UNIVERSITY OF CENTRAL FLORIDA

BARTTest Users Manual

A TEST SUITE FOR RADIATIVE-TRANSFER CODES

Author:
Michael D. HIMES

Supervisor:
Dr. Joseph HARRINGTON

June 18, 2018

Contents

1 Team Members

- Michael D. Himes, University of Central Florida (mhimes@knights.ucf.edu).
- Joseph Harrington, University of Central Florida (jh@physics.ucf.edu).
- Patricio Cubillos, Space Research Institute, Austrian Academy of Sciences (patricio.cubillos@oeaw.ac.at)
- Jasmina Blecic, New York University Abu Dhabi (jasmina@nyu.edu).
- Ryan Challener, University of Central Florida (rchallen@knights.ucf.edu).

2 Introduction

This document describes BARTTest, a collection of tests for radiative-transfer (RT) codes. The tests include simple forward models with analytically-verifiable answers, realistic forward models for an HD 189733b-like planet with various types of temperature profiles, and retrievals on synthetic and real data. The simple models test the presence, shape, and strength of lines and other basic aspects of RT codes. The realistic forward models and retrievals provide results that can be compared with other RT codes.

BARTTest is configured by default to test BART, the Bayesian Atmospheric Radiative Transfer code. The configuration and input files are provided as reference for users to see the exact setup for each test so that it can be reproduced for other RT codes. This manual provides a description of each test as well as each individual file so that the user can easily locate any desired input file.

This manual describes the BARTTest program usage for the user (Sections 3 through 5) and potential contributors (Section 6). Section 3 indicates the system requirements and how to obtain the code. It also details the commands to run the program. Section 4 describes the directories and files contained in BARTTest. Section 5 shows how to execute BARTTest from start to finish. Section 6 details how a user would contribute the results of their RT code to BARTTest.

2.1 BARTTest Overview

BARTTest is organized as follows:

```
BARTTest
├── code-output
│   ├── 01BART
│   └── <other codes>
├── doc
├── lib
├── results
│   ├── 01BART
│   ├── <other codes>
│   └── plots
└── tests
    ├── 00inputs
    │   ├── filters
    │   ├── par
    │   └── TLI
    ├── f01oneline
    ├── f02fewline
    ├── f03multiline
    ├── f04broadening
    ├── f05abundance
    │   └── atm
    ├── f06blending
    ├── f07multicia
    ├── f08isothermal
    ├── f09comparison
    ├── r01isothermal
    ├── r02noninverted
    ├── r03inverted
    └── r04WASP12b
```

Each directory and the files within will be discussed in detail in Section 4.

2.2 License

3 Setup

3.1 System Requirements

The requirements to run BARTTest are dependent on what tests the user wishes to execute. The isothermal, comparison, and retrieval tests all have greater requirements for storage and RAM.

Requirements (excluding isothermal, comparison, and retrieval tests):

-

Requirements (all tests):

-

3.2 Execution Methods

The user has a number of ways to run the tests depending on where they are at in the process. The general process is as follows when running BARTTest for BART:

- Pull BART from Github and compile
- Pull line lists (note: necessary for isothermal, comparison, and retrieval tests)
- Run the tests
- Produce plots of the results

The program is controlled by a Makefile in the top-level directory; all commands to execute BARTTest thus take the form of make <command>, executed from the top-level directory. The configured commands are listed in Table 3.2.

These are the built-in methods of executing BARTTest. If the user wishes to run the package in some other manner, it is up to them to configure it as such.

4 Descriptions of Directories and Files

This section describes the files and subdirectories in BARTTest.

4.1 Top-level Files

The BARTTest directory contains 3 files: LICENSE, Makefile, and README.

- LICENSE is identical to Section 2.2. It contains the usage license.
- Makefile is the main driver of the program; it handles commands used to execute tests, download line lists, and so on. See Section 3.2 for more information.
- README is a very brief version of this manual.

