Solving the Alhazen-Ptolemy Problem, Compiling Cassini-VIMS Titan Data, and Modernizing Orbital Integration

A Thesis
Presented in Partial Fulfillment of the Requirements for the
Degree of Master of Science
with a
Major in Physics
in the
College of Graduate Studies
University of Idaho
by
William Miller

Major Professor: Jason Barnes, Ph.D.
Committee Members: Matthew Hedman, Ph.D.; F. Marty Ytreberg, Ph.D.
Department Administrator: John Hiller, Ph.D.

December 2022

Abstract

I present the first complete analytical solution to finding the specular point on a spherical surface, known as the Alhazen-Ptolemy problem, for the cases where either the observer of the light source may be approximated as infinitely distant, and where they must both be treated as finitely distant. I also present `Vulcan`, a new orbital integration code which offers the benefits of modernization to existing orbital integration algorithms: efficient parallelization, advanced memory management, and concurrent output. Lastly, I present my work compiling the Titan data from Cassini's Visible Infrared Mapping Spectrometer (VIMS) instrument to create a set of composite maps which summarize the dataset for future work.

Table of Contents

List of Figures

Statement of Contribution

The content of chapter 2 was published in the Planetary Science Journal under the title *Solving the Alhazen-Ptolemy Problem: Determining Specular Points on Spherical Surfaces for Radiative Transfer of Titan's Seas* under the authorships of myself, Jason W. Barnes, and Shannon M. MacKenzie. It was Jason's idea to employ the Euler rotation, and it was Shannon who determined that we would need to solve the 'both-finite case' for use in the `SRTC++` (Spherical Radiative Transfer in C++) model. I performed the formulation, analysis, and symbolic solving for both the 'one-finite' and 'both-finite' cases. I also implemented those solutions as standalone routines in both Python and C++, as well as integrated routines in `SRTC++`.

The as-yet unpublished work of chapter 3 is authored by myself and Jason W. Barnes. The `Jcube` codebase and the `cylindermap` utility are the work of Jason, I contributed modifications to `cylindermap` and `Jcube` including memory optimizations and parallelization resulting in a fifty-fold decrease in computation time, and implemented the routines to output files compatible with the file standard of the Planetary Data System's Version 4. I also created the composite maps for the first 22 flybys, the remainder were created by undergraduate students who are not authored because it is the changes I made to the processing pipeline and their implications that are the matter of the chapter, not the maps themselves. Furthermore, I created every demonstrative map included herein.

The work of chapter 4 is authored solely by me.

Chapter 1 Introduction

1.1 Solving the Alhazen-Ptolemy Problem

In 150 C.E. Ptolemy formulated a deceptively simple problem: given a light source, a spherical reflector, and an observer, where on the surface of the sphere will the observer see the light, i.e. where is the the specular point? Known as the Alhazen-Ptolemy problem thanks to the geometric solution presented by the 11th-century mathematician Alhazen, this problem lacked a complete analytical solution for nearly the next two thousand years. The earliest algebraic solution was found by Jack Elkin in 1965, but as with most subsequent solutions, it was incomplete, relying on numerical or inspective methods to complete the calculation. Jason Barnes, Shannon MacKenzie, and I created a formulation which allows a complete solution. I have solved the Alhazen-Ptolemy problem under this formulation for two cases: the case where either the source, observer, or both are so distant they can be approximated as infinitely distant, and the case where both must be treated as finitely distant. This formulation, my solutions, and their implications are presented in Chapter 2.

1.2 Modernizing Orbital Integration

When it comes to orbital integration, the problem is time. There is a direct relationship between the duration of an integration and the time it takes to complete, limiting either the size of or the sampling granularity for the parameter space being investigated. Broadly speaking, the solution to this problem lies in the improvement of the algorithm employed in integrating, but even the best algorithms can benefit significantly from optimization, particularly when it comes to algorithms whose implementation was optimized under the constraints of computing architecture which no longer exists, such as `MERCURY`'s hybrid symplectic integrator. The 'low-hanging fruit' of beating orbital integration's inherent problem of time is the modernization of the implementations of existing algorithms. I have accomplished this modernization with `Vulcan`. `Vulcan` forms a codebase in which any integration scheme can be easily added while receiving the benefits of efficient parallelization, modern memory management, and concurrent file output. The construction of `Vulcan` is intended to support any algorithm – even future algorithms – in the interest of solving the optimization problem once-and-for-all, or at least until quantum computing renders it obsolete. I discuss `Vulcan` in detail in Chapter 4.

1.3 Compiling Cassini-VIMS Titan Data

The observations of Titan conducted by the flagship Cassini mission's Visible Infrared Mapping Spectrometer (VIMS) during more than a hundred flybys over the course of the thirteen-year mission have proven invaluable to furthering our understanding of the only other body in our solar system with active hydrology (in the form of methane). This enormous dataset is difficult to work with, requiring substantial processing to synthesize useful data products for scientific investigation. And although there have already been many useful analyses published it is important to lower these barriers to enable future work, especially as many contributors to the mission which began 25 years ago have moved on or retired. I have integrated the creation of data products compatible with Version 4 of the Planetary Data System standard into Jason Barnes' `Jcube` code base. With my additions we have summarized the observations from the prime mission and most of the extended mission into PDS4 data products. The creation of these maps is the matter of Chapter 3.

Chapter 2 Solving the Alhazen-Ptolemy Problem

Miller, W., Barnes J., MacKenzie S. 2021, Planetary Science Journal, 2, 63

Given a light source, a spherical reflector, and an observer, where on the surface of the sphere will the light be directly reflected to the observer, i.e. where is the the specular point? This is known as the Alhazen-Ptolemy problem, and finding this specular point for spherical reflectors is useful in applications ranging from computer rendering to atmospheric modeling to GPS communications. Existing solutions rely upon finding the roots of a quartic equation and evaluating numerically which root provides the real specular point. We offer a formulation, and two solutions thereof, for which the correct root is predeterminable, thereby allowing the construction of the fully analytical solutions we present. Being faster to compute, our solutions should prove useful in cases which require repeated calculation of the specular point, such as Monte-Carlo radiative transfer, including reflections off of Titan's hydrocarbon seas.

2.1 Introduction

The specular point is the specific location on a surface where the outgoing vector of a mirror reflection coincides exactly with the vector to an observer. At such a point, the light from a point source is seen by the observer and, if the source has spatial extent, an image is produced in the reflection. Since superposition allows a source with spatial extent to be divided into point sources – each with a single specular point which in aggregate comprise the specular region – the same methods for finding a specular

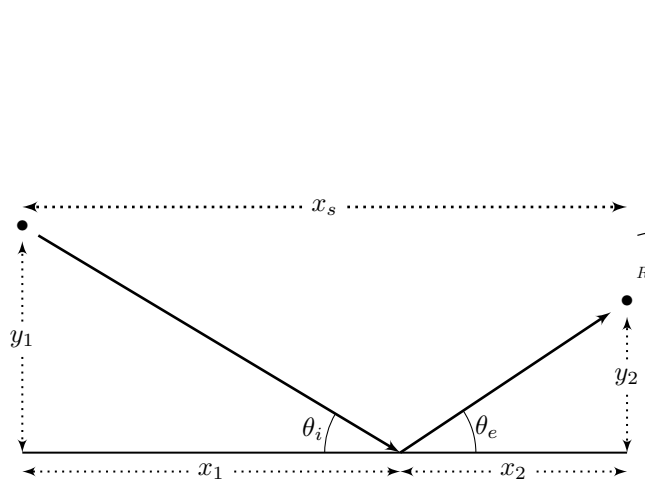

Figure 2.1: Specular point for planar surface.

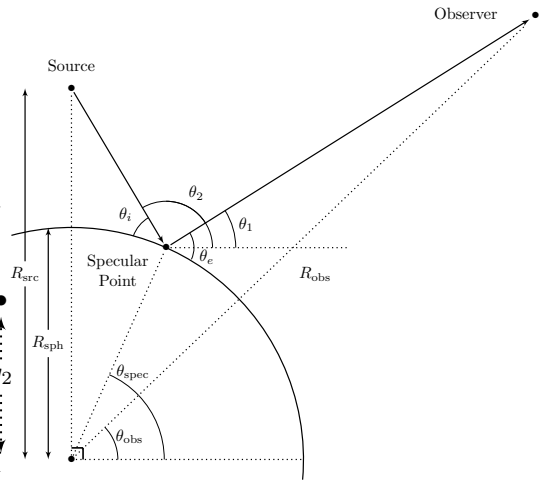

Figure 2.2: Specular point on a spherical surface.

point can be repeatedly applied to determine the region of specular reflection corresponding to a source with spatial extent.

2.1.1 Applications

Perhaps the most notable use for finding the specular point for an arbitrary source-observer configuration is in Global Positioning Systems. Unfortunately, the tolerance of GPS requires the Earth be approximated as an ellipsoid instead of a sphere, rendering analytical solution of the spherical-surface case less useful (Southwell & Dempster, 2018; Prakash et al., 1994). However, our formulation can be extended to elliptical geometries more easily than previous approaches, arguing positively of its usefulness in this application.

Unlike the Earth, Saturn's moon Titan – which harbors liquid hydrocarbons that allowed the Cassini mission to record numerous specular reflections (Stephan et al., 2010; Barnes et al., 2014) – can be well-approximated as a sphere. The `SRTC++` model (Barnes et al., 2018) does exactly that and finding a faster method for computing the specular reflections due to scattering off of Titan's ubiquitous atmospheric haze aerosols in `SRTC++` is the direct motivation for this work. There may also be utility for removing 'sun-glint' from imaging data by employing a method similar to those in Kay et al. (2009) but with the higher precision enabled by our analytical solutions.

Another notable application is the rendering of computer graphics with raytracing (Inakage, 1986), a case where computation speed is critical. Particularly when rendering a reflected image (instead of a simple point source) because the rendering fidelity is much more sensitive to the accuracy of the specular point calculation. The increased interest in ray-traced rendering, particularly with regard to the video game and movie industries, serves only to increase the usefulness of a faster method for handling even partially reflective surfaces.

2.1.2 Planar Surface

For a planar surface the specular point (Figure 2.1) can be found trivially. In the coordinate system centered on the specular point with the x-axis parallel to the surface, the positions of the source and observer must satisfy

$$\theta_i \stackrel{!}{=} \theta_e \tag{2.1}$$

where θ_i is incidence angle and θ_e is emission angle. In the configuration shown in Figure 2.1 they are defined as

$$\theta_i = \tan^{-1}\left(\frac{y_1}{x_1}\right), \quad \text{for} \quad 0 \leq \theta_i \leq \frac{\pi}{2},$$

$$\theta_e = \tan^{-1}\left(\frac{y_2}{x_2}\right), \quad \text{for} \quad 0 \leq \theta_e \leq \frac{\pi}{2}, \quad \text{and}$$

$$x_s = x_1 + x_2.$$

where x_s is the separation between the source and observer in the x-direction. Then the position of the source and observer relative to the specular point can be found by combining the above and rearranging,

$$x_2 = \frac{x_s y_2}{y_2 + y_1},$$
$$x_1 = \frac{x_s y_1}{y_1 + y_2}.$$

2.1.3 Alhazen-Ptolemy Problem

The case of a spherical surface (Figure 2.2) is far more complicated. First formulated in 150 CE by Ptolemy (Weisstein, 2002), it is known as the Alhazen-Ptolemy problem because the first method of solving it (a geometric approach using conic sections) was provided – and proven – by *Ibn al-Haythum* in the 11^{th} century (Katz, 1995). Elkin (1965) provided the first algebraic solution and similar formulations by Riede (1989), Smith (1992), and Waldvogel (1992) confirmed Elkin's solution. Neumann (1998) proved that ruler-and-compass construction of a general solution, a method which permits only circular arcs and straight lines, is impossible. That is, given only an unmarked and idealized compass and straight edge there is no way to determine the specular point.

2.1.3.1 Existing methods

The solution in Elkin (1965) produces a quartic equation for the distance from either the source or the observer to the specular point (though the he does not term it such), the roots of which can be used to retrieve a collection of distances from the specular point to the other point of interest. The resulting x-y pairs must then be analyzed to find the real specular point. There are several methods which are useful in retrieving the specular point from the generated x-y pairs: minimizing the source-specular-observer path length, employing a numerical root finding method with an initial 'guess' (e.g. Newton's method), or 'by probe'.

Fujimura et al. (2019) provides a formulation which works well for the method of path-length minimization, but suffers numerical instability in cases where the radius of the sphere is very small relative to source or observer distance (e.g. in the **one-finite** case discussed below). Glaeser (1999) demonstrates the numerical root finding approach (using an algorithm from Schwarze 1990) and notes that it is subject to "numerical instabilities that may lead to a considerable loss of accuracy" under certain circumstances. Elkin (1965) shows the 'by probe' method, but this is almost impossible to

capture algorithmically. These limitations beg the creation of a better, faster method.

2.1.3.2 Our approach

The salient problem, and in fact the only obstacle to a fully analytical solution, is that of deducing which root to use. We present a formulation that allows predetermination of the 'correct' root and enables the construction of a piecewise function which directly produces the location of the physical specular point. We find this formulation superior to previous work because of its analytical branch deduction – something lacking from every other formulation. The fully analytical approach avoids numerical instability and allows significantly faster computation.

We present two analytical solutions for this formulation: one that requires the distance to either the source or observer be approximateable as infinite (the "**one-finite**" case) and a second that works for any arbitrary configuration of source and observer (the "**both-finite**" case). Furthermore, we demonstrate that the complete solution can be expressed using no more than 3 roots in the **both-finite** case and no more than 2 roots in the **one-finite** case. We apply an Euler rotation and make use of the particulars of the resulting geometry to simplify the analysis, which also serves to minimize the associated computational expenses.

2.2 Euler Rotation

In both cases, it is useful to place either the source or observer at polar angle $\pi/2$ (i.e. above the north pole) for two reasons: first, it reduces the subsequent analysis to two dimensions and second, it allows a simplification which will be used in Sections 2.3 and 2.4. The simplest way to accomplish this is to start in Cartesian coordinates with the origin at the center of the sphere and then apply two sequential rotations. The first rotation uses the angles

$$\alpha = \frac{\pi}{2} - \operatorname{atan2}\left(y_{\mathrm{src}}, x_{\mathrm{src}}\right) \quad \text{and} \quad \beta = \frac{\pi}{2} - \sin^{-1}\left(\frac{z_{\mathrm{src}}}{R_{\mathrm{src}}}\right) \tag{2.4}$$

to define new coordinates in a rotated frame (denoted by the prime, $'$),

$$x'_k = z_k \sin\left(\beta\right) - x_k \cos\left(\beta\right) \sin\left(\alpha\right) - y_k \cos\left(\beta\right) \cos\left(\alpha\right), \tag{2.5a}$$

$$y'_k = x_k \cos\left(\alpha\right) + y_k \sin\left(\alpha\right), \text{ and} \tag{2.5b}$$

$$z'_k = x_k \sin\left(\beta\right) \sin\left(\alpha\right) + y_k \cos\left(\alpha\right) \sin\left(\beta\right) + z_k \cos\left(\beta\right). \tag{2.5c}$$

Here α is the azimuthal angle of the source (measured from the positive x-axis), and β is the inclination angle (measured from the positive z-axis). The second rotation is not strictly necessary; the critical portion is the placement of source or observer at $\pi/2$, accomplished by the first rotation. However, this second rotation further reduces the problem because it enforces $-\pi/2 < \theta_{\text{obs}} < \pi/2$ (in the rotated frame), allowing us to ignore some of the roots in Equations 2.13 and 2.21. This rotation takes

$$\gamma = \text{atan2}\left(y'_{\text{src}}, x'_{\text{src}}\right) \tag{2.6}$$

to establish coordinates in the twice-rotated, $''$, frame to be

$$x''_k = z'_k \sin\left(\gamma\right) - y'_k \cos\left(\gamma\right), \tag{2.7a}$$

$$y''_k = y'_k \sin\left(\gamma\right) - z'_k \cos\left(\gamma\right), \text{ and} \tag{2.7b}$$

$$z''_k = z'_k. \tag{2.7c}$$

With $k \in \{\text{src}, \text{obs}\}$ (i.e. for x''_{src}, x''_{obs}, etc) and where $\text{atan2}\left(y, x\right)$ defines the angle from the positive x-axis to (x, y). At the end, once the specular point has been calculated in this new coordinate system, we will apply these two rotations in reverse for $k \in \{\text{src}, \text{obs}, \text{spec}\}$ to recover the specular point in the original system. The rotated system (x''_k, y''_k, z''_k) will be used for the remainder of this paper.

In the **one-finite** case, the rotation should be performed to place the coordinate of finite radius at $\pi/2$. This allows full use to be made of the approximation $R_{\text{obs}} \gg R_{\text{sph}}$ or $R_{\text{src}} \gg R_{\text{sph}}$. In the **both-finite** case the rotation can place either coordinate at $\pi/2$, but we will place the source at $\pi/2$ for notational consistency.

2.3 One-Finite Case

If either the source or the observer can be considered infinitely distant, then after applying the Euler rotation in Equation 2.5 to place the finitely distant of the two at $\pi/2$ – as Figure 2.2 shows – θ_1 and θ_2 can be defined such that

$$\theta_1 = \tan^{-1}\left(\frac{R_{\text{obs}} \sin\left(\theta_{\text{obs}}\right) - R_{\text{sph}} \sin\left(\theta_{\text{spec}}\right)}{R_{\text{obs}} \cos\left(\theta_{\text{obs}}\right) - R_{\text{sph}} \cos\left(\theta_{\text{spec}}\right)}\right) \quad \text{and} \tag{2.8a}$$

$$\theta_2 = \tan^{-1}\left(\frac{R_{\text{src}} \sin\left(\theta_{\text{src}}\right) - R_{\text{sph}} \sin\left(\theta_{\text{spec}}\right)}{R_{\text{src}} \cos\left(\theta_{\text{src}}\right) - R_{\text{sph}} \cos\left(\theta_{\text{spec}}\right)}\right), \tag{2.8b}$$

where θ_e and θ_i are then given by

$$\theta_e = \frac{\pi}{2} - (\theta_{\rm spec} - \theta_1) \qquad \text{and} \qquad \theta_i = \frac{\pi}{2} - (\theta_2 - \theta_{\rm spec})\,.$$

2.3.1 Formulation

Since incidence angle (θ_i) must always be equal to emission angle (θ_e), then in the limit where $R_{\rm obs} \gg R_{\rm sph}$ we get

$$\theta_2 = 2\theta_{\rm spec} - \theta_{\rm obs}. \tag{2.9}$$

With the introduction of a constant,

$$c \equiv \frac{R_{\rm sph}}{R_{\rm src}}, \tag{2.10}$$

and substituting in θ_2 from Equation 2.8b, Equation 2.9 becomes

$$\frac{\sin(\theta_{\rm src}) - c\sin(\theta_{\rm spec})}{\cos(\theta_{\rm src}) - c\cos(\theta_{\rm spec})} = \tan(2\theta_{\rm spec} - \theta_{\rm obs})\,. \tag{2.11}$$

Making use of the identities

$$\tan(x-y) = \frac{\sin(2x) - \sin(2y)}{\cos(2x) + \cos(2y)} \qquad \text{and} \qquad \sin x\cos 2y \pm \sin 2y\cos x = \sin(2y \pm x)\,,$$

and the fact that the Euler rotation in Equation 2.5 results in $\theta_{\rm src} = \pi/2$, we get

$$\cos(4\theta_{\rm spec}) + c\sin(3\theta_{\rm spec}) + \cos(2\theta_{\rm obs}) = c\sin(\theta_{\rm spec} - 2\theta_{\rm obs})\,. \tag{2.12}$$

2.3.2 Solution

Equation 2.12 has sixteen unique, non-trivial solutions, courtesy of Wolfram Mathematica (Wolfram Research, 2020). We neglect eight of these because they represent special cases that are not physically meaningful for our purposes. The other eight are presented in Equation 2.14 using the

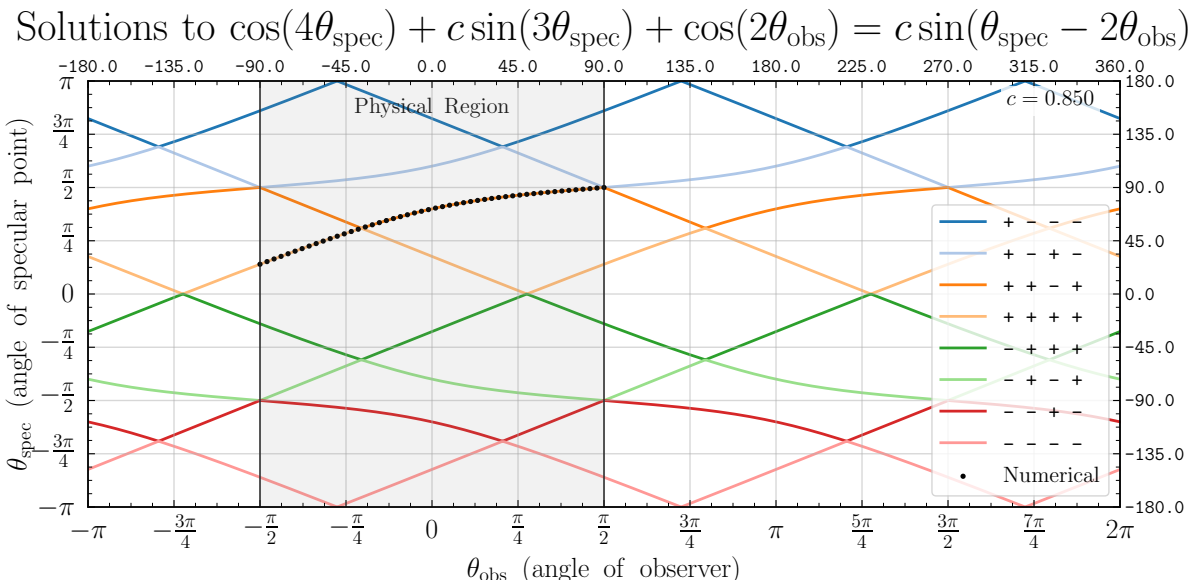

Figure 2.3: Plots of the solutions to Equation 2.12 where $R_{\text{sph}}/R_{\text{src}}$ (c) is 0.85. Note that the numerical method is only applied to $[-\pi/2, \pi/2]$ because the Euler rotation in Section 2.2 enforces θ_{obs} to be between $-\pi/2$ and $\pi/2$. θ_{obs} is the angle of the observer, θ_{spec} is the angle of the specular point. Both are in the rotated reference frame and measured from the positive x-axis.

coefficients we define in Equation 2.13.

$$D_0 \equiv 1152\left(c^2-4\right)\left(c^2+c^2\cos\left(2\theta_{\text{obs}}\right)-846\cos\left(2\theta_{\text{obs}}\right)-1\right)+c^2\sin\left(2\theta_{\text{obs}}\right)^2 \tag{2.13a}$$

$$D_1 \equiv \left(-4\left(160-128c^2+4c^4+96\cos\left(2\theta_{\text{obs}}\right)-96c^2\cos\left(2\theta_{\text{obs}}\right)\right)^3+\left(-16\left(c^2-4\right)^3-D_0\right)^2\right)^{1/2} \tag{2.13b}$$

$$D_2 \equiv \sqrt[3]{16\left(c^2-4\right)^3+D_0-D_1} \tag{2.13c}$$

$$D_3 \equiv \frac{1}{3\sqrt[3]{4}D_2}\left(40-32c^2+c^4+24\cos\left(2\theta_{\text{obs}}\right)-24c^2\cos\left(2\theta_{\text{obs}}\right)\right)+\frac{D_2}{24\sqrt[3]{2}} \tag{2.13d}$$

$$D_4 \equiv \sqrt{\frac{4-c^2}{6}+D_3} \tag{2.13e}$$

$$\theta_{\text{spec}} = \pm \begin{cases} \cos^{-1}\left(-\frac{D_4}{2} - \sqrt{-\frac{4-c^2}{3} - D_3 - \frac{c\sin(2\theta_{\text{obs}})}{8D_4}}\right) \\ \cos^{-1}\left(-\frac{D_4}{2} + \sqrt{-\frac{4-c^2}{3} - D_3 - \frac{c\sin(2\theta_{\text{obs}})}{8D_4}}\right) \\ \cos^{-1}\left(+\frac{D_4}{2} - \sqrt{-\frac{4-c^2}{3} - D_3 + \frac{c\sin(2\theta_{\text{obs}})}{8D_4}}\right) \\ \cos^{-1}\left(+\frac{D_4}{2} + \sqrt{-\frac{4-c^2}{3} - D_3 + \frac{c\sin(2\theta_{\text{obs}})}{8D_4}}\right) \end{cases} \tag{2.14}$$

The only differences between the solutions are the different signs associated with four terms. The possible combinations are $\pm - - -$, $\pm - + -$, $\pm + - +$, and $\pm + + +$. These solutions are plotted in Figure 2.3 for a source at 3000 km from the center of a sphere of radius 2575 km (i.e. Titan) and observer at much greater distance. After comparing these solutions against the numerical method, as in Figure 2.3, it is clear the branch that produces the physical specular point (on the exterior of the sphere) is

$$\theta_{\text{spec}} = \begin{cases} \cos^{-1}\left(\frac{D_4}{2} - \sqrt{-\frac{4-c^2}{3} - D_3 + \frac{c\sin(2\theta_{\text{obs}})}{8D_4}}\right) & \theta_{\text{spec}} < \frac{\pi(\cot^{-1}(c) - \theta_{\text{obs}})}{\pi + 2\cot^{-1}(c)} \\ \cos^{-1}\left(\frac{D_4}{2} + \sqrt{-\frac{4-c^2}{3} - D_3 + \frac{c\sin(2\theta_{\text{obs}})}{8D_4}}\right) & \theta_{\text{spec}} \geq \frac{\pi(\cot^{-1}(c) - \theta_{\text{obs}})}{\pi + 2\cot^{-1}(c)} \end{cases} \tag{2.15}$$

2.3.3 Observer at finite distance

By altering the Euler rotation in 2.5 to place the observer at $\theta_{\text{obs}} = \pi/2$ instead of the source, this same approach can be used in the limit where $R_{\text{src}} \gg R_{\text{sph}}$ and $R_{\text{obs}} \not\gg R_{\text{sph}}$. In this case Equation 2.12 becomes

$$\cos(4\theta_{\text{spec}}) + c\sin(3\theta_{\text{spec}}) + \cos(2\theta_{\text{src}}) = c\sin(\theta_{\text{spec}} - 2\theta_{\text{src}}) \tag{2.16}$$

and the solutions thereof will similarly have θ_{obs} replaced with θ_{src}.

2.4 Both-Finite Case

2.4.1 Formulation

If $R_{\text{obs}} \not\gg R_{\text{sph}}$ and $R_{\text{src}} \not\gg R_{\text{sph}}$ then the solution becomes even more complicated. In this case, Equation 2.9 becomes

$$\theta_2 + \theta_1 = 2\theta_{\text{spec}}, \tag{2.17}$$

and Equation 2.8 becomes

$$\theta_1 = \tan^{-1}\left(\frac{R_{\text{obs}}\sin(\theta_{\text{src}}) - R_{\text{sph}}\sin(\theta_{\text{spec}})}{R_{\text{obs}}\cos(\theta_{\text{src}}) - R_{\text{sph}}\cos(\theta_{\text{spec}})}\right) \quad \text{and} \tag{2.18a}$$

$$\theta_2 = \tan^{-1}\left(\frac{R_{\text{src}}\sin(\theta_{\text{obs}}) - R_{\text{sph}}\sin(\theta_{\text{spec}})}{R_{\text{src}}\cos(\theta_{\text{obs}}) - R_{\text{obs}}\cos(\theta_{\text{spec}})}\right), \tag{2.18b}$$

which yields

$$\tan^{-1}\left(\frac{R_{\text{obs}}\sin(\theta_{\text{src}}) - R_{\text{sph}}\sin(\theta_{\text{spec}})}{R_{\text{obs}}\cos(\theta_{\text{src}}) - R_{\text{sph}}\cos(\theta_{\text{spec}})}\right) + \tan^{-1}\left(\frac{R_{\text{src}}\sin(\theta_{\text{obs}}) - R_{\text{sph}}\sin(\theta_{\text{spec}})}{R_{\text{src}}\cos(\theta_{\text{obs}}) - R_{\text{obs}}\cos(\theta_{\text{spec}})}\right) = 2\theta_{\text{spec}} \ .$$

Again making use of $\theta_{\text{src}} = \pi/2$ and defining a new constant,

$$b \equiv \frac{R_{\text{sph}}}{R_{\text{obs}}}, \tag{2.19}$$

the **both-finite** equivalent of Equation 2.12 is

$$\frac{c\sin(\theta_{\text{obs}} + \theta_{\text{spec}}) - cb\sin(2\theta_{\text{spec}}) + \cos\theta_{\text{obs}} - b\cos\theta_{\text{spec}}}{c\cos(\theta_{\text{obs}} + \theta_{\text{spec}}) - cb\cos(2\theta_{\text{spec}}) + \sin\theta_{\text{obs}} - b\sin\theta_{\text{spec}}} = \tan(2\theta_{\text{spec}}) \ . \tag{2.20}$$

With $b = 0$ (and with some manipulation) we can recover Equation 2.12. Doing so is nontrivial and will be neglected for brevity, but it is clearly demonstrated by the agreement between the two methods when $b \approx 0$ as shown in section 2.5.2.

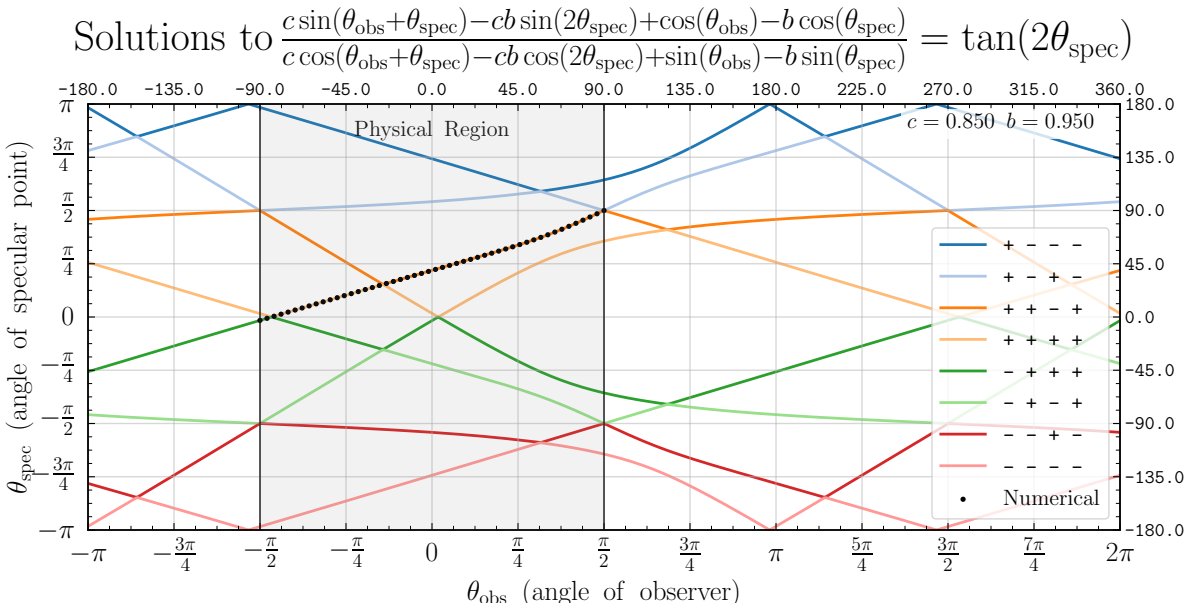

Figure 2.4: Solutions to Equation 2.22 for R_{sph}/R_{src} (c) of 0.85 and R_{sph}/R_{obs} (b) of 0.95. Note that the numerical method is only applied to $[-\pi/2, \pi/2]$ because the Euler rotation in Section 2.2 enforces θ_{obs} to be between $-\pi/2$ and $\pi/2$. θ_{obs} is the angle of the observer, θ_{spec} is the angle of the specular point. Both are in the rotated reference frame and measured from the positive x-axis.

2.4.2 Solution

Similar to Equation 2.12, Equation 2.20 produces sixteen unique, non-trivial solutions – only eight of which are physically meaningful (the others represent special cases). These are presented in Equation 2.22 using the coefficients defined in Equation 2.21

$$E_0 \equiv b^2 + c^2 + 2bc\sin\theta_{\rm obs} - 4 \tag{2.21a}$$

$$E_1 \equiv 24\cos\left(\theta_{\rm obs}\right)^2\left(b^2 - bc\sin\theta_{\rm obs} - 2c^2 + 2\right) + E_0^2 \tag{2.21b}$$

$$E_2 \equiv 72E_0\cos^2\theta_{\rm obs}\left(b^2 - bc\sin\theta_{\rm obs} + 4c^2 - 4\right) + \tag{2.21c}$$

$$432\cos^2\theta_{\rm obs}\Big((b - c\sin\theta_{\rm obs})^2 - b^2(c^2-1)\cos^2\theta_{\rm obs}\Big) + 2E_0^3 \tag{2.21d}$$

$$E_3 \equiv \left(\sqrt{E_2^2 - 4E_1^3} + E_2\right)^{1/3} \tag{2.21e}$$

$$E_4 \equiv \frac{\sqrt[3]{4}\,E_3^2 + \sqrt[3]{16}\,E_1}{24E_3} \tag{2.21f}$$

$$E_5 \equiv \sqrt{\frac{b^2\cos\left(\theta_{\rm obs}\right)^2}{4} + \frac{E_3}{12\sqrt[3]{2}} + \frac{E_1}{6\sqrt[3]{4}\;E_3} - \frac{E_0}{6}} \tag{2.21g}$$

$$E_6 \equiv \frac{(b\cos\theta_{\rm obs})^3 - bE_0\cos\theta_{\rm obs} - 4\cos\theta_{\rm obs}\,(b - c\sin\theta_{\rm obs})}{16E_5} \tag{2.21h}$$

$$\theta_{\rm spec} = \pm\begin{cases} \cos^{-1}\left(\dfrac{b\cos\theta_{\rm obs} - 2E_5}{4} - \dfrac{1}{2}\sqrt{\dfrac{3(b\cos\theta_{\rm obs})^2 - 2E_0}{12} - E_4 - E_6}\right) \\ \cos^{-1}\left(\dfrac{b\cos\theta_{\rm obs} - 2E_5}{4} + \dfrac{1}{2}\sqrt{\dfrac{3(b\cos\theta_{\rm obs})^2 - 2E_0}{12} - E_4 - E_6}\right) \\ \cos^{-1}\left(\dfrac{b\cos\theta_{\rm obs} + 2E_5}{4} - \dfrac{1}{2}\sqrt{\dfrac{3(b\cos\theta_{\rm obs})^2 - 2E_0}{12} - E_4 + E_6}\right) \\ \cos^{-1}\left(\dfrac{b\cos\theta_{\rm obs} + 2E_5}{4} + \dfrac{1}{2}\sqrt{\dfrac{3(b\cos\theta_{\rm obs})^2 - 2E_0}{12} - E_4 + E_6}\right) \end{cases} \tag{2.22}$$

As in Equation 2.14, the only differences between the solutions are the signs on four terms. The combinations of these signs are $\pm - - -$, $\pm - + -$, $\pm + - +$, and $\pm + + +$ (the same as in 2.14). They are plotted in Figure 2.4. Comparison with the numerical method indicates the branch that produces the physical specular point (on the exterior of the sphere) is

$$E_7 \equiv \frac{3(b\cos\theta_{\rm obs})^2 - 2E_0}{12} - E_4 + E_6$$

$$\theta_{\text{spec}} = \begin{cases} -\cos^{-1}\left(\dfrac{b\cos\theta_{\text{obs}} + 2E_5}{4} + \dfrac{1}{2}\sqrt{E_7}\right) & -\dfrac{\pi}{2} \le \theta_{\text{obs}} < L_1 \\ \cos^{-1}\left(\dfrac{b\cos\theta_{\text{obs}} + 2E_5}{4} + \dfrac{1}{2}\sqrt{E_7}\right) & L_1 \le \theta_{\text{obs}} \le L_2 \quad \text{and} \quad \dfrac{\partial E_7}{\partial \theta_{\text{obs}}} \le 0 \\ \cos^{-1}\left(\dfrac{b\cos\theta_{\text{obs}} + 2E_5}{4} - \dfrac{1}{2}\sqrt{E_7}\right) & L_2 < \theta_{\text{obs}} \le \dfrac{\pi}{2} \quad \text{or} \quad \dfrac{\partial E_7}{\partial \theta_{\text{obs}}} > 0 \end{cases} \tag{2.23}$$

where the limits L_1 and L_2 are given by

$$L_1 = \tan^{-1}\left(\frac{c + b\sqrt{1 - b^2 + c^2}}{b^2 - c^2}\right) \tag{2.24a}$$

$$L_2 = \tan^{-1}\left(\frac{c - b\sqrt{1 - b^2 + c^2}}{b^2 - c^2}\right) \tag{2.24b}$$

It should be noted that L_1 may occur at less than $-\pi/2$, and in such cases only the second two parts in Equation 2.23 are necessary. Additionally, while there is an exact solution for the partial derivative of E_7, in practical computation it is robust to use the approximation

$$\frac{\partial E_7}{\partial \theta_{\text{obs}}} \approx E_7\big|_{\theta_{\text{obs}}} - E_7\big|_{\theta_{\text{obs}} - \zeta}$$

where ζ is directly related to the branch-deduction precision (assuming it is larger than the computation precision), e.g. $\zeta = 10^{-9}$ will result in the correct branch being chosen within 10^{-9}. Our **both-finite** implementation in the following section makes use of this approach.

2.5 Computation

Implementations of these solutions in both Python and C++ can be found in this GitHub Repository[1]. Also in the repository are Wolfram Language Package files for easily reading the solutions into Mathematica. The C++ implementations were used for performance benchmarking. The solution for the **one-finite** case performs significantly better than that of the **both-finite** case, but in many applications the difference will be negligible because both routines are relatively minimal – especially compared to numerical methods.

[1] https://github.com/WMiller256/Alhazen-Ptolemy

2.5.1 Performance

Excluding branch deductions (and after compiling with `gcc` option `-O3`), our C++ implementation of the **one-finite** solution requires 76 floating point operations, 6 trigonometric and 5 `sqrt` calls with one expensive exponentiation operation to find a complex cube root. The corresponding **both-finite** code requires 131 floating point operations, 5 trigonometric and 5 `sqrt` calls, and one expensive cube root.

Benchmarked on an hexacore Intel i7-8750H the **one-finite** solution averaged 23 ns per iteration and the **both-finite** solution averaged 121 ns per iteration for 10^9 random combinations of b, c, and θ_{obs}. The code used for benchmarking is included in the linked GitHub repository. In cases where $L_1 \leq \theta_{\text{obs}} \leq L_2$ the **one-finite** implementation has a more significant speed advantage because the **both-finite** implementation's branch deductions are most expensive in this region.

2.5.2 Precision

The agreement between the **one-finite** solution and the **both-finite** solution is shown in Figure 2.5. When using the 64-bit `double` type, the machine precision for either calculation is roughly 10^{-7} due to the trigonometric functions. The use of the `long double` type (which is implementation dependent but usually corresponds to between 80-bit and 128-bit storage) can be used to improve this precision to 10^{-15}. It should be noted that any numerical methods will have the same or similar precision limitations. Figure 2.5 does not make use of the extra precision and therefore the point at which the average error exceeds 10^{-6} can be considered the point at which the disagreement between the two methods exceeds the machine precision. This occurs at around $b/c = 2\times10^{-5}$, i.e. when either R_{obs} or R_{src} is 50,000 times larger than the other. At a ratio of 500 the accuracy drops to $\pm1.72\times10^{-2}$ degrees, but only for the nearly worst-case-scenario of $c = 0.95$. As c decreases the accuracy of θ_{spec} at the same b/c improves, e.g. for $c = 0.1$ the accuracy is $\pm1.72\times10^{-6}$ degrees at $b/c = 500$.

2.6 Conclusion

2.6.1 SRTC

Our method is of particular use in simulating radiative transfer on Titan, i.e. in the `SRTC++` model (Barnes et al., 2018). Specifically, calculating specular reflections due to atmospheric scattering allows us to compensate for the adjacency effect – when atmospheric scattering redirects light reflected off bright material adjacent to spectrally dark regions (e.g. lakes) and causes them to appear brighter than they should. This effect has been well documented on Earth (Odell & Weinman, 1975; Tanre et al.,

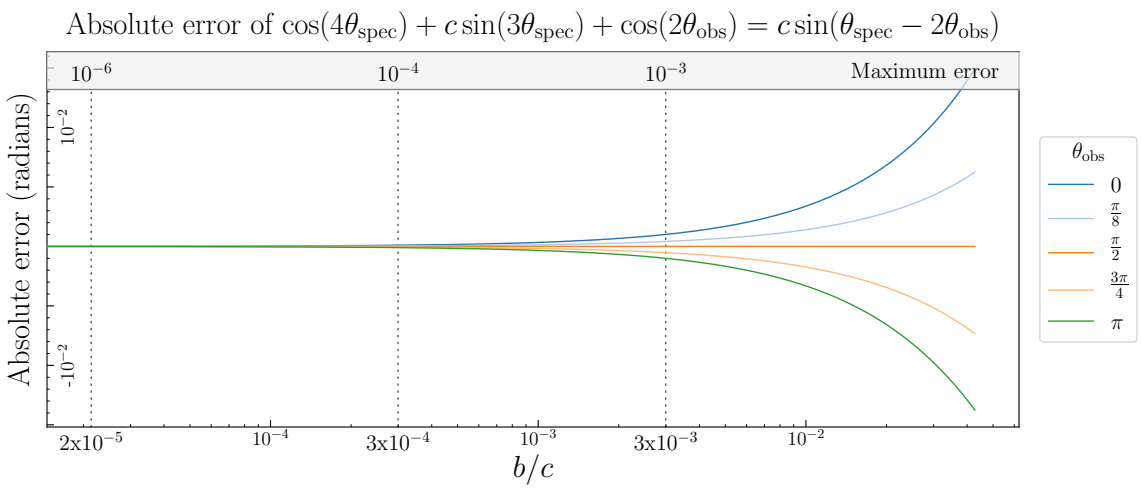

Figure 2.5: The agreement between the **one-finite** and **both-finite** solutions as a function of the ratio between b and c with the maximum error across all θ_{obs} for a given b/c. Note that even though θ_{obs} is confined to $[-\pi/2, \pi/2]$, we are showing the error for θ_{obs} between 0 and π. We do so to demonstrate that for $-\pi/2 > \theta_{\text{obs}}$ or $\theta_{\text{obs}} < \pi/2$ (in the reference frame before the second Euler rotation) the persistent error is symmetrically reversed.

1981; Minomura et al., 2001; Sterckx et al., 2011; Kiselev et al., 2015) and Karkoschka & Schroder (2016) found it necessary to compensate for it in their reflectivity analysis of *Huygens* data.

Realistically simulating Titan's atmosphere and surface requires compensation for the adjacency effect, and many high fidelity simulations. In such simulations, the specular point must be calculated for every scattering event (and there are usually billions). The effect of our method on the computation speed of these simulations is negligible (e.g. about 1 minute over 4 days of execution time). The same cannot be said of existing methods.

2.6.2 Rendering

Another notable application is in rendering computer graphics. Computation speed is critical here as well, and often multiple light sources or reflections must be accounted for – each with a separate specular point. Sources with non-negligible spatial extent are also common, requiring the calculation of multiple specular points during each refresh. We have formulated this paper to emphasize spherical surfaces, but in fact the analysis in the rotated reference frame can be applied to any reflective surfaces of revolution, provided the intersection between the surface and the plane formed by the source, observer, and specular point forms a circle.

2.6.3 Extension to elliptical geometry

Extending our approach to elliptical geometry would drastically broaden its usefulness by providing an analytical, branch-deducing method for calculating specular points on any closed conic section. As mentioned previously, this extension would enable our solution to be used in GPS communications. But it would also expand the usefulness for computer rendering because most planar slices of regular surfaces of revolution produce one or several non-circular conic sections. We leave the elliptical case to future work.

2.6.4 In Summary

We have presented a formulation of the Alhazen-Ptolemy problem in which the physical specular point is predeterminable, we find this formulation to be superior to previous approaches because it does not rely on numerical methods for branch deduction. Furthermore, we have shown that the solution can be directly written as a piecewise function with only 3 branches in the **both-finite** case and only 2 branches in the **one-finite** case. Employing an Euler rotation, normalizing the radius of the sphere to unity, and translating the rotated coordinate system to center on the specular point are critical in this formulation and in enabling robust and efficient computational implementation. Our method is useful for studying the ethane composition of Titan's lakes with `SRTC++`, rendering 3D computer graphics, and lays the groundwork for extension to elliptical geometry.

Chapter 3 Compiling Cassini-VIMS Titan Data

William J. Miller, Jason W. Barnes

During her 13-year tenure in the Saturn system, Cassini completed 127 imaging flybys of Titan. These immense and invaluable data have greatly furthered our understanding of Titan, of particular import are the observations from the prime instrument; the Visible Infrared Mapping Spectrometer (VIMS). There are significant processing barriers to overcome in synthesizing useful products from the VIMS data. To enable their dissemination and lower these barriers I have integrated the creation of data products compatible with Version 4 of the Planetary Data System (PDS) standard into the `Jcube` codebase. We have completed creating PDS data products for 89 flybys.

3.1 Introduction

The VIMS dataset is comprised of 694 gigabytes of imaging data in 352 wavelengths in the visible and near- infrared. This massive dataset has proven invaluable to furthering our understanding of Titan, revealing a body with active hydrology in the form of methane. These data have shown rivers and seas of methane (Barnes et al., 2007, 2006; Stephan et al., 2010), clouds (Griffith et al., 2005; Rodriguez et al., 2009), dunes (Barnes et al., 2008), and even allowed the measurement of waves in Jinpo Lacus and Kraken Mare (Barnes et al., 2011).

3.1.1 The Problem

The process by which these data can be digested into something scientifically useful is by no means simple. The raw data are delivered from the calibration pipeline as Integrated Software for Imagers and Spectrometers (ISIS) Version 2 files. ISIS2 files are themselves fairly workable – albeit somewhat dated – the ISIS2 format is supported by the C++ and Python Application Programming Interfaces (API) of the Geospatial Data Abstraction Library (GDAL). The problems arise when mosaicing (3.2.2) or coloring the raw images to generate summary products for publication and dissemination.

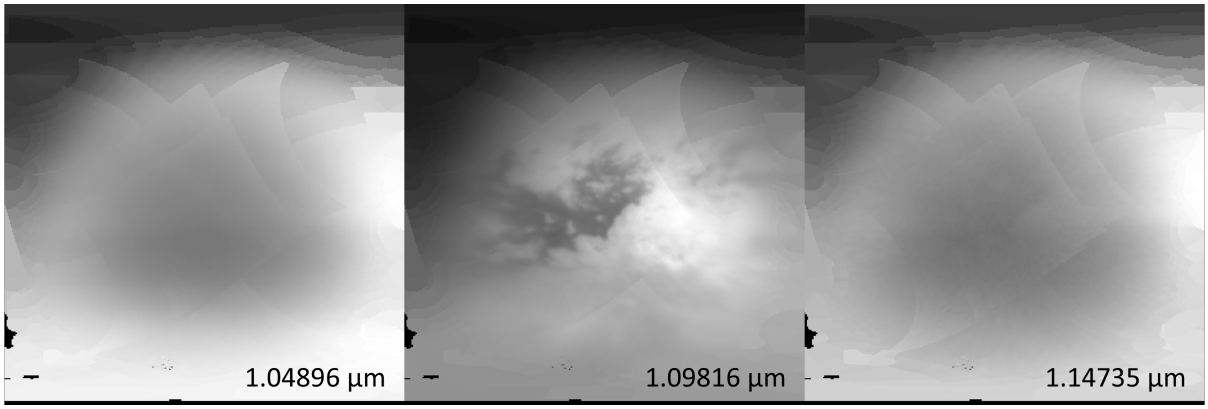

Figure 3.1: The 1-µm atmospheric window. The 1.09816 band is the only one of the three shown here which receives sufficient surface signal to be useful. There are only four VIMS bands in this window.

The main issue of coloring comes from the optical characteristics of Titan's atmosphere, namely the numerous and broad absorption bands in the visible and infrared ranges. Figure 3.1 illustrates the narrowness of the 1-µm atmospheric window. We use data from five of these atmospheric windows to create colored images, those five windows only span about 50 VIMS wavelengths. Hence, useful colored images depicting surface detail cannot be created from a full cube, but only from an appropriate subset of planes. Figure 3.2 shows two composite maps colored using a scheme which assigns blue to 1.3 µm, green to 2 µm, and red to 5 µm – our standard scheme.

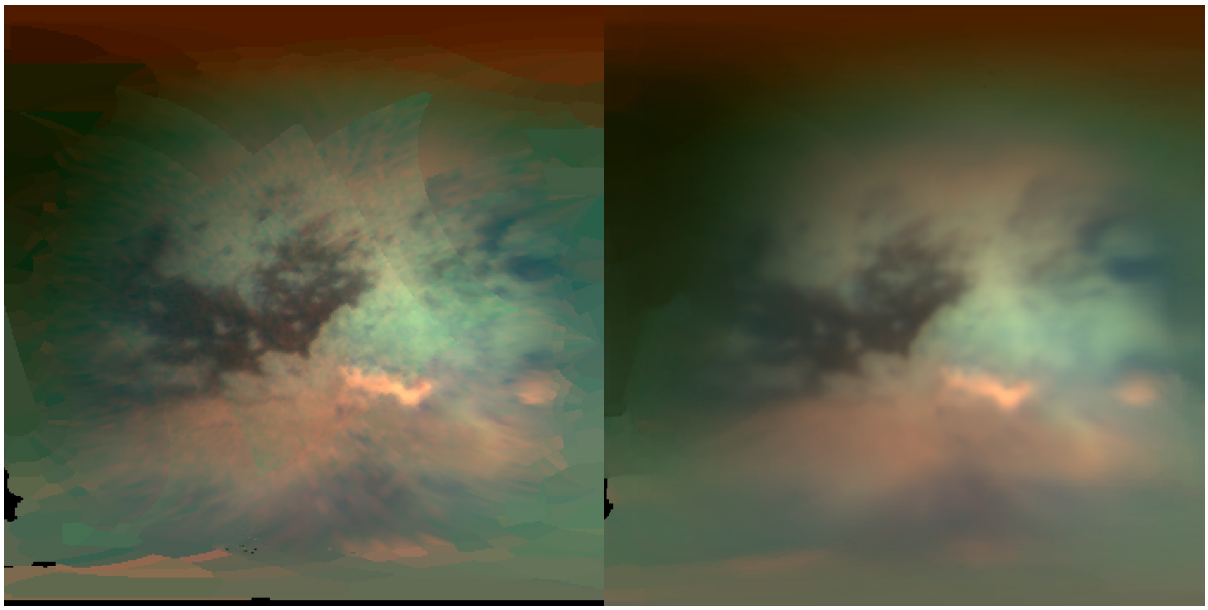

Figure 3.2: Colored and cylindrically-projected global maps from the T8 flyby using Lanczos interpolation, with sequential replacement (left) and coadd (right) meshing. Blue has been assigned to 1.3 µm, green to 2 µm, and red to 5 µm.

The second significant barrier to creating summary products is the staggering variety of viewing geometries. This is not unique to the Cassini mission; any spacecraft will create a vast array of viewing geometries, either during a flyby or an orbit. For the Titan data, this works to our advantage because it provides us coverage at multiple scales with temporal adjacency – allowing us to create composites which put higher resolution data in broader context. It also works to our disadvantage because it makes the creation of said composites more complicated.

The challenge is that every single observation, which in the case of the VIMS instrument is not only every image but every single *pixel*, the spacecraft has a different three-axis position and orientation. Fortunately, the techniques for compensating for the different viewing geometries and projecting the different observations onto a common canvas are well-established. Unfortunately, that doesn't make them easy. The spacecraft and target position and orientation must first be extracted from the SPICE (Spacecraft, Planet, Instrument, Camera-matrix, and Events) kernels to define the viewing geometry. This geometry is used to create a homography (transformation matrix) between the observation and a common canvas, and then the homography is applied to 'project' the observation onto the canvas.

In short, a researcher interested in using data from VIMS who does not already possess the necessary software infrastructure to perform the processing must

1. Ingest the ISIS2 files
2. Extract the relevant information from the SPICE kernels
3. Compose a homography for each pixel
4. Apply the homographies
5. Compose the projected cubes
6. Create colored images from a subset of planes
7. Conduct their investigation

Furthermore, since there are hundreds of cubes per flyby, practically speaking the above must be automated by the development of redundant software infrastructure. I expect most researchers would much rather skip to number 7, given the option.

3.1.2 The Project

The goal of this project is to create a comprehensive set of data products – specifically, cylindrically projected maps – spanning all 127 flybys.[1] These products are stratified by sequence, interpolation, mesh, and coverage. For each flyby, there can be a total of 24 products as shown in Figure 3.3. That being said, not every flyby will have products from both ingress and egress or for every coverage, but every sequence will have both Lanczos and nearest neighbor interpolations, which will each have coadd and sequential replace meshes for every coverage. At minimum, a sequence will contain a global Lanczos and nearest neighbor for both coadd and sequential replace (4 products in total).

