The Curious Case of V CVn

by

© **Michael T. Power**

A project submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of Master of Science.

Department of Physics and Physical Oceanography
Memorial University

August 2024

St. John's, Newfoundland and Labrador, Canada

Abstract

From many decades of observations, the star V Canum Venaticorum (V CVn) has strange behaviour regarding the maxima and minima of its light curve and linear polarisation, which have an approximately inverse relationship (sometimes with a lead/lag time between them) and an almost constant polarisation position angle. One theory proposed to explain this strange behaviour is the existence of a bow shock driven by a spherically symmetric dusty wind from the star. In this theory, the wind is assumed to vary with time due to radial pulsations. This work uses a new framework developed in Zeus3D, a multi-physics magnetohydrodynamics code, to test this theory. The results of this work show that when a time-varying stellar wind is at its maximum brightness the polarisation signal is at a minimum due to a dense, symmetric shell which forms around the star. Conversely, when the brightness is at a minimum, the symmetric shell around the star is much less dense, and the polarisation is instead dominated by the asymmetric bow shock structure, causing the polarisation signal to attain a maximum value. Numerically reproducing the observed inverse relationship between the polarisation and light curve provides a strong theoretical argument that a variable stellar wind bow shock is the solution to the curious case of V CVn.

In Loving memory of my grandmother.

It is with a heavy heart that I dedicate this work to my dearest late grandmother, Mary Neville, whom I lost during my first semester here at Memorial University. She was the matriarch of our small family, the glue that held it together. Starting long before I was born, she cooked dinner for everyone in our family every Sunday and often for unrelated community members and was always there when you needed her. She was one of the most brilliant people I've ever had the pleasure of knowing; she excelled at everything, whether it was acing her provincial exams in grade 12, destroying everyone in tarabish or cribbage, English, mathematics, taxes, poetry, she knew it all, even learning how to play the piano in her eighties. Above all else, she was kind, wise, and completely selfless; she sacrificed her education by getting a job and helping all her siblings financially through teaching and nursing college. On top of all that, very few grandchildren can say that they thoroughly enjoyed drinking and partying with their grandmother, but I can say that without a second thought.

Due to her love of poetry, I dedicate a poem to her by my great-great-grandfather. He was a farmer with no formal education but another person in a long line who, like my grandmother, worked hard to get me to where I am today and who clearly was quite intelligent.

True Friendship & False

George Mahan

circa 1870

I often hear a plaintive sound, with the singing birds of Spring,
As in the shades and woodland glades, their music sweet they sing.

In the morning early, and in the evening late,
They'd discourse true love and friendship, with music to their mate.

If each mortal here would learn sincere, a lesson from each bird,
The world would be a happy place, we'd hear no angry word;
The human race would be united, and get peace from Heaven above,
And true friendship with each other would soon ripen into love.

There are two kinds of friendship, and there are two kinds of love;
The false is of this world, but true love's from Heaven above.

The true love is in Heaven, therein a secret lies,
It can't end or change like eternity, for such love never dies.

There is a false deceitful love, it can only last a time,
It depends upon our nature, when love is changed to crime;
Such creatures are unworthy, they never can improve,
For their hearts with pride are scornful, such hearts are not for love.

They can't enjoy its raptures, because the heart is lost;
They are like a vessel in distress, by the foaming billows tossed;
No doubt they may get married, and life's gloomy footpath brave,
And enjoy a life of misery, until they meet the grave.

The same love by which He fathoms space,
As He in silence views the false and true, and divides the human race.
But those who possess true friendship, and His righteous laws do keep,
Shall enjoy a bright home of celestial bliss when in death we fall asleep.

All men are created out of dust, what use for selfish pride?
And our day of death uncertain while through life's lonely vale we glide;
The rich are not for Heaven, for their comforts here they have,
And they'll rot and crumble into dust, as the poor man in his grave.

So let patience and forbearance lead all true men through life,
Banish selfish interference, which only ends in strife.
God made us all for happiness, and never to rebel,
So strangers to true friendship shall find a home in Hell.

Lay summary

Stars constantly blow off gas from their outer layers, releasing it as they move. Just as when you clap your hands or crack a whip, the gas coming from the star hits the gas and dust that the star is moving through in space, making a shock wave. This research comes down to modelling the complicated physics of these shock waves in a computer. These shock waves that the stars make change the light they produce by obscuring and altering its qualities; you can think of this like a car's headlights on a foggy night. So, when the light from these distant stars is observed here on Earth, it doesn't look how we expect it to. However, with this research, there's hope to change that by understanding how these shock waves alter the light from these stars.

The results of this work give strong evidence that the cause of the unexpected light being observed here on Earth depends on these shocks and the way that the stars blow off their gas. This is studied numerically with a code called Zeus3D.

Acknowledgements

The author would like to thank many people and entities who made this research possible. Most notably, Dr. Hilding Neilson for partially funding this research and for his vast knowledge of astrophysical phenomena. Next, Dr. David Clarke for his illuminating discussions on computational MHD, as well as Tahere Parto and Galina Sherren for their incredibly useful editorial comments. Also, the author would like to thank the National Sciences and Engineering Research Council of Canada for funding provided by a Canada Graduate Scholarship-Master's (CGS M) and Memorial University of Newfoundland for providing funding via a Dean's Scholarship, as well as the Centre for Analytics, Informatics and Research at Memorial University for providing high-performance computing facilities.

This work has made use of data from the European Space Agency (ESA) mission *Gaia* (`https://www.cosmos.esa.int/gaia`), processed by the *Gaia* Data Processing and Analysis Consortium (DPAC, `https://www.cosmos.esa.int/web/gaia/dpac/consortium`). Funding for the DPAC has been provided by national institutions, in particular, the institutions participating in the *Gaia* Multilateral Agreement.

Statement of contribution

The design and identification of this research topic were ongoing collaborative efforts between the author and Dr. Hilding Neilson, who was in a supervisory role.

All practical aspects of this research were independently undertaken by the author, with Dr. Hilding Neilson contributing in a supervisory role.

The data analysis and codes made for this thesis were the sole effort of the author.

The manuscript was prepared solely by the author.

Table of contents

List of tables

List of symbols

$\mathbb{1}$	The identity tensor.
$\vec{B}$	The magnetic field vector.
$\hat{C}(\tau_l)$	The normalised cross-correlation function.
$\vec{E}$	The electric field vector.
e_0	The ISM internal energy.
e	The internal energy of a fluid.
e_T	The total energy of a fluid.
$e_w(t)$	The time-varying internal energy of a stellar wind.
f	A generic integrand.
G	Newton's gravitational constant.
i	Inclination angle.
I_0	The unpolarised light intensity of a star.
I_1	The first intensity Stokes parameter.
I_2	The second intensity Stokes parameter.
I_3	The third intensity Stokes parameter.
I	The sum of the first intensity Stokes parameter and the unpolarised light intensity.
k_B	The Boltzmann constant.
$\vec{L}$	The Lorentz force.
$\mathcal{L}$	The radiative cooling function.
M_A	The Mach number.
$\bar{\dot{M}}$	The average mass-loss rate of a stellar wind.
$\dot{M}_w$	The mass-loss rate of a stellar wind.
$\dot{M}(t)$	The time-varying mass-loss rate of a stellar wind.
$\dot{M}_{\mathrm{Max}}$	The maximum mass-loss rate of a stellar wind.
$\dot{M}_{\mathrm{Min}}$	The minimum mass-loss rate of a stellar wind.
$\bar{m}$	The average mass of particles in a gas.
n_0	The ISM particle number density.
N	The number of random points used in a Monte Carlo method.
P_0	The ISM thermal pressure.
P	The thermal pressure of a gas.

List of symbols

P_B	The magnetic pressure of a gas.
P_R	The residual polarisation from scattering in a density field.
P_w	The thermal pressure of a stellar wind.
Q	The normalised first Stokes parameter.
R_{SO}	The standoff distance to a stellar wind bow shock from a star.
$\bar{R_{\mathrm{SO}}}$	The average standoff distance to a stellar wind bow shock from a star.
R_w	The radius of a stellar wind bubble.
R_{sim}	The radial extent of a simulation.
$\mathcal{R}_y$	The rotation matrix about the y-axis.
r	The radial coordinate in a spherical polar system.
$\hat{r}$	The radial coordinate unit vector in a spherical polar system.
$\vec{r}_P$	The position vector corresponding to a scattering point within a density field.
$\vec{s}$	The momentum vector of a gas.
$\mathbb{S}$	The shear tensor.
t	The time coordinate.
t_{sim}	The total time for a simulation.
T_0	The ISM temperature.
T_w	The stellar wind temperature.
U	The normalised second Stokes parameter.
$\vec{v}$	The velocity field of a gas.
$\vec{v}_{\mathrm{ISM}}$	The velocity field of the ISM.
V	A volume.
V_{zone}	The volume of a computational zone.
v_w	The speed of a stellar wind.
$\vec{v}_*$	The velocity vector of a star.
$\bar{v}_w$	The average speed of a stellar wind.
$v_{w,\mathrm{Max}}$	The maximum speed of a stellar wind.
$v_{w,\mathrm{Min}}$	The minimum speed of a stellar wind.
$\hat{x}$	The x-direction unit vector in a Cartesian space.
$\vec{x}_i$	The coordinate of a point in a Cartesian space.

List of symbols

$x_{1a}(i)$	The a-grid 1-direction coordinate in Zeus3D.
$x_{2a}(j)$	The a-grid 2-direction coordinate in Zeus3D.
$x_{3a}(k)$	The a-grid 3-direction coordinate in Zeus3D.
$\hat{y}$	The y-direction unit vector in a Cartesian space.
$\hat{z}$	The z-direction unit vector in a Cartesian space.
β_{AD}	The ambipolar diffusion coefficient.
γ	The ratio of specific heats.
γ_P	The geometric polarisation factor.
δt	A time step.
$\delta\theta$	The difference in angle between zone faces in Zeus3D.
δ_{ij}	The Kronecker delta.
ζ	The ratio of the standoff distance to the wind bubble radius.
η	The ratio of a stellar wind's maximum to minimum mass-loss rate.
θ	The polar angle.
$\hat{\theta}$	The unit vector for the polar angle.
λ	The ratio of a stellar wind's maximum to minimum speed.
μ	The viscosity.
π	The ratio of circumference to diameter.
ρ_0	The mass density of the ISM.
$\rho_w(t)$	The time-varying mass density of a stellar wind.
σ_0	A convenient constant related to the Thomson scattering cross-section.
σ_T	The Thomson scattering cross-section.
τ	The period of V CVn.
$\bar{\tau}$	The average optical depth.
ϕ	The azimuthal angle.
$\hat{\phi}$	The unit vector for the azimuthal angle.
χ	The scattering angle.
ψ	The polarisation position angle.
ω	The angular frequency.

List of abbreviations

CFL	Courant–Friedrichs–Lewy.
EM	Electromagnetic.
ISM	Interstellar medium.
K-HI	Kelvin-Helmholtz instability.
MHD	Magnetohydrodynamics.
MUTCI	Monotonic, up-winded, time-centered interpolation.
PPI	Piecewise parabolic interpolation.
R-TI	Rayleigh-Taylor instability.
V CVn	V Canum Venaticorum.

Chapter 1

Background and Introduction

> Exploration is in our nature. We began as wanderers, and we are wanderers still. We have lingered long enough on the shores of the cosmic ocean. We are ready at last to set sail for the stars.
>
> *Carl E. Sagan*

In physics, especially astrophysics, one must often study phenomena that exhibit unexpected behaviour or defy conventional wisdom to gain a deeper understanding of the universe. This thesis addresses a star that exhibits unexpected relationships between its observed parameters. The star in question is known as V Canum Venaticorum (V CVn). Semi-regular variable stars have the property that their brightness changes with time and pulsates periodically. However, the variation in brightness may change from cycle to cycle. Studying these stars further can provide insight into the behaviour of large, cool stars that our sun will eventually become. Another very important possibility with semi-regular variables is their potential use as standard candles. Standard candles calibrate distances in the universe, which allows us to

measure important quantities such as the expansion rate of the universe. The more standard candles that exist, the better the constraints and measurements of important cosmological quantities become. Therefore, understanding semi-regular variable stars may be a vital component in our understanding of the universe (Luna et al., 2021).

Runaway stars are another interesting, yet strange category of objects that occur in the universe. A runaway star has a large velocity with respect to its local interstellar medium (ISM). These stars can attain such velocities in a number of ways proposed by theories, such as gravitational interaction and ejection from star clusters (Gies and Bolton, 1986), ejection from binary systems when the companion explodes as a supernova (Hoogerwerf et al., 2001), or a gravitational slingshot effect from an interaction with a supermassive black hole (Brown et al., 2006). Like a boat travelling at high speed through water, runaway stars moving through their local ISM at extreme speeds set up a bow-shaped arc as they pass through the material.

V Canum Venaticorum (V CVn) is a semi-regular variable star located in the constellation Canes Venatici around 501 pc from Earth. In galactic coordinates that define longitude and latitude with the Sun as the origin, the star is located at about $(107.89^o, +70.77^o)$. V CVn is of spectral type M4-M6e, meaning it is a red star (Carroll and Ostlie, 2007) and, therefore, can herald information about the future of our very own star, the sun (Wenger et al., 2000; Gaia Collaboration et al., 2016; Babusiaux et al., 2023; Gaia Collaboration et al., 2023).

When one normally thinks of light, the light from the Sun often comes to mind. When it is radiated away from the sun, all light is electromagnetic (EM) waves, which vibrate in all directions perpendicular to the direction of travel. This is known as

unpolarised light. However, if the EM waves which comprise the light interact with particles and scatter, such as those in our atmosphere or gas and dust in space, one would find that the directions in which the light vibrates change. After scattering interactions, the light may only vibrate in certain directions instead of all directions perpendicular to the direction of motion. This light is said to be polarised (Young and Freedman, 2019; Bohren and Huffman, 1983). As it turns out, polarisation can be an incredibly powerful tool in the astrophysicists' kit, as it can probe and give information about the mechanics of circumstellar environments and other phenomena (Clarke, 2010a).

The curious behaviour of V CVn is detailed in Neilson et al. (2014) and Neilson et al. (2023). They take multiple decades of observations (Wolff et al., 1996; Magalhães et al., 1986; Poliakova, 1981) for V CVn and show that the brightness and polarisation have a roughly inverse relationship, with a slight lead or lag time. This is to say when V CVn reaches its maximum brightness, the polarisation nears its minimum, and when the brightness reaches a minimum, the polarisation nears its maximum value. One could imagine that, more often than not, a higher brightness leads to more photons and, therefore, more scattering events. Thus, maximum brightness should be correlated with maximum polarisation, and conversely minimum brightness with minimum polarisation. V CVn does not behave as expected with regard to these measurements. Moreover, Neilson et al. (2014) and Neilson et al. (2023) noticed that the polarisation position angle is roughly constant (see section 2.4 for the mathematical details of polarisation), which causes them to claim that this must imply the existence of an asymmetric structure which is mostly stable in time. They argue that since V CVn is a runaway star (Gaia Collaboration et al., 2016,0; Babusiaux et al., 2023) the presence of a stellar wind bow shock is likely, which is

an asymmetric mostly stable structure that can cause a large polarisation signal to manifest (Shrestha et al., 2018,0). Some other proposals attempt to explain the curious behaviour of V CVn, such as Safonov et al. (2019), who claim that there are 'blobs' around the star and that the brightness changes in V CVn may be non-radial. This would mean that when the star is at maximum brightness on the far side, the Earth observes minimum brightness, but the blob behind the star, which will see maximum brightness, will scatter a maximal amount of polarised light toward the Earth. This would imply maximum polarisation at minimum brightness. The reverse situation would occur when the star's far side is at minimum brightness, but the close side is at maximum brightness. The Earth would observe maximum brightness, but a minimum amount of polarised light would be scattered from the blob on the far side.

Although the blob theory is appealing, Neilson et al. (2023) argue that no star of V CVn's brightness variability pulsates non-radially. There are a handful of other theories involving magnetic fields, rapid rotation, *etc.*, but Neilson et al. (2023) argue that these scenarios are also unlikely due to the nature of V CVn. Thus, this thesis aims to numerically test the hypothesis that a spherically symmetric wind, which varies with time, is the cause of the inverse relationship between the polarisation and brightness of V CVn.

This work aims to address two problems. The first problem is the computation of the full range of hydrodynamics in an interaction between the stellar wind from V CVn and the ISM it is moving through. Secondly, the calculation of the polarisation signal directed toward Earth. In a simplistic sense, one may imagine this problem as a sphere which gives off a spherically symmetric gas which may vary in density,

temperature, and velocity (the star)[1], placed at a fixed location in a gas of very low density and temperature (the ISM), moving with a constant velocity.[2] The structure which forms in this scenario is known as a bow shock, and it is an illustrative exercise to follow the theoretical work of Baranov et al. (1971) and Wilkin (1996) to describe the analytic nature of a simplified case of this.

Following Baranov et al. (1971), if one assumes that in the star's direction of motion, then there exists a point where the ram pressure equalizes, which is a distance R_{SO} away from the star in the coordinates of its rest frame. The ram pressure is the momentum flux portion of Cauchy's stress tensor. Therefore, it is physically nothing more than a measure of the momentum flux in the i-direction through a surface defined by a normal vector in the j-direction. However, for the purposes being discussed, since the stellar wind meets the ISM head-on, the surface normal for this interaction is in the same direction as the momentum flux. Thus, the form that the ram pressure takes for the stellar wind is $P_w = \rho_w v_w^2$ and $P_0 = \rho_0 v_\star^2$ for the ISM. Here, ρ_w is the density of the stellar wind, v_w is the speed of the stellar wind, ρ_0 is the density of the ISM, and v_* is the speed of the ISM in the rest frame of the star. Equating the ram pressure of the stellar wind and ISM yields

$$\rho_w v_w^2 = \rho_0 v_\star^2. \tag{1.1}$$

[1]It is not a requirement that the gas ejected from the star is spherically symmetric. Rotation, magnetic fields, non-radial pulsation, and other physical phenomena would cause an asymmetry in the wind. The routines developed for this thesis can handle these cases, but it is just the most likely case that the winds from V CVn are essentially spherically symmetric. See Neilson et al. (2023) for the argument.

[2]The one-dimensional analogue of this problem for a constant wind is just a 1D shock tube problem, the Riemann problem known as the Sod Shock (Sod, 1978). Therefore, one familiar with fluid mechanics may think of the constant stellar wind case as a 2D Sod Shock in polar coordinates. It is non-trivial, but thinking about it this way can give an *a priori* idea of the type of structures one would expect in the final results!

Consider a sphere of radius R_{SO} (known as the standoff distance to the bow shock); the total mass loss of the stellar wind through a surface with a spherically symmetric density and velocity profile is

$$\dot{M}_w = 4\pi R_{SO}^2 \rho_w v_w. \tag{1.2}$$

Solving equation (1.2) for the density of the wind, ρ_w, substituting this expression into equation (1.1) and then rearranging, gives an analytic expression for the standoff distance to the bow shock,

$$R_{SO} = \sqrt{\frac{\dot{M}_w v_w}{4\pi \rho_0 v_*^2}}. \tag{1.3}$$

Moving forward, one may follow Wilkin (1996), which is a detailed calculation of momentum balance that assumes the bow shock is extremely thin compared with the distance to the star (equivalent to assuming infinitely efficient cooling). This results in an analytic expression for the shape of the bow shock in polar coordinates, namely,

$$r(\theta) = R_{SO} \csc(\theta) \sqrt{3[1 - \theta \cot(\theta)]}. \tag{1.4}$$

Figure 1.1 gives an intuition for the shape of a stellar wind bow shock according to equation (1.4).

Chapter 2 outlines the methodology used for the research. §2.1 gives background on the code Zeus3D. §2.2 details issues involving the boundary conditions and their solutions. §2.3 outlines the mathematical models of the simulations. §2.4 overviews the mathematics and implementation of linear polarisation in Zeus3D. Chapter 3 concerns the results of the work. §3.1 is the discussion, §3.2 gives all results of the

static wind velocity model of V CVn, §3.3 gives all results of the variable wind velocity model of V CVn, and §3.4 provides normalised cross-correlation functions for each model of V CVn, which elucidate the relationship between the polarisation and brightness (mass-loss rate) of V CVn over the entire dataset as opposed to small time-slices. Chapter 4 is a reiteration of the conclusions and a discussion of future work. Finally, Appendix A is the modular code and execution script for the computation of the normalised cross-correlation functions in §3.4.[3]

[3]This is the only code displayed within this thesis, as it will not be uploaded to `https://mike-power666.github.io/`.

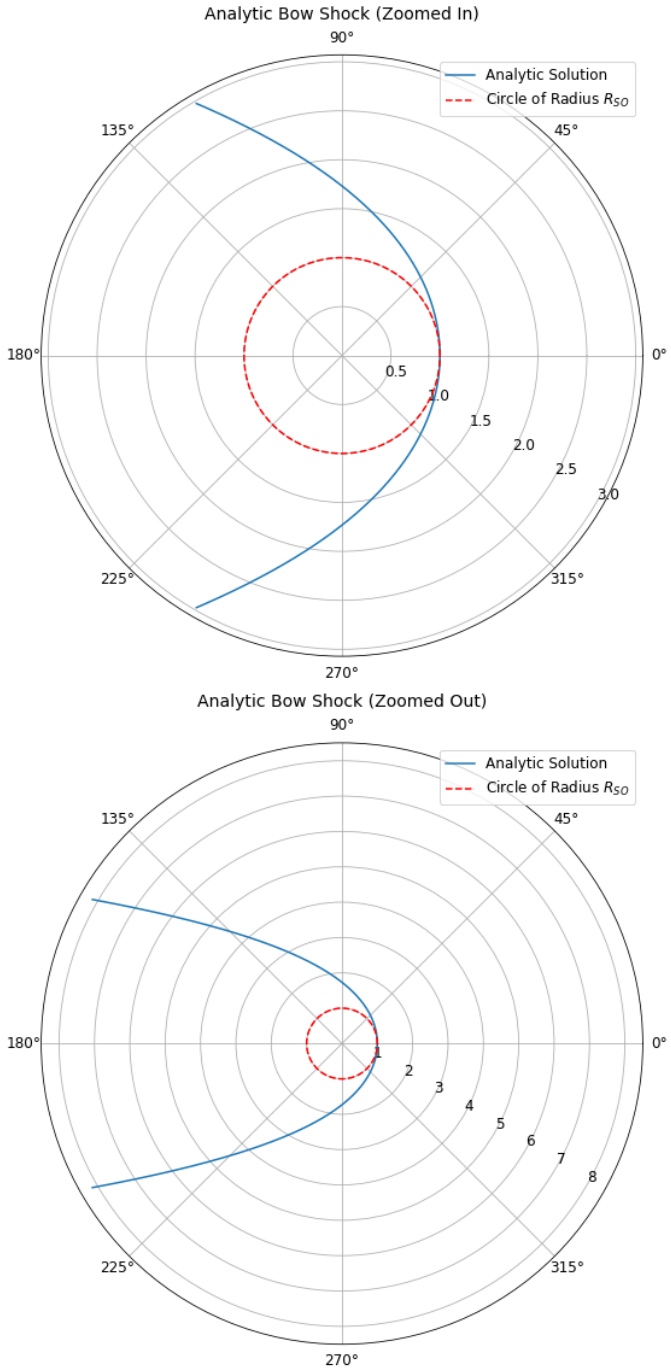


Figure 1.1: A zoomed-in (top) and zoomed-out (bottom) plot of the analytic bow shock defined by equation (1.4) with the standoff distance R_{SO} normalised to unity. One may notice that a circle of radius R_{SO} is always smaller than the analytic bow shock model, except at the apex of the bow shock. This is a visual aid to show that the approximation of a spherical wind until the point where the ram pressure of the wind and ISM equalize is acceptable.

Chapter 2

Methodology

> You don't understand something until you compute it.
>
> *Michael L. Norman*

2.1 Some Zeus3D Background

To tackle the complexities involved in modelling the physics of the solar wind from V CVn interacting with the local ISM as the star moves, numerical methods must be employed to solve the equations that govern the situation. Accordingly, this work uses Zeus3D, one of the most robust and well-tested codes in computational astrophysics.

Zeus3D is a computational (magneto)hydrodynamics code that accounts for many physical phenomena, including viscosity, gravity, molecular cooling, and ambipolar diffusion; these are just its single fluid capabilities (Clarke, 2016). The equations

which Zeus3D solves are as follows, conservation of mass:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = 0; \tag{2.1}$$

conservation of momentum:

$$\frac{\partial \vec{s}}{\partial t} + \nabla \cdot \left[\vec{s}\vec{v} + (p + p_B)\mathbb{1} - \vec{B}\vec{B} - \mu\mathbb{S} \right] = -\rho \nabla \phi; \tag{2.2}$$

conservation of flux, the induction equation:

$$\frac{\partial \vec{B}}{\partial t} + \nabla \times (\vec{E} - \beta_{\mathrm{AD}} \vec{L} \times \vec{B}) = 0; \tag{2.3}$$

and the choice of either conservation of internal energy, which keeps pressures numerically positive-definite but total energy may not be conserved to machine round off error (Clarke, 2010b):

$$\frac{\partial e}{\partial t} + \nabla \cdot (e\vec{v}) = -p \nabla \cdot \vec{v} + \mu\mathbb{S} : \nabla \vec{v} - \mathcal{L} + 2\beta_{\mathrm{AD}} L^2; \tag{2.4}$$

or the conservation of total energy, which conserves total energy to machine round off error but may cause negative pressures to manifest (Clarke, 2010b):

$$\frac{\partial e_T}{\partial t} + \nabla \cdot \left[(e_T + p - p_B)\vec{v} - \mu\mathbb{S} \cdot \vec{v} + \vec{E} \times \vec{B} + \beta_{\mathrm{AD}} B^2 \vec{L} \right] = -\mathcal{L} + \beta_{\mathrm{AD}} L^2. \tag{2.5}$$

Here, ρ represents the mass density of the fluid, t is the time, $\vec{v}$ is the velocity field, $\vec{s} = \rho\vec{v}$ is the momentum field, p is the thermal pressure, $p_B = \frac{1}{2}B^2$ is the magnetic pressure[1], $\mathbb{1}$ is the identity tensor, $\vec{B}$ is the magnetic induction (magnetic field), μ is the viscosity (Von Neumann and Richtmyer, 1950), $\mathbb{S} = \partial_j v_i + \partial_i v_j - \frac{2}{3}\delta_{ij} \nabla \cdot \vec{v}$ is

[1]Note that in Zeus3D, μ_0, the permeability of free space, is taken to be unity.

the shear tensor (Stone and Norman, 1992), ϕ is the usual Newtonian gravitational potential defined by $\nabla^2\phi = 4\pi G\rho$, $\vec{E} = -\vec{v}\times\vec{B}$ is the induced electric field, $\beta_{\rm AD}$ is the ambipolar diffusion coupling constant (Power, 2018), $\vec{L} = (\nabla\times\vec{B})\times\vec{B}$ is the Lorentz Force, e is the internal energy density, $\mathcal{L}$ is the cooling function (Raga et al., 1997), and $e_T = e + \frac{1}{2}\rho v^2 + \frac{1}{2}B^2 + \phi$ is the total energy density.

The system of partial differential equations must then be closed with an equation of state to relate the thermodynamic variables. For most applications in astrophysics, the ideal gas law is an excellent approximation of reality. Therefore, Zeus3D uses it to close the system (although other equations of state may be employed if desired). The ideal gas law used by computational astrophysicists is cast into a slightly different form than one usually encounters due to the use of intensive variables in computational (magneto)hydrodynamics. Therefore, for the sake of clarity and completeness, the ideal gas law is

$$p = (\gamma - 1)e, \tag{2.6}$$

where γ is the ratio of specific heats.

One of Zeus3D's advantages, which was convenient for this project, is the ability to add new physics or alter how the code's main (M)HD cycle operates relatively simply. Often, it is impossible with large codes to directly change the underlying (M)HD algorithm without chaos quickly ensuing, but Zeus3D comes equipped with the ability to do "microsurgery" (Clarke, 2010c) on the source code itself. The algorithmic blocks which form the source code allow the user to more easily control how new code behaves and interacts with the vast number of existing algorithms. Since the work addressed in this thesis required the development of a large amount of original code within the framework of Zeus3D, it is, therefore, essential to dive

deep into the details of how Zeus3D solves the (M)HD equations if the reader is to understand the subtleties of the numerical work comprising the majority of this thesis.

Zeus3D is a fully conservative grid code that uses a staggered mesh and an operator splitting scheme for numerical stability (Clarke, 1996).[2] For this thesis, discussing some of the finer details of the staggered mesh and the process by which Zeus3D solves the (M)HD equations will be sufficient. A quick thought experiment is helpful to elucidate the meaning of 'grid code'. Consider a physical problem in a 2D plane that is L_x units in the x-direction and L_y units in the y-direction. One creates a 'grid' to solve the equations by subdividing the region into several smaller 2D planes of length L_x/n_x and L_y/n_y, in each direction, called 'zones'. The analogy extends to 3D, and it is also not necessary for the geometry to be Cartesian, nor is it essential for the zones to be uniform. Nonetheless, Zeus3D subdivides a computational domain into many smaller zones, with their shapes and sizes determined by the overall grid geometry and the user's choice of coordinates. The complication in a grid code like Zeus3D is the staggered mesh approach. Figure 2.1 shows a typical computational 'zone' within the Zeus3D framework in 2D (but the principles easily extend to 3D or other geometries); a zone has a few defining features which must be understood. In the staggered mesh approach, Zeus3D has two grids with which one must concern themselves, the 'a-grid' and the 'b-grid'. The left-bottom-back zone corner defines a quantity located purely on the a-grid (this will be the bottom-left corner in the 2D example of Figure 2.1. In contrast, the zone centre defines a quantity located purely on the b-grid (this is still the zone centre in the 2D example of Figure 2.1). The importance of these grid distinctions comes into play during the manipulation and definition of the primitive (M)HD variables undergoing evolution on the grid.

[2]The interested reader is encouraged to explore the in-depth description of these schemes numerically within the cited reference if they wish to learn the finer details.

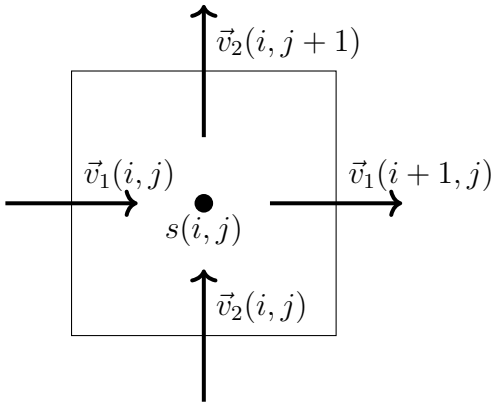


Figure 2.1: A generic representation of a 2D zone in Zeus3D. $\vec{v}_1$ represents the 1-component of a vector, $\vec{v}_2$ represents the 2-component of the same vector. The variable s represents a scalar. The notation (i, j) represents the zone drawn. Therefore, the vector components for this zone are located at the left face and the bottom face. The scalars for this zone are simply located at the centre. Note how the velocity components of adjacent zones are technically located at this zone's top and right faces, hence the designation $(i, j+1)$ and $(i+1, j)$. One must always be mindful of the zone geometries when working in Zeus3D.

It turns out that within a staggered mesh scheme, one must treat vectors and scalars differently. Pure vector quantities such as a velocity or magnetic field must have their components at the zone faces. Examining Figure 2.1, one will notice that the vector quantity, $\vec{v}$, denoting a generic vector quantity, has its 1-component located at the 1-face (the component points in the 1-direction) and has its 2-component at the 2-face (the component points in the 2-direction). Scalars, such as the temperature or internal energy density, on the other hand, are far more intuitive. They are defined at the zone centres and, therefore, lay directly on the b-grid, while vectors defined on the zone faces have a mixture of a-grid and b-grid parentage. A complication deliberately avoided here is the nature of quantities derived from vectors. Dot-product identities convert vectors into scalars, so pure b-grid coordinates define these scalars at the

zone centres (an example being the velocity divergence). However, cross-product definitions create some trouble for vectors, as they place them on the zone edges.[3] To re-iterate the crucial points, Zeus3D defines vector quantities at zone faces (so that each component is perpendicular to its respective face) and defines scalars at the zone centre. One subtlety, which becomes very important, is that the components of a vector and the magnitude of the same vector (which is a scalar quantity) reside in different locations. Thus, when position vectors are required to define a quantity like a velocity through something like an angle, θ, one must be extremely careful in correctly choosing these position vectors such that numerical consistency is maintained.

