university of groningen / faculty of science and engineering / kapteyn astronomical institute

University of Groningen

Bachelor's thesis

Enhancing and expanding the applicability of ReverseRADEX for accurate astrophysical insights

Supervisor:
Prof. Dr. Floris F.S. van der Tak (SRON Netherlands Institute for Space Research, University of Groningen)

2nd Examiner:
Dr. Tim Lichtenberg (Kapteyn Astronomical Institute, University of Groningen)

Author:
Roos H. Voorhoeve (s4571401)

Abstract

Observations of molecular emission lines from interstellar gas clouds, particularly in the infrared and (sub)millimeter parts of the electromagnetic spectrum, contain valuable data regarding overall physical factors such as kinetic temperature and gas density. The existing software for efficiently extracting this information is limited and in need of enhancement. ReverseRADEX is a tool developed for obtaining such conditions from molecular line spectra, that can be applied to molecular clouds and star- and planet-forming regions. ReverseRADEX, was developed and thereby tested only for CO and the aim of this thesis is to extend its usefulness to other interstellar molecules, to increase the information that can be obtained about conditions such as kinetic temperature and gas density. Variability in accuracy in the results when testing different molecules, shows that further modifications and extended testing are needed to make ReverseRADEX reliable for real observational data. The thesis highlights the need for more advanced development and modifications for a wider range of applications in Astrophysics.

November 30, 2023

Contents

1 Introduction

Due to transitions in molecules in interstellar space, spectral lines can be observed in specific parts of the electromagnetic spectrum. The region between about 1.0 and 1000 μm, i.e. (far) infrared and (sub)millimeter, provides most information to study molecular clouds and star- and planet-forming regions. Molecular rotational and vibrational transitions that cause lines in this part of the spectrum happen at temperatures below 1000 K. In addition, at infrared wavelengths there is less extinction and lines are easier to observe. The extinction can be in the range of 0.1 to a few magnitudes per kiloparsec, however exact values vary depending on the specific characteristics of the region (Draine 2011).
The study of those line spectra, i.e. spectroscopy, provides a lot of information to characterize phenomena unfolding in those clouds and regions. To be more precise, it provides a wealth of knowledge about the composition, temperature, densities and several other physical conditions in those regions. For example, it is already known that when a solar-type star forms, it starts at temperatures of about 10 K and a density between 10^3 to 10^4 cm^{-3}, when cores collapses and the forming continues, the temperature can rise to tens or hundreds of kelvins and densities in the core can increase to 10^8 to 10^9 cm^{-3}. Besides, those regions show a rich and varied chemistry, which includes the presence of complex organic molecules[1] (Jørgensen et al. 2020).

The values of those conditions can be obtained by comparing line intensities in observed spectra to what is predicted by theoretical computations (LeBlanc 2010). These computations involve adjusting the molecule's abundance to enhance the alignment between theoretical spectral lines and the observed ones, thereby maximizing the accuracy of the fit. Depending on what condition you want to calculate, different parameters need to be considered. For example, when computing density, Einstein's A coefficient and collisional data are used and when computing kinetic temperature, the upper state energy in kelvin is used.

As observations do not directly contain relevant information about the source, astronomers use models to infer physical conditions from the line intensities. However, the tools to get the information quickly and in a practical manner are scarce. One of these tools is RADEX, which is a programme for non-LTE (local thermal equilibrium) models of interstellar line spectra (Van der Tak et al. 2007). Non-LTE circumstances occur when a celestial object lacks equilibrium between radiation and matter at specific wavelengths, this requires more complex models like RADEX to calculate properties (for further details see section 2.1.1).
RADEX is a radiative transfer code, used to calculate the intensities of atomic and molecular lines that are produced in a uniform medium. This programme is based on statistical equilibrium calculations covering both radiative and collisional processes. In addition, radiative contributions from background sources are included in the calculations.
The input consists of the geometry of the object, a molecular data file, the kinetic temperature (K), the density of collisional partner(s) (cm^{-3}), background temperature (K), column density (cm^{-2}), and line width (km/s). The output first summarizes the input parameters, followed by a line-by-line listing of upper state energy (K), frequency (GHz), wavelength (μm), excitation

[1] In this thesis, complex organic molecules are defined as carbon-bearing molecules with at least six atoms (Herbst et al. 2009).

temperature (K), optical depth, peak intensity (K), and line flux (both in units of K km/s and erg/s cm^2) (see section 2.1.3).

In an earlier bachelor's thesis the programme of RADEX was reversed (Van der Mooren 2021). As a result, line intensities can be specified as input and some of the input values of RADEX have become the output, such are the kinetic temperature (K), the column density (cm^{-2}) and the density of collisional partner(s) (cm^{-3}). This way the optimization process of matching model spectra to observed line spectra is automated, which ensures that the values are computed faster. That is to say, one run in ReverseRADEX produces the same results that would require multiple runs in RADEX.
ReverseRADEX has been developed specifically for CO and the goal now is to extend it to other molecules. This programme can be highly valuable to obtain those values about line intensities, for which no easy solutions are currently available. The extension to other molecules increases its value, as the information that can be obtained with the spectral lines of CO is limited and will be extended when data from other molecules can be applied as well. This way even more information about temperature and densities can be collected.

This thesis is dedicated to testing and improving ReverseRADEX for broader molecular applicability and higher accuracy, to the extent required. Hundreds of interstellar molecules are known and new ones are still being discovered. H_2 is the most abundant, therefore its density is widely used to estimate physical conditions in interstellar space.
Furthermore, this thesis aims to enhance usability by making ReverseRADEX more accessible and easier to use. In this way, it can be used with certainty in practical applications by astronomers and other interested parties.

The structure of this thesis is as follows. Section 2 discusses the required background information and theories, including a further explanation on RADEX and RevereseRADEX. Section 3 introduces the method used and how the research was conducted. Section 4 presents the results with the appropriate explanations. Section 5 discusses the result and suggests possible follow-up steps. Finally, section six presents the conclusion.

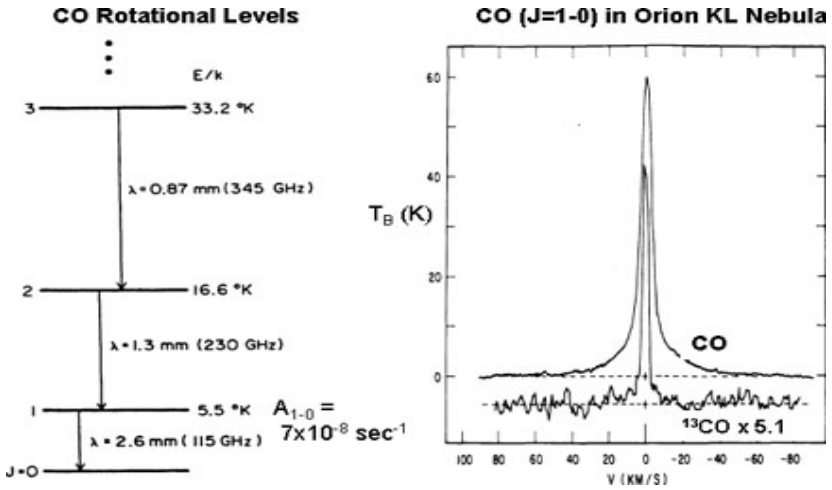

Figure 1: The rotational levels of CO on the left, with corresponding wavelengths. The graph on the right shows the CO emission lines in the direction of the Orion Kleinmann-Low nebula[2]

[2] Image credits: https://ned.ipac.caltech.edu/level5/Sept12/Scoville/Scoville8_1.html

2 Theory

2.1 RADEX

As announced in section 1, RADEX is a software tool, designed for non-LTE modeling of line spectra. It serves as a radiative transfer programme, specifically employed to simulate the excitation and radiation transfer processes within interstellar molecular clouds. In the following sections, the features and applications of RADEX will be discussed in greater detail (Van der Tak 2017).

2.1.1 Non-LTE

When gas is very dense, collisions between particles become crucial. At these high densities, the equations that describe the behavior of atoms or molecules can be simplified by using the Boltzmann equation (see section 2.3.1) .
In these high density conditions, the distribution of particles in different energy states depends mainly on the kinetic temperature of the gas (T_{kin}). This simplification happens because in these conditions, collisions are the dominant factor affecting the distribution of states. However, in more general scenarios, other processes, like radiation, also play a role. In these cases, the populations of different states not only depend on temperature, but also on factors like gas density and the entire range of internal radiation that interacts with the atom or molecule through absorption and stimulated emission.
A Local Thermodynamic Equilibrium (LTE) in stellar astrophysics means that not only internal states and molecular motions but also ionization balance, molecular abundances, and the local radiation source function at all frequencies are assumed to be in equilibrium at the same temperature. The local aspect referred to each layer of depth in a layered atmosphere where temperature and density could vary. In these cases the excitation temperature (T_{ex}), that is the temperature used in the Boltzmann equation, equals T_{kin} everywhere (for further explanation on the Boltzmann equation see section 2.3.1).
There exists also quasi-LTE, where a constant or fixed T_{ex} is assumed. For these cases, T_{ex} is the parameter in the Boltzmann distribution, which has a different value for each species in the quasi-LTE case (Van der Tak et al. 2020).
Non-LTE assumes that the system is not in thermodynamic equilibrium at every point, which means that various physical conditions, such as temperature and density can vary throughout the region or cloud. For non-LTE T_{ex} generally has a different value for each transition. For both quasi-LTE and non-LTE T_{ex} can deviate significantly from T_{kin}.
A celestial object is in a state of non-LTE when the interactions between particles are not frequent enough to establish equilibrium, which is often the case in low-density environments. In addition, at typical temperatures in molecular clouds, collisional ionization is usually not a significant factor. Therefore, non-LTE calculations are more complex, as the Boltzmann distribution no longer holds. This is where RADEX will come into play, as it allows properties, e.g. T_{ex}, to be calculated in these circumstances.

2.1.2 Applications

The output of RADEX helps to understand a wide range of astronomical problems, the main application is to the interstellar medium with a focus on star- and planet-forming regions (Van der Tak et al. 2020). With the created spectra of molecular lines, physical conditions as well as chemical composition can be studied. Therefore, the understanding of complex processes in different astronomical environments is broadened. RADEX is particularly useful in the field of

molecular astrophysics. A couple of circumstances where RADEX can be applied will be outlined below.

Primarily, the programme is applicable to molecular clouds. Molecular clouds are a type of interstellar cloud, which can be divided into diffuse and dense molecular clouds. For diffuse clouds the typical density is between 100 and 500 n_H (cm^{-3}) and the typical temperature is in the range of 30 to 100 K. For dense clouds the density is larger than 10^4 n_H (cm^{-3}) and the temperature lies between 10 to 50 K (Snow et al. 2006). The high density permits the formation of molecules, while the high column density prevents their destruction by UV radiation. The most commonly formed molecule is molecular hydrogen (H_2), hence these clouds are characterized by high H_2 density. In the inner regions star formation takes place, possible due to the high density. Star-Forming regions consist mainly of gas and dust, where dust grains play a very important role in regulating physics and chemistry. Some molecules are formed on their surfaces and dust absorbs and scatters starlight, which causes starlight passing through interstellar clouds to be attenuated. RADEX allows, by obtaining the line spectra, to model the excitation and radiative transfer of the molecular lines in the different parts of these dense clouds. Besides the temperature and density, abundance of various molecular species is especially important in these places. This is crucial to learn more about those regions' physical conditions and chemical composition and is made possible by the use of RADEX.

