

UNIVERSITY OF HAWAII

MCMC SIMULATION OF INTERSTELLAR GRAIN  
CHEMISTRY

*Author:*

Pavel SENIN

*Supervisors:*

Dr. Kim BINSTED,

Dr. Jacqueline KEANE

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF  
MASTER OF SCIENCE

December 3, 2007

# Contents

<b>1</b>	<b>Context</b>	<b>4</b>
1.1	Thesis goal . . . . .	4
1.2	Structure of the thesis . . . . .	4
<b>2</b>	<b>Survey.</b>	<b>5</b>
2.1	Background. . . . .	5
2.2	Granular models and previous work. . . . .	7
<b>3</b>	<b>Interstellar grain chemistry model.</b>	<b>9</b>
3.1	The chemical model. . . . .	9
3.2	Accretion . . . . .	9
3.3	Surface mobility and chemistry . . . . .	10
<b>4</b>	<b>Software model design and implementation.</b>	<b>12</b>
4.1	General principles. . . . .	12
4.2	Software implementation. . . . .	15
4.3	Software features. . . . .	17
<b>5</b>	<b>Results and evaluation.</b>	<b>19</b>
<b>6</b>	<b>Future work</b>	<b>19</b>

## Abstract

Chemical reactions on interstellar dust grains play a crucial role in interstellar chemistry by promoting the formation of organic products. While many of the reaction rates are poorly understood and molecular formation routes are difficult to isolate in the laboratory, computer simulations of these reactions allows us to better understand the nature and evolution of interstellar molecules in molecular clouds. The work presented here is part of the CASS 2006 initiative which aimed to involve graduate students from diverse regions of science into the field of Astrobiology. This particular collaboration resulted in the development of computational software that implements a MCMC stochastic model of the surface chemistry. While the current model only simulates grain-surface chemistry, we show the capabilities and vast potential of coupling the grain-surface with a full-scale interstellar gas chemistry model. We have improved upon the original IDL model by implementing a Java based product that provides numerous options for set-up and tuning of simulations along with real-time visualization of the model results.

# ACKNOWLEDGEMENTS

I would like to thank to Dr. Kim Binsted for the given opportunity to join Computational Astrobiology Summer School 2006 at UH and for being a great supervisor. I highly appreciate the confidence that you have in me.

Many thanks to Dr. Jacqueline Keane for the constant support, interest and enthusiasm for my research.

I thank to Dr. Philip Johnson for helping me with all software measurements and interpretation.

# 1 Context

## 1.1 Thesis goal

This thesis describes the design and implementation of Java-based software called *ICLOUDS* which stands for *I*nterstellar *CLOUDS*. The goal of developing this system is to facilitate a computational model that is capable of simulating the complex stochastic process of chemical reactions occurring on interstellar dust grains. The reason for creating these simulations is straightforward: the computer simulations allow us to experiment with different setups at relatively low cost compared to wet-lab experiments which are expensive and not always precise since mimicking interstellar conditions (temperature, density) is very difficult.

## 1.2 Structure of the thesis

Section 2, “Survey”, begins with an introduction to the interstellar medium and a description of the chemistry that occurs on the dust grain surfaces. This section provides the background information for readers unfamiliar with those particular areas. Then an overview of current research trends is presented. Emphasis is placed on the aspects relevant to designing the ICLOUDS simulation toolkit.

Chapter 3, “Interstellar grain chemistry model”, details the process of designing the theoretical model.

Chapter 4, “Software model design and implementation” discusses the general principles behind the software design and details the software model flow along with features provided.

Chapter 5, “Results and Evaluation”, describes the accomplished simulations results and compares those with the real observational data.

Chapter 6, “Conclusions and future work”, summarizes the contributions of the thesis to the astrochemistry field and discuss several areas for future work.

## 2 Survey.

### 2.1 Background.

Though the space between stars in our galaxy appears to be completely void of matter, there exist a very dilute gas medium and low density of grain particles. The interstellar medium varies in temperature, density and composition. Typical densities in the interstellar medium are one particle per cubic centimeter ( $1\text{cm}^{-3}$ ), but some regions have densities that are 10,000 times greater than average. The generic name given to a conglomeration of gas, plasma and dust in space is “interstellar cloud”; more precisely, an interstellar cloud is a denser-than-average region of the interstellar medium.

Interstellar clouds are generally classified by their density, size and temperature. Depending on the state of the hydrogen, clouds are divided into 3 regions: neutral ( $H$ -I regions), ionized ( $H$ -II or  $H^+$  regions, ie. a plasma), or molecular ( $H_2$  or molecular clouds). Neutral and ionized clouds are called diffuse clouds, while molecular clouds are also referred to as dense clouds. Dense regions arise from diffuse clouds that develop sufficient density to be self-shielded from ultraviolet radiation, heating and shocks which dissociates molecular hydrogen. A typical molecular cloud in the Milky Way may contain up to several million solar masses of cold gas and span tens of parsecs across ( $1\text{parsec} = 30.857 \times 10^{12}\text{km}$ ). The average temperatures within these clouds are 10 - 20 Kelvins above absolute zero and the gas is mostly in the form of hydrogen molecules, with trace amounts of other molecules such as carbon monoxide ( $CO$ ). The millimeter-wavelength emission of  $CO$  molecules allows mapping and studying of these clouds.

As mentioned, in addition to gas, interstellar clouds contain dust grains composed of a core and mantle, Figure 1. The dust core is typically composed of silicate, carbon or iron compounds. The mantle is a conglomerate of atoms and molecules frozen out from the interstellar gas. These grains are microscopic, about  $10^{-5}$  cm across, irregularly shaped and are even less common in space than atoms or molecules; there is 1 dust grain for every 1 trillion gas particles. However, considering the vast volume of observed clouds in space, the total number of dust particles within a cloud can be considerable.

The Molecular clouds were discovered in our Milky Way Galaxy in the mid 1960’s by astronomers

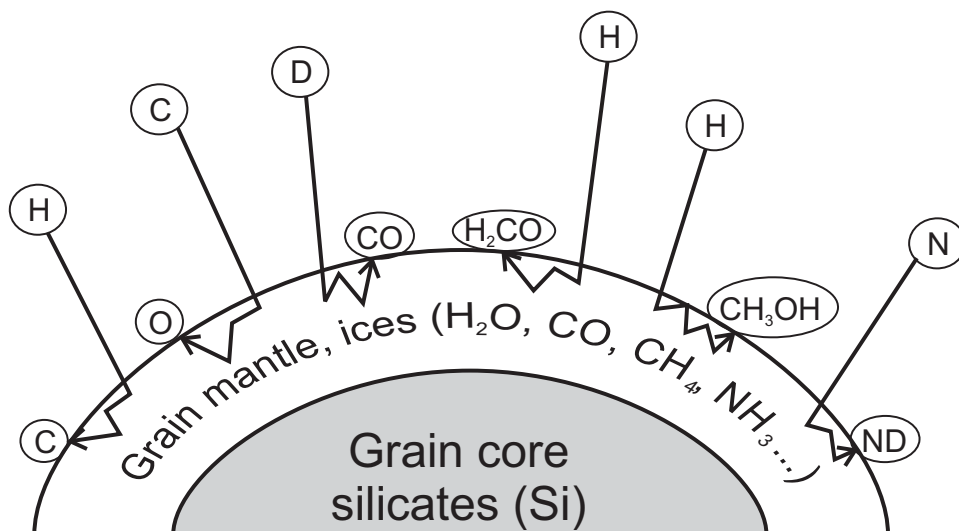


Figure 1: The conceptual drawing of a dust grain along with complex molecules formation routes. Once a reactive species arrives at the surface, it diffuses or tunnels to the site with the coreactant and reaction happens (see Figure 3).

using radio telescopes tuned to the millimeter wavelengths at which many kinds of molecules in space emit radiation. Today interstellar molecules are observed spectroscopically either through emission lines or through absorption lines in a medium in front of continuum radiation source, such as a star or a quasar, Figure 2. These lines can be observed via radio-, millimeter-, or infrared wavelengths, depending on the molecule transition and the type of spectral line.

Since the first observations in 1960's the picture of the interstellar medium has evolved into a rather complex one driven by the newest telescopes. There are many conditions that are not always straightforward to explain such as being hot or cold, dense or diffuse, ionized or neutral, moreover there are many regions with molecular features that overlap and become difficult to interpret. As for the role of Interstellar clouds, they are the birthplace of all stars, planets, comets, asteroids and by extrapolation ... us!

While many physical aspects of the star formation process are still poorly understood, another challenging problem from both observational and theoretical grounds is the chemical evolution of the material. Within the cloud medium, high-resolution spectroscopy has identified more than 100 different gas-phase molecules. The gas-phase chemistry can qualitatively explain the presence

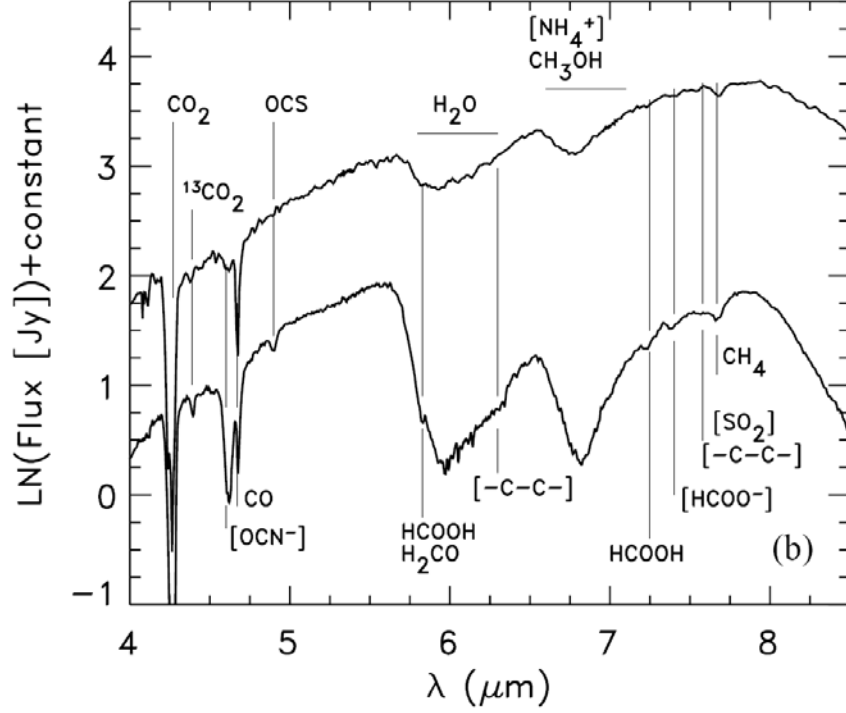


Figure 2: The rich mid-infrared ISO/SWS spectra of the massive protostars NGC 7538 showing the richness of ice absorption features: IRS9 (top) and W 33A (bottom). Data taken from Whittet et al. (1996), Gibb et al. (2000), Boogert&Ehrenfreund (2004). Identifications labeled in square brackets are uncertain.

of many molecules, but not all of them. For example the most abundant molecule in interstellar clouds - the simplest hydrogen molecule ( $H_2$ ) is exclusively formed on the grain surface. Many others species, such as methanol, can only be explained by chemistry on dust grain surfaces. This paper will present an effort to develop a stochastic model of interstellar molecular cloud chemistry.

## 2.2 Granular models and previous work.

### *Absorption and chemistry.*

The surface of the interstellar grain is considered by the contemporary science to be bare or covered with monolayers of molecules. Typical grain size is considered to be about  $\approx 100 \mu m$  but there is a wide distribution of sizes. The process of accretion of species from the gas phase in a typical cloud with density  $\approx 10^4$  occurs with a frequency about one species per day. Newly accreted and



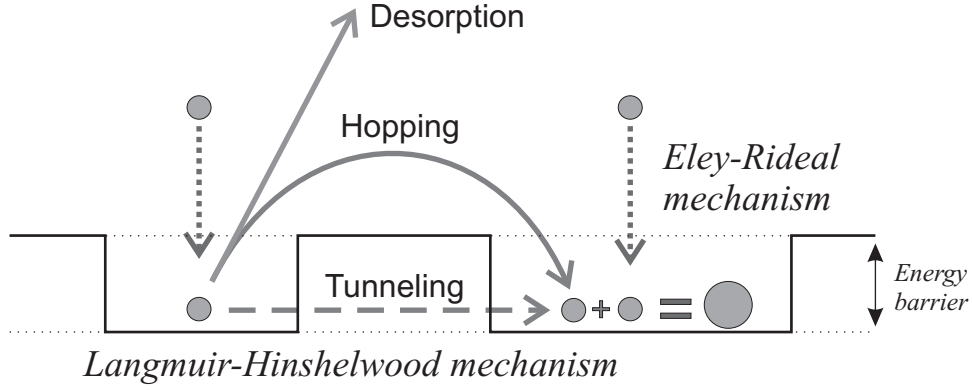


Figure 3: Grain surface chemistry processes. While moving species ruled by the Langmuir-Hinshelwood mechanism through the hopping or tunneling, non-moving species react through the Eley-Rideal mechanism.

bonded to a surface species does not stay stationary but can travel to other binding sites either by thermal hopping over the barriers between sites or by quantum tunneling through them, Figure 3. As for the barrier width it is the actual energy value sufficient for the species evaporation. (Tielens & Hagen 1982, Tielens & Allamandola 1987, Hasegawa et al. 1992). The shape of the grain in such models is considered to be flat with about  $10^6$  binding sites. The ability to diffuse is governed by the mass of the atom and in general only  $H$ ,  $O$ ,  $C$ ,  $D$  and  $N$  are light enough to diffuse across the grain surface. If after diffusing to an other site the particle finds a species to react with, reaction will occur if the activation barrier is low or zero. The chemistry model based on such assumptions is known as Langmuir-Hinshelwood chemistry. As for the species that are tightly bound to the surface the chemistry is ruled by Eley-Rideal mechanism, (Tielens 1993), Figure 3.

