RADTRANS - USER GUIDE

A. T. HANNINGTON[1], A. P. WHITWORTH[1]

15/11/2019

CONTENTS

1 Preface 2
2 Rad Trans 2
2.1 Structure . . . 2
2.2 Usage . . . 2
3 Python Plotting Packages 3
4 Unit Tests 3
4.1 Usage . . . 4
5 Known Bugs 4
5.1 Rad Trans . . . 4
5.2 Python Plotting Packages . . . 4
5.3 Unit Tests . . . 5

LIST OF FIGURES

LIST OF TABLES

[1] *School of Physics and Astronomy, Cardiff University, Queen's Building, The Parade, Cardiff, Wales, UK, CF24 3AA*

1 PREFACE

`Rad Trans` is a Monte Carlo Radiative Transfer (MCRT) program, designed and created by Prof. A. P. Whitworth, with subsequent development lead by A. T. Hannington. The program was developed to tackle the problem of MCRT applied to Filamentary Molecular Clouds, and to perform these calculations as quickly as possible.

The current version of `Rad Trans` is 1D, with only radial dependence being utilised in an x-y plane. The current specified geometry is that of a 1D cylindrically symmetric filament of infinite length.

This short document is meant to act as a brief user guide for `Rad Trans`. Please refer to Whitworth's `Rad Trans` documentation for a discussion of the underlying physics.

It should be noted that in its current form, the developed subroutines perform test cases only, and currently dose not perform a general MCRT calculation. However, the `Detailed Balance` subroutine largely follows the structure that would be require for a general MCRT calculation, and it should be possible to generalise from there.

2 RAD TRANS

2.1 Structure

`Rad Trans`, in its current version, is split between three main `FORTRAN 90` files (extension ".f90"). These are `RadTrans_Maincode.f90`, `RadTrans_Subroutines.f90`, and `RadTrans_Constants.f90`. As indicated by the naming convention, these three files contain the main program, all other subroutines, and code and physical constants, respectively.

`RadTrans_Maincode.f90` currently has the four main integrated (numerical) tests (i.e. full numerics tests) coded in after the compulsory preamble. That is to say, the test subroutines are the final calls made within this subroutine, and the preamble would be necessary for any MCRT tests with `Rad Trans`.

`RadTrans_Subroutines.f90` has been split into Unit Tested (see Section 4) and non-Unit Tested subroutines. Where possible, we have refactored the subroutines in this file to be as general as possible extracting features such as Ray Tracing, which are geometry specific, and creating separate subroutines. This allows easier adaptation of the tests to a general MCRT calculation with a differing setup.

`RadTrans_Constants.f90` is split into `CONSTANTS` and `PHYSICAL_CONSTANTS`. As these names imply, the former contains all of the `Rad Trans` specific constants and on/off "switches" for various functionalities, and the latter contains physical constants such as the speed of light, pi, etc. .

2.2 Usage

`Rad Trans` can be compiled and run in two simple ways. The recommended route is through the use of the accompanying Makefile (dependent on GNU's `GFORTRAN`), as follows.

To compile and run normally, enter into the terminal:

1. make clean

2. make
3. ./run

To compile and run with debug flags, enter into the terminal:

1. make clean
2. make debug
3. ./run

To clean up and delete intermediary files and the executable "./run":

1. make clean

It is, of course, possible to use `Rad Trans` using a direct compile and run method. For example, using `GFORTRAN`:

1. gfortran -03 -o run RadTrans_MainCode.f90 RadTrans_Subroutines.f90 RadTrans_Constants.f90
2. ./run

3 PYTHON PLOTTING PACKAGES

All of the plotting packages which accompany `Rad Trans` have been developed in a `Python 3.x` environment. To utilise them, first compile and run `Rad Trans` (see Section 2.2) with **all** option flags set to "1", and utilising the `Detailed Balance` test. Then run the Python plotting packages as normal, for example by using:

1. python3 PlotCellTemperatures.py

4 UNIT TESTS

The purpose of Unit Testing is to provide an industry-standard set of test cases that ensure that `Rad Trans` is operating as intended. Unit Tests should be performed on each separate subroutine and test that, given the fiducial case/simplest case, the subroutine behaves as expected. New test cases should be developed in parallel with new subroutines, to ensure the final results can be trusted.

As previously mentioned, `Rad Trans` is comprised of four (numerical) integrated tests. These tests are **not** Unit Tested — we could not develop Unit Tests that would reliably test the results of these integrated tests beyond what is already performed within `Rad Trans`.

4.1 Usage

The Unit Tests for `Rad Trans` are written in `Python 3.x`, and thus it is necessary to convert the `FORTRAN90` code into an format able to be interpreted by Python. As such, we use the `NumPy` package `F2Py` to perform this conversion. We then perform the unit tests using `PyTest`.

We have produced a standard compilation shell script for this process, which performs the two compilation commands necessary. This script is called `f2py_RadTrans.sh`, and can be executed (in Linux) using:

1. bash f2py_RadTrans.sh

We have also produced a standardised script for running the unit tests, called `pytest_unitTest.sh`, which can run using:

1. bash pytest_unitTest.sh

Please note, that `PyTest` is not installed as standard with `Python 3.x` and thus will need to be installed for use within your environment.

5 KNOWN BUGS

5.1 Rad Trans

1. Constant optical depth system producing number of interactions not consistent with optical depth set. I believe this to be an issue with the average chi value. Analytic forms of the equations to find this value have been given, but not implemented.

5.2 Python Plotting Packages

5.2.1 *Plot Abs(orption) Emiss(ion) Probabilities .py*

1. Not generalised to centering of vlines on different lambda value. Some tweaking of "FUDGE" factor will be needed, and possibly further corrections.

5.2.2 *Plot Cell Temperatures .py*

1. NONE

5.2.3 *Plot Density .py*

1. NONE

5.2.4 *Plot Dust Properties .py*

1. Blackbody plots need to be normalised!

5.2.5 *Plot NMSV_DB-test .py*

1. These plots are not linear in some cases...

5.2.6 *Plot Probabilities .py*

1. NONE

5.2.7 *Plot Timings .py*

1. Fit doesn't seem correct. Intercept non-zero and large linear coeff. .

5.3 Unit Tests

1. Tolerance of the Unit Tests on Schuster Densities is perculiarly high (approx. 5%). I feel this is likely due to a mismatch in density between the two programs: this program evaluates the density at the boundary, RadTrans_*.f90 evaluates the density at each boundary and takes an average for the cell itself.

REFERENCES