Astrobear is a hydro model machine that has been adapted to many purposes. The topic of interest to me has been simulating various types of flows (of several classes of shapes) into external media (also of several classes of shapes). This set of notes is a primer that documents the highlights of what I've learned about how to configure Astrobear to conduct an outflow parameter study, primarily in November 2104. These notes are written as a blog to myself, but I'm happy to share them with other users.

Many fantastic people at the University of Rochester made my work very productive and fruitful. There's no way to organize the list sensibly. So here are the names: Adam Frank, Baowei Liu, and Martin Huarte-Espinosa. My very sincere thanks to all of them. Three students at UW have also been instrumental in helping out: Keira Brooks, Brenda Bushell, and most recently Rebecca Kemmerer. Thanks guys!

Bruce Balick

RUNNING ASTROBEAR

works in tandem with display software such as VisIt to create and display hydro models of collimated stellar outflows into a geometrically structured environment. Astrobear os the hydro machine and it runs on supercomputers. VisIt is designed to operate as stand-alone software that works on laptops and desktops. It reads the arrays of hydro state variables produced by Astrobear and -- when it isn't crashing -- displays them in very flexible formats.

Astrobear is executed remotely from within a directory on the host supercomputer. The name of that directory should be a good description of the initial parameters, especially if you plan to run a series of similar models. For example, I created a directory with the name ' tapOBL15R2n4e2v200namb4e4 ' for a sim in which a tapered conical jet is launched obliquely into a AGB wind with initial conditions specified in a file within the directory called "problem.data". This complex name is a succinct pasteup of parameter choices that allow me to distinguish the various model outcomes on my extensive parameter study.

STRUCTURE OF THE UNIX DIRECTORY FOR EACH RUN

Immediately before executing Astrobear the controlling directory on the host computer has these files:

astrobear	communication.data	out	problem.data	solver.data
Bondi_alpha.tab	global.data	physics.data	scales.data	TABLES

You will have downloaded the various files with the help of the Astrobear group. All of these files are essential, and they must be in the same directory that you have open when you start the Astrobear execution. Once in place, it is very rare that you will change any of these files other than 'problem.data '.

Before execution you MUST configure the initial conditions within the file ' problem.data '. You may also make a few changes to the file ' global.data ' with all

sorts of technical parameters, though this is infrequent. The directory ' out ' must be present but completely empty since this is where Astrobear leaves its results as it runs (I routinely use ' rm out/* ' for this purpose). If it happens to be there before execution of Astrobear, remove the file ' std.out ' as well. Then you're good to go.

After execution this directory will look like:

```
astrobear        communication.data  out           problem.data  solver.data  TABLES
Bondi_alpha.tab  global.data         physics.data  scales.data   std.out
```

You will see a new file called ' std.out ' in which error messages and some basic information about the execution of Astrobear can be found. If the run is successful then all of the results are a series of "chombos" contained within the ' out ' directory. These files will be passed to visualization software for display and your inspection. (The chombos are a time series of intermediate and final results. Each of them is an ' hdf ' file that is jam packed with arrays of the density, speed, temperature, etc. at each of the chombo time steps. You determine the maximum model time and the number of chombos within ' global.data ')

PREPARING FOR ASTROBEAR EXECUTION

This is usually very simple: Copy an old ' problem.data ' from a previous run, modify up to about a dozen values within it with a simple text editor, and start the run.
Here's an example of one the ' problem.data ' file that I created within the directory ' tapOBL15R2n4e2v200namb4e4 '. Note that the usable part of the file begins with ' &ProblemData ' and ends with ' / '. Extra lines at the end are not used.

```
&ProblemData
! MODEL: tpaer 15deg  (n=4e2, r=1000AU, T=1000K) @ 200 km/s
! into AGB wind of density 4e4 at r = Rjet
! folder tapOBL15R2n4e2v200namb4e4
!
!      BACKGROUND or “AMBIENT” SECTION. Values apply to origin
tamb = 1d3           ! ambient temp, 1cu = 0.1K (100K=1000cu)
namb = 4e4           ! ambient central density cm^-3. Usually 400 for 1/r^2 or
torus.
stratified = t       ! true = add a 1/r^2 background 'AGB stellar wind'
torus      = f       ! true - add torus to the background
torusalpha = 0.7     ! alpha and beta specify the geometry
torusbeta  = 10d0    ! see Frank & Mellema, 1994ApJ...430..800F
rings      = f       ! true - add radial density modulations to AGB wind
!
!     FLOW  DESCRIPTION SECTION, values apply at origin at t=0
outflowType  = 2    ! TYPE OF FLOW    1 cyl jet, 2 conical wind, 3 is clump
njet  = 4d2          ! flow density at launch zone, 1cu = 1cm^-3
Rjet  = 0.5          ! flow radius at launch zone, 1cu = 500AU (outflowType=1
only)
vjet  = 2e7          ! flow velocity , 1cu = cm/s (100km/s=1e7cu)
tjet  = 1d3          ! flow temp, 1cu = 0.1K (100K=1000cu)
tt    = 0.0d0        ! flow accel time, 1cu = 8250y (0.02 = 165y)
```

```
open_angle = 90d0    ! conical flow open angle (deg)
tf    = 15d0        ! conical flow Gaussian taper (deg) for njet and vjet; 0=
disable
lObliqueCone = t    ! T = tapered flow launched 45 deg to x and y axis
(torus=f,tf>0)
sigma = 0d0         ! !toroidal.magnetic.energy / kinetic.energy, example 0.6

!
!     OTHER PARAMETERS
lcooling = t         ! radiative cooling?
buff  = 8            ! central refinement of a grid with a resolution 1/2
/

extra command lines, to use move them into the problem.data file
! jet  flow  modulation (A*t^a + B*t^b) * exp(-ct), t = time in cu (not
tested)
balick_a  = 0d0     !A, default=0
balick_aa = 0d0     !a, default=0
balick_b  = 0d0     !B, default=0
balick_bb = 0d0     !b, default=0
balick_c  = 0d0     !c, default=0
```

The ambient (external or environmental) medium is static and described by a temperature `tamb` and a specific density distribution. The geometry of the ambient density distribution is controlled with the variables `stratified`, `torus`, `torusalpha`, `torusbeta`, and `rings`.

If `stratified = t` then

n(r) = `namb` (r/`Rjet`)^-2

where namb and Rjet are parameters found within "problem.data" . This mimics a pre-existing constant stellar wind, also known as a stratified wind or alternately an AGB wind.

If `torus = t` then a torus is added to the background whose peak density is given by `namb`. The functional form of the torus's density distribution can be controlled using `torusalpha` and `torusbeta` (`see Frank & Mellema, 1994ApJ...430..800F`).

If `rings = t` then the initial density AGB wind is modulated by a periodic function to simulate a series of rings that are often found around AGB stars and some prepNe using Hubble images,

If `stratified = t`, `torus = f`, and `rings = f` then the ambient density is constant.

The description of the injected flow is more complex. The flow will have one of several geometries defined by '`outflowType '`. The flow is initially injected onto the grid at a radius `Rjet` (`Rjet` is the radius of the injection orifice) with a density `njet` , speed `vjet` , and temperature `tjet`. The parameter governs the injection rate at the outset. A value of zero means no acceleration of the flow. Larger values (in units of 8250y) result in a more gradual speed of injection which is useful if you find odd behaviors developing early in the time series. Sigma governs the strength of the magnetic field injected into the flow.

If the flow is a cylindrical jet (`outflowType = 1`) then it is normally launched with constant `njet` and `vjet` at the orifice of radius `Rjet`. The density and velocity can be modulated with time using either '`tt `' or the five '`balick `' parameters listed at the end

of the ' problem.data ' file, but not both.

If the flow is a clump (`outflowType = 3`) then `Rjet` is its effective radius. However, the clump's density is not uniform. Rather, the clump density decreases with radius as

nclump(r) = `njet` [1-(`Rjet`/r)^2)] within r = `Rjet`.

That is, the clump density distribution is parabolic and falls to zero at the outer edge of the clump.

If the flow is conical (`outflowType = 2`) then `open_angle` governs the angular width of the injected flow. For example, an opening angle of 30 deg results in a uniform flow (isodensity and isospeed) that is contained within zenith angles of 30 degrees. An opening angle of 90 deg is a spherical wind. Note: any parameters in ' problem.data ' that do not apply to models of cylindrical jets (`outflowType = 1`) or clumps (`outflowType = 3`) will be ignored. These include `open_angle`, `tf`, and `lObliqueCone`.

A variation of the conical or spherical flow is the "tapered" cone. In this case the flow density and velocity that enter the grid from the central launch sphere are modulated ("tapered") by Gaussian of the form

taper function = exp -[(zenith angle - theta)/tf)^2]

where theta is zero for a flow symmetric about the y axis.

The flow can be injected into the grid at theta=45 degrees (`lObliqueCone = t`, a 45-deg zenith angle) if desired. This can ONLY be used with `outflowType = 2` and `tf > 0`). This diagonal injection can be useful for avoiding instabilities that may develop on the axes of the grid, which is not uncommon. It has only a partial effect on wide cones (`tf > 45`) and no effect on spherical flows (`open_angle = 90d0 & tf = 0`). That is, the primary purpose of setting `lObliqueCone = t` is diagnostic in nature. [In principle, changing the injection angle of a tapered flow should have no effect on the model outcome. In practice that may not be the case when the flow axis and the y axis are aligned. So changing the injection angle to 45 deg provides a way to distinguish artificial from real features in tapered flows.]

If this all sounds complex, it isn't. You'll get used to it quickly and easily.

PREPARING FOR VISUALIZATION OF ASTROBEAR's RESULTS

The fun begins after Astrobear finishes producing its chombos. The workflow is this: the chombos contained in the directory ' out ' on the host computer (typically 200 to 500 Mbytes) are downloaded to the disk on the laptop that I use to run VisIt (in my case a Mac laptop with a large external hard disk). I place the chombos in a directory (a 'folder' on the Mac) of the same name that I used on the supercomputer. I also download and store the problem.data file in the same folder since no other record of record the parameter settings used in the models is kept in the chombos.

I have been using VisIt (https://wci.llnl.gov/simulation/computer-codes/visit) to visualize the results of Asdtrobear contained in the sets of chombos. After the chonmbos and the problem.data files are on my laptop I configure certain Visit 'session' files that Visit uses to connect to and open each set of chombos. I pray intensely as I do this since the process should be simple but it isn't. Visit doesn't always open the chmobos and I've never figured out why. One of Visit's many shortcomings is that it error reporting is not helpful. Visit finds all sorts of petty excuses to fail. It might complain about the

chombos not being hdf files (which they are) on one day and then open them easily the next. I find that I must restart Visit on my Mac many times daily. (Baowei insists that he doesn't have those problems when he runs VisIt on his Linux desktop.)

OK. Enough whining. And, unless you're Baowei, even larger frustrations await you even after Visit has opened the sets of chombos.

In sharp contrast to the simplicity and ease of running Astrobear, visualization of the chombos in the ' out ' directory can be extremely messy with VisIt. To say the least, VisIt takes some training thanks to its multitude of highly varied display options and its archaic architecture. VisIt is a fiendishly cantankerous tool. So, at this point, I leave the reader with Baowei. To quote the score of Bob Dylan's 115th Dream, "I just said 'good luck'!".