Table 1: Makefile Commands

Command	Description
bart	Downloads BART from Github to BARTTest/../BART/ and compiles.
hitran_linelists	Downloads necessary line lists from HITRAN for the isothermal and comparison tests.
oneline	Executes the oneline test (see Section 4.6.2).
fewline	Executes the fewline test (see Section 4.6.3).
multiline	Executes the multiline test (see Section 4.6.4).
broadening	Executes the broadening test (see Section 4.6.5).
abundance	Executes the abundance test (see Section 4.6.6).
blending	Executes the blending test (see Section 4.6.7).
multicia	Executes the multicia test (see Section 4.6.8).
isothermal	Executes the isothermal test (see Section 4.6.9).
comparison	Runs all of the comparison test cases (see Section 4.6.10). Note: this takes a long time.
comparison_tli	Generates the TLI file used for the comparison test. Note: this takes a long time.
comparison_iso	Runs the isothermal emission and transmission comparison cases.
comparison_noinv	Runs the noninverted emission and transmission comparison cases.
comparison_inv	Runs the inverted emission and transmission comparison cases.
plots	Produces plots of the code output.
retrievals	Runs all of the retrievals (see Section 4.6.11). Note: this takes a long time.
retrieval_iso	Runs the synthetic retrievals for an isothermal atmosphere.
retrieval_noinv	Runs the synthetic retrievals for a noninverted atmosphere.
retrieval_inv	Runs the synthetic retrievals for an inverted atmosphere.
retrieval_WASP12b	Runs a retrieval using WASP-12b secondary eclipse data.
tests_noiso	Runs all of the forward modeling tests except isothermal and comparison.
tests_plusiso	Runs all of the forward modeling tests except comparison.
some	Combination of the 'bart' and 'tests_noiso' commands.
all	Downloads BART, downloads line lists, runs every test. Note: this takes a long time.
clean	Deletes all of the code output and plots produced by BART.

4.2 code-output

The code-output directory contains the data files produced by executing each test. Each subdirectory is associated with the output from a particular RT code, and these are numbered according to the time that the output was received. For example, there is a directory "01BART" which contains BART's output for each of the tests.

4.3 doc

The doc directory contains documentation related to BARTTest. The main file of note is barttest-user-manual.pdf, which you are currently reading.

4.4 lib

The lib directory contains various functions and scripts used in the execution of BARTTest.

Files in the directory:

- abuncomp.py: Contains a function used by comparison.py to compute the factor difference of line intensity between spectra.
- comparison.py: Script used to compute the factor difference of line intensity between spectra. Used in the abundance test (see Section 4.6.6).
- makeplots.py: Contains functions used to plot the results of various tests. Produces plots of results.
- opacityconv.py: Contains a function to read a (binary) Transit opacity file and convert it to Numpy arrays.
- voigtcomp.py: Contains a function to compare the line profile from the broadening test with the theoretical Voigt profile corresponding to the setup (temperature, pressure, molecular properties, layer abundances). Used in the broadening test (see Section 4.6.5).
- voigt.py: Contains functions to compute a Gaussian, Lorentzian, and Voigt profile. Used by voigtcomp.py to compute the theoretical Voigt profile.

4.5 results

The results directory contains the plots of the output of each test. It is similar in structure to code-output: each subdirectory is associated with the plots of the output of a particular RT code, and these are numbered according to the time that the output was received.

The plots subdirectory contains plots of the output of multiple codes for the comparison and retrieval tests. Subdirectories within the plots subdirectory are named according to the codes that are being compared and what is being compared ('comparison-iso' or 'retrieval-inv'). For example, a comparison of 'BART', 'code', and 'othercode' for the comparison test would be located at BARTTest/results/plots/BART_code_othercode_comparison-iso/

4.6 tests

The tests/ directory contains the setups for each test. The subdirectory 00inputs/ contains the inputs used for tests. The remaining subdirectories are the tests; subdirectories named f##name are forward model tests, while those named r##name are retrieval tests. Each subdirectory is detailed below. Note that paths are given with respect to BARTTest/tests/.

The forward model tests are configured for Transit, the radiative transfer package of BART. The retrieval tests are configured for BART. These input and configuration files are provided so that the user may reference these for clarity on the exact setup for each test; the following sections will describe each file in each subdirectory.

In general, each forward model test will have a Transit configuration file (i.e., broadening_emission.trc), a Pylineread configuration file (i.e., broadening.plc), and an atmospheric model (i.e., broadening.atm). Some tests have multiple configuration files or atmospheric models. For the retrieval tests, each directory will have a BART configuration file (i.e., iso_emission.brt), a Pylineread configuration file (i.e., iso_emission.plc), and two text files containing the eclipse or transit depths and uncertainties (i.e., iso_ecldepths.txt and iso_ecluncs.txt).

4.6.1 00inputs

00inputs contains the input files used for Transit (the radiative transfer package of BART) to execute the tests.

New (fake) molecules are defined to be used in the tests:

- CG1, similar to H2
- CG2, similar to He
- CG3, similar to N2
- LG1, similar to H2O
- LG2, similar to CH4
- LG3, similar to CO
- LG4, similar to CO2

Note that due to the similarity between these molecules, the user may use the real molecule in place of the fake one as far as calculations are concerned (mass, isotopes, isotope ratios, etc.). However it is important to note that the line lists for these fake molecules do not match the real line lists of the real molecules.

Files in 00inputs/:

- CIA_HITRAN_H2He_0200-9900K_0.5-500um.dat: An opacity table for collision-induced absorptions between H2 and He molecules.
- litran.dat: A modified version of the HITRAN definitions file used by the pylineread module (reads line lists to build a file of line transitions) of Transit. Defines the fake molecules listed above.
- molecules.dat: A modified version of the molecule definitions file used by Transit. Defines the fake molecules listed above.

Subdirectories in 00inputs/:

- filters/: Contains filters used for the retrieval tests.
- par/: Contains line lists for the fake molecules and files used to pull linelists from the HITRAN website.
- TLI/: Contains the Transition Line Information files that Transit uses as input. It is initially empty; running the tests will populate the directory as they are created for each test.

4.6.2 f01oneline

This test uses a model atmosphere that has one layer of 100% LG1, and all remaining layers 85% CG1 and 15% CG2. LG1 has only one line at 2.29 um. The test is executed in eclipse geometry.

Files in f01oneline/:

- oneline.atm: Atmospheric model used by Transit.
- oneline.plc: Configuration file for pylineread. Creates line list input file for Transit.
- oneline_emission.trc: Configuration file for Transit. Produces emission spectrum.

4.6.3 f02fewline

This test uses a model atmosphere that has one layer of 100% LG2, and all remaining layers 85% CG1 and 15% CG2. LG2 has three lines: 2.15 um, 2.29 um, 3.20 um. The test is executed in both eclipse and transit geometries.

Files in f02fewline/:

- fewline.atm: Atmospheric model used by Transit.
- fewline.plc: Configuration file for pylineread. Creates line list input file for Transit.
- fewline_emission.trc: Configuration file for Transit. Produces emission spectrum.
- fewline_transmission.trc: Configuration file for Transit. Produces transmission spectrum.

4.6.4 f03multiline

This test uses a model atmosphere that has one layer of 100% LG2, one layer of 100% LG3, one layer of 100% LG4, and all remaining layers 85% CG1 and 15% CG2. LG2 has lines as in f02fewline; LG3 has lines at 2.38 um, 2.50 um, and 2.54 um; LG4 has lines at 2.86 um, 3.02 um, and 3.78 um. The test is executed in both eclipse and transit geometries.

Files in f03multiline/:

- multiline.atm: Atmospheric model used by Transit.
- multiline.plc: Configuration file for pylineread. Creates line list input file for Transit.
- multiline_emission.trc: Configuration file for Transit. Produces emission spectrum.
- multiline_transmission.trc: Configuration file for Transit. Produces transmission spectrum.

