Fourier Demos

Andreas Stahel

Version of 8th February 2018

1 Introduction

In the booklet [Stew13] by Ian Stewart find an excellent description of Fourier methods:

- **What does it say?**
  Any pattern in space and time can be thought of as a superposition of sinusoidal pattern with different frequencies.
- **Why is it that important?**
  The component frequencies can be used to analyze the pattern, create them to order, extract important features, and remove random noise.
- **What did it lead to?**
  Fourier's technique is very widely used, for example in image processing and quantum mechanics. It is used to find the structure of large biological molecules like DNA, to compress image data in digital photography, to clean up old or damaged audio records and to analyze earthquakes. Modern variants are used to store fingerprint data efficiently and to improve medical scanners.

For the Fourier series of a T–periodic function $f(t)$ use the real Fourier coefficients

$$a_n = \frac{2}{T}\int_0^T f(t)\,\cos(n\,\omega\,t)\,dt \quad \text{and} \quad b_n = \frac{2}{T}\int_0^T f(t)\,\sin(n\,\omega\,t)\,dt$$

or the complex Fourier coefficients

$$c_n = \frac{1}{T}\int_0^T f(t)\,e^{-in\omega t}\,dt \quad \text{with} \quad c_n = \frac{1}{2}\,(a_n - i\,b_n)$$

where $\omega = \frac{2\pi}{T}$. The trigonometric and the complex Fourier series are given by

$$f(t) \sim \frac{a_0}{2} + \sum_{n=1}^{\infty}\left(a_n\cos(n\omega t) + b_n\sin(n\omega t)\right) \quad \text{and} \quad f\,(t) \sim \sum_{n=-\infty}^{\infty} c_n e^{in\omega t}$$

For real valued functions $f(t)$ we have

$$f\,(t) \sim c_0 + 2\,\mathrm{Re}\left(\sum_{n=1}^{\infty} c_n\,e^{in\omega t}\right)$$

2 Fourier Series Applet

On the web site www.falstad.com/fourier/ find an applet to play with Fourier series, as shown in Figure 1. You can:

- choose different functions
- modify the Fourier coefficients
- listen to the corresponding sound
- change the number of coefficients to be used

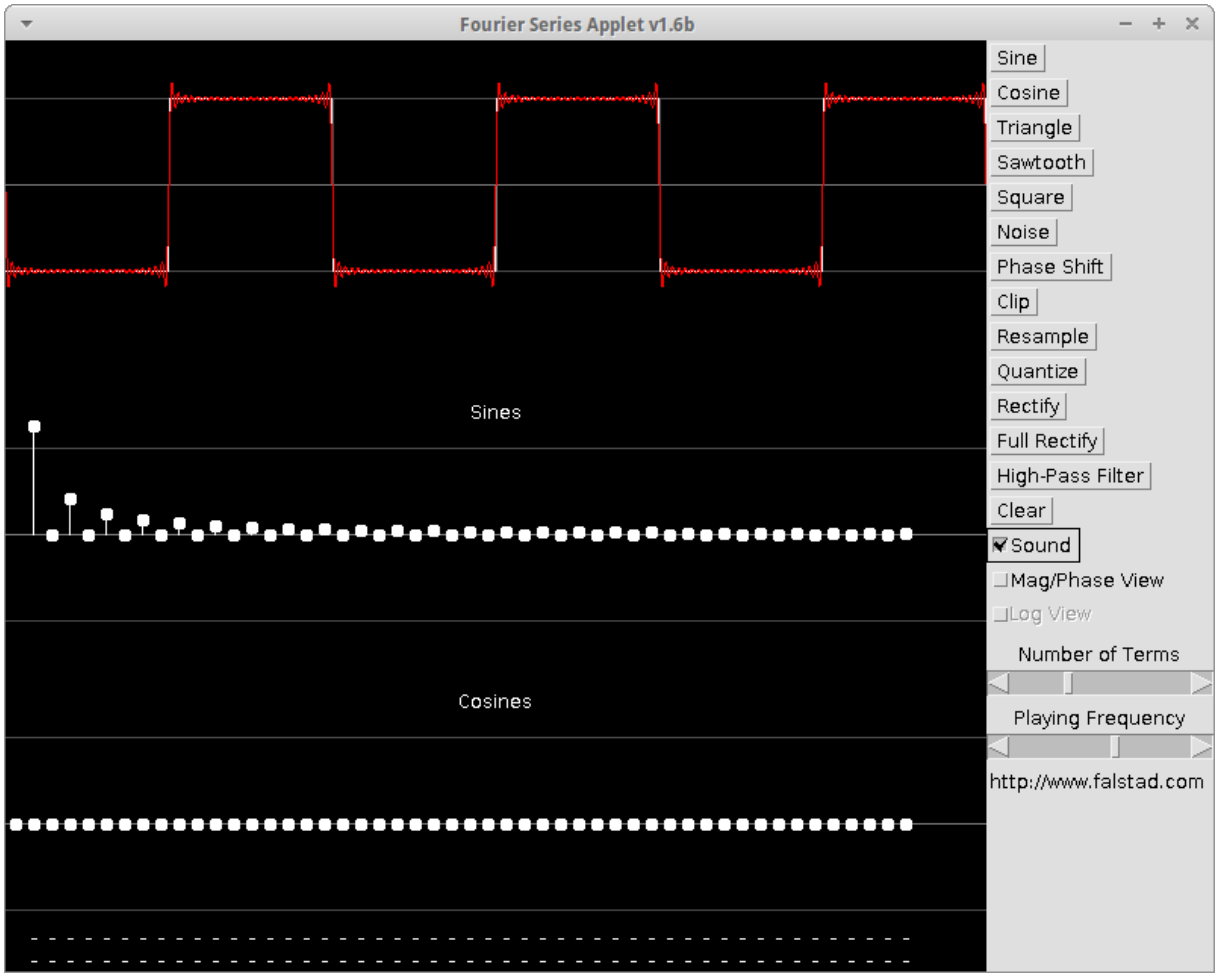

Figure 1: Fourier Series Applet

3 Phasor Factory

On the site www.jhu.edu/signals/phasorapplet2/phasorappletindex.htm one used to find a working Phasor Factory. Unfortunately it does not work any more. The complex Fourier approximation can by visualized as curves in the complex plane $\mathbb{C}$. Each contribution $c_n \exp(i\,n\,t)$ corresponds to a circle with radius $|c_n|$, parameterized with angular velocity n and a phase shift of $\phi_n = \arg(c_n)$. These circles can be be added to obtain curves in the complex plane $\mathbb{C}$.

$$
\begin{aligned}
f_1(t) &= c_1 \exp(i\,t) = |c_1|\,(\cos(t + \arg(c_1)) + i \sin(t + \arg(c_1))) \\
&\quad \text{circle in } \mathbb{C} \text{ with radius } |c_1| \text{ and angular velocity } 1 \\
f_2(t) &= c_1 \exp(i\,t) + c_2 \exp(i\,2\,t) \\
&\quad \text{sum of two circles in } \mathbb{C} \text{ with radii } |c_1| \text{ and } |c_2| \text{ and angular velocities } 1 \text{ and } 2 \\
f_3(t) &= f_2(t) + c_3 \exp(i\,3\,t) \quad \text{sum of three circles in } \mathbb{C} \\
f_N(t) &= \sum_{n=1}^{N} c_n \exp(i\,n\,t) \quad \text{sum of } N \text{ circles in } \mathbb{C}
\end{aligned}
$$

Using *Octave*/`MATLAB` one can generate animations, with a snapshot shown in Figure 2. On the left find the real and imaginary part of the Fourier expression

$$f_N(t) = \sum_{n=1}^{N} c_n \exp(i\,n\,t) \in \mathbb{C}$$

This is the superposition of N circles. On the right the real part of $f_N(t)$ is plotted as function of time t. To show the approximation of a rectangular function by 5 Fourier terms use

```
Phasor(5,'rectangle')
```

Since two of the five term in the Fourier approximation are zero Figure 2 displays the superposition of only three circles.

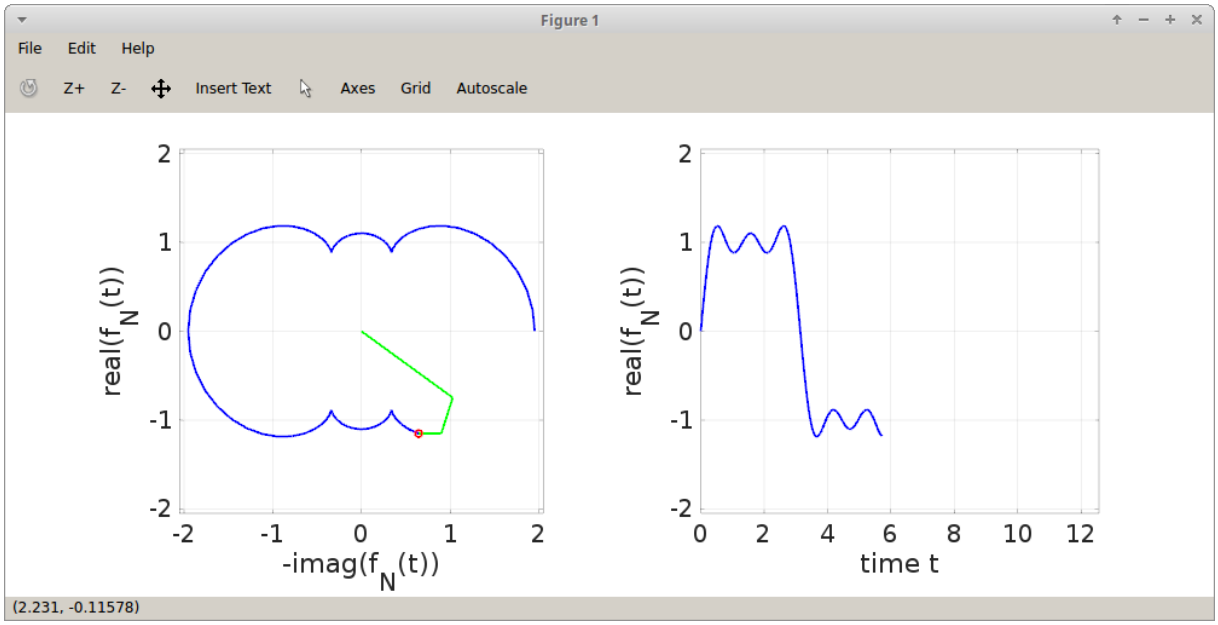

Figure 2: Phasor Factory

Phasor.m

```
function Phasor(N,func,plotpoints)
% Phasor(N,func,plotpoints)
%   generate a phasor animation with N contributions for the function 'func'
%   the constant contribution of the Fourier approximation is not used
%
%   possible values for func: 'rectangle', 'sawtooth', 'triangle', 'halfwave'
%
%   func can also be a vector of complex Fourier coefficients
%     2*real(sum_{k=1}^N c(k)*exp(i*k*t))
%
%   plotpoints: optional argument for the number of plot points to be used
%
%   Example 1:
%       Phasor(3,'rectangle')
%
%   Example 2:
%       N=1024; c = fft(linspace(0,2*pi,N)<1)/N;
%       Phasor(20,c(2:N),501)

%% Copyright (C) 2018  Andreas Stahel  <Andreas.Stahel@bfh.ch>
%%
%% This program is free software; you can redistribute it and/or modify
%% it under the terms of the GNU General Public License as published by
%% the Free Software Foundation; either version 2 of the License, or
%% (at your option) any later version.
%%
%% This program is distributed in the hope that it will be useful,
%% but WITHOUT ANY WARRANTY; without even the implied warranty of
%% MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%% GNU General Public License for more details.
%%
%% You should have received a copy of the GNU General Public License
%% along with this program; If not, see <http://www.gnu.org/licenses/>.

if nargin<2
  help Phasor
  error('invalid call of Phasor')
end%if

if nargin<=2
  t = linspace(0,4*pi,201);
else
  t = linspace(0,4*pi,plotpoints);
end%if

xt = []; yt = xt; xpt =  xt; ypt = xt;

if isnumeric(func)
   N = min(N,length(func));
   c = func(1:N);
   else
   switch lower(func)
   case{'rectangle','rect'}
      c = -i/2 * (4/pi./[1:N]).*mod([1:N],2);
   case{'sawtooth','saw'}
      c = i*(-1).^[1:N]./[1:N];
```

```
    case{'triangle' ,'tria'}
       cc = mod([1:N],2);
       c = 4/pi^ 2*([1:N].^(-2)).*mod([1:N],2);
    case{'halfwave','half'}
       nn = 1024; c = fft(max(sin(linspace(0,2*pi,nn)),0))/nn;
       c = c(2:end);
    otherwise
       error('invalid value for function name, use rectangle, sawtooth, triangle, halfwave')
    end%switch
end%if
omega = 1:length(c);

%% determine the range to be displayed
dom = 2.1*max(abs(sum(diag(c(1:N))*exp(i*omega(1:N)'*t))));

fig = get(gcf);
if exist('OCTAVE_VERSION')
  set(gcf,'position',[fig.position(1),fig.position(2) 1000,400])
  else
  set(gcf,'position',[fig.Position(1),fig.Position(2) 1000,400])
end%if
for ii = 1:length(t)
  xt = [xt,t(ii)];
  yt = [yt,2*real(sum(c(1:N).*exp(i*omega(1:N)*t(ii))))];
  subplot(1,2,2)
  plot(xt,yt,'b');
  axis([0,4*pi,-dom,dom])
  xlabel('time t'); ylabel('real(f_N(t))')
  drawnow()
  subplot(1,2,1)
  wheel = 0;
  for nn = 1:N
    wheel = [wheel;2*sum(c(1:nn).*exp(i*omega(1:nn)*t(ii)))];
  end%for
  xpt = [xpt,imag(-wheel(end))];
  ypt = [ypt,real(wheel(end))];
  plot(xpt,ypt,'b',imag(-wheel),real(wheel),'g',imag(-wheel(end)),real(wheel(end)),'or')
  axis(dom*[-1 1 -1 1])
  xlabel('-imag(f_N(t))'); ylabel('real(f_N(t))')
  axis square
end%for
```

4 Audio Spectrum with *Octave* or MATLAB

With MATLAB/*Octave* you can record sound by using the buult-in sound card. Then compute the spectrum and display, all in real time! Find the graphical result in Figure 3 and the corresponding code below.

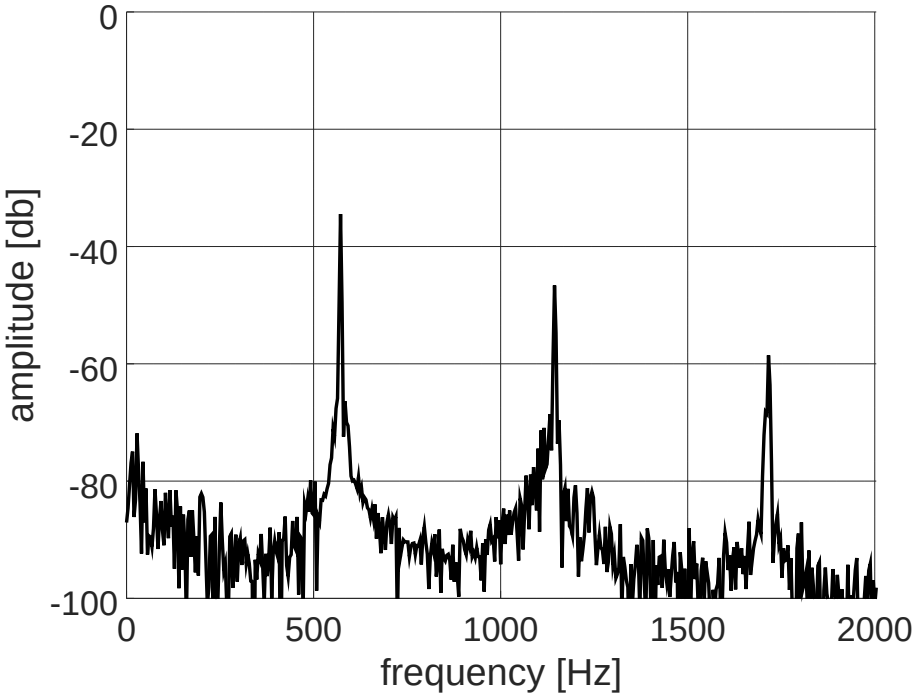

Figure 3: Audio Spectrum on a dezibel scale

To make this code work on your computer use the command `audiodevinfo()` to determine the ID of your audio device.

AudioSpectrumAnalyszerDezibel.m

```
%% code by Andreas Stahel, Bern University of Applied Sciences
%% loosely based on the code spectrum_analyser.m from playrec
%% see http://www.playrec.co.uk/

sf = 44100; % set the sampling frequency
T = 1/4;  % length of time interval for one data set
n0 = sf*T; % number of data points in one sample
freq = [0:n0-1]/T; % set of all frequencies
n = find(freq>2000,1); % display only frequencies smaller than 2.0 kHz

% to find the ID of sound device
%aa = audiodevinfo()
%aa.input.ID
% ID = 4  % on my old laptop
ID = 14;  % on my current Laptop
% ID = 7; % on my desktop
% ID = 2 % on laptop of Pierre-André with Windows
RecObj = audiorecorder(sf,16,1,ID);

%%%%%%%% set up the graphics
fftFigure = figure(1);   % initialize graphics window
```

```
clf
% set the axis for the graphics with all properties, e.g. scaling
fftAxes = axes('parent',fftFigure,'xlimmode','manual','ylimmode','manual',...
              'xscale','linear','yscale','linear',...
              'xlim',[0 (n-1)/T],'ylim',[-100 0]);
% initialize the data, at first with zero values
fftLine = line('XData',[0:(n-1)]/T,'YData',zeros(1,n));
xlabel('frequency [Hz]'); ylabel('amplitude [db]'); % label the axis
drawnow(); % draw the graphics window

keyDownListener = @(src,event) keyboard;  % to replace Octave's kbhit()
% set(fftFigure,'HitTest','on')
set(fftFigure,'KeyPressFcn',keyDownListener)

% loop with the measurements
for k = 1:20/T  % at most 20 sec, stop by hitting a key
   recordblocking(RecObj,T)  % record for T seconds
   s = getaudiodata(RecObj); % get the raw data out
   spec = abs(fft(s(:,1)))/(2*n0);     % compute the spectrum
   set(fftLine,'YData',20*log10(spec(1:n)));  % write the data to the window
   drawnow();                          % update the display
end%for
```

References

[Stew13] I. Stewart. *Seventeen Equations that Changed the World.* Profile Books Limited, 2013.