Maxima by Example:
Ch.10: Fourier Series, Fourier and Laplace Transforms *

Edwin L. Woollett

September 16, 2010

Contents

Feedback from readers is the best way for this series of notes to become more helpful to new users of Maxima. *All* comments and suggestions for improvements will be appreciated and carefully considered.

```
The Maxima code examples in this chapter were generated
using the XMaxima graphics interface on a Windows XP computer, and copied
into a fancy verbatim environment in a latex file which uses the fancyvrb
and color packages.
We use qdraw.mac for plots (see Ch.5), which uses draw2d defined
in share/draw/draw.lisp.
```

10.1 Introduction

In chapter 10 we discuss the Fourier series expansion of a given function, the computation of Fourier transform integrals, and the calculation of Laplace transforms (and inverse Laplace transforms).

10.2 Fourier Series Expansion of a Function

10.2.1 Fourier Series Expansion of a Function over $(-\pi, \pi)$

A Fourier series expansion designed to represent a given function $\mathbf{f}(\mathbf{x})$ defined over a finite interval $(-\pi, \pi)$, is a sum of terms

$$\mathbf{f}(\mathbf{x}) = \frac{1}{2}\,\mathbf{a_0} + \sum_{n=1}^{\infty} [\mathbf{a_n} \cos(n\,x) + \mathbf{b_n} \sin(n\,x)] \tag{10.1}$$

and the constant coefficients $(\mathbf{a_n},\ \mathbf{b_n})$ are

$$\mathbf{a_n} = \frac{1}{\pi} \int_{-\pi}^{\pi} \mathbf{f}(\mathbf{y}) \cos(n\,y)\,\mathbf{dy}, \quad \mathbf{b_n} = \frac{1}{\pi} \int_{-\pi}^{\pi} \mathbf{f}(\mathbf{y}) \sin(n\,y)\,\mathbf{dy}. \tag{10.2}$$

(For a derivation of these equations see Sec.10.2.8 and Eqs.(10.20) and (10.21).)

Whether or not you are working with a function which is periodic, the Fourier expansion will represent a periodic function for all $\mathbf{x}$, in this case having period $2\,\pi$.

The first term of the expansion

$$\frac{1}{2}\,\mathbf{a_0} = \frac{1}{2\,\pi} \int_{-\pi}^{\pi} \mathbf{f}(\mathbf{y})\,\mathbf{dy} = \langle \mathbf{f}(\mathbf{x}) \rangle \tag{10.3}$$

is the (un-weighted) average of $\mathbf{f}(\mathbf{x})$ over the domain $(-\pi, \pi)$. Hence $\mathbf{a_0}$ will always be twice the average value of the function over the domain.

If $\mathbf{f}(\mathbf{x})$ is an even function $(\mathbf{f}(-\mathbf{x}) = \mathbf{f}(\mathbf{x}))$, then only the $\cos(n\,x)$ terms contribute. If $\mathbf{f}(\mathbf{x})$ is an odd function $(\mathbf{f}(-\mathbf{x}) = -\mathbf{f}(\mathbf{x}))$, then only the $\sin(n\,x)$ terms contribute.

If you are trying to find a fourier expansion for a function or expression $\mathbf{f}(\mathbf{x})$ which is a homemade function which involves "if..then..else" constructs, it is necessary to do the preliminary work "by hand".

On the other hand, if the given function is a smooth function defined in terms of elementary functions and polynomials, or includes **abs** of elements of the function, one can use the **fourie** package to do most of the work in obtaining the desired Fourier series expansion. This package is **calculus/fourie.mac**, and there is also a short **demo** file **fourie.dem**.

Although many secondary functions defined in this mac file are usable once you load the package, they are not documented in the Maxima manual. The Maxima manual gives a brief definition of the primary tools available, but gives no examples of use, nor is there an **example** file for such primary functions as **fourier**, **foursimp**, and **fourexpand**.

If the Fourier series integrals needed for the coefficients are too difficult for **integrate**, you should use the Quadpack function **quad_qawo** described in Chapter 8, Numerical Integration.

10.2.2 Fourier Series Expansion of $f(x) = x$ over $(-\pi, \pi)$

We first use the coefficient formulas Eq.(10.2) to find the Fourier expansions of this simple linear function $f(x) = x$ by hand.

Because the average value of $\mathbf{f(x)}$ over the domain $(-\pi, \pi)$ is zero, $\mathbf{a_0 = 0}$. Because $\mathbf{f(x)}$ is an odd function, we have $\mathbf{a_n = 0}$ for all $\mathbf{n > 0}$.

```
(%i1) (declare(n,integer), assume(n > 0 ), facts() );
(%o1)                      [kind(n, integer), n > 0]
(%i2) define(b(n),integrate(x*sin(n*x),x,-%pi,%pi)/%pi );
                                          n
                                   2 (- 1)
(%o2)                      b(n) := - --------
                                        n
(%i3) map('b,makelist(i,i,1,7));
                           2    1 2    1 2
(%o3)                [2, - 1, -, - -, -, - -, -]
                           3    2 5    3 7
(%i4) fs(nmax) := sum( b(m)*sin(m*x),m,1,nmax)$
(%i5) map('fs,[1,2,3,4] );
                         2 sin(3 x)
(%o5) [2 sin(x), 2 sin(x) - sin(2 x), ---------- - sin(2 x) + 2 sin(x),
                             3

                         sin(4 x)   2 sin(3 x)
                       - -------- + ---------- - sin(2 x) + 2 sin(x)]
                            2           3
```

The list contains, in order, the lowest approximation $2\sin(x)$ which retains only the $n = 1$ term in the expansion, the two term approximation $2\sin(x) - \sin(2x)$, which includes the $n = 1, 2$ terms, etc. We now load the **draw** package and the **qdraw** package (the latter available with Ch. 5 material on the author's webpage) to make two simple plots. We first make a plot showing the function $f(x) = x$ in blue, the one term approximation (`fs(1) = 2 sin(x)`) in red, and the two term approximation (`fs(2) = 2 sin(x) - sin(2 x)`) in green.

```
(%i6) (load(draw), load(qdraw) )$
            qdraw(...), qdensity(...), syntax: type qdraw();

(%i7) qdraw( xr(-5.6,5.6),yr(-4,4),
            ex([x,fs(1),fs(2)],x,-%pi,%pi),key(bottom) )$
```

In this plot (see next page), we have taken control of the **x** and **y** range, using the approximate fudge factor **1.4** to relate the horizontal canvas extent to the vertical canvas extent (see our discussion in Ch. 5 if this is all new to you) to get the geometry approximately correct. In the next plot we include one, two, three and four term approximations.

```
(%i8) qdraw( xr(-5.6,5.6),yr(-4,4),
            ex([x,fs(1),fs(2),fs(3),fs(4)],x,-%pi,%pi),key(bottom) )$
```

with the four term approximation in purple being a closer approximation to $f(x) = x$ (see the figure on the next page).
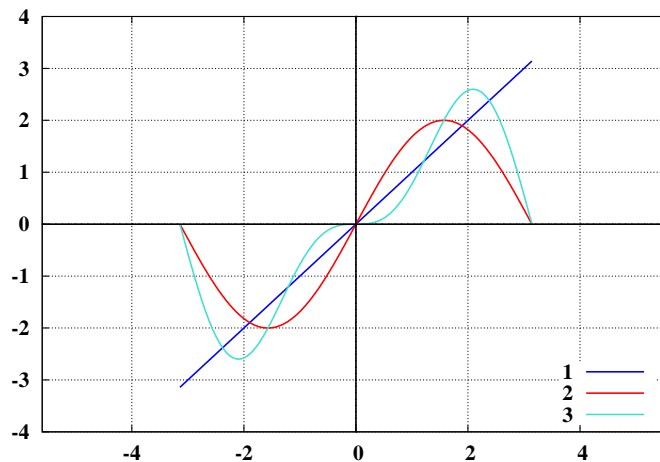
Figure 1: One and Two Term Approximations
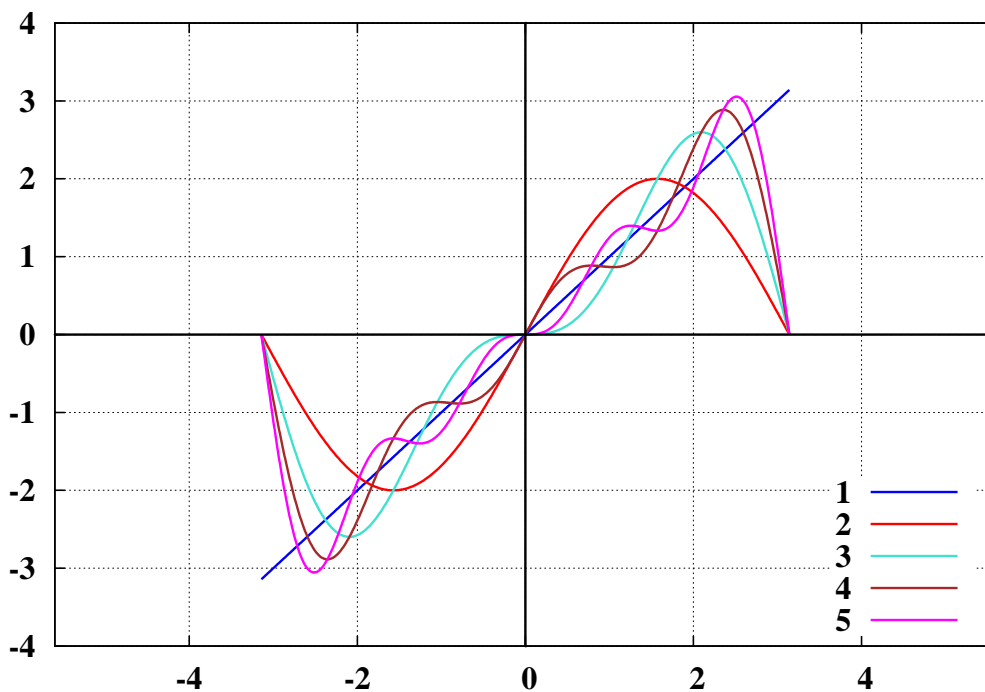
Figure 2: 1, 2, 3, and 4 Term Approximations

10.2.3 The calculus/fourie.mac Package: fourier, foursimp, fourexpand

Now let's show how to get the expansions of this simple linear function using the package **calculus/fourie.mac**. A curious feature of **fourie.mac** is the role that the symbol **n** plays in the calculation of the fourier coefficients. If you look at the code of **calculus/fourie.mac** (about a four page text file), you will see that the function **fourier** calls either **fourcos**, **foursin**, or **fourcoeff**, and each of these latter functions declare **n** to be a local variable, and then have the statement (inside the **block**) `assume( n > 0 )`, which "leaks out" of the **block** to become a global assumption. These functions then call a package function **adefint(..)** to integrate the function to be expanded times either $\cos(n\pi x/p)$ or $\sin(n\pi x/p)$, (where **p** will be π for our expansion domain here).

5

Finally, after dealing with possible instances of **abs(..)** in the function to be expanded, **adefint** calls either **ldefint** or **integrate** with an argument involving these same **n** dependent trig functions.