2.2 The Boundary Problem

This work's first and possibly most daunting task was representing the star V CVn numerically within Zeus3D. At the very best, if one could have a method of determining the dynamics of the solar wind as a function of time near the stellar surface, then in Zeus3D, one would require the smallest length scale to be on the order of the star's radius. The largest length scale would have to be on the order of the standoff distance to the bow shock, which manifests. Thus, the length scales within the problem would span many orders of magnitude, and dynamically important physics would be happening within each. Static or adaptive mesh refinement (Colella, 1982; Berger and Colella, 1989), which changes the size of the zones in Zeus3D according to the dynamics and associated length scales required to resolve the physics adequately, while certainly essential, would still be far too computationally costly to resolve the problem from the stellar radius to the standoff distance of the bow shock.

[3] I will avoid discussing this as it mainly concerns magnetic field quantities. However, this would become important if concerned with a hydrodynamical quantity such as vorticity!

This scale problem necessitates an approximation to work in tandem with a mesh refinement scheme. The approximation used in this work follows Mackey et al. (2021), where a 'wind bubble' resolves the wind coming from the star. Essentially, for the wind bubble approximation, one calculates the standoff distance to the bowshock, which sets the overall scale of the problem. One then determines an appropriate minimum scale for the problem, which is some factor, η, less than the standoff distance to the bow shock. One chooses this factor, η, large enough to have sufficient computational zones between the wind bubble, which now acts as a boundary condition internal to the grid, and the bow shock to resolve the necessary physics. Coupled with the wind bubble, one may introduce static or adaptive mesh refinement to save more computational resources while ensuring an adequate level of resolution to capture all physics sufficiently.

2.2.1 Boundary Conditions Within the Computational Domain

In its current form, Zeus3D cannot handle boundary conditions within a computational domain, which the wind bubble approximation to model V CVn requires. Zeus3D handles boundary conditions around the physical grid boundary perfectly and simply (including magnetic conditions). However, there is no direct or intuitive method to apply the same routines within the grid. Therefore, the answer to boundary conditions within the grid requires much understanding of how Zeus3D functions and subsequent code development. Within a computational scheme, a boundary condition is nothing more than several zones which remain at a fixed value or a prescribed value that fluctuates with time in a predetermined way. The latter of these cases is

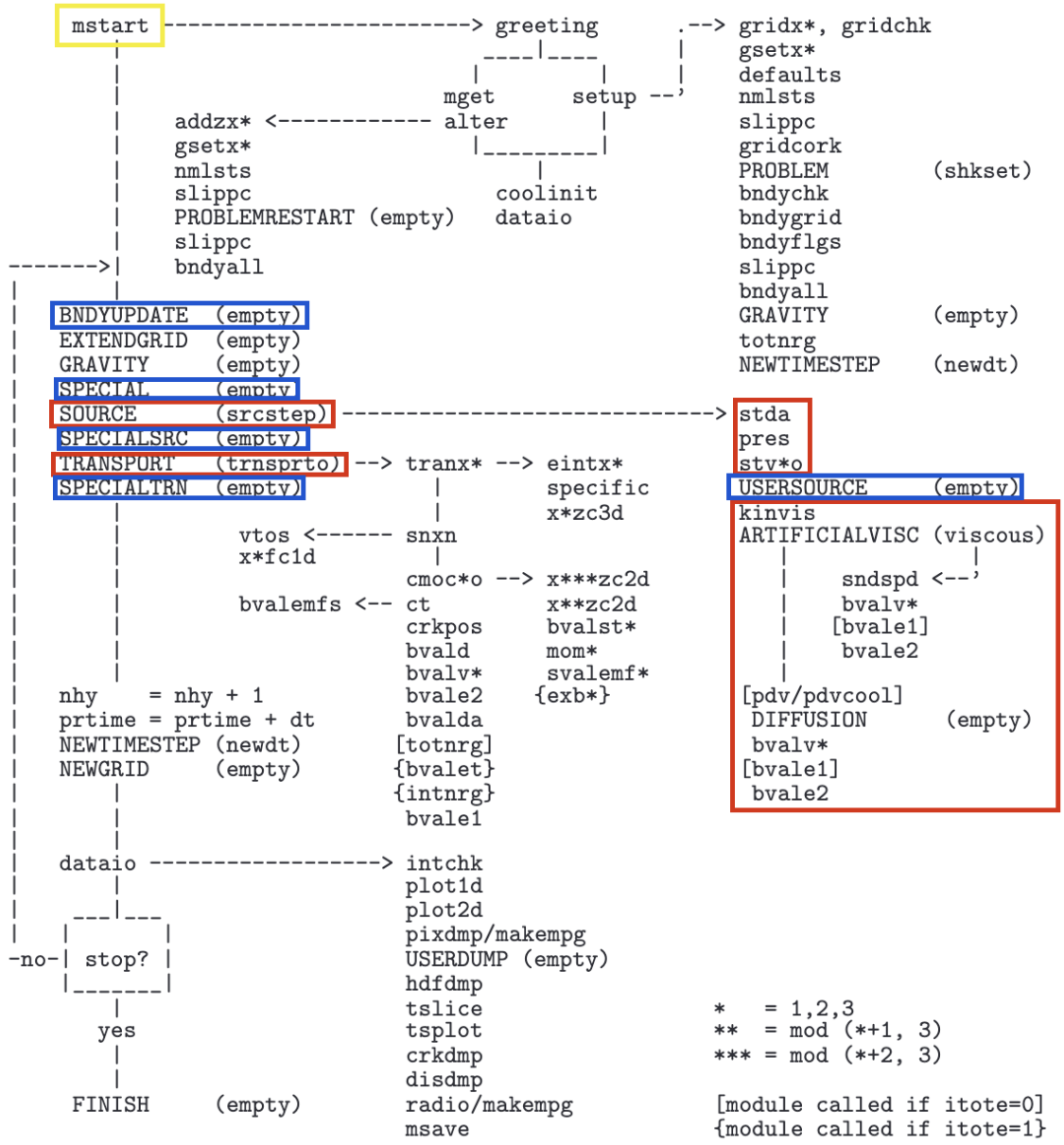


Figure 2.2: The Zeus3D flow chart ©David A. Clarke, reproduced from Clarke (2010c) with permission.

where the interests of this thesis lay. To approach the problem of defining boundary conditions internal to the grid, one must understand how Zeus3D progresses through a (M)HD cycle. Thus, examining Figure 2.2, one may follow the flow chart from the beginning, 'mstart' (yellow), downward to the beginning of the section which has been highlighted, 'BNDYUPDATE (empty)' (blue). During the pre-compilation step, a user inputs a file called 'zeus36.mac' (the Mac file), which allows the user to select the major physics involved in the simulation. For example, the Mac file controls which coordinate system is used, whether the user is doing magnetohydrodynamics or regular hydrodynamics, if gravity is turned on, and much more. More importantly for this work, Zeus3D also comes equipped with the ability to insert user-supplied subroutines into various locations within the code by aliasing the name of the subroutines within the Mac file and linking them during pre-compilation.

As a toy example to enhance understanding, if a user wished to add or take away mass from each computational zone in the simulation, this would change the continuity equation (2.1) to be

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = \dot{\rho}_{\text{toy}}. \tag{2.7}$$

Here, $\dot{\rho}_{\text{toy}}$ represents the density rate of change as a function of position and time, which the user desires. For simplicity, assume there exists a subroutine called 'mass-change.f', which removes a mass density of

$$\rho_{\text{toy}} = \dot{\rho}_{\text{toy}} \delta t \tag{2.8}$$

from every computational zone after each time step, δt.[4] Since this influences the source term of the continuity equation, the user would need to apply their routine when Zeus3D calculates the source terms for the (M)HD cycle. Following the flowchart in Figure 2.2, 'SPECIAL' (blue) is not aliased '(empty)', so the user could apply their source term here. Next, in the flowchart, 'Source' (red) is aliased to '(srcstep)'. The routines, 'srcstep', are listed by following the flow chart to the right. Starting with 'stda', this routine calculates the synchrotron age in each zone. Moving onto 'pres', this routine calculates the combinations of the different types of pressures (thermal, magnetic, *etc.*) and interpolates them to a zone centre. Next, the 'stv∗o' routines calculate the source terms, such as the pressure gradient, gravitational potential, *etc.*, for the equation of motion in the $* = 1, 2, 3$ direction. Following this, the 'USER-SOURCE' (blue) is not aliased '(empty)'. This is where the user has a choice to put their source terms if it is important for them to be influenced by the subsequent block of routines starting with 'kinvis' (red), which applies viscosity to the equations. However, after the viscosity routines, follow the flow chart back to 'SPECIALSRC' (blue), which is not aliased '(empty)'. The user also has the choice to put their routines after the viscous step but before the dynamical transport step, 'TRANSPORT' (red) '(trnsprto)'. In this toy example, what should the user do? When influencing the physics of the simulations in such a direct manner, one should never assume anything. The only way to truly know where a routine belongs is to open Zeus3D in a debugger and alias the routine to all the places it could possibly belong to one at a time. During the investigation within a debugger, one must carefully step through each (M)HD cycle, noting how the new routine influences the variables on a numerical and physical level. Then, each case must be compared to the other, and one must draw upon expertise

[4]This is not physical and would almost certainly be unstable. However, that doesn't matter, as the purpose of this exercise is to make a simple example of how Zeus3D functions with respect to changes in the (M)HD cycle.

and physical intuition (or test problems if they exist) to determine the best place to alias the routine. In the above toy example, it could be the case that the user aliases 'masschange.f' to 'SPECIAL'. After opening a debugger, the user may notice that placing their mass loss routine before the source terms can be calculated may cause numerical instabilities to manifest. Therefore, the subroutine would likely have to be aliased to another location in the Zeus3D (M)HD cycle, but to re-iterate, one should always investigate where routines which influence the (M)HD cycle belong through rigorous testing and comparisons in a debugger.

Once one understands how the (M)HD cycle of Zeus3D functions, then one can determine a method of creating boundary conditions internal to the grid. For computational efficiency, the zones defined by the wind bubble should only be reset as often as required and not anymore. However, the variable values must remain completely unchanged before and after any step in the Zeus3D (M)HD cycle which would change their values.

Once again, following the Zeus3D flow chart of Figure 2.2, there are five aliases which correspond to the locations in the (M)HD cycle where the zones of the wind bubble can be reset (blue). The first alias which appears is 'BNDYUPDATE'. On the initial (M)HD cycle, there is no need to reset the wind bubble zones, as they would have just been defined during the grid setup. However, it may be necessary on (M)HD cycles after the first. This depends upon the other aliases. The second alias which appears is 'SPECIAL'. Like 'BNDYUPDATE', this is not needed on the first (M)HD cycle, and if 'BNDYUPDATE' is aliased, then 'SPECIAL' will be redundant as it occurs directly afterwards. Thus, if 'SPECIAL' is aliased, 'BNDYUPDATE' can take

its place. Therefore, 'SPECIAL' should not be aliased. The third alias is 'USER-SOURCE', which appears after variables are updated via a source step and before they are updated with viscosity. Since the variables in the wind bubble zones will be updated by the source step governed by the 'stv∗o' routines, 'USERSOURCE' should be aliased to reset the wind bubble zones so that the source step does not numerically influence the viscous step near the wind bubble in a non-physical manner.[5] The fourth alias is 'SPECIALSRC', and as it falls after the viscous step and before the transport step, the routine to reset the wind bubble zones should be aliased to 'SPECIALSRC'. The fifth and final alias is 'SPECIALTRN', which comes after the transport step and before calculating the new time step. Since the new time step depends upon the dynamics of the entire grid due to the CFL limit[6] (Courant et al., 1928), and the transport step just changed all of the variables in the grid; the routine which resets the wind bubble zones must be aliased to 'SPECIALTRN'. Since 'BNDYUPDATE' comes directly after 'SPECIALTRN' and no physics has been applied to change the variables between these steps, it is completely unnecessary to alias the routine to 'BNDYUPDATE'.

Therefore, to minimize computational costs, the routine that resets the wind bubble zones must be aliased to 'SPECIALSRC', 'SPECIALTRN', and 'USERSOURCE' (for safety). Thus, by continuous resets of the zone variables in this manner, the wind

[5]As it turns out, during the debugging and investigations, having 'USERSOURCE' aliased to reset the wind bubble zones has no noticeable influence on the physics. So, it can be relatively safely disregarded for computational efficiency. However, it is best to err on the side of caution. There are very good reasons why this alias has no noticeable influence, but it is far too much of an unrelated side tangent in an already complicated segment.

[6]The history of the CFL limit, which governs the time step in all computational fluid dynamics grid codes to prevent numerical instabilities, is astounding. One would assume that it would have been discovered after the invention of computers, but this is not the case at all. The original paper cited is from 1928. The reader is encouraged to delve into the English translation of the original paper of Courant et al. (1956) to appreciate this almost prescient work, which came many decades before it would be used in machines yet to be invented.

bubble meets the standard of the definition of a computational boundary condition.

Both Figures 2.3 and 2.4 display what the computational domain in Zeus3D would look like for V CVn in Cartesian and spherical geometries, respectively. For Cartesian geometry, one may show, using Figure 2.3, that working in the rest frame of the star, one requires inflow conditions where the ISM will be moving toward the wind bubble representing V CVn with a velocity of $-\vec{v}_*$. Outflow conditions would be required at all other grid boundaries to allow material to flow unimpeded out of the grid and no longer influence the dynamics. These conditions are trivial in Zeus3D and require almost no thought since the inflow conditions and ISM are simply set with a velocity that a single Cartesian vector component can represent. However, in this Cartesian case, the boundary condition internal to the grid representing the wind bubble of V CVn is highly non-trivial because it requires the representation of a sphere using finite Cartesian zones, which can never be numerically lossless. These numerical inaccuracies can be mitigated, but careful thought is necessary to do so, and the methods involved with this are discussed in section 2.2.2.

In spherical geometry, one may show, using Figure 2.4, that working in the rest frame of the star, the boundary conditions for the wind bubble within the grid will be trivial. Since the wind bubble itself is defined in spherical coordinates, any vector defined within will have no issues representing its components in spherical coordinates. However, the outer boundary of the computational domain in the case of spherical coordinates is non-trivial (so is the setup of the atmosphere) since the velocity of the ISM will be defined by a Cartesian vector component requiring it to be resolved in spherical coordinates. This will cause numerical errors to manifest, but they can be mitigated. The methods involved in this mitigation are also discussed in section 2.2.2.

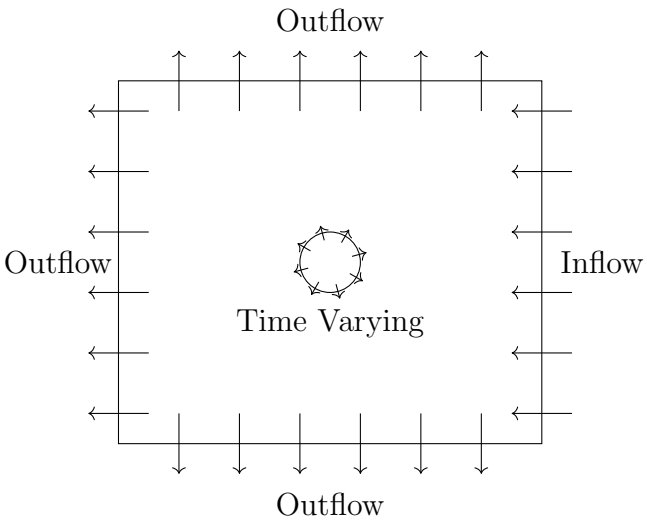


Figure 2.3: A Cartesian coordinate representation of the computational domain in Zeus3D for the case of V CVn. Here, the conditions on the boundary of the computational domain are trivial since the inflow will be along a Cartesian direction. However, the conditions within the grid, which act as the star V CVn, are not trivial since this will be a sphere subdivided into finite Cartesian cubes. This will always lead to numerical errors, and one must reduce them as much as possible.

2.2.2 Resolving Boundary Zone Geometry Conflicts

The numerical losses from zones with a mismatch between the simulation's geometry and the boundary's geometry are important to minimize. Since the work of this thesis involves supersonic flow conditions at the boundaries, any inconsistencies and numerical losses gather quickly and abruptly. Therefore, boundaries must be treated with care.

Figure 2.5 is a mock-up of a Zeus3D zone which straddles the wind bubble boundary, which is represented by the dashed circular line (blue). Zones cut by the wind bubble boundary are an issue since the portion of the zone within the boundary should have

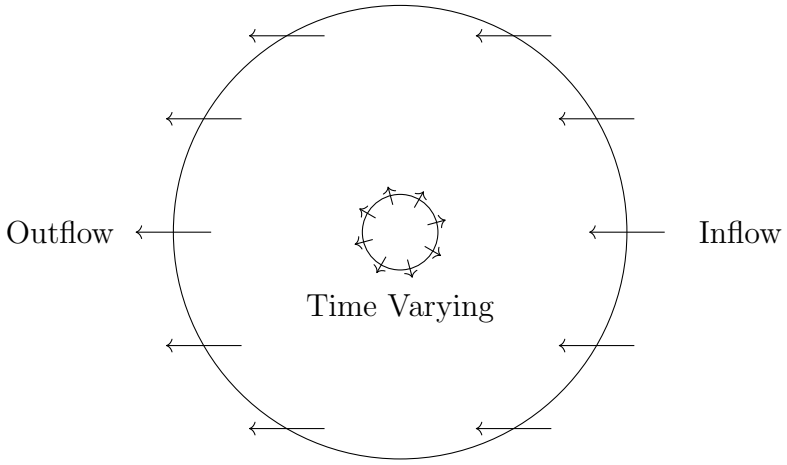


Figure 2.4: A spherical coordinate representation of the computational domain in Zeus3D for the case of V CVn. Here, the conditions on the boundary of the computational domain are non-trivial since the inflow will be along a Cartesian direction. This requires breaking a velocity vector represented in Cartesian coordinates into spherical coordinates. Numerically, in Zeus3D, this will lead to errors due to how zones are defined (see Figure 2.1), but one must reduce these errors as much as possible. However, the internal boundary is trivial since it is a sphere represented in spherical coordinates.

values corresponding to the star. Yet, the values within the same zone on the outside of the boundary should have values associated with the ISM. Since it is not possible to infinitely subdivide a zone and have a perfect boundary when the geometries mismatch, one must carefully consider the numerical methods and structure of Zeus3D zones to alleviate this problem. The first step in doing this is understanding what a value even means in Zeus3D.[7]

In Zeus3D, the fields and variables themselves are not evolved, but rather, their zone averages. A zone average is defined in terms of an integral over the volume of the

[7]Algorithmic details will *not* be discussed in detail within this thesis for the sake of brevity, as there are many thousands of lines of code with several subtleties. Suppose one wishes to learn more about the "gotchas" and nuances in the multitude of algorithms developed in Zeus3D for this work. In that case, a short letter will be released in the Journal of Computational Physics this Summer (2024), and the full code will be released in an open-source form on my GitHub, `https://mike-power666.github.io/` when the paper is released.

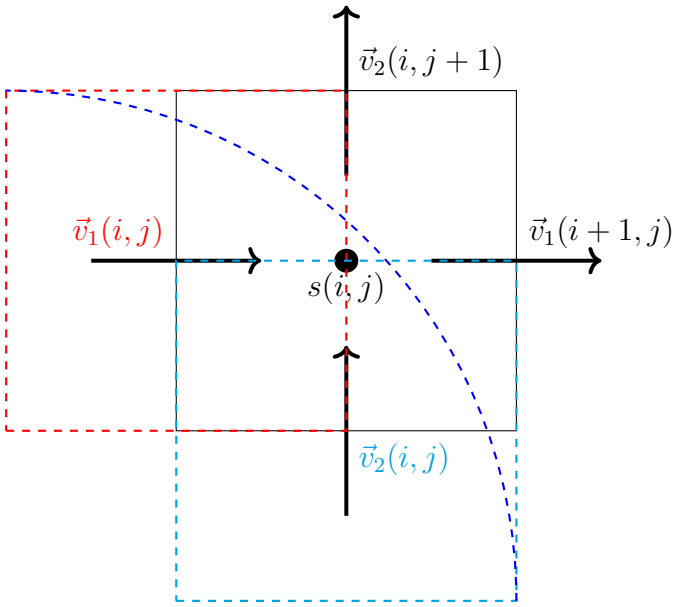


Figure 2.5: An exaggerated mock example of a Zeus3D zone (black) which straddles the wind bubble boundary (blue). Scalars are defined at the centre of the zone. The 1-components of vectors for this zone are defined on the left face, so they have an effective zone which defines them, centred at this position (red). The 2-components of vectors for this zone are defined on the bottom face, so they have an effective zone which defines them, centred at this position (cyan).

zone defined around a quantity. For scalars, this volume is about the zone centre, and for vector quantities, this volume is about the centre of the face where each vector component is defined. Thus, in general, these integrals can be written as

$$s(i,j,k) \equiv \langle s(x,y,z) \rangle = \frac{1}{V_{\text{zone}}} \iiint s(x,y,z)\, dV_{\text{zone}}, \tag{2.9}$$

for some scalar, s, and

$$\vec{v}_l(i,j,k) \equiv \langle \vec{v}_l(x,y,z) \rangle = \frac{1}{V_{\text{zone},l}} \iiint \vec{v}_l(x,y,z)\, dV_{\text{zone},l}, \tag{2.10}$$

for each component of a vector, $\vec{v}_l$. By these integrals, a zone which straddles the boundary will have a weighted average value depending on how much of the zone is internal to the boundary and how much of the zone belongs to the ISM. Therefore, the Monte Carlo method will be the simplest way to determine the correct zone-averaged quantities for zones straddling the wind bubble boundary.

Without going into deep computational detail, a Monte Carlo method randomly samples an integration zone to give an estimate of the integral, which, in its most basic and naive form (assuming a Gaussian error, which may not be true), is (Press et al., 2003)

$$\iiint f\, dV \approx V \langle f \rangle \pm V \sqrt{\frac{\langle f^2 \rangle - \langle f \rangle^2}{N}}, \tag{2.11}$$

where

$$\langle f^k \rangle = \frac{1}{N} \sum_{i=1}^{N} f^k(\vec{x}_i), \tag{2.12}$$

and $f(\vec{x}_i)$ is the integrand evaluated at the randomly sampled point $\vec{x}_i$. For this work, we know precisely the values the integrand takes. The integrand is constant in the wind bubble's interior and exterior, so it does not have any sharp, small regions where

the value changes abruptly. Thus, two options for the Monte Carlo method are used in the code. The first is simply a uniform sampling of the region with the number of points the user chooses. The second is a random sampling with the number of points the user chooses. The former is likely more accurate (due to the known geometry) but comes at a higher computational cost. At the same time, the latter is more computationally efficient and almost as accurate on average. In practice, the method the user selects doesn't really matter, as both produce relatively accurate, efficient zone averages for the wind bubble boundary.

For the other geometry of concern, namely that of spherical coordinates, one encounters another problem entirely. As mentioned, the boundary conditions in spherical coordinates internal to the grid represented by the wind bubble are trivial since the wind bubble is a sphere. However, the ISM and the outer simulation boundary have velocity vectors, which would be in the $-\hat{z}$ direction but must be defined in spherical coordinates. In spherical coordinates, the transformation representing the Cartesian unit vectors in the physicists' convention from Arfken et al. (2012) is

$$\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} \sin(\theta)\cos(\phi) & \cos(\theta)\cos(\phi) & -\sin(\phi) \\ \sin(\theta)\sin(\phi) & \cos(\theta)\sin(\phi) & \cos(\phi) \\ \cos(\theta) & -\sin(\theta) & 0 \end{bmatrix} \begin{bmatrix} \hat{r} \\ \hat{\theta} \\ \hat{\phi} \end{bmatrix}. \tag{2.13}$$

Since the velocity of the ISM in the rest frame of the star will be $\vec{v}_{\rm ISM} = -v_*\hat{z}$, the velocity which must be set at each zone in Zeus3D for the ISM and outer boundaries on a spherical grid is

$$\vec{v}_{\rm ISM} = -v_*\cos(\theta)\hat{r} + v_*\sin(\theta)\hat{\theta}. \tag{2.14}$$

As simple as this may seem, numerically, it is not so. Consider Figure 2.5 once again. One may notice that the location of the $\vec{v}_1$ and $\vec{v}_2$ components do *not* occur at the same location and will have separate values of θ due to the staggered mesh approach. Since $\theta \in [0, \pi]$, which increases counterclockwise from the z-axis in the Zeus3D simulations, the location of the 1-component of vectors will have a value, θ_1, which is less than the value, θ_2, of the location of the 2-component of vectors. Therefore, one may perform a Taylor expansion about the angle θ_1 and take the difference between the angles to be $\theta_2 - \theta_1 = \delta\theta$. Performing this Taylor expansion to first-order, one finds the velocity of the ISM (note that the angles are zone-dependent) for each zone is

$$\vec{v}_{\rm ISM} = -v_* \cos(\theta_1)\hat{r} + v_*[\sin(\theta_1) + \delta\theta \cos(\theta_1)]\hat{\theta}. \tag{2.15}$$

This is a perturbation on the velocity, which causes it to not sit perfectly in the $-\hat{z}$ direction (and it changes the direction in a way which depends upon the zone location, governed by θ_1). However, this deviation *is* dependent upon the resolution of the grid, since infinite grid resolution corresponds to $\delta\theta \to 0$. Therefore, it can be mitigated, albeit with a higher computational cost.

One might be tempted to think that the use of the zone centre as a common location so that the values of θ are the same would be best. Still, this simplistic assumption, which would ensure that there is no rotation of the ISM velocity vector, is unfortunately incorrect for reasons which are not obvious (Clarke, 2023). The reason, is due to how Zeus3D transports momentum. One can imagine neighbouring zones in the ISM near, but not necessarily on the boundary, which will not change from a certain time step, n, to the next, $n+1$, if the stellar wind dynamics are not influencing them yet. This means the momentum vector for a given zone $\vec{s} = \rho\vec{v}$ should be the same at

each time step. Of course, in Zeus3D, this must be thought of in a component manner. From Clarke (1996) (with a naive notation to exemplify the point), compressional momentum is transported as follows (using the 1-component as an example)

$$s_1^{n+1}(i,j,k) = s_1^n(i,j,k) - \frac{\mathcal{J}_{1,1}(i,j,k) - \mathcal{J}_{1,1}(i-1,j,k)}{dx_{1,b}(i)}. \tag{2.16}$$

Here,

$$\mathcal{J}_{1,1} = \frac{1}{2}[\mathcal{M}_1(i,j,k) + \mathcal{M}_1(i+1,j,k)]\overline{v_1^n(i,j,k)}^i, \tag{2.17}$$

where

$$\overline{v_1^n(i,j,k)}^i = \begin{cases} v_{1,L} & (v_{1,L} > 0 \text{ and } v_{1,R} > 0), \\ v_{1,R} & (v_{1,L} < 0 \text{ and } v_{1,R} < 0), \\ 0 & \text{otherwise}, \end{cases} \tag{2.18}$$

and

$$v_{1,L} = \frac{v_1^n(i,j,k) + \delta v_1(i,j,k)}{1 + \frac{\delta v_1(i,j,k) dt^n}{dx_{1,b}(i)}}, \tag{2.19}$$

$$v_{1,R} = \frac{v_1^n(i+1,j,k) - \delta v_1(i+1,j,k)}{1 + \frac{\delta v_1(i+1,j,k) dt^n}{dx_{1,b}(i+1)}}, \tag{2.20}$$

$$\delta v_1(i,j,k) = \begin{cases} \frac{dv_1(i,j,k) dv_1(i-1,j,k)}{dv_1(i,j,k) + dv_1(i-1,j,k)} & [dv_1(i,j,k) dv_1(i-1,j,k) > 0], \\ 0 & \text{otherwise}, \end{cases} \tag{2.21}$$

$$dv_1(i,j,k) = v_1^n(i+1,j,k) - v_1^n(i,j,k). \tag{2.22}$$

These equations generate an implicit, monotonic, up-winded, time-centered interpolation (MUTCI)[8], which in Zeus3D is typically either Van Leer interpolation (shown) (van Leer, 1977), or piece-wise parabolic interpolation (PPI) (Colella and Woodward,

[8]Note that these equations only work for Cartesian zones. One must insert the metric factors at the correct locations for the spherical zones under discussion. However, the equations get extraordinarily bulky.

1984), where $\mathcal{M}_1(i, j, k)$ represents a 1-average of the mass flux (there are similar sets of equations for the 2- and 3-transport of momentum). What it comes down to is that within the set of equations above, the velocities are sometimes estimated at the zone centre, which, if the velocity at the face used the location of the zone centre to estimate its velocity, leads to a cascade of numerical inconsistencies during interpolation. One may explore how to minimize the numerical problems discussed above using some of the algebraic methods shown and try to optimize from there. Alternatively, one may use a debugger to explore the zones numerically. Both exercises are very useful.[9] One last thing to note about the boundary conditions is that the transport step of Zeus3D (the equations above are an example) requires multiple zones in each direction to complete the interpolations. This is already taken care of in Zeus3D at the typical simulation boundaries through the inclusion of 'ghost zones', which are an extension of the boundary zones. However, for the wind bubble boundary created for this work, one must either have a mathematical prescription of the wind throughout the entire wind bubble as a function of radius, or one must assume that the wind is constant throughout the wind bubble. As long as one is only concerned with treating the wind bubble as a boundary condition, assuming that the wind is constant throughout will produce the correct results for transport.

[9]This is certainly a 'cliffhanger', but as mentioned earlier, in-depth discussion of numerical algorithms and reasoning will be limited to the code (for the sake of clarity and brevity), which will be available on `https://mike-power666.github.io/` (Summer 2024). However, this is a nice view 'under the hood' of what one must understand to know what Zeus3D is doing!

2.3 Mathematics for the Initialization of Primitive Variables

To begin a simulation in Zeus3D, or any other code for that matter, one must initialize all variables on the grid, including the boundary conditions. For pure hydrodynamics, the primitive variables are the density, ρ, the velocity, $\vec{v}$, and the internal energy, e. These primitive variables may not be directly observable by telescopes or any other device; therefore, some work is required to determine the value of each primitive variable. For the work of this thesis, there are two distinct prescriptions for the stellar wind, but both share a common average mass-loss rate. The difference between the prescriptions for the stellar wind comes down to which variables are fixed and which change with time. In the first prescription (§2.3.1), the wind velocity is fixed, and the wind density varies for the mass-loss rate to change with time. The wind velocity and density vary with time in the second prescription (§2.3.2). Each will be discussed in separate subsections since the mathematics governing each case differs.

Since V CVn is a semi-regular variable star, the mass-loss rate (taken as a proxy for the brightness) should be modelled to vary periodically with a period equivalent to the observational data of the star. Since V CVn is an AGB star, one may use the relation from De Beck, E. et al. (2010) for mass-loss rates. The relationship defines the average mass loss of an AGB star, $\dot{M}$, (in solar masses) as a function of its period, τ, (in days) as

$$\log(\dot{M}) = \begin{cases} -7.37 + 3.42 \times 10^{-3}\tau & \tau \lesssim 850, \\ -4.46 & \tau \gtrsim 850. \end{cases} \tag{2.23}$$

2.3.1 Conditions for a Static Wind Velocity

If one does not model the stellar wind directly by determining a model and then solving the governing equations for its time evolution (see Henny and Cassinelli (1999) for an example), then a periodic prescription must be assumed. For this work and the case of a static wind velocity, the assumed form of the time-varying mass-loss rate of the wind is

$$\dot{M}(t) = \alpha + \beta \sin^2(\omega t). \tag{2.24}$$

Here, α and β represent (yet to be determined) constants, $\omega = \pi/\tau$ is used so that there will be only one full variation over the period, τ, due to the use of $\sin^2(\omega t)$, which allows the mass-loss rate to vary to any degree which the user wishes. In a case where $\sin(\omega t)$ governs the variation in the mass-loss rate, the variation has a limit of twice its average (and this limit would set the minimum mass-loss rate to zero). Two more parameters are required to determine the primitive variables uniquely. For this case, the fixed speed of the stellar wind v_w will act as the first, and the second will be the level to which the mass loss varies, defined by the ratio

$$\eta = \frac{\dot{M}_{\mathrm{Max}}}{\dot{M}_{\mathrm{Min}}}. \tag{2.25}$$

To progress, one must integrate the general mass-loss rate, equation 2.24, in the sense of a time average over a full period. Doing this, one finds

$$\dot{\bar{M}} = \langle \dot{M}(t) \rangle = \frac{1}{\tau} \int_0^\tau [\alpha + \beta \sin^2(\omega t)]\, dt. \tag{2.26}$$

Since the integral of $\sin^2(\omega t)$ over a period is $1/2$, this gives the relationship

$$\dot{\bar{M}} = \alpha + \frac{1}{2}\beta. \tag{2.27}$$

From the form of equation 2.24, one may notice that the minimum possible mass-loss rate is

$$\dot{M}_{\text{Min}} = \alpha. \tag{2.28}$$

Also by inspection, the maximum possible mass-loss rate is

$$\dot{M}_{\text{Max}} = \alpha + \beta, \tag{2.29}$$

which, when combined with the definition of the mass loss ratio of equation 2.25 and equation 2.28, gives

$$\alpha + \beta = \eta\alpha. \tag{2.30}$$

Together, equations 2.27 and 2.30 define a system of equations in two unknowns, namely,

$$\begin{bmatrix} 1 & \frac{1}{2} \\ 1-\eta & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \dot{\bar{M}} \\ 0 \end{bmatrix}. \tag{2.31}$$

By Cramer's rule, the solutions are

$$\alpha = \frac{2}{\eta+1} \begin{vmatrix} \dot{\bar{M}} & \frac{1}{2} \\ 0 & 1 \end{vmatrix},$$

$$\alpha = \frac{2}{\eta+1}\dot{\bar{M}}, \tag{2.32}$$

and

$$\beta = \frac{2}{\eta+1} \begin{vmatrix} 1 & \dot{\bar{M}} \\ \eta-1 & 0 \end{vmatrix},$$

$$\beta = 2 \cdot \frac{\eta-1}{\eta+1}\dot{\bar{M}}. \tag{2.33}$$

These solutions give an equation for the mass-loss rate in terms of the known initial parameters, namely,

$$\dot{M}(t) = 2 \cdot \frac{1 + (\eta - 1)\sin^2(\omega t)}{\eta + 1} \dot{\bar{M}}. \tag{2.34}$$

To determine the primitive variables, one must utilize the mass-loss rate for the wind bubble defined in equation 1.2 with the definition of the mass-loss rate for a static wind velocity, equation 2.34. Combining these equations yields

$$\rho_w(t) = 2 \cdot \frac{1 + (\eta - 1)\sin^2(\omega t)}{\eta + 1} \cdot \frac{\dot{\bar{M}}}{4\pi R_w^2 v_w}. \tag{2.35}$$

Since, in this case, the wind velocity is already set, the last primitive variable to solve for is the internal energy of the wind, e_w. This can be accomplished by use of the ideal gas law, equation 2.6, taking the mean molecular weight of the gas to be $\bar{m}$, and the density of the wind, equation 2.35. This procedure gives

$$e_w(t) = \frac{k_B T_w}{(\gamma - 1)\bar{m}} \rho_w(t),$$

$$e_w(t) = \frac{2 k_B T_w}{(\eta + 1)(\gamma - 1)\bar{m}} \cdot [1 + (\eta - 1)\sin^2(\omega t)] \cdot \frac{\dot{\bar{M}}}{4\pi R_w^2 v_w}. \tag{2.36}$$

2.3.2 Conditions for a Variable Wind Velocity

Similar to the static wind velocity conditions, one must set an average mass-loss rate, $\dot{\bar{M}}$, the mass-loss rate ratio, η, and the average stellar wind velocity, $\bar{v}_w$. However, the stellar wind velocity for these conditions should vary with time. Thus, in a similar vein to η, one may define a ratio which governs the strength of the wind velocity variations,

$$\lambda = \frac{v_{w,\mathrm{Max}}}{v_{w,\mathrm{Min}}}. \tag{2.37}$$

For these conditions, since both ρ_w and v_w vary with time, setting the form of $\dot{M}(t)$ beforehand does not work. Rather, one must set the type of functions that determine the behaviour of ρ_w and v_w, then calculate $\dot{M}(t)$ from there.[10]

For this work, to model V CVn, the functions should vary periodically about an average value. Therefore, the functional forms of the variation are

$$\rho_w(t) = \bar{\rho}_w + \kappa \sin(\omega t), \tag{2.38}$$

and

$$v_w(t) = \bar{v}_w + \sigma \sin(\omega t). \tag{2.39}$$

Here, the oscillation frequency is defined in the usual way $\omega = 2\pi/\tau$; both κ and σ are constants yet to be determined. Also, since the integral of $\sin(\omega t)$ over a period τ vanishes, it is clear why the first term in each of the functions is their respective average value.

Three unknown values must be determined before the forms the primitive variables take can be known. They are the average value of the wind density, $\bar{\rho}_w$, κ and σ. To begin, one must use the relationship governing the strength of the wind velocity variation, equation 2.37, and solve for the unknown constant σ. Due to the sinusoidal nature of the wind velocity variation, it is clear that

$$v_{w,\mathrm{Max}} = \bar{v}_w + \sigma, \tag{2.40}$$

[10]Why choose the functional form of $\rho_w(t)$ and $v_w(t)$? Why not $\dot{M}(t)$ and $v_w(t)$ to match the initial conditions, or even $\dot{M}(t)$ and $\rho_w(t)$? It is simply an arbitrary choice but with good reason. Since the equations for a time-varying stellar wind are not being solved numerically to determine the conditions, one must choose some arbitrary functional form for the variations. As it turns out, the current choice is far simpler algebraically but no more arbitrary than any other!

and

$$v_{w,\mathrm{Min}} = \bar{v}_w - \sigma. \tag{2.41}$$

Combining equations 2.37 and 2.40, one finds that

$$\lambda v_{w,\mathrm{Min}} = \bar{v}_w + \sigma.$$

Using equation 2.41 gives

$$\lambda(\bar{v}_w - \sigma) = \bar{v}_w + \sigma,$$

$$\bar{v}_w(\lambda - 1) = \sigma(\lambda + 1),$$

$$\sigma = \frac{\lambda - 1}{\lambda + 1}\bar{v}_w. \tag{2.42}$$

Using this relationship, for completeness, one may find that

$$v_{w,\mathrm{Max}} = \frac{2\lambda}{\lambda + 1}\bar{v}_w, \tag{2.43}$$

and

$$v_{w,\mathrm{Min}} = \frac{2}{\lambda + 1}\bar{v}_w. \tag{2.44}$$

Next, one must determine the functional form of the mass-loss rate. Combining equations 1.2, 2.38, and 2.39 yields two useful forms of the equation, namely,

$$\dot{M}(t) = 4\pi R_w^2[\bar{\rho}_w + \kappa\sin(\omega t)][\bar{v}_w + \sigma\sin(\omega t)]. \tag{2.45}$$

$$\dot{M}(t) = 4\pi R_w^2[\bar{\rho}_w\bar{v}_w + (\bar{\rho}_w\sigma + \bar{v}_w\kappa)\sin(\omega t) + \kappa\sigma\sin^2(\omega t)]. \tag{2.46}$$

Similar to the previous section, integration and averaging over a period gives

$$\dot{\bar{M}} = \langle \dot{M}(t) \rangle = \frac{4\pi R_w^2}{\tau} \int_0^\tau \left[\bar{\rho}_w \bar{v}_w + (\bar{\rho}_w \sigma + \bar{v}_w \kappa) \sin(\omega t) + \kappa\sigma \sin^2(\omega t) \right] dt,$$

$$\dot{\bar{M}} = 4\pi R_w^2 (\bar{\rho}_w \bar{v}_w + \frac{1}{2} \kappa\sigma). \tag{2.47}$$

Two unknowns remain. Therefore, one must utilize the remaining initial parameter, η. To do this, the maximum and minimum values of the mass-loss rate must be known. By inspection, the maximum mass-loss rate must occur at a time when the trigonometric functions are maximized. Therefore, one finds that

$$\dot{M}_{\text{Max}} = 4\pi R_w^2 (\bar{\rho}_w + \kappa)(\bar{v}_w + \sigma). \tag{2.48}$$

There are two possible cases for the minimum value of the mass-loss rate due to the nature of the functional form. The first case occurs when the trigonometric functions vanish. The second case occurs when $\sin(\omega t) = -1$ and simultaneously $\sin^2(\omega t) = 1$. By inspecting the functional form of the mass-loss rate, one may determine a condition that governs which case must be used, namely,

$$\dot{M}_{\text{Min}} = \begin{cases} 4\pi R_w^2 \bar{\rho}_w \bar{v}_w & \kappa\sigma \geq \bar{\rho}_w \sigma + \bar{v}_w \kappa, \\ 4\pi R_w^2 (\bar{\rho}_w - \kappa)(\bar{v}_w - \sigma) & \text{otherwise.} \end{cases} \tag{2.49}$$

Since each of the variables in the above condition are *individually* positive-definite, one may determine if this condition is ever satisfied. Rewriting the condition as

$$\kappa(\sigma - \bar{v}_w) \geq \bar{\rho}_w \sigma,$$

$$\sigma - \bar{v}_w \geq \frac{\bar{\rho}_w \sigma}{\kappa}.$$

By using equation 2.42,

$$\left(\frac{\lambda-1}{\lambda+1}-1\right)\bar{v}_w \geq \frac{\bar{\rho}_w\sigma}{\kappa}.$$

$$\frac{-2}{\lambda+1} \geq \frac{\bar{\rho}_w\sigma}{\bar{v}_w\kappa}.$$

$$-2 \geq \frac{\bar{\rho}_w\sigma}{\bar{v}_w\kappa}(\lambda+1). \tag{2.50}$$

This is not true, as the right-hand side of this equality is positive-definite. Therefore, the minimum mass loss has a single value independent of any cases defined by

$$\dot{M}_{\text{Min}} = 4\pi R_w^2(\bar{\rho}_w-\kappa)(\bar{v}_w-\sigma). \tag{2.51}$$

Now, starting with the mass loss ratio, equation 2.25,

$$\eta = \frac{\dot{M}_{\text{Max}}}{\dot{M}_{\text{Min}}} = \frac{4\pi R_w^2(\bar{\rho}_w+\kappa)(\bar{v}_w+\sigma)}{4\pi R_w^2(\bar{\rho}_w-\kappa)(\bar{v}_w-\sigma)},$$

$$\eta = \frac{(\bar{\rho}_w+\kappa)(\bar{v}_w+\sigma)}{(\bar{\rho}_w-\kappa)(\bar{v}_w-\sigma)},$$

$$\eta(\bar{v}_w-\sigma)\bar{\rho}_w - \eta(\bar{v}_w-\sigma)\kappa = (\bar{v}_w+\sigma)\bar{\rho}_w + (\bar{v}_w+\sigma)\kappa,$$

$$(\bar{v}_w+\sigma-\eta\bar{v}_w+\eta\sigma)\bar{\rho}_w + (\bar{v}_w+\sigma+\eta\bar{v}_w-\eta\sigma)\kappa = 0,$$

$$[(1-\eta)\bar{v}_w+(1+\eta)\sigma]\bar{\rho}_w + [(1+\eta)\bar{v}_w+(1-\eta)\sigma]\kappa = 0.$$

Using equation 2.42 to simplify, one finds that

$$\left[(1-\eta)\bar{v}_w+(1+\eta)\frac{\lambda-1}{\lambda+1}\bar{v}_w\right]\bar{\rho}_w + \left[(1+\eta)\bar{v}_w+(1-\eta)\frac{\lambda-1}{\lambda+1}\bar{v}_w\right]\kappa = 0.$$

Multiplying through by $\lambda + 1$ and dropping the common $\bar{v}_w$ yields

$$[(\lambda+1)(1-\eta) + (1+\eta)(\lambda-1)]\bar{\rho}_w + [(\lambda+1)(1+\eta) + (1-\eta)(\lambda-1)]\kappa = 0,$$

which simplifies considerably after the expansion of terms, leading to

$$(\lambda - \eta)\bar{\rho}_w + (\lambda + \eta)\kappa = 0. \tag{2.52}$$

From earlier, equation 2.47 can be manipulated into a more useful form by substitution of equation 2.42. This gives

$$\bar{v}_w\bar{\rho}_w + \frac{1}{2}\cdot\frac{\lambda-1}{\lambda+1}\bar{v}_w\kappa = \frac{\dot{\bar{M}}}{4\pi R_w^2}. \tag{2.53}$$

The equations 2.52 and 2.53 form a system in the two unknown variables $\bar{\rho}_w$ and κ, namely,

$$\begin{bmatrix} 2 & \frac{\lambda-1}{\lambda+1} \\ \lambda-\eta & \lambda+\eta \end{bmatrix} \begin{bmatrix} \bar{\rho}_w \\ \kappa \end{bmatrix} = \begin{bmatrix} \frac{\dot{\bar{M}}}{2\pi R_w^2 \bar{v}_w} \\ 0 \end{bmatrix}. \tag{2.54}$$

By Cramer's rule, the solutions for the system are

$$\bar{\rho}_w = \frac{\lambda+1}{\lambda(\lambda+3)+\eta(3\lambda+1)} \begin{vmatrix} \frac{\dot{\bar{M}}}{2\pi R_w^2\bar{v}_w} & \frac{\lambda-1}{\lambda+1} \\ 0 & \lambda+\eta \end{vmatrix},$$

$$\bar{\rho}_w = \frac{2(\lambda+\eta)(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)}\cdot\frac{\dot{\bar{M}}}{4\pi R_w^2\bar{v}_w}, \tag{2.55}$$

and

$$\kappa = \frac{\lambda+1}{\lambda(\lambda+3)+\eta(3\lambda+1)} \begin{vmatrix} 2 & \frac{\dot{\bar{M}}}{2\pi R_w^2\bar{v}_w} \\ \lambda-\eta & 0 \end{vmatrix},$$

$$\kappa = \frac{2(\eta-\lambda)(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \cdot \frac{\dot{\bar{M}}}{4\pi R_w^2 \bar{v}_w}. \tag{2.56}$$

Note that the expression for κ, imposes a physical condition on the variables since it must be positive definite. Therefore, the initial conditions must always ensure that the mass loss ratio is greater than the wind velocity ratio. Mathematically,

$$\eta > \lambda. \tag{2.57}$$

All that remains is to write out the full expressions for the mass-loss rate as a function of time and all primitive variables. Starting with the mass-loss rate, one must combine equations 2.42, 2.45, 2.55, and 2.56 which gives

$$\dot{M}(t) = 4\pi R_w^2 \left\{ \bar{\rho}_w \bar{v}_w + \left[\frac{\lambda-1}{\lambda+1} \bar{\rho}_w \bar{v}_w + \kappa \bar{v}_w \right] \sin(\omega t) + \frac{\lambda-1}{\lambda+1} \kappa \bar{v}_w \sin^2(\omega t) \right\},$$

$$\begin{aligned} \dot{M}(t) = & \left[\frac{2(\lambda+\eta)(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} + \frac{2(\lambda+\eta)(\lambda-1)+2(\eta-\lambda)(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \sin(\omega t) \right. \\ & \left. + \frac{2(\eta-\lambda)(\lambda-1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \sin^2(\omega t) \right] \dot{\bar{M}}. \end{aligned}$$

Expansion of the coefficient of $\sin(\omega t)$ yields the mass-loss rate as a function of time in terms of the initial conditions,

$$\dot{M}(t) = 2 \cdot \frac{(\lambda+\eta)(\lambda+1) + 2\lambda(\eta-1)\sin(\omega t) + (\eta-\lambda)(\lambda-1)\sin^2(\omega t)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \dot{\bar{M}}. \tag{2.58}$$

Now, one must calculate the primitive variables. Combining equations 2.38, 2.42, 2.55, and 2.56 gives the time variation of the wind density,

$$\rho_w(t) = \frac{2(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \cdot \left[(\lambda+\eta) + (\eta-\lambda)\sin(\omega t) \right] \cdot \frac{\dot{\bar{M}}}{4\pi R_w^2 \bar{v}_w}. \tag{2.59}$$

For the sake of completeness, by inspection one can determine the maximum and minimum values of the wind density,

$$\rho_{w,\mathrm{Max}} = \frac{4\eta(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \cdot \frac{\dot{M}}{4\pi R_w^2 \bar{v}_w}, \tag{2.60}$$

and

$$\rho_{w,\mathrm{Min}} = \frac{4\lambda(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)} \cdot \frac{\dot{M}}{4\pi R_w^2 \bar{v}_w}. \tag{2.61}$$

By taking a ratio, one finds a simple relationship for the strength of the density variation

$$\frac{\rho_{w,\mathrm{Max}}}{\rho_{w,\mathrm{Min}}} = \frac{\eta}{\lambda}. \tag{2.62}$$

Combining equations 2.39 and 2.42 gives the time variation of the wind velocity,

$$v_w(t) = \bar{v}_w \left[1 + \frac{\lambda-1}{\lambda+1}\sin(\omega t)\right]. \tag{2.63}$$

Finally, combining the ideal gas law, equation 2.6, with the density variation of the wind, equation 2.59, one finds the internal energy variation of the wind,

$$e_w(t) = \frac{2k_B T_w(\lambda+1)}{[\lambda(\lambda+3)+\eta(3\lambda+1)](\gamma-1)\bar{m}} \cdot \left[(\lambda+\eta)+(\eta-\lambda)\sin(\omega t)\right] \cdot \frac{\dot{M}}{4\pi R_w^2 \bar{v}_w}. \tag{2.64}$$

2.3.3 A Numerical Curiosity

How does one determine how small to make the wind bubble? As discussed earlier, attempting to simulate a wind bubble, which is the radius of the star in question, is folly. Therefore, one must make an approximation of some kind.[11] If one wishes

[11] What I'm about to discuss I found quite strange and genuinely surprising. I did not do the math like this or think about the problem in the way I'm going to present it until one time when I was playing with changing the average mass-loss rate of the star. I was trying to decrease the mass-loss rate to decrease the density of the stellar wind and shrink the gap between the wind density and the

to capture all of the physics involved in simulations of stellar wind bow shocks, then the size of the simulation must be enough to capture the bow shock itself. Therefore, the standoff distance, $\bar{R}_{\rm SO}$, is the metric by which one chooses the scales within the simulation. With this in mind, one must decide how large to make the simulation, which, presumably, should be some multiple of the standoff distance. Similarly, one should presumably choose the radius of the wind bubble R_w as some multiple of the standoff distance to the bow shock. One should choose this multiplier such that the wind bubble is sufficiently small to capture the physics and sufficiently large so that the scales within the simulation aren't too computationally expensive.

To codify this thought process, one may define a scale factor, ζ, such that

$$\zeta = \frac{\bar{R}_{\rm SO}}{R_w}, \tag{2.65}$$

where obviously, $\bar{R}_{\rm SO} \gg R_w$. Assuming that one chooses ζ for a given simulation,[12] the radius of the wind bubble is

$$R_w = \frac{1}{\zeta}\sqrt{\frac{\dot{\bar{M}}\bar{v}_w}{4\pi\rho_0 v_*^2}}. \tag{2.66}$$

Now, one may notice that all of the primitive variables in the previous section (except the velocity, of course, that's chosen *a priori*) depend upon the ratio $\dot{\bar{M}}/R_w^2$. One may also now notice that $R_w^2 \propto \dot{\bar{M}}$. Therefore, not one of the primitive variables depends

density of the bow shock. To my surprise, changing the mass-loss rate did not change the wind's density. So, I went to pen and paper to find out why.

[12]For example, in the literature (Mackey et al., 2012; van Marle et al., 2006; Freyer et al., 2003), the size of the wind bubble is usually chosen by making it take up a small number of zones at the focal point of the simulation. This would constitute a certain ζ since the zone sizes are determined by the simulation's scale, which is the standoff distance. Generally, one chooses to make the wind bubble an order of magnitude or two smaller than the standoff distance. This means that typically, $\zeta \in [10, 100]$.

on the average mass-loss rate of the star. One may recalculate the formulae for the dependent primitive variables from the previous sections to be:

$$\rho_w(t) = \frac{2\rho_0\zeta^2}{\eta+1}\cdot[1+(\eta-1)\sin^2(\omega t)]\cdot\left(\frac{v_*}{v_w}\right)^2; \tag{2.67}$$

$$e_w(t) = \frac{2k_BT_w\rho_0\zeta^2}{(\eta+1)(\gamma-1)\bar{m}}\cdot[1+(\eta-1)\sin^2(\omega t)]\cdot\left(\frac{v_*}{v_w}\right)^2; \tag{2.68}$$

for the static wind velocity configuration and

$$\rho_w(t) = \frac{2\rho_0\zeta^2(\lambda+1)}{\lambda(\lambda+3)+\eta(3\lambda+1)}\cdot\Big[(\lambda+\eta)+(\eta-\lambda)\sin(\omega t)\Big]\cdot\left(\frac{v_*}{v_w}\right)^2; \tag{2.69}$$

$$e_w(t) = \frac{2k_BT_w\rho_0\zeta^2(\lambda+1)}{[\lambda(\lambda+3)+\eta(3\lambda+1)](\gamma-1)\bar{m}}\cdot\Big[(\lambda+\eta)+(\eta-\lambda)\sin(\omega t)\Big]\cdot\left(\frac{v_*}{v_w}\right)^2; \tag{2.70}$$

for the variable wind velocity configuration.

Indeed, due to the numerics, the primitive variables do not depend upon the average mass-loss rate of the star. One may wonder if this is physically reasonable. On the surface, no, of course not. However, the simulation still depends heavily on the mass-loss rate of the star; it's just at a much deeper level. As mentioned, the standoff distance to the bow shock sets the scale for the simulation, which means that the mass-loss rate directly impacts the overall scale of the simulation. This, in turn, implies that *any* non-ideal effect such as viscosity (even numerical grid viscosity) will influence the dynamics according to the dimensionless number associated with the effect. For the example of numerical viscosity, this would be the Reynolds number, which depends upon the characteristic length scale of the flow. The time scale also depends upon the length scale through the simulation crossing time of the star. Also, turbulence depends upon the flow's characteristic length scales (Landau and Lifshitz,

1987). Therefore, though it is not obvious on the surface, the mass-loss rate still has a physical meaning and influence upon the dynamics, physical interpretation, and results of the simulations.[13]

2.4 A Geometric View of Polarisation

This section follows the seminal work of Brown and McLean (1977) and Brown et al. (1978) but clarifies some of the geometry, calculations, and understanding during the derivations while also proving that the results of the first and second papers are equivalent under certain assumptions and a particular geometric orientation.

Figure 2.6 introduces the geometry used in Brown and McLean (1977). Here, the unprimed coordinate system represents a set of coordinates centred on the astronomical object of interest, where the z-axis is the symmetry axis (Brown and McLean (1977) assumes azimuthal (ϕ) symmetry). The primed coordinate system represents a set of coordinates centred on the astronomical object, but with the x' axis pointing in the direction of Earth. The y and y' axes coincide; therefore, the plane made with the z' and y' axes represents the plane of the sky on Earth. The angle of inclination, i, is defined between the x and z' axes. Therefore, the un-primed coordinate system is related to the primed coordinate system via a passive rotation of $\xi = \pi/2 - i$ clockwise about the y axis when viewed looking in the $+\hat{y}$ direction from the origin.

To explain this geometry further and its relation to Earth, one should perform the following thought experiment. Consider a flat disc which is symmetric about the z axis (this is to say, the normal vector defining the plane of the disc is $\hat{z}$). If the axial

[13] Despite this justification, it certainly still feels weird and non-intuitive at best...

inclination is $i = 0$, then the direction to Earth (the x' axis) points along the $-\hat{z}$ axis, meaning the disc is in the plane of the sky. Therefore, one would see a circle projected onto the plane of the sky from Earth. Conversely, if the axial inclination is $i = \pi/2$, the direction to Earth would point in the $\hat{x}$ direction, meaning that the axis of symmetry of the disc is in the plane of the sky, so one would view the disc from Earth edge on. This would look like a straight line. The latter case in this thought experiment is equivalent to V CVn. The motion of V CVn with respect to Earth is essentially completely tangential. It has a negligible radial component to its velocity when considering the magnitude of the tangential velocity and the observational errors. Therefore, taking the direction of motion of V CVn to be along the $\hat{z}$ direction, this has to be in the plane of the sky to keep all of its velocity tangential. Therefore, the axial inclination of V CVn is roughly $i = \pi/2$.

Equation (1) from Brown and McLean (1977) comes from applying the coordinate rotation mentioned above. One must solve for the scattering angle, χ, which is the angle between a radial vector, $\hat{r}$, to any point in an envelope and the Earth, which is in the $\hat{x}'$ direction. Since the important quantity is the cosine of the scattering angle, one may make clever use of the properties of a dot product. Therefore, it is sufficient to find $\cos(\chi) = \hat{r} \cdot \hat{x}'$. From Arfken et al. (2012),

$$\begin{bmatrix} \hat{r} \\ \hat{\theta} \\ \hat{\phi} \end{bmatrix} = \begin{bmatrix} \sin(\theta)\cos(\phi) & \sin(\theta)\sin(\phi) & \cos(\theta) \\ \cos(\theta)\cos(\phi) & \cos(\theta)\sin(\phi) & -\sin(\theta) \\ -\sin(\phi) & \cos(\phi) & 0 \end{bmatrix} \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix}. \tag{2.71}$$

This gives the radial unit vector in terms of the Cartesian set in the unprimed system,

$$\hat{r} = \sin(\theta)\cos(\phi)\hat{x} + \sin(\theta)\sin(\phi)\hat{y} + \cos(\theta)\hat{z}. \tag{2.72}$$

Now, one must determine the value of $\hat{x}'$ in terms of the unprimed Cartesian set. To do this, one may exploit the definition of the primed coordinate system, which, as mentioned, is a passive rotation of $\xi = \pi/2 - i$ clockwise about the y axis when viewed looking in the $+\hat{y}$ direction from the origin. Mathematically, since rotation matrices are orthogonal,

$$\hat{x}' = \mathcal{R}_y^{-1}(\tfrac{\pi}{2} - i)\hat{x} = \mathcal{R}_y^{\intercal}(\tfrac{\pi}{2} - i)\hat{x},$$

$$\hat{x}' = \begin{bmatrix} \cos(\frac{\pi}{2} - i) & 0 & \sin(\frac{\pi}{2} - i) \\ 0 & 1 & 0 \\ -\sin(\frac{\pi}{2} - i) & 0 & \cos(\frac{\pi}{2} - i) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

By exploiting the even and odd nature of the trigonometric functions, as well as a shift by a quarter period, one finds that

$$\hat{x}' = \begin{bmatrix} \sin(i) & 0 & \cos(i) \\ 0 & 1 & 0 \\ -\cos(i) & 0 & \sin(i) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix},$$

$$\hat{x}' = \begin{bmatrix} \sin(i) \\ 0 \\ -\cos(i) \end{bmatrix},$$

$$\hat{x}' = \sin(i)\hat{x} - \cos(i)\hat{z}. \tag{2.73}$$

Thus, one may now determine the cosine of the scattering angle by a dot product, reproducing equation (1) from Brown and McLean (1977), namely,

$$\cos(\chi) = -\cos(\theta)\cos(i) + \sin(\theta)\sin(i)\cos(\phi). \tag{2.74}$$

The next few steps in Brown and McLean (1977) follow trivially because the density distribution is assumed to have azimuthal (ϕ) symmetry. Therefore, under an integral $\cos(\phi) \to 0$, and $\cos^2(\phi) \to \pi$. After this, everything else in Brown and McLean (1977) follows by utilizing the rotation matrix relating the coordinate systems defined above with some trigonometric and geometric tricks. Thus, with the geometry clarified, it is left as an exercise for the reader to derive the important results, equations (19), (22), and (23)[14] from Brown and McLean (1977) cast into a more useful form for Zeus3D, namely,

$$\gamma_p = \frac{\int_0^\infty \int_0^\pi \rho(r,\theta) \cos^2(\theta) \sin(\theta)\, dr d\theta}{\int_0^\infty \int_0^\pi \rho(r,\theta) \sin(\theta)\, dr d\theta}, \tag{2.75}$$

$$\bar{\tau} = \frac{\pi \sigma_0}{\bar{m}} \int_0^\infty \int_0^\pi \rho(r,\theta) \sin(\theta)\, dr d\theta, \tag{2.76}$$

$$P_R \simeq \bar{\tau}(1 - 3\gamma_p) \sin^2(i). \tag{2.77}$$

Here, γ_p is a geometric factor which governs the polarisation due to scattering in a density field with azimuthal symmetry, $\rho(r,\theta)$, $\bar{m}$ is the average particle mass, σ_0 is related to the wavelength independent Thomson scattering cross section, $\sigma_T = 6.66 \times 10^{-25}\,\mathrm{cm}^2$, by $\sigma_0 = 3\sigma_T/(16\pi)$, and $\bar{\tau}$ is effectively the optical depth of the envelope averaged over the angular coordinate.

The most important quantity is 2.75 since it effectively governs the polarisation signal of the density field. Consider a purely radial density field (a completely spherically

[14]Note that my equation 2.76 differs from Brown and McLean (1977) by a factor of 2. I can only assume that there is a typo in their equation (22), which also causes a typo in equation (21). Perhaps they meant to define the optical depth with a coefficient of 3/16, not 3/32. Either way, there is an inconsistency in either their definition of the average optical depth or the equation for the residual polarisation. One can prove this by working the simple algebra to derive equation (21) from (5) and then (23) with definition (22) from Brown and McLean (1977). For the remainder of this thesis, I will assume that the definition of the average optical depth is incorrect (missing a factor of 2) and that the polarisation equation is correct. However, either choice of changing the residual polarisation by a factor of 2 or changing the definition of the average optical depth by a factor of 2 results in self-consistency among the algebraic equations.

symmetric configuration). The integrals are separable, which causes the integral over the radial coordinate of the density to divide out, and one is left with integrals over the polar angle. The numerator has a value of 2/3, and the denominator integrates to 2. Therefore, the gamma value for a spherically symmetric density field is 1/3. Now, examining equation 2.77, one may notice that the residual polarisation varies as $1 - 3\gamma_p$. This means that a perfectly spherical configuration causes no polarisation signal. In general, from this analysis, one may note that *symmetric structures about the symmetry axis in the density profile tend to make the polarisation signal vanish, whereas asymmetric structures about the symmetry axis in the density profile will tend to produce a polarisation signal.*

All of the calculations of Brown and McLean (1977) only consider the azimuthal symmetry case, meaning the density distributions are symmetric about the z axis (the direction of motion for V CVn). While this is extremely useful for two-dimensional calculations, one must go further to calculate the polarisation signal from a full three-dimensional density profile. Brown et al. (1978) provide the exact calculation for the case of axial inclination $i = \pi/2$. In other words, when the direction of motion of V CVn is in the plane of the sky. Conveniently, as previously discussed, this is precisely the case for V CVn. Thus, from Brown et al. (1978), one acquires the ingredients to calculate the Stokes parameters and, therefore, the residual polarisation, P_R, and associated position angle, ψ. The equations are as follows, the intensity Stokes parameters:

$$I_1 = \frac{I_0\sigma_0}{\bar{m}} \int_0^{\infty} \int_0^{\pi} \int_0^{2\pi} \rho(r,\theta,\phi)[1 + \sin^2(\theta)\cos^2(\phi)] \sin(\theta)\, dr d\theta d\phi; \tag{2.78}$$

$$I_2 = \frac{I_0\sigma_0}{\bar{m}} \int_0^{\infty} \int_0^{\pi} \int_0^{2\pi} \rho(r,\theta,\phi)[\sin^2(\theta)\sin^2(\phi) - \cos^2(\theta)] \sin(\theta)\, dr d\theta d\phi; \tag{2.79}$$

$$I_3 = \frac{I_0 \sigma_0}{\bar{m}} \int_0^{\infty} \int_0^{\pi} \int_0^{2\pi} \rho(r, \theta, \phi)[\sin(2\theta)\sin(\phi)]\sin(\theta)\, dr d\theta d\phi; \tag{2.80}$$

with:

$$I = I_0 + I_1; \tag{2.81}$$

the normalised Stokes parameters:

$$Q = \frac{I_2}{I}; \tag{2.82}$$

$$U = \frac{I_3}{I}; \tag{2.83}$$

the residual polarisation:

$$P_R = \sqrt{Q^2 + U^2}; \tag{2.84}$$

and the polarisation position angle, ψ, defined by:

$$\tan(2\psi) = \frac{U}{Q}. \tag{2.85}$$

Here, I_0 represents the undimmed light of the star. The claim made by Brown et al. (1978) is that these generalized three-dimensional results for the Stokes parameters collapse to the simpler case of Brown and McLean (1977) under certain assumptions. Thus, if one assumes for V CVn that the direction of motion is in the plane of the sky, $i = \pi/2$, the net polarisation is small so that $I = I_0 + I_1 \simeq I_0$ and the density field is symmetric about the azimuthal coordinate (ϕ, which is the critical assumption of Brown and McLean (1977)), one finds for equations 2.79 and 2.80 that

$$I_2 = \frac{I_0 \sigma_0 \pi}{\bar{m}} \int_0^{\infty} \int_0^{\pi} \rho(r, \theta)[\sin^2(\theta) - 2\cos^2(\theta)]\sin(\theta)\, dr d\theta,$$

since $\sin^2(\phi) \to \pi$ under the azimuthal integral, and

$$I_3 = 0,$$

since $\sin(\phi) \to 0$ under the azimuthal integral. From these intensity Stokes parameters, one finds that $U = 0$ (expected for axis-symmetry (Shrestha et al., 2018)) and

$$Q \simeq \frac{I_2}{I_0} = \frac{\sigma_0 \pi}{\bar{m}} \int_0^\infty \int_0^\pi \rho(r,\theta)[\sin^2(\theta) - 2\cos^2(\theta)] \sin(\theta)\, dr d\theta,$$

$$Q \simeq \frac{\sigma_0 \pi}{\bar{m}} \int_0^\infty \int_0^\pi \rho(r,\theta)[1 - 3\cos^2(\theta)] \sin(\theta)\, dr d\theta.$$

$$Q \simeq \frac{\sigma_0 \pi}{\bar{m}} \left[\int_0^\infty \int_0^\pi \rho(r,\theta) \sin(\theta)\, dr d\theta \right] \left[1 - 3 \frac{\int_0^\infty \int_0^\pi \rho(r,\theta) \cos^2(\theta) \sin(\theta)\, dr d\theta}{\int_0^\infty \int_0^\pi \rho(r,\theta) \sin(\theta)\, dr d\theta} \right].$$

Using the definitions of equations 2.76 and 2.75, one gets

$$Q \simeq \bar{\tau}(1 - 3\gamma_p). \tag{2.86}$$

Since $U = 0$, equation 2.84 gives the condition that $P_R = |Q|$. Thus,

$$P_R \simeq |\bar{\tau}(1 - 3\gamma_p)|, \tag{2.87}$$

which is precisely the absolute value of equation 2.77 when the axis of symmetry (the direction of motion of V CVn) is in the plane of the sky ($i = \pi/2$) as desired.

