Abstract: - Periodic functions appear quite often in engineering and hence mathematicians have paid attention to its representation by trigonometric series. In this note we present some simple functions that may be easily implemented in computer algebra systems and facilitate the understanding of how the corresponding trigonometric series approaches to the periodic function.

Key-Words: - Periodic function, computer algebra system.

1 Background

Computer Algebra Systems (CAS), e.g. DERIVE™, MATHEMATICA® and MATLAB® [4,11,17] have done for Algebra and Calculus what calculators did for arithmetic in saving time and avoiding mistakes in calculations.

CAS have smoothened the task of simplifying expressions, solving equations, taking derivatives, finding antiderivatives, and much more (cf. [15]).

In general, CAS may help in saving time and approaching Mathematics to Engineering, making its lecturing teaching more appealing without affecting the mathematical level and scope, [6,7,8].

In fact while for the engineering students, the constraints imposed by the engineering drawing may create the perception that the more they know, the less creative they become, our experience tells us that the exposure of the students to graphics created by CAS enhances their capability of graph representation and facilitates their understanding of mathematical problems and theory.

In this way plotting the partial and final solution of problems make the interpretation of mathematical expressions easier, as well as facilitating to understand the relationships between the problems solved and the reality that these problems refer to.

Traditionally, accurate and detailed drawing were assumed as the most important engineering skills. However engineering world has changed drastically once that time-consuming hand drawings has given way to computer drafting.

One of the typical examples that shows how graphics help to understand approximation techniques is the representation of periodic functions which are smooth enough by means of Fourier series [1,3,12,13,14,16]. CAS are a great ally for both goals: obtaining the Fourier series and representing them.

Let us recall that the Fourier series of a $2\pi$-periodic function $f$ is the trigonometric function given by

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left( a_n \cos(nx) + b_n \sin(nx) \right)$$

where the coefficients $a_0$, $a_n$, $b_n$ are given by means of the Euler’s formulae which in turn are obtained with the aid of the orthogonality relationships of the sine and cosine functions.

And according to the Dirichlet conditions each piecewise regular function $f$ which has a finite number of finite discontinuities and a finite number of extrema can be expanded into a Fourier series which converges to $f$ at the continuous points and the mean of the positive and negative limits at the discontinuity points.

On the other hand it is well known that a piecewise regular function $f$ defined on $[a,b]$, $a>0$, can be represented by a (non-unique) sine or cosine series. In practice, the selected series type will be a consequence of the contour conditions of the problem.

For example, if the following heat equation (cf. [1,5]) with null contour conditions is to be solved:
where \(0<x<L\), \(t>0\) and \(k>0\) is a constant which has a certain physical meaning, a solution comes given by the expression

\[
    u(x,t) = \sum_{n=1}^{\infty} B_n \sin\left(\frac{n\pi x}{L}\right) e^{-\left(\frac{n\pi}{L}\right)^2 kt}
\]

where the \(B_n\) are the coefficients of the sine series development of \(f(x) = u(x,0)\) in \([0,L]\).

Because of this interest in representing periodic functions by means of sine and cosine series our aim in this paper is to present an easily implemented algorithm that will ease the task of generating adequate Fourier series.

### 2 Generating periodic functions

Let us recall that a real function \(f(x)\) is said to be periodic if there is some real number \(p>0\) called period such that

\[
    f(x) = f(x + np)
\]

for \(n = 1, 2, \ldots\). Hence the sine function is an example of periodic function with least period (called principal period) \(2\pi\) and each constant function is periodic but does not have a principal period.

Periodic signals play an important role in many scientific and technological applications (cf. [2,5,10,13]). However standard programs do not incorporate, up to now, an implemented function to generate periodic functions. And thus it is not unusual to find periodic functions \(f\) represented in only one interval of length the period of \(f\) whereas its Fourier series is fully represented as shown for instance in Figure 1.