Here is a small test of that leakage for both an **assume** statement and a **declare** statement.

```
(%i1) facts();
(%o1)                                      []
(%i2) f(m) := block([n],declare(n,integer),assume( n > 0 ),
              if m < 2 then n :2 else n:3,(2*n*m) )$
(%i3) f(1);
(%o3)                                       4
(%i4) facts();
(%o4)                        [kind(n, integer), n > 0]
(%i5) is(n>0);
(%o5)                                     true
```

Another curious feature of the **fourie.mac** package is that **n** is not declared to be an integer, and unless the user of the package does this first, the integration routine may ask questions about signs which may seem irrelevant to the result. We avoid that trap here and use **declare(n, integer)** before calling **fourier**.

```
(%i1) facts();
(%o1)                                      []
(%i2) (load(fourie), facts() );
(%o2)                                      []
(%i3) (declare(n,integer), facts() );
(%o3)                            [kind(n, integer)]
(%i4) clist : fourier(x,x,%pi);
(%t4)                              a  = 0
                                    0


(%t5)                              a  = 0
                                    n


                                          n
                                   2 (- 1)
(%t6)                        b  = - --------
                              n        n


(%o6)                             [%t4, %t5, %t6]
(%i7) facts();
(%o7)                        [kind(n, integer), n > 0]
(%i8) fs(nmax) := fourexpand(clist,x,%pi, nmax )$
(%i9) map( 'fs, [1,2,3,4] );
                                     2 sin(3 x)
(%o9) [2 sin(x), 2 sin(x) - sin(2 x), ---------- - sin(2 x) + 2 sin(x),
                                          3
                                  sin(4 x)   2 sin(3 x)
                                - -------- + ---------- - sin(2 x) + 2 sin(x)]
                                     2           3
(%i10) b(n);
(%o10)                               b(n)
```

Some comments: you can use **foursimp** to get extra simplification of the fourier coefficients (if needed) before defining what I call **clist** (for coefficient list) which is the list **fourexpand** needs to generate the expansions you want. Note **fourier** produces a separate output for a_0 and a_n, with the second meant for $n = 1, 2, \ldots$, and there is never an output for b_0. We have defined the small function **fs(nmax)** to make it easier to call

fourexpand with any value of **nmax** and to allow the function to be mapped onto a list of integers to show us the first few approximations. Note that once you call **fourier**, the assumption $n > 0$ becomes a global fact. Also note that the **fourie.mac** package does not define a Maxima function $b(n)$, although you could use:

```
(%i11) define(b(n),rhs(%t6) );
                                             n
                                  2 (- 1)
(%o11)                   b(n)  := - --------
                                        n
(%i12) map( 'b, makelist(i,i,1,7) );
                             2    1  2     1  2
(%o12)              [2,  - 1, -,  - -, -,  - -, -]
                             3    2  5     3  7
```

If you look at the Maxima code in **fourie.mac**, you see that because we have an "odd" function $f(x) = x$, **fourier** calls the package function **foursin**, which calls package function **adefint**, which calls the core Maxima function **ldefint** for this case.

Those expansion expressions can then be used for plots as above.

10.2.4   Fourier Series Expansion of a Function Over $(-p, \, p)$

The Fourier series expansion of a function defined over the interval $-p \le x \le p$ (and whose Fourier expansion will represent a function which has a period $2\,p$) can be found from the expansion over the interval $(-\pi, \, \pi)$ which we have been using above by a simple change of variables in the integrals which appear in Eqs.(10.2) and (10.1).

However, we will simply use the results derived in Sec.10.2.8, and written down in Eqns (10.18) and (10.19).

$$f(x) = \frac{1}{2}\, a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{\pi\, n\, x}{p}\right) + b_n \sin\left(\frac{\pi\, n\, x}{p}\right) \right] \tag{10.4}$$

with the corresponding coefficients (for a_n, $n = 0, 1, ...$, for b_n, $n = 1, ...$):

$$a_n = \frac{1}{p} \int_{-p}^{p} f(y) \cos\left(\frac{\pi\, n\, y}{p}\right) dy, \quad b_n = \frac{1}{p} \int_{-p}^{p} f(y) \sin\left(\frac{\pi\, n\, y}{p}\right) dy. \tag{10.5}$$

We need to warn the user of the package **fourie.mac** that they use the symbol a_0 to mean *our* $a_0/2$.

Our a_0 is defined by

$$a_0 = \frac{1}{p} \int_{-p}^{p} f(x)\, dx \tag{10.6}$$

The **fourie.mac** package's definition of *their* a_0 is

$$[a_0]_{\text{fourie.mac}} = \frac{1}{2\,p} \int_{-p}^{p} f(x)\, dx \tag{10.7}$$

which defines the average value of the function over the domain and which becomes the first term of the fourier series expansion they provide.

7

10.2.5 Fourier Series Expansion of the Function $|x|$

We define the function to have period 4 with $f(x) = |x|$ for $-2 \le x \le 2$. This function is an even function of x so the **sin** coefficients b_n are all zero. The average value of $|x|$ over the domain $(-2, 2)$ is greater than zero so we willl have a non-zero coefficient a_0 which we will calculate separately. We do this calculation by hand here.

Note that the Maxima function **integrate** cannot cope with **abs(x)**:

```
(%i13) integrate(abs(x)*cos(n*%pi*x/2),x,-2,2)/2;
                  2
                 /
                 [              %pi n x
                 I    abs(x) cos(-------) dx
                 ]                  2
                 /
                  - 2
(%o13)           --------------------------
                             2
```

so we will split up the region of integration into two sub-intervals: $-2 \le x \le 0$, in which $|x| = -x$, and the interval $0 \le x \le 2$ in which $|x| = x$. We will use the formula Eq. (10.5) for the coefficients a_n (note that $a_n = 0$ if **n** is even) and the expansion formula Eq. (10.4).

```
(%i1) (declare(n,integer),assume(n > 0), facts() );
(%o1)                   [kind(n, integer), n > 0]
(%i2) a0 :integrate(-x,x,-2,0)/2 + integrate(x,x,0,2)/2;
(%o2)                            2
(%i3) an : integrate((-x)*cos(n*%pi*x/2),x,-2,0)/2 +
           integrate(x*cos(n*%pi*x/2),x,0,2)/2;
                              n
                      4 (- 1)         4
(%o3)                 -------- - -------
                        2  2       2  2
                      %pi  n     %pi  n
(%i4) an : (ratsimp(an), factor(%%) );
                              n
                      4 ((- 1)  - 1)
(%o4)                 --------------
                          2  2
                       %pi  n
(%i5) define(a(n),an);
                                   n
                         4 ((- 1)  - 1)
(%o5)             a(n) := --------------
                              2  2
                           %pi  n
(%i6) map( 'a, [1,2,3,4,5] );
                     8             8              8
(%o6)          [- ----, 0, - ------, 0, - -------]
                    2          2              2
                  %pi        9 %pi        25 %pi
(%i7) fs(nmax) := a0/2 + sum(a(m)*cos(m*%pi*x/2),m,1,nmax)$
```

```
(%i8) map('fs, [1, 3, 5] );
                %pi x              3 %pi x              %pi x
          8 cos(-----)      8 cos(-------)      8 cos(-----)
                  2                    2                    2
(%o8) [1 - -------------, - -------------- - ------------- + 1,
                  2                    2                    2
              %pi                  9 %pi                %pi
                          5 %pi x              3 %pi x              %pi x
                    8 cos(-------)      8 cos(-------)      8 cos(-----)
                            2                    2                    2
                  - -------------- - -------------- - ------------- + 1]
                          2                    2                    2
                      25 %pi                9 %pi                %pi
(%i9) (load(draw),load(qdraw))$
          qdraw(...), qdensity(...), syntax: type qdraw();

(%i10) qdraw( ex([abs(x),fs(1)],x,-2,2),key(bottom) )$
(%i11) qdraw( ex([abs(x),fs(1),fs(3)],x,-2,2),key(bottom) )$
(%i12) qdraw( ex([abs(x),fs(5) ],x,-2,2),key(bottom) )$
```

The first function in the plot list is $|x|$, represented by **abs(x)**, which appears in the color blue. We see that the expansion out to $n = 5$ provides a close fit to $|x|$. Here is that comparison:
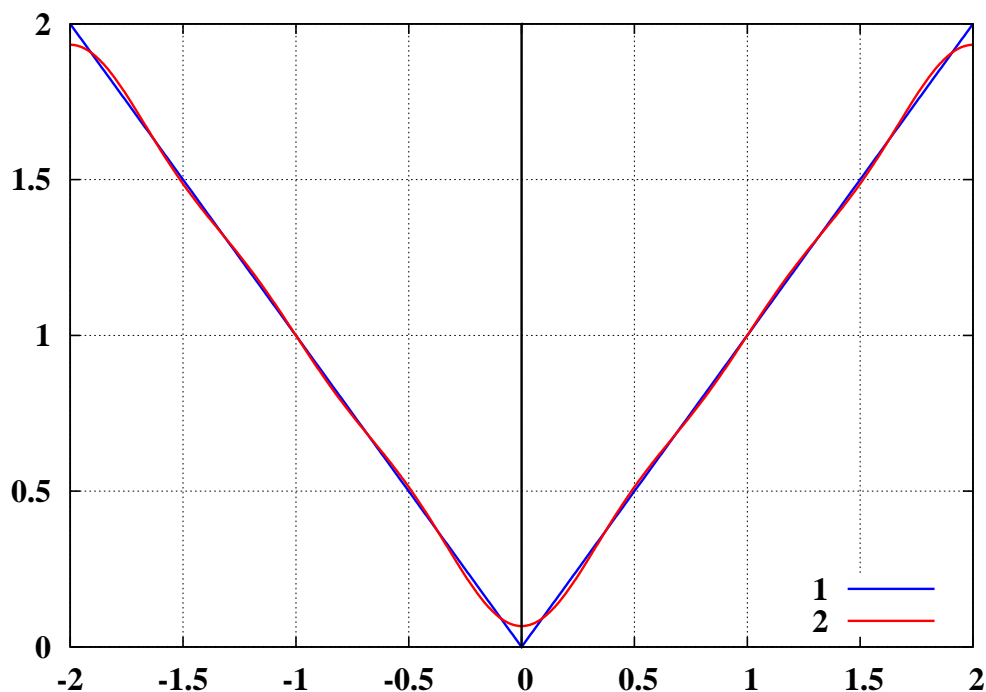

Figure 3: $n = 5$ Approximation to $|x|$

We now try out the package **fourie.mac** on this example:

```
(%i1) ( load(fourie),facts() );
(%o1)                              []
(%i2) (declare(n,integer),facts());
(%o2)                      [kind(n, integer)]
```

```
(%i3) fourier(abs(x),x,2);
(%t3)                                    a   = 1
                                          0


                                             n
                                      4 (- 1)        4
(%t4)                         a   = -------- - -------
                               n       2  2       2  2
                                     %pi  n      %pi  n


(%t5)                                    b   = 0
                                          n


(%o5)                             [%t3, %t4, %t5]
(%i6) clist : foursimp(%);
(%t6)                                    a   = 1
                                          0


                                                n
                                      4 ((- 1)  - 1)
(%t7)                         a   = --------------
                               n          2  2
                                        %pi  n


(%t8)                                    b   = 0
                                          n


(%o8)                             [%t6, %t7, %t8]
(%i9) facts();
(%o9)                        [kind(n, integer), n > 0]
(%i10) fs(nmax) := fourexpand(clist,x,2,nmax )$
(%i11) map( 'fs, [1,3,5] );
                    %pi x            3 %pi x            %pi x
             8 cos(-----)       8 cos(-------)     8 cos(-----)
                     2                   2                 2
(%o11) [1 - ------------, - -------------- - ------------ + 1,
                 2                 2                2
               %pi             9 %pi            %pi
                              5 %pi x            3 %pi x        %pi x
                       8 cos(-------)     8 cos(-------)   8 cos(-----)
                               2                  2              2
                     - -------------- - -------------- - ------------ + 1]
                             2                 2                2
                         25 %pi            9 %pi             %pi
```

Notice that $(a_0)_\text{fourie} = 1 = \frac{1}{2}(a_0)_\text{ourdef}$, (see Eq. (10.7)) so that **fourie.mac**'s expansion starts off with the term a_0 rather than $\frac{1}{2}a_0$. Of course, the actual end results look the same, with the first term in this example being **1**, which is the average value of $|x|$ over the domain $(-2, 2)$.