#### ***Evaporation.***

The evaporation of molecules from the grain surface is very slow in dense molecular clouds for all species except atomic and molecular hydrogen and helium due to the low cloud temperature, however mechanisms such as sputtering in shocks, sputtering and evaporation due to the cosmic ray bombardment, heating from exothermic chemical reactions, photodesorption, grain-to-grain collisions, explosions, etc. cause enrichment of the gas phase by the evaporation of surface chemistry products. (Williams 1993, 1998).

#### ***Previous work.***

In the early studies interstellar chemistry was clearly divided by two grounds: solely gas-phase chemistry and static gas-phase chemistry coupled with a dynamic grain-surface chemistry. In the first approach it was believed that grains played a passive role in interstellar chemistry by acting simply as accumulators of molecules that condensed out from the gas phase. A first well known exception was  $H_2$ , which had been shown to be formed exclusively on grains and changed the common assumptions, [3], [4]. The work by Pickles & Williams (1977), Tielens and Hagen (1982) and d’Hendecourt et al. (1985) introduce two methods that are in use today - the Monte-Carlo (stochastic) approach and the rate equations approach. These studies highlighted that a substantial fraction of the cataloged gas phase species can not in fact be exclusively formed through gas phase chemistry or that their abundances are considerably greater than what a gas phase origin can account for. When both methods if started from the purely gas phase propagate the picture of a complex on-grain chemical population in agreement with observations, the preference today is given to the Monte-Carlo method not only because its theoretical and computational simplicity but because it better approximates (mimics) the nature of interstellar on-grain chemistry. It was found that it is considerably difficult to propagate all of the time-dependent chemistry rate changes in the rate-equations model.

This work takes as the basis for construction the original model and computational software designed by J. Keane and A. G. G. M. Tielens which adopted the Monte-Carlo sampling, [1]. The model was presented at the Computational Astrobiology Summer School held August 2006 and deals with two processes: accretion of species from gas phase and on-surface chemistry. During the after-presentation discussion there were two issues with the current software implementation that were pointed out: first there is need for software stochastic method improvement with dynamic gas phase coupling, and the second issue was making the software available for the public domain.

While the first problem could be solved by the redesigning of the original code, the overall cost of IDL platform (that starts from \$3000 for the basic single-user package) eliminates the second goal completely. Our choice of Object-Oriented design paradigm and Java development platform was quite natural due to the reasons that will be discussed in the Software model Design section.

To meet the second goal, public domain availability, the Google Code public project hosting site was selected as the SVN repository host for the process of development. The software from the very early stages is available for download from <http://code.google.com/p/iclouds>. The source

code, Javadocs, User Manual and “jump-start” batch files and configuration are available within the distribution. Preference was given to the Google hosting site due to a number of Google features such as Issue/Bug tracker, Wiki engine, Blogger and Google-talk groups that were used to make the project as user, and collaboration friendly as possible.

### 3 Interstellar grain chemistry model.

#### 3.1 The chemical model.

The following section of the writing is adopted from the original article by J. Keane and A. G. G. M. Tielens [1] and highlights the underlying theory.

The “scoundrel approach” that was adopted by authors, considers that chemical reactions occurring on the grain surface are solely responsible for the formation of the molecular species investigated here, [5]. The model implements the Monte Carlo accretion limited method [6]. Two essential and simplifying assumptions rule the chemistry processing:

- the accretion timescale  $\ll$  the timescales for reaction of the accreted atom  $H$ ,  $N$ ,  $C$ , or  $O$ , with other species;
- there is no desorption of grain mantle species and thus the surface chemistry is not coupled back to the gas-phase.

The reactions network considered here is based on reactions that occur only between weakly bound sites: from one physisorbed to another physisorbed site.

#### 3.2 Accretion

The gas composition is limited to the species that dominate the bulk composition of the gas, i.e.,  $H$ ,  $C$ ,  $CO$ ,  $O$ ,  $O_2$ ,  $N$ , and  $N_2$ . A gas-phase species  $i$  accretes on to the grain at a rate:

$$R_{acc}(i) = n_i \bar{v}(i) \pi a^2 \tag{1}$$

where  $n_i$  and  $\bar{v}(i)$  are the gas-phase abundance relative to hydrogen and the mean velocity of the species  $i$ , and  $a$  is the grain size. Thermal velocities of the gas phase species are given by:

$$\bar{v}(i) = \sqrt{\frac{3kT_g}{m_i}}$$

where  $m_i$  is the mass of the gas species  $i$ ,  $T_g$  is the temperature of the gas (10K) and  $k$  is the Boltzmann constant. It is assumed that the species once accreted stays on the surface. Accretion is a random process with a probability given by:

$$P_{acc}(i) = \frac{R_{acc}(i)}{\sum_j R_{acc}(j)}$$

where the summation is over all gas phase species. The accretion timescale,  $\tau_{acc}$ , is then given by:

$$\tau_{acc}^{-1} = \sum_j R_{acc}(j) \quad (2)$$

For a 1000 Å grain,  $\tau_{acc} \approx 10^5$  at  $n = 10^4 \text{ cm}^{-3}$  and  $T = 10\text{K}$ .

### 3.3 Surface mobility and chemistry

It is assumed that all species are physisorbed rather than chemisorbed on the grain surfaces. The mobility of an accreted species, i.e. the ability to diffuse from one physisorbed site to another, is crucial to the likelihood of a reaction occurring on the grain surface. Theoretical studies of H diffusion on ice surfaces show that quantum mechanical tunneling dominates the diffusion process at 10K with a diffusion timescale which varies between between  $10^{-12}\text{s}$  and  $10^{-9}\text{s}$  depending on the assumptions [3][7][8]. Atoms heavier than  $H$  ( $N$ ,  $C$ , and  $O$ ) are presumed to diffuse through thermal hopping:

$$\tau_{hop}^{-1} = v_0 \exp \left[ -\frac{E_{dif}}{kT_d} \right]$$

where  $T_d$  is the dust temperature (10K) and  $E_{dif}$  represents the barrier that must be overcome in order for the species to diffuse from one site to another.  $v_0$  is the characteristic vibrational frequency (i.e., represented as an harmonic oscillator) of the grain surface species, and a value of  $10^{12}\text{s}^{-1}$  is have chosen to use.  $E_{dif} \approx 0.3E_b = 240\text{K}$ , where  $E_b$  denotes the binding energy of the species to the grain surface, which is typically 800K for atoms on an icy grain surface [6]. This leads to a thermal hopping timescale of  $\approx 10^{-2}\text{s}$  for these atoms. The time it takes a diffusing species to visit

all sites on the grain surface (i.e. scan the surface) is:

$$\tau_{scan} = N_{\tau}$$

where depending on the diffusing species  $\tau$  is either  $\tau_{tun}$  or  $\tau_{hop}$ . For a 1000 Å grain,  $N \approx 10^6$  and  $\tau_{scan}$  is then  $\leq 10^{-3}$  for atomic H and  $10^4 s$  for C, N, and O. Both of these are less than the accretion timescale. We note that heavier atoms, radicals or molecules are more strongly bound to an icy surface [6] and hence, on an accretion timescale, such species are immobile.

The process of roaming the grain surface is essentially that of a random walk. An accreted species will remain on the grain surface for a time equal to time it takes the species to thermally desorb (evaporation timescale):

$$\tau_{evap} = v_0^{-1} \exp \left[ \frac{E_b}{kT_d} \right]$$

where relevant binding energies are given by [8]. The desorption timescale of atomic H is then  $\approx 10^{-3} s$ , i.e. much less than the accretion timescale but larger than  $\tau_{scan}$ . For atomic C, N, and O as well as all other species the evaporation timescale is long compared to all other relevant timescales (at 10K). A mobile H atom can visit each reacting site then at most:

$$\eta = \tau_{evap} / \tau_{scan}$$

which for the parameters above is  $\approx 10^6$ . Of course, if the atom reacts rapidly,  $\eta$  may be much less. For the other atoms, the relevant timescale to consider is the accretion timescale, and  $\eta$  is then 10.

Model assumes that species with unpaired electrons react upon “collision” on a grain surface. Reaction of atoms with molecules having paired electrons may also occur upon collision depending on kinetic considerations [6]. The probability for an H atom ( $i$ ) to react with a non-radical species ( $j$ ) depends on the barrier against reaction (activation barrier,  $E_{a_j}$ ):

$$P_{i,j} = \tau v_0 \exp \left[ -\frac{2a}{\hbar} \sqrt{2mE_{a_j}} \right]$$

where  $\tau$  represents the time spent in a site (in this case  $\tau_{tun}$ ). If the reactant is not an H atom (i.e. either N, C, or O) then the term in the exponent is replaced by the likelihood of thermally hopping over the activation barrier ( $E_a/kT_d$ ). The probability of an atom reacting within one evaporation timescale ( $\tau_{evap}$ ) is then given by:

$$\phi_{i,j} = \theta_j k P_{i,j}$$

where  $\theta_j$  is the surface concentration of the co-reactant and  $k$  is the number of times the atom enters a specific site before it either reacts or evaporates. We note that the diffusion timescale drops out, and thus the reaction probability in each visit is independent of the exact rate at which a species diffuses across the surface. A faster diffusion timescale limits, of course, the reaction probability but that is exactly counteracted by the increase in the collision rate. The potential of a reaction occurring is then determined by the rate of finding a coreactant:

$$\phi_{i,j} = \theta_j$$

where  $\tau_i$  is either due to tunneling or to thermal hopping, depending on the nature of the diffusing species. The key point is that a reaction will occur within an evaporation timescale for  $H$ .

The overall reaction probability of species  $i$  with species  $j$  must then be weighted by all the reactions that are possible for species  $i$ :

$$R_{i,j} = \phi_{i,j} / \sum_{l=1}^n \phi_{i,l} \quad (3)$$

where the summation is over all reactions possible. Now, within this competition of surface species for a newly accreted  $H$ , when there is another radical on the surface, atomic  $H$  will react with this radical even if the surface is otherwise completely covered ( $\theta = 1$ ) with reaction partners with  $E_a$  as low as 450K. On the other hand, if there is no radical “waiting on” the surface,  $H$  will “select” among the coreactants according to the activation barrier and the surface concentration of the coreactants involved. We stress once more that the reaction probability (Equation 3) is independent of the diffusive timescale and the total reaction time (e.g., the exact value of  $k$ )

## 4 Software model design and implementation.

### 4.1 General principles.

During the preliminary problem analysis we found that the interstellar cloud environment, a grain itself and each of interstellar medium species could be viewed as objects and the process of accretion and surface chemistry that model describes is an evolution of these objects in time. This perspective naturally fits an object-oriented design (OOD) paradigm which was expected to benefit future development through reusability and extensibility granted by the modular design, [12][13][14].

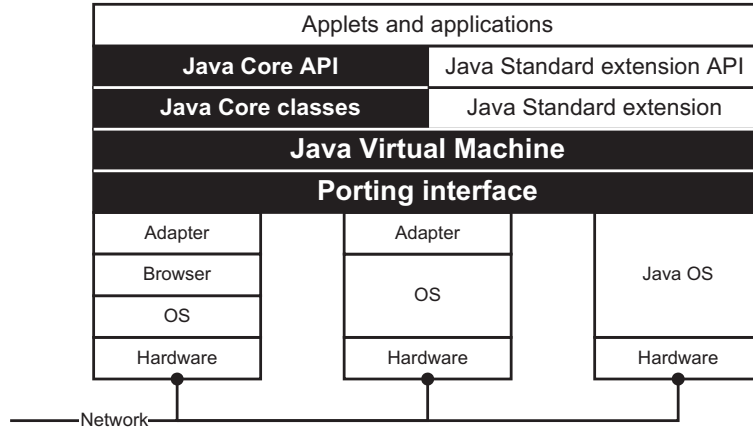


Figure 4: The JVM architecture. JVM is able to run on a variety of platforms providing the same run-time environment for the user program.

