CS170A -- HW#3 -- assignment and solution form -- Octave

Your name: Shirley Xuemin He

Your UID: 204468663

Please upload only this notebook to CCLE by the deadline.

Policy for late submission of solutions: We will use Paul Eggert's Late Policy: N days late $\Leftrightarrow$ 2^N points deducted} The number of days late is $N = 0$ for the first 24 hrs, $N = 1$ for the next 24 hrs, etc., and if you submit an assignment H hours late, $2^{\lfloor H/24 \rfloor}$ points are deducted.

Problem 1: Fourier Music

Remember the k-th note in the the **equal-tempered scale** has frequency c^k times the frequency of C, where $c = 2^{1/12} = 1.059463\ldots$:

note	frequency (Hz)	frequency/C	interval
Middle C	261.63	$c^0 = 1.000$	~
C#	277.18	$c^1 = 1.059$	minor 2nd
D	293.66	$c^2 = 1.122$	major 2nd
D#	311.13	$c^3 = 1.189$	minor 3rd
E	329.63	$c^4 = 1.260$	major 3rd
F	349.23	$c^5 = 1.334$	4th
F#	369.99	$c^6 = 1.414$	diminished 5th
G	392.00	$c^7 = 1.498$	5th
G#	415.30	$c^8 = 1.587$	minor 6th
A	440	$c^9 = 1.682$	major 6th
A#	466.16	$c^{10} = 1.782$	minor 7th
B	493.88	$c^{11} = 1.888$	major 7th
High C	523.25	$c^{12} = 2.000$	octave/8th

The equal-tempered scale can be extended to higher or lower octaves by multiplying the frequencies in it by a power of 2. (For example, the note 'A' occurs at frequencies 220, 440, 880, etc., because 440 = 2 × 220, 880 = 4 × 220, etc.

However, it is not right to call the frequency 660 = 3 × 220 an 'A'; the frequency 660 is an 'E', because 600 = 2 × 330, and 330 is the frequency for 'E'.) Only the frequencies $2^k \times 440$ (for integer k) are called 'A'.

Remember: if a signal is sampled at frequency `Fs`, then the *Nyquist frequency* at the end of this first half is `Fs/2`.

1.0 Read in the Mystery Tune

Read in the file `mystery.wav`, creating a vector `y` that contains a sound track (audio sequence) for the first few seconds of a mystery tune.

As discussed in class, if `Fs` is an integer sampling frequency. In Octave and Matlab, executing `sound(y, Fs)` will play `y` at the frequency `Fs`.

Change `y` to contain only the first of the 2 stereo audio tracks.

In [ ]:

```
% pkg install audio-1.1.4.tar
% pkg load audio
```

In [43]:

```
[y Fs] = auload('mystery.wav');

sound(y,Fs)   %  play the mystery tune
```

In [44]:

```
y = y(:,1);   %  only keep the first track for this assignment

n = length(y)
```

```
n =  860000
```

1.1 The Fast Fourier Transform

The length of `y` is $n = 86000$. Find the integer factors of n by using the Matlab function `factor()`.

Then find out how much cpu time it takes to compute `fft(y);`

(To do this, run it 100 times and compute the average time required, using `cputime`.)

In [46]:

```
factors_of_n = factor(n)
t_y = cputime;
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
```

```
fft(y);

fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
```

```
fft(y);

fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
fft(y);
e_y = (cputime-t_y)/100;

printf("fft(y) takes %d s", e_y);
```

```
factors_of_n =

    2    2    2    2    2    5    5    5    5   43

fft(y) takes 0.0470895 s
```

1.2 The Fast Fourier Transform?

First, define a new vector `z` that contains `y(1:(n-13))`. Find the integer factors of $(n - 13)$.

Find out how much cpu time it takes to compute `fft(z);` using the method above.

Explain the difference in timing; this is discussed on pp.235-236 of the Course Reader.

In [48]:

```
z = y(1:(n-13));
factors_of_nminus13 = factor(n-13)
t_z = cputime;
fft(z);
fft(z);
fft(z);
```

```
fft(z);

fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
```

```
fft(z);

fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
fft(z);
e_z = (cputime-t_z)/100;

printf("fft(z) takes %d s", e_z);
```

```
factors_of_nminus13 =  859987
fft(z) takes 0.339489 s
```

In [5]:

```
% fft(z) takes much longer because n-13 = 859987 is a prime number,
% so FFT cannot be decomposed.
```

1.3 Plot the First Half of the Power Spectrum

Plot the frequency spectrum for `y` by plotting the power of the 'first half' of the squared power of its Fourier transform, using `Fs` as the sampling frequency.

Assume for this assignment that the *power* of a complex value z is its complex absolute value: i.e., $\mathrm{power}(z) = |z| = \sqrt{z\,\bar{z}}$.

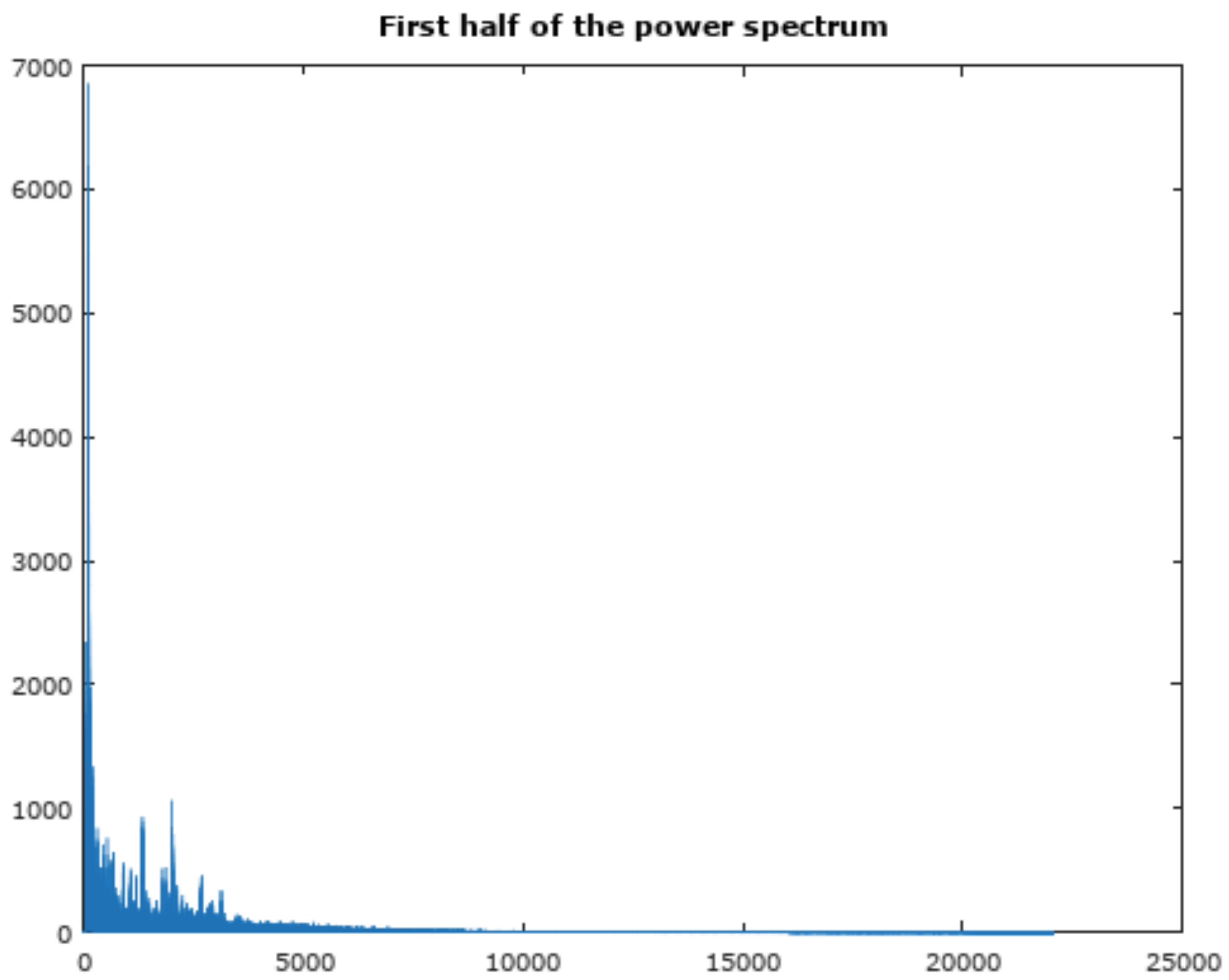
Here by "*first half*" of a sequence $s = [s_0, \ldots, s_{n-1}]$ we actually mean $[s_1, \ldots, s_m]$ where $m = \lfloor n/2 \rfloor$. For example, the "first half" of $[0, 1, 2, 3, 4, 5, 6, 7]$ is $[1, 2, 3, 4]$.

To increase the information displayed by your plot, omit the very first element in the `power` vector (which is just the sum of the data).

In [ ]:

```
fftY = fft(y);
firstHalf = 2:(n/2+1);
power = abs(fftY(firstHalf));

NyquistFrequency = Fs/2;
freqs = linspace(1, NyquistFrequency, n/2);

%%% This following plot makes the program hang
% plot(freqs, power);
% title("First half of the power spectrum")
```

So I include a screenshot of the graph instead of the actual plot:



1.4 Write a script to find the top 10 Notes (corresponding to Spikes)

There are altogether about $(5 \times 12) + 1 = 61$ notes in the equal-tempered scale with frequencies between low C (with frequency near 131) and the very high C with frequency near $131 \times 2^5 = 4192$.

Suppose we also define **dividing lines** between notes. For example, C has frequency 261.63 and C# has frequency 271.18. The dividing line between these two notes is $\Delta = (261.63 + 271.18)/2 = 266.40$.

Just as every note in the equal-tempered scale has frequency c^k times the frequency of C, we can define **equal tempered dividing lines** between all adjacent notes as having frequency c^k times the frequency Δ.

Write a Matlab function `note_power(power,Fs)` that distills a power spectrum vector `power` into an 61×1 vector that gives, for each equal-tempered note, the maximum value among all entries in the vector within the dividing lines above and below this note. Find the top 10 notes (the 10 notes with highest power).

For the mystery clip, use your function to print the power value for each of these 10 notes.

In [50]:

```
function [toptennotes] = note_power(power,Fs)

sigma = 2^(1/12);
k = (0:11)';
equal = sigma.^k;
middle = 261.63 * equal;
equaltemperedscale = [middle/2 ; middle ; middle*2 ; middle*4 ; middle*8 ; 261.6
3*16];

first60 = equaltemperedscale(1:60);
last60 = equaltemperedscale(2:61);
dividinglines = (first60 + last60) / 2;

bins = histc(power, dividinglines);

endfunction
```

1.5 Identify the Key, and find the relationship of the top 10 Notes to the Key

Find the note whose spike in the power spectrum is largest; this is called the **key**.

For each of the top 10 notes, compute the **ratio** of their frequency to that of the key. Each ratio will be a power c^k; for each note, give the value of k.

A major fifth has a ratio c^7 , which is approximately 3/2 -- the tonic/dominant ratio.

A minor fifth (diminished fifth) has a ratio c^6 , which is approximately 1.4.

A major third has a ratio c^4 , which is approximately 5/4 -- the ratio of harmony.

In [8]:

```
c = 2^(1/12)

major_third = c^4
minor_fifth = c^6
major_fifth = c^7
```

```
c =  1.0595
major_third =  1.2599
minor_fifth =  1.4142
major_fifth =  1.4983
```

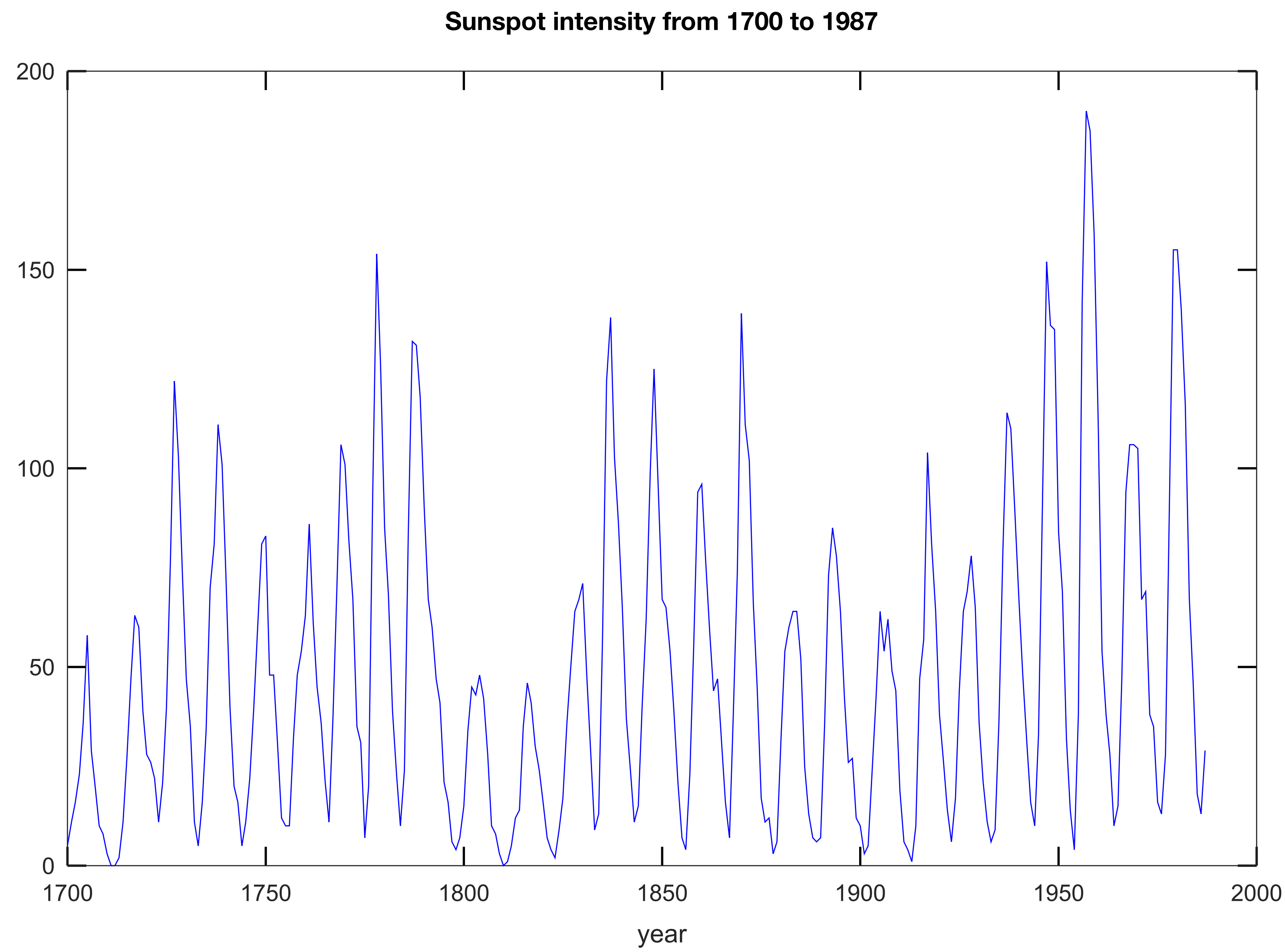
Problem 2: Sunspots

Sunspots have effects on power grids and communications on earth, and have had high intensity recently. They often appear in the news, and there is a sunspot observatory at Big Bear near Los Angeles (http://phys.org/news/2016-10-tracking-sunspots-solar-insight.html).

In class we studied a classic example of Fourier analysis: determining the periodicity of sunspot data. The plot of the data shows it is clearly periodic.

In [9]:

```
% sunspot activity from 1700 to 1987.

years = 1700 : 1987;
sunspots = [ ...
     5    11    16    23    36    58    29    20    10     8     3     0     0
2    11    27 ...
    47    63    60    39    28    26    22    11    21    40    78   122   103
73    47    35 ...
    11     5    16    34    70    81   111   101    73    40    20    16     5
11    22    40 ...
    60    81    83    48    48    31    12    10    10    32    48    54    63
86    61    45 ...
    36    21    11    38    70   106   101    82    67    35    31     7    20
93   154   126 ...
    85    68    39    23    10    24    83   132   131   118    90    67    60
47    41    21 ...
    16     6     4     7    15    34    45    43    48    42    28    10     8
3     0     1 ...
     5    12    14    35    46    41    30    24    16     7     4     2     9
17    36    50 ...
    64    67    71    48    28     9    13    57   122   138   103    86    65
37    24    11 ...
    15    40    62    99   125    96    67    65    54    39    21     7     4
23    55    94 ...
    96    77    59    44    47    31    16     7    38    74   139   111   102
66    45    17 ...
    11    12     3     6    32    54    60    64    64    52    25    13     7
6     7    36 ...
    73    85    78    64    42    26    27    12    10     3     5    24    42
64    54    62 ...
    49    44    19     6     4     1    10    47    57   104    81    64    38
26    14     6 ...
    17    44    64    69    78    65    36    21    11     6     9    36    80
114   110    89 ...
    68    48    31    16    10    33    93   152   136   135    84    69    32
14     4    38 ...
   142   190   185   159   112    54    38    28    10    15    47    94   106
106   105    67 ...
    69    38    35    16    13    28    93   155   155   140   116    67    46
18    13    29 ...
];

plot( years, sunspots, 'b' )
title('Sunspot intensity from 1700 to 1987')
xlabel('year')
```

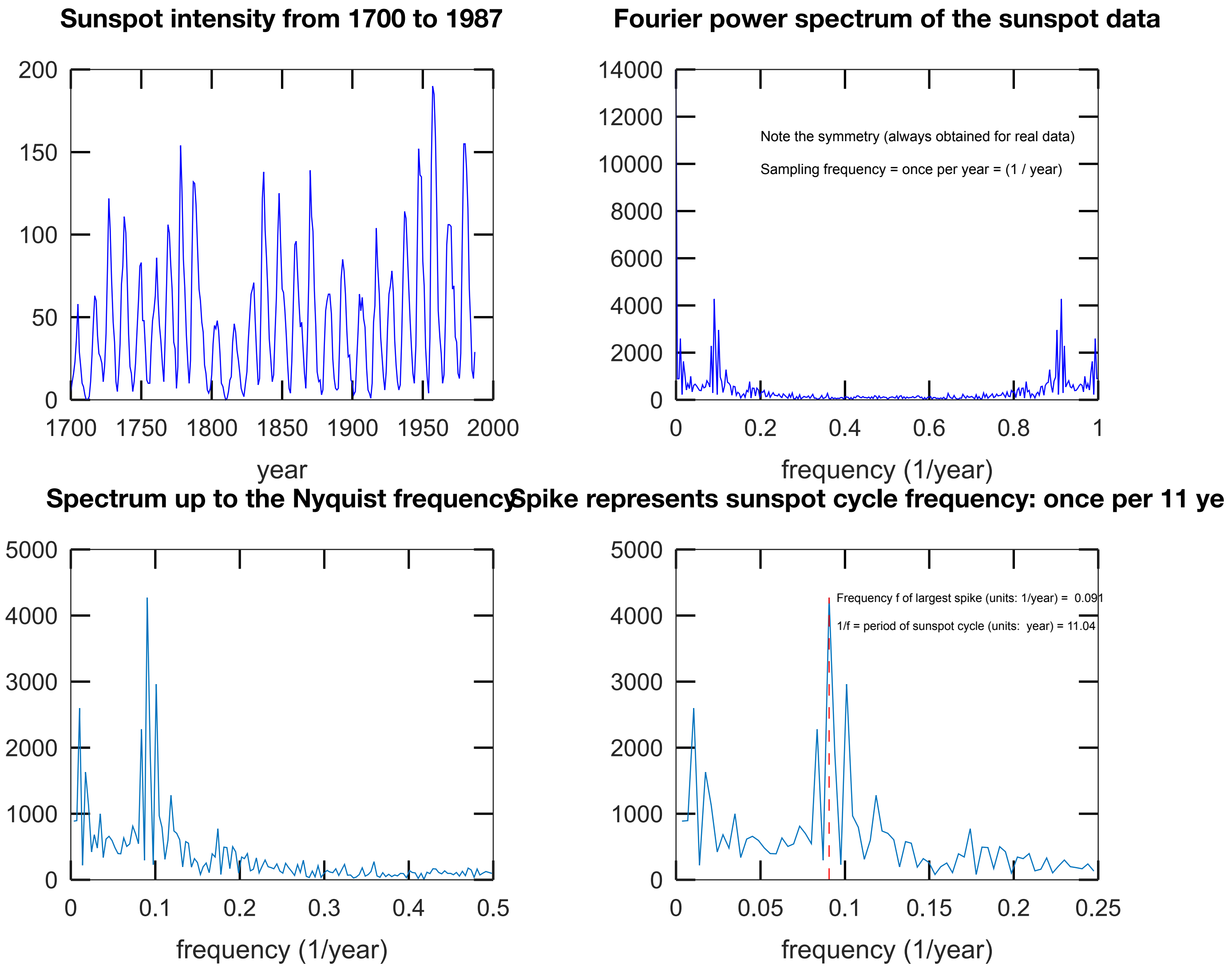


We can use Fourier analysis to find the frequency of the cycles shown in the plot above. Based on the script below, sunspots go through a cycle that has frequency (0.091/year) which is (1/(11 years)). Thus the period of sunspot activity is about 11 years long.

In [10]:

```
subplot(2,2,1)
plot( years, sunspots, 'b' )
title('Sunspot intensity from 1700 to 1987')
xlabel('year')

power_spectrum = abs(fft(sunspots));
n = length(sunspots);
sampling_frequency = 1;  % one sample / year
frequencies = linspace( 0, sampling_frequency, n );

subplot(2,2,2)
plot( frequencies, power_spectrum, 'b' )
text( sampling_frequency/5, max(power_spectrum)*0.80, 'Note the symmetry (always obtained for real data)', 'fontsize', 6 )
text( sampling_frequency/5, max(power_spectrum)*0.70, 'Sampling frequency = once per year = (1 / year)', 'fontsize', 6 )
title( 'Fourier power spectrum of the sunspot data' )
xlabel('frequency (1/year)')

n_over_2 = floor(n/2);
```

```
subplot(2,2,3)
plot( frequencies(2:n_over_2), power_spectrum(2:n_over_2) )
title( 'Spectrum up to the Nyquist frequency' )
xlabel('frequency (1/year)')
% text( sampling_frequency/8, max(power_spectrum)*0.90, 'Nyquist frequency = sam
pling frequency / 2', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.70, 'We ignore the 1st Fouri
er coefficient here;', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.60, 'it is always just the s
um of the input data.', 'fontsize', 6 )

search_interval = 2:floor(n/4);  %  skip the first coefficient, which is always
the sum of the input data
spike_location = 1+find( power_spectrum(search_interval) == max(power_spectrum(s
earch_interval)) );
spike_frequency = frequencies(spike_location);
period_of_sunspot_cycle = 1/spike_frequency;

subplot(2,2,4)
plot( frequencies(search_interval), power_spectrum(search_interval) )
text( spike_frequency*1.05, power_spectrum(spike_location)*1.00,
      sprintf('Frequency f of largest spike (units: 1/year) = %6.3f',spike_frequ
ency), 'fontsize', 5 )
text( spike_frequency*1.05, power_spectrum(spike_location)*0.90,
      sprintf('1/f = period of sunspot cycle (units:  year) = %5.2f',period_of_s
unspot_cycle), 'fontsize', 5 )
title( 'Spike represents sunspot cycle frequency: once per 11 years' )
xlabel('frequency (1/year)')
hold on
plot( [spike_frequency spike_frequency], [0 power_spectrum(spike_location)], 'r-
-' )
hold off
```



Sunspot data is available from WDC-SILSO, Belgian Royal Observatory, Brussels -- in the directory http://www.sidc.be/silso/datafiles (http://www.sidc.be/silso/datafiles), which contains daily, monthly, and yearly sunspot indices:

- **yearly data** (since 1700) is in: http://www.sidc.be/silso/DATA/SN_y_tot_V2.0.csv (http://www.sidc.be/silso/DATA/SN_y_tot_V2.0.csv)
- **monthly data** (since 1749) is in: http://www.sidc.be/silso/DATA/SN_m_tot_V2.0.csv (http://www.sidc.be/silso/DATA/SN_m_tot_V2.0.csv)
- **daily data** (since 1818) is in: http://www.sidc.be/silso/DATA/SN_d_tot_V2.0.csv (http://www.sidc.be/silso/DATA/SN_d_tot_V2.0.csv)

2.0 Get the yearly and monthly data

Define a function to read in a .csv file across the network using `urlread()`, `writefile()`, and `csvread()`.

Use your function to read in the yearly, monthly, and daily data.

In [11]:

```
urlwrite("http://www.sidc.be/silso/DATA/SN_y_tot_V2.0.csv", "yearly.csv");
urlwrite("http://www.sidc.be/silso/DATA/SN_m_tot_V2.0.csv", "monthly.csv");
urlwrite("http://www.sidc.be/silso/DATA/SN_d_tot_V2.0.csv", "daily.csv");
data_year = dlmread("yearly.csv", ";");
data_month = dlmread("monthly.csv", ";");
data_day = dlmread("daily.csv", ";");
yearly = data_year(:,2);
monthly = data_month(:,4);
daily = data_day(:,5);
```

2.1 Compare the Belgian yearly sunspot data with the classical sunspot data

Is the yearly data close to the `sunspots` vector analyzed above? (up to 1987) Find the L2-distance between the two vectors.

In [12]:

```
l2dist = norm(sunspots' - yearly(1:288),2)
```

```
l2dist =  645.78
```

In [13]:

```
% No.
```

2.2 Analyze the yearly data

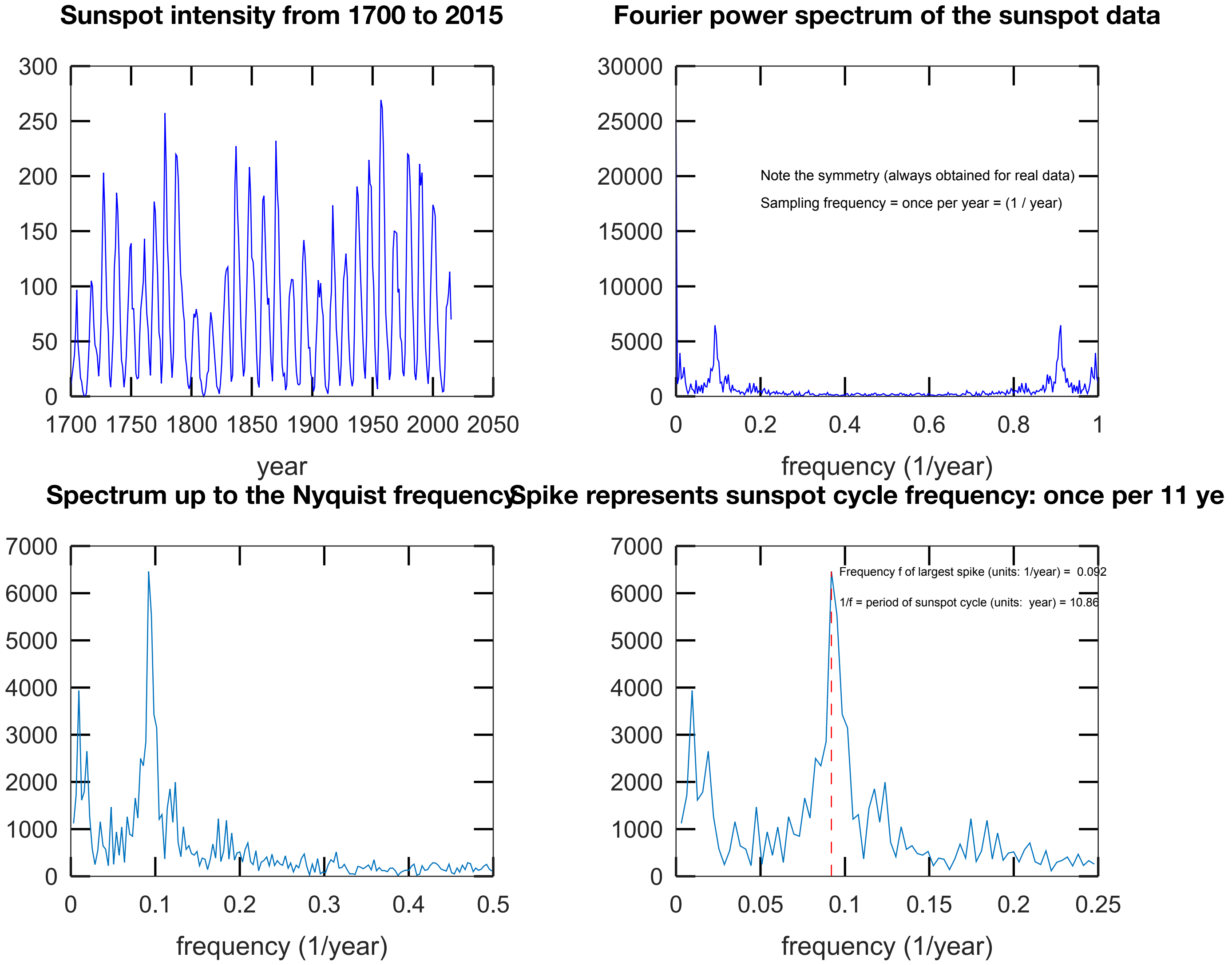
Find the period of yearly sunspot intensity, using analysis like that shown above. Do the two values of the period agree?

In [14]:

```
years = 1700:2015;

subplot(2,2,1)
plot( years, yearly, 'b' )
title('Sunspot intensity from 1700 to 2015')
xlabel('year')

power_spectrum = abs(fft(yearly));
n = length(yearly);
sampling_frequency = 1;  %  one sample / year
frequencies = linspace( 0, sampling_frequency, n );

subplot(2,2,2)
plot( frequencies, power_spectrum, 'b' )
text( sampling_frequency/5, max(power_spectrum)*0.80, 'Note the symmetry (always
```

```
obtained for real data)', 'fontsize', 6 )

text( sampling_frequency/5, max(power_spectrum)*0.70, 'Sampling frequency = once
per year = (1 / year)', 'fontsize', 6 )
title( 'Fourier power spectrum of the sunspot data' )
xlabel('frequency (1/year)')

n_over_2 = floor(n/2);
subplot(2,2,3)
plot( frequencies(2:n_over_2), power_spectrum(2:n_over_2) )
title( 'Spectrum up to the Nyquist frequency' )
xlabel('frequency (1/year)')
% text( sampling_frequency/8, max(power_spectrum)*0.90, 'Nyquist frequency = sam
pling frequency / 2', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.70, 'We ignore the 1st Fouri
er coefficient here;', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.60, 'it is always just the s
um of the input data.', 'fontsize', 6 )

search_interval = 2:floor(n/4);  %  skip the first coefficient, which is always
the sum of the input data
spike_location = 1+find( power_spectrum(search_interval) == max(power_spectrum(s
earch_interval)) );
spike_frequency = frequencies(spike_location);
period_of_sunspot_cycle = 1/spike_frequency;

subplot(2,2,4)
plot( frequencies(search_interval), power_spectrum(search_interval) )
text( spike_frequency*1.05, power_spectrum(spike_location)*1.00,
      sprintf('Frequency f of largest spike (units: 1/year) = %6.3f',spike_frequ
ency), 'fontsize', 5 )
text( spike_frequency*1.05, power_spectrum(spike_location)*0.90,
      sprintf('1/f = period of sunspot cycle (units:  year) = %5.2f',period_of_s
unspot_cycle), 'fontsize', 5 )
title( 'Spike represents sunspot cycle frequency: once per 11 years' )
xlabel('frequency (1/year)')
hold on
plot( [spike_frequency spike_frequency], [0 power_spectrum(spike_location)], 'r-
-' )
hold off

printf("Period of sunspot cycle = %d years", period_of_sunspot_cycle);
```

```
Period of sunspot cycle = 10.8621 years
```



2.3 Analyze the monthly data

Find the period of sunspot intensity using the **monthly** data. At which frequencies are there spikes?

Remember, the sampling frequency here is (1/month) = (12/year).

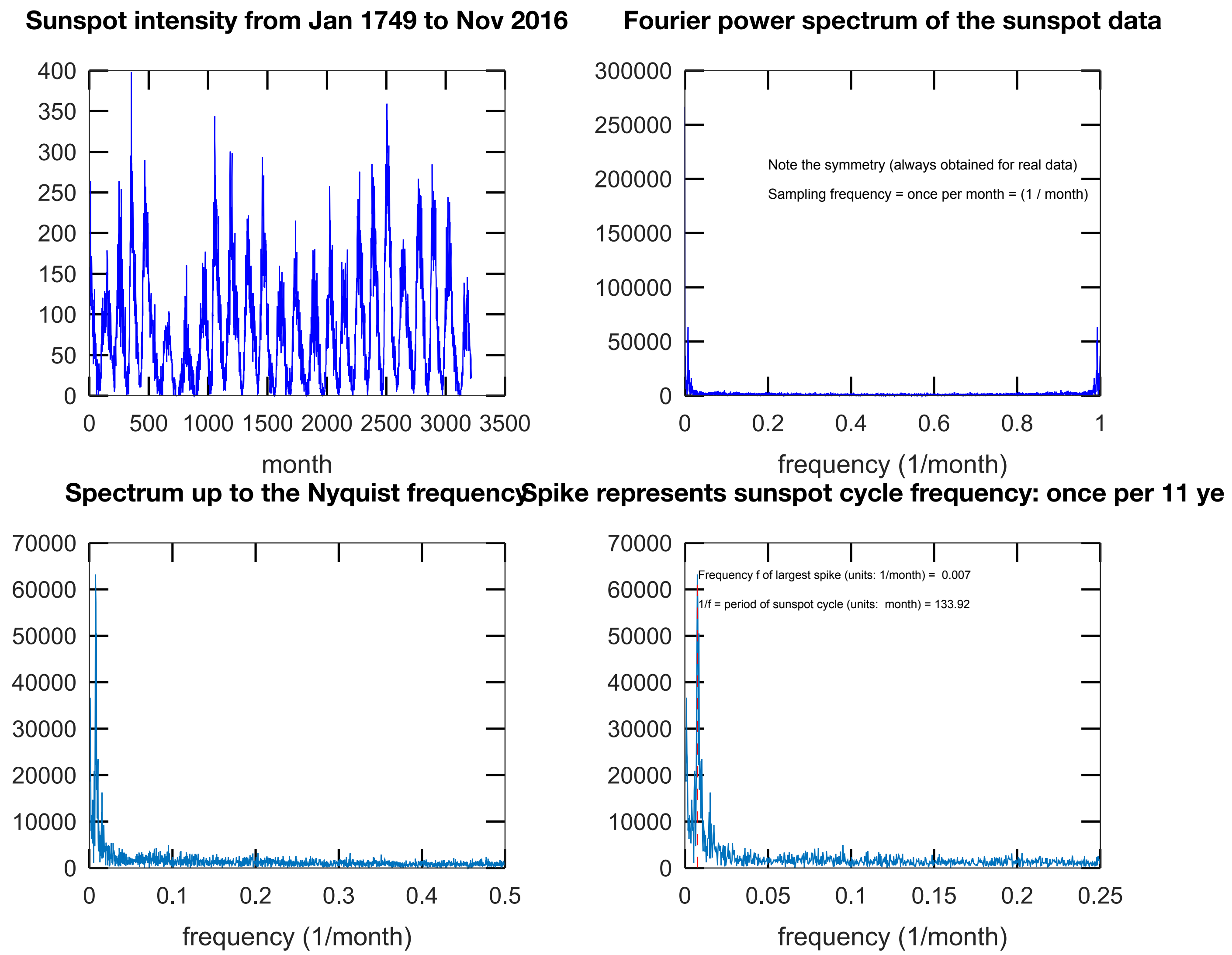
Is there a period at a frequency that is close to the period for the yearly data?

In [15]:

```
months = 1:3215;

subplot(2,2,1)
plot( months, monthly, 'b' )
title('Sunspot intensity from Jan 1749 to Nov 2016')
xlabel('month')

power_spectrum = abs(fft(monthly));
n = length(monthly);
sampling_frequency = 1;  % one sample / month
frequencies = linspace( 0, sampling_frequency, n );

subplot(2,2,2)
plot( frequencies, power_spectrum, 'b' )
text( sampling_frequency/5, max(power_spectrum)*0.80, 'Note the symmetry (always
```

```
text( sampling_frequency/5, max(power_spectrum)*0.80, 'Note the symmetry (always
obtained for real data)', 'fontsize', 6 )
text( sampling_frequency/5, max(power_spectrum)*0.70, 'Sampling frequency = once
per month = (1 / month)', 'fontsize', 6 )
title( 'Fourier power spectrum of the sunspot data' )
xlabel('frequency (1/month)')

n_over_2 = floor(n/2);
subplot(2,2,3)
plot( frequencies(2:n_over_2), power_spectrum(2:n_over_2) )
title( 'Spectrum up to the Nyquist frequency' )
xlabel('frequency (1/month)')
% text( sampling_frequency/8, max(power_spectrum)*0.90, 'Nyquist frequency = sam
pling frequency / 2', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.70, 'We ignore the 1st Fouri
er coefficient here;', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.60, 'it is always just the s
um of the input data.', 'fontsize', 6 )

search_interval = 2:floor(n/4);  %  skip the first coefficient, which is always
the sum of the input data
spike_location = 1+find( power_spectrum(search_interval) == max(power_spectrum(s
earch_interval)) );
spike_frequency = frequencies(spike_location);
period_of_sunspot_cycle = 1/spike_frequency;

subplot(2,2,4)
plot( frequencies(search_interval), power_spectrum(search_interval) )
text( spike_frequency*1.05, power_spectrum(spike_location)*1.00,
      sprintf('Frequency f of largest spike (units: 1/month) = %6.3f',spike_freq
uency), 'fontsize', 5 )
text( spike_frequency*1.05, power_spectrum(spike_location)*0.90,
      sprintf('1/f = period of sunspot cycle (units:  month) = %5.2f',period_of_
sunspot_cycle), 'fontsize', 5 )
title( 'Spike represents sunspot cycle frequency: once per 11 years' )
xlabel('frequency (1/month)')
hold on
plot( [spike_frequency spike_frequency], [0 power_spectrum(spike_location)], 'r-
-' )
hold off

printf("Spike frequency = %d per month", spike_frequency);
printf("Period of sunspot cycle = %d years", period_of_sunspot_cycle/12);
```

```
Spike frequency = 0.00746733 per month
Period of sunspot cycle = 11.1597 years
```



In [16]:

```
% The period of the monthly data is very close to the period of the yearly data,
% approximately 11 years.
```

2.4 Analyze the daily data

Find the period of sunspot intensity using the **daily** data. At which frequencies are there spikes?

Remember, the sampling frequency here is (1/day) = (365.25/year).

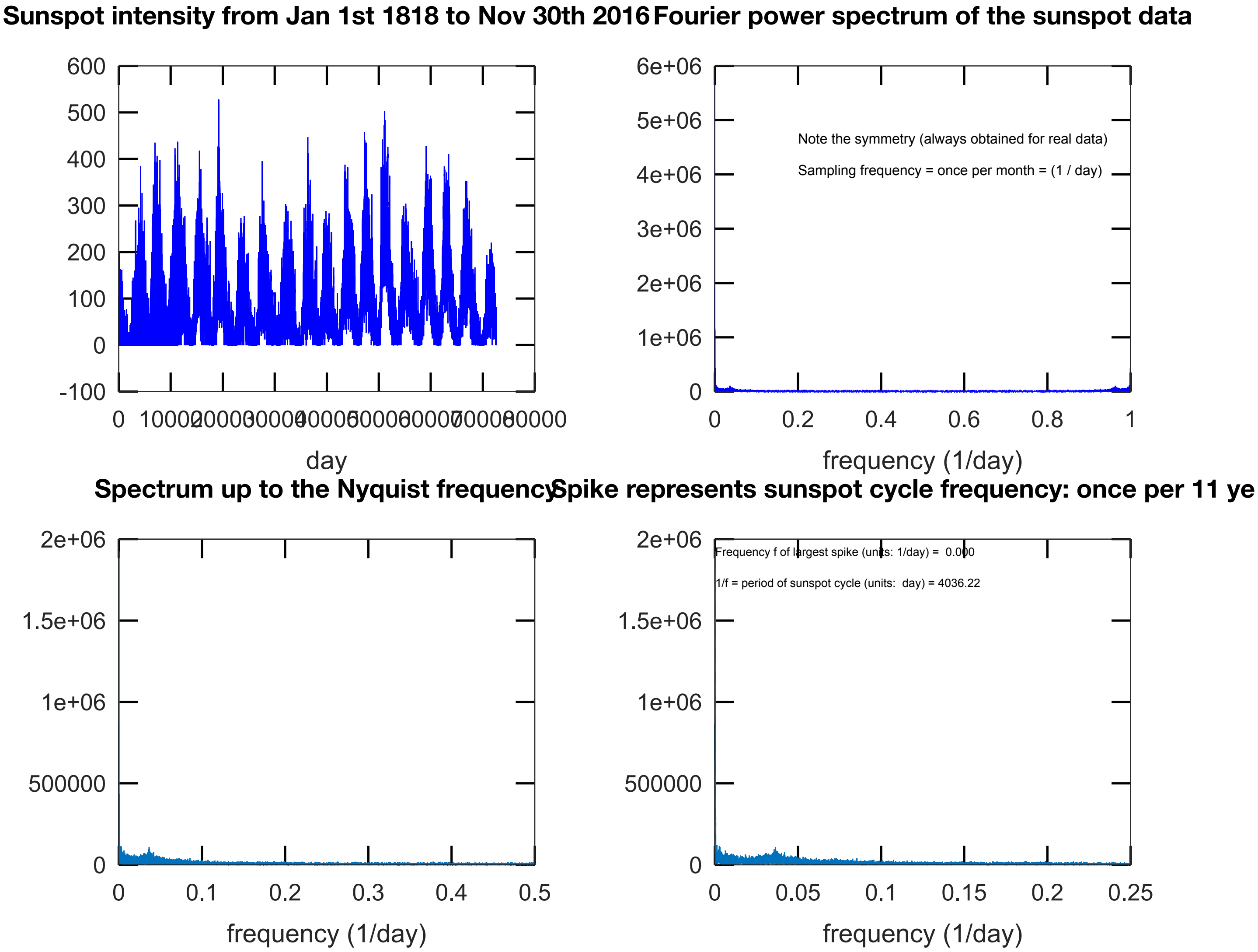
Is there a period at a frequency that is close to the period for the yearly data?

In [17]:

```
days = 1:72653;

subplot(2,2,1)
plot( days, daily, 'b' )
title('Sunspot intensity from Jan 1st 1818 to Nov 30th 2016')
xlabel('day')

power_spectrum = abs(fft(daily));
```

```
n = length(daily);

sampling_frequency = 1;  %  one sample / day
frequencies = linspace( 0, sampling_frequency, n );

subplot(2,2,2)
plot( frequencies, power_spectrum, 'b' )
text( sampling_frequency/5, max(power_spectrum)*0.80, 'Note the symmetry (always
obtained for real data)', 'fontsize', 6 )
text( sampling_frequency/5, max(power_spectrum)*0.70, 'Sampling frequency = once
per month = (1 / day)', 'fontsize', 6 )
title( 'Fourier power spectrum of the sunspot data' )
xlabel('frequency (1/day)')

n_over_2 = floor(n/2);
subplot(2,2,3)
plot( frequencies(2:n_over_2), power_spectrum(2:n_over_2) )
title( 'Spectrum up to the Nyquist frequency' )
xlabel('frequency (1/day)')
% text( sampling_frequency/8, max(power_spectrum)*0.90, 'Nyquist frequency = sam
pling frequency / 2', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.70, 'We ignore the 1st Fouri
er coefficient here;', 'fontsize', 6 )
% text( sampling_frequency/8, max(power_spectrum)*0.60, 'it is always just the s
um of the input data.', 'fontsize', 6 )

search_interval = 2:floor(n/4);  %  skip the first coefficient, which is always
the sum of the input data
spike_location = 1+find( power_spectrum(search_interval) == max(power_spectrum(s
earch_interval)) );
spike_frequency = frequencies(spike_location);
period_of_sunspot_cycle = 1/spike_frequency;

subplot(2,2,4)
plot( frequencies(search_interval), power_spectrum(search_interval) )
text( spike_frequency*1.05, power_spectrum(spike_location)*1.00,
      sprintf('Frequency f of largest spike (units: 1/day) = %6.3f',spike_freque
ncy), 'fontsize', 5 )
text( spike_frequency*1.05, power_spectrum(spike_location)*0.90,
      sprintf('1/f = period of sunspot cycle (units:  day) = %5.2f',period_of_su
nspot_cycle), 'fontsize', 5 )
title( 'Spike represents sunspot cycle frequency: once per 11 years' )
xlabel('frequency (1/day)')
hold on
plot( [spike_frequency spike_frequency], [0 power_spectrum(spike_location)], 'r-
-' )
hold off

printf("Spike frequency = %d per day", spike_frequency);
printf("Period of sunspot cycle = %d years", period_of_sunspot_cycle/365);
```

```
Spike frequency = 0.000247756 per day
Period of sunspot cycle = 11.0581 years
```



In [18]:

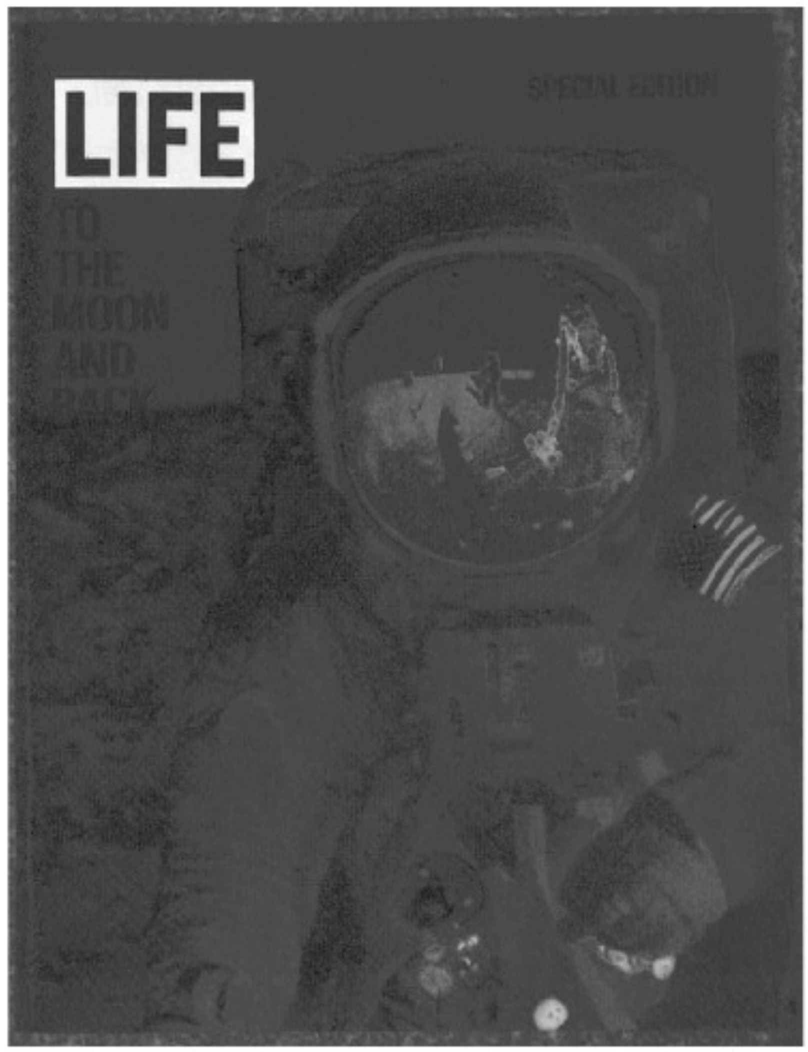
```
% The period of the monthly data is very close to the period of the yearly data,
% approximately 11 years.
```

Problem 3: A Photoshop Detector

A simple way to check if a $m \times n$ RGB image has been faked (with Photoshop, say) is to use the `reshape` function to convert it into a $(m\,n) \times 3$ matrix, compute the 3 principal components of its 3×3 covariance matrix, project the reshaped image on the second principal component, and then reshape the result back to a $m \times n$ grayscale image. If this grayscale image has bright or dark spots, it is possible that the color distribution in that area (and perhaps also that part of the image) has been altered.

For example, the image `LIFE_projected_on_2nd_PC.jpg` shows the result of projecting the image `LIFE.jpg` on the 2nd PC, and treating the result as a grayscale image. Notice the flag on Aldrin's shoulder and some buttons on his suit are bright, suggesting that they have been retouched. The buttons are bright red in `LIFE.jpg`.

LIFE magazine cover



LIFE cover projected on 2nd PC

3.1 Write a Photoshop detector

Write a function that takes the filename of an input image, and returns its 2nd PC image.

As a test, show the output of your program on `LIFE.jpg`.

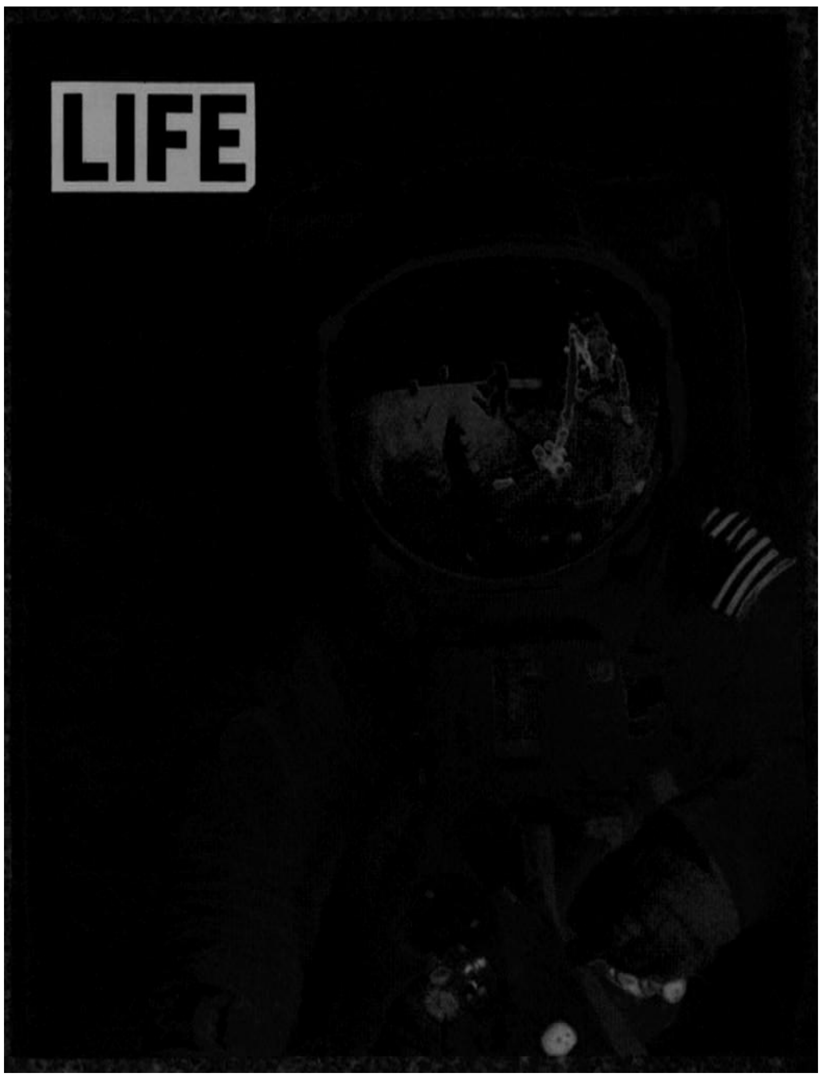
(With Octave, you can include the output image in this notebook with a <img src="..."> HTML tag.)

In [53]:

```
function [PC2image] = photoshop_detector(origimage)

orig = imread(origimage);
% [x, map] = rgb2ind(orig);
[m n l] = size(orig);
reshapedorig = double(reshape( orig , m*n, 3));

C = cov(reshapedorig);
[U,S,V] = svd(C);
PC2 = V(:,2);
projected = reshapedorig * PC2;
reshapedprojected = reshape( uint8( projected ), m, n);
% PC2image = ind2gray(reshapedprojected);
PC2image = reshapedprojected;

endfunction
```

In [54]:

```
PC2LIFE = photoshop_detector("LIFE.jpg");
imwrite(PC2LIFE, 'LIFE_PC2.jpg', 'jpg');
```



2nd PC image for LIFE magazine cover

3.2 Jennifer in Paradise

Show the output of your program on the photo `Jennifer_in_Paradise.jpg` included in the assignment .zip file. This image is allegedly the first image ever actually photoshopped -- a photo of the wife of the original developer of Photoshop, famous as a photoshopped image (http://www.telegraph.co.uk/technology/2016/02/05/the-most-famous-photoshopped-images-of-all-time/).

In [58]:

```
PC2Jennifer = photoshop_detector("Jennifer_in_Paradise.jpg");
imwrite(PC2Jennifer, 'Jennifer_PC2.jpg', 'jpg');
```

2nd PC image for "Jennifer in Paradise"

3.3 Find and comment on an image of Hillary Clinton or Donald Trump that has been photoshopped

Using e.g. Google image search, find an image of Hillary or Donald whose 2nd PC has clearly highlighted areas. Comment on the highlighting.

In [59]:

```
Hillary = imread("https://i.ytimg.com/vi/-dY77j6uBHI/maxresdefault.jpg");
imwrite(Hillary, 'Hillary.jpg', 'jpg');
PC2Hillary = photoshop_detector("Hillary.jpg");
imwrite(PC2Hillary, 'Hillary_PC2.jpg', 'jpg');
```

2nd PC image for Hillary's image

In [60]:

```
% We can see from the highlightings that the facial features, earings, hair, and the collar
% of Hillary had been modified so as to make her stand out more in the picture.
```

Problem 4: Graphs as Matrices

The goal of this problem is to analyze the graph (actually, a numeric matrix H). You can read it from one of the attached files `hero_social_network_50.tsv`, `hero_social_network_50.m`, `hero_social_network_50.py`.

This H matrix represent a core part of Marvel's ***Hero Social Network*** (a graph of social connections between superheroes. The entry h_{ij} of H is the strength of edges, so higher values mean that heroes i and j appear together more often.

The entire Marvel set of heroes is amazingly large; this dataset includes 206 heroes in the graph, obtained from the subset who appear in at least 50 comics. Despite their frequent appearance, this matrix is still pretty sparse -- apparently superheroes don't have hundreds of friends.

4.0 Read in the Hero Social Network

For example, you can read the file with ***dlmread()*** in Matlab; in Python either ad hoc parsing or perhaps the ***csv*** module.

Notice that the network is undirected, so H is real symmetric, and all entries of H are nonnegative.

In [2]:

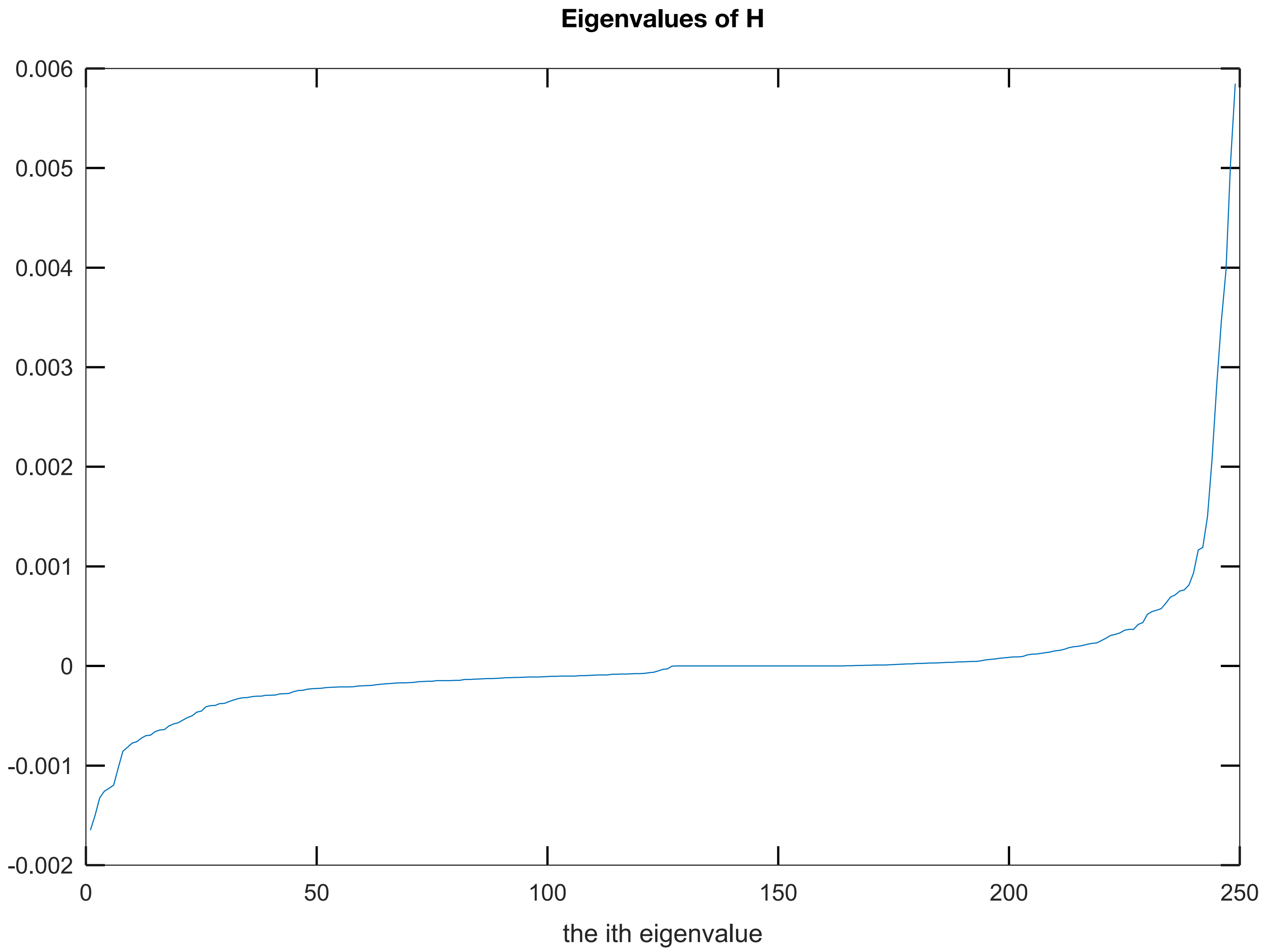
```
hero_social_network_50;
```

4.1 Eigenvalues

Compute the ***eigenvalues*** of H and plot them -- so that the i-th eigenvalue λ_i is plotted at position $(x, y) = (i, \lambda_i)$.

In [3]:

```
[Q L] = eig(Hero_network);
evals = diag(L);          % evals not ordered from large to small
plot(1:length(evals),evals);
title("Eigenvalues of H");
xlabel("the ith eigenvalue");
```



4.2 Largest Eigenvalue

Find the ***largest eigenvalue*** r of H. Is it equal to the ***spectral norm*** $||H||_2$?

In [4]:

```
MaxEval = max(evals)
SpectralNorm = norm(Hero_network,2)
printf("The largest eigenvalue of H is equal to the spectral norm of H.");
```

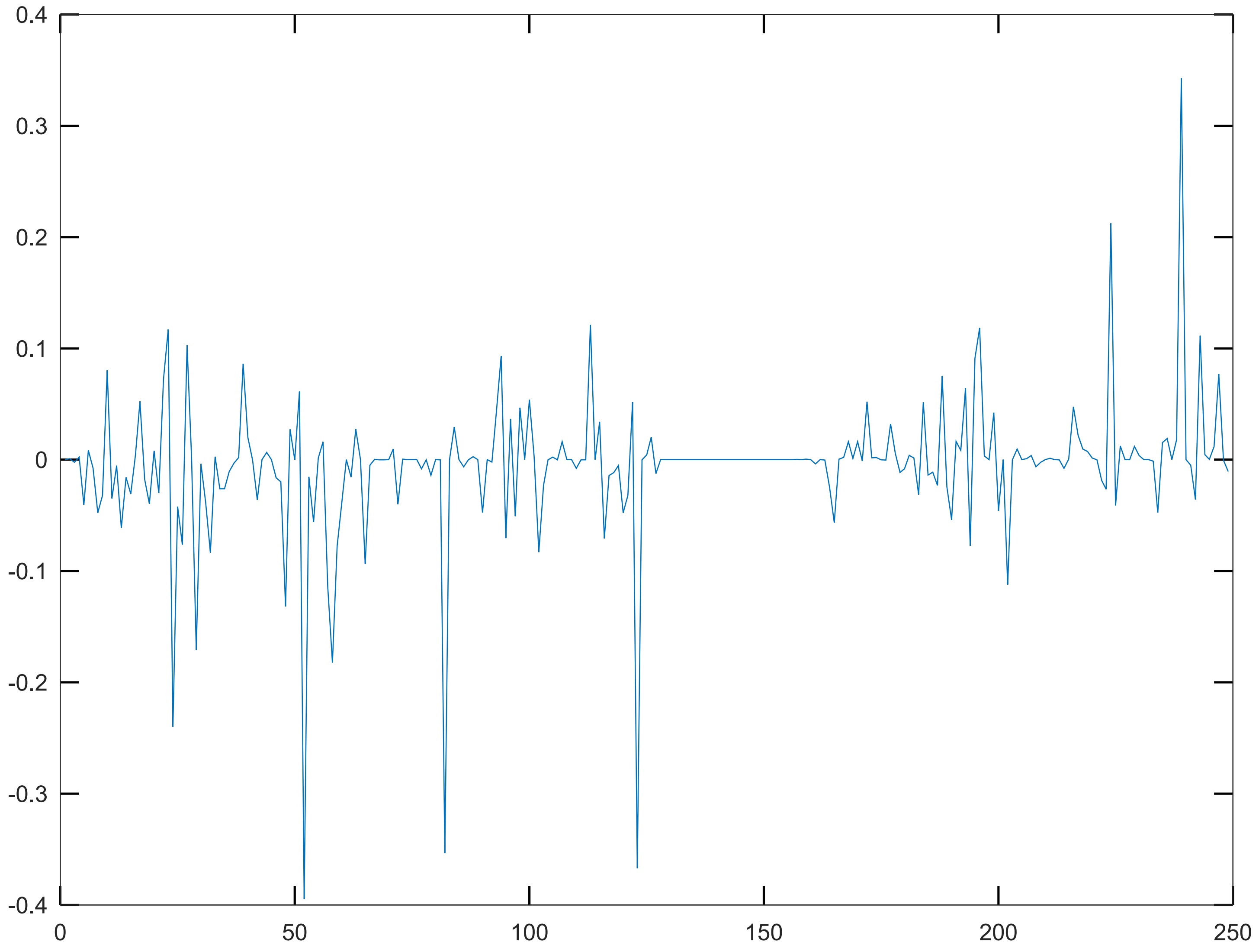
```
MaxEval =  0.0058488
SpectralNorm =  0.0058488
The largest eigenvalue of H is equal to the spectral norm of H.
```

4.3 Dominant Eigenvector

Find the ***eigenvector e*** of H that has this largest eigenvalue. Plot the sequence of entries of the eigenvector, showing that they are nonnegative. Which entry i has the largest value? (Who is the i-th hero?)

In [5]:

```
MaxEigenvector = Q(length(evals),:);
plot(1:249,MaxEigenvector)
```



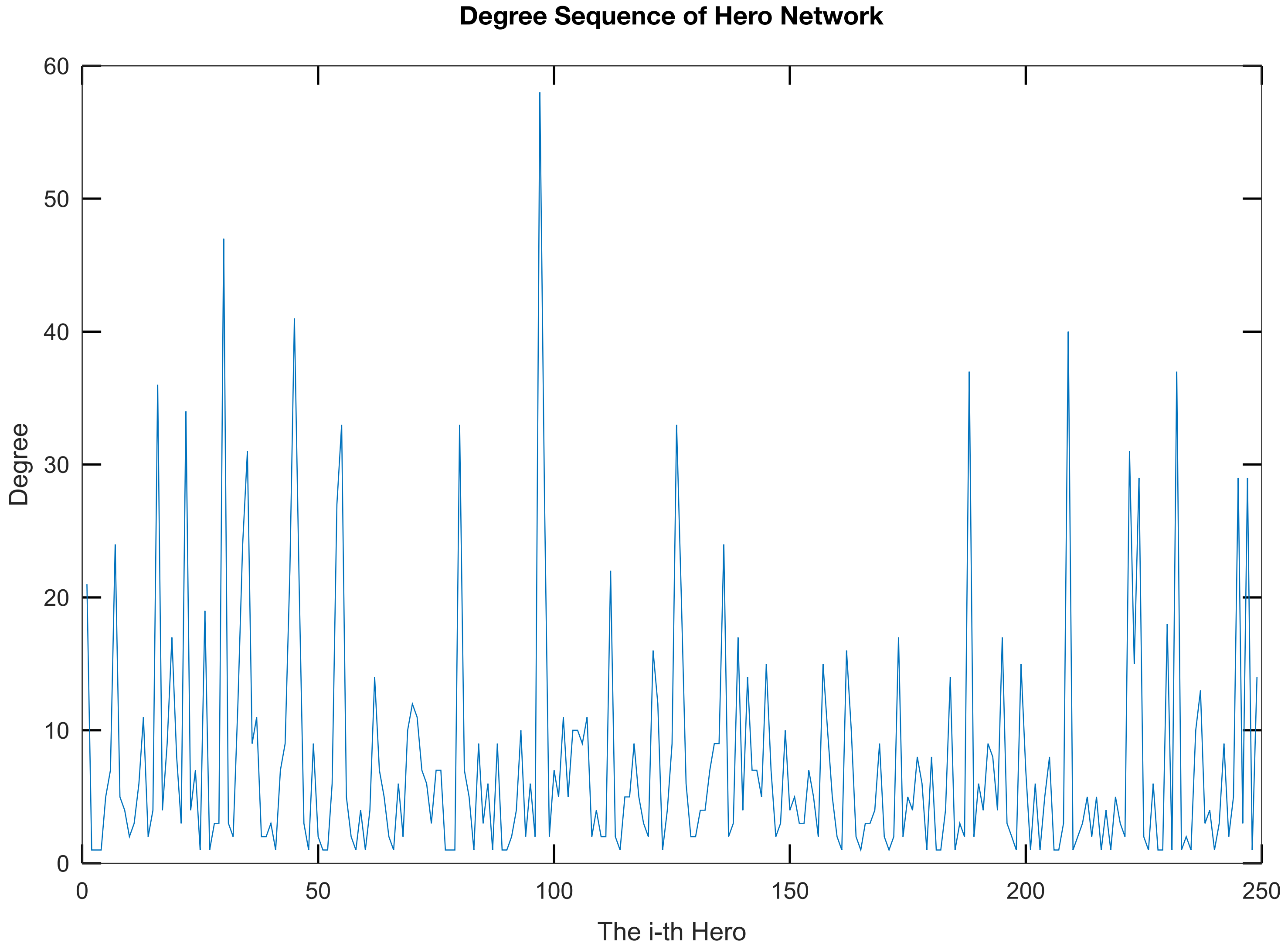
4.4 Node Degree Sequence

The ***degree sequence*** is a vector $\mathbf{d}$ of integer values whose i-th entry d_i is the *degree* of the i-th node. That is, d_i is the number of other superheroes to which the i-th hero has nonzero edges. Plot the sequence of degrees. What is the name of the hero i who has the largest value?

In [6]:

```
d = zeros(1,length(Hero_network));

for i = 1:length(Hero_network)
    for j = 1:length(Hero_network)
        if (i != j && Hero_network(i,j) > 0)
            d(1,i) = d(1,i) + 1;
        endif
    endfor
endfor

plot(1:length(Hero_network),d)
title("Degree Sequence of Hero Network")
xlabel("The i-th Hero")
ylabel("Degree")

printf('%s has the largest degree of %d.',names{maxindex},maxdegree);
```

```
error: 'maxindex' undefined near line 1 column 49
```



4.5 Graph Laplacian -- and Connected Components

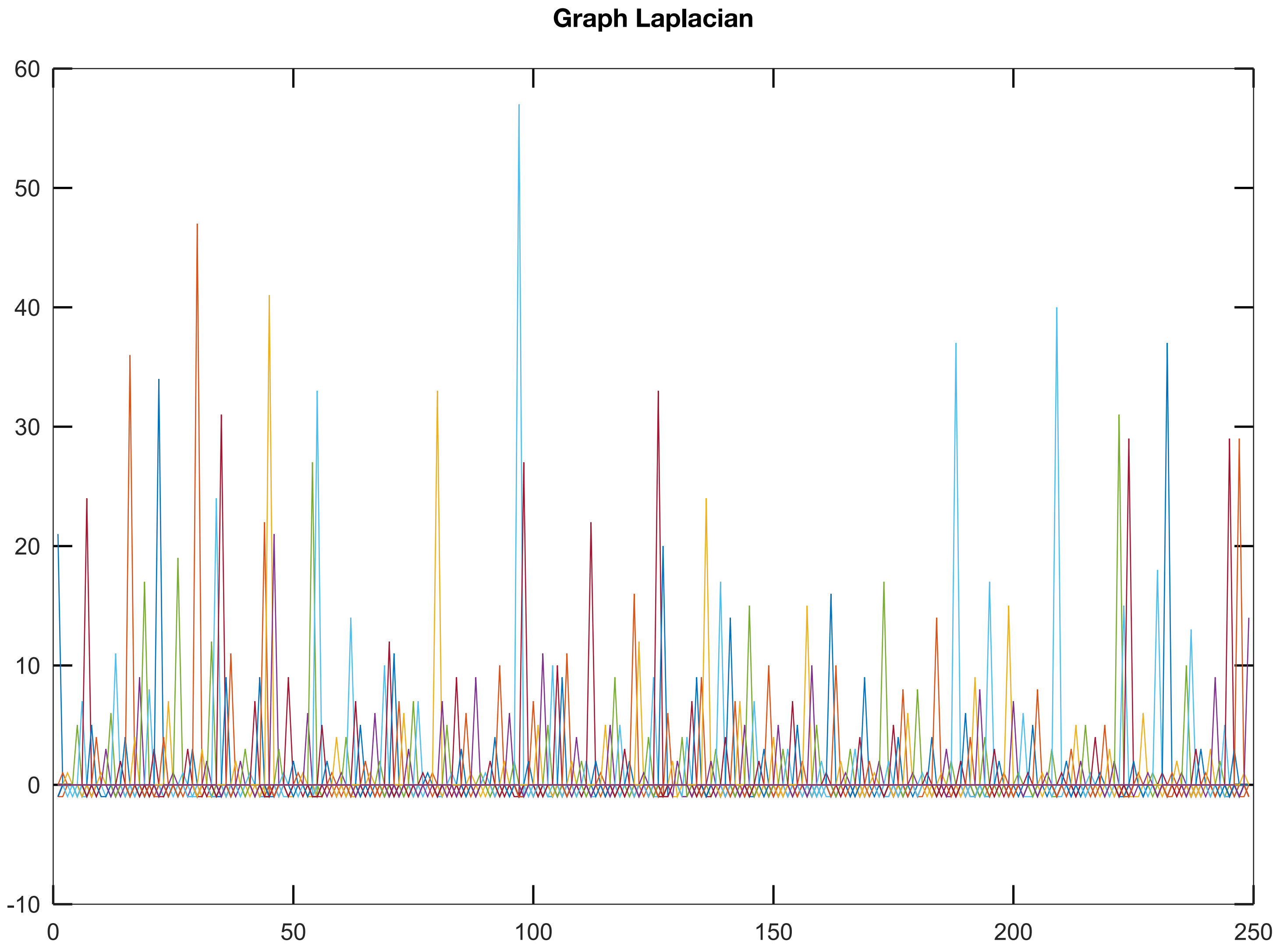
Compute the ***graph laplacian*** $L = D - A$, where $D = diag(\mathbf{d})$ is the diagonal matrix determined by the degree sequence, and A is the zero-one adjacency matrix for H (The entries of A are zero wherever they are zero in H, and are one wherever nonzero in H.)

What is the number of zero eigenvalues of the graph laplacian L, if we assume that all eigenvalues whose absolute value is below 1e-10 are zero? (This is supposed to be the number of connected components in the graph.)

In [8]:

```
A = zeros(size(Hero_network));

for i = 1:length(Hero_network)
    for j = 1:length(Hero_network)
        if (Hero_network(i,j) > 0)
            A(i,j) = 1;
        endif
    endfor
endfor

D = diag(d);

L = D - A;
plot(L)
title("Graph Laplacian")

[Qa La] = eig(L);

nZeros = 0;

evals_a = diag(La);

for k = 1:length(evals_a)
    if (abs(evals_a(k)) < 1^(-10))
        nZeros = nZeros + 1;
    endif
endfor

printf('There are %d zero eigenvalues of the graph laplacian L.', nZeros)
```

```
There are 41 zero eigenvalues of the graph laplacian L.
```



In [ ]: