---

basics

install

1. Ask for an invitation to become a member of the group that shares the latest version of HDUST in dropbox (link to the zipped directory). Latest version is 2.02 (04/02/2014).

2. Download the zipped file and unzip it. The directory that is created is `distribution_current`; from here on, I will suppose that `distribution_current` has been unzipped/located in our home directory. It is recommended that you maintain the structure of subdirectories; more information about this can be found in the project directory section.

3. Compilation of the code.

   ```
   $ cd ~/distribution_current
   ```

   - `xdr`: library used for preprocessing and dust

     ```
     $ cd ~/distribution_current/xdr/v1.06
     ```

     This directory contains a Makefile (`makefile`). Open this file and change only the following two lines (if necessary):

     ```
       8 CC = gcc
      10 F77 = gfortran
     ```

     The numbers indicate the position of each line; those numbers are only indicative. What is important is to find the word before the `=` (the last occurence of it - because all previous occurences are overriden) in a line that does not start with `#` (because those lines are ignored) and change the part after the `=`.

     Then give the commands:

     ```
     $ rm *.a *.o
     $ make
     ```

     [dP: We could add a `make clean` option instead of the first command...]

   - `hdust` **serial version**:

     ```
     $ cd ~/distribution_current/fortran/hdustv2.02
     ```

     Open the `Makefile` and make sure the following lines are as quoted:

     ```
      12 #### GFORTRAN
      13 CFLAGS = -C -g -fbounds-check -ffpe-trap=invalid,zero,overflow
      16 FC = gfortran
     ```

     Now you can compile:

     ```
     $ make clean
     $ make
     ```

   - `hdust` **parallel version**:

     ```
     $ cd ~/distribution_current/fortran/hdustparv2.02
     ```

     Open the `Makefile` and make sure the following lines are as quoted:

     ```
      13 #### ALPHACRUCIS
      14 FC = mpif90
      15 CFLAGS = -O2
     ```

     Now you can compile:

     ```
     $ make clean
     $ make
     ```

   [dP: Parallel version should be integrated into the serial version.]

4. `precalcs`: only necessary when you use dust. *[DMF: Follow the same process as for hdust]*: *[dP: How can I? There is no `Makefile` in this directory.]*

   ```
   $ cd ~/distribution_current/runs/precalcs
   $ make clean
   ```

```
$ make
```

before the first run

Before attempting to run HDUST, you have to set the paths to the necessary libraries and compilers. A correct `.bashrc` is probably located by default in your home directory, if you made the right choices when applying for an account in `alphacrucis` (i.e. which compiler you think you will use more, which software you expect to use more etc.), but I let my account be created as general as possible and had problems therafter, which were solved when I finally added the following lines to my `.bashrc`:

```
##### DESCOMENTE SOMENTE AS LINHAS PARA O COMPILADOR QUE SERA USADO
##### MPI COMPILER
  ### INTEL
      source /opthub/intel/bin/compilervars.sh  intel64
      PATH=/opthub/intel_openmpi/bin:$PATH
      LPTH=/opthub/intel_openmpi/lib:/opthub/intel/lib/intel64
      LPTH=$LPTH:/opthub/intel/composer_xe_2011_sp1.9.293/mkl/lib/intel64/:/lib64

  ### OPEN64
    # PATH=/opt/open64/bin:/opt/open64_openmpi/bin:$PATH
    # LPTH="/usr/lib64/:/opt/open64/open64-gcc-4.2.0/lib64
    # LPTH=$LPTH:/opt/open64/open64-gcc-4.2.0/lib
    # LPTH=$LPTH:/opt/open64/lib/gcc-lib/x86_64-open64-linux/5.0/64"

  ### GNU
    # PATH=/opthub//gnu_openmpi/bin:$PATH
    # LPTH="/opthub/gnu_openmpi/lib"

  ### PGI
    # PATH=/opthub//pgi/linux86-64/12.2/bin:$PATH
    # LPTH="/opthub/pgi/linux86-64/12.2/lib"
    # MANPATH=$MANPATH:/opthub/pgi/linux86-64/12.2/man
    # LM_LICENSE_FILE=$LM_LICENSE_FILE:/opthub/pgi/license.dat

##### TORQUE/MAUI
      PATH=/opthub/torque/bin:/opthub/torque/sbin:/opthub/maui/bin:$PATH

##### SPH STUFF
      PATH=/sto/home/carciofi/runs/SPH/:$PATH

##### MPI
      ulimit -s unlimited

##### export all necessary paths at once
      LD_LIBRARY_PATH=$LPTH
      unset LPTH
      export PATH LD_LIBRARY_PATH MANPATH LM_LICENSE_FILE
```

[dP: Only one of `INTEL`, `OPEN64`, `GNU`, `PGI` blocks should be kept. It's a minimal version of `.bashrc`, so I believe I should remove the rest of them for simplicity reasons. Have there been any problems with `gnu_openmpi`? If not, then we could have this uncommented by default.]

project directory

The project directory is located at `~/distribution_current/runs/hdust/bestar2.02`. You can rename this directory freely, but the names of the subdirectories below it should be kept as they are in the original copy. The files located in the subdirectories can be renamed and/or modified.

