ME201/MTH281/ME400/CHE400
Convergence of Fourier Series

Version 1.7

1. Introduction

This notebook provides a general framework for the visualization of partial sums of Fourier series. To use it for a particular function, you must define the function in Section 2 below. This is the only place in the notebook where specific information about the function is given. Section 3 contains the definitions of the terms and partial sums of the series, and Section 4 defines a function which produces a graph of the nth partial sum for any n. Section 5 defines a computationally efficient routine for producing a sequence of all partial sums up to a specified value n. A number of examples are given in Section 6. In Section 7, we take a different approach to the visualization by using the powerful *Mathematica* command Manipulate, which is new to versions 6 and 7. As you will see there, one line of code produces the graph sequence along with a convenient set of controls for visualizing the sequence.

2. Definition of Function and Fourier Coefficients

In[1]:=
```
Clear[fbas, f, a, b];
```

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. We will do this first for the square wave. In the code below, the name of the basic function defined for [-L,L] is fbas, and the periodically extended function is called f. The example defined below is the square wave. There are many ways to define it. We use the built-in function sign[x], which is -1 for negative x, 0 for x = 0, and 1 for positive x. We then extend it periodically.

In[2]:=
```
fbas[x_] := Sign[x]
```

We next specify L, the half-length of the period.

In[3]:=
```
L = 1;
```

Now we periodically extend the function.

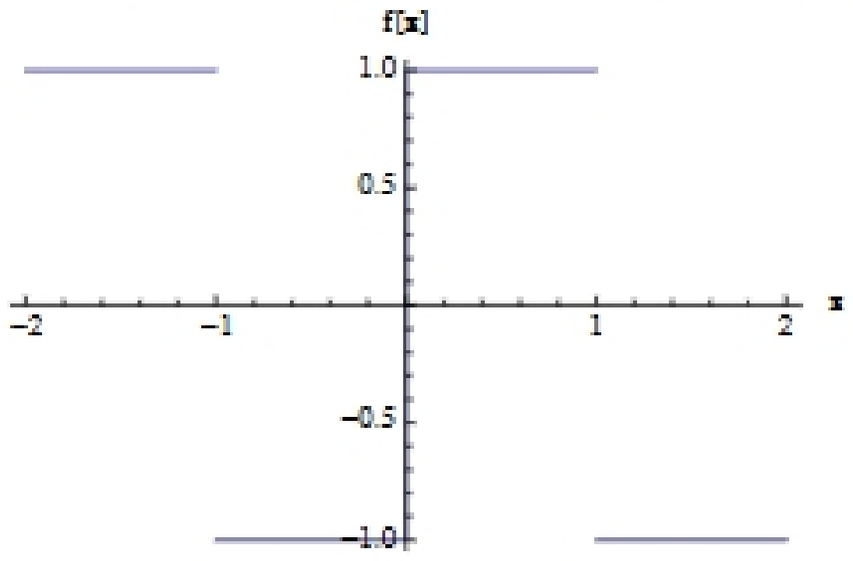
In[4]:=
```
f[x_] := fbas[Mod[x, 2 * L, -L]]
```

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]. To check that this gives the function we want, we plot f over $[-2L, 2L]$. We make the line for the plotted function slightly thicker than the *Mathematica* default by using the Thickness function with the Option PlotStyle.

In[5]:=
```
SetOptions[Plot, ImageSize -> 250];
```

In[6]:=
```
Plot[f[x], {x, -2 L, 2 L}, AxesLabel → {"x", "f[x]"}, PlotStyle -> Thickness[0.004]]
```

Out[6]=



For purposes of plotting yet to come, we specify the minimum and maximum of the function f, denoted by fmin and fmax.

In[7]:=
```
fmin = -1.0;
```

In[8]:=
```
fmax = 1.0;
```

We calculate here the Fourier coefficients for the function f (in this case the square wave). For most of the simple functions we work with (including this one), the coefficient integrals could be evaluated analytically. However, we write our code in terms of numerical integration, so that it will work for (almost) any integrable function. It is important NOT to calculate the integral for each coefficient each time the function is called. If that is done, calculations involving the series, such as plotting, would be unacceptably slow. We evaluate the coefficients once and for all and store the values in the functions a[n] and b[n]. Although we don't know a priori how many coefficients will be needed in a particular problem, we calculate the first ncoeffmax coefficients, where we set the default value of ncoeffmax below to 101. If we need more coefficients than that, we might want to think about finding another way to do the problem! 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. The function calcoeff, defined below, calculates and stores all of the Fourier coefficients up to order ncoeffmax.

In[9]:=
```
ncoeffmax = 101;
```

In[10]:=
```
calcoeff :=
  Module[{}, a[0] = (1.0 / (2 L)) NIntegrate[fbas[x] , {x, -L, L}, AccuracyGoal → 8]; b[0] = 0.0;
   Do[(a[n] = (1.0 / L) NIntegrate[fbas[x] Cos[(n π x) / L], {x, -L, L}, AccuracyGoal → 8];
     b[n] = (1.0 / L) NIntegrate[fbas[x] Sin[(n π x) / L], {x, -L, L}, AccuracyGoal → 8]), {n,
     1, ncoeffmax}]];
```

If you have analytical expressions for your Fourier coefficients and you prefer to use them, define the function a[n] as the nth Fourier cosine coefficient and b[n] as the nth Fourier sine coefficient. If you do this, then do not execute the command calcoeff.

Now we execute calcoeff to calculate the coefficients.

In[11]:=
```
calcoeff;
```

In order to check on the accuracy of the numerical integration in this case, we also use the analytical expression for the coefficients and then compare the two. As we showed in class, for this function the cosine coefficients are all zero, and the sine coefficients are zero for even n, and $4/(n\pi)$ for odd n. We store the coefficient values in the functions analyta[n] and analytb[n].

In[12]:=
```
Do[(analyta[n] = 0; analytb[n] = If[EvenQ[n], 0, 4.0 / (n π)]), {n, 0, 101}];
```

We now compare the results. We first check to see that all of the a[n]'s are zero as they should be for this odd function.

In[13]:=
```
Table[{n, a[n]}, {n, 0, 101}]
```

Out[13]=
```
{{0, 0.}, {1, 0.}, {2, 0.}, {3, 0.}, {4, 0.}, {5, 0.}, {6, 0.}, {7, 0.}, {8, 0.},
 {9, 0.}, {10, 0.}, {11, 0.}, {12, 0.}, {13, 0.}, {14, 0.}, {15, 0.}, {16, 0.}, {17, 0.},
 {18, 0.}, {19, 0.}, {20, 0.}, {21, 0.}, {22, 0.}, {23, 0.}, {24, 0.}, {25, 0.}, {26, 0.},
 {27, 0.}, {28, 0.}, {29, 0.}, {30, 0.}, {31, 0.}, {32, 0.}, {33, 0.}, {34, 0.}, {35, 0.},
 {36, 0.}, {37, 0.}, {38, 0.}, {39, 0.}, {40, 0.}, {41, 0.}, {42, 0.}, {43, 0.}, {44, 0.},
 {45, 0.}, {46, 0.}, {47, 0.}, {48, 0.}, {49, 0.}, {50, 0.}, {51, 0.}, {52, 0.}, {53, 0.},
 {54, 0.}, {55, 0.}, {56, 0.}, {57, 0.}, {58, 0.}, {59, 0.}, {60, 0.}, {61, 0.},
 {62, 0.}, {63, 0.}, {64, 0.}, {65, 0.}, {66, 0.}, {67, 0.}, {68, 0.}, {69, 0.},
 {70, 0.}, {71, 0.}, {72, 0.}, {73, 0.}, {74, 0.}, {75, 0.}, {76, 0.}, {77, 0.},
 {78, 0.}, {79, 0.}, {80, 0.}, {81, 0.}, {82, 0.}, {83, 0.}, {84, 0.}, {85, 0.},
 {86, 0.}, {87, 0.}, {88, 0.}, {89, 0.}, {90, 0.}, {91, 0.}, {92, 0.}, {93, 0.},
 {94, 0.}, {95, 0.}, {96, 0.}, {97, 0.}, {98, 0.}, {99, 0.}, {100, 0.}, {101, 0.}}
```

Now we check to see that the b[n]'s are all zero for even n.

In[14]:=
```
Table[{n, b[n]}, {n, 0, 100, 2}]
```

Out[14]=
```
{{0, 0.}, {2, -8.43745 × 10^-17}, {4, 1.59121 × 10^-17}, {6, -3.44631 × 10^-17}, {8, -2.49405 × 10^-16},
 {10, 8.49173 × 10^-16}, {12, -1.98425 × 10^-15}, {14, 1.53841 × 10^-15}, {16, 0.},
 {18, 4.26519 × 10^-16}, {20, 5.7863 × 10^-16}, {22, 3.82782 × 10^-16}, {24, 0.}, {26, 0.}, {28, 0.},
 {30, 0.}, {32, 0.}, {34, -1.85433 × 10^-16}, {36, -1.5619 × 10^-16}, {38, -1.2537 × 10^-16},
 {40, -9.65625 × 10^-17}, {42, -7.13447 × 10^-17}, {44, -5.01672 × 10^-17}, {46, 0.}, {48, 0.},
 {50, 0.}, {52, 0.}, {54, 0.}, {56, 0.}, {58, 0.}, {60, 0.}, {62, 0.}, {64, 0.}, {66, 0.},
 {68, 0.}, {70, 0.}, {72, 0.}, {74, 0.}, {76, 0.}, {78, 0.}, {80, 0.}, {82, 0.}, {84, 0.},
 {86, 0.}, {88, 0.}, {90, 0.}, {92, 0.}, {94, 0.}, {96, 0.}, {98, 0.}, {100, 0.}}
```

While some values are not exactly zero, those values are negligibly small. Finally we check the b values for odd n. We construct a table with values of n, of the numerical b[n], and the difference of the numerical and analytical values of b[n].