```
> restart:
  libname:="C:\\Maple\\lib",libname:
  with(WalshPaley):
```

The function **FCoeff(n,f)** gives us the **n**-th Fourier coefficient of the function **f**.
The returned value is exact and simplified.
Calculate the 100-th Fourier coefficient of the function f(x)=x*exp(x)

```
> SetDim(7):
> FCoeff(100,x*exp(x));
```

$$
\begin{aligned}
&1+\frac{3}{64}e^{\frac{125}{128}}-\frac{37}{64}e^{\frac{91}{128}}+\frac{93}{64}e^{\frac{35}{128}}+\frac{71}{64}e^{\frac{57}{128}}-\frac{95}{64}e^{\frac{33}{128}}-\frac{73}{64}e^{\frac{55}{128}}+\frac{3}{2}e^{\frac{1}{4}}\\
&+\frac{67}{64}e^{\frac{61}{128}}-\frac{33}{64}e^{\frac{95}{128}}-\frac{65}{64}e^{\frac{63}{128}}+\frac{39}{64}e^{\frac{89}{128}}-\frac{97}{64}e^{\frac{31}{128}}-\frac{5}{64}e^{\frac{123}{128}}+\frac{99}{64}e^{\frac{29}{128}}\\
&+\frac{75}{64}e^{\frac{53}{128}}+e^{\frac{1}{2}}-\frac{41}{64}e^{\frac{87}{128}}+\frac{7}{64}e^{\frac{121}{128}}-\frac{101}{64}e^{\frac{27}{128}}+\frac{1}{2}e^{\frac{3}{4}}-\frac{77}{64}e^{\frac{51}{128}}\\
&+\frac{103}{64}e^{\frac{25}{128}}+\frac{43}{64}e^{\frac{85}{128}}-\frac{63}{64}e^{\frac{65}{128}}-\frac{9}{64}e^{\frac{119}{128}}-\frac{105}{64}e^{\frac{23}{128}}+\frac{79}{64}e^{\frac{49}{128}}\\
&+\frac{107}{64}e^{\frac{21}{128}}-\frac{45}{64}e^{\frac{83}{128}}-\frac{5}{4}e^{\frac{3}{8}}+\frac{11}{64}e^{\frac{117}{128}}-\frac{31}{64}e^{\frac{97}{128}}-\frac{109}{64}e^{\frac{19}{128}}+\frac{81}{64}e^{\frac{47}{128}}\\
&+\frac{111}{64}e^{\frac{17}{128}}+\frac{47}{64}e^{\frac{81}{128}}+\frac{61}{64}e^{\frac{67}{128}}+\frac{35}{64}e^{\frac{93}{128}}-\frac{13}{64}e^{\frac{115}{128}}-\frac{7}{4}e^{\frac{1}{8}}-\frac{69}{64}e^{\frac{59}{128}}\\
&-\frac{1}{64}e^{\frac{127}{128}}+\frac{113}{64}e^{\frac{15}{128}}-\frac{3}{4}e^{\frac{5}{8}}-\frac{83}{64}e^{\frac{45}{128}}+\frac{49}{64}e^{\frac{79}{128}}-\frac{115}{64}e^{\frac{13}{128}}+\frac{15}{64}e^{\frac{113}{128}}\\
&+\frac{85}{64}e^{\frac{43}{128}}+\frac{117}{64}e^{\frac{11}{128}}-\frac{1}{4}e^{\frac{7}{8}}-\frac{59}{64}e^{\frac{69}{128}}-\frac{51}{64}e^{\frac{77}{128}}-\frac{119}{64}e^{\frac{9}{128}}+\frac{17}{64}e^{\frac{111}{128}}\\
&-\frac{87}{64}e^{\frac{41}{128}}+\frac{121}{64}e^{\frac{7}{128}}+\frac{29}{64}e^{\frac{99}{128}}+\frac{53}{64}e^{\frac{75}{128}}-\frac{123}{64}e^{\frac{5}{128}}-\frac{19}{64}e^{\frac{109}{128}}+\frac{89}{64}e^{\frac{39}{128}}\\
&+\frac{125}{64}e^{\frac{3}{128}}-\frac{55}{64}e^{\frac{73}{128}}+\frac{21}{64}e^{\frac{107}{128}}+\frac{57}{64}e^{\frac{71}{128}}-\frac{127}{64}e^{\frac{1}{128}}-\frac{91}{64}e^{\frac{37}{128}}-\frac{23}{64}e^{\frac{105}{128}}\\
&+\frac{25}{64}e^{\frac{103}{128}}-\frac{27}{64}e^{\frac{101}{128}}
\end{aligned}
\tag{1}
$$