Another important implementation of RADEX is the interstellar medium. For example, obtained data on diffuse and translucent clouds also help astronomers to investigate their composition and physical and chemical properties. Translucent clouds have higher densities than diffuse ones, but lower densities than dense clouds. Their densities fall in the range of 500 to 5000 n_H (cm^{-3}) (Snow et al. 2006).
It should be noted that these are only some examples and other applications are possible.

2.1.3 Input parameters

RADEX uses different input variables as discussed in section 1, some parameters will be explained further. It starts with entering a file from The Leiden Atomic and Molecular Database (LAMDA) of the molecule that is taken into consideration and the desired name of your output file. LAMDA collects spectroscopic information and collisional rate coefficients for molecules, atoms, and ions of astrophysical and astrochemical interest (Van der Tak et al. 2020). The files follow a homogeneous data format that is used by RADEX, the format is straightforward and flexible[3].

After the molecular file you have to fill in the range of the frequency (GHz) in which you want the spectral lines, the kinetic temperature (K), the column density (cm^{-2}), the line width (km/s) and geometry. In addition, the number of collisional partners and the corresponding densities (cm^{-3}) are also indicated. It is possible to apply multiple collision partners simultaneously. However, the number of available partners is restricted to seven options; H_2, p-H_2, o-H_2, electrons, H (atoms), He, and H^+ (Van der Tak et al. 2007). Collisional rate coefficients for all relevant partners must be provided. Most cases suffice with H_2 as the only partner.
Finally, the background temperature is also one of the input variables of RADEX. If you enter a positive value, it means a blackbody spectrum at that temperature, for instance, like the

[3] The LAMDA files can be found here: https://home.strw.leidenuniv.nl/~moldata/

CMB. A value of zero represents the average interstellar radiation field. For a negative value the user has to supply a list of observed flux densities. This thesis takes into account only the CMB (2.73 K), due to shortcomings in ReverseRADEX (section 2.2.7)

2.2 ReverseRADEX

RADEX was reversed in an earlier bachelor's project, which resulted in the programme ReverseRADEX (Van der Mooren 2021). The reverse implies that the inputs and outputs have changed, as elaborated in section 2.2.1.

2.2.1 Input parameters

To use this programme, some input values that also come with RADEX still need to be specified, such are the background temperature, linewidth, geometry and a LAMDA file for the corresponding molecule. Additionally, a file with (observed) line intensities must be provided, this has to contain the frequencies (GHz), the intensities, with the preferred unit specified by a #1, #2 or #3 in the first line of the file, that represents the radiation temperature in K, flux in K km/s or flux in erg/s cm^2 respectively (Langevelde 2008). Furthermore, the standard deviation is optional to enter, this represents the observational uncertainty. If no value is specified, the value 1 is automatically assigned with the corresponding selected unit of the intensity.
Besides, the boundaries for parameters that you want to fit need to be supplied. The possible parameters are the kinetic temperature (K), the column density (cm^{-2}) and the collisional partners with their corresponding density (cm^{-3}). Initial guesses are also given with the parameters, those are used if the parameters do not need to be fitted.
You can either enter all the input by using a configuration file or fill them in manually. As output the code will provide values for the parameters that are picked to fit. Besides the values for those parameters combined with uncertainties, two types of plots are created, one about the uncertainty in the reproduction of the observations and a corner plot of the parameter space. Currently the code is only developed and tested for the CO molecule and data created by SpectralRadex is used, instead of real observations.

2.2.2 Wrapper

The original version of RADEX was largely written in Fortran. Fortran is a compiled programming language, which means that it translates machine code from source code. This can lead to faster execution times compared to Python, for example. The latter is an interpreted language, here for more steps are needed to get machine code, which can make the code run slower. The main advantage of Fortran is that it is very efficient and accurate when dealing with numerical operations. On the other hand, this language is generally more difficult to read, as it is very close to machine code (Decyk 2007). The advantage of Python is that it is easier to read for people who are not experts in programming.
Therefore, Python is more accessible to use and several Python wrappers have been created for RADEX. In Van der Mooren (2021), several of those wrappers have been tested. Eventually, the wrapper SpectralRadex is used (Holdship 2020). This one turned out to be the most accurate and even faster than RADEX itself. Since that project was already done two years ago, new wrappers have also been created by now, an example is radex-python (Megias 2023). However, other wrappers are disregarded for this thesis, but can certainly be of interest for follow-up studies.

2.2.3 Algorithms

ReverseRADEX makes use of three algorithms to get the estimates of the physical conditions, which are being fit. The algorithms are connected by a chain (see figure 2). The aim of the first step is to find a global minimum in a chi-square landscape, this is done by the brute-force method (see section 2.2.4). Accuracy can be improved in this step, for this reason this method will be explained in further detail in the following section.
The chain continues with a non-linear least squares algorithm, i.e. Levenberg-Marquardt, this step will refine the parameter estimates. The programme ends with an MCMC algorithm, to estimate the uncertainties.
The advantages of chaining the algorithms is that this will reduce biases and the computation time of the intensive MCMC algorithm, because it will only need to obtain uncertainty estimates, instead of having to search the whole parameter space for estimates.
The last two algorithms mentioned are not further explored or improved upon in this thesis and are therefore not considered in more detail. For further explanations, please see Van der Mooren (2021).

2.2.4 Brute-force method

The initial guess of the parameters is done by the brute-force method. This is a simple grid search of the entire parameter space, specified by the user. The grid search is complemented by SpectralRadex's feature set, which makes this wrapper even more convenient. The appropriate number of points to assess for each parameter is dynamically adapted by the process. While minimizing the computational time, considering the user-defined bounds and imposing minimum and maximum constraints to ensure efficient sampling. The goal of this step is to find initial parameter estimates, to pass on to the next algorithm to further refine the outcome. This is done by applying the chi-square formula (1) to the observed and LAMDA data files.

$$\chi^2 = \sum_i \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

This formula checks the difference between the observed (O) and expected (E) value of line strengths and divides it by the uncertainty. A smaller result indicates an observed value that is closer to the expected data of the LAMDA files, a value of zero means that the corresponding line strengths are equal. For the best fit, this formula searches for the lowest value.
The aim of finding a proper first estimate can be effectively achieved if there are enough observations to restrict parameter degeneracy. The degeneracy can be a concern in this step, because only the smallest value of chi-square is chosen, which minimizes the difference between observed and standard data. Due to the stepsizes in the grid search, there may be even better values between those steps. However, the process would be greatly slowed down by introducing smaller steps, so this is a compromise between accuracy and speed.
As far as the code has been tested, it has not yet caused any issues. Thus, a single-mode solution is provided and there is no need for an overly fine sample yet.

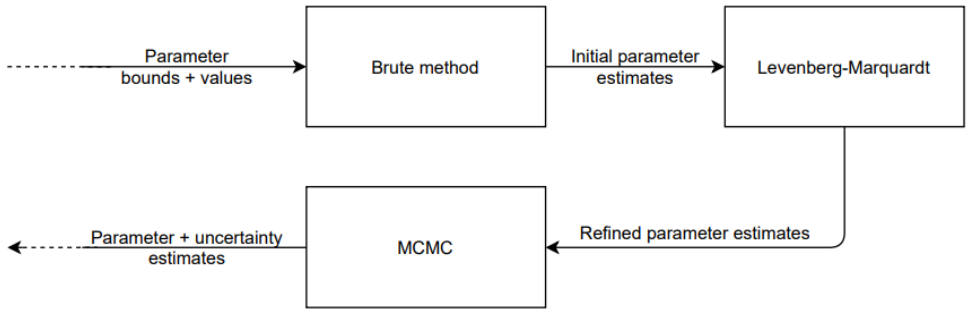

Figure 2 : A flow diagram of the algorithm chain used in ReverseRADEX (Van der Mooren 2021)

2.2.5 Applications ReverseRADEX

ReverseRADEX is applicable to similar situations as described for RADEX in section 2.1.2. Which of the two programmes is more practical depends on what data is available. In some cases it can be easier to observe line intensities, than to obtain the kinetic temperature or column density. Developments and new instruments available can provide more data that can be used in ReverseRADEX, allowing it to be used more often (2.2.6). Whether line intensities or RADEX's parameters are available, in both cases it is interesting to learn more about physical conditions in different environments. Hence, ReverseRADEX is applicable to the similar regions and astrophysical problems as RADEX and is preferred over RADEX when line intensities are more easily obtainable.

2.2.6 Observational instruments

To observe line intensities of molecules in interstellar space, different telescopes and instruments can be used. The main types of telescopes for this purpose are radio telescopes, infrared telescopes and (sub)millimeter telescopes, e.g. Five-hundred-meter Aperture Spherical radio Telescope, Spitzer Space Telescope and Atacama Large Millimeter/submillimeter Array (ALMA) respectively.
Which telescope is best to use depends on which part of the electromagnetic spectrum you want to observe. As the names suggest, the radio telescope is suited for radio and microwave lengths, the infrared telescope for the infrared part of the spectrum and the last one for the submillimeter and millimeter wavelength range. As introduced in section 1, these telescopes cover the range where most of the vibrational and rotational transitions take place, namely 1.0 to 1000 µm. Therefore, this range is the most valuable as it provides the most information.
When gathering data there are several restrictions to take into consideration. A high spectral resolution is essential to distinguish individual molecules and to measure velocities of gas motions. The telescopes must be highly sensitive to detect weak molecular lines.
Besides, the earth's atmosphere absorbs radiation of several wavelengths, primarily in the ultraviolet, optical and infrared regions of the electromagnetic spectrum. As those regions are important when it comes to observing spectral lines, observatories are located at high-altitude observatories or on spacecraft to minimize atmospheric interference (Karttunen et al. 2003).
As an example, ALMA is located at an altitude of 5000 m and one of its surveys has a spectral resolution of 0.25 km/s and about 300 lines per GHz can be identified (Jørgensen et al. 2020)

2.2.7 Restrictions and requirements

There are certain restrictions involved in using ReverseRADEX. First of all, the code is not supported on windows. This problem could already have been caused by the use of SpectralRadex. In addition, certain versions of Python packages and libraries are required. If certain versions are not complied with, it is not guaranteed that the code is still completely functional.
Similar to RADEX, there are seven possible collision partners that are supported, namely H_2, H, e−, p–H_2, o–H_2, H^+ and He. Whereas RADEX has multiple options for the background temperature, it is currently not supported by ReverseRADEX to use other values than the CMB (2.73 K), since only this value is considered to be adopted for SpectralRadex (Holdship et al. 2020).
Furthermore, the user is expected to take into account effects on the observed data themselves, e.g. doppler shift. Finally, ReversRADEX operates only as a terminal application or as a .ipynb notebook and lacks a graphical user interface.

2.2.8 Areas for improvement

As discussed in the section 2.2.4, the initial guess of RevereseRADEX is done with brute-force. Therefore, important values with high potential could possibly be ignored in the grid search.
One way to solve this is to decrease the step sizes, which causes the first step to increase in time. This initial step can be done in a more accurate and efficient manner.
A different global optimization method may improve the results, without significantly increasing the computation time. A high potential method could be the Bees Algorithm, which is an intelligent optimization technique that is part of the swarm algorithms field (Baronti 2022). By means of a parametric objective function, it searches for the values that minimize or maximize the output. This algorithm will be discussed in greater detail in section 2.4 .
Furthermore, the code has not yet been tested for other molecules nor for real data. To ensure that it can be properly applied to both options, improvements can also be made in the user-friendliness of the programme and the ease of solving errors that arise when using it.
Finally, there are also potential improvements in the other two algorithms, but that is beyond the scope of this project.

2.3 Molecules

Interpreting molecular line observations is the main objective of this thesis. Therefore, it is important to understand certain concepts. Various quantities exist to describe molecules and their line spectra. A few of them that are provided in LAMDA files and that are relevant for this thesis will be explained further. Besides, the distinction between linear and nonlinear molecules will be explained.

2.3.1 Properties

Spectral lines are caused by radiation and transitions within a molecule, transitions can be vibrational and rotational. When a molecule emits a photon it falls back from a higher energy level to a lower energy level, an emission line is created. In the case where energy is absorbed

through a collision, the molecule gets to a higher energy level, the process creates an absorption line (Draine 2011).
Rotational transitions represent changes in angular momentum, following allowed selection rules. Vibrational transitions involve changes in the vibrational energy of a molecule, typically associated with absorption or emission of infrared radiation, following the fundamental vibrational modes of the molecule (Gupta 2016). The energy levels of all transitions are quantized and the differences correspond to characteristic frequencies. These frequencies give rise to the unique spectral lines and are provided in the LAMDA files.
The energy in Kelvin is also represented in those files. This value is related to the temperature, as an increase in energy implies an increase in kinetic temperature. Therefore, this value influences the population of different energy levels within molecules, which can affect the prominence and behavior of spectral lines observed in a system at that temperature. The probability distribution that describes the statistical distribution of particles in different energy states in a system is given by the Boltzmann distribution. It provides the probability of finding a particle in a particular energy state and is given by:

$$P_i = \frac{e^{-E_i/(kT)}}{Z} \qquad with \qquad Z = \sum_i e^{-E_i/(kT)} \tag{2}$$

E is the energy of the corresponding state, T is the excitation temperature and k is the Boltzmann constant.
For LTE T_{ex} has to be equal to T_{kin}, but for non-LTE T_{ex} is in general smaller than T_{kin}. Besides, this parameter has a different value for each transition in non-LTE circumstances.
Another important parameter displayed in the LAMDA files is the Einstein's A coefficient. This number is used to describe the rate of spontaneous emission of photons from an excited molecular state, to a lower energy state. It quantifies the probability per unit time of this emission for a particular transition in s^{-1}. An increase in this number signifies an increase in the optical depth and hence an increase in density.
Furthermore, the number of collisional transitions and the corresponding temperatures are provided by LAMDA. This collisional data is also an indicator of the density, as a higher collisional rate suggests a higher density. Therefore, the combination of the collisional data and the Einstein's A coefficients determine the density.
The frequencies of the lines function mainly to distinguish them from each other, depending on which parameters you want to calculate (e.g. T_{kin}) you need lines that add new information to the calculation, i.e. lines that have different values for the described properties.

2.3.2 Linear vs nonlinear molecules

Based on molecular geometry a distinction can be made between linear, i.e. diatomic, and nonlinear molecules. Both have distinct characteristics that differentiate them from each other and also affect the spectral lines they produce.
One of the characteristics is the symmetry. Linear molecules have a high degree of symmetry, with only one principal axis of rotation. Because of this, they have two rotational degrees of freedom, which refers to the number of ways a molecule can rotate in space. These transitions are quantized and simplify their energy level diagrams and their spectral behavior. By contrast, nonlinear molecules have lower symmetry, as they possess multiple axes of rotation and have three rotational degrees of freedom. Therefore, they introduce more complexity to analyze their line spectra (Delaire et al. 2000).
Furthermore, linear molecules have distinct energy spacings due to their symmetrical geometry, leading to specific wavenumber positions. Conversely, nonlinear molecules have more complex energy spacings due to their asymmetrical geometry.

In conclusion, linear and nonlinear molecules differ in their molecular structure and symmetry, which results in different spectral characteristics. These differences have influence on the spectral lines in various ways. As linear molecules produce typically simpler line spectra than nonlinear molecules this distinction is being taken into account in the methodology of this thesis.

2.4 BeesAlgorithm

As discussed in 2.2.3 and 2.2.4, the initial guess step of RevereseRADEX is done with brute force. Therefore, important values with high potential may be missed when executing the grid search. This initial step can be done in a more accurate manner. There are several tools that this can be done by, MAGIX is one of them.
It serves as a framework with an easy interface to connect existing numerical codes, employing an iterating engine to minimize deviations between model results and observational data. This way best-fit values can be determined in a given parameter space (Möller et al. 2020).
Considering the advantages and disadvantages, no further attempt will be made to implement MAGIX into ReverseRADEX, for it is an external dependency that is too generalized, slow, inconsistent and produces inaccurate parameter estimates as well as offers no support for the Windows platform (Van der Mooren 2021).

Another option which has not yet been investigated is BeesAlgorithm, which is a search method that solves optimization problems. This algorithm has potential to replace the initial guess step and will therefore be explained further.
From examining other algorithms not many possibilities came up, the only other possibilities resemble BeesAlgorithm, examples are Ant Colony Optimization, Grey Wolf Optimizer and Whale Optimization. Hence, BeesAlgorithm will be investigated first in this thesis.

BeesAlgorithm is a search method that solves optimization problems. It is inspired by nature and mimics the behavior of honey bees (Pham et al. 2005). This metaphor uses how bees help each other find nectar and the different tasks involved. It starts with random exploration of the fields, done by a part of the population. The bees are looking for sources that are near the hive and with an abundance of nectar. When the bees return they communicate their findings by doing a ‘waggle dance’. In response to this, a number of foragers join the scout bee to exploit the advertised area. The highest promising areas are visited by the largest number of foragers. The process repeats itself, to ensure that a large number of the colony will harvest the highly profitable sources.

The algorithm starts by sampling the solution space, done by artificial bees that function as a population of agents. Then the scout bees start random exploration, which involves searching the whole space with uniform probability.
Each scout bee evaluates the visited site, which represents a solution, via the fitness function, the scouts that find regions with the highest fit in this global search will return with the waggle dance that recruits forager bees.
In this step exploitation takes place, the search will continue near the most promising regions in a local search. Both local and global search are repeated until the best fitting solution is found or a certain amount of iterations have passed.

In the local search, neighborhood shrinking causes the size of the hood to decrease when it fails to bring improvement in fitness. If it becomes clear that there are no improvements in the area, the site will be abandoned and a new random solution will be generated. The process is repeated until the best value is found. A visualization of the process is shown in figure 3.

An example where BeesAlgorithm is applied successfully is to the selection features for manufacturing data. The problem in this process was about filtering irrelevant information from data caused by noise or data redundancy. BeesAlgorithm was applied to select an optimal set of features for a pattern classification. A combination of features that produces the lowest classification error had to be found. According to the results the algorithm worked properly (Pham et al. 2007).

A somewhat similar situation is the case for ReverseRADEX, where it is about finding the global minima in a chi-square landscape. The smallest values derived by the formula represent the smallest differences between the observed and synthetic data. The BeesAlgorithm brings benefits that rectifies where the brute-force method falls short.
The algorithm will always try to find a solution that maximizes or minimizes the objective function. In this case that function should be the chi-square formula (see section 2.2.4), applied to the data.

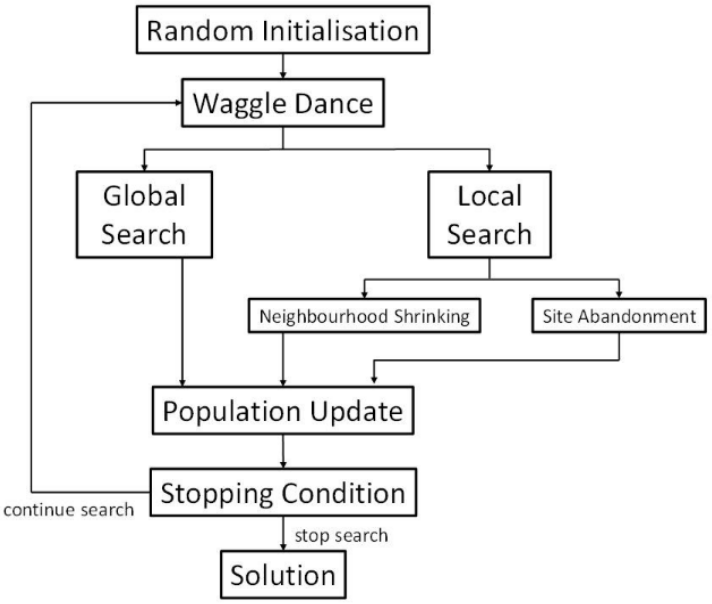

Figure 3 : A flowchart of BeesAlgorithm (Pham et al. 2005)

3 Methods

This section describes how ReverseRADEX was developed and tested for molecules other than CO. To get the code working for different molecules, it will be tested on several aspects and modified to ensure that it can be applied to line spectra other than just CO. The following sections discuss how this is done and what factors are taken into account, broken down into different steps and components.

3.1 Input Data

Firstly, the code is tested by inserting spectral data files produced by RADEX. In this manner the expected values are already known, making it convenient to verify the results. When the results are as foreseen, the code can be tested for real observations. This has not yet been done for any molecule, including CO, so this molecule will also be tested then.
For the molecular data file the LAMDA database is continued to be used. Although the files are very similar, they are not identical. To preserve the code operating, it needs to be generalized, so that other files will also be read successfully. This will be the first step to fix, before different molecules will be observed.

3.2 Background temperature

As explained in section 2.1.3, the background temperature is a constant parameter for RADEX as well as ReverseRADEX. Even though this parameter can have different values in RADEX, varying from negative to positive values, this is not yet supported by ReverseRADEX. Therefore, this parameter is fixed at 2.73 K throughout this research, which is the temperature for the CMB.

3.3 Molecules

As explained in section linear vs nonlinear molecules, there is an overall difference between the line spectra of linear and nonlinear molecules. Due to the fact that linear molecules have simpler line spectra, only those kinds of molecules will be tested in this thesis.
The molecules that will be tested are HCl, O_2 and CH^+, the specific choice of these molecules, besides being linear, is arbitrary. Some characteristics in which the molecules differ is that CH^+ is a light hydride ion (13.02 g/mol), while CO is a heavy neutral molecule (28.01 g/mol). HCl is heavier than CO, weighing 36.46 g/mol, and has hyperfine structure. Furthermore, O_2 has a $^3\Sigma$ ground state, which influences the allowed rotational transitions.
Table 1 shows the frequencies of the lowest transitions of CO, HCl, O_2 and CH^+, taken from the LAMDA files. They cover different frequency ranges and the gaps between the frequencies also vary. Both CO and CH^+ have spectral lines that are widely spaced. On the other hand, HCl and O_2 have multiple spectral lines which hardly differ from each other, which can make it more difficult to distinguish between them.

	CO	HCl	O_2	CH^+
Frequencies (GHz)	115.2712018 230.5380000 345.7959899 461.0407682 576.2679305 691.4730763 806.6518060 921.7997000 1036.9123930 1151.9854520 1267.0144860	625.90160300 625.91875600 625.93200700 1251.43434000 1251.43434000 1251.44705650 1251.45056850 1251.45193000 1251.45193000 1251.46391000 1251.48091000	52.02142300 52.54241810 53.06693350 53.59577700 54.13003000 54.67118400 55.22138600 55.78381300 56.26477280 56.36339700 56.96821200	835.1375040 1669.2812909 2501.4404523 3330.6296656 4155.8719545 4976.2013942 5790.6657535 6598.3290559 7398.2740442 8189.6045320 8971.4476267

Table 1: The frequencies in GHz of the lowest transitions of CO, HCl, O_2 and CH^+ based on their LAMDA files

3.4 Configuration files

When using ReverseRADEX, a configuration file can be used to enter all input parameters and files, instead of filling everything in manually for each run. In the current code, when using a configuration file, only the file named config.ini is read, even if a different name is used on the command line. As it is more convenient to create different configuration files, than changing the config.ini one, every time you want to run the code with different input, it will be made operative for other file names. If a wrong name is used, the code should stop running and not automatically use the config.ini. Besides, it is important that manual input also continues to work.

3.5 Testing

When the code works for all LAMDA files and different configuration files can be used, the testing of different molecules will be executed. This also includes varying different parameters and possibilities for input, so that as many upcoming bugs within the code as possible can be fixed. Depending on how the outcomes turn out, different trials will be conducted on the following aspects;

- Various number of RADEX produced lines, varying from four up to eight spectral lines
- Manually changes in input lines, to simulate a situation that could possibly be closer to real data, where small differences are more likely to occur
- Similar frequencies, this is only possible to test with HCl and O_2 , where differences of less than one GHz are available (table 1). In the other cases the gap between frequencies is clearly distinguishable
- Different bandwidths, which is the deviation an observed frequency can have from a frequency in a LAMDA file. This has to be changed within the code and is done for values between 1 and 0.001 GHz
- Different spectral ranges of the lines, the range in which lines occur differs per molecule, the ranges used vary from 50 to 5000 GHz
- With and without a provided standard deviation of 10%
- Varying the initial estimates in the configuration file

- Fitting two instead of three parameters, to see if this affects the parameters that will still be fitted, indicating the correct value with the initial guess

For convenience, the background temperature, linewidth and geometry are kept constant, with values of 2.73 K, 1.0 km/s and 1(uniform sphere) respectively. In addition, H_2 is the only considered collision partner and its density has a value of 10^4 cm^{-3} for O_2 , 10^5 cm^{-3} for CH^+ and 10^8 cm^{-3} for HCl. So three parameters are fitted by default, except in cases where testing is done to fit fewer parameters.

3.6 Data analysis

For visualization of the data, the two different plots produced by ReverseRADEX and the numerical outcomes displayed are used. Based on comparisons of these 3 different results, the data will be analyzed and conclusions will be drawn from this.

3.7 Different initial guess algorithm

As discussed in section 2.4, the initial guess step of RevereseRADEX can be replaced by several tools, based on the theory the option that will be further examined and implemented in the code is BeesAlgorithm. This algorithm should increase the accuracy of the initial step, without a significant increase in time.
This step will be implemented after the rest of the methodology has been successfully performed.

4 Results

The results are divided into three sections. Firstly, the changes made as preparation of the code are presented (section 4.1), this contains updates to run the molecular files, getting different configuration files working and acquirements of the spectral data files. Then the findings of testing for molecules, among the various factors described in the method, wil be shown (section 4.2).
Finally, section 4.3 provides a beginning to a possible next step to make the code work more accurately.

4.1 Preparations

4.1.1 Input molecular file

The code is generalized to accommodate all the currently available LAMDA files that were previously unsupported. The start and end points for where the code is to be read are indicated now by multiple options that are common in the LAMDA files, defined by ‘trans’ and ‘numbr’. For the existing files it holds, if other variants of the start and end statements used in the code appear, they can be added. This should be done in the file ReverseRADEX/user_input/read_user_dat.py and then be inserted in the definition get_molfile_frequencies in lines 124 and 125, as shown in figure 4.
If a data file is used that is not from LAMDA, it must be in the same format to be suitable.

```
110     def get_molfile_frequencies(self, molecular_file):
111         """get all the frequencies as floats in a list from the
112         selected molfile.
113 
114         Args:
115             molecular_file (str): file location on system of molecular file.
116 
117 
118         Returns:
119             list: list of frequencies with float type.
120         """
121 
122         contents_molfile = self.get_file_lines(molecular_file)
123 
124         trans = ('!TRANS', '! TRANS', '!Transition','!TRANSITION', '! TRANSITION')
125         numbr = ('!NUMBER', '! NUMBER', '!Number')
126         index_lower = None
127         index_upper = None
128         for molfile_line in contents_molfile:
129             if molfile_line.startswith(trans):
130                 index_lower = contents_molfile.index(molfile_line)
```

Figure 4: The function that reads the LAMDA file, ReverseRADEX/user_input/read_user_dat.py

4.1.2 Configuration files

A couple of modifications enabled the use of other names for configuration files. The user can enter ‘python main.py -config’ followed by any name that represents a configuration file that is in the directory where you run ReverseRADEX.
The file needs to be in the same format as the original config.ini file and be recognized as a .ini file, typified by the colors of the texts, the format can be seen in figure 5. If it meets both requirements, it can have any arbitrary name and does not have to specifically end with ‘.ini’. Different configuration files were also used while testing the molecules and worked properly.

In addition, an ‘help’ argument is added. When the user runs the script with the –help option they will see a description that clarifies how to insert the configuration file.
The part of the code adapted for this purpose can be found in the appendix A.

When only ‘ python main.py’ is entered on a command line, the programme still asks to fill in all input manually and it will not automatically read the original config.ini file, which was the case before.

```
# Configure the ReverseRADEX settings
[PATHS]
reverseradex_dir = '/Users/users/voorhoeve/VIRGO01/RADEX/REVERSERADEX'
# path to LAMDA compliant format molecular data file
MolecularFile = '/Users/users/voorhoeve/VIRGO01/RADEX/data/ch+.dat'
# path to (observed) spectral data file
SpectraFile = '/Users/users/voorhoeve/VIRGO01/RADEX/radexdata/radexdatach+.dat'

[CONSTANT_PARAMETERS]
# background radiation field temperature [K]
BackgroundTemperature = 2.73
# line width [km/s]
LineWidth = 1.0
# (1=uniform sphere, 2=LVG, 3=slab)
Geometry = [1, 'uniform sphere']

# [name parameter, value if not fit, (bound_low, bound_upp), fit parameter?]
[VARIABLE_PARAMETERS]
# kinetic temperature: 0.1 < tkin < 1e4 [K]
Tkin = ['tkin', 100.0, (30.0, 500.0), True]
# column density: 1e5 < cdmol 1e25 [cm^-2]
Coldens = ['cdmol', 1e14, (1e10, 1e18), True]
# volume densities: 1e-3 < coll partner < 1e13 [cm^-3]
# NOTE: the indentation is needed to correctly interpret 'Voldens' as a dict.
Voldens = {'h2':(1e5, True),
           'h':(0.0, False),
           'e-':(0.0, False),
           'p-h2':(0, False),
           'o-h2':(0.0, False),
           'h+':(0.0, False),
           'he':(0.0, False),
           'min_max':(1e3, 1e8)}
```

Figure 5: The format of the configuration files

4.1.3 Acquirements spectral data file

For the data file the user has to supply, several points are important to pay attention to and be aware of. The general explanation of the input file can already be found in Van der Mooren (2021). When entering a file you need to make sure there are no empty lines at the bottom, as those will be read and will cause an error. To make the programme more user friendly, comments are added to tell the user empty lines need to be removed and the code gets shut down by an error saying ‘empty lines found in data file’, see figure 6.

```
    def get_file_lines(self, data_file_location):
        """get the file lines of user supplied data file in a list.

        Args:
            data_file_location (str): file location on system of data file.


        Returns:
            list: list of file lines of the user supplied data file.
        """
#         return data_file_lines
        with open(data_file_location, 'r') as data_file:
            # Read all lines from the file
            data_file_lines = data_file.readlines()

            # Check for empty lines
            if any(line.isspace() for line in data_file_lines):
                print("Warning: The file contains empty lines. Please remove them to continue the process.")
            # You can choose to stop the code here if you want
            # Uncomment the following line to stop the code
                raise ValueError("Empty lines found in the data file.")

        return data_file_lines
```

Figure 6: The part of the code that checks for empty lines in the data file, ReverseRADEX/user_input/read_user_dat.py

In addition, lines need to provide new information to be relevant for the fitting process (see section 2.3.1). When the code is executed with lines that have a difference in frequency that is too small, this will not be recognized as such new lines or information. The code stops with an error of not fitting shapes, as the second frequency will not be used as input so it misses one value.
This was tested with a difference in frequency of about 0.1 GHz, the code still worked in this case. This is demonstrated by the spectral lines with frequencies 56.2648 GHz and 56.3634 GHz of O_2, where a difference of 0.0986 GHz was still sufficient. As compared to the spectral lines of HCl, of 2499.8480 GHz and 2499.8876 GHz, which were interpreted as an error with a difference of 0.0396 GHz. Therefore, the difference between frequencies that is sufficient is approximately between 0.1 and 0.04 GHz.
In all cases there was a significant difference in the corresponding intensities, which did provide new information about the spectral lines besides the frequencies. For HCl, the intensities had values of 5.145×10^{-5} and 2.567×10^{-3} erg/s cm^2 , this amounts to a difference of $2.578\ 10^{-2}$ erg/s cm^2. Therefore, these specific lines were not expected to produce an error, the direct cause of this could not be found nor resolved. When using the programme, you should take this into account when providing observed spectral lines.

A variable in the code that might have something to do with this problem is the bandwidth, defined in ReverseRADEX/user_input/read_user_dat.py. If there is an observed line that cannot be found within the bandwidth in a LAMDA file, that line is not included. However, this shows a different error than the shape error that is referred to, this new error is discussed further in section 4.2.4.

What could be a possible explanation is that multiple observed lines are appended to one line of a LAMDA file, because they fit in the range of the bandwidth. In this case multiple lines will have the same index in the LAMDA file, which results in duplicate observed frequencies, and then it only returns the first match. So only one LAMDA line is used for multiple data lines, while there were several possibilities. Thus, the extra lines do not provide new information that is needed in the calculations. This problem will only occur if a molecule has spectral lines that are very close to each other, for the molecules tested this is the case for HCl and O_2. Table 2 shows some of their spectral lines and here it can be seen that the differences between some lines are very small (i.e smaller than 1GHz). Different bandwidths have been tried and do not solve this problem. Further causes of failure to read similar lines and the possible role of bandwidth will also be discussed in section 4.2.4.

4.2 Molecule Testing

Running the code never gives you the exact same output twice, even if nothing changes to the input values. This was observed in many trials of the CO molecule and subsequently with other molecules. The differences are caused by the MCMC algorithm, which uses random parameter combinations to decide which combinations of parameters is best. It is not completely random, as it tries to find the best combination by prioritizing which combinations are closer to the observations and not continuing to try parameter combinations that are further away. In this process, small differences may arise every run.
However, the differences are small enough for the estimates to lie well within the 16% and 84% standard deviations, i.e. one sigma below and above the best estimate, so the value for an

identical run is always in that range. An example is shown in figure 7, which shows output of two times the exact same run of CH^+. For the kinetic temperature the difference is only 0.00193 log10(tkin(K)), this is a power of ten smaller than all the error margins. For the column density the difference is 0.02499 log10(cdmol(cm^{-2})), which is also smaller than those corresponding errors. Lastly, the difference for the H_2 density is 0.0238 log10(H_2(cm^{-3})), this is two powers of ten smaller than the provided errors.
After this general observation, the factors examined will now be discussed in more detail

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 1.99605 | -0.01399 | +0.01509
log10(cdmol): 14.31489 | -0.50218 | +1.14067
log10(h2): 4.70761 | -1.12546 | +0.46263
```

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 1.99412 | -0.01361 | +0.01512
log10(cdmol): 14.28990 | -0.46411 | +1.09908
log10(h2): 4.73141 | -1.06015 | +0.44050
```

Figure 7: Two runs of identical configuration files of CH^+

4.2.1 Initial estimate

The input of ReverseRADEX contains an initial estimate in the configuration files in the range of the values for the kinetic temperature, column density and volume density of the collision partner(s). The specified initial estimate is used only if it is chosen not to fit the parameter, otherwise the code is indifferent to changing this value. When you provide the programme with a different initial estimate for one of the three variable parameters, the output of all the three parameters changes, only because that happens for every run, as already covered.

4.2.2 Uncertainties

When no uncertainties are provided in the observed data file, the programme assigns the value 1 to it, with the corresponding selected unit of the intensity (2.2.1). This is very big in relation to the values covered by the uncertainty, which range from powers of 10^{-2} to 10^{-10} (erg/s cm^2). So even for the highest values in the range, the uncertainty is powers of ten greater than the original value. In the cases where the code is tested without provided uncertainties, the results are very arbitrary. This can be clearly seen in the corner plots, figure 8. Figure 8.a does not show a clear peak for the median value. Instead, it is a more constant line with insignificant minima and maxima. The values are still reasonably close at times, but by the corner plot this seems to be more of chance than a distinct best value.
Conversely, when uncertainties are provided, the corner plot shows a more evident peak or maximum, see figure 8.b. An uncertainty of 10% of the original value was used during testing. Once again, values created by RADEX are used, for real observations the uncertainty can be different and so will be the output.
Even though the values in the first case seem more random than the second, the results with a clear median are not necessarily getting better. The uncertainties decrease but the end values are in some cases even better when they are connected to the quite indistinct corner plot.

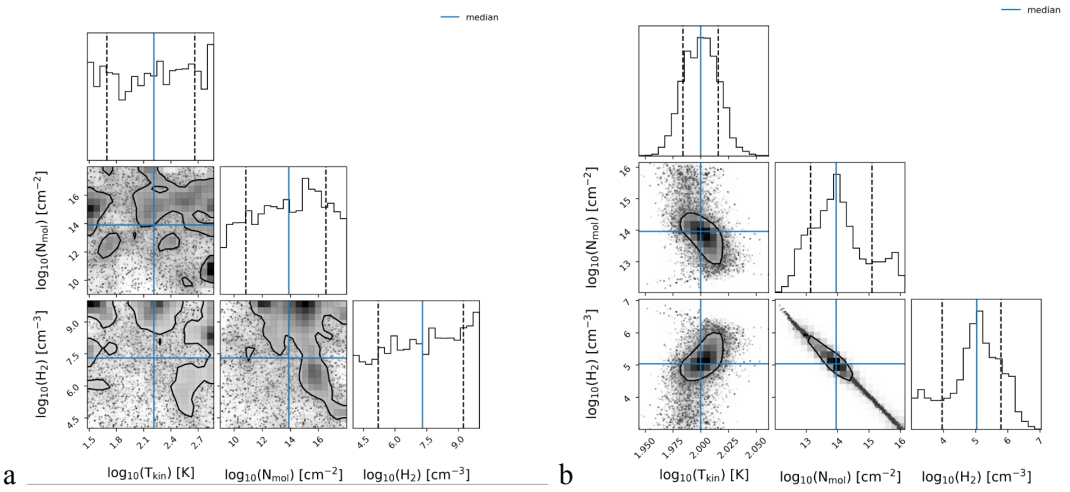

Figure 8: The corner plot of similar input without provided uncertainties (a) and with 10% uncertainties (b)

4.2.3 The number of spectral lines provided

Throughout all the runs, the number of spectral lines in the observed data file are also altered at times. This starts from four lines, which is the lowest number to still be able to fit three parameters, and is done till eight lines. To judge this factor properly, the focus was on the runs with very accurate outcomes, to see how they evolve when adding or subtracting lines. Accurate outcomes implies that the difference between estimate and expected value is well within the error margins. By testing this specific factor, supplied uncertainties of 10% were used.

For CH^+ very accurate values were obtained for five, six and seven lines. With rounding to two decimal places, the kinetic temperature was equal to the expected value. The column density was no more than 0.45 log10(cdmol(cm^{-2})) off the expected value and for the H_2 density it was no more than 0.43 log10(H_2(cm^{-3})) off the expected value. Adding more lines did not show an increase, nor a decrease in accuracy of the results. The differences were as small as the differences that always appear when running the code (section 4.2). For four lines the outcome became worse. This is shown in table 2, for five, six and seven lines, the outcomes are close to the expected values and with the errors taken into account, they fall within that range. By contrast, this is not the case for four lines, even with the error margins taken into account, the expected outcomes do not fall within these ranges. The values in the table are rounded up to two decimal places, the complete numbers can be found in Appendix B.

The spectra graphs show that the data points and the RADEX median parameters no longer overlap when 4 lines are used, whereas they did with the other number of spectral lines (see Figure 9).

By contrast, HCl got very accurate values for four lines. Namely, 2.0 for log10(T_{kin}(K)), 13.82 for log10(cdmol((cm^{-2})) and 8.06 for log10(H_2(cm^{-3})), where the expected values were 2, 14 and 8 respectively. Here the values are also rounded and full numbers with errors can be found in Appendix B too.

Therefore, it is not generally true that more lines are better. The correlation between the number of lines and accuracy of the result is different for each molecule, which depends on the quality of the data and the fitness of the supplied lines to the parameters.

In both cases the lines were spread over relatively large ranges, which might have a positive effect.

	4 lines	5 lines	6 lines	7 lines	Expected values
log10(T_{kin}(K))	1.74 (-0.00967, +0.00999)	2.00 (-0.01576, +0.01534)	2.00 (-0.01444, +0.01467)	2.00 (-0.01435, +0.01461)	2
log10(cdmol(cm^{-2}))	15.69 (-0.62995, +0.67694)	14.06 (-0.90010, +1.49218)	14.45 (-0.58524, +1.38347)	14.21 (-0.90757, +1.54870)	14
log10(H_2(cm^{-3}))	3.39 (-0.68303, +0.63554)	4.95 (-1.46091, +0.82453)	4.57 (-1.37012, +0.55557)	4.80 (-1.51103, +0.85621)	5

Table 2: The estimates for CH^+ with different number of spectral lines

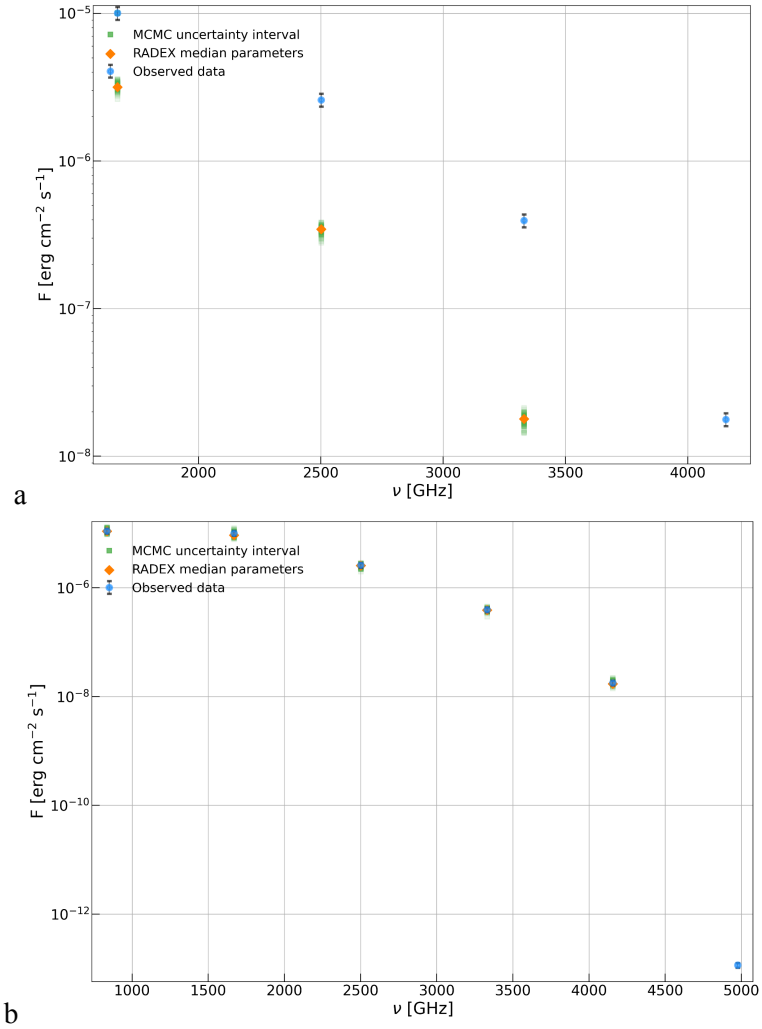

Figure 9: The spectrum graphs of four (a) and seven (b) spectral lines of CH^+

4.2.4 Manual changes

As discussed in 4.1.3, the bandwidth has influence on how far frequencies may deviate from the values in LAMDA files. To see what is possible with this variable and what could possibly happen if real observations are used with most likely larger deviations, testing was done with manual differences in the RADEX data and different bandwidths.
The bandwidth has normally a value of 0.001 GHz, and is tested for values between 0.001 and 1 GHz, as it has to be smaller than 1. However, the bigger the value is, the worse the output gets. The size of the changes in data is correlated with the bandwidth. For a change of 0.1 GHz in one of the frequencies applied with a bandwidth of 0.1 GHz, the output is of similar degree of accuracy, including the uncertainties, as in the case of a bandwidth of 0.001 GHz and no change in frequency.
If the bandwidth is too small to the corresponding change in data, the spectral line will not be recognized and will stop the code from working. Therefore, the bandwidth has to be adapted to the expected deviation from real data
The margin in frequencies is also dependent on the molecule and its LAMDA file. The concern is that the frequency needs to be coupled to one of those from the LAMDA. So if there are many spectral lines that are very similar, larger differences are possible. However, lines might be coupled to a different frequency than what the original frequency should have been.

4.2.5 Additional factors

Fitting two parameters instead of three has been tested. For configuration files whose outcomes fit only one parameter perfectly, that is to say it was equal to the expected value, the code is executed to fit only the other two parameters. The parameter for which a good value had already been obtained, was excluded from the fitting process, by changing this in the configuration file. This ensures that the provided initial guess is used in the process, instead of the value being calculated. When one parameter was supplied in the configuration file, the outcomes did not change significantly, bearing in mind that outcomes are never exactly equal. Therefore, fitting one parameter less does not affect the accuracy of the other parameters.

For some molecules spectral lines from different spectral ranges have been tried in separate executions, this gave no particulars. For some molecules bigger ranges exist, or bigger ranges were needed to obtain enough workable lines, i.e. lines with big enough differences between each other to be read as different lines. For O_2 the range varied from 50 up to 1500 GHz and for HCl the range was from 500 to 2500 GHz. The largest range is used for CH^+, namely 500 to 5000 GHz, as this molecule has few spectral lines even for this large range, compared to the other two. The outcomes of applying lines from different parts of the mentioned ranges have some changes in them, this is just because different data is provided. Depending on the quality of the provided spectral lines, the fits will get better or worse, there is no general conclusion to be drawn here.

4.2.6 The overall quality of the results

Besides being tested on different variations of different factors, the accuracy of the outcomes were checked in all cases. How well fitted these were, varies from one situation to another. In the worst cases, the expected value was not even in the range of the standard deviations and

values were off the correct ones by a factor of 10^3. Acting on the variations to make them more favorable for each molecule sometimes improved the results, but no clear trend could be seen in doing so.
As discussed in section 4.2.2, with or without provided uncertainties gives very different corner plots, where the difference is whether or not a clear median is visible. However, providing specified uncertainties does not guarantee an outcome closer to the expected value, as it varies which case had the best final estimate of the parameters. This is surprising and may suggest that some algorithm is unstable. This also applies to the other tested variations. There are no outstanding trends that can be observed. The outcomes are reasonably close, that is to say that in general the right values fall at least between the 16 and 84% standard deviations, where overall at least a factor of ten is still taken into account. However, there are also cases where this is not even the case and the values are too far off to even be in that range. While those cases could have been proper runs when you take into account all the provided input.

The 16 to 84% standard deviations also vary a lot for different circumstances and for the different parameters and can reach high values. For example the errors for the column density varied from -0.06469, + 0.07669 to -4.36104, +3.96618, where these values are also in log scale. This shows that in some cases this can even reach up to several factors of 10 from the estimated value. This happens mainly for the two density parameters and is particularly the case when there are no provided uncertainties.
Depending on how accurate the user wants the results to be in order to be of value, the code now suffices in some cases, but is not yet reliable to use for real data, for which the right values are not known. Therefore no real data has been tested now.

4.3 Different initial guess algorithm

As the results were not yet satisfactory, the first improvement was adopted to optimize the first algorithm, according to section 3.7. Given limited time and in some areas shortcomings in programming knowledge, it has not been possible to fully implement this algorithm. The code that is written so far and debugged to some extent can be found in Appendix C.

5 Discussion

The discussion reveals specific topics of the thesis that could be improved and offers potential prospects in pursuit of improvement. Since the results were generally not satisfactory, the code should first be improved until the desired result is obtained.
Section 5.1 will discuss the possible causes and solutions to the accuracy of the outcomes. Section 5.2 will address follow-up steps that can be taken if the entire code works more properly for all molecules.

5.1 Improving code performance

Given that it is not clear why the code does not produce consistent valid results, several steps will be reconsidered. The extent to which the accuracy of the outcomes needs to be improved depends on the user and the scientific goal. In some cases it suffices when the value falls within a factor of ten of the expected outcome, in other cases it should be more precise. This also depends on the quality and amount of available data.

5.1.1 Wrapper

To start with the applied Python wrapper. Before SpectralRadex was applied to ReverseRADEX it was tested. In essence, good values are produced by this wrapper. However, since then, new wrappers have been developed that might be more efficient. An example that could be worth studying further is the wrapper radex-python (Megias 2023). Compared to SpectralRadex, this code is written much more compactly and consists of only one file, whereas SpectralRadex uses multiple directories with multiple files. This makes it easier to retrieve and debug errors that occur when ReverseRADEX calls radex-python instead of SpectralRadex.
Additionally, in SpectralRadex no other background temperatures than the CMB are supported. However, different background temperatures are possible in radex-python and there could be other wrappers that support this too. The implementation of a different wrapper to ReverseRADEX can make it applicable to circumstances where other sources than the CMB need to be considered to determine the background temperature, for example star formation activity. Therefore, this modification can make ReverseRADEX more widely applicable.

Since a wrapper is called at several places in the code and sometimes errors arise that cannot be accessed properly for this reason, it could be a good option to develop a new reverse programme in Fortran code. The advantage is that this is in the same language as the original code and so the whole code is easier to make it fit to RADEX. In addition, Fortran is a better language than Python when it comes to scientific computing, especially when dealing with large datasets. The disadvantage is that Fortran is a dated programming language and so is not used very often anymore. So to be able to execute this, someone has to be proficient in this language. In addition, for the same reason, adjustments cannot be made quickly by others, which means the code is not very accessible to adapt or improve.

5.1.2 Algorithms

The attempt to implement Bees Algorithm can be further developed to optimize the first algorithm in the chain. It is not yet certain that this will work perfectly. Therefore, it has to be tested when it operates. If not, other algorithms could be given a chance, like the ones mentioned, i.e. Ant Colony Optimization, Grey Wolf Optimizer and Whale Optimization.

However, it is not yet certain whether changing the initial guess will solve all problems and will make sure that accurate values are produced by default. This should be further determined on the basis of new results obtained and analyzed.
The problem of parameter degeneracy may be solved by applying such different algorithms that search more precisely an initial guess. This will only be the case if it appears that there is one best value. When degeneracy still exists, it can be considered to pass on multiple initial estimates to the next algorithm. To succeed in doing this, the entire code must be adjusted accordingly and there will probably be a need for an extra algorithm in between the current steps. Since modifying an existing code causes many difficulties, it would probably be easier to set up completely new code to make it work.
When change in the initial algorithm is not successful, the other algorithms may also have room for improvement, but those will not be considered in this thesis.

5.1.3 User Friendly updates

Some updates are added to the code to make it more user friendly, focusing on areas where a possible complicated error could occur during use.
In more places printed warnings or clearly stated errors could be added, to ensure that it can be easily acted upon and also people who are not so deep into the code understand how to solve certain errors. This applies to errors mainly caused by something not being provided properly somewhere in the input and the code not understanding it.
An example of an error or warning that could be useful is when the boundaries are not properly declared. Then the value is on the edge of the range, by adjusting the range, a better value can come out.
Adding warnings or errors will not solve all problems, as some errors may occur that are more complex than properly entering a data file.
Another possibility of an addition could be to make the bandwidth part of the input. The bandwidth basically represents the error line frequencies can have, to still be matched with a line in a LAMDA file. Depending on the used instruments this may vary and the user probably has an indication of what the value could be. This is probably one of the main reasons that ReverseRADEX works properly for CO and not yet for other molecules. CO's spectral lines have a regular pattern and are widely spaced, so the bandwidth does not cause any difficulties when CO's data is applied. Conversely, other molecules can have spectral lines with different information in a smaller spectral range and the bandwidth should be adapted in such cases. Therefore, it can be useful to change this as an input parameter, rather than having to search the code to change its value.
Lastly, a graphical user interface could be created to make the programme more accessible.

5.2 Next steps

If the code works properly, that is to say the outcome is similar to what is entered in RADEX for different molecules than CO, there are several steps that can be taken thereafter.

5.2.1 Different uncertainties

Even if corner plots show a more distinct median when uncertainties are indicated, the values of the parameter estimates are not necessarily getting better. If this is fixed through adjustments, different variations of this can also be tested. Now no errors or errors of 10% are used. Then, testing can also be done with smaller or larger uncertainties to see how that affects the outcomes and the plots.

In addition, the default number of 1 for uncertainties, when none is provided, is not a realistic estimate of the uncertainties. This can be replaced by using a certain percentage as the error margin of the intensities to calculate an uncertainty that gives a more realistic impression. This can be replaced by using a certain percentage of the intensities as a margin for the error, to calculate an uncertainty that gives a more realistic impression.

5.2.2 Different background temperatures

In the scenario where a different wrapper is applied to the code it might be possible to use other background temperatures. Testing different background temperatures would be necessary first to see how this affects all the parameters. Then the code can be applied for circumstances with other background temperatures as well, allowing it to be used more broadly. For example, when the average interstellar radiation field has to be taken into account.

5.2.3 Real data

When all the previous steps have been fulfilled successfully, the testing of real data can begin. As a preview manually changes of RADEX data can be tested first, like carried out in this project too. I expect that real data is likely to bring up some new errors and adjustments may be needed to keep the code working. The bandwidth will probably also play a role here, because in real data other uncertainties will emerge than in the RADEX-produced lines. As suggested in section 5.1.3, the implementation of making bandwidth a variable to be entered by the user will already make it more accessible and make more of the data efficiently usable. Suggestions for real data sources are ALMA line surveys, ALMA provides a great database on their website with data about various molecules[4]. An example of a specific survey of ALMA is the Protostellar Interferometric Line Survey (PILS). The aim of this survey is to investigate the origin of complex organic molecules nearby star forming regions, in this case of the protostellar binary IRAS 16293-2422. The frequency range goes from 329 to 363 GHz (Jørgensen et al. 2016).

5.2.4 Different molecules

In this thesis the focus was on testing linear molecules. As they have simpler line spectra. In a follow up study this can be extended to nonlinear molecules. The LAMDA has supporting files for this, containing triatomic molecules and larger molecules, e.g. CH_3OCHO or PH_3.
In parallel, multiple linear molecules can also be added in testing, since only three of those have now been used.

[4] https://almascience.nrao.edu/alma-data

6 Conclusion

Spectral lines of molecules are crucial when studying molecular clouds and star- and planet-forming regions, as they provide valuable information about the composition, temperature, densities and other physical conditions of those regions. Relevant spectral lines occur mostly in mid and far infrared and (sub)millimeter regions of the electromagnetic spectrum.

Physical conditions are obtained by comparing observed spectra with theoretical calculations, adjusting the abundance of elements to maximize the correspondence between theoretical and observed lines. Existing tools such as RADEX are used to calculate physical conditions, but a more efficient method is needed. ReverseRADEX, developed in an earlier thesis, aims to estimate the kinetic temperature, the molecular column density and H_2 volume density from observed line intensities.

The aim of this thesis was testing ReverseRADEX for different molecules and increasing its usability. The research was carried out by testing linear molecules for various factors that affected the input of ReveresRADEX. Only computed data from RADEX was used.

The results of the tests were analyzed, based on the numerical and graphical output of ReverseRADEX.

The findings indicate variability in the accuracy of the results, which goes from the expected values to outcomes that are off by a factor of 10^3, depending on the parameters and conditions that are tested. From the results no clear trend can be observed. Hence, improvements have been suggested, including implementing different algorithms for initial estimates and adding warnings and error messages to improve the user experience.

Future steps include testing the programme for different molecules, different uncertainties, background temperatures and eventually using real observational data. Moreover, several modifications can be done to extend the programme to be used to nonlinear wider range of applications in astrophysics.

In conclusion, the code of ReverseRADEX is still too unreliable to use for real data. A lot of adjustments and testing need to be done before the outcome is of any real use.

Acknowledgements

The completion of my bachelor’s project marks the closure of a valuable learning journey. I would like to express my gratitude to my supervisor Prof Dr Floris F.S. van der Tak for his guidance throughout my research and his support during the more difficult stages of the project. In addition, I would like to thank Dr Tim Lichtenberg for serving as second examiner for this bachelor’s project and Drs Martin G.R. Vogelaar for having so much patience and time for what he called a "corona victim", with limited programming skills. Thank you to Filip van der Mooren for help and for being open to using his code.
Finally, I would like to thank my loving environment for the support and assistance I needed during this project, thanks in part to them I persisted and gained new perspectives.

Software: RADEX (van der Tak et al. 2007), ReverseRADEX (van der Mooren, 2021), oVirt(Omer Frenkal and Tomas Jelinek, 2013), SpectralRadex (Holdship et al. 2020), NumPy (Harris et al. 2020), SciPy (Virtanen et al. 2020), Pandas (McKinney 2010; Reback et al. 2020), emcee (ForemanMackey et al. 2013), Matplotlib (Hunter 2007), corner (Foreman-Mackey 2016), BeesAlgorithm (DT Pham 2006).

References

Baronti, L. et al. (2022) *BeesAlgorithm - A Python Implementation*, *PyPI*. Available at: https://pypi.org/project/bees-algorithm/ (Accessed: 19 November 2023).

Black, J.H. (1994) *Energy budgets of diffuse clouds, NASA/ADS*. Available at: https://ui.adsabs.harvard.edu/abs/1994ASPC...58..355B/abstract (Accessed: 17 November 2023).

Caputi, K. (2023) *Energy Levels and Spectral Lines, Kapteyn Instituut | Onderzoek | rijksuniversiteit Groningen*. Available at: https://www.astro.rug.nl/~karina/Teaching_files/kcaputi_ism202223_lect3.pdf (Accessed: 17 November 2023).

Castellani, M. (2021) *Baa 2021 tutorial,* YouTube. Available at: https://www.youtube.com/watch?v=vr_AgMM5wGg (Accessed: 17 November 2023).

Chaisson, Eric J. , Fernie, John Donald , Brecher, Kenneth and Aller, Lawrence Hugh. *"star"*. *Encyclopedia Britannica,* (6 Nov. 2023). Available at: https://www.britannica.com/science/star-astronomy. Accessed 17 November 2023.

Condon, J. and Ransom, S. (2018) *Chapter 7 Spectral Lines, 7 Spectral Lines‣ Essential Radio Astronomy*. Available at: https://www.cv.nrao.edu/~sransom/web/Ch7.html (Accessed: 17 November 2023).

Condie, K.C. (2022) *Molecular clouds, Molecular Clouds - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/earth-and-planetary-sciences/molecular-clouds (Accessed: 17 November 2023).

Decyk, Viktor & Norton, Charles & Gardner, Henry. (2007). *Why Fortran?. Computing in Science & Engineering*. 9. 68 - 71. 10.1109/MCSE.2007.89.

Delaire, J.A. and Nakatani, K. (2000) *Linear and nonlinear optical properties of ... - ACS publications, Linear and Nonlinear Optical Properties of Photochromic Molecules and Materials.* Available at: https://pubs.acs.org/doi/10.1021/cr980078m (Accessed: 17 November 2023).

Dere, K.P. and Mason, H.E. (1993) *Nonthermal velocities in the solar transition zone observed with the high-resolution telescope and Spectrograph - Solar Physics, SpringerLink.* Available at: https://link.springer.com/article/10.1007/BF00627590 (Accessed: 17 November 2023).

Draine, B.T. (2011) *Physics of the interstellar and Intergalactic Medium*. Princeton, NJ: Princeton University Press.

Dorigo, M. (2018) *Ant colony optimization, Ant Colony Optimization*. Available at: https://www.aco-metaheuristic.org/ (Accessed: 17 November 2023).

Erkut , O. and Hardalaç, F. (2021) *Comparison of Ant Colony Optimization and Artificial Bee Colony Algorithms for Solving Electronic Support Search Dwell Scheduling Problem.*

Available at: https://ieeexplore.ieee.org/document/9659666/ (Accessed: 17 November 2023).

Grupen, C. (2020) *Astroparticle Physics*. Cham: Springer.

Gupta, V.P. (2016) *Interaction of radiation and matter and electronic spectra, Principles and Applications of Quantum Chemistry.* Available at: https://www.sciencedirect.com/science/article/abs/pii/B9780128034781000091 (Accessed: 19 November 2023).

Herbst, E. and van Dishoeck, E.F. (2009) *Complex organic interstellar molecules - Leiden University*. Available at: https://home.strw.leidenuniv.nl/~ewine/e-prints/ARAA_published.pdf (Accessed: 30 November 2023).

Holdship, J. and the UCL Astronomy Group (2020). Version used: 0.3.2 (Mar 2021); original release/first (known) commit (Jul 2020). Url: https://github.com/uclchem/SpectralRadex.

Implementation of whale optimization algorithm (2021) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/implementation-of-whale-optimization-algorithm/ (Accessed: 17 November 2023).

Introduction to ant colony optimization (2020) GeeksforGeeks. Available at: https://www.geeksforgeeks.org/introduction-to-ant-colony-optimization/ (Accessed: 17 November 2023).

Jørgensen, J.K., Belloche, A. and Garrod , R.T. (2020) *Astrochemistry during the formation of stars , Annual reviews.* Available at: https://www.annualreviews.org/doi/full/10.1146/annurev-astro-032620-021927 (Accessed: 18 November 2023).

Jørgensen, J.K. et al. (2016) *The Alma Protostellar Interferometric Line Survey (PILS): First results from an unbiased submillimeter wavelength line survey of the class 0 protostellar binary IRAS 16293-2422 with alma*, arXiv.org. Available at: https://arxiv.org/abs/1607.08733 (Accessed: 20 November 2023).

Joseph, A. (2023). *Solar/planetary formation and evolution.* In Elsevier eBooks (pp. 1–54). https://doi.org/10.1016/b978-0-323-95717-5.00001-3

Karttunen, H. (2003) *Fundamental astronomy.* Berlin: Springer.

Koç, E. (2010) *The bees algorithm theory, improvements and applications*. Available at: https://orca.cardiff.ac.uk/55027/1/U585416.pdf (Accessed: 17 November 2023).

Kovetz, E.D. et al. (2019) *Astrophysics and cosmology with line-intensity mapping,* arXiv.org. Available at: https://arxiv.org/abs/1903.04496 (Accessed: 17 November 2023).

Langevelde , H.Ja. and Tak, F. (2008) *Radiation bookkeeping: A guide to astronomical molecular spectroscopy ...* Available at: https://var.sron.nl/radex/radex_manual.pdf (Accessed: 17 November 2023).

LeBlanc, F. (2010) *An introduction to Stellar Astrophysics*. Hoboken, N.J: Wiley.

Mathis, J. S. (2009, 20 mei). *Molecular Cloud | Astronomy, star formation & Interstellar Medium.* Encyclopedia Britannica. https://www.britannica.com/science/molecular-cloud

Megias, A. (2023) *Andresmegias/Radex-Python: Radex line fitter in python,* GitHub. Available at: https://github.com/andresmegias/radex-python (Accessed: 17 November 2023).

Mirjalili, S. et al. (2014) *Grey Wolf optimizer, Advances in Engineering Software.* Available at: https://www.sciencedirect.com/science/article/abs/pii/S0965997813001853 (Accessed: 17 November 2023).

Muders, D. (2010) *Home | Max-Planck-Institut für radioastronomie, Spectral line observing.* Available at: https://www.mpifr-bonn.mpg.de/948273/Muders-Spectral-Line-Observing.pdf (Accessed: 17 November 2023).

National Research Council, Division on Engineering and Physical Sciences, Board on Physics and Astronomy, Space Studies Board, Astronomy and Astrophysics Survey Committee (2001) *Astronomy and astrophysics in the new millennium: Panel reports, 7 Report of the Panel on Ultraviolet, Optical, and Infrared Astronomy from Space | Astronomy and Astrophysics in the New Millennium: Panel Reports | The National Academies Press*. Available at: https://nap.nationalacademies.org/read/9840/chapter/9#332 (Accessed: 18 November 2023).

Pagani, L. et al. (2020) *Radio Telescope Total Power Mode: Improving Observation Efficiency, Astronomy & Astrophysics.* Available at: https://www.aanda.org/articles/aa/full_html/2020/11/aa38976-20/aa38976-20.html (Accessed: 17 November 2023).

Pham, D.T. et al. (2020) *An analysis of the search mechanisms of the Bees algorithm, Swarm and Evolutionary Computation.* Available at: https://www.sciencedirect.com/science/article/abs/pii/S2210650220303990 (Accessed: 19 November 2023).

Pham, D.T. and Castellani, M. (2005) *The bees algorithm webpage , BAWebPage.* Available at: http://beesalgorithmsite.altervista.org/index.html (Accessed: 17 November 2023).

Snow, T.P. and McCall, B.J. (2006) *Diffuse atomic and molecular clouds , ResearchGate.* Available at: https://www.researchgate.net/publication/234147918_Diffuse_Atomic_and_Molecular_Clouds (Accessed: 19 November 2023).

Tak, F. van der et al. (2005) *Leiden atomic and molecular database, Leiden Observatory - Leiden University.* Available at: https://home.strw.leidenuniv.nl/~moldata/ (Accessed: 17 November 2023).

Van Der Tak, F. F. S., Black, J. H., Schöier, F. L., Jansen, D. J., & Van Dishoeck, E. F. (2007). *A computer program for fast non-LTE analysis of interstellar line spectra. Astronomy and Astrophysics*, 468(2), 627–635. https://doi.org/10.1051/0004-6361:20066820

Van der Tak, F. (2017) *Fvdtak/radex: Radex is a program for non-LTE models of Interstellar Line Spectra,* GitHub. Available at: https://github.com/fvdtak/RADEX (Accessed: 17

November 2023).

Van der Tak, F. et al. (2007) *Radex, Radex: Non-LTE molecular radiative transfer in homogeneous interstellar clouds*. Available at: https://personal.sron.nl/~vdtak/radex/index.shtml (Accessed: 17 November 2023).

Van der Tak, F. (2020) *The Leiden atomic and molecular database (LAMDA): Current status, recent updates, and future plans, MDPI.* Available at: https://www.mdpi.com/2218-2004/8/2/15 (Accessed: 17 November 2023).

Van der Mooren, F. (2021) *ReverseRADEX: A tool to quickly gauge global physical conditions of a gas cloud, Student Theses Faculty of Science and Engineering.* Available at: https://fse.studenttheses.ub.rug.nl/25088/ (Accessed: 17 November 2023).

Van der Mooren, F. (2021) *Fimomili/REVERSERADEX: ReverseRADEX is a tool to quickly gauge the physical conditions in a gas cloud from line spectra*., GitHub. Available at: https://github.com/FimomilI/ReverseRADEX (Accessed: 17 November 2023).

(2016) *7. Non-LTE – Basic Concepts - University of Hawai'i.* Available at: https://home.ifa.hawaii.edu/users/kud/teaching_16/7_Non_LTE.pdf (Accessed: 17 November 2023).

Möller, T. and Panoglou, D. (2020) *Magix manual.* Available at: https://magix.astro.uni-koeln.de/sites/magix/files/files/MAGIX_Manual.pdf (Accessed: 20 November 2023).

Mirjalili, S. et al. (2014) *Grey Wolf optimizer, Advances in Engineering Software*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0965997813001853 (Accessed: 17 November 2023).

Appendices

A Calling configuration files

```
#%% Read input from config.ini or ask for user input from terminal
parser = argparse.ArgumentParser(
    prog='Reverse RADEX',
    description='ReverseRADEX is a tool to quickly gauge the physical conditions in a gas cloud from line spectra.',
    epilog=''
)

parser.add_argument('-config', default=None, help='Specify the configuration file name (e.g., "your_config.ini")')
args = parser.parse_args()

if args.config is not None:
    config = ConfigParser()
    config_file = args.config
    config.read(config_file)

    paths     = 'PATHS'
    constants = 'CONSTANT_PARAMETERS'
    variables = 'VARIABLE_PARAMETERS'

    # file locations.
    user_molfile = eval(config[paths]['MolecularFile'])
    user_datfile = eval(config[paths]['SpectraFile'])
    print(user_molfile, user_datfile)

    # constant parameters.
    Tbg             = eval(config[constants]['BackgroundTemperature'])
    dv              = eval(config[constants]['LineWidth'])
    geom, geom_name = eval(config[constants]['Geometry'])

    # variable parameters.
    temp_kin = eval(config[variables]['Tkin'])
    coldens  = eval(config[variables]['Coldens'])
    voldens  = eval(config[variables]['Voldens'])
```

Where the config files gets called, this can be found in ReverseRADEX/main.py

```
else:  #### Catch user input from terminal ####
    user_molfile = input_constant.molfile_input()
    user_datfile = input_constant.datafile_input()

    # constant parameters.
    Tbg             = input_constant.background_radiation_input()
    dv              = input_constant.line_width_input()
    geom, geom_name = input_constant.geometry_input()

    # variable parameters.
    temp_kin = input_variable.kinetic_temperature_input()
    coldens  = input_variable.column_density_input()
    voldens  = input_variable.collision_densities_input()
```

The second part, when manually application of the input variables is preferred. This part has stayed the same.

B Different number of spectral lines

Estimates for HCl for four spectral lines, with expected values of 2, 14, 8

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 2.06544 | -0.41760 | +0.42078
log10(cdmol): 13.81756 | -2.72816 | +2.84872
log10(h2): 8.06363 | -2.14139 | +1.98694
 *** Warning: Some lines have very high optical depth
```

Estimates for CH^+ for 4, 5, 6 and 7 lines, with expected values of 2, 14, 5

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 1.73695 | -0.00965 | +0.00999
log10(cdmol): 15.69318 | -0.62997 | +0.67694
log10(h2): 3.39315 | -0.68303 | +0.63554
```

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 1.99987 | -0.01576 | +0.01534
log10(cdmol): 14.06104 | -0.90010 | +1.49218
log10(h2): 4.94573 | -1.46091 | +0.82453
```

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 1.99556 | -0.01444 | +0.01467
log10(cdmol): 14.44672 | -0.58524 | +1.38347
log10(h2): 4.57397 | -1.37012 | +0.55557
/net/virgo01/data/users/voorhoeve/RADEX/REVERSERADEX/save_plot/plot.py:
```

```
Parameter estimates and accompanying upper and lower uncertainties,
Percental:    50%    |    16%    |    84%    |
log10(tkin): 1.99736 | -0.01435 | +0.01561
log10(cdmol): 14.21054 | -0.90757 | +1.54870
log10(h2): 4.79095 | -1.51103 | +0.85621
```

C Implementation BeesAlgorithm

The following images show the attempt of the implementation of BeesAlgorithm to replace the first algorithm. This can be found in ReverseRADEX/fitting/find_initial_guess.py file, where the original code has been maintained as much as possible. Many different things have been tried to debug the code and to connect it to the main programme. Commented attempts and trials to do so are still in there.

```
# module imports
from numpy import (
    concatenate,
    geomspace,
    linspace,
    loadtxt,
    append,
    array,
    where,
    log10,
    full,
    ones,
    ix_
)
#import numpy as np
from spectralradex.radex import run_grid

from multiprocessing import cpu_count, Pool
import warnings
from bees_algorithm import BeesAlgorithm, BeesAlgorithmTester


def data_file_extraction(user_data_file, uncertainty):
    """extract the line strength column (with uncertainties) from the
    user supplied data file. These uncertainties are only used for
    calculating the chi^2 values so the default uncertainties = 1 (or
    any other constant) since they have no effect then.

    Args:
        user_data_file (str): user supplied data file directory.

        uncertainty (str): uncertainties included ('yes' -OR- 'no')


    Returns:
        tuple: 1 numpy array with line strenghts and 1 numpy array
        with line strength uncertainties.
    """
    data = loadtxt(user_data_file).T
    if uncertainty == 'no':
        line_strenghts = data[1]
```

```
        line_strenghts = data[1]
        return (line_strenghts, ones(line_strenghts.shape[0]))
    else:
        line_strenghts, line_strenght_uncertanties = data[1:]
        return (line_strenghts, line_strenght_uncertanties)

    return


# FIXME: use *args for y_err based on uncertainty?

def chi_squared(y_fit, y_obs, y_err, uncertainty):
    """calculate the chi^2 values between the user data file and the
    spectralRadex grid calculations (y_err as a default is equal to an
    array of ones).

    Args:
        y_fit (numpy array): the spectralRadex grid fit line strengths
        for all transition lines [T_R (K) -OR- FLUX (K*km/s) -OR-
        FLUX (erg/cm2/s)].

        y_obs (numpy array): the observed line strengths read from data
        file [T_R (K) -OR- FLUX (K*km/s) -OR- FLUX (erg/cm2/s)].

        y_err (numpy array): the observed line strength uncertainties
        read from data file [T_R (K) -OR- FLUX (K*km/s) -OR-
        FLUX (erg/cm2/s)].

        uncertainty (str): are uncertainties included ('yes', 'no').


    Returns:
        [numpy array]: chi_squared values
    """
    if uncertainty == 'no':
        return ( (y_obs - y_fit)**2 ).sum(axis=1)
    else:
        return ( ( (y_obs - y_fit) / y_err )**2 ).sum(axis=1)

    return
```

```
def find_initial_parameter_guesses(kinetic_temperature, column_density,
                                   voldens, volume_density,
                                   constant_parameters,
                                core_count=cpu_count()):
    """

    Args:
        summary = [name [str], value [float], (lim_low, lim_upp) [floats],
        fit (bool)]

        kinetic_temperature (list): summary of kinetic temperature.

        column_density (list): summary of column density.

        voldens (list): list of summaries of all collision partners.

        volume_density (list): list of summaries of all collision partners.

        constant_parameters (list): list of required constant parameters
        for spectralRadex and user data file information.


    Returns:
        list: list of lists of parameters that now contains the initial
        parameter guesses for the values to be written to "parameters.xml"
        for MAGIX.
    """
    _, Tkin_value, Tkin_limits, Tkin_fit = kinetic_temperature
    _, cd_value, cd_limits, cd_fit     = column_density
    (user_molfile, Tbg, dv, freq_min, freq_max, geom,
     units, matching_index, user_datfile,
     uncertainties) = constant_parameters

    # Create an instance of the BeesAlgorithm
    # Adjust the parameters and configurations as needed for your specific problem
###   ba = BeesAlgorithm(num_bees=50, num_elite_bees=10, num_iterations=100)

    # Define the objective function for the BeesAlgorithm
    def objective_function(parameters):
        # Construct parameter values based on the BA population
        current_tkin, current_cd, *current_voldens = parameters
        current_tkin = 10 ** current_tkin
```

```
        current_tkin = 10 ** current_tkin
        current_cd = 10 ** current_cd
        current_parameters = [current_tkin, current_cd] + [10 ** vd for vd in current_voldens]

        # Construct grid guess parameters
#         parameters_to_fit = [Tkin_fit, cd_fit]
#         number_of_parameters_to_fit = 0
#         for fit in parameters_to_fit:
#             if fit == True:
#                 number_of_parameters_to_fit += 1
#         grid_guess_parameters = {}
#         for collision_partner in voldens:
#         # exclude the volume density bounds
#             if collision_partner != 'min_max':
#                 value, fit = voldens[collision_partner]
#                 if fit is False:
#                     grid_guess_parameters[collision_partner] = value
#                 else:
#                     number_of_parameters_to_fit += 1
#                     grid_guess_parameters[collision_partner] = geomspace(
#                         voldens_min, voldens_max, num_points_voldens + 1,
#                         endpoint=False
#                 )[1:]

#         grid_guess_parameters['tkin']      = Tkin_grid
#         grid_guess_parameters['cdmol']     = cd_grid
#         grid_guess_parameters['molfile']   = user_molfile
#         grid_guess_parameters['tbg']       = Tbg
#         grid_guess_parameters['linewidth'] = dv
        # grid_guess_parameters['fmin']      = freq_min
        # grid_guess_parameters['fmax']      = freq_max
        # grid_guess_parameters['geometry']  = geom
        grid_guess_parameters = {
            'tkin': array([ 34.8,  64.6,  94.4, 124.2, 154. , 183.8, 213.6, 243.4, 273.2, 303. , 332.8, 362.6, 392.4, 422.2, 452. , 481.8,
511.6, 541.4,571.2, 601. , 630.8, 660.6, 690.4, 720.2]),  #current_parameters[0]),
            'cdmol': array([6.81292069e+14, 4.64158883e+16, 3.16227766e+18, 2.15443469e+20,1.46779927e+22]),  #current_parameters[1]),
            'molfile': user_molfile,
            'tbg': Tbg,
            'linewidth': dv,
            'fmin': freq_min,
            'fmax': freq_max,
            'geometry': geom
```

```
            'geometry': geom
        }

        for collision_partner, param_value, _ in zip(volume_density, current_parameters[2:], voldens):
            if collision_partner[2]:  # Check if this partner is fitted
                grid_guess_parameters[collision_partner[0]] = 10 ** param_value
        print('guess:', grid_guess_parameters)
        core_count = 1
        pool = Pool(processes=core_count)
        # Run the grid for this set of parameters
    #   guess = {'a':3, 'b':4}
        for key, value in grid_guess_parameters.items():
            #print('key, vlaue', key, value)
            if key== 'h2':
                newvalue = array(1e4)
                grid_guess_parameters[key] = newvalue
        #grid_guess_parameters['molfile'] = "/Users/users/voorhoeve/VIRGO01/RADEX/data/co.dat"

        #print('array guesses?', grid_guess_parameters)
        print("guess;", grid_guess_parameters)
        grid_output_DataFrame = run_grid(grid_guess_parameters, target_value=units, pool=pool)
        print('==================================================')
        print(' dataframe',grid_output_DataFrame)
        grid_output = grid_output_DataFrame.to_numpy()

        # Calculate chi-squared and return it
        y_observed, y_uncertainties = data_file_extraction(user_datfile,
                                                           uncertainties)
        parameters_to_fit = [Tkin_fit, cd_fit]
        number_of_parameters_to_fit = 0
        for fit in parameters_to_fit:
            if fit == True:
                number_of_parameters_to_fit += 1

        grid_output_cut = grid_output[:,number_of_parameters_to_fit:]
        grid_output_to_compare = grid_output_cut[ix_(full(grid_output_cut.shape[0], True), matching_index)]
        chi2 = chi_squared(grid_output_to_compare[:, number_of_parameters_to_fit:], y_observed[None,:], y_uncertainties[None,:],
uncertainties)
        return chi2.sum()
```

```
#     lower_bounds = array([log10(Tkin_limits[0]), log10(cd_limits[0])] + [log10(vd[0]) for vd in voldens])
#     upper_bounds = array([log10(Tkin_limits[1]), log10(cd_limits[1])] + [log10(vd[1]) for vd in voldens])
  #  print(array(Tkin_limits[0]),type(array(Tkin_limits[0])), log10(array(Tkin_limits[0])))
  #  print(array(cd_limits[0]),type(array(cd_limits[0])), log10(array(cd_limits[0])))
   # print(voldens,type(voldens))
   # print([voldens[vd][0] for vd in voldens]) #List comprehension, begint rechts links wordt uitgevoerd
    newlist = []
    for k in voldens:
        value = voldens[k][0]
        if value == 0:
            newlist.append(0.0000001)
        else:
            newlist.append(log10(value))
   # print(newlist)
    lower_bounds = array([log10(array(Tkin_limits[0])), log10(array(cd_limits[0]))] + [0]*len(newlist)) # + [log10(array(voldens[vd][0])) for vd in voldens])
    upper_bounds = array([log10(array(Tkin_limits[1])), log10(array(cd_limits[1]))] + newlist) #+ [log10(array(voldens[vd][0])) for vd in voldens])
    print(lower_bounds)
    print(upper_bounds)
#     Tkin_min, Tkin_max       = Tkin_limits
#     cd_min, cd_max           = cd_limits
#     voldens_min, voldens_max = voldens['min_max']
    #ook nog voor de andere parameteres? of alles in 1?
    print('++++++++++++++++++++++++++++++++++++++++++++++')
    print(objective_function)
    ba = BeesAlgorithm(objective_function, lower_bounds, upper_bounds, ns=50, nb=10, ne=5, nrb=5, nre=10, stlim=10)

    #ba = BeesAlgorithm(objective_function,lower_bounds[0],upper_bounds[0]) #num_bees=50, num_elite_bees=10, num_iterations=100)
  #  ba.set_bounds(lower_bounds, upper_bounds)

    # Run the Bee Algorithm to find initial parameter guesses
    best_parameters, _ = ba.run(objective_function)
    print('best_parameters', best_parameters)
    # Convert the best parameters back to their original scale
    best_tkin, best_cd, *best_voldens = best_parameters
    best_tkin = 10 ** best_tkin
    best_cd = 10 ** best_cd
    best_parameter_estimates = [best_tkin, best_cd] + [10 ** vd for vd in best_voldens]

    return best_parameter_estimates

```