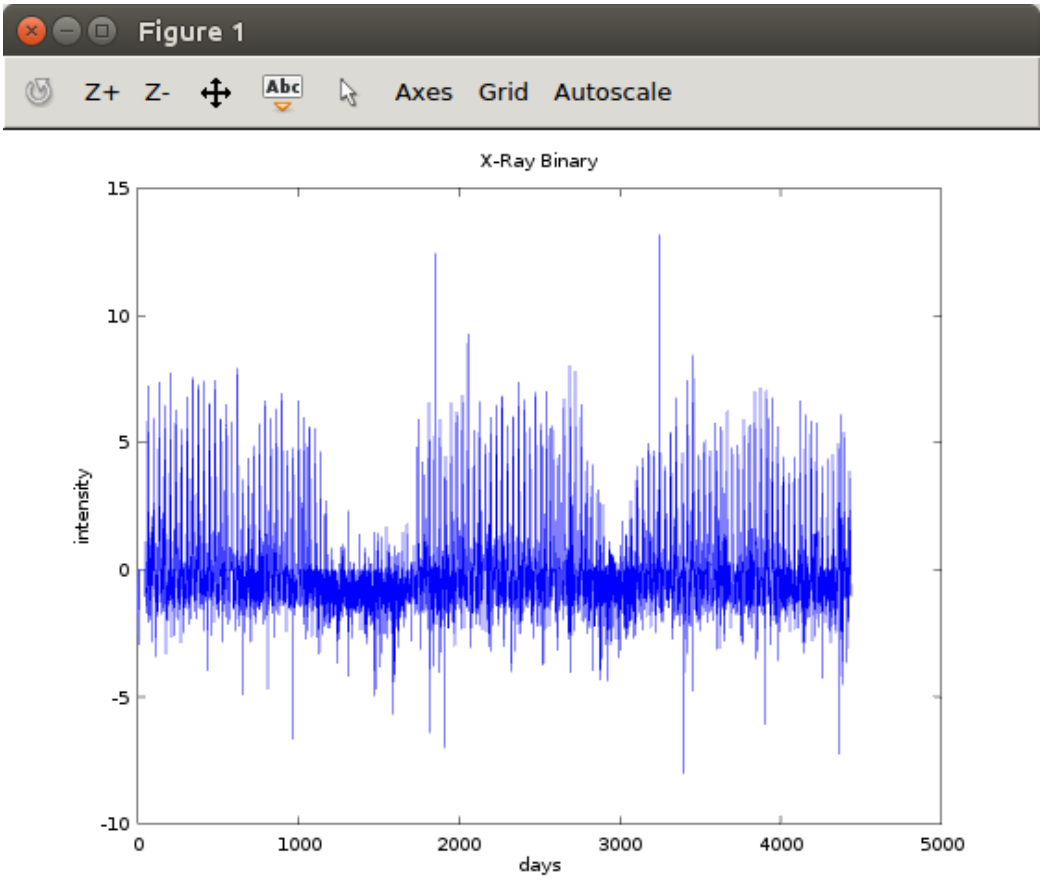
Fourier Analysis

Erin Goeke

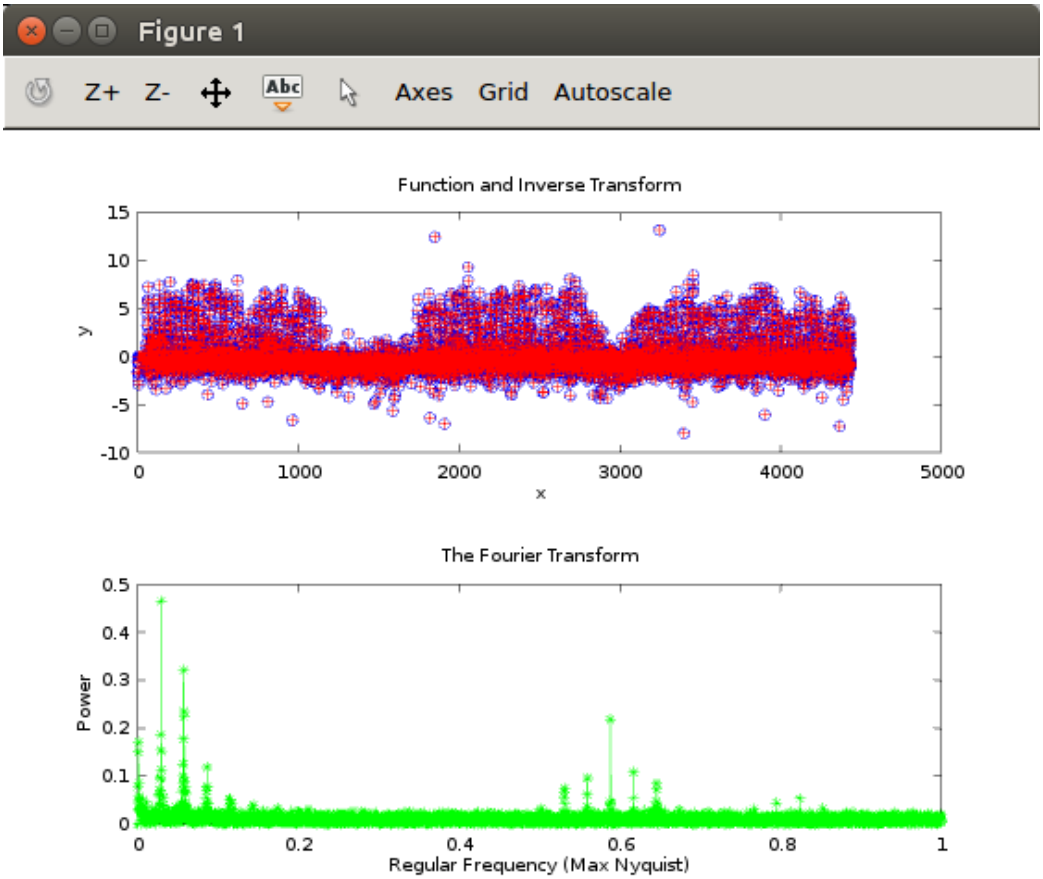
April 11, 2017

1 Astronomy Project

Here is the graph of our entire binned data train for the binary system I was given! Not very helpful because we have peaks and troughs which are on a much larger time domain than what we would like to investigate



This is the transform and inverse transform for the larger data, which most likely contains some erroneous low frequencies which correspond to the long wavelength which as stated above, we don't care about.



To get rid of information we don't need I cut our data up into a small section which should tell us more about what is going on, on a monthly basis. I chose the half days from 2000 to 2800.
To see what the frequency and period look like for this large time domain I have also included the graphs below:

