Uncertainties in the Determination of Jupiter's Core

Alexander C. English
Mentor: David J. Stevenson

Abstract

This summer, the Juno spacecraft completed its five-year journey to our solar system's largest planet after a successful Jupiter Orbital Insertion (JOI) on Independence Day. As the probe continues to orbit Jupiter for the remainder of its mission, it will be streaming data back to Earth to better our understanding of the gas giant. Our focus is on learning more about Jupiter's interior—specifically, the radius and mass of its core (defined as "the central concentration of elements heavier than hydrogen and helium"[1]), about which we know very little[2]. Juno's Gravity Science Experiment (GSE) employs an instrument that will detect slight discrepancies in the spacecraft's orbit around Jupiter in order to measure the distribution of matter within the planet. This will give us a value for Jupiter's moment of inertia (α), currently unknown, to about 0.2% accuracy. Combining this measurement with Jupiter's observables—the precisely known mean radius (R) and total mass (M) of the whole planet—will allow us to narrow the range of possible values that the core's radius and mass can take on. However, this will depend on the accuracy to which we know the equation of state of the materials that comprise most of Jupiter. In other words, this project quantifies the extent to which uncertainties in the equation of state affect the determination of Jupiter's core. This analysis is of interest because it will provide a better understanding of Jupiter's structure and give insight into the processes by which the planet formed, thereby teaching us about the origins of our solar system.

---

1. The Equation of State and Density Function

An equation of state describes how a substance's atoms interact with each other, expressing the state of matter that arises from specific physical circumstances. Jupiter is made up primarily of hydrogen, whose atoms repel each other against the force of gravity, resulting in hydrostatic equilibrium, giving the planet its shape and external gravity field. Hydrogen has a very complex equation of state[3], but coincidentally it is not very different from an equation called the n = 1 polytrope, which relates pressure (P) to density (ρ) by the following: $P = K\rho^2$, where K is some constant. This equation only applies in the region of Jupiter external to the core, known as the envelope, which can be treated as a fluid in hydrostatic equilibrium (i.e. at rest), giving rise to a general solution for the density

function: $\rho(x) = A\frac{sinx}{x} + B\frac{cosx}{x}$, where $x \equiv kr$, and $k \equiv \sqrt{2\pi G/K}$ (G is the gravitational constant). By requiring that hydrostatic equilibrium be maintained at the border of the core and envelope, as well as the density function equaling zero when evaluated at Jupiter's radius, we can solve for A and B, and derive formulas for the core mass (m_c), envelope mass (m_e), and α depending only on our choice of k and the core radius (r_c).

2. Comparing Jupiter's Gravitational Harmonic and Moment of Inertia

For context, it should be explained that Juno is not only measuring Jupiter's moment of inertia, but also its gravitational harmonic (J_2), a separate yet similar property of the planet. However, J_2 is already known to much greater precision than is moment of inertia, so it is of interest to see how these two values behave in relation to each other. To do this, we can use lambda ($\Lambda_{2,0}$), an attribute that is in one-to-one correspondence with J_2 and for which we have the necessary equations to solve in terms of only k and r_c. Next, we need a reference value for lambda that will be kept fixed, which, based on many current models, should correspond to a core mass of about 10 Earth masses and a very small core radius. Note that while this value of lambda is probably fairly close to Jupiter's actual lambda, the exact number is less important than the determination of how the moment of inertia changes when lambda is kept constant. In my approach, I first choose reasonable radii for Jupiter's core (one-third of Jupiter's radius, one-fourth, etc.) and then determine what value of k results in the same reference value of lambda for each core radius. Next, I take my paired values of k and r_c and plug them into my previous equations to determine the moment of inertia and the core mass for each case.

r_c	k	K	α	m_c
R/3	4.32069*10^{-8}	222766	0.247286	3.00436*10^{26}
R/4	4.35222*10^{-8}	221004	0.248913	1.66540*10^{26}
R/5	4.36060*10^{-8}	220441	0.249291	1.16295*10^{26}
R/6	4.36348*10^{-8}	220218	0.249374	9.33123*10^{25}
R/7	4.36465*10^{-8}	220117	0.249380	8.12835*10^{25}
R/8	4.36519*10^{-8}	220067	0.249367	7.43486*10^{25}
R/9	4.36546*10^{-8}	220042	0.249349	7.00532*10^{25}
R/10	4.36560*10^{-8}	220028	0.249332	6.72419*10^{25}
R/12	4.36573*10^{-8}	220015	0.249305	6.39470*10^{25}
R/15	4.36579*10^{-8}	220009	0.249279	6.16173*10^{25}
R/16	4.36580*10^{-8}	220008	0.249274	6.11655*10^{25}
R/18	4.36581*10^{-8}	220007	0.249264	6.05180*10^{25}
R/20	4.36581*10^{-8}	220007	0.249257	6.00894*10^{25}
R/25	4.36582*10^{-8}	220006	0.249247	5.94860*10^{25}
0	4.36582*10^{-8}	220006	0.249227	5.86775*10^{25}

Figure 1: Table of values in SI units for fixed $\Lambda_{2,0}$ = 0.154949

Changing k in order to keep lambda constant is akin to slightly changing the composition of Jupiter's envelope, most likely by altering the heavy element makeup of the envelope. As the core radius increases, one can see from the table that the core mass does as well, which makes sense because if it behaved otherwise, the envelope mass would have to be the same or greater, dramatically changing J_2 and α. It follows that α should increase as the core radius increases because by definition, this model assumes that the core has a constant density, thus a larger core would necessarily result in a less central concentration of mass. This holds true for smaller core radii, but the trend reverses somewhere around R/7. But more important than these observations is *how much* α changes, and a quick glance reveals that this change is on the order of a few hundredths of a percent. Only when the core radius becomes very large, probably unreasonably so, do we see more significant changes in α. This not only confirms to us that J_2 and α are reasonably good approximations of each other (which we already knew), but it also means that changes in α are not very dependent on the core radius.

3. Perturbing the Equation of State

Thus far, our analysis has been "unperturbed," meaning that it has relied solely on the simple n = 1 polytrope. However, the reality is that we don't precisely know Jupiter's equation of state, which is much more complex[4]. To account for this, we will employ perturbation theory by adding a small correction term to our density function in order to simulate a change in the equation of state[5]. The new density function then becomes $\rho(x) = A\frac{sinx}{x} + B\frac{cosx}{x} + \frac{1}{x}\int_{x_c}^{x} g(z)\sin(x-z)dz,$ where $x \equiv kr,$ $x_c \equiv kr_c,$ $g(z) \equiv \frac{1}{z}\frac{d}{dz}\left(\frac{z^2}{\rho_0}\frac{d}{dz}(y(z){\rho_0}^2)\right)$, ρ_0 is the unperturbed density, and $y(z)$ is a small correction function of choice. Now, if we once again force the density to equal zero at Jupiter's radius and make the core mass and envelope mass add up to the total mass of the planet, we can solve for A and B and consequently for m_c, m_e, and α. Next, we can change $y(z)$ for each pair of values of k and r_c that kept lambda constant in the unperturbed case in order to keep α unchanged. For this reason, it makes sense to choose a $y(z)$ that has an adjustable parameter, such as β in $(sinz + \beta cosz)/100$. The idea here is ultimately to use Juno's measurement of Jupiter's moment of inertia in order to examine the range of possible radii and masses for Jupiter's core. In other words, we are trying to understand what differing models for the *interior* of the planet would result in the same *exterior* value of its moment of inertia, which is what Juno will give us. On the next page is a plot of a density profile comparing the unperturbed function to the perturbed, as well as a table showing what $y(z)$ functions keep α the same as the unperturbed case and the resultant perturbed core masses.

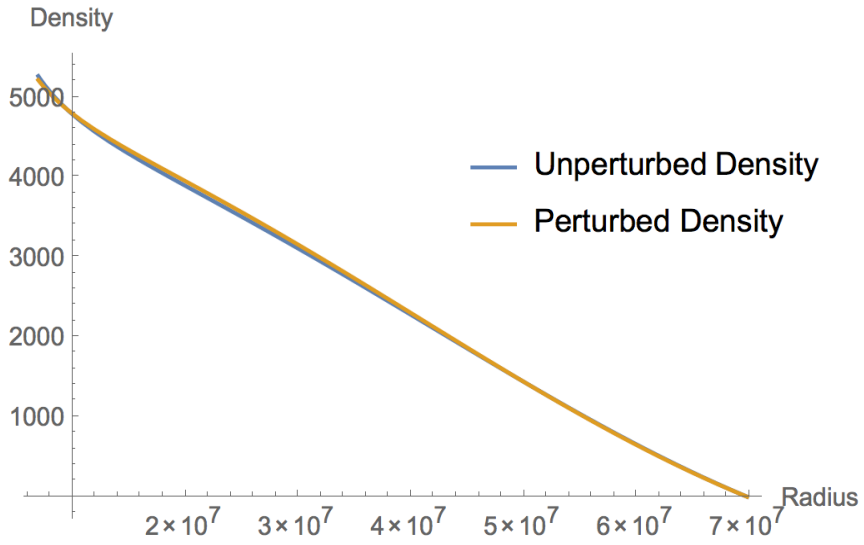


Figure 2 (left): Unperturbed and perturbed plots of density vs. radius outside the core for k = 4.36560*10^{-8}, r_c = R/10, β = 1.0100 (SI units)

Figure 3 (below): Table of values in SI units for $y(z)$ and resulting m_c, keeping α fixed from the unperturbed case

r_c	k	α	$y(z)$	m_c unperturbed	m_c perturbed
R/3	4.32069*10^{-8}	0.247286	(sin(z)-0.4621*cos(z))/100	3.00436*10^{26}	3.03311*10^{26}
R/4	4.35222*10^{-8}	0.248913	(sin(z)+0.5841*cos(z))/100	1.66540*10^{26}	1.61031*10^{26}
R/5	4.36060*10^{-8}	0.249291	(sin(z)+0.8490*cos(z))/100	1.16295*10^{26}	1.08187*10^{26}
R/6	4.36348*10^{-8}	0.249374	(sin(z)+0.9391*cos(z))/100	9.33123*10^{25}	8.43241*10^{25}
R/7	4.36465*10^{-8}	0.249380	(sin(z)+0.9763*cos(z))/100	8.12835*10^{25}	7.19628*10^{25}
R/8	4.36519*10^{-8}	0.249367	(sin(z)+0.9940*cos(z))/100	7.43486*10^{25}	6.48946*10^{25}
R/9	4.36546*10^{-8}	0.249349	(sin(z)+1.0041*cos(z))/100	7.00532*10^{25}	6.05502*10^{25}
R/10	4.36560*10^{-8}	0.249332	(sin(z)+1.0100*cos(z))/100	6.72419*10^{25}	5.77229*10^{25}
R/12	4.36573*10^{-8}	0.249305	(sin(z)+1.0167*cos(z))/100	6.39470*10^{25}	5.44350*10^{25}
R/15	4.36579*10^{-8}	0.249279	(sin(z)+1.0211*cos(z))/100	6.16173*10^{25}	5.21330*10^{25}
R/16	4.36580*10^{-8}	0.249274	(sin(z)+1.0215*cos(z))/100	6.11655*10^{25}	5.16877*10^{25}
R/18	4.36581*10^{-8}	0.249264	(sin(z)+1.0229*cos(z))/100	6.05180*10^{25}	5.10564*10^{25}
R/20	4.36581*10^{-8}	0.249257	(sin(z)+1.0239*cos(z))/100	6.00894*10^{25}	5.06411*10^{25}
R/25	4.36582*10^{-8}	0.249247	(sin(z)+1.0250*cos(z))/100	5.94860*10^{25}	5.00588*10^{25}
0	4.36582*10^{-8}	0.249227	(sin(z)+1.0270*cos(z))/100	5.86775*10^{25}	4.92953*10^{25}

