Computational Modeling of the Detection of Extra-Solar Planets

Introduction

The purpose of this project was to build several different functions and scripts that allow the user to partially if not completely model a star system containing an orbiting planet, and to simulate two of the major ways of detecting these extra-solar planets. The first major portion of this project is a function that simulates the orbit of the two bodies around their center of mass. The second member of this project takes the position and any other relevant data from the orbit plotting function and imports this into another function that calculates the radial velocity of the star along each point in its orbit. Then the function outputs a graph of the observed velocity, which represents the Doppler shift in the stars spectra as it travels towards and away from us. The final piece of this project takes information about the planet and graphs a visualization of the transit. The transit is the measured decrease in photons received from the star as the planet passes between our planet and its host star. The overall goal of this project was to build three working functions that correctly simulate the observations made by astronomers as well as be able to correctly predict characteristics of undiscovered planets.

Analysis

This project contained many challenging pieces, which were mainly due to the difficulty of the physics involved. The orbital plotting function and the radial velocity function contained many principles from celestial mechanics and were difficult to complete without a relevant textbook. I was able to find many different websites and references in other texts to the material that I needed and was able to reconcile enough material to build a model.

The orbital plotting function went through several stages so that the complexity was added slowly and manageably. Initially, the function simply graphed two spheres of different radii, one stationary and the other orbiting the first in a circular orbit. I achieved this using spherical coordinates, which I converted back to Cartesian in order to graph, to build the mesh spheres and then parameterized the orbiting one using the polar description of a circle. This made the orbiting model look more realistic and gave a basic frame for the actual physics driven model of the system. This was accurate for planets in circular orbits, but the user was required to specify many things and it was not generalized for the large portion of elliptically orbiting planets.

The next step in creating an accurate plotting function was to generalize the function for elliptical orbits, and to include the physical parameters of the star and planet to get an accurate description of the system. Using the eccentricity and period of the orbit (e and T respectively), the following description of the orbit may be used:

(1)

(2)

These equations define the orbit of the planet as an ellipse orbiting its star. Equation 2 is Kepler’s third law of planetary motion. The angle is called the true anomaly, which is the angle between the closest point in the planets orbit to its star and its current location. Equation 1 defines the position of the planet as a distance r and an angle away from the star.

This more accurately represents the system, but in reality both the planet and the star are orbiting their combined center of mass. This adds a whole other level of complexity to the problem. The first issue is the calculation of the center of mass of the system, which is relatively simple. Taking the center of mass to be the origin of the coordinate system, r to be the distance from the origin to the planet, R to be the distance from the origin to the star, a as the distance between them, m as the mass of the planet and M as the mass of the star we can derive a simple relation detailing the locations of the two bodies:

(3)

(4)

Now using a concept labeled reduced mass we can calculate the orbits of the two bodies. This problem is called a two-body problem, and the simplest way to solve it is by simplifying it to a one-body problem. We can do this by creating a fictional system that contains characteristics of the original system. We replace the star with a star that has a mass that is equal to the sum of the mass of the planet and the star. The planet orbiting this new star has a mass that is the reduced mass. The reduced mass is defined as:

(5)

This new system is essentially a solution to our original problem, but it has transformed it into the model where the planet is orbiting the star instead of the center of mass. Equation 1 is actually this representation because the measured period of the planet is actually the period of this reduced mass planet, and since the equation is mass independent of mass it remains unchanged. In this new system, r actually represents the distance between the two orbiting bodies. So substituting back into equations 3 and 4 , where a is the distance between the two bodies, we obtain expressions for the orbits of both the planet and its parent star relative to the systems center of mass. This method simulates our system with very little error.

The radial velocity function takes information passed off from the orbit plotting function and uses it to calculate the radial velocity of the star as it travels along its path. This is useful because it is one of the methods used for detecting extra-solar planets. The velocity of the star is not always directly toward us, it is always tangent to its path, and we can detect this because the stars light gets Doppler shifted by its motion. As the star moves toward us, we see the light at a higher frequency, and then as the star moves away we see the light at a lower frequency. This yields a curve of the stars velocity relative to our position that is sinusoidal in nature. We may derive an expression for the radial velocity of the star by using Newton’s third law and centripetal acceleration. From this we get the equation:

(6)

Here is the mass of the planet, is the mass of the star, i is the angle of inclination, is the semi-major axis of the planets orbit, and G is the standard gravitational constant. Using this relation we may calculate the peak radial velocity of the star. Now we may multiply this by a sinusoidal function of the true anomaly because it actually varies if it is traveling in an orbit. This may be only an approximation of the graph for a planet traveling in an elliptical orbit, but it is a good approximation for planets with only slightly eccentric orbits.

The transit function illustrates another method of detecting extra-solar planets. This function takes the radii of the planet and the star and creates an animation of the planet passing in front of the star and a plot of the star’s brightness versus time. In practice, astronomers look at stars and if they see a periodic and significant decrease in brightness, this can be caused by a transiting extra-solar planet. This only occurs if the planet passes between our planet and its own star. Also, this method tends to detect very large planets that orbit very closely to their parent star. While this function was not a particularly difficult concept physics wise, it involved a lot of very messy geometry and trigonometry. The geometric description was the best way to go about calculating the area shared by the two circles as the planet circle passes across the edge of the star. The two areas where this occurred had very difficult descriptions for the enclosed area. The area had to be broken up into two different sections because I was unable to find one description for the entire interface. The two equations are used either before of after the highest point on the smaller circle has passed into the bigger circle. I will not show these equations here because of their length and complexity. For all of the relevant equations and derivations for these two algorithms, see the attached work.

One of the elements that I would have liked to add if I had more time would be a way to include the impact parameter. The impact parameter is the distance away from the central axis of the star that the planet is transiting. Because the distance between the star and the planet is so great, it can be assumed that the path across the surface of the star is linear. Using the impact parameter, the transit function could be generalized more to include transits that are in orbital planes that are not parallel to our line of sight, which is extremely common. This generalization would be lengthy and difficult using the method that I have developed.

Conclusion

Overall I was able to accomplish what I set out to do with tis project. I built a set of functions that accurately describes the orbit of a two body system, and simulates the data one would retrieve when attempting to detect the orbiting planet. There are several aspects of these functions require special attention. Because I was unable to include the impact parameter in the transit function, the transit for certain planets could be drastically misrepresented. This is due to the fact that the impact parameter has a very large effect on the time of the transit and even the depth of the transit. If the impact parameter is large, it shortens the transit time by a quite a bit, and if it makes it so that the planet never completely passes into the interior of the stars cross-section, the depth of the transit will be less. That being said, as long as the impact parameter isn’t relatively large, it will only have an affect on the duration of the transit. In the radial velocity function, the derivation that I found for the radial velocity of the star used centripetal acceleration, and I am unsure as to if it correctly represents elliptical orbits as well. One other puzzling part of this equation is that it yields a maximum value that is very close to the measured value, but it is slightly off. The calculated value is about 10 m/s off of what the measured value is. In the database where the information about these planets is listed it quotes a possible error value of plus or minus 11 m/s but other than that I was unable to determine the cause of this. For the majority of the testing, I used the exoplanet TReS 3b. This planet is a characteristic sample of the current exoplanet database. With a little more time and expertise, these functions could have very accurately represented all of the different aspects of exoplanet detection.

References

<http://msemac.redwoods.edu/~darnold/math50c/mathjax/spherical/index.xhtml>

<http://en.wikipedia.org/wiki/Elliptic_orbit>

[http://exoplanets.org](http://exoplanets.org/)

<http://www.mathopenref.com/ellipsefoci.html>

[http://www.astro.cornell.edu/academics/courses/astro1101/java/Finding%20Exosolar%20Planets.htm](http://www.astro.cornell.edu/academics/courses/astro1101/java/Finding Exosolar Planets.htm)

<http://www.physics.nmt.edu/~rsonnenf/phys241/compmech_rev1.47.pdf>

<http://en.wikipedia.org/wiki/Reduced_mass>

<http://en.wikipedia.org/wiki/Two-body_problem>

Matlab Guide, 2nd Edition; Desmond J. Higham and Nicholas J. Higham

Code

%master script

%Runs all functions and does all complex calculations within functions.

%Also does any plotting

clear all

close all

%note: for most testing, I used the planet tres 3b, which is the current

%data enterd. The user should only need to do any sort of unit conversion

%when entering the radii of the two bodies and the mass of the two bodies.

