Maxima by Example: Ch.7: Symbolic Integration Tools *

Edwin L. Woollett

January 31, 2009

Contents

*This version uses Maxima 5.17.1. This is a live document. Check `http://www.csulb.edu/~woollett/` for the latest version of these notes. Send comments and suggestions to `woollett@charter.net`

7 Symbolic Integration Tools

7.1 Symbolic Integration: integrate(...)

Although the process of finding an integral can be viewed as the inverse of the process of finding a derivative, in practice finding an integral is more difficult. It turns out that the integral of fairly simple looking functions cannot be expressed in terms of well known functions, and this has been one motivation for the introduction of definitions for a variety of "special functions", as well as efficient methods for numerical integration (quadrature) discussed in Ch. 8.
In the following, we always assume we are dealing with a real function of a real variable and that the function is single-valued and continuous over the intervals of interest.
The Maxima manual entry for **integrate** includes (we have made some changes and additions):

> Function: **integrate(expr, var)**
> Function: **integrate(expr, var, a, b)**
> Attempts to symbolically compute the integral of **expr** with respect to **var**. **integrate(expr, var)** is an indefinite integral, while **integrate(expr, var, a, b)** is a definite integral, with limits of integration **a** and **b**. The integral is returned if **integrate** succeeds. Otherwise the return value is the noun form of the integral (the quoted operator **'integrate**) or an expression containing one or more noun forms. The noun form of **integrate** is displayed with an integral sign if **display2d** is set to **true** (which is the default).
>
> In some circumstances it is useful to construct a noun form by hand, by quoting **integrate** with a single quote, e.g., **'integrate(expr, var)**. For example, the integral may depend on some parameters which are not yet computed. The noun may be applied to its arguments by **ev(iexp, nouns)** where **iexp** is the noun form of interest.
>
> The Maxima function **integrate** is defined by the lisp function **$integrate** in the file /src/simp.lisp. The indefinite integral invocation, **integrate(expr,var)**, results in a call to the lisp function **sinint**, defined in src/sin.lisp, unless the flag **risch** is present, in which case the lisp function **rischint**, defined in src/risch.lisp, is called. The definite integral invocation, **integrate(expr,var,a,b)**, causes a call to the lisp function **$defint**, defined in src/defint.lisp. The lisp function **$defint** is available as the Maxima function **defint** and can be used to bypass **integrate** for a definite integral.
>
> To integrate a Maxima function $\mathbf{f}(\mathbf{x})$, insert $\mathbf{f}(\mathbf{x})$ in the **expr** slot.
>
> **integrate** does not respect implicit dependencies established by the **depends** function.
>
> **integrate** may need to know some property of the parameters in the integrand. **integrate** will first consult the **assume** database, and, if the variable of interest is not there, **integrate** will ask the user. Depending on the question, suitable responses are **yes;** or **no;**, or **pos;**, **zero;**, or **neg;**. Thus, the user can use the **assume** function to avoid all or some questions.

7.2 Integration Examples and also defint, ldefint, beta, gamma, erf, and logabs

Example 1

First the indefinite integral $\int \mathbf{sin^3 x\,dx}$:

```
(%i1) integrate (sin(x)^3, x);
                                   3
                               cos (x)
(%o1)                          ------- - cos(x)
                                  3
(%i2) ( diff (%,x), trigsimp(%%) );
                                       3
(%o2)                                sin (x)
```

Notice that the indefinite integral returned by **integrate does not include** the arbitrary constant of integration which can always be added.

If the returned integral is correct (up to an arbitrary constant), then the first derivative of the returned indefinite integral should be the original integrand, although we may have to simplify the result manually (as we had to do above).

Example 2

Another indefinite integral, $\int \mathbf{x\,(b^2 - x^2)^{-1/2}\,dx}$:

```
(%i3) integrate (x/ sqrt (b^2 - x^2), x);
                                        2    2
(%o3)                           - sqrt(b  - x )
(%i4) diff(%,x);
                                       x
(%o4)                          -------------
                                     2    2
                               sqrt(b  - x )
```

Example 3

The definite integral can be related to "area under a curve" and is the more accessible concept, while the indefinite integral is a function whose derivative is the original integrand.

Here is a definite integral, $\int_0^{\pi} \mathbf{\cos^2 x\,e^x\,dx}$:

```
(%i5) i3 : integrate (cos(x)^2 * exp(x), x, 0, %pi);

                                     %pi
                                3 %e       3
(%o5)                           ------- - -
                                   5       5
```

Instead of **integrate**, you can try **ldefint** (think Limit definite integral), which may provide an alternative form of the answer (if successful).

