# Lecture 6:

# The Convergence of Fourier Series

Jorge Aarao

PCMI, Summer 2003

This worksheet contains a procedure (found in the Maple Manual) that computes and displays the n-th Fourier approximant for
a function f defined on a given interval. Also to be found are pictures of the Dirichlet kernel, and a visual demonstration of
Gibbs' phenomenon. Finally, at the end we have a procedure that computes the L^2 norm of the difference f - S(f,N).

Whenever appropriate the code will be given in the form of a procedure.

## Computing Fourier Coefficients

The procedure An takes as its input a function (func), its range (xrange), and an integer n, and returns the coefficient An.
Likewise, the procedure Bn returns the coefficient Bn.

```
> An:=proc(func, xrange::name=range, n)
>    local l; global A;
>    l:= rhs( rhs(xrange) ) - lhs( rhs(xrange) );
>    A[n]:= (2/l)*int(func*cos(n*x),xrange);
> end proc:
> An(x,x=-Pi..Pi, 3);
```

$$0$$

```
> An(x^2, x=-Pi..Pi, 0);
```

$$\frac{2\pi^2}{3}$$

```
> Bn:=proc(func, xrange::name=range, n::posint)
>    local l; global B;
>    l:= rhs( rhs(xrange) ) - lhs( rhs(xrange) );
>    B[n]:= (2/l)*int(func*sin(n*x),xrange);
> end proc:
> Bn(x,x=-Pi..Pi, 3);
```

$$\frac{2}{3}$$

```
> Bn(x,x=-Pi..Pi, 2);
```

$$-1$$

```
>
>
```

## Displaying a Single Approximant

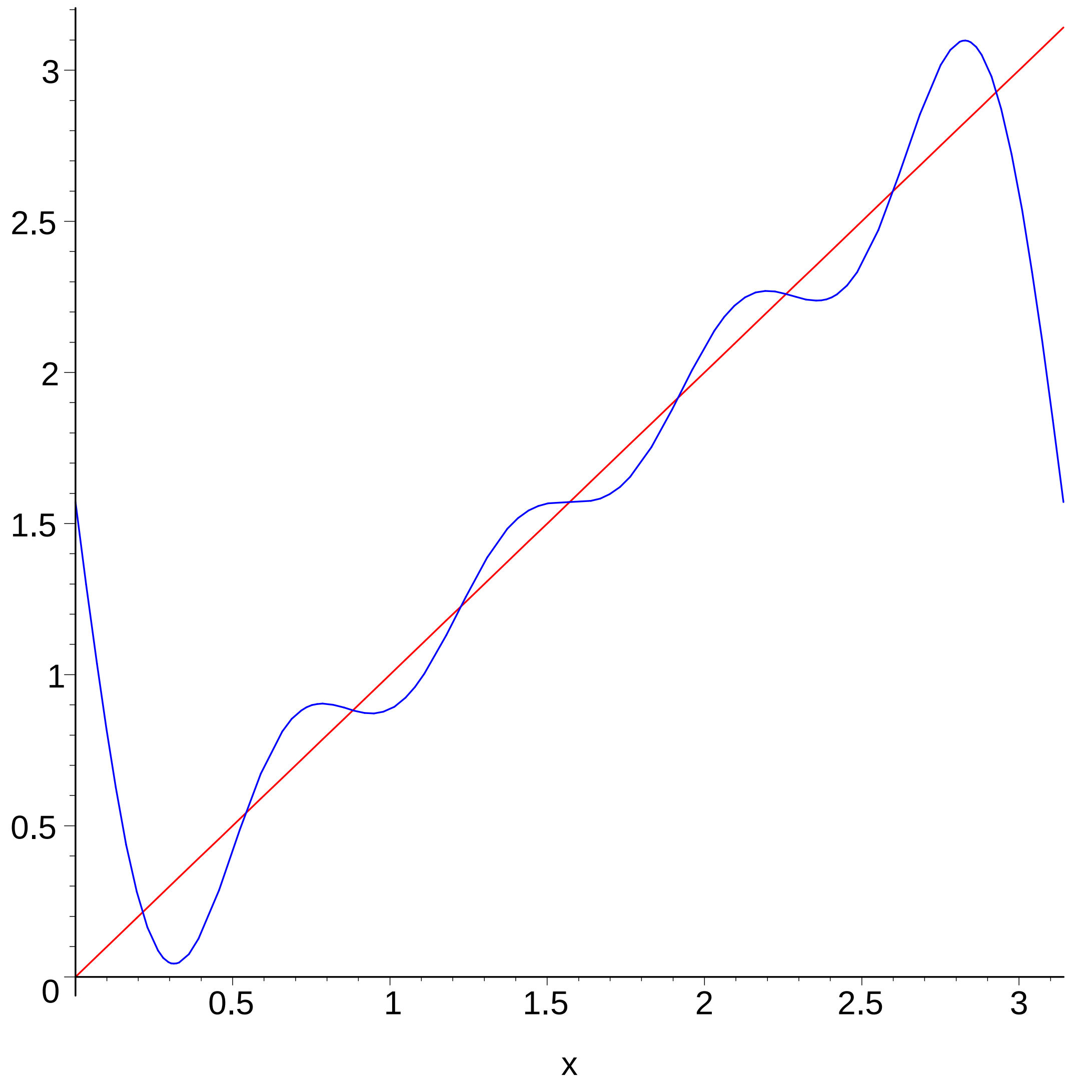
The procedure below displays the n-th Fourier approximant to **f**, together with the graph of **f** itself.

```
> Approximantn:=proc(func, xrange::name=range, n::posint)
```

```
  local x, a, b, l, k, p, partsum;
  global q;
  a:= lhs( rhs(xrange) );
  b:= rhs( rhs(xrange) );
  l:= b-a;
  x:= 2*Pi*lhs(xrange)/l;

  partsum := (1/l) * evalf(Int(func, xrange));
 for k from 1 to n do
    partsum:=
 partsum+(2/l)*evalf(Int(func*sin(k*x),xrange))*sin(k*x)+
              (2/l)*evalf(Int(func*cos(k*x),xrange))*cos(k*x);
 od;
 q[n]:=plot(partsum, xrange, color=blue, args[4..nargs]);
 p:=plot(func, xrange, color=red, args[4..nargs]);
 plots[display]([q[n],p]);
 end:
```

```
> Approximantn(x,x=0..Pi, 4);
```




```
>
>
```

## Animating the approximants

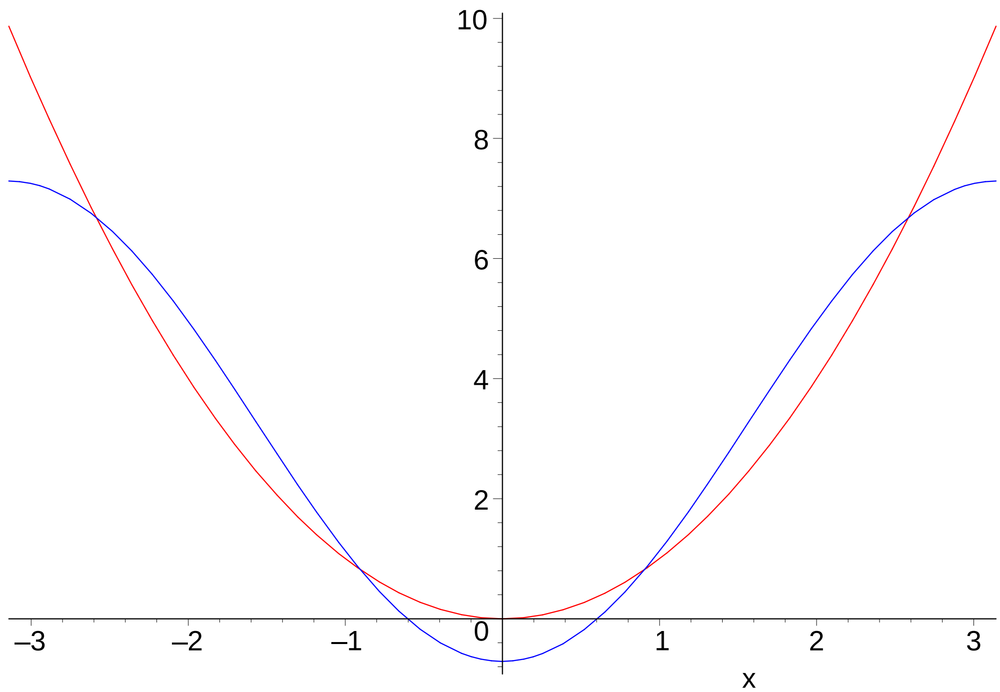
This procedure shows an animation of the Fourier approximants, and can be found in the Maple manual.

```
> FourierPicture:=proc(func, xrange::name=range, n::posint)
   local x, a, b, l, k, j, p, q, partsum;

   a:= lhs( rhs(xrange) );
   b:= rhs( rhs(xrange) );
   l:= b-a;
   x:= 2*Pi*lhs(xrange)/l;

   partsum := (1/l) * evalf(Int(func, xrange));
  for k from 1 to n do
     partsum:=
```

```
  partsum+(2/l)*evalf(Int(func*sin(k*x),xrange))*sin(k*x)+
               (2/l)*evalf(Int(func*cos(k*x),xrange))*cos(k*x);
     q[k]:=plot(partsum, xrange, color=blue, args[4..nargs]);
  od;
  q:=plots[display]([seq(q[k],k=1..n)],insequence=true);
  p:=plot(func, xrange, color=red, args[4..nargs]);
  plots[display]([q,p]);
  end:
> FourierPicture(x^2, x=-Pi..Pi, 20);
```

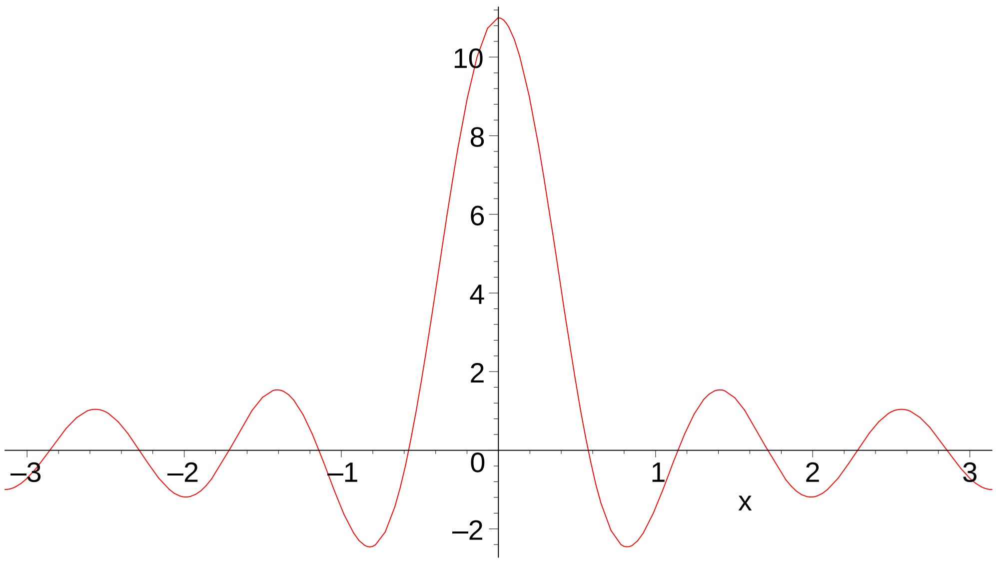



```
> #f:=x->piecewise(-Pi<x and x<0, 0, 0<x and x<Pi, 1);
> #FourierPicture(f(x), x=-Pi..Pi, 40);
>
```

# The Dirichlet Kernel

The following procedure plots the graph of the Dirichlet kernel over the interval -Pi to Pi.

```
> DN:=proc(N)
>   local j; global Dir;
>   Dir[N]:=1;
>   for j from 1 to N do
>     Dir[N]:=Dir[N]+2*cos(j*x);
>   end do;
>   plot(Dir[N], x=-Pi..Pi);
> end proc:
> DN(5);
```




```
> Int(Dir[5], x=-Pi..Pi);
```

$$\int_{-\pi}^{\pi} 1 + 2\cos(x) + 2\cos(2\,x) + 2\cos(3\,x) + 2\cos(4\,x) + 2\cos(5\,x)\,dx$$

```
> evalf(%);
```

$$6.283185307$$

```
>
>
```

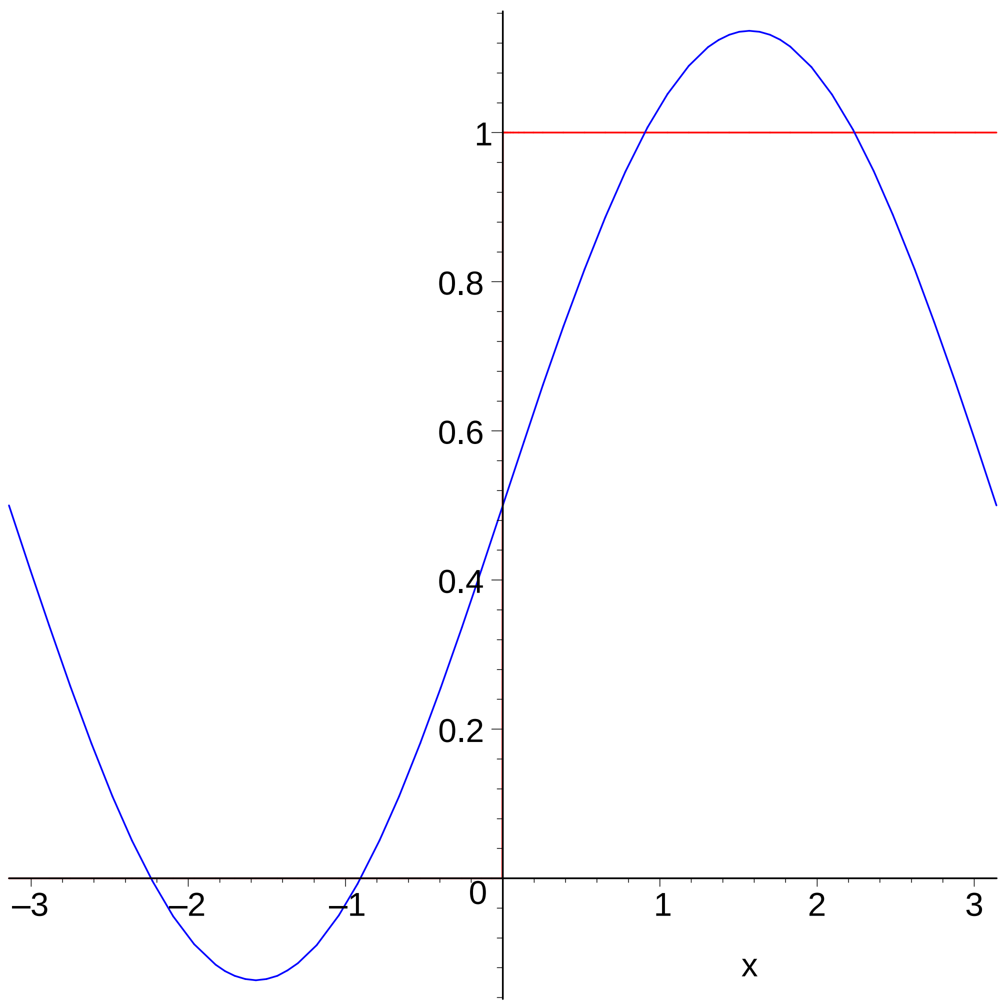
# Gibbs' Phenomenon

Whenever we compute the Fourier approximants of a discontinuous function f, there is a little bump that shows up
near every discontinuity of f. This is known as Gibbs' phenomenon, and can be easily verified graphically.

```
> f:=x->piecewise(-Pi<x and x<0, 0, 0<x and x<Pi, 1);
```

$$f := x \rightarrow \text{piecewise}(-\pi < x \textbf{ and } x < 0, 0, 0 < x \textbf{ and } x < \pi, 1)$$

```
> FourierPicture(f(x), x=-Pi..Pi, 40);
```




```
>
>
```

# Mean Square Convergence

This procedure computes the L^2 norm of  f - S(f,N) over the interval from -Pi to Pi.
It calls the procedures An and Bn defined previously.

```
> MeanSquare:=proc(func, N)
>    local g, j; global M;
```

```
>   g:=An(func, x=-Pi..Pi, 0)/2;
>   for j from 1 to N do
>     g:=g+An(func, x=-Pi..Pi, j)*cos(j*x)+Bn(func, x=-Pi..Pi,
  j)*sin(j*x);
>   end do;
>   M:=evalf(sqrt(int((func -g)^2, x=-Pi..Pi)));
> end proc:
> MeanSquare(x, 4);
```

1.667700900

```
> MeanSquare(x, 16);
```

0.8725625505

```
> MeanSquare(x, 64);
```

0.4413882174

```
> MeanSquare(x^2, 4);
```

0.4236902112

```
> MeanSquare(x^2, 16);
```

0.06101716507

```
> MeanSquare(x^2, 64);
```

0.007903769815

```
>
```