most important theorem of this section. It states how we compute the invariant distribution, if it exists, when the Markov chain is irreducible and aperiodic, independent of the choice of the initial distribution. The statement comes from [Norris] and is slightly adjusted to our notations so far. Due to Perron-Frobenius theorem, we already know for our specific case, such a stationary distribution indeed exists.

Theorem 3 (Convergence to invariant distribution)**.** *Let $(X_0, X_1, \dots)$ be an irreducible and aperiodic Markov chain with state space $\{s_1, \dots, s_k\}$, any initial distribution $\mu^{(0)}$ and transition matrix M. Suppose M has an invariant distribution π. Then*

$$P(X_n = j) \to \pi_j$$

as $n \to \infty$ for all j. In particular,

$$M^n_{i,j} \to \pi_j$$

as $n \to \infty$ for all i, j.

Proof. The proof can be found in [Norris, sect 1.8, theorem 1.8.3].

Remark: The last assertion can be rephrased as when $n \to \infty$, $M^n \to \Pi$, where Π is the limiting matrix whose entries are constant along the columns, i.e. the rows of Π are repetitions of the same vector π.

Remark: In Chapter 4 of this theses, a motivation for why using powers of matrices to find the invariant eigenvector will be given, for the case that the matrix is diagonalisable.

Remark: It is not explicitly said in [Norris], but the invariant distribution as in the above theorem is unique. At the start of this subsection, I already showed the distribution must be unique. In [Häggström, end of Section 5] one can find a more explicit proof, without the use of the Perron-Frobenius theorem.

3 Numerical radiative transfer

One of the fields in astronomy, is to know how radiation is distributed in astronomical objects. This boils down to solve equations for radiative transfer. For example, one of the simplest forms of the radiation equation is [R&L, eq 1.23]:

$$\frac{dI_\nu}{ds} = -\alpha_\nu I_\nu + j_\nu$$

Here the intensity I depends, at least, on two vectors for position in space and direction of radiation and depends on the frequency ν. Also the coefficients of absorption α_ν and emission j_ν can vary drastically throughout the object we are considering. In practice, for almost all physical cases the analytical solution of the equation for radiative transfer is not known, or can not be known. Therefore many numerical methods are devised for solving the equations of radiative transfer. One of this methods is currently being worked on, called SimpleX [Paardekoper, Kruip, Icke] and [Ritzerfeld]. In this method, one is not looking at the differential equation anymore, but goes back to the basics of how radiation is transfered through the medium. For this one specifies a point distribution that represents the object, and prescribes a recipe of how much radiation is being transfered from one point to another. This process is assumed to be a Markov process, and solving the radiation equation will now be equivalent to finding the invariant distribution of the associated Markov matrix. How this is done will be explained in the next subsections. Solving the differential equation above is analogous to solving an eigenvalue problem.

3.1 The model

The model of the radiative transfer we are using can be quickly summarized as follows.

- Specify a point process that represents the object in question.
- Perform a tesselation with these points as vertices to build a graph.
- Store all information about the links between points in a matrix M.
- Define a weighting according to the process you are interested in. Multiply all M_{ij} by such weighting.
- Add sources and sinks. Connect each sink with each source.
- Normalize each row of M such that each row sums up to unity.

I will now address each of these points one by one and explain them in slightly more detail.
First we specify a point process that describes the object as accurate as possible. This is not a trivial task, it depends on the density distribution of the object, and you want to make the choice of points Poisson random. Also the number of points you are going to take must be high enough to represent the object as good as possible, but not too much because of the computational obstacles you will encounter. These choices for the point process choice is described in [Ritzerfeld], I will not elaborate on this further in this thesis. From now on it is assumed such a point process has been established.
Next we need to find a recipe of how radiation is being transferred between the points. For this, one takes the points of the point distribution as vertices for a graph, and the edges of the graphs can be constructed by various methods, I will discuss two of them, the Voronoi-Delaunay triangulation and the N-nearest neighbours.
In the SimpleX code, one uses a method called Delaunay-Voronoi triangulation. Here one first constructs the Voronoi diagram of the point distribution: each point of the point distribution is called a nucleus, and each Voronoi cell around that nucleus contains all points in space that are more near that nucleus than any other nucleus. The set of points that have exactly two nearest nuclei is called a Voronoi wall. Next, one says that two nuclei are connected whenever the nuclei are seperated by a common Voronoi wall. See the figure below for an example. This way one constructs a graph with the nuclei as its vertices. This type of partition of space can be made

in any dimension, the figure below shows it for a tesselation in a plane, but in practice you use the three dimensional variant. Many properties are known for this type of partition of space, as is described in [Okabe, et. al.], [Møller] and [Ritzerfeld]. For example, in the three dimensional case, the mean number of edges coming out of each vertex is 15.54 . . . , and Ritzerfeld has shown that the mean edge length is proportional to the mean free path of a photon, a very nice physical property.
Another method is the N-nearest neighbours. Here, each vertex is connected with a prescribed number of N nearest neighbours, according to Euclidean distance. This method has a few advantages over the previous method, and some disadvantages. The advantages are that building this graph is faster than building a Delaunay grid, and you can vary the number of edges coming out of each point. The disadvantages are that this kind of partition of space has no physical meaning, and the graph is more directed: in the Delaunay-Voronoi case, all points are interconnected, but in case of the nearest neighbours, when y is one of the neigbours of a point x, x is not necessarily a neighbour of y, see image below. This thesis will be about whether this method is still meaningful in astronomical applications, even with these disadvantages. When one has constructed the graph by some method (like the two above, but maybe any other method), we store the connections in a matrix M. Whenever a point i is connected to a point j we set $M_{ij} = 1$, and zero otherwise. Thus when using the N-nearest neighbours method, each

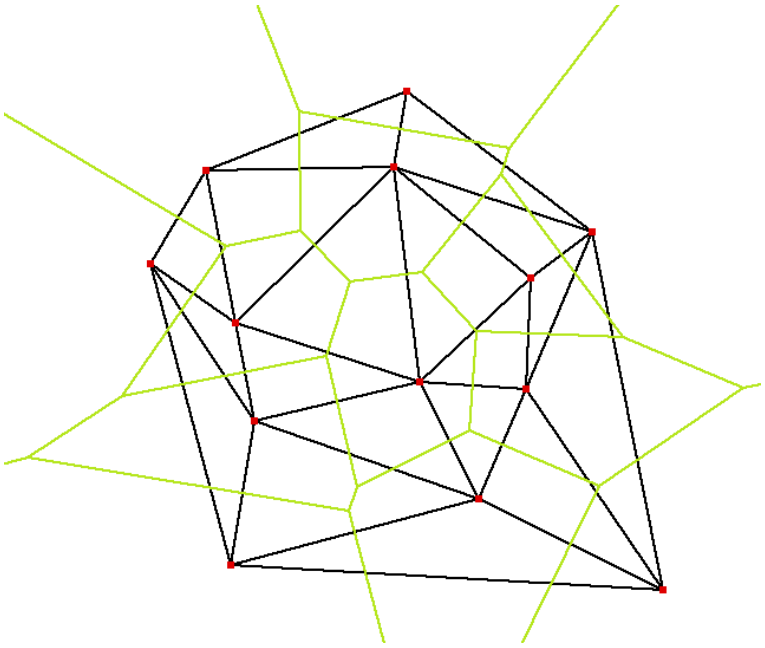


Figure 3: The construction of the Voronoi-Delaunay graph. The green lines represent the Voronoi cells, and two points are connected with a black line whenever the points share a Voronoi wall.