Among all available object-oriented languages Java was a natural choice due to the cross-platform compatibility. By choosing Java based development we ensured the ability of our software to run with little or no change on multiple hardware platforms. This option granted by the abstraction layer that Java Virtual Machine (or JVM) provides over the particular hardware and operating system implementation. Basically it is a “soft computer” that runs user programs within. This machine is “virtual” because it is implemented over the real hardware platform and operating system shown in Figure 4. Among other Java features, JVM robustness and reliability, rich options of Collection Framework, generic interfaces and easy threads management along with Javadocs and variety of tools for code development and quality control were taken in account.

While designing the software we facilitated the achievement of the following design goals:

- decomposability - each particular problem was divided into the smaller subproblems;
- composability - constructing a system as the whole from several subsystems;
- understandibility and decoupling - minimizing the number of related modules needed to understand and implement a particular module;
- continuity - minimizing the number of modules that need be changed when making a small change for a problem specification;
- protection - minimizing the number of modules affected by a run-time exception.

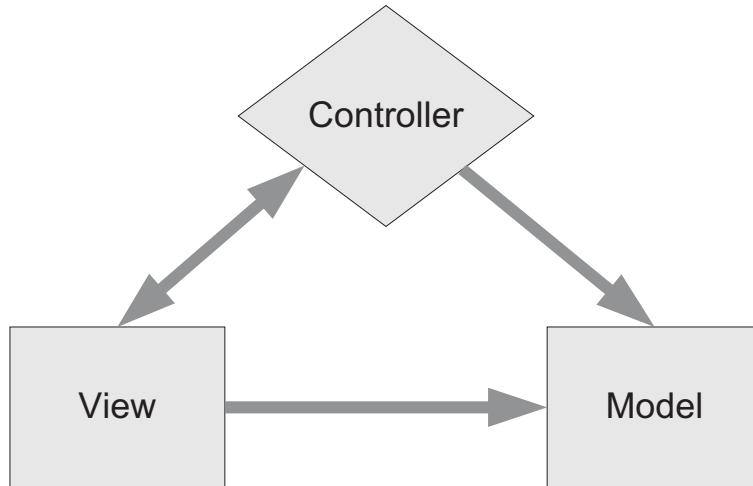


Figure 5: The MVC architecture overview. By dividing the application into a Model, View, and Controller we can separate the presentation from the model logic and data processing (computation).

As for the general design pattern, before starting the project we had a working software implementation in IDL that was stable and mature enough to highlight the design limitations. This situation ensured that the right model architecture was adopted. Among different software design patterns we considered, the *Model-View-Controller* was chosen as the constructing paradigm resulting in an efficient and flexible implementation. The *MVC* design pattern is shown in Figure 5 is a popular object oriented design paradigm which is defined by the SmallTalk environment, [16]. The goal of the MVC design pattern is to separate an application between the data it has, the way it presents the data, and the way it allows interaction with the data. To achieve the separation, MVC design pattern proposes to separate an application into three components, i.e., Model, View, and Controller which will be described in turn:

- **Model:** the core of an application. This part maintains the state and data that the application contains. It does the actual data manipulation, computation and reacts to the Controller component input, but knows nothing about the way the data is displayed. Whenever a significant change happens in the model data, the Model updates all the Views registered to it. The Java provides the Observer-Observable framework that allows to accomplish such association, The Observers (*Views*) notified each time when the Observable (*Model*) fires the change.



- View: the user interface which could be an attached Java-based GUI module or a web-application within the browser backed by the Application Server or even a remote client, which displays the information contained in the model to the user. The View has to be a registered Observer with the Model to access the information.
- Controller: this component handles the user interactions with the view and transforms the user interactions on the view to the actions to the model which then reacts and fires changes into the View component. In our case we are going to extend the Controller functionality through the implementation of external data injection to the Model from the full-scale gas model.

By adopting such an architectural decision we managed to separate the data (Model) and user interface (View) from each other. Future changes to the user interface and migration toward the web-based interface will not impact the data handling, and the data can be reorganized without changing the user interface.

## 4.2 Software implementation.

The model flow is a Monte-Carlo Markov Chain simulation, i.e. each step solely depends on previous one and each of the steps is random. While there is no algorithm available to run such a model in parallel we managed to shorten the overall multiple-simulation experiment time by encapsulating the whole simulation process within the single Java class that implements Java Runnable interface. This decision enables us to eliminate between-runs and set-up pauses by running multiple and scheduling a number of simulations on the multi-CPU (multi-core) architecture. The queue handled by the Java Executor Service interface within the Controller component.

The data flow within the model could be viewed as the functional structure shown in Figure 6. The simulation is set up by the three main parts within the configuration file: the cloud physical parameters, the cloud population and the chemistry network. These three sections are wrapped by XML container and provided to the software before the start of a simulation. The convenient XML-formatted config is easy to understand and maintain, moreover the file could be easily reused to set up different simulations by overriding the specific parameters within the command-line arguments

or in the case of GUI usage by calling appropriate methods implemented within the Model from the Controller component.

Before setting up and starting a simulation run, software parses the configuration file, extracts and checks for consistency in all the parameters needed for simulation setup. The reaction network provided will be checked for consistency of atom amount between coreactants and products. Once all configured parameters are found to be valid, the main method initializes the internal structures for keeping accounting of parameters and starts the simulation.

The way simulation goes is naturally mimics a real-world accretion and chemistry process through the stochastic sampling, Figure 6.

### ***Accretion - sampling from the gas.***

At the first stage the software samples a species population from the virtual cloud medium into the intermediate storage. As shown in Figure 6, “*Sampler #1*” takes as an input the cloud general setup parameters (density, temperature, etc.) and the species population data. All species within the cloud have an equal probability to be sampled at the each sampling iteration. These probabilities are weighted by the species abundances and its thermal velocities. It should be noted that all samplings within the software done by using the custom random number generator that is an implementation of the “Mersenne Twister” algorithm which is expected to be “truly random” on the long run due to the extremely large period, about  $2^{19937}$ , [17], [18].

The purpose of having an intermediate storage for sampled species is to break the software simulation cycle into chunks so in the future it can be coupled with gas-phase models. It is planned to adjust the cloud population and cloud general parameters to fit the gas-phase model results at each chunk sampling event. Among future options is to insert at this point the simulation for the evaporation mechanism that would simulate a process of species depletion from the surface.

When the amount of sampled species within the intermediate storage meets the preconfigured number, the software proceeds to the chemistry processing phase.

### ***Surface chemistry.***

The software instantiates about a dozen storages at this stage before proceeding further - the surface *active population* and the surface *buried population* (Figure 6) are the main ones, the software will

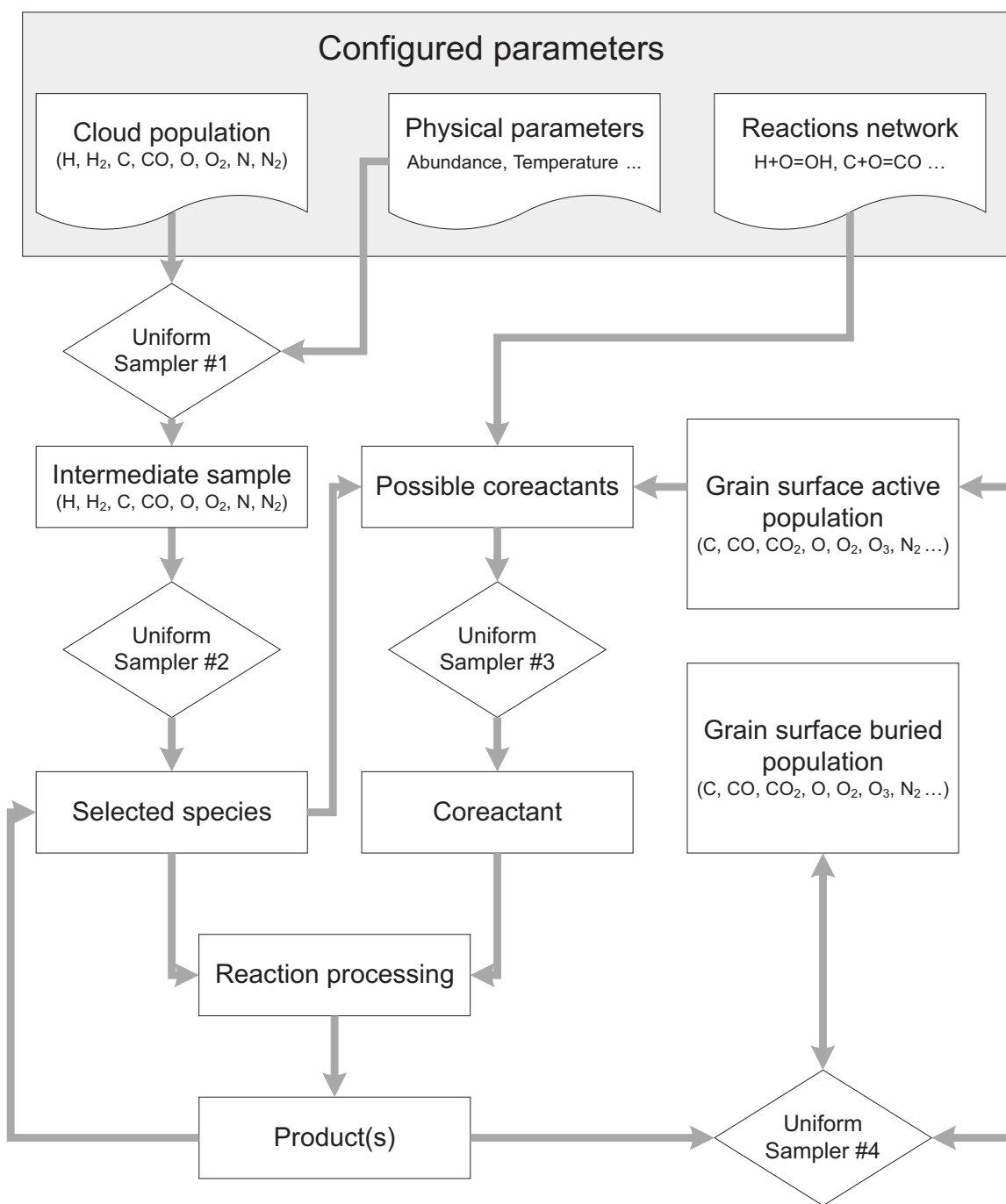


Figure 6: The sampling flow within the software. Multiple independent samplers mimics the real-life random process driven by the parameters provided.

keep these from iteration to iteration. Various temporary and statistics data storages will be renewed at the each iterations step.

- ***“Selected species” sampling.***

The species that drive the chemistry, *selected species*, could be picked from two sources, the previous chemical reaction products list or from the intermediate storage, Figure 6. More precisely, if the previous reaction products list is not empty the selected species picked from there in the FIFO order, if it is found empty, the selected species picked from the intermediate storage. Species are sampled from intermediate storage (*“Sampler #2”*.) with an equal probability that is weighted by species abundances.

- ***Chemistry processing.***

Once the species is selected, it will drive the chemistry. The configured reactions network is used to scan through the active population in order to find the possible coreactants list. At the next step the software samples the coreactant from this list by using the abundance-weighted probabilities (*“Sampler #3”*.) When coreactants are found for picking, they are removed from storages and the products depending on its type, active or passive, placed into one of the two storages - the grain surface population (surface or buried) or in the product list. Species is placed into the product list if it is reactive, i.e. able to drive chemistry. If species is proactive, the software samples among all grain placing sites and makes a decision about landing the species. If the sampled site is occupied by a species this species will be transferred into the buried population and the newly landed species will take its place among the active grain surface population.

The surface chemistry processing ends when all accreted species from the intermediate storage and all products are used. At this point the software checks, whether or not the accretion event limit is reached, if so, the main loop terminates and software proceed to the statistics output phase, if the limit is not yet reached it starts sampling from the gas again.

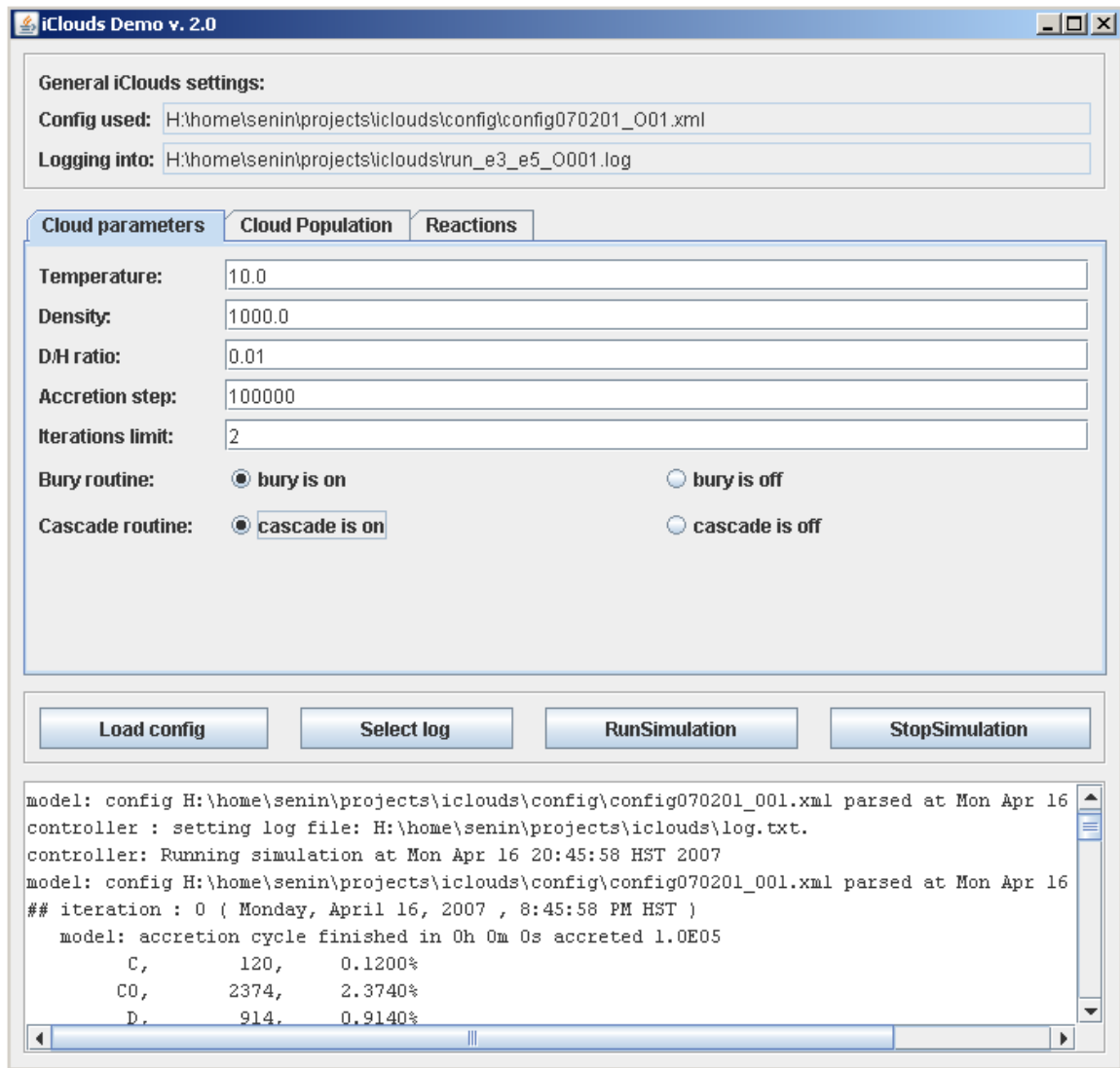


Figure 7: The ICLOUDS software screenshot showing the “Cloud parameters” panel.

### 4.3 Software features.

As pointed before, ICLOUDS software hosted within the public accessible domain and can be downloaded as the distribution package from homepage or checked out in source codes from the SVN repository. While we are developing the software using Eclipse development platform and providing the necessary Eclipse development data files, [24], the software is completely independent from any specific system setup. Platform-independent building scripts are included within the distribution. The scripts are designed to be used with Apache Ant build tool, and it is possible to build and run the system on virtually any platform that supports Java, [25].

ICLOUDS software can be built and run in two modes: command-line (CLI) and graphical (GUI) user interfaces. The GUI facilitates simulations setup and run, Figure 7, and visualization of the cloud population and chemistry network with computed derivatives parameters such as thermal velocities, accretion probabilities and chi values, Figures 8, 9. The CLI version is intended to be used in the batch-scheduled experiments and allows individual parameters setup through the command line. This option allows a same configuration file to be reused for a set of different experiments.

As the output of the simulation ICLOUDS provides two log files, one of the files is the *run statistics* that contains information about the experiment setup and final grain population data. The second log file is optional and provides the grain population *evolution statistics* data through making the periodical populations snapshots. This period is adjustable and measured in accretion events. While the *run statistics* is a text file that is human-readable and designed to be intuitively self-describing, the *evolution statistics* is provided by employing Java Logging API and can be in the text or XML format and contains information about snapshots time, logger class name, etc... Since Java Logging API provides a variety of log granularity levels we exercise this option to introduce logging levels into our code. By adjusting the granularity from the *none* to the *finest* level the logging mechanism gradually improves output and at the *finest* logging level the log file will contain full information about sampling and decision events that happened along with complete populations accounting. This option was found extremely valuable while designing and debugging the chemistry network used in our experiments.

In order to analyze the output more rigorously than by viewing logs produced, we developed and including into the distribution Java-based log parser(s) that convert the log data into the

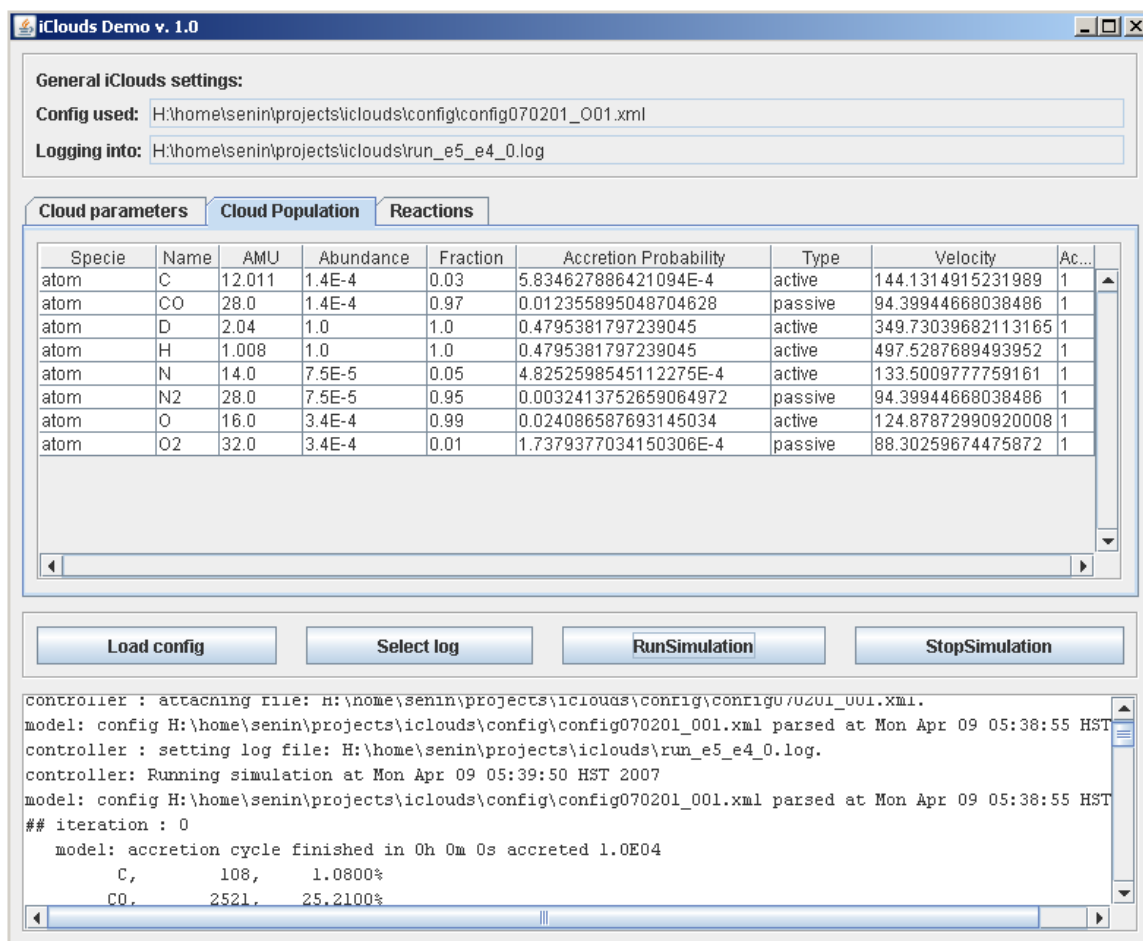


Figure 8: The ICLOUDS software screenshots of the “Cloud population” panels showing an overview of pre-configured cloud population and computed by software values.

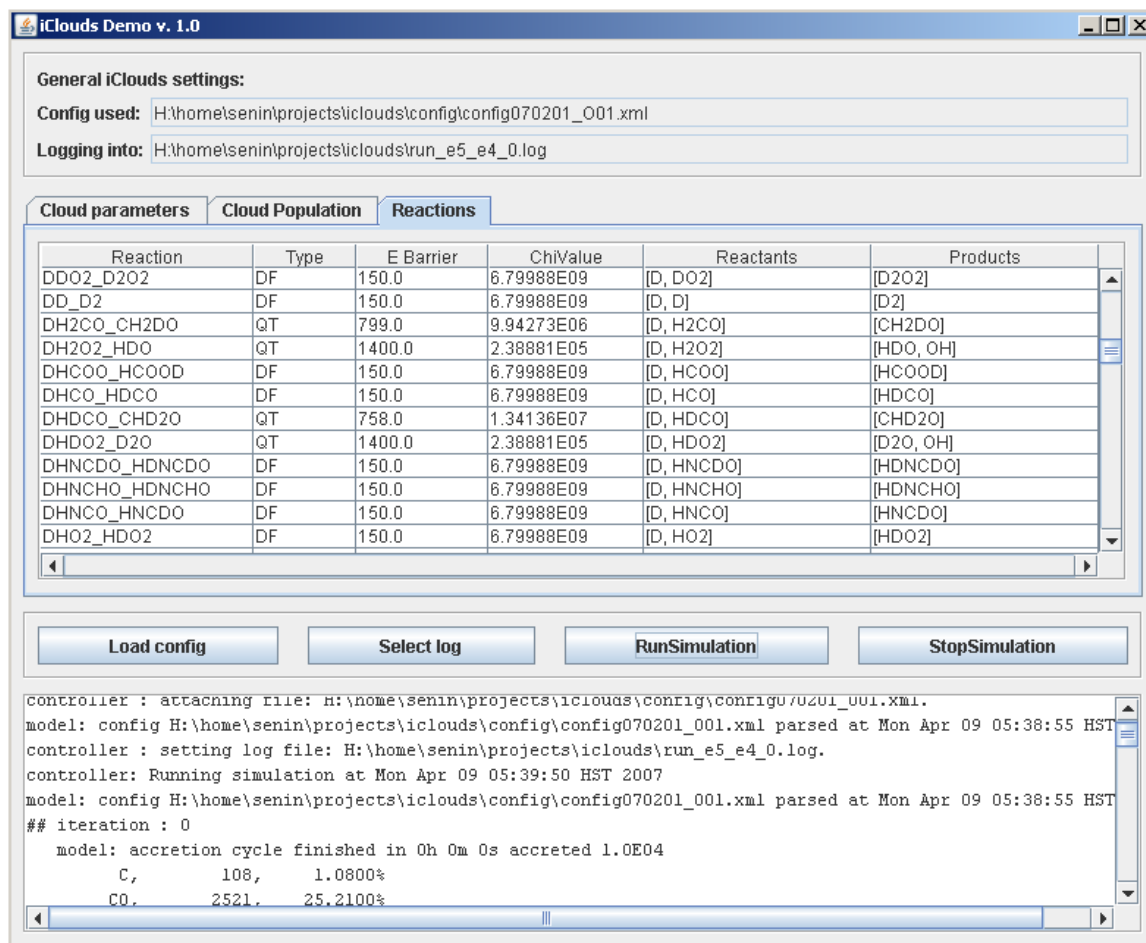


Figure 9: The ICLOUDS software screenshots of the “Reactions” panels showing an overview of configured chemistry network and computed by software values.



conventional .csv format which is readable by the most of the analytical software, Figure 10. In our case, the R (“GNU S”), a freely available language and environment for statistical computing and graphics, was chosen due to its computational and plotting capabilities, [15]. We provide core scripts within our distribution which can be used to produce plots which commonly used in Astrochemistry field.