4.6.5 f04broadening

This test looks at the accuracy of the Voigt profile for one line. One layer of the model atmosphere has 0.01% LG1 and 99.99% CG3, and all remaining layers 100% CG3. The layer with LG1 is at a pressure of 0.33516 bars and a temperature of 1442.58 K. The analytical Voigt profile is computed for this situation by calculating the half width at half max (HWHM) of a Gaussian due to thermal broadening and the HWHM of a Lorentzian due to pressure broadening. These are used as parameters for the Faddeeva function; the real part of this function is equal to a Voigt profile. This is compared to the opacity spectrum of the line as computed by Transit.

Files in f04broadening/:

- broadening.atm: Atmospheric model used by Transit.
- broadening.plc: Configuration file for pylineread. Creates line list input file for Transit.
- broadening_emission.trc: Configuration file for Transit. Produces emission spectrum.

4.6.6 f05abundance

This test explores the effect of varying the abundance of a molecule in an optically-thin situation. If the atmosphere is optically thin and the line is weak, then double the abundance should result in a line that is twice as intense.

This test uses a uniform atmosphere composed of x% LG1 and 100-x% CG1 at all layers. X ranges from 0.01% to 0.1% in steps of 0.01%. All 11 cases are executed in eclipse geometry.

Files in f05abundance/:

- abundance_line.plc: Configuration file for pylineread. Creates line list input file for Transit.
- abundance_noline.plc: Configuration file for pylineread. Creates line list input file for Transit. Line is removed for this case.
- abundance_*_emission.trc: Configuration files for Transit. * ranges from 1e-4 to 1e-3 in steps of 1e-4.
- abundance_noline_emission.trc: Configuration file for Transit. The line is "removed" (made very, very weak) for this case.
- README: Brief readme about the test.

Subdirectories in f05abundance/:

- atm/: Directory containing the atmospheric models. Naming convention is, i.e. 1e-4_uniform.atm

4.6.7 f06blending

This test uses two lines in close proximity to look at the result of two lines blending together. There is a LG1 line at 2.28919 um and a LG2 line at 2.28921 um. LG1 and LG2 only exist at one layer of the atmospheric model.

Files in f06blending/:

- blending.atm: Atmospheric model used by Transit.
- blending.plc: Configuration file for pylineread. Creates line list input file for Transit.
- blending_emission.trc: Configuration file for Transit. Produces emission spectrum.

4.6.8 f07multicia

Transit incorporates collision-induced absorptions (CIAs) via a text file opacity table in a certain format. Other codes may also handle CIAs in a similar manner. This test uses a atmosphere uniformly composed of 85% H_2 and 15% He at all altitudes. The forward model is generated for three cases: no CIAs, H_2-He CIAs, and H_2-H_2 and H_2-He CIAs.

Files in f07multicia/:

- multicia.atm: Atmospheric model used by Transit.
- multicia.plc: Configuration file for pylineread. Creates line list input file for Transit.
- noCIA_emission.trc: Configuration file for Transit. Produces emission spectrum.
- oneCIA_emission.trc: Configuration file for Transit. Produces emission spectrum.
- twoCIA_emission.trc: Configuration file for Transit. Produces emission spectrum.

4.6.9 f08isothermal

This test checks the case of isothermal emission. For isothermal emission, the result should be a blackbody spectrum corresponding to the temperature of the atmosphere. Full line lists are used for multiple molecules (real, not fake as in previous tests).

Files in f08isothermal/:

- isothermal.atm: Atmospheric model used by Transit.
- isothermal.plc: Configuration file for pylineread. Creates line list input file for Transit.
- isothermal_emission.trc: Configuration file for Transit. Produces emission spectrum.

4.6.10 f09comparison

This test generates forward models for an HD 189733b-like planet for three different PT profiles (isothermal, noninverted, and inverted) in both eclipse and transit geometry. The atmosphere contains H, He, C, N, O, H_2, CO, CO_2, CH_4, H_2O, N_2, and NH_3. Line lists are included for H_2, CO, CO_2, CH_4, H_2O, and NH_3. CIAs are included for H_2-H_2 and H_2-He.

Files in f09comparison/:

- comparison.plc: Configuration file for pylineread. Creates line list input file for Transit. Note: requires line lists. See Section 3.2 for information on downloading line lists if the user wishes to download them. Be aware that this will take a long time and use a lot of hard drive space.
- inv.tea: Atmospheric model used by Transit. Inverted atmosphere.
- inv_emission.trc: Configuration file for Transit. Produces emission spectrum for the case of an inverted atmosphere.
- inv_transmission.trc: Configuration file for Transit. Produces transmission spectrum for the case of an inverted atmosphere.
- iso.tea: Atmospheric model used by Transit. Isothermal atmosphere.
- iso_emission.trc: Configuration file for Transit. Produces emission spectrum for the case of an isothermal atmosphere.
- iso_transmission.trc: Configuration file for Transit. Produces transmission spectrum for the case of an isothermal atmosphere.
- noinv.tea: Atmospheric model used by Transit. Noninverted atmosphere.
- noinv_emission.trc: Configuration file for Transit. Produces emission spectrum for the case of a noninverted atmosphere.
- noinv_transmission.trc: Configuration file for Transit. Produces transmission spectrum for the case of a noninverted atmosphere.

4.6.11 r01isothermal

For this test, the isothermal emission and transmission spectra generated for f09comparison were binned according to some filters (see BARTTest/tests/00inputs/filters/). Noise was added to the

eclipse/transit depths by calculating the theoretical photon noise and pulling from a Gaussian centered on zero and a standard deviation equal to the photon noise. The test is to perform a retrieval of atmospheric parameters and abundances using this eclipse/transit depth data and compare the result to the known inputs.

Note that for the isothermal emission retrieval, the retrieved abundances will likely not match the known inputs. This is because, in principle, the abundances do not effect the resulting spectrum. Due to noise being added to an otherwise perfect blackbody, retrieval algorithms will end up trying to fit the noise. The important result is that the retrieval PT profile is isothermal within 3σ.

Files in r01isothermal/:

- iso_ecldepths.txt: Text file containing the eclipse depths. Comments will point to the corresponding filters.
- iso_ecluncs.txt: Text file containing the eclipse depth uncertainties. Comments will point to the corresponding filters.
- iso_emission.brt: BART configuration file to perform the retrieval. Isothermal emission case.
- iso_tradepths.txt: Text file containing the transit depths. Order corresponds to the filter filters in 00inputs/filters/.
- iso_trauncs.txt: Text file containing the transit depth uncertainties. Order corresponds to the filter filters in 00inputs/filters/.
- iso_transmission.brt: BART configuration file to perform the retrieval. Isothermal transmission case.

4.6.12 r02noninverted

For this test, the noninverted emission and transmission spectra generated for f09comparison were binned according to some filters (see BARTTest/tests/00inputs/filters/). Noise was added to the eclipse/transit depths by calculating the theoretical photon noise and pulling from a Gaussian centered on zero and a standard deviation equal to the photon noise. The test is to perform a retrieval of atmospheric parameters and abundances using this eclipse/transit depth data and compare the result to the known inputs.

Files in r02noninverted/:

- noinv_ecldepths.txt: Text file containing the eclipse depths. Comments will point to the corresponding filters.
- noinv_ecluncs.txt: Text file containing the eclipse depth uncertainties. Comments will point to the corresponding filters.
- noinv_emission.brt: BART configuration file to perform the retrieval. Noninverted emission case.
- noinv_tradepths.txt: Text file containing the transit depths. Order corresponds to the filter filters in 00inputs/filters/.
- noinv_trauncs.txt: Text file containing the transit depth uncertainties. Order corresponds to the filter filters in 00inputs/filters/.
- noinv_transmission.brt: BART configuration file to perform the retrieval. Noninverted transmission case.

4.6.13 r03inverted

For this test, the inverted emission and transmission spectra generated for f09comparison were binned according to some filters (see BARTTest/tests/00inputs/filters/). Noise was added to the eclipse/transit depths by calculating the theoretical photon noise and pulling from a Gaussian centered on zero and a standard deviation equal to the photon noise. The test is to perform a retrieval of atmospheric parameters and abundances using this eclipse/transit depth data and compare the result to the known inputs.

Files in r03inverted/:

- inv_ecldepths.txt: Text file containing the eclipse depths. Comments will point to the corresponding filters.
- inv_ecluncs.txt: Text file containing the eclipse depth uncertainties. Comments will point to the corresponding filters.
- inv_emission.brt: BART configuration file to perform the retrieval. Inverted emission case.
- inv_tradepths.txt: Text file containing the transit depths. Order corresponds to the filter filters in 00inputs/filters/.
- inv_trauncs.txt: Text file containing the transit depth uncertainties. Order corresponds to the filter filters in 00inputs/filters/.
- inv_transmission.brt: BART configuration file to perform the retrieval. Inverted transmission case.

4.6.14 r04WASP12b

WASP-12b is a well-observed exoplanet with observations across the NIR and MIR. It is therefore a good testcase for retrieval algorithms as the plethora of data points allows for the constraining of parameters.

This test uses a combination of space- and ground-based observations of the dayside emission of WASP-12b to perform a retrieval of atmospheric parameters and abundances. Eclipse depths/uncertainties and the filters corresponding to the observation are cited at the top of their respective files.

Files in r04WASP12b/:

-
-
-
-

5 Example

6 How to Use This Software to Test RT Codes

We would like you to contribute the results of your code to BARTTest!

BARTTest has been designed with one main goal in mind: to provide verification that an RT code works as intended. To do so, there are numerous tests that are offered in the package (see Section 4.6).

It is not absolutely necessary for all tests to be executed by some RT code. Rather, the comparison test (see Section 4.6.10) is the most important of the forward modeling tests due to the realistic setup; this best reflects the real cases RT codes are used for. Similarly, the WASP-12b case is the most important of the retrieval tests since it is based on real data. The more codes that yield the same (or similar) answer, the more confident we as a community can be that our codes are all implementing the same mathematics, and thus the more confident we can be in the results of our codes. If there are discrepancies between the results of different codes, the simpler forward modeling tests and synthetic retrieval tests may be able to pin down the cause of the differing results.

In order to contribute results, the user must run their RT code for the exact same cases as those set up in BARTTest/tests/ and then format their code output a specific way. The specific format and naming convention is, of course, the format and naming convention that is adopted by BART. Rather than attempting to explain this format, the user is encouraged to look at the output of BART in BARTTest/code-output/01BART/. If the user does not wish to reformat the output of their code to match this particular format, then they must include a file (in BARTTest/code-output/<yourcode>/) to produce plots of the results of their code (in BARTTest/results/<yourcode>/). This is to ensure that the plots provided can be replicated.

Contributing results to BARTTest will follow a particular format. The code-output/ and results/ directories will have subdirectories for each RT code following a specific naming convention of ##code/ (e.g., 01BART/) as described in Section 4.2. The user may name the directory for their code output/results as they wish. The number preceeding the name of the code is determined by the order in which results are received via a pull request on Github; the user may pick a number but it may be changed depending on the order it is received (e.g., if two users submit results around the same time). If the user does not pick a brief name for their code, the authors will pick one as they see fit.

If the user converts the output of their code to the same format as BART, plots can be easily produced using existing files in BARTTest. BARTTest/lib/makeplots.py can be easily reconfigured to point to the results of a different RT code and produce plots of it as long as it is in the same format.

As mentioned previously, to submit the results of an RT code, the user must submit a pull request on Github. Merging will be handled by the authors. Do not submit the code itself–only the output of the code and plots of that output!