As the plot shows, the perturbed density is slightly greater on average than its unperturbed counterpart (also, note how the density goes to zero as the radius approaches Jupiter's radius). Remember that these functions are only valid in the envelope, meaning that the extra envelope mass that arises from perturbing the density has to be made up for by a less massive core, which is indeed the case in the table (for all core radii except R/3, which is probably around where this model begins to break down). More interesting is the trend that β in the $y(z)$ function follows throughout the table. While it decreases consistently (up until the very large core radii, at which point it begins to drop more dramatically), this change is on the order of a few percent or even less, meaning that this model is not very dependent on the choice of core radius.

4. A Physically Motivated Model

Now that we've gone to the trouble of creating a model with a customizable perturbation to the equation of state, it's a good idea to make a strategic choice for $y(z)$ to better represent a model of Jupiter's interior. We let $y(z) = 0.015(1 + z - 0.2z^2)/(1 + e^{5(z-2.5)})$ to denote a thermal correction to the equation of state due to a temperature difference inside Jupiter of about 1000K at 1 megabar pressure (and at higher pressures, but not lower)[1]. This perturbation signifies a deviation from published models, and results in the following table when α is kept at its unperturbed value.

r_c	k unperturbed	k perturbed	α	m_c unperturbed	m_c perturbed
R/6	$4.36348*10^{-8}$	$4.28870*10^{-8}$	0.249374	$9.33123*10^{25}$	$8.16853*10^{25}$
R/8	$4.36519*10^{-8}$	$4.28794*10^{-8}$	0.249367	$7.43486*10^{25}$	$6.31780*10^{25}$
R/10	$4.36560*10^{-8}$	$4.28732*10^{-8}$	0.249332	$6.72419*10^{25}$	$5.63880*10^{25}$
R/12	$4.36573*10^{-8}$	$4.28688*10^{-8}$	0.249305	$6.39470*10^{25}$	$5.33010*10^{25}$

Figure 4: Table of values in SI units for perturbed k and m_c arising from the $y(z)$ perturbation over mid-range core radii

Because $y(z)$ is fixed, the only parameter that can be changed to keep α constant is k, which decreases on average by a couple percent for mid-range core radii. This is accompanied by a more significant decrease in the core mass of about 10-20% (more substantial than in the previous perturbation). Keep in mind that the core's contribution to the mass of the entire planet is only on the order of a few percent, which is why a fairly large change in core mass relative to the core does not have much of an effect on α.

Discussion

Although more investigation is required in order to fully understand how changes in the equation of state trade off with the radius and mass of Jupiter's core, this analysis is a good start. We now have a way to create a perturbation of choice to the equation of state and examine how this affects the core, which was the initial goal of this project. However, this approach is only a model, and can always be improved upon for greater accuracy. One such correction would be to treat Jupiter as a non-uniform rotating body, which would complicate the physics involved but be more realistic. Though Juno will measure Jupiter's moment of inertia to greater precision than is currently known, it has been demonstrated that—depending on the envelope's equation of state—the core can take on a wide range of masses and radii. To narrow this range, further examination of the heavy element makeup of Jupiter's core and envelope and the equation of state will be required. But it will be well worth it to better understand the creation of Jupiter and our solar system as a whole.

Acknowledgements

I want to acknowledge my mentor, Professor David Stevenson, for guiding me through this project. His oversight was essential in my progress and success, especially at times when I was stuck. Dave does not coddle you, and his firm insistence on you forging your own path through solving a problem is the surest way to learn how to do research. Dave often knows your capabilities better than yourself, which is a true testament to his talent and experience, as well as to his role as a mentor.

I also want to thank the California Institute of Technology's Summer Undergraduate Research Fellowships program for funding my research stipend and providing a wonderful summer of helpful events and fun activities. Caltech's campus truly feel like a home away from home.

References

1. Stevenson, D. J. Unpublished. Limits on Determining the Core of Jupiter.
2. Guillot, T., D. J. Stevenson, W. B. Hubbard, and D. Saumon 2004. The interior of Jupiter. In *Jupiter - The Planet, Satellites and Magnetosphere*.
3. Saumon, D., G. Chabrier, and H. M. van Horn 1995. An Equation of State for Low Mass Stars and Giant Planets. *Astrophys. J. Suppl.* 99, 713.
4. Hubbard, W. B., and B. Militzer 2016. A Preliminary Jupiter Model. In *The Astrophysical Journal.*
5. Stevenson, D. J. Unpublished. P41B-2072, Limits to the Core Mass of Jupiter.

Appendix

The appendix consists of a Mathematica notebook detailing the math behind my equations and how I used them in approaching this project. It is on the following pages.

SURF 2016
Uncertainties in the Determination of Jupiter's Core
Student: Alec English
Mentor: David Stevenson
Paper: Stevenson, D. J. Unpublished. Limits on Determining the Core of Jupiter.

Part I: Calculating values of k and rc for a fixed value of lambda of Jupiter.
Starting with the n = 1 polytrope for the equation of state of Jupiter (P = Kρ^2), I succeed in developing a formula for Jupiter's lambda (Λ2,0), that depends only on the radius of the core (rc) and the model parameter in the equation of state (K). From there, I can choose a reasonable reference value for lambda that corresponds to a fractional change in Jupiter's radius of about 0.03, and then vary the core radius and K to keep this reference value the same. I then calculate the resulting core mass (mc, which is also only dependent on the core radius and K) and the moment of inertia for each pair of values and compile them.

Constants (gravitational, mean radius of Jupiter, mass of Jupiter):

In[1]:=
```
G = 6.674 * 10^(-11)
```
Out[1]= 6.674×10^{-11}

In[2]:=
```
R = 6.98 * 10^7
```
Out[2]= 6.98×10^{7}

In[3]:=
```
M = 1.9 * 10^27
```
Out[3]= 1.9×10^{27}

Model parameter (K) in the equation of state as a function of k (Equation 4). K and k can directly be derived from each other. I use Kk[k_] because K alone is a special symbol in Mathematica.

In[4]:=
```
Kk[k_] := 2 * Pi * G / (k^2)
```

A and B coefficients in the *unperturbed case* as functions of k. These were derived by solving the equation involving Jupiter's radius and delta (giving B in terms of A), and the equation involving Jupiter's mass and the A squared + B squared term (which is why both M and R are present in these functions). Note that B as a function of k includes the A function within it, due to it be subsituted back into the radius and delta equation once A is known.

Below, I test that my A and B functions match the values I computed by hand for the k value corresponding to a delta of 0.03 to make sure my function is working.

In[5]:=
```
A[k_] := M * (k^2) * Cos[Pi - (k * R)] / (4 * Pi * R)
```

In[6]:=
```
B[k_] := A[k] * Tan[Pi - (k * R)]
```

```
A[4.36582 * 10^(-8)]
```

```
4110.44
```

```
B[4.36582 * 10^(-8)]
```

```
388.561
```

Core mass as a function of k and core radius. From here on out, all the equations will be dependent on these two variables. This equation is the exact same one I calculated by hand using the continuity equation, except with A and B also dependent on k, instead of the above constants (because delta is no longer constant).

In[7]:=
```
mc[k_, rc_] := (4 * Pi / (k^3)) (A[k] * Sin[k * rc] -
    (A[k] * k * rc * Cos[k * rc]) + B[k] * Cos[k * rc] + (B[k] * k * rc * Sin[k * rc]))
```

This “rho” function represents $\Delta\rho/\rho c$ found in equation A10 of the appendix (because this term appears a lot and is complicated enough for this to be useful). Determined simply by expanding the expression and simplifying the result.

In[8]:=
```
rho[k_, rc_] := 1 - ((rc^3) (M - mc[k, rc]) / (mc[k, rc] * ((R^3) - (rc^3))))
```

Recurrence formulas for the derivatives of spherical bessel functions (because I couldn’t find out how/was too lazy to find a Mathematica shortcut for this). These are directly out of that textbook you showed me, nothing too fancy.

In[9]:=
```
j2prime[x_] := SphericalBesselJ[1, x] - (3 * SphericalBesselJ[2, x] / x)
```

In[10]:=
```
y2prime[x_] := SphericalBesselY[1, x] - (3 * SphericalBesselY[2, x] / x)
```

“Theta” represents the huge nasty coefficient attached to A2 in appendix equation A10. Notice how it contains my “rho” equation, as well as the spherical bessel functions and their derivatives.

In[11]:=
```
theta[k_, rc_] :=
  (k * rc * j2prime[k * rc] - ((2 - 3 * rho[k, rc]) (SphericalBesselJ[2, k * rc])) /
   ((2 - 3 * rho[k, rc]) * (SphericalBesselY[2, k * rc]) - k * rc * (y2prime[k * rc])))
```

“Phi” represents the coefficient attached to q in equation A10. It also contains rho and the spherical bessels. By substituting in theta and phi, solving for J2 and q using equations A11 and A12 is more manageable.

In[12]:=
```
phi[k_, rc_] := (((rho[k, rc]) ((k * rc)^2) / ((k * R)^2))) /
   ((2 - 3 * rho[k, rc]) * SphericalBesselY[2, k * rc] - (k * rc * y2prime[k * rc]))
```

The “mu” and “delta” functions represent the denominators of the 2 expressions for A2 that equations A11 and A12 give rise to once everything else thus far has been plugged in and rearranged. Note how they both contain the theta function above.

In[13]:=
```
mu[k_, rc_] := SphericalBesselJ[2, k * R] + (theta[k, rc] * SphericalBesselY[2, k * R])
```

In[14]:=
```
delta[k_, rc_] := 3 * k * R (j2prime[k * R] + (theta[k, rc] * y2prime[k * R]))
```

By setting the 2 expressions for A2 equal to each other, we can rearrange them into a single expression for lambda (J2/q), which involved the mu, delta, and phi functions, and thus all previously defined functions in this notebook somewhere inside them. This is the function we ultimately care about, and for which we will vary k and rc to keep a constant value.

In[15]:=
```
lambda[k_, rc_] := ((mu[k, rc] (2 + (3 * phi[k, rc] * k * R * y2prime[k * R]))) -
    (delta[k, rc] ((1 / 3) + phi[k, rc] * SphericalBesselY[2, k * R]))) /
  ((9 * mu[k, rc]) + delta[k, rc])
```

Our reference value for lambda (what we want it to always equal), is in the limiting case when core radius is 0 and k corresponds to a delta of 0.03. You can't actually use 0 for rc because a divide by zero error occurs, so instead we make it negligibly small. This also confirms that the lambda function works, because the answer equals the answer you get when you use the simple equation that arises from this limiting case (equation 21).

Now, what I did was set reasonable values of the core radius in terms of R (the mean radius of the whole planet) ranging from R/3 (way too big) to 0, and then by trial and error found the value of k that gave the right value of lambda up to 6 decimal places. Here, we just see that last case, for which we already by definition know what k and rc are due to the limiting case.

In[93]:=
```
lambda[4.36582 * 10^(-8), 0.00000000000000001]
```

Out[93]=
```
0.154949
```

Alpha is the moment of intertia function (equation 18) divided by (M*R^2), which is the coefficient that we care about. Instead of A and B being constants, they also are allowed to vary, so the integral is a bit more complicated than before. Here, I took each pair of k and rc values that resulted in the right lambda, and evaluated alpha and core mass (a function that I already defined above for use in lambda) for those values, inputting the results into a table on word. Left here are just the values during the limiting case, which does allow for rc to be set exactly to 0.

In[17]:=
```
alpha[k_, rc_] :=
  ((2 / 5) mc[k, rc] * (rc^2) + (8 * Pi / (3 * (k^5))) * Integrate[(A[k] * (x^3) * Sin[x]) +
      (B[k] * (x^3) * Cos[x]), {x, k * rc, k * R}]) / (M * (R^2))
```

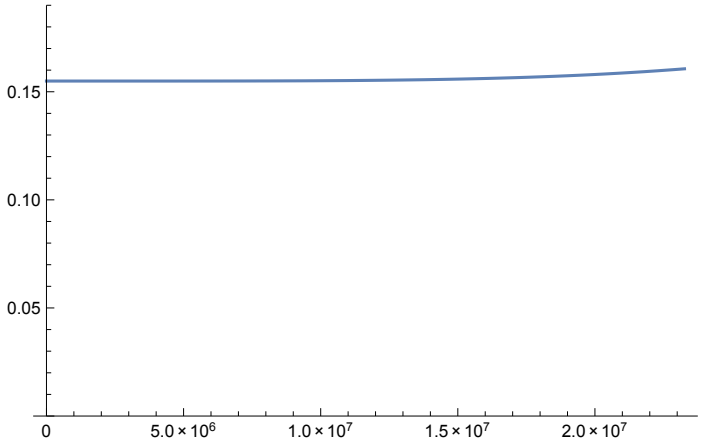
In[18]:=
```
alpha[4.36582 * 10^(-8), 0.00000000000000001]
```

Out[18]=
```
0.249227
```

In[19]:=
```
mc[4.36582 * 10^(-8), 0.00000000000000001]
```

Out[19]=
```
5.86775 × 10^25
```

```
Plot[lambda[4.3658200*^-8, rc], {rc, 0, R / 3}, PlotRange → {0, 0.19}]
```



Here is the function for envelope mass (Equation 9), which can be calculated now that the A and B functions are defined.

In[21]:=
```
me[k_, rc_] :=
  (4 * Pi / (k^3)) NIntegrate[(A[k] * x * Sin[x]) + (B[k] * x * Cos[x]), {x, k * rc, k * R}]
```

To check my work, I made a formula for the total mass of Jupiter (which should always add up to our constant M), that simply adds the core mass (mc) and the envelope mass (me) together. This function should always return 1.9*10^27 as its result, no matter what values of k and rc are chosen, and indeed it does exactly that.

In[22]:=
```
m[k_, rc_] := mc[k, rc] + me[k, rc]
```

In[23]:=
```
m[1000, 1000]
```

Out[23]= 1.9×10^{27}

Finally, I defined my density function (equation 5).

In[117]:=
```
Rho1[k_, r_] := (A[k] * Sin[k * r] / (k * r)) + (B[k] * Cos[k * r] / (k * r))
```

Part II: Perturbing the density function for modeling Jupiter's interior

I succeed in creating a new density function based on a perturbation of choice (which is why this analysis is useful) to the n = 1 polytrope equation of state on Mathematica. This perturbation is represented by g(z), which is determined by a small correction function, y(z) (37, 38), that I choose to be of the form (sinz + beta*cosz)/100, where beta is a parameter that I can vary to fit my results. Next, I set to work finding what values of beta dresult in a moment of inertia measurement for the perturbed density profile that is the same for each paired value of core radius (rc) and the model parameter (K) from the previous, unperturbed case. The idea here is to combine the precisely known total mass and radius of Jupiter with Juno's measurement of the planet's moment of inertia in order to examine the range of possible radii and masses for Jupiter's core. In other words, I am trying to understand what differing models for the interior of the planet would result in the same exterior value of its moment of inertia,

which is what Juno will give us.

Equation (38) in your paper, this represents the unperturbed density function which is used to determine g(z). Note how the function is dependent on both k and z because k uniquely determines A and B in the unperturbed case.

In[24]:=
```
rho0[k_, z_] := (A[k] * Sin[z] / z) + (B[k] * Cos[z] / z)
```

A y(z) function of choice! This is what makes this notebook useful. y represents a slight change in the equation of state which perturbs the whole density equation. y ranges from 0 to Pi and should take on values of a few percent in that range. It is helpful to create a y function with a parameter that you can vary aside from simply changing its amplitude. The parameter here is the coefficient in front of cos(z).

In[132]:=
```
y[z_] := (Sin[z] + 1.01 * Cos[z]) / 100
```

Once y is chosen, g(z) can be determined by the following equation based on (38). g is dependent on both y and rho0, so it is a function of k and z.

In[61]:=
```
g[k_, z_] := (1 / z) (Dt[((z^2) / rho0[k, z]) * Dt[y[z] * ((rho0[k, z])^2), z], z])
```

Now we will set about solving for the perturbed values of A and B. To do this, we must solve two equations. The first is that the total mass of Jupiter is equal to the core mass plus the envelope mass, and the second is that the density evaluated at Jupiter's outer radius equals zero.

S represents the term attached to A that results from solving (39) for the core mass, expanding and factoring the terms for A and B independent of each other.

In[27]:=
```
S[k_, rc_] :=
  (4 * Pi / (k^3)) (((k * rc)^2) * y'[k * rc] + (1 - 2 * y[k * rc]) ((k * rc)^3) / 3)
```

T represents the term attached to B when solving (39) for the core mass.

In[28]:=
```
T[k_, rc_] :=
  (4 * Pi / (k^3)) ((k * rc) * y'[k * rc] + (1 - 2 * y[k * rc]) (1 + ((k * rc)^2) / 2))
```

U represents the extra term that arises from plugging the perturbed density function (37) into the equation for envelope mass (9). When this is done and the envelope mass is calculated, the first two terms (the ones attached to A and B) remain the same, but U is added on to the end. Because Mathematica seems to have trouble directly calculating double integrals of that complexity (which is what arises when (37) goes into (9)), I split U into U1 and U2, which represent two applications of integration by parts on the resulting equation to eliminate the double integral and make Mathematica able to calculate specific values.

In[73]:=
```
U1[k_, rc_] := (4 * Pi / (k^3))
   (((Sin[k * R] - (k * R) Cos[k * R]) * NIntegrate[Cos[z] * g[k, z], {z, k * rc, k * R}]) -
     NIntegrate[(Sin[x] - x * Cos[x]) * (Cos[x] * g[k, x]), {x, k * rc, k * R}])
```

```
In[72]:= U2[k_, rc_] := (4 * Pi / (k^3)) (((((k * R) * Sin[k * R] + Cos[k * R]) *
            (-NIntegrate[Sin[z] * g[k, z], {z, k * rc, k * R}])) +
         NIntegrate[(x * Sin[x] + Cos[x]) (Sin[x] * g[k, x]), {x, k * rc, k * R}])

In[74]:= U[k_, rc_] := U1[k, rc] + U2[k, rc]
```

V represents the term attached to A when the envelope mass is calculated and the terms are factored.

```
In[76]:= V[k_, rc_] :=
  (4 * Pi / (k^3)) (Sin[k * R] - (k * R) Cos[k * R] - Sin[k * rc] + (k * rc) Cos[k * rc])
```

W represents the term attached to B when the envelope mass is calculated and the terms are factored.

```
In[77]:= W[k_, rc_] :=
  (4 * Pi / (k^3)) ((k * R) Sin[k * R] + Cos[k * R] - (k * rc) Sin[k * rc] - Cos[k * rc])
```

P represents the term attached to A when the density (equation 37) is calculated at Jupiter's outer radius, and Q represents the term attached to B.

```
In[78]:= P[k_] := Sin[k * R]

In[79]:= Q[k_] := Cos[k * R]
```

Omega is the term that the sum of P and Q equal.

```
In[80]:= Omega[k_, rc_] := -NIntegrate[g[k, z] * Sin[(k * R) - z], {z, k * rc, k * R}]
```

Now that the previous functions have been defined, I can solve for the perturbed A and B coefficients in the density equation using algebra. Note that A and B are now dependent on both k and rc, unlike in the unperturbed case. Also, the A function uses the B function in its definition for simplicity. Below, I test the same reference value for k that I used to test my A and B functions in the unperturbed case, with the core radius value that resulted in my reference value of lambda in the unperturbed case. The results differ slightly from before.

```
In[81]:= B[k_, rc_] := ((M * P[k]) - (U[k, rc] * P[k]) - (Omega[k, rc] * (S[k, rc] + V[k, rc]))) /
    ((T[k, rc] * P[k]) + (W[k, rc] * P[k]) - (Q[k] * (S[k, rc] + V[k, rc])))

In[82]:= A[k_, rc_] := (Omega[k, rc] / P[k]) - ((Q[k] / P[k]) * (B[k, rc]))

In[99]:= A[4.36582 * 10^(-8), 0.0001]

Out[99]= 4325.59

In[97]:= B[4.36582 * 10^(-8), 0.001]

Out[97]= 333.274
```

Here is the approximate core mass function for the perturbed case (Equation 39). I can do this now that A and B are known. Unlike the perturbed case, where I calculated the core mass exactly, this is not an exact formula because the derivation becomes so nasty when the equation of state is altered that it is not worth analyzing perfectly when the core mass represents only a few percent of the total mass of the

planet.

In[102]:=
```
Mc[k_, rc_] :=
  (4 * Pi / (k^3)) * (y'[k * rc] * (A[k, rc] * ((k * rc)^2) + B[k, rc] * (k * rc)) +
     (1 - 2 * y[k * rc]) * (B[k, rc] * (1 + ((k * rc)^2) / 2) + A[k, rc] * ((k * rc)^3) / 3))
```

I1 summed with I2 represents the double integral term that arises when the perturbed density function is inserted into (equation 23) when calculating the moment of inertia of Jupiter. It is split in two because I performed an integration by parts, just like I did with my U function, though this is more complicated application than in the previous case. Ia, Ib, Ic, Id, Ie, and Ig (because If is a special command in Mathematica) are simply components of the full answer that are used for simplicity to keep track of everything.

In[108]:=
```
Ia[k_, rc_] :=
 3 ((k * R)^2) * Sin[k * R] - 6 * Sin[k * R] + 6 (k * R) * Cos[k * R] - ((k * R)^3) * Cos[k * R]
```

In[109]:=
```
Ib[k_, rc_] := NIntegrate[Cos[z] * g[k, z], {z, k * rc, k * R}]
```

In[110]:=
```
Ic[k_, rc_] := NIntegrate[
   (3 (x^2) * Sin[x] - 6 * Sin[x] + 6 * x * Cos[x] - (x^3) * Cos[x]) * Cos[x] * g[k, x],
   {x, k * rc, k * R}]
```

In[111]:=
```
I1[k_, rc_] := (8 * Pi / (3 * (k^5))) ((Ia[k, rc] * Ib[k, rc]) - Ic[k, rc])
```

In[112]:=
```
Id[k_, rc_] :=
  ((k * R)^3) * Sin[k * R] - 6 (k * R) * Sin[k * R] - 6 * Cos[k * R] + 3 ((k * R)^2) * Cos[k * R]
```

In[113]:=
```
Ie[k_, rc_] := -NIntegrate[Sin[z] * g[k, z], {z, k * rc, k * R}]
```

In[114]:=
```
Ig[k_, rc_] := NIntegrate[
   ((x^3) * Sin[x] - 6 x * Sin[x] - 6 * Cos[x] + 3 * (x^2) * Cos[x]) * Sin[x] * g[k, x],
   {x, k * rc, k * R}]
```

In[115]:=
```
I2[k_, rc_] := (8 * Pi / (3 * (k^5))) ((Id[k, rc] * Ie[k, rc]) + Ig[k, rc])
```

Now I can define my new alpha function, which is just (equation 23) for the perturbed case. From here, I performed lots of trial and error calculations to find which values of beta in my original y(z) function kept alpha the same as in the unperturbed case for the same paired values of k and rc that kept lambda at its constant reference value in the unperturbed case. I also calculated the perturbed core mass once I had beta to see how much it changed from the unperturbed case to keep alpha constant.

In[116]:=
```
Alpha[k_, rc_] := ((2 / 5) * Mc[k, rc] * (rc^2) + ((8 * Pi / (3 * (k^5))) *
      NIntegrate[(A[k, rc] * (x^3) * Sin[x]) + (B[k, rc] * (x^3) * Cos[x]),
       {x, k * rc, k * R}]) + I1[k, rc] + I2[k, rc]) / (M * (R^2))
```

Here is a perturbed envelope mass equation based on previous algebra.

In[119]:=
```
Me[k_, rc_] := (A[k, rc] * V[k, rc]) + (B[k, rc] * W[k, rc]) + U[k, rc]
```

And to check my work, as before, I make a total mass of Jupiter equation that adds the core mass to the envelope mass and should always equal 1.9*10^27.

In[121]:=
```
Mj[k_, rc_] := Mc[k, rc] + Me[k, rc]
```

In[122]:=
```
Mj[4.36560 * 10^(-8), R / 10]
```

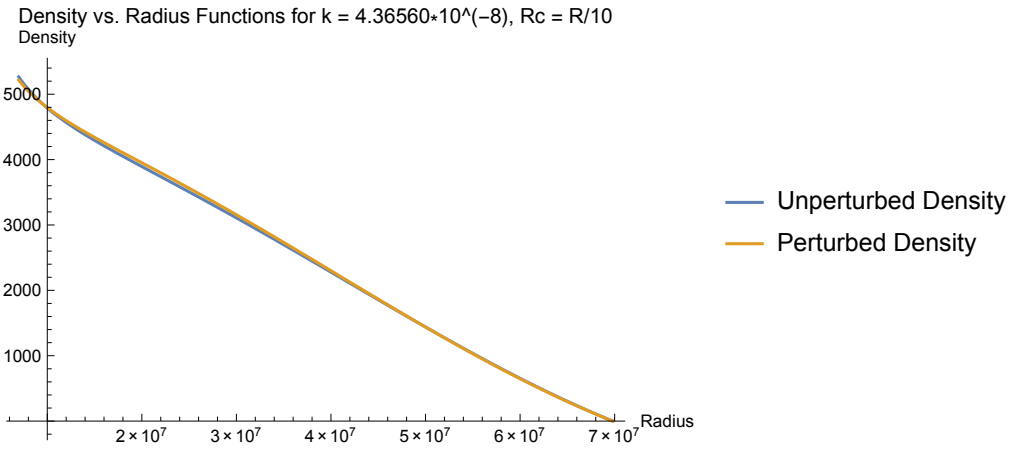
Out[122]= 1.9×10^{27}

Finally, I define my perturbed density function (equation 37).

In[118]:=
```
Rho2[k_, rc_, r_] := (A[k, rc] * Sin[k * r] / (k * r)) + (B[k, rc] * Cos[k * r] / (k * r)) +
  ((NIntegrate[g[k, z] * Sin[(k * r) - z], {z, k * rc, k * r}]) / (k * r))
```

In[133]:=
```
Plot[{Rho1[4.36560 * 10^(-8), r], Rho2[4.36560 * 10^(-8), R / 10, r]}, {r, R / 10, R},
 PlotLabel → "Density vs. Radius Functions for k = 4.36560*10^(-8), Rc = R/10",
 PlotLegends → {"Unperturbed Density", "Perturbed Density"},
 AxesLabel → {"Radius", "Density"}]
```

Out[133]=



Here is a physically motivated y(z) for which I also examined alpha. In this case, I changed k to make sure alpha remained unchanged. A plot of g(z) with the perturbed k value is included.

In[135]:=
```
y[z_] := 0.015 (1 + z - 0.2 z z) / (1 + Exp[5 (z - 2.5)])
```

In[137]:=
```
g[4.28732 * 10^(-8), z]
```

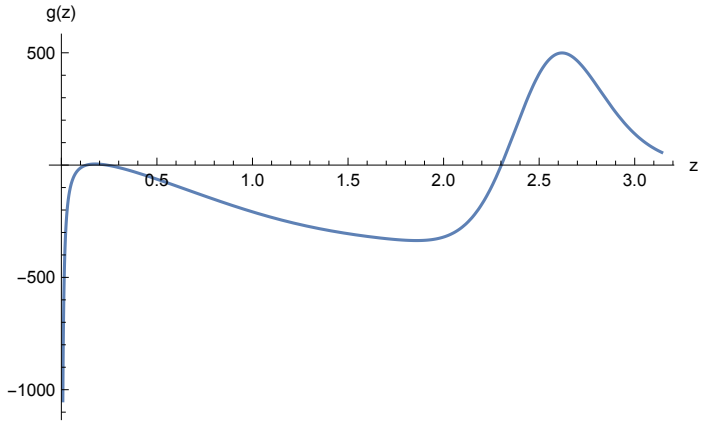
Out[137]=
$$\frac{1}{z}\left(\frac{1}{\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}}\right.$$
$$z^2\left(\left(\left(0.03\,\left(1+z-0.2\,z^2\right)\left(-\frac{591.24\,\text{Cos}[z]}{z^2}+\frac{3937.48\,\text{Cos}[z]}{z}-\frac{3937.48\,\text{Sin}[z]}{z^2}-\right.\right.\right.\right.$$
$$\left.\left.\left.\left.\frac{591.24\,\text{Sin}[z]}{z}\right)^2\right)\Big/\left(1+e^{5\,(-2.5+z)}\right)+\frac{1}{1+e^{5\,(-2.5+z)}}\right.$$
$$0.03\,\left(1+z-0.2\,z^2\right)\left(\frac{1182.48\,\text{Cos}[z]}{z^3}-\frac{7874.96\,\text{Cos}[z]}{z^2}-\frac{591.24\,\text{Cos}[z]}{z}+\right.$$
$$\left.\frac{7874.96\,\text{Sin}[z]}{z^3}+\frac{1182.48\,\text{Sin}[z]}{z^2}-\frac{3937.48\,\text{Sin}[z]}{z}\right)$$
$$\left.\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)+\frac{1}{1+e^{5\,(-2.5+z)}}0.06\,(1-0.4\,z)\right)$$

$$\left(-\frac{591.24\,\text{Cos}[z]}{z^2}+\frac{3937.48\,\text{Cos}[z]}{z}-\frac{3937.48\,\text{Sin}[z]}{z^2}-\frac{591.24\,\text{Sin}[z]}{z}\right)$$
$$\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)-\frac{1}{\left(1+e^{5\,(-2.5+z)}\right)^2}0.3\,e^{5\,(-2.5+z)}\left(1+z-0.2\,z^2\right)$$
$$\left(-\frac{591.24\,\text{Cos}[z]}{z^2}+\frac{3937.48\,\text{Cos}[z]}{z}-\frac{3937.48\,\text{Sin}[z]}{z^2}-\frac{591.24\,\text{Sin}[z]}{z}\right)$$
$$\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)-$$
$$\left(0.006\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/\left(1+e^{5\,(-2.5+z)}\right)-$$
$$\left(0.15\,e^{5\,(-2.5+z)}\,(1-0.4\,z)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/\left(1+e^{5\,(-2.5+z)}\right)^2+$$
$$\left(0.75\,e^{10\,(-2.5+z)}\left(1+z-0.2\,z^2\right)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/\left(1+e^{5\,(-2.5+z)}\right)^3-$$
$$\left(0.375\,e^{5\,(-2.5+z)}\left(1+z-0.2\,z^2\right)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/$$
$$\left(1+e^{5\,(-2.5+z)}\right)^2\Bigg)-$$
$$\left(z^2\left(-\frac{591.24\,\text{Cos}[z]}{z^2}+\frac{3937.48\,\text{Cos}[z]}{z}-\frac{3937.48\,\text{Sin}[z]}{z^2}-\frac{591.24\,\text{Sin}[z]}{z}\right)\right.$$
$$\left(\frac{1}{1+e^{5\,(-2.5+z)}}0.03\left(1+z-0.2\,z^2\right)\left(-\frac{591.24\,\text{Cos}[z]}{z^2}+\frac{3937.48\,\text{Cos}[z]}{z}-\right.\right.$$
$$\left.\left.\frac{3937.48\,\text{Sin}[z]}{z^2}-\frac{591.24\,\text{Sin}[z]}{z}\right)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)+\right.$$
$$\left(0.015\,(1-0.4\,z)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/\left(1+e^{5\,(-2.5+z)}\right)-$$
$$\left.\left(0.075\,e^{5\,(-2.5+z)}\left(1+z-0.2\,z^2\right)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/\right.$$
$$\left.\left(1+e^{5\,(-2.5+z)}\right)^2\right)\Bigg)\Big/$$
$$\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2+\frac{1}{\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}}$$
$$2\,z\left(\frac{1}{1+e^{5\,(-2.5+z)}}0.03\left(1+z-0.2\,z^2\right)\right.$$
$$\left(-\frac{591.24\,\text{Cos}[z]}{z^2}+\frac{3937.48\,\text{Cos}[z]}{z}-\frac{3937.48\,\text{Sin}[z]}{z^2}-\frac{591.24\,\text{Sin}[z]}{z}\right)$$
$$\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)+$$
$$\left(0.015\,(1-0.4\,z)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/\left(1+e^{5\,(-2.5+z)}\right)-$$
$$\left(0.075\,e^{5\,(-2.5+z)}\left(1+z-0.2\,z^2\right)\left(\frac{591.24\,\text{Cos}[z]}{z}+\frac{3937.48\,\text{Sin}[z]}{z}\right)^2\right)\Big/$$

$$\left(1 + e^{5\,(-2.5+z)}\right)^2\Bigg)\Bigg)$$

In[138]:=
```
Plot[%126, {z, 0, Pi}, AxesLabel → {"z", "g(z)"}]
```

Out[138]=



Summary of useful equations:

-Unperturbed A and B coefficients: A[k_], B[k_]
-Perturbed A and B coefficients: A[k_, rc_], B[k_, rc_]
-Unperturbed core mass: mc[k_, rc_]
-Perturbed core mass: Mc[k_, rc_]
-Lambda: lambda[k_, rc_]
-Unperturbed alpha: alpha[k_, rc_]
-Perturbed alpha: Alpha[k_, rc_]
-Unperturbed envelope mass: me[k_, rc_]
-Perturbed envelope mass: Me[k_, rc_]
-Unperturbed total mass: m[k_, rc_]
-Perturbed total mass: Mj[k_, rc_] (because M is already defined as a constant)
-Unperturbed rho function: Rho1[k_, r_]
-Perturbed rho function: Rho2[k_, rc_, r_]
-Correction functions to perturbed equation of state: y[z_], g[k_, z_]

Remember that nearly all of these functions are dependent on an input value of k and rc. This may seem contrary to much of Professor Stevenson's paper, wherein the symbol x is often used to represent k*r, which pops up a lot. However, for the purposes of my analysis thus far, much of which has been a trial and error examination of how the variables behave apart from one another, it has been more useful to have k and r separated in these equations. They are not obsolete if you are using the x notation, though; you just need to separate x into k and r.