Stratospheric Chemistry Kinetics

Last update	March 20, 2019
Supervisor	Dr. Nikos Daskalakis, room S3132, e-mail:daskalakis@uni-bremen.de
Lab	Building NW1, room S3133 (on the map it's S3121)

Contents

1 Introduction

For understanding the atmospheric processes and composition, scientists have utilized different tools and methods to help them study this complex system. Among those tools are atmospheric models. Models have proven to be extremely useful tools for understanding, among others, the atmospheric observations.

There are many different kinds of models available for studying the atmosphere: Chemistry and Transport models, Planetary models, General Circulation Models, Eulerian and Lagrangian models, etc. Among the simplest of the models, are the chemical kinetics box models. These are basically "computational chemistry labs", where we can code the reactions that we think take place, starting concentrations of pollutants, and the model will calculate the quantities at the end of our simulation.

In this experiment you are going to try and simulate the kinetics of the stratospheric chemistry of O_3 using a box model. You will have to create the box model using a tool that automates the process in a great extend. This is the Kinetic Pre-Processor (KPP, Sandu and Sander, 2006).

Using KPP you will have to simulate the basic chemistry of ozone as proposed by Sidney Chapman (Chapman, 1929), and then try to extend it in order to account for the missing sinks and better represent the actual stratospheric chemistry of O_3.

2 Experiment formalities

On the day of the experiment, please report to Dr. Nikos Daskalakis (room S3132) by 09:30. You will have to finish the on-site work by 14:45.

There is a dedicated desktop workstation prepared for your experiment work in room S3133, where you will perform the experiment.

Since our offices are relatively new, there is no room S3132/S3133 on the interactive campus map. You can find us at the same location where the former room S3121 existed.

3 How to prepare for the experiment

3.1 Literature search

Before coming to perform the experiment there are a few things that you should look up. First of all, we are going to try and simulate the stratospheric chemistry of Ozone, as proposed by Sidney Chapman in the 1930's. For this experiment, knowing the reactions in not enough. A preliminary knowledge of the Chapman cycle, its strengths and shortcomings, is mandatory for this experiment.

For you to be able to evaluate your results you should have an idea of what the O_3 concentration in the stratosphere is.

Look at the literature in order to find the basic chemical cycle ozone in the stratosphere as proposed by Sidney Chapman. Figure out what reactions are needed to complete the cycle, and which reactions should be included in the cycle in order to "correct" for the missing sinks of Ozone in the stratosphere.

It is advisable to have at hand a list of all the reactions you believe need to be in the model for each step of the experiment. If you try to find the reactions during the experiment it is most certain that the provided time will not be enough.

3.2 Explanation of terms used in this manual

model	A model is a description of a system using mathematical concepts and language.
box model	A box model of an atmospheric species describes the abundance of said species inside a box representing a selected atmospheric domain.
compile	Compilation of a code is the translation of "human readable" code to "machine language". This is done using a compiler.
compiler	A program that converts instructions (code) to machine language so that it can be read and executed by computers.

4 Background

4.1 Ozone in the atmosphere

Ozone is one of the naturally produced oxidants of the atmosphere playing an important role in atmospheric chemistry and in human health. It is mainly found in two regions in the atmosphere, the troposphere and the stratosphere, having a different role in each region. Figure 1 illustrates the vertical distribution of Ozone in the Earth's atmosphere.