Here we chose to use the package function **foursimp** to simplify the appearance of the coefficients. We see that **fourie.mac** is able to cope with the appearance of **abs(x)** in the integrand, and produces the same coefficients and expansions as were found "by hand".

10.2.6   Fourier Series Expansion of a Rectangular Pulse

We define $f(x)$ to be a function of period 4, with $f = 0$ for $-2 \leq x < -1$, $3/2$ for $-1 \leq x \leq 1$ and $f = 0$ for $1 < x \leq 2$.

```
(%i1) f(x):= if x >= -1 and x <= 1 then 3/2 else 0$
(%i2) map('f, [-3/2,-1,0,1,3/2] );
                                3  3  3
(%o2)                      [0, -, -, -, 0]
                                2  2  2
(%i3) (load(draw),load(qdraw) )$
             qdraw(...), qdensity(...), syntax: type qdraw();

(%i4) qdraw( yr(-0.5,2), ex1(f(x),x,-2,2,lw(5),lc(blue) ) )$
```

The plot of $f(x)$ shows a square pulse with height $3/2$ above the x axis and with a width of 2 units over the interval $-1 \leq x \leq 1$. Over the rest of the domain $-2 \leq x \leq 2$, $f(x)$ is defined to be zero.
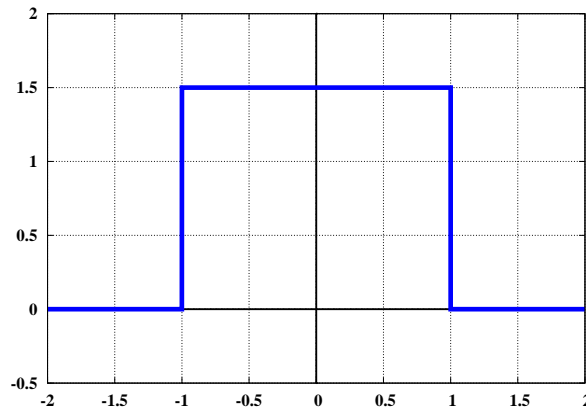
Figure 4: Rectangular Pulse of Height $3/2$

Although we can make a plot of the function $f(x)$ as defined , Maxima's **integrate** function cannot do anything useful with it, and hence neither can **fourie.mac** at the time of writing.

```
(%i5) integrate(f(x),x,-2,2);
             2
            /
            [                                      3
(%o5)       I    (if (x >= - 1) and (x <= 1) then - else 0) dx
            ]                                      2
            /
             - 2
```

Hence we must compute the Fourier series expansion "by hand". We see that $f(x)$ is an even function $(f(-x) = f(x))$, so only the $\cos(n\pi x/2)$ terms will contribute, so we only need to calculate the a_n coefficients.

```
(%i6) (declare(n,integer), assume(n>0),facts() );
(%o6)                    [kind(n, integer), n > 0]
```

```
(%i7) a0 : (1/2)*integrate( (3/2),x,-1,1 );
                              3
(%o7)                         -
                              2
(%i8) define(a(n),(1/2)*integrate((3/2)*cos(n*%pi*x/2),x,-1,1));
                                    %pi n
                              3 sin(-----)
                                      2
(%o8)                   a(n) := ------------
                                   %pi n
(%i9) map( 'a, makelist(i,i,1,7) );
                      3         1        3            3
(%o9)              [---, 0, - ---, 0, -----, 0, - -----]
                     %pi       %pi     5 %pi        7 %pi
```

We see that for $n > 0$, $a_n = 0$ for n even, and the non-zero coefficients have $n = 1, 3, 5, 7, ...$ Hence we only get a better approximation if we increase n_{max} by 2 each time.

```
(%i10) fs(nmax) := a0/2 + sum( a(m)*cos(m*%pi*x/2),m,1,nmax )$
(%i11) map( 'fs, [1,3] );
                %pi x                 3 %pi x            %pi x
           3 cos(-----)           cos(-------)      3 cos(-----)
                   2       3           2                   2       3
(%o11)     [------------- + -, - ------------- + ------------- + -]
               %pi          4        %pi             %pi          4
(%i12) qdraw( yr(-0.5,2),ex([f(x),fs(1),fs(3) ],x,-2,2) )$
(%i13) qdraw( yr(-0.5,2),ex([f(x),fs(11) ],x,-2,2) )$
```

The plot with the approximations **fs(1)** and **fs(3)** was drawn first.
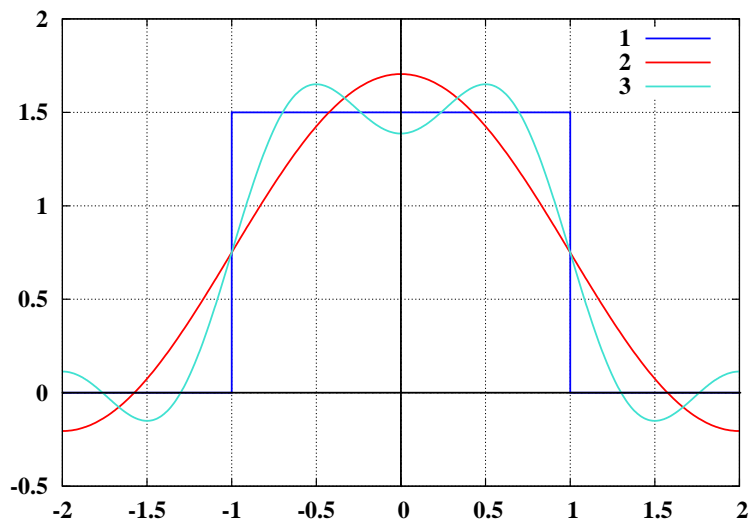
Figure 5: $n_{max} = 1, 3$ Approx. to Rectangular Pulse

Then a plot showing the **fs(11)** approximation:
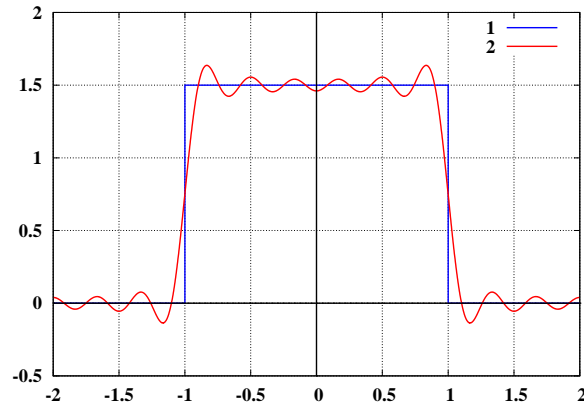
Figure 6: $\mathbf{nmax} = \mathbf{11}$ Approx. to Rectangular Pulse

10.2.7 Fourier Series Expansion of a Two Element Pulse

We compute the Fourier series expansion of a function which has the definition over the interval $(-10, 10)$ (and whose Fourier series expansion will be a function with period 20 for all **x**) given by: for $-10 \le x < -5$, $\mathbf{f} = 0$, and for $-5 \le x < 0$, $\mathbf{f} = -5$, and for $0 \le x \le 5$, $\mathbf{f} = 5$, and for $5 < x \le 10$, $\mathbf{f} = 0$. First we define such a function for our plots.

```
(%i1) f(x):= if x >= -5 and x < 0 then -5
             elseif x >= 0 and x <= 5 then 5 else 0$
(%i2) map('f,[-6,-5,-1,0,1,5,6]);
(%o2)                    [0, - 5, - 5, 5, 5, 5, 0]
```

and plot the function

```
(%i3) ( load(draw),load(qdraw) )$
             qdraw(...), qdensity(...), syntax: type qdraw();
(%i4) qdraw( yr(-8,8), ex1(f(x),x,-10,10,lw(5),lc(blue) ) )$
```

which looks like
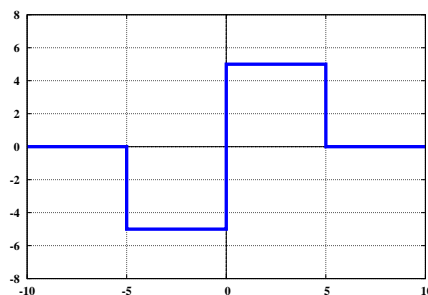
Figure 7: Two Element Pulse with Period 20

Since we have an odd function, only terms like $b_n \sin(n \pi x/10)$ will contribute to the expansion, but, just for practice, we find a_n too.

```
(%i5) a0 : (1/10)*( integrate( -5, x, -5, 0 ) +
                          integrate( 5, x, 0, 5 )  );
(%o5)                              0
(%i6) an : (1/10)*(integrate( -5*cos( n*%pi*x/10 ), x, -5, 0 ) +
                      integrate(5*cos(n*%pi*x/10), x, 0, 5 )  );
(%o6)                              0
(%i7) bn : ( (1/10)*(integrate( -5*sin(n*%pi*x/10), x, -5, 0 ) +
                integrate( 5*sin(n*%pi*x/10), x, 0, 5 )  ),
                ratsimp(%%) );
                                %pi n
                       10 cos(-----) - 10
                                  2
(%o7)                - -------------------
                             %pi n
(%i8) define( b(n), bn );
                                   %pi n
                          10 cos(-----) - 10
                                     2
(%o8)                  b(n) := - -------------------
                                    %pi n
(%i9) map('b,makelist(i,i,1,7));
                     10    10     10          2      10       10
(%o9)              [---, ---, -----, 0, ---, -----, -----]
                     %pi   %pi   3 %pi       %pi   3 %pi   7 %pi
(%i10) fs(nmax) :=  sum( b(m)*sin(m*%pi*x/10), m, 1, nmax )$
(%i11) map('fs,[1,2,3]);
                 %pi x           %pi x             %pi x
           10 sin(-----)   10 sin(-----)    10 sin(-----)
                    10              5                10
(%o11) [-------------, -------------- + -------------,
              %pi            %pi               %pi
                          3 %pi x           %pi x            %pi x
                   10 sin(-------)   10 sin(-----)    10 sin(-----)
                             10               5                10
                   --------------- + ------------- + -------------]
                        3 %pi            %pi              %pi
(%i12) qdraw(  xr(-15, 15), yr(-10, 10),
                    ex( [f(x), fs(1), fs(2) ], x, -10, 10 )  )$
```

The plot of **f(x)** with the two lowest approximations looks like
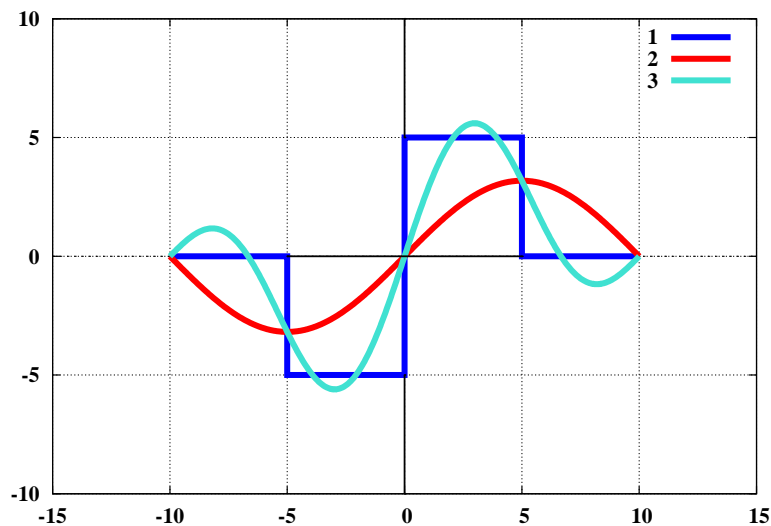
Figure 8: Two Lowest Order Approximations