%The mass already has the conversion from jupiter masses and solar masses included

%All other parameters can be taken dircetly from exoplanet.org

%Enter parameters here

%eccentricity

e=0;

%orbital period in days

T=1.31;

%mass of planet

m=1.91\*1.898e27;

%mass of star kg

M=0.915\*1.989e30;

%orbital inclination degrees

i=81.85;

%star radius in meters

rho=0.812\*6.955e8;

%planet radius in meters

rho1=1.336\*6.9911e7;

%time of transit(day)

timetransit=0.057;

%number of iterations(controls speed)

iterations=500;

%number of orbits i.e. 2pi is 1 orbit(I reccommend 6pi)

theta1=linspace(0,6\*pi,iterations);

%creates planetary spheres for plotting

[x,y,z,x1,y1,z1,C,C1]=planetBuilder(rho,rho1);

%caluclates motion of bodies

[rp,xp,yp,rs,xs,ys]=orbitCalculator(e,T,m,M,theta1);

%calculates radial velocity shift

[vs,maxvs]=radialVelocity(rp,m,M,i,e,theta1);

%autoscaling of axes

maximum=max(rp)\*1.2;

minimum=-maximum;

%creating figures

fig1=figure;

fig2=figure;

set(fig1,'OuterPosition',[100 100 800 800]);%makes larger plot

%sets black background

whitebg([0 0 0])

%plots orbits

for t=1:iterations

figure(fig1);

subplot(1,2,1)

title('Orbital Plot')

%planet plot

mesh(x1+xp(t),y1+yp(t),z1,C1)

%plot parameters

axis square

hold on

axis([minimum maximum minimum maximum minimum maximum])

grid on

%star plot

mesh(x+xs(t),y+ys(t),z,C)

%animates plot view to gain perspective on the orbit

if t<iterations/3

view(90,90\*((3\*t)/iterations))

else

view(90,90)

end

%plot parameters

hold off

axis square

grid on

title('Orbital Plot')

subplot(1,2,2)

title(' Relative Radial Velocity of Star')

hold on

plot(theta1(t),vs(t),'.','markersize',10)

pause(.00001)%animation does not work without pause

end

%plotting transit

if timetransit==0

else

depth=transit(rho,rho1,timetransit,fig2);

end

fprintf('Transit Depth: %f\n',depth)

fprintf('Maximum Radial Velocity: %f m/s\n',maxvs)

function [x,y,z,x1,y1,z1,C,C1]=planetBuilder(rho,rho1)

%planetBuilder creates the meshgrid spheres that represent the planet and

%its star. It accepts the radii of the two bodies, where rho is the radius

%of the planet and rho1 is the radius of the star and outputs the x y z

%mesh coordinates. C and C1 are the colors for the mesh spheres.

%build spheres

theta=linspace(0,2\*pi,40);

phi=linspace(0,pi,40);

[theta,phi]=meshgrid(theta,phi);

%star mesh sphere coordinates

x=rho\*sin(phi).\*cos(theta);

y=rho\*sin(phi).\*sin(theta);

z=rho\*cos(phi);

%planet mesh sphere coordinates

x1=rho1\*sin(phi).\*cos(theta);

y1=rho1\*sin(phi).\*sin(theta);

z1=rho1\*cos(phi);

%colors for planet and star

C=nan(40,40,3);%star red

C(:,:,1) = 1;

C(:,:,2) = 0;

C(:,:,3) = 0;

C1=nan(40,40,3);%planet green

C1(:,:,1) = 0;

C1(:,:,2) = 1;

C1(:,:,3) = 0;

end

function [rp,xp,yp,rs,xs,ys]=orbitCalculator(e,T,m,M,theta1)

%orbitCalculator takes the eccenticity, period, mass of the planet, mass of

%the star, and the parameter theta1 and calculates the positions of the

%planet and the star in their respective orbits with respect to their

%center of mass.

%builds location for movement of planet

a=(((T^2)/(365.25^2))^(1/3))\*1.49597871\*10^11;%%major axis of orbit(AU->m)

r=(a\*(1-e^2)./(1+e.\*cos(theta1)));%position of reduced mass planet

%planetary motion coordinates

rp=(M/(m+M)).\*r;%position in polar

xp=rp.\*cos(theta1);

yp=rp.\*sin(theta1);

%stellar motion coordinates

rs=-(m/(m+M)).\*r;%position in polar

xs=rs.\*cos(theta1);

ys=rs.\*sin(theta1);

end

function [vs,maxvs]=radialVelocity(rp,m,M,i,e,theta1)

%radialVelocity takes the position vector of the star, the mass of the

%planet and the mass of the star, the eccentricity of the orbit, the angle of inclination

%and createsa plot of the relative orbital

%semi-major axis of planet orbit also converts distance in meters to AU

ap=max(rp)/((1+e)\*149597871000);

mp=m/(1.898\*10^27);%mass of planet in mjupiter

ms=M/(1.989\*10^30);%mass of star in msun

vs=sin(theta1).\*sind(i)\*28\*(mp)/(((ap)^(1/2))\*((ms)^(1/2)));

maxvs=max(vs);

end

function [depth]=transit(R,r,T,figh)

%transit plots an animation of a planet of radius r transiting a star of

%radius R as well as a plot of the stars intesity over time. T is the

%time ellapsed during the orbit. figh is the handle of the figure that the

%transit will be plotted in.

%time correlation of data points

timestep=linspace(0,T,1000);

%auto scale setting

minimum=(R^2)\*pi-(r^2)\*pi;

lower=minimum/((R^2)\*pi);

min1=-2\*R-1;

max1=2\*R+1;

%motion of planet

x1=linspace(min1,max1,1000);

%little circle plot

xr=linspace(-r,r,100);

yr1=sqrt(r^2-(xr).^2);

yr2=-sqrt(r^2-(xr).^2);

%big circle plot

xR=linspace(-R,R,100);

yR1=sqrt(R^2-(xR).^2);

yR2=-sqrt(R^2-(xR).^2);

for iter =1:998%must be shorter because of indexing

figure(figh);

clf

%plots big circle

subplot(1,2,1)

plot(xR,yR1,'r',xR,yR2,'r')

title('Transit')

hold on

axis square

axis([min1 max1 min1 max1])

%plots little circle

plot(xr+x1(iter+2),yr1,'b')

plot(xr+x1(iter+2),yr2,'b')

hold off

%calculating relexant intersection points and angles

xint=(R^2-r^2+x1(iter+2)^2)/(2\*x1(iter+2));

yint=sqrt(R^2-xint^2);

theta=atan(abs(yint/xint));

alpha=atan(abs(yint/(x1(iter+2)-xint)));

beta=atan(abs((x1(iter+2)-xint)/yint));

%calculating intensities

if x1(iter+2)<-(R+r)%to the left of star

intense(iter)=pi\*(R^2);

end

if x1(iter+2)>-(R+r)&&x1(iter+2)<-(R-r)%first intersection

if yint-yint0>0%checks rate of change of intersection of two cirles yint

iarea=theta\*(R^2)+alpha\*(r^2)-abs(yint\*xint)-abs(yint\*(x1(iter+2)-xint));

intense(iter)=pi\*(R^2)-iarea;

else

iarea=theta\*(R^2)-abs(xint\*yint)+abs((x1(iter+2)-xint)\*yint)+beta\*(r^2)+(pi/2)\*(r^2);

intense(iter)=pi\*(R^2)-iarea;

end

end

if x1(iter+2)>-(R-r)&&x1(iter+2)<(R-r)%inside star

intense(iter)=pi\*(R^2)-pi\*(r^2);

end

if x1(iter+2)>(R-r)&&x1(iter+2)<(R+r)%second intersection

if yint-yint0>0%checks rate of change of intersection of two cirles yint

iarea=theta\*(R^2)-abs(xint\*yint)+abs((x1(iter+2)-xint)\*yint)+beta\*(r^2)+(pi/2)\*(r^2);

intense(iter)=pi\*(R^2)-iarea;

else

iarea=theta\*(R^2)+alpha\*(r^2)-abs(yint\*xint)-abs(yint\*(x1(iter+2)-xint));

intense(iter)=pi\*(R^2)-iarea;

end

end

if x1(iter+2)>(R+r)%to the right of star

intense(iter)=pi\*(R^2);

end

%pecrengtage brightness value

intense1(iter)=intense(iter)/((R^2)\*pi);

%plots intesity vs time

hold on

subplot(1,2,2)

axis([0 T lower-.03 1.03])

plot(timestep(1:iter),intense1(1:iter),'-')

title('Relative Flux vs Time')

axis([0 T lower-.03 1.03])

xlabel('Time Elapsed')

hold off

%store y int for slope check

yint0=yint;

%time scaling for different regions

if intense1(iter)==1

pause(.000005)

else

pause(.005)

end

end

depth=1-min(intense1);

end