A Refined Model of Pluto's Atmosphere
(implemented using OSBERT and AMELIA)

by
Stephen Michael Slivan

S.M., Elec. Eng. and Comp. Sci., MIT (1986)
S.B., Comp. Sci. and Eng., MIT (1986)

Submitted in partial fulfillment
of the requirements for the
degrees of

Master of Science
in Earth and Planetary Sciences
and
Bachelor of Science
in Earth, Atmospheric, and Planetary Sciences

at the

Massachusetts Institute of Technology
September 1989

Signature of Author _____
Department of Earth, Atmospheric, and Planetary Sciences
Friday, August 11, 1989

Certified by _____
Professor James L. Elliot
Thesis Supervisor

Certified by _____
Edward W. Dunham
Principal Research Scientist; Thesis Co-Supervisor

Accepted by _____
Professor Thomas H. Jordan
Chairman, Departmental Graduate Committee

Lindgren

A Refined Model of Pluto's Atmosphere (implemented using OSBERT and AMELIA)

by
Stephen Michael Slivan

Submitted to the
Department of Earth, Atmospheric, and Planetary Sciences
on August 11, 1989 in partial fulfillment of the requirements for the Degrees of
Master of Science in Earth and Planetary Sciences
and
Bachelor of Science in Earth, Atmospheric, and Planetary Sciences

Abstract

Photometric data of the UT 9 June 1988 stellar occultation by Pluto obtained with the Kuiper Airborne Observatory [ELLI89]show near the 40% light-level a sudden sharp drop in the observed flux from the star, to a level below that which would be expected from a clear isothermal atmosphere, evidence that the lowest portion of the atmosphere probed by the occultation is not clear and isothermal. It has been proposed that the drop was caused by an extinction layer present in an isothermal atmosphere [ELLI89]. A second suggestion is that the starlight encountered a sharp temperature gradient in a clear atmosphere [HUBB89]. This thesis project involved developing a refined atmospheric model to explore these two alternatives, and comparing models of limiting cases to the KAO data.

First, a geometric reduction of the occultation was performed using the OSBERT program, removing an approximation of constant radial in-shadow velocity over the time interval of the occultation. Then the fairly sophisticated AMELIA atmospheric model was developed to use this information and more rigorously compute model occultation light curves corresponding to atmospheres with haze and/or linear temperature gradients.

Finally, the KAO data were modeled in the two limiting cases of (1) an isothermal atmosphere with a sharp turn-on to a low-lying extinction layer, and (2) a clear atmosphere with a sharp positive temperature gradient in its lowest layer. Least-squares minimization fits were effected on the models using the KAO data.

Results suggest that both the extinction model and nonisothermal model can fit the data very well above and through the observed sharp drop, but that below the drop the extinction model more closely corresponds to the observed data, as the light curve in the nonisothermal case doesn't quite reach the required minimum intensity.

Thesis Supervisor: Dr. James L. Elliot
Title: Professor of Astronomy and Physics
Thesis Co-supervisor: Dr. Edward W. Dunham
Title: Principal Research Scientist

Acknowledgments

During the course of this project I managed to disrupt the lives of a number of people, who for the most part seem to have since recovered. I wish to acknowledge here the more extreme cases:

Ted Dunham repeatedly makes himself available and gives of his time to patiently work to un-confuse me, even in the face of unprecedented density on my part.

We dragged *Tim Shepard* off to Mauna Kea for three weeks last year to help with the astrometric observations; subsequently I commandeered his room in order to use his fast dialup line. He provided me with a copy of the plotting tool he developed for his own thesis [SHEP90], and also cheerfully fields my dumb UNIX questions.

Jack Wisdom permits me the use of POINCARE (so that AMELIA fits can be done in less than a normal human life expectancy) even though it significantly taxes the processor, and also patiently explained to me how numerical integration really should be done.

Leslie Young, who authored the SAFIT fitting program used for the final light curve fits, donated two marathon sessions in the terminal room to customize it for use with AMELIA on Jack's machine.

Thanks, gang.

Table of Contents

List of Figures

List of Tables

Chapter One

Introduction

1.1 Overview

On UT 9 June 1988, Pluto was observed from the Kuiper Airborne Observatory to occult a 12^{th} magnitude star, revealing unambiguously for the first time that the planet has a relatively extensive, albeit tenuous, atmosphere [ELLI89]. High-quality photometric data of the occultation were obtained using the SNAPSHOT CCD imaging system with the KAO 0.9 meter telescope.

The light curve thus obtained from the data shows near the 40% light-level a sudden sharp drop in the observed flux from the star, to a level below that which would be expected from a clear isothermal atmosphere. The drop is evidence that the lowest portion of the atmosphere probed by the occultation is not clear and isothermal. It has been proposed that the drop is caused by an extinction layer present in an otherwise isothermal atmosphere, with a sharp upper boundary due to a sudden 'turn-on' of aerosols [ELLI89]. A second suggestion is that a lightcurve of the observed morphology could be produced without haze, if instead the starlight encounters a sharp positive temperature gradient near the planet surface [HUBB89, ESHL89].

This thesis project involved developing a refined atmospheric model to explore these two alternatives, and comparing models of the limiting cases to the KAO light curve data. Two major pieces of software were designed and implemented as part of this project. The first, OSBERT, was subsequently used to perform the refined geometrical reduction for the light

curve data, and the second, AMELIA, was used to compute refined model light curves to be fit to the observed data.

1.2 Methodology and Thesis Organization

Chapter 2 gives an overview of the occultation prediction and data acquisition for the KAO observations.

Chapter 3 describes the photometric data reduction to produce the light curve, as well as the refined geometric reduction of the occultation as observed from the KAO. This refined reduction was performed in order to remove the approximation of constant radial in-shadow velocity of the KAO during the time interval of the occultation made in [ELLI89], and involved computing an actual in-shadow distance of the KAO from the shadow center to correspond to each observed data point. The OSBERT program, which uses observer, star, and occulting body position information to compute observer in-shadow radii, was designed and implemented expressly to perform these calculations, and is described in detail in Appendix B. Also, the LIBSS library of routines, which OSBERT in turn calls upon to do some of its fundamental computations, is described in Appendix C.

Chapter 4 describes a model of Pluto's atmosphere, whose lower layer can be characterized by extinction due to aerosols, or a linear temperature gradient, or combinations thereof. The model is based on the fairly sophisticated AMELIA model lightcurve generation software developed and implemented for this work, which takes into account change in gravitational acceleration with radius, the far-limb ray contribution, and supports model atmospheres with up to three distinct thermal layers having linear temperature gradients, as well as the extinction in the lowest layer. Finally, the KAO data

are modeled in the two limiting cases of (1) a sharp turn-on to a low-lying extinction layer, and *no* temperature gradient, and (2) a sharp positive temperature gradient in the lowest layer, with *no* extinction. The derivations of the AMELIA model and its implementation are described in detail in Appendix A.

Chapter 5 presents the results of the least-squares minimization fits effected on the models using the KAO data. They suggest that both the extinction model and nonisothermal model can fit the data very well above and through the observed sharp drop, but that below the drop the extinction model more closely corresponds to the observed data, as the light curve in the nonisothermal case doesn't quite reach the required minimum intensity.

Chapter Two

Event and Observations

This chapter includes an overview of the preparations made for obtaining the occultation data, as well describing the data collection itself. The occultation prediction is outlined in Section 2.1, while Section 2.2 contains a detailed description of the observations recorded.

2.1 Event prediction

The UT 1988 June 9 occultation by Pluto was identified several years ago [MINK85] as one of a number of predictions of possible occultations by the planet. It was the eighth event on the list for Pluto; thus the otherwise unnamed target star, lying at the eastern end of constellation Virgo at $\alpha_{2000} = 14^h52^m09\overset{s}{.}9$, $\delta_{2000} = +00°45'03''$, came to be known as "P8". The star was one of the brighter candidates on the list, and the predicted geometry of the event looked to be one of the most promising for yielding an occultation visible from somewhere on Earth.

Subsequent photometry showed P8 to be substantially brighter than originally reported [BOSH86]. Although still rather faint at magnitude +12.3, a good signal-to-noise ratio could be expected for successful observations of the event, and it would also be possible to use smaller portable telescopes to record additional chords [MILL79]. Though observations of only one chord will still yield information about the occulting body and its atmosphere (if present), such additional chords are particularly valuable because taken together, they can provide geometrical constraints important for extracting maximum information from all the observed data.

In order to observe an occultation, the observer must be positioned within the occulting body's shadow; thus the preparation task involves predicting the location of the shadow on Earth so that observing equipment can be located to record the event. In the case of a stellar occultation, the star acts as a light source at infinite distance, so the diameter of the occulting body's shadow will be essentially the same as that of the body itself. Thus, the radius of the body sets an upper limit on positional errors allowable in the predicted shadow track position for the event. The case of Pluto is a particularly challenging one, not only because it's an intrinsically small body (less than 1200 kilometers radius) but also because of its great distance from Earth. Pluto's apparent radius as seen from Earth is only about 0.″6, meaning that measurements of the relative positions of Pluto and P8 need to be at least that accurate if observers located on the predicted center line are to be assured of being *somewhere* in the actual shadow track. Reliably attaining that accuracy is in itself a tough task for current astrometry, but since it's not necessarily even possible to arrange observations from the predicted center line, even less positional uncertainty would be required to afford observers elsewhere in the predicted track a guaranteed place in the shadow.

For the Pluto/P8 event, refined predictions of the track of Pluto's shadow were based on 55 plates taken with the 61-inch Astrometric Reflector at the U.S. Naval Observatory's Flagstaff Station [WASS88]. These calculations indicated that the shadow would cross the south Pacific, New Zealand, and Australia, at about 18 $km \cdot sec^{-1}$. The center of its nearly east-to-west track passed near latitude -22°, east longitude -170°.

A test flight on the Kuiper Airborne Observatory two nights before the occultation event

afforded an opportunity to use the SNAPSHOT CCD imaging system[3] with the KAO 0.9-meter telescope to record frames as additional positional data for Pluto and P8. Immediately after the aircraft returned to Honolulu late that night, analysis was begun on these images in order to provide some late-date additional information toward pinning down the location of the upcoming occultation's shadow track. Fortunately, the shadow track prediction based on these CCD frames corresponded well with that based on the earlier USNO plates.

Pluto's satellite Charon was near southern elongation at the time of the occultation. Because the mass of Charon is not insignificant with respect to that of Pluto, the center-of-mass of the Pluto/Charon system (the *barycenter*, about which the two bodies orbit) is significantly displaced from the center of Pluto itself; this difference cannot be neglected when trying to achieve the accuracy needed to accurately refine a prediction of an occultation by Pluto. Unfortunately, the information needed to pin down Pluto's motion has been difficult to obtain because Pluto and Charon are not resolvable into separate objects through terrestrial telescopes, due to their being so close together. Further, the center-of-light of the composite entity that *is* visible corresponds *neither* to the center of Pluto *nor* to the system barycenter, and the offset from the center-of-light to the barycenter changes continuously as Charon moves in its 6.4-day orbit about Pluto. Thus Charon's presence significantly complicated the prediction effort because in order to produce a meaningful refined prediction, explicit allowance had to be made for the satellite's unknown effect on Pluto's barycentric motion.

---

[3]This same instrumentation would be used for the occultation observations themselves; for details see Section 2.2.

2.2 Observations

In preparation for observing the occultation event, the NASA Kuiper Airborne Observatory, normally based at the Ames Research Center near San Francisco, California, was flown to Hickam Air Force Base in Honolulu, Hawaii. Though the predicted shadow track of the occultation did cross a few land areas from which ground-based observations might be staged, it was felt that the flexibility of being able to choose an observing location based on late-date information, and to position the KAO's 0.9-meter telescope over land or ocean as needed, would prove very valuable toward obtaining high-quality data of this rare event.

The specific intent of the occultation flight plan developed for the KAO was to put the aircraft on the center line of the predicted occultation ground track at the event's predicted midtime of UT $10^h37^m30^s$. It was clear that limits on the range of the aircraft would make it impossible to return directly to Honolulu after the event, since the travel required merely to reach the shadow track about 5000 kilometers south from Hawaii would use more than half of its maximum fuel capacity. The occultation observations would be conducted nearer to the island of American Samoa, so after the occultation the KAO would land there instead and return to Hawaii the following day.

The attempt to reach the predicted center line didn't succeed, however, because shortly before scheduled takeoff from Honolulu, jet fuel began spewing from a vent in the KAO's wing fuel tank. Though the flow stopped of its own accord within a couple of minutes, the subsequent evacuation of the aircraft and fuel cleanup caused an unavoidable delay in takeoff. Once finally airborne, about 15 minutes behind schedule, strong head winds en route prevented the KAO from making up any of the time lost on the ground, so the

occultation observations were carried out while the aircraft was still about 120 kilometers north of the predicted center line.

On the KAO's 0.9-meter telescope, the direct CCD chip on the SNAPSHOT photometer [DUNH85] was set up with an optical reduction factor of 3.3×, which produced an image scale of 1.″1 per pixel. In order to achieve a high signal-to-noise ratio by detecting the maximum amount of light from the occulted star, no filter was used during observations of the occultation, so the wavelength response was that of the CCD chip itself [DUNH85]. The SNAPSHOT data system clock was synchronized with WWVH just before takeoff.

Beginning four hours prior to the occultation observations, a series of 44 full-sized frames (480 rows × 390 columns of pixels), each exposed for 30 seconds, were taken of the field containing Pluto and P8. 22 of these images were exposed with no filter; the other 22 were exposed through an 'R' filter [BESS76]. At the time these frames were recorded, Pluto and P8 were separated by 12″; these frames would be used for resolved photometry of the two objects.

For the occultation itself, a series of 50×50 pixel frames containing Pluto and P8 were recorded at intervals of 0.2 second. Data recording began 10 minutes before the predicted occultation midtime at UTC $10^{h}27^{m}30^{s}$ and lasted for 20 minutes, which produced a total of 6000 images. Following the occultation data series, 16 full-sized frames were exposed with no filter before observations were concluded. On these supplementary frames the images of Pluto and P8 are unresolved.

The airborne seeing at the time of the event produced an image diameter of about 4″ for 50% enclosed light. Since the images of Pluto and Charon were separated by less than an arc second, all subsequent photometry of Pluto effected on these images includes the light

from Charon as well. (Subsequent references to "the light from Pluto" should be understood to mean the *combined* light from Pluto and Charon unless specifically stated otherwise.)

Chapter Three

Data Reduction

This chapter describes the massage of the observational data in preparation for its being compared and contrasted with light curves predicted using modeled atmospheres. A description of the method of photometry used to obtain the light curve is given in Section 3.1, and Section 3.2 describes the reduction of the event's geometry as it pertains to the lightcurve data.

3.1 Photometry

A light curve for the occultation event was derived from the raw CCD images using a 'synthetic aperture' approach. First, the technique of 'marginal analysis' [ELLI88] was used to locate a center for the image of P8 on each frame. Then, the pixel values within a circular synthetic aperture of radius 5 pixels about that center position (corresponding to a field aperture 11″ in diameter) were summed, yielding for each image a total light level corresponding to that of P8 plus that of the sky background visible through the aperture.

In order to subtract the background light level, the contribution of sky brightness to the total light level within the synthetic aperture was estimated by summing for each raw image the pixel values within a circular annulus of sky (inner radius 5 pixels and outer radius 15 pixels) *surrounding* that image's synthetic aperture location. This background total was scaled by the ratio of the areas of the aperture to the annulus, and subtracted from the image's total in-aperture light level to yield a light level for P8 by itself. No explicit corrections to 'flatten' the field were used, as the area of the CCD detector used to record

these images was chosen specifically to afford a uniform response over the region containing P8 and Pluto.

That section of the occultation light curve resulting from this photometry showing both immersion and emersion is shown in Figure 3-1, in which time is plotted along the horizontal axis, and derived total signal level in ADU[4] per 0.2 second integration is plotted along the vertical axis. The signal-to-noise level for these data, defined as the ratio of the unocculted stellar signal to the root-mean-square noise, is approximately 33:1 for each integration. The light curve appears to be symmetric about the occultation midtime, which occurred at UTC 1988 June 9 $10^h37^m27^s$.

3.2 Geometry of the Occultation

3.2.1 Preliminary geometrical solution

In order to divine a structure for Pluto's atmosphere from the obtained light curve shown in Figure 3-1, it was first necessary to establish the relationship of the observed occultation to the center of Pluto. To accomplish this, an approximate solution for the occultation geometry was generated from three data sets: the KAO data first described in [ELLI89], occultation data obtained by the Lowell Observatory expedition to Charters Towers in northeastern Australia, and published timings for the chord obtained from Hobart, Tasmania [WATS88].

Though customarily half-intensity of the occulted star is chosen as an easily-determined light level at which to define a geometrical reference radius, the Tasmanian chord only

---

[4]'Analog-to-Digital Units'; 1 ADU ≈ 14 electrons at gain 1.0. See [DUNH85] for description.

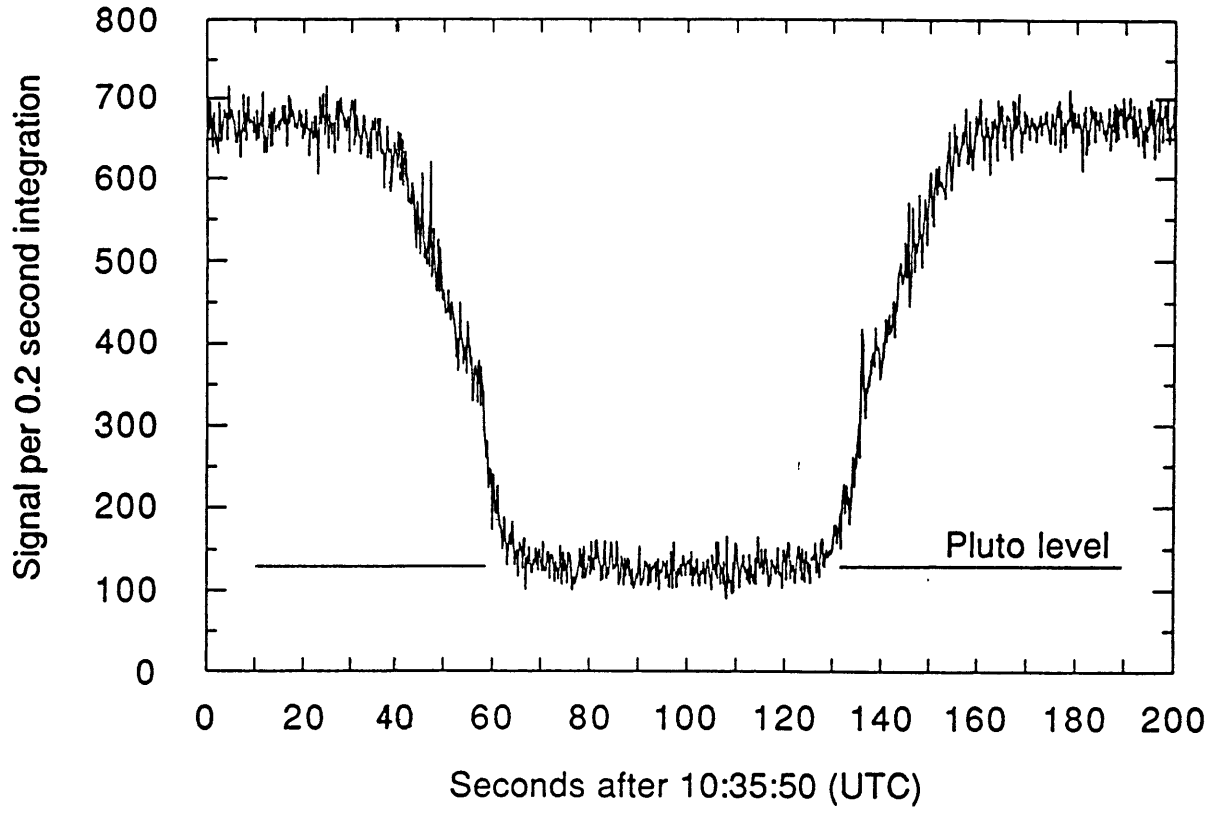

Figure 3-1: Light curve of stellar occultation by Pluto[5]

probed part-way down into Pluto's atmosphere, and thus the resulting light curve did not fall to a level corresponding to half-intensity as observed from the other two sites. Instead, the 0.775 light level, corresponding to half of the drop reported from Tasmania, was used to define the reference radius.

Isothermal fits to the portions of the data above the sharp drops shown in Figure 3-1 yield the 0.775 light times given in Table 3-I for the KAO. The KAO geodetic coordinates, also

[5]figure by A. Bosh, from [ELLI89]

Table 3-I: Station Locations and Occultation times

Station	Event[a]	Longitude (east)	Latitude	Altitude (km)	Time for 0.775 light (UTC)	Radial residual (km)
Charters Towers	I	146°18′26.″3	-20°00′31.″3	0.285	$10^h40^m43^s.7$	-16
	E				$10^h42^m10^s.8$	-12
Hobart	I	147°25′52.″5	-42°50′57.″3	0.310	$10^h40^m32^s.5$	15
	E				$10^h41^m27^s.5$	-16
KAO[b]	I	-170°40′12″	-20°25′06″	12.5	$10^h36^m35^s.1$	8
	E	-170°29′48″	-20°14′48″	12.5	$10^h38^m18^s.8$	21

NOTES:

a) 'I' indicates immersion; 'E' indicates emersion.

b) Positions of the KAO at the event times are raw data from the Omega navigation system, which have a systematic error of approximately 10 to 15 km.

given in Table 3-I, have a systematic error of about 15 kilometers, which was the error recorded by the aircraft's inertial navigation system upon landing in American Samoa after the occultation. Unfortunately, only the magnitude of the error and not its direction was available from the navigator. The coordinates and times of 0.775 light of the other sites are also given in Table 3-I.

A least-squares fit of a circle to the six occultation times from the three data sets used was carried out by minimizing the sum of the squared radial residuals, yielding a reference radius at the 0.775 light level of 1271 ± 9 kilometers. This radius is as measured *in Pluto's shadow* crossing the Earth; in order to find the corresponding distance *in Pluto's atmosphere* measured from the center of the planet itself this value must be corrected for refraction effects by Pluto's atmosphere. (It is important to clearly identify a radius value as to whether it's in-shadow or at-planet; the distinction is made explicitly in this work.)

The radii of the KAO immersion and emersion points are 8 and 21 kilometers outside this circular solution, respectively. Considering that the uncertainty in the KAO position is greater than that of the ground stations, the residuals of the KAO chord can be reduced considerably by assuming that the KAO was actually south of the positions given in Table 3-I by about 15 kilometers. Subsequent geometrical analysis uses positions that correspond to this shift, noting that the difference in the limb velocity in the region of interest is 0.2 $km \cdot sec^{-1}$. The resulting chord as seen from the KAO is shown in Figure 3-2, along with chords corresponding to the other observations used in this preliminary geometrical solution. The dashed circle corresponds to the level in the atmosphere at which dimming of P8 is first detected in the KAO data; the solid circle similarly corresponds to the radius at which light from the star is last detectable.

3.2.2 Radial tagging of data points using OSBERT

Since the KAO chord shown in Figure 3-2 is non-central, it's not strictly appropriate to make an approximation that equal intervals of elapsed-time between individual integrations of data correspond to equal intervals of radius in the occulting body's shadow. In the case of a Pluto occultation this is a particular issue since the extent of the atmosphere is not at all insignificant with respect to Pluto's radius. In order to meaningfully compute a physical model of the atmosphere with which to compare the data, such a model should be computed as a function of in-shadow radius of the observer rather than as a function of occultation elapsed time. To fit such a model requires some method of associating each integrated data point with its corresponding actual radius in Pluto's shadow. A procedure for astrometric analysis of occultations is described in [SMAR77] and makes use of the *fundamental plane* of the occultation, which passes

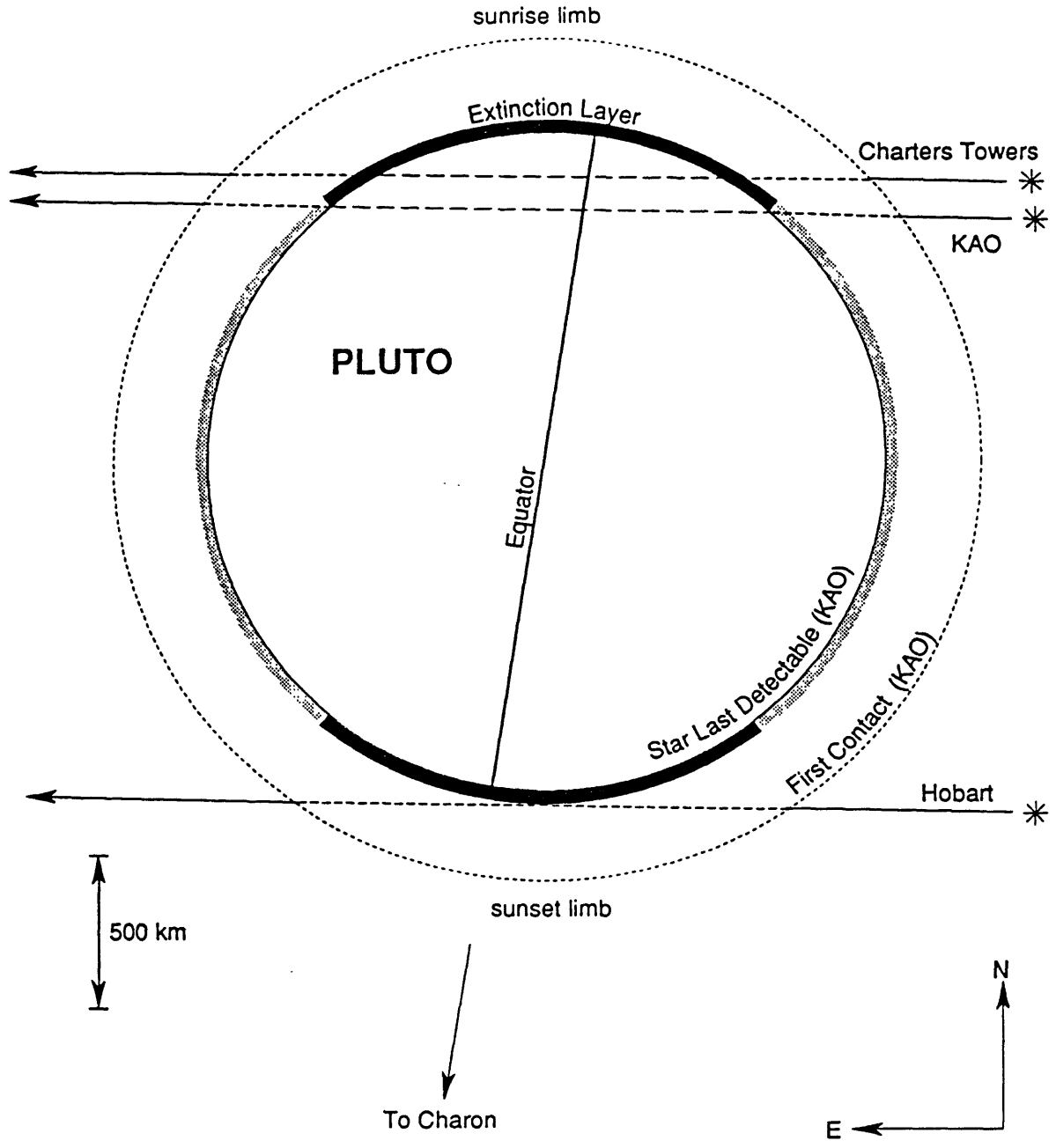

Figure 3-2: Occultation chords used for preliminary geometry[6]

[6]figure by A. Bosh, from [ELLI89]

through the center of the Earth and is perpendicular at each instant to the line from the center of the occulting body to the occulted star. For this work, radial tagging of the observed data was accomplished using the OSBERT program, a tool which was expressly developed to implement the calculations described in [SMAR77]. A detailed description of OSBERT and the calculations it performs is given in Appendix B.

The information needed to compute an in-shadow radius at the time of each data point includes observer position, an ephemeris for the occulting body, corrected as necessary with respect to the occulted star, and occulted star position information from which *its* ephemeris can be computed. In this case, the midtime of each 0.2 sec data integration was used as the time for which the KAO, Pluto, and P8 positions would be calculated to compute the shadow radius corresponding to that data point. For observer positions, the KAO positions at 0.775 light for immersion and emersion given in Table 3-I were supplied to OSBERT, from which latitude, longitude, and altitude were linearly interpolated or extrapolated as needed to compute the KAO position at the time of interest. The position information available for occulted star P8 comes from [MINK85] and consists of only its celestial coordinates: $\alpha_{B1950.0} = 14^h49^m36^s.890$, $\delta_{B1950.0} = +0°57'19''.37$. The Pluto ephemeris used was calculated by Larry Wasserman at Lowell Observatory; that portion used for occultation reduction is given in Table 3-II. Positions given in the table are raw values without any post-occultation corrections.