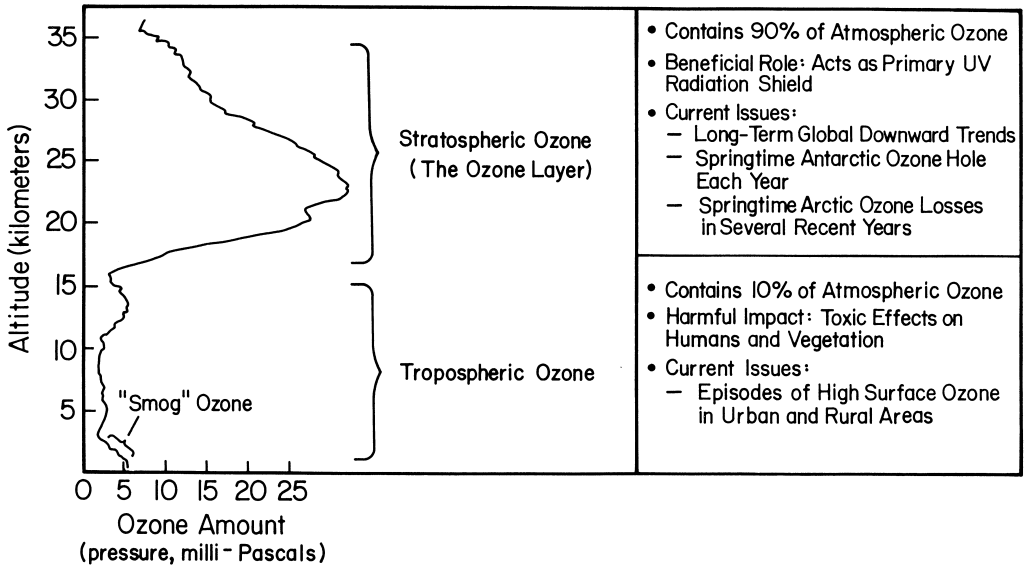

Figure 1: Atmospheric Ozone vertical distribution (image from NOAA)

In the troposphere ozone acts as a greenhouse gas, trapping the outgoing radiation from the Earth's

surface, having a warming effect. Moreover, high concentrations of Ozone can be toxic for living organisms at the earths surface.

In the stratosphere, where most of the atmospheric Ozone resides (about 90% of the total atmospheric Ozone), ozone acts as a "shield" for ultraviolet sunlight, protecting life on the surface from its harmful effects.

4.2 Stratospheric Ozone chemistry

Because of the importance of stratospheric Ozone it has been the subject of studies since scientists discovered its existence in the early 1900s. Shortly after its discovery, British physicist Sydney Chapman proposed a photochemical mechanism that could explain the vast amounts of Ozone in the stratosphere.

The Chapman mechanism, as it is known, consists of the following reactions:

$$
\begin{aligned}
O_2 + hv &\longrightarrow O + O \\
O + O_2 + M &\longrightarrow O_3 + M \\
O_3 + hv &\longrightarrow O + O_2 \\
O + O_3 &\longrightarrow O_2 + O_2 \\
O + O + M &\longrightarrow O_2 + M
\end{aligned}
$$

Models based only on the Chapman mechanism overpredict stratospheric ozone levels, as shown in Figure 2.

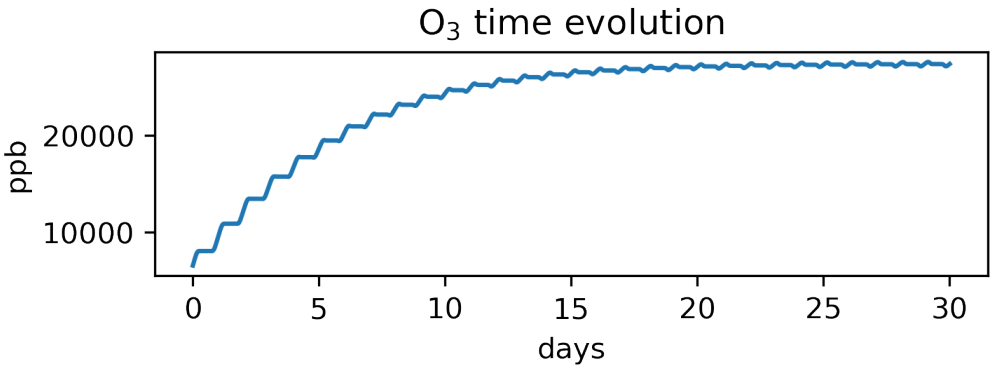

Figure 2: Simulation of stratospheric ozone concentration based on the Chapman mechanism. This simulation was performed using a KPP model.

We now know that there were missing sinks of Ozone from the Chapman mechanism, like the reactions with nitrogen oxides, chlorine, bromine and hydrogen. Models that take into account these reactions correctly predict the stratospheric Ozone concentrations, as shown in Figure 3.

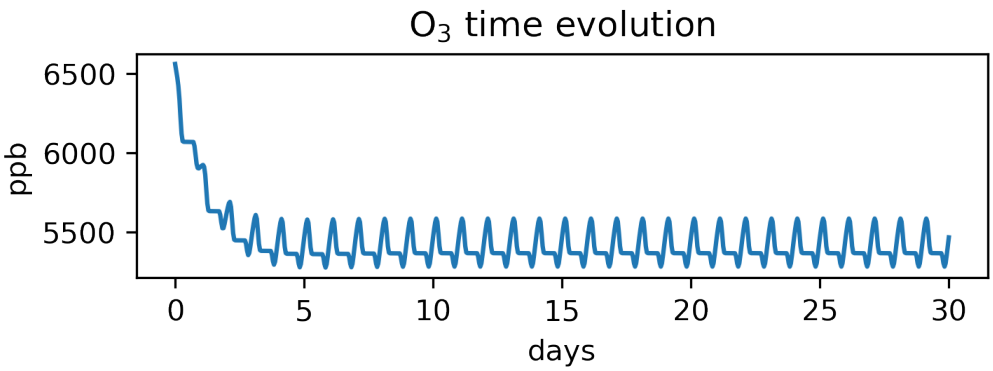

Figure 3: Simulation of stratospheric ozone concentration based on an analytical mechanism. This simulation was performed using a KPP model.

5 Experiment

Based on the information you can find in the appendices of this document (Sections A.2,A.3), you are asked to create a simple box model of the stratospheric chemistry of Ozone at first as proposed by Sidney Chapman.

5.1 Task 1

First prepare all the necessary files needed by KPP to produce the box model (Section A.2). During this task you will need extra information, such as the reaction rates and starting concentrations for the reactions you are going to simulate. These will be provided on site by the supervisor. Make sure to setup the simulation for 30 days, giving this way adequate time for the species to come to equilibrium in the model.

Then compile and run the model (Section A.3).

After successful completion of the model simulation a python notebook will be provided that will help you process the output of the model and create the figures of the temporal evolution of the concentration of Ozone and other species included in your simulation.

5.2 Task 2

At this point you should be able to identify the shortcomings of the Chapman mechanism. You have to find (by searching in the literature) which reactions need to be added in the mechanism to properly represent the stratospheric Ozone concentration. *It is advisable to do the literature search before the experiment. If done during the experiment there will not be adequate time left to finish within the 6 hour period.*

Create a new box model (under a new directory) and start (gradually) adding the new reactions. Add one "family" of reactions at a time, creating a new model every time you add a catalytic cycle.

HINT: *There are four main catalytic cycles that can be added in this step.*

After adding each of the "reaction families", run the model and plot the temporal evolution of Ozone concentrations. Try and comment on the differences you see with each addition.

The goal is to reach as close to what is shown in Figure 3. Have in mind though that this figure was created using a model of more than 100 reactions.

6 What should be on the report

The report for this experiment should consist of the following:

- Introduction of the problem (Do NOT copy what is in this document!)
- Description of the model
- The different chemical mechanisms used in the experiment (reactions) and the results they produced (based on the output of the python notebook that we will provide).
- Discussion on the results and the differences produced by the different mechanisms.
- Conclusions
- Appendix containing all the kpp, eqn, spc and def files you created during this experiment.

Appendix

A KPP

A.1 What is KPP

Quoting from the creator's website:

> *The KPP kinetic preprocessor is a software tool that assists the computer simulation of chemical kinetic systems. The concentrations of a chemical system evolve in time according to the differential law of mass action kinetics. A numerical simulation requires an implementation of the differential laws and a numerical integration in time.*
>
> *KPP translates a specification of the chemical mechanism into Fortran77, Fortran90, C, or Matlab simulation code that implements the concentration time derivative function, its Jacobian, and it Hessian, together with a suitable numerical integration scheme. Sparsity in Jacobian/Hessian is carefully exploited in order to obtain computational efficiency.*

KPP incorporates a library with several widely used atmospheric chemistry mechanisms; the users can add their own chemical mechanisms to the library. KPP also includes a comprehensive suite of stiff numerical integrators. The KPP development environment is designed in a modular fashion and allows for rapid prototyping of new chemical kinetic schemes as well as new numerical integration methods.

Essentially, KPP is a tool that will help us create a box model for calculating the "life" of ozone in the stratosphere.

A.2 Preparing all the inputs KPP needs

Note: All the files described here are **examples** and are not the exact files you will have to create for this experiment!

You can find the official KPP user manual here.

KPP will already be installed on the system you are going to work for this experiment. As described above, KPP is a pre-processor that will create the box model you will use to calculate the chemistry of the stratosphere.

