Generating Fourier Series with Mathematica

Generating Fourier series with Mathematica is extremely simple provided, of course, that you are familiar with Mathematica. For example, for an even, unit amplitude, square wave with unit period, the Fourier series is given by

$$f(t) = \frac{4}{\pi} \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{2n-1} \cos((2n-1)2\pi t)$$ \hspace{1cm} (1)

A Mathematica command to define this function is

$$f[t_, nTerms_] := \frac{4}{\pi} \text{Sum}[-1^{(n+1)} \cos[(2 n - 1)2 \pi t]/(2 n - 1), \{n, 1, nTerms\}]$$

This command defines a function \( f \) with two arguments. The first, \( t \), is the time and the second, \( nTerms \), is the number of terms to be used when evaluating the series. After issuing this command one can plot the function \( f \) with, for example, a command such as

$$\text{Plot}[f[t, 20], \{t, 0, 2\}]$$

This says to plot the function using the first twenty harmonics with time ranging from zero to two. Alternatively, one can evaluate the function at a point. For example, the statement \( f[0.123, 15] \) would evaluate the series at the point 0.123 using the first 15 non-zero harmonics.

However, how would one plot the series in Matlab? Unfortunately you have to work a bit harder (at least given my limited knowledge of Matlab) but it is straightforward and generating such plots is a good way to check that you have the series correct. Here is a transcript, with comments, of a session which generates a plot of the series shown above. The first 20 non-zero harmonics are used.

```matlab
% Set the number of harmonics in the series.
nTerms=20;

% n = a vector of integers of all the harmonic numbers
```
n=1:nTerms;

% an = coefficients of harmonics
an=4*(-1).ˆ(n+1)./(pi*(2*n-1));

% t = time at which function is sampled
t=linspace(0,2,401);

% f = ultimately will contain the Fourier series of the
% desired function. Initialize to zero. Length
% is the same as the number of time samples.
f = zeros(1,length(t));

% Use a loop to add each harmonic to the series.
for n=1:nTerms
    f = f + an(n)*cos((2*n-1)*2*pi*t);
end

% Plot the function.
plot(t,f)
xlabel(‘time, t’)
ylabel(‘f(t)’)

The plot generated by these commands is shown below.
Generating the Fourier series for other functions would be done in a similar fashion. The general approach is

1. Generate a vector giving the integer values of the indices.

2. From that vector generate the $a_n$ and $b_n$ coefficients.

3. Create a vector with the desired time samples.

4. Initialize to zero a vector representing the function and ensure it has the same length as the number of time steps. Alternatively, if there is a non-zero $a_v$ term, the vector could be initialized with that constant using a command such as

   $$f = a_v \times \text{ones}(1, \text{length}(t));$$

   where we assume $a_v$ has been set previously to the desired value. However, this command is rather inefficient in that it potentially involves a lot of multiplication when all we need to do is generate a vector with the same value in every element. A better way to do this is the \texttt{repmat()} command which merely repeats a value the specified number of times. The command would be

   $$f = \text{repmat}(a_v, 1, \text{length}(t));$$

5. Use a for loop to add to the function the contributions from each harmonic.