Fourier Analysis I

Erin Goeke

March 17, 2017

1 Introduction

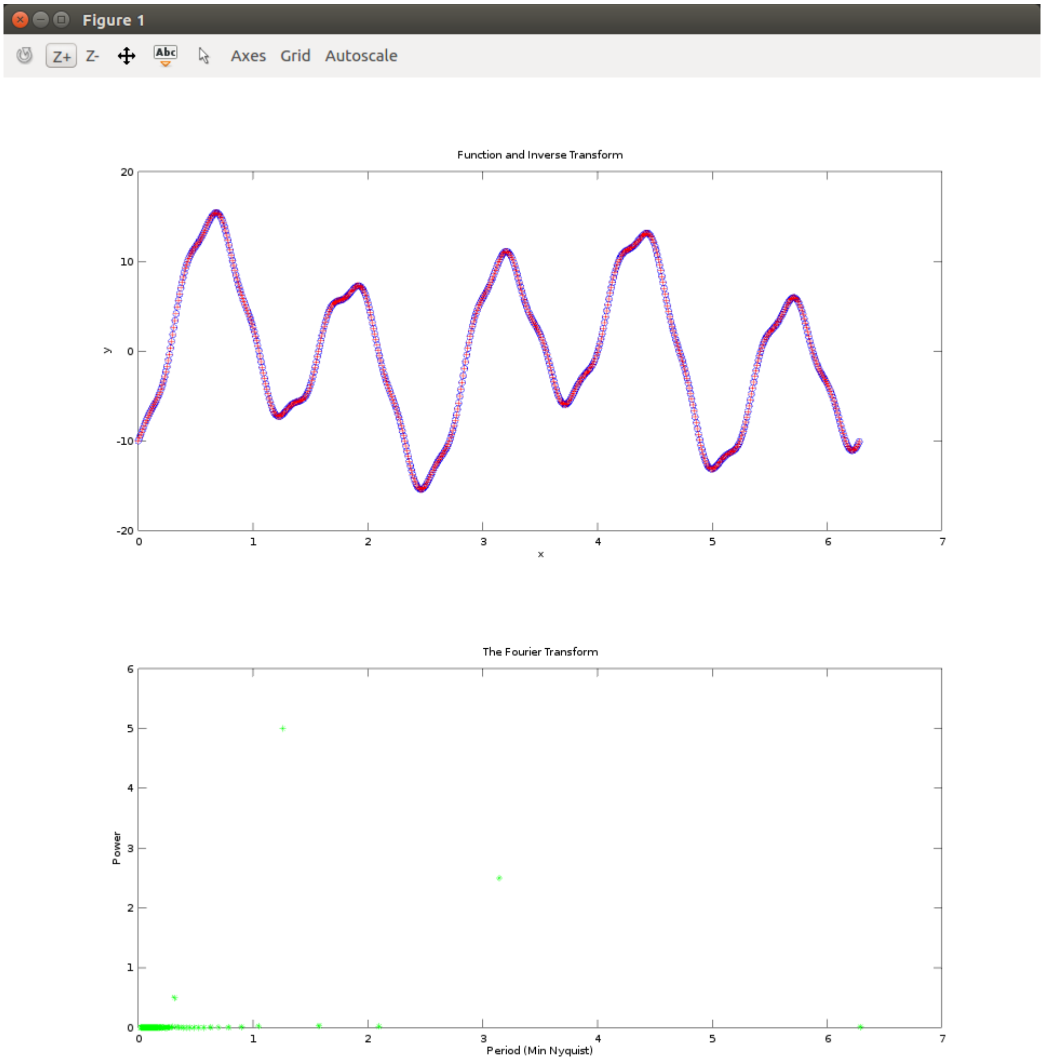
Yikes, I have been bad. Thank you for always being patient with me; I will try *very* hard on the rewrite.

2 Testing Fourier Transforms

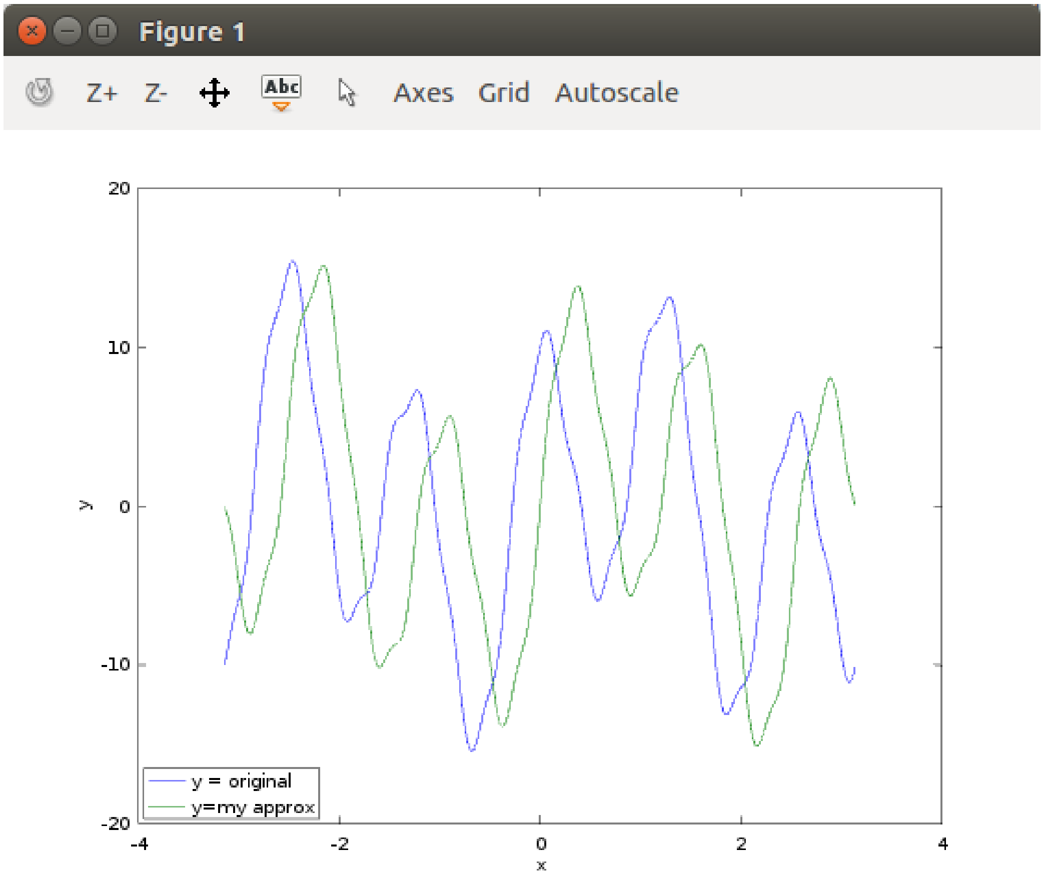
The code I have prepared for creating Fourier Transforms is located in Section 4.

3 Estimating stuff

Howdy! Justin gave me this domain and range which he called 'erinData.mat' and I loaded it into my workspace. It gave me a graph that looks like:



Using the relationship between wavelength and frequency I got: ¿¿ f = @(x)sin(20*x) + 5*sin(2*x)+10*sin(5*x);
This has the right amplitude and a good approximate frequency, however there is a phase shift which I can't account for.



4 Matlab Code

4.1 plotFT.m

```
function plotFT(y,g,iy,tau,fflag,nyqflag,powflag,perflag)
% The inputs are:
%   y = the original data train (required)
%   g = the FT (just put 0 if you don't want this plotted)
%   iy = the inverse FT (just put 0 if you don't want this plotted)
%   tau = the spacing of the data train (required)
%   fflag = 0 if you want angular freq; 1 if you want regular freq
%   nyqflag = 0 if you want to plot ALL data; 1 if you want to ignore aliases
%   powflag = 0 if you want to plot real/imag parts of g; 1 to plot power
%   perflag = 0 to plot vs. frequency; 1 to plot vs. period

clf;

% Define important variables
N = length(y);
x = (0:N-1)*tau;
alph = 2*pi/N/tau*(0:N-1);
pow = sqrt(conj(g).*g);

% Determine whether we will be using frequency or angular freq.
f = alph;
period = 2*pi./f;
xname = 'Angular Frequency';
if fflag
  f = alph/2/pi;
  xname = 'Regular Frequency';
end
if perflag
  xname = 'Period';
end



% Plot the original function, and (if present) the inverse transform
if length(g)==N
  subplot(2,1,1)
end
```

```

plot(x,y,'bo')
hold on
if length(iy)==N
   plot(x,iy,'r+')
   title('Function and Inverse Transform')
else
   title('Original Data Train')
end
xlabel('x')
ylabel('y')
hold off

% Plot the transform, if desired

if length(g)==N
  subplot(2,1,2)
  nmax = N;
  if nyqflag
     nmax = round(N/2);
     if perflag
       xname = strcat(xname,' (Min Nyquist)');
     else
       xname = strcat(xname,' (Max Nyquist)');
     end
  end
  if powflag
    if perflag
      plot(period(1:nmax),pow(1:nmax),'g*')
    else
      plot(f(1:nmax),pow(1:nmax),'g*')
    end
    ylabel('Power')
  else
    if perflag
      plot(period(1:nmax),real(g(1:nmax)),'r*',period(1:nmax),imag(g(1:nmax)),'g*')
    else
```

```
      plot(f(1:nmax),real(g(1:nmax)),'r*',f(1:nmax),imag(g(1:nmax)),'g*')
    end
    ylabel('Real and Imaginary Coefficients')
  end
  title('The Fourier Transform')
  xlabel(xname)
end

end
```

4.2 inverseTransform.m

```
% inverse Fourier Transform program

% inputs: g = array containing range of fourier transform
% outputs: y = range of approximation of original function by inverse
% fourier transform

% Written by Justin and Lizzie 3/13/17
% Editted by eggoeke 3/14/17
function [y] =inverseTransform(g)
% create N as the number of terms used
N=length(g);
% allocate space for exponential array
exponentials=ones(N,1);
for n=1:N
    % declare entire exponential array for each term
    exponentials(:,1)=exp(i*(n-1)*(0:N-1)'*2*pi/N);
    % multiply g by exponentials for the term to get
    % their sum on the domain and assign to a slot in y
    y(n)=g*exponentials;
end
end
```

4.3 fouG.m

```
% Fourier transform code

% Input: y = array of the range of the function inverse is
% calculated for
% Output: g = array of of amplitudes of present in y in relation to
% alpha (analogous to n in fourier series; only continuous)

% Written by Erin and Jen 3/13/17
% Edited by eggoeke 3/14/17
function [g]= fouG(y)
% compute the number of terms given in y
N = length(y);
% allocate space for an array which will contain exponential terms
% for summation and space for the fourier transform range
expC = ones(N,1);
g = ones(1,N);
for n = 1:N
    % declare entire array exponential terms for each term
    expC(:,1) = exp(-i.*(n-1).*(0:(N-1))'.*2*pi./N);
    % multiply y by exponential array to get sum, multiply by
    % scalar and set to corresponding g term (amplitude) for
    % frequency(?)
    g(n) = 1/N.*(y*expC);
end
end
```