![Fig. 1](image1)

For this reason in [9] we aimed to find an easily implemented method to generate and plot periodic functions. The idea to generate a \((b-a)\)-periodic function defined over the whole real axis starting from a function \(f(x)\) defined at an interval \([a,b]\) was just to substitute the variable \(x\) by the expression

\[
    x - \left(\frac{b-a}{b-a}\right) x - \left(\frac{b-a}{b-a}\right)
\]

In this way if the DERIVE program is to be used as CAS the \(\text{MOD}(x,k)\) command, which stands for \(x\) modulo \(k\), is pretty handy and just have to substitute the variable \(x\) by

\[
    a + \text{MOD}(x-a,b-a).
\]

For instance, in order to extend the function \(e^x\), \(x \in [0.5,1.5]\) to an 1-periodic function we may use the command

\[
    \text{SUBST}(\text{EXP}(x),x,0.5 + \text{MOD}(x-0.5,1.5-0.5)).
\]

Simplifying the above expression and plotting the function generated in this way, Figure 2 is obtained.

![Fig. 2](image2)

This function can also be implemented within MATHEMATICA. For instance, by means of

\[
    \text{Plot}[\text{Exp[Mod[x,1.5-0.5,0.5]],\{x,3,3\},\text{PlotRange}\to\{-4,4\}];
\]

we obtain the graph shown in Fig. 3. It may be seen that we find an unexpected linking of the function at its discontinuity points due to the way in which MATHEMATICA generates graphs.

\[\text{(*)}\]

[\(\text{I}(x)\)] represents the function integer part, i.e. \(\text{I}(t) = n\) where \(n\) is the integer number such that \(n \leq t < n+1\).

\[\text{(*)}\]

In DERIVE versions previous to the 5.0 the command LIM should be used instead of SUBST with the same structure and arguments.
MATLAB also enables to generate graphs of periodic functions in a similar way. In the above particular case (see Fig. 4) this may be done by means of

\[ L = \text{exp}(0.5 + \text{mod}(x-0.5,1.5-0.5)) \]
\[ \text{ezplot}(L,[-3,3]) \]

Hence we find that MATLAB produces the same linking effect as MATHEMATICA does at the discontinuity points.

### 3 Even and odd extensions

In order to get the sine and cosine series representations of a piecewise regular function \( f \) defined on \( ]a, b[ \subset \mathbb{R}, a \geq 0 \), firstly we must extend \( f \) to an interval \( ]-b, b[ \) so that the extended function is even or odd. This may be done with the aid of the following two functions, \( f_E \) for the even extension and \( f_O \) for the odd extension, defined by means of:

\[
\begin{align*}
  f_E(x) &= f(-x) \chi_{]-b,-a[} + f(x) \chi_{[a,b[} \\
  f_O(x) &= -f(-x) \chi_{]-b,-a[} + f(x) \chi_{[a,b[}
\end{align*}
\]

where \( \chi_{[a,b[} \) is the characteristic function of the interval \( ]a, b[ \), that is to say

\[
\chi_{[a,b[}(x) = \begin{cases} 
1 & \text{if } x \in ]a, b[ \\
0 & \text{if } x \notin ]a, b[ 
\end{cases}
\]

This extending function is easily implemented in CAS. In case we are dealing with the DERIVE program we will have in mind that the function characteristic of the interval \( ]a, b[ \) is given by means of its function \( \text{CHI}(a,x,b) \). Hence the above two extending functions may be implemented by defining:

\[
\begin{align*}
\text{EVEN\_EXT}(f(x),x,a,b):= & f(x) \cdot \text{CHI}(a,x,b) + \text{SUBST}(f(x),x,-x) \text{CHI}(-b,x,-a), \\
\text{ODD\_EXT}(f(x),x,a,b):= & -f(x) \cdot \text{CHI}(a,x,b) - \text{SUBST}(f(x),x,-x) \text{CHI}(-b,x,-a)
\end{align*}
\]

Therefore if we are to get an even extension and an odd extension of the function \( e^x, x \in ]0.5,1.5[ \) we may obtain them respectively by means of

\[
\begin{align*}
(e^x)_E &= e^x \chi_{]0.5,1.5[} + e^{-x} \chi_{]-1.5,-0.5[}, \\
(e^x)_O &= e^x \chi_{]0.5,1.5[} - e^{-x} \chi_{]-1.5,-0.5[}.
\end{align*}
\]

Thus if we take advantage of the two functions that we have built up in this section we just have to introduce the following two functions:

\[
\begin{align*}
\text{EVEN\_EXT}(&\text{EXP}(x),x,0.5,1.5), \\
\text{ODD\_EXT}(&\text{EXP}(x),x,0.5,1.5).
\end{align*}
\]

Plotting them we see in Fig. 5 their graphs.
Here (1) represents the original function, (1) and (2) form the even extension, and (1) together with (3) is the odd extension.