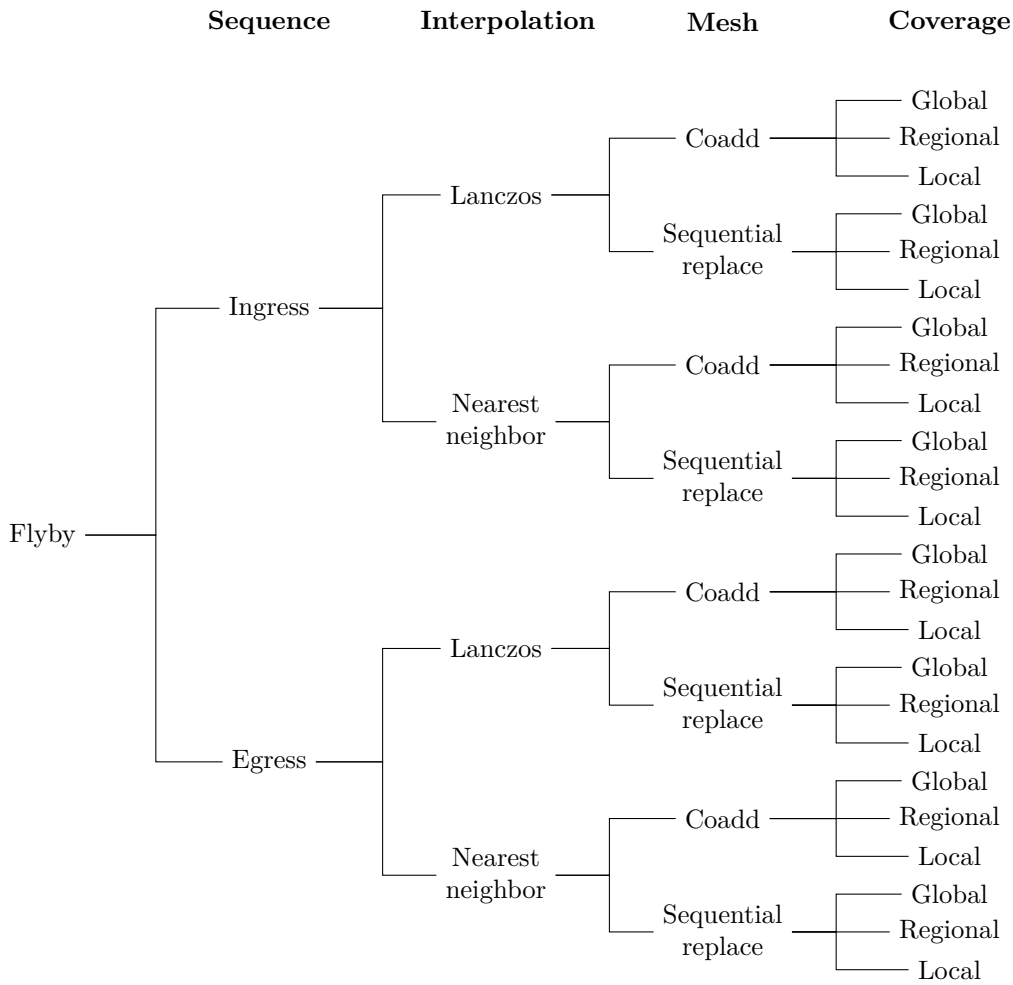

Figure 3.3: The stratification of data products for each flyby. Not every flyby will feature a complete tree, but will at minimum contain global coverage of each mesh and interpolation for at least one sequence.

[1]Several flybys are excluded due to lack of usable data, e.g. the Tc flyby which was reserved for Huygens insertion.

These products will summarize the data from every flyby into an easily ingestable set of maps: with a few simple lines of Python or C++, they can be read into memory for analysis. Furthermore, data viewers such as `pds4_viewer` are capable of immediately opening and displaying the maps without any configuration or modification. A prospective researcher armed with access to these maps would be able to go from concept to investigation in a matter of minutes, regardless of their technical background.

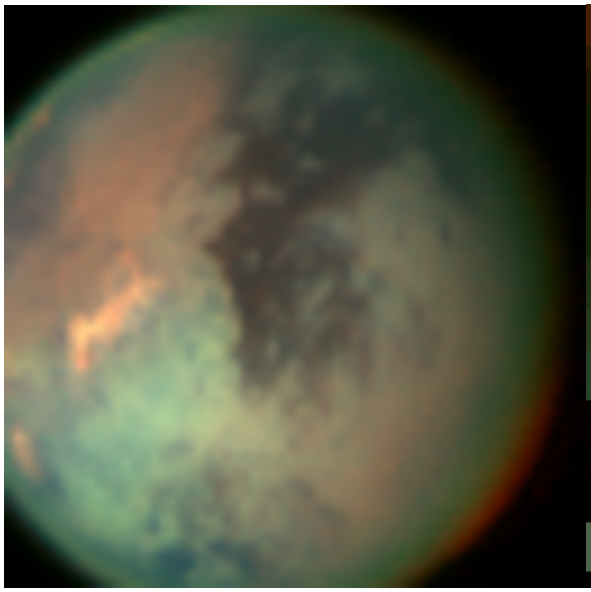

Figure 3.4: Colored version of the calibrated 1509136601_1 ISIS cube

Figure 3.5: Lanczos-interpolated cylindrical projection of the 1509136601_1 ISIS cube.

3.2 Processing

After retrieving the calibrated data from the VIMS server, the process of generating a composite map is as follows,

1. Project the ISIS2 cube files onto a cylindrical canvas.
2. Color the projected cubes to create a set of images for review.
3. Review the cylindrically-projected images to determine which to include in the composites. The set will be different for sequential replace and coadd.
4. Mesh the cubes into a single composite map.
5. Iterate the reviewing, filtering, and meshing as needed.

Much of this processing can be, and is, delegated to computing resources through automated processes.

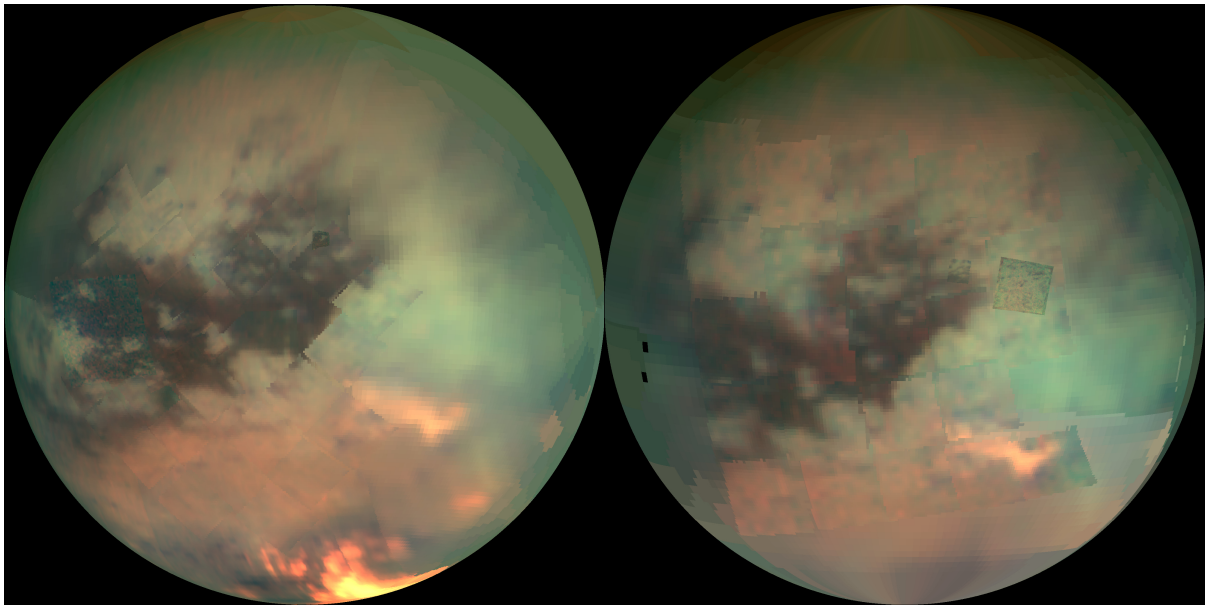

Figure 3.6: Comparison between orthographic projections of the Lanczos-interpolated, sequential replacement, global composites from Ta (left) and T3 (right). Note the bright orange cloud feature to the south in the Ta observations is no longer present by T3.

However, the determinations of which observations to include over what coverage and at what resolution must be made by a human, making this a tedious and time-consuming process.

3.2.1 Projecting

The projection step is required in order to mesh different observations into a single composite. Without projecting the ISIS cubes, there is no relationship between pixels and spatial coordinates (latitude and longitude). It also allows for direct comparison between features in different observations, even entirely different flybys.

This is of particular use in identifying and studying ephemeral features such as clouds. Take for example the comparison between the orthographic projections of the sequential replace, Lanczos composites from Ta and T3 in Figure 3.6. Ta shows a prominent bright orange feature (a cloud) in the south, while T3 does not. Thanks to the projection we can at a glance observe the extent of the cloud. In fact, since the images are aligned to the same canvas, we could do that same analysis quantitatively – which would be impossible with unprojected images.

Projected cubes can also be directly compared against observations from other instruments, most importantly those from Cassini's radar instrument (RADAR). Combining RADAR and VIMS data provides both spectral and altimetry information of the same features, which is particularly useful in identifying and analyzing lakes.

There are many applications for quantitatively analyzing the temporal variations on Titan's surface, this is merely one example. The data products we are creating will enable such analyses in unprecedented fashion.

3.2.2 Mosaicing

The process of mosaicing is equally critical to creating a manageable set of data products because it allows an entire flyby to be viewed in a single image. Similarly invaluable is the ability to put high resolution data into broader context by mosaicing observations of differing resolutions. Furthermore, it allows the combination of observations from different flybys into a single composite, which is particularly useful in summarizing the highest resolution data from multiple flybys.

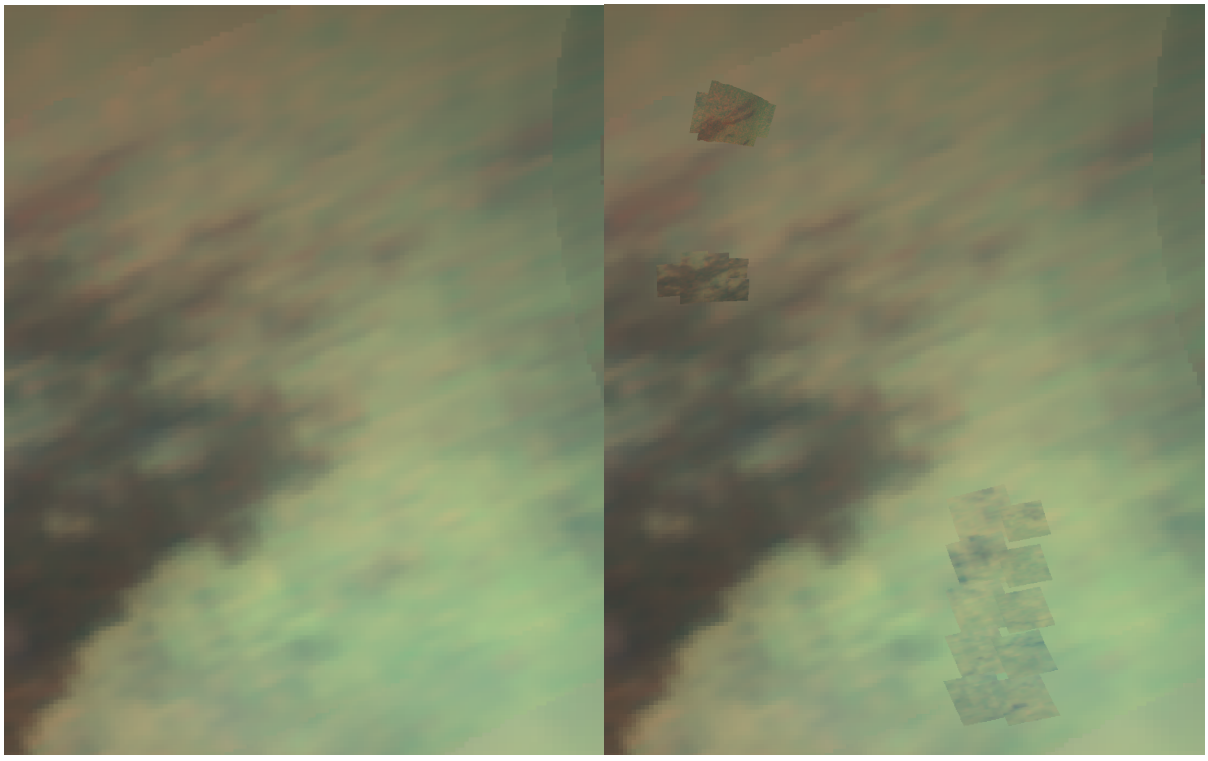

Figure 3.7: Demonstration of sequential replacement, the left panel shows the lower-resolution observations only and the right shows the higher-resolution observations layered over the others. The high resolution data are summarized in context by this method. The data are from the ingress sequence of flyby Tb.

Condensing hundreds of cubes comprising hundreds of millions of pixels into one image is quite the task, and its usefulness cannot be understated. Fortunately, this process is very straightforward once the cubes have been projected. The process of mosaicing is also described as 'meshing'. We use two meshing methods:

Coadd – pixels at the same latitude and longitude are averaged into a single value. This method provides better signal-to-noise at the cost of lower spatial resolution.

Sequential Replacement – the cubes are layered in a prescribed order with the highest resolution on top and then flattened. This method provides higher spatial resolution, but at a lower signal-to-noise ratio. Figure 3.7 demonstrates this approach.

3.3 Results

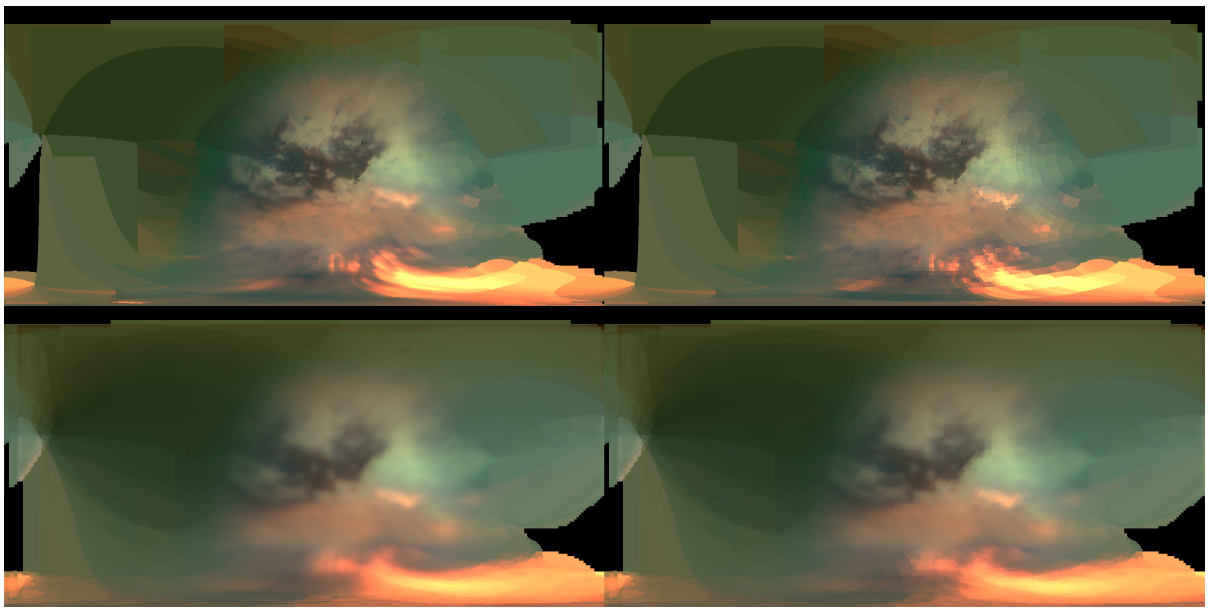

Figure 3.8: Final global maps for the Ingress sequence of the Ta flyby. Top row shows the sequential replacement mesh, bottom row shows the coadd mesh. Left column shows Lanzcos interpolation, right column shows nearest neighbor.

Presently, we have completed creating final data products for 89 flybys, totaling 528 individual composites, 274 from the prime mission (T0 - T44) and 254 from the extended mission (T45 - T126). The products from the prime mission are being finalized for delivery to the PDS, and those from the extended mission will be delivered upon their completion. Figure 3.8 shows the complete set of global cylindrically-projected composites from the Ta flyby, as they will be delivered.

Orthographic projection offers a number of advantages over cylindrical projection and in many cases is

more appropriate than cylindrical; it allow the observations from a flyby to be placed in the context of the original viewing geometry of the sequence.[2] This is particularly useful for illustrating specular geometries, especially when there is a specular reflection, and for analyzing haze properties near the terminator (it is in fact critical for this analysis). We may also later include orthographic projections of these products (see Figure 3.9) as well as the individual cylindrically-projected cubes. But for now that is left to future work.

3.4 Conclusion

I have integrated the ability to generate PDS4 compliant files into our existing processing infrastructure, enabling the creation and dissemination of a stratified collection of composite maps summarizing Cassini's VIMS observations of Titan. This set will make it substantially easier for future researchers to work with the data it covers, and enhance the quality of future analyses which leverage it by facilitating temporal comparisons of features or regions of interest.

We have completed this set of data products for the prime mission, flybys T0 through T44, and a majority of the extended mission, T45 through T126. I am preparing the products from the prime mission for final deliver to the PDS. The remaining maps will be delivered upon their completion.

[2]The viewing geometry changes constantly, as mentioned previously, but we can pick a viewing geometry either early in ingress or late in egress which is representative for the purposes of summarization.

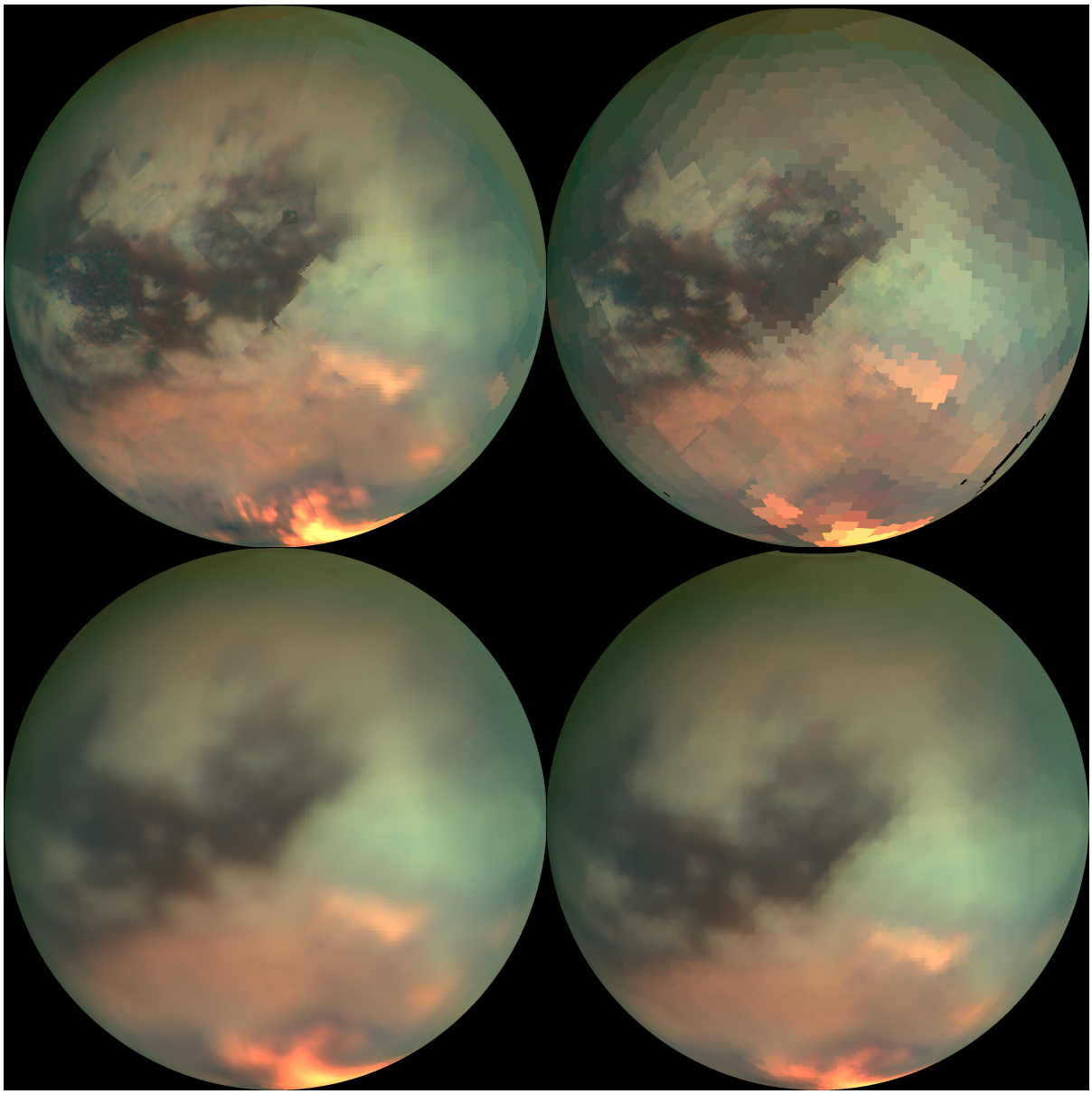

Figure 3.9: Orthographic projections of the global maps for the Ingress sequence of the Ta flyby. Top row shows the sequential replacement mesh, bottom row shows the coadd mesh. Left column shows Lanzcos interpolation, right column shows nearest neighbor.

Chapter 4 Modernizing orbital integration

William J. Miller

There is tremendous unrealized potential arising from the advantages offered by modern computing hardware in the implementation of orbital integrators. `Vulcan` creates a structure in which any integration scheme can be implemented and given full advantage of its efficient parallelization, advanced memory management, and concurrent file output. This speeds integration by potentially hundreds of times - reducing the time needed for an Euler-Symplectic integrator to simulate the solar system for a gigayear from a month to a few hours. Furthermore, new calculations - and even entirely new integration algorithms - can be added without significantly changing the source code.

4.1 Introduction

The driving force behind the development of new orbital and N-body integration methods is computational efficiency. Greater efficiency allows the study of a wider parameter space, or the same space at higher resolution. It allows better understanding of how the chaos of a system varies within its parameter space. More efficient integration relaxes the lower bound on the simulation time-step, enabling simulations of systems with faster-orbiting bodies, more close encounters, or a wider disparity in orbital period between bodies. The challenge is to balance the drive for greater efficiency with the need for accuracy. Existing orbital integrators can be stratified by both algorithmic and implementation differences, and achieve varying degrees of success in improving the computational efficiency without sacrificing accuracy.

The most prevalent integrators, `MERCURY` (Chambers, 1999), `SyMBA` (Duncan et al., 1998a), and `RMVS3` (Levison & Duncan, 1994), share a common restriction imposed by their age - they were designed for computer architectures which no longer exist, and consequently, were driven by design restrictions that are incongruous with modern computing hardware: they lack parallelization, memory management is geared toward minimizing memory footprint, and writing to file is performed in serial. The strength of these integrators is in their algorithmic improvements - symplectic and hybrid-symplectic integrators are an order of magnitude faster than more primitive methods (e.g. Bulirsch-Stoer, Runge-Kutta) (Chambers, 1999). Subsequent integration schemes have generated further algorithmic improvements (Wisdom, 2017) (Hernandez, 2016) (González et al., 2000) (Peláez et al., 2007). But still none fully leverage the potential of modern hardware.

The most successful attempt at improving implementation efficiency by taking advantage of modern hardware is the `Rebound` project (Rein, H. & Liu, S.-F., 2012) - it features intra-step parallelization using the OpenMP library and the memory management approach optimizes for speed instead of footprint. However, it still falls short of fully realizing the potential of modern hardware - this is detailed in subsection 4.3.

The divergent development of `SMercury` (Lissauer et al., 2012) and `Mercury-T` (Bolmont et al., 2015) highlights an additional drawback of existing implementations: each implementation is constructed on a different code base, marrying two algorithms from existing implementations requires prohibitive overhaul to the code base - even adding a new calculation can create a development branch which is incompatible with other branches - as is the case with the addition of spin and tidal calculations to `MERCURY`.

No single algorithm or implementation exists which can simultaneously maximize efficiency and accuracy for all use cases. This has resulted in the broad array of algorithms and implementations outlined above and creates a need which `Vulcan` is intended to fulfill - a structure which can provide the advantages of modern hardware to any integrator or integration scheme. `Vulcan` is built for efficiency in both execution and development time: it features efficient parallelization and memory management for small-N simulations and intelligent, concurrent file output with the ability to add new calculations in just a few lines of code.

4.2 Practical Details

Large-scale N-body integrators such as `PKDGRAV` (Potter et al., 2017), `GADGET-2` (Springel, 2005), and `GANDALF` (Hubber et al., 2018) (as well as its predecessor `SEREN` Hubber et al., 2011) feature parallelization schemes which are well optimized for the simulation of many particles, but not for small-scale N-body integrations. Most of the prevalent orbital integrators which are geared towards a small number of bodies do not employ parallelization, and those that do (i.e. `Rebound`) are restricted to intra-step parallelization - where the operations *within* a step are performed in parallel, but where there is no thread-continuity *between* steps.

I posit that one reason for this lack is the expense of thread sleeping and resumption - sleeping takes about 100 clock cycles, resuming takes at least as much, often more. Modern processors can usually perform between 4 (Intel Core 2 Pentium series) and 32 (Intel Skylake-X series) flop/cycle (Dolbeau, 2018). So at the lower bound, sleeping and resuming a thread corresponds to between 800 and 6,400 flop - depending on the architecture. Since calculating the gravitational interactions of a system of n

massive bodies requires

$$\frac{28n!}{2(n-2)!}$$

floating point operations, then a corresponding Bulirsch-Stoer integration (Stoer & Bulirsch, 1980) requires

$$\sum_{i=1}^{n_s-2} 26 \cdot i + \sum_{i=1}^{n_s} \left(48 + \frac{28n!}{2(n-2)!}\right) \cdot i + 68 \cdot n_s + 3\left(\frac{28n!}{2(n-2)!}\right) \cdot n_s$$

floating point operations per step. Where n_s is the maximum number of substeps. For small n, on most architectures this is on the scale of the cost to sleep and resume the threads. Significant improvement is not realized until the ratio between the hardware concurrency (number of logical cores) and number of bodies is small. Hence, intra-step parallelization does not offer much improvement for small-scale N-body integration (see Figure 4.1). This is not true for inter-step parallelization which employs 'busy-waiting' or 'spin-locking' (normally considered an anti-pattern Gamma et al., 1995) to guarantee against sleeping between steps. Implementing this is non-trivial.

It should be noted, however, that the spin-lock timeout period in C++ is implementation dependent, and consequently the magnitude of the pessimization in intra-step parallelization varies based on the specific simulation and the integrator used.

4.3 Vulcan

The continued advancement of the algorithms used in orbital integration, as well as the variety of calculations which are useful in different simulations – from the common (e.g. spins, tides, general relativity) to the exotic (e.g. YORP and Yarkovsky) – demand an implementation that brings the advantages of modern computing hardware to any integration scheme. `Vulcan` seeks to do exactly that.

There are three main avenues of improvement which I have pursued in the development of this code: efficient parallelization of calculations, memory optimization, and concurrent file output. I have implemented these improvements congruent with my requirements for extensibility:

1. Every `Integrator` must arbitrarily benefit from the three primary optimizations
2. `Vulcan` must support any integration scheme

3. `Vulcan` must support any arbitrary set of calculations, provided the necessary inputs can be encapsulated in a `Body` object

4. The set of calculations must be flexible between time steps.

5. `Vulcan` must support variable and fixed time steps.

6. `Vulcan` must support different time steps for different bodies.

To this end, I have implemented a structure wherein an arbitrary calculation can be added to the integration with as little as 5 lines of code (excluding the code for the calculation itself). Additionally, any new integration scheme (i.e. a new integration algorithm, a new combination of existing algorithms, or a new implementation of an existing scheme) can be added by simply sub-classing the abstract base class, `Integrator`, and will receive full benefit of the parallelization, memory management, and file output optimizations.

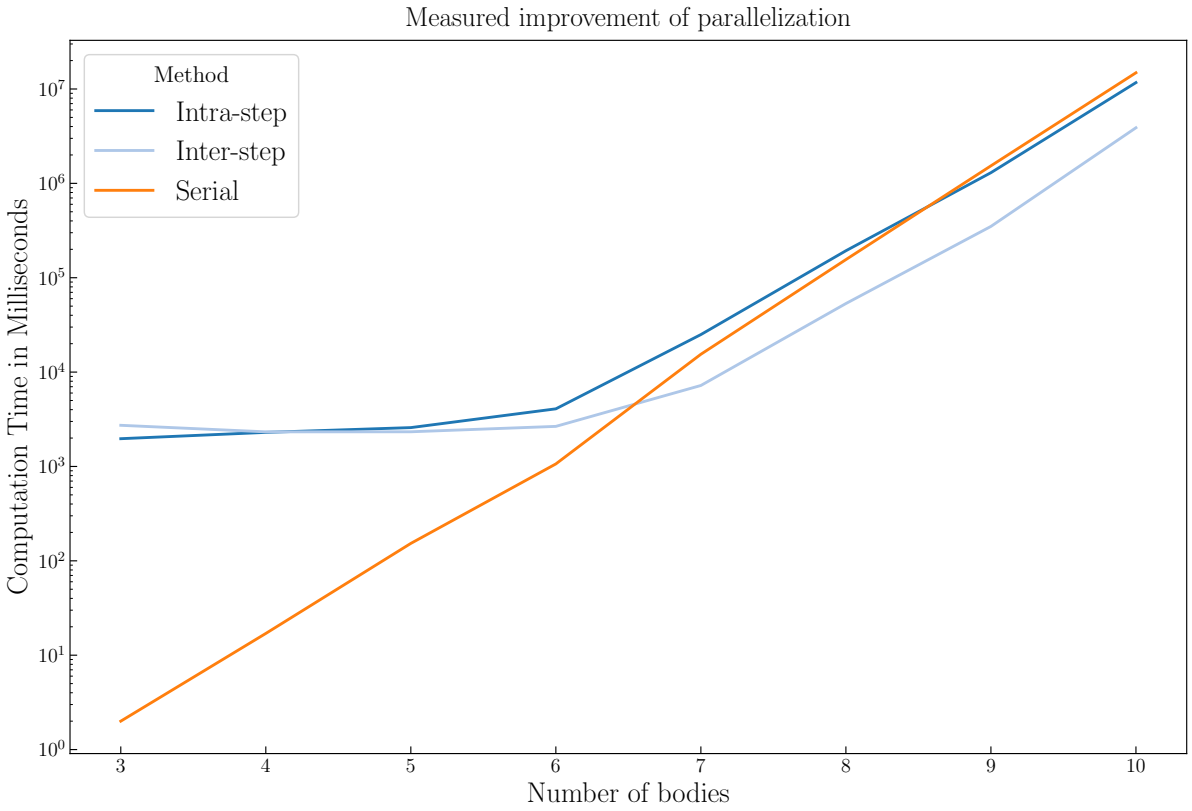

Figure 4.1: Realized changes in computation time for serial, intra-step parallelized, and inter-step parallelized execution of `Vulcan`'s Bulirsch-Stoer integrator. Testing conducted with memory optimizations disabled: parallel integrations with 6 bodies or fewer are memory-limited. The intra-step method does not surpass serial if fewer than 9 bodies are included. The inter-step method is always faster than serial when not memory-limited.

4.3.1 Parallelization

Efficient parallelization of the calculations demands inter-step thread continuity. If the threads are relinquished to the thread scheduler between steps they may be assigned to other tasks in the interim, bottle-necking the remaining threads to fewer logical cores or forcing a delay to allow completion of lower priority tasks. Furthermore, upon resumption the calculations may be redistributed, and calculations that switch cores will cause further slowdown. Thread schedulers are quite good at all of this, and in most cases there is no significant pessimization - which is in fact the foundation of the popular technique of thread-pooling (Garg et al., 2002). In cases where the expense of the calculation (or task) is roughly the same or less than normal thread-scheduling (such as small-scale N-body integration), a different technique is needed. When faced with such a calculation, thread pools will generally revert to serial execution, but then they sacrifice the potential advantages.

In `Vulcan`, the threads are forced into 'busy-waiting' between steps – something which is normally considered an anti-pattern (Gamma et al., 1995). This prevents them from being relinquished to the thread scheduler and guarantees they will not have to sleep and resume between calculations. This allows my code to better realize the full potential offered by parallelization compared to other parallelization techniques (pipes, OpenMP, threadpools, etc), even with a small number of bodies.

4.3.2 Memory Optimization

The utility of memory caching is most notable in cases where there is a discrepancy between the time steps of individual bodies. In cases which are suitable for the use of 'ghost' bodies (i.e. bodies who are affected by the massive bodies but do not affect them or each other (Lissauer et al., 2012)) this caching makes use of computing space which would otherwise be wasted, as shown in Figure 4.2.

The idea here is to cache the necessary values from each time step and use them to do the 'ghost' calculations while the main simulation progresses instead of waiting until all the calculations for 'ghosts' have been finished before advancing the simulation to the next step. In a use case such as that in (Lissauer et al., 2012), this would enable the simulation of 68 different Earth obliquities without any decrease in the overall speed of the simulation, and a twofold increase in the overall speed above that. This would either allow double the parameter space to be sampled, or halve the time needed to sample the same space.

4.3.3 Concurrent file output

The most expensive part of any orbital integration is the file IO. Write speed strongly restricts the temporal granularity of the results. Most prevalent integrators will write to file on a fixed simulation

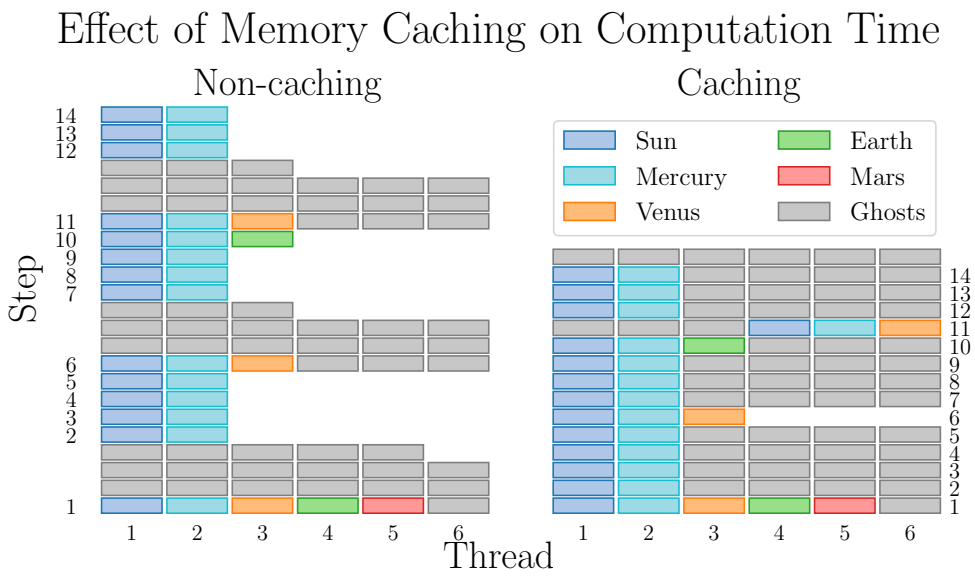

Figure 4.2: Illustration of the improvement of appropriate memory caching for an architecture whose memory scheduler can perform 6 parallel operations (typical for modern machines). The test system is comprised of our solar system and 12 'ghost' Earths. In this case, the caching version completes 14 steps for every 9 steps of the non-caching version.

time interval, or write to memory on a fixed interval and dump to file on a longer interval. In `Vulcan`, I employ a more advanced scheme: the parameters are written to memory on a given simulation time interval, but they are dumped to file on a given *wall time* interval, e.g. once every hour. These memory dumps are performed concurrent to the main simulation, meaning the simulation is only slowed down to a factor of $\frac{n_{\text{threads}}}{n_{\text{threads}} - 1}$ for the duration of the write.

Additionally, only the last time point in each memory dump is written with full precision - the rest use a much lower precision. This way even if a simulation is interrupted and needs restarting there will be no error due to loss of precision but the disk operations can be performed between 2 and 4 times faster, enabling a higher temporal granularity.

4.3.4 Realized Improvement

The real test of any theory is the result of its application in practice, and `Vulcan` passes this test with flying colors. Figure 4.3 shows the performance increase from parallelization for different architectures for a solar-system Bulirsch-Stoer integration. It bears noting that the Intel i7-8750H and AMD Ryzen 5 2060 are hyperthreading, and the AMD Opterons are not. The Intel i7-8750H and AMD Ryzen 5 2600 were built for processing speed, and sacrifice memory speed. The AMD Opteron systems are older, but were better optimized for memory speed – so while their raw performance is worse, they leverage the

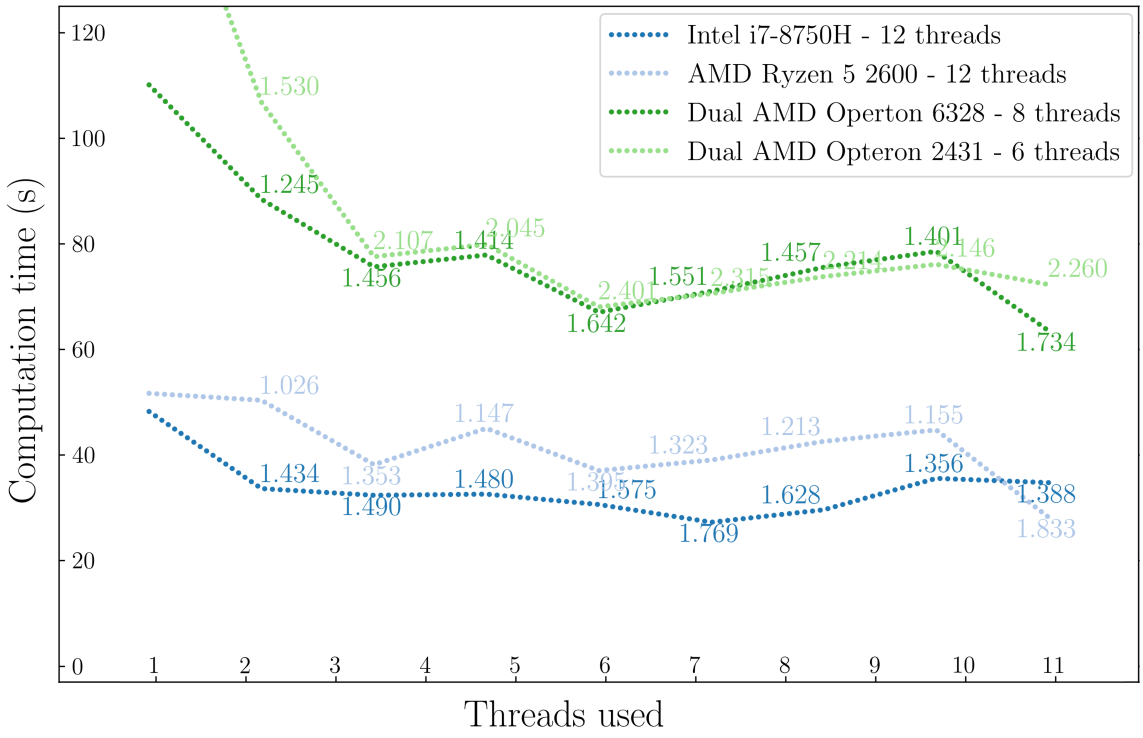

Figure 4.3: Comparison of the performance of `Vulcan`'s Bulirsch-Stoer integrator simulating 50,000 years of the solar system across several architectures.

parallelization better. With a system built for memory speed, the realized performance improvement will increase even further.

4.4 Results

Of course, the true measure of any orbital integrator is not speed, but accuracy. After all, what good is getting an answer more quickly if that answer is wrong? The most important and useful metric for the accuracy of an orbital integrator is the relative energy error of the system, i.e. the change in energy relative to the initial energy:

$$\Delta E = \frac{E - E_0}{E_0} \tag{4.1}$$

where E is the energy at time t and E_0 is the energy at time $t = 0$.

In a perfect integrator, the net change would be zero across any timescale. In practice, relative energy errors on the order of 10^{-6} are generally sufficient since the physical precision is generally limited to 10^{-7} (unless specific efforts are made to employ greater than 64-bit precision). Both Chambers (1999) (`MERCURY`) and Duncan et al. (1998b) (`SyMBA`) benchmarked their accuracy against a modified outer solar

system model in which the masses of the four outer planets are increased by a factor of 50. This is a very chaotic system, as Chambers (1999) notes, and even small variations in the initial parameters or in the time-step result in drastically different outcomes. This makes it a perfect system on which to test `Vulcan`.

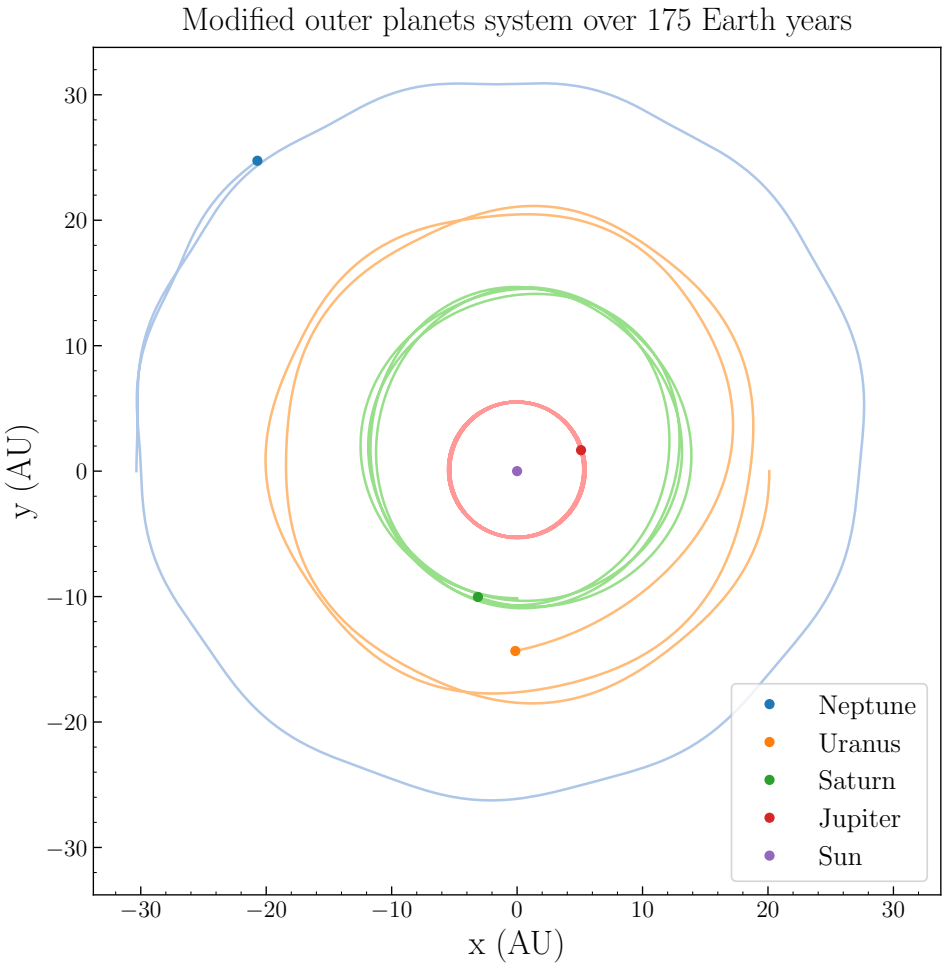

Figure 4.4: 175 Earth year integration of the five-body, modified outer planets system, where the masses of the four outer planets have been increased by a factor of 50. Uranus has just experienced a close encounter with Saturn which will lead to it's ejection within 25 Earth years.

Figure 4.4 shows the evolution of this system using `Vulcan`'s Bulirsch-Stoer integrator over a period of 175 Earth years. 175 years is too short to benchmark the energy error, but with this configuration Uranus is ejected at around 200 Earth years due to a close encounter with Saturn so the 175-year version is more suitable for illustration. The accuracy benchmark completed 2000 Earth years of integration time. Longer integration would not be meaningful for this system because Uranus, Neptune, and Saturn are all ejected by the 2000 year mark. The absolute maximum relative energy error for this integration was 3.0×10^{-7} — on par with the results from both `MERCURY` and `SyMBA`. More importantly, as with both

MERCURY and SyMBA, Vulcan's Bulirsch-Stoer integrator did not demonstrate any accumulation of energy error.

4.5 Conclusion

At present, I have fully implemented and rigorously tested both a Runge-Kutta and a Bulirsch-Stoer integrator which satisfy all the design requirements outlined in 4.3. The performance and results from the Bulirsch-Stoer integrator are discussed in 4.2, 4.3.4, and 4.4.

While the Bulirsch-Stoer integrator provides a meaningful proof-of-concept for the optimizations described herein, it offers no advantage over the established standard of MERCURY's hybrid symplectic integrator and, generally speaking, the MERCURY integrator should be preferred over Vulcan's Bulirsch-Stoer. The same cannot necessarily be said of MERCURY's Bulirsch-Stoer integrator, which suffers a tremendous speed disadvantage: Vulcan's Bulirsch-Stoer integrator took 38 days to complete a gigayear simulation of our solar system on an 8-core, 3.6 GHz Intel i7-9700K, compared to 27 days for MERCURY's hybrid symplectic integrator. Chambers (1999) states that symplectic integrators are roughly an order of magnitude faster than their numerical counterparts. Since this integration features very few close encounters, it is reasonable to expect that the MERCURY's Bulirsch-Stoer integrator would require 8 to 9 months to complete a gigayear[1].

Consequently, Vulcan needs a hybrid symplectic integrator before competing with MERCURY and rigorous testing thereof before being applied to scientific investigation. Said integrator and its testing suite will be the next developments in the Vulcan code.

[1] I was unable to measure this directly, for obvious reasons

Chapter 5 Summary

5.1 Solving the Alhazen-Ptolemy Problem

I have presented solutions to the Alhazen-Ptolemy problem for the one-finite and both-finite cases under a useful formulation created in collaboration with Jason Barnes and Shannon MacKenzie. These solutions are immediately useful for simulating radiative transfer on Titan with the `SRTC++` model, and I have integrated an implementation of them into that code. The locations of specular points returned by my solutions are both more accurate and take less time to compute than any alternative method; critical for enabling `SRTC++` to model specular reflections.

The modifications I have made to `SRTC++` will allow us to compensate for the adjacency effect in future investigations and facilitate the analysis of ethane composition in Titan's lakes – which was the direct motivation for this work.

5.2 Modernizing Orbital Integration

I have modernized orbital integration with the `Vulcan` code base, offering all existing orbital integration schemes the benefits of efficient parallelization even with small numbers of bodies, modern memory management, and concurrency in outputting results.

I have fully implemented and rigorously tested a Bulirsch-Stoer integrator which has demonstrated proof-of-concept for the aforementioned optimizations and which has realized a near ten-fold improvement in computational efficiency over the Bulirsch-Stoer integrator featured in the `MERCURY` code. `Vulcan`'s Bulirsch-Stoer beats both `MERCURY`'s hybrid symplectic integrator and `SyMBA`'s numerical integrator in terms of relative energy error for a modified outer-planets test system while competing with the hybrid integrator in terms of computation time. `Vulcan` will enable future orbital integrations to cover significantly larger parameter spaces or the same spaces with much higher granularity.

5.3 Compiling Cassini-VIMS Titan Data

I have compiled Titan data from Cassini's VIMS instrument into summary data products and integrated the process for creating products compliant with Version 4 of the PDS standard into the `Jcube` code base. These composite maps span different coverages, interpolation styles, and meshing methods to ensure that

they support the broadest feasible range of analyses.

The global views provide large-scale context and allow for global comparisons across different flybys, the regional views provide summaries of observations whose resolution is too good to be accurately depicted in the global context but whose extent goes beyond that of the local views. The local views put the highest resolution data into context without sacrificing spatial resolution to do so.

Lanczos interpolations provide the highest spatial resolution, but sacrifice some spectral resolution while nearest neighbor interpolations preserve the spectral resolution without enhancing the spatial resolution. Sequential replacement meshes give higher resolution but lower signal-to-noise while coadd meshes give better signal-to-noise at the cost of lower resolution.

The delivery of these products with enable future researchers to conduct meaningful scientific investigations without paying the time costs of developing code to process the raw data.

References

Barnes, J. W., Brown, R. H., Radebaugh, J., et al. 2006, Geophysical Research Letters, 33

Barnes, J. W., Radebaugh, J., Brown, R. H., et al. 2007, Journal of Geophysical Research: Planets, 112

Barnes, J. W., Brown, R. H., Soderblom, L., et al. 2008, Icarus, 195, 400

Barnes, J. W., Soderblom, J. M., Brown, R. H., et al. 2011, Icarus, 211, 722

Barnes, J. W., Sotin, C., Soderblom, J. M., et al. 2014, Planetary science, 3, 3

Barnes, J. W., MacKenzie, S. M., Young, E. F., et al. 2018, The Astronomical Journal, 155, 264

Bolmont, E., Raymond, S. N., Leconte, J., Hersant, F., & Correia, A. C. 2015, Astronomy & Astrophysics, 583, A116

Chambers, J. E. 1999, Monthly Notices of the Royal Astronomical Society, 304, 793

Dolbeau, R. 2018, The Journal of Supercomputing, 74, 1341

Duncan, M. J., Levison, H. F., & Lee, M. H. 1998a, The Astronomical Journal, 116, 2067

—. 1998b, The Astronomical Journal, 116, 2067

Elkin, J. M. 1965, The Mathematics Teacher, 58, 194

Fujimura, M., Hariri, P., Mocanu, M., & Vuorinen, M. 2019, Computational Methods and Function Theory, 19, 135

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. 1995, Design patterns: elements of reusable object-oriented software (Pearson Education India)

Garg, R. P., Sharapov, I. A., & Sharapov, I. 2002, Techniques for optimizing applications: high performance computing (Sun Microsystems Press Palo Alto)

Glaeser, G. 1999, Journal for Geometry and Graphics, 3, 121

González, A. B., Martın, P., & López, D. J. 2000, Applied numerical mathematics, 35, 1

Griffith, C. A., Penteado, P., Baines, K., et al. 2005, science, 310, 474

Hernandez, D. M. 2016, Monthly Notices of the Royal Astronomical Society, 458, 4285

Hubber, D. A., Batty, C. P., McLeod, A., & Whitworth, A. P. 2011, Astronomy & Astrophysics, 529, A27

Hubber, D. A., Rosotti, G. P., & Booth, R. A. 2018, Monthly Notices of the Royal Astronomical Society, 473, 1603

Inakage, M. 1986, The Visual Computer, 2, 379

Karkoschka, E., & Schroder, S. E. 2016, Icarus, 270, 260

Katz, V. J. 1995, Mathematics Magazine, 68, 163

Kay, S., Hedley, J. D., & Lavender, S. 2009, Remote sensing, 1, 697

Kiselev, V., Bulgarelli, B., & Heege, T. 2015, Remote Sensing of Environment, 157, 85

Levison, H. F., & Duncan, M. J. 1994, Icarus, 108, 18

Lissauer, J. J., Barnes, J. W., & Chambers, J. E. 2012, Icarus, 217, 77

Minomura, M., Kuze, H., & Takeuchi, N. 2001, Journal of the Remote Sensing Society of Japan, 21, 260

Neumann, P. M. 1998, The American Mathematical Monthly, 105, 523

Odell, A., & Weinman, J. 1975, Journal of Geophysical Research, 80, 5035

Peláez, J., Hedo, J. M., & de Andrés, P. R. 2007, Celestial Mechanics and Dynamical Astronomy, 97, 131

Potter, D. P., Stadel, J., & Teyssier, R. 2017, Computational Astrophysics and Cosmology

Prakash, W., Varma, A., & Bhandari, S. 1994, Computers & Geosciences, 20, 1467

Rein, H., & Liu, S.-F. 2012, Astronomy & Astrophysics, 537, A128

Riede, H. 1989, Praxis Math, 31, 65

Rodriguez, S., Le Mouélic, S., Rannou, P., et al. 2009, Nature, 459, 678

Schwarze, J. 1990, in Graphics gems, Academic Press Professional, Inc., 404–407

Smith, J. D. 1992, The Mathematical Gazette, 76, 189

Southwell, B. J., & Dempster, A. G. 2018, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 11, 639

Springel, V. 2005, Monthly Notices of the Royal Astronomical Society, 364, 1105

Stephan, K., Jaumann, R., Brown, R. H., et al. 2010, Geophysical Research Letters, 37

Sterckx, S., Knaeps, E., & Ruddick, K. 2011, International Journal of Remote Sensing, 32, 6479

Stoer, J., & Bulirsch, R. 1980, Introduction to Numerical Mathematics, Springer, Berlin, Germany

Tanre, D., Herman, M., & Deschamps, P. 1981, Applied Optics, 20, 3676

Waldvogel, J. 1992, Elemente der Mathematik, 47, 108

Weisstein, E. W. 2002, Alhazen's Billiard Problem, Wolfram Research, Inc.

Wisdom, J. 2017, Monthly Notices of the Royal Astronomical Society, 464, 2350

Wolfram Research, I. 2020, Mathematica, Version 12.1