Table 3-II: Geocentric DE-130 ephemeris of Pluto

TDT	RA	Dec.	Distance (AU)	Epoch
1988 06 08.	14 49 44.0545	+ 0 57 19.783	28.8826310	B1950.0
1988 06 09.	14 49 39.0546	+ 0 57 19.414	28.8923516	B1950.0
1988 06 10.	14 49 34.1287	+ 0 57 18.222	28.9022935	B1950.0
1988 06 11.	14 49 29.2789	+ 0 57 16.202	28.9124538	B1950.0

Corrections to the positions, as from retrospectively examining occultation chords from observers at different locations, are supplied to OSBERT separately. The offsets used for this reduction are given in Table 3-III; they include contributions from both a correction to the Pluto ephemeris used and from the correction to the KAO position described in Section 3.2.1. These values are those which result in a KAO chord minimum in-shadow radius of 855 km along with half-light in-shadow radii which are symmetric about the occultation midtime; it is the application of this correction which yields a half-light in-shadow radius ρ_{ref} of 1169.91 km.

Table 3-III: Ephemeris corrections supplied to OSBERT

ΔRA (sec)	ΔDec (arcsec)	Epoch
0.0361476	0.233965	J1988.4379

The OSBERT computations also require use of a few geometrical constants; the values used in this reduction are given in Table 3-IV.

Table 3-IV: Adopted constants for geometry reduction

Quantity	Value	Reference
Astronomical unit (AU)	$1.49597870 \times 10^{11}$ m	[AA1989, p. K6]
Earth equatorial radius of reference spheroid ($a_{\oplus}$)	6378140 m	[AA1989, p. K6]
Flattening of reference spheroid (f)	0.00335281	[AA1989, p. K6]
Undulation of the geoid	25 m	[KING83, p. 151], giving Goddard Earth Model 10B, 1978
$\Delta T = TDT - UT1$	56.1 sec	linearly interpolated from [AA1989, p. K9]

The computed output was split into immersion and emersion data sets, to correspond exactly with the immersion/emersion split of the observed light curve data to be fit against the models.

Chapter Four

Modeling Pluto's Atmosphere and Occultation Light Curve

To use the occultation light curve as information about the structure of Pluto's atmosphere a traditional modeling approach is used, in which model light curves are computed from a well-defined model atmosphere. Parameter values defining the atmosphere are incrementally adjusted until the model light curve which most closely matches the observed data is produced; values corresponding to this 'best-fit' model will then hopefully be related to the structure of Pluto's atmosphere itself.

Section 4.1 reviews those qualitative aspects of the observed light curve which bear upon the development of models of Pluto's atmosphere which correspond to the data. The parameter sets used to define the two specific atmospheric models fitted to the KAO data in this work are detailed in Section 4.2. (Results of the model-fitting are given in Chapter 5.)

4.1 Morphology of Light Curve

The light curve of Figure 3-1 shows a gradual decline in its 'shoulders' near full intensity which is characteristic of an occultation by a clear isothermal atmosphere, in which the observed dimming of the starlight is caused primarily by differential refraction [BAUM53]. However, as the observed light level from P8 drops slightly below 40% intensity the slope of the light curve increases; here the curve becomes much steeper than would be expected from such an atmosphere. The flat bottom of the light curve similarly does not correspond to the isothermal case. It must be that this lower atmosphere is not

clear and isothermal; it is the behavior of model light curves in this lower portion upon which the limiting-case comparison of the extinction and nonisothermal models effected in this work will focus.

The signal level from Pluto alone, shown in the figure as a solid line labelled "Pluto level", was determined from photometry effected on the CCD images of Pluto and P8 obtained several hours prior to the occultation, on which the images of the planet and star are separable. The observed light curve falls to a bottom level corresponding to the signal level observed from Pluto alone, within a measurement error of 1% of the Pluto signal (equivalent to 0.2% of the signal from P8). This is taken to indicate, within this uncertainty, that a total occultation of the star by Pluto was observed, and means in particular that any model which is to fit the observed data well must be able to reach this bottom level.

4.2 Implementation of Models

4.2.1 Overview

The particular set of parameters used to describe the two Pluto atmospheric models used for this work was chosen to satisfy the broader requirements for exploring combinations of extinction and temperature gradients, as well as the one-or-the-other limiting cases examined here. This general model for use with Pluto consists of two layers with independently adjustable lapse rates, the upper of which is clear atmosphere. The lower layer is additionally allowed a nonzero opacity, if the model is to include extinction. Transition regions for temperature and opacity between the two layers, of independently adjustable thicknesses, are included as well. The parameters used to describe these model atmospheres are given in Table 4-I.

Table 4-I: Parameters describing model Pluto atmospheres and occultation light curves

1. C_{STAR} - (ADU) photometric count from unocculted star
2. C_{BGND} - (ADU) photometric count from background sky
3. ρ_{ref} - (km) reference radius in shadow for light curve flux $\phi = \phi_{ref} = 0.5$
4. Δr_{τ} - (km) half-height of opacity transition layer
5. κ_{34} - (10^{-8}cm^{-1}) reference opacity at r_{34} (parameter 10, below) (if there were no opacity transition zone)
6. f - (unitless) parameter referring to amount of aerosols being created at radius of interest vs. amount created above and merely falling through radius of interest. (see AMELIA Equation (A.18) on page 65)
7. Δr_{34} - (km) half-height of temperature transition region between upper layer ("Layer 3") and lower layer ("Layer 4")
8. Γ_3 - (K · km^{-1}) lapse rate of upper layer
9. Γ_4 - (K · km^{-1}) lapse rate of lower layer
10. r_{34} - (km) radius of boundary between upper and lower layers
11. T_{34} - (K) temperature at layer boundary r_{34}, if there were no transition region

4.2.2 AMELIA parameter 'unwrapper'

The AMELIA implementation of a parameterized atmospheric model[7] was used to compute the model lightcurve corresponding to the atmosphere determined by a given collection of values for the parameters of Table 4-I. The detailed derivations of the atmospheric and lightcurve generation computations implemented with AMELIA are given in Appendix A.

As described in Section A.2.2, using this software requires that the mapping from variable external model parameters to internal AMELIA parameters be specified; the following

[7]from whence come the as yet unexplained references in Table 4-I to "Layer 3" and "Layer 4"

description of that mapping for the models of interest here uses the parameter notation defined there on page 52.

The approach used here is to put the lower boundary of AMELIA Layer 1 up high enough so that the outermost of the star's light rays detectably dimmed by refraction have radii of closest approach to Pluto already below the layer's lower boundary. Thus, the width of its temperature transition region to underlying Layer 2 is irrelevant and is arbitrarily set to 0:

$r_{12} \equiv 3000$ km (just to get it out of our way)

$\Delta r_{12} \equiv 0$ km

Layers 2 and 3 of AMELIA are "glued together" into one layer, which will be used as the upper clear layer in the fitted models; the boundary between Layers 2 and 3 is arbitrarily set to be halfway up the combined layer:

$$r_{23} \equiv \frac{r_{12}+r_{34}}{2}$$

$\Delta r_{23} \equiv 0$ km

$\Gamma_2 \equiv \Gamma_3$

Finally, AMELIA Layer 4 will be the extinction/nonisothermal layer, with adjustable opacity and lapse rate.

Of the remaining parameter unwrapper relations which need to be specified, those for C_{STAR}, C_{BGND}, ρ_{ref}, Δr_τ, f, Δr_{34}, Γ_3, Γ_4, and r_{34} are all trivial direct assignments. The reference opacity κ_{34} is specified externally in units of 10^{-8}cm^{-1}, so its value needs to be scaled back to cm^{-1} for use in the optical depth computation. The remaining internal parameter T_{12} is related to external parameter T_{34} in the following way:

$$T_{12} = T_{34} - \Gamma_3(r_{12} - r_{34})$$

Although all of the external model parameters are in theory able to be fitted parameters, for

the two models used in this work only two or three parameters will actually be fit; the others will be 'configured' to fixed values appropriate for the model in use. Both models will use $\Gamma_3=0$, making the upper clear layer isothermal with temperature T_{34}. Similarly, $\rho_{ref}=1169.91$ km as computed from the geometry data reduction, and the C_{STAR} and C_{BGND} photometric counts from the fits of [ELLI89, Table II]. The separate fits to immersion and emersion effected for this work will use the corresponding immersion/emersion values tabulated there. Specific handling of the remaining external parameters depends on which of the two models (extinction or nonisothermal) is being fitted, and is described below separately.

4.2.3 Isothermal model, with haze

For this hazy isothermal case, the lapse rate Γ_4 for the lowest layer is set to 0, so the value of Δr_{34}, specifying the thickness of the temperature transition layer to lapse rate Γ_3, is irrelevant.

The parameters Δr_τ and f were assigned fixed values, chosen so as to approximate the occultation lightcurve resulting from fits to the extinction model of [ELLI89]. The opacity in that original model has an abrupt turn-on; to approximate this effect with AMELIA while still maintaining continuity of opacity with radius, the opacity transition layer is assigned a half-height of only 20 meters. This is smaller than the radial resolution of the observed data in that region of the lightcurve, and causes r_{34} to effectively be the radius of the top of the extinction layer. A value of 0.3 for the other opacity parameter f was chosen by examining trial light curves produced with AMELIA using values of 0.0, and 0.1 through 10.0 stepping by factors of about 3, also noting in the process that $f > 0.3$ appears to be unsatisfactorily highly correlated with the value chosen for κ_{34}. This model assumes an atmosphere composed exclusively of methane [HUBB88, ELLI89].

Fixing Δr_τ and f in this way leaves three fitted parameters: the reference opacity κ_{34}, the radius of the top of the extinction layer r_{34}, and the atmospheric temperature T_{34}.

4.2.4 Clear model, with temperature gradient

For this clear nonisothermal case, the reference opacity κ_{34} is set to zero, so the values of the remaining opacity-related parameters Δr_τ and f are irrelevant.

The parameters T_{34} and Δr_{34} were assigned fixed values, chosen so as to approximate the occultation data lightcurve using the temperature profile given as a solid line in [HUBB89, Fig. 3]. The model fitted assumes an atmosphere of carbon monoxide [YELL89] which also contains enough methane to hold the temperature of the outer isothermal gas at 106 K, thus fixing T_{34} at that value. A value for the temperature transition layer thickness was determined by experimentation; initially, a trial model light curve computation using a temperature transition zone half-height of 7 km and a lapse rate for the underlying nonisothermal layer Γ_4 of -7.5 $K \cdot km^{-1}$ was run. The model lightcurve obtained using this atmosphere did in fact display a sudden drop in the expected place, but the morphology of the lightcurve diverged from the KAO data in two ways: (1) the flux past the drop did not fall sharply enough, and (2) the model light curve 'bottomed out' and leveled off before reaching the flat bottom evidenced in the observed lightcurve. Subsequent experimentation with transition layer sizes showed that shrinking the transition layer would cause a corresponding increase in the steepness of the lightcurve drop. A half-height of 0.3 km produces a slope which seems to correspond fairly well to that of the data; this is the half-height that would subsequently be used in the fitted model.

Fixing T_{34} and Δr_{34} in this way leaves two fitted parameters: the radius of the temperature transition layer r_{34} and the lapse rate of the low nonisothermal layer Γ_4.

The second mismatch seen in the trial models, that of the flux level at the bottom of the lightcurve, was unable to be completely overcome. In this case it appears that the lapse rate itself is the model parameter which affects this bottom level, and though by increasing the lapse rate it was possible to decrease the flux level, successive increases to the lapse rate had progressively smaller effects on the lightcurve.

Chapter Five

Results and Conclusions

5.1 Model fits to the data

The two models described in Chapter 4 were fitted by least-squares minimization to KAO data intervals spanning 94.4 seconds of elapsed observing time. These intervals, one each for immersion and emersion, correspond to the shadow radius interval 855 km to 1985 km, or Pluto radius interval 1160 km to 1985 km; the outermost 300 km or so correspond to flux which is not detectably refracted. The immersion/emersion split was chosen to be at the minimum shadow radius of the data, so that there is no overlap between the intervals; this split point corresponds closely to the midtime of the occultation as observed from the KAO.

Table 5-I: Adopted parameter values for model atmospheres

Parameter	Symbol	Value	Reference
Mass of Pluto	M_{Pluto}	$(1.29 \pm 0.11) \times 10^{25}$g	[ELLI89]
Radius of Pluto	R_{Pluto}	1142 ±21 km	[THOL88]
Earth/Pluto distance	D	4.323×10^{9} km	[ELLI89]
Mean molecular weight of methane	$\bar{\mu}$	16 amu	
Refractivity of Methane[a]	ν_{STP}	4.38×10^{-4}	[ELLI89]
Mean molecular weight of carbon monoxide	$\bar{\mu}$	28 amu	
Refractivity of carbon monoxide	ν_{STP}	3.34×10^{-4}	

NOTES:

a) Mean value, integrated over the spectral response of the SNAPSHOT CCD chip.

The extinction model fits are shown in Figures 5-1 and 5-2, along with data and residuals;

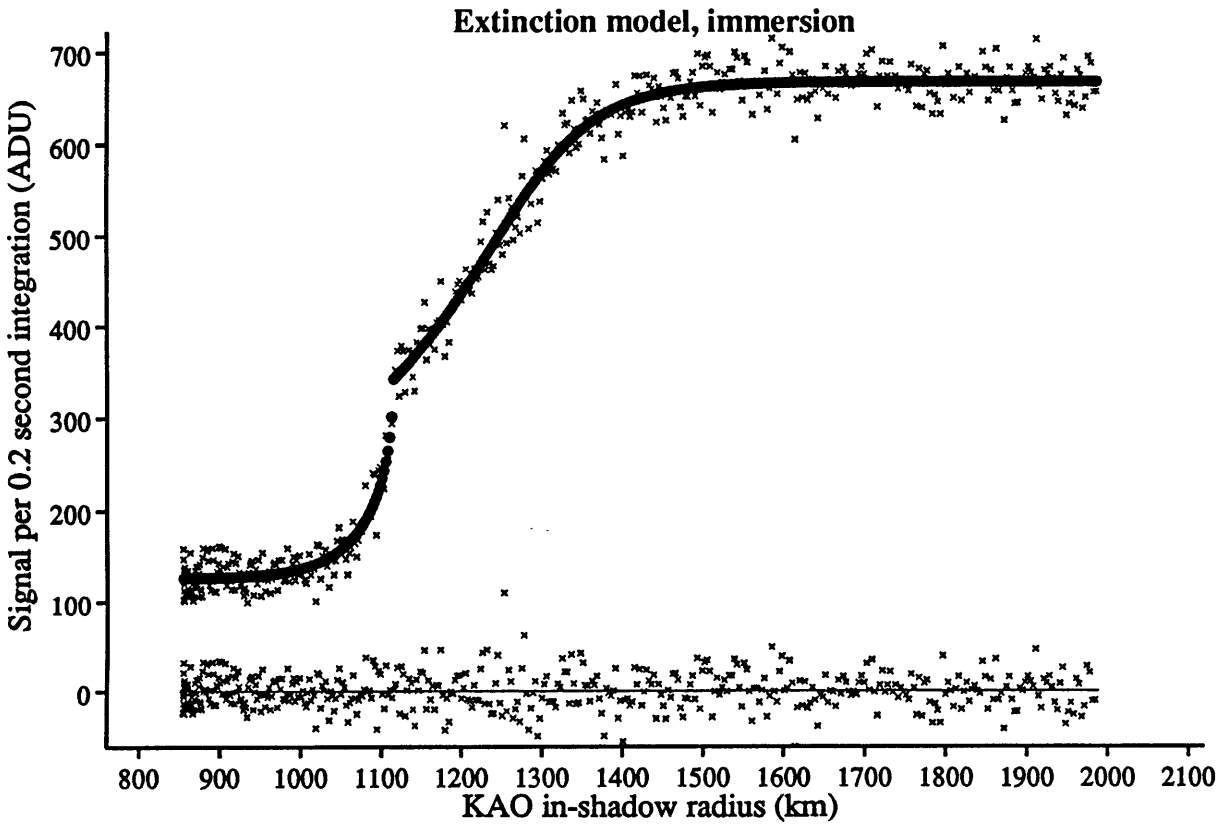

Figure 5-1: Isothermal-extinction model, data, and residuals (immersion)

corresponding plots for the nonisothermal model are shown in Figures 5-3 and 5-4. Note that the observed lightcurve data in these figures are exactly those plotted in [ELLI89, Fig. 3], but that the horizontal time axis in each of the original plots has been replaced by an axis indexed by the KAO's radial distance from the center of Pluto's shadow as computed using OSBERT. Since the KAO chord in Figure 3-2 was non-central, this 'radius-domain' plotting of the data has the effect of stretching out the upper part of the curve while at the same time compressing the lower portion, as compared with the earlier plots using the 'time-domain'.

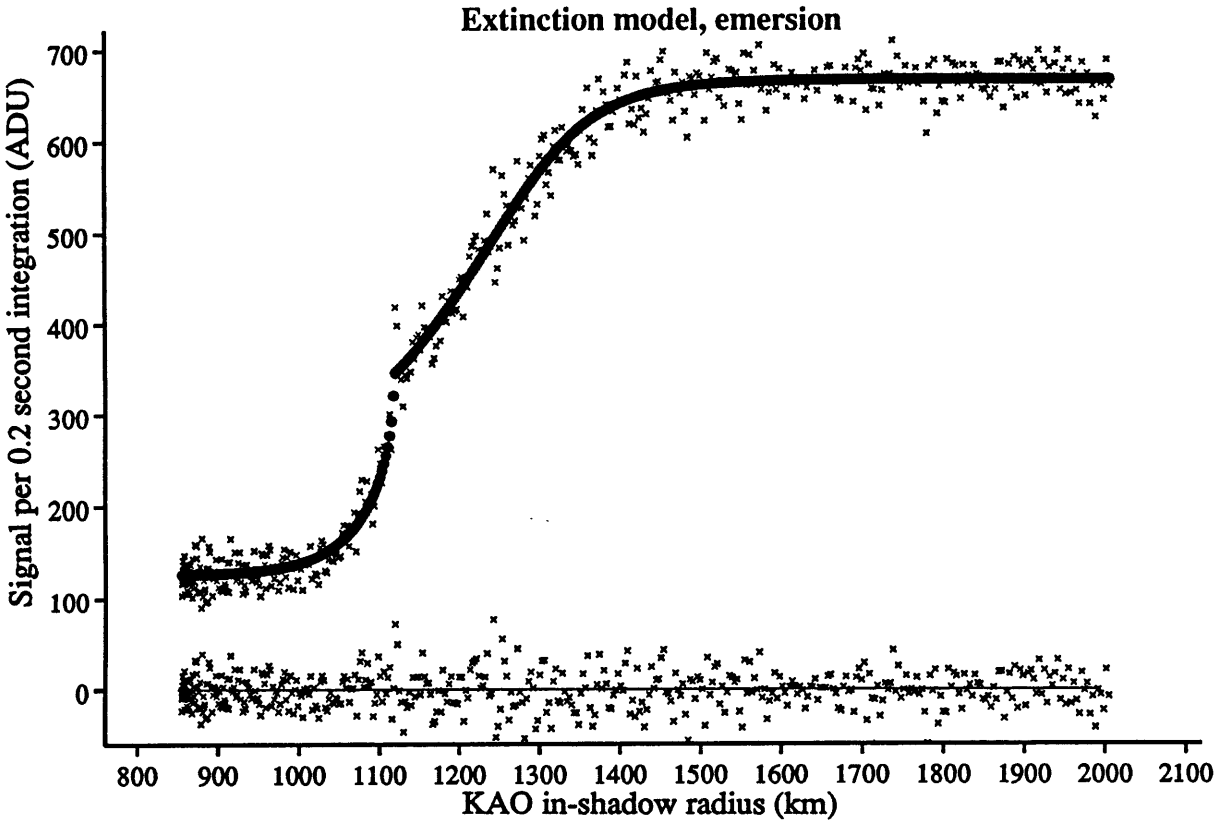

Figure 5-2: Isothermal-extinction model, data, and residuals (emersion)

5.1.1 Isothermal-extinction model

The "hazy model" light curve was initially fit with three free parameters: the temperature of the isothermal atmosphere, the radius of opacity turn-on, and the reference opacity value at the turn-on radius. Initial values for these parameters were determined "bye eye" from trial model light curve plots compared with the KAO data. Though the parameter values for both immersion and emersion converged in a straightforward manner, the *formal errors* for the temperature and radius values tended to oscillate once value convergence was attained. It is thought that this is an artifact caused by the fitting algorithm interacting with

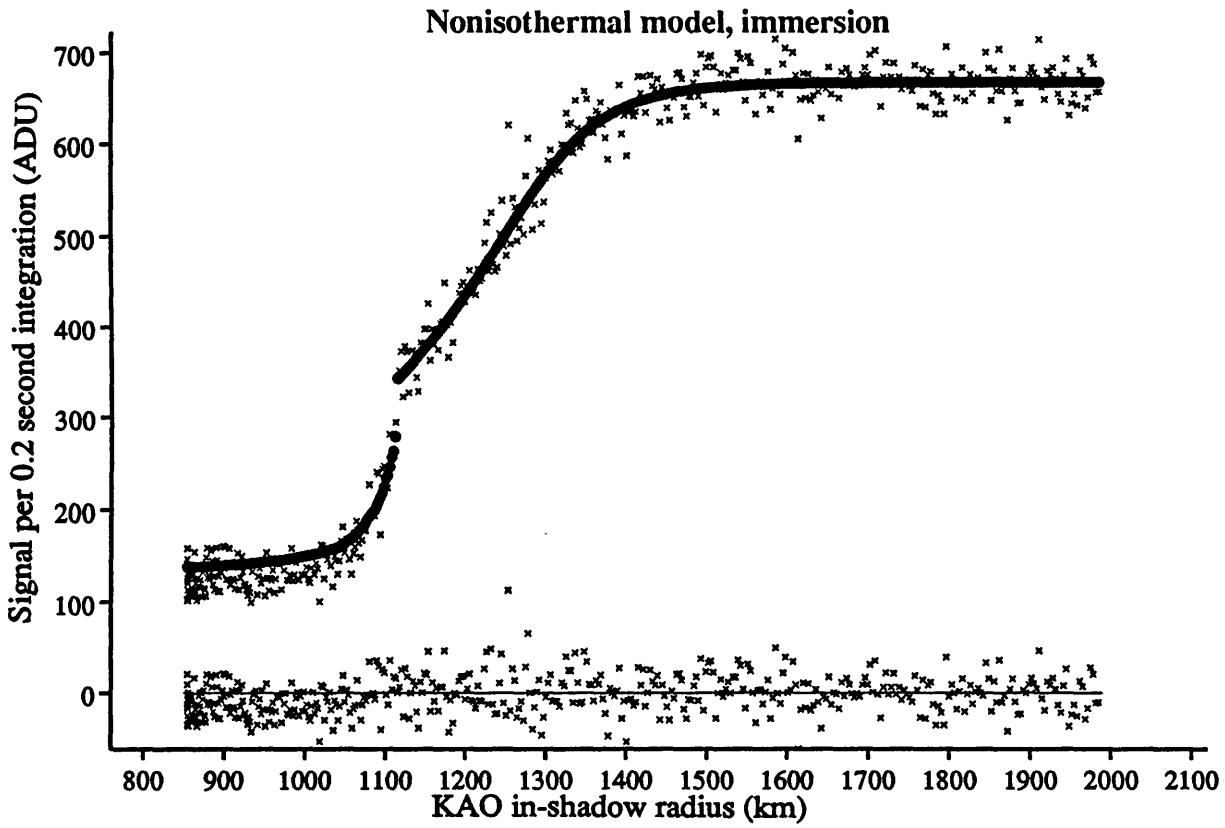

Figure 5-3: Clear nonisothermal model, data, and residuals (immersion)

the data, as it adjusts the fitted radius back and forth across the sharp drop of the lightcurve. To better define the formal errors, a second fit was used with the radius parameter fixed to the converged value from the initial fit. The values of the fitted extinction model parameters are given in Table 5-II, along with their formal errors.

5.1.2 Clear nonisothermal model

For the nonisothermal case modeled here, the upper atmosphere temperature was fixed at 106 Kelvin, so the two free parameters were radius of the upper boundary of the nonisothermal layer and its lapse rate. As was done for the extinction model, initial values

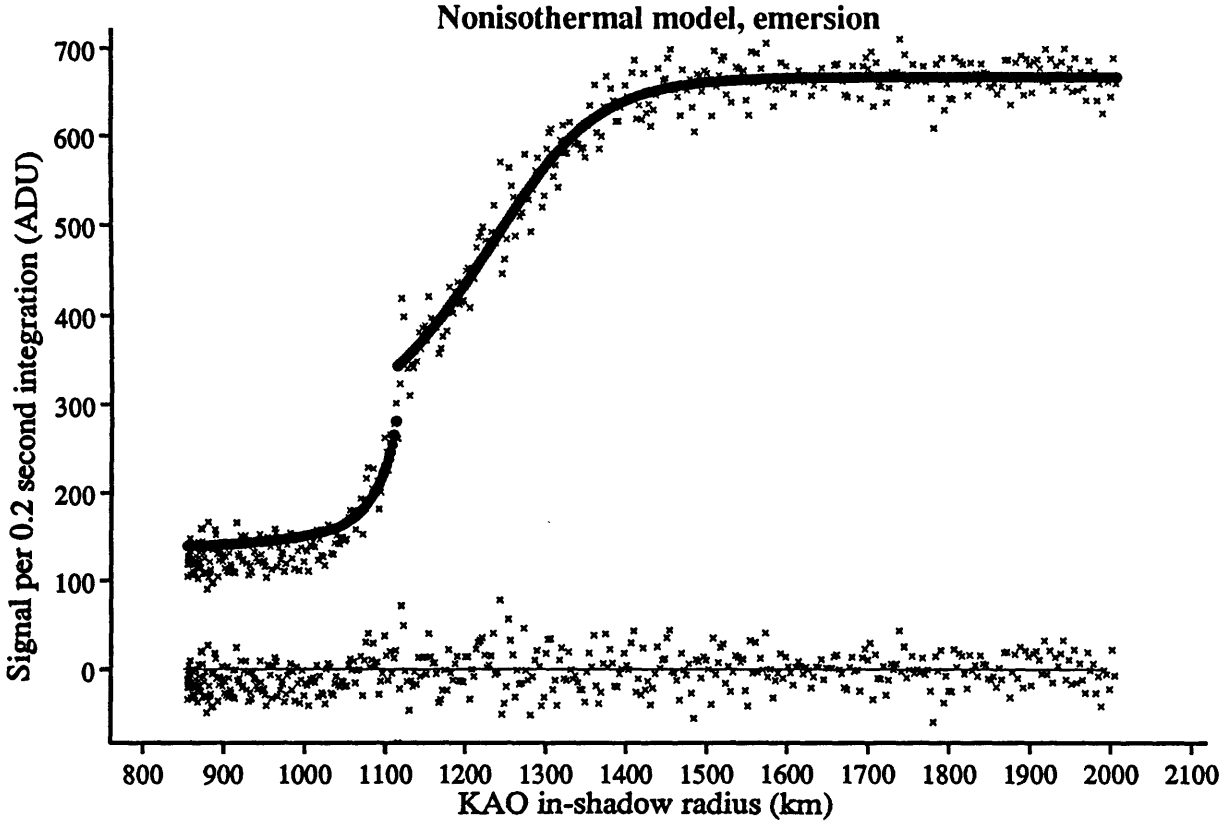

Figure 5-4: Clear nonisothermal model, data, and residuals (emersion)

Table 5-II: Results of isothermal-extinction model fits

Model Parameter	Units	Immersion	Emersion	Weighted Mean
Isothermal atmosphere				
Temperature at extinction turn-on (T_{34})	Kelvin	59.89 ±0.25	60.75 ±0.44	60.10±0.22
Extinction Layer				
Center of turn-on transition layer (r_{34})	km	1213.94 ±0.33	1216.47 ±0.84	1214.68 ±0.31
Opacity at r_{34} (κ_{34})	10^{-8}cm^{-1}	2.98 ±0.16	2.61 ±0.14	2.77 ±0.11

for parameters came from trial plots, but unlike the first model, the fits themselves had a

tougher time trying to reach a 'best fit' to the data: as mentioned in the nonisothermal model description, it appears that it is the lapse rate of the nonisothermal layer which determines at what flux level the model light curve levels off at its bottom. Though initially the fitting program improved the fit by using a progressively more negative lapse rate, each change in the lapse rate produced successively less improvement, until finally the goodness-of-fit actually indicated the fits were getting *worse* even though the lapse rate was still being increased. Table 5-III gives the results of the "best fit" able to be achieved with this model, and Figures 5-3 and 5-4 show that the model light curve corresponding to even this 'least-worst' fit does not correspond well in its lower portion to the morphology of the corresponding part of the KAO data light curve.

Table 5-III: Results of clear nonisothermal model fits

Model Parameter	Units	Immersion	Emersion	Weighted Mean
Nonisothermal atmosphere[a]				
Center of temperature transition layer (r_{34})	km	1213.44 ±0.14	1212.53 ±0.66	1213.40 ±0.14
Lapse rate of nonisothermal layer (Γ_4)	$K \cdot km^{-1}$	-37.01 ±6.30	-32.41 ±6.93	-34.93 ±4.66

NOTES:

a) Upper isothermal layer set to 106 Kelvin.

5.2 Discussion of Results

First, results from these two limiting-case models are compared and contrasted with each other, Then, the results of the extinction model are compared and contrasted with those from the extinction model described in [ELLI89].

5.2.1 Extinction model vs. Nonisothermal Model

The upper portion of the extinction light curve is essentially indistinguishable from that of the nonisothermal model; this is because that portion of the curve corresponds to the clear isothermal atmosphere layer, whose structure is determinable only to $\frac{T}{\bar{\mu}}$. The value of this ratio corresponding to the best-fit extinction model is 3.8, identical to the value for the nonisothermal model fixed by assuming the CH_4 + CO composition of $\bar{\mu}=28$ amu and $T=106$ K.

The results of primary interest here though have to do with the sharp drop and bottom level of the models. It appears that the steep drop of the data can be made to match well with both the clear model, by keeping the change in height over which the lapse rate changes small, and in the extinction model, by analogously keeping the change in height over which the extinction becomes significant small. The radius of the significant boundary in each of the two best-fit models correspond fairly well at 1213 km and 1215 km for the nonisothermal and extinction models, respectively. On the other hand, the residuals plots show clearly that the immersion and emersion best fits of the light curve corresponding to the clear model, characterized by a single lower nonisothermal layer of constant gradient (Figures 5-3 and 5-4), are in each case inferior to those of the isothermal atmosphere with extinction (Figures 5-1 and 5-2). In the nonisothermal case, the best-fit lapse rate is not steep enough to allow the bottom of the light curve to reach the required zero intensity. There is an additional difficulty with this particular result, which becomes apparent when reconciling this model with the radius of Pluto as derived from observations of the Pluto-Charon mutual eclipse events: the steepness of the temperature gradient needed to achieve best-fit to the data forces the temperature to implausible values long before reaching the

radius of 1142 ±21 km [THOL88], even with the estimated 20 km uncertainty in the absolute positioning of the radius scale determined by the model described here. Nor does the limiting case extinction-only model appear to be a fit without discrepancy; though the extinction model immersion plot residuals show quite an even distribution about the zero line throughout the interval plotted, the extinction emersion plot shows in the range of 1000 km to 1050 km in the shadow (corresponding to about 1185 km to 1197 km at Pluto, ±20 km) a small but definite negative tendency of the residuals.

5.2.2 Extinction model vs. previous models

Though the two particular models investigated in this work correspond to each other fairly well with respect to the physical structure and lower boundary of the upper clear isothermal layer, comparing this structure with that determined using the extinction model of [ELLI89] reveals some significant differences whose origin is not completely clear at this time. The most severe disagreement occurs with respect to the radius scale of the atmosphere, when comparing the radii of half-light (1238 km vs. 1214 km) and of the top of the extinction layer (1215 km vs. 1189 km), both of which are approximately 25 km higher in the new models fit here. A second discrepancy appears in the temperature of the atmosphere (60 K vs. 67 K), although the values do marginally correspond within their estimated uncertainty of 6 K, and the new model's temperature for a pure methane atmosphere corresponds to that of the 61 K "warm model" of [HUBB88]. It is clear though that the new model *does* fit the data, and it does so with residuals comparable to those of the fits of the original model.

Some difference is to be expected between the results of these two extinction models, since the variation of gravitational acceleration with height and the varying of the component of

star apparent velocity perpendicular to the limb of Pluto are handled explicitly in the new model, whereas the earlier model assumed them constant for purposes of computation, to the results of which a correction was incorporated. It is felt though that the magnitude of the differences revealed in the fitting results is uncomfortably large and is thus likely due to some error or inconsistency in one or the other of the models, or in the geometric data.

The first suspect of course is the modeling software itself, particularly as it's new. A problem with OSBERT's radial tagging of the data points certainly could cause a shift of radii in the model atmosphere, or, a problem in the *input* information could produce such symptoms. It is felt that the first scenario, an OSBERT error, is unlikely inasmuch as that program performs simple arithmetic and table-lookup operations which were easily checked exhaustively and independently. The fortuitous circumstance of an easily observable lunar occultation at just the time OSBERT was being completed afforded an valuable opportunity for a 'field trial'; the program computed circumstances for the occultation of λ Aqr on UT 18 Nov 1988 yielding an immersion time accurate to within 2 seconds of the observed time.[8]

On the other hand, the least well-established of the event geometry-related input data supplied to OSBERT and subsequently used by the new model are the KAO positions during the observations, and the new model crucially depends on the observer location relative to the occulting body's shadow. It is now known that there are significant discrepancies among the raw KAO position data. Indications are that the preliminary geometrical solution of the occultation first computed for [ELLI89] and described in detail in Section 3.2.1 needs to be reexamined, particularly the information having to do with the location of the KAO during the event.

---

[8]For this computation, a lunar radius of 1738 km was used.

It's also true that computational errors in the AMELIA model light curve function could cause discrepancies between the results of the old and new models. However, the bending angle and flux results of a clear isothermal test case have been checked as being self-consistent, using an assumed reference number density, when James Elliot successfully and independently duplicated the AMELIA results using the Mathematica Program to perform the numerical integrations based on the original equations given in Appendix A. Admittedly this does not in any way check AMELIA's computation of the reference number density, but the half-light number density of 8.4×10^{13} molecules$\cdot$cm^{-3} computed by the new model corresponds extremely well with the 8.3×10^{13} molecules$\cdot$cm^{-3} given in [ELLI89, Table IV] for the old model and thus does not even present the appearance of being broken.

5.3 Conclusions

The results of the least-squares minimization fits effected on the models using the KAO data suggest that both an extinction model of essentially the same type as that described in [ELLI89], and a nonisothermal model consisting of an abrupt transition to a temperature layer of steeply negative but constant lapse rate, can fit the observed data very well above and through the observed sharp drop. However, below the drop the extinction model more closely corresponds to the observed data, as the light curve in the nonisothermal case is unable to drop to the required minimum intensity. Additionally, the information about the physical structure of Pluto atmospheres corresponding to the two limiting case models explored in this work as summarized in Tables 5-IV and 5-V shows that a nonisothermal model of this type which does in fact approximate the observed light curve with a dimming of the required steepness, even though it stops dropping before it reaches zero flux, results

in such a strongly negative lapse rate that implausibly low temperatures are reached too high in the atmosphere to be reconciled with constraints on Pluto's physical structure obtained from other observations.

Table 5-IV: Structure for Pluto's Atmosphere (Extinction Model)

Clear Atmosphere at the half-light level	
radius of half-light	1238 ±20 km
temperature	60 ±6 Kelvin
scale height	55.5 ±6.9 km
number density	8.4×10^{13} molecules · cm^{-3}
Extinction Layer	
top	1215 km[a]
unit optical depth	1207 km[a]

a) Although the placement of the radius scale has an uncertainty of ±20 km, the relative radii of various levels should have uncertainties no greater than 1 to 2 km.

Table 5-V: Structure for Pluto's Atmosphere (Nonisothermal Model)

Clear Atmosphere at the half-light level	
radius of half-light	1237 ±20 km
temperature	106 Kelvin (adopted)
scale height	55.1 ±6.9 km
number density	1.1×10^{14} molecules · cm^{-3}
Clear Model Nonisothermal Layer	
top	1213 km[a]
lapse rate	-35 K · km^{-1}
radius of temperature = 40 K	1211 km[a]

a) Although the placement of the radius scale has an uncertainty of ±20 km, the relative radii of various levels should have uncertainties no greater than 1 to 2 km.

Significant differences between the best-fit extinction model of this work and that of [ELLI89] are unexplained at this time, but appear to be at least in part related to an inconsistent treatment of the event geometry in that first model. In any case, these differences do not appear to significantly impact this work's basic comparison of extinction and nonisothermal mechanisms as explanations for the morphology of the observed Pluto stellar occultation light curve.

Appendix A

AMELIA function

AMELIA

Release 1.0

Application User Guide and Internal Design Documentation

Stephen Slivan

MIT, Department of Earth, Atmospheric, and Planetary Sciences,

Cambridge, MA 02139

(slivan@lcs.mit.edu)

Concept and original derivations from handwritten notes by Ted Dunham.

$Revision: 0.12.1.1 $; Last submitted $Date: 89/08/12 20:09:20 $ $Author: slivan $

Roadmap:

- Section A.1 is an overview of what AMELIA does.
- Section A.2 contains specific information about using AMELIA, helpful for a user (hopefully). It includes descriptions of its software interface and configuration file format.
- Section A.3 contains the detailed derivations and other information about the equations and algorithms implemented with AMELIA.

A.1 Overview

Concept: AMELIA (*Atmospheric Model with Extinction Layer and Intractable Arithmetic*) refers to a sophisticated function coded in C, which implements a parameterized atmospheric model. It was developed to satisfy the requirements for modeling stellar occultation lightcurves corresponding to a Pluto atmosphere clear and isothermal in its upper part, but including also a lower layer which may be characterized by extinction due to aerosols, and/or a nonzero temperature gradient. Additional capability is available for modeling lightcurves produced by an atmosphere with up to three nonisothermal temperature layers, each having its own constant gradient of temperature with radius, hopefully making AMELIA suitable for modeling Uranus' atmosphere.

As an isolated function AMELIA can't run "stand-alone"; it's intended to be called by some application, such as a fitting or plotting program. Its results can be used as a modeled lightcurve to be fit to observational stellar occultation data, or to generate plots of such modeled lightcurves.

In order to use AMELIA to generate modeled lightcurve flux values, the calling application supplies the following input information to fix the models parameters:

- temperature profile parameters
- opacity profile parameters
- reference radius & normalized flux level
- photometric counts from occulted star and background

The `amelia` function itself takes as one of its calling arguments an in-shadow radius representing the distance of the observer from the center of the occulting body's shadow; the function computes a lightcurve flux value for that radius.

The atmospheric model explicitly handles the variation with height of gravitational acceleration g, but still assumes constant mean molecular weight μ for the atmospheric gas. For lightcurve generation, ray convergence appropriate for a spherical planet is used, and the flux contribution from the far-limb ray is included. As the gas composition is taken to be constant, its refractivity ν_{STP} will be constant as well. Further, note that the approximation $H(r) << r$ is *not* made.

A.2 External Aspects

AMELIA's interaction with an application is characterized by the following:

- the `amelia` function call itself
- a model parameter value 'unwrapping function'
- a 'configured values' file

In this section, each of these three aspects is described in detail.

Before proceeding it will be useful to define a uniform notation for the parameters associated with the various layers of the AMELIA implementation of this atmospheric model:

Atmospheric layers are numbered sequentially, beginning with 1 for the outermost layer, and increasing for lower layers. A transition region has a 2-digit number identifying the two layers joined by the region (e.g., region 12 joins layers 1 and 2). The following convention is used for denoting layer boundaries, transition layer thicknesses, and lapse rates:

r_{mn} — radius of center of temperature transition zone *mn*. Radius at which a discontinuous slope in temperature profile would exist between atmospheric layers m and n in the absence of a transition-zone layer. Layer m sits above r_{mn}; layer n lies below it.

Δr_{mn} half-height of transition-zone layer between atmospheric layers m and n. Center of zone lies at radius r_{mn}, so zone extends from radius $r_{mn}-\Delta r_{mn}$ up to $r_{mn}+\Delta r_{mn}$.

Γ_n lapse rate corresponding to atmospheric layer n

A.2.1 Software interface

The `amelia` function itself has the interface specification given in Figure A-1.

Figure A-1: C interface specification of `amelia` function

```
int
amelia(changed,fpvals,rs,rtnl)
   boolean changed;  /* TRUE iff one or more of [fpvals]
                        changed since last call */
   double fpvals[];  /* current values for fitted parameters */
   double rs[];      /* radii in shadow (km) */
   double *rtnl;     /* returned value of light-curve flux */
 /*-------------------------------------------------------
 REQUIRES: [fpvals] correspond with .mfit file being used.
   Radii (from an OSBERT file) [rs] be as follows:
      rs[0] = radius immediately preceding rs[1] in OSBERT file
      rs[1] = target radius of interest
      rs[2] = radius immediately following rs[1] in OSBERT file
 MODIFIES: *rtnl
 EFFECT: Computes a scaled star flux value corresponding to
   shadow radius rs[1] computed as per the current
   atmospheric model.  Integration-time smoothing capability
   is included.  The amelia function returns SUCCEED if no
   errors were detected while performing the computations;
   FAIL is returned otherwise.
 IMPLEMENTATION NOTES:
   It is important to note that the four GLOBAL STRUCTURES
   [mpvals], [mpold], [mcvals], and [global] are defined
   outside of all C functions used here and thus would COLLIDE
   with same-named variables if they are also defined
   outside of all functions.

   Local state memory has been implemented to save some
   computations by re-using values of ns34, k34m, and n0
   when possible.
  -------------------------------------------------------*/
```

This interface was determined by the needs of the SAFIT fitting program, with which AMELIA has been expressly designed to be used.

Note that the radius argument `rs` is in fact a 'triple' of three radii which SAFIT sends along from the user-specified 'coordinate file' corresponding to the 'data file'; this is used to support integration-time smoothing of the flux values produced by AMELIA. As of this writing, the only source of such a radius coordinate file is the OSBERT program (see Appendix B). Also, note that a configured value N_INT controls whether integration-time smoothing is used; in particular, if N_INT set to 1, the particular values supplied for `rs[0]` and `rs[2]` will not affect the computed flux value.

A.2.2 Parameter "wrapper"

In order to try to accommodate some moderate degree of generality with respect to a user selecting a set of model external parameters (e.g., to be fit), while at the same time keeping a fixed set of parameters[9] which are used internally to specify the atmospheric model in use, the design of AMELIA includes a 'parameter unwrapping' module. Its only purpose is to convert a given set of assigned values of the external parameters to the set of corresponding values of the AMELIA internal parameters. Thus, when a user specifies an external parameter set, it's necessary to also specify the order in which a set of values for these external parameters is passed to `amelia` via its `fpvals` argument, as well as the precise mapping from external to internal parameters, and to code up a corresponding unwrap module which implements the conversion. The specification for the unwrapping function is given in Figure A-2. The internal model parameters which must be computed and returned to AMELIA through `rtnl` (structure of type `MPTYPE`, typedef'd in "amelia.h") by the wrapper are given in Table A-I (possibly helpful to refer to Figures A-4 and A-5 on pp. 68 and 71). Currently the unwrapper function exists as a separate source module named "unwrap.c"; see it as an example of a coded unwrapper function.

[9]NOT a 'set of fixed parameters'!

Figure A-2: C interface specification of `unwrap_params` function

```
EXCEPTION
unwrap_params(p,rtn1)
   double p[];
   MPTYPE *rtn1;
 /*-----------------------------------------------------------
 SIGNALS: not_possible
 REQUIRES:
 MODIFIES: *rtn1
 EFFECT: Computes parameter values used by model computations
   from external parameter values provided by the calling
   application program.
 IMPLEMENTATION NOTES:
  -----------------------------------------------------------*/
```

A.2.3 Configuration file

The AMELIA function expects that a configuration file named "AMELIA.CONFIG" be available for reading in the directory in which the calling program is being run.[10] Note that the lexical form of the C-FITS style header has been adopted for use here; the sample file shown as Figure A-3 was used for testing and serves as an example of what entries are expected in the configuration file.

Note that improper selection of integration accuracy and iterative solution convergence criteria can introduce significant numerical errors in the computed model; experience has shown it prudent to plot out a model lightcurve intended to be used with a fitting program before trying the fit so that ''ringing'' or other numerical artifacts which could confuse the fitting process can be detected.

[10]This rather inelegant situation arises because no reasonable method could be devised by which SAFIT would take the configuration file name and pass it along to AMELIA.

Table A-I: Model parameters used by AMELIA

1. C_{STAR} - photometric count from unocculted star
2. C_{BGND} - photometric count from background sky
3. Θ_{STAR} - angular diameter of occulted star (not used in Release 1)
4. b - limb-darkening parameter for star (not used in Release 1)
5. ρ_{ref} - (km) reference radius in shadow for $\phi = \phi_{ref}$ (ϕ_{ref} is configured value PHI_REF)
6. Δr_{τ} - (km) half-width of opacity transition zone
7. κ_{34} - (cm^{-1}) opacity at r_{34} (item 17) if there were no opacity transition zone
8. f - (unitless) parameter referring to amount of aerosols being created at radius of interest vs. amount created above and merely falling through radius of interest. (see Equation (A.18), page 65)
9. Δr_{12} - (km) half-width of temperature transition region between Layer 1 and Layer 2
10. Δr_{23} - (km) half-width of temperature transition region between Layer 2 and Layer 3
11. Δr_{34} - (km) half-width of temperature transition region between Layer 3 and Layer 4
12. Γ_2 - ($K \cdot km^{-1}$) lapse rate of Layer 2
13. Γ_3 - ($K \cdot km^{-1}$) lapse rate of Layer 3
14. Γ_4 - ($K \cdot km^{-1}$) lapse rate of Layer 4
15. r_{12} - (km) radius of boundary between Layer 1 and Layer 2
16. r_{23} - (km) radius of boundary between Layer 2 and Layer 3
17. r_{34} - (km) radius of boundary between Layer 3 and Layer 4
18. T_{12} - (K) temperature of isothermal Layer 1

Figure A-3: Sample AMELIA configuration file

```
        Parameter configuration file used by AMELIA,
        in computing atmospheric model.

        THIS FILE FOR HAZY ISOTHERMAL MODEL
        Pluto atmosphere.

GRAVCONS = 6.672e-11     /* (m^3 kg^-1 sec^-2) gravitation constant */
BOLTZMAN = 1.38062e-23   /* (J K^-1) Boltzmann constant             */
M_AMU    = 1.66053e-27   /* (kg) mass of atomic mass unit           */
LOSCHMIT = 2.68684e+19   /* (molecules cm^-3) Loschmidt's number    */
        (LOSCHMIT from Allen, "Astrophysical Quantities", 3rd Ed.)

MOL_WT   = 16            /* (amu) mean mol. wt. of atmospheric gas  */
        (methane)
NU_STP   = 4.38e-04      /* (unitless) refractivity of gas at STP   */
        (from [ELLI89]: mean value for methane, integrated over
         spectral response of SNAPSHOT CCD chip)

M_PLANET = 1.29e+22      /* (kg) mass of planet                     */
D2PLANET = 4.323e+09     /* (km) distance from observer to planet   */
        (Pluto)

PHI_REF  = 0.5   /* (unitless) Normalized reference flux value
                    corresponding to reference radius in shadow    */
N_INT    = 1     /* number of bins for integration-time smoothing  */
N_SD     = 1     /* number of bins for star-diameter smoothing     */


R_BOTTOM = 900.0 /* (km) radius of lower bound for atmosphere      */
        (TBD for Pluto)

        The idea for specifying an R_BOTTOM for a given model has
        to do with finding the radius (at planet; NOT in shadow)
        below which computed values start to overflow HUGE (bending
        angle, optical depth, ...), or at which obviously weird
        things are happening (e.g. T<0).
        This is purely a numerical check, with no INHERENT physical
        meaning (e.g. increasing values for bending angle will be
        invalid from a physical standpoint long before they get to
        HUGE.)
        AMELIA checks the planet radius at which it is about to try
        to do a computation; if it's below R_BOTTOM it will print
        a warning message.
        R_BOTTOM should be at or slightly above the lowest radius
        for which whatever computations you're doing don't blow up.
        Obviously this
        value will depends on the particular model parameters; one
        empirical way to determine the value of R_BOTTOM is to use
        a test driver to call calc_all and compute values at
        decreasing planet radii until it dies.
```

Figure A-3, continued

```
SIGNIFDG = 3
        Specifies number of significant decimal digits for
        bending angle computations.  Used to determine past
        what point are approximations made in computing bending
        angles no longer valid, and print warning messages. NOTE
        THAT THIS VALUE DOES NOT AFFECT ANY COMPUTATIONS MADE.

        The following 9 values are configured parameters which are
        passed to the 'ode_integrate' function, which performs
        numerical integrations to compute bending angles & derivative,
        optical depth, and number density:
        Part 1.  For isothermal layer:
A_ISO = 0.005 /* fractional accuracy for ode_integrate */
T_ISO = 0.01   /* (radians) trialstep for ode_integrate */
M_ISO = 0.0    /* (radians) minstep for ode_integrate */
        Part 2.  For clear, nonisothermal layers:
A_CNON = 0.005 /* fractional accuracy for ode_integrate */
T_CNON = 0.01   /* (radians) trialstep for ode_integrate */
M_CNON = 0.0    /* (radians) minstep for ode_integrate */
        Part 3.  For hazy, nonisothermal layer:
A_HNON = 0.02  /* fractional accuracy for ode_integrate */
T_HNON = 0.01   /* (radians) trialstep for ode_integrate */
M_HNON = 0.0    /* (radians) minstep for ode_integrate */

        The following eight values deal with the 'calc_phi' function
        computing normalized flux values:
HBEGN_PN = 1.02 /* high bound, solving for rgnear */
ASOLV_PN = 0.1  /* (km) accuracy to use for rgnear */
GBRAC_PN = 1.2  /* 'grow' factor for rgnear search range */
NBRAC_PN = 50   /* number of times to try growing rgnear range */
LBEGN_PF = 0.98 /* low bound, solving for rgfar */
ASOLV_PF = 0.1  /* (km) accuracy to use for rgfar */
GBRAC_PF = 1.2  /* 'grow' factor for rgfar search range */
NBRAC_PF = 50   /* number of times to try growing rgfar range */

        The following nine values deal with the 'calc_n0' function
        computing the reference number density at r12:
HBEGN_NN = 1.02      /* high bound, solving for rgnear */
GBRAC_NN = 1.2       /* 'grow' factor for rgnear search range */
NBRAC_NN = 50        /* number of times to try growing rgnear range */
CSOLV_NN = 0.1       /* (km) convergence to use for rgnear */
CNVRG_N0 = 0.005     /* fractional convergence for n0 */
LBEGN_NF = 0.995     /* low bound, solving for rgfar */
GBRAC_NF = 1.05      /* 'grow' factor for rgfar search range */
NBRAC_NF = 50        /* number of times to try growing rgfar range */
CSOLV_NF = 0.1       /* (km) convergence to use for rgfar */
```

A.2.4 Global information

The identifiers `mpvals`, `mpold`, `mcvals`, and `global` denote globally-visible structures defined in module amelia.c and declared in all other AMELIA modules, which hold model parameter values (current and previous call), model configured values, and other miscellaneous global information, respectively:

1. Application code must not redefine these identifiers.
2. These identifiers are external and global, and are thus visible to *all* modules linked with AMELIA. Use of these values by application code is discouraged; in particular, if the application *modifies* any of these values, unpredictable results may occur.

A.2.5 AMELIA Messages

The lexical format of most warning and error messages produced by AMELIA is

function_name: text of message

where *function_name* is the name of the function producing the message.

A.2.5.1 Warning messages

These are for user information only; they do not affect computations made and do not raise any signal internally.

calc_t (warning): computed unphysical T=*temp* at r=*radius*
: Function `calc_t`, which computes atmosphere temperature at a given planet radius, is about to return and has noticed that it's returning a computed temperature <= 0° Kelvin at that radius.

calc_all (warning): Supplied graze radius *radius* below bottom *radius*
: `calc_all`, which computes bending angle, derivative thereof, optical depth, and number density, has been called with a planet radius argument which is below the current lower limit of computability. (See also configuration file, keyword "R_BOTTOM".)

calc_all (warning): Computed bending (rg=*radius*) *angle* > max *angle*
: `calc_all` is about to return and has noticed that the bending angle value (radians) it has computed for graze radius 'rg' violates the small-

angle approximation $\cos\theta \approx 1$ ("$s \approx x$", on page 70) made in its computation, to configured value "SIGNIFDG" significant figures.

A.2.5.2 Some error messages

Each corresponds to some error condition which raises an internal signal and indicates that the computation has failed at some level. Since one of the design ground rules of the AMELIA code is to perform extensive internal consistency and boundary condition checks, and to "make some noise" when anything questionable is detected, there are *many* such messages; this is only a partial listing and includes those most likely to be seen and easiest to fix:

get_config: Unable to open configuration file *filename*
: The named configuration file was not found or otherwise unopenable. Check first to see that you've got the expected file in the directory from which you're running the application program.

Duplicate configuration value found for *key*
: The configuration file contains more than 1 entry for the keyword. Remove the extra entry from your configuration file.

Lexical error in configuration value *key*
: A formatting error was detected when the value of *key* was interpreted as a numeric value. Fix the value in your configuration file with a text editor.

get_config: semantic error in *key*
: Normally indicates that a configured numeric value, which has been successfully parsed, was found to have an illegal or meaningless value (e.g. you've configured the mass of planet M_PLANET ≤ 0). Fix the offending value in the configuration file.

Missing configured value *key*
: No entry for the key was found in the configuration file. Add one!

finish_params: *condition* not allowed
: When inspecting the internal model parameters computed from the fitted parameter values by the user-written `unwrap_params` function, the invalid *condition* was detected (e.g., "star_count<=0"). Time to debug your `unwrap_params` module.

finish_params: isothermal boundary must be above opacity
: A special case of "*condition* not allowed"

finish_params: overlapping transition layers illegal

A special case of "*condition* not allowed". The current combination of layer boundaries and transition region thicknesses results in a overlap somewhere between transition layers. Probably an unwrapper problem, though it's possible you're just sending it a bad set of parameters.

rkqc: stepsize shrunk to 0

bsstep: stepsize shrunk to 0

If one of these two messages is seen[11], a numerical integration has failed, in the following way: the integrator tries to compute a result with some specified accuracy (for AMELIA, these accuracies are specified in the configuration file). It does this by taking each "integration step" of size small enough to match the accuracy required; if a given step size is too large it tries something smaller. If the step size decreases to zero then this error is signaled. Examine the configured minimum step sizes and accuracy in the configuration file. (This error also has been seen to occur where the derivative being integrated isn't sufficiently smooth; this problem was solved by breaking up the offending integrals into sums of integrals, putting the discontinuities at integral boundaries. Note that this involved significant recoding of AMELIA.)

rkqc: unscaled accuracy overflow for equation index *n*

bsstep: unscaled accuracy overflow for equation index *n*

Another potentially messy integration failure. In this case, the integrator is trying to compute "goodness of integral" for its current guess of step size, and in scaling its internal "error" value, overflowed HUGE. Same recommendations as for "stepsize shrunk to 0".

rkqc: scaled accuracy overflow for equation index *n*

bsstep: scaled accuracy overflow for equation index *n*

Closely related to "unscaled accuracy overflow". Integrator is about to scale the error of the "least accurately computed" of the equations it's integrating by dividing the error by the (configured) desired accuracy, and finds that the value overflows HUGE. Same recommendations as for "stepsize shrunk to 0".

proc1name: *proc2name* signals '*exception*'

Catch-all internal error message; normally indicates that the error is non-trivial, or error cause is non-unique. It indicates that a call to *proc2name* by *proc1name* has failed, with *proc2name* signalling

---

[11]which one you get, `rkqc` or `bsstep`, depends on which integration stepper routine is called by the "lightcurve.o" object file you're using

exception. Since it's not unlikely that something fairly complex is going on, the primary purpose of this message is to provide some straightforward hints as to where in the source code to start poking around with a debugger to look for the underlying problem. Note that these commonly would come in bunches, since when a low-level procedure call fails, it signals its caller, which in turn fails and signals its caller, and so forth, up until the top-level `amelia` function finally returns `FAIL`. A real-life example while modeling for Pluto:

```
iso_derivs: x outside allowed range
rk4: *derivs1 signals 'bounds'
rkqc: rk4-1 signals 'not_possible'
ode_integrate: stepper signals 'not_possible'
calc_normall: ode_integrate(iso) signals 'not_possible'
n0_solvand: calc_normall (wsn) signals 'not_possible'
calc_n0: bracket_root (n0) signals 'not_possible'
finish_params: calc_n0 signals 'not_possible'
amelia: finish_params signals 'not_possible'
```

In this case, the actual problem occurred while `calc_n0`, trying to iteratively find the physically meaningful root of function `n0_solvand`, called `bracket_root` to bracket the interval in which the root would be found; unfortunately, the particular parameter set for the atmosphere in use for that call caused there to be *no* such root.[12] Since `bracket_root` has no way to detect this problem, it worked very hard to do its job anyway, and diligently tried to return a bracketing interval around the nearest root it *could* find, which in this case happened to be farther out at about radius 3300km from Pluto, but which alas had no physical significance. The barrage of messages from `iso_derivs`, `ode_integrate`, and so forth, are the result of internal consistency and bounds checks violated as AMELIA attempted to perform atmospheric model computations to zero in on this bad root.

A.3 Model Derivations

The model parameter notation defined in Section A.2 will be used throughout the following derivations.

---

[12] `n0_solvand` implements Equation (A.57) and can be a nasty function! See footnote on page 80.

A.3.1 Atmosphere

A.3.1.1 Temperature

The atmosphere model developed includes provisions for four separate layers, each characterized by its own linear temperature gradient (the *lapse rate* $\Gamma = \frac{-dT(r)}{dr}$). The outermost layer is isothermal, so by assigning some particular temperature to this outer layer, the constraints imposed by assigning temperature gradients to the underlying three layers completely specify the temperature profile of the model.

In order that the model occultation lightcurve to be obtained using this atmosphere be continuous, the temperature profile for the atmosphere as a whole must be continuous as well. Furthermore, the *first derivative* of temperature with respect to radius must also be continuous.[13] In order to satisfy these requirements in the atmospheric model, an additional transition-zone layer is included at each of the three inter-layer boundaries where the temperature profile would otherwise be discontinuous in slope. Within such a transition layer a quadratic curve is used to represent the temperature profile and thus "smooth over" the break in slope. In general, *cubic* curves are the lowest-order curves that can be made to simultaneously meet the continuity of position and slope requirements to join two arbitrary segments (four conditions, position and tangent vector at each end of the curve) and also ensure that the ends of the curve pass through specified points on the profile. In the cases here though, the segments to be joined are not completely arbitrary: the points at which the extensions of the adjoining zones' linear temperature gradients intersect the center of the transition layer exactly coincide. This additional constraint zeroes out the cubic term, so the actual curve to join the segments is only quadratic in r.

---

[13]Subsequently it was realized that the *second* derivative would be needed as well (for Equation (A.61)); note that the transition layers as defined do *not* have continuity of second derivatives. It's not yet clear whether this will prove to be a problem.

An expression for the temperature curve $T(r)$ within within the transition zone joining the bottom of layer m with the top of layer n was derived by solving the four continuity equations as a linear system of four equations assuming values will be assigned for T_{mn}, Γ_m, Γ_n, and Δr_{mn}. The result and its first two derivatives are given by

$$T(r) = T_{mn} + \frac{\Gamma_n - \Gamma_m}{4}\Delta r_{mn} - \frac{\Gamma_n + \Gamma_m}{2}(r - r_{mn}) + \frac{\Gamma_n - \Gamma_m}{4\Delta r_{mn}}(r - r_{mn})^2 \qquad (A.1)$$

$$\frac{dT(r)}{dr} = \frac{-(\Gamma_n + \Gamma_m)}{2} + \frac{\Gamma_n - \Gamma_m}{2\Delta r_{mn}}(r - r_{mn}) \qquad (A.2)$$

$$\frac{d^2T(r)}{dr^2} = \frac{\Gamma_n - \Gamma_m}{2\Delta r_{mn}} \qquad (A.3)$$

Note that the transition layer half-widths Δr_{12}, Δr_{23}, and Δr_{34} are among the parameter values expected from the ''parameter unwrapper''.

A.3.1.2 Number density

Qualitatively, the density of the atmosphere of a planet decreases with increasing altitude, by virtue of its existing in the planet's gravitational field. In order to obtain an analytical form for the *number density* $n(r)$ as gas particles per unit volume, it is necessary to begin with some well-defined constraints. One such equation of state is the ideal gas law:

$$pV = n_{mol} R_0 T \qquad (A.4)$$

where n_{mol} is the number of moles of ideal gas and R_0 is the 'gas constant'. Use $R_0 = kN_A$ and $n = \frac{n_{mol} N_A}{V}$ to get an alternate form

$$p(r) = n(r)\, kT(r) \qquad (A.5)$$

where N_A is Avagadro's number and k is the 'Boltzmann constant'.

A second constraint is provided by an assumption of hydrostatic equilibrium. This seems a

reasonable approach, since deviations from equilibrium should be self-correcting over short time-scales:

$$\frac{dp(r)}{dr} = -\rho g(r) = -\bar{\mu} M_0 n(r) g(r) \qquad (A.6)$$

$\bar{\mu}$ represents the mean molecular weight of the ideal gas atmosphere of interest here, and is assumed to be constant over r. M_0 is the mass of the atomic mass unit.

Equate the first derivative with respect to r of Equation (A.5) with Equation (A.6), and rearrange towards getting an expression for $n(r)$:

$$\frac{dp(r)}{dr} = k\frac{d[n(r)T(r)]}{dr} = -\bar{\mu} M_0 n(r) g(r)$$

$$n(r)\frac{dT(r)}{dr} + T(r)\frac{dn(r)}{dr} = \frac{-\bar{\mu} M_0}{k} n(r) g(r)$$

$$\frac{1}{n(r)}\frac{dn(r)}{dr} = \frac{d}{dr}[\ln n(r)] = -\left[\frac{1}{T(r)}\frac{dT(r)}{dr} + \frac{\bar{\mu} M_0}{kT(r)} g(r)\right]$$

To get $n(r)$ from $\ln n(r)$, integrate from a reference radius r_1 to r:[14]

$$\int_{r_1}^{r} \frac{d}{dr}[\ln n(r)]\, dr = \int_{\ln n(r_1) = \ln n_1}^{\ln n(r)} d[\ln n(r)] = -\int_{r_1}^{r} \left[\frac{1}{T(r)}\frac{dT(r)}{dr} + \frac{\bar{\mu} M_0}{kT(r)} g(r)\right] dr$$

The usual approximation taking $g(r)$ to be constant throughout the thickness of the atmosphere is only valid if the atmosphere thickness is small compared to the planet's radius, and thus is not valid for modelling Pluto's atmosphere. Instead, replace the acceleration due to gravity by its explicit dependence on radius $g(r) = \frac{GM_P}{r^2}$ [KLEP73, p. 83] on the right-hand side, where M_P represents the mass of the planet, and rearrange a bit more to finally yield an expression for $n(r)$:

[14] A particular value for r_1 will be pinned down in Section A.3.1.4.

$$\ln\frac{n(r)}{n_1} = -\int_{r_1}^{r}\left[\frac{1}{T(r)}\frac{dT(r)}{dr}+\frac{\bar{\mu}M_0GM_P}{kT(r)}\cdot\frac{1}{r^2}\right]dr$$

$$n(r) = n_1\cdot\exp\left\{-\int_{r_1}^{r}\left[\frac{1}{T(r)}\frac{dT(r)}{dr}+\frac{\bar{\mu}M_0GM_P}{kT(r)}\cdot\frac{1}{r^2}\right]dr\right\} \qquad (A.7)$$

Note that this equation cannot specify n_1, the scale factor for the number density profile; it only defines the functional form of $n(r)$. Computing a value for n_1 is possible only by using a reference flux level of starlight refracted through the atmosphere, which can't be handled until after introducing the modelled lightcurve (Section A.3.2.4). For now, define a scaled *normalised number density* $\tilde{n}(r)$ such that $\tilde{n}(r_1) \equiv 1$:

$$\tilde{n}(r) \equiv \frac{n(r)}{n_1} \qquad (A.8)$$

a quantity which prove useful for a number of upcoming computations. (Honest!)

Using Equation (A.8), Equation (A.7) becomes

$$\tilde{n}(r) = \exp\left\{-\int_{r_1}^{r}\left[\frac{1}{T(r)}\frac{dT(r)}{dr}+\frac{\bar{\mu}M_0GM_P}{kT(r)}\cdot\frac{1}{r^2}\right]dr\right\} \qquad (A.9)$$

Equation (A.9) can be simplified somewhat if the atmospheric layer of interest is taken to be isothermal, with $\frac{dT(r)}{dr} = 0$ (as will be true for the uppermost layer of the AMELIA model):

$$\tilde{n}(r) = \exp\left\{-\int_{r_1}^{r}\frac{\bar{\mu}M_0GM_P}{kT(r)}\cdot\frac{1}{r^2}dr\right\}$$

$$\tilde{n}(r) = \exp\left\{\frac{-\bar{\mu}M_0GM_P}{kT}\cdot\left(\frac{1}{r_1}-\frac{1}{r}\right)\right\} \qquad (A.10)$$

A.3.1.3 Opacity

In order to compute the extinction coefficient corresponding to an atmospheric model incorporating a "hazy" layer along with the atmospheric gas, an expression for the *opacity* $\kappa(r)$ is required. If the opacity is taken to be due to aerosol particles of radius r_a cm and of number density $n_a(r)$ particles·cm^{-3}, and if there's no self-shadowing of these particles, the opacity $\kappa(r) = n_a(r)\pi r_a^2$. If the (nonzero) opacity at some reference radius r_2 is known, then

$$\frac{\kappa(r)}{\kappa(r_2)} = \frac{n_a(r)}{n_a(r_2)} \tag{A.11}$$

The task now is to determine an expression for the number density of aerosols in terms of the atmosphere's $n(r)$ and $T(r)$.

If the aerosol particles are forming either above or within the layer of interest according to a source function $\sigma(r)$ particles·cm^{-3}·sec^{-1}, and are falling toward the planet at their positive terminal velocity $v(r)$ cm·sec^{-1}, a continuity equation can be written:

$$\frac{d}{dr}[n_a(r)\cdot v(r)] = -\sigma(r)$$

Integrate this relation from r_2 (the reference radius) to r to get an equation representing the source flux in particles·cm^{-2}·sec^{-1}:

$$n_a(r)\,v(r) = n_a(r_2)\,v(r_2) - \int_{r_2}^{r} \sigma(r)\,dr \tag{A.12}$$

To proceed will require obtaining an expression for $\sigma(r)$.

If the mechanism by which these aerosol particles are formed is some interaction with the atmospheric gas itself, one can expect that in the haze layer, the source function is directly proportional to the gas number density:

$$\sigma(r) = \sigma(r_2) \cdot \frac{n(r)}{n(r_2)} \tag{A.13}$$

However, note that for use in Equation (A.12), $\lim_{r \to \infty} \sigma(r)$ must converge to zero at least linearly, otherwise the integral diverges and the flux of aerosols becomes infinite. Since $\lim_{r \to \infty} n(r)$ is a nonzero constant, Equation (A.13) is *not* valid above the haze layer.

Fortunately, the model definition specifies that the atmosphere above the opacity transition zone is clear, so define r_2 to be up at the top of the haze at r_{34} and use the following 'piecewise' functional form for $\sigma(r)$:

$$(r > r_2) \quad \sigma(r) = 0 \tag{A.14}$$

$$(r = r_2) \quad \sigma(r) = n_a(r) v(r) \delta(r_2) \tag{A.15}$$

$$(r < r_2) \quad \sigma(r) = \sigma(r_2) \cdot \frac{n(r)}{n(r_2)} \tag{A.16}$$

Note that the Dirac δ-function used in the $r=r_2$ case has units of $1/r$, so that $\int_0^\infty n_a(r) v(r) \delta(r_2) dr = n_a(r_2) v(r_2)$.

Having disposed of any aerosol considerations above r_2, now figure out what's going on underneath. For radii within the haze layer, substituting Equation (A.16) into Equation (A.12) yields the source flux

$$n_a(r)\, v(r) \;=\; n_a(r_2)\, v(r_2) - \int_{r_2}^{r} \sigma(r_2) \cdot \frac{n(r)}{n(r_2)} dr \tag{A.17}$$

Rearrange for a more convenient form:

$$\frac{n_a(r)}{n_a(r_2)} \;=\; \frac{v(r_2)}{v(r)} \left[1 - \frac{\sigma(r_2) H(r_2)}{n_a(r_2)\, v(r_2)} \int_{r_2}^{r} \frac{n(r)}{n(r_2)} \frac{1}{H(r_2)} dr \right]$$

To clean up the notation, define a new opacity parameter f, to refer to the ratio between the distributed source function (amount of aerosols the source function would make in one scale height at r_2), and the strength of the δ-function source at r_2:

$$f = \frac{\sigma(r_2) H(r_2)}{n_a(r_2) v(r_2)} \tag{A.18}$$

Substituting for f here yields

$$\frac{n_a(r)}{n_a(r_2)} = \frac{v(r_2)}{v(r)} \left[1 - f \int_{r_2}^{r} \frac{n(r)}{n(r_2)} \frac{1}{H(r_2)} dr \right] \tag{A.19}$$

Equation (A.19) has an additional dependence on terminal velocity; an expression for $v(r)$ is given in [TOON80, Equation (2)]:

$$v(r) = \text{constant} \cdot \frac{1}{r^2 n(r) [T(r)]^{1/2}}$$

Substituting this relation, along with Equation (A.11), into Equation (A.19) yields

$$\frac{\kappa(r)}{\kappa(r_2)} = \left(\frac{r}{r_2} \right)^2 \frac{n(r)}{n(r_2)} \left(\frac{T(r)}{T(r_2)} \right)^{1/2} \left[1 - f \int_{r_2}^{r} \frac{n(r)}{n(r_2)} \frac{1}{H(r_2)} dr \right] \tag{A.20}$$

It's now necessary to substitute for $n(r)$. Recall that Equation (A.7), the expression for $n(r)$ in a nonisothermal layer, contains a dependence on n_1 whose value can't yet be specified. This dependency on n_1 thus inherited by Equation (A.20) through $n(r)$ can be removed completely by using Equation (A.8), by virtue of the fact that the $n(r)$ in Equation (A.20) only appear as ratios:

$$\frac{\kappa(r)}{\kappa_2} = \left(\frac{r}{r_2} \right)^2 \frac{\tilde{n}(r)}{\tilde{n}(r_2)} \left(\frac{T(r)}{T(r_2)} \right)^{1/2} \left[1 - f \int_{r_2}^{r} \frac{\tilde{n}(r)}{\tilde{n}(r_2)} \frac{1}{H(r_2)} dr \right] \tag{A.21}$$

$$\text{with} \quad H(r_2) = \frac{k T(r_2) r_2^2}{\bar{\mu} M_0 M_P G}$$

A.3.1.4 Summary

Temperature summary: (refer to Figure A-4)

Region 1 (isothermal): $(r_{12}+\Delta r_{12}) \le r < \infty$

Layer 1 is the outermost layer, and is isothermal, so $\Gamma_1 = 0$.

$$T(r) = T_{12} \tag{A.22}$$

Region 12 (transition between 1 and 2): $(r_{12}-\Delta r_{12}) < r < (r_{12}+\Delta r_{12})$

$$T(r) = T_{12}+\frac{\Gamma_2-\Gamma_1}{4}\Delta r_{12}-\frac{\Gamma_2+\Gamma_1}{2}(r-r_{12})+\frac{\Gamma_2-\Gamma_1}{4\Delta r_{12}}(r-r_{12})^2 \tag{A.23}$$

with T_{12} = a fixed assigned value

Region 2: $(r_{23}+\Delta r_{23}) \le r \le (r_{12}-\Delta r_{12})$

$$T(r) = T_{12}-(r-r_{12})\Gamma_2 \tag{A.24}$$

Region 23 (transition between 2 and 3): $(r_{23}-\Delta r_{23}) < r < (r_{23}+\Delta r_{23})$

$$T(r) = T_{23}+\frac{\Gamma_3-\Gamma_2}{4}\Delta r_{23}-\frac{\Gamma_3+\Gamma_2}{2}(r-r_{23})+\frac{\Gamma_3-\Gamma_2}{4\Delta r_{23}}(r-r_{23})^2 \tag{A.25}$$

with $T_{23} = T_{12}-\Gamma_2(r_{23}-r_{12})$

Region 3: $(r_{34}+\Delta r_{34}) \le r \le (r_{23}-\Delta r_{23})$

$$T(r) = T_{23}-(r-r_{23})\Gamma_3 \tag{A.26}$$

Region 34 (transition between 3 and 4): $(r_{34}-\Delta r_{34}) < r < (r_{34}+\Delta r_{34})$

$$T(r) = T_{34}+\frac{\Gamma_4-\Gamma_3}{4}\Delta r_{34}-\frac{\Gamma_4+\Gamma_3}{2}(r-r_{34})+\frac{\Gamma_4-\Gamma_3}{4\Delta r_{34}}(r-r_{34})^2 \tag{A.27}$$

with $T_{34} = T_{23}-\Gamma_3(r_{34}-r_{23})$

Region 4: $r_{surface} \le r \le (r_{34}-\Delta r_{34})$

$$T(r) = T_{34}-(r-r_{34})\Gamma_4 \qquad (A.28)$$

also, $T_{surface} = T_{34}-\Gamma_4(r_{surface}-r_{34})$

Note here that the temperatures represented by T_{mn} in the above relations *do not* in general correspond to the model atmospheric temperatures $T(r_{mn})$. Since T_{mn} is the temperature at r_{mn} in the absence of a transition layer, the two values correspond only in the case that the transition layer if of zero thickness.

Number density summary: The reference radius corresponding to the reference number density n_1 first introduced in deriving Equation (A.7) can be chosen to be anywhere in the isothermal layer. For convenience make it the *bottom* of the isothermal layer:

$$r_1 = r_{12}+\Delta r_{12} \qquad (A.29)$$

so Equation (A.8) is now pinned down as

$$\tilde{n}(r) \equiv \frac{n(r)}{n(r_{12}+\Delta r_{12})} \qquad (A.30)$$

For future computational convenience, identify the integral contained in Equation (A.9) as an auxiliary function U

$$U(r) = \int_{r_{12}+\Delta r_{12}}^{r} \left[\frac{1}{T(r)} \frac{dT(r)}{dr} + \frac{\overline{\mu} M_0 G M_P}{kT(r)} \cdot \frac{1}{r^2} \right] dr \qquad (A.31)$$

producing

$$\tilde{n}(r) = \exp\left\{ -U(r) \right\} \qquad (A.32)$$

Equation (A.10) for the isothermal layer is unchanged. Finally, note that once a value for the reference number density n_1 has been computed, the actual number density $n(r)$ can be easily obtained from $\tilde{n}(r)$ by

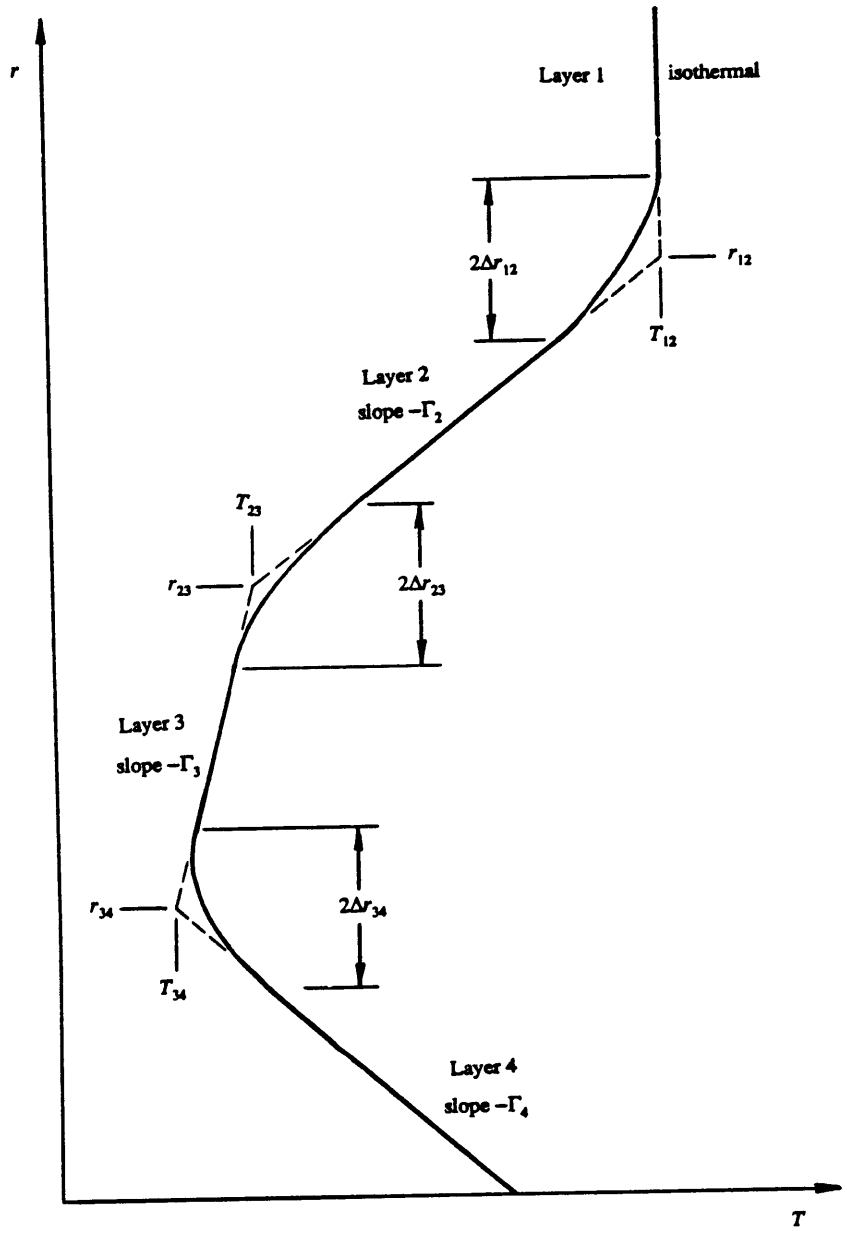

Figure A-4: Model Atmosphere Temperature Profile

$$n(r) = n_1 \cdot \tilde{n}(r) \tag{A.33}$$

Opacity summary: (refer to Figure A-5) In the AMELIA model, the haze layer corresponds to atmosphere Layer 4, the lowest layer, so reference radius r_2 in Equation (A.21) corresponds to atmosphere radius r_{34}. A linear transition region centered on r_{34} is also included to satisfy continuity of opacity (but not its first derivative) between $\kappa=0$ in atmosphere Layer 3 and computed opacity $\kappa(r)$ at the top of Layer 4; this transition layer is of thickness $2\Delta r_\tau$. In summary,

Regions 1,2,3 (clear atmosphere): $(r_{34}+\Delta r_\tau) \le r < \infty$

$$\kappa(r) = 0 \tag{A.34}$$

Opacity transition region (straddles r_{34}): $(r_{34}-\Delta r_\tau) < r < (r_{34}+\Delta r_\tau)$

$$\kappa(r) = \frac{r_{34}+\Delta r_\tau - r}{2\Delta r_\tau} \cdot \kappa(r_{34}-\Delta r_\tau) \tag{A.35}$$

Region 4 (haze layer): $r \le (r_{34}-\Delta r_\tau)$

$$\kappa(r) = \kappa_{34} \cdot \left(\frac{r}{r_{34}}\right)^2 \frac{\tilde{n}(r)}{\tilde{n}(r_{34})} \left(\frac{T(r)}{T(r_{34})}\right)^{1/2} \left[1 - f\int_{r_{34}}^{r} \frac{\tilde{n}(r)}{\tilde{n}(r_{34})} \frac{1}{H(r_{34})} dr\right] \tag{A.36}$$

$$\text{with scale height } H(r_{34}) = \frac{kT(r_{34}){r_{34}}^2}{\mu M_0 M_P G}$$

For future computational convenience, identify the integral contained in Equation (A.36) as an auxiliary function Y

$$Y(r) = \int_{r_{34}}^{r} \frac{\tilde{n}(r)}{\tilde{n}(r_{34})} \frac{1}{H(r_{34})} dr \tag{A.37}$$

finally producing

$$\kappa(r) = \kappa_{34} \cdot \left(\frac{r}{r_{34}}\right)^2 \frac{\bar{n}(r)}{\bar{n}(r_{34})} \left(\frac{T(r)}{T(r_{34})}\right)^{1/2} \left[1 - f \cdot Y(r)\right] \qquad (A.38)$$

A.3.2 Refraction and Extinction

A.3.2.1 Refractive bending angle

The geometry of a light ray from a star at infinity being refracted by a planetary atmosphere is shown in Figure A-6.

Note that the closest approach (''graze'') distance of the refracted light ray is denoted by the symbol r_G; this will always be distance as measured at the planet.

The derivation in [GOLD63, equation 9] yields the following expression for the refractive bending angle θ ($\theta < 0$):

$$\theta = \int_{-\infty}^{\infty} \hat{\delta} \cdot \nabla \ln N(r)\, ds$$

where

$\hat{\delta}$ is a unit vector in the direction perpendicular to the ray,
towards the direction of deflection.
$N(r)$ is the refractive index of the atmosphere, here a function of r only
s is the arc length along the ray

Since $N(r)$ is a function of r only, the gradient $\nabla \ln N(r)$ evaluates to $\hat{r} \frac{1}{N(r)} \frac{\partial N(r)}{\partial r}$

[HILD76, p. 313], yielding

$$\theta = \int_{-\infty}^{\infty} \hat{\delta} \cdot \hat{r} \frac{1}{N(r)} \frac{dN(r)}{dr} ds$$

Make the following substitutions:

- Approximate $s \approx x$, so $ds \approx dx$.
- Approximate the path of the ray as being a straight line, so $\hat{\delta} \cdot \hat{r} = \frac{r_G}{r}$.

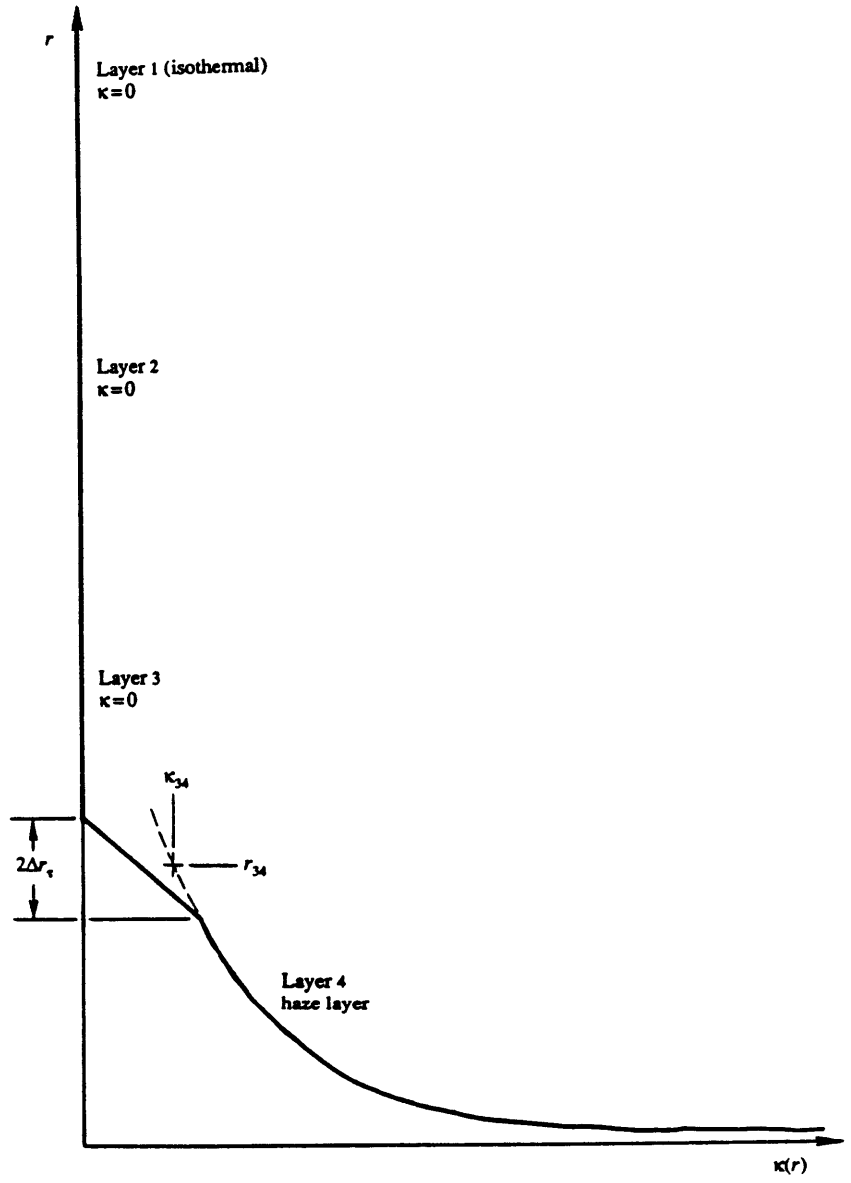

Figure A-5: Model Atmosphere Opacity Profile

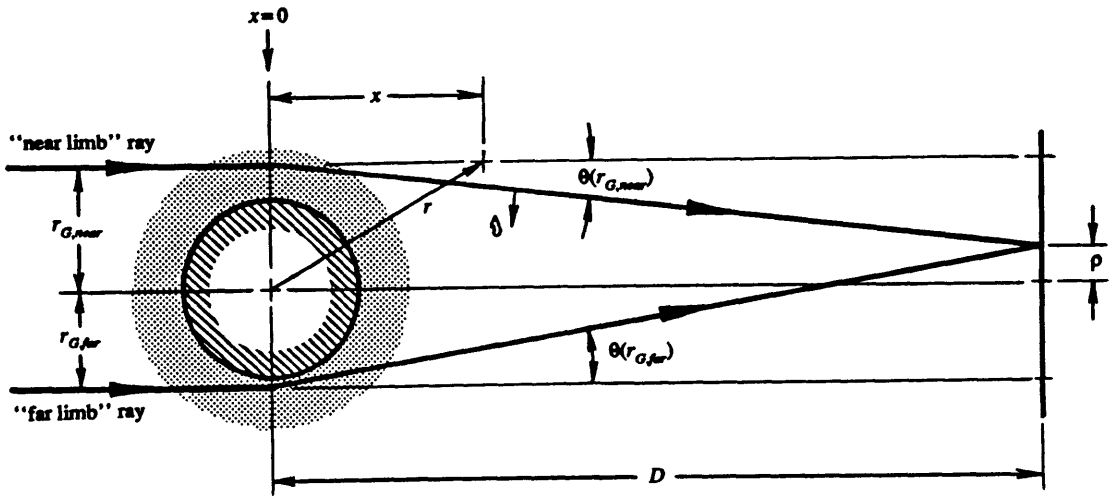

Figure A-6: Geometry for refraction of ray

- $N(r) = 1 + \nu(r)$, where $\nu(r)$ is the *refractivity* and is proportional to number density, so $\frac{dN(r)}{dr} = \frac{d\nu(r)}{dr}$.

Since $N(r) \approx 1$, the result is

$$\theta(r_G) \approx \int_{-\infty}^{\infty} \frac{r_G}{r} \frac{d\nu(r)}{dr} dx$$

Now, use

$$\nu(r) = \frac{n(r)}{N_0} \nu_{STP}$$

where

$n(r)$ = number density of gas
N_0 = Loschmidt's number
ν_{STP} = refractivity of gas at standard temperature and pressure

to get

$$\theta(r_G) \approx \frac{\nu_{STP}}{N_0}\int_{-\infty}^{\infty}\frac{r_G}{r}\frac{dn(r)}{dr}dx = \frac{2\nu_{STP}}{N_0}\int_{0}^{\infty}\frac{r_G}{r}\frac{dn(r)}{dr}dx \qquad (A.39)$$

As written this integral extends to infinity, which leads to some difficulty when attempting to compute its value numerically. Fortunately, it is possible to change variables and reduce it to a finite integral since that part of the atmospheric model extending to infinity is isothermal. Define r_{iso} to be the radius of the lower boundary of the model's outermost isothermal layer. (This is the *actual* boundary; *above* any transition zone to the underlying layer.) Denote the number density at this radius $n_{iso} = n(r_{iso})$. Figure A-6 shows that r, r_G, and x are related by

$$r^2 = r_G^2 + x^2 \qquad (A.40)$$

so use Equation (A.40) to introduce x_{iso} such that $r_{iso}^2 = r_G^2 + x_{iso}^2$. Write the expression for the bending angle θ as a sum of contributions from the non-isothermal atmosphere and the outermost isothermal layer:

$$\theta(r_G) = \theta_{non}(r_G) + \theta_{iso}(r_G) \qquad (A.41)$$

$$\theta_{non}(r_G) = \frac{2\nu_{STP}}{N_0}\int_{0}^{x_{iso}}\frac{r_G}{r}\frac{dn(r)}{dr}dx \qquad (A.42)$$

$$\theta_{iso}(r_G) = \frac{2\nu_{STP}}{N_0}\int_{x_{iso}}^{\infty}\frac{r_G}{r}\frac{dn(r)}{dr}dx \qquad (A.43)$$

Note that if the graze radius r_G is greater than the lower isothermal limit r_{iso}, the ray doesn't travel deep enough into the atmosphere to encounter the non-isothermal layers, and thus $\theta_{non} = 0$ and $x_{iso} = 0$.

These expressions for θ_{non} and θ_{iso} have a dependence on n_{iso} which would prove inconvenient, since it will be necessary to compute some form of bending angles as part of *determining* n_{iso} (i.e., before it's known). Since r_{iso} is defined to be $r_{12} + \Delta r_{12}$, it corresponds

exactly to r_1 (Equation (A.29)), meaning $n_{iso} = n_1$. To take advantage of this, define a scaled *normalised bending angle* $\tilde{\theta}(r_G)$:

$$\tilde{\theta}(r_G) \equiv \frac{\theta(r_G)}{n_1} \tag{A.44}$$

From Equation (A.30), $\frac{dn(r)}{dr} = n_{iso} \cdot \frac{d\tilde{n}(r)}{dr}$; substituting into Equations (A.41) through (A.43) yields

$$\tilde{\theta}(r_G) = \tilde{\theta}_{non}(r_G) + \tilde{\theta}_{iso}(r_G) \tag{A.45}$$

$$\tilde{\theta}_{non}(r_G) = \frac{2\nu_{STP}}{N_0}\int_0^{x_{iso}} \frac{r_G}{r}\frac{d\tilde{n}(r)}{dr}dx \tag{A.46}$$

$$\tilde{\theta}_{iso}(r_G) = \frac{2\nu_{STP}}{N_0}\int_{x_{iso}}^{\infty} \frac{r_G}{r}\frac{d\tilde{n}(r)}{dr}dx \tag{A.47}$$

The task now is to get rid of the upper limit of infinity in equation (A.47). Substitute for $\tilde{n}(r)$ using Equation (A.10):

$$\tilde{\theta}_{iso}(r_G) = \frac{2\nu_{STP}}{N_0}\int_{x_{iso}}^{\infty} \frac{r_G}{r}\frac{d}{dr}\left[\exp\left\{ \frac{-\bar{\mu}M_0GM_P}{kT}\left(\frac{1}{r_{iso}} - \frac{1}{r}\right)\right\}\right] dx$$

Define

$$C_1 \equiv \frac{\bar{\mu}M_0GM_P}{kT} \tag{A.48}$$

and note that $C_1 = \frac{r^2}{H(r)}$ with $H(r) = \frac{kT}{\bar{\mu}M_0 g(r)}$:

$$\tilde{\theta}_{iso}(r_G) = \frac{2\nu_{STP}}{N_0}\int_{x_{iso}}^{\infty}\frac{r_G}{r}\exp\left\{\frac{-C_1}{r_{iso}}+\frac{C_1}{r}\right\}\frac{-C_1}{r^2}dx$$

$$\tilde{\theta}_{iso}(r_G) = \frac{-2\nu_{STP}C_1 r_G}{N_0}\int_{x_{iso}}^{\infty}\frac{1}{r^3}\exp\left\{\frac{-C_1}{r_{iso}}\right\}\exp\left\{\frac{C_1}{r}\right\}dx$$

$$\tilde{\theta}_{iso}(r_G) = \frac{-2\nu_{STP}r_G}{N_0 C_1}\exp\left\{\frac{-C_1}{r_{iso}}\right\}\int_{x_{iso}}^{\infty}\left(\frac{C_1}{r}\right)^3\exp\left\{\frac{C_1}{r}\right\}\frac{dx}{C_1}$$

Introduce a new variable of integration ψ as shown in Figure A-7.

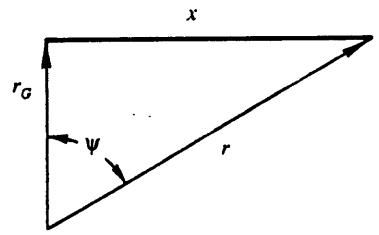

Figure A-7: Change of variables from x to ψ

It will be necessary to substitute for $\frac{C_1}{r}$ and $\frac{dx}{C_1}$, so express these in terms of ψ:

$$r = \frac{r_G}{\cos\psi} \;;\quad \frac{1}{r} = \frac{\cos\psi}{r_G} \;;\quad \text{so}\quad \frac{C_1}{r} = \frac{C_1\cos\psi}{r_G}$$

$$x = r_G\tan\psi \;;\quad \frac{x}{C_1} = \frac{r_G\tan\psi}{C_1} \;;\quad \text{so}\quad \frac{dx}{C_1} = \frac{r_G d\psi}{C_1\cos^2\psi}$$

Making these substitutions in the expression for $\tilde{\theta}_{iso}$ yields

$$\tilde{\theta}_{iso}(r_G) = \frac{-2\nu_{STP}r_G}{N_0 C_1}\exp\left\{\frac{-C_1}{r_{iso}}\right\}\int_{\psi_i}^{\pi/2}\left(\frac{C_1}{r_G}\right)^3\cos^3\psi\exp\left\{\frac{C_1\cos\psi}{r_G}\right\}\frac{r_G}{C_1}\frac{d\psi}{\cos^2\psi}$$

$$\tilde{\theta}_{iso}(r_G) = \frac{-2\nu_{STP}}{N_0}\exp\left\{\frac{-C_1}{r_{iso}}\right\}\int_{\psi_i}^{\pi/2}\frac{C_1\cos\psi}{r_G}\exp\left\{\frac{C_1\cos\psi}{r_G}\right\}d\psi \qquad (A.49)$$

This new integral is well-behaved over the new interval of integration 0 to $\pi/2$.

In summary, from Equations (A.45) through (A.47),

$$\tilde{\theta}(r_G) = \frac{2\nu_{STP}}{N_0}\int_0^{x_{iso}} \frac{r_G}{r}\frac{d\tilde{n}(r)}{dr}dx - \frac{-2\nu_{STP}}{N_0}\exp\left\{\frac{-r_{iso}}{H(r_{iso})}\right\}\int_{\psi_i}^{\pi/2}\frac{r_G}{H(r_G)}\cos\psi\exp\left\{\frac{r_G\cos\psi}{H(r_G)}\right\}d\psi \quad (A.50)$$

Finally, note that once a value for the reference number density n_1 has been computed, the actual bending angle $\theta(r_G)$ can be easily obtained from $\tilde{\theta}(r_G)$ by

$$\theta(r_G) = n_1 \cdot \tilde{\theta}(r_G) \quad (A.51)$$

A.3.2.2 Optical depth

To obtain an expression for the optical depth $\tau(r_G)$ seen by a light ray of closest approach (''graze'') distance r_G, integrate $\kappa(r)$ along the path taken by the ray. In Figure A-8, the positive x direction is defined to be the direction in which the light ray is traveling, and $x = 0$ corresponds to the coordinate of the planet center on the x axis. The ray path may be approximated to be a straight path since the bending of the ray by refraction won't affect τ enough to be significant here. The integral is

$$\tau(r_G) = \int_{-\infty}^{\infty}\kappa(r)dx$$

To express κ as a function of x, use Equation (A.40) to rewrite the integral in terms of r_G and x:

$$\tau(r_G) = \int_{-\infty}^{\infty}\kappa([r_G^2+x^2]^{1/2})dx = 2\int_0^{\infty}\kappa([r_G^2+x^2]^{1/2})dx \quad (A.52)$$

Though Equation (A.52) has an upper limit of integration of infinity, in this model the opacity κ is zero above radius $r_{34}+\Delta r_\tau$ (page 64). Thus the integral in fact need only be taken out to the x that corresponds to that radius:

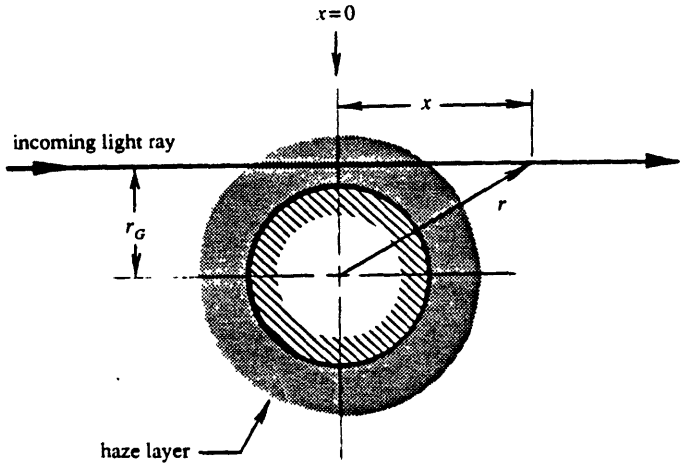

Figure A-8: Geometry of optical depth

$$x_\tau = ((r_{34}+\Delta r_\tau)^2 - r_G^2)^{1/2}$$

$$\tau(r_G) = 2\int_0^{x_\tau} \kappa([r_G^2+x^2]^{1/2})\,dx \qquad (A.53)$$

A.3.2.3 Model normalised flux from star

It now becomes necessary to explicitly deal separately with the rays refracted from the near limb and the far limb to shadow radius ρ. From the event geometry (Figure A-6), corresponding to that in [HUBB88, equations (5) and (6)], with bending angle α there corresponding to –θ here, see that

$$-D\theta(r_{G,near}) = r_{G,near} - \rho \qquad (A.54)$$

$$-D\theta(r_{G,far}) = r_{G,far} + \rho \qquad (A.55)$$

where

D is observer-to-planet distance
$r_{G,near}$ is graze radius of near limb ray
$r_{G,far}$ is graze radius of far limb ray
ρ is observer radius from shadow center

[HUBB88, equations (7) and (8)] yield the normalised flux ϕ taking into account ray convergence and the far limb contribution; to this include the effect of the opacity due to aerosols to get

$$\phi(\rho) = \left(\frac{e^{-\tau_{near}}}{1+D\left[\frac{d\theta(r_G)}{dr_G}\right]_{r_{G,near}}} \right) \frac{r_{G,near}}{\rho} + \left(\frac{e^{-\tau_{far}}}{1+D\left[\frac{d\theta(r_G)}{dr_G}\right]_{r_{G,far}}} \right) \frac{r_{G,far}}{\rho} \qquad (A.56)$$

τ_{near} and τ_{far} are the optical depths traversed by the near and far limb rays, respectively, computable using Equation (A.53).

A.3.2.4 Number density scale factor

As was pointed out in the derivation for Equation (A.7), the introduction of the light curve model makes it possible to determine a value for the number density scale factor n_1 used in Equations (A.30) and (A.51). Begin by defining a reference radius in the shadow ρ_{ref}, such that the near limb opacity is unimportant ($\tau_{near} \approx 0$) with respect to its corresponding reference flux level $\phi_{ref} \equiv \phi(\rho_{ref})$. Equation (A.56) can be used to write an expression for ϕ_{ref}; rearranging terms in that equation, setting $\tau_{near} \approx 0$, and substituting for D from Equations (A.54) and (A.55) yields

$$\phi(\rho_{ref}) - \frac{\left(\frac{r_{G,far}}{\rho_{ref}}\right) e^{-\tau_{far}}}{1-(r_{G,far}+\rho_{ref})\left(\frac{1}{\tilde{\theta}(r_{G,far})}\right)\left[\frac{d\tilde{\theta}(r_G)}{dr_G}\right]_{r_{G,far}}} \qquad (A.57)$$

$$= \frac{\left(\frac{r_{G,near}}{\rho_{ref}}\right) e^{-0}}{1-(r_{G,near}-\rho_{ref})\left(\frac{1}{\tilde{\theta}(r_{G,near})}\right)\left[\frac{d\tilde{\theta}(r_G)}{dr_G}\right]_{r_{G,near}}}$$

The iterative process of Figure A-9 may now be used to compute a value for n_1.

A.3.2.5 Summary and Implementation Strategy

The approach taken to implement the computation of bending angles and optical depth was suggested by Jack Wisdom and involves integrating the *derivatives* of $\tilde{\theta}$, $\frac{d\tilde{\theta}}{dr_G}$, and τ, along with those of the two auxiliary functions U and Y, simultaneously as a set of ordinary differential equations. The "Fourth-Order Runge-Kutta Stepper with Adaptive Stepsize" given in [PRES86] was adapted for use by AMELIA to handle the actual computations.[16] Integration of each the model's derivatives takes place using the auxiliary variable of integration ψ as defined in Figure A-7; derivatives of quantities which were derived in terms of r or x have been rewritten with a change of variable to ψ using

$$\frac{dr}{d\psi} = \frac{r_G \sin\psi}{\cos^2\psi} \qquad \frac{dx}{d\psi} = \frac{r_G}{\cos^2\psi}$$

The limits of integration in the implementation are $-\pi/2$ to 0, rather than from 0 to $+\pi/2$ as

[16]The Bulirsch-Stoer stepper from [PRES86] was implemented as well, and can be substituted for use in AMELIA at compile-time.

Figure A-9: Algorithm for iterative solution for n_1

```
BEGIN { compute n1 }

   set far limb contribution (to θ) in (A.57) = 0

   REPEAT

      compute r_G,near from (A.57) iteratively[15] using
      reference flux level and shadow radius

      compute θ(r_G,near) from geometry (A.54)

      n1 := θ(r_G,near) / θ~(r_G,near)  (Equation (A.44))

      IF value for n1 has converged

         THEN BREAK

         ELSE BEGIN

            compute r_G,far from geometry (A.55);
            this will be used in (A.57) in next iteration
            to compute the far limb contribution to flux

         END;

   UNTIL FALSE;

END;
```

is done in the formal derivation for $\tilde{\theta}_{iso}$. This change was made because the initial values for the integrals are known at ψ=-π/2 (e.g., $\tilde{\theta}_{iso} = 0$), whereas they're not known if integration begins at ψ=0, and manifests itself internally in computations of the bounds of integration, in range checks, and also in the sign of the odd $\frac{dY}{d\psi}$ auxiliary function used in computing optical depths in the haze layer.

[15]Experience shows this can be a particularly ill-behaved function to solve iteratively, depending on the particular set of values of model parameters being used. In its 'nice' form, when the atmosphere is isothermal, the function has one discontinuity to infinity and only two roots, and AMELIA appears to have no trouble distinguishing and grabbing the desired root. However, when a set of increasingly nonisothermal Pluto models were run through AMELIA, the actual *morphology* of the function changed as it 'grew' local minima and maxima and closely-spaced unwanted roots (they appear in pairs). Watch out!

Following are the derivatives implemented in "atmosphere.c":

Bending angle and derivative in isothermal layer: (C_1 defined in Equation (A.48)) These derivatives must be well-defined over the full interval $-\pi/2 \le \psi \le 0$. From Equation (A.49):

$$\frac{d\tilde{\theta}_{iso}}{d\psi} = \frac{-2\nu_{STP}}{N_0} \cdot \frac{C_1 \cos\psi}{r_G} \cdot \exp\left\{ \frac{C_1 \cos\psi}{r_G} - \frac{C_1}{r_{12}+\Delta r_{12}} \right\} \qquad (A.58)$$

and the derivative of *that*, with respect to r_G:

$$\frac{d\tilde{\theta}_{iso}}{dr_G d\psi} = \frac{-2\nu_{STP}}{N_0} \cdot \exp\left\{ \frac{C_1 \cos\psi}{r_G} - \frac{C_1}{r_{12}+\Delta r_{12}} \right\} \cdot \frac{-C_1 \cos\psi}{r_G^2} \left[\frac{C_1 \cos\psi}{r_G} + 1 \right] \qquad (A.59)$$

Bending angle and derivative in nonisothermal layers: These derivatives must be well-defined over the interval $-\pi/2 < \psi \le 0$. (Won't need to evaluate them at $\psi = -\pi/2$ because the nonisothermal layers of the model have an upper bound of $r < \infty$.) From Equations (A.46) and (A.32):

$$\frac{d\tilde{\theta}_{non}}{d\psi} = \frac{-2\nu_{STP}}{N_0} \cdot \frac{r_G}{\cos\psi} \left[\frac{1}{T(r)} \frac{dT}{dr} + \frac{\bar{\mu} M_0 G M_P}{kT(r) r^2} \right] \cdot \exp\left\{ -U(\psi) \right\} \qquad (A.60)$$

and the derivative of *that*, with respect to r_G:

(A.61)

$$\frac{d\tilde{\Theta}_{non}}{dr_G d\psi} = \frac{-2\nu_{STP}}{N_0} \cdot \exp\left\{-U(\psi)\right\}\left(\frac{1}{\cos\psi}\left[\frac{1}{T(r)}\frac{dT}{dr} + \frac{\bar{\mu}M_0 GM_P}{kT(r)r^2}\right] + \right.$$

$$\frac{r_G}{\cos\psi} \cdot \frac{\bar{\mu}M_0 GM_P \cos\psi}{kT(r)^2 r_G^2}\left[\frac{d^2T}{dr^2} - \frac{2\cos\psi}{r_G}\frac{dT}{dr} - \frac{2}{T(r)}\left(\frac{dT}{dr}\right)^2\right] +$$

$$\left. r_G\left[\frac{1}{T(r)}\frac{dT}{dr} + \frac{\bar{\mu}M_0 GM_P}{kT(r)r^2}\right]^2\right)$$

From Equation (A.31):

$$\frac{dU}{d\psi} = \left[\frac{1}{T(r)}\frac{dT}{dr} + \frac{\bar{\mu}M_0 GM_P}{kT(r)r^2}\right] \cdot \frac{r_G \sin\psi}{\cos^2\psi} \qquad (A.62)$$

Optical depth in haze layer (below opacity transition region): These derivatives must be well-defined over the interval $-\pi/2 < \psi \leq 0$. Note that the Y auxiliary function is for use in computing the opacity $\kappa(r)$ in the haze layer (Equation (A.38)). From Equation (A.37):

$$\frac{dY}{d\psi} = \frac{\tilde{n}(r)}{\tilde{n}(r_{34})H(r_{34})} \cdot \frac{r_G \sin\psi}{\cos^2\psi} \qquad (A.63)$$

From Equation (A.53):

$$\frac{d\tau}{d\psi} = \frac{2r_G}{\cos^2\psi} \cdot \kappa(r) \qquad (A.64)$$

In order to use these derived relations to compute a normalised flux value, the algorithm of Figure A-10 is implemented as function `calc_phi` (module "lightcurve.c").

Figure A-10: Algorithm for computation of normalised flux

```
BEGIN { compute φ(ρ) }
   { note that n(r12+Δr12), κ(r34−Δrτ), and ñ(r34) have already
     been (re)computed as needed during first call to amelia
     function with this parameter set, and were saved in
     global structure "mpvals" for use here }
   compute r_G,near iteratively from (A.54)
   compute [dθ(r_G)/dr_G]_r_G,near  by integrating (A.59) and (A.61)
   compute τ_near by integrating (A.64)

   compute r_G,far iteratively from (A.55)
   compute [dθ(r_G)/dr_G]_r_G,far  by integrating (A.59) and (A.61)
   compute τ_far by integrating (A.64)

   compute φ(ρ) from (A.56) and return the result
END
```

A.3.3 Model "observed flux"

A.3.3.1 Integration time smoothing

Though Equation (A.56) will compute for a given radius in the shadow ρ an *instantaneous* flux value for the model, a set of observed flux values with which one may wish to compare computed model values will in fact be *integrated* values, equally spaced in *time* rather than in radius. Therefore, it may be necessary to "integrate" the model's normalized flux values as well, over the shadow radii traversed by the observer over the time of the integration; AMELIA computes such 'integration-time smoothing" in the following way:

Start with the set of discrete times $t[k]$ representing the mid-times of the integrations of the observed data values, and the associated set of discrete radii in the shadow $\rho[k]$; the idea is to compute the corresponding set of integrated flux values $\phi_{int}[k]$.

To approximate a single integrated value $\phi_{int}[n]$ corresponding to a single mid-time $t[n]$ (at in-shadow radius $\rho[n]$), average the model flux over the radius range $\rho_-[n] \equiv \frac{\rho[n-1]+\rho[n]}{2}$ to $\rho_+[n] \equiv \frac{\rho[n]+\rho[n+1]}{2}$. In order to compute an approximate average normalized value of the continuous but not analytically expressible function $\phi(\rho)$ over the radius interval $\rho_-[n]$ to $\rho_+[n]$, compute N_{int} equally-spaced discrete values of ϕ within that radius interval and average those values:

$$\phi_{int}[n] \approx \frac{1}{N_{int}} \sum_{j=1}^{N_{int}} \phi(\rho_j[n]) \qquad (A.65)$$

$$\text{with} \quad \rho_j[n] = \rho[n] + \left(\frac{\rho_+[n]-\rho_-[n]}{N_{int}}\right)\left(j - \frac{N_{int}+1}{2}\right)$$

A.3.3.2 Scaling to observed flux

In order to produce a final model flux value $\Phi[n]$ to be compared with the actual observed flux from data, the integrated normalized flux value $\phi_{int}[n]$ needs to be scaled to the observed photometric counts from the star, and a contribution from sky background photometric count needs to be added:

$$\Phi[n] = \phi_{int}[n] \cdot C_{STAR} + C_{BGND} \qquad (A.66)$$

where

C_{STAR} = photometric count from unocculted star
C_{BGND} = photometric count from background sky

Note that this scaling assumes there's no gradient in the background counts.

Appendix B

OSBERT program

osbert

Release 1.0

User Guide and Internal Design Documentation

Stephen Slivan

MIT, Department of Earth, Atmospheric, and Planetary Sciences,

Cambridge, MA 02139

(slivan@lcs.mit.edu)

$Revision: 1.11.1.1 $; Last submitted $Date: 89/07/30 21:55:52 $ $Author: slivan $

Roadmap:

- Section B.1 is an overview of what OSBERT does.
- Section B.2 contains detailed information about using OSBERT helpful for a *user*. It includes input and output file format descriptions.
- Section B.3 contains detailed information about how OSBERT performs its computations, including equations and relations implemented.
- Section B.4 contains other design documentation possibly useful for the aspiring OSBERT maintainer.[17] (This is the least coherent section; it's mostly chunks of the Preliminary and Revised Design Documents that survived the actual writing of the program code.)

---

[17]Those unfamiliar with Steve's style of coding in C are recommended to read Appendix D, "A Style of C for 'Programming in the Large'", before slogging through any C source files. It contains generally useful information about using the 'cluster'-style of abstract data types and stand-alone procedures in C.

B.1 Overview

Concept: OSBERT (*Occultation Star/Body Evaluator for Radial Tagging*) is a stand-alone C program to tag by star/body radial distance and position angle occultation data initially tagged by time. This is useful in the case that the chord described by the star passing behind the occulting body, as seen by the observer, does not pass through the geometric center of the shadow. Its results can be used to refine an occultation prediction for a given site, or as part of occultation data reduction:

- remove an "equal intervals = equal radial distance" assumption in atmospheric isothermal fits and inversions, and in ring-profile plots
- 'sky-plane' plot of an event
- other plots (e.g. radial distance vs. time)
- information for more elaborate isothermal/haze atmospheric models & fits

B.1.1 Information Produced by OSBERT

All output information is provided in software-readable form; specifically, a C-FITS[18] template file defined ahead-of-time is "filled in". For output format details see Section B.2.3. Primary results are:

- UTC time corresponding to the first result output data pair, time interval between result pairs, and total number of result pairs. This defines the set of times for which output values have been computed; call this set $T(i)$.
- Distance in kilometers from occulted star to occulting body center as seen by the observer (called $R(T)$ in this document), for each time $T(i)$ of a data point.
- Position angle in radians (called $\theta(T)$ in this document), measured from North

[18]FITS-related format used by this group

through East, of the ray on the fundamental plane beginning at the shadow center and extending through the occulting star's position, for each time $T(i)$ of a data point.

B.1.2 Input Information Typically Available to User

- UTC date and start-time for beginning of data, offset from start-time to mid-time of first data exposure, interval separating beginning of each integration, and number of data points.
- Observer *geodetic* latitude, longitude, and altitude above sea level for some set of UTC t_{obs}. This information may be either constant over time as in the case of a ground-based observation, or may be a set of values indexed by time in the case of an observer in motion (e.g. KAO). It may or may not already exist in software-readable form (e.g, the KAO 'Housekeeping Tape').
- RA and Dec of occulted star for some standard epoch, possibly with other information used for critical computations of its apparent position (components of proper motion, parallax, radial velocity).
- Occulting body geocentric RA, Dec, and distance for some set of TDT t_{body}. Like observer location, this information may or may not already exist as a software-readable ephemeris. May also have some external correction offsets to apply to body position as well.

B.2 External Specification

B.2.1 Running OSBERT

To run the OSBERT program, use the following command-line syntax:[19]

---

[19]Throughout this section, terminal input and output are printed in a fixed-width font, with sample terminal input from the user being underlined.

```
% osbert [infile]
```

where *infile* is an optional command-line argument. (Note that the string *infile* is not case-folded. Similarly, throughout the OSBERT program no case-folding is done in order to be consistent with UNIX, under which OSBERT was developed and is likely being run.) You get[20]

```
Hello and welcome to osbert:
Occultation Star/Body Evaluator for Radial Tagging
```

If specified, *infile* is the name of an ASCII indirect file in the format of a C-FITS header whose keywords in turn specify file names for one or more of the input files required by OSBERT (data file format details in Section B.2.2):

- data times file; keyword 'TIMEFILE'
- observer positions; keyword 'OBSFILE'
- occulted star ephemeris; keyword 'STARFILE'
- occulting body ephemeris; keyword 'BODYFILE'
- independent star/body position corrections; keyword 'CORFILE'
- output file to be created; keyword 'OUTFILE'

If no *infile* is specified in the run string, the user is prompted at the terminal for an *infile* name:

```
     Enter name for indirect file,
NONE to manually enter file names,
                or QUIT to abort: QUIT
```

Entering only a <ret> here causes the prompt to be repeated.

If a name for *infile* is supplied, OSBERT will attempt to use it as the indirect file. If QUIT is entered, OSBERT exits. If NONE is entered, the user is prompted at the terminal for each individual data file name:

---

[20]All sample output from OSBERT in this document is in English, which is accurate if the program you're using was NOT compiled with the '-DHawaiian' flag.

```
Enter name for data times file, or QUIT to abort: P8.TIM
Enter name for observer position file, or QUIT to abort: P8.OBS
Enter name for star ephemeris file, or QUIT to abort: P8.STAR
Enter name for body ephemeris file, or QUIT to abort: PLUTO.BOD
Enter name for position corrections file, or QUIT to abort: P8.COR
Enter name for output file, or QUIT to abort: P8.OUT
```

Entering 'QUIT' to any file name prompt will return the user to the prompt for a name for *infile* described above. Entering only a <return> causes the prompt to repeat.

The file name for TIMEFILE may be given as 'NONE', in which case the user will be prompted at the terminal later on for the information required to determine the times for which output values are to be calculated. (See Section B.2.2.2 for details.) NONE is not allowed as a filename entry for any file other than the TIMEFILE; its use gives the error

<bell>NONE disallowed as filename for [keyword]

and treats the entry as if it were left blank.

If an *infile* is specified but is not openable (e.g. file not found or other error) the error message

<bell>Error trying to open indirect file *infile*

is written to the terminal, followed by the prompt for a different *infile* name.

If an openable *infile* is specified in which one or more of the input file keywords is missing or is present but with no string value, or if a file specified in the indirect file cannot be opened, the user is prompted at the terminal for manual entry of names for only those input data files OSBERT didn't get from the indirect file.

If a file-opening error of some sort occurs when OSBERT tries to open a file whose name has just been entered interactively by the user, the error message

Error trying to open data file *data file name*

is followed by a repeat of the prompt.

Note that the values for all these filename keywords are preserved in the final output file produced by OSBERT. This means that it is possible to use the final output file directly as the indirect input file for another run of OSBERT, if for some reason it becomes necessary to reproduce the computations.

B.2.2 Input file formats

Described here are the lexical formats and specific units in which OSBERT expects to find the various data files it will use for radial tagging. Since the raw data files will come from many different sources, each with its own particular format, raw files which do not conform to these OSBERT expectations should have their lexical format and units[21] converted by means of a *separate* software tool written specifically to convert that particular file format to the appropriate OSBERT format described here.[22] (Having separate tools relieves one of having to relink OSBERT each time a file with a different raw format is to be used with the program.) The OSBERT input file formats are all ASCII text files, and are in fact a superset of the C-FITS header and ASCII table file formats already in use by the group. This makes OSBERT compatible with C-FITS in the sense that a C-FITS file which happens to already contain the information OSBERT is expecting may be used directly by OSBERT. If one has some other format of raw data instead, the ASCII text format of OSBERT files simplifies the tasks of developing the lexical conversion tools and verifying that they're producing correct output.

---

[21] including time-scale conversions which may be necessary in an oddball case, such as having EDT but needing UTC

[22] a yet unaddressed issue is the library organization of these tools

B.2.2.1 OSBERT ASCII format

These are the general specifications of the format in which OSBERT expects to find its ASCII text input files, and they apply to each of the following individual input file descriptions:

- blank lines (empty or spaces only) are ignored.
- 'comment' lines are those whose first eight characters are spaces, or whose first non-space character is a forward slash. OSBERT ignores comment lines.
- 'header' lines are of the form

  [keyword] = [value] / [comment]

  The keyword is NOT optional; the first non-space character of a line may not be an equal sign. The value is lexically optional (but will not be semantically optional if OSBERT needs it). The comment is optional; if included it must be separated from the rest of the line by a front-slash. If a front-slash character is to be included as part of a character string [value], then the string must be enclosed in a pair of double quotes; double quotes are optional for string values otherwise. Note that if an opening double quotes is not closed, the value will be taken to be an empty string.
- If any ASCII data lines are to follow the last header line, a single line consisting of only "END" is required to separate the last header line from the first data line.
- data lines consist of *entries* separated by one or more spaces, each of which in turn consist of *fields* of characters, with each field similarly separated by one or more spaces. Any number of entries may appear on the line; the only restriction is that the last field of each entry must have a decimal point (to distinguish it as being that entry's last field). For example, this line has three entries, having three fields, two fields, and one field, respectively:

  18 36 32.67 +48 56.3 J1988.

B.2.2.2 Data times file (TIMEFILE)

A header containing the following keyword values:

- DATE-OBS - UTC date of start of first datum (yyyy mm dd)
- STRTTIME - UTC time of start of first datum (hh., hh mm., or hh mm ss., optionally with 0 or more digits after the optional decimal point)
- NAXIS3 - total number of data taken
- CDELT3 - time interval between 2 successive start-of-data (sec)
- CRVAL3 - offset from STRTTIME to center of first datum (sec)

If CRVAL3 is omitted its value will default to zero; all of the other keywords values are explicitly required.

If a lexical or semantic error is detected in a TIMEFILE header line, an appropriate message (one of the following) is printed and OSBERT is aborted:

Duplicate found in TIMEFILE for keyword *keyword*
Lexical error in TIMEFILE for keyword *keyword*
TIMEFILE missing keyword *keyword*
Semantic error in TIMEFILE values: offset > interval
Semantic error in TIMEFILE values: offset < 0.0
Semantic error in TIMEFILE values: interval <= 0.0
Semantic error in TIMEFILE values: num_data < 1

If TIMEFILE="NONE", then the user is prompted at the terminal for this information:

```
Enter value for UTC start date (yyyy mm dd): 1988 08 22
Enter value for UTC start time (hh mm ss.ss): 17 31 12
Enter value for offset (sec) to midtime of first datum: 0.05
Enter value for interval (sec) between successive data: 0.2
Enter value for number of data: 200
```

Entering only a <return> in response to a prompt causes the prompt to be repeated. Entering "QUIT" to any prompt aborts the OSBERT run.

If a lexical or semantic error is detected on a value entered interactively, an appropriate message (one of the following) is printed at the terminal and its prompt is repeated:

Lexical error in value for keyword *keyword*

```
Semantic error in TIMEFILE values: offset > interval
Semantic error in TIMEFILE values: offset < 0.0
Semantic error in TIMEFILE values: interval <= 0.0
Semantic error in TIMEFILE values: num_data < 1
```

B.2.2.3 Observer position (OBSFILE)

A header containing the following keyword values:

- L_ORDER - order of interpolation to use for longitudes
- F_ORDER - order of interpolation to use for latitudes
- AL_ORDER - order of interpolation to use for altitudes
- UNDULATN - value (in meters) for 'undulation of the geoid' for the area covered by the location(s) in this position file (see global contour map included with documentation for procedure `gd2gc` in Section C.2.1).[23]

Values for L_ORDER, F_ORDER, and AL_ORDER are required; UNDULATN is optional and defaults to 0.0.

If a lexical or semantic error is detected in the OBSFILE header, an appropriate message (one of the following) is printed and OSBERT is aborted:

```
Duplicate found in OBSFILE for keyword keyword
Lexical error in OBSFILE for keyword keyword
OBSFILE missing keyword keyword
OBSFILE interpolation order < 0
```

These header lines are followed by one or more ASCII data lines, with one data line for each tabulated observer position. Each data line has 4 data:

```
UTC             East Longitude    Geodetic latitude    Altitude(m)
TIME string     THETA string      PHI string           double string
```

Note that east longitude has now been accepted as some sort of standard usage for astronomy.

---

[23]approximation made here that this value constant for all locations in this OBSFILE; cases may exist where this would have to be modified if using correct values were very important (e.g. super-critical observations spanning hours made from an airborne telescope).

Altitude is taken to be meters above mean sea level. For specifics of valid lexical formats for conversion into the data types TIME, THETA, and PHI, see on-line 'man' pages for libss.a (TIME$utc_parse, THETA$dms_parse, PHI$dms_parse).

Example data line, corresponding to UTC 15 Sep 1988 13:08:46.00, longitude *W* 78°52′32.″0, latitude *N* 43°03′13″, altitude 183 *m*:

```
1988 09 15 13 08 46.00  -78 52 32.0  +43 03 13.  183.
```

If a lexical or semantic error is detected in an OBSFILE data line, an appropriate message (one of the following) is printed and OSBERT is aborted:

```
No data lines found in OBSFILE
OBSFILE data line missing 1 or more fields
In OBSFILE, TIME$utc_parse(<time>) [UTC] signals bounds
In OBSFILE, TIME$utc_parse(<time>) [UTC] signals bad_format
In OBSFILE, THETA$dms_parse(<angle>) [longitude] signals bounds
In OBSFILE, THETA$dms_parse(<angle>) [longitude] signals bad_format
In OBSFILE, PHI$dms_parse(<angle>) [latitude] signals bounds
In OBSFILE, PHI$dms_parse(<angle>) [latitude] signals bad_format
OBSFILE: s2d(<real>) [altitude] signals bad_character
OBSFILE: s2d(<real>) [altitude] signals extra_decimal
OBSFILE: s2d(<real>) [altitude] signals float_overflow
OBSFILE: s2d(<real>) [altitude] signals float_underflow
```

Note that for a non-moving observer, OBSFILE would only have one data line. The keywords would specify 0^{th} order interpolation for all three data values, followed by END and the single data line. In that case, the particular value given for the UTC entry would be unimportant.

B.2.2.4 Occulted star position (STARFILE)

A header containing the following keyword values:

- EPOCH - Epoch of RA & Dec, as a TIME epoch (e.g. B1950.0)
- A_STAR - RA of star (THETA-format hms string)
- D_STAR - Dec of star (PHI-format dms string)
- M_EPOCH - Epoch of proper motion values, as a TIME epoch
- MA_STAR - RA proper motion (arcsec per century)

- MD_STAR - Dec proper motion (arcsec per century)
- N_STAR - radial velocity of star, if known (km · sec^{-1})
- P_STAR - parallax of star, if known (arcsec)

Values for EPOCH, A_STAR, and D_STAR are required; the others are optional. For specifics of valid lexical formats for conversion into the data types TIME, THETA, and PHI, see on-line 'man' pages for **libss.a** (TIME$ep_parse, THETA$hms_parse, PHI$dms_parse).

If a lexical or semantic error is detected in the STARFILE header, an appropriate message (one of the following) is printed and OSBERT is aborted:

Duplicate found in STARFILE for keyword *keyword*
Lexical error in STARFILE for keyword *keyword*
STARFILE missing keyword *keyword*

B.2.2.5 Occulting body position file (BODYFILE)

A header containing keyword values for:

- A_ORDER - order of interpolation for RA
- D_ORDER - order of interpolation for Dec
- R_ORDER - order of interpolation for geocentric distance

All three of these values must be present.

If a lexical or semantic error is detected in the BODYFILE header, an appropriate message (one of the following) is printed and OSBERT is aborted:

Duplicate found in BODYFILE for keyword *keyword*
Lexical error in BODYFILE for keyword *keyword*
BODYFILE missing keyword *keyword*
BODYFILE interpolation order < 0

These header lines are followed by one or more ASCII data lines, with one line for each tabulated body position. Each data line has 5 data:

TDT	Right ascension	Declination	Distance(AU)	Epoch
TIME string	THETA string	PHI string	double string	TIME string

The RA, Dec, and distance in A.U. are taken to be geocentric values, and the RA and Dec values in this file are taken to be light-time corrected. For specifics of valid lexical formats for conversion into the data types TIME, THETA, and PHI, see on-line 'man' pages for libss.a (TIME$tdt_parse, THETA$hms_parse, PHI$dms_parse, TIME$ep_parse).

Example line, corresponding to TDT 12 Aug 1988 0^h, RA $16^h24^m39^s.2631$, Dec +56°08′21.″332, geocentric distance 4.9312 A.U., and epoch of RA and Dec B1950.0:

```
1988 08 12  16 24 39.2631  +56 08 21.332  4.9312  B1950.0
```

If a lexical or semantic error is detected in a BODYFILE data line, an appropriate message (one of the following) is printed and OSBERT is aborted:

```
No data lines found in BODYFILE
BODYFILE data line missing 1 or more fields
```

B.2.2.6 File of corrections to body ephemeris (CORFILE)

File header containing the following keyword values:

- D_EPOCH - Epoch of corrections
- DA_ORDER - order of interpolation for RA corrections
- DD_ORDER - order of interpolation for Dec corrections

All three values must be present.

If a lexical or semantic error is detected in the CORFILE header, an appropriate message (one of the following) is printed and OSBERT is aborted:

```
Duplicate found in CORFILE for keyword keyword
Lexical error in CORFILE for keyword keyword
CORFILE missing keyword keyword
CORFILE interpolation order < 0
```

These header lines are followed by one or more ASCII data lines, with one line for each tabulated body correction. Each data line has 4 data:

TDT	Δα (seconds)	Δδ (arcsec)	Epoch
TIME string	float string	float string	TIME string

The sense of the corrections is such that the values given are applied to the occulting body position. The corrections are taken to be applied to geocentric values, and also to be light-time corrected. For specifics of valid lexical format for conversion into the data type TIME see on-line 'man' pages for **libss.a** (TIME$tdt_parse, TIME$ep_parse).

Example line, corresponding to TDT 12 Aug 1988 14:12:16.00, $\Delta\alpha$ = +0^{s}.03600, $\Delta\delta$ = -0".2368, and epoch of corrections B1950.0:

```
1988 08 12 14 12 16.00  0.03600  -0.2368  B1950.0
```

If a lexical or semantic error is detected in a CORFILE data line, an appropriate message (one of the following) is printed and OSBERT is aborted:

```
No data lines found in CORFILE
CORFILE data line missing 1 or more fields
```

B.2.3 Output file format

OSBERT outputs a C-FITS file. The C-FITS header template file `osbert_out.ots` is used to produce this file; the header portion includes among its keywords the following keyword values:

- TIMEFILE, OBSFILE, BODYFILE, STARFILE, CORFILE, OUTFILE: file names of data input files
- CTIME1: UTC date and time (as a `TIME` string) corresponding to first output data pair.
- CRVAL1: 'offset to midtime of first datum'. Because of the way CTIME1 is defined, being the actual time of interest rather than a reference time from which to offset, CRVAL1 will always be 0.0 and will be written out as such.
- CDELT1: interval between successive data points given output values
- NAXIS1: total number of data points for which are given output values
- DEL_T: boolean value; TRUE if extrapolated ΔT values used in UT1 computations (includes UT Julian Date and GMST/GAST computations) for the input data set.

- DEL_AT: boolean value: TRUE if extrapolated ΔAT values used in UTC computations for the input data set.

The data portion contains the radial distance in kilometers on the fundamental plane from the center of the body shadow to the star position, and the position angle in radians of the star position with respect to the shadow center, measured North through East. (In terms of the 'sky plane' described in [ELLI78], the shadow center is taken to be (0,0), so the star position is (ξ',η').)

B.2.4 Establishing genealogy of the OSBERT you're running

You can see which levels of source revisions were used for the object files linked to a particular executable copy of OSBERT with the Unix 'ident' command:

```
% ident osbert

osbert:
     Header: body_cvt.c,v 1.5 88/10/26 18:59:51 slivan Submitted $
     Header: d2s.c,v 1.3 88/10/11 15:12:11 slivan Released $
     Header: failure.c,v 1.3 88/10/11 15:14:13 slivan Released $
     Header: gd2gc.c,v 1.2 88/10/11 13:20:09 slivan Released $
     Header: hour_angle.c,v 1.2 88/10/11 13:27:39 slivan Released $
     Header: interpolate.c,v 1.1 88/10/26 18:55:55 slivan Submitted $
     Header: strsegment.c,v 1.1 88/10/05 11:50:25 slivan Submitted $
     Header: obs_cvt.c,v 1.5 88/10/26 18:59:51 slivan Submitted $
     Header: osbert.c,v 1.4 88/10/29 15:20:13 slivan Exp $
     Header: out_unparse.c,v 1.1 88/10/12 15:53:17 slivan Exp $
     Header: parse3f.c,v 1.3 88/10/11 15:15:25 slivan Released $
     Header: pdataline.c,v 1.2 88/10/18 11:24:25 slivan Submitted $
     Header: phi.c,v 1.4 88/10/11 15:16:55 slivan Released $
     Header: precess.c,v 1.5 88/10/26 18:19:29 slivan Released $
     Header: round.c,v 1.2 88/10/11 15:22:26 slivan Released $
     Header: s2d.c,v 1.3 88/10/11 15:22:18 slivan Released $
     Header: search.c,v 1.1 88/10/11 13:48:23 slivan Released $
     Header: star_cvt.c,v 1.2 88/10/26 19:00:12 slivan Submitted $
     Header: theta.c,v 1.7 88/10/11 15:21:34 slivan Released $
     Header: time.c,v 1.2 88/10/11 15:02:13 slivan Submitted $
     Header: time_cvt.c,v 1.2 88/10/26 18:59:27 slivan Submitted $
     Header: trunc.c,v 1.2 88/10/11 15:20:34 slivan Released $
```

B.3 Internal Design - details relevant to application

B.3.1 General information

Accuracy/precision - the OSBERT program code uses double-precision arithmetic for all mathematical computations (as do the procedures from **libss.a** which OSBERT uses). This eliminates machine round-off as a source of loss of significant digits, but still requires that additional special handling (included in both OSBERT and **libss.a**) be present to prevent 'decimal trash' in the rightmost figures from fouling up results. The user of the results will be responsible for determining the actual precision of the values produced by OSBERT, based on the precision of the data supplied as input.

OSBERT also uses certain constant values defined in the include file `constants.h` associated with **libss.a**:

- astronomical unit (m)
- Earth equatorial radius (m)
- number of arcsec per radian

Notation used in algorithm descriptions:

λ_{Obs} = longitude of observer
ϕ_{Obs} = geodetic latitude of observer
Alt_{Obs} = altitude of observer above mean sea level

ϕ'_{Obs} = geocentric latitude of observer
ρ'_{Obs} = geocentric distance of observer

α_{star} = apparent RA of occulted star
δ_{star} = apparent Dec of occulted star

ρ_{body} = geocentric distance of occulting body
α_{body} = apparent geocentric RA of occulting body (uncorrected)
δ_{body} = apparent geocentric Dec of occulting body (uncorrected)

α'_{body} = corrected α_{body}
δ'_{body} = corrected δ_{body}

B.3.2 Computation of times for which output is calculated

This information is computed from the following information available in the headers of Snapshot data files:

- UTC start-time of first datum
- offset to center-time of first datum
- interval between each start-of-datum
- number of data

The $T(i)$ that OSBERT uses as data times are defined as the center-time of the data, and these are calculated as described by the following pseudo-code:

```
BEGIN { time_cvt }
   T[0] := TIME$utc_parse(start time of first datum);
   T[0] := T[0] + (offset to center of first datum);
   FOR i := 1 TO (number of data - 1) DO BEGIN
      T[i] := T[0] + (i*interval);
   END;
END.
```

Note that the `TIME` cluster is left to deal with the details of conversion of UTC time to a `TIME` object. (See on-line 'man' pages for **libss.a** for `TIME` details.)

B.3.3 Computation of observer geocentric position

Uses the `gd2gc` procedure in **libss.a** to compute geocentric latitude and distance of the observer (See on-line 'man' pages for **libss.a** for `gd2gc` details):

```
BEGIN { obs_cvt }
   search for entries in OBSFILE data around time T;
   interpolate to get λ_Obs, φ_Obs, Alt_Obs at time T;
   gd2gc(φ_Obs, Alt_Obs, UNDULATN) to get φ'_Obs, ρ'_Obs;
END;
```

B.3.4 Computation of apparent coordinates of occulted star

IMPLEMENTATION NOTE: initial version of OSBERT handles only precession to date of interest; proper motion, parallax, and radial velocity values if present will be parsed and read in from the input file but *not* used in any calculation. Uses the `precess` procedure in libss.a to handle precession to equinox of date. (See on-line 'man' pages for libss.a for details of `precess`.)

```
BEGIN { star_cvt }
   precess(EPOCH,T,A_STAR,D_STAR) to get α_star, δ_star
      at epoch T;
END;
```

B.3.5 Computation of apparent coordinates of occulting body

$$\alpha'_{body} = \alpha_{body} + \Delta\alpha$$
$$\delta'_{body} = \delta_{body} + \Delta\delta$$

→ potential problem here as $|\delta'_{body}|$ could be > 90°. If so, the condition is taken to be an error.

```
BEGIN { body_cvt }
   interpolate tabulated data to TIME T to get α_body, δ_body, ρ_body, Epoch_0;

   use cor_cvt(T) to get Δα, Δδ, epoch_C (epoch for corrections);

   { apply corrections }
   precess(Epoch_0, epoch_C, α_0, δ_0) to get α_1, δ_1;
   α_2 := α_1 + Δα;
   δ_2 := δ_1 + Δδ;

   { precess to instantaneous epoch T }
   precess(epoch_C, T, α_2, δ_2) to get α'_body, δ'_body;
END;
```

B.3.6 OSBERT's computations on the Fundamental Plane

Code in OSBERT 'main' itself does the computations for Smart's "fundamental plane" [SMAR77, page 369], computing shadow center and observer position in the Fundamental Plane of the occultation. The final desired results are

- $R(T)$, the distance on that plane between these two points
- $\theta(T)$, the position angle of the observer with respect to the shadow center

This is essentially a fairly straightforward implementation of stuff found in [SMAR77, Section 204]:

{page 371}

$$x_{shadow} := \rho_{body} \cdot \cos \delta'_{body} \cdot \sin (\alpha'_{body} - \alpha_{star})$$
$$y_{shadow} := \rho_{body} \cdot (\sin \delta'_{body} \cdot \cos \delta_{star} - \cos \delta'_{body} \cdot \sin \delta_{star} \cdot \cos (\alpha'_{body} - \alpha_{star}))$$

{page 372}

$$x_{Obs} := \rho'_{Obs} \cdot \cos \phi'_{Obs} \cdot \sin HA_{star}$$
$$y_{Obs} := \rho'_{Obs} \cdot (\cos \delta_{star} \cdot \sin \phi'_{Obs} - \sin \delta_{star} \cdot \cos \phi'_{Obs} \cdot \cos HA_{star})$$

{final result}

$$R(T) := \sqrt{(x_{Obs} - x_{shadow})^2 + (y_{Obs} - y_{shadow})^2}$$
$$\theta(T) := \text{position angle} = \frac{\pi}{2} - \text{atan}(y/x) = \text{atan}(x/y)$$
$$:= \text{atan2}(x_{Obs} - x_{shadow}, y_{Obs} - y_{shadow})$$

```
BEGIN { osbert main }
   Open indirect file to get data file names;
   Open data files;
   Prompt for data files
      (those omitted or for which got an error);

   use time_cvt to get set of TIMEs T[]
   FOR each TIME T for which computation to be done DO BEGIN
      use obs_cvt to get observer position and geocentric distance
         at TIME T;
      use star_cvt to get star position;
      use body_cvt to get occulting body position & distance in AU;
      make distance to body units of Earth's equatorial radius;
      compute hour angle of occulted star;
      compute position of observer on Fundamental Plane;
      compute position of shadow center on fundamental plane;
```

```
      compute distance & position angle, shadow-to-observer;
   END;

   use out_unparse to write the results to the output file;
END;
```

B.3.7 precision of results

The radius and angle results of OSBERT are encoded to integers before being written to the C-FITS output file. The original computations of these `double` values are limited to 14 significant decimal digits of precision on the VAX/750 (plus a fifteenth to round them properly). The internal procedure `out_unparse` uses 32-bit `long` integers as the representation for its encoded output values, which appears to reliably yield 12 significant decimal digits in the final decoded `double` output values. An earlier implementation using `shorts` yielded only 7 digits. The scaling factors and offset values for the encoding and decoding are written to the output file header to their limiting machine precision of 14 significant digits.

B.4 Internal Design - other information

OSBERT was developed on a VAX 750 running BSD 4.2 Unix. It was developed such that it would conform to those restrictions which apply to GC-type Unix systems:

- AT&T Unix systems' C (e.g., like the GC's) is a superset of the C described in [KERN78]; there are a very few aspects of C which were added almost immediately after C began to be used outside of Bell Labs which are *not* included in the original book. In practice, it is nowadays effectively impossible to find a compiler in current use which doesn't support these early modifications. Two of these extensions are used by OSBERT: the `enum` enumerated type definition, and the `void` type for function return values. Other than `enum` and `void`, OSBERT conforms to [KERN78].

- 14-character file names
- Identifiers unique within 7 characters

B.4.1 portability aspects

The modules of OSBERT use the **libss.a** library and the `fits_mgr.o` object module; thus the pathnames for the associated `#include` files in the sources as well as for those two binaries in the makefile are dependent on the external directory structure.

B.4.2 diagnostic printouts capability

Embedded in the source code are diagnostic printf statements that were used during testing and verification of OSBERT; they may be activated by compiling with a -DDIAGNOSTICS flag.

B.4.3 Significant changes from Preliminary Design

1. Moving of lowest-level raw input file parsing and units-conversion *outside* of OSBERT to separate software tools (see Section B.2.2)
2. Handling of star ephemeris (erroneously had as a fixed position in the Preliminary Design)
3. Output position angle of star w.r.t. shadow center as well as distance from shadow center
4. Defines details of user interaction (config. file / manual entry)
5. Former name REBINNER deemed misleading
6. Changes to reflect use of TIME, THETA, PHI clusters

B.4.4 Information Required for Fundamental Plane Computations

- A set of TIME values (call them $T(i)$) corresponding to the data points to be tagged.

- Observer geocentric latitude, longitude, and geocentric distance for each $T(i)$.
- Occulting body Geocentric RA, Dec, and distance for each $T(i)$, corrected with respect to occulted star, also corrected for light-travel time.[24]
- Ephemeris of apparent RA and Dec of occulted star, for each $T(i)$. (*Apparent* position is required for the calculations to work properly.)
- Some independently-known values related to geometry of Earth (equatorial radius $a_{\oplus}$, length of astronomical unit).

B.4.5 Global Design

B.4.5.1 Design Issues

The great variance in information sources, castings (e.g. geocentric vs. geodetic, apparent coordinates vs. reference equinox, UTC vs. UT vs. TDT) and software-readable formats dictates that a great deal of flexibility be available within the overall design of OSBERT. Specifically, the core of the computations should be as completely independent as possible from all details of the raw input and output file formats.

It seems that in critical cases where an interpolation is necessary, it would be best to leave it to be the *last* step, after any other computations (e.g. converting geodetic latitude to geocentric). (I suspect that rigorous precession is safely commutative though, as it is simply a rotation on a sphere.) This complicates the implementation however, so Release 1 will be implemented to interpolate first and perform other calculations later.

From a program-as-a-whole point of view, localizing the interactions with details of

---

24

- planetary aberration = stellar aberration + light-time correction
- stellar aberration = diurnal aberration + annual aberration + secular aberration
- annual aberration includes e-terms

input/output file formats should drive the design of OSBERT. The most straightforward way to handle this localization is to include a separate entity for each input and output information set, to shield the rest of the program completely from specifics of input and output file format. Such an entity's only purpose in life is to convert its information between the well-defined form in which the main computation entity wants it and whatever form the anarchistic outside world has decreed the information is in. In fact, such an entity may best be handled by including as part of it an independent software tool, external to OSBERT, so that relinking OSBERT isn't required each time a input file of some different format is to be used, and so that this parsing software can be easily saved and re-used later on. The file standards between these tools and the OSBERT program itself are to be ASCII text files of explicit format. An ASCII header/data format resembling the C-FITS style possibly could be made to work here. By defining an interface to the rest of the program which is clear even to plankton, this separation allows for changing the input parsing to match any given information set without requiring someone to go mucking around in the core computation code. Such a separation is also useful for localizing certain goofs should debugging become necessary (e.g., the writer of a parsing tool needs nothing more complex than an ordinary text editor to check the output of his/her program).

As OSBERT is an interactive program, implementation of a 'self-destruct' feature (in the spirit of [SLIV86, pp. 153-154]) would not be inappropriate. In particular, the user should have available a graceful means to abort and exit at any point in the user/OSBERT interaction.

B.4.5.2 OSBERT Design

The following procedures are used by OSBERT (see Figure B-1):

`time_cvt`	accepts file containing information from which a TIME T is calculated for each data point (or image, or whatever).
`obs_cvt`	accepts TIME T and input format of observer's location information, producing observer's longitude λ_{Obs}, geocentric latitude ϕ'_{Obs}, and geocentric distance ρ'_{Obs}.
`star_cvt`	accepts TIME T and input format of celestial coordinates of the occulted star, producing apparent coordinates of the star for the OSBERT computations.
`body_cvt`	accepts TIME T and input format of occulting body celestial coordinates information, producing the corrected geocentric apparent coordinates: RA α'_{body} and Dec δ'_{body}, and the geocentric distance ρ_{body}.
`cor_cvt`	accepts TIME T and input format of any independent corrections to the body ephemeris (as may arise, for example, from retrospectively revising the location of the occultation chord on the shadow), and produces correction offsets to be added to the body's coordinates.
`out_unparse`	accepts the results of the OSBERT computations $R(T)$, $\theta(T)$ and mangles them into a C-FITS file output format for the user.

With respect to opening the input files, it will be convenient to have the top-level code in OSBERT which handles the user interactions be that which actually opens the files, to facilitate error-handling in the case of file-opening errors. The lower-level modules inside OSBERT thus only see already-opened files. Body and star coordinates from different sources will almost surely require an offset with respect to each other so that they agree with independent computations of precisely where on the disc the star's occultation chord should lie. This particular breakdown has the star-vs.-body correction applied individually to the coordinates of the body for each T rather than just applying a correction once to the star; this correlates better conceptually with the possibility of allowing the future application of a non-constant (over T) correction to the system.

OSBERT is only doing the computation for one value of T at a time, so there's no need for

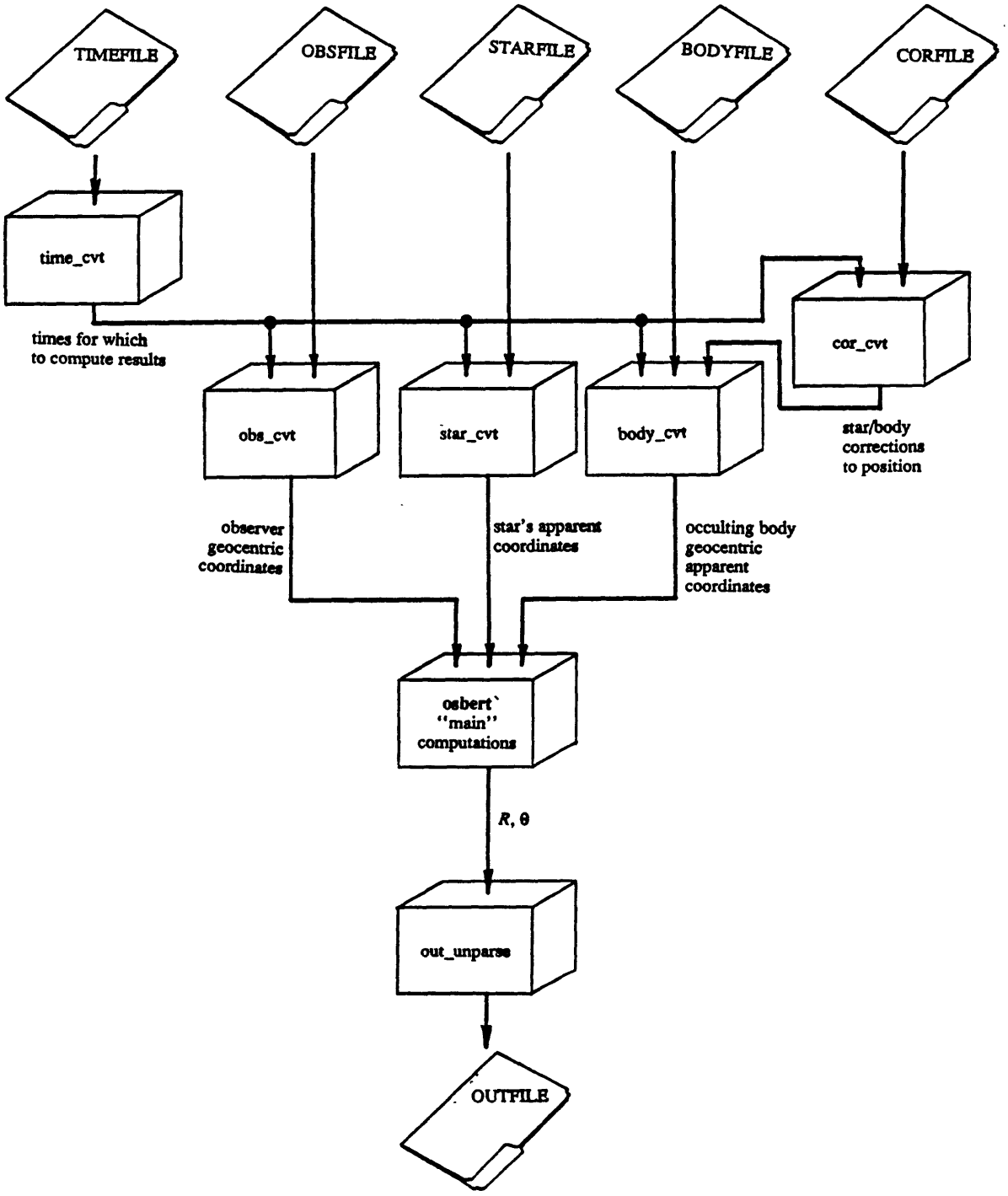

Figure B-1: OSBERT conceptual modular decomposition

each of the input entities to run off and produce processed values for ALL T at once. On the other hand, any of the input entities which do interpolation need to have more than one T's worth of *raw* information available simultaneously to do the interpolation. The input entities will keep internally a complete set of the raw input data, but OSBERT will only request *processed* input one T at a time.

This leads naturally to an implementation of OSBERT which conceptually would run something like this:

```
BEGIN
   time_cvt computes all T of interest;
   star_cvt converts & checks raw coordinates of occulted star;
   obs_cvt, cor_cvt, and body_cvt read in all their raw data
      for later interpolation;

   FOR i := 1 TO numdata DO BEGIN
      { T[i] is UT1 for data point #i }
      obs_cvt computes φ'Obs, λObs, ρ'Obs for T;
      star_cvt computes αstar, δstar for time T;
      body/cor_cvt computes α'body, δ'body, ρbody for T;
      OSBERT computes the results Y[i] and θ[i] for time T;
   END;

   set of all Y[i], θ[i] converted to final form by out_unparse;

END. { OSBERT }
```

B.4.6 OSBERT main procedure

Internally, the computations require uniform units of input values; in particular, distances must be in units of Earth's (geoid) semi-major axis, since this is required for Smart's relations to work

B.4.7 Procedure `time_cvt`

This entity accepts the file containing information from which a TIME T is calculated for each R, θ to be computed.

B.4.7.1 Design Issues

T(i) - The data times themselves will typically NOT be available in UT1; rather, they're very likely to be in UTC (since that's what's broadcast over WWV).

B.4.8 Procedure `obs_cvt`

`obs_cvt` accepts time T and input format of observer's location information, producing observer's longitude λ_{Obs}, geocentric latitude ϕ'_{Obs}, and geocentric distance ρ'_{Obs}. Specifically, it must

1. parse the information's input format
2. make the latitude geocentric
3. interpolate positions to TIME T

B.4.8.1 Design Issues

format of available information

- interval of tabulation not necessarily regular (as in the case of positions recorded by hand by the KAO navigator). It will be possible to implement some special kind of interpolation which could deal with irregular intervals, but need not be implemented for Release 1.[25]

OSBERT will be written to handle the general case of an observer in motion, so that in the case of a fixed observer, the `obs_cvt` entity would return values independent of T for λ_{Obs}, ϕ'_{Obs}, and ρ'_{Obs}.

---

[25]unless required for the Pluto/P8 occultation

The positions at times $T(i)$ will need to be interpolated from the observer's positions tabulated at times t_{Obs} (the actual interpolation could be short-cut in the case of a stationary observer). Noted that the order to which interpolation should be done for a given table is actually a function of the table (depends on how quickly the function changes and how frequently it's tabulated); it will be included as part of the input file.

B.4.9 Procedure `star_cvt`

`star_cvt` accepts input format of celestial coordinates α, δ (and optionally other information related to the position: μ_α, μ_δ, ν, π) of the occulted star and produces the star's apparent coordinates in the form required by the OSBERT computation entity. Full precision would require accounting for proper motion, parallax, and whatever else necessary [AA1988, pp. B39-B41]; Release 1 will deal only with precession. Specifically, `star_cvt` needs to

- parse the input format
- ***restriction on Release 1: No proper motion handling implemented.*** Future enhancement to compute and apply the correction for proper motion (in the proper epoch)
- precess the position to apparent coordinates
- ***restriction on Release 1:*** Unclear about handling of aberration: *stellar aberration = diurnal aberration + annual aberration [e-terms here] + secular aberration.* (Noted that for 'critical' computations, AA also worries about $\mathbf{E_B}$ barycentric position of $\oplus$, $\mathbf{E_B}$ velocity, and $\mathbf{S_B}$ barycentric position of Sun.) Annual aberration requires information about barycentric positions of Sun and Earth, and may have e-terms as well. Noted that 'astrometric' ephemerides before 1984 had e-terms built in, but no longer. Diurnal aberration is easily calculated, but will be same for star & body of occultation, and needs to be clear whether it's contained already in the star & body positions being used.

B.4.9.1 Design Issues

The star has an ephemeris, since apparent coordinates are required for the calculations.

"epoch of proper motion" vs. "epoch of position" issue? (suspect one can't precess a p.m. correction, but can precess a corrected position) (Perhaps "Differential Precession and Nutation" [AA1988, page B21] could be used here, but it looks messier than precessing to epoch of corrections, applying the corrections, then precessing back to final epoch, particularly given the fact that a rigorous precession procedure is already available in **libss.a**.)

B.4.10 Procedure `body_cvt`

`body_cvt` accepts TIME *T*, input format of occulting body celestial coordinates information, (and any independent corrections to these coordinates), producing the corrected geocentric apparent coordinates: RA α'_{body} and Dec δ'_{body}, and the geocentric distance ρ_{body}. Specifically, it must

1. parse the input format of the ephemeris file
2. interpolate position to TIME *T*
3. apply the correction to the position
4. fix up results as required by OSBERT (e.g. precess to apparent coordinates at TIME T)
5. ***restriction on Release 1:*** Not clear precisely what to do with aberration: *planetary aberration = stellar aberration + light-time correction.* For the time being, leave it out and let it be absorbed into the corrections to the body ephemeris.

B.4.10.1 Design Issues

The input format for the ephemeris is particularly ill-defined, as it is subject to change for *each* occultation event (some events may have machine-readable information from various sources each in their own unique format, while the ephemerides for others will exist only on paper). Design proceeds based on the assumptions that all ephemerides will at least

1. minimally contain RA, Dec, and geocentric distance, with the RA and Dec given in either a standard epoch or apparent coordinates.
2. have a uniform tabulation interval within itself. (knowing that different sets may have different intervals)
3. be computed for TDT.

An ephemeris of choice for a given event may exist only on paper, and in such a case would need to be typed into the VAX at least semi-manually in order to provide it to OSBERT in a software-readable format. The modularization of OSBERT input parsing has the effect of making the precise file format for an ephemeris much less of an issue. It will be advantageous to have a file created manually from a paper ephemeris be people-readable and editable to facilitate error detection and correction.

Noted that OSBERT will assume the ephemeris is already light-time corrected.

B.4.11 Procedure `cor_cvt`

Parses the input format of the ephemeris corrections, so `body_cvt` can apply them.

B.4.11.1 Design Issues

An issue arises here having to do with how the coordinate corrections are supplied to the OSBERT program. As the value of the offset may change over successive OSBERT runs (particularly when the body ephemeris changes), these two values should probably not be compiled in as a #DEFINE. Interactive entry each time the program is run is inconvenient

and open up the possibility of introducing user errors, and would be completely inadequate in the case of a non-constant tabulated set of offsets. This thus seems a good place to use some sort of ASCII "configuration file" which can be changed easily when required but relieves the users of having to worry about the offsets at all at run time if the set currently configured is the one to be used. (It was decided to make the corrections be resident in their very own independent input data file; this together with putting the input file name into the OSBERT "indirect" file implemented seems to satisfy `cor_cvt`'s requirements well.)

Issue of epoch of corrections vs. epoch of position: precess coordinates to epoch of corrections, apply corrections, precess results to required final results.

As a change of offset is a fairly subtle one, it will be prudent to include information about which 'corrections' input file, and which ephemeris file, were used in the header of the final output file. (Finally it was decided to include names of ALL input files in the output file header.)

B.4.12 Procedure `out_unparse`

`out_unparse` accepts the results of the OSBERT computations $R(T)$ and $\theta(T)$ and mangles them into the required output format for the user of the radial tag data.

B.4.12.1 Design Issues

Details of output file format:

- IF .hsp is C-FITS, use it.
- OTHERWISE, need to decide whether to use .hsp or .ts?

θ will be given to `out_unparse` in radians, and R will be given in kilometers. Values for the various time-related entries (start of data, interval) will be given in a format similar

to that used in '.ts' format C-FITS templates (e.g. /usr/fits/occultation_data/time_series1.ts). A C-FITS header template will be used to create the header of the output file; this file will define the details of the output format needed.

Arbitrarily choose $\theta = 0$ for $R = 0$.

CRVAL3, when used as part of the C-FITS header of raw data, refers to the offset to be added to the 'start' time to arrive at the midtime of the exposure giving rise to the first datum. In OSBERT, the start time written out is exactly the time for which the first computation was done, so CRVAL1 will always be 0.0.

Support in internal design for future modification: writing partial data set to output file. Although OSBERT computes and writes out data values for all times specified in the input data set (TIMEFILE), the `out_unparse` procedure takes as two arguments starting and ending output value array indices (*start* and *finish*) which could be used to specify the writing of results for only a part of the input data set. Currently, the OSBERT main module calls `out_unparse` with hard-coded arguments of 0 and (num_data-1), respectively. A third argument, of type `TIME`, is to be the `TIME` value corresponding to the results at array index *start*; it's currently coded to be called with OSBERT main's *t[0]*. However, at this time *no* explicit internal support exists for restricting the values of keywords DEL_T and DEL_AT to apply to only a partial data set.

Appendix C

libss.a library

libss.a

Release 2.0

Application User Guide and Internal Design Documentation

Stephen Slivan

MIT Department of Earth, Atmospheric, and Planetary Sciences,

Cambridge, MA 02139

(slivan@lcs.mit.edu)

$Revision: 2.0.1.1 $; Last submitted $Date: 89/08/06 16:41:46 $ $Author: slivan $

Roadmap:[26]

Sections C.1 through C.3 comprise the '**libss.a** User's Guide' part of the document, with information useful to one deciding if and how to use the data types and procedures in the library in an application.

- Section C.1.1 covers the **TIME** cluster, whose objects represent a specific date and time.
- Section C.1.2 covers the **THETA** and **PHI** clusters, whose objects represent angles with certain explicit constraints, such as those used as celestial coordinates.

---

[26]Those unfamiliar with Steve's style of coding in C are recommended to read Appendix D "A Style of C for 'Programming in the Large'" before reading on in this one. It contains generally useful information about using the 'cluster'-style of abstract data types and stand-alone procedures in C.

- Section C.2.1 covers the **gd2gc** procedure, which converts geodetic latitude, altitude above sea level, (and undulation of the geoid, if available) into geocentric latitude and geocentric distance.
- Section C.2.2 covers the **hour_angle** procedure, which computes the hour angle of an object at specified RA for an observer at a specified east longitude at a specified time.
- Section C.2.3 covers the **precess** procedure, which precesses an RA and Dec in some given initial epoch to the corresponding RA and Dec in a second epoch.
- Section C.2.4 covers a set procedures useful for interpolating tabular values. **search** is given a time and locates a position in an array of tabulated values, and a set of interpolation procedures (**float_interpolate, double_interpolate, THETA_interpolate,** and **PHI_interpolate**) use the result from **search** and perform Lagrangian interpolation of the specified order to specified TIME t.
- Section C.3 describes how to use the **failure** procedure for aborting a program on a fatal error. It also includes information about some of the *#include* files used in the implementation of the routines in this library (i.e., *#include* files which aren't explicitly associated with any particular procedure or cluster): constants (mathematical, physical, and machine-dependent) and lexical definitions for the C language which may be useful for, but aren't necessarily *required* in application code using this library.

C.1 Clusters (abstract data types)

C.1.1 TIME: Parse strings representing times and convert between time scales

The TIME object defined in **libss.a** is useful for representing a specific date and time, and provides the applications programmer with an easy means of dealing with the messiness of conversion to, from, and between the various time scales currently in use. The basic concept is that the mapping of moments in time to TIME objects is 1:1 (within the finite precision available from a specific implementation), regardless of the particular time scale from which the TIME object at a given moment was originally computed. Thus, the application can use the operations available in the TIME cluster to accept input information given in any of UTC, UT1, TDT, epochs, or Julian dates (UT or DT) by converting them all to TIME objects, perform simple arithmetic or comparison operations on those values if necessary (e.g. searching for a target TIME in an array of TIMEs representing the independent argument of an ephemeris; see Section C.2.4), and then also convert TIME objects back to human-readable form using any of the available time scales.

The simplest application here would be to parse a string representing a moment of time given in one time scale, such as UTC, and immediately unparse it to its corresponding string in another time scale, such at TDT; presto, instant time scale conversions!

(Note that *sidereal* time is in fact an *angle*, and is thus more appropriately represented with a THETA object (Section C.1.2). Facilities for obtaining the sidereal time, as a THETA object corresponding to a given TIME, are included in the TIME cluster.)

C.1.1.1 operations available

TIME$tdt_parse given a formatted string representing a moment in time given in Terrestrial Dynamical Time, return a valid TIME object corresponding to that moment. The string must contain year, month, and date fields, and optionally may contain hours, minutes, and seconds. The procedure is fairly forgiving with respect to details of string formatting; see procedure specifications for details. This is how, if you have an time represented as a string (such as "1988 08 17 13 07.0" for August 17, 1988 at 13:07.0) you can directly create from it a valid TIME. Similar procedures are available for two other time scales: UTC (**TIME$utc_parse**), and UT1 (**TIME$ut1_parse**). (Special handling for leap-seconds, which don't have a unique UTC representation, is included; see procedure specifications for TIME$utc_parse for details.)

TIME$ep_parse Similar to the above parse functions except that it parses strings representing epochs (e.g. B1950.0, J1988.5, 2000.00). (Note that B1950.0 ≠ UT 0^h 01-Jan-1950!)

TIME$tdt_unparse
Inverse of parse function; returns a string representing TDT time corresponding to a given TIME object. Useful for producing text output of values which is easily understandable to a human being. The caller specifies directly the string formatting; see the procedure specifications for details. Corresponding procedures are available for UTC (**TIME$utc_unparse**), UT1 (**TIME$ut1_unparse**), and epochs (**TIME$ep_unparse**).

TIME$jdt2time, TIME$time2jdt
procedures to convert TIME to and from TDT Julian Date, represented as a double-float value.

TIME$jdu2time, TIME$time2jdu
similar to above; converts TIME from and to UT Julian Date, represented as a double-float value.

TIME$gmst returns a THETA representing Greenwich Mean Sidereal Time corresponding to given TIME object.

TIME$gast returns a THETA representing Greenwich Apparent Sidereal Time corresponding to given TIME object.

TIME$add_interval
returns a new TIME object representing a time a given number of seconds later (earlier) than a supplied TIME argument. (Negative number of seconds yields earlier TIME.)

TIME$difference

returns a double-float representing the number of seconds difference between two given TIMEs.

TIME$difratio fairly specialized procedure which finds use in tabular interpolation (Section C.2.4); given 4 TIMEs it returns ratio of two time differences. Available in case application programmer needs to code their own interpolation procedure(s).

comparison operations **[TIME$gt, TIME$lt, TIME$eq]**
functions returning **boolean** values; return TRUE if first TIME is [later than, earlier than, or the same time as] the second TIME, respectively.

C.1.1.2 Some notes on accuracy & precision

The precision and accuracy of internal mathematical computations within the TIME cluster is explicitly addressed; for critical users who wish to know the precision and accuracy which can be expected from a particular implementation, see the module specifications and also the include file "libss.h" (info. in section C.3).

Users (critical users in particular) also will need to consider that some information used in time computations is not independently calculable and is available only in tabular-lookup form; in the TIME module this is the case for computations of GAST (not GMST), UT1, and UTC (leap-seconds). (In fact, the information needed for precise UT1 computations isn't even physically available until well after its associated time has passed.) This means that there is a very real possibility that the table entries needed for a specific application may not be present in TIME's internal tables; for the TIME user, this aspect manifests itself in the following ways:

1. Procedures TIME$utc_parse and TIME$utc_unparse return an extra **boolean** value which is set TRUE if an extrapolation beyond the current known end of it's leap-second (ΔAT) table was made, flagging the application that there may have been leap seconds added to Bureau of Standards' UTC which aren't reflected in TIME's internal table.

2. Procedures TIME$ut1_parse, TIME$ut1_unparse, TIME$jdu2time,

TIME$time2jdu, TIME$gmst, TIME$gast return an extra **boolean** value which is set TRUE if any ΔT values flagged as being 'extrapolated' values (as are predicted values published in the *Astronomical Almanac*) in the internal ΔT table were used in the operation. This flags the application in case the critical user wants to check to see if a newer reference table may now be available which has actual measured values to replace the predicted ones currently being used by TIME.

3. Those same six procedures (TIME$utl_parse, TIME$utl_unparse, TIME$jdu2time, TIME$time2jdu, TIME$gmst, and TIME$gast) have been implemented so that they can make an approximation to the exact value required if a needed ΔT value is simply *missing* from the table, *if* the non-critical user explicitly indicates that such an approximation is adequate for the application. Each of these procedures has as one of its input arguments a **boolean** flag which, if set FALSE, will make an approximation ($\Delta T = 0^{s}.0$) and then return normally if the exact information is unavailable. Critical users will set the flag TRUE, which will raise the EXCEPTION 'critical_exception' instead if precise data is unavailable. For details of approximations made by a specific procedure under specific conditions, see that procedure's code specifications.

4. Procedure TIME$gast also uses a table lookup of 'equation of equinoxes' (available in *Astronomical Almanac* in the tabulated "Universal and Sidereal Times"); behavior on a failure of this lookup also is selected by the 'critical' **boolean** flag argument. If a needed value is missing from the table and the flag is FALSE, the approximation will be made that the value equals $0^{s}.0$ and TIME$gast thus will return with the value of GMST instead; if the critical flag is TRUE, a 'critical_exception' EXCEPTION will be raised.

→ If the critical user discovers that required table information isn't already available to TIME, but the user can find that information published somewhere (usually the most recent *Astronomical Almanac*), he/she may *add* it to the internal static data structure inside

TIME where such tabulated data is kept and thus prevent the 'critical_exception' from occurring[27]. Users taking this route are taking on the task of augmenting the appropriate table and re 'make'-ing **libss.a**; the static tables are kept in files "dattable.c" (ΔAT information for UTC), "dttable.c" (ΔT information for UT1), and "etable.c" (equation of equinoxes information for GAST). Each of these files contains instructions on how to extend its table. NOTE that each of these files are to be kept source-controlled under RCS (as are all the sources for **libss.a**) so that changes made don't accidentally slip by unrecorded.

C.1.1.3 Notes on algorithms & relations used

1. TIME$tdt_parse, TIME$tdt_unparse: Relations used are correct for TDT; no special handling is provided for differentiating between ET (dynamical time scale used 1960 through 1983) and TDT (dynamical time scale used beginning 1984). *Astronomical Almanac 1988*, page B5, includes

   > For most purposes, ET up to 1983 December 31 and TDT from 1984 January 1 can be regarded as a continuous time-scale.

2. TIME$utc_parse, TIME$utc_unparse: [AA 1988, pgs. B5 & K9]

   $TDT = TAI + 32^{s}.184$
   $TAI = UTC + \Delta AT$

   Offset between TAI and TDT fixed in 1972 to be $32^{s}.184$. Value for ΔAT needs ultimately to be looked up in the *Astronomical Almanac* and included in the TIME cluster's static tables (see source file "dattable.c").

3. TIME$ut1_parse, TIME$ut1_unparse: ([AA1988], pp. B5 & K9)

   $UT = TDT - \Delta T$

   Value for ΔT needs ultimately to be looked up in the *Astronomical Almanac* and included in the TIME cluster's static tables (see file "dttable.c"). ΔT is a continuously varying function of time; value used is interpolated to second-order from the contents of TIME's internal ΔT table.

---

[27] or alternately may decide that the approximation isn't such a bad idea after all

4. TIME$ep_parse, TIME$ep_unparse: [AA1988], page B4, has

$$Julian\,epoch = \mathrm{J}[2000.0 + \frac{JD - 2451545.0}{365.25}]$$

$$Besselian\,epoch = \mathrm{B}[1900.0 + \frac{JD - 2415020.31352}{365.242198781}]$$

5. TIME$jdt2time, TIME$time2jdt: nothing weird here; straightforward addition of offset.

6. TIME$jdu2time, TIME$time2jdu: For jdu2time: supplied UT1 Julian Date is converted to its associated UT1 'ordinary date' string, in typical TIME format. This resulting string is fed to TIME$ut1_parse, to create the resulting TIME object. Time2jdu implements the inverse operations.

7. TIME$gmst: computing GMST from UT1: Use the UT1 date to compute the Julian date, from which this relation (adopted by IAU in 1982 and given in [AA1988], page B6) may be used to compute the GST directly:

$$GMST\ at\ 0^h\ UT1 =$$
$$24110.54841 + 8640184.812866\,T_U + 0.093104\,T_U^2 - 6.2\times10^{-6}T_U^3$$
$$T_U = \frac{JD - 2451545.0}{36525}$$

(2451545.0 is the JD for 2000 Jan. 1, 12^h UT) To offset GMST at 0^h UT1 to desired time UT1, use (from [AAS1984], p. S16)

$$\frac{GMST}{UT1} = 1.00273\,79093\,50795 + 5.9006\times10^{-11}\cdot T_U - 5.9\times10^{-15}\cdot T_U^2$$

8. TIME$gast: ([AA1988], p. B6)

$$GAST = GMST + equation\,of\,equinoxes$$

To convert to GAST, information from the tabulated "equation of the equinoxes" is required. Values for "equation of equinoxes" need ultimately to be looked up in the *Astronomical Almanac* and included in the TIME cluster's static tables (see file "etable.c"). For 1988, these values are tabulated in [AA1988] on pp. B8-B15, and are to be "interpolated to the required time if full precision is required"; the TIME cluster interpolates them to second order.

C.1.2 THETA & PHI: Parse strings representing angles, and compute with angles

There are two types of angle objects implemented in **libss.a**, THETA and PHI. Although the range of angles each may represent is different (THETAs range from 0 to 2π while PHIs range from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$), the operations available for objects of either type are similar and will be treated here together. Objects of type THETA will be useful for representing right ascensions, hour angles, sidereal times, and longitudes, while PHI objects are well-suited to representing declinations and latitudes. (Note that a specific moment in time, as on a specific date, is *not* uniquely representable by only an angle; see section C.1.1 for information about the **TIME** type which is appropriate for such cases.)

C.1.2.1 operations available

<type>$create given a double-float argument in radians, returns a valid object of the angle type. This is how, if you have an angle represented in radians, you create valid PHIs and THETAs to use in those remaining operations which require them as arguments.

<type>$dms_parse
given a formatted string representing an angular value in degrees, arcmin, arcsec, return a valid object of the angle type. This is how, if you have an angle represented as a string (such as "42 22 17.6") you can directly create from it a valid THETA or PHI. The procedure is fairly forgiving with respect to details of string formatting; see procedure specifications for details. A similar procedure **THETA$hms_parse** is available for THETAs only; it accepts a string representing hours, minutes, and seconds.

<type>$dms_unparse
'unparse' refers to the inverse of the 'parse' operation: given a angle object, the procedure returns a formatted string representing the value in degrees, arcmin, arcsec. Useful for producing text output of values which is easily understandable to a human being. The caller specifies directly the string formatting; see the procedure specifications for details. Two additional unparse procedures are available for objects of type THETA: **THETA$ra_unparse** (for "right ascension") returns an hours, minutes, seconds string in the range 0^h to 24^h, and **THETA$ha_unparse** (for "hour angle") returns a similar string but in the range -12^h to $+12^h$.

<type>$add adds two angles to return a third.

<type>$sub subtracts a second angle from a first, returning the result as an angle.

<type>$cmult multiplies an angle by a given double-float value, returning the result as an angle object. (Note that this is *not* multiplication of two angles!)

trigonometric functions [<type>$sin, <type>$cos, <type>$tan]
each function returns the indicated double-float result.

The precision and accuracy of internal mathematical computations within these clusters is explicitly addressed; for critical users who wish to know the precision and accuracy which can be expected from a particular implementation, see the module specifications and also the include file "libss.h" (info. in section C.3).

C.2 Procedures

C.2.1 gd2gc: Compute geocentric latitude and distance

The gd2gc procedure takes as arguments

- topocentric latitude (a PHI object)
- altitude above mean sea level (double float, representing meters)
- undulation of the geoid at location if interest (double float, representing meters)

and returns two values:

- geocentric latitude (a PHI object)
- geocentric distance (double float, representing units of equatorial radius of Earth)

C.2.1.1 Algorithm

Symbol key:

ϕ = geodetic latitude of observer
h = height of observer above reference spheroid

ϕ' = geocentric latitude of observer
ρ' = geocentric distance of observer

$a_\oplus$ = equatorial radius of spheroid
f = flattening of reference spheroid

gd2gc implements these relations

$$C = \{\cos^2\phi + (1-f)^2\sin^2\phi\}^{(-1/2)}$$
$$S = (1-f)^2 C$$

$$\tan\phi' = \frac{(a_\oplus S + h)}{(a_\oplus C + h)}\tan\phi$$

$$\rho' = \frac{(a_\oplus C + h)\cos\phi}{a_\oplus \cos\phi'} = \frac{(a_\oplus S + h)\sin\phi}{a_\oplus \sin\phi'}$$

('C' form used for $\phi' < 45°$;
'S' form used for $\phi' \geq 45°$.)

which are derived from the following relations given in [AA1988], page K11. They involve no approximations, so they'll work for large *Alt* as well as small ones:

$$a_\oplus \rho \cdot \cos\phi' \cdot \cos\lambda = (a_\oplus C + h) \cdot \cos\phi \cdot \cos\lambda$$
$$a_\oplus \rho \cdot \cos\phi' \cdot \sin\lambda = (a_\oplus C + h) \cdot \cos\phi \cdot \sin\lambda$$
$$a_\oplus \rho \cdot \sin\phi' = (a_\oplus S + h) \cdot \sin\phi$$

Noted that strictly, h refers to a height above the reference spheroid and differs from the height above mean sea level (i.e. the geoid) by the 'undulation of the geoid' at that point; this value varies from about -105m to +80m. The value of undulation for a particular location on the Earth can be estimated from geological maps designed for just such purposes. (e.g., a photocopy of one such map is included with this document as Figure C-1, from [KING83], page 151.)

Fig 68 *Contour map of the height of the geoid in metres relative to a spheroid of flattening 1/298.257, as given by the Goddard Earth Model 10B (GEM 10B). The depressed areas are shaded and the elevated areas white. The contour lines are at 10 m intervals.*

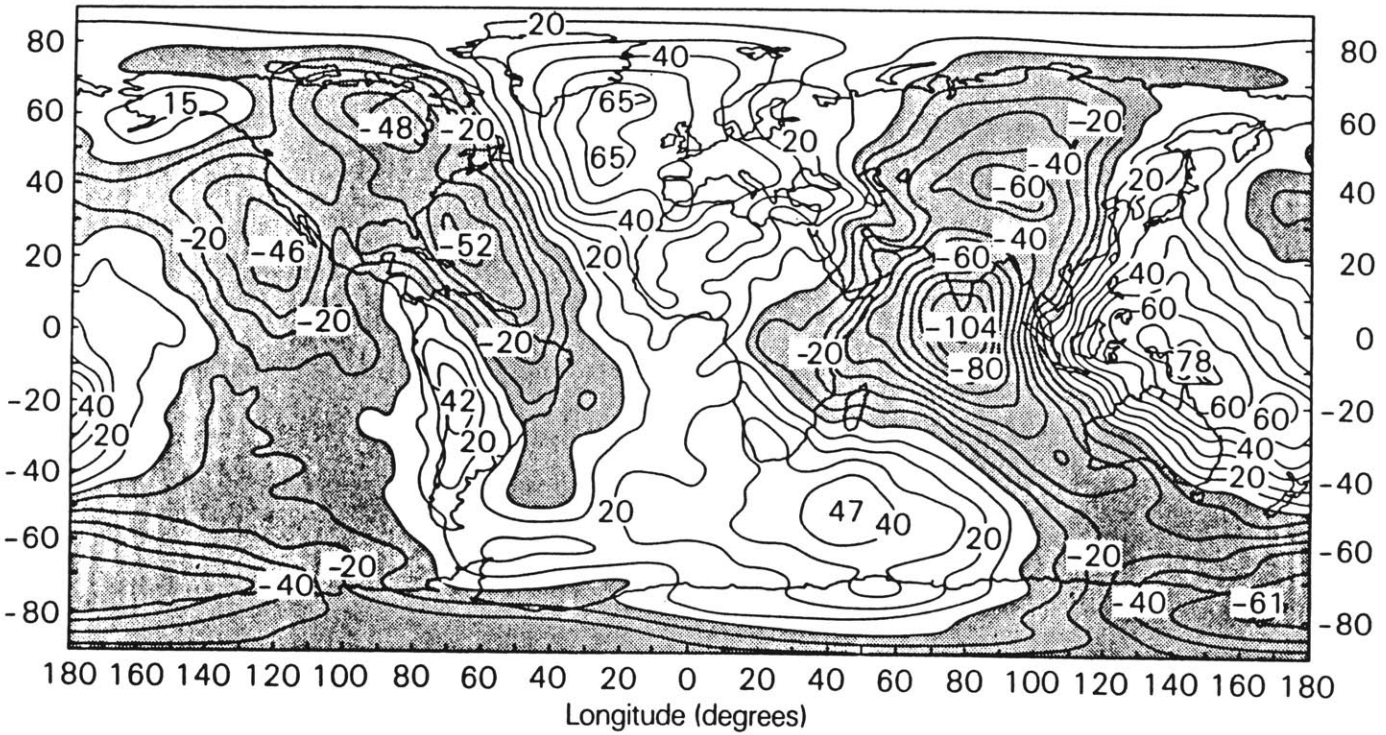

Figure C-1: Contour map of undulation of the geoid

C.2.2 hour_angle: Compute hour angle of object at a given time

The **hour_angle** procedure takes as arguments

- east longitude of observer (a THETA object)
- right ascension of object (a THETA object)
- target time (a TIME object)
- 'critical' flag (see below)

and returns two values

- a THETA object corresponding to the hour angle
- a boolean flag set TRUE only if 'extrapolated' values from the ΔT table (in TIME) were needed for a computation of UT1.

The 'critical' flag: Computation of hour angle involves Greenwich Sidereal Time; specifically, greatest accuracy requires use of Greenwich Apparent Sidereal Time (GAST), which may vary a few seconds from Greenwich Mean Sidereal Time (GMST). GMST is directly computable from relations, but GAST requires a table lookup of 'equation of equinoxes' information (see **TIME** cluster, TIME$gast and TIME$gmst) and thus may not be available if the lookup table (in **TIME**, see included source file "etable.c") doesn't currently hold the date of interest. On the other hand, the accuracy obtained by substituting GMST instead may be completely adequate for all but the most critical of applications. For this reason, an additional (boolean) argument serves as a switch with which the user may distinguish between critical and non-critical applications. For a non-critical application the switch is set FALSE, in which case if a GAST is unavailable its corresponding GMST will be substituted instead. The switch is set TRUE for a critical application, so that a 'critical_exception' will be raised if it is not possible to compute the precise GAST, thus informing the user that the "etable.c" information in **TIME** needs to be extended (see Section C.1.1.2).

C.2.2.1 Algorithm

$$HA_{object} := GAST - \alpha_{object} + \lambda_{Obs}$$

where

$GAST$ = Greenwich Apparent Sidereal Time at time t of interest
α_{object} = right ascension of object
λ_{Obs} = east longitude of observer

C.2.3 precess: Precess coordinates from one given epoch to another epoch

The **precess** procedure takes the following arguments

- initial epoch (a TIME)
- final epoch (a TIME)
- right ascension in initial epoch (a THETA)
- declination in initial epoch (a PHI)

and returns two values

- right ascension in final epoch (a THETA)
- declination in final epoch (a PHI)

C.2.3.1 Algorithm

This module is a straightforward implementation of the "Reduction for precession - *rigorous* formulae" given in [AA1988], page B18. Note that the values of constants ζ_A, z_A, and θ_A are given for reduction to and from the standard epoch t_0 = J2000.0, so reduction from arbitrary epoch t_1 to arbitrary epoch t_2 involves an intermediate computation of the position at t_0.

C.2.4 search & interpolate: Tabular search and interpolation

Interpolation of a value from a table of available values involves two steps: first, the **search** procedure finds the position of interest in the table. Then in the second step, values from the table position thus identified are used by the **<type>_interpolate** procedure to

actually compute the interpolated value. <type> is the name of the type of the result; interpolation procedures have been implemented in **libss.a** for types float, double, THETA, and PHI.

Note that by separating the search and interpolate operations, repeated interpolations to the same time of different data sets sharing identical times of tabulation (e.g., interpolating the RA, Dec, and distance of a body at a given time from an ephemeris) can be accomplished with only *one* search operation followed by the three calls to interpolate.

The **search** procedure takes the following arguments

- (address of) array of TIME values to search in
- the array indices of TIME array between which to search
- target TIME for which interpolation will be computed

and returns the array index of the element of the TIME array which is the earliest time available which is later than the target time.

The **<type>_interpolate** procedure takes the following arguments

- (address of) array of TIME values corresponding to tabulation times
- indices of bounds of portion of TIME array of interest
- (address of) array of data values (of type <type>) to use in interpolation
- array index returned from **search**
- target TIME to which to interpolate the value of interest
- order of interpolation to use

and returns a value of type <type> which is the interpolated value.

All the **<type>_interpolate** procedures' C sources are in file "interpolate.c".

C.2.4.1 Algorithms

search: Straightforward iterative implementation of binary search

<type>_interpolate: Computation depends on order of interpolation indicated by input

argument. Note that in the following formulae, p is the fraction of the tabulation interval for which the interpolation is to be computed, and the f_n refer to tabulated values of the "function" being interpolated (e.g., f_0 refers to the tabulated value at the index returned by the **search** procedure, f_{-1} is the preceding tabulated value, f_1 is the following tabulated value).

- order 0: result is the *nearest tabulated value*.
- order 1: result is linear interpolation

  $p \cdot f_0 + (1-p) \cdot f_{-1}$
- order 2: result is Lagrangian quadratic interpolation (3-point)

  $$\frac{p(p-1)}{2} \cdot f_{-2} + (1-p^2) \cdot f_{-1} + \frac{p(p+1)}{2} \cdot f_0$$
- order 3: result is Lagrangian cubic interpolation (4-point)

  $$\frac{-p(p-1)(p-2)}{6} \cdot f_{-2} + \frac{(p+1)(p-1)(p-2)}{2} \cdot f_{-1} + \frac{-p(p+1)(p-2)}{2} \cdot f_0 + \frac{p(p-1)(p+1)}{6} \cdot f_1$$

C.3 Miscellany

C.3.1 failure: procedure for aborting an application

Can be used when detecting errors deigned as 'fatal' after which the program should not be permitted to continue; it provides an easy way to print an informational message and force a core file to be dumped. The **failure** procedure takes a string argument; application should *#include* the file **libss.h**. Calling **failure** causes the supplied failure message string to be printed at the primary output stream, and then aborts the program, explicitly causing a core dump in the process.

C.3.2 Include files

The following include files are associated with **libss.a**. The first two files, **exceptions.h** and **libss.h**, are among those files necessary for compiling application programs which use procedures in the library (see the specifications of the particular procedures you're using); the others are used internally when compiling the library itself but contain information possibly useful for applications as well.

C.3.2.1 exceptions.h

To be used with any procedure or cluster which returns exceptions (which turns out to be almost all of them in **libss.a**). Herein is the definition of the **EXCEPTION** enumerated type, listing all possible values for a variable of that type.

C.3.2.2 libss.h

To be included by applications which use any of the procedures or data types in **libss.a**, it includes type definitions of C function return values, and *#defines* that are used to ensure uniqueness in 7 characters of externally callable procedure names.

Of particular note here is that **libss.h** is also the location of the d information determining the limits of accuracy and precision of objects of type **TIME**, **THETA**, and **PHI**. The constants explicitly define the numeric precision (given in number of decimal digits after the decimal point) to which the implementation is "guaranteed" to give correct results of a computation's result when expressed in a particular unit of measurement. Rounding, tolerances, and string lengths are handled explicitly; see the file libss.h itself for detailed explanation.

C.3.2.3 lexical.h

(file *#include*d by libss.h) Useful lexical substitutions used in the C source code for procedures in **libss.a**: **signal, boolean, TRUE, FALSE, AND, OR, NOT.**

C.3.2.4 constants.h

Useful constants (mathematical and physical); having this include file available facilitates those applications where the value used for a constant in the application program MUST be identical that used in the implementation of the procedures in **libss.a**. Includes definitions for π, c (speed of light), number of arcsec per radian, radius and flattening of Earth's reference spheroid, astronomical unit, constants for diurnal parallax and aberration.

C.3.2.5 tiny.h

Machine-dependent constants: includes definitions for smallest nonzero representable double-float value ("TINY"; kin of "HUGE" in math.h) and exponents of TINY and HUGE.

Appendix D

A Style of C for 'Programming in the Large'

(preliminary draft)

Stephen M. Slivan
MIT, Department of Earth, Atmospheric, and Planetary Sciences,
Cambridge, MA 02139
(slivan@lcs.mit.edu)

The book [LISK86] presents a programming method based on abstractions. The applicability of this approach transcends the particular language the authors use to illustrate it (CLU); this document includes information on one particular incarnation of that methodology using the less sophisticated C language:

- The information here is intended to provide a sort of 'bridge' between the CLU examples in [LISK86] and software written in C by highlighting differences between CLU examples in the book and particular attempts to apply their underlying concepts using the C language.
- The content of this preliminary draft is aimed primarily at those programmers writing high-level applications in C, who are using library or other code already designed and implemented using this style (such as the **libss.a** library described in Appendix C).
- The information included will also be helpful for one who finds it necessary to slog through and understand the source files of software written in this style.
- This document does not attempt a detailed conceptual treatment of the value of, or rationale behind, use of abstraction and specification in large software development. (For this, see [LISK86].)

- This document does not attempt to justify the particular lower-level design and style decisions made in applying the larger concepts to C code.

D.1 Overview of Concepts

D.1.1 Data Abstraction

A data abstraction consists of a set of abstract objects along with a set of operations which characterize those objects. The behavior of the objects and operations is specified at a high-level, so that they don't depend on details of their underlying implementation. The key idea is that the applications programs use these objects based strictly on these high-level specifications.

[Example: The C language implements an abstract object called a *double*, which represents a floating-point value. The underlying representation is some number of bits of binary mantissa, exponent, sign, and so forth. Operations to create, add, subtract, multiply, and perform all primitive operations on doubles are provided with the data type to manipulate the mantissa and exponent bits, and it is these operations that are called by C programs; thus higher-level programs need not be (indeed, *should* not be) dependent on the particular bit representation used by a given machine.]

The concept of data abstraction simply enables hiding of internal detail and complexity that appears at higher levels, affording user-defined types the status of abstract objects.

For more detailed information about this idea, and about the value of abstraction in general, I refer you to [LISK86, Chapters 1 & 4]. The CLU language used there has features built directly into the language which support these ideas; what I've done here is try to incorporate the most useful concepts into "C" code.

The CLU language used in [LISK86] provides explicit support for implementing data abstractions as *clusters*, complete with compiler-enforced type-safety. No such support is available in C, so clusters can be implemented only as a type definition and set of

functions, with type-safety maintained only by programming conventions honored by mutual consent between cluster implementor and cluster user.

D.1.2 Procedure Abstraction

A procedure abstraction abstracts a single event or task, rather than a data object.

Nearly all high-level programming languages support some method of implementing procedural abstractions; and the C language provides a rudimentary facility as well with its *function*.

D.2 Aspects of Application in C

D.2.0.1 header files

Header files are provided to be *#included* by compilation units of application software using cluster and procedure modules. The name(s) of the header file(s) needed are listed in the module specifications under "Include files for applications"; files thusly listed must be *#included* for your application to compile and run properly. Such a header typically contains type declarations of C function return values for the externally-callable procedure(s) in the module; this simplifies the task of the application programmer having to remember to declare them and get them right on their own (a frequent source of errors in C code). These header files may also contain *#defines* used to 'alias' identifier names as seen by the C compiler, in order that all identifiers are kept unique within the first seven characters (for GC compatibility). This is used particularly in renaming cluster operations, where the first part of an operation name is the type name (see Section D.2.2.2), and allows the applications programmer to use the extended procedure names exactly as they appear in the documentation.

D.2.1 Implementation of Procedure Abstractions

A procedure is implemented using the C *function* facility.

D.2.1.1 module specifications

A procedure module implements a procedural abstraction. A general description of what a given procedure does is included in a corresponding "Application User Guide" paper document (e.g. Appendix C), which includes algorithm descriptions, in order to help the applications programmer select any procedures which would be useful for an application program. Once identified, the applications programmer will find the detailed information necessary for integrating modules in those modules' code specifications, which are available as 'man' pages derived from the comments at the beginning of the modules' C source code files.

The module specifications include

- list of header files the application program must include for proper operation of the procedure(s) used from the module
- Notes revealing any special handling or weird aspects of use of the module; particularly those limitations which are implementation-dependent.

Each procedure within a module has its own procedure specifications; they include

- SIGNALS: list of signals which might be raised (see Section D.2.3)
- REQUIRES: any specific requirements which must be satisfied if the procedure is to work properly
- MODIFIES: list of arguments which are modified (the returned values; see Section D.2.1.2)
- EFFECT: detailed description of the effect of the operation, including conditions under which the various signals will be raised.
- IMPLEMENTATION NOTES: other special information potentially helpful for the user or maintainer of the procedure.

D.2.1.2 passing of procedure results

As the single C *function* return value is (usually) used to raise exceptions (see Section D.2.3), the actual results from a procedure are (usually) written out to variables through pointers (addresses) passed in the procedure's argument list explicitly for this purpose. By convention in such argument lists return values have been collected to be at the end of the argument list and have the name 'rtn*x*' with $x = 1, 2, \ldots$ up to the number of return values.

Note also that by convention, in no case is any single argument in the argument list used for *both* input to *and* output from the procedure. The MODIFIES clause in the module specifications is intended to provide explicit information about which arguments are and which are not modified by the procedure.

D.2.2 Implementation of Data Abstractions

A 'cluster' module implements a (presumably useful) higher-level data type. By using operations in such a module, you are provided with means by which to create objects of the type, as well as to perform certain operations on objects of that type. The idea is that you can create objects and use these type-operations without having to waste time and effort worrying about the ugly details of how these types have actually been implemented underneath, which in turn allows you to concentrate on your higher-level application program. In fact, the ONLY operations available on an object of a given type are those supplied in the module, INCLUDING those operations which create a valid object of the type in the first place.

(It's very possible that someone will come up with an application to use an existing data type for which one or more primitive operations required on the type are not yet available; in that case it would be necessary to add operations to that existing module.)

D.2.2.1 module specifications

(See also [LISK86, Chap. 8]) In the module specifications for a cluster you'll find information you'll need to use the data type in your program design. Included are

- brief description of the data type & some sample uses for it
- list of operations (procedures) available with the data type
- list of header files to *#include* in application code using objects of the type
- notes giving any additional information relevant to use of the type

Following this general information are specific procedure specifications for each operation. Included for each procedure are

- SIGNALS: list of signals which might be raised (see Section D.2.3)
- REQUIRES: any specific requirements which must be satisfied if the procedure is to work properly
- MODIFIES: list of arguments which are modified (the returned values; see "PASSING OF PROCEDURE RESULTS", below)
- EFFECT: detailed description of the effect of the operation, including conditions under which the various signals will be raised.
- IMPLEMENTATION NOTES: other special information potentially helpful for the user or maintainer of the procedure.

****** IMPORTANT POINT ABOUT DATA TYPES IMPLEMENTED USING THIS 'C' VARIANT OF 'CLUSTERS': ******

NOTE that contained in the specifications for *all* operations of a user-defined type there is an *implicit* REQUIRES clause, to the effect that any data object supplied where a specific type is required must in fact be a VALID data object, created using one of the procedures supplied which creates or returns one (e.g. TYPE$create, TYPE$parse, ...) and *not* a 'fake' one made from some other type. The unfortunate lack of type-safety for user-defined types in the C language makes it CRUCIAL that this requirement be explicitly adhered to by the application programmer, since the compiler won't catch you if you start taking dangerous liberties. (The C implementation of clusters: 'data abstraction by mutual consent'.)

Proper Behavior Of Operations In C Cluster Modules As Per Their Specifications Can NOT Be Expected If Fake Objects Are Used As Arguments.

D.2.2.2 cluster procedure names

Cluster operation names are of the form

<abstraction name>$<procedure name>

which resembles CLU syntax. Note that this syntax *immediately* identifies the data type (and thus the source code file) in which the procedure is coded by use of the abstraction name, saving time in case it becomes necessary to locate the source code.

D.2.3 Exceptions

(See also [LISK86, Chap. 5]) An exception is an 'exceptional' return condition from a procedure; it does not necessarily indicate an error condition, though flagging errors is one typical use for them. The EXCEPTION type is defined in the header file "exceptions.h" (a file name you'll find in the list of "Include files for application" for any procedure module or cluster module which contains procedures that signal exceptions). EXCEPTION is simply an enumerated type of the union set of exceptions that are raised by the various procedures in the modules.

In general, a procedure can terminate its invocation either by raising an exception, or by completing normally and returning one or more results. Since the C language has the limitation of only returning one value directly from a procedure, the convention has been adopted to typically use this return value to signal (return) an EXCEPTION value (which will be exception value "ok" for a normal return).[28]

---

[28]In a few cases, where a procedure only returns one value, and it is likely to be used in-line, AND the abstract procedure implemented doesn't signal any exceptions per se, you'll find procedures written as 'functions', which return an actual result instead; e.g. THETA$sin(x) and TIME$eq(t1,t2). Such atypical cases are explicitly accounted for in the type-declaration header file associated with the module.

One may find the C **switch** statement to be a convenient construct to use for calling these procedures and detecting and handling the various possible exceptions; of course setting a return-code variable of type **EXCEPTION** and then using an '**if...elseif...else**' construct will work as well. It is recommended that the 'default' case always be included to trap any 'unhandled exceptions' which may unexpectedly arise, by terminating the application with a call to the **failure** procedure (Section D.2.3.2). (See also [LISK86, Sec. 5.2.4])

D.2.3.1 sample usage in "C" program

One example of a call to a cluster operation that creates a **PHI** object as defined in Appendix C:

```
#include "libss.h"

main()
  {PHI f;
   char phi_string[DmsStrlen];

   /* ...code which gets a string from somewhere representing an
      angular value to be made into a PHI... */
   strcpy(phi_string,"23 26 39.26");

   switch(PHI$dms_parse(phi_string,&f))
     {case ok:
         break;
      case bad_format:
         printf("Input string is of bad format");
         break;
      case bounds:
         printf("Input string field out of bounds");
         break;
      default:
         failure("main: unhandled exception from PHI$dms_unparse");
         break;}

   /* code down here to use [f] for whatever... */}
```

D.2.3.2 failure

When 'Failure' is indicated in a procedure specification, it means that the condition is treated as a non-recoverable error. A special aborting procedure **failure** is called, which prints a message at the terminal briefly identifying the cause of the failure, and then aborts the program and explicitly causes core to be dumped. The **failure** procedure takes a single string pointer argument and is available for use by your application as one of the procedures in **libss.a** (See Appendix C).

References

[AAS1984] U.S. Naval Observatory.
Supplement to The Astronomical Almanac 1984.
U.S. Government Printing Office, Washington, D.C., 1983.

[AA1988] U.S. Naval Observatory.
The Astronomical Almanac for the Year 1988.
U.S. Government Printing Office, Washington, D.C., 1987.

[AA1989] U.S. Naval Observatory.
The Astronomical Almanac for the Year 1989.
U.S. Government Printing Office, Washington, D.C., 1988.

[BAUM53] Baum, W.A. and A.D. Code.
A Photometric Observation of the Occultation of σ Arietis by Jupiter.
The Astronomical Journal 58(1208):108-112, May, 1953.

[BESS76] Bessell, M. S.
UBVRI Photometry with a Ga-As Photomultiplier.
Publications of the Astronomical Society of the Pacific 88:557-560, 1976.

[BOSH86] Bosh, A.S., J.L. Elliot, S.E. Kruse, R.L. Baron, E.W. Dunham and L.M. French.
Signal-to-noise Ratios for Possible Stellar Occultations by Pluto.
Icarus 66(3):556-560, June, 1986.

[DUNH85] Dunham, E.W., R.L. Baron, J.L. Elliot, J.V. Vallerga, J.P. Doty and G.R. Ricker.
A High-speed, Dual-CCD Imaging Photometer.
Publications of the Astronomical Society of the Pacific 97(598):1196-1204, December, 1985.

[ELLI78] Elliot, J.L., E. Dunham, L.H. Wasserman, R.L. Millis, and J. Churms.
The Radii of Uranian Rings α, β, γ, δ, ε, η, 4, 5, and 6, from their Occultations of SAO 158687.
The Astronomical Journal 83(8):980-992, August, 1978.

[ELLI88] Elliot, J.L., E.W. Dunham, R.L. Baron, A.W. Watts, S.P. Kruse, W.R. Rose, and C.M. Gillespie.
Image Quality on the Kuiper Airborne Observatory, I: Results of the First Flight Series.
in preparation , 1988.

[ELLI89] Elliot, J.L., E.W. Dunham, A.S. Bosh, S.M. Slivan, L.A. Young, L.H. Wasserman, and R.L. Millis.
Pluto's Atmosphere.
Icarus 77(1):148-170, January, 1989.

[ESHL89] Eshleman, Von R.
Pluto's Atmosphere: Models Based on Refraction, Inversion, and Vapor-Pressure Equilibrium.
preprint , 1989.

[GOLD63] Goldsmith, D.W.
Differential Refraction in Planetary Atmospheres with Linear Scale Height Gradients.
Icarus 2(4):341-349, November, 1963.

[HILD76] Hildebrand, F.B.
Advanced Calculus for Applications, Second Edition.
Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1976.

[HUBB88] Hubbard, W.B., D.M. Hunten, S.W. Dieters, K.M. Hill, and R.D. Watson.
Occultation Evidence for an Atmosphere on Pluto.
Nature 336:452-454, December, 1988.

[HUBB89] Hubbard, W.B., R.V. Yelle, and J.I. Lunine.
Nonisothermal Pluto Atmosphere Models.
submitted to *Icarus* , 1989.
(preprint).

[KERN78] Kernighan, Brian W. and Dennis M. Ritchie.
The C Programming Language.
Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1978.

[KING83] King-Hele, Desmond.
Observing Earth Satellites.
Macmillan London, Ltd., London, 1983.

[KLEP73] Kleppner, Daniel and Robert J. Kolenkow.
An Introduction to Mechanics.
McGraw-Hill Book Company, New York, NY, 1973.

[LISK86] Liskov, Barbara and John Guttag.
Abstraction and Specification in Program Development.
MIT Press, Cambridge, Massachusetts, 1986.

[MILL79] Millis, R.L. and J.L. Elliot.
Direct Detection of Asteroid Diameters from Occultation Observations.
In T. Gehrels (editor), *Asteroids*, pages 98-118. The University of Arizona Press, Tuscon, 1979.

[MINK85] Mink, D.J. and A. Klemola.
Predicted Occultations by Uranus, Neptune, and Pluto: 1985-1990.
The Astronomical Journal 90(9):1894-1899, September, 1985.

[PRES86] Press, William H., Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling.
Numerical Recipes: The Art of Scientific Computing.
Cambridge University Press, New York, NY, 1986.

[SHEP90] Shepard, Timothy J.
TCP Packet Trace Analysis.
Master's thesis, Massachusetts Institute of Technology, 1990.
(in preparation).

[SLIV86] Slivan, Stephen M.
A Metaphor-based User Interface for a Patient Data Management System (FLYSWATTER/150).
Master's thesis, Massachusetts Institute of Technology, May, 1986.

[SMAR77] Smart, W.M.
Textbook on Spherical Astronomy, Sixth Edition.
Cambridge University Press, New York, New York, 1977.

[THOL88] Tholen, D.J. and M.W. Buie.
Circumstances for Pluto-Charon mutual events in 1989.
The Astronomical Journal(?) volume?(number?):pages?, month?, year?
(in press).

[TOON80] Toon, Owen B., R.P. Turco, and James B. Pollack.
A Physical Model of Titan's Clouds.
Icarus 43(3):260-282, September, 1980.

[WASS88] Wasserman, L.H., R.L. Millis, O.G. Franz, A.R. Klemola, and C.C. Dahn.
.
submitted to *Bulletin of the American Astronomical Society* , 1988.

[WATS88] Watson, R.D., K.M. Hill, and S.W. Dieters.
IAU Circular No. 4612.
1988

[YELL89] Yelle, Roger V. and Jonathan I. Lunine.
Evidence for a Molecule Heavier than Methane in the Pluto Atmosphere.
submitted to *Nature* , 1989.
(preprint).