Figure 4: Although point B is a nearest neighbour of point A (green line), point B does not have point A as one of its nearest neighbours.

row of M will have N nonzero entries.
The transport of radiation is now as follows: each package of radiation in a vertex i will move along one of the outgoing edges with some probability to another vertex j. The assignment of probabilities is the next step of the model, it completely depends on what type of radiation you are interested in. Some of them are considered in [Ritzerfeld], like ballistic radiation, where also the direction of where radiation came from initially is considered. But the main point is, that you need to prescribe some sort of weighting in each edge along each vertex of the graph. This information is stored in the matrix M, where M_{ij} gives the probability of going from vertex i to vertex j. This way, each row of M will sum to unity.
We now have a grid with vertices and edges and probabilities along edges. Now we define two types of points: sources and sinks. A source is a point that only has outgoing lines: it only emits radiation, but won't recieve any. You may think of a star surrounded by a cloud of dust: the star is the source and radiates, and all other points around it recieve, absorb and re-emits the radiation further away. A sink is a point that only has have incoming edges: it only recieves radiation, but won't emit any. Sinks are either totally absorbant points within the object, or points on the boundary of the object.
We will place the sinks in some way around the object artificially, and will then connect each sink with each source. This connection is virtual and the the sinks have the following purposes. Firstly, this way we have photon conservation at the equilibrium case: the total number of photons in the system remains constant. Secondly, the amount of packages that is sent by the sinks to each source will depend on the luminosity of the source: if we have two sources and one source is twice as luminuous as the other, it will recieve twice as many packages of the sinks as the other one does. And finally, the radiation must eventually leave the object in order to make it possible that the object can be observed. The sinks at a fixed distance around the object will collect such escaped radiation.
Since we now have added two special types of points, we need to adjust the matrix M accordingly. When a point x has been chosen to be the source, all points that have an common edge with x will be forbidden to send radiation to it: $M_{i,x} = 0$ for all i. Next for each sink y, we forbid that y sends radiation to any other point by setting $M_{y,i} = 0$ for each i and then virtually connect each sink with each source by setting $M_{y,x} = p$, where p is a probability value corresponding to the luminosity of the source x.
To ensure we have a probability matrix, we normalize each row of M so that all row elements sums to unity.
We now have modelled the radiative transfer and stored all the information in an $n \times n$ matrix M, where n is the number of points in the point distribution. The question now is: what is the invariant distribution? This is the distribution vector μ of length n, where each component corresponds with the amount of radiation packages that resides in the corresponding point. When applying the matrix onto this vector, in order to have an equilibrium with photon conservation and all, the same vector must come back. In other words, solve the following eigenvalue problem: $\mu M = \mu$.

3.2 Connections with mathematical theory

We now have build a probability matrix of size n. We make now the assumption that the whole process as described above follows the Markov property: the probability that radiation will flow from point i to any of it's connected points j is independent of how the radiation got to point i. Remark that this does not necessarily has to be true, as mentioned in [Ritzerfeld], if you want to take into account the direction from which the radiation has come from initially, the process does not follow the Markov property. But from here on, I assume it does. Now, by the construction of sources and sinks, the matrix is constructed to be irreducibel: you can go to any point, from any point, with positive probability. Also the matrix is aperiodic.
The mathematical theory in the previous chapters now gives a few important results. The Perron-Frobenius theorem tells us, that since each row of M sums to unity, the maximal eigenvalue r is equal to 1. Also, the theorem states that there exists a positive eigenvector of M corresponding to the eigenvalue $r = 1$. Hence we know that there exists a solution of

our eigenvalue problem $\mu M = \mu$. This is an important result, since we now know at least a solution must exist. But yet another important observation is, that the solution is also unique. There is only one eigenvector with positive entries. Every other eigenvector will have no physical meaning, since it will have a negative component, which corresponds to the negative number of photons in a point, which is absurd. However, the other eigenvectors may tell you something about how fast you can the dominant eigenvector, and maybe we are able to combine the other eigenvectors somehow to get something meaningful.
So the question now is, how do we find the eigenvectors of a very large matrix? One significant property of our matrix, is that it is sparse. Typically, we use an astronomical amount of points in the point distribution, thus the matrix will be very large. Only a few entries per row of the matrix will be nonzero: if you used Voronoi-Delaunay, around 15 to 16 entries out of the thousands from each row will be nonzero, or if you used N-nearest neighbours, that number would be around N. The remainder of the matrix is zero. This property will be exploited to find the eigenvectors fast, and how that works is the topic of the next chapter.

4 Numerical Linear Algebra

As is made clear in the previous chapter, the matrix I am considering is large and sparse. From this matrix I am interested in the eigenvector associated with eigenvalue 1, the maximal eigenvalue. The standard direct method is to solve the characteristic equation

$$\det(M - \lambda I) = 0$$

where I is the identity matrix. The eigenvalues are the roots of this polynomial in λ. But the matrix is large, say $10^3 \times 10^3$, and there exists in general no formula for finding the roots of a polynomial of degree greater than 4. One could also try to solve the problem by setting up thousands of linear systems, but solving this will be extremely unpleasant and must be avoided at all costs. We have to use iterative methods for finding the eigenpairs of the matrix. I will discuss two methods, the power method for finding the eigenvector corresponding to the dominant eigenvalue, and the Arnoldi algorithm for finding more eigenvalues and their associated eigenvectors. In a later chapter of this thesis, the spectra of various point distributions and allocations of sources and sinks are considered, for which (preferably) all of the eigenvalues are wanted. The power method only gives the top one, but the Arnoldi can give more, and uses the idea of the power method. A version of this method is also used in Matlab for eigenvector finding. [1]
We have a large, sparse, irreducible and aperiodic matrix M which, in general, is not symmetric. We know from previous chapters that the dominant eigenvalue is given by the Perron-Frobenius theorem: $\lambda_1 = 1$. Assume that the eigenvalues are ordered such that

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$$

For the desciption of the methods, I will keep the notation λ_1 for the dominant eigenvalue, even though we already know it's particular value in our case. The methods also work for matrices that are not stochastic.

Remark: Before we go on, a small remark on notation has to be made. In the discussion on Markov chains in the first chapter, the matrix was constructed in such a way that we need to find the so called left eigenvector. If we altered the definition for the construction of the transition matrix, by saying that if the probability of point i to point j is nonzero, then this nonzero number is stored in the i-th *column* and j-th row of the matrix, in stead of the other way around. In that case we need to find the right eigenvector for our purposes. In the literature for eigenvector finding methods, one seeks the right eigenvectors, but in the literature of Markov

[1] What will follow is a collection of information found in [Saad], [Radke] and [Press, et. al.]. All theorems are taken from [Saad], but rewritten in a form that fits this thesis. The intention of this chapter is to give an overview of both methods and why they work.

chains, one uses the definition so that you need to find the left eigenvectors. The left eigenvectors for a matrix are the same as the right eigenvectors of the transpose of the matrix, so nothing has changed on the kind of eigenvectors or eigenvalues I am after. If the method finds the right eigenvectors, one needs to transpose the matrix accordingly.

4.1 The power iteration method

The power iteration method essentially does the following: start with any initial vector v_0, let the matrix M act on this vector iteratively, and when you do this long enough, you will end up with the eigenvector corresponding to the largest eigenvalue. Note that the proof of this for the stochastic case is already given in the chapter of Markov theory! There you compute the powers of M and let that act onto the initial distribution vector, but this is equivalent with iteration. In fact it is more efficient to form v_k by multipliying M with v_{k-1} rather than compute the next power of M at every step. There is another way to proof that this method will work, whenever there exists a dominant maximal eigenvalue. The algorithm for the general case is as follows, the proof is for the case when the matrix is diagonalizable.[2]

Algorithm 1 (The power iteration method)**.**

1. *Choose an initial vector* v_0
2. *For* $k = 1, 2, 3, \ldots$ *compute:*
   (a) $v_k = \frac{Mv_{k-1}}{\|Mv_{k-1}\|}$
   (b) $\lambda_k = \frac{v_k^T M v_k}{v_k^T v_k}$
3. *stop when* $\|Mv_k - v_k\lambda_k\| < \tau\,|\lambda_k|$

Theorem 4. *In the power method algorithm above the* λ_k *will converge to the dominant eigenvalue* λ_1 *and* v_k *will converge to the corresponding eigenvector of matrix* M*.*

Proof. Let v_0 be a random vector. Suppose the exact eigenvectors of M are $\{x_1, x_2, \ldots, x_n\}$, with corresponding eigenvalues $\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. Since the eigenvectors of M form a basis for $\mathbb{R}^n$, v_0 can be written as a linear combination of these eigenvectors:

$$v_0 = \sum_{k=1}^{n} c_k x_k$$

Multiply this vector with M gives:

$$Mv_0 = \sum_{k=1}^{n} c_k M x_k = \sum_{k=1}^{n} c_k \lambda_k x_k$$

Repeating this m times yields :

$$M^m v_0 = \sum_{k=1}^{n} c_k M^m x_k = \sum_{k=1}^{n} c_k \lambda_k^m x_k = \lambda_1^m \left(c_1 x_1 + \sum_{k=2}^{n} c_k \left(\frac{\lambda_k}{\lambda_1} \right)^m x_k \right)$$

Since M has an dominant eigenvalue, $\left(\frac{\lambda_k}{\lambda_1}\right)^m \to 0$ for $k > 0$ as $m \to \infty$. The rate of convergence thus depend on the ratio $\frac{\lambda_2}{\lambda_1}$, since a high ratio means that the k-th power will slowly converges to zero as k goes to infinity. From $2(a)$, we then easily see:

$$v_k = \frac{Mv_{k-1}}{\|Mv_{k-1}\|} = \frac{M^k v_0}{\|M^k v_0\|} \to x_1$$

[2]The proof for the general case uses Jacobi factorisation of M, and the proof is rather lengthy and is beyond the scope of this chapter. The proof that uses the fact that M is diagonalizable illustrates the dependence on other eigenvectors for the convergence of the method.

as $k \to \infty$.

The second step follows from the definition of eigenvectors and eigenvalues:

$$Mv_k = \lambda_k v_k$$

Multiplying both sides with the transpose of v_k and rearranging will give the assertion. And since $v_k \to x_1$ as $k \to \infty, \lambda_k \to \lambda_1$ as $k \to \infty$. □

We see from the proof that the convergence will depend on the second largest eigenvalue λ_2, for when λ_2 is almost as large as λ_1, the ratio $\frac{\lambda_2}{\lambda_1} \approx 1$ and it's powers will vanish slowly.
The τ parameter in the algorithm is the value of tolerance. It is just something you tell the algorithm within what accuracy you want to find the dominant eigenpair, say within the first 6 digits.
The sparsity of the matrix M is exploited in this algorithm in step a, since the number of operations for multiplying the matrix with a vector is of the order of the number of nonzero entries in the sparse matrix.

4.2 Krylov subspace iteration and the Arnoldi algorithm

The power method computes the dominant eigenpair, but can not compute other eigenpairs. In the algorithm, we essentially overwrite the vector $M^{m-1}v_0$ with the vector $M^m v_0$. It turns out that it will be usefull to use the iterates. We make the following definition:

Definition 7. *Let A be a square matrix, v_0 a vector and m an integer. Then the **Krylov subspace** is defined as*

$$\mathcal{K}_m(A, v_0) \equiv span\left\{v_0, Av_0, A^2v_0, \ldots, A^{m-1}v_0\right\}.$$

The vectors $A^k v_0$ in the Krylov space are not orthogonal, but an orthonormal base can be constructed by means of a Gram-Schmidt procedure. This is essentially what Krylov subspace iterative methods will do: build a orthonormal basis and use the resulting vectors for constructing the eigenvectors and eigenvalues. The Arnoldi algorithm will compute the orthonormal basis for matrices that are not symmetrical, like ours. In this algorithm a version of Gram-Schmidt procedure is followed, and entries for an upper Hessenberg matrix iare constructed, from which the eigenvalues for M can be derived. All of this will be stated and mostly proved in the next theorems.
We start by stating the Arnoldi algorithm, and then give a small discussion of it.

Algorithm 2 (The Arnoldi algorithm)**.**

1. *Choose a normalized vector v_1 and $m \in \mathbb{N}$.*
2. *for $j = 1, 2, \ldots, m$ do*
   - *(a) $h_{i,j} = Av_j \cdot v_i$ for $i = 1, 2, \ldots, j$.*
   - *(b) $w_j = Av_j - \sum_{i=1}^{j} h_{i,j} v_i$*
   - *(c) $h_{j+1,j} = \|w_j\|_2$*
     - *if $h_{j+1,j} = 0$, STOP*
   - *(d) $v_{j+1} = \frac{w_j}{h_{j+1,j}}$*

A couple of new parameters are defined in this algorithm: $h_{i,j}$, which is simply the innerproduct of all previously constructed orthonormal vector with the product of A onto the most recent constructed vector, and w_j, which is the projection step. In item 2(b) one can recognize the Gram-Schmidt procedure. The algorithm stops if w_j in this step vanishes. The assignment w_j

in this algorithm is merely for notational reasons, $h_{i,j}$ will be used shortly. Written to full form, the orthonormal vectors are:

$$v_{j+1} = \frac{Av_j - \sum_{i=1}^{j} (Av_j \cdot v_i)\, v_i}{\left\| Av_j - \sum_{i=1}^{j} (Av_j \cdot v_i)\, v_i \right\|} \quad (*)$$

Theorem 5. *The set of vectors $\{v_1, v_2, \ldots, v_m\}$, generated by the Arnoldi algorithm, forms an orthonormal basis for $\mathcal{K}_m(A, v_1)$.*

Proof. By step $2(d)$, the vectors are orthonormal by construction. It remains to show that they span $\mathcal{K}_m(A, v_0)$. According to a proposition from [Saad], the Krylov subspace $\mathcal{K}_m(A, v_0)$ is the subspace of all vectors which can be written as $x = p(A)v_0$, where p is a polynomial of degree not exceeding $m-1$. The proof of this theorem will follow from the claim that each vector v_j is of the form $q_{j-1}(A)\, v_0$, where q_{j-1} is a polynomial of degree $j-1$. This will be shown by induction on j. For $j = 0$, we have $v_0 = q_0(A)\, v_0$, with $q(t) = 1$. Assume the claim is true for general j. Then we have, by $(*)$:

$$\begin{aligned} v_{j+1}h_{j+1,j} &= Av_j - \sum_{i=1}^{j} (Av_j \cdot v_i)\, v_i \\ &= A \cdot q_{j-1}(A)\, v_0 - \sum_{i=1}^{j} h_{i,j} q_{i-1}(A)\, v_0 \\ &= q_j(A)\, v_0 \end{aligned}$$

Hence, by the induction step and induction hypothesis, the claim is true for all j. □

The next theorem will show the use of the coefficients $h_{i,j}$ constructed by the algorithm. A matrix H is called upper Hessenberg if all its entries below the lower subdiagonal are zero.

Theorem 6. *Let A, $\{v_1, \cdots, v_m\}$ and $h_{i,j}$ be as generated by the Arnoldi algorithm. Construct the upper Hessenberg $m \times m$ matrix via $(h_{i,j})$ and let V_m be the $n \times m$ matrix with the orthonormal vectors v_j as column vectors. Then we can find the following relations:*

1. $AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T$
2. $V_m^T A V_m = H_m$

Proof. The first relation can be derived from $2(b)$ and $2(d)$ from the algorithm, for $j = 1, 2, \ldots, m$:

$$Av_j = \sum_{i=1}^{j+1} h_{i,j} v_i$$

The second relation is easily derived from the first, by multiplying both sides with V_m^T and make use of the orthonormality of the vectors $v_1, \ldots, v_m$. □

Now to the punchline: the m approximate eigenvalues and eigenvectors for A are now derived from the upper Hessenberg matrix H_m. The eigenvalues of H_m approximate the eigenvalues of A, and if x is an eigenvector of H_m corresponding to eigenvalue λ, then $V_m x$ is the approximate eigenvector of A. To find the eigenvectors and eigenvectors of the Hessenberg matrix, a whole other kind of procedure is used: the QR-factorization. I will not go in much detail, but the idea of this method is as follows: you start with the simillarity transformation $AV = VH$, where H is upper Hessenberg, $V^H V = I$. The algorithm is as follows:

Algorithm 3 (QR decomposition)**.**

1. *for i = 1,2,... until convergence do*
   (a) *Factor $H_i = Q_i R_i$*

(b) Set $H_{i+1} = R_i Q_i$ *and* $V_{i+1} = V_i Q_i$

2. *end*

Here Q is a orthogonal matrix and R an upper triangular matrix. The matrix H is now made upper triangular, its eigenvalues are listed on its diagonal.

The whole procedure of finding the first few eigenvalues and eigevectors of A is now summarized below.

- Construct the Krylov subspace $\mathcal{K}_m(A, v_0)$
- Use the Arnoldi algorithm to construct a orthonormal set of vectors v_i such that $\mathcal{K}_m(A, v_1) = \text{span}\{v_1, v_2, \ldots, v_m\}$, and set $V_m = [v_1 v_2 \ldots v_m]$.
- Construct the upper Hessenberg matrix H_m
- Use QR-factorization to construct the eigenpairs (x_i, λ_i) of H_m
- The eigenvalue of A is approximated by λ_i
- The eigenvector of A is approximated by $V_m x_i$

When does the Arnoldi algorithm breaks down? From step 2(d) in the algorithm, we see that this will happen when the vector w_j has zero norm at step j. The following theorem address the conditions under which this occurs, but in order to proof that theorem, we need a lemma and another definition. The **minimal polynomial** of a vector v_0 is the nonzero monic polynomial p of lowest degree, such that $p(A)v_0 = 0$.

Lemma 1. *The Krylov subspace* $\mathcal{K}_m$ *is of dimension* m *if and only if the degree of the minimal polynomial of* v_0 *with repsect to* A *is larger than* $m - 1$.

Proof. The vectors $v_0, Av_0, \ldots, A^{m-1}v_0$ form a basis, they are linearly independent: there is at least one a_i such that $\sum_{i=0}^{m-1} a_i A^i v_0$ is nonzero. This is equivalent with the condition that there be no polynomial of degree smaller or equal to $m - 1$ for which $p(A)v_0 = 0$. This proves the result. □

Theorem 7. *Arnoldi's algorithm breaks down at step* j *if and only if the minimal polynomial of* $v_1 (= v_0)$ *is of degree* j. *In this case the approximate eigenvalues and eigenvectors are exact.*

Proof. Let the degree of the minimal polynomial be equal to j, then assume that w_j is not equal to zero. Then v_{j+1} can be defined and as a result, $\mathcal{K}_{j+1}$ would be of dimension $j + 1$. By the previous lemma, the degree of the minimal polynomial is larger than j, which is not true. This leads to the contradiction, and the assumption of w_j being nonzero is false. Hence $w_j = 0$ and the Arnoldi algorithm stops.
Let $w_j = 0$. The degree d of the minimal polynomial of v_1 is $d \leq j$. But we cannot have that $d < j$, for then we have that $w_d =$, by the previous proof, and the algorithm should have stopped at the earlier step number d. So $d = j$. This completes the first half of the theorem.
The second part of the theorem is more demanding and beyond the scope of this thesis, and I refer to [Saad] proposition 4.3 and proposition 6.7 for the complete proof. □

Remark: If the algorithm has not stopped before the m-th iteration, one might be able to increase the number m to have more eigenvectors. *Remark:* A couple of questions that arises from the discussion above: how good are these approximate eigenvectors and eigenvalues, and how long do I have to iterate untill I get good results. This all depends on your definition of what a good approximate eigenvector is. [Saad] considers the distance between a particular eigenvector from the subspace $\mathcal{K}_{\Updownarrow}$. However, this consideration leads to extremely technical results, and therefor I will only refer to chapter 6.7 for Saad if one is interested in how this works. Also, [Saad] concludes at the end of that chapter: T̈he conclusion is that the eigenvalues that are in the outermost part of the spectrum are likely to be well approximated.Ö̈ther sources also conclude that the convergence of the eigenvectors using Arnoldi algorithm is not yet fully understood and worked out for the unsymmetric case.

5 Practical implementation

Back to the problem at hand. In Chapter 3 the problem has been posed, namely finding the stationary vector of the stochastic transition matrix M, and how to do that efficiently was explained in Chapter 4. What will follow are a few practical examples, with various kind of point distributions and allocations of the sinks and different luminosities of the sources. I am considering the nearest neighbour method, due to its practical implementation, and because it is much faster to generate than the Voronoi-Delaunay method. The main question now is, despite of the physical disadvantages of the nearest neighbour method, do we still get realistic results? The following questions are posed beforehand and answered in the rest of this chapter:

1. What does the stationary eigenvector look like?
2. What is the distribution of eigenvalues for different kinds of point distributions?
3. Can we approximate the stationary eigenvector by a combination of the other eigenvectors, for example to try and reduce the poisson noise added by the random allocation of the point distribution?

These questions will be answered in the next subsections, but first I will present some coding considerations.

5.1 Matlab/coding considerations

For construction of the matrix M and searching its eigenpairs, I am using the program Matlab. Matlab stands for Matrix Laboratory, and is used widely to cope with large matrices, and knows how to deal with sparse systems. One of the reasons of choosing Matlab rather than starting from scratch with C, Fortran or Python, is that it already has a built-in function called `eigs`, which can compute the first m eigenvectors and eigenvectors for sparse matrices. It uses a mathematical equivalent version of the Arnoldi Algorithm, called the Implicitly Restarting Arnoldi algorithm (IRMA).
The data for the point distribution and the N nearest neighbours has been provided by my thesis supervisor V. Icke. The data represents two small galaxies that are in collision with each other. Also the allocation of sinks around the objects is given by V. Icke. If the object consist of P points, Q sinks, R sources and N neighbours, the input data I will have consists of $P + Q + R$ rows and $N + 3$ columns: each row corresponds with a point in the point distribution, the first three colums are that point's cartesian coordinates X, Y and Z, and the final N columns are the N nearest neigbours of that point. This data is constructed in such a way that the first R rows corresponds with the sources, next P normal points and the last Q rows are the sinks. The sources in this case will be the centre of the halo of each of the galaxies.
The full Matlab code is given in the appendix at the end of this thesis. Sufficient comments are made there, so you can easily see what part of the code does what part of my research. Note that you can turn off pieces of this code if you are interested in only a few aspects, e.g. if you only want to know the stationary distribution, you may want to ignore the rest of the code, since the whole code can take a couple of minutes to compile on a standard laptop. Below an overview of various steps of the code.
The construction of the matrix is quite easily done with the provided data. As is shown in the first chapter, the probability of going from point i to point j will be stored in the matrix at entry M_{ij}. I am only going to consider equal distribution along the neighbours, that is if a point is connected with k neighbours, the probability will be set as $1/k$. Since I am going to add sources and sinks later on, first I set all entries of the matrix to 1 whenever a connection from one point to another exists. At the very end, when everything is correctly connected, all rows of the matrix will be normalised by summing the row vector (that tells us with many points a point is connected to, which is the number k) and dividing each entry in that row by that amount.
Since Matlab can make use of the sparsity of the matrix, we can build a sparse matrix right away with a slight trick, as is shown in the appendix. Next, the allocation of sources and sinks is

being done: first we make sure that all points are forbidden to send anything to the source, such that sources only have outgoing lines. This is easily done by setting all entries in the columns of the matrix corresponding to the sources to 0. Next we handle the sinks. In the data in the last Q rows it is stored with which points the sources are connected, but of course this definition needs to be the other way around: we need to know which points are going to send packages to the sinks. How this is done can be seen in the appendix. Each sink is going to be connected with each source. The amount of radiation to the source from the sink depends on the luminosity of the source: if a source A is twice as bright as source B, all sinks send twice as much to source A than to source B. Now all connections have been made and as stated above, all rows need to be normalized such that each rows sums to unity. As has been remarked in chapter 3, we finally transpose the matrix so we get the right kind of eigenvectors. This was the construction of the matrix in detail. All that is left to do is doing some science with it.

5.2 Results and discussion

As described above, all information has been stored in a matrix. I started this chapter by posting a few questions. I will answer these one by one using some field examples.

5.2.1 What does the stationary distribution look like?

From the constructed matrix, we need to find the eigenvector corresponding to eigenvalue 1. From the previous chapters, I know there exists exactly one such eigenvector. The function `eigs` in Matlab can produce the first few eigenvalues with corresponding eigenvectors. By calling the command `[V,D] = eigs(A,k)` the first k eigenvectors of matrix A will be stored as column vectors in matrix V and the corresponding eigenvalues will be stored on the diagonal of D. The eigenvalue with highest modulus is stored in the first entry of D, which as we have seen must be equal to one. The corresponding eigenvector, the stationary vector, is stored in the first column of V. The function normalizes the vectors on size, but since we are looking for distribution vectors, we normalize the vector again such that all entries sums to unity.
To visualize this eigenvector, I chose the following method. Each entry is mapped into a colormap from red to blue, where a high number will correspond with a more red color than a lower number. However, most entries are around the same number, so distinction between those values are quite hard. But since the entries are number between 0 and 1, taking the logarithm of the entries makes this distinction more apparent. Nothing has changed with the eigenvector on itself, just the way it is visualized. Since we have stored the coordinates of each point, we can make a 3D scatterplot, and each point will recieve the color corresponding with the logarithm of the value in the eigenvector, which gives the amount of radiation residing at that point. This leads to an intuitive visualization: the redder the area, the more radiation there is in that area.
In the figure on the next page is an example of a typical result. There are two sources with equal luminosity (arrows points at the location of the two sources), 2000 normal points and 200 sinks on a fixed distance in a sphere around the object. The points are connected with their 20 nearest neighbours. The color indicates the logarithm of the value of the distribution vector. As you can notice, the amount of radiation around the sources is high and declines as you move more and more away to the outer rims of the galaxies. That is exactly what you expect, since there is a smaller probability of going to the edge directly rather to be scattered around the object. I conclude that the method at least gives me realistic expected results.

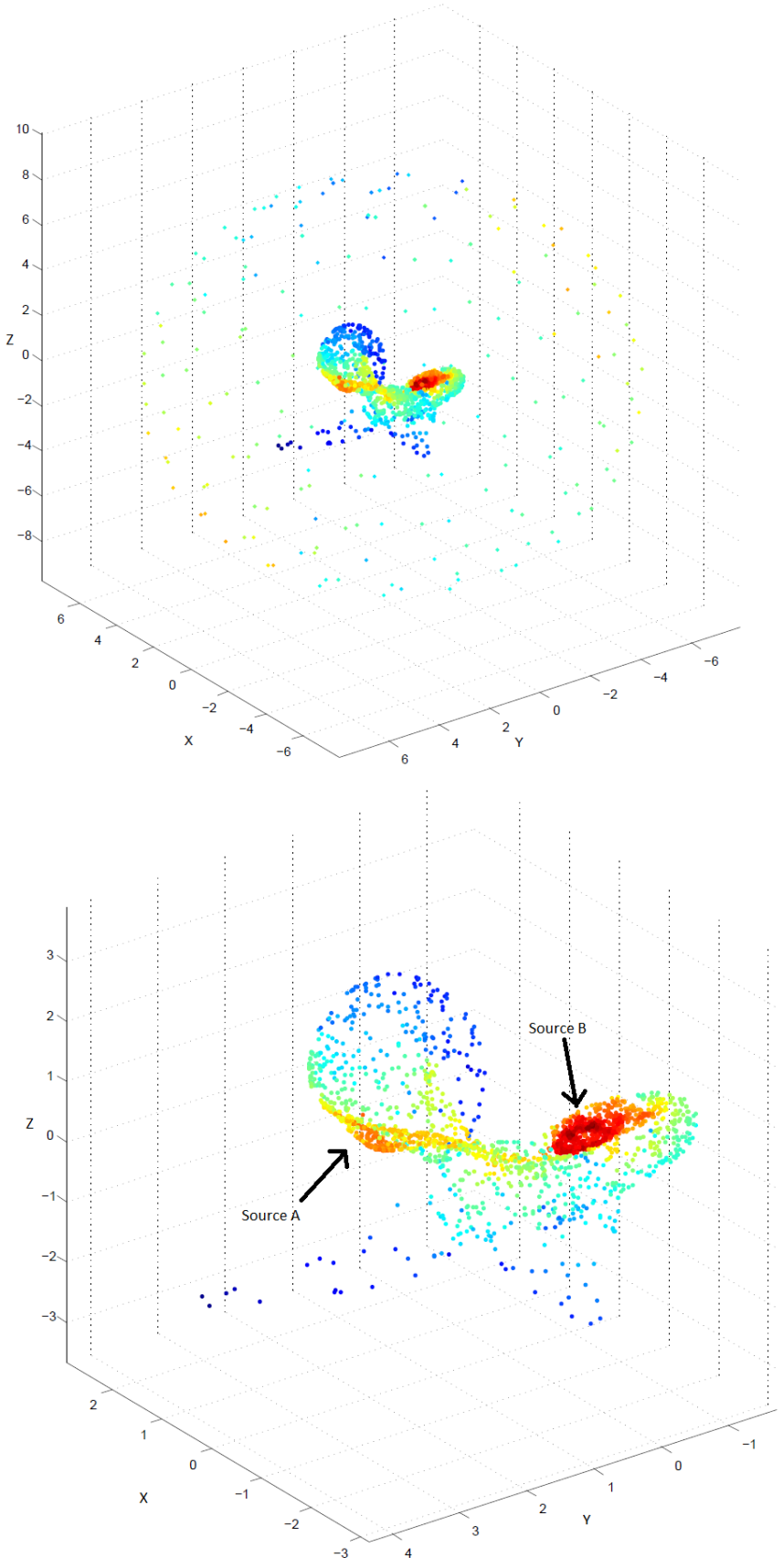


Figure 5: Stationary distribution vector, two colliding galaxies with in total 2 sources, 200 sinks 2000 normal points.

5.2.2 What is the distribution of eigenvalues for different kinds of point distributions?

This question is related to the question how fast our method will give us the stationary eigenvector. As is shown in chapter 3, one measure of seeing this is to compute the top eigenvalues and see if they are close to the eigenvalue 1. For the distribution above, the first 2200 eigenvalues are computed so I can get an idea of the full spectrum of the sparse matrix, and to get an idea how fast the eigenvalues will divert from eigenvalue 1 in absolute value, since the eigenvalues can be complex numbers. This is shown in the figure below. Below a table of the true values of the top 20 eigenvalues.

λ_1	1.0000	λ_8	0.9818 + 0.0004i	λ_{15}	0.9664
λ_2	0.9959	λ_9	0.9818 - 0.0004i	λ_{16}	0.9628
λ_3	0.9950	λ_{10}	0.9782 + 0.0007i	λ_{17}	0.9545
λ_4	0.9935	λ_{11}	0.9782 - 0.0007i	λ_{18}	0.9513 + 0.0004i
λ_5	0.9933	λ_{12}	0.9740	λ_{19}	0.9513 - 0.0004i
λ_6	0.9875	λ_{13}	0.9697	λ_{20}	0.9493 + 0.0010i
λ_7	0.9849	λ_{14}	0.9667	λ_{21}	0.9493 - 0.0010i

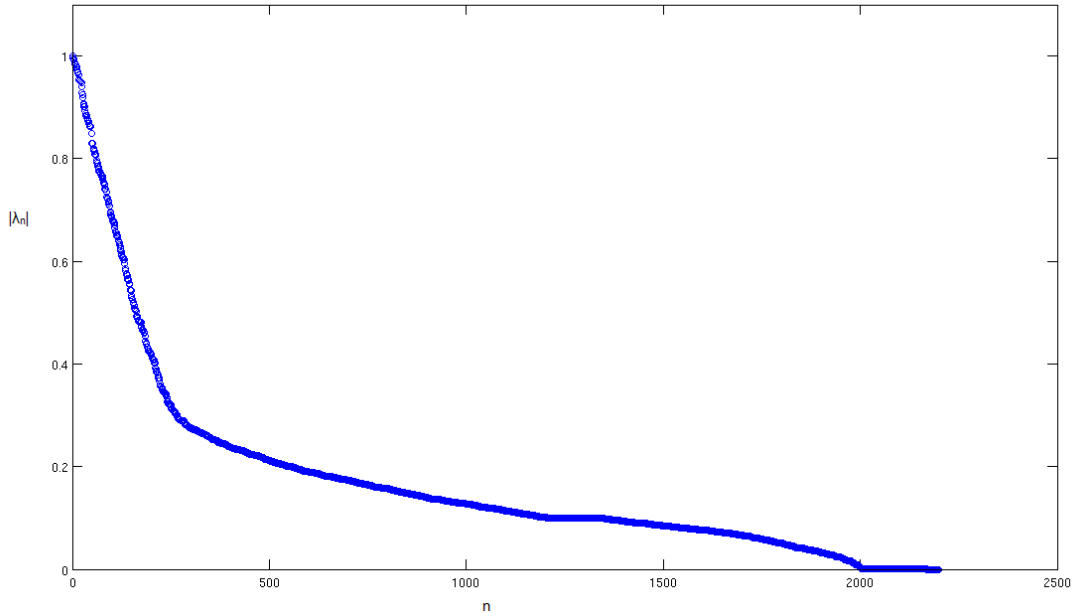


Figure 6: Graph of top 2200 eigenvalues of the distribution corresponding to figure 5

I also made the check and confirmed indeed that any other eigenvector not corresponding to eigenvalue 1 has a negative component. As can be seen, many eigenvalues are very close to one, with the second largest eigenvalue to be 0.9959. This suggests a slow convergence of the eigenvectors, which may be a problem if we scale the object with many more points than just 2202. I also looked at different type of distributions, and played with the following three parameters:

1. the way points are connected with the sinks
2. the number N of nearest neighbours
3. the luminosity of the two sources

Below a few results are given for each of these parameters.
First we need to think about the role of the sinks. We placed the sinks evenly around the object in a fixed distance. The first idea was to use the nearest neighbour strategy: connect the N nearest *normal* points with the sink. but this distribution is quite anisotropic and might not be

physical. The next idea, and used from now on, is to connect the N normal points with a sink that defines the largest angles. In the figures below are these two schemes showed, where you can see how the sinks are connected with the rest of the object.

Next I built the stationary vector with $N = 10, 20$ or 40 nearest neighbours, keeping the sinks as above and the sources equally luminous. The results are given in the plots on the next page, where you see the top down view of the two galaxies, the sinks are removed for clarity. From the plots I conclude the $N = 20$ case gives a more realistic satisfying outcome from the way radiation is distributed: too many neighbours makes the radiation more dispersed throughout the object, and otherwise the radiation is too concentrated around the sources. I am going to use this number of neighbours from now on. Also, I have plotted the eigenvalues for each of these cases. What you immediately see, is that the number of eigenvalues that have relatively high absolute value decreases faster with increasing number of nearest neighbours. Finally I played with the luminosity of the sources, keeping all of the above strategies for the sinks and number of neighbours. I made 11 runs by changing the way all the sinks send radiation back to the two sources. If we denote sourceA and sourceB by the two different sources, I first set the probability of going from the sink to source A to 1.0, and from sink to source B to 0.0. In each next run, I decrease the probability from the sink to source A by 0.1, and increase the other by 0.1. At each run, I again plotted the eigenvalues, but I see no significant difference in the way the spectrum is distributed: it follows exactly the same curve, only slight variations in individual eigenvalues. This may seem to be expected, since only a small portion of the matrix is changed in value (not in position). The results for two extreme cases are given in a next page.

5.2.3 Using the other eigenvectors to approximate the stationary eigenvector

One of the ideas that crossed the mind before starting with this thesis, was to somehow use the other eigenvectors to get something that looks like the stationary eigenvector. The reasoning for this began as follows: suppose two eigenvalues are very close together, or in our case if the second largest eigenvalue was very close to one, what can we say about their respective eigenvectors? Does the eigenvector corresponding to the second eigenvalue look like the stationary distribution? The answer for this is given by the Perron-Frobonius theorem stated in the earlier chapter. The answer is no, since the stationary distribution vector, the eigenvector corresponding to the eigenvalue one, is the only eigenvector with positive entries. This means that any other eigenvector, even the one with eigenvalue close to 1, must have a negative component, which has absolutely no physical meaning. Another idea was to get some kind of linear combination of the other eigenvectors, say the top twenty, that is very close to the stationary vector. Again this was futile, since the eigenvectors are nearly orthogonal to this vector, and therefore have minus signs on different entries. You must undergo a lot of trouble only to get a linear combination out of them that at least has all nonnegative components, and this procedure is just not useful.

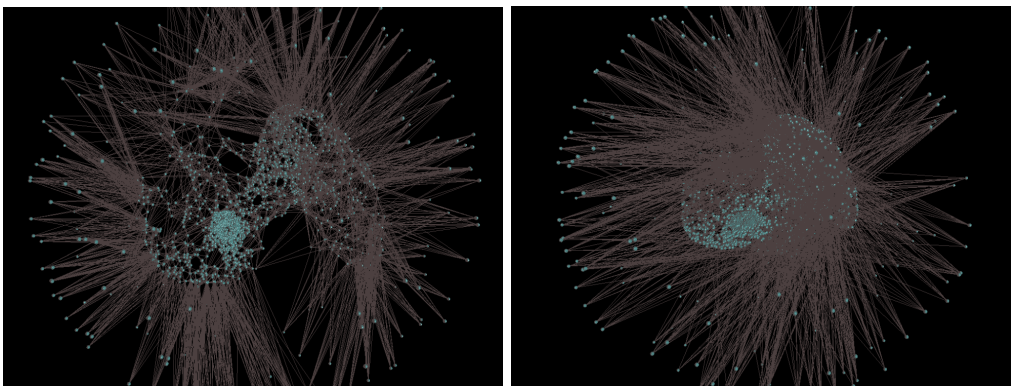


Figure 7: Two schemes for the connection of the sinks with the normal points. Left: Least physical distance. Right: Largest angular distance.

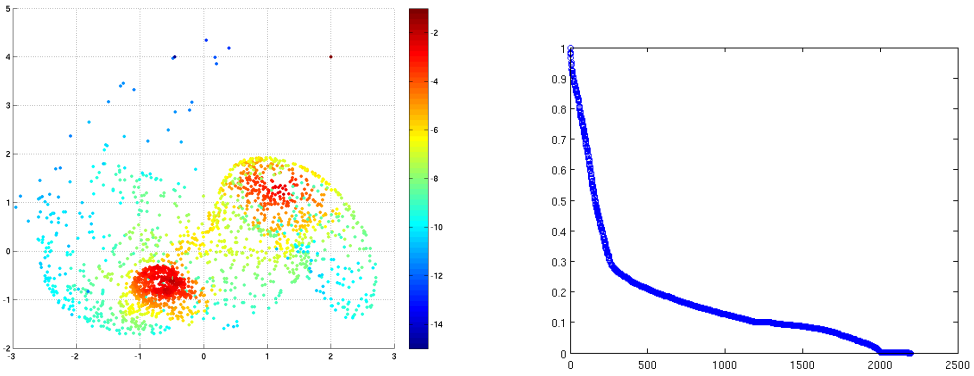


Figure 8: The stationary distribution and spectrum for N=10

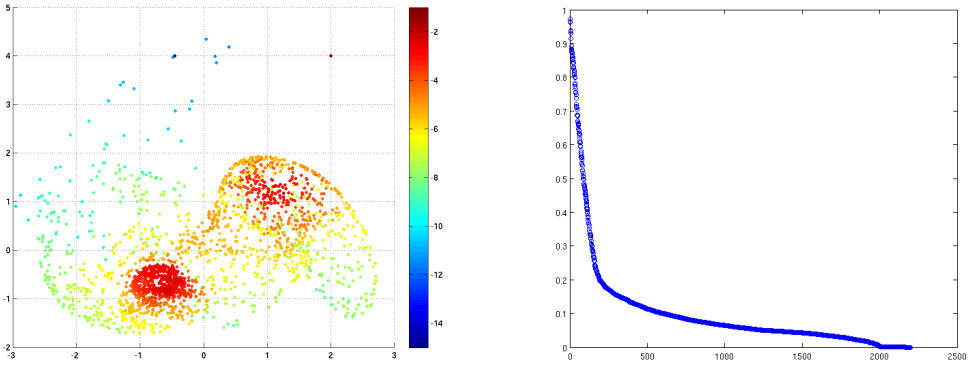


Figure 9: The stationary distribution and spectrum for N=20

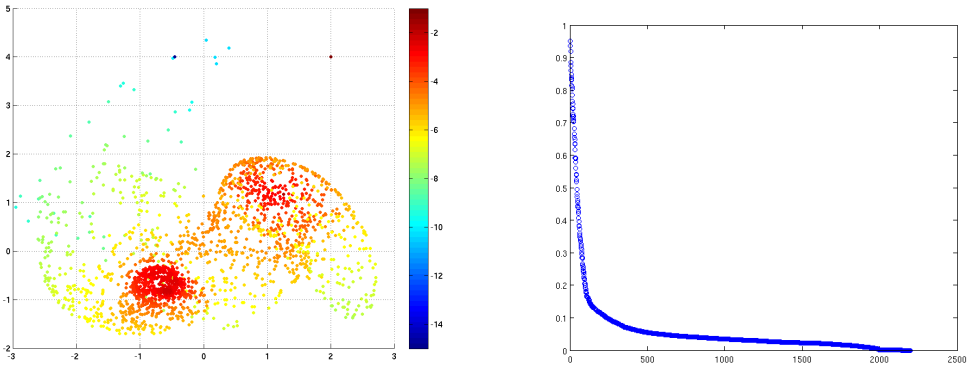


Figure 10: The stationary distribution and spectrum for N=40

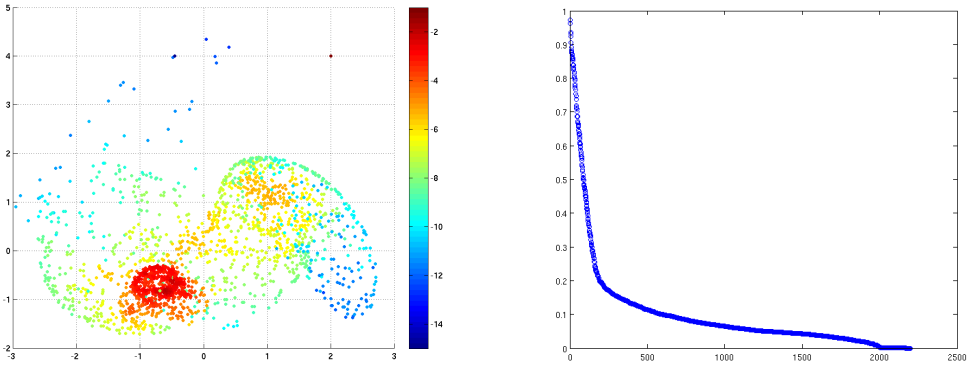


Figure 11: The stationary distribution and spectrum when source A recieves 10% from the sinks

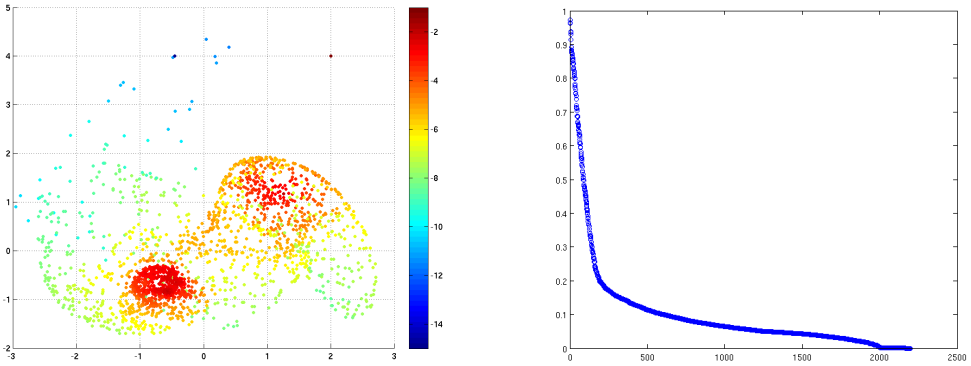


Figure 12: The stationary distribution and spectrum when source A recieves 50% from the sinks

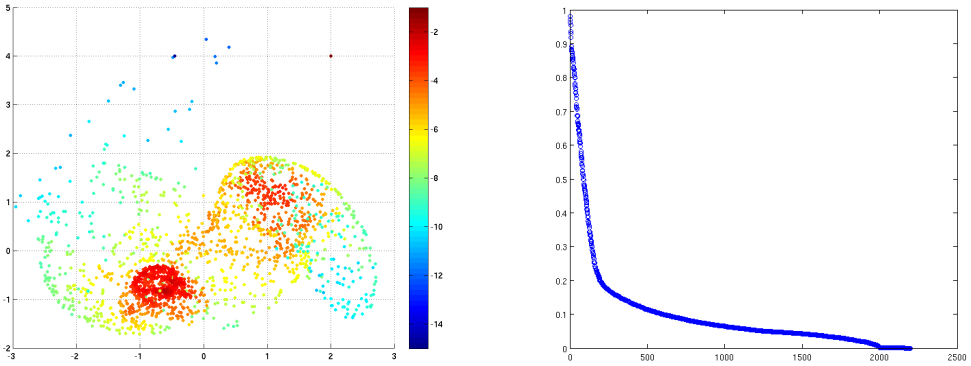


Figure 13: The stationary distribution and spectrum when source A recieves 90% from the sinks

6 Conclusion and further remarks

6.1 Quick summary

In the first few chapters I presented some theory on Markov chains and numerical linear algebra, which forms the basis of many questions, answers and strategies for a model of radiative transfer. The model can be summarized as follows: given a point distribution of an astronomical object, like two colliding galaxies, we constructed a directed graph between each point using a prescribed number of N nearest neighbours. Next two special type of points are introduced: sources with only outgoing lines and sinks with only incoming lines and virtual connections with each source. Next a scheme is made that represents the physics one is interested in, that is reflected in the way how one assigns all the probabilities of going from one point to another. All that information is stored in a stochastic transition matrix, and the information you are after is the eigenvector corresponding to eigenvalue 1. The Perron-Frobenius theorem tells us that when the matrix is irreducible and aperiodic, there always exists such eigenvector with eigenvalue one, it is unique and consists of only nonnegative components. By the sparsity of the matrix, one can use a version of the Arnoldi algorithm to find this eigenvector and also other eigenvectors and eigenvalues efficiently. To this end some examples were given. The goal was to experiment with the parameters of the model, namely the sinks, the number of nearest neighbours and the luminosity of the sources, to see if the method gives some physical satisfying results. Also the spectra of the sparse matrix was considered. The conclusion is that this model gives results that reflects what we actually see and expect, and therefore this method can be used to give model radiative transfer. For more on the latter we give a small discussion in the next subsection.

6.2 Discussion

The model that is presented is far from perfect. There are no true physical grounds on the choice of the number of neighbours, apart from computational considerations. However, in the SimpleX code, one uses the Voronoi-Delaunay tesselation, which leads to the fact that each point has approximately 15 or 16 neighbours. In my experimentations, I concluded that taking 20 nearest neighbours gave me more satisfying results, so there might be a connection there. Although one has to be careful at this point, you can always keep on tweaking this number untill you get a perfect agreement with reality, but you still have no idea of why this should work on physical grounds.

One of the aspects I also have considered, but did not work out far enough, is the possibility of absorption by the individual points. I made two considerations. First you can add a nonzero diagonal in the transition matrix: this gives the portion of radiation that is absorbed by the point itself. Say a point absorbs 40% of the radiation, then the other 60% will be re-emitted with it's nearest neighbours or sinks. In essence you build loops at points. This part I have done myself, the result is in the figure below, but there is something going on with the spectrum. In stead of decreasing like $1/N$, it contains bumps. The effect is more extreme the larger the absorption coefficient. I have not thought of a reason of this. Of course in the case of total absorption, the matrix is no longer irreducible, and in a few iterations you have your stationary distribution: all the components of the initial vectors that corresponds with the normal points keeps exactly the same value, apart of those with recieve radiation of the sources. At the first iteration all of the sinks have radiated everything to the sources and are then empty. At the second iteration all sources have radiated everything to their nearest neighbours and are then empty. And after that each iteration does not change anything. So there are no unique stationary distributions anymore, and you cannot say anything useful about it. Also the spectrum is broken in sections, there are a lot of eigenvalue 1, then with eigenvalue 0.67 etcetera. In the figure below this is shown for three cases: 20% absorption, 80% absorption and total absorption.

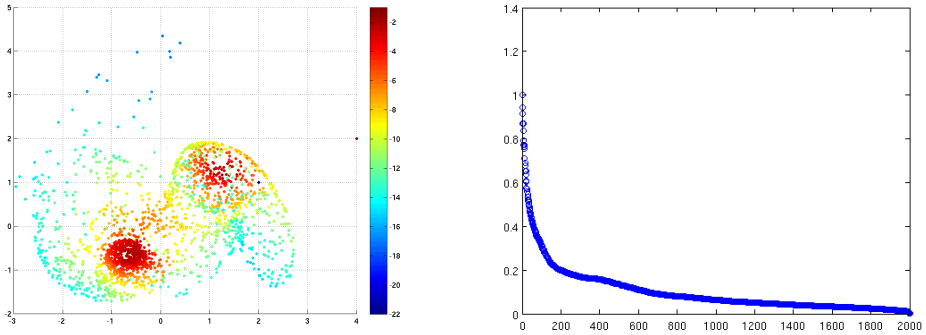


Figure 14: The stationary distribution and spectrum when all points absorb 20% of the radiation at each iteration

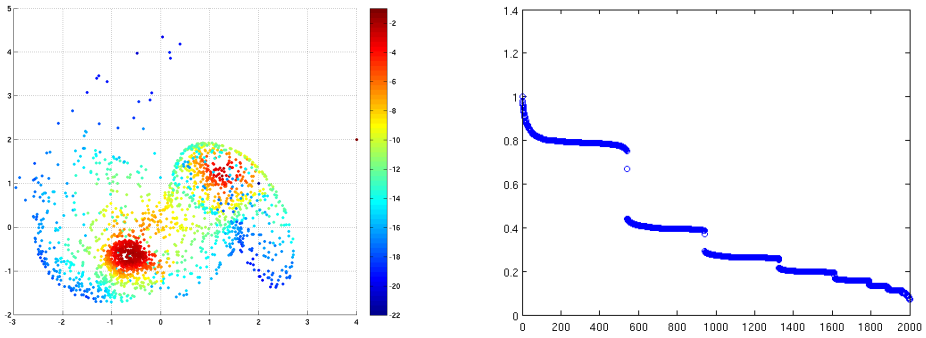


Figure 15: The stationary distribution and spectrum when all points absorb 80% of the radiation at each iteration

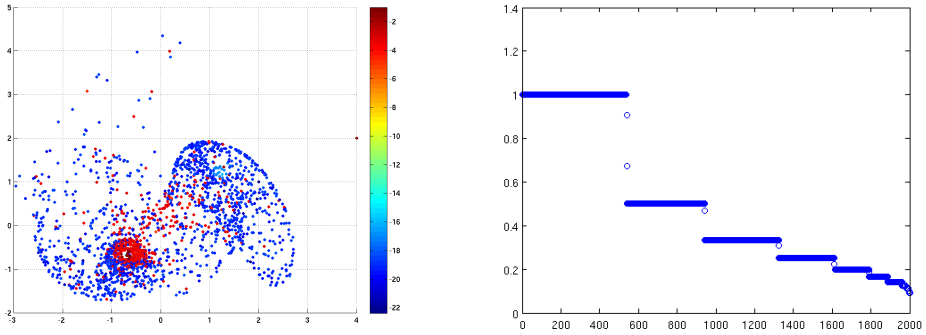


Figure 16: The stationary distribution and spectrum when all points absorb 100% of the radiation at each iteration

Another way for modelling absorption is to introduce the idea of halfsinks at each point: in stead of keeping the radiation at each point to re-emit at another iteration, we virtually connect each point with each of the sources, with the probability that reflects the absorption coefficient of that point: if the point is to absorp 40% of radiation, it first sends 40% back to the source, and then re-emit the remainder to it's nearest neighbours or sinks. I have not worked out this idea.

There are a lot of parameters one can tweak: the number of nearest neighbours, the location of the sinks, the way absorption is handled, the way probabilities are assigned between two connected points. But the most important thing to remark is the following: that given some astronomical phenomena, like colliding galaxies, you can think of some point process that resembles the problem. By using the presented method, you may be able to get back the distribution of radiation as observed with the telescope by tweaking all such parameters. That way you may get an insight on how radiation is being transfered within the object.

References

[Häggström] Häggström, O., *Finite Markov chains and algorithmic applications*, London Mathematical Society, student texts 52

[Norris] Norris, J., *Markov Chains*, Cambridge University press, 1997

[Press, et. al.] Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P.,*Numerical Recipes in C: The Art of Scientific Computing* Cambridge, UK: Cambridge University Press, 1992

[Saad] Saad, Y. *Numerical Methods for Large Eigenvalue Problems - 2nd Edition*, Society for Industrial and Applied Mathematics, 2011

[R&L] Rybicky, G.B,. Lightman, A.P., *Radiative processes in astrophysics*, Wiley, New York

[Okabe, et. al.] Okabe, A., Boots, B., Sugihara, K., & Chiu, S., *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams,* 2nd edn.,John Wiley & Sons, 1999

[Møller] Møller, J., *Random Tesselations in* R^d, Adv. Appl. Prob. 21, 37-73, 1989

[Seneta] Seneta, E., *Non-negative matrices and Markov chains*, Springer-Verlag, 1981

[Radke] Radke, R.J., A matlab implementation of the implicitly restarted arnoldi method for solving large-scale eigenvalue problems, Master thesis, Houston, Texas, 1996

[Meyer] Meyer, C.D., *Matrix analysis and applied linear algebra*, Society for Industrial and Applied Mathematics, 2001

[Paardekoper, Chael, Icke] Paardekooper, J.-P., Kruip, C. J. H., & Icke, V. 2010, A&A, 515, 79

[Ritzerfeld] Ritzerfeld, N.G.H., *The simplicity of transport: triangulating the first light*, Leiden, 2007

A The Matlab code

In this appendix my matlab code is presented. If you care to use this, please note that the file to be read in must have the following structure: say you have a total of N points, with $n_{sinks}, n_{sources}, n_{normal}$ points. Say you have for each point K connections (K nearest neighbours, or otherwise). Then you need to build a array with N rows, $K+3$ columns. In the first three columns, you save the spatial coordinates of each of the points. In the last K colomns, you insert the rownumbers of the points it is connected is. So: row 59 corresponds with point number 59, and say $\{2, 45, 367, 399, 567, 2235\}$ are it's six nearest neighbours ($K = 6$), than the final six columns of the data must consist of the numbers 2, 45, etc., in row 59.

In the code I also present two tricks for colormethods of the stationary eigenvector. Each comment is preceded with a % sign and colored blue. The normal code is black.

%This is the matlab code for the Bachelorthesis of Rick Vooys. In this code, a dataset will be read, analysed and reconstructed to a sparse matrix. This matrix will then be reconfigured

according to theory. After that the eigenvalues and eigenvectors will be calculated and a plot of the stationary vector will be constructed and displayed.

%READ IN DATA data = importdata('markov-0002-txt.txt'); %put between the quotes the filename

bsize = size(data); %to identify how many neighbours we are dealing with

datasize = bsize(2);%the number of columns of the data, the number of nearest neighbours equals datasize-3.

data(:,4:datasize) = data(:,4:datasize)+1; %datafile starts by counting with zero, matlab starts with one, so we add 1 to each entry.

numberofneighbours = datasize-3; %Number of nearest neighbours

% SPATIAL COORDINATES

X= data(:,1);

Y= data(:,2);

Z= data(:,3);

%From here we assume there are 2202 points in the datafile, one also could have used no.points=bsize(1); but that gave a weird error. So if you need a different number of points, change it accordingly! Of this 2202 points, the first few (2 here) should be the sources, then all the normal points (2000) and then all the sinks (final 200). If you have a different structure of this, change the following accordingly!

% CONSTRUCTION OF THE SPARSE MATRIX

rows = repmat((1:2202)',[1 numberofneighbours]); %build 2202xN array

columns = data(1:2202,(1:numberofneighbours)+3); %fetch the locations of the nearest neighbours

Matrix = logical(sparse(rows,columns,1,2202,2202)); %A trick with logicals. If punt B is a nearest neighbour of punt A, put a 1 in that row and column, otherwise, do nothing

Matrix = +Matrix; %convert logical to double

% SOURCES

for j=1:2202

Matrix(j,1) = 0; %Deny all points to send anything to the first source

Matrix(j,2) = 0; %Same for the second source

end

% SINKS

for sinknumber= 2003:2202

for j=1:numberofneighbours

kolomnrj = data(sinknumber,j+3); %identify the points that are linked with each sink

Matrix(kolomnrj,sinknumber) = 1; %connect such point with the sink

Matrix(sinknumber,kolomnrj) = 0; %do not let the sinks pass radiation back to the normal points

end

Matrix(sinknumber, 1) = 0.9; %SOURCE 1 will have a relative luminosity of 0.9 (change if needed)

Matrix(sinknumber, 2) = 0.1; %SOURCE 2 will have 1-luminosityofsource 1 (change accordingly)

```
end

%NORMALISATION (each row must sum op to unity)

for i = 1:2202

Matrix(i,:) = Matrix(i,:)/sum(Matrix(i,:));

end

%EIGENSPACE EXPLORATION

Matrix=transpose(Matrix); %Our definition was wrong, see thesis. This is the ductape of this code.

[V,D,flag] = eigs(Matrix,k); %Determine the first k eigenvalues (stored in D) and eigenvectors (stored in V), flag tells us if all eigenvalues converged

Statvector = V(:,1)/sum(V(:,1)); %normalise the stationary vector (such that it sums to one)

%PLOT THE STATIONARY VECTOR

scatter3(X,Y,Z,5,log10(Statvector), 'filled') %first the coordinates of each point, then the size of each point, then the vector to assign colors with (we take the log10, gives more insightfull plot, and say that each point must be fully colored (otherwise just colored circles in stead of disks)

%TRICK 1: Color adjustment Add two fake points, give it both a fixed maximum and minimum color, and add it to the plot. Note that this is only done in my thesis where I am not also going to plot all the sinks, for clarity.

Statvector(2003) = 10^(-1); %this just happen to be a great maximum value Statvector(2004) = 10^(-20); %dito minimum value

%we have to create the points in the plot, so add these coordinates

X(2003) = 2;

Y(2003) = 4;

Z(2003) = 0;

Z(2004) = 2;

Y(2004) = 4;

Z(2004) = 1;

%TRICK 2: lonely points. If a point is not a nearest neighbour of any other point, it will have lost all it's content in one iteration and will be 0 afterwards. In this particular dataset, there was 1 such point. So we ignore this point in the making of the plot (do not ignore it in the analysis of eigenspectra though!) This one was number 1065, and deletion of this point is as follows:

X(1065)=[];

Y(1065)=[];

Z(1065)=[];

Statvector(1065)=[];

%now we plot it again, without the sinks (note we added two pints, and deleted one, so there are a total of 2003 points to be plotted

scatter3(X(1:2003),Y(1:2003),Z(1:2003),10,log10(Statvector(1:2003)),'filled')

%SPECTRUM PLOTTING: construct a graph showing the eigenvalues, in absolute value, in descending order. Here k=2200 above.

for i = 1:2200
```

```
D2200(i)=D(i,i);

end

D2200=sort(abs(D2200),'descend');

plot(D2200,'o')

%END OF MAIN PROGRAM

%ALTERNATIVE: ABSORPTION During the matrix building phase, before normalisation. Add the following

%ADDING ABSORPTION: connect points with themselves

alpha = 0.3 %absorption coefficient, between 0.0 and 1.0. Change if needed

for i = 3:2002

Matrix(i,:)=Matrix(i,:)*(1-alpha)/sum(Matrix(i,:)); %Matrix only consists of 1's and 0's, after the point has absorpted a portion alpha, a total of 1-alpha of radiation remains to be redistributed along the neighbours.

Matrix(i,i)=alpha; % now whole of row i sums up to unity.

end
```