

# Fourier Analysis Using A Spreadsheet

Robert A. Dory and Jeffrey H. Harris

For many tasks involving Fourier analysis, a spreadsheet is an excellent tool. Using three examples, this article shows that for data sets involving perhaps a hundred points and a few tens of coefficients, a spreadsheet method is practical and quick on personal computers such as the Macintosh SE or the IBM PC AT and compatibles. With enhanced hardware such as floating-point co-processor chips, the numbers may be extended by about an order of magnitude. This article presents three spreadsheet templates that should permit rather easy generalization to other orthogonal basis functions, provided those can be defined in terms of standard functions or recurrence formulas.

Fourier analysis is used here to illustrate provision of modularity to spreadsheet element definitions, so that the available replication processes allow simple extension of the ranges of the calculation; in this case, the number of harmonics and integration nodes.

Fourier analysis of a real function also illustrates some techniques in spreadsheet use. These include selective application of array/vector arithmetic and the important trick of formulating formula blocks that can be replicated (blocks copied with relative references adjusted appropriately and absolute references unchanged) to extend the range or resolution of the calculation.

For a moderate problem, programming and debugging of functional Fourier analysis templates like the Fourier templates discussed below might take a spreadsheet user less time than finding the user's manual and the object code for an existing Fourier analysis program.

The reader is assumed to have some experience with Excel or a similar spreadsheet program so the general operations given here can be carried out without detailed instructions. The examples use the R1C2 option for addressing the elements of the template; this is a matter of taste, but has the property that variable names can take the form *a0* and *b1*, which otherwise conflict with the slightly more compact normal addressing scheme A1, AB137, ...

Two templates for the Excel spreadsheet program illustrate the trade-off between extensive use of memory for possible reduction of calculation time and, conversely, the deliberate recalculation of data to reduce memory requirements.

Fast Fourier transform (FFT) techniques are not used here because the logic required is not straightforwardly well suited to the simpler tools available in the spreadsheet. By development of function macros for retrograde indexing, formula entry and logic, the FFT or the related

fast Hartley transform (FHT) should be practical.

The latter could give an interesting challenge to readers; an article on the FHT in the April, 1988 issue of Byte magazine, p. 93, by Mark O'Neill would be a fine starting place for development of this process, which uses only real (non-complex) arithmetic.

The procedure used here is straightforward integration with the formulas:

$$f(x) = \sum f_n \cos nx + \sum g_n \sin nx, \text{ with } n = 0, 1, 2, \dots$$

$$f_n = \int f(x) \cos nx \, dx / \int \cos^2 nx \, dx, \text{ from } 0 \text{ to } x_M = 2\pi,$$

$$g_n = \int f(x) \sin nx \, dx / \int \sin^2 nx \, dx, \text{ from } 0 \text{ to } x_M = 2\pi,$$

where  $x_M = 2\pi$  is the periodicity interval of  $f(x)$ . With this approach, the basis functions of the expansion could be readily changed for application of another basis, such as Legendre or Bessel functions, especially if the higher index elements can be calculated with sufficient accuracy using two-term recurrence formulas as in the Fourier case. Accuracy of this is discussed further under Example 2.

The integrations use the trapezoidal rule for non-uniform (arbitrary) grid spacing:

$$\int g(x) dx = 1/2 \sum (x_{j+1} - x_{j-1}) g(x_j)$$

where the nodes are  $x_j$ , for  $j = 0, j_M$  and for the special terms  $j = 0$  and  $j_M$ , the corresponding  $x_{-1}$  and  $x_{j_M+1}$  are to be taken as  $x_0$  and  $x_{j_M}$ . If a new set of basis functions called for a weight function, it could be grouped conveniently with the factor  $(x_{j+1} - x_{j-1})$ .

A test function is provided. It is given by  $y_1$ , with parameters  $a_1$ ,  $b_0$ ,  $b_2$  and "noise," where

$$y_1 = a_1 \sin x + b_0 + b_2 \cos 2x + \text{noise} \cdot (R - 1/2)$$

and  $R$  is the pseudo-random function with uniform distribution in  $[0, 1)$ .

## Fourier Template 1

Sine and cosine values are tabulated at the integration nodes and recurrence formulas are used to obtain the higher harmonic functions  $\sin nx_j$  and  $\cos nx_j$ . The accuracy should be adequate for reasonable values of  $nx_j$ , since Excel arithmetic is done to roughly 15 decimal places. For small values of  $n$  and  $j_M$ , for example, ten and 50, Excel recognizes that the basis data do not need to be recalculated when changes are made only to test data  $f(x_j)$ . The process is then reasonably speedy. However, when the number of nodes increases by a factor of three or four, tables of dependency become large and Excel may then not be able to avoid some recalculation.

Figure 1 shows one arrangement. We assume that a limited number of harmonics are needed, but that

Robert A. Dory and Jeffrey H. Harris are Research Physicists at Oak Ridge National Laboratory, Oak Ridge, Tenn.

# SPREADSHEETS

COMPUTERS IN PHYSICS NOV/DEC 1988

Fourier Example 1.											
21	points	p	2	dx0	0.314	b0	-0.25	a1	12.34	b2	0.123
j	x	dx	y	si	co	si 2	co 2	si 3	co 3	si 4	co 4
0	0.000	0.314	-0.127	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
1	0.314	0.628	3.663	0.309	0.951	0.588	0.809	0.809	0.588	0.951	0.309
2	0.628	0.628	7.041	0.588	0.809	0.951	0.309	0.951	-0.309	0.588	-0.809
3	0.942	0.628	9.695	0.809	0.588	0.951	-0.309	0.309	-0.951	-0.588	-0.809
19	5.969	0.628	-3.964	-0.309	0.951	-0.588	0.809	-0.809	0.588	-0.951	0.309
20	6.283	0.314	-0.127	0.000	1.000	0.000	1.000	0.000	1.000	0.000	1.000
(pπ) Non uniform intervals OK. Uses quadratic forms to do integrations											
Coefficients		fn	gn	0	1	2	3	4			
Cosine	fn	-0.250	0.000	0.000	0.123	0.000	0.000	0.000			
Sine	gn	0.000	12.340	0.000	0.000	0.000	0.000	0.000			
noise	0.0	Test function : y = b0 + a1 sin x + b2 cos 2x + noise * (rand()-.5)									
factor	1.00	norm	12.566	6.283	6.283	6.283	6.283	6.283	6.283	6.283	6.283

1. This is the Fourier analysis template 1. A two-column block from "Sin n Cos n" through "Coefficients Cosine and Sine" can be copied to blank columns on the right to add calculation for another n value.

acceptable accuracy may require a substantial number of integration nodes  $x_i$ . Integration nodes are arranged downward, with harmonic numbers across, because that fits the screen format best. If the data to be analyzed are generated by another program, Fortran for example, they can be stored one number per row, so that they can conveniently be pasted into Excel using the Macintosh Clipboard.

At the bottom of the column blocks labeled "si n" and "co n" are formulas for the integrations. The numbers labeled "norm" are the values of  $\int \sin^2 nx dx$  and  $\int \cos^2 nx dx$  multiplied by an overall normalizing number "factor." For the present sine and cosine basis and assumed integration interval  $[0, p\pi]$ , these values are all  $p\pi$  except those for  $n = 0$ , which are 0 and  $2p\pi$ . To save time, these could be changed to constant values 1, 0, and 2 and "factor" to  $p\pi$ ; they are as indicated to provide for use with another set of orthogonal functions.

The "norm" elements are entered in the form " $=\text{sum}(si^2 dx)$ " using the command key and a carriage return, to make them *array function formulas*, which can act, for example, to give the inner product of the elements of the vectors "sin" and "dx" [that is,  $\sum (sin)_i \cdot (sin)_i \cdot (dx)_i$ ]. Function formulas are set off by braces: [...].

The rows labeled "Cosine Coefficients" and "Sine Coefficients" are filled with similar array function formulas having the forms:

" $=\text{SUM}(y \cdot R[-23]C:R[-3]C \cdot dx)/R[+3]C$ " and  
 " $=\text{SUM}(y \cdot R[-24]C:R[-4]C \cdot dx)/R[+2]C$ "

where the divisor picks up the value of "norm" from the same column and the symbol "R[-25]C:R[-5]C" calls out the basis function vector from the same column.

If more integration nodes are needed, one can insert the appropriate number of blank rows somewhere after

2. A partial listing of the 'program' for template 1. As shown, columns 5 and 6, are too narrow to give complete formulas in all rows. The notation "RC:R[2]C[5]" denotes a 3x6 range of cells, from the cell containing the reference to the cell five columns to the right and two rows down.

Fourier Example 1. Formulas:1							
	2	3	4	5	6	7	
2	=COUNT(n)	points	p	2	dx0	=p*PI/(points-1)	
3	j	x	dx	y	si	co	
4	0		=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+b2*CO	=SIN(x)	=COS(x)	
5	1	=R[-1]C+dx0	=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+b2*CO	=SIN(x)	=COS(x)	
6	2	=R[-1]C+dx0	=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+b2*CO	=SIN(x)	=COS(x)	
25	(pπ)			Non-uniform intervals OK			
26	Coefficients			0	1	1	
27	Cosine	fn	=SUM(y * dx)/norm			=SUM(y * R[-23]	
28	Sine	gn	0			=SUM(y	
29	noise	0	Test function : y = b0				+a1 sin x + b2 cos 2x + noi
30			2pπ	pπ	pπ	pπ	
31	factor	1	norm	=SUM(dx)	=SUM(R	=SUM(R[-27]C:R[-	

3. A second partial listing of the 'program' for template 1. For a nicely labeled display, row 3, columns 8 and 9, which contain the values of n, have been assigned number formats "Si "0 and "Co "0 which preface the n value with the text in quotes.

Fourier Example 1. Formulas:2			
	7	8	9
2	=p*PI/(points-1)	b0	-0.25
3	eo	2	2
4	=COS(x)	=si*RC[-1]+co*RC[-2]	=co*RC[-2]-si*RC[-3]
5	=COS(x)	=si*RC[-1]+co*RC[-2]	=co*RC[-2]-si*RC[-3]
6	=COS(x)	=si*RC[-1]+co*RC[-2]	=co*RC[-2]-si*RC[-3]
24	=COS(x)	=si*RC[-1]+co*RC[-2]	=co*RC[-2]-si*RC[-3]
25	Uses quadratic forms to do integrations		
26	1	=R[-23]C	=R[-23]C
27	=SUM(y * R[-23]	=SUM(y * R[-23]C:R[-3]C * dx)/r	
28		=SUM(y * R[-24]C:R[-4]C * dx)/norm	
29	x + b2 cos 2x + noi	se * (rand()-.5)	
30	pπ	pπ	pπ
31	=SUM(R[-27]C:R[-1]	=SUM(R[-27]C:R[-7]C * 2 * dx)	=SUM(R[-27]C:R[-7]C * 2 * dx)

Fourier Example 2.											
21	points	p	2	(dx)	0.314	a1	1	b0	-0.25	b2	0.123
j	x	dx	y	Test function : $y = b0 + a1 \sin x + b2 \cos 2x + \text{noise} * (\text{rand}(0-5))$							
0	0.000	0.314	-0.127	noise 0.0							
1	0.314	0.628	0.159	Coefficients							
2	0.628	0.628	0.376	Cosine							
3	0.942	0.628	0.521	Sine							
4	1.257	0.628	0.602	norm 6.283							
5	1.571	0.628	0.627								
19	5.969	0.628	-0.460								
20	6.283	0.314	-0.127	Uses quadratic forms to do integrations			Non-equidistant pts OK				
(pπ)											

4. Fourier analysis template 2 is a space-saving program that uses half the memory of the previous example; it also has half the speed, but the added time may be unimportant.

the second row of nodes and before the last one. Then the relative duplication command ("Fill Down") can be used to populate the template from the second node down through the last. This should provide the required higher integration accuracy when new data  $y(x_j)$  are added.

If another harmonic value is required, the user can copy the two-column rectangle from "si n co n" down through the "norm" line into the two blank columns at the right of the presently filled zone. To make this possible, the sine and cosine formulas were written in mixed form " $= \text{si} \cdot \text{RC}[-1] + \text{co} \cdot \text{RC}[-2]$ ," which means multiply the element on this row of the vector named "si" by the element of this row and one column back, and add the corresponding element of vector "co" times the element two columns back. These formulas are not array function formulas because the intent is to evaluate a column vector row by row using data from the same row of other column vectors. The vectors "si" and "co" were defined to be columns C6 and C7, elements R4 through R24 in the present case, and should not be confused with functions SIN(argument) and COS(argument).

#### Fourier Template 2

To reduce memory usage (by about half) in the case of large data sets  $f(x_j)$ , the basic functions in this example are not determined once and stored. Instead, they are recalculated for each coefficient and case. Calculation time is about doubled, but still not large; with a

Macintosh II and use of a floating point co-processor chip the calculation time for about a thousand elements and a half dozen coefficients is roughly a minute.

The approach here is generally the same as in the first example, except that the coefficients are entered as the array function formulas

" $\{ = \text{SUM}(y \cdot \text{SIN}(n \cdot X)) / \text{norm} \}$ " and

" $\{ = \text{SUM}(y \cdot \text{COS}(n \cdot X)) / \text{norm} \}$ " where the

row array  $n$  is defined as labeled, and  $\text{norm}$  is set to  $p\pi$  for the Fourier transform to eliminate some repetition of the sine and cosine calculations. An easily misleading complication arose in this example. An initial assumption proved incorrect: that the proper element of a row array  $\text{norm}$ , as in Example 1, would be used by the array formula defining  $f_n$  and  $g_n$ . Instead, the first element of the  $\text{norm}$  array was used even for succeeding columns. This feature could be called a bug in Excel (version 1.04), but might be specified somewhere in fine print in the manual. The work-around chosen here was to reduce  $\text{norm}$  to a scalar and then insert the necessary factor 2 in the formula for  $f_0$ .

Integration nodes could be added to this template in the same manner as in the first example. Providing more coefficients is also about the same: insert new columns after the  $n = 1$  calculation and replicate the  $n, f_n, g_n$  and  $\text{norm}$  columns to the right, as far as the last one.

Fourier Example 2. Formulas:1						
2	2	3	4	5	6	7
2	=COUNT(j)	points	p	2	(dx)	=p*PI/(points-1)
3	j	x	dx	y	Test function : $y = b0 + a1 \sin x$	
4	0	0	=R[1]C[-1]-RC[-1]	=b0+a1*SIN(x)+		
5	1	=R[-1]C+R2C7	=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+	Coefficients	
6	2	=R[-1]C+R2C7	=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+	Cosine	
7	3	=R[-1]C+R2C7	=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+	Sine	
8	4	=R[-1]C+R2C7	=R[1]C[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+		
24	20	=R[-1]C+R2C7	=RC[-1]-R[-1]C[-1]	=b0+a1*SIN(x)+	Uses quadratic forms to do integrations	
25		(pπ)				

5. A partial listing of the 'program' for template 2. Columns 5 and 7 are not shown fully.

Fourier Example 2. Formulas:2						
2	8	9	10	11	12	
2	a1	1	b0	-0.25	b2	0.123
3	+b2					
4					noise	0
5	n	0	1	2	3	4
6	$f_n$	=SUM(y * dx) / norm / 2	=SUM(y * COS(R[-1]C * x) * dx) / norm	=SUM(y * (	=SUM(y * (	=SUM(y * (
7	$g_n$	0	=SUM(y * SIN(R[-2]C * x) * dx) / norm	=SUM(y * (	=SUM(y * (	=SUM(y * (
8	norm	=p*PI				
9						

6. A second partial listing of the 'program' for example 2. Columns 8, and 11 through 13 are not shown fully. See the note in the text concerning the change in use of  $\text{norm}$  from Example 1.

**Fourier Example 3.**

Test function:  $y = b0 + a1 \sin x + b2 \cos 2x + \text{noise} * (\text{rand}() - .5)$

21	points	p	2
a1	1	b0	-0.25
b2	0.123	noise	0.0
(dx)	0.3142	kmax	20

j	x	y	yfl
0	0.000	-0.127	-0.127
1	0.314	0.1585	-0.46
2	0.628	0.3758	-0.8

19	5.969	-0.46	0.1585
20	6.283	-0.127	-0.127

(pπ)                      y(flipped)

Nota bene:                      Equal intervals required.  
Ref:                      Numerical Methods that Work  
F. S. Acton; Harper, Row; 1970.

Acton's Compaction					
dj	n	0	1	2	3
0.05	phin	0	0.3142	0.6283	0.9425
dphi	ephin2	2.00	1.90	1.62	1.18
0.31	eKp2	0	0	0	0
k	eKp1	0	0	0	0
20	e20	-0.127	-0.127	-0.127	-0.127
19	e19	-0.714	-0.701	-0.665	-0.609
18	e18	-2.1	-2.006	-1.749	-1.368
1	e1	-114.4	32.361	1E-15	-2E-16
0	e0	-119.5	30.65	1.103	-0.127
csum		-5.127	-0.127	1.103	-0.127
dsum		0	10	6E-16	-2E-16
fn		-0.25	2E-15	0.123	0
err		9E-16		2E-17	
qn		0	1	6E-17	0
err				-7E-16	

7. Fourier template 3 is a compact analysis as described by F. S. Acton in the book *Numerical Methods That Work*. The method can accept an arbitrary number of equally space data points.

**Fourier Example 3 Formulas:1**

4	5	8	9	10	11
4			Acton's	Compaction	
5	p	2	dj	n	0
6	b0	-0.25	=1/kmax	phin	=R[-1]C*dphi
7	noise	0	dphi	ephin2	=2*COS(phin)
8	kmax	=MAX(j)	=2*pi*dj	eKp2	0
9	y	yfl	k	eKp1	0
10	=b0+a	=INDEX(y,points-j,1)	kmax-j	=RC[-1]	=ephin2*R[-1]C-R[-2]C+yfl
11	=b0+a	=INDEX(y,points-j,1)	kmax-j	=RC[-1]	=ephin2*R[-1]C-R[-2]C+yfl
29	=b0+a	=INDEX(y,points-j,1)	kmax-j	=RC[-1]	=ephin2*R[-1]C-R[-2]C+yfl
30	=b0+a	=INDEX(y,points-j,1)	kmax-j	=RC[-1]	=ephin2*R[-1]C-R[-2]C+yfl
31		y(flipped)	csum	=R[-1]C-R[-2]C*ephin2/2	=R[-1]C-R[-2]C*ephin2/2
32		Equal intervals required.	dsum	=R[-3]C*SIN(phin)	=R[-3]C*SIN(phin)
33			fn	=dj*(R[-2]C-R30C5)	=2*dj*(R[-2]C-R30C5)
34			err	=(b0-R[-1]C)/b0	
35			qn	=2*dj*R[-3]C	=2*dj*R[-3]C
36			err		=(a1-R[-1]C)/a1

8. A partial listing of the 'program' for template 3. Notice that a single sine and cosine pair is evaluated for each harmonic coefficient. In the second column shown, the data vector y is 'flipped' to simplify the downward recurrence for the harmonic coefficients.

**Discussion**

In spreadsheet programming, higher efficiency is possible with more careful planning and improved mathematical technique. The Fourier analysis process has been the focus of a great deal of optimization in the last couple of decades. The FFT process earlier permits very rapid analysis, at the expense of substantially more involved logic, restriction to uniform data spacing and to a number of data points that can be decomposed into a product of powers of small prime numbers [2<sup>m</sup>·3<sup>n</sup>...].

A compact and elegant process arising from a suggestion in F. S. Acton's book *Numerical Methods That Work*, Harper and Row, 1970, provides an intermediate level of speed and complexity. It applies to uniformly spaced data with arbitrary number of points, and gives an attractively simple template requiring only one sine and cosine per harmonic. The scaling of calculation time is as  $j_M$  times  $nmax$  rather than  $j_M \ln j_M$  for the FFT process; it can be competitive then when only a few tens of data points are needed and the required highest harmonic,  $nmax$ , is not large. The speed is about twice that of Example 1, and four times that of Example 2.; storage requirements are slightly (10%) under those for Example 1, twice that of Example 2.

The process is based on recurrence formulas for cosines and sines:

$$\cos n\phi = (2 \cos\phi) \cdot \cos(n+1)\phi - \cos(n+2)\phi$$

and

$$\sin n\phi = (2 \cos\phi) \cdot \sin(n+1)\phi - \sin(n+2)\phi$$

with the observation that:

$$\sum_{k=0}^K f(k\phi_n) \cos k\phi_n = [c_0 - c, \cos\phi_n]$$

and

$$\sum_{k=0}^K f(k\phi_n) \sin k\phi_n = c, \sin\phi_n,$$

when  $c_k = (2 \cos\phi_n) \cdot c_{k+1} - c_{k+2} + f(k\phi_n)$ , for  $k = K, \dots, 1, 0$ , with  $c_{K+1} = c_{K+2} = 0$ .

**FOR FURTHER READING**

*Numerical Methods That Work*, mentioned above, gives a fine discussion of the merits of using recurrence formulas to extend tables of standard functions. A more recent book is *Numerical Recipes* by Press, Flannery, Teolkowsky and Vetterling, Cambridge Press, 1986.

Readers' suggestions for useful or especially effective texts on numerical/mathematical methods would be appreciated. They can be sent in care of Computers In Physics, American Institute of Physics, 335 East 45th Street, New York, N.Y., 10017.