For KPP to create the box model you will need to create a series of input files. Here I will try to explain everything that you need to include to those files and how to use them to create the model.

Note: Make a directory inside which you will create all the files needed. This directory, as well as all the individual files that you will create from now on, have to have the same basename as the model you will create. This means that if you plan to name your model "my_model", then make a directory named "my_model" and inside that directory make all the files needed, named "my_model.kpp", "my_model.spc", "my_model.eqn" e.t.c.

First of all you will need to create a `chapman.kpp` file. This file will contain all the basic information KPP needs to start creating the model.

In this file you have to define the name of the model, the programming language in which the model will be created, the integrator for the chemistry and the driver of the model. For the purpose of the experiment we have already created the driver you will use. Your file should look like this:

```
#MODEL      chapman
#LANGUAGE   Fortran90
#INTEGRATOR rosenbrock
#DRIVER     general
```

The second file you will need to create is the file containing the chemical equations that the model will be asked to solve. This file has to have the same basename as the kpp file, with the `eqn`. In that file you will need to write the reactions that will take place, with the reaction rate coefficients. The format is as follows:

```
#EQUATIONS
// This is a comment line. This will not be interpreted
<R1> reactant + reactant = product  : rate ; {This is an inline comment}
<R2> reactant + hv = product : rate * SUN;
```

Note the `SUN` factor with which we multiply the photochemical reaction rate. This is a factor ranging from 0 to 1 and depends on the time of day. During nighttime it is zero, and during noon it is one.

Note: Before you start the experiment, you should revise the reactions you think should go in the model wiht the supervisor. At this point you will be provided with all the rates you might need for preparing the equation file.

The program already knows all the atoms of the periodic table of elements. But it does not know the molecules of the reactants and pollutants. You will have to create a file describing the species that are used in the reactions. This file again has to have the same basename as the model, this time with the `spc` suffix.

An example of an spc file is:

```
#include atoms

#DEFVAR
O   = O;            { Oxygen atomic ground state }
O1D = O;            { Oxygen atomic excited state }
O3  = O + O + O;    { Ozone }

#DEFFIX
M   = O + O + N + N;{ Atmospheric generic molecule }
O2  = O + O;        { Molecular oxygen }
```

In the first line of the file we include the definitions of all the atoms. Then we specify all the species whose concentrations do change according to the law of mass action kinetics under the *#DEFVAR* definition. Then we specify the species whose concentrations are determined by physical and not chemical factors under the *#DEFFIX*.

Every line ends with the semicolon `(;)`. Everything included in curly brackets is a comment.

The final file you will have to create is the definitions file (suffix `def`). This file will contain several information for the simulation, like starting concentrations for all the species used in the simulation, for which species will the model create a report of their concentrations, the timeframe of the simulation, the timestep etc. An example of this file is:

```
#include chapman.spc
#include chapman.eqn

#LOOKATALL                      {File Output}
#MONITOR O3;O2;                 {Screen Output}

#CHECK O; N;                    {Check Mass Balance}

#INITVALUES                     {Initial Values}
CFACTOR = 1.     ;              {Conversion Factor}
O3  = 5.326E+11 ;
O2  = 1.697E+16 ;

#INLINE F90_INIT
        TSTART = (12*3600)
        TEND = TSTART + (24*3600) * 30
        DT = 3600
        TEMP = 300
#ENDINLINE
```

The file begins with two *#include* statements. Make sure to include here the files you created during the previous steps of preparing the model.

The *#LOOKATALL* statement tells the program to save the concentrations for all the species involved in the simulation.

The *#MONITOR* statement defines the concentrations of the species that will be print on the screen during the simulation. This is useful for monitoring the evolution of species we are interested in.

The *#CHECK* statement tells to the model to always be aware of the mass balance of the species specified there.

After the *#INITVALUES* statement you have to specify the starting concentrations of all the species that are involved in your simulation. You will be provided starting concentrations for the species that you decide to include.

The *#CFACTOR* is a conversion factor for harmonization of the concentration units to the reaction rate units. The supervisor of the experiment will provide you with a proper factor for the provided concentrations.

Then a piece of Fortran90 code is included in this file. Here we specify the starting and ending time of the simulation, as well as the timestep. All the times are in seconds. Also here we specify the temperature (ambient) for the model to use (in Kelvin). Make sure to use a proper temperature for the stratosphere. Also you will have to adjust the timestep according to the reactions you plan to add to the simulation (for monitoring extremely fast radical reaction we need a smaller timestep).

After you have created all the files, the contents of the folder should look like this:

```
daskalakis@lamoslab:~/kpp$ ls
chapman.def  chapman.eqn  chapman.kpp  chapman.spc
daskalakis@lamoslab:~/kpp$
```

A.3 Create and compile the Fortran model

When all the input files for the experiment are ready, you will need to run the KPP tool in order to create the box model. You do that by writing in the command line:

```
daskalakis@lamoslab:~/kpp\$ kpp chapman.kpp
```

If all the files you created are correct, then you should get a report like the following:

```
This is KPP-2.2.3.

KPP is parsing the equation file.
KPP is computing Jacobian sparsity structure.
KPP is starting the code generation.
KPP is initializing the code generation.
KPP is generating the monitor data:
    - chapman_Monitor
KPP is generating the utility data:
    - chapman_Util
KPP is generating the global declarations:
    - chapman_Main
KPP is generating the ODE function:
    - chapman_Function
KPP is generating the ODE Jacobian:
    - chapman_Jacobian
    - chapman_JacobianSP
KPP is generating the linear algebra routines:
    - chapman_LinearAlgebra
KPP is generating the Hessian:
    - chapman_Hessian
    - chapman_HessianSP
KPP is generating the utility functions:
    - chapman_Util
KPP is generating the rate laws:
    - chapman_Rates
KPP is generating the parameters:
    - chapman_Parameters
KPP is generating the global data:
    - chapman_Global
KPP is generating the stoichiometric description files:
    - chapman_Stoichiom
    - chapman_StoichiomSP
KPP is generating the driver from general.f90:
    - chapman_Main
KPP is starting the code post-processing.

KPP has succesfully created the model "chapman".
```

If you now list the contents of the directory you should get something like this:

```
daskalakis@lamoslab:~/kpp$ ls
chapman.def              chapman_JacobianSP.f90     chapman_Monitor.f90
chapman.eqn              chapman.kpp                chapman_Parameters.f90
chapman_Function.f90     chapman_LinearAlgebra.f90  chapman_Precision.f90
chapman_Global.f90       chapman_Main.f90           chapman_Rates.f90
chapman_Hessian.f90      chapman.map                chapman.spc
chapman_HessianSP.f90    chapman_mex_Fun.f90        chapman_Stoichiom.f90
chapman_Initialize.f90   chapman_mex_Hessian.f90    chapman_StoichiomSP.f90
chapman_Integrator.f90   chapman_mex_Jac_SP.f90     chapman_Util.f90
chapman_Jacobian.f90     chapman_Model.f90          Makefile_chapman
daskalakis@lamoslab:~/kpp$
```

Notice all the files that are newly created by the pre-processor. These are all the Fortran90 files your newly created model consists of.

You can now compile your model using `make -fMakefile_chapman`. If no error occur, you should get something like the following:

```
daskalakis@lamoslab:~/kpp$ make -fMakefile_chapman
gfortran -cpp -O  -c chapman_Precision.f90
gfortran -cpp -O  -c chapman_Parameters.f90
gfortran -cpp -O  -c chapman_Global.f90
gfortran -cpp -O  -c chapman_Function.f90
gfortran -cpp -O  -c chapman_JacobianSP.f90
gfortran -cpp -O  -c chapman_Jacobian.f90
gfortran -cpp -O  -c chapman_HessianSP.f90
gfortran -cpp -O  -c chapman_Hessian.f90
gfortran -cpp -O  -c chapman_StoichiomSP.f90
gfortran -cpp -O  -c chapman_Stoichiom.f90
gfortran -cpp -O  -c chapman_Rates.f90
gfortran -cpp -O  -c chapman_Monitor.f90
gfortran -cpp -O  -c chapman_Util.f90
gfortran -cpp -O  -c chapman_LinearAlgebra.f90
gfortran -cpp -O  -c chapman_Initialize.f90
gfortran -cpp -O  -c chapman_Integrator.f90
gfortran -cpp -O  -c chapman_Model.f90
gfortran -cpp -O  -c chapman_Main.f90
gfortran -cpp -O  chapman_Precision.o chapman_Parameters.o chapman_Global.o
chapman_Function.o chapman_JacobianSP.o chapman_Jacobian.o chapman_HessianSP.o
chapman_Hessian.o chapman_StoichiomSP.o chapman_Stoichiom.o chapman_Rates.o
chapman_Util.o chapman_Monitor.o chapman_LinearAlgebra.o chapman_Main.o
chapman_Initialize.o chapman_Integrator.o chapman_Model.o -o chapman.exe
daskalakis@lamoslab:~/kpp$
```

After compiling you have to run the model. To do that you type in the command line:

```
daskalakis@lamoslab:~/kpp\$ ./chapman.exe
```

While the model runs you should get on your screen the concentrations calculated in each timestep for each of the species you specified in the *#MONITOR* command in your `def` file.

B Technical knowledge

This experiment will be performed mainly in the Linux terminal. You should get yourselves acquainted with the terminal interface and the basic commands one needs to navigate the environment. Also, knowledge of a terminal text editor will be useful (e.g. vi, pico/nano).

B.1 Linux operating system

Linux is a family of free and open-source operating systems based on the Linux kernel. It provides both GUI (Graphical User Interface) and a console. For this lesson you will be using both the GUI and the console.

The GUI is more or less intuitive for its use, and the basic usage is similar to what other operating systems offer, with windows, menus etc. Here we will give you a small tour in the Linux terminal (console), as most users are not familiar with it.

B.2 Linux basic file structure

Linux has a tree-like file structure. The `ROOT` directory (`/`) looks like this:

```
daskalakis@lamosnb2 ~$ tree / -L 1
/
├── bin
├── boot
├── dev
├── etc
├── home
├── init
├── lib
├── lib64
├── media
├── mnt
├── opt
├── proc
├── root
├── run
├── sbin
├── srv
├── sys
├── tmp
├── usr
└── var

19 directories, 1 file
daskalakis@lamosnb2 ~$
```

As you can see, everything is located under `ROOT (/)` directory. Your personal space will be located under the `/home/<class>/<username>` directory. This is where all your personal files and folders will be, and also where you will perform the experiment.

B.3 Basic Linux commands

The terminal is a basic application in all Linux systems. You will be using a virtual terminal with a Linux shell for the purpose of this course. A shell is a command-line interpreter that provides a command line user interface for the operating system. There are many different shells that a user can choose from, with one of the most popular ones being the `BASH shell`.

Through the terminal the user communicates with the system through commands. The basic commands you will need to get started with the terminal are the following:

cd	[**c**hange **d**irectory] Used to navigate in the Linux filesystem
cal	[**cal**endar] shows current month's calendar
cal *year*	shows the calendar of the current year
cat	[con**cat**enate] prints on the screen the contents of a file. E.g. `cat filename` will print on the screen the contents of file `filename`
cat -v	also show the non printable characters in the file
chgrp	[**ch**ange **gr**ou**p**] changes the group of ownership of the file.
chown	[**ch**ange **own**er] changes the owner of the file.
chmod	[**ch**ange **mod**e] changes the permissions of the file.
cmp	[**c**o**mp**are] compares two files and shows the location of the first difference it finds.
cp	[**c**o**p**y] used to copy a file
cp -r	used to copy a folder
mv	[**m**o**v**e] used to move or rename a file or folder
find	looks in the disk for files that fulfill the criteria we provide
grep	[**g**lobal **r**egular **e**xpression **p**rint] looks in the files for a string
logname	prints on the screen the username of the current user
ls	[**l**i**s**t] lists all the contents of the current folder
ls -a	[**l**i**s**t **a**ll] lists all the contents of the current, including the hidden files and folders.
ls -l	[**l**i**s**t **l**ong list] lists all the contents of the current folder with information about ownership, permissions, size, etc
ls -la	combination of the previous two
passwd	changes the password of the current user
ps	[**p**roce**s**s] prints on the screen the running processes
pwd	[**p**rint **w**orking **d**irectory] prints on the screen the path to the current directory.
rm	[**r**emove] deletes a file
rm -r	deletes a file or folder
tail	shows on screen the last lines of a file
tail -f	shows on screen the last lines of a file, tracking the changes
head	prints on the screen the first lines of a file (header)
logout	self explanatory
clear	clears the screen from all previous commands and/or messages

man [**man**ual] shows the manual page of a command
mkdir [**m**a**k**e **dir**ectory] creates a folder

Of course this list is far from complete, but it should suffice for the introducing you to the Linux terminal.

B.4 Basic VI usage

As mentioned before, knowing how to use a text editor in the terminal can be really useful. Here I will try and give you an introduction to the vi editor.

vi is a modal editor. This means that it has several "modes". There are two primary modes: *insert mode* where you type text into the editor and it is committed to the document, and *normal mode* where you enter arguments via the keyboard that perform a variety of functions.

In order to start editing a file (new or existing) in vi you just have to write in the command line:

```
daskalakis@lamosnb2 ~\$ vi filename
```

vi always starts in normal mode. This means that you cannot start typing into your file right away.

Following are some of the commands vi can accept when in normal mode:

Cursor movement commands:

h move one character left
j move one character right
k move one line down
l move one line up
[Enter] move to the beginning of the next line
$ move to the last column of the current line
0 move to the first column of the current line
^ move to the first non-empty column of the current line
w move to the beginning of the next word or punctuation mark
W move to the column after the next empty character
b move to the beginning of the previous word
B move to the beginning of the previous word ignoring punctuation marks
e move to the end of the next word
E move to the end of the next word ignoring punctuation marks
H move to the top of the screen
M move to the middle of the screen
L move to the bottom of the screen

Screen movement commands:

G	move to the last line of the document
#G	move to line number #
z+	move so that the line currently at the bottom of the screen goes to the top of the screen
z	move current line to the middle of the screen
z-	move current line to the bottom of the screen
Ctrl-f	move one screen down
Ctrl-b	move one screen up
Ctrl-d	move half a screen down
Ctrl-u	move half a screen up

Text insert commands:

All the following commands allow to enter *insert mode* and edit the document. When in insert mode, at the bottom left of the screen you can see the `-- INSERT --` indication, which means you are editing the document. To return to the *normal mode* hit the `Esc` key.

r	replace the character under the cursor with the next character you hit on the keyboard
i	start inserting text where the cursor is
a	insert text starting from the next column after the cursor
o	insert a new line below the current line and start inserting text at the beginning of that line
R	replace text until the `Esc` key is pressed
I	insert text starting from the first non-blank column of the line. If there is no text in the line, insert at the beginning of the line
A	insert text starting at the end of the current line
O	insert a new line above the current line and insert at the beginning of that line

Commands for deleting text:

x	delete the current character
dd	delete the current line
dw	delete the current word, starting from the position of the cursor
db	delete from the position of the cursor until the beginning of the current word
D	delete from the cursor up to the end of the current line

All the above commands save whatever was last deleted in a buffer, and can be used (pasted) again in the document.

Commands for copying:

yy	copy the current line

Commands for pasting:

p	paste after the cursor
P	paste before the cursor

Search commands:

/	search for text forward (case sensitive)
?	search for text backwards (case sensitive)
f	search for a character forward in the current line
F	search for a character backwards in the current line

Other commands:

.	redo the last command
u	undo
ctrl-r	redo (after undo)
ZZ	save and quit
ZQ	discard changes and quit

When in *normal mode*, you can also use a variety of commands after hitting colon `(:)`. Some of these commands are:

:w	save
:x	save and quit
:wq	save and quit
:q	quit
:q!	quit discarding changes
:#	go to line number #
:$	go to the end of the document

References

Chapman, S. (1929). A theory of upper atmospheric ozone. *Memoirs of the Royal Meteorological Society*, 3,(26):103–125.

Sandu, A. and Sander, R. (2006). Technical note: Simulating chemical systems in fortran90 and matlab with the kinetic preprocessor kpp-2.1. *Atmospheric Chemistry and Physics*, 6(1):187–195.

B.5 Basic PICO usage

PICO or NANO is a very simple, very basic text editor for use in the Linux terminal. It is the equivalent of notepad for Windows. It offers no syntax highlighting or high level editing, but can be

used for easy and fast modification of files.

For opening a file with pico you type:

```
pico <filename>
```

The application will open and you can directly start modifying the contents of the file. At the bottom of the screen you can see the options available. For exiting the application you type `Ctrl-X`. For saving you type `Ctrl-O`. It offers a basic search function with `Ctrl-W`. For saving and exiting you type `Ctrl-OX`.

If you find yourselves struggling with vi, you can use pico for its ease of use.