The directory `bestar2.02` contains 10 subdirectories. *[dP: Daniel's manual refers to an additional, obsolete directory `line`. Is that why I don't have it?]* Each directory contains one or more files, each of which can be used for a run of HDUST with different model parameters. Which of the files will be used is defined in the master file.

1. `mod[N]`, where `[N]` is a string of 2-3 digits that is derived from `hdust_bestar2.02.inp` as described in § model ID + SUFFIX. In the directory with this name, there will be at least one file named `mod[N][R].txt` that describes the envelope, in particular the parameters of the circumstellar ambience (disk and/or dust and/or wind). The data in this file has to be consistent with the data given in `gridcells`. The optional `[R]` (it can also be empty) in the file name `mod[N][R].txt` will be explained below (§ model ID + SUFFIX; § list of runs).

2. `simulation`: This directory controls the simulation parameters: number of photons, files for the initial gas or dust temperature at the first step of the simulation *[dP: that is what is meant by "step 1" in Daniel's manual, isn't it?]*, spectral band and spectrum resolution

for the observables.

`HDUST` must in two separate phases: an initial iteration phase (`STEP = 1`) and one or more post-processing phases (`STEP > 1`). This is defined inside the `SIMULATION` section of the input file, through the first variable that is declared therein, `STEP`:

- iteration phase
  - `STEP = 1`: The program iterates the solution to calculate the H level populations, temperatures, and hydrostatic equilibrium density (where required) for all cells. The results are output in the `mod[N][R]nn.temp`, and `mod[N][R]nn.dust` files, where `nn` is the iteration number. Some samples for `STEP = 1` are located in `bestar2.02/simulation/`, e.g. `step1.txt`, `step1_long.txt`, `step1_refine.txt`.
- post-processing phase: In this mode, the program caclulates the emergent spectrum. There are three different modes for this:
  - `STEP = 2`: The program performs a simulation for a specified wavelength range. The initial position of the emitted photon is sampled using the total emissivity of each cell and the luminosity of the star. In this method, photons are destroyed when they are absorbed by the gas. *[dP: I found no sample.]*
  - `STEP = 4`: The program employs a radiative equilibrium procedure, in which photons are emitted from the star using the entire frequency distribution of the stellar spectrum. In this method the photons are not destroyed when they are absorbed by gas, but rather are reemitted in the same position with a frequency given by the local emissivity. A sample for `STEP = 4` is located in `bestar2.02/simulation/step4.txt`.
  - `STEP = 5`: The program performs a simulation for a specified wavelength range, like `STEP = 2`, but instead an equal number of (weighted) photons is emitted by each cell. This is usually the most efficient method (especially when images are desired). A lot of samples for `STEP = 5` are located in `bestar2.02/simulation/`, e.g. `halpha.txt`, `ir.txt`, `radio_cm.txt`, `radio_mm.txt`, `sed.txt`.

Reminders:

- Always keep the value of `N_batch` equal to zero (`N_batch = 0`). `HDUST` automatically divides the number of photons for each parallel processor.
- In complete simulations (photosphere+disk) of the lines, enable the photospheric line of option 2 (two = read the file with the spectra from SYNSPEC). If it is disabled, then we actually have emission from a pure disk. *[dP: Problem here; I don't understand. Check files and ASK. And maybe add more information.]*

3. `source`: It describes the central star: ***L***, ***M***, limb darkening, spectral lines etc.

   Reminders:

   - Define the spectra (including the photospheric lines). *[dP: Same here; I don't understand.]*

4. `composition`: Chemical composition of the gas
5. `gridcells`: The grid of the simulation is defined within this directory. The selected (in master file) file has to be modified if it is a disk, a wind or another 3D distribution. *[dP: I had trouble realising why I could not make a run with perturbation of type 1, spherical region; it came out that I had to change the number of grid cells in the `N_phi` in `gridcells`.]*
6. `fullsed`: The results of the simulation; SEDs in various bands
7. `observers`: Describes the parameters of the observer (latitude, angle ***i***), as well as the azimuthal angle for non-symmetrical simulations, as in the case of the disk of ζ-Tauri.
8. `controls`: Controls the "physics" of the simulation. In general it is used for debugging.
9. `images`: This directory contains a(t least one) file with parameters for the images (resolution, spacial scale, ***λ***). It must be consistent with the data given in `simulation`.
10. `perturbations`: Optional. Causes deviation from the default parameters, e.g. regions of lower or higher density, presence of blobs, holes in the disk etc.

All 10 subdirectories HAVE TO BE IN the project directory, as each one contains possible input files with information on the field their name suggests. Each subdirectory contains at least one input file. From within the possible input files in each subdirectory, you have to choose one and

define its name in the master file).

run

1. Go to the directory where the HDUST executable is located:

   ```
   $ cd ~/distribution_current/runs/hdust/
   ```

   In that directory we should find the project directory and the master file.

2. Create the project directory, i.e. a directory for the input and output files of your project. You may simply use the sample directory bestar2.02 that is already there or you can copy it. Whatever the name, we will call it as the project directory.

3. Create a master file. You may simply use the sample master file hdust_bestar2.02.inp that is already there or you can copy it.

4. Make the desired modifications on the input files located in the subdirectories of the project directory and/or select the desired input files in the master file.

5. You run the (serial version of the) program with the command:

   ```
   $ ./hdustv2.02.bc input = hdust_bestar2.02.inp
   ```

---

master file

sample

In any input file, all lines starting with an exclamation mark (!) are considered as comments and have no effect in the forecoming run. A sample master file looks like this: (In order for it to be a valid master file, N has to be substituted by a number, as explained below.)

```
        PROJECT = bestar2.02
        MODEL = N

SUFFIX='' SIMULATION='step1' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVE
SUFFIX='b' SIMULATION='sed' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVEF
SUFFIX='b' SIMULATION='ir' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS
SUFFIX='b' SIMULATION='halpha' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSEF

SUFFIX='' SIMULATION='step1_long' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' AD
SUFFIX='a' SIMULATION='step1_refine' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls'
SUFFIX='a' SIMULATION='sed' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVEF
SUFFIX='a' SIMULATION='ir' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS
SUFFIX='a' SIMULATION='halpha' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSEF
```

[dP: Is it true that each line corresponds to one run of HDUST? Or does every new SUFFIX corrspond to a new run? I mean if I add new line characters, e.g. before GRIDCELLS, will there be a problem?]

model ID + SUFFIX

In the beginning of the master file, you have to define the name of the project folder, and the model number.

The model number defines the name of the directory of the input file that describes the circumstellar envelope. It is the same directory where the output files will be saved. The model number is defined in the line

```
        MODEL = N
```

where N ∈ [1,999]. If N is a single digit number, a zero will be prepended to the model number for the name. This means that there must be a directory mod[N] inside the project directory (see § project directory, item 1).

Each line below the PROJECT and MODEL lines correspond to one run of HDUST. The first directive of each line is SUFFIX. The strings defined for PROJECT, MODEL and SUFFIX are able to define where all input parameters will be found. So for example, if PROJECT = bestar2.02, MODEL = 2 and SUFFIX = c, the HDUST executable will attempt to find the file bestar2.02/mod02/mod02c.txt, where all information necessary for the run can be found.

overrides

However, it was considered more practical to be able to separate this large file into pieces/sections of information on the values of the numerous parameters. Therefore in each run's line, the keyword SUFFIX is followed by keywords defining the rest of the input files for each run. The data given on those files may as well be located in mod[N][R].txt, where N is the MODEL and R is the SUFFIX. If they cannot be found in mod[N][R].txt, then it is essential that you define the relevant keywords.

The sections of the input data are divided into the essential sections and the optional sections:

- essential: SIMULATION, SOURCE, COMPOSITION, GRIDCELLS, FULLSED, OBSERVERS. The essential sections have to be included either in mod[N][R].txt or defined as overrides in the run's line within the master file.*[dP: I don't understand the reference to FULLSED. If FULLSED contains the results of the simulation (as mentioned in Daniel's manual), then why is it essential? I believe that it is not among the essential keywords, since the program runs very well without defining this keyword. I guess that if the FULLSED keyword were be defined, then the only thing that would change would be the name of the output files in directory ./bestar2.02/fullsed/. Yet, I do not know what is the defualt name; it probably will not have a .txt extension, as all files in that directory end in .sed2. Need more info.]*
- optional: CONTROLS, IMAGES, PERTURBATIONS. If the optional keywords are absent from both mod[N][R].txt and they do not exist as keyword overrides in the master file, then HDUST will use some standard values.

If an override is set, then HDUST will look elsewhere for the appropriate section of the input file. For example, if SOURCE = zeta_tau, HDUST will look for information on the source (section IV of the master file) in ./bestar2.02/source/zeta_tau.txt.

If any of these keywords are appended in the run's line, then they override any relevant section given in mod[N][R].txt. If a keyword is essential and the relevant section can not be found in mod[N][R].txt, then it is absolutely necessary that we define those overrides.

The file mod[N][R].txt may contain all information needed for a single run, therefore none of the rest subdirectories of the project directory are really essential. This means that we could as well copy+paste all data given in the files defined by the keywords other than SUFFIX inside mod[N][R].txt, and get rid of all the subdirectories and files beneath them. But that's simply not practical.

list of runs

You may have noticed that in the sample master file there is a keyword that I did not include neither in the list of the essential keywords nor in the list of the optional keywords, i.e. the ADDSUFFIX keyword. Each run of HDUST produces a series of output files, one of which can be used as the master file for a new run. So the SUFFIX and ADDSUFFIX keywords essentially define uniquely the input and the output of each run.

The list of runs can be arbitrarily long. As an example, we will examine the lines for the list of runs from the sample master file.

```
37 SUFFIX='' SIMULATION='step1' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='obse
38 SUFFIX='b' SIMULATION='sed' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='obser
39 SUFFIX='b' SIMULATION='ir' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='observ
40 SUFFIX='b' SIMULATION='halpha' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='ob
41
42 SUFFIX='' SIMULATION='step1_long' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' ADDSUFFIX=
43 SUFFIX='a' SIMULATION='step1_refine' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVE
44 SUFFIX='a' SIMULATION='sed' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='obser
45 SUFFIX='a' SIMULATION='ir' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='observ
46 SUFFIX='a' SIMULATION='halpha' SOURCE='sample_be' COMPOSITION='pureH' GRIDCELLS='grid' CONTROLS='controls' OBSERVERS='ob
```

Suppose that above the list of runs, the project name and model ID were given as: PROJECT = bestar2.02, MODEL = 2. With such a master file, HDUST will run 9 times:

1. The first time, HDUST will follow the directives given in ***line 37***: It will first search for the file ./bestar2.02/mod02/mod02.txt. Some sections of ./bestar2.02/mod02/mod02.txt will be overriden with data taken from the designated files:
    - SIMULATION: ./bestar2.02/simulation/step1.txt
    - SOURCE: ./bestar2.02/source/sample_be.txt
    - COMPOSITION: ./bestar2.02/composition/pureH.txt
    - GRIDCELLS: ./bestar2.02/gridcells/grid.txt
    - CONTROLS: ./bestar2.02/controls/controls.txt
    - OBSERVERSS: ./bestar2.02/observers/observers.txt

    In the end of the run, a new file will be created whose name will include the string defined for the ADDSUFFIX keyword: ./bestar2.02/mod01/mod01b.txt.
2. The second time, HDUST will follow the directives given in ***line 38***: It will first search for the file ./bestar2.02/mod02/mod02b.txt (asSUFFIX='b'), which was just created in the previous run. The overrides that will take place will be the same as in previous run, except for SIMULATION:

- SIMULATION: ./bestar2.02/source/sed.txt

  Recall that HDUST runs in two separate pahses, as explianed in § project directory, item 2: One is the iteration pahse, which was completed in the first of this list of runs, using ./bestar2.02/simulation/step1.txt. So, in this second run of the list, we perfomr the post-processing pahse (STEP = 5) for a specified wavelength range.

3. The third time, HDUST follows the directives given in ***line 39***: Again, it uses the file ./bestar2.02/mod02/mod02b.txt, which was created in the first run. It will run in the post-processing mode again (STEP = 5), as in the second time above, but now using a different wavelength range, defined in ./bestar2.02/source/ir.txt.

4. The fourth time, HDUST follows the directives given in ***line 40***. Similarly to the two previous runs, HDUST will in the post-processing mode, using a different wavelength range, ./bestar2.02/source/halpha.txt.

5. The fifth run corresponds to ***line 42***. Now we are done post-processing the first run, and we will run a new iteration phase. So we start again with the initial model file (./bestar2.02/mod02/mod02.txt, no SUFFIX here) and our simulation file is a new one (./bestar2.02/simulation/step1_long.txt) with STEP = 1. In the end of the run a new file ./bestar2.02/mod01/mod01b.txt will be created.

6. . . .

---

known problems

akreisL

In akreisL a problem comes out in file read_master.f90, subroutine set_file_pointers, input variable mode:

```
Fortran runtime error: Actual string length is shorter than the  \
    declared one for dummy argument 'mode' (0/5)
```

In particular this happens right after reading controls.txt.