The analogous to the CHI function of DERIVE are not implemented within MATHEMATICA nor in MATLAB.

However the above extending implementations may be analogously defined within these two programs. In order to get this we should implement a function that will play the role of CHI in DERIVE. Hence if we are to work with MATHEMATICA we may define for instance

\[
\text{chi}(x, a, b) := \text{If}\[\text{And}[x > a, x < b], 1, 0]\]

Clearly this function may also be defined by a difference of two step functions:

\[
\text{chi}(x, a, b) = \text{UnitStep}[x-a] - \text{UnitStep}[x-b]
\]

Within MATLAB we do not find the step function implemented. Thus this procedure requires that we should define it and proceed as follows:

\[
\begin{align*}
\text{syms} \ x \\
H &= 1/2*(\text{abs}(x)/x+1) \\
H &= 1/2*\text{abs}(x)/x+1/2 \\
\text{CHI} &= \text{subs}(H, x, x-a) - \text{subs}(H, x, x-a) \\
\text{CHI} &= 1/2*\text{abs}(x-a)/(x-a) - 1/2*\text{abs}(x-a)/(x-a)
\end{align*}
\]

4 Obtaining sine and cosine series

In Section 2 we have recalled a method to generate a \((b-a)\)-periodic extension of a function \(f : a, b [\to \mathbb{R}\). And in Section 3 we have seen how to generate an even or odd function which coincides with a given function \(f\) defined on \(a, b\) whenever 0 does not belong to \(a, b\).

If we combine the two techniques and after getting and even or odd extension of \(f\) defined over \(-b, b\) we may form a \(2b\)-periodic even extension \(f^E\) by means of

\[
f^E(x) = f^E(\left(x - (2b)\right)\left(\frac{x+b}{2b}\right)).
\]

This function happens to be even and \(2b\)-periodic. Analogously we may get the \(2b\)-periodic odd extension by means of the function

\[
f^O(x) = f^O(x - (2b)\left(\frac{x+b}{2b}\right)).
\]

If we find the Fourier series of each one of the two periodic functions generated in this way, we will obtain a Fourier cosine series and a Fourier sine series, respectively, corresponding to the \(2b\)-periodic even and odd extension of \(f, f^E\) and \(f^O\).

It is worth noting that, under adequate smooth conditions, by Dirichlet’s theorem any of these two series do represent \(f\) in \([a, b]\) at its continuity points.

All this process can be visualized using any of the CAS mentioned in this paper. In this way if we are to use the DERIVE program and consider the partial sums of Fourier series of order 10 for the function \(e^x\), \(x \in ]0.5,1.5[\), we proceed as follows:

\[
\begin{align*}
#1: \ & \text{EVEN\_EXT}(\text{EXP}(x), x, 0.5, 1.5) \\
#2: \ & \text{ODD\_EXT}(\text{EXP}(x), x, 0.5, 1.5) \\
#3: \ & \text{SUBST}(#1, x, -1.5+\text{MOD}(x+1.5, 1.5+1.5)) \quad (*) \\
#4: \ & \text{SUBST}(#2, x, -1.5+\text{MOD}(x+1.5, 1.5+1.5)) \\
#5: \ & \text{FOURIER}(#1, x, -1.5, 1.5, 10) \\
#6: \ & \text{FOURIER}(#2, x, -1.5, 1.5, 10)
\end{align*}
\]

The so obtained partial sums of the Fourier cosine and sine series are approximations of two \(2b\)-periodic extensions of \(f\) which logically coincide with \(f\) when restricted to \([a, b]\) but not in the larger \([-b, b]\) where \(f\) in fact does not either exist.

In Fig. 6 we may observe how #5 is an approximate representation of the even extension of the original function \(e^x\) defined in \(]0.5,1.5[\).

\[\text{Fig. 6}\]

\[\text{(*)} \quad \text{The lines #3 and #4 may be omitted for the graph representation but both of them appear in the theoretical construction.}\]
And in Figure 6 we find how #6 is an approximate representation of the odd extension.

5 Reducing the Gibbs phenomenon

The graphs of the partial sums shown in Figures 6 and 7 of the 2b-periodic extensions of a function originally defined in ]-b,b[ by means of

\[
    f_E(x) = f(-x)\chi_{]-b,-a]} + f(x)\chi_{]a,b[}
\]

\[
    f_O(x) = -f(-x)\chi_{]-b,-a]} + f(x)\chi_{]a,b[}
\]

reveal a deviation presented with respect to \( f \) close to \( x = a \) and \( x = b \) where we find an “overshooting”. These overshootings, known as the Gibbs’ phenomenon (cf. \[1,14\]), are due to that the functions \( f_E \) and \( f_O \) are not continuous at those points.

If \( a > 0 \) we may avoid this phenomenon in \( x = a \) by using another extending functions in \(-b,b[\) as for example:

\[
    f_E^E(x) = f(-x)\chi_{]-b,-a]} + f(a)\chi_{]a,b[} + f(x)\chi_{]a,b[}
\]

\[
    f_O^O(x) = \begin{align*}
    -f(-x)\chi_{]-b,-a]} - f(a)\chi_{]a,0]} + f(a)\chi_{]a,b[} + f(x)\chi_{]a,b[}
\end{align*}
\]

These two functions are easily implemented in any CAS. In the case of DERIVE this may be done by means of the following two commands:

\[
\begin{align*}
\text{EVEN\_EXT\_IMP}(u,x,a,b) &:= \\
&= u\cdot\text{CHI}(a,x,b)+\text{SUBST}(u,x,-x)\text{CHI}(-b,-x-a)+ \\
&+\text{SUBST}(u,x,a)\text{CHI}(-a,x,a),
\end{align*}
\]

\[
\begin{align*}
\text{ODD\_EXT\_IMP}(u,x,a,b) &:= \\
&= u\cdot\text{CHI}(a,x,b)-\text{SUBST}(u,x,-x)\text{CHI}(-b,-x-a)+ \\
&+\text{SUBST}(u,x,a)\text{CHI}(0,x,a)-\text{CHI}(-a,x,0)).
\end{align*}
\]

In both cases the Fourier series are different from the aforementioned ones, despite the fact that both of them go on being cosine and sine series respectively. However we may see in Figure 8 how the new Fourier series approach in a better way to \( f \) near \( x=a \).

If \( a = 0 \) and \( f(a) \neq 0 \), the Gibbs’ phenomenon will appear in the odd extension in the proximity of \( x = 0 \) and this cannot be avoided.

A slight modification of the above enables to avoid the Gibbs’ phenomenon in \( x = b \). An option is to generate a 3b-periodic extension of the function originally defined in \( ]a,b[ \) for example:

\[
\begin{align*}
    f_E^E(x) & = +f(-x)\chi_{]-b,-a]} + f(a)\chi_{]a,b[} + f(x)\chi_{]a,b[} + \\
    & + f(b)\chi_{]a,b[} + f(b)\chi_{]b,\frac{3b}{2}]} \\
    f_O^O(x) & = \\
    -f(-x)\chi_{]-b,-a]} - f(a)\chi_{]a,0]} + f(a)\chi_{]a,b[} + f(x)\chi_{]a,b[} - \\
    - f(b)\chi_{]a,b[} + f(b)\chi_{]b,\frac{3b}{2}]}.
\end{align*}
\]
6 Conclusion
The methods exposed in Section 4 may be easily modified to generate half-range sine and cosine extensions of a function originally defined on $]-L, L[$ by working with $\text{CHI}(0, x, L)f(x)$ instead of $f$. And then we should consider a $2L$-periodic even or odd extension of this function considered as defined in $]0, L[$ since this is just a particular case of the above exposition with $a = 0$.

All CAS systems here presented have proven to be useful in dealing with Fourier sine and cosine series representation.

References: