```
function [freq,coeff,APspec] = fourier_coeff(fun,t0,T,M,N,method,res,num_P)

%

% Calculate the Fourier coefficients of the series expansion of a function,

% and the amplitude and phase spectra. The script contains some theory and

% 3 different methods to calculate the coefficients.

%

%

%USAGE

%-----

% fourier_coeff(fun,t0,T)

% fourier_coeff(fun,t0,T,M)

% fourier_coeff(fun,t0,T,M,N)

% fourier_coeff(fun,t0,T,M,N,method)

% fourier_coeff(fun,t0,T,M,N,method,res)

% fourier_coeff(fun,t0,T,M,N,method,res,num_P)

%

%

%INPUT

%-----

% - FUN   : character string representing the function with "t" as the

%   independent variable (e.g. '10*cos(2*pi*3*t-pi/4)'). Dot-arithmetic

%   operators must be used (.* ./ .\ .^). FUN must be defined for [T0,T0+T]

% - T0    : initial "t" for the definition of FUN

% - T     : period of the function

% - M     : number of frequencies (default: 5)

% - N     : number of data points per period (default: 100)

% - METHOD: 1 (least-squares), 2 (integrals [default]) or 3 (FFT)
```

```
% - RES   : 1 (plot results) or 0 (do not plot [default])
% - NUM_P : number of periods the function will be sampled at. Only
%   effective when RES=1 (default: 1)
%
%
%OUTPUT
%------
% - FREQ : frequencies
% - COEFF: Fourier series coefficients in the form in the form [a0 a1 ...
%   aM b1 ... bM], where
%   f(t) = a0 + \sum_{m=1}^M { a_m cos(2pi m t/T) + b_m sin(2pi m t/T) }
%   So the corresponding frequencies are: 0, 1/T, 2/T, ..., M/T
% - APSPEC: the first column contains the amplitude spectrum, and the
%   second column the phase spectrum
% - If RES=1:
%   Figure the original function and the Fourier series expansion, and
%   another with the amplitude and phase spectra
%
%
%
%=====================================================================
%          Fourier expansion of a periodic function f(t)
%=====================================================================
%
% T: period of f(t)
% M: number of harmonics (the equalities hold when M->infinity)
% w0 = 2pi/T
```

```
%
%
% Three equivalent forms:
%
%
% 1) Sine-cosine form
%    ----------------
%  f(t) = a0 + \sum_{m=1}^M { a_m cos(m w0 t) + b_m sin(m w0 t) }
% Fourier coefficients:
%   a0  = 1/T \int_0^T f(t) dt (DC term)
%   a_m = 2/T \int_0^T f(t) cos(m w0 t) dt
%   b_m = 2/T \int_0^T f(t) sin(m w0 t) dt
%
%
% 2) Amplitude-phase form
%    --------------------
%  f(t) = A0 + \sum_{m=1}^M { A_m cos(m w0 t - phi_m) }
% Fourier coefficients:
%   a0    = A0 (DC term)
%   a_m   = A_m cos(phi_m)
%   b_m   = A_m sin(phi_m)
%   |A_m| = sqrt( a_m^2 + b_m^2 )  (amplitude)
%   phi_m = arctan(b_m/a_m)        (phase)
% Spectral plots:
% - Amplitude spectrum: amplitude vs. harmonic frequency
% - Phase spectrum: phase vs. harmonic frequency
%
```

```
%
% 3) Complex exponential form
%    ------------------------
%  f(t) = \sum_{m=-M}^M { c_m exp^{i m w0 t} }
% Complex Fourier coefficients:
%   c_m = 1/T \int_0^T f(t) exp^{-i m w0 t} dt
%
%
% Relationship between froms (3) and (1):
%   c0     = a0
%   c_m    = ( a_m - ib_m )/2       , m=1,2,...,M
%   c_{-m} = c_m* = ( a_m + ib_m )/2 , m=1,2,...,M
%
%   a0  = c0
%   a_m = c_m + c_{-m} = 2 real(c_m)     , m=0,1,...,M
%   b_m = i(c_m - c_{-m}) = -2 imag(c_m) , m=1,...,M
%
%
% Relationship between froms (3) and (2):
%   c_m = |c_m|exp(i theta_m)
% For m=0,1,...,M:
%    |c_m| = |c_{-m}| = |A_m|/2
%    tan(theta_m)    =  tan(phi_m)
%    tan(theta_{-m}) = -tan(phi_m)
%
% Obs.: All the integrals above must be made within a period, that is, can
% be from t0 to t0+T for an arbitrary t0
```

```
%
% REFERENCE:
% Tan, Li. Digital signal processing: fundamentals and applications.
% Academic Press, USA, 2008 - pp. 709-711
%
%
%==============================================================================
%               Calculating the Fourier coefficients
%==============================================================================
%
% This function calculates the Fourier coefficients using three methods:
%
%
% 1) This method explores the fact that Fourier coefficients give the best
%    least-squares fit when a function is expanded in a set of orthonormal
%    functions:
% Sampling the function: f(tn) = f_n, tn = n T/N, n=1,2,...N
% => f_n = a0 + \sum_{m=1}^M { a_m cos(2 pi m n/N) + b_m sin(2 pi m n/N) }
% Problem statement:
%   Given a column vector f = [f1 f2 ... fN]', find the set of
%   coefficients coef = [a0 a1 ... an b1 ... bn]' that best fit the
%   expansion above.
% Definition:
% A = [1 cos(w0.t1) ... cos(n.w0.t1) sin(w0.t1) ... sin(n.w0.t1)]
%     [1 cos(w0.t2) ... cos(n.w0.t2) sin(w0.t2) ... sin(n.w0.t2)]
%     [...   ...    ...   ...         ...    ...   ...     ]
%     [1 cos(w0.tN) ... cos(n.w0.tN) sin(w0.tN) ... sin(n.w0.tN)]
```

```
% Therefore, we can use ordinary least-squares to find the "coef" matrix
% such that: f=A.coef => coef = A^(-1)f
%
%
% 2) This method finds the a_m and b_m coefficients from the sine-cosine
%    form above, using all the ways Matlab offers to calculate integrals.
%
%
% 3) This method utilizes the Discrete Fourier Transform (DFT)
%
% Discrete Fourier Transform (DFT) of f(t):
%   F_m = \sum_{n=0}^{N-1} { f_n exp^{-2 pi i m n/N} }
%   - m = 0,1,...,N-1
%   - f(t) was sampled in N samples: f_n = f(t_n), n = 0,1,...,N-1
%
% Inverse DFT: f_n = 1/N \sum_{m=0}^{N-1} { F_m exp^{2 pi i m n/N} }
%
% REFERENCE:
% >> help fft.m
%
%
% From the integral definition of c_m above, one can relate it to F_m:
%  Let us choose t_n/T = 0,1/N,...,(N-1)/N = n/N, n=0,1,...,N-1
%  => c_m \approx 1/T \sum_{n=0}^{N-1} { f_n exp^{-2 pi i m n/N } \Delta t
%  But \Delta t/T = 1/N. Therefore:
%  N c_m \approx F_m
%
```

```
% Guilherme Coco Beltramini (guicoco@gmail.com)
% 2011-Jul-27, 12:09 am


thresh = 10^(-8); % threshold to consider the values 0
% (for numerical precision in the phase estimation)


% Input
%==========================================================================
if nargin<8
   num_P = 1;
elseif ~isnumeric(num_P) || num_P<1 || floor(num_P)~=ceil(num_P)
   error('Invalid number of periods')
end
if nargin<7
   res = 0;
elseif ~isequal(res,0) && ~isequal(res,1)
   error('Invalid results option')
end
if nargin<6
   method = 2;
elseif ~isequal(method,1) && ~isequal(method,2) && ~isequal(method,3)
   error('Invalid method')
end
if nargin<5
   N = 100;
```

```
elseif ~isnumeric(N) || N<1 || floor(N)~=ceil(N)
   error('Invalid number of data points per period')
end
if nargin<4
   M = 5;
elseif ~isnumeric(M) || M<1 || floor(M)~=ceil(M)
   error('Invalid number of frequencies')
end


% Initialize
%============================================================================

t     = linspace(t0,t0+num_P*T,num_P*N)';
num_P = N; % number of data points per period

w0 = 2*pi/T;
f_inline = inline(fun,'t');
try
   y = f_inline(t);
catch ME
   if ~isempty(strfind(ME.message,'Inner matrix dimensions must agree'))
      disp('Dot-arithmetic operators must be used (.* ./ .\ .^)')
   end
   error(ME.identifier,ME.message)
end
y = y(:); % y must be a column vector
```

```
if method==3 && num_P<M+1
    method = 2;
    fprintf('Changing to method %d\n',method)
end


% Calculate the Fourier coefficients
%===============================================================================

switch method

%===============================================================================
case 1    % METHOD 1
%===============================================================================

A = zeros(num_P,2*M+1);
A(:,1) = 1;
t_aux = t(1:num_P);
for m=2:M+1
    A(:,m)   = cos(w0*(m-1)*t_aux);
    A(:,m+M) = sin(w0*(m-1)*t_aux);
end

% Same as the for loop above (apparently it takes Matlab the same time):
%t_aux = w0*repmat((1:M),N,1).*repmat(t,1,M);
%A = zeros(N,2*M+1); A(:,1) = 1/2;
%A(:,2:M+1) = cos(t_aux);
%A(:,M+2:2*M+1) = sin(t_aux);
```

```
coeff = A\y(1:num_P);


%==========================================================================
case 2    % METHOD 2
%==========================================================================

coeff = zeros(2*M+1,1);

% Five ways of calculating the integrals, in decreasing order of time to
% evaluate the coefficients:
% 1) trapz(x,y)
% 2) quadl(fun,a,b)
% 3) quad(fun,a,b)
% 4) quadv(fun,a,b)
% 5) quadgk(fun,a,b)

% 1) trapz:
% - depends on sampling
% - can be applied to any set of data
% - faster
% yaux = A(1:num_P,:).*repmat(y(1:num_P),1,2*M+1);
% for m=1:2*M+1
%     coeff(m) = trapz(t(1:num_P),yaux(:,m));
% end
% coeff(1) = 2*coeff(1);
% coeff    = 2/T*coeff;
```

```
% 2-5) quadX:
% - do not depend on sampling
% - function must be known explicitly
% - slower
coeff(1) = 1/T*quadl(f_inline,t0,t0+T);
f_aux = inline(['(' fun ')' '.*cos(w0*m*t)'],'t','m','w0');
for m=1:M
   coeff(m+1) = quadl(f_aux,t0,t0+T,[],[],m,w0);
end
f_aux = inline(['(' fun ')' '.*sin(w0*m*t)'],'t','m','w0');
for m=1:M
   coeff(m+M+1) = quadl(f_aux,t0,t0+T,[],[],m,w0);
end
coeff(2:end) = 2/T * coeff(2:end);


%==========================================================================
case 3    % METHOD 3
%==========================================================================

coeff = fft(y(1:num_P),num_P)/num_P; % complex Fourier coefficients

% FFT = fft(X,N) => FFT(1)=DC term, FFT(2)=FFT(N), FFT(3)=FFT(N-1), ...
% FFT(k)=FFT(N-k+2), ..., FFT(N/2)=FFT(N/2+2) if N is even
%                   FFT((N+1)/2)=FFT((N+3)/2) if N is odd
% FFT(k) corresponds to increasing frequencies as k increases from 2 to N/2
% for even N, or from 2 to (N+1)/2 for odd N.
```

```
%

% FFTsh = fftshift(fft(X,N)) =>

% - N even: FFTsh((N+2)/2)=DC term, FFTsh(2)=FFTsh(N), FFTsh(3)=FFTsh(N-1),

% ..., FFTsh(k)=FFTsh(N-k+2), ..., FFTsh(N/2)=FFTsh(N/2+2)

% - N odd: FFTsh((N+1)/2)=DC term, FFTsh(1)=FFTsh(N), FFTsh(2)=FFTsh(N-1),

% ..., FFTsh(k)=FFTsh(N-k+1), ..., FFTsh((N-1)/2)=FFTsh((N+3)/2)

% FFTsh(k) corresponds to decreasing frequencies as k increases from 2 to

% N/2 for even N or (N-1)/2 for odd N.


coeff = coeff(1:M+1);

coeff(2:end) = coeff(2:end).*exp(-2*pi*1i*t0/T*(1:M)).';

coeff = [real(coeff(1)) ; 2*real(coeff(2:end)) ; -2*imag(coeff(2:end))];


end



% Amplitude and phase spectra

%==========================================================================

freq  = (0:1:M)'/T;

tmp   = coeff;

Aspec = sqrt( tmp(1:(M+1)).^2 + [0;tmp(M+2:end).^2] );

tmp(abs(tmp)<thresh) = 0;

Pspec = [0 ; atan2(tmp(M+2:end),tmp(2:M+1))];

%Pspec = [0 ; atan(tmp(M+2:end)./tmp(2:M+1))];

Pspec(isnan(Pspec)) = 0;

APspec = [Aspec Pspec];
```

```
% Show results
%==========================================================================
if res

   % Approximate value for the funtion
   %---------------------------------
   fseries = fourier_series(coeff,t,T);

   figure
   plot(t,y,'k',t,fseries,'r.')
   legend('Original function','Fourier series')
   grid on
   title('Fourier series expansion')
   xlabel('t')

   figure
   subplot(1,2,1)
   plot(freq,Aspec)
   grid on
   title('Amplitude spectrum: (a_m^2+b_m^2)^{1/2}')
   xlabel('Frequency (Hz)')

   subplot(1,2,2)
   plot(freq,Pspec*180/pi)
   grid on
   title('Phase spectrum: \phi_m (degrees)')
   xlabel('Frequency (Hz)')
```

end