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

$$-0.0000117110 \tag{2}$$

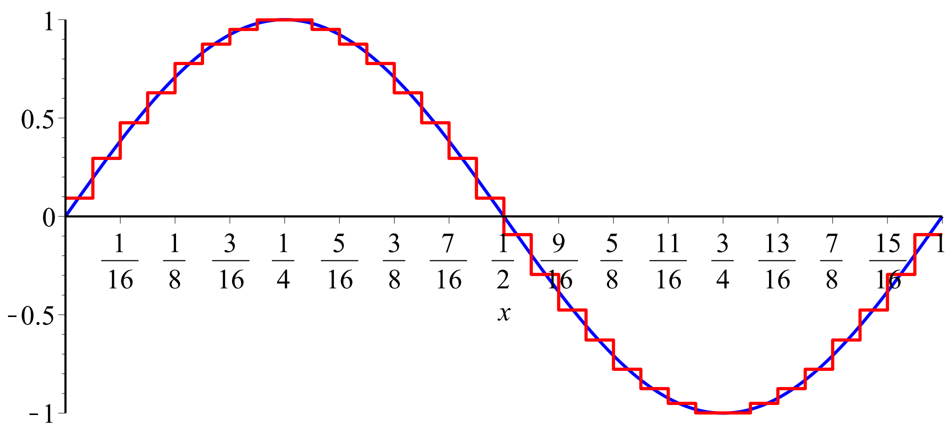
We recommend the use of the command **FCoeffs(n1,n2,f)** to obtain the Fourier coefficient from **n1** to **n2**.
We can use the option floating to calculate the Fourier coefficients numerically.
Calculate numerically the Fourier coefficient of the function f(x)=x*exp(x) for indices from 20 to 25.

```
> FCoeffs(20,25,x*exp(x),floating);
```

[0.00432196850520003, -0.00145232453480000, -0.000743672194800010,
0.000231108645200012, 0.00216472749519998, -0.000727271224799979] **(3)**

With the commands **SumVal(sor)** or **SumFunc(sor,x)** where **sor** is **FCoeffs(0,n-1,f)** we obtain the **n**-th partial sum of Fourier series of the function **f** with respect to the Walsh-Paley system.
Plot the 28-th partial sum of Fourier series for f(x)=sin(2*PI*x) and calculate the partial sum for x=0.4.

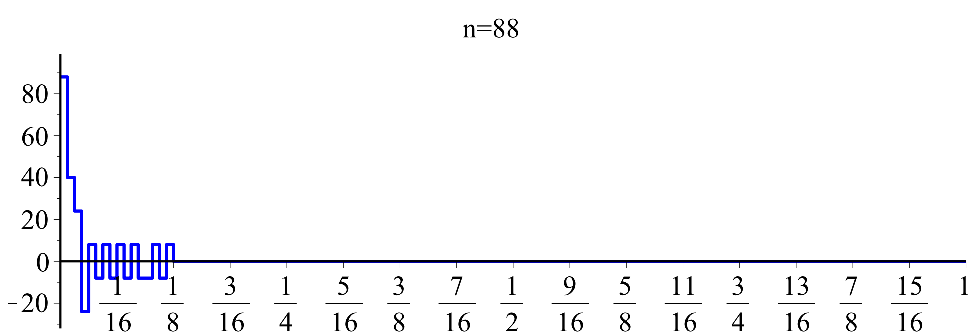
```
> f:=x->sin(2*Pi*x):n:=28:fPlot:=plot(f(x),x=0..1,color=blue):
  FourierPlot:=lineplot(SumVal(FCoeffs(0,n-1,f(x),floating)),color=
  red):
  plots[display]([fPlot, FourierPlot],tickmarks=[[seq(i/16=i/16,i=
  1..16)],default]);
  SumFunc(FCoeffs(0,n-1,f(x)),0.4);
```



$$\frac{2\left(2\cos\left(\frac{7}{16}\pi\right)+2\cos\left(\frac{1}{16}\pi\right)+6\cos\left(\frac{3}{16}\pi\right)-3\sqrt{2}-1-2\cos\left(\frac{5}{16}\pi\right)\right)}{\pi} \quad \textbf{(4)}$$

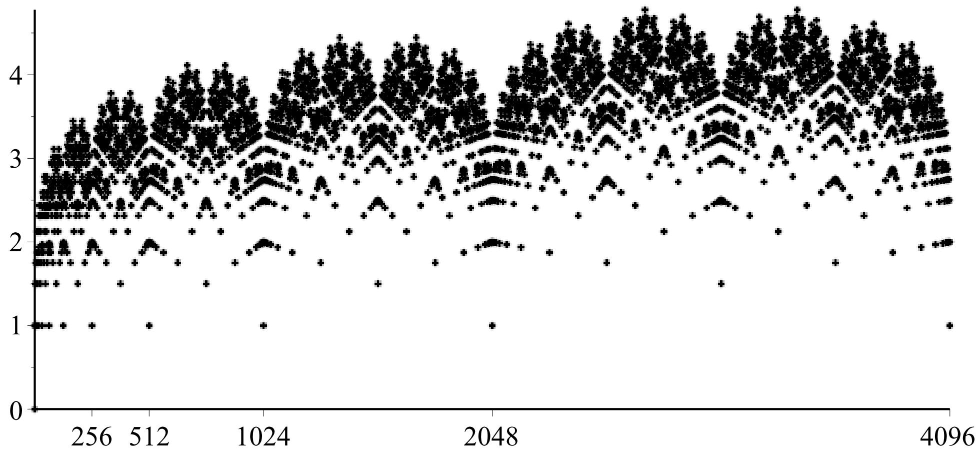
The Dirichlet kernels with respect to the Walsh-paley system can be obtained by the comands **DiriVal(n)** and **DiriFunc(n,x).**
DiriPVal(n) gives us a row vector with the values of the Dirichlet kernels of index **n** at the dyadic intervals. The values are calculated recursively, hence we do not deal with setting the dimension.
The function **DiriFunc(n,x)** gives us the value of the Dirichlet kernels of index **n** at the point **x**.
Create an animation with the first 100 Dirichlet kernels.

```
> plots[display](seq(lineplot(DiriVal(n),tozero,title=cat("n=",n),
  color="blue"), n=0..99), insequence=true,tickmarks=[[seq(i/16=
  i/16,i=1..16)],default]);
```



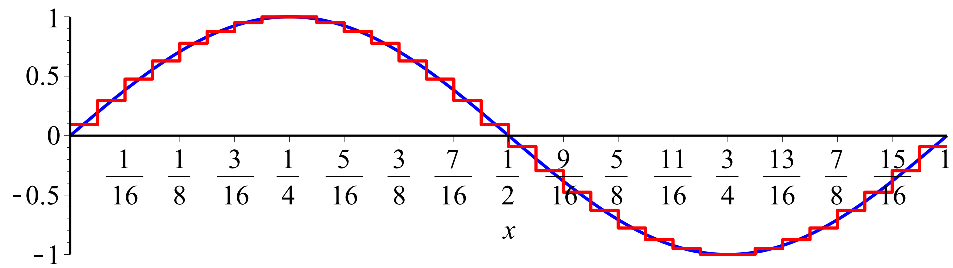
`LebConst(n)` gives us the Lebesgue point of index `n`.
The Lebesgue points are calculated recursively, hence we do not deal with setting the dimension.
Plot the Lebesgue points from 0 up to 4096.

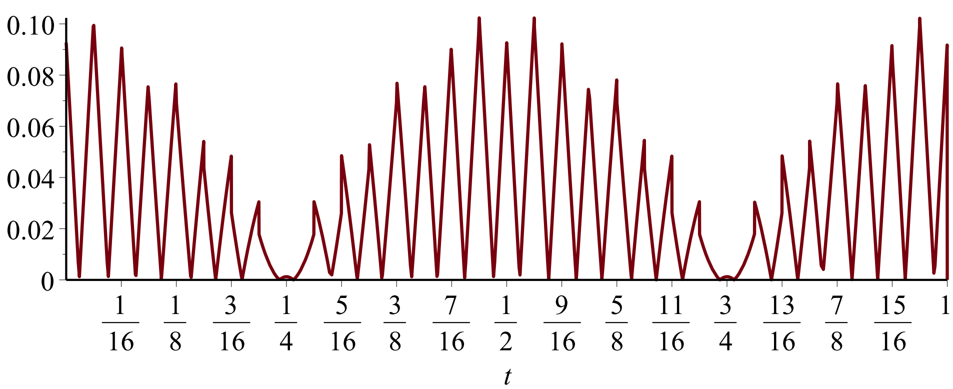
```
> m:=2^12:
  plots[pointplot]({seq([n, LebConst(n)], n = 0 .. m)}, symbol=
  diamond, symbolsize=4,tickmarks=[[seq(2^i,i=8..12)],default]);
```



The commands **FSumVal(n,f)** and **FSumFunc(n,f,x)** gives us the **n**-th partial sum of Fourier series of the function **f**.
This commands compute the partial sums from the values of Dirichlet kernels using convolution.
Like Dirichlet kernels it is not necessary to set dimensions.
We can use the option floating to calculate the values numerically.
Plot the 28-th partial sum of Fourier series for f(x)=sin(2*PI*x) and the absolute difference between the function and the partial sum..

```
> f:=x->sin(2*Pi*x):n:=28:
  fPlot:=plot(f(x),x=0..1,color=blue):
  FourierPlot:=lineplot(FSumVal(n,f(x),floating),color=red):
  plots[display]([fPlot, FourierPlot],tickmarks=[[seq(i/16=i/16,i=
  1..16)],default]);
  plot(abs(f(t)-FSumFunc(n,f(x),t)),t=0..1,tickmarks=[[seq(i/16=
  i/16,i=1..16)],default]);
```

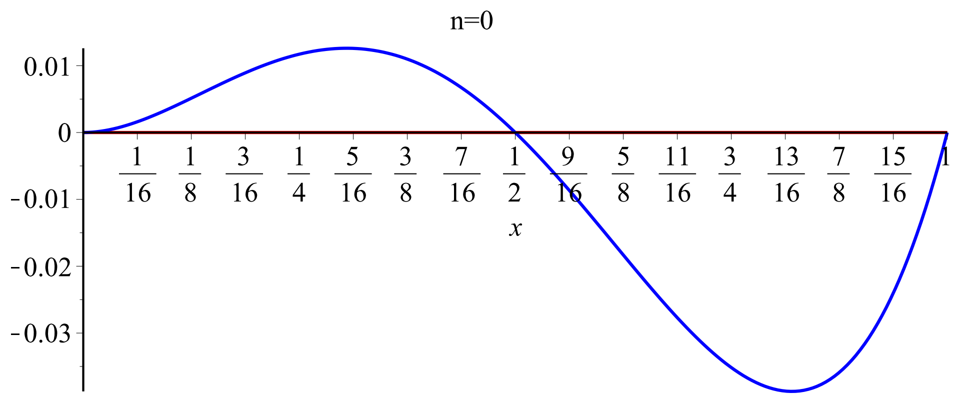




The command **FSumsVal(k,f)** returns a matrix with the values of the partial sums of Fourier series from 1 up to 2^k.
We suggest to use this matrix to create animations.

```
> SetDim(9):
  f:=x-> x^2*(x-1)*(x-1/2):
  n:=500:
```

```
> fPlot:=plot(f(x),x=0..1,color=blue,tickmarks=[[seq(i/16=i/16,i=1.
  .16)],default]):
  FSV:=FSumsVal(9,f(x),floating):
  for i from 1 to n do
      FourierPlots[i]:=lineplot(FSV[i], title=cat("n=",i), color=
  red)
  end do:
  FourierPlots[0]:=lineplot([0], title=cat("n=",0), color=red):
  FourierSeq:=plots[display](seq(FourierPlots[i], i=0..n),
  insequence=true):
  plots[display](FourierSeq,fPlot);
```



Due to their importance in applications, the package is able to handle separately the 2^n-th partial sums of Fourier series. For that we have the commands **SVal(n,f)** and **SFunk(n,f)** which are used just like **FSumsVal(2^n,f)** and **FSumsFunk(2^n,f)** respectively, but they work faster.

```
> f:=x->x^2-0.5*x:n:=5:
  fPlot:=plot(f(x),x=0..1,color=blue):
  FourierPlot:=lineplot(SVal(n,f(x),floating),color=red):
  plots[display]([fPlot, FourierPlot],tickmarks=[[seq(i/16=i/16,i=
  1..16)],default]);
  plot(abs(f(t)-SFunc(n,f(x),t,floating)),t=0..0.99999999,
  tickmarks=[[seq(i/16=i/16,i=1..16)],default]);
```