2.4.1 Implementation of Polarisation in Zeus3D

The implementation of the polarisation calculations in Zeus3D is quite straightforward. If one works in two dimensions, then Zeus3D will write the values of $\gamma_{p,u}$ and

$\gamma_{p,l}$ (the subscripts u and p here represent upper and lower, corresponding to the numerator and denominator of γ_p respectively) assuming azimuthal symmetry to a file at each time step after the hydrodynamic cycle completes. If one works in three dimensions, Zeus3D will write the values of the normalised Stokes parameters Q and U to a file at each time step after the hydrodynamic cycle completes.

The user has three choices for computing the integrals that define the polarisation parameters, each with strengths and weaknesses. The first option has the highest numerical accuracy and computational cost. If the user desires, an algorithm for Romberg integration (Press et al., 2003) (including a Richardson Extrapolation step (Richardson, 1911; Richardson and Gaunt, 1927)) has been implemented to solve each of the integrals defining the polarisation. If the user chooses Romberg integration as the calculation method, they must also include a maximum error allowed by the integration routines. Therefore, one should proceed with caution in using this method, as in a simulation with 10^6 hydrodynamic cycles, the Romberg integrator will be called 10^6 times. If the maximum error the user selects is too small, the computational cost will be extremely high. If one is not careful, this can consume most of the computational time. The second option available to the user is a Monte Carlo integrator (Press et al., 2003). These routines are severely more efficient than a Romberg integrator but will produce far less accurate results. It is not recommended to use the Monte Carlo integration scheme. While it does include importance sampling, it is not necessarily wise to use a Monte Carlo integrator on a density field that has shocks that change location with time. This is because shocks are incredibly thin regions, which will contribute a large value to the integrals. Since Monte Carlo methods are based on random sampling, it is not always guaranteed to include small regions with

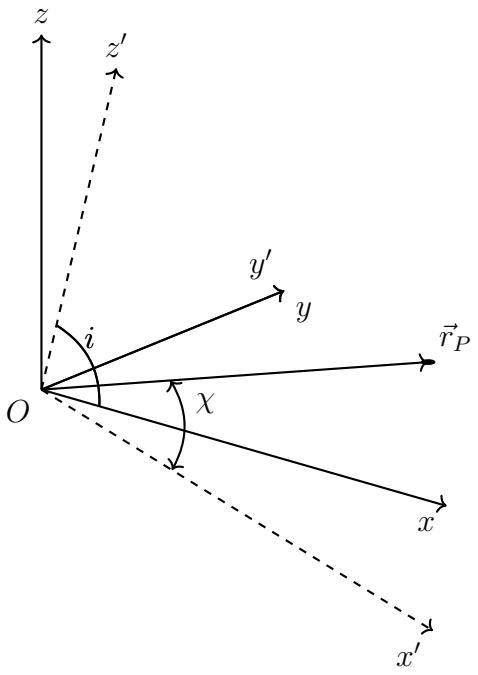


Figure 2.6: The important aspects of the geometry defined in Brown and McLean (1977). The plane of the sky is the plane defined by $\hat{y}$ and $\hat{z}'$. Earth is toward $\hat{x}'$. The angle, i, defines the axial inclination, which is $\angle xOz'$. Finally, the scattering angle, χ, is defined as the angle between the radial vector, which defines an arbitrary scattering point, $\vec{r}_P$, and the vector defining the direction to Earth, $\hat{x}'$.

high contributions. The third option available to the user is the most computationally efficient but comes with some inaccuracy. It is a simple Riemann sum, using the computational zones as the integration regions. Therefore, $dr \rightarrow x_{1a}(i+1) - x_{1a}(i)$, $d\theta \rightarrow x_{2a}(j+1) - x_{2a}(j)$, and $d\phi \rightarrow x_{3a}(k+1) - x_{3a}(k)$ are the differentials approximated by finite differences of their values defined by the a-grid in Zeus3D, and pure b-grid coordinates will determine the function values. In practice, since the Zeus3D variables are defined as zone averages, this is essentially the same as Monte Carlo integration if it selected fixed points as opposed to random points. Therefore, this scheme ensures the capturing of shocks and any other small in size but large in contribution region of the integrand.

Chapter 3

Results and Discussion

> The first principle is that you must not fool yourself—and you are the easiest person to fool.
>
> *Richard P. Feynman*

3.1 Discussion

Tables 3.1 and 3.2 detail the initial values for the constant and variable wind velocity models, respectively. In both simulations, the stellar wind is spherically symmetric and has a mass-loss rate which varies periodically. The star moves directly along the z-axis (from left to right in the simulations). In the rest frame of the star, this implies that the ISM is moving with the star's velocity but in the opposing direction. Therefore, the ISM is moving from right to left in the simulations.

Figures 3.1 through 3.12 represent the time evolution of the static wind velocity case, while Figures 3.13 through 3.24 represent the time evolution of the variable wind velocity case. The interaction between the wind from V CVn and the oncoming

ISM produces an incredibly rich structure and physics, far more complicated than the analytic bow shock model would indicate. By looking at the $\theta = 0$ slice plots (Figures 3.2, 3.5, 3.8, 3.11, for the static wind speed case; 3.14, 3.17, 3.20, 3.23, for the variable wind speed case), one may discern the different regions of the interaction. The three vertical (cyan) lines in each of the $\theta = 0$ slice plots represent the minimum ($R_{\mathrm{SO,\ Min}}$), average ($\bar{R}_{\mathrm{SO}}$), and maximum ($R_{\mathrm{SO,\ Max}}$) standoff distances calculated from the analytic bow shock model using the minimum, average, and maximum values of the mass-loss rate, $\dot{M}(t)$, and the velocity of the wind, $v_w(t)$.

Moving from left to right in the $\theta = 0$ slice plots, the flat line for each variable represents the prescribed values of the variables in the wind bubble boundary (hence the flat line). However, one may notice that in the time evolution of the slice plots, the values in the wind bubble region change. This is expected as the boundary values are time-dependent. Immediately following the wind bubble boundary zone is a region where the flow tends to get less dense and cool, and thermal pressure decreases while the velocity increases. This is a rarefaction fan. In this region, the supersonic wind can spread out and rarefy unimpeded; one may think of this as the opposite of a compression region. The oscillations in the rarefaction fan are due to the oscillations in the boundary conditions. In the case of a stellar wind that is constant in time, one would see a smooth, non-oscillatory curve connecting the wind bubble region with the next region of the interaction. The time variations in the boundary conditions perturb this structure, and the perturbations continue through the rarefaction fan. The next region of the interaction begins at the discontinuity, which occurs in all of the variables. This discontinuity is the reverse shock. Here, the material being driven by the stellar wind is compressed, which vastly increases its density and thermal pressure, heated, which causes the temperature to change by orders of magnitude, and

decelerated to a locally sub-sonic speed. The region to the right of the reverse shock is where the heated and compressed stellar wind material gathers.

Further to the right of the reverse shock is a curious, thin region most easily noticed as a discontinuity in density. Though the density is discontinuous, the thermal pressure is *not*. This indicates a contact discontinuity, which separates the material driven by the stellar wind from the material coming from the ISM. No material may be transferred across a contact discontinuity, as they have the property that the velocity normal to the contact surface vanishes. One may notice by looking at the 1-velocity that this is indeed the case since for the $\theta = 0$ slices, the 1-direction is the normal direction. Following to the right of the contact discontinuity is a highly compressed, hot region of ISM gas. This region comprises ISM material swept up by the forward shock, which is the discontinuity in all variables to the right. Once again, one may notice that from the ISM (to the right of the forward shock), the shock sweeps up this material, heating and compressing it, which changes its density, thermal pressure, and temperature by vast amounts. Also, locally, the region between the contact and forward shock is sub-sonic. Of course, to the right of the forward shock is the yet-to-be-disturbed ISM.

The heat maps show the density, thermal pressure, temperature, velocity divergence, Mach number, and flow speed from left to right and top to bottom, respectively. Of particular note are the plots of velocity divergence and Mach number. Though they may not seem illuminating initially, the velocity divergence is an excellent measure of the location of shocks since they cause rapid compression of the material. Anywhere a sudden compression of material exists, the velocity divergence will take on

a large (relatively speaking) negative value. Thus, regions in the heat map of velocity divergence with the highest absolute value will represent the shock structure of the interaction. The Mach number, defined as the ratio of the local gas speed to sound speed, showcases sub-sonic and super-sonic flow regions. Since the heat map is logarithmic, zero represents the local sonic transition. Therefore, in areas with a logarithmic Mach number less than zero, the flow is sub-sonic, while in areas with a logarithmic Mach number greater than zero, the flow is super-sonic. One may notice that sub-sonic flow regions typically occur with a high absolute velocity divergence. This is the case, as the shocks normally cause the local sound speed to rise drastically while decelerating the flow, thereby causing the Mach number to decrease.

Once again, observing the heat maps for both cases, particularly the density, thermal pressure, and velocity divergence, one may notice an extremely interesting dynamic structure in the left half of the simulation, behind the star, with respect to its direction of motion. The velocity divergence and thermal pressure plots show that a rarefied region of extremely low thermal pressure develops around the star, surrounded by an asymmetric shock structure which connects to the reverse shock in the front of the star. As time progresses, the star forms a comet-like tail, as one may see by examining the density heat maps. Even more interesting in the region behind the star is the presence of weaker oblique shocks and a cascade of even weaker shocks that change dynamically as the simulation evolves. One may see this structure by following the evolution of the velocity divergence through time. One may notice that by examining the density and temperature plots, the contact discontinuity that separated the stellar wind from the ISM at the front of the star persists into the tail. The hot swept-up ISM material, which travels through the forward shock, is dynamically transported backwards into the outer envelope of the tail. In contrast, the stellar wind material

which passes through the reverse shock is transported into the cometary structure. As this material moves through the cometary structure, it then passes through the weaker oblique shocks, but it is still mostly supersonic. Here, a very interesting mixing occurs, which can be most easily noticed in the Mach number, density, and speed plots. The material passing through the oblique shock, still mostly supersonic, forms a shear layer with the subsonic gas, which passes through the shock structure directly behind the star. Due to the density, temperature, and relative velocity differences at this interface, as time progresses, this interface becomes unstable due to the shear flow, and Kelvin-Helmholtz instabilities (Drazin, 2002) develop and begin to mix the layers. As this mixing occurs, the weaker cascading shocks travelling through this tail region sweep up and mix the Kelvin-Helmholtz unstable regions even more. All of the dynamic features occurring within the tail region of the interaction cause a large amount of turbulence in the flow. Due to its stochastic nature, this turbulence will manifest as an asymmetry in three-dimensional simulations.[1]

Below the slice plots is a plot of the polarisation as a percentage per optical depth and the star's mass-loss rate, which acts as a proxy for its brightness. The polarisation increases downward while the mass-loss rate increases upward. This is to help show the inverse relationship between the two curves. It is notable that, even as time passes by following the plots in either model, the star's polarisation and mass-loss rate exhibit an inverse relationship. The reason for this inverse relationship comes from the structure of the density data. One may notice a few details by examining both the density heat maps and the density from the slice plots. The material in

[1]Although I've been as descriptive as possible, one cannot fully appreciate and understand the rich hydrodynamic structure within the cometary tail of the star from the time slice images alone. A lot is going on, and it's hard to keep track. To enrich the experience and fully understand the structure, one should visit `https://mike-power666.github.io/` and examine the animations of the interaction.

the rarefaction fan nearest the wind bubble has extremely high density in all plots. This is because the star has a large mass-loss rate with a low wind speed. Looking at the heat map, one may notice that this region, especially nearest the wind bubble, is quite spherical. Thus, in calculating the polarisation, this region, having a high density and spherical shape, will tend to decrease the polarisation signal, pushing it toward a vanishing spherical value. Moving further from the wind bubble boundary, yet staying in the rarefied region both in the direction of motion and in the tail, the density continues to decrease, and this region has a prolate shape, meaning it is elongated along the z-axis, the direction of motion. This manifests as an asymmetry that contributes to a polarisation signal. Next, one encounters the reverse shock, which compresses the stellar wind material and causes the density to be higher than some parts in the rarefaction fan.

Since a bow shock is, by nature, an asymmetric structure, this shocked region contributes the majority of the asymmetry in the density profile and thereby contributes the highest polarisation signal, pushing it away from zero. Also of interest is the cometary structure of the tail, which has a shock structure that varies with time in all cases. Due to this, the tail has non-negligible density, contributing to the polarisation in a capacity that depends upon the shock structure at the given time. One may picture the manifestation of the inverse relationship in the following way. The material making up the bow shock in the region after the reverse shock sustains its highly compressed value with time. As mentioned, since the bow shock is an asymmetric structure, this contributes the largest polarisation signal. While the bow shock is asymmetric, the material in the rarefaction fan near the wind bubble is highly symmetric. Since it has a much higher density, it dominates the integral, pushing the polarisation signal toward zero. However, the density coming from the wind bubble

varies with time due to the mass loss of the star varying with time. Thus, when the mass-loss rate is at a maximum, so too is the density of the wind. Therefore, the density in the rarefaction fan almost entirely takes over the integral, pushing the polarisation toward zero, thus causing it to attain a minimum. However, as the mass-loss rate approaches a minimum, the contribution of the region near the star, while still symmetric, does not dominate the integral as much since the density is lower and the value of the density in the compressed region after the reverse shock stays relatively stable. This causes the asymmetric bow shock to dominate more in the integral, causing a maximum polarisation signal at minimum mass loss. This inverse relationship persists in both the constant wind velocity case and the variable wind velocity case. Whenever the mass loss is at a maximum, the polarisation is roughly at a minimum. Whenever the mass loss is at a minimum, the polarisation is roughly at a maximum. Although the physics of the current simulations have been very basic, and the polarisation being simple Thomson scattering, this is an excellent numerical demonstration of how the variable wind bow shock hypothesis works in practice to give the inverse relationship observed between the signals of V CVn.

Another aspect of dynamic interest in this interaction is the onset of a Rayleigh-Taylor inability (R-TI), which develops at the interface of the contact discontinuity located at the apex of the bow shock. In 1950, Taylor (1950) discovered that the instability found by Lord Rayleigh (Rayleigh, 1879) involving the interface between a fluid of higher density sitting on top of a fluid of lower density in a gravitational field is far more general and happens under any acceleration, not just gravitational (Sharp, 1984). According to Drazin (2002), "there is instability if and only if the net acceleration is directed from the lighter toward the heavier fluid." The R-TI begins with a perturbation in the density profile, likely caused here by numerical viscosity.

It then grows as 'fingers' in the linear regime of the instability and quickly grows into 'mushroom-type clouds' which are susceptible to the Kelvin-Helmholtz instability. In the case of the simulations presented here in this work, a net acceleration is caused by the material passing through the forward shock and decelerating the contact discontinuity at the apex of the bow shock. Due to the high density of the stellar wind being swept up by the reverse shock, the left-hand side of the contact discontinuity (closer to the star) is far denser than the right-hand side of the contact discontinuity (the ISM material swept up by the forward shock). Since the shocked ISM material is causing a deceleration to the contact discontinuity, this is a net acceleration pointing from the direction of the much less dense shocked ISM to the much more dense shocked stellar wind across the contact. The numerical viscosity then initiates the R-TI in the contact discontinuity, which first acts as fingers. When the mushroom-type structures begin to form, which are Kelvin-Helmholtz unstable in the non-linear regime of the R-TI, they don't have enough time to grow too much because they are swept into the shear flow of the shocked ISM material making its way into the tail. However, this behaviour, along with the mixing of the 'fingers', can cause large perturbations to grow at the interface of the contact discontinuity, leading to the tail destabilizing slightly and changing the morphology. This can be seen toward the end of the simulation of the constant stellar wind velocity case, Figure 3.15.[2] This result should be taken cautiously, as these simulations are only two-dimensional. Full three-dimensional simulations should be examined if one wishes to probe if this instability is truly real. There is some precedent for the onset of this R-TI in the literature, in the work of Villaver et al. (2012), who used the other version of Zeus3D from Stone and Norman (1992), as well as Meyer et al. (2014). Lastly, another interesting aspect of the results is the temperature structure in the tail. Due to the shock heating, the

[2]If one wishes to see and understand this behaviour, it is recommended to view the animation for the density heat map on `https://mike-power666.github.io/`.

tail attains an extremely high temperature. This indicates numerical evidence of a "Mira-like tail" (Neilson et al., 2014).

3.2 Static Wind Velocity Results

Table 3.1: Constant Wind Velocity Simulation Parameters

Stellar Wind Parameters			
Variable	**Value (Common)**	**Value (CGS)**	**Meaning**
$\dot{M}$	$2.0 \times 10^{-7}\,\mathrm{M_\odot\,yr^{-1}}$	$1.2610 \times 10^{+19}$	Average mass-loss rate
$\bar{v}_w$	$25\,\mathrm{km\,s^{-1}}$	$2.5000 \times 10^{+06}$	Average Wind Speed
τ	195 days	$1.6848 \times 10^{+07}$	Period
ω		1.8647×10^{-07}	Frequency
T_w		$3.2000 \times 10^{+03}$	Wind Temperature
R_w		$2.0000 \times 10^{+15}$	Wind Bubble Radius
η	10		mass-loss rate Ratio
λ			Wind Speed Ratio
ζ	12.9057		Wind Bubble Size Ratio
ISM Parameters			
Variable	**Value (Common)**	**Value (CGS)**	**Meaning**
v_*	$150\,\mathrm{km\,s^{-1}}$	$1.5000 \times 10^{+07}$	ISM Speed
n_0		$1.0000 \times 10^{+01}$	ISM Number Density
ρ_0		1.6735×10^{-23}	ISM Density
T_0		$2.5000 \times 10^{+01}$	ISM Temperature
e_0		5.1774×10^{-14}	ISM Internal Energy
Other Parameters			
Variable	**Value (Common)**	**Value (CGS)**	**Meaning**
$\bar{m}$	1.0078 u	1.6736×10^{-24}	Mean Particle Mass
$\bar{R}_{\mathrm{SO}}$	1725.38 AU	$2.5811 \times 10^{+16}$	Standoff Distance
R_{Sim}	0.162 Pc	$5.0000 \times 10^{+17}$	Radius of Simulation
t_{Sim}	22181 Julian Years	$7.0000 \times 10^{+11}$	Total Simulation Time

Time-Varying Parameters	
Variable	**Functional Form (CGS)**
$\dot{M}(t)$	$(2.2927 \times 10^{+18}) + (2.0635 \times 10^{+19}) \cdot \sin^2(\omega t)$
$\rho_w(t)$	$(1.8245 \times 10^{-20}) + (1.6420 \times 10^{-19}) \cdot \sin^2(\omega t)$
$v_w(t)$	
$e_w(t)$	$(7.2248 \times 10^{-09}) + (6.5023 \times 10^{-08}) \cdot \sin^2(\omega t)$

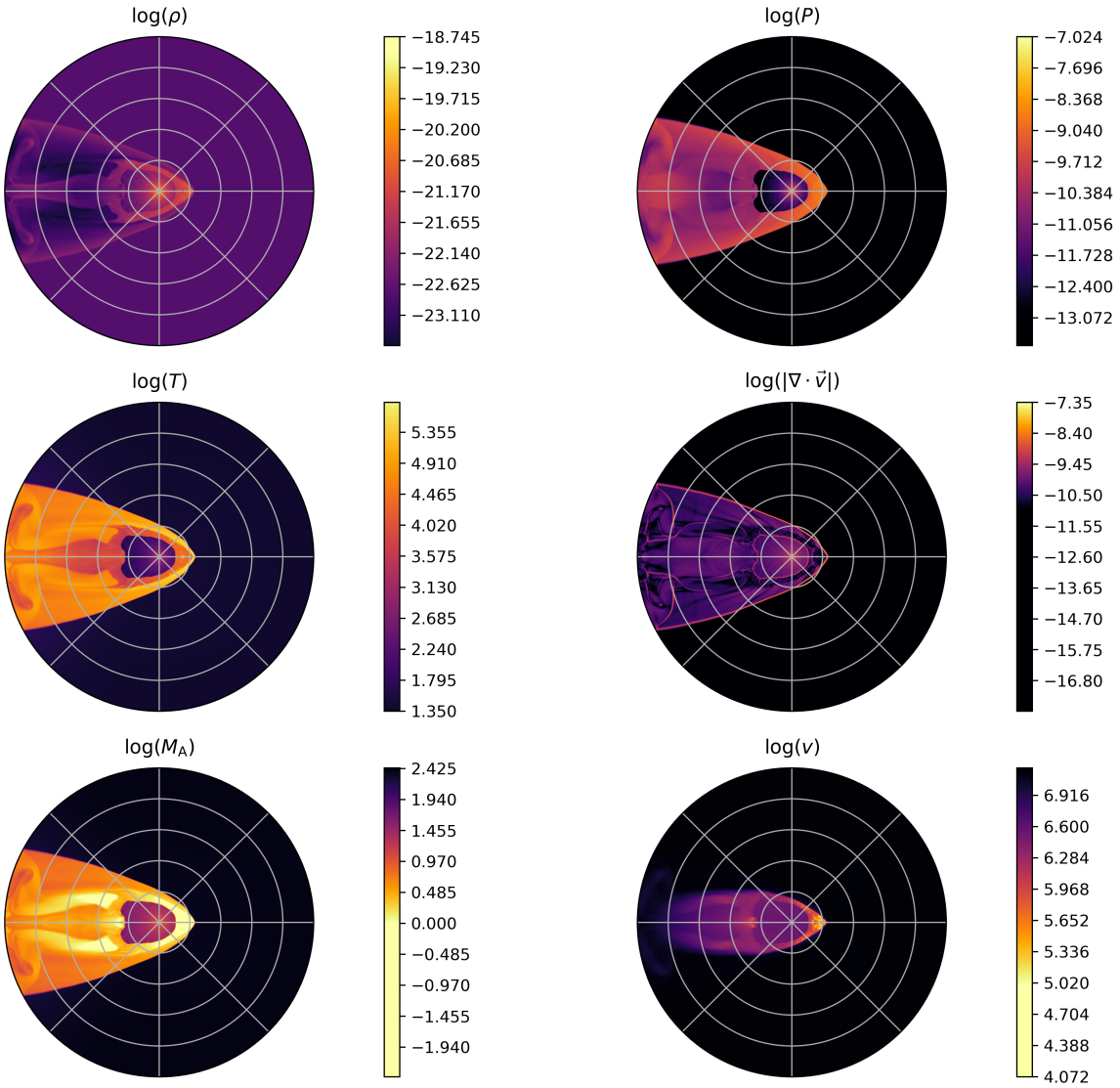


Figure 3.1: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 8.7500 \times 10^{10}$ s for the static wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

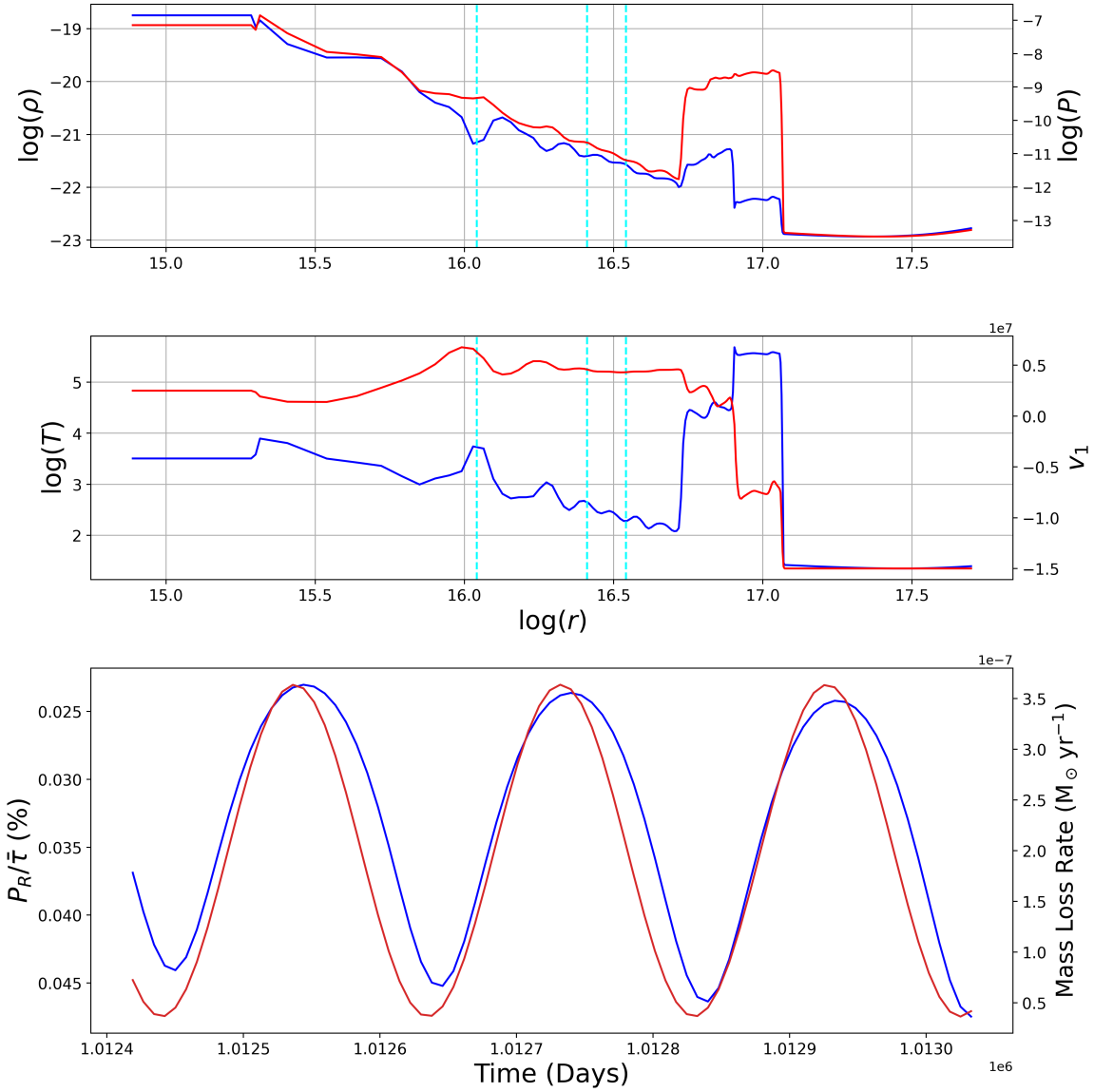


Figure 3.2: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 8.7500 \times 10^{10}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

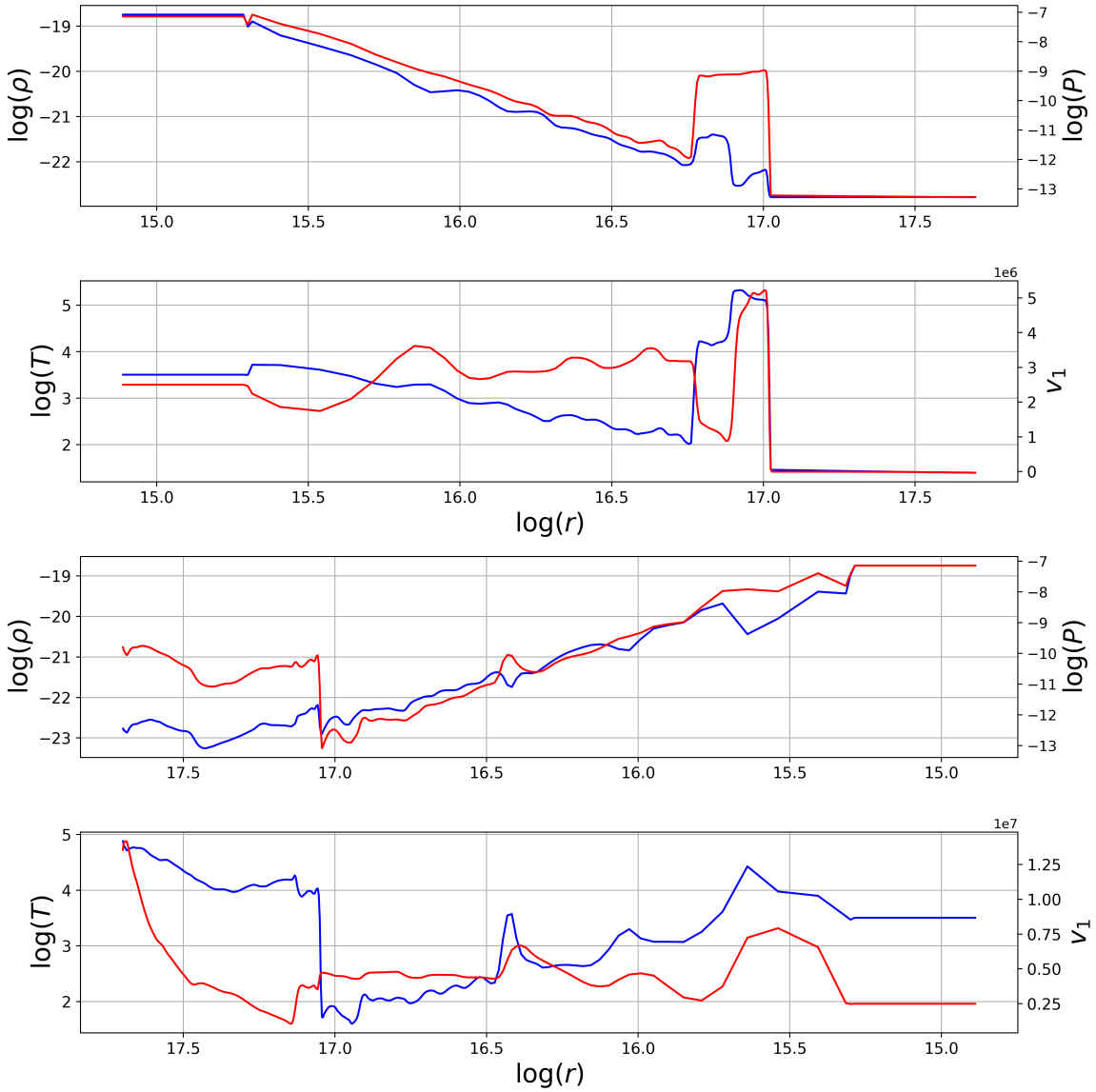


Figure 3.3: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 8.7500 \times 10^{10}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 8.7500 \times 10^{10}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed, so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

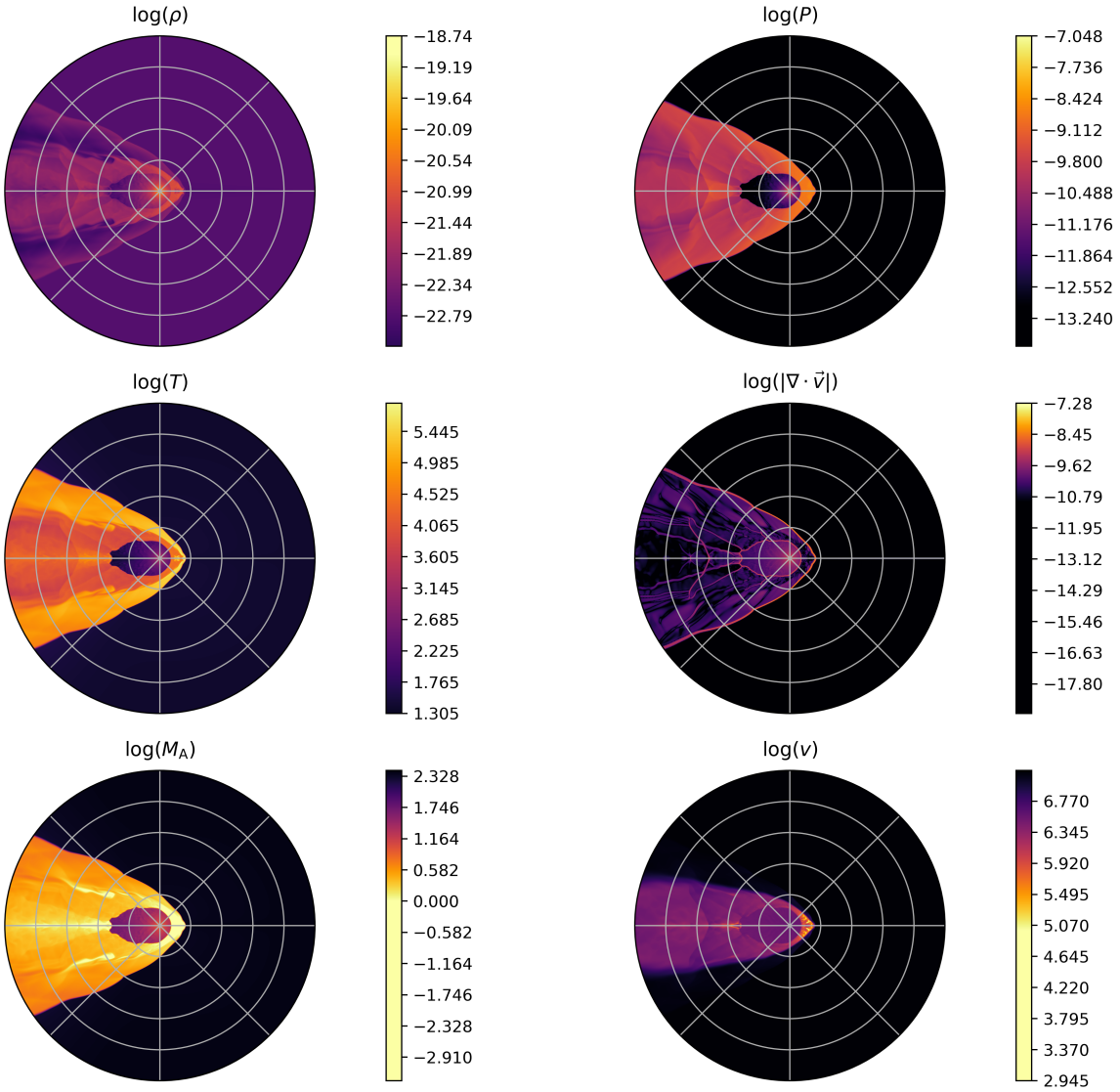


Figure 3.4: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 1.7500 \times 10^{11}$ s for the static wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

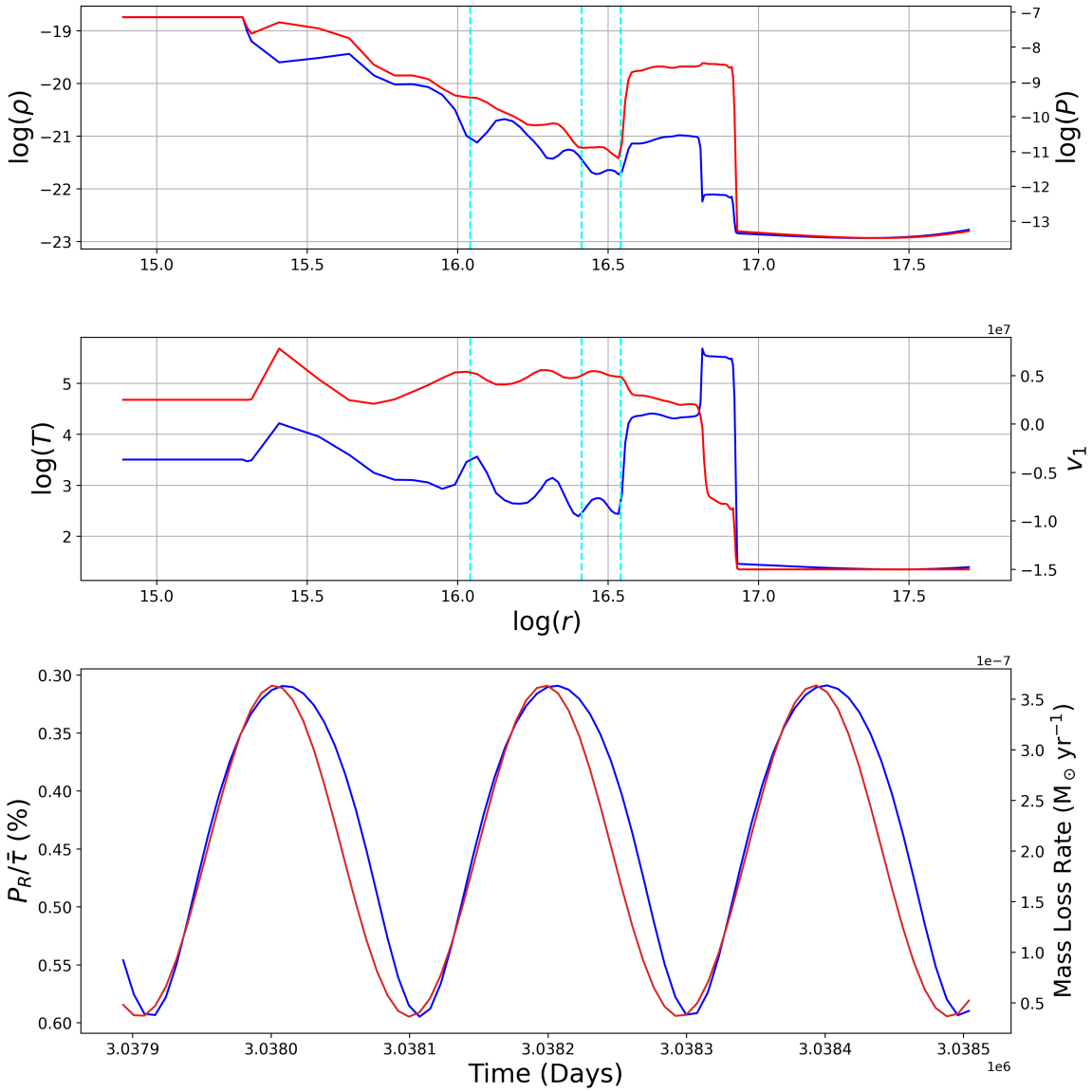


Figure 3.5: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 1.7500 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

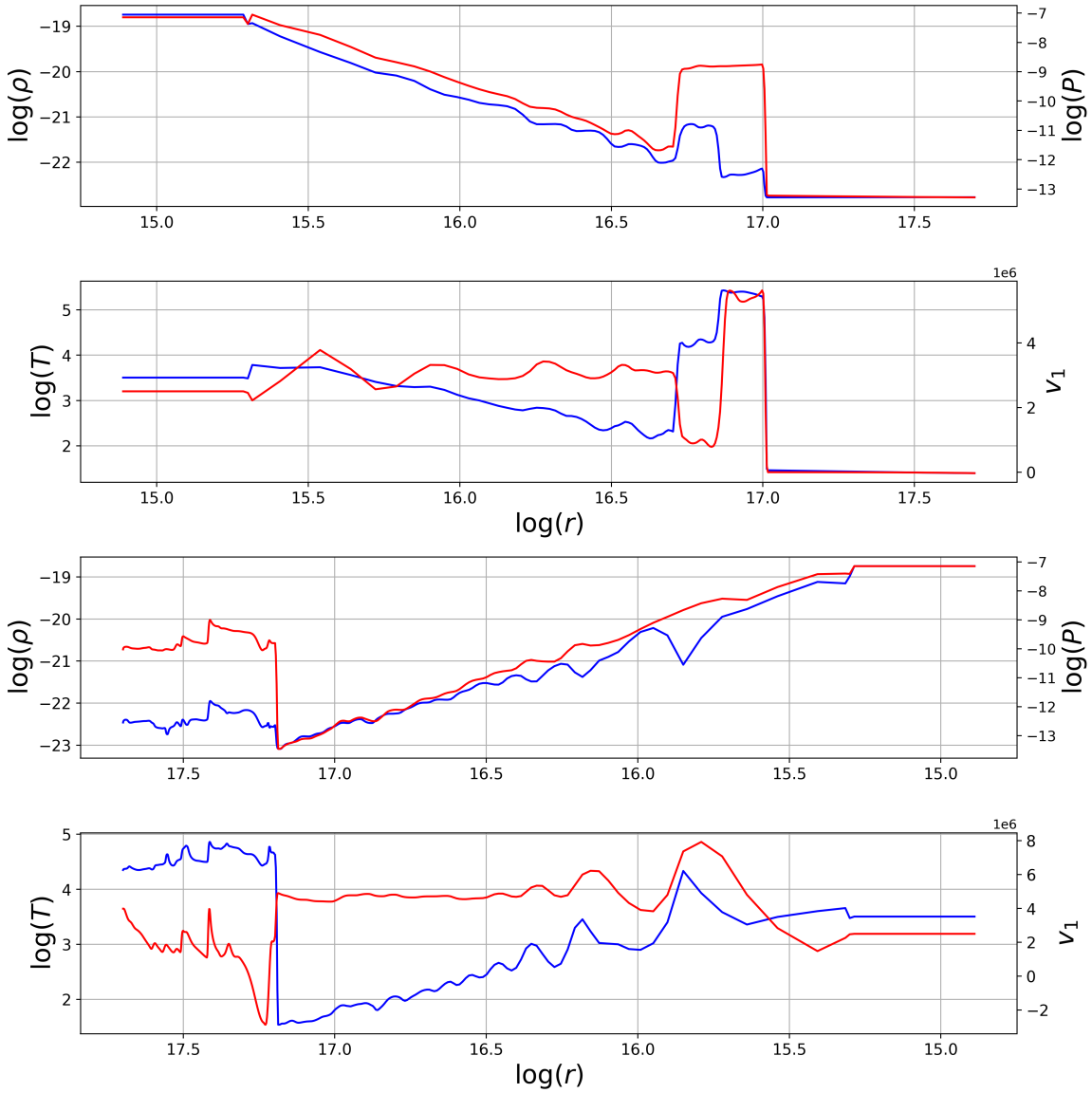


Figure 3.6: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 1.7500 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 1.7500 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed, so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

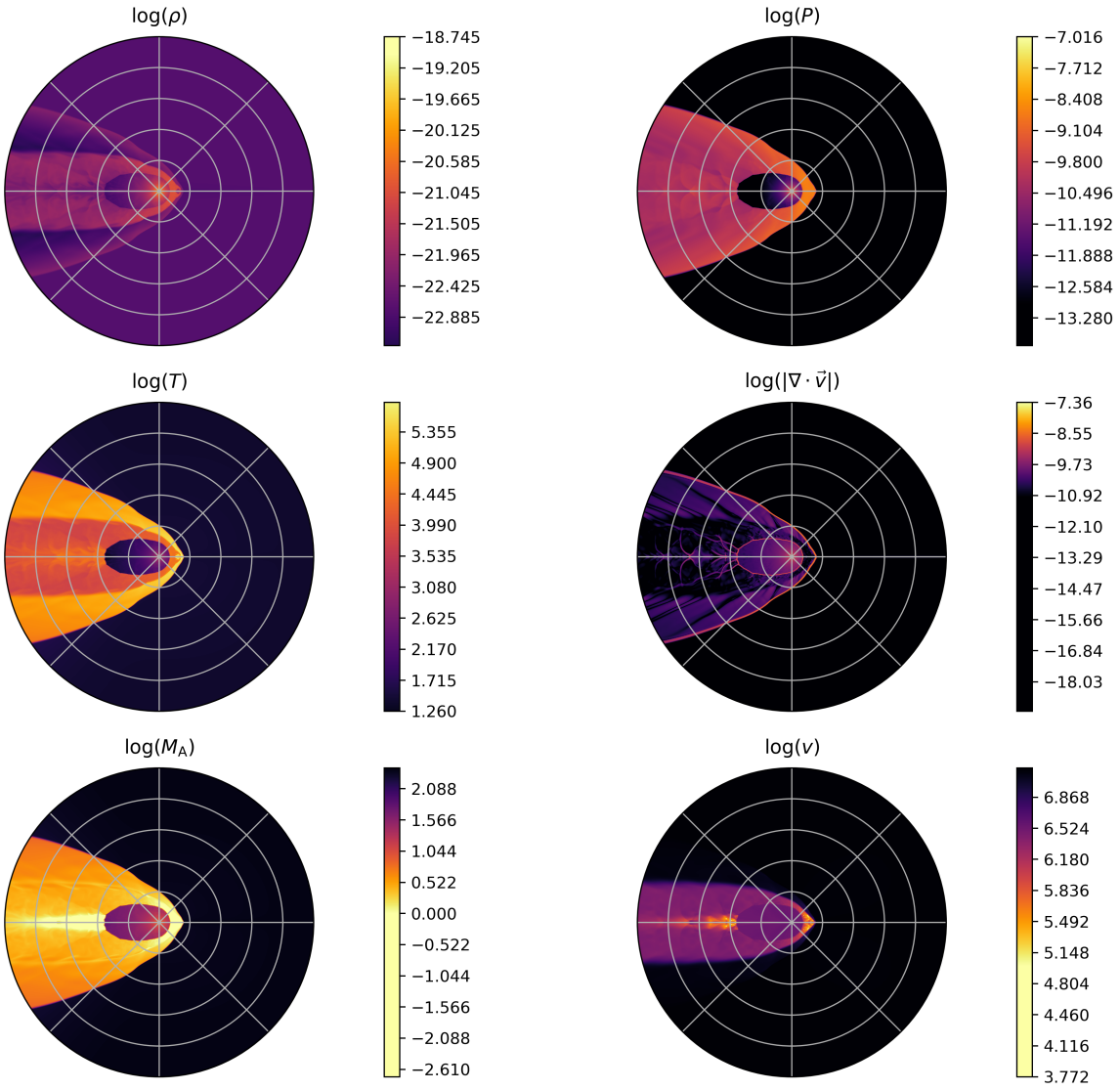


Figure 3.7: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 4.3750 \times 10^{11}$ s for the static wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

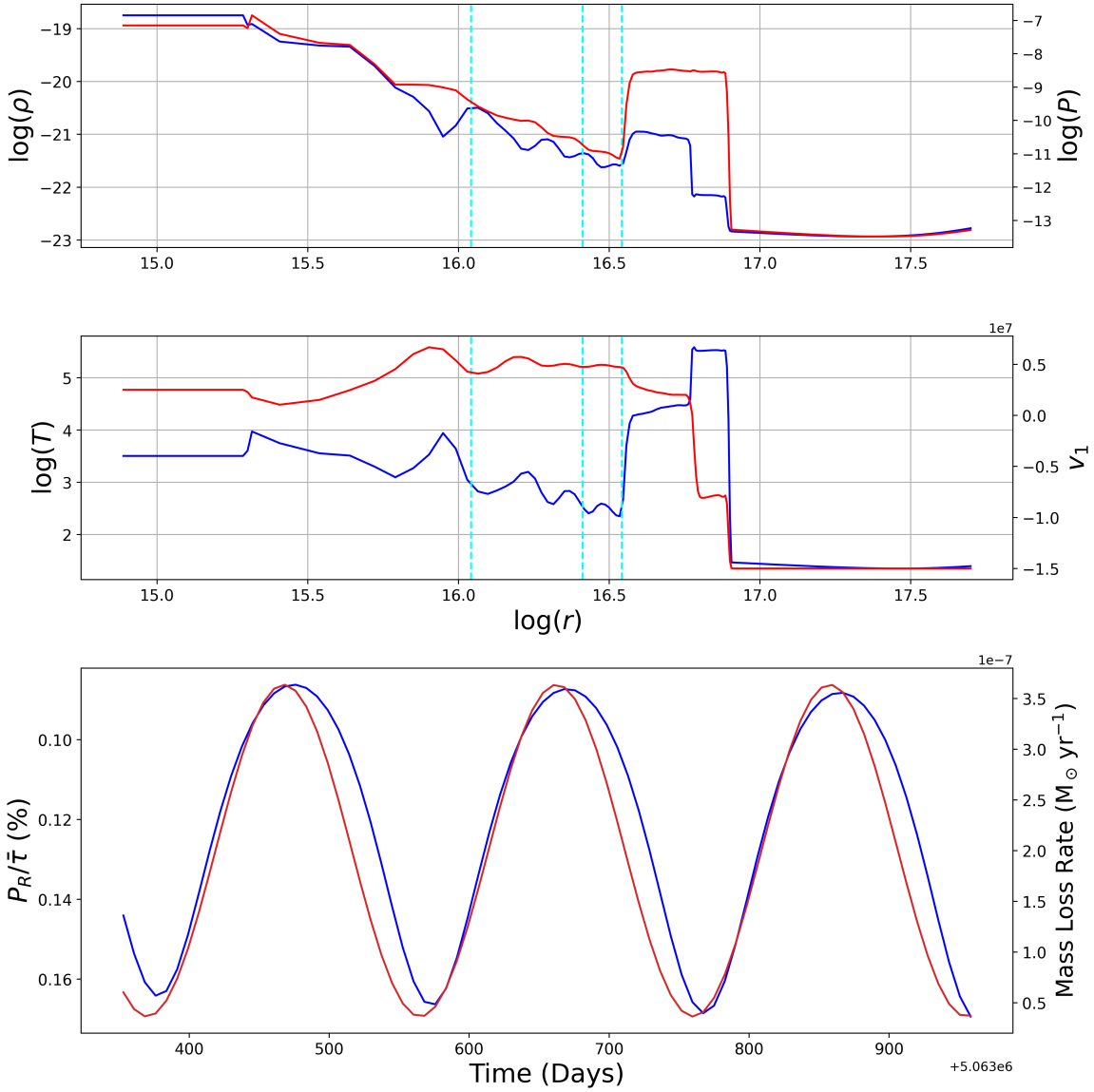


Figure 3.8: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 4.3750 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

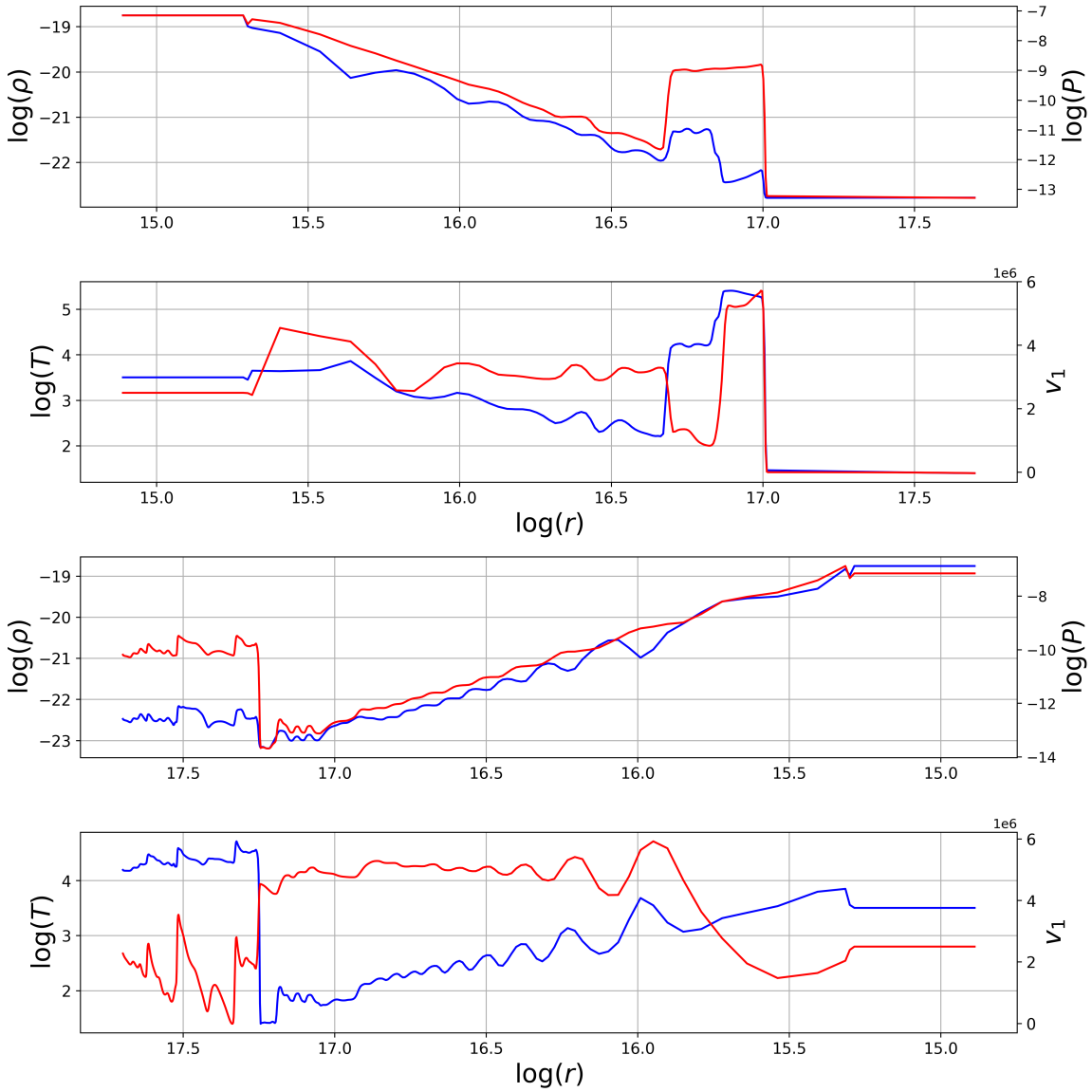
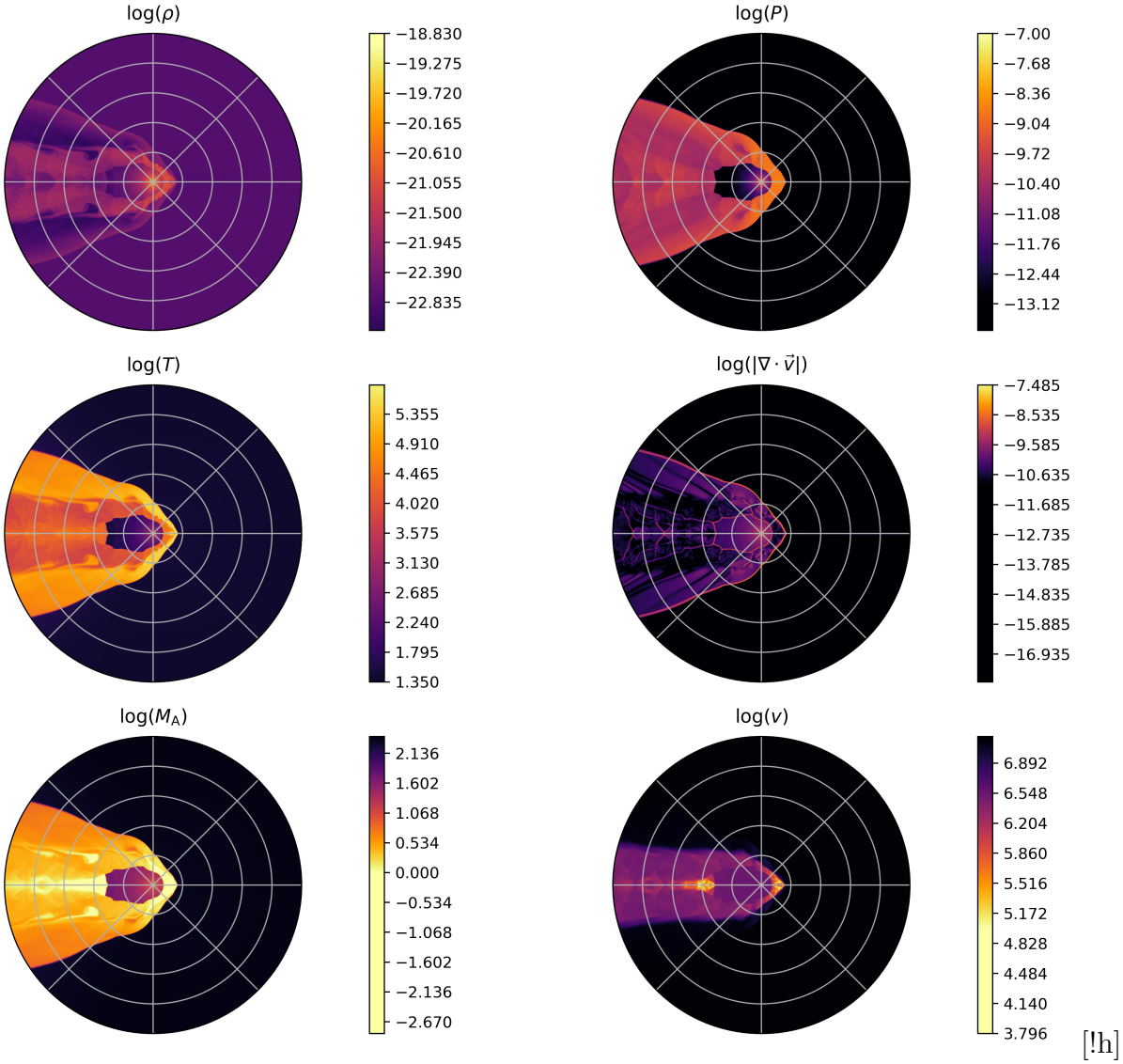


Figure 3.9: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 4.3750 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 4.3750 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed, so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.



[!h]

Figure 3.10: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 7.0000 \times 10^{11}$ s for the static wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

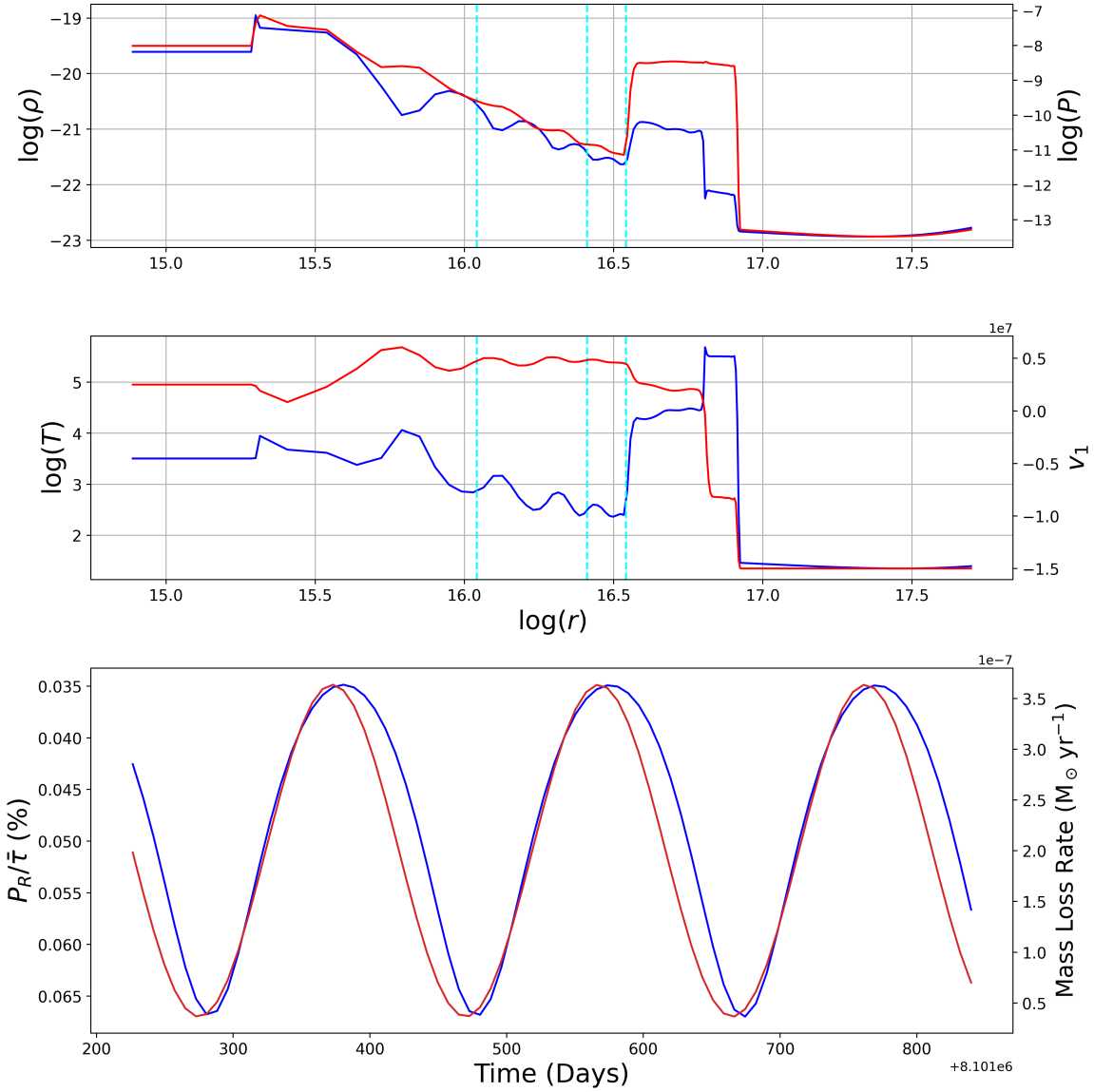


Figure 3.11: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 7.0000 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

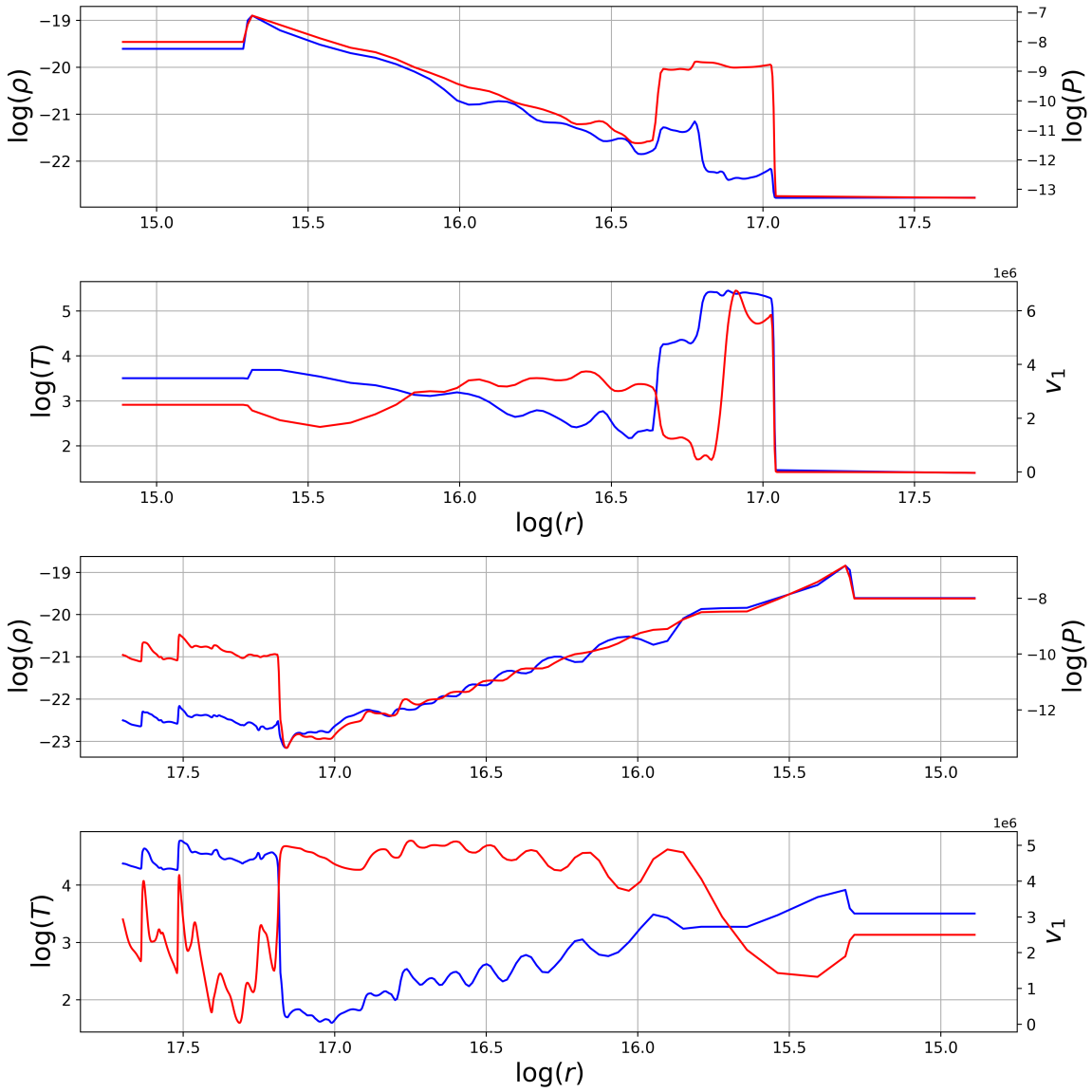


Figure 3.12: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 7.0000 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. Once again, from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 7.0000 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