From the Maxima manual:

> Function: **ldefint(expr, x, a, b)**
> Attempts to compute the definite integral of **expr** by using **limit** to evaluate the indefinite integral of **expr** with respect to **x** at the upper limit **b** and at the lower limit **a**. If it fails to compute the definite integral, **ldefint** returns an expression containing limits as noun forms.
>
> **ldefint** is not called from **integrate**, so executing **ldefint(expr, x, a, b)** may yield a different result than **integrate(expr, x, a, b)**. **ldefint** always uses the same method to evaluate the definite integral, while **integrate** may employ various heuristics and may recognize some special cases.

Here is an example of use of **ldefint**, as well as the direct use of **defint** (which bypasses **integrate**):

```
(%i6) ldefint (cos(x)^2 * exp(x), x, 0, %pi);
                                      %pi
                                   3 %e      3
(%o6)                              ------- - -
                                      5      5
(%i7) defint (cos(x)^2 * exp(x), x, 0, %pi);
                                      %pi
                                   3 %e      3
(%o7)                              ------- - -
                                      5      5
```

Example 4

Here is an example of a definite integral over an infinite range, $\int_{-\infty}^{\infty} \mathbf{x^2}\, \mathbf{e^{-x^2}}\, \mathbf{dx}$:

```
(%i8) integrate (x^2 * exp(-x^2), x, minf, inf);
                                    sqrt(%pi)
(%o8)                               ---------
                                        2
```

To check this integral, we first ask for the indefinite integral and check it by differentiation.

```
(%i9) i1 : integrate(x^2*exp(-x^2),x );
                                                    2
                                                 - x
                          sqrt(%pi) erf(x)    x %e
(%o9)                     ---------------- - --------
                                 4                2
(%i10) diff(i1,x);
                                          2
                                     2  - x
(%o10)                              x  %e
```

Thus the indefinite integral is correct. The second term, heavily damped by the factor $\mathbf{e^{-x^2}}$ at $\pm\infty$, does not contribute to the definite integral. The first term is proportional to the (Gauss) error function, $\mathbf{erf(x)}$, in which $\mathbf{x}$ is real. For (in general) complex $\mathbf{w = u + i\,v}$,

$$\mathbf{Erf(w)} = \frac{\mathbf{2}}{\sqrt{\boldsymbol{\pi}}} \int_{\mathbf{0}}^{\mathbf{w}} \mathbf{e^{-z^2}}\, \mathbf{dz} \tag{7.1}$$

in which we can integrate over any path connecting $\mathbf{0}$ and $\mathbf{w}$ in the complex $\mathbf{z = x + i\,y}$ plane, since the integrand is an entire function of $\mathbf{z}$ (no singularities in the finite $\mathbf{z}$ plane. You can make your own plot using your favorite method. The author is particularly enamoured of the **qdraw** package, discussed in Chapter 5 of Maxima by Example.

```
(%i11) ( load(draw),load(qdraw) )$
                qdraw(...), qdensity(...), syntax: type qdraw();
(%i12) qdraw(yr(-2,2),ex1(erf(x),x,-5,5,lw(5),lc(red),lk("erf(x)") ) )$
```

with the result:

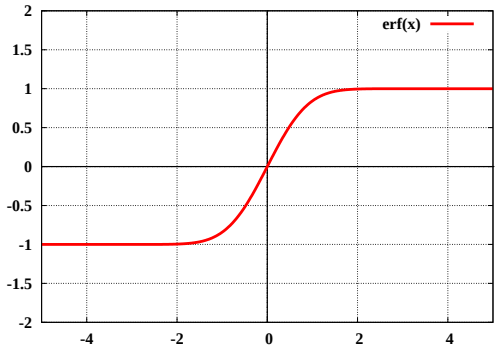

Figure 1: $\mathbf{erf(x)}$

Maxima's **limit** function confirms what the plot indicates:

```
(%i13) [limit(erf(x),x, inf), limit(erf(x),x, minf)];
(%o13)                              [1, - 1]
```

and hence we have the definite integral desired.

Example 5: Use of assume

We next calculate the definite integral $\int_0^\infty \mathbf{x^a}\,(\mathbf{x+1})^{-\mathbf{5/2}}\,\mathbf{dx}$.

```
(%i15) (assume(a>1),facts());
(%o15)                                  [a > 1]
(%i16) integrate (x^a/(x+1)^(5/2), x, 0, inf );
   2 a + 2
Is ------- an integer?
      5
no;
Is  2 a - 3  positive, negative, or zero?
neg;
                                          3
(%o16)                        beta(a + 1, - - a)
                                          2
```

The combination of $\mathbf{assume(a>1)}$ and $\mathbf{2\,a-3<0}$ means that we are assuming $\mathbf{1<a<3/2}$.
In response to the first question asked, if the quantity $\mathbf{b=(2\,a+2)/5}$ is equal to the integer $\mathbf{1}$, then $\mathbf{a}$ would also equal