The expansion **fs(11)** does pretty well as a rough approximation

```
(%i13) qdraw( xr(-15, 15), yr(-10, 10),
                ex(  [ f(x), fs(11) ], x, -10, 10 ) )$
```
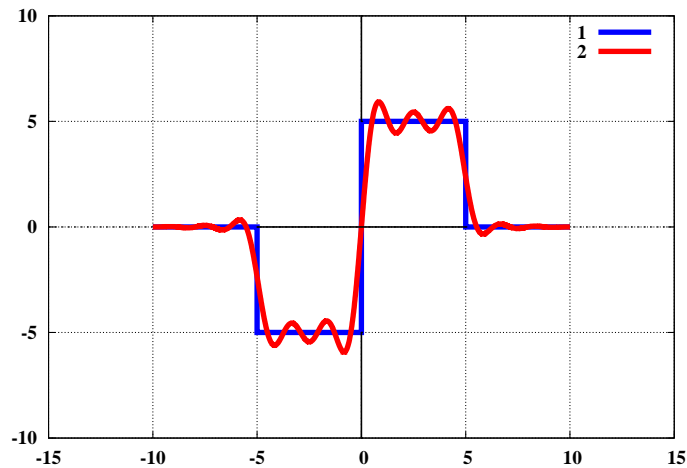
Figure 9: $\mathbf{nmax = 11}$ Approx. to Two Element Pulse

Whether you are working with a periodic function or not, the Fourier series expansion always represents a periodic function for all **x**. For this example, the period is **20**, and if we make a plot of **fs(11)** over the range **[-10, 30]**, we are including two periods.

```
(%i14) qdraw(yr(-10, 10),ex( fs(11), x, -10 , 30 ) )$
```

which looks like
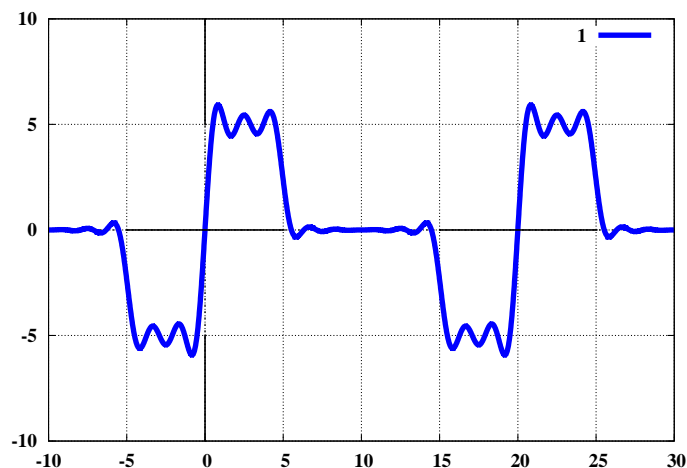
Figure 10: $\mathbf{nmax = 11}$ Approx. Drawn for Two Periods

10.2.8 Exponential Form of a Fourier Series Expansion

The exponential form of a Fourier series can be based on the completeness and orthogonality (in the Hermitian sense) of the set of exponential functions

$$\left\{ \exp\left(\frac{2\pi i n x}{b-a} \right) \right\} \qquad (n = 0, \pm 1, \pm 2, \cdots) \tag{10.8}$$

on the interval (a, b). Define the function $\phi_n(x)$ as

$$\phi_n(x) = \exp\left(\frac{-2\pi i n x}{b-a} \right). \tag{10.9}$$

This function has the properties

$$\phi_{-n}(x) = \phi_n(x)^*, \qquad \phi_n(x)\phi_n(x)^* = 1. \tag{10.10}$$

in which the asterisk indicates the complex conjugate.

The (Hermitian) orthogonality of the $\{\phi_n(x)\}$ over (a, b) is expressed by

$$\int_a^b \phi_n(x)\phi_m(x)^* \, dx = (b-a)\,\delta_{nm}, \tag{10.11}$$

in which the Kronecker delta symbol is

$$\delta_{nm} = \left\{ \begin{array}{ll} 1 & \text{for} \quad n = m \\ 0 & \text{for} \quad n \neq m. \end{array} \right\} \tag{10.12}$$

Eq.(10.11) is clearly true for $n = m$, using Eq.(10.10). For $m \neq n$, we can simplify this integral by using the exponential product law $e^A e^B = e^{A+B}$, letting $r = m - n \neq 0$, and changing the variable of integration $x \to y$ via $x = (b-a)y + a$. The resulting integral in terms of y will then be over the interval $(0, 1)$. The differential $dx \to (b-a)\,dy$ and $(b-a)$ comes outside the integral. Also, inside the exponential, $x/(b-a) \to a/(b-a) + y$, and we can take outside an exponential function of a constant. Thus the integral is proportional to $\int_0^1 \exp(2\pi i r y)\,dy$, which is proportional to $\exp(2\pi i r) - 1 = \exp(2\pi i)^r - 1 = 0$.

We now write some function of x as a linear combination of the $\phi_n(x)$ with coefficients C_n to be determined.

$$f(x) = \sum_{n=-\infty}^{\infty} C_n \phi_n(x). \tag{10.13}$$

To find the $\{C_n\}$, we multiply both sides of Eq.(10.13) by $\phi_m(x)^*\,dx$, integrate both sides over the interval (a, b), and use orthogonality, Eq.(10.11).

$$\int_a^b f(x)\phi_m(x)^*\,dx = \sum_{n=-\infty}^{\infty} C_n \int_a^b \phi_n(x)\phi_m(x)^*\,dx$$

$$= \sum_{n=-\infty}^{\infty} C_n (b-a)\,\delta_{nm}$$

$$= (b-a)\,C_m.$$

16

Hence we have for the coefficients

$$C_n = \frac{1}{(b-a)} \int_a^b f(x)\, \phi_n(x)^* \, dx.$$

(10.14)

Inserting these coefficients into Eq.(10.13) we can write

$$f(x) = \sum_{n=-\infty}^{\infty} C_n \phi_n(x)$$

$$= \sum_{-\infty}^{\infty} \exp\left(\frac{-2\pi i n x}{b-a}\right) \frac{1}{b-a} \int_a^b f(y) \exp\left(\frac{2\pi i n y}{b-a}\right) dy$$

$$= \frac{1}{b-a} \sum_{-\infty}^{\infty} \int_a^b f(y) \exp\left(\frac{2\pi i n (y-x)}{b-a}\right) dy.$$

We now separate out the $n = 0$ term and combine the $n = \pm m$ terms into a sum over the positive integers.

$$f(x) = \frac{1}{b-a} \int_a^b f(y)\, dy + \frac{1}{b-a} \sum_{n=1}^{\infty} \int_a^b f(y) \left[\exp\left(\frac{2\pi i n (y-x)}{b-a}\right) + \exp\left(\frac{-2\pi i n (y-x)}{b-a}\right)\right] dy$$

$$= \frac{1}{b-a} \int_a^b f(y)\, dy + \frac{2}{b-a} \sum_{n=1}^{\infty} \int_a^b f(y) \cos\left(\frac{2\pi n (y-x)}{b-a}\right) dy$$

Using the trig identity $\cos(A - B) = \cos A \cos B + \sin A \sin B$, we recover the trigonometric form of the Fourier series expansion of a function over the interval (a, b), which will represent a function which has period equal to $(b$ - $a)$ for all values of x.

$$f(x) = \frac{1}{2} a_0 + \sum_{n=1}^{\infty} \left[a_n \cos\left(\frac{2\pi n x}{b-a}\right) + b_n \sin\left(\frac{2\pi n x}{b-a}\right) \right]$$

(10.15)

The coefficients are given by the integrals

$$a_n = \frac{2}{b-a} \int_a^b f(y) \cos\left(\frac{2\pi n y}{b-a}\right) dy, \quad b_n = \frac{2}{b-a} \int_a^b f(y) \sin\left(\frac{2\pi n y}{b-a}\right) dy.$$

(10.16)

The expansion starts with the term

$$f(x) = \frac{1}{2} a_0 + \cdots = \frac{1}{b-a} \int_a^b f(y) + \cdots$$

(10.17)

which is the (unweighted) average of $f(x)$ over (a, b).

If we specialize to a function defined over the interval $(\textbf{-p, p})$, whose Fourier expansion will represent a function which has period $\textbf{2 p}$ for all $\textbf{x}$, the above results have the replacements $2/(b-a) \to 2/(2p) \to 1/p$, and the appropriate exspansion equations are

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

(10.18)

with coefficients

$$a_n = \frac{1}{p} \int_{-p}^{p} f(y) \cos\left(\frac{\pi n y}{p}\right) dy, \quad b_n = \frac{1}{p} \int_{-p}^{p} f(y) \sin\left(\frac{\pi n y}{p}\right) dy.$$

(10.19)

If we specialize further to $p = \pi$, the appropriate equations are

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}[a_n\cos(nx) + b_n\sin(nx)] \tag{10.20}$$

with coefficients

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi}f(y)\cos(ny)\,dy, \quad b_n = \frac{1}{\pi}\int_{-\pi}^{\pi}f(y)\sin(ny)\,dy. \tag{10.21}$$

10.3   Fourier Integral Transform Pairs

10.3.1   Fourier Cosine Integrals and fourintcos(..)

Given some function $f(x)$ defined for $x \geq 0$, we define the Fourier cosine transform of this function as

$$F_C(f,\omega) = \frac{2}{\pi}\int_0^{\infty}\cos(\omega x)\,f(x)\,dx \tag{10.22}$$

The given function $f(x)$ can then be written as an integral over positive values of ω:

$$f(x) = \int_0^{\infty}F_C(f,\omega)\,\cos(\omega x)\,d\omega \tag{10.23}$$

The two equations (10.22) and (10.23) are an example of a "Fourier transform pair", which include conventions about where to place the factor of $2/\pi$.

Here is a simple example. We let the function be $f(x) = \sin(x)\,e^{-x}$, defined for $x \geq 0$. Let the letter **w** be used to stand for ω. We first calculate the Fourier cosine transform $F_C(f,\omega)$ (we use the symbol **fcw** in our work here) using **integrate** and then show that the inverse integral over ω gives back the original function.

```
(%i1)  f:sin(x)*exp(-x)$
(%i2)  integrate(f*cos(w*x),x,0,inf);
                                              2
                         2            2 w
(%o2)                   ------   -  --------
                         4            4
                        w   + 4    2 w   + 8
(%i3)  fcw : (2/%pi)*ratsimp(%);
                              2
                          2 (w   - 2)
(%o3)                  -  ------------
                              4
                         %pi (w   + 4)
(%i4)  integrate(fcw*cos(w*x),w,0,inf);
Is   x   positive, negative, or zero?
p;
                              - x
(%o4)                       %e      sin(x)
```

We can also use the package **fourie.mac**, which we explored in the section on Fourier series. This package provides for the calculation of our Fourier cosine transform integral via the **fourintcos(expr,var)** function. The function **fourintcos(f,x)** returns an answer with the label a_z which contains the Fourier cosine integral $F_C(f, z)$, with the letter **z** being the package convention for what we called **w** (ω),

```
(%i5) load(fourie);
(%o5) C:/PROGRA~1/MAXIMA~3.1/share/maxima/5.18.1/share/calculus/fourie.mac
(%i6) fourintcos(f,x);
                                        2
                           2           z
                       2 (------ - ------)
                           4           4
                          z  + 4    z  + 4
(%t6)              a  = --------------------
                        z           %pi

(%o6)                       [%t6]
(%i7) az : ratsimp(rhs(%t6));
                              2
                         2 z   - 4
(%o7)                 - --------------
                              4
                        %pi z  + 4 %pi
(%i8) (2/%pi)*ratsimp(%pi*az/2);
                              2
                         2 (z   - 2)
(%o8)                 - ------------
                              4
                        %pi (z  + 4)
```

Thus **fourintcos(expr,var)** agrees with our definition of the Fourier cosine transform.