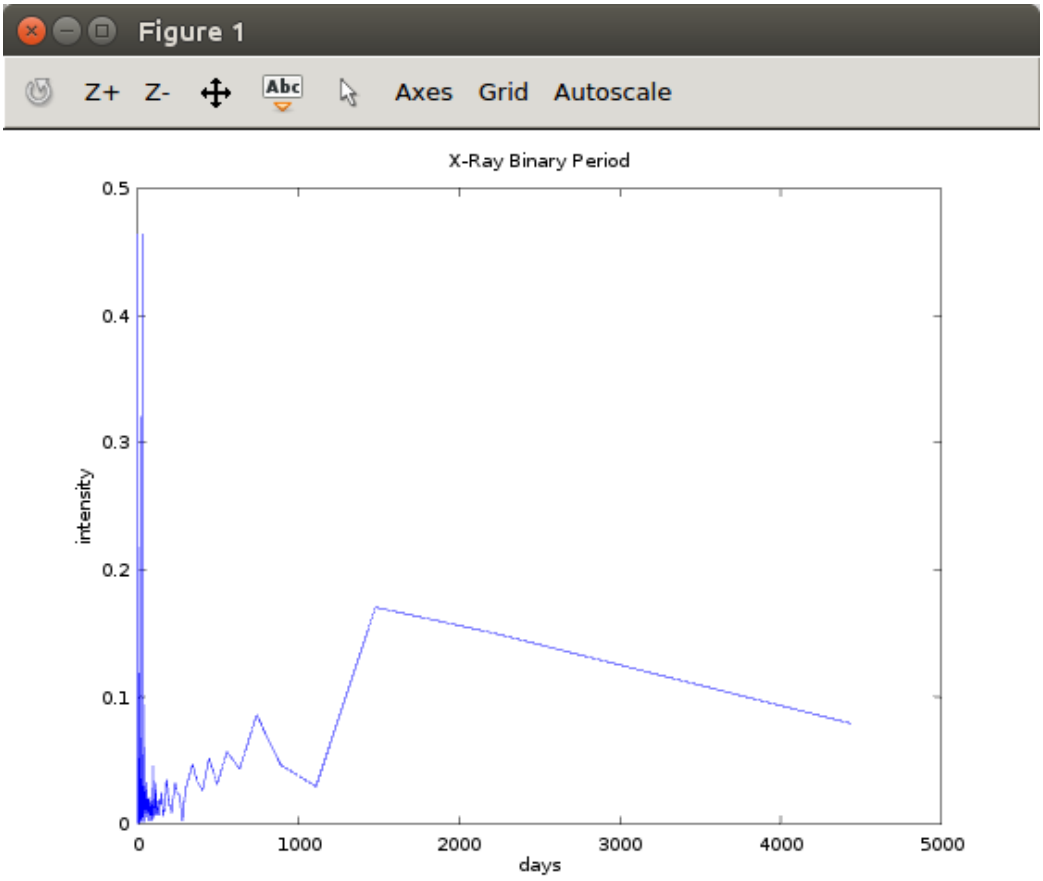
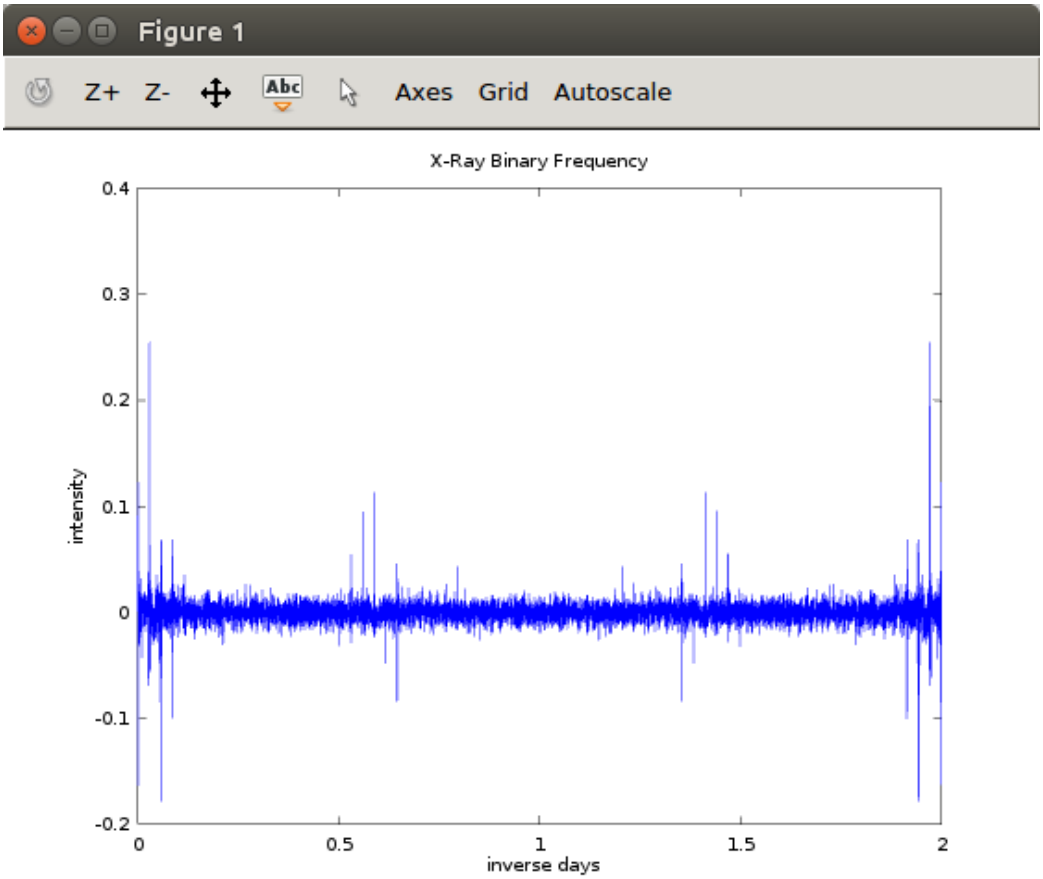
