

# ME 201/MTH 281/ME400/CHE400

## Fourier Series for a Square Wave

### *Mathematica 7*

#### ■ 1. Introduction

This notebook looks at the numerical convergence of the Fourier series for a square wave. Later we will also take a graphical look at the convergence. This notebook is written so that it easily can be modified to study the convergence of the Fourier series of a different function. The specific function used is defined in section 2.

#### ■ 2. Definition of Function and Fourier Coefficients

In the definitions below, we first define the function over a basic interval  $[-L,L]$ , specify the value of  $L$ , and then extend the function periodically. To illustrate the procedures as we go, we will use the square wave. The name used for the basic function defined over  $[-L,L]$  is `fbas`. The periodically extended function is called `f`.

```
fbas[x_] := -1 /; -L <= x < 0 (* Square Wave *)
```

```
fbas[x_] := 1 /; 0 <= x < L
```

```
L = 1;
```

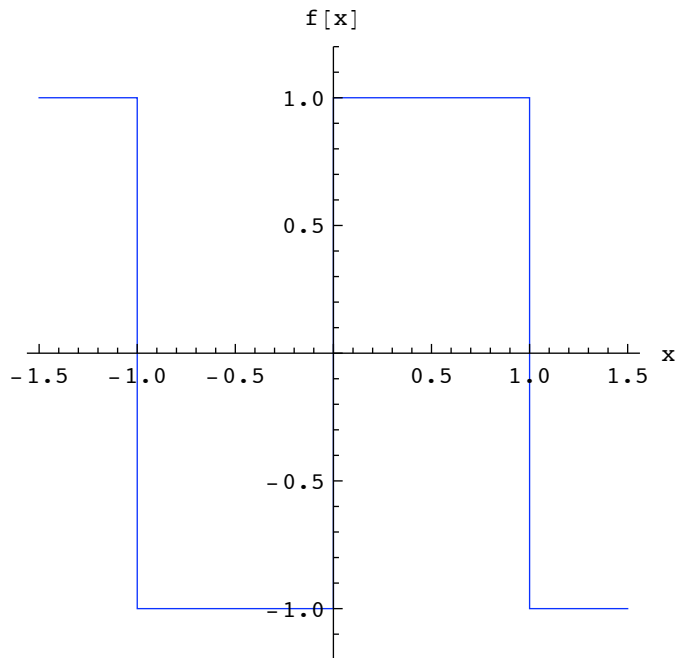
```
f[x_] := fbas[Mod[x, 2 * L, -L]]
```

The function `Mod[x, 2*L, -L]` (1) adds  $L$  to  $x$ , (2) calculates the remainder on dividing (1) by  $2*L$ , (3) subtracts  $L$  from (2). This has the effect of shifting the original  $x$  by  $\pm 2L$  until it falls into the range  $[-L,L]$ . This modified  $x$  is then used as the argument of the original function `f[x]`.

We plot the periodically extended function over the range  $[-1.5L, 1.5L]$ .

```
SetOptions[Plot, ImageSize -> 250];
```

```
Plot[f[x], {x, -1.5 * L, 1.5 * L}, PlotStyle -> RGBColor[0, 0, 1], PlotPoints -> 500,
PlotRange -> {-1.2, 1.2}, AxesLabel -> {"x", "f[x]"}, AspectRatio -> 1.0]
```



We give here the Fourier coefficients for the function  $f$  (in this case the square wave). We assume these have been calculated elsewhere, and the values are supplied here in the definitions of the coefficients. We denote the cosine coefficients by  $a[n]$  and the sine coefficients by  $b[n]$ . For convenience in later calculations, we define  $b[0] = 0$ .

```
a[n_] = 0.0;
```

```
b[0] = 0;
```

```
b[n_] := If[EvenQ[n], 0, (4 / (n * π))]
```

EvenQ is a logical function which returns True if its argument is even, False otherwise.

### ■ 3. Terms and Partial Sums of Fourier Series

Now we define the  $n$ th term of the Fourier series, called fourterm.

```
fourterm[x_, n_] :=
  N[a[n] * Cos[(n * π * x) / L] + b[n] * Sin[(n * π * x) / L]]
```

The  $n$ th partial sum of the series is foursum, given by

```
foursum[x_, n_] := Sum[fourterm[x, k], {k, 0, n}]
```

#### ■ 4. Numerical Values of the Partial Sums

We construct a table of values showing how the sum of the series depends on both  $x$  and on the number of terms kept in the partial sum. We use 10 values of  $x$ , namely 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, and 0.5. We consider the  $k$ th partial sum with the indexing being such that we add up the first  $k$  non-zero terms to get the  $k$ th partial sum. For example, for  $k = 20$ , we will add up the harmonics corresponding to  $n = 1, 3, 5, \dots, 39$ . Our table will be for  $k = 10, 20, 40, 80$ , and 160.

```
TableForm[Table[foursum[i * 0.05, 10 * 2^j - 1], {i, 1, 10}, {j, 1, 5}], TableHeadings ->
  {"x=0.05", "x=0.10", "x=0.15", "x=0.20", "x=0.25", "x=0.30", "x=0.35", "x=0.40",
   "x=0.45", "x=0.50"}, {"k = 10", "k = 20", "k = 40", "k = 80", "k = 160"}]
```

	k = 10	k = 20	k = 40	k = 80	k = 160
x=0.05	1.17981	0.902406	0.94973	0.974644	0.987293
x=0.10	0.90114	0.949097	0.974327	0.987134	0.993563
x=0.15	1.06875	0.965128	0.982495	0.991239	0.995618
x=0.20	0.946452	0.973003	0.986472	0.993232	0.996616
x=0.25	1.04469	0.977534	0.988751	0.994374	0.997187
x=0.30	0.960851	0.980352	0.990167	0.995082	0.997541
x=0.35	1.03559	0.982154	0.991071	0.995535	0.997767
x=0.40	0.966631	0.983278	0.991634	0.995817	0.997908
x=0.45	1.03214	0.983897	0.991944	0.995972	0.997986
x=0.50	0.968248	0.984094	0.992043	0.996021	0.998011

Notice that a large number of terms are required for accuracy. This is a consequence of the slow decay of the Fourier coefficients with increasing  $n$  -- they drop off only like  $1/n$ . Notice also that more terms are required for accuracy for points near a discontinuity, such as  $x = 0.05$ . For example, there is still a 5% error at  $x = 0.05$  even when  $k = 40$ , whereas the error at  $x = 0.5$  (maximum distance from a discontinuity) is slightly less than 1% for  $k = 40$ . Finally notice the surprisingly large overshoot at  $x = 0.05$  when  $k = 10$  -- nearly 18% above the actual value of the function. This is not a numerical error. The overshoot is real. Surprisingly, it does not go away as  $k$  increases, but simply moves ever closer to the discontinuity. This is called the Gibbs phenomenon, after the American mathematical physicist Willard Gibbs who first explained the overshoot. Very shortly, we will see a graphical view of all of this, which will provide more insight than the bare numbers above.