10.3.2 Fourier Sine Integrals and fourintsin(..)

Given some function $f(x)$ defined for $x \geq 0$, we define the Fourier sine transform of this function as

$$F_S(f, \omega) = \frac{2}{\pi} \int_0^\infty \sin(\omega\,x)\,f(x)\,dx \tag{10.24}$$

The given function $f(x)$ can then be written as an integral over positive values of ω:

$$f(x) = \int_0^\infty F_S(f, \omega)\,\sin(\omega\,x)\,d\omega \tag{10.25}$$

The two equations (10.24) and (10.25) are another "Fourier transform pair", which include conventions about where to place the factor of $2/\pi$.

Here is a simple example. We let the function be $\mathbf{f(x)} = \cos(\mathbf{x})\,\mathbf{e}^{-\mathbf{x}}$, defined for $\mathbf{x} \geq 0$. Let the letter $\mathbf{w}$ stand for ω. We first calculate the Fourier sine transform $\mathbf{F_S(f, \omega)}$ (we use the symbol **fsw** in our work here) using **integrate** and then show that the inverse integral over ω gives back the original function.

```
(%i1)  f:cos(x)*exp(-x)$
(%i2)  assume(w>0)$
(%i3)  integrate(f*sin(w*x),x,0,inf);
                                    3
                                   w
(%o3)                           ------
                                  4
                                 w  + 4
(%i4)  fsw : (2/%pi)*%;
                                     3
                                  2 w
(%o4)                          ------------
                                    4
                               %pi (w  + 4)
(%i5)  integrate(fsw*sin(w*x),w,0,inf);
Is   x   positive, negative, or zero?


p;
                                   - x
(%o5)                            %e    cos(x)
```

We can also use the package **fourie.mac**, which we used above. This package provides for the calculation of our Fourier sine transform integral via the **fourintsin(expr,var)** function. The function **fourintsin(f,x)** returns an answer with the label b_z which contains the Fourier sine integral $\mathbf{F_S(f, z)}$, with the letter $\mathbf{z}$ being the package convention for what we called $\mathbf{w}$ (ω).

```
(%i6)  load(fourie);
(%o6)  C:/PROGRA~1/MAXIMA~3.1/share/maxima/5.18.1/share/calculus/fourie.mac
(%i7)  facts();
(%o7)                              [w > 0]
(%i8)  (forget(w>0),facts());
(%o8)                                []
(%i9)  fourintsin(f,x);
                                        3
                                     2 z
(%t9)                       b   = ------------
                             z        4
                                  %pi (z  + 4)

(%o9)                              [%t9]
(%i10) bz : rhs(%t9);
                                     3
                                  2 z
(%o10)                         ------------
                                    4
                               %pi (z  + 4)
```

Thus **fourintsin(expr,var)** agrees with our definition of the Fourier sine transform.

10.3.3 Exponential Fourier Integrals and fourint

Given some function $f(x)$ defined for $-\infty < x < \infty$, we define the exponential Fourier transform of this function as

$$F_{Exp}(f, \omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{i\omega x} dx \tag{10.26}$$

The given function $f(x)$ can then be written as an integral over both positive and negative values of ω:

$$f(x) = \int_{-\infty}^{\infty} F_{Exp}(f, \omega) e^{-i\omega x} d\omega \tag{10.27}$$

The two equations (10.26) and (10.27) are another "Fourier transform pair", which include conventions about where to place the factor of 2π as well as which member has the minus sign in the exponent.

If the given function is even, $f(-x) = f(x)$, then the exponential Fourier transform has the symmetry

$$F_{Exp}(f, -\omega) = F_{Exp}(f, \omega) \tag{10.28}$$

and can be expressed in terms of the Fourier cosine transform:

$$F_{Exp}(f, \omega) = \frac{1}{2} F_C(f, \omega). \tag{10.29}$$

If the given function is odd, $f(-x) = -f(x)$, then the exponential Fourier transform has the symmetry

$$F_{Exp}(f, -\omega) = -F_{Exp}(f, \omega) \tag{10.30}$$

and can be expressed in terms of the Fourier sine transform:

$$F_{Exp}(f, \omega) = \frac{i}{2} F_S(f, \omega). \tag{10.31}$$

If the given function is neither even nor odd, the function can always be written as the sum of an even function $f_e(x)$ and an odd function $f_o(x)$:

$$f(x) \equiv f_e(x) + f_o(x) = \frac{1}{2}(f(x) + f(-x)) + \frac{1}{2}(f(x) - f(-x)) \tag{10.32}$$

and can be expressed in terms of the Fourier cosine transform of $f_e(x)$ and the Fourier sine transform of $f_o(x)$:

$$F_{Exp}(f, \omega) \equiv F_{Exp}(f_e + f_o, \omega) = \frac{1}{2} F_C(f_e, \omega) + \frac{i}{2} F_S(f_o, \omega) \tag{10.33}$$

The **fourie.mac** package function **fourint(expr,var)** displays the non-zero coefficients a_z and/or the b_z, in terms of which we can write down the value of $F_{Exp}(f, z)$ (with our conventions):

$$F_{Exp}(f, z) = \frac{1}{2} a_z + \frac{i}{2} b_z \tag{10.34}$$

10.3.4 Example 1: Even Function

We calculate here the exponential fourier transform $F_{Exp}(f, \omega)$ when the function is $f(x) = \cos(x) e^{-|x|}$ which is an even function $f(-x) = f(x)$ and is defined for $-\infty < x < \infty$.

We first use **integrate**, by separating the integral into two pieces, **i1** is the integral over $(-\infty, 0)$ and **i2** is the integral over $(0, \infty)$ (we ignore the overall factor of $1/(2\pi)$ initially).

```
(%i1)  assume(w>0)$
(%i2)  i1:integrate(exp(%i*w*x)*cos(x)*exp(x),x,minf,0);
                           2            3
                          w  + 2    %i w
(%o2)                     ------ - ------
                           4         4
                          w  + 4    w  + 4
(%i3)  i2:integrate(exp(%i*w*x)*cos(x)*exp(-x),x,0,inf);
                            3        2
                          %i w      w  + 2
(%o3)                     ------ + ------
                           4         4
                          w  + 4    w  + 4
(%i4)  i12:ratsimp(i1+i2);
                               2
                            2 w  + 4
(%o4)                       --------
                              4
                             w  + 4
(%i5)  2*ratsimp(i12/2);
                                2
                             2 (w  + 2)
(%o5)                        ----------
                                4
                               w  + 4
(%i6)  iexp:%/(2*%pi);
                                 2
                               w  + 2
(%o6)                       ------------
                                 4
                            %pi (w  + 4)
```

Hence, direct use of **integrate** provides the exponential Fourier transform

$$\mathbf{F_{Exp}}(\cos(x)\exp(-|x|),\,\omega) = \frac{\omega^2 + 2}{\pi\,(\omega^4 + 4)} \tag{10.35}$$

We can use **integrate** to calculate the inverse Fourier exponential transform, recovering our original even function of x.

```
(%i7)  integrate(exp(-%i*w*x)*iexp,w,minf,inf);
Is   x   positive, negative, or zero?
p;
                                 - x
(%o7)                         %e     cos(x)
(%i8)  integrate(exp(-%i*w*x)*iexp,w,minf,inf);
Is   x   positive, negative, or zero?
n;
                                  x
(%o8)                         %e   cos(x)
```

Next we use **integrate** again directly to calculate the Fourier cosine transform of the given even function, now considered as defined for $x \geq 0$ and confirm that the required exponential Fourier transform in this case is correctly given by $\frac{1}{2}\mathbf{F_C}(\mathbf{f}, \omega)$.

```
(%i9) i3:ratsimp(integrate(cos(x)*exp(-x)*cos(w*x),x,0,inf));
                               2
                              w  + 2
(%o9)                         ------
                               4
                              w  + 4
(%i10) i3:(2/%pi)*i3;
                                2
                             2 (w  + 2)
(%o10)                       ------------
                                  4
                             %pi (w  + 4)
```

Output **%o10** is the value of $\mathbf{F_C}(\mathbf{f}, \omega)$, and one half of that value is the required exponential Fourier transform.

Next we use **fourint(expr,var)** with this even function.

```
(%i1) load(fourie);
(%o1) C:/PROGRA~1/MAXIMA~3.1/share/maxima/5.18.1/share/calculus/fourie.mac
(%i2) fourint(cos(x)*exp(-abs(x)),x);
                               2
                              z          2
                         2 (------ + ------)
                             4          4
                            z  + 4   z  + 4
(%t2)                 a  = -------------------
                       z          %pi

(%t3)                        b  = 0
                              z

(%o3)                        [%t2, %t3]
(%i4) ratsimp(rhs(%t2));
                               2
                            2 z  + 4
(%o4)                      --------------
                               4
                           %pi z  + 4 %pi
(%i5) (2/%pi)*ratsimp(%pi*%/2);
                               2
                            2 (z  + 2)
(%o5)                       ------------
                                 4
                            %pi (z  + 4)
```

which confirms that for an even function, the required exponential Fourier transform is correctly given (using **fourint**) by $\mathbf{F_{Exp}}(\mathbf{f}, \mathbf{z}) = \mathbf{a_z}/\mathbf{2}$.

10.3.5 Example 2: Odd Function

We calculate here the exponential fourier transform $\mathbf{F_{Exp}}(f, \omega)$ when the function is $f(x) = \sin(x)\,e^{-|x|}$ which is an odd function $f(-x) = -f(x)$ and is defined for $-\infty < x < \infty$.

We first use **integrate**, by separating the integral into two pieces, **i1** is the integral over $(-\infty, 0)$ and **i2** is the integral over $(0, \infty)$ (we ignore the overall factor of $1/(2\,\pi)$ initially).

```
(%i1) ( assume(w>0), facts());
(%o1)                              [w > 0]
(%i2) i1:ratsimp(integrate(exp(%i*w*x)*sin(x)*exp(x),x,minf,0));
                              2
                             w  + 2 %i w - 2
(%o2)                        ----------------
                                   4
                                  w  + 4
(%i3) i2:ratsimp(integrate(exp(%i*w*x)*sin(x)*exp(-x),x,0,inf));
                              2
                             w  - 2 %i w - 2
(%o3)                      - ----------------
                                   4
                                  w  + 4
(%i4) iexp:ratsimp(i1+i2)/(2*%pi);
                                 2 %i w
(%o4)                        ------------
                                   4
                             %pi (w  + 4)
(%i5) facts();
(%o5)                              [w > 0]
```

Hence, direct use of **integrate** provides the exponential Fourier transform

$$\mathbf{F_{Exp}}(\sin(x)\exp(-|x|),\ \omega) = \frac{2\,\mathrm{i}\,\omega}{\pi\,(\omega^4 + 4)} \tag{10.36}$$

We can use **integrate** to calculate the inverse Fourier exponential transform, recovering our original odd function of x.

```
(%i6) integrate(exp(-%i*w*x)*iexp,w,minf,inf);
Is   x  positive, negative, or zero?
p;
                                    - x
(%o6)                             %e    sin(x)
(%i7) integrate(exp(-%i*w*x)*iexp,w,minf,inf);
Is   x  positive, negative, or zero?
n;
                                    x
(%o7)                             %e  sin(x)
(%i8) facts();
(%o8)                              [w > 0]
```

Next we use **integrate** again directly to calculate the Fourier sine transform of the given odd function, now considered as defined for $x \geq 0$ and confirm that the required exponential Fourier transform in this case is correctly given by $\frac{i}{2} F_S(f, \omega)$.

```
(%i9)  ratsimp(integrate(sin(x)*exp(-x)*sin(w*x),x,0,inf));
                              2 w
(%o9)                        ------
                              4
                             w  + 4
(%i10)  (2/%pi)*%;
                              4 w
(%o10)                    ------------
                              4
                          %pi (w  + 4)
```

Output **%o10** is the value of $F_S(f, \omega)$, and multiplying by $i/2$ yields the required exponential Fourier transform.

Next we use **fourint(expr,var)** with this odd function.

```
(%i11)  load(fourie);
(%o11)  C:/PROGRA~1/MAXIMA~3.1/share/maxima/5.18.1/share/calculus/fourie.mac
(%i12)  fourint(sin(x)*exp(-abs(x)),x);
(%t12)                           a   = 0
                                  z

                                     4 z
(%t13)                        b   = ------------
                               z        4
                                    %pi (z  + 4)

(%o13)                          [%t12, %t13]
```

which confirms that for an odd function, the required exponential Fourier transform is correctly given (using **fourint**) by $F_{Exp}(f, z) = (i \, b_z)/2$.

10.3.6 Example 3: A Function Which is Neither Even nor Odd

We calculate here the exponential fourier transform $\mathbf{F_{Exp}}(f, \omega)$ when the function is $f(x) = \cos^2(x - 1)\,e^{-|x|}$ which is defined for $-\infty < x < \infty$ and is neither even nor odd.

We first use **integrate**, by separating the integral into two pieces. **i1** is the integral over $(-\infty, 0)$ and **i2** is the integral over $(0, \infty)$ (we ignore the overall factor of $1/(2\,\pi)$ initially).

```
(%i1) ( assume(w>0), facts());
(%o1)                               [w > 0]
(%i2) i1:ratsimp(integrate(exp(%i*w*x)*cos(x-1)^2*exp(x),x,minf,0));
                       5                              4
(%o2) - ((%i cos(2) + %i) w  + (- 2 sin(2) - cos(2) - 1) w

                                  3                           2
 + (- 4 %i sin(2) - 2 %i cos(2) - 6 %i) w  + (8 sin(2) - 6 cos(2) + 6) w
 + (- 4 %i sin(2) - 3 %i cos(2) + 25 %i) w + 10 sin(2) - 5 cos(2) - 25)
      6      4      2
/(2 w  - 10 w  + 38 w  + 50)
(%i3) i2:ratsimp(integrate(exp(%i*w*x)*cos(x-1)^2*exp(-x),x,0,inf));
                       5                            4
(%o3) ((%i cos(2) + %i) w  + (- 2 sin(2) + cos(2) + 1) w

                                3                           2
 + (4 %i sin(2) - 2 %i cos(2) - 6 %i) w  + (8 sin(2) + 6 cos(2) - 6) w
 + (4 %i sin(2) - 3 %i cos(2) + 25 %i) w + 10 sin(2) + 5 cos(2) + 25)
      6      4      2
/(2 w  - 10 w  + 38 w  + 50)
(%i4) i12 : rectform(ratsimp(i1 + i2));
                 4                2
      (cos(2) + 1) w  + (6 cos(2) - 6) w  + 5 cos(2) + 25
(%o4) ----------------------------------------------------
                    6      4      2
                   w  - 5 w  + 19 w  + 25

                                                       3
                              %i (4 sin(2) w  + 4 sin(2) w)
                          + ----------------------------
                                 6      4      2
                                w  - 5 w  + 19 w  + 25
(%i5) i12 : map('ratsimp,i12);
                 4                2
      (cos(2) + 1) w  + (6 cos(2) - 6) w  + 5 cos(2) + 25   4 %i sin(2) w
(%o5) ---------------------------------------------------- + -------------
                    6      4      2                           4      2
                   w  - 5 w  + 19 w  + 25                    w  - 6 w  + 25
(%i6) i12 : realpart(i12)/(2*%pi) + %i*imagpart(i12)/(2*%pi);
                 4                2
      (cos(2) + 1) w  + (6 cos(2) - 6) w  + 5 cos(2) + 25
(%o6) ----------------------------------------------------
                        6      4      2
               2 %pi (w  - 5 w  + 19 w  + 25)

                                                2 %i sin(2) w
                                          + --------------------
                                                 4      2
                                            %pi (w  - 6 w  + 25)
```

The final **i12** is the desired exponential Fourier transform.

We can now calculate the inverse Fourier transform.

```
(%i7) integrate(exp(-%i*w*x)*i12,w,minf,inf);
Is  x  positive, negative, or zero?
p;
                      - x
                    %e    (sin(2) sin(2 x) + cos(2) cos(2 x) + 1)
(%o7)               -------------------------------------------
                                          2
```

As an exercise in the manipulation of trigonometric functions, we now go back and forth between the coefficient of `%e^(-x)` in this result and our starting function (for positive x).

```
(%i8) cos(x-1)^2;
                                    2
(%o8)                          cos (x - 1)
(%i9) trigreduce(%);
                            cos(2 (x - 1)) + 1
(%o9)                       ------------------
                                    2
(%i10) ratsimp(%);
                            cos(2 x - 2) + 1
(%o10)                      ----------------
                                   2
(%i11) trigexpand(%);
                    sin(2) sin(2 x) + cos(2) cos(2 x) + 1
(%o11)              -------------------------------------
                                      2
```

which gets us from the original coefficient to that returned by our inverse transform integral. Now we go in the opposite direction:

```
(%i12) trigreduce(%);
                            cos(2 x - 2) + 1
(%o12)                      ----------------
                                   2
(%i13) factor(%);
                           cos(2 (x - 1)) + 1
(%o13)                     ------------------
                                   2
(%i14) trigexpand(%);
                          2              2
                    - sin (x - 1) + cos (x - 1) + 1
(%o14)              -------------------------------
                                   2
(%i15) trigsimp(%);
                                 2
(%o15)                        cos (x - 1)
```

which returns us from the transform integral coefficient to our original coefficient. Hence we have verified the correctness of the exponential Fourier transform for this case.

Next we use **fourint(expr,var)** with this function which is neither even nor odd.

```
(%i16) load(fourie);
(%o16) C:/PROGRA~1/MAXIMA~3.1/share/maxima/5.18.1/share/calculus/fourie.mac
(%i17) fourint(cos(x-1)^2*exp(-abs(x)),x);
                         4                 2
              (cos(2) + 1) z  + (6 cos(2) - 6) z  + 5 cos(2) + 25
(%t17)     a  = ---------------------------------------------------
            z                        6      4       2
                            %pi (z  - 5 z  + 19 z  + 25)


                                   4 sin(2) z
(%t18)                        b  = --------------------
                               z        4      2
                                   %pi (z  - 6 z  + 25)


(%o18)                             [%t17, %t18]
(%i19) az : rhs(%t17);
                         4                 2
              (cos(2) + 1) z  + (6 cos(2) - 6) z  + 5 cos(2) + 25
(%o19)        ---------------------------------------------------
                               6      4       2
                       %pi (z  - 5 z  + 19 z  + 25)
(%i20) bz : rhs(%t18);
                              4 sin(2) z
(%o20)                        --------------------
                                   4      2
                              %pi (z  - 6 z  + 25)
(%i21) iexp_f : az/2 + %i*bz/2;
                    4                 2
         (cos(2) + 1) z  + (6 cos(2) - 6) z  + 5 cos(2) + 25
(%o21) -----------------------------------------------------
                          6      4       2
                 2 %pi (z  - 5 z  + 19 z  + 25)
                                                           2 %i sin(2) z
                                                       + --------------------
                                                              4      2
                                                         %pi (z  - 6 z  + 25)
```

To compare this with our result using **integrate** we need to replace **z** by **w**:

```
(%i22) subst(z=w,iexp_f) - i12;
(%o22)                                 0
```

which shows that, except for notation, the results are the same. We have confirmed that for a general function, the required exponential Fourier transform is correctly given (using **fourint**) by $F_{Exp}(f, z) = a_z/2 + (i\, b_z)/2$.

10.3.7 Dirac Delta Function $\delta(x)$

It is conventional to define the Dirac delta function (unit impulse function) $\delta(x)$ by

$$\delta(-x) = \delta(x) \quad \text{even function} \tag{10.37}$$

$$\delta(x) = 0 \quad \text{for} \quad x \neq 0 \tag{10.38}$$

$$\int_{-a}^{b} \delta(x)\, dx = 1 \quad \text{for} \quad a, b > 0 \tag{10.39}$$

28

These equations imply

$$\delta(\mathbf{y} - \mathbf{x}) = \delta(\mathbf{x} - \mathbf{y}) \tag{10.40}$$

and

$$\int \mathbf{f}(\mathbf{y})\,\delta(\mathbf{y} - \mathbf{x})\,d\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{10.41}$$

for any well behaved function $\mathbf{f}(\mathbf{x})$ (we are more careful below), provided the range of integration includes the point $\mathbf{x}$. For since $\delta(\mathbf{y} - \mathbf{x})$ is zero except at the single point $\mathbf{y} = \mathbf{x}$, we can evaluate $\mathbf{f}(\mathbf{y})$ at that single point and take the result outside the integral. The resulting integral which is left is $\int \delta(\mathbf{y} - \mathbf{x})\,d\mathbf{y}$, and the change of variables $\mathbf{y} \to \mathbf{t} = \mathbf{y} - \mathbf{x}$ for fixed $\mathbf{x}$ yields the integral $\int \delta(\mathbf{t})\,d\mathbf{t} = 1$.

Now let $\mathbf{c}$ be a positive constant (independent of the integration variable $\mathbf{x}$). In the integral $\int_{-\mathbf{a}}^{\mathbf{b}} \delta(\mathbf{c}\,\mathbf{x})\,d\mathbf{x}$, we change variables, $\mathbf{x} \to \mathbf{y} = \mathbf{c}\,\mathbf{x}$ so $d\mathbf{x} = d\mathbf{y}/\mathbf{c}$.

$$\int_{-\mathbf{a}}^{\mathbf{b}} \delta(\mathbf{c}\,\mathbf{x})\,d\mathbf{x} = \frac{1}{\mathbf{c}} \int_{-\mathbf{c}\,\mathbf{a}}^{\mathbf{c}\,\mathbf{b}} \delta(\mathbf{y})\,d\mathbf{y} = \frac{1}{\mathbf{c}} \tag{10.42}$$

since $(\mathbf{c}\,\mathbf{a}) > 0$ and $(\mathbf{c}\,\mathbf{b}) > 0$.

If, on the other hand, $\mathbf{c}$ is a negative constant, $\mathbf{c} = -|\mathbf{c}|$, and we make the same change of variables,

$$\int_{-\mathbf{a}}^{\mathbf{b}} \delta(\mathbf{c}\,\mathbf{x})\,d\mathbf{x} = \frac{1}{\mathbf{c}} \int_{-\mathbf{c}\,\mathbf{a}}^{\mathbf{c}\,\mathbf{b}} \delta(\mathbf{y})\,d\mathbf{y} = \frac{1}{\mathbf{c}} \int_{|\mathbf{c}|\,\mathbf{a}}^{-|\mathbf{c}|\,\mathbf{b}} \delta(\mathbf{y})\,d\mathbf{y} = -\frac{1}{\mathbf{c}} \int_{-|\mathbf{c}|\,\mathbf{b}}^{|\mathbf{c}|\,\mathbf{a}} \delta(\mathbf{y})\,d\mathbf{y} = \frac{1}{|\mathbf{c}|} \tag{10.43}$$

Evidently, we can always write (since both $\mathbf{x}$ and $\mathbf{y}$ are dummy integration variables

$$\int \delta(\mathbf{c}\,\mathbf{x})\,d\mathbf{x} = \int \frac{\delta(\mathbf{x})}{|\mathbf{c}|}\,d\mathbf{x} \tag{10.44}$$

or simply (with the understanding that this kind of relation only makes sense inside an integral)

$$\delta(\mathbf{c}\,\mathbf{x}) = \frac{\delta(\mathbf{x})}{|\mathbf{c}|} \tag{10.45}$$

We can find a sometimes useful representation of the Dirac delta function by using the exponential Fourier transform pair Eqs.(10.26) and (10.27). If we use $\mathbf{y}$ as the (dummy) variable of integration in Eq.(10.26), and insert into Eq.(10.27), we get (interchanging the order of integration)

$$\begin{aligned}
\mathbf{f}(\mathbf{x}) &= \int_{-\infty}^{\infty} \left[\frac{1}{2\,\pi} \int_{-\infty}^{\infty} e^{i\,\omega\,\mathbf{y}}\,\mathbf{f}(\mathbf{y})\,d\mathbf{y} \right] e^{-i\,\omega\,\mathbf{x}}\,d\omega \\
&= \int_{-\infty}^{\infty} \mathbf{f}(\mathbf{y}) \left[\frac{1}{2\,\pi} \int_{-\infty}^{\infty} e^{i\,\omega(\mathbf{y}-\mathbf{x})}\,d\omega \right] d\mathbf{y} \\
&= \int_{-\infty}^{\infty} \mathbf{f}(\mathbf{y})\,\delta(\mathbf{y} - \mathbf{x})\,d\mathbf{y}
\end{aligned}$$

Hence we can write

$$\delta(\mathbf{x}) = \frac{1}{2\,\pi} \int_{-\infty}^{\infty} e^{i\,\omega\,\mathbf{x}}\,d\omega \tag{10.46}$$

One use of such a representation of the one dimensional Dirac delta function is to derive in a different way the result of Eq.(10.45). From Eq.(10.46) we have

$$\delta(\mathbf{c}\,\mathbf{x}) = \frac{1}{2\,\pi} \int_{-\infty}^{\infty} e^{i\,\omega\,\mathbf{c}\,\mathbf{x}}\,d\omega \tag{10.47}$$

29

Again, suppose c is a constant and $c > 0$. Then if we change variables $\omega \to y = \omega\, c$, we have $d\omega = dy/c$. When $\omega = -\infty, y = -\infty$. When $\omega = \infty, y = \infty$. Hence

$$\delta(c\, x) = \frac{1}{c} \left(\frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\, y\, x}\, dy \right) = \frac{1}{c}\, \delta(x) \tag{10.48}$$

If, on the other hand, $c < 0$, $c = -|c|$ in Eq.(10.47), and we make the same change of variables, when $\omega = -\infty$, $y = \infty$, and when $\omega = \infty$, $y = -\infty$. Hence, for this case,

$$\delta(c\, x) = \frac{1}{c} \left(\frac{1}{2\pi} \int_{\infty}^{-\infty} e^{i\, y\, x}\, dy \right) = \frac{1}{c} \left(-\frac{1}{2\pi} \int_{-\infty}^{\infty} e^{i\, y\, x}\, dy \right) = \frac{1}{|c|}\, \delta(x). \tag{10.49}$$

Combining the two cases leads us again to Eq.(10.45).

Although the representation Eq.(10.46) can be useful for formal properties of the Dirac delta function, for actual calculation of integrals one is safer using a limiting form of that result, or even simpler versions, as we will see. If we replace the infinite limits in Eq.(10.46) by finite limits, we can work with the representation

$$\delta(x) = \lim_{L \to \infty} d_L(x) \tag{10.50}$$

where

$$d_L(x) = \frac{1}{2\pi} \int_{-L}^{L} e^{i\, x\, y}\, dy = \frac{\sin(x\, L)}{\pi\, x} \tag{10.51}$$

which can then be used as

$$\int f(x)\, \delta(x)\, dx = \lim_{L \to \infty} \int f(x)\, d_L(x)\, dx = \lim_{L \to \infty} \int f(x)\, \frac{\sin(x\, L)}{\pi\, x}. \tag{10.52}$$

However, this will only be useful if the resulting integrals can be done.

An easier approach to the careful use of the Dirac delta function for the calculation of integrals is to create a mathematically simple model of a function of x which has the required properties and whose use will result in integrals which can easily be done. Mathematically rigorous treatments (see the Theory of Distributions) of the Dirac delta function justify the use of such models .

Here we follow the approach of William E. Boyce and Richard C. DiPrima, in their textbook "Elementary Differential Equations and Boundary Value Problems", Fifth Edition, John Wiley & Sons, Inc, New York, 1992, Section 6.5, "Impulse Functions".

Let

$$d_\epsilon(x) = \left\{ \begin{array}{cc} 1/(2\,\epsilon) & \text{for} \quad -\epsilon < x < \epsilon \\ 0 & \text{for} \quad x \le -\epsilon \quad \text{or} \quad x \ge \epsilon \end{array} \right\} \tag{10.53}$$

which defines a rectangular pulse with base length $2\,\epsilon$, height $1/(2\,\epsilon)$ with area "under the curve" equal to 1. This is an even function of x centered on $x = 0$ which gets narrower and taller as $\epsilon \to 0$.

Then

$$d_\epsilon(y - x) = \left\{ \begin{array}{cc} 1/(2\,\epsilon) & \text{for} \quad (x - \epsilon) < y < (x + \epsilon) \\ 0 & \text{for} \quad y \le (x - \epsilon) \quad \text{or} \quad y \ge (x + \epsilon) \end{array} \right\} \tag{10.54}$$

This model can then be used in the form (with the range of integration assumed to include the point $\mathbf{y} = \mathbf{x}$):

$$\int \mathbf{f}(\mathbf{y})\,\boldsymbol{\delta}(\mathbf{y} - \mathbf{x})\,d\mathbf{y} = \lim_{\epsilon \to 0} \int \mathbf{f}(\mathbf{y})\,d_\epsilon(\mathbf{y} - \mathbf{x})\,d\mathbf{y} = \lim_{\epsilon \to 0} \frac{1}{2\,\epsilon} \int_{\mathbf{x}-\epsilon}^{\mathbf{x}+\epsilon} \mathbf{f}(\mathbf{y})\,d\mathbf{y}. \tag{10.55}$$

The integral should be done before taking the limits in this safer approach.

However, if $\mathbf{f}(\mathbf{y})$ is a continuous function and possesses a derivative in a complete neighborhood of $\mathbf{x}$, we can use the mean value theorem of calculus to write this limit as

$$\lim_{\epsilon \to 0} \frac{1}{2\,\epsilon}\,\mathbf{f}(\hat{\mathbf{y}})\,(2\,\epsilon) = \lim_{\epsilon \to 0} \mathbf{f}(\hat{\mathbf{y}})). \tag{10.56}$$

where $(\mathbf{x} - \epsilon) < \hat{\mathbf{y}} < (\mathbf{x} + \epsilon)$ As $\epsilon \to 0$, $\hat{\mathbf{y}} \to \mathbf{x}$, and $\mathbf{f}(\hat{\mathbf{y}}) \to \mathbf{f}(\mathbf{x})$, which recovers Eq. (10.41), and which we repeat here for emphasis.

$$\int \mathbf{f}(\mathbf{y})\,\boldsymbol{\delta}(\mathbf{y} - \mathbf{x})\,d\mathbf{y} = \mathbf{f}(\mathbf{x}) \tag{10.57}$$

for any function $\mathbf{f}(\mathbf{x})$ which is continuous and possesses a derivative in a complete neighborhood of $\mathbf{x}$, provided the range of integration includes the point $\mathbf{x}$.

10.3.8 Laplace Transform of the Delta Function Using a Limit Method

Use of Eq. (10.57) allows us to immediately write down the Laplace transform of $\boldsymbol{\delta}(\mathbf{t} - \mathbf{t_0})$ for $\mathbf{t_0} > 0$, the integral $\int_0^\infty \mathbf{e}^{-s\mathbf{t}}\,\boldsymbol{\delta}(\mathbf{t} - \mathbf{t_0})\,d\mathbf{t}$, since the function $\mathbf{e}^{-s\mathbf{t}}$ which multiplies the delta function is continuous and possesses a derivative for all values of $\mathbf{t}$.

However, as a consistency check, let's also use the limit method just discussed to derive this result.

$$\mathcal{L}\{\boldsymbol{\delta}(\mathbf{t} - \mathbf{t_0})\} \equiv \int_0^\infty \mathbf{e}^{-s\mathbf{t}}\,\boldsymbol{\delta}(\mathbf{t} - \mathbf{t_0})\,d\mathbf{t} = \mathbf{e}^{-s\,\mathbf{t_0}} \tag{10.58}$$

provided s and $\mathbf{t_0}$ are positive.

```
(%i1) assume(s>0,e>0,t0>0)$
(%i2) i1: integrate(exp(-s*t),t,t0-e,t0+e)/(2*e);
                    e s - s t0       - s t0 - e s
                  %e                %e
                  ------------   -  --------------
                       s                  s
(%o2)             ------------------------------
                              2 e
(%i3) limit(i1,e,0,plus);
                            - s t0
(%o3)                     %e
```

As a bonus, the limit of Eq.(10.58) as $\mathbf{t_0} \to \mathbf{0}^+$ then gives us the Laplace transform of $\boldsymbol{\delta}(\mathbf{t})$.

$$\mathcal{L}\{\boldsymbol{\delta}(\mathbf{t})\} \equiv \int_0^\infty \mathbf{e}^{-s\mathbf{t}}\,\boldsymbol{\delta}(\mathbf{t})\,d\mathbf{t} = 1 \tag{10.59}$$

10.4   Laplace Transform Integrals

10.4.1   Laplace Transform Integrals: laplace(..), specint(..)

A subgroup of definite integrals is so useful in applications that Maxima has functions which calculate the Laplace transform (**laplace** and **specint**) of a given expression, as well as the inverse Laplace transform (**ilt**).

laplace(expr, t, s) calculates the integral

$$\int_0^\infty e^{-st} f(t) \, dt = \mathcal{L}\{f(t)\} = F(s) \tag{10.60}$$

where $f(t)$ stands for the **expr** argument (here assumed to depend explicitly on t) of **laplace**. **laplace** assumes that s is a very large positive number. If there are convergence issues of the integral, one allows s to be a complex number $s = x + iy$ and the assumption is made that the integral converges to the right of some line $\mathrm{Re}(s) = \mathrm{const}$ in the complex s plane.

10.4.2   Comparison of laplace and specint

specint(exp(- s*t) * expr, t) is equivalent to **laplace(expr, t, s)** in its effect, but has been specially designed to handle special functions with care. You need to prepare **specint** with **assume** statement(s) that s is both positive and larger than other positive parameters in **expr**.

$\mathcal{L}\{1\} = 1/s$

We start the following comparison of **laplace** and **specint** with an **assume** statement (needed by **specint** alone).

```
(%i1)  assume( s>0, a > 0, b > 0, s > a, s > b )$
(%i2)  laplace(1,t,s);
                                    1
(%o2)                               -
                                    s
(%i3)  specint(exp(-s*t),t);
                                    1
(%o3)                               -
                                    s
(%i4)  ilt(%,s,t);
(%o4)                               1
```

$\mathcal{L}\{t\} = 1/s^2$

```
(%i5)  laplace(t,t,s);
                                    1
(%o5)                               --
                                     2
                                    s
(%i6)  specint(exp(-s*t)*t,t);
                                    1
(%o6)                               --
                                     2
                                    s
(%i7)  ilt(%,s,t);
(%o7)                               t
```

$\mathcal{L}\{e^{a\,t}\} = 1/(s-a)$

```
(%i8) laplace(exp(a*t),t,s);
                                    1
(%o8)                             -----
                                  s - a
(%i9) specint(exp(-s*t)*exp(a*t),t);
                                    1
(%o9)                             -----
                                  s - a
```

$\mathcal{L}\{\sin(a\,t)/a\} = (s^2 + a^2)^{-1}$

```
(%i10) laplace(sin(a*t)/a,t,s);
                                    1
(%o10)                            -------
                                   2    2
                                  s  + a
(%i11) (specint(exp(-s*t)*sin(a*t)/a,t),ratsimp(%%) );
                                    1
(%o11)                            -------
                                   2    2
                                  s  + a
```

$\mathcal{L}\{\cos(a\,t)\} = s\,(s^2 + a^2)^{-1}$

```
(%i12) laplace(cos(a*t),t,s);
                                    s
(%o12)                            -------
                                   2    2
                                  s  + a
(%i13) (specint(exp(-s*t)*cos(a*t),t ), ratsimp(%%) );
                                    s
(%o13)                            -------
                                   2    2
                                  s  + a
```

$\mathcal{L}\{t\sin(a\,t)/(2\,a)\} = s\,(s^2 + a^2)^{-2}$

```
(%i14) laplace(sin(a*t)*t/(2*a),t,s);
                                     s
(%o14)                          ----------
                                  2     2 2
                                 (s  + a )
(%i15) (specint(exp(-s*t)*sin(a*t)*t/(2*a),t ),ratsimp(%%) );
                                      s
(%o15)                       ------------------
                              4      2 2      4
                             s  + 2 a  s  + a
(%i16) map('factorsum,%);
                                     s
(%o16)                          ----------
                                  2     2 2
                                 (s  + a )
```

$$\mathcal{L}\{e^{a\,t}\cos(b\,t)\} = (s - a)\,((s - a)^2 + b^2)^{-1}$$

```
(%i17) laplace(exp(a*t)*cos(b*t),t,s);
                              s - a
(%o17)                  --------------------
                         2             2    2
                        s  - 2 a s + b  + a
(%i18) map('factorsum,%);
                              s - a
(%o18)                  --------------
                              2    2
                        (s - a)  + b
(%i19) (specint(exp(-s*t)*exp(a*t)*cos(b*t),t ),ratsimp(%%) );
                              s - a
(%o19)                  --------------------
                         2             2    2
                        s  - 2 a s + b  + a
(%i20) map('factorsum,%);
                              s - a
(%o20)                  --------------
                              2    2
                        (s - a)  + b
```

$$\mathcal{L}\{\sqrt{t}\ \mathbf{bessel\_j}\,(1, 2\,\sqrt{a\,t})\} = \sqrt{a}\,s^{-2}\,e^{-a/s}$$

```
(%i21) expr : t^(1/2) * bessel_j(1, 2 * a^(1/2) * t^(1/2));
(%o21)              bessel_j(1, 2 sqrt(a) sqrt(t)) sqrt(t)
(%i22) laplace(expr,t,s);
                                      - a/s
                            sqrt(a) %e
(%o22)                      ---------------
                                   2
                                  s
(%i23) specint(exp(-s*t)*expr,t);
                                      - a/s
                            sqrt(a) %e
(%o23)                      ---------------
                                   2
                                  s
(%i24) ilt(%,s,t);
                                   - a/s
                         sqrt(a) %e
(%o24)              ilt(---------------, s, t)
                               2
                              s
```

We find in the above example that **ilt** cannot find the inverse Laplace transform.

$$\mathcal{L}\{\mathrm{erf}(\sqrt{t})\} = (\,s\,\sqrt{s+1}\,)^{-1}$$

Here again **ilt** fails.

```
(%i25) assume(s>0)$
(%i26) laplace(erf(sqrt(t)),t,s);
                                   1
(%o26)                      -------------
                            s sqrt(s + 1)
(%i27) specint(exp(-s*t)*erf(sqrt(t)),t);
                                   1
(%o27)                      ----------------
                                1        3/2
                            sqrt(- + 1) s
                                 s
(%i28) radcan(%);
                                   1
(%o28)                      -------------
                            s sqrt(s + 1)
(%i29) ilt(%,s,t);
                                  1
(%o29)            ilt(-------------, s, t)
                       s sqrt(s + 1)
```

$$\mathcal{L}\{\mathrm{erf}(t)\} = e^{s^2/4}\,(1 - \mathrm{erf}(s/2))\,s^{-1}$$

Here **laplace** succeeds, and **ilt** and **specint** fail.

```
(%i30) laplace(erf(t),t,s);
                             2
                            s
                            --
                            4        s
                          %e   (1 - erf(-))
                                         2
(%o30)                    ------------------
                                  s
(%i31) ilt(%,s,t);
                            2
                           s
                           --
                           4        s
                         %e   (1 - erf(-))
                                        2
(%o31)            ilt(----------------, s, t)
                             s
(%i32) specint(exp(-s*t)*erf(t),t);
                          - s t
(%o32)            specint(%e      erf(t), t)
```

35

10.4.3 Use of the Dirac Delta Function (Unit Impulse Function) delta with laplace(..)

The **laplace** function recognises **delta(arg)** as part of the expression supplied to **laplace**. We have introduced the Dirac delta function (unit impulse function) in Sections 10.3.7 and 10.3.8. Here are three examples of using **delta** with **laplace**.

$\mathcal{L}\{\delta(t)\} = 1$

```
(%i1) laplace(delta(t),t,s);
(%o1)                                      1
(%i2) ilt(1,s,t);
(%o2)                                 ilt(1, s, t)
```

$\mathcal{L}\{\delta(t - a)\} = e^{-as}$

(for $a > 0$):

```
(%i3) laplace(delta(t-a),t,s);
Is   a   positive, negative, or zero?
p;
                                        - a s
(%o3)                                 %e
```

$\mathcal{L}\{\delta(t - a)\sin(b\,t)\} = \sin(a\,b)\,e^{-as}$

(for $a > 0$):

```
(%i4) laplace(delta(t-a)*sin(b*t),t,s);
Is   a   positive, negative, or zero?
p;
                                         - a s
(%o4)                          sin(a b) %e
```

In this last example, we see that **laplace** is simply using

$$\int_0^\infty f(t)\,\delta(t - a)\,dt = f(a) \tag{10.61}$$

with $a > 0$ and $f(t) = e^{-st}\sin(b\,t)$.

10.5 The Inverse Laplace Transform and Residues at Poles

10.5.1 ilt: Inverse Laplace Transform

The Maxima function: **ilt (expr, s, t)** computes the inverse Laplace transform of **expr** with respect to s and in terms of the parameter t. This Maxima function is able to compute the inverse Laplace transform only if **expr** is a rational algebraic function (a quotient of two polynomials).

In elementary work with Laplace transforms, one uses a "look-up table" of Laplace transforms, together with general properties of Laplace and inverse Laplace transforms, to determine inverse Laplace transforms of elementary and special functions.

For advanced work, one can use the general formula for the inverse Laplace transform

$$f(t > 0) = \frac{1}{2\pi i} \int_{\sigma - i\infty}^{\sigma + i\infty} e^{st} F(s) \, ds \qquad (10.62)$$

where the contour of integration is a vertical line in the complex $s = x + iy$ plane (ie., along a line $x = \sigma$), to the right of all singularities (poles, branch points, and essential singularities) of the integrand. Here we assume the only type of singularities we need to worry about are poles, isolated values of s where the approximate power series expansion for s near s_j is $g(s_j)/(s - s_j)^m$, where $g(s_j)$ is a finite number, and m is the "order" of the pole. If $m = 1$, we call it a "simple pole". At a particular point (x, y) of the contour, the exponential factor has the form $e^{st} = e^{ity} e^{tx}$. Because $t > 0$ the integrand is heavily damped by the factor e^{xt} when $x \rightarrow -\infty$, and the contour of integration can be turned into a closed contour by adding a zero contribution which is the integral of the same integrand along a large arc in the left hand s plane (in the limit that the radius of the arc approaches ∞), and the resulting closed contour integral can then be evaluated using the residue theorem of complex variable theory. The (counter-clockwise) closed contour integral is then given by $2\pi i$ times the sum of the residues of the integrand at the isolated poles enclosed by the contour.

The factor $2\pi i$ coming from the integral part of Eq.(10.62) is then cancelled by the factor $1/(2\pi i)$ which appears in the definition (Eq.(10.62)) of the inverse Laplace transform, and one has the simple result that **the inverse Laplace transform is the sum of the residues of the poles of the integrand**.

10.5.2   residue: Residues at Poles

We can use the Maxima **residue** function to illustrate how the sum of the residues at all the poles of the integrand reproduces the answer returned by **ilt**.

> **residue (expr, z, z_0)**
> Computes the residue in the complex plane of the expression **expr** when the variable z assumes the value **z_0**. The residue is the coefficient of **(z - z_0)^(-1)** in the Laurent series for **expr**.

```
(%i1) residue (s/(s^2+a^2), s, a*%i);
                                 1
(%o1)                            -
                                 2
(%i2) residue (sin(a*x)/x^4, x, 0);
                                  3
                                 a
(%o2)                          - --
                                 6
```

We start with some arbitrary rational algebraic function of s, use **ilt** to determine the corresponding function of t, and then compute the inverse Laplace transform (a second way) by summing the values of the residues of the Laplace inversion integrand in the complex s plane.

```
(%i3) fs : -s/(s^3 +4*s^2 + 5*s +2)$
(%i4) ft : ( ilt(fs,s,t),collectterms(%%,exp(-t),exp(-2*t) ) );
                            - t         - 2 t
(%o4)               (t - 2) %e    + 2 %e
```

```
(%i5) (laplace(ft,t,s), ratsimp(%%) );
                                         s
(%o5)                      - -------------------
                              3      2
                             s  + 4 s  + 5 s + 2
(%i6) fs - %;
(%o6)                                 0
(%i7) fst : exp(s*t)*fs;
                                         s t
                                    s %e
(%o7)                      - -------------------
                              3      2
                             s  + 4 s  + 5 s + 2
(%i8) partfrac(fst,s);
                           s t        s t        s t
                      2 %e        2 %e          %e
(%o8)                 ------- - ------- + --------
                       s + 2     s + 1            2
                                            (s + 1)
```

The inverse Laplace transform **integrand** above was called **fst**, and we used **partfrac** on **fst** to exhibit the poles of this integrand. We see that there is one simple pole at $s = -2$ and one pole of order 2 located at $s = -1$.

To use the Maxima function **residue**, we need to tell Maxima that $t > 0$.

```
(%i9) assume( t > 0 )$
(%i10) r1 : residue(fst,s,-2);
                                         - 2 t
(%o10)                            2 %e
(%i11) r2 : residue(fst,s,-1);
                                          - t
(%o11)                          (t - 2) %e
(%i12) r1 + r2 - ft;
(%o12)                                0
```

which shows that the sum of the residues reproduces the function of t which **ilt** produced.

This concludes our discussion of the inverse Laplace transform.