RAA 2019 Vol. **19** No. **12**, 176(10pp) doi: 10.1088/1674–4527/19/12/176
© 2019 National Astronomical Observatories, CAS and IOP Publishing Ltd.
http://www.raa-journal.org *http://iopscience.iop.org/raa*

***R**esearch in*
***A**stronomy and*
***A**strophysics*

A new data structure for accelerating kinetic Monte Carlo method

Xu-Li Zheng[1,4], Dong-Hui Quan[1,2], Hai-Long Zhang[1], Xiao-Hu Li[1], Qiang Chang[1] and Olli Sipilä[3]

[1] Xinjiang Astronomical Observatory, Chinese Academy of Sciences, Urumqi 830011, China; *765947958@qq.com*
[2] Department of Chemistry, Eastern Kentucky University, Richmond, KY, USA
[3] Max-Planck-Institute for Extraterrestrial Physics (MPE), Giessenbachstr. 1, 85748 Garching, Germany
[4] School of Astronomy and Space Science, University of Chinese Academy of Sciences, Beijing 100049, China

Received 2019 April 14; accepted 2019 June 10

Abstract The kinetic Monte Carlo simulation is a rigorous numerical approach to study the chemistry on dust grains in cold dense interstellar clouds. By tracking every single reaction in chemical networks step by step, this approach produces more precise results than other approaches but takes too much computing time. Here we present a method of a new data structure, which is applicable to any physical conditions and chemical networks, to save computing time for the Monte Carlo algorithm. Using the improved structure, the calculating time is reduced by 80 percent compared with the linear structure when applied to the osu-2008 chemical network at 10 K. We investigate the effect of the encounter desorption in cold cores using the kinetic Monte Carlo model with an accelerating data structure. We found that the encounter desorption remarkably decreases the abundance of grain-surface H_2 but slightly influences the abundances of other species on the grain.

Key words: astrochemistry — molecular processes — methods: numerical — ISM: molecules — ISM: abundances

1 INTRODUCTION

Good models can play a central role in astrochemistry while creating, judging, or predicting observations. Generally, simple models take less computing time and give less accuracy, while advanced models might significantly improve the modeling level but take a considerable amount of time. Modelers spare no effort to improve the modeling accuracy within reasonable computing time. For the case of gas-grain chemistry, there exist at least five kinds of models, which are the rate-equation (RE) model, the modified rate-equation (mRE) model, the master-equation (MaE) model, the moment-equation (MoE) model and the Monte Carlo model. Details of these models are beyond the scope of this article and interested readers please refer to Du & Parise (2011). Relatively, the RE approach (Hasegawa et al. 1992), which is based on chemical kinetics, is rapid in calculating the abundances of chemical species (both in the gas phase and on the grain surface). However, the simulative results of this approach differ considerably from those given by the more rigorous stochastic approaches while modeling the chemistry of the grain species with low abundances (much less than one atom/molecule per grain). This discrepancy is known as the finite-size (FS) problem (Hasegawa & Herbst 1993; Taquet et al. 2012). The mRE approach, which partly solved the FS problem, is the most efficient and applicable to large chemical networks (Chang & Herbst 2012). However, the mRE approach is the least rigorous. The MaE is more rigorous than the mRE approach. This approach calculates the temporal evolution of the probability of states in the phase space. However, it has not been used in a large gas-grain network because as the number of species in the reaction network increases, the number of states increases exponentially (Stantcheva & Herbst 2004). The MoE approach, which combines the advantages of the Monte Carlo approach and the mRE approach to some extent, is more rigorous and rather efficient at low order (Du & Parise 2011). The kinetic Monte Carlo approach is more rigorous than the MoE approach (Chang & Herbst 2012; Lipshtat & Biham 2003). They can be utilized to calculate the precise evolution of any species and yield more accurate results for low-abundance species on the grain (Tielens & Charnley 1997; Herbst & Shematovich 2003; Stantcheva & Herbst 2004). However, it takes a lot of computing time when the simulative time is long, or the number of reactions is large, or the temperature is high (Hincelin et al. 2013; Viti et al. 2004; Garrod & Herbst 2006; Gao et al. 2017). Hence, solutions for reducing computing time in the Monte Carlo models are necessary. New mechanisms have

been applied to specific physical and chemical conditions to save time (Chang et al. 2017). Methods from other areas such as computer science are also needed to improve the efficiency of the algorithm.

The encounter desorption mechanism describes how a grain-surface H_2 molecule (hereafter gH_2) desorbs from the grain-surface gH_2 monolayer rather than from the ice: $gH_2 + gH_2 \rightarrow H_2 + gH_2$ (Hincelin et al. 2015). This encounter desorption has a rather high reaction rate and would influence the abundance of gH_2 and other grain-surface species in our prediction. The physical condition of the encounter desorption mechanism is at a rather low temperature and the abundances of most species on the grain are below one unit, which encounters the FS problem in rate-equation simulation. The kinetic Monte Carlo model, which yields better results for this condition, is going to be utilized to investigate the effect of the mechanism. When taking account of the encounter desorption, we have to consider the problem that the high rates of accretion of H_2 and desorption of gH_2 are too time-consuming to apply to the kinetic Monte Carlo model in a full gas-grain network such as osu-2008 (Hassel et al. 2008). Therefore, the kinetic Monte Carlo model has not been utilized to investigate the effect of the encounter desorption, making in depth understanding difficult. Chang et al. (2017) proposed to apply the quasi-steady-state approximation to the kinetic Monte Carlo model to solve the computationally time-consuming problem caused by the high rates of accretion of H_2 and the desorption of gH_2. However, the model with the quasi-steady-state approximation is still time-consuming. Therefore, we use a new data structure to further save computing time to fulfil the aim of investigating the effect of the encounter desorption in the kinetic Monte Carlo model.

This paper is organized as follows. The encounter desorption and quasi-steady-state approximation are presented in Section 2. Section 3 describes the kinetic Monte Carlo model and Section 4 describes a new pyramid-searching data structure for accelerating the kinetic Monte Carlo approach. The results are included in Section 5 and Section 6 closes with conclusions.

2 ENCOUNTER DESORPTION AND THE QUASI-STEADY-STATE APPROXIMATION

We usually think that the desorption of gH_2 is on the water ice substrate and utilizes the desorption energy of H_2 due to its value on water ice, 440 K (Cazaux & Tielens 2004). However, the abundance of gH_2 is also high on the grain mantle at low temperatures (Hasegawa et al. 1992; Watson & Salpeter 1972; Pickles & Williams 1977; Tielens & Allamandola 1987), and Morata & Hasegawa (2013) suggested that the amount of gH_2 on the grain is about one monolayer at 10 K. The possibility that one gH_2 molecule comes across another gH_2 molecule is high, and one of them subsequently desorbs from the gH_2 molecules. This process is called encounter desorption. The desorption energy of gH_2 on the gH_2 substrate is 23 K (Cuppen & Herbst 2007; Garrod & Pauly 2011). The encounter desorption is much more efficient than the desorption directly from the water ice, since the desorption energy of gH_2 on the gH_2 substrate is about 20 times smaller than that on the water substrate (Hincelin et al. 2015).

In this paper, we will study how encounter desorption influences the species on the grain. Hincelin et al. (2015) already tested the encounter desorption in a rate-equation model with a large gas-grain chemical network using the Nautilus code. However, when most of the abundances of grain-surface species are below one unit, the Monte Carlo model is more precise. Since the microscopic Monte Carlo is too time-consuming (Chang et al. 2005), we employ the kinetic Monte Carlo model. There are still two remaining problems: first, the high rates of accretion of H_2 and desorption of gH_2 will make the simulation take too much computing time. Therefore, an alternative solution is necessary. Second, a highly efficient method to improve the kinetic Monte Carlo model in a full gas-grain network is needed to further decrease the computing time.

The first problem is solved by Chang et al. (2017) using the quasi-steady-state approximation (Gillespie 1976). The quasi-steady-state approximation is an assumption that the abundance of gH_2 is a constant. Because the accretion rate of H_2 is large due to the large abundance of H_2, the computing time required to simulate a kinetic Monte Carlo model that includes gas-phase H_2 accretion is overwhelmingly large. In Section 3, the relation of the reaction rates with the computing time will be given in detail. Therefore, the accretion of H_2 is normally absent in most kinetic Monte Carlo models. In the quasi-steady-state approximation, species whose accretion and desorption rates are much more than the loss rates of these species via reactions on grain surfaces are treated as transient species. The abundances of transient species on grain surfaces are purely determined by the accretion and desorption of these species themselves. The gH_2 can be taken as a transient species because H_2 is overwhelmingly much more abundant than other species, causing a rather high accretion rate of H_2 and desorption rate of gH_2. Moreover, because the abundance of H_2 can hardly change if the physical conditions are kept constant, the abundance of gH_2 can also be assumed to be constant. Therefore, the calculation of the changing molecular number of gH_2 is neglected to save computing time.

The second problem can be solved by introducing a new data structure to improve the algorithm. We will discuss the highly efficient pyramid-searching data structure in Section 4. The method is widely applied to decrease the computing time despite the types of reactions and physical conditions in the simulation.

3 KINETIC MONTE CARLO MODEL

The kinetic Monte Carlo model is implemented to simulate the reactions, one at a time, both in the gas and on the grain. In the process, we use two random numbers to decide how long the reaction will take and which reaction will occur. We have two steps:

1. Find the reaction time interval
   We use the following formula to calculate the reaction time for the occurring reaction
   $$t = \frac{-\log(r_2)}{\sum_{i=1}^{\max} R_i}\,. \tag{1}$$
   r_2 is a random number, which is equally distributed between 0 and 1. R_i represents the reaction rate of reaction i.
2. Find which reaction will occur
   We generate a random number r_1, which is equally distributed between 0 and 1, and use it to compare the fraction of the sum of selected reaction rates over the total rates ($\frac{\sum_{i=1}^{k} R_i}{\sum_{i=1}^{\max} R_i}$). As expressed in the following equation, when $\sum_{i=1}^{k} R_i$ is just above $r_1 \times \sum_{i=1}^{\max} R_i$ while the previous $\sum_{i=1}^{k-1} R_i$ is below $r_1 \times \sum_{i=1}^{\max} R_i$, we assume that the reaction k will occur.
   $$\sum_{i=1}^{k-1} R_i < r_1 \times \sum_{i=1}^{\max} R_i < \sum_{i=1}^{k} R_i\,. \tag{2}$$

The kinetic Monte Carlo model is a loop of step 1 and 2, as shown above. Each loop describes how a reaction will occur and how long it will occur. The reactions occur one by one. As the two steps accumulate for a long simulative time, the results follow a normal distribution. For the case of a large reaction rate like the accretion of H_2, making a large $\sum_{i=1}^{\max} R_i$ and a small time interval t, it takes more loops to accumulate to a specific simulative time than the case that $\sum_{i=1}^{\max} R_i$ is small. More loops need more computing time, which is the reason for taking into account the accretion of gH_2 and the desorption of H_2 needs an excessive amount of time.

In detail, when a reaction occurs, the number of reactants will decrease, and the number of products will increase. During this process, the rates of the reactions related to the changed species will also be changed. When the reaction is completed and the program goes on to search for the next reaction to occur in the next loop, the updated rates should be used to calculate the sum of the new reaction rates. The part of updating the corresponding changed reaction rates is called the $updating\ part$.

The algorithm consists of the $searching\ part$, which finds the reaction that occurs and the time this reaction lasts, and the $updating\ part$, in which the related reaction rates are updated.

In the searching part, the normal method to compute the $\sum_{i=1}^{k-1} R_i$, $\sum_{i=1}^{k} R_i$ and $\sum_{i=1}^{\max} R_i$ is to add the rates from the first to the last. But when the number of reactions is very large, the required computing time is long. For example, if the number of reactions in a large network is over five thousand, it takes much more time for the program to compute the sum of the reaction rates than the time in a small network with only nine reactions. Therefore, we look into the algorithm to find new time-saving methods. We find that each time when the updated $\sum_{i=1}^{k-1} R_i$, $\sum_{i=1}^{k} R_i$ and $\sum_{i=1}^{\max} R_i$ are computed, most of the reaction rates stay the same and only some reaction rates are changed. In this situation, we introduce a pyramid-index-searching structure to the kinetic Monte Carlo model to skip the reactions with unchanged rates to calculate the $\sum_{i=1}^{k} R_i$ and to lead the $\sum_{i=1}^{k} R_i$ to assist in searching for the occurring reaction in $searching\ part$ in Section 4.

4 THE PYRAMID-INDEX-SEARCHING METHOD

4.1 Data Structure

Data structure is a concept of how the data are processed and stored. A logical data structure is based on how the data are processed and a physical data structure is based on how the data are stored according to the logical structure. When the data are read, the specific algorithm employs the logical data structure. An appropriate data structure can improve efficiency of the algorithm (Baloukas et al. 2009; Tarjan 1983, 1984). The logical data structure is determined by the relationship between the data. There are two types of logical data structures: a linear structure and a nonlinear one.

The physical data structure that is used to store the reactions in the old model is an array, which is rapid to read. But when one reaction rate is changed, all the reaction rates have to be read from the first to the $k^{\rm th}$ to calculate $\sum_{i=1}^{k} R_i$. To skip the unchanged rates, we adopt a new pyramid-searching data structure to store all the reactions. The new data structure is made of data structs (a concept in data structure, which are employed to store several items associated with each other in a unit) that are organized like a pyramid. Each data struct includes the following information: reaction number, reaction rate, $\sum_{i=1}^{k} R_i$ (sum of the rates up to the reaction, 3rd lower captain means the third captain that is led by this captain), upper level struct, three lower level structs, $\sum_{i=1}^{k}(R_{\rm inew} - R_{\rm iold})$ (the sum of the rate differences), in the formula $R_{\rm inew}$ represents the newly updated reaction rate, the $R_{\rm iold}$ represents the old reaction rate in the last loop, $\sum_{i=1}^{k} R_{i0}$ (a constant, the sum of the reaction rates in the first loop), etc. By adding $\sum_{i=1}^{k} R_{i0}$ to $\sum_{i=1}^{k}(R_{\rm inew} - R_{\rm iold})$, which means the initial $\sum_{i=1}^{k} R_i$ increased by the changes (sum of the rate differ-

Table 1 Elements of the Struct

No	Elements	Notes	New	Old
1	reaction no.	for explaining the reaction	√	√
2	reactants	for explaining the reaction	√	√
3	products	for explaining the reaction	√	√
4	parameters	for explaining the reaction	√	√
5	reaction type	for explaining the reaction	√	√
6	reaction checked times	for explaining the reaction	√	√
7	reaction rate coefficient	for explaining the reaction	√	√
8	reaction rate	for explaining the reaction	√	√
9	upper captain's no.	for building the structure	√	×
10	1st lower captain's no.	for building the structure	√	×
11	2nd lower captain's no.	for building the structure	√	×
12	3rd lower captain's no.	for building the structure	√	×
13	1st lower captain's $\sum_{i=1}^{k} R_i$	for building the structure	√	×
14	2nd lower captain's $\sum_{i=1}^{k} R_i$	for building the structure	√	×
15	3rd lower captain's $\sum_{i=1}^{k} R_i$	for building the structure	√	√
16	$\sum_{i=1}^{k} R_{i0}$	for building the structure	√	×
17	$\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$	for building the structure	√	×

ences) during the loops, we can get the correct $\sum_{i=1}^{k} R_i$ at any loop. Table 1 lists all the elements of the data struct.

In the pyramid structure (as displayed in Fig. 1(a) and Fig. 2(a)), at the bottom all of the reactions are considered as reaction "soldiers". Since these soldiers do not have lower structs, the corresponding values are all defined to be zeros. The reaction soldiers are grouped by a certain number of soldiers and the number is defined according to the size of the reaction network. We use a relatively big number, e.g. 25 (meaning that there are 25 members in a group), for big reaction networks, for example the osu-2008, and a small number, e.g. 1 for very small networks. As described in the previous part, when a reaction occurs, it may affect the reaction rates of other reactions. We apply a value, which is the difference of the newly updated rate minus the old rate in the last loop ($R_{\mathrm{inew}} - R_{\mathrm{iold}}$), to record the change in the rate, and we use the sum of the rate differences ($\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$) to record the changes from the first loop to this loop. Therefore, for each reaction soldier, one element is reserved for recording the sum of the rate differences as the reaction rates are progressed. This element is represented by $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$. For the last soldier in a group, its $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$ records the sum of the rate differences of the soldiers from the first reaction soldier in the group to itself while other soldiers only record their own $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$. This is due to the last soldier in the group leading the $\sum_{i=1}^{k} R_i$ ($\sum_{i=1}^{k} R_i$ = $\sum_{i=1}^{k} R_{i0}$ + $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$) of the group. The $\sum_{i=1}^{k} R_i$ assists the searching by reading $\sum_{i=1}^{k} R_i$ directly rather than by calculating R_i to get the sum.

In the level that is just above the bottom, "captains" (named sol-captains, i.e. soldiers' captains) are assigned to lead the reaction soldiers so that each sol-captain leads three groups of soldiers. The captains are virtual reactions that do not have real reaction rates. The virtual reaction includes null reactants and products, which are defined as zeros in the program. The rates are also zeros but the sum of the rates stores the corresponding values ($\sum_{i=1}^{k} R_i$) from the three last soldiers in the three lower groups under this captain. According to the definition of $\sum_{i=1}^{k} R_i$, the last soldier's $\sum_{i=1}^{k} R_i$ is the maximum within its group. Also, the last soldier from the third group under this captain has the maximum $\sum_{i=1}^{k} R_i$. The $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$ of this captain is defined as the sum of the rate differences from the first reaction soldier to the last reaction soldier in its third lower group. Meanwhile, each captain has the $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$ of all reactions up to its last soldier like $\sum_{i=1}^{k} R_i$. The three lower level structs of the sol-captains are pointed to the structs of the last soldiers in the three groups.

Above the sol-captains' level, there are captains of captains (named cap-captains, i.e. captains' captains), which follow a similar structure but they only lead three lower captains (not groups of reaction soldiers). All elements follow the same rules as those for sol-captains except the $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$, which is read directly from the value of the 3rd captain under it. In this way, each captain stores $\sum_{i=1}^{k} R_i$ of all reactions up to its last leading reaction in the bottom and this is also true for $\sum_{i=1}^{k}(R_{\mathrm{inew}} - R_{\mathrm{iold}})$. This is repeated until there is only one captain needed. Thus this captain is named "chief captain" and is on the top of the pyramid structure. Virtual soldiers with null reactants, products and $\sum_{i=1}^{k} R_i$ are added to the bottom as needed to complete a full pyramid.

4.2 Searching for the Reaction that Occurs

To illustrate how we skip the unchanged reaction rates to calculate $\sum_{i=1}^{k} R_i$, we use a small network containing nine reactions. The data structure from the old model as well as the accelerated model is shown in Figure 1(b) and Figure 1(a), respectively.

In the linear structure, we add the reaction rates from the first to the specific reaction to get the $\sum_{i=1}^{k} R_i$ satisfying Equation (1) and the fifth reaction is the occurring

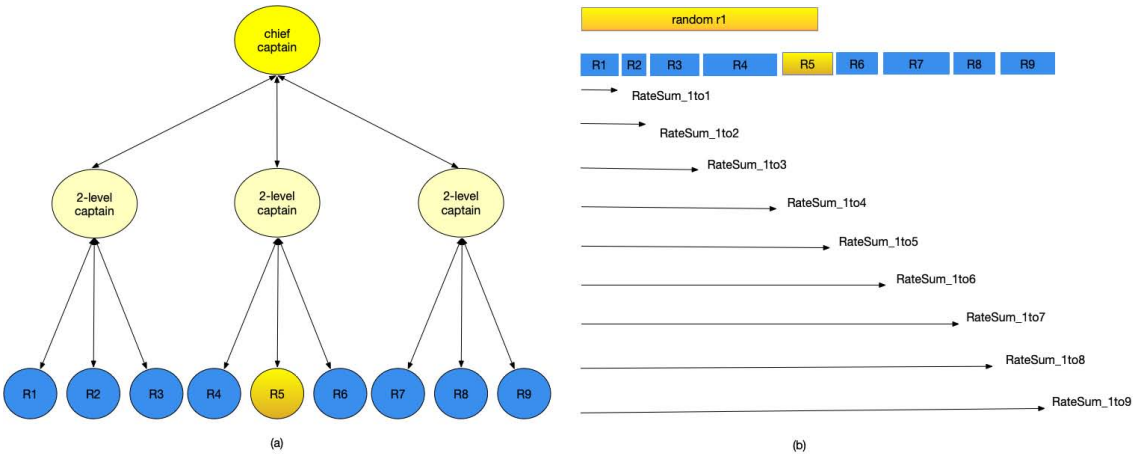

Fig. 1 (a) Data structure from the accelerated model using the pyramid-index-searching structure. The chief captain leads three $2^{\rm nd}$ captains, and the $2^{\rm nd}$ captains lead nine reaction soldiers. (b) Data structure from the old model. In the example here, r_1 is 0.5, and the arrows with long tails represent $\sum_{i=1}^{k} R_i . \sum_{i=1}^{5} R_i$ satisfies Eq. (1).

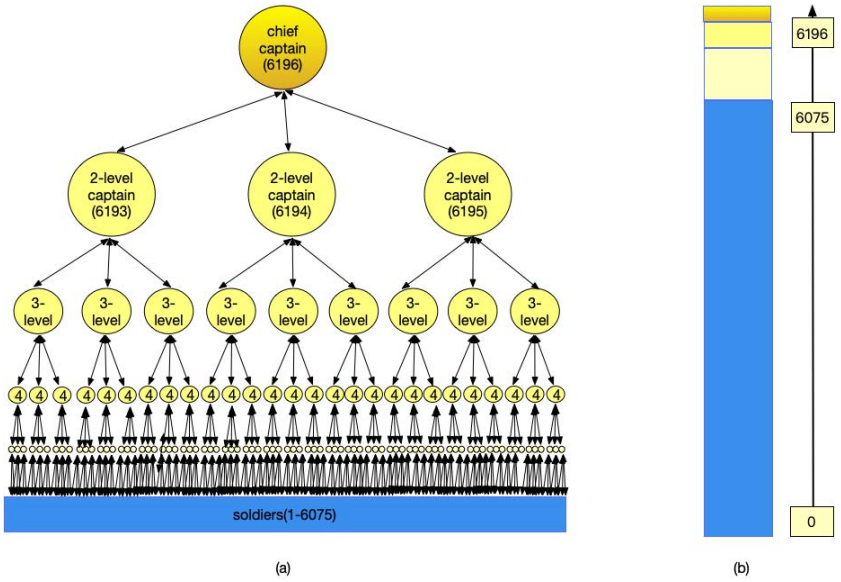

Fig. 2 (a) Pyramid data structure for 5942 reactions. The number of reaction soldiers on the bottom (the $6^{\rm th}$ level) is 6075, in which there are 5942 real reactions and 133 virtual reactions to make a complete pyramid. They are divided into groups of 25 members. The number of captains in the $5^{\rm th}$ level is 81, ranging from 6076 to 6156. The number of captains in the $4^{\rm th}$ level is 27, ranging from 6157 to 6183. The number of captains in the $3^{\rm rd}$ level is 9, ranging from 6184 to 6192. The number of captains in the $2^{\rm nd}$ level is 3, ranging from 6193 to 6195. The number of the chief captain is 6196. (b) The array to store the pyramid data structure. The blue rectangle represents the soldiers, and the yellow rectangles signify the captains. The structs are built according to the sequence shown by the arrow on the right of the array.

reaction in this example. In computer science, the average searching length (the time complexity of the searching part, and the time complexity is a concept to evaluate the computing time in the program) is a measure of the consumed time for a searching method. It is usually assumed that each operation which takes a unit of time is defined as 1. Here the searching length of the old model in the general cases is $\frac{1}{9}\times$(1+2+...+9)$\times$2=10. Here, the probability of choosing one reaction in the network of nine reactions. 1, 2,..., 9 represent the comparing times of specific reactions in different positions in the network. Multiplying by 2 denotes the time taken by two operations, adding and comparing, for each round. Consequently, the searching length in the old model is 10.

Using the pyramid-index-searching structure, the data struct (a captain or a soldier) that satisfies $r_1 \times \sum_{i=1}^{\max} R_i < \sum_{i=1}^{k} R_i$ will be found for each level, and the interval in which the reaction that occurs lies is zoomed out by each level of searching. This means finding the first $\sum_{i=1}^{k} R_i$ larger than $r_1 \times \sum_{i=1}^{\max} R_i$ through the captains in the level.

The searching process begins from the first level, chief captain, whose three $\sum_{i=1}^{k} R_i$ are $\sum_{i=1}^{3} R_i$, $\sum_{i=1}^{6} R_i$ and $\sum_{i=1}^{9} R_i$. Then the 2nd lower $\sum_{i=1}^{k} R_i$, which is $\sum_{1}^{6} \mathrm{Rate}$ for this case, is compared with r_1 (assuming that $\sum_{i=1}^{\max} R_i$ is 1). Following the same example as we discussed in the old model, the $\sum_{i=1}^{6} R_i$ is bigger than the r_1. Then the 1st lower $\sum_{i=1}^{k} R_i$, which is $\sum_{i=1}^{3} R_i$, is compared with r_1 to find out if $\sum_{i=1}^{6} R_i$ is the first one larger than r_1. Here $\sum_{i=1}^{3} R_i$ is smaller than r_1. Therefore $\sum_{i=1}^{6} R_i$ is the first $\sum_{i=1}^{k} R_i$ that is larger than r_1 in this level. So, $\sum_{i=1}^{k} R_i$ of the occurring reaction is among $\sum_{i=1}^{4} R_i$, $\sum_{i=1}^{5} R_i$ and $\sum_{i=1}^{6} R_i$.

We then move forward to the next lower level. Using the same example, only the 2nd captain in this level needs to be checked, because the occurring reaction must be under this captain. As defined in the previous parts, this lower level captain also stores three $\sum_{i=1}^{k} R_i$, which are $\sum_{i=1}^{4} R_i$, $\sum_{i=1}^{5} R_i$ and $\sum_{i=1}^{6} R_i$. The one in the middle, $\sum_{i=1}^{5} R_i$, is first compared with r_1. In this example, it is larger than r_1. Following the same method as we had in the upper level, next the $\sum_{i=1}^{k} R_i$ on the left is compared with r_1. Here $\sum_{i=1}^{4} R_i$ is smaller than r_1. Consequently, we conclude that reaction 5 is the reaction that occurs. The average searching length for using the improved method in general cases is $\frac{1}{9}\times$(4+4+3+4+4+3+3+3+2) = $\frac{10}{3}$. Here, the $\frac{1}{9}$ represents the probability of choosing one reaction among nine reactions. 4, 3, 2 represent the comparing times of specific reactions in different positions in the network. In each level, there are at most two comparing operations: on one hand, if the 2nd lower $\sum_{i=1}^{k} R_i$ is larger than r_1, the 1st lower $\sum_{i=1}^{k} R_i$ needs to be compared with r_1; On the other hand, if the 2nd lower $\sum_{i=1}^{k} R_i$ is smaller than r_1, the 3rd lower $\sum_{i=1}^{k} R_i$ is already the first value that is larger than r_1. Therefore, only one comparison is necessary. The number of operations to identify the occurring reaction is then calculated. For example, if reaction (1) is the real case, then four operations are needed; if reaction (2), then four operations are needed, etc. Comparing the average searching length using the two structures, it can be seen that the pyramid-index-searching method saves $\frac{2}{3}$ of the searching time in this case.

4.3 Application to a Large Chemical Reaction Network

In this study, we use the osu-2008 reaction network for interstellar clouds. The network includes various types of reactions: bimolecular gas phase reactions, cosmic ray (CR) ionization and dissociation, far ultraviolet (FUV) photodissociation and ionization, thermal desorption and CR desorption, and bimolecular grain surface reactions (Semenov et al. 2010). The total number of reactions is 5942. In the following part, we analyze how the pyramid structure will improve the model efficiently using the large network. The reaction soldiers are grouped by 25 members in this case. The structure for 5942 reactions is illustrated in Figure 2(a). As seen from Figure 2(a), we know the pyramid structure consists of five levels.

For this network, the average searching length of the model with pyramid structure is $\frac{1}{243}\times$(10+10+9+10+10+9+9+9+8+10+10+9+...+7+7+6+7 +7+6+6+6+5)+$\frac{1}{25}\times$(1+2+...+25)$\times$2= $\frac{103}{3}$. Here, $\frac{1}{81}$ represents the probability of choosing one group of reaction soldiers led by the sol-captain in the pyramid structure. 10, 9, 8, 7, 6, 5 signify the comparing times to find the specific group of reaction soldiers led by the sol-captain in different positions. $\frac{1}{25}$ corresponds to the probability of choosing one reaction once a time in the reaction group with 25 group members in the last level. 1, 2...25 indicate the comparing times of specific soldiers in different positions in the group. Moreover, using the old model it is $\frac{1}{5942}\times$(1+2+...+5942)$\times$2=5943. Here, $\frac{1}{5942}$ represents the probability of choosing one reaction among 5942 reactions. 1, 2,..., 5942 signify the comparing times of specific soldiers in different positions in the network. The pyramid structure method is efficient when simulating a large network using the kinetic Monte Carlo model. In the next section, we will discuss how it is implemented into the kinetic Monte Carlo model using C language.

4.4 Physical Data Structure for Pyramid Structure and Implementation in the Kinetic Monte Carlo Algorithm

The common physical structure for a nonlinear logical structure like a pyramid structure is a pointer (or pin), which is a special data type to point to another data type in C language to create the relationships between structs in the pyramid structure. For the pyramid data structure we defined in Section 4.1, the pointer of the current data struct points to its upper captain and to three lower captains or groups of soldiers. However, there are too many pointers in the program when applied to a big network, making it extremely difficult to find the correct physical address of the data. Then we look into the data and operations again, finding that they have two important characteristics:

1) The data themselves are stable without changing the sequence during the whole simulation.

2) The data have stable relationships. They have a fixed upper captain and three lower captains or groups of soldiers.

Accordingly, we use an array of structs as the physical data structure. The array is stable and easy for the computer to find the physical address. Structs of captains are built after the structs of reaction soldiers as part of the array of structs, which is shown in Figure 2(b).

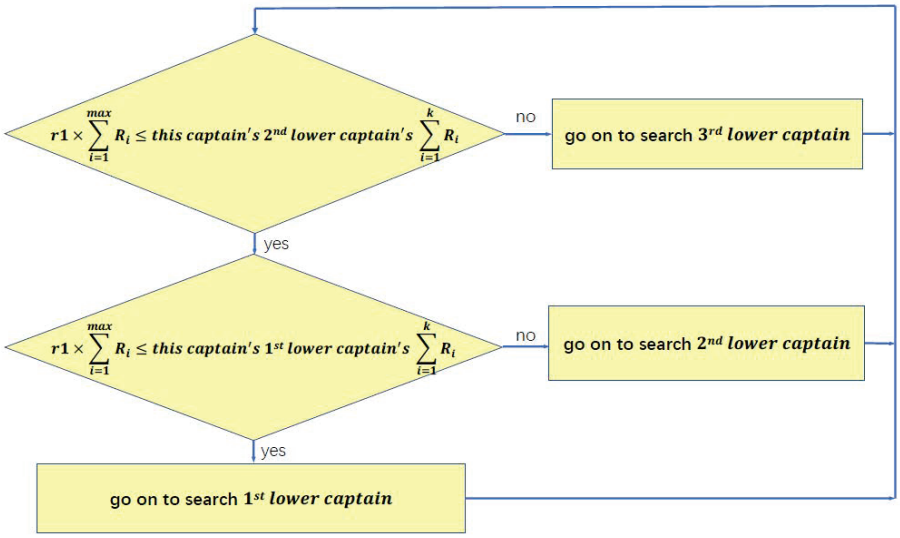

Fig. 3 Main process for searching for captains.

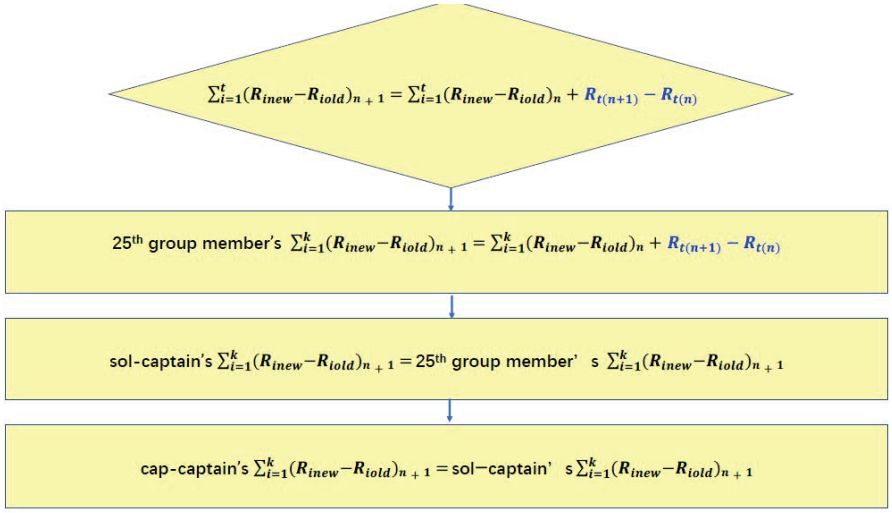

Fig. 4 The main updating process.

The specific process of the code is divided into three parts.

1) *Building*

The struct array is used to store the structs of captains and soldiers. When programming, we use cycles to define all data structs from the bottom to the top of the pyramid, i.e., from the soldiers to the chief captain.

2) *Searching*

To find the reaction that occurs, the pyramid data structure is searched from the chief captain to the soldiers. This process is already discussed in Section 4.2. The main process for captains is depicted in Figure 3. The searching stops at the level of sol-captains.

As a result, the captain which satisfies Equation (1) is found. The reaction that occurs must be one of the reaction soldiers under it. Then, $\sum_{i=1}^{k} R_i$ for each reaction soldier under the captain is compared with $r_1 \times \sum_{i=1}^{\max} R_i$ to find which reaction occurs.

3) *Updating*

In the old model, the data struct only contains information on the reaction and $\sum_{i=1}^{k} R_i$. Updating simply needs to replace the reaction rates whenever they are changed. In the accelerated method, the updating process is similar except that other related captains' elements (reaction rate and $R_{\text{inew}} - R_{\text{iold}}$) should also be updated. As illustrated in Table 1, there is an element, $\sum_{i=1}^{k}(R_{\text{inew}} - R_{\text{iold}})$, which stores the sum of the rate difference. The last reaction soldier in the group has a special $\sum_{i=1}^{k}(R_{\text{inew}} - R_{\text{iold}})$, which is the sum of all rate differences from the first reaction to this reaction. The main process for updating is depicted in Figure 4. In Figure 4, $\sum_{i=1}^{t}(R_{\text{inew}} - R_{\text{iold}})_{n+1}$ represents reaction t's $\sum_{i=1}^{k}(R_{\text{inew}} - R_{\text{iold}})$ in the $(n+1)$ loop. $R_{t(n+1)}$ and $R_{t(n)}$ signify the t reaction's rate in the

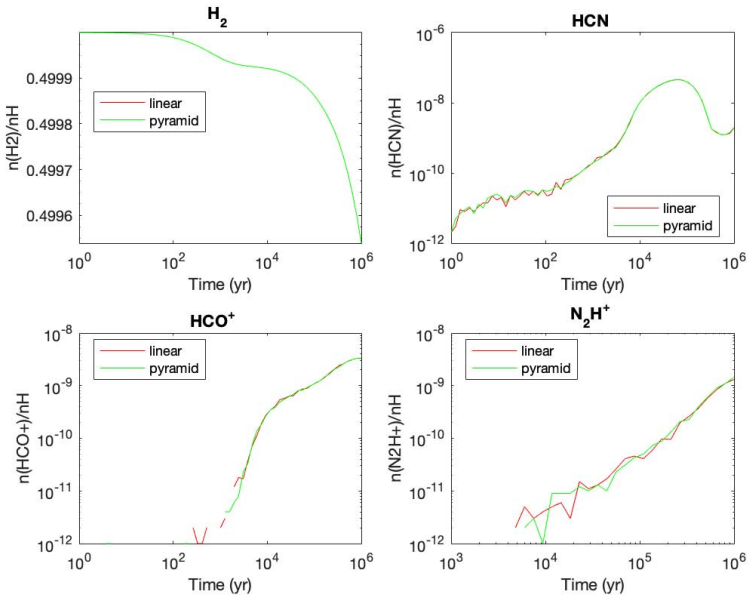

Fig. 5 Comparison of the results from the old model and the improved model. Red lines represent results from using the linear structure, and green lines signify results from implementing the pyramid structure.

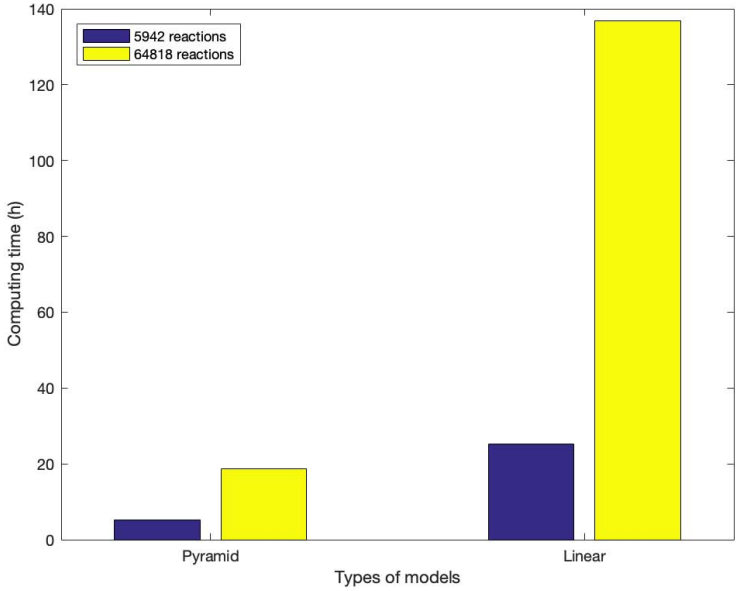

Fig. 6 Comparison of the computing time for the two models that use the pyramid structure and the linear structure, respectively.

$(n+1)$ loop and (n) loop, respectively. 25^{th} group member corresponds to the last member in one reaction group.

It takes more time for the new structure to update because the change of one reaction rate will influence the whole structure. So when changing the corresponding reaction rates, we have to change all the relevant captains to make them obtain the right value of $\sum_{i=1}^{k} R_i$ from the three lower captains (or the last soldiers in three groups of reaction soldiers).

Here we compare the time of the old model with the time of the accelerated model with pyramid structure. The whole time complexity using the old linear structure is $\frac{1}{5942}$ $\times$(1+2+...+5942)$\times$2+2n=5943+2n, in which 5943 is the time for searching and n is the average number of reactions that need to be updated. Since each updating is one operation (changing the reaction rate), it takes time n to find n reactions to be updated and another n's time to update n reactions. The whole time complexity using the pyramid structure is $\frac{1}{243}\times$(10+10+9+10+10+9+9+9+8+10+10+9+...+7+7+6+7+7+6+6+6+5)+$\frac{1}{25}\times$(1+2+...+25)$\times$2+5n+605=$\frac{103}{3}$+5n+605, in which $\frac{103}{3}$ is the time for searching, 605 is the time

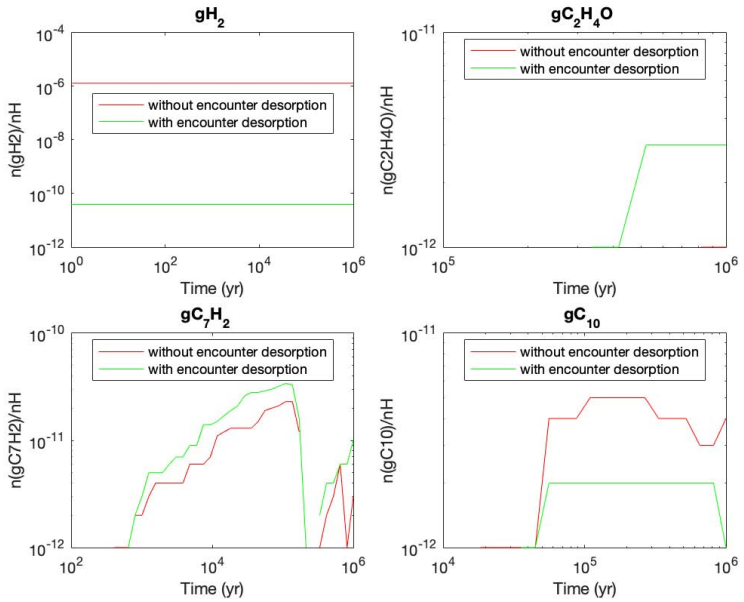

Fig. 7 Comparison of the results with the encounter desorption and without the encounter desorption using the improved model with the quasi-steady-state approximation. The red lines represent the results without encounter desorption, and the green lines signify the results with the encounter desorption.

to update the structure only about captains, $5n$ is the time taken for updating the reaction's and the last group soldier's rate and $\sum_{i=1}^{k}(R_{\rm inew} - R_{\rm iold})$, since there are five operations: (1) find the reaction to update; (2) change the reaction rate; (3) calculate the difference (to get $R_{\rm inew} - R_{\rm iold}$); (4) add to its $\sum_{i=1}^{k}(R_{\rm inew} - R_{\rm iold})$; (5) add the value to the $25^{\rm th}$ soldier's (the last soldier in one group) $\sum_{i=1}^{k}(R_{\rm inew} - R_{\rm iold})$. From these calculations, we conclude that when n is smaller than 1768, implementing the pyramid structure method is much more efficient than the method that relies on the old linear structure. This is usually the case because in most chemical reaction networks the number of reactions related to a specific reaction is generally much smaller than 1768.

5 RESULTS AND DISCUSSION

5.1 A comparison between the old model and the improved kinetic Monte Carlo model

To test the accuracy of the new structure, we use the osu-2008 network without the reaction of encounter desorption. The network consists of 5942 reactions. The temperature is 10 K, and the initial density of H atoms is $2.0\times10^4\,{\rm cm}^{-3}$.

As displayed in Figure 5, results from both models have very good agreement with each other. Due to random fluctuations, the abundance of N_2H^+ exhibits small differences at the beginning. In comparison, 96% of the abundances of the species agree with each other better than 90%, and all the gas-phase species' abundances are the same as those in the model with the linear structure. The computing time varies due to the calculated performance of different computers but the improved model is always much more efficient. When using a computer with a CPU main frequency of 3501 Mhz, it took about 8 h 58 min to finish the simulation with the improved model, while it took 45 h 1 min with the old model. We also tested a large gaseous network containing isotope D with 64 818 reactions for further comparison (Sipilä et al. 2013, 2015). The total time that we simulated in the two models is 4.17×10^5 yr using the networks with 5942 and 64 818 reactions, respectively. The physical parameters were kept fixed in all simulations. The differences in the computing time are compared in Figure 6. In Figure 6, the computing time of the simulation using the pyramid and the linear structure in the network with 5942 reactions is 5.19 h and 25.27 h, respectively. The computing time of the simulation using the pyramid and the linear structure in the network with 64 818 reactions is 18.77 h and 136.91 h, respectively. This demonstrates that for a larger network consuming a large amount of calculation time, the pyramid structure remarkably improves the efficiency.

5.2 A Comparison between the Results with and without Encounter Desorption using the Kinetic Monte Carlo Model

The aim is to investigate the effect of the encounter desorption ($gH_2 + gH_2 \rightarrow H_2 + gH_2$). The physical conditions were the same as those in Section 5.1.

We utilized the quasi-steady-state approximation both in the data structure accelerated model with the encounter desorption and that without the encounter desorption.

From Figure 7, it can be seen that the abundance of gH_2 is more without the encounter desorption in the simulation. The straight lines are due to the influence of the quasi-steady-state approximation, which assumes that the abundance of gH_2 is constant in the whole process. The encounter desorption affects 4.1% of the grain-surface species by a factor of more than two. The largest difference lies in gC_2H_4O, gC_7H_2 and gC_{10}, with the ratios (the fraction of the abundances of species in the pyramidal structure over the abundances of species in the linear structure) of 0.33, 0.3 and 4.0, respectively. The reason for the change of abundances for other species is due to the change of the abundance of gH_2.

6 CONCLUSIONS

We introduce a new data structure for searching the specific reaction to improve the efficiency of the kinetic Monte Carlo model, which saves a lot of computing time and is also applicable to other kinds of networks. We use the model with the data structure to investigate the effect of the encounter desorption. In the future, we plan to apply the method to a very large gas-grain reaction network consisting of tens of thousands of reactions including deuterium.

The pyramid structure also has other forms. It may have more levels and more branches, i.e., one captain can lead more lower captains or groups of soldiers. The new data structure makes it possible to apply the kinetic Monte Carlo model to larger chemical networks or more complex physical environments. The new model with the data structure offers great advantages over the old model which takes a lot of computing time. The data structure extends the number of specific applications where the kinetic Monte Carlo model can be utilized.

Acknowledgements The authors acknowledge the constructive and fruitful comments from the anonymous referee, which significantly improved the quality of this paper. This work is supported by the CAS "Light of West China Program" (2017-QNXZ-B), the Heaven Lake Hundred-Talent Program of Xinjiang Uygur Autonomous Region of China, the National Natural Science Foundation of China (Nos. 11673054 11873082, U1531125, 11803080, 11503075, 11543002, 11673054 and 11703075), the National Key Basic Research Program of China (973 Program 2015CB857100), the National Key Basic Research and Development Program (2018YFA0404704) and Youth Innovation Promotion Association CAS. The research is administrated by the Chinese Academy of Sciences (CAS). The algorithm and debugging work utilized the Taurus High Performance Computing Cluster of Xinjiang Astronomical Observatory, CAS.

References

Baloukas, C., Risco-Martin, J. L., Atienza, D., et al. 2009, Journal of Systems & Software, 82, 590
Cazaux, S., & Tielens, A. G. G. M. 2004, ApJ, 604, 222
Chang, Q., Cuppen, H. M., & Herbst, E. 2005, A&A, 434, 599
Chang, Q., & Herbst, E. 2012, ApJ, 759, 147
Chang, Q., Lu, Y., & Quan, D. 2017, ApJ, 851, 68
Cuppen, H. M., & Herbst, E. 2007, ApJ, 668, 294
Du, F., & Parise, B. 2011, A&A, 530, A131
Gao, Z.-F., Wang, N., & Shan, H. 2017, Astronomische Nachrichten, 338, 1060
Garrod, R. T., & Herbst, E. 2006, A&A, 457, 927
Garrod, R. T., & Pauly, T. 2011, ApJ, 735, 15
Gillespie, D. T. 1976, Journal of Computational Physics, 22, 403
Hasegawa, T. I., & Herbst, E. 1993, MNRAS, 263, 589
Hasegawa, T. I., Herbst, E., & Leung, C. M. 1992, ApJS, 82, 167
Hassel, G. E., Herbst, E., & Garrod, R. T. 2008, ApJ, 681, 1385
Herbst, E., & Shematovich, V. I. 2003, Ap&SS, 285, 725
Hincelin, U., Chang, Q., & Herbst, E. 2015, A&A, 574, A24
Hincelin, U., Wakelam, V., Commerçon, B., Hersant, F., & Guilloteau, S. 2013, ApJ, 775, 44
Lipshtat, A., & Biham, O. 2003, A&A, 400, 585
Morata, O., & Hasegawa, T. I. 2013, MNRAS, 429, 3578
Pickles, J. B., & Williams, D. A. 1977, Ap&SS, 52, 443
Semenov, D., Hersant, F., Wakelam, V., et al. 2010, A&A, 522, A42
Sipilä, O., Caselli, P., & Harju, J. 2013, A&A, 554, A92
Sipilä, O., Caselli, P., & Harju, J. 2015, A&A, 578, A55
Stantcheva, T., & Herbst, E. 2004, A&A, 423, 241
Taquet, V., Ceccarelli, C., & Kahane, C. 2012, A&A, 538, A42
Tarjan, R. E. 1983, in Cbms-nsf Regional Conference Series in Applied Mathematics
Tarjan, R. E. 1984, Journal of the Acm, 31, 245
Tielens, A. G. G. M., & Allamandola, L. J. 1987, in Astrophysics and Space Science Library, 134, Interstellar Processes, eds. D. J. Hollenbach, & H. A. Thronson, Jr., 397
Tielens, A. G. G. M., & Charnley, S. B. 1997, Origins of Life and Evolution of the Biosphere, 27, 23
Viti, S., Collings, M. P., Dever, J. W., McCoustra, M. R. S., & Williams, D. A. 2004, MNRAS, 354, 1141
Watson, W. D., & Salpeter, E. E. 1972, ApJ, 174, 321