3.3 Variable Wind Velocity Results

Table 3.2: Variable Wind Velocity Simulation Parameters

Stellar Wind Parameters			
Variable	**Value (Common)**	**Value (CGS)**	**Meaning**
$\dot{M}$	$2.0 \times 10^{-7}\,\mathrm{M_\odot\,yr^{-1}}$	$1.2610 \times 10^{+19}$	Average mass-loss rate
$\bar{v}_w$	$25\,\mathrm{km\,s^{-1}}$	$2.5000 \times 10^{+06}$	Average Wind Speed
τ	195 days	$1.6848 \times 10^{+07}$	Period
ω		3.7293×10^{-07}	Frequency
T_w		$3.2000 \times 10^{+03}$	Wind Temperature
R_w		$2.0000 \times 10^{+15}$	Wind Bubble Radius
η	10		mass-loss rate Ratio
λ	3		Wind Speed Ratio
ζ	12.9057		Wind Bubble Size Ratio
ISM Parameters			
Variable	**Value (Common)**	**Value (CGS)**	**Meaning**
v_*	$150\,\mathrm{km\,s^{-1}}$	$1.5000 \times 10^{+07}$	ISM Speed
n_0		$1.0000 \times 10^{+01}$	ISM Number Density
ρ_0		1.6735×10^{-23}	ISM Density
T_0		$2.5000 \times 10^{+01}$	ISM Temperature
e_0		5.1774×10^{-14}	ISM Internal Energy
Other Parameters			
Variable	**Value (Common)**	**Value (CGS)**	**Meaning**
$\bar{m}$	1.0078 u	1.6736×10^{-24}	Mean Particle Mass
$\bar{R}_{\mathrm{SO}}$	1725.38 AU	$2.5811 \times 10^{+16}$	Standoff Distance
R_{Sim}	0.162 Pc	$5.0000 \times 10^{+17}$	Radius of Simulation
t_{Sim}	22181 Julian Years	$7.0000 \times 10^{+11}$	Total Simulation Time

Time-Varying Parameters	
Variable	**Functional Form (CGS)**
$\dot{M}(t)$	$(1.1114 \times 10^{+19}) + (1.1541 \times 10^{+19}) \cdot \sin(\omega t)$ $+(3.2693 \times 10^{+18}) \cdot \sin^2(\omega t)$
$\rho_w(t)$	$(8.8442 \times 10^{-20}) + (4.7622 \times 10^{-20}) \cdot \sin(\omega t)$
$v_w(t)$	$(2.5000 \times 10^{+06}) + (1.2500 \times 10^{+06}) \cdot \sin(\omega t)$
$e_w(t)$	$(3.5022 \times 10^{-08}) + (1.8858 \times 10^{-08}) \cdot \sin(\omega t)$

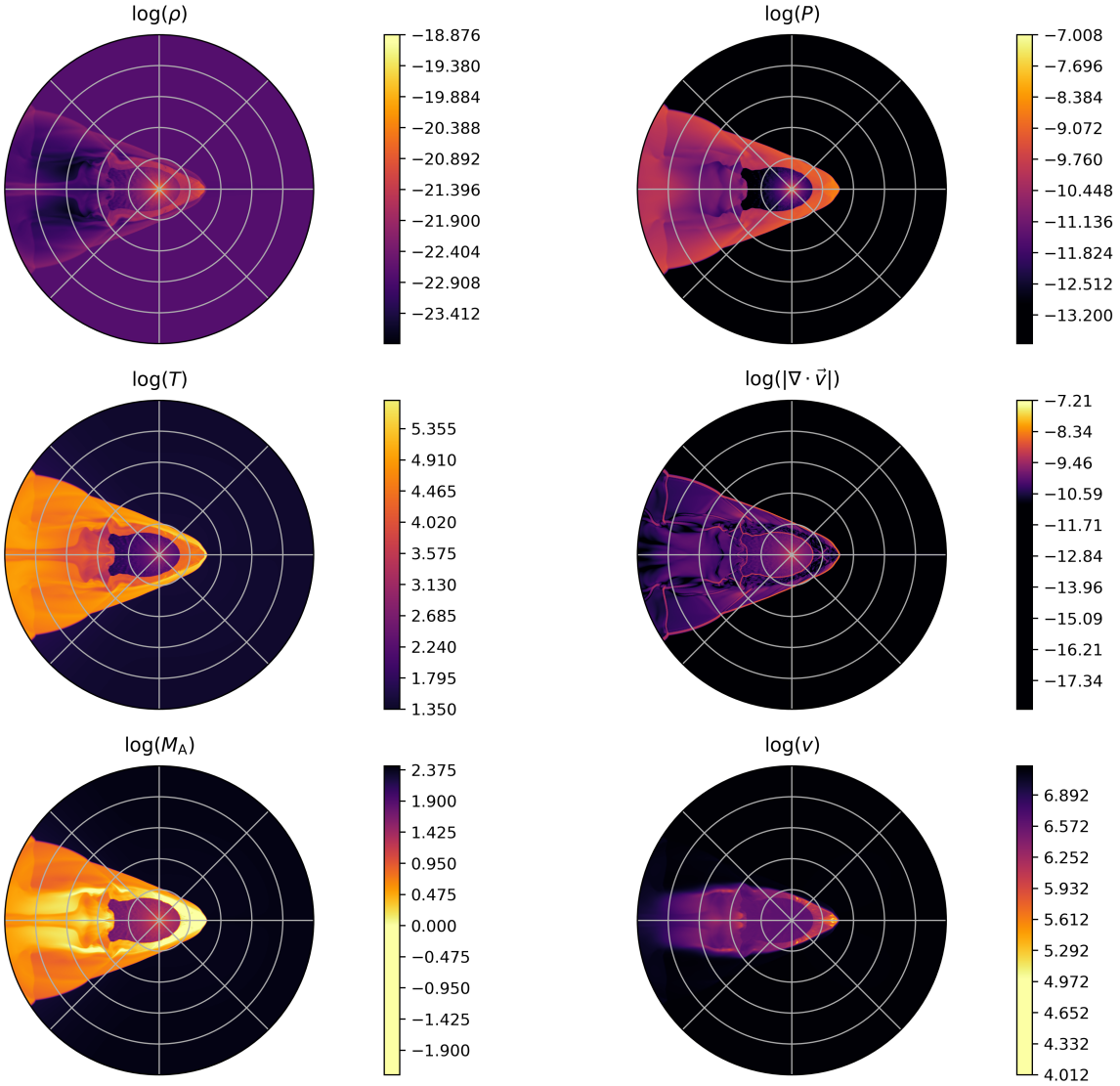


Figure 3.13: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 8.7500 \times 10^{10}$ s for the variable wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

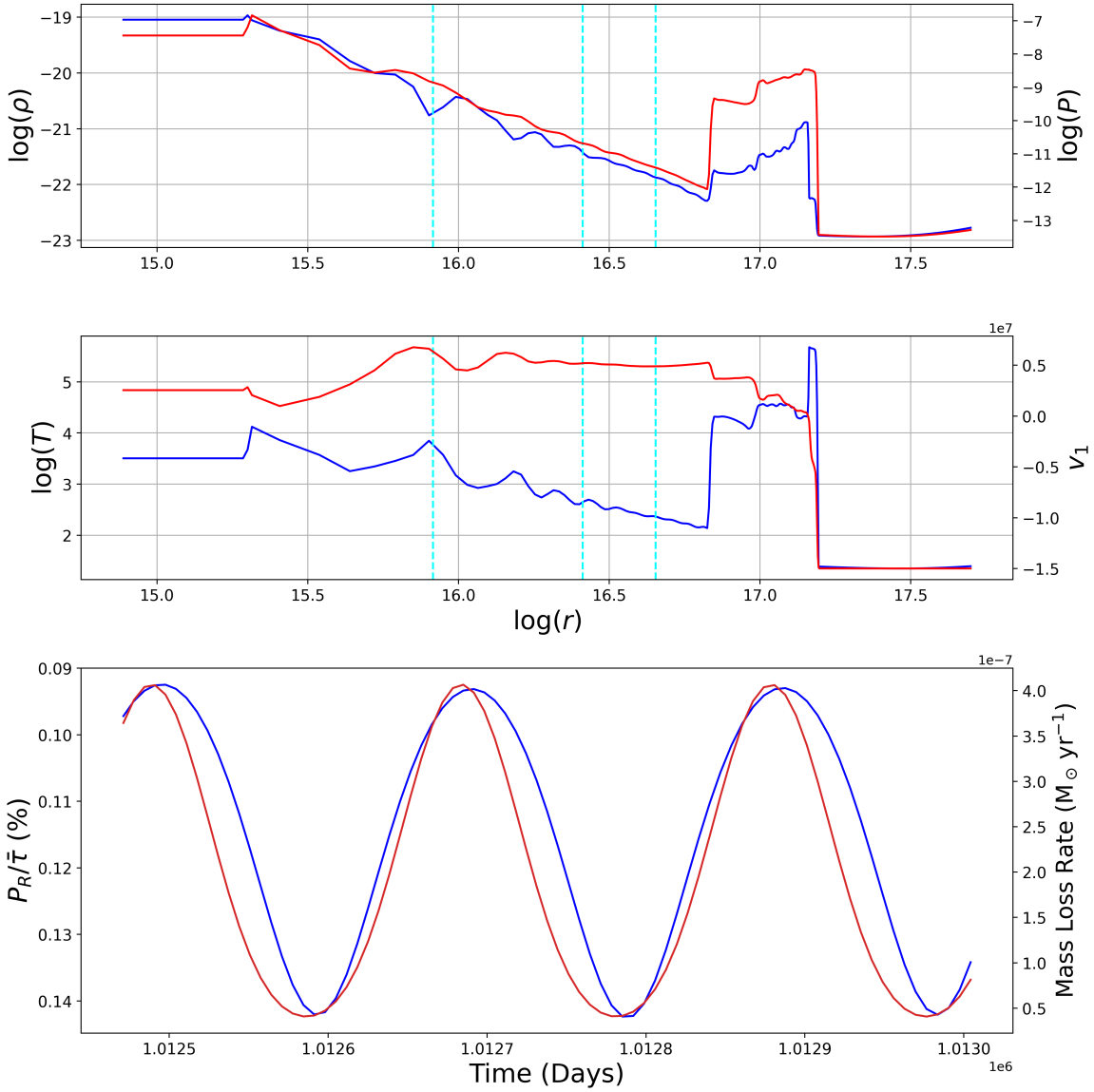


Figure 3.14: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 8.7500 \times 10^{10}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

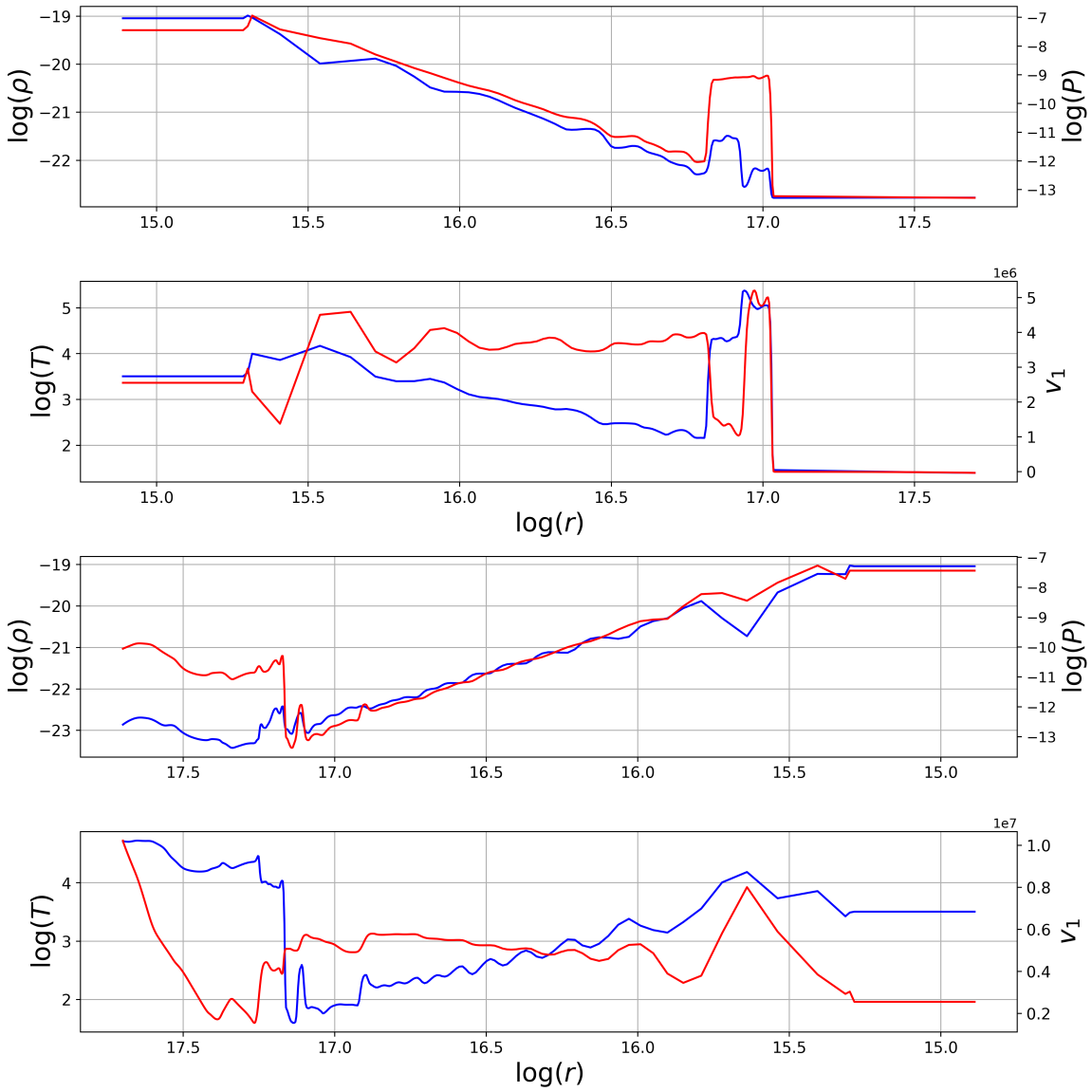


Figure 3.15: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 8.7500 \times 10^{10}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 8.7500 \times 10^{10}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed, so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

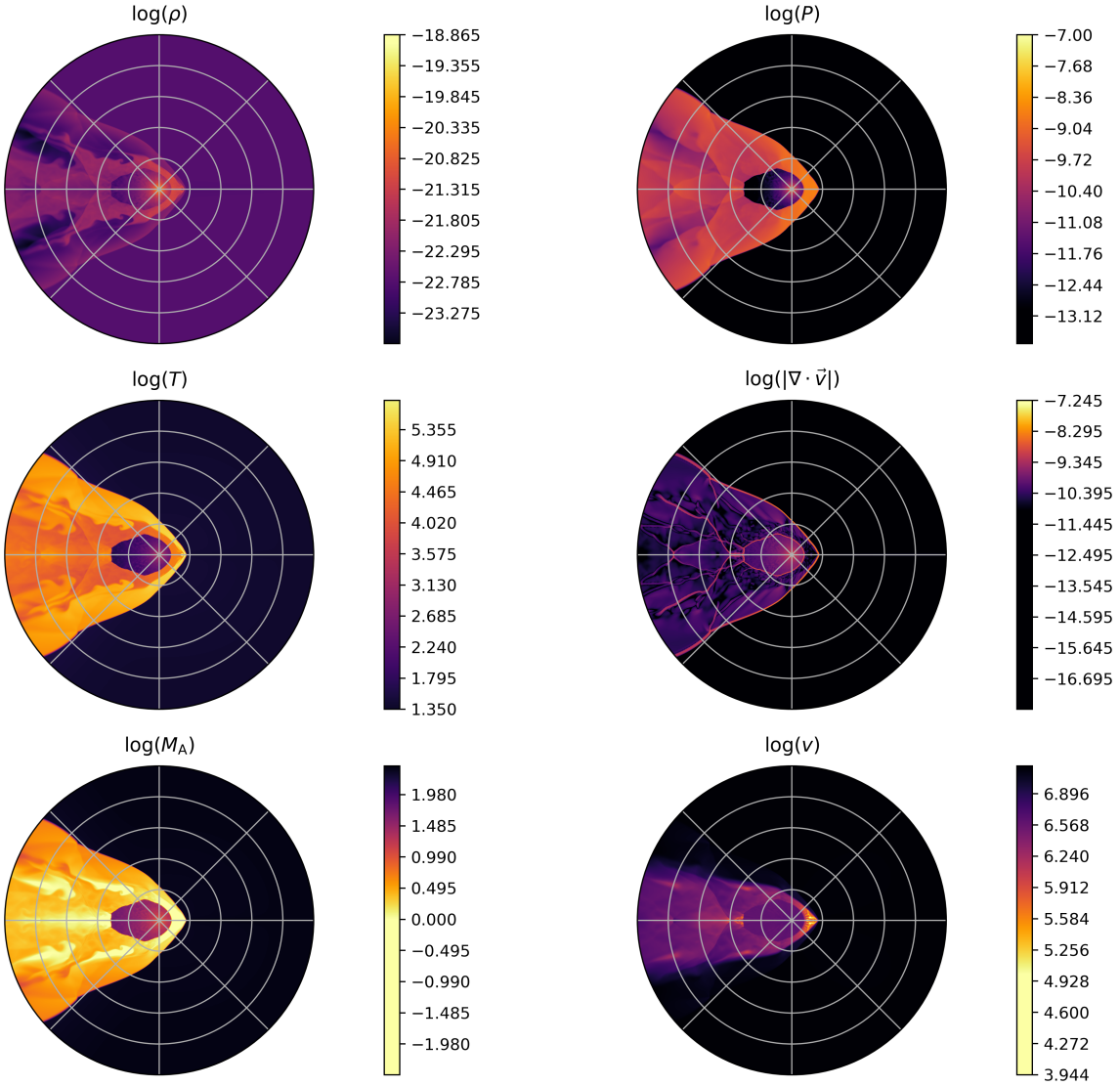


Figure 3.16: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 1.7500 \times 10^{11}$ s for the variable wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

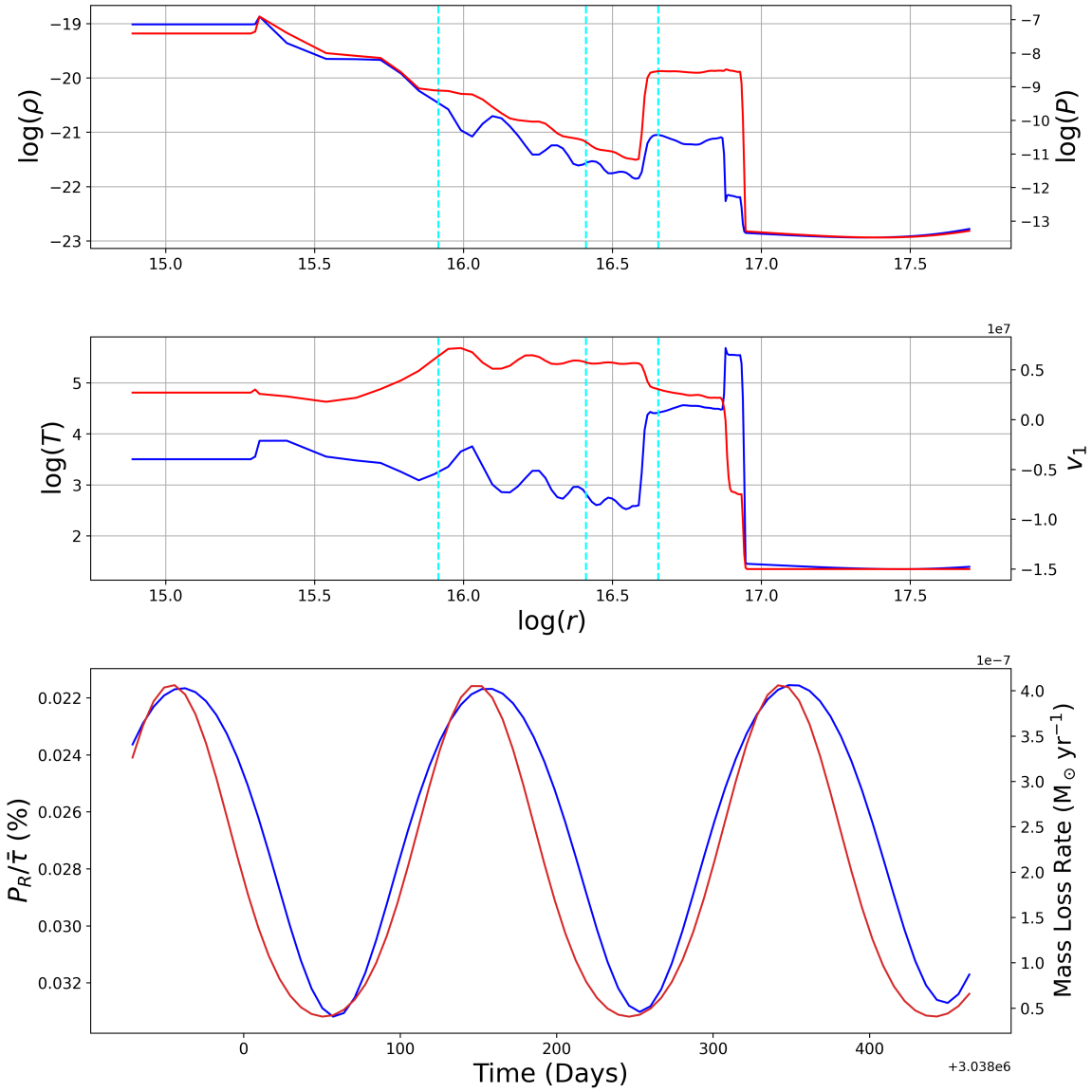


Figure 3.17: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 1.7500 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

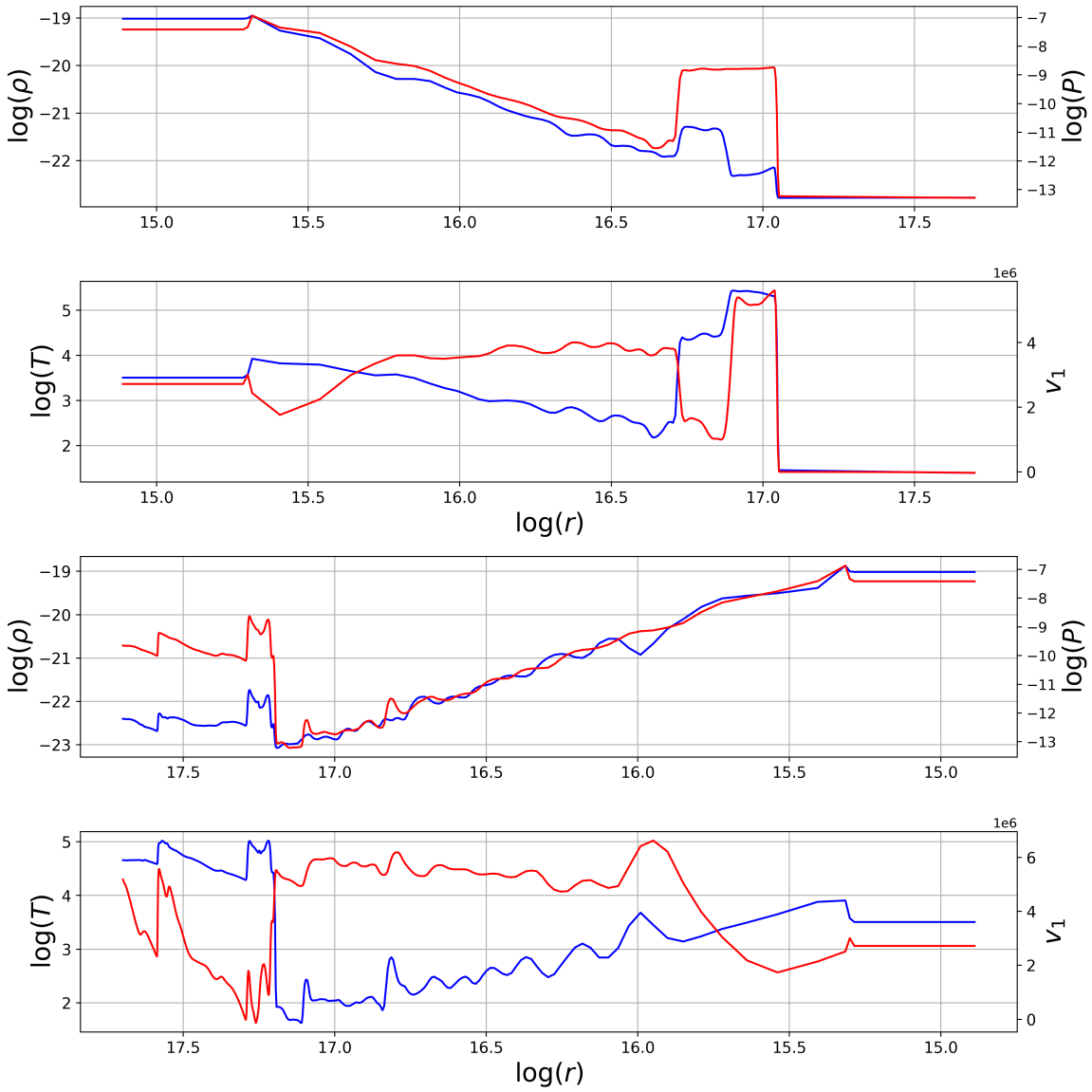


Figure 3.18: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 1.7500 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 1.7500 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed, so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

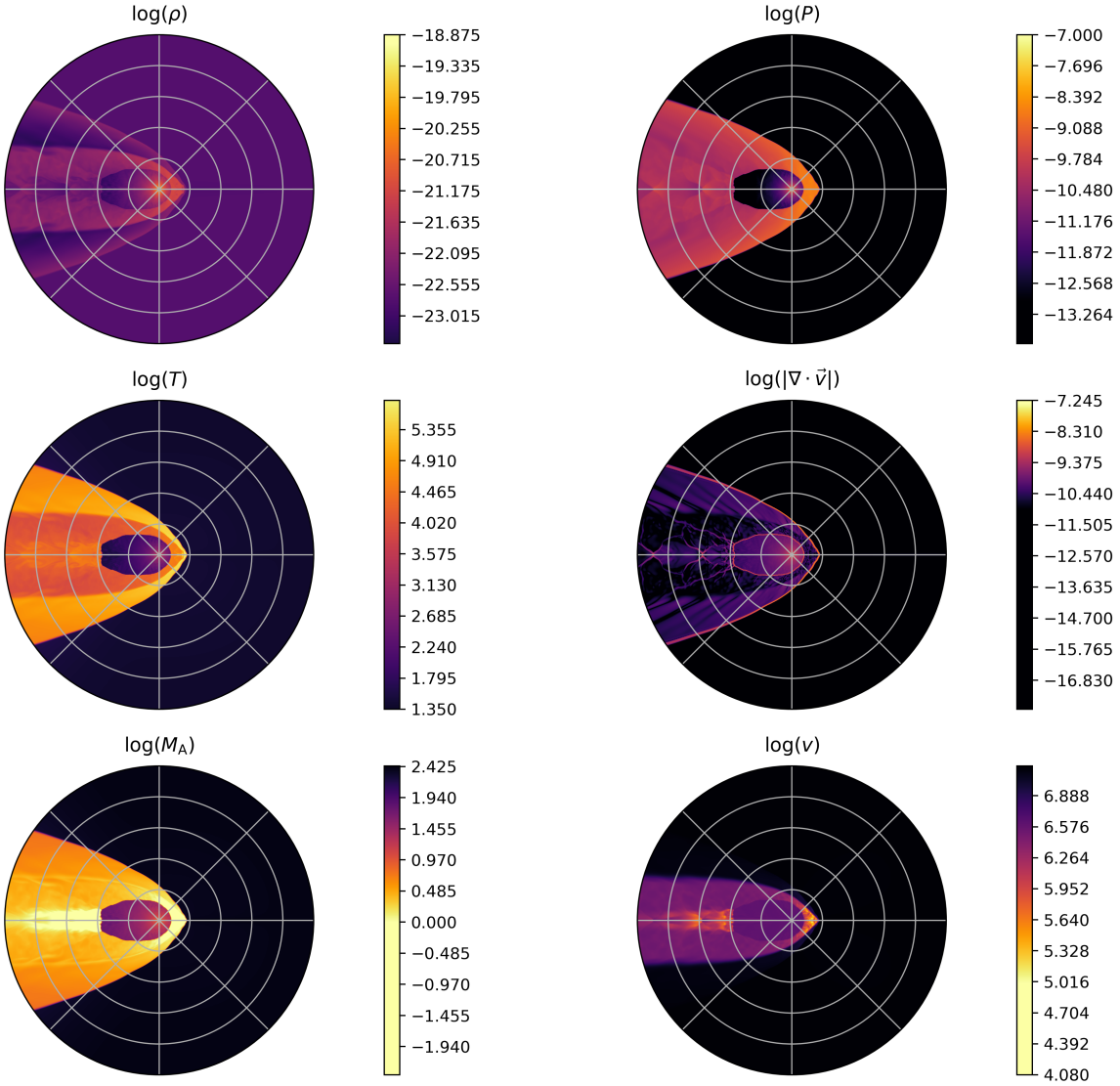


Figure 3.19: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 4.3750 \times 10^{11}$ s for the variable wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

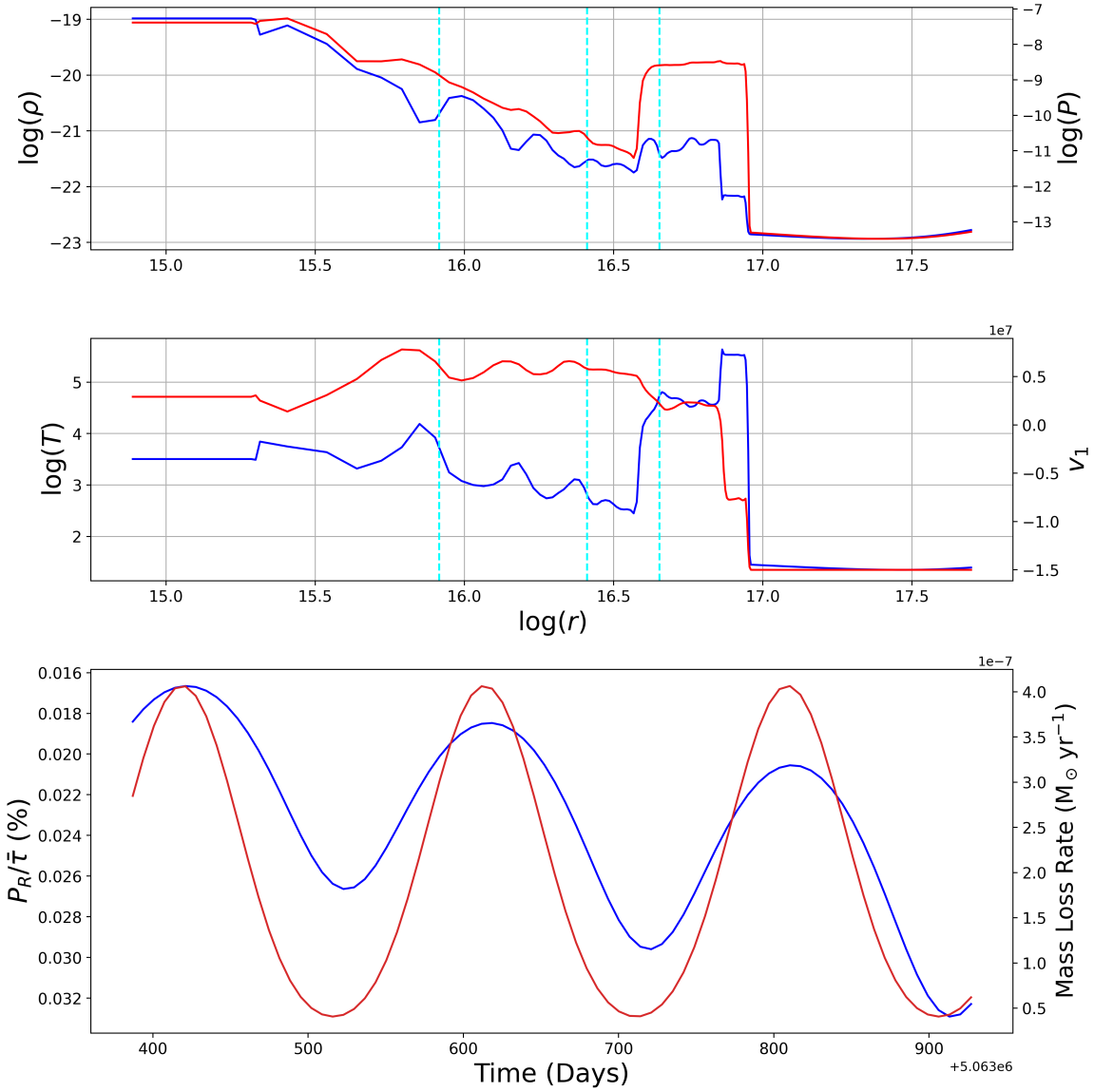


Figure 3.20: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 4.3750 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

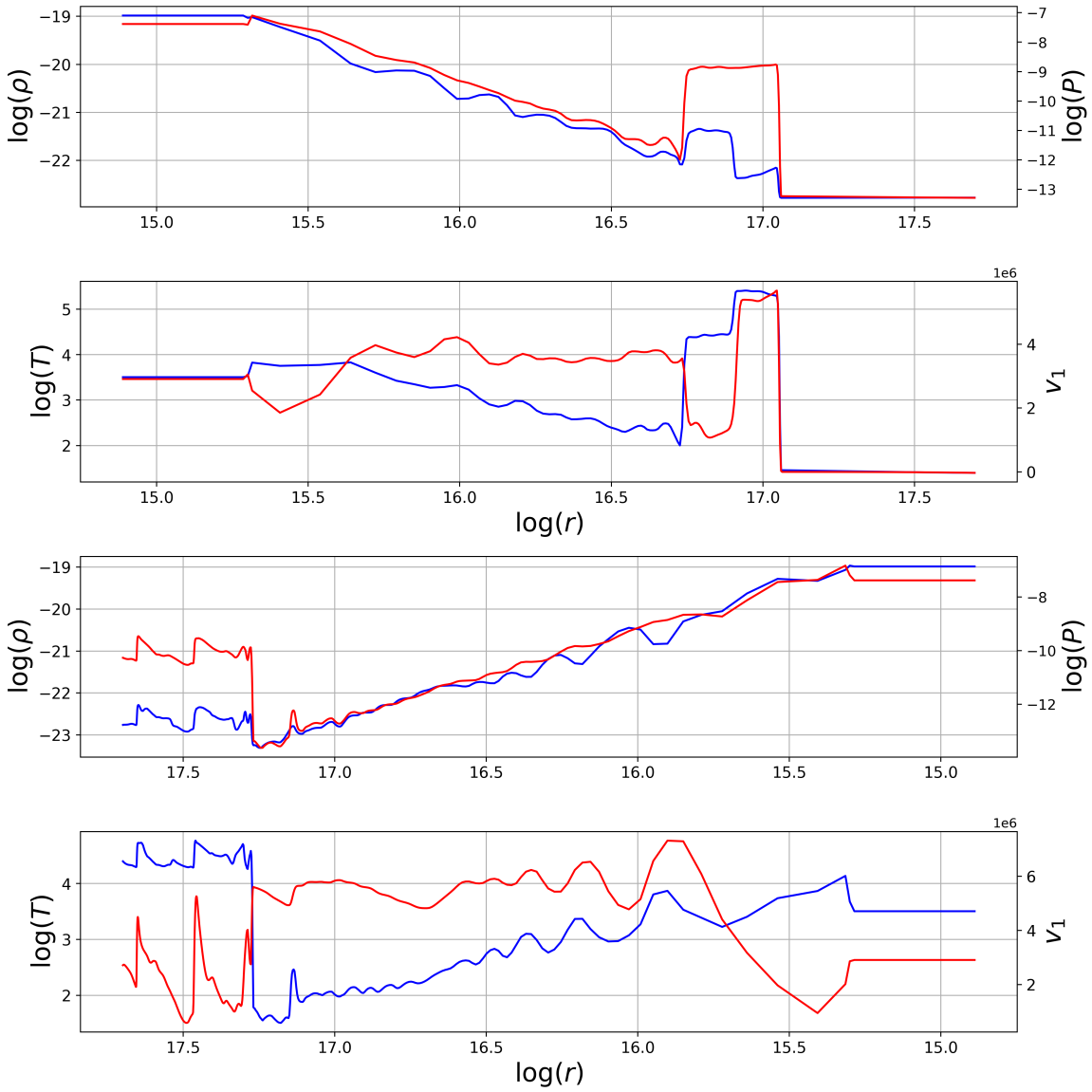


Figure 3.21: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 4.3750 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 4.3750 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed, so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

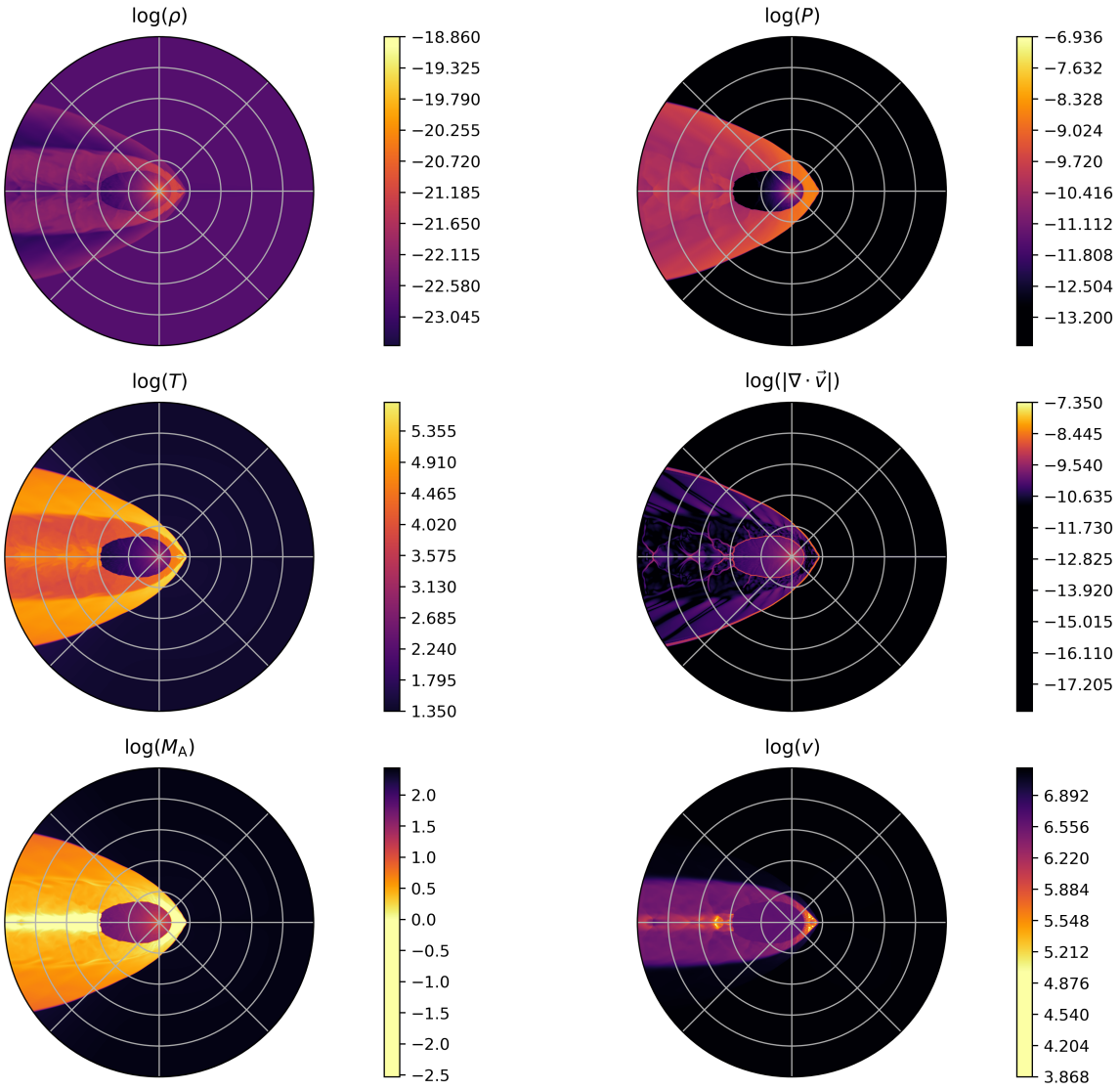


Figure 3.22: From left to right, top to bottom, this Figure represents a heat map of the logarithms of density, thermal pressure, temperature, absolute velocity divergence, Mach number, and speed, at a problem time of $t = 7.0000 \times 10^{11}$ s for the variable wind velocity case. Each concentric circle (gray) indicates a distance of 1.0×10^{17} cm, and each radial line (gray) indicates an angular increment of $\pi/4$ radians. All units are cgs unless otherwise indicated.

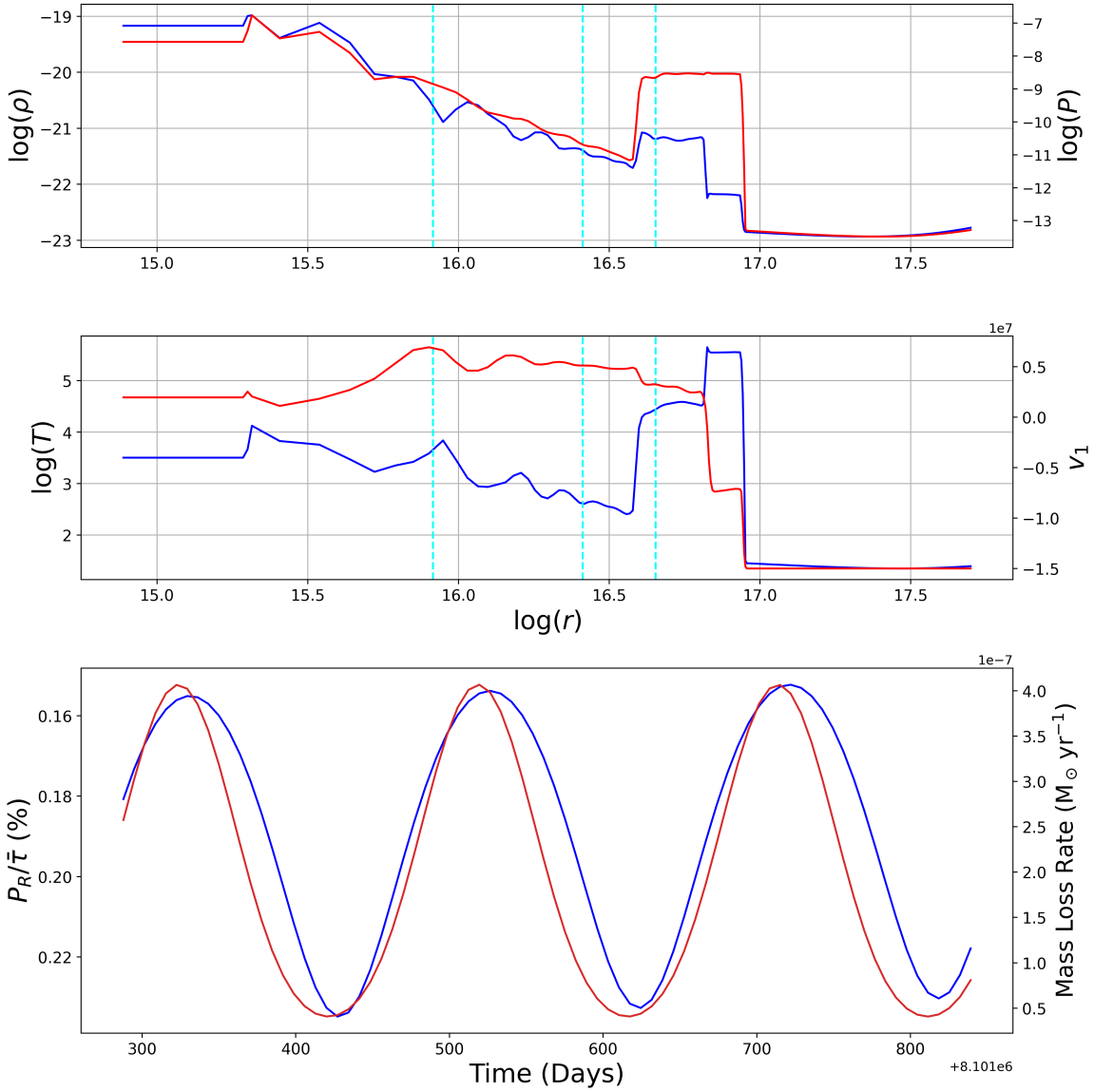


Figure 3.23: From left to right, top to bottom, the 1-dimensional slices along $\theta = 0$ (the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 7.0000 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. The bottom plot from left to right is the residual polarisation as a fraction of the optical depth in percentage (blue) and the mass-loss rate of the star (red). The time coordinates for this plot are the nearest 80 points to the time of the slice. All units are cgs unless otherwise indicated.

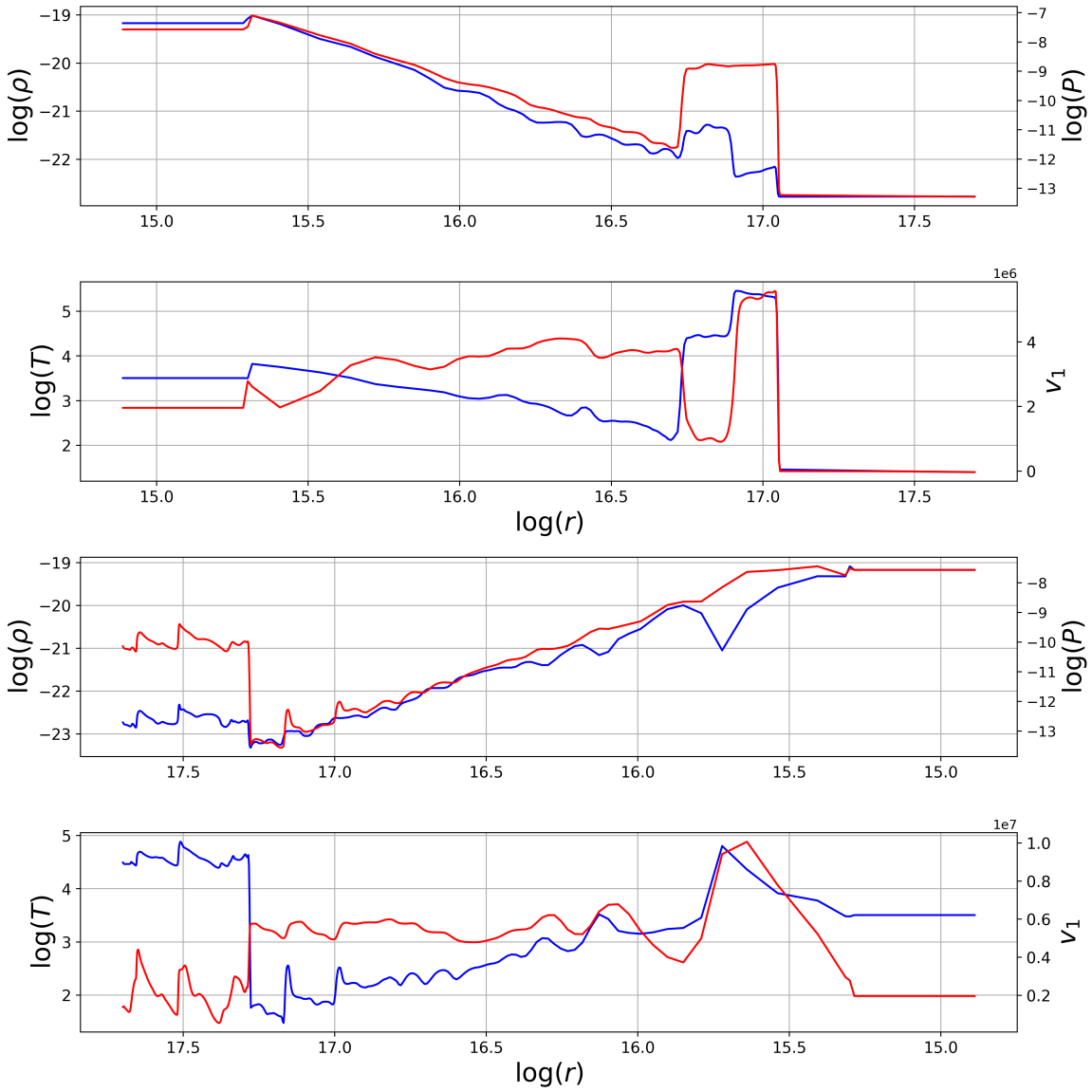


Figure 3.24: From left to right, top to bottom, the 1-dimensional slices along $\theta = \pi/2$ (perpendicular to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 7.0000 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate. Once again from left to right, top to bottom for the lower two plots, the 1-dimensional slices along $\theta = \pi$ (opposed to the direction of motion of V CVn) of the logarithms of the density (blue), thermal pressure (red), temperature (blue), and radial velocity (red) at time $t = 7.0000 \times 10^{11}$ s. All are plotted against the logarithm of the radial coordinate, but the axis is reversed so it is easier to imagine going from the star through the tail. All units are cgs unless otherwise indicated.

3.4 Cross-Correlation Functions for Polarisation and Mass-Loss Rate Data

While the polarisation and mass-loss rate plots at various short time slices allow one to gain an understanding of the relationship between the variables, the entire dataset of over 20000 years cannot be shown in this way on a singular plot due to how quickly the data oscillates compared to the total period. Therefore, another method of displaying the entire dataset coupled with the plots is required to fully understand the relationship. The most useful and intuitive process for showing the relationship between the variables in the whole dataset is a normalised cross-correlation function. The definition of a cross-correlation function (Press et al., 2003) is

$$C(\tau_l) = \int_{-\infty}^{\infty} P_R(t)\dot{M}(t+\tau_l)\,dt. \tag{3.1}$$

Here, τ_l represents the lag. For this work, the normalisation of the cross-correlation function has a mean which vanishes, and a standard deviation of unity. This causes the interpretation of the function to be easier to understand. When the normalized cross-correlation function has a strong positive value, it means that the dataset has a strong linear relationship. Conversely, when the normalized cross-correlation function has a strong negative value, it means that the dataset has a strong inverse relationship. Weak values or vanishing values imply little or no correlation in the dataset, respectively. Mathematically,

$$\hat{C}(\tau_l) = \frac{C(\tau_l) - \mu(C(\tau_l))}{\sigma(C(\tau_l))}. \tag{3.2}$$

Here, $\hat{C}(\tau_l)$ is the normalised cross-correlation function, $\mu(C(\tau_l))$ represents the mean of the cross-correlation function, and $\sigma(C(\tau_l))$ represents the standard deviation of the cross-correlation function.[3] One may show, using Figures 3.25 and 3.26, that the normalised cross-correlation function for each model of V CVn has a strong negative peak near zero, and at zero lag, they also have a strong negative value. This implies that the functions have a strong inverse relationship across the entire dataset, due to the mean and standard deviation of $\hat{C}(\tau_l)$. However, one may notice that the cross-correlation function does not peak at the origin; rather, it peaks in the negative direction near the origin. Thus, while the actual data is strongly inversely related as is, it is not naturally maximally inversely related. Instead, the polarisation leads the mass-loss rate by a small number of days for each of the models.

[3]If one wishes to see the calculation, the code is contained within Appendix A. This code will not be uploaded to `https://mike-power666.github.io/`, as it is not maximally efficient (this should really be accomplished with FFT), nor does it completely meet this author's usual coding standards (it was made relatively rapidly and therefore is a bit messy)! If one wishes to acquire this code for some reason, look on `https://mike-power666.github.io/` for a current email address, and the code may be provided upon request.

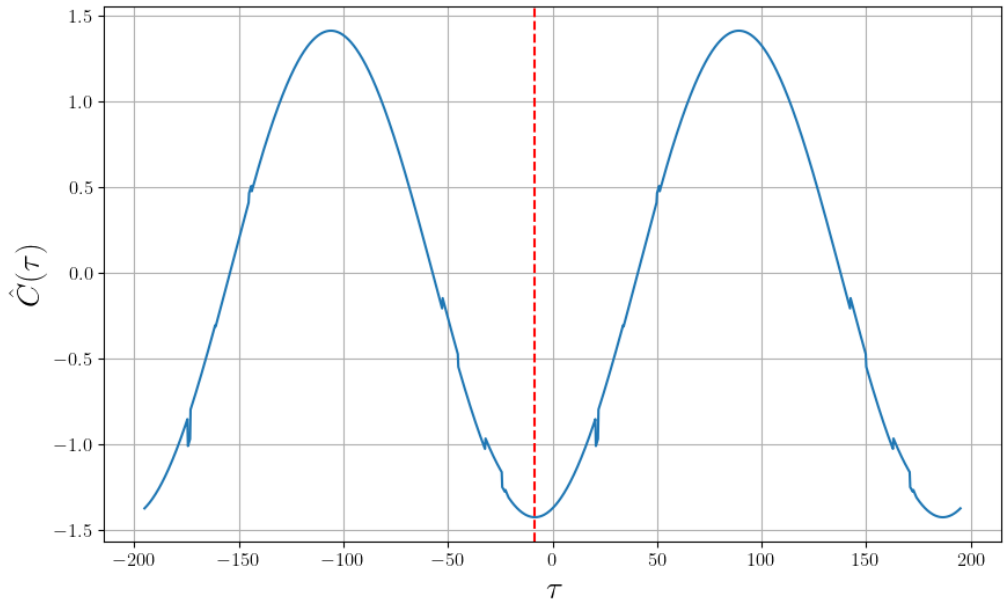


Figure 3.25: The normalised cross-correlation function of the residual polarisation and mass-loss rate for the static wind case of V CVn (blue) plotted against the lag time, τ_l in days. The dashed line (red), located at $\tau_l \approx -8.5$ days with a corresponding value of $\hat{C}(\tau_l) \approx -1.42274$, represents a local minimum of the function. Due to the normalisation condition, this function value means that the residual polarisation and mass-loss rate have an extremely strong inverse relationship. Since the local minimum doesn't occur exactly at zero lag, it means that the polarisation leads the mass-loss rate by about 8.5 days. However, they still have an extremely strong inverse relationship across the dataset.

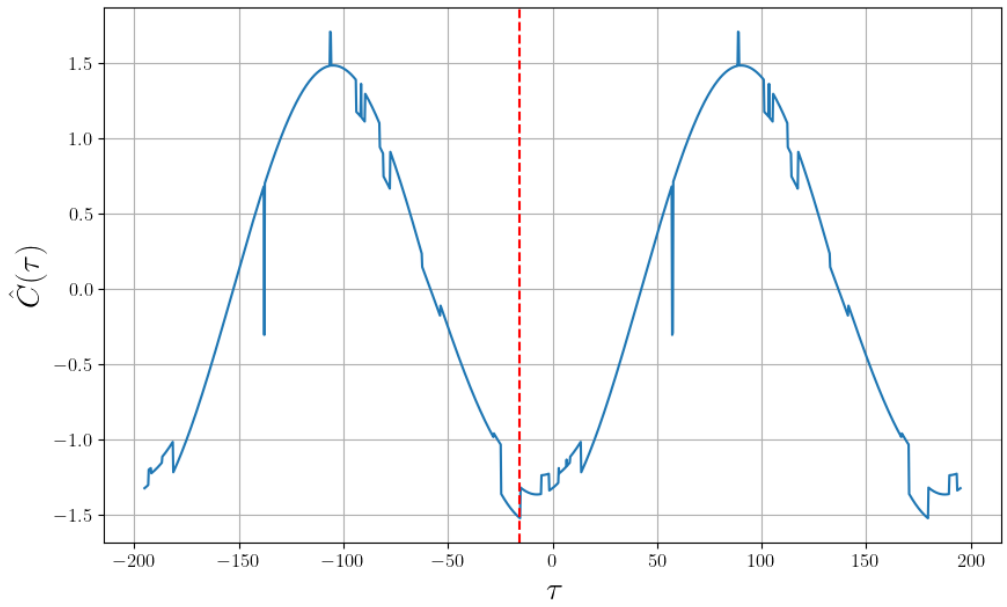


Figure 3.26: The normalised cross-correlation function of the residual polarisation and mass-loss rate for the variable wind case of V CVn (blue) plotted against the lag time, τ_l in days. The dashed line (red), located at $\tau_l \approx -15.5$ days with a corresponding value of $\hat{C}(\tau_l) \approx -1.52166$, represents a local minimum of the function. Due to the normalisation condition, this function value means that the residual polarisation and mass-loss rate have an extremely strong inverse relationship. Since the local minimum doesn't occur exactly at zero lag, it means that the polarisation leads the mass-loss rate by about 15.5 days. However, they still have an extremely strong inverse relationship across the dataset.

Chapter 4

Conclusions and Future Work

> The reward for work well done is the opportunity to do more.
>
> *Jonas E. Salk*

With the new framework for Zeus3D developed for this thesis, one may use its full capabilities to simulate the interaction of stellar winds, which may vary with time, and the ISM. Any physics that Zeus3D is capable of simulating works with the routines, and Zeus3D can now also calculate the Stokes parameters of a density field as it evolves with time. Some of the features of this new framework for stellar wind interactions include, but are not limited to, magnetic fields, stellar rotation, general asymmetric wind profiles, time-varying wind profiles, non-uniform ISM, multiple wind bubbles (for wind-wind interactions; Cartesian coordinates only), and external animation and plotting routines.

The results of both numerical models for the wind of the star V CVn show that the polarisation signal produced varies inversely with its mass-loss rate (a proxy for the star's brightness). Observations of V CVn show that polarisation and brightness

are roughly inversely correlated. Therefore, this work provides strong numerical evidence that when the mass loss of the star is at its maximum, the envelope around the star is dense, meaning it dominates the polarisation integrals, and quite symmetric, meaning it tends to push the polarisation signal to zero causing it to attain a minimum. Conversely, when the mass-loss rate is at a minimum, the stellar wind material and the compressed ISM material in the post-shock regions of the asymmetric bow shock begin to contribute much more to the polarisation integral in comparison to the far less dense symmetric shell around the star at a minimum mass-loss rate. Therefore, the asymmetric structure of the bow shock pushes the polarisation signal to a maximum. Though the physics used in these models has been as simple as possible, additions such as Mie scattering will only strengthen the polarisation signal, thereby strengthening the results. Thus, it is extremely plausible that the curious case of V CVn is solved by the existence of an asymmetric stellar wind bow shock driven by a dusty pulsating wind.

Moving forward, there are many possibilities for continuing this work. The first and most obvious is full three-dimensional simulations if one has access to sufficient computational power. The problem with needing to simulate a bow shock for tens of thousands of years, driven by a wind which varies on the order of two hundred days, is that the wind evolution requires a time step small enough to resolve it adequately. In three dimensions, this constraint severely increases the computational cost. However, three-dimensional simulations immediately give access to several pieces of information. Most important is the calculation of the full suite of Stokes parameters. Having all Stokes parameters changes the polarisation to $P_R = \sqrt{Q^2 + U^2}$, but now gives access to information about the polarisation position angle defined by $\tan(2\psi) = U/Q$. The polarisation is not the only strange behaviour of V CVn; it is also the near-constant

position angle. In three dimensions, asymmetries in the envelope caused by instabilities and turbulence will manifest as a variation in the polarisation position angle. However, these variations will likely be quite small, and therefore, one may find a nearly constant position angle and contribute more evidence to the variable wind bow shock hypothesis.

Second, more realistic physics should be included. The simplest of which is radiative cooling. Zeus3D is already capable of radiative cooling (Raga et al., 1997), and one may turn it on by simply aliasing the cooling flags. This will cause the hot, dense regions between the forward and reverse shocks to cool and decrease the thickness of the bow shock. Also, bow shocks from slow winds and fast relative ISM velocity are subject to radiative instabilities, which tend to shred the bow shock. It would be interesting to see how this would affect the polarisation signal. Other extremely important physics would include Mie scattering by either adding a routine to calculate it in Zeus3D or using the established framework to export the time variation of the density field for use by the SLIP code of Shrestha et al. (2021). Mie scattering is severely more realistic and will help amplify the polarisation signal coming from the bow shock, which will likely cause the signal to become closer to the observed strength. Moreover, one could also include the effects of multiple scattering instead of single scattering. Once again, this process will aim to increase the polarisation signal from the bow shock, likely lifting it closer to observed levels.

Finally, the most difficult and possibly the least impactful addition would be that of a proper stellar wind model, which could be fed into Zeus3D to govern the time variation of the wind bubble boundary conditions. As one may have noticed, the two models used have different methods of varying the mass-loss rate. However, the

results show qualitatively similar behaviour in the polarisation plots. Therefore, it doesn't seem like the process by which the wind pulsates with time would have a massive impact on the nature of the inverse relationship. However, it could affect the signal's strength and morphological features in the tail and shock structure. There is, however, a caveat. If, for some reason, the velocity of the stellar wind and the density were to vary out of phase with one another, it could be the case that the maximum density will not always coincide with the maximum mass loss. One may think of this by considering a case where the velocity is at a maximum, but the density is at some intermediate value and decreasing. This could correspond to a local maximum mass loss but average density. This would cause the polarisation signal not to attain a minimum at this point (since it depends on the density of the dusty spherical shell around the star). Therefore, this could manifest as a lead/lag time in the polarisation and mass loss space. It is thought that V CVn has multiple periods, most notably $\tau = 184\,\text{days}$ and $\tau = 195\,\text{days}$. One could contrive a model similar to the one in this thesis, which has velocity and density variations with different periods. One would have to devise a method of justifying this physically, but it could be an interesting experiment, even if it is unjustifiable!

References

Arfken, G. B., Weber, H. J., and Harris, F. E. (2012). *Mathematical Methods for Physicists: A Comprehensive Guide*. Academic Press, London, 7th edition.

Babusiaux, C., Fabricius, C., Khanna, S., et al. (2023). Gaia Data Release 3. Catalogue validation. *Astronomy & Astrophysics*, 674:A32.

Baranov, V. B., Krasnobaev, K. V., and Kulikovskii, A. G. (1971). Soviet phys.–dokl. *Soviet Physics–Doklady*, 15:791.

Berger, M. and Colella, P. (1989). Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84.

Bohren, C. F. and Huffman, D. R. (1983). *Absorption and Scattering of Light by Small Particles*. Wiley-Interscience.

Brown, J. C. and McLean, I. S. (1977). Polarisation by thomson scattering in optically thin stellar envelopes. i. source star at centre of axisymmetric envelope. *Astronomy and Astrophysics*, 57:141.

Brown, J. C., McLean, I. S., and Emslie, A. G. (1978). Polarisation by thomson scattering in optically thin stellar envelopes. ii. binary and multiple star envelopes and the determination of binary inclinations. *Astronomy and Astrophysics*, 68:415–427.

Brown, W. R., Geller, M. J., Kenyon, S. J., and Kurtz, M. J. (2006). Hypervelocity stars. i. the spectroscopic survey. *The Astrophysical Journal*, 647(1):303.

Carroll, B. W. and Ostlie, D. A. (2007). *An Introduction to Modern Astrophysics*. 2nd (international) edition.

Clarke, D. (2010a). *Stellar Polarimetry*. Wiley-VCH, Berlin.

Clarke, D. A. (1996). A Consistent Method of Characteristics for Multidimensional Magnetohydrodynamics. *Astrophysical Journal*, 457:291.

Clarke, D. A. (2010b). On the reliability of zeus-3d. *The Astrophysical Journal Supplement Series*, 187(1):119.

Clarke, D. A. (2010c). *ZEUS-3D User Manual*. Institute for Computational Astrophysics, Saint Mary's University, Halifax, NS, Canada. Revised versions: 7/11; 11/13; 3/15.

Clarke, D. A. (2016). *What is ZEUS-3D?* Institute for Computational Astrophysics, Saint Mary's University, Halifax, NS, Canada. Copyright © David A. Clarke, 2016.

Clarke, D. A. (2023). Private communication. Discussion regarding computational magnetohydrodynamics using the code Zeus3D.

Colella, P. (1982). Local refinement techniques for elliptic problems on cell-centered grids. *Journal of Computational Physics*, 37(3):367–382.

Colella, P. and Woodward, P. R. (1984). The piecewise parabolic method (ppm) for gas-dynamical simulations. *Journal of Computational Physics*, 54(1):174–201.

Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen differenzengleichungen der mathematischen physik. *Mathematische Annalen*, 100(1):32–74.

Courant, R., Friedrichs, K., and Lewy, H. (1956). On the partial difference equations of mathematical physics. *IBM Journal of Research and Development*, 11(2):215–234.

De Beck, E., Decin, L., de Koter, A., Justtanont, K., Verhoelst, T., Kemper, F., and Menten, K. M. (2010). Probing the mass-loss history of agb and red supergiant stars from co rotational line profiles* - ii. co line survey of evolved stars: derivation of mass-loss rate formulae. *A&A*, 523:A18.

Drazin, P. G. (2002). *Introduction to Hydrodynamic Stability*. Cambridge University Press, Cambridge.

Freyer, T., Hensler, G., and Yorke, H. W. (2003). Massive Stars and the Energy Balance of the Interstellar Medium. I. The Impact of an Isolated 60 M_{solar} Star. *Astrophysical Journal*, 594(2):888–910.

Gaia Collaboration, Prusti, T., de Bruijne, J. H. J., Brown, A. G. A., Vallenari, A., Babusiaux, C., Bailer-Jones, C. A. L., et al. (2016). The Gaia mission. *Astronomy & Astrophysics*, 595:A1.

Gaia Collaboration, Vallenari, A., Brown, A. G. A., Prusti, T., de Bruijne, J. H. J., et al. (2023). Gaia Data Release 3. Summary of the content and survey properties. *Astronomy & Astrophysics*, 674:A1.

Gies, D. R. and Bolton, C. T. (1986). The Binary Frequency and Origin of the OB Runaway Stars. *Astrophysical Journal Supplement*, 61:419.

Henny, J. G. L. M. L. and Cassinelli, J. P. (1999). *Introduction to Stellar Winds*. Cambridge University Press, Cambridge.

Hoogerwerf, R., de Bruijne, J. H. J., and de Zeeuw, P. T. (2001). On the origin of the O and B-type stars with high velocities. II. Runaway stars and pulsars ejected from the nearby young stellar groups. *Astronomy and Astrophysics*, 365:49–77.

Landau, L. D. and Lifshitz, E. M. (1987). *Fluid Mechanics*. Pergamon Press, Oxford, UK, 2nd edition.

Luna, G., Zijlstra, A., Marigo, P., Girardi, L., and Pastorelli, G. (2021). Semi-regular red giants as distance indicators. *Astronomy & Astrophysics*, 656:A66.

Mackey, J., Green, S., Moutzouri, M., Haworth, T. J., Kavanagh, R. D., Zargaryan, D., and Celeste, M. (2021). Pion: Simulating bow shocks and circumstellar nebulae. *Monthly Notices of the Royal Astronomical Society*, 504:983–1008.

Mackey, J., Mohamed, S., Neilson, H. R., Langer, N., and Meyer, D. M. (2012). Double bow shocks around young, runaway red supergiants: Application to betelgeuse. *Astrophysical Journal Letters*, 751.

Magalhães, A. M., Coyne, G. V., and Benedetti, E. K. (1986). Polarimetry of stars with infrared excesses. ii. *The Astronomical Journal*, 91:919.

Meyer, D. M., Gvaramadze, V. V., Langer, N., Mackey, J., Boumis, P., and Mohamed, S. (2014). On the stability of bow shocks generated by red supergiants: The case of irc -10414. *Monthly Notices of the Royal Astronomical Society: Letters*, 439.

Neilson, H., Steenken, N., Simpson, J., Ignace, R., Shrestha, M., Erba, C., and Henson, G. (2023). A multiyear photopolarimetric study of the semi-regular variable v cvn and identification of analog sources. *Astronomy and Astrophysics*, 677.

Neilson, H. R., Ignace, R., Smith, B. J., Henson, G., and Adams, A. M. (2014).

Evidence of a mira-like tail and bow shock about the semi-regular variable v cvn from four decades of polarization measurements. *Astronomy and Astrophysics*, 568.

Poliakova, T. A. (1981). Study on the structure of stellar atmospheres. *Leningradskii Universitet Vestnik Matematika Mekhanika Astronomiia*, 2:105.

Power, M. T. (2018). On the theory of ambipolar diffusion, with applications to astrophysical jets. Bachelor's Thesis. Saint Mary's University, Halifax, N.S. http://library2.smu.ca/handle/01/27887.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2003). *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2nd edition.

Raga, A. C., Mellema, G., and Lundqvist, P. (1997). An axisymmetric, radiative bow shock model with a realistic treatment of ionization and cooling. *The Astrophysical Journal Supplement Series*, 109(2):517.

Rayleigh, L. (1879). On the dynamical theory of gravitation. *Proceedings of the Royal Society of London*, 10:170–177.

Richardson, L. F. (1911). The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 210:307–357.

Richardson, L. F. and Gaunt, J. A. (1927). The deferred approach to the limit. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 226:299–361.

Safonov, B. S., Dodin, A. V., Lamzin, S. A., and Rastorguev, A. S. (2019). The circumstellar envelope of the semiregular variable star v cvn. *Astronomy Letters*, 45:453–461.

Sharp, D. H. (1984). An overview of rayleigh-taylor instability. *Physica D: Nonlinear Phenomena*, 12:3–18.

Shrestha, M., Neilson, H. R., Hoffman, J. L., and Ignace, R. (2018). Polarization simulations of stellar wind bow-shock nebulae - i. the case of electron scattering. *Monthly Notices of the Royal Astronomical Society*, 477:1365–1382.

Shrestha, M., Neilson, H. R., Hoffman, J. L., Ignace, R., and Fullard, A. G. (2021). Polarization simulations of stellar wind bow shock nebulae - ii. the case of dust scattering. *Monthly Notices of the Royal Astronomical Society*, 500:4319–4337.

Sod, G. A. (1978). A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics*, 27(1):1–31.

Stone, J. M. and Norman, M. L. (1992). ZEUS-2D: A Radiation Magnetohydrodynamics Code for Astrophysical Flows in Two Space Dimensions. I. The Hydrodynamic Algorithms and Tests. *Astrophysical Journal Supplement*, 80:753.

Taylor, G. I. (1950). The instability of liquid surfaces when accelerated in a direction perpendicular to their planes. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 201(1065):192–196.

van Leer, B. (1977). Towards the ultimate conservative difference scheme. ii. monotonicity and conservation combined in a second-order scheme. *Journal of Computational Physics*, 23:276.

van Marle, A. J., Langer, N., Achterberg, A., and García-Segura, G. (2006). Forming a constant density medium close to long gamma-ray bursts. *Astronomy and Astrophysics*, 460(1):105–116.

Villaver, E., Manchado, A., and García-Segura, G. (2012). The interaction of asymptotic giant branch stars with the interstellar medium. *Astrophysical Journal*, 748.

Von Neumann, J. and Richtmyer, R. D. (1950). A Method for the Numerical Calculation of Hydrodynamic Shocks. *Journal of Applied Physics*, 21(3):232–237.

Wenger, M., Ochsenbein, F., Egret, D., Dubois, P., Bonnarel, F., Borde, S., Genova, F., Jasniewicz, G., Laloë, S., Lesteven, S., and Monier, R. (2000). The simbad astronomical database: The cds reference database for astronomical objects. *Astronomy and Astrophysics Supplement Series*, 143(1):9–22.

Wilkin, F. P. (1996). Exact analytic solutions for stellar wind bow shocks. *The Astrophysical Journal*, 459:L31–L34.

Wolff, M. J., Nordsieck, K. H., and Nook, M. A. (1996). An ultraviolet interstellar polarization survey: Stars with high color excess. *The Astronomical Journal*, 111(2):856.

Young, H. D. and Freedman, R. A. (2019). *University Physics with Modern Physics*. Pearson.

Appendix A

A Cross-Correlation Calculator

```
!-------------------------------------------------------------------------------
! Memorial University of Newfoundland, MSc Student
!-------------------------------------------------------------------------------
!
! PROGRAM: Cross_Correlation
!
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> This is the main program to execute the steps to calculate a
!> cross-correlation function between the polarisation signal of V CVn and
!> its mass-loss rate.
!
! REVISION HISTORY:
! 25/06/2024 - Initial Version
!-------------------------------------------------------------------------------

program Cross_Correlation
    use Load_Data_Module
    use Correlation_Module
    implicit none

    real(8) :: tau_start, tau_end, tau_step

    ! Call Load_Data, giving access to the Thomson arrays.
    call Load_Data('ThomsonScattering_tr.txt')

    ! Initialize the lags at which to calculate the cross-correlation function (in days).
    tau_start = -195.0d0
    tau_end   =  195.0d0
    tau_step  =    2.5d-1

    ! Call Correlation begin the calculation.
    call Correlation(tau_start, tau_end, tau_step)

end program Cross_Correlation
```

```
!-------------------------------------------------------------------------------
! Memorial University of Newfoundland, MSc Student
!-------------------------------------------------------------------------------
!
! MODULE: Load_Data
!
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> This module opens a file and loads the data into arrays, while avoiding
!> "gotchas".
!
! REVISION HISTORY:
! 25/06/2024 - Initial Version
```

```
!------------------------------------------------------------------------------

module Load_Data_Module
    implicit none
    real(8), dimension(:), allocatable :: t, gam
    integer :: nmax
contains

    subroutine Load_Data(filename)
        character(len=*), intent(in) :: filename
        integer :: i, unit_num, num_lines
        character(len=100) :: line ! This should be adjusted for dynamical allocation.
        logical :: logic_dummy

        ! Determine a unit number for the file.
        do unit_num = 10, 99
            inquire(unit=unit_num, opened=logic_dummy)
            if (logic_dummy .eqv. .false.) exit ! Exit the loop if the unit number was not in use.

            ! Idiot proofing if too many files are opened.
            if ( unit_num .eq. 99 ) then
                print *, "Too many files opened. Load_Data failed."
                stop
            end if
        end do

        ! Open the file.
        open(newunit=unit_num, file=filename, status='old', action='read', iostat=i)
        ! Idiot proofing.
        if (i /= 0) then
            print *, "Error opening file: ", trim(filename)
            stop
        end if

        ! Determine number of lines in the file for array allocation.
        num_lines = 0
        do
            read(unit_num, '(A)', iostat=i) line
            if (i /= 0) exit ! Exit loop at end of file or error.
            num_lines = num_lines + 1
        end do

        ! Allocate memory for arrays.
        allocate(t(num_lines), gam(num_lines))

        ! Rewind the file to read data.
        rewind(unit_num)

        ! Read data from the file into allocated arrays.
        do i = 1, num_lines
            read(unit_num, *) t(i), gam(i)
        end do

        ! Close the file.
        close(unit_num)

        ! Set nmax to the number of lines read.
        nmax = num_lines

        print *, "Data loaded successfully from file: ", trim(filename)
    end subroutine Load_Data

end module Load_Data_Module
```

```
!------------------------------------------------------------------------------
! Memorial University of Newfoundland, MSc Student
!------------------------------------------------------------------------------
!
! MODULE: Correlation
!
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> This module controls the calculation, normalization, and data output of the
!> cross-correlation function. It is parallelized for OpenMP.
!
! REVISION HISTORY:
! 26/06/2024 - Initial Version
!------------------------------------------------------------------------------

module Correlation_Module
    use Data_Output_Module
    use OMP_Lib
    use Load_Data_Module
    use Normalization_Module
    implicit none
```

```
24      real(8), dimension(:), allocatable :: tau, C
25      integer :: tau_max_index
26
27  contains
28
29      subroutine Correlation(tau_start, tau_end, tau_step)
30          use Romberg_Module
31
32          real(8), intent(in) :: tau_start, tau_end, tau_step
33          integer :: j
34          real(8) :: tau_in_seconds, t_start, t_end, maxerror
35          real(8), dimension(:), allocatable :: temp_array
36
37          ! Initialize start and end times of the integral.
38          t_start = t(1)
39          t_end = t(nmax)
40
41          ! Initialize a maximum error for the integration.
42          maxerror = 1.0d-5
43
44          ! Calculate tau_max_index, the number of entries for the tau and C arrays.
45          tau_max_index = int((tau_end - tau_start) / tau_step) + 1
46
47          ! Allocate memory for tau and C arrays
48          allocate(tau(tau_max_index), C(tau_max_index))
49
50          ! Calculate tau and C arrays
51          !$omp parallel do private(j, tau_in_seconds) shared(tau, C)
52          do j = 1, tau_max_index
53              tau_in_seconds = (tau_start + real(j-1, kind=8) * tau_step) * 2.4d1 * 3.6d3 ! Convert tau to seconds.
54              tau(j) = tau_in_seconds
55              ! Call Romberg module here to calculate C(j).
56              call Romberg(t_start, t_end, maxerror, 1, tau(j), C(j))
57          end do
58          !$omp end parallel do
59
60          ! Calculate the statistically normalized Correlation function.
61          allocate(temp_array(tau_max_index))
62          call Normalization(C, tau, tau_max_index, temp_array)
63
64          ! Call Data_Output to prepare a file for plotting.
65          call Data_Output(tau, temp_array, 'Cross_Correlation_Data.txt')
66          deallocate(temp_array)
67
68      end subroutine Correlation
69
70  end module Correlation_Module
```

```
1   !-------------------------------------------------------------------------------
2   ! Saint Mary's University, Honours Student
3   !-------------------------------------------------------------------------------
4   !
5   ! MODULE: Romberg
6   !
7   !> @Mike-Power666
8   !> Michael Power
9   !
10  ! DESCRIPTION:
11  !> A Romberg numerical integrator. This program will output the numerical value
12  !> of the definite integral of your function.
13  !
14  ! REVISION HISTORY:
15  ! 29/04/2017 - Initial Version
16  ! 14/05/2018 - Revision 1
17  !> Expunged all goto statements from the original FORTRAN77 version of this
18  !> algorithm and replaced them with while loops and more clear exit conditions.
19  ! 26/06/2024 - Revision 2
20  !> Refactored the original algorithm to be more modular and specialized it to
21  !> the cross-correlation problem. Also, an interesting bug was found when the
22  !> array, T, was passed into EM in modern fortran. See the EM subroutine for
23  !> a more extensive comment.
24  !-------------------------------------------------------------------------------
25
26  module Romberg_Module
27      use Integrand_Module
28      implicit none
29
30    contains
31
32      subroutine Romberg(A, B, MaxError, flag, param, Result)
33          integer, intent(in) :: flag
34          real(8), intent(in) :: A, B, MaxError, param
35          real(8), intent(out) :: Result
36          real(8), dimension(:,:), allocatable :: T
37          real(8) :: R, RError, H
38          integer :: Q, j, k, m
39
```

```
40          ! Initialize constants and arrays.
41          Q = 250
42          allocate(T(Q, 0:Q))
43
44          ! Compute the first E.M. sum.
45          H = (B - A) / 2.0d0
46          T(1, 0) = H * (Integrand(A, flag, param) + Integrand(B, flag, param)) / 2.0d0 + H * Integrand(A + H, flag, param)
47
48          ! Initialize the setp m to one and call the E.M. subroutine to compute T(2,0).
49          m = 1
50          call EM(m, T, A, B, flag, param)
51
52          ! Top of Romberg Ratio Loop
53          do while (.true.)
54              ! Increment the step m by one and then call the E.M. subroutine to compute T(m+1,0).
55              m = m + 1
56              call EM(m, T, A, B, flag, param)
57
58              R = abs((T(m-1, 0) - T(m, 0)) / (T(m, 0) - T(m+1, 0) + 1.0d-99))
59
60              ! If the Romberg ratio is greater than three, we've found the optimal m value to continue with.
61              ! Otherwise, back to the top of the loop to iterate once again.
62              if (R > 3.0d0) exit
63
64              ! Warn the user if we iterate too much...
65              if (m > 20) then
66                  print *, 'Cannot converge quickly enough... Change variables?'
67                  exit
68              end if
69          end do
70          ! Bottom of Romberg Ratio Loop
71
72          ! Initialize the step k to zero.
73          k = 0
74
75          ! Top of Richardson Extrapolation Loop
76          do while (.true.)
77              ! Increment the step by one.
78              k = k + 1
79
80              ! Warn the user if Richardson extrapolation fails.
81              if ( (m+k) > Q ) then
82                  print *, 'Richardson extrapolation has failed... Increase Q?'
83                  exit
84              end if
85
86              ! Create the Richardson extrapolation table.
87              do j = 1, k
88                  T(m+k, j) = ((4.0d0**j) * T(m+k, j-1) - T(m+k-1, j-1)) / ((4.0d0**j) - 1.0d0)
89              end do
90
91              ! Evaluate the Richardson error term.
92              RError = 2.0d0 * abs((T(m+k, k) - T(m+k, k-1)) / (T(m+k, k) + T(m+1, k-1)))
93
94              ! If the Richardson error term is less than the maximum tolerated error, we're done!
95              ! Otherwise, call the E.M. subroutine to compute T(m+k+1, 0) and then back to the top of the loop!
96              if (RError < MaxError) exit
97
98              call EM(m+k, T, A, B, flag, param)
99          end do
100
101         ! Bottom of Richardson Extrapolation Loop
102         Result = T(m+k, k)
103
104         deallocate(T)
105
106     end subroutine Romberg
107
108     subroutine EM(m, T, A, B, flag, param)
109       integer, intent(in) :: m, flag
110       real(8), intent(in) :: A, B, param
111       real(8), dimension(:,:), allocatable, intent(inout) :: T
112       ! Note that since T(:,:) has non-standard indexing, one must be very cautious. If it isn't defined
113       ! as an allocatable array, or passed as a pointer, the array indexing will default to normal within
114       ! the subroutine and then back to non-standard outside of the subroutine, which gets very confusing.
115       ! This was a result of technical changes to 'allocatable' on dummy arguments in 1997. This comment
116       ! serves as a reminder of this niche problem for the future.
117       real(8) :: Sum, H
118       integer :: i
119
120       ! Calculate the E.M. sum for the (m+1)th step.
121       Sum = 0.0d0
122       H = (B - A) / (2.0d0**(m+1))
123
124       do i = 1, (2**(m+1)-1), 2
125          Sum = Integrand(A + real(i, kind=8) * H, flag, param) + Sum
126       end do
127
```

```
        T(m+1, 0) = T(m, 0) / 2.0d0 + H * Sum

    end subroutine EM

  end module Romberg_Module
```

```
!-------------------------------------------------------------------------------
! Saint Mary's University, Honours Student
!-------------------------------------------------------------------------------
!
! MODULE: Integrand
!
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> The integrand calculator associated with a Romberg integration algorithm.
!
! REVISION HISTORY:
! 29/04/2017 - Initial Version
! 26/06/2024 - Revision 1
!> Refactored the original algorithm to be more modular and specialized it to
!> the cross-correlation problem.
!-------------------------------------------------------------------------------

module Integrand_Module
    implicit none

contains

    function Integrand(x, flag, param) result(return_value)
        integer, intent(in) :: flag
        real(8), intent(in) :: x, param
        real(8) :: return_value

        ! Correlation function for first model.
        if ( flag == 1 ) then
            return_value = polarisation(x) * Mass_Loss_Rate_1(x + param)
        end if

        ! Correlation function for second model.
        if ( flag == 2 ) then
            return_value = polarisation(x) * Mass_Loss_Rate_2(x + param)
        end if

        ! Mean of the correlation function.
        if ( flag == 3 ) then
            return_value = Mean_Integrand(x)
        end if

        ! Standard deviation of the correlation function.
        if ( flag == 4 ) then
            return_value = Variance_Integrand(x, param)
        end if

        ! Romberg test case. Solution to compare with is Ln(|x^3+1|)/3.
        if ( flag == 99 ) then
            return_value = x**2 / (1.0d0 + x**3 + 1.0d-99)
        end if

    end function Integrand

    function polarisation(x) result(return_value)
        use Load_Data_Module
        use Linear_Int_Module
        real(8), intent(in) :: x
        real(8), dimension(:), allocatable :: pol
        real(8) :: return_value

        ! Allocate the polarisation array.
        allocate(pol(nmax))

        ! Calculate the absolute polarisation as a fraction of the optical depth (in %).
        pol = abs(1.0d0 - 3.0d0 * gam)

        ! Calculate the value of the polarisation.
        return_value = Linear_Int(x, t, pol, nmax)

        ! Deallocate the polarisation array.
        deallocate(pol)

    end function polarisation

    function Mass_Loss_Rate_1(x) result(return_value)
        real(8), intent(in) :: x
        real(8) :: c_1, c_2, omega, return_value

```

```
        ! Initialize constants.
        c_1   = 3.63636363d-8
        c_2   = 9.0d0 * 3.63636363d-8
        omega = 1.864668d-7

        ! Calculate the mass-loss rate.
        return_value = c_1 + c_2 * sin(omega * x)**2

    end function Mass_Loss_Rate_1

    function Mass_Loss_Rate_2(x) result(return_value)
        real(8), intent(in) :: x
        real(8) :: c_1, c_2, c_3, omega, return_value

        ! Initialize constants.
        c_1   = 1.762712d-7
        c_2   = 1.830508d-7
        c_3   = 4.745763d-8
        omega = 3.729336d-7

        ! Calculate the mass-loss rate.
        return_value = c_1 + c_2 * sin(omega * x) + c_3 * sin(omega * x)**2

    end function Mass_Loss_Rate_2

    function Mean_Integrand(x) result(return_value)
        use Correlation_Module
        use Linear_Int_Module
        real(8), intent(in) :: x
        real(8) :: return_value

        ! Calculate the correlation function divided by the interval size.
        return_value = Linear_Int(x, tau, C, tau_max_index) / (tau(tau_max_index) - tau(1))

    end function Mean_Integrand

    function Variance_Integrand(x, mean_of_function) result(return_value)
        use Correlation_Module
        use Linear_Int_Module
        real(8), intent(in) :: x, mean_of_function
        real(8) :: return_value, interpolated_C_value

        ! Calculate the interpolated correlation function value.
        interpolated_C_value = Linear_Int(x, tau, C, tau_max_index)

        ! Calculate the full integrand for standard deviation.
        return_value = (interpolated_C_value - mean_of_function)**2 / (tau(tau_max_index) - tau(1))

    end function Variance_Integrand

end module Integrand_Module
```

```
!-------------------------------------------------------------------------------
! Saint Mary's University, Honours Student
!-------------------------------------------------------------------------------
!
! MODULE: Linear_Int
!
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> A simple linear interpolator for an array of function values, y. Note that
!> the array of associated variable values must be monotonically increasing
!> for the binary search algorithm to function properly.
!
! REVISION HISTORY:
! 30/04/2017 - Initial Version
! 26/06/2024 - Revision 1
!> Refactored the original algorithm to be more modular.
!-------------------------------------------------------------------------------

module Linear_Int_Module
    use Binary_Search_Module
    implicit none

contains

    function Linear_Int(int_value, x, y, num_elements) result(return_value)
        integer, intent(in) :: num_elements
        real(8), dimension(num_elements), intent(in) :: x, y
        real(8), intent(in) :: int_value
        integer :: index
        real(8) :: x_1, y_1, x_2, y_2, return_value

        ! Call binary search.
        call Binary_Search(x, num_elements, int_value, index)
```

```
36
37          ! If the binary search returned the max or min value for the array, return the
38          ! lest there be a segfault corresponding value.
39          if ( index == num_elements ) then
40              return_value = y(num_elements)
41          elseif ( index == 1 ) then
42              return_value = y(1)
43          else
44              ! Set the values from the binary search for the linear interpolation.
45              x_1 = x(index - 1)
46              y_1 = y(index - 1)
47              x_2 = x(index)
48              y_2 = y(index)
49
50              ! Compute the linear interpolation.
51              return_value = (y_2 - y_1) * (int_value - x_1) / (x_2 - x_1) + y_1
52          end if
53
54      end function Linear_Int
55
56  end module Linear_Int_Module
```

```
1   !-------------------------------------------------------------------------------
2   ! Saint Mary's University, Honours Student
3   !-------------------------------------------------------------------------------
4   !
5   ! MODULE: Binary_Search
6   !
7   !> @Mike-Power666
8   !> Michael Power
9   !
10  ! DESCRIPTION:
11  !> A classic, simple, and efficient binary search algorithm. The array being
12  !> searched bust be monotonic and increasing. This algorithm returns the index
13  !> of the first entry greater than or equal to the value passed.
14  !
15  ! REVISION HISTORY:
16  ! 30/04/2017 - Initial Version
17  ! 26/06/2024 - Revision 1
18  !> Refactored the original algorithm to be more modular.
19  !-------------------------------------------------------------------------------
20
21  module Binary_Search_Module
22      implicit none
23
24  contains
25      subroutine Binary_Search(monotonic_array, num_elements, value, index)
26          implicit none
27          integer, intent(in) :: num_elements
28          real(8), dimension(num_elements), intent(in) :: monotonic_array
29          real(8), intent(in) :: value
30          integer, intent(out) :: index
31          integer :: low, high, mid
32
33          ! Initialize indices.
34          low = 1
35          high = num_elements
36
37          ! Binary search.
38          do while (low < high)
39              mid = (low + high) / 2
40              if (monotonic_array(mid) < value) then
41                  low = mid + 1
42              else
43                  high = mid
44              end if
45          end do
46
47          ! Check if the element has been found.
48          if (monotonic_array(low) >= value) then
49              index = low
50          else
51              print *, 'Binary search failed... Is your array monotonically increasing?'
52          end if
53
54      end subroutine Binary_Search
55
56  end module Binary_Search_Module
```

```
1   !-------------------------------------------------------------------------------
2   ! Memorial University of Newfoundland, MSc Student
3   !-------------------------------------------------------------------------------
4   !
5   ! MODULE: Normalization
6   !
```

```
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> Calculates the statistical normalization of a cross-correlation function.
!> That is, Norm(C(tau)) = [C(tau) - mean(C(tau))] / sigma(C(tau)).
!> The mean and variance are calculated by virtue of a Romberg integrator.
!
! REVISION HISTORY:
! 28/06/2024 - Initial Version
!------------------------------------------------------------------------------

module Normalization_Module
    implicit none

contains

    subroutine Normalization(C, tau, tau_max_index, norm)
        use Romberg_Module
        real(8), dimension(:), allocatable, intent(in) :: C, tau
        integer, intent(in) :: tau_max_index
        real(8), dimension(tau_max_index), intent(out) :: norm
        real(8) :: mean_of_C, variance_of_C

        ! Call Romberg to calculate the mean.
        call Romberg(tau(1), tau(tau_max_index), 1.0d-9, 3, 0.0d0, mean_of_C)

        ! Call Romberg to calculate the variance.
        call Romberg(tau(1), tau(tau_max_index), 1.0d-9, 4, mean_of_C, variance_of_C)

        ! Calculate the statistically normalized cross-correlation function.
        norm = (C - mean_of_C) / sqrt(variance_of_C)

    end subroutine Normalization

end module Normalization_Module
```

```
!------------------------------------------------------------------------------
! Memorial University of Newfoundland, MSc Student
!------------------------------------------------------------------------------
!
! MODULE: Data_Output
!
!> @Mike-Power666
!> Michael Power
!
! DESCRIPTION:
!> This module opens a file and dumps the data into arrays, while avoiding
!> "gotchas".
!
! REVISION HISTORY:
! 28/06/2024 - Initial Version
!------------------------------------------------------------------------------

module Data_Output_Module
    implicit none
contains

    subroutine Data_Output(tau, C, filename)
        real(8), dimension(:), intent(in) :: tau, C
        character(len=*), intent(in) :: filename
        integer :: i, unit_num, num_lines
        logical :: logic_dummy

        ! Determine a unit number for the file.
        do unit_num = 10, 99
            inquire(unit=unit_num, opened=logic_dummy)
            if (.not. logic_dummy) exit ! Exit the loop if the unit number is not in use.

            ! Idiot proofing if too many files are opened.
            if (unit_num == 99) then
                print *, "Too many files opened. Data_Output failed."
                stop
            end if
        end do

        ! Open the file.
        open(newunit=unit_num, file=filename, status='replace', action='write', iostat=i)
        ! Idiot proofing.
        if (i /= 0) then
            print *, "Error opening file: ", trim(filename)
            stop
        end if

        ! Write data to the file.
        num_lines = size(tau)

```

```
        do i = 1, num_lines
            write(unit_num, '(2e25.15)') tau(i), C(i)
        end do

        ! Close the file.
        close(unit_num)

        print *, "Data successfully written to file: ", trim(filename)
    end subroutine Data_Output

end module Data_Output_Module
```

```
#-------------------------------------------------------------------------------
# Memorial University of Newfoundland, MSc Student
#-------------------------------------------------------------------------------
#
# Program: Cross_Correlation_Plotter
#
#> @Mike-Power666
#> Michael Power
#
# DESCRIPTION:
#> This program simply plots a normalized cross-correlation function and finds
#> a local minima.
#
# REVISION HISTORY:
# 29/06/2024 - Initial Version
#-------------------------------------------------------------------------------

import numpy as np
import matplotlib.pyplot as plt

# Function to read data from the file.
def read_data(filename):
    data = np.loadtxt(filename)
    tau = data[:, 0] / (24 * 3600) # Convert lag time back to days.
    C = data[:, 1]
    return tau, C

# Function to find the index of the local minimum near zero.
def find_local_minima(tau, C):
    zero_index = np.argmin(np.abs(tau))  # Find the index closest to zero
    window_size = 100  # Adjust the window size as needed
    window_start = max(0, zero_index - window_size)
    window_end = min(len(tau), zero_index + window_size)
    local_min_index = window_start + np.argmin(C[window_start:window_end])

    local_min_tau = tau[local_min_index]
    local_min_C = C[local_min_index]
    print(f"Local minimum near zero found at  = {local_min_tau:.15f}, Ć() = {local_min_C:.15f}")

    return float(local_min_tau), float(local_min_C)

# Plotting function.
def plot_data(tau, C, output_file):
    plt.rcParams['text.usetex'] = True  # Enable LaTeX rendering.
    plt.rcParams['font.size'] = 12  # Set default font size for tick marks.
    plt.rcParams['axes.labelsize'] = 20  # Set font size for labels.

    plt.figure(figsize=(10, 6))
    plt.plot(tau, C, label=r'$\hat{C}(\tau)$')
    plt.xlabel(r'$\tau$')
    plt.ylabel(r'$\hat{C}(\tau)$')
    plt.grid(True)

    # Find and plot the local minimum near zero.
    local_min_tau, local_min_C = find_local_minima(tau, C)
    plt.axvline(x=local_min_tau, color='red', linestyle='--', label='Local Minima Near Zero')
    plt.savefig(output_file, format='png')

# Main function.
def main():
    input_file = 'Cross_Correlation_Data.txt'
    output_file = 'Cross_Correlation_Plot.png'

    tau, C = read_data(input_file)
    plot_data(tau, C, output_file)

if __name__ == "__main__":
    main()
```

```
#-------------------------------------------------------------------------------
# Memorial University of Newfoundland, MSc Student
#-------------------------------------------------------------------------------
#
```

```
# Script: Cross_Correlation_Calculator
#
#> @Mike-Power666
#> Michael Power
#
# DESCRIPTION:
#> This bash script compiles, executes, and then cleans up the directory
#> containing the cross-correlation function calculator. Note that it tries to
#> use all cores on a given machine, therefore caution is advised!
#
# REVISION HISTORY:
# 29/06/2024 - Initial Version
#-------------------------------------------------------------------------------

#!/bin/bash

# Determine the number of CPU cores.
NUM_CORES=$(sysctl -n hw.physicalcpu)

# Export OMP_NUM_THREADS to use all available CPU cores.
export OMP_NUM_THREADS=$NUM_CORES

# Compile Fortran files with OpenMP and warnings enabled.
gfortran -fopenmp -Wall -Wextra -c Binary_Search.f90
gfortran -fopenmp -Wall -Wextra -c Data_Out.f90
gfortran -fopenmp -Wall -Wextra -c Linear_Int.f90
gfortran -fopenmp -Wall -Wextra -c Load_Data.f90
gfortran -fopenmp -Wall -Wextra -c Integrand.f90
gfortran -fopenmp -Wall -Wextra -c Romberg.f90
gfortran -fopenmp -Wall -Wextra -c Normalization.f90
gfortran -fopenmp -Wall -Wextra -c Correlation.f90
gfortran -fopenmp -Wall -Wextra -c Cross_Correlation.f90

# Check if compilation was successful before proceeding.
if [ $? -ne 0 ]; then
    echo "Compilation failed... Exiting."
    exit 1
fi

# Link object files with OpenMP and create executable.
gfortran -fopenmp Binary_Search.o Linear_Int.o Load_Data.o Integrand.o Romberg.o \
Correlation.o Data_Out.o Normalization.o Cross_Correlation.o -o Cross_Correlation_Calculator

# Check if linking was successful before proceeding.
if [ $? -ne 0 ]; then
    echo "Linking failed.... Exiting."
    exit 1
fi

# Run the executable.
./Cross_Correlation_Calculator

# Clean up object files and executable after running.
rm *.o Cross_Correlation_Calculator

# Run the python plotting script.
python Cross_Correlation_Plotter.py

echo "Execution and cleanup completed."
```