Since the software was treated as an open-source package from the very beginning, in order to meet the best open-source community practices, such quality assurance tools like CheckStyle [26], JUnit [27], EMMA [28], JDepend [29], [30], FindBugs [31] and PMD [32] were set up and constantly used in the coding process. The set of cross-platform Ant scripts that run these tools manually or as the Cron scheduled job is provided within the distribution package. The natural way of monitoring the software development process and general trends was provided by the HackyStat system, which sensors are embedded within the scripts, Figures 13, 14, [33].

We documented every bit of our Java code within the software using Javadocs. This Javadocs can be built from the source distribution package by calling the Ant script provided. The general manual for the software usage and system configuration provided within the distribution and online as the Wiki.

## 5 Results and evaluation.

### *General achievements.*

The overall development priority for this phase of the project was given to the model code development. The majority of the work was spent ensuring that the core code is stable in a long run and produces meaningful results along with providing valuable features for the putative users. We kept all crucial components well-tested and were monitoring the software quality by using the Hackystat system. At the time of writing this report the Java code size is about 12’000 lines and it took about 200 hours of active development.

Current ICLOUDS software version provides many features within the distribution packages such as the fast set-up and run instructions and extensive analysis toolkit. By following these guidelines one can quickly and easily set up and run simulations, parse log files and extract specific

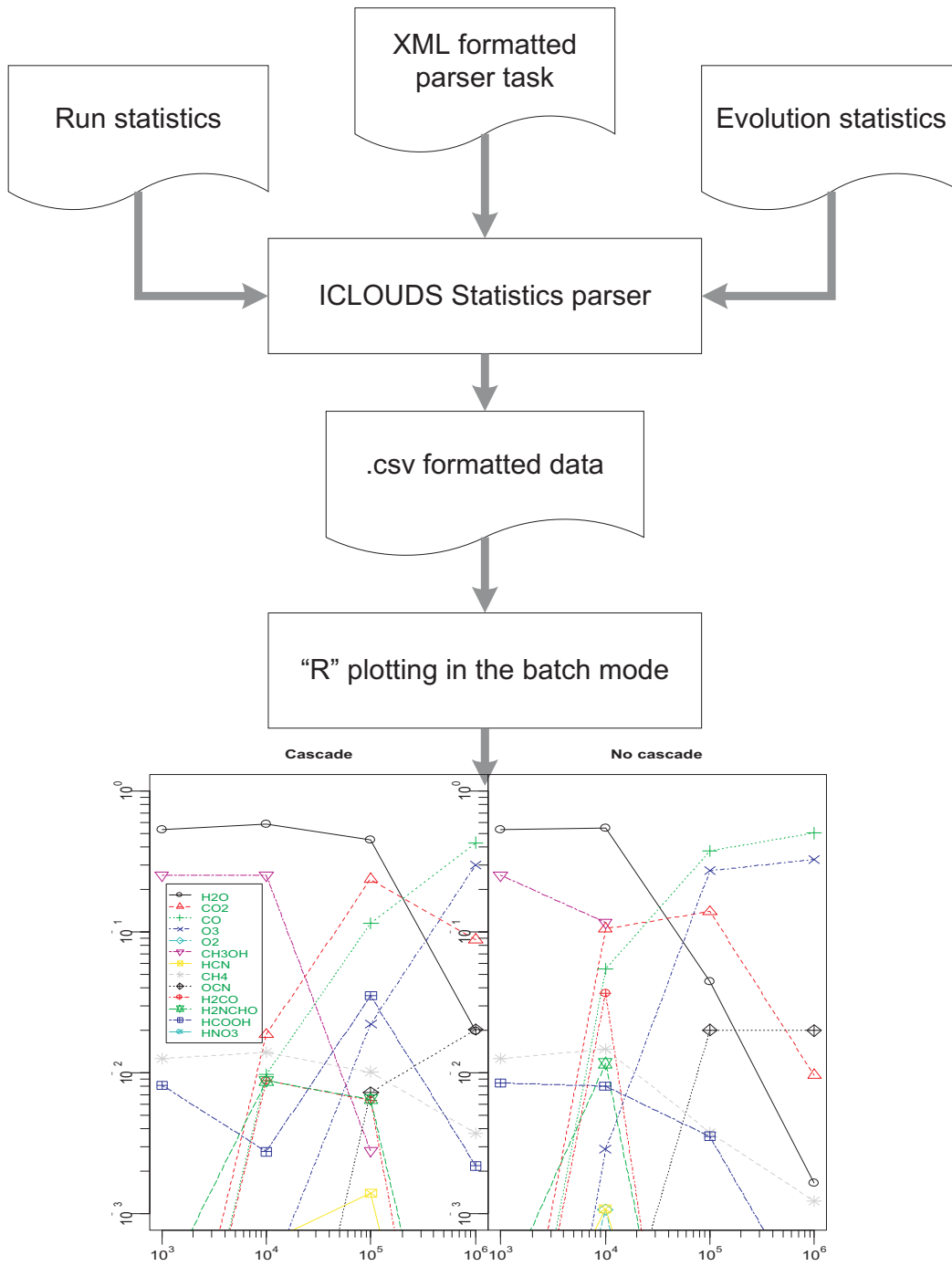
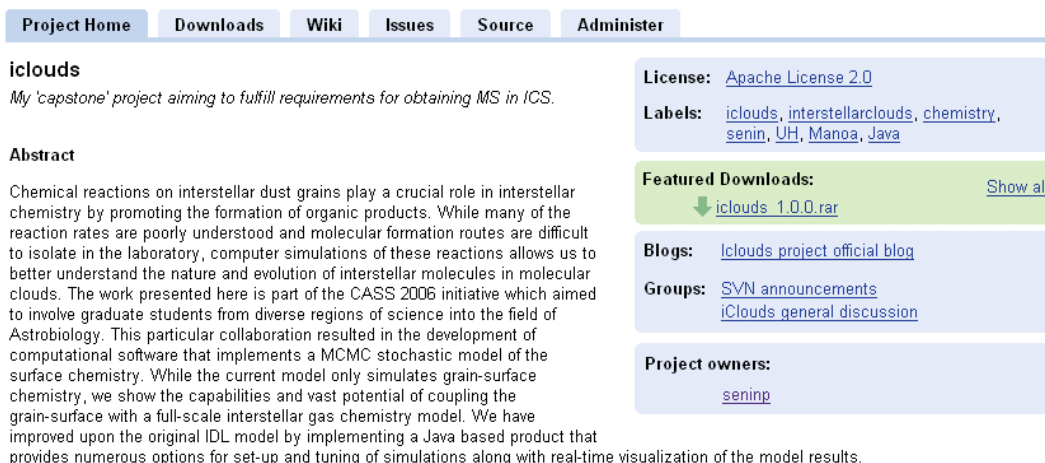


Figure 10: ICLOUDS provides an easy configurable simulations parser and R scripts for the plotting.



**iclouds**  
My 'capstone' project aiming to fulfill requirements for obtaining MS in ICS.

**Abstract**  
Chemical reactions on interstellar dust grains play a crucial role in interstellar chemistry by promoting the formation of organic products. While many of the reaction rates are poorly understood and molecular formation routes are difficult to isolate in the laboratory, computer simulations of these reactions allows us to better understand the nature and evolution of interstellar molecules in molecular clouds. The work presented here is part of the CASS 2006 initiative which aimed to involve graduate students from diverse regions of science into the field of Astrobiology. This particular collaboration resulted in the development of computational software that implements a MCMC stochastic model of the surface chemistry. While the current model only simulates grain-surface chemistry, we show the capabilities and vast potential of coupling the grain-surface with a full-scale interstellar gas chemistry model. We have improved upon the original IDL model by implementing a Java based product that provides numerous options for set-up and tuning of simulations along with real-time visualization of the model results.

**License:** [Apache License 2.0](#)

**Labels:** [iclouds](#), [interstellarclouds](#), [chemistry](#), [seninp](#), [UH](#), [Manoa](#), [Java](#)

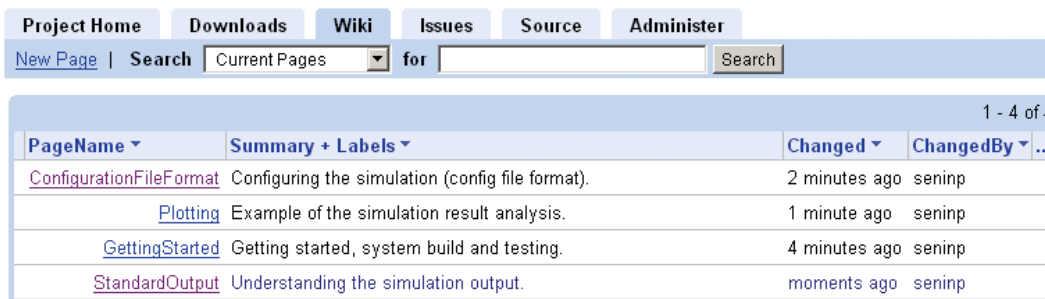
**Featured Downloads:** [iclouds\\_1.0.0.rar](#) [Show all](#)

**Blogs:** [iclouds project official blog](#)

**Groups:** [SVN announcements](#), [iClouds general discussion](#)

**Project owners:** [seninp](#)

Figure 11: ICLOUDS project home page at GoogleCode host the downloads section, wiki engine, issue tracking, and SVN repository interface.



PageName	Summary + Labels	Changed	ChangedBy
<a href="#">ConfigurationFileFormat</a>	Configuring the simulation (config file format).	2 minutes ago	seninp
<a href="#">Plotting</a>	Example of the simulation result analysis.	1 minute ago	seninp
<a href="#">GettingStarted</a>	Getting started, system build and testing.	4 minutes ago	seninp
<a href="#">StandardOutput</a>	Understanding the simulation output.	moments ago	seninp

Figure 12: ICLOUDS project Wiki engine provides guide for the system build, testing, simulation setup and results interpretation.

information along with generating standard plots using **R**.

The binaries, source code, all supportive scripts, examples and manuals are available online at the public hosting provided by the GoogleCode hosting, Figure 11. The homepage provides easy navigation and contains the documentation for the users and developers, Figure 12.

### *Computational results.*

The computational results section goes here.

## 6 Future work

While the initial goals are met, the theoretical model is implemented and software output rigorously checked and tested, there are some issues that we want to emphasize as the goals for the future software development:

- *The performance and memory usage issues.*

Because of the considerably long running time, we did an informal performance measurement (profiling) aiming the computational time improvement. We have run the test simulation over three different configurations: Intel Pentium 4 workstation, AMD64 workstation and Intel Itanium 2 shared-memory supercomputer. The memory consumption never went more than 256 Mb, but surprisingly, among others the AMD64 (x86-64) architecture found to be the fastest performer as seen in Table 1, the only reason for this could be the faster memory read-write cycle along with efficient 64 bits operations. Taking this reasoning as the starting point we altered standard Java garbage collection mechanism to perform more aggressively (concurrent low pause collector) and we gained about 10 - 20% in performance. In our opinion to improve the overall performance further we will need to alter the software memory management behavior. Specifically in the next iteration we plan to experiment with two options in order to improve the overall garbage collection time: the first is to alter objects initialization in order to help the garbage collector in distinguishing old and young object generations and the second is arbitrary garbage collection calling from the simulation loop.

Table 1: The ICLOUDS performance (test run,  $10^5$  density,  $10^7$  accretion).

Hardware and OS configuration	Simulation run time
Pentim 4, 3.2Ghz, HT, PC-3200 DDR, Open Suse Linux 10.0, Java HotSpot(TM) Client VM (build 1.6.0-b105, mixed mode, sharing)	3h 7m 44s
AMD64 “Venice” 3000+, 1.8Ghz, DDR-400, Open Suse Linux 10.2, Java HotSpot(TM) 64-Bit Server VM (build 1.6.0-b105, mixed mode)	0h 47m 39s
SGI Altix 350, Itanium 2, 1.6Ghz, PC-2100 DDR, Suse Enterprise Server 9, BEA JRockit(R) (build R27.1.0-109-73164-1.5.0_08-20061129-1425-linux-ia64)	1h 36m 23s

- *The code quality.*

The overall development priority for this project term was given to quantitative goals instead of qualitative. We put the major effort to make the code stable along with providing necessary features for the complete analysis of the computation results. While the ICLOUDS 2.0 code is stable and clean in terms of code style it has a low test coverage. Our code test coverage was degrading over the time of development, Figure 13 and should be fixed in the next development iteration. Among other priorities the next step will involve the analysis and cleaning the code from PMD and FindBugs warnings, Figure 14.

- *Web user interface development.*

We are planning to implement the web-interface (WUI) in the one of the next development iterations with a simulation scheduling and monitoring support. The general idea is to encapsulate the Model component within the Java Application server, implement the Controller component as the servlet and design the View component using Java Server Faces technology.

As for the overall model evolution through the coupling with the gas phase model, we have done the experiment of the actual gas-phase coupling by external parameters data injection. While this proves the general coupling concept, we are looking forward to improve the overall simulation precision through dynamic parameters estimation by facilitating a gas-phase model. It is not clear whether we will code the gas-phase model within the ICLOUDS package or employ a third-party model to extract the data needed, but we will put the most of our effort to implement this feature.

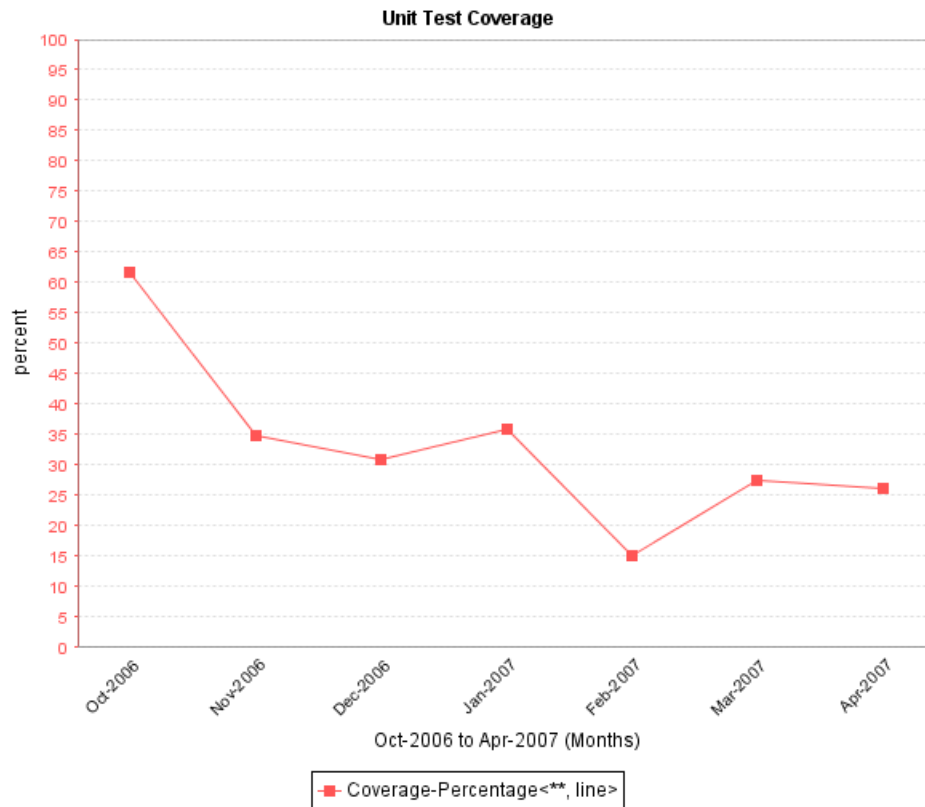


Figure 13: ICLOUDS unit test code coverage statistics.

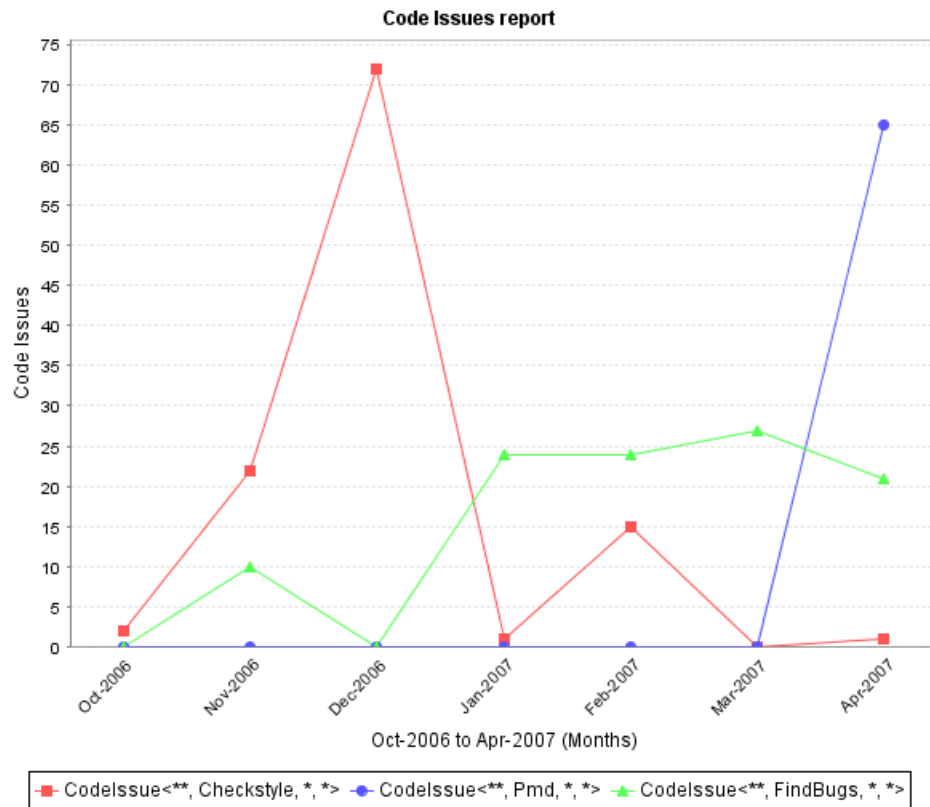


Figure 14: ICLOUDS code issues statistics.

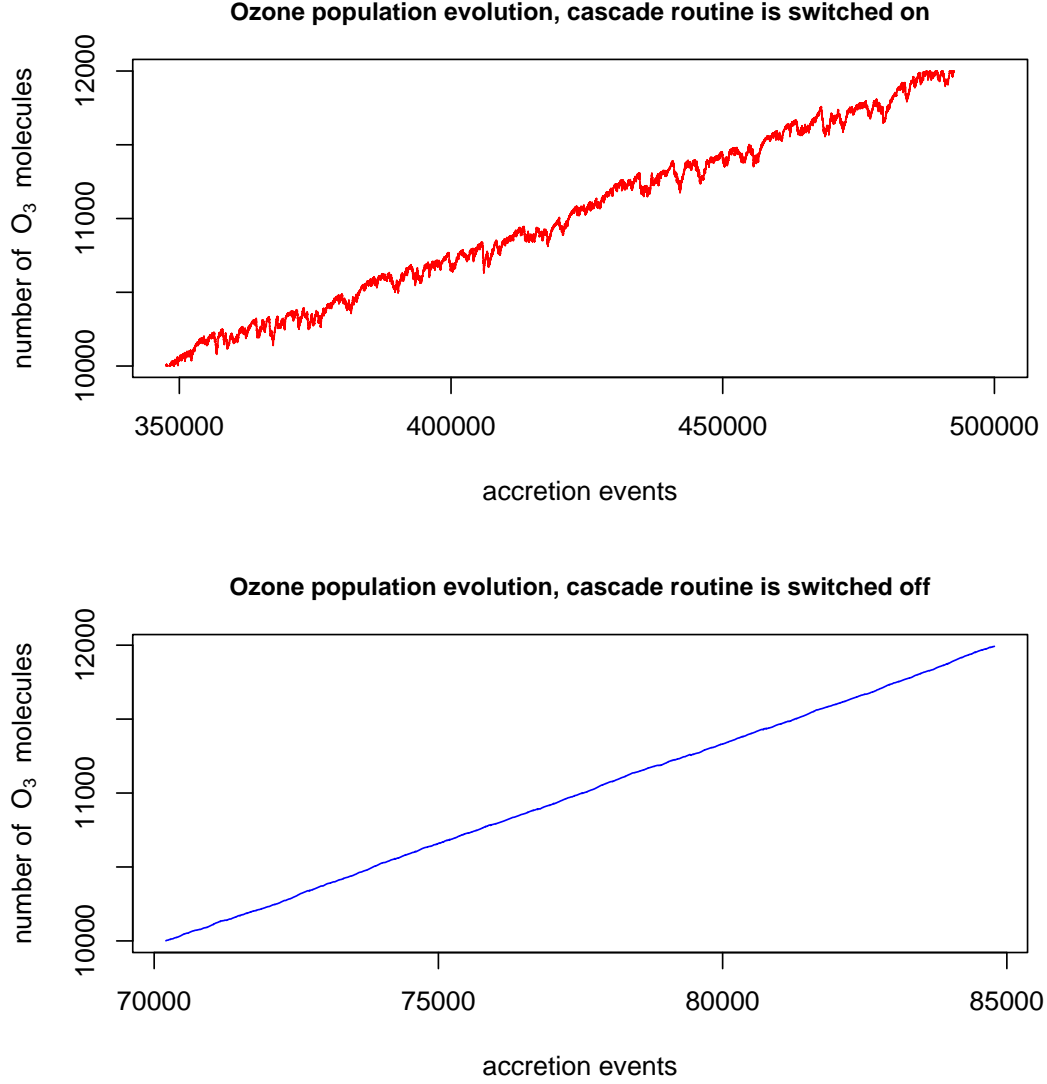


Figure 15: Illustration of the effect of the  $O_3$  cascade on the ozone molecules on the grain surface. The sawtooth behavior (upper panel) results from free H formed from reactions  $O_3 + OH(OD)$  and  $H_2 + OH(OD)$ . H remains on surface cycling between these reactions and continuously reducing the number of  $O_3$  molecules. The lower panels shows the behavior of the ozone molecules when the cascade is switched off.



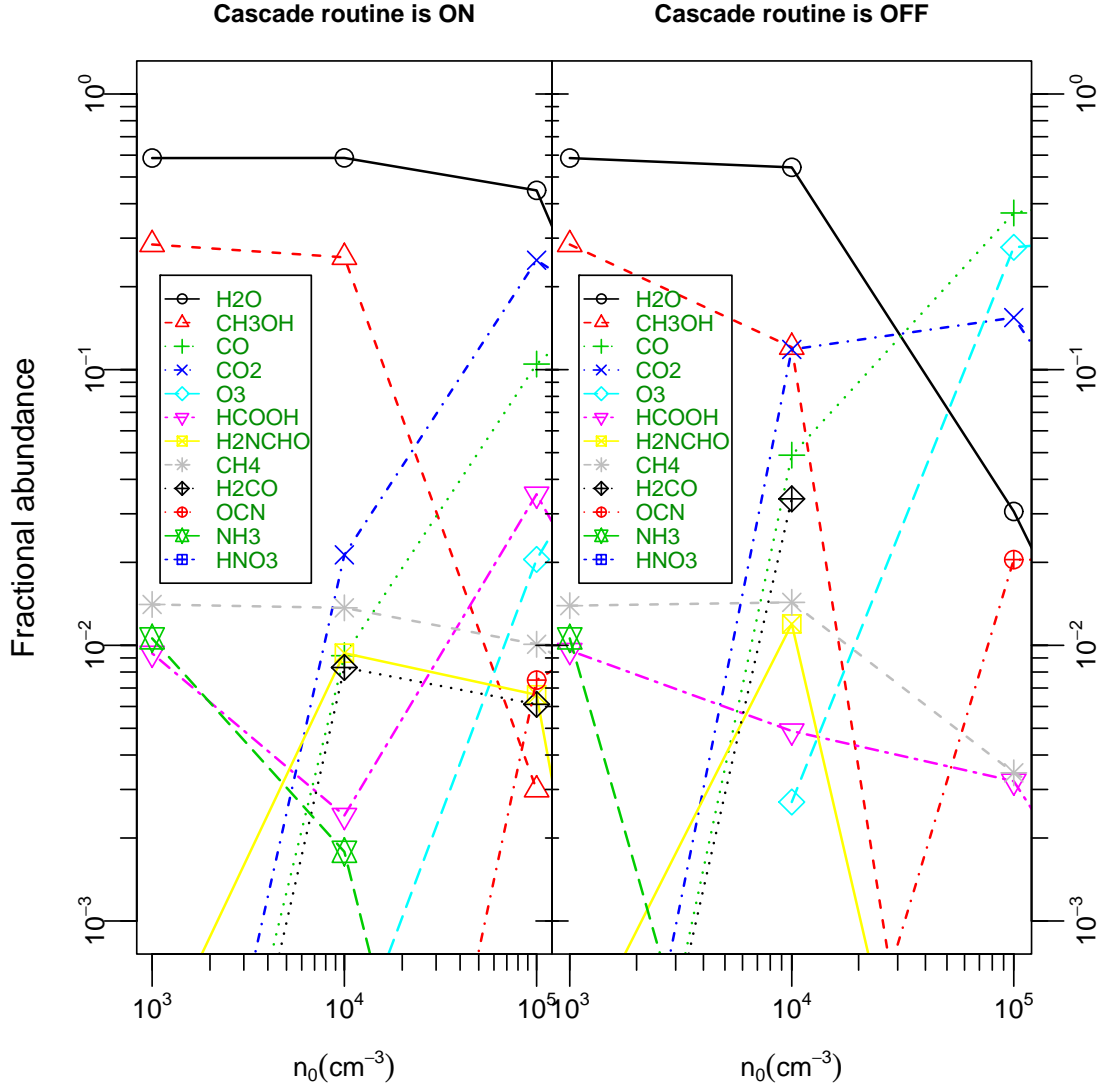


Figure 16: Fractional abundances of the major solid-state H-bearing grain surface species as a function of hydrogen density for a  $D$ -to- $H$  ratio of 0.01. Left panel shows the chemistry that results from the  $O_3 - OH$  cascade and right panel shows the grain surface composition that occurs when the cascade is switched off.

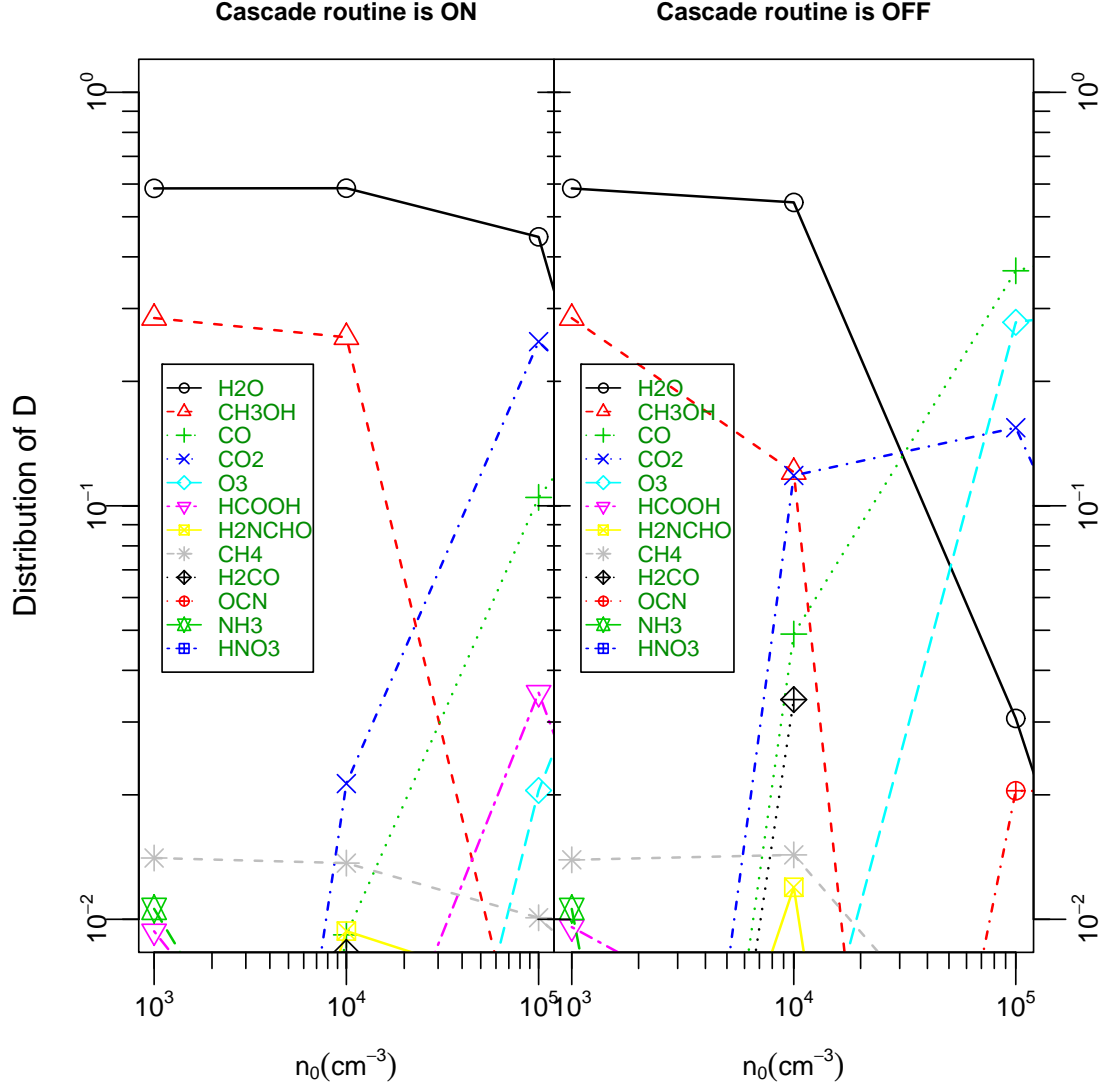


Figure 17: The distribution of deuterium amongst the D-bearing species as a function of density, for  $D$ -to- $H$  ratio of 0.01. Left panel shows the chemistry that results from the  $O_3 - OH$  cascade and right panel shows the grain surface composition that occurs when the cascade is switched off.

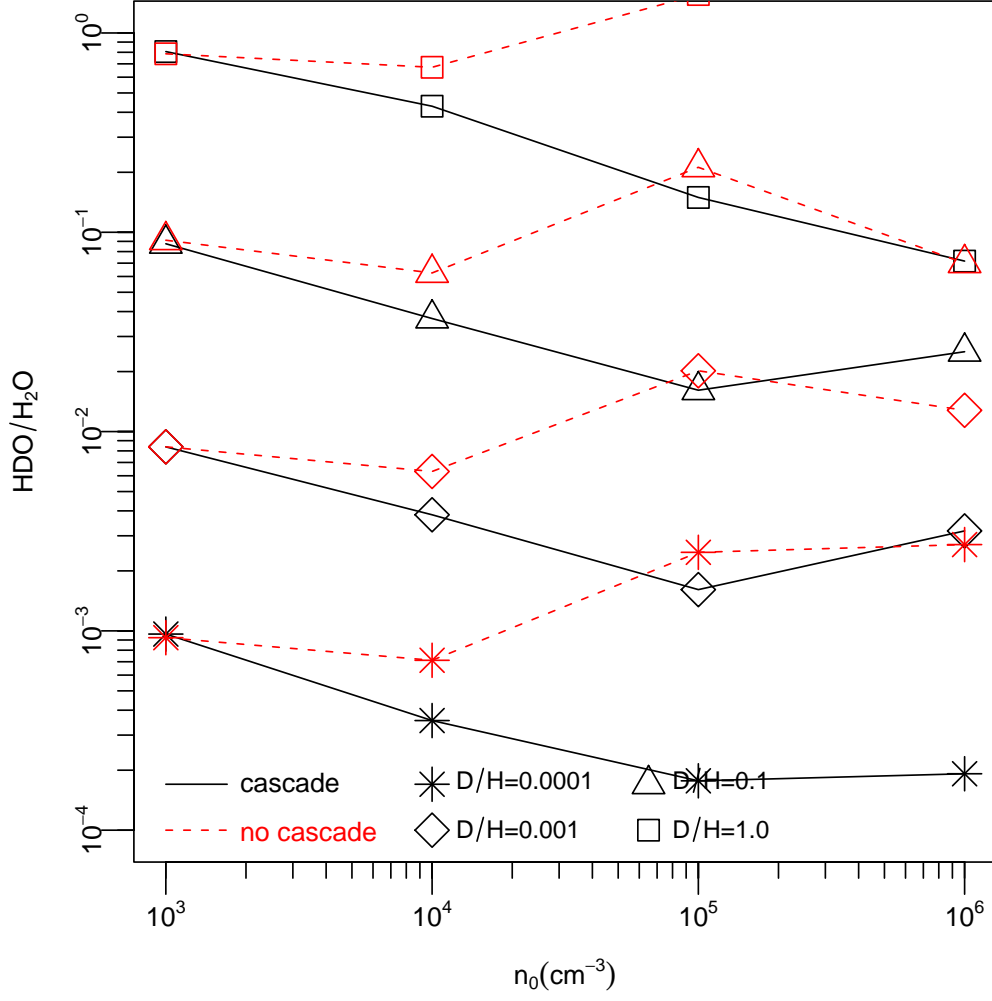


Figure 18: The fraction of  $HDO$  relative to  $H_2O$  as the function of density, for different ratios for the cascade and no cascade case when the  $O_3$  barrier is 450K.

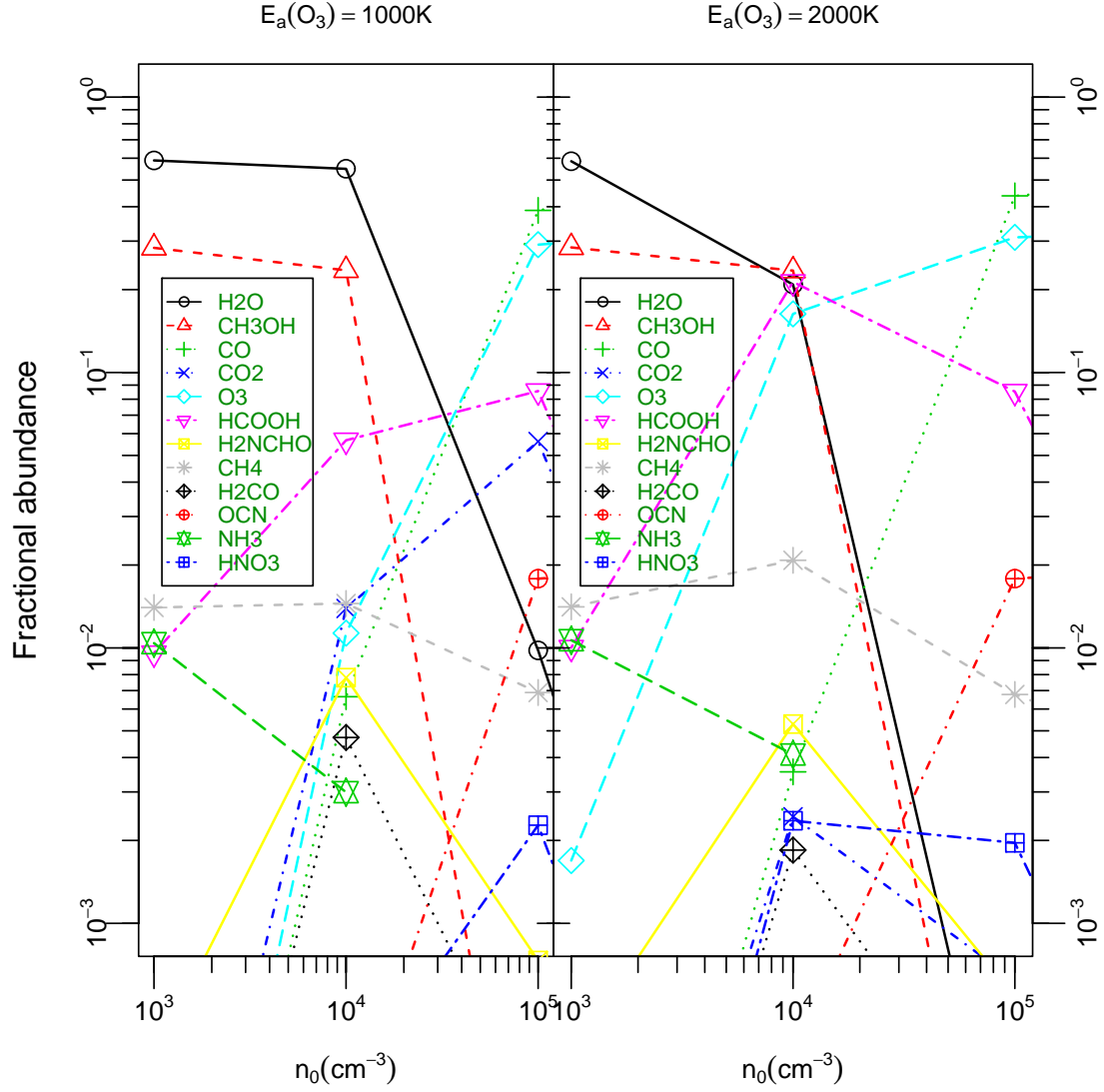


Figure 19: Fractional abundances of the major solid-state H-bearing grain surface species with the cascade and the  $\text{O}_3 + \text{H}$  or  $\text{D}$  activation barrier is increased, as the function of density for  $\text{D}$ -to- $\text{H}$  ratio of 0.01.

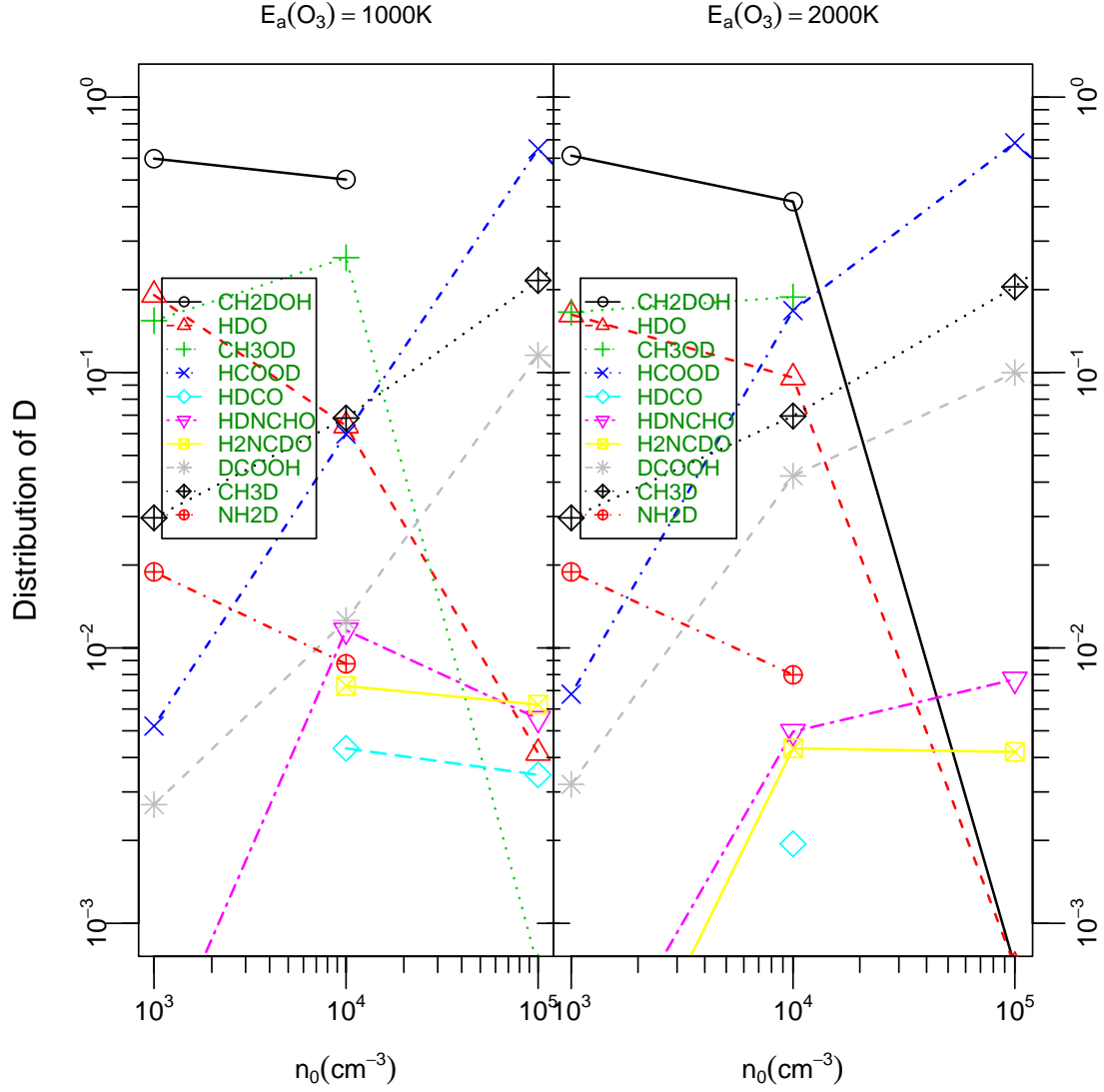


Figure 20: The distribution of deuterium amongst the D-bearing species when the cascade is on and the  $O_3 + H$  or  $D$  activation barrier is increased, as the function of density for  $D$ -to- $H$  ratio of 0.01.

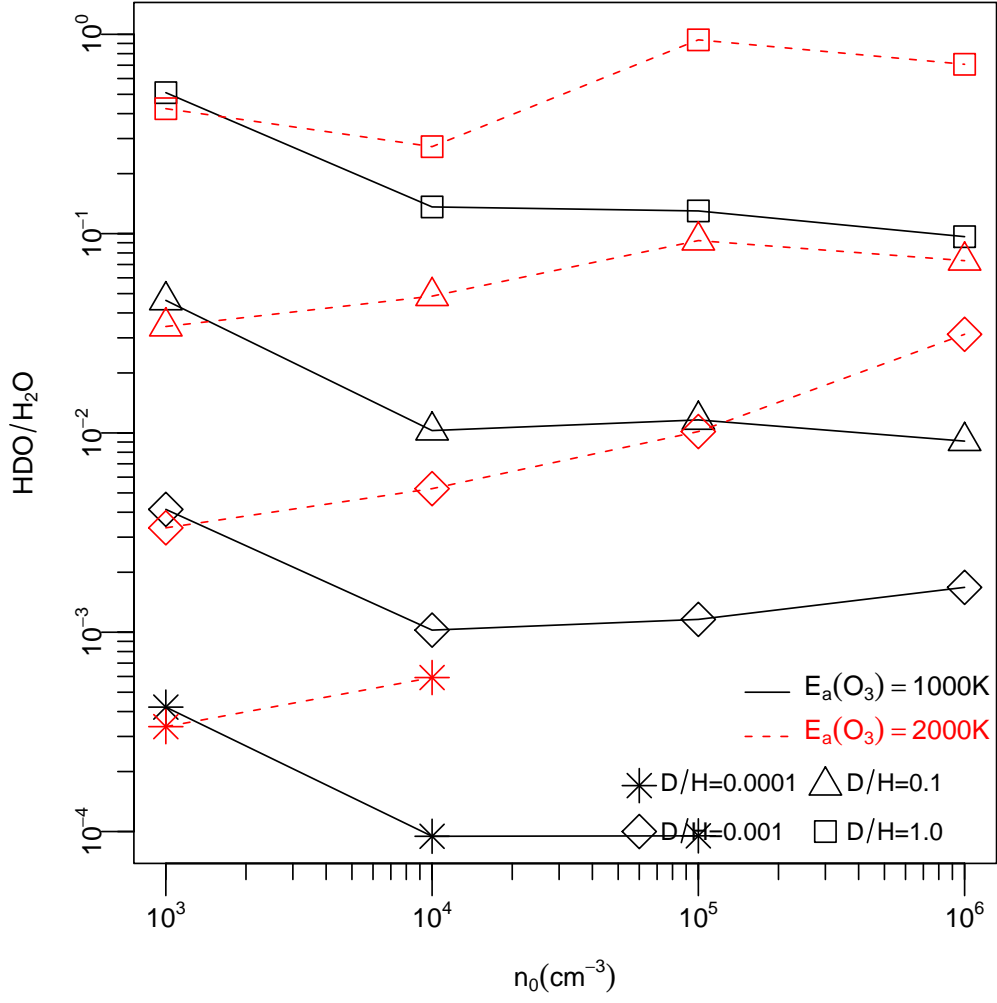


Figure 21: The fraction of  $HDO$  relative to  $H_2O$  versus density for increased  $O_3 + H$  or  $D$  activation barriers, without the  $O_3$  cascade.

## References

- [1] J. Keane & A. G. G. M. Tielens *Modeling grain surface chemistry in dense molecular clouds*. Astronomy & Astrophysics, The European Southern Observatory, 2006 (To be filled manually).
- [2] P. Ehrenfreund, & S.B. Charnley *Organic Molecules in the Interstellar Medium, Comets, and Meteorites: A Voyage from Dark Clouds to the Early Earth*. Annu. Rev. Astron. Astrophys. 38, 427-483.
- [3] D. Hollenbach, & E.E. Salpeter, *H<sub>2</sub> formation on grains*. 1971, ApJ 163, 155.
- [4] L. Snyder, *Origin and Evolution of the Biosphere*. 1997, p115.
- [5] S.B. Charnley, A.G.G.M. Tielens, T.J. Millar, . 1992, ApJ 399, L71.
- [6] A.G.G.M. Tielens & W. Hagen, . 1982, A&A 114, 245.
- [7] M.A. Leitch-Devlin, & D.A. Williams, . 1984, MNRAS 210, 577.
- [8] A.G.G.M. Tielens, & L.J. Allamandola, *Physical Processes in Interstellar Clouds*. 1987, eds. G.E. Morfill & M. Scholer, p.333.
- [9] A.C.A. Boogert, A.G.G.M. Tielens, C. Ceccarelli, A.M.S. Boonman, E.F. van Dishoeck, J.V. Keane, D.C.B. Whittet, Th. de Graauw *THE ENORMOUS ABUNDANCE OF D<sub>2</sub>CO IN IRAS 16293-2422*. A&A, 2000, 359, 1169.
- [10] W. K. Hastings *Monte carlo sampling methods using Markov chains and their application*. Biometrika, vol. 57, pp. 1701-1762, 1994.
- [11] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 6, pp. 721-741, 1984.
- [12] Bertrand Meyer, *Object-Oriented Software Construction*. Englewood Cliffs, NJ.: Prentice-Hall, Inc., 1988.
- [13] Ivar Jacobson, M. Christerson, P. Jonsson & and G. Overgaard, *Object-Oriented Software Engineering*. Reading, Mass., Addison-Wesley, 1992.

- [14] B. Stroustrup, *The C++ Programming Language*. Addison-Wesley Series in Computer Science.
- [15] R Development Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2006. <http://www.R-project.org>
- [16] A. Goldberg, D. Robson, *Smalltalk-80 The language and its implementation*. Addison-Wesley Series in Computer Science
- [17] M. Matsumoto and T. Nishimura, *Dynamic Creation of Pseudorandom Number Generators*. Monte Carlo and Quasi-Monte Carlo Methods 1998, Springer, 2000, pp 56–69.
- [18] M. Matsumoto and T. Nishimura, *Sum-discrepancy test on pseudorandom number generators*. Mathematics and Computers in Simulation, Vol. 62 (2003), pp 431-442.
- [19] David Flanagan, *Java in a Nutshell, Fifth Edition*. O'Reilly, Fifth Edition, March 2005.
- [20] Allan Vermeulen at all., *The Elements of Java Style*. Cambridge University Press (January 2000).
- [21] Joshua Bloch, *Effective Java Programming Language Guide*. Prentice Hall PTR; 1st edition (June 5, 2001).
- [22] Brian Goetz at all, *Java Concurrency in Practice*. Addison-Wesley Professional (May 9, 2006).
- [23] Mersenne Twister Home Page, <http://www.math.sci.hiroshima-u.ac.jp/m-mat/MT/emt.html>
- [24] Home page, Eclipse an open development platform, <http://www.eclipse.org/>
- [25] Home page, Ant a Java build tool, <http://ant.apache.org/>
- [26] Home page, Checkstyle code style development tool, <http://checkstyle.sourceforge.net/>
- [27] Home page, JUnit testing framework, <http://www.junit.org/index.htm>
- [28] Home page, Emma Java code coverage measurement tool, <http://emma.sourceforge.net/>
- [29] Mike Clark, *Pragmatic Project Automation: How to Build, Deploy, and Monitor Java Apps*. The Pragmatic Programmers; 1 edition (August 2004).



- [30] Home page, JDepend development tool, <http://clarkware.com/software/JDepend.html>
- [31] Home page, FindBugs development tool, <http://findbugs.sourceforge.net/>
- [32] Home page, PMD java code analyzer, <http://sourceforge.net/projects/pmd>
- [33] Home page, Hackystat Developer Services, <http://www.hackystat.org/>

## List of Figures

1	The conceptual drawing of a dust grain along with complex molecules formation routes.	5
2	The observational data. . . . .	6
3	Grain surface chemistry processes. . . . .	7
4	The JVM architecture overview. . . . .	13
5	The MVC architecture overview. . . . .	13
6	The sampling flow within the software. . . . .	20
7	The ICLOUDS screenshot showing the “Cloud parameters” panel. . . . .	20
8	The ICLOUDS screenshots of the “Cloud population” panel. . . . .	20
9	The ICLOUDS screenshots of the “Reactions” panels. . . . .	20
10	Results analysis. . . . .	20
11	ICLOUDS project at GoogleCode. . . . .	20
12	ICLOUDS project Wiki engine. . . . .	20
13	ICLOUDS unit test code coverage statistics. . . . .	22
14	ICLOUDS code issues statistics. . . . .	22
15	Illustration of the effect of the $O_3$ cascade. . . . .	23
16	Fractional abundances of the major H-bearing species. . . . .	23
17	The distribution of deuterium amongst the D-bearing species . . . . .	23

18	The fraction of $HDO$ relative to $H_2O$ as the function of density. . . . .	23
19	Fractional abundances of the major H-bearing species, 1K, 2K. . . . .	23
20	The distribution of deuterium amongst the D-bearing species, 1k, 2K. . . . .	23
21	The fraction of $HDO$ relative to $H_2O$ versus density for increased barriers. . . . .	28