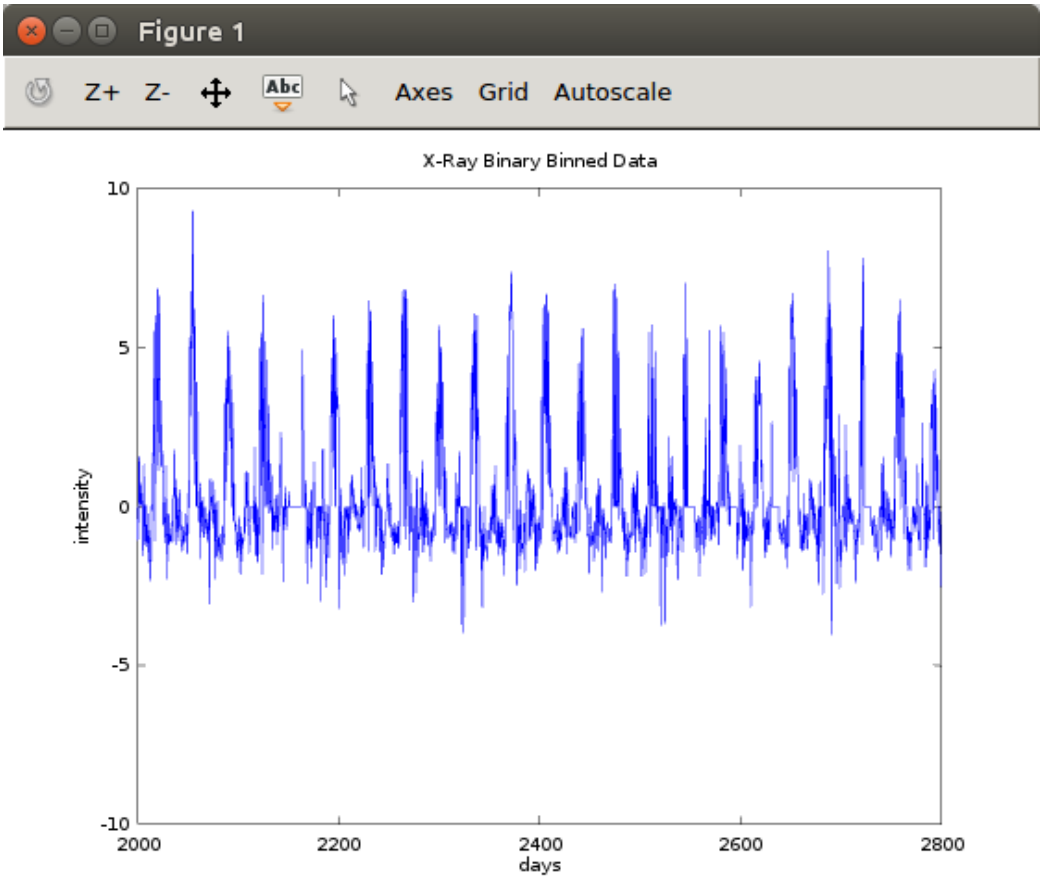


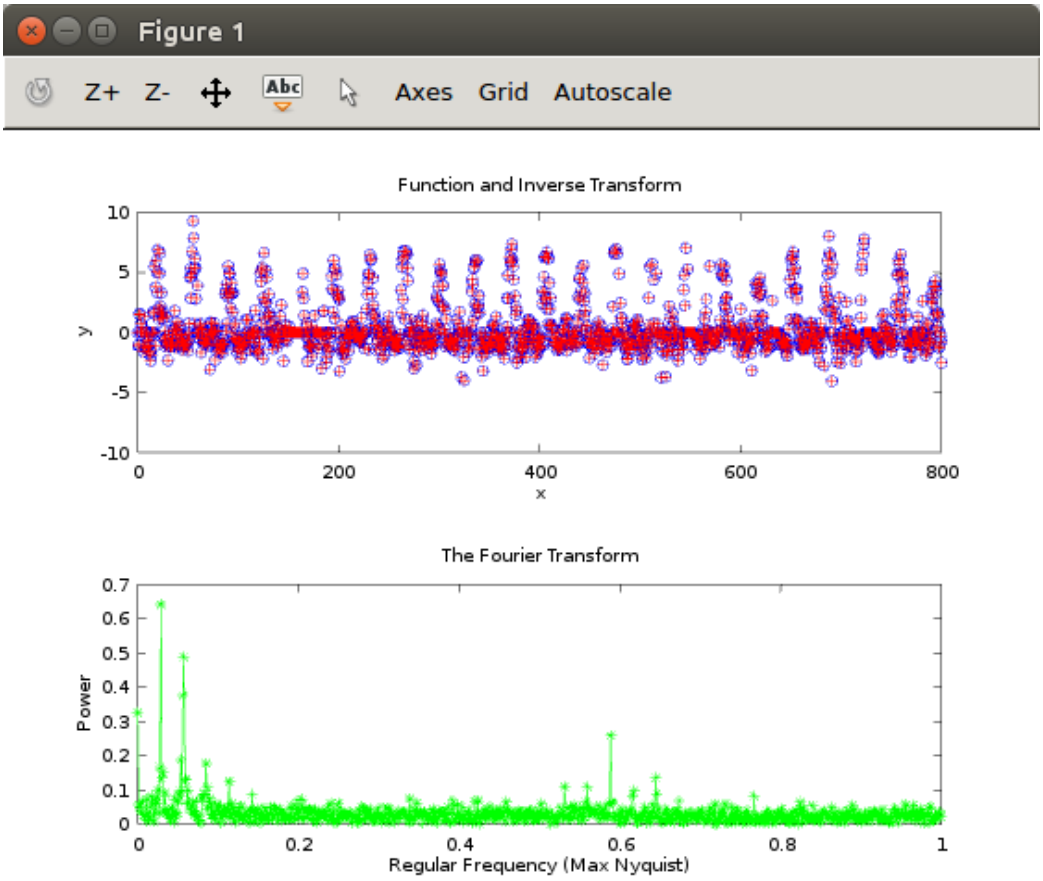
Figure 1
Z+
Z-
Abc
Axes
Grid
Autoscale
X-Ray Binary Period
intensity
days
0
0.1
0.2
0.3
0.4
0.5
0
1000
2000
3000
4000
5000



Not very useful for our context, however these graphs still do hold a lot of information.
Here is the smaller section of our binned data which I chose to work with!



As we can see already there is less information in our Fourier Transform, but in this case less is *more* because I don't have to deal with extra, confusing information I don't care about.



Here is the frequency and period graph for the smaller binned data!

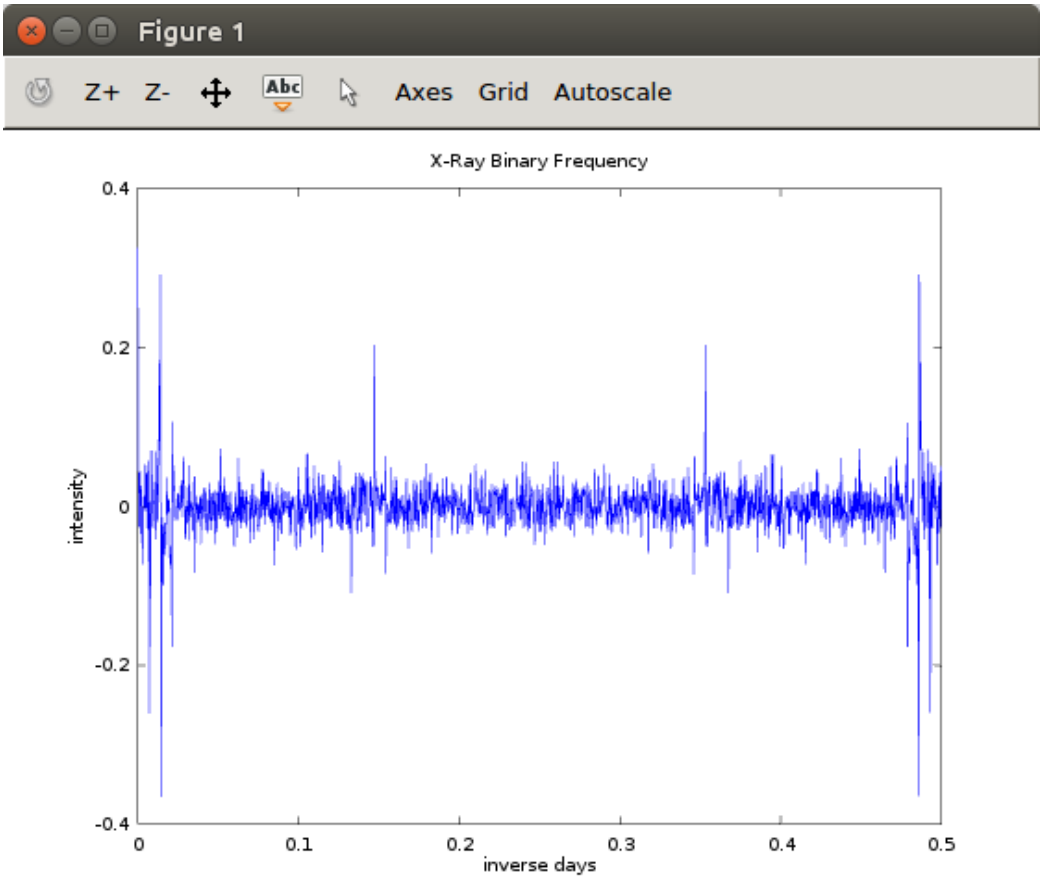
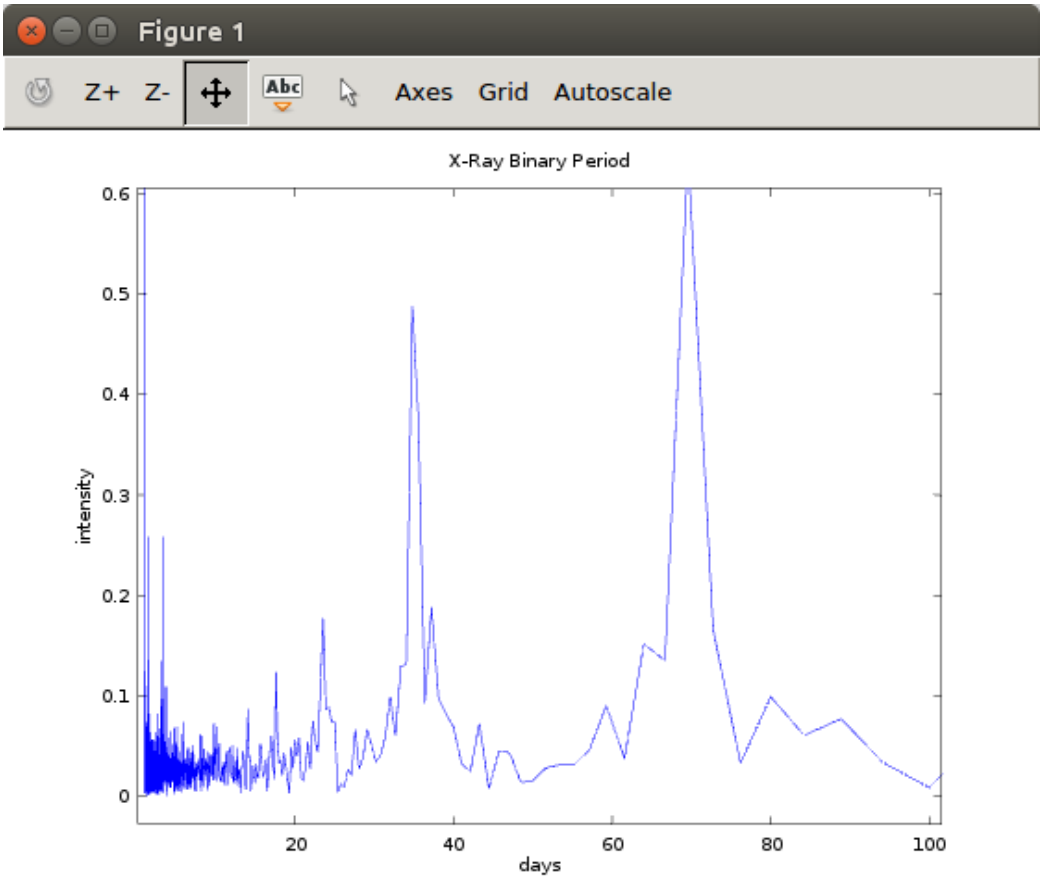


Figure 1
Z+
Z-
Abc
Axes
Grid
Autoscale
X-Ray Binary Frequency
intensity
inverse days
0.4
0.2
0
-0.2
-0.4
0
0.1
0.2
0.3
0.4
0.5



There is some action happening around the 35 day mark and what appears to be a harmonic of it around the 17 day mark. I would guess that this would have to do with the orbiting masses blocking each other and the difference in their respective intensities showing a difference in overall intensity over time. This would also make sense because the objects are so close and massive that they would probably only take about a month to complete a full orbit. I do not know what the higher frequency is at this time.

1.1 fourier.m

```
1 function [a,b]=fourier(alpha,beta,tol,funky,p,nterm)
2 %This is a program to evaluate fourier coefficients
3 % INPUTS
4 % alpha= begin interval
```

```
% beta= end of interval
% n= number of divisions
% funky= function to be approximated
%
% OUTPUTS
% a= coefficient
% b= coefficient
% written on Feburary, 20 2017

%loop until desired coefficient number is reached
for i=1:nterm

    a(i)=fconvint(alpha,beta,tol,funky,p,0,i);
    b(i)=fconvint(alpha,beta,tol,funky,p,1,i);
end
end
```

1.2 fconvint.m

```
function [new] = fconvint(alpha,beta,tol,f,p,trigflag,i
    )
%alpha = beginning of interval
%beta = end of interval
%tol = the tolerance each term is to be calculated to
%f = .m file name which makes an array of the function
%p = parameters for f
%trigflag = determines whether the coefficient for sine
    or cosine
%is to be calculated (0 = cos; 1 = sin)
%i = which term is to be calculated

%Written by Math Phys Class Feb 22, 2017
%Editted by eggoeke on 3/14/2017

%create start number of slices
n = 10;
%make linear space
```

```
17 x=linspace(alpha,beta,n);
18 %grab function array
19 y=feval(f,x,p);
20 %dot integrand with trig function it is describing and
      scale
21 %for all intervals
22 if trigflag==0
23     y=y.*cos((i-1)*2*pi*x/(beta-alpha))*2/(beta-alpha);
24 else
25     y=y.*sin(i*2*pi*x/(beta-alpha))*2/(beta-alpha);
26 end
27 %get preliminary integral to compare future integrals
      with
28 old = fsimp(x,y);
29 n=n*10;
30 %same as above only n the number of slices is increased
31 x=linspace(alpha,beta,n);
32 y=feval(f,x,p);
33 if trigflag==0
34     y=y.*cos((i-1)*2*pi*x/(beta-alpha))*2/(beta-alpha);
35 else
36     y=y.*sin(i*2*pi*x/(beta-alpha))*2/(beta-alpha);
37 end
38 new = fsimp(x,y);
39 %rinse, wash and repeat until the difference betweem
      the old and new
40 %integrals is less than the tolerance
41 while((abs(old-new) > tol )&(n < 10000000000))
42     old = new;
43     n=n*10;
44     x=linspace(alpha,beta,n);
45     y=feval(f,x,p);
46     if trigflag==0
47         y=y.*cos((i-1)*2*pi*x/(beta-alpha))*2/(beta-
              alpha);
48     else
49         y=y.*sin(i*2*pi*x/(beta-alpha))*2/(beta-alpha);
50     end
```

```
51      new = fsimp(x,y);
52  end
53  n;
54  end
```

1.3 plotfou.m

```
1   function plotfou(x,funky,p,a,b)
2   % This program plots a function as well as the terms in
        a fourier
3   %   series (that have been calculated previously by
        another function
4   % Written by Don Smith 2/22/2017
5   % Editted by eggoeke on 3/14/17
6
7   % INPUTS
8   %  x = the array of x-values over which the function
        will be plotted
9   %  funky = the name of the function.  There must be a
        file funky.m
10  %  p = the array of parameters that the function funky
        needs
11  %  a = the array of "a" fourier coefficients
12  %  b = the array of "b" fourier coefficients
13
14  % grab array of function approximating and plot
15  y = feval(funky,x,p);
16  plot(x,y,'ko')
17  hold on
18  % make arrays for containing the cosine and sine terms
19  zc = zeros(size(x));
20  zs = zeros(size(x));
21
22  % the first one for cosine is always weird: half a0
23  zc = zc + a(1)/2;
24
25  % create and add up the cosine terms
```

```
for n=2:length(a)
      term = a(n)*cos((n-1)*2*pi*x/(x(length(x))-x(1)));
      zc = zc + term;
end

% create and add up the sine terms
for n=1:length(b)
       term = b(n)*sin(n*2*pi*x/(x(length(x))-x(1)));
       zs = zs + term;
end

% add together the two for the approximation
total = zc + zs;

% plot the approximation
plot(x,total,'cx')

hold off
end
```

1.4 binner.m

```
%% Author: eggoeke <eggoeke@nephele>
%% Created: 2017-04-03
%% Help from Don

function [out] = binner (in, bin)
  floor = in(1,1); % Define time axis origin
  in(:,1) = in(:,1) - floor; % Subtract first time
     value from all times
  binnum = round(in(:,1)/bin-0.5); % Define an array of
      bin numbers for each data
  maxbin = max(binnum); % What is the highest bin
     number?

  out = zeros(maxbin,3); % Define the output array:
     time, avg, num points in bin
```

```
12    out(:,1) = bin*(0:maxbin-1)'; % Define new time axis
          with evenly spaced points
13
14    for i=0:maxbin-1 % Step through each bin
15      ninbin = binnum(binnum==i); % Identify the points
            in this bin
16      if length(ninbin>0) % If there are more than zero
            points in the bin
17        out(i+1,3) = length(ninbin); % Number of points
              in the bin
18        out(i+1,2) = sum(in((binnum==i),2))/out(i+1,3); %
              Average of all points in bin
19      end
20    end
21    % Subtract off the average value of all bins that
          have points in them
22    out((out(:,3)>0),2) = out((out(:,3)>0),2) - mean(out
          ((out(:,3)>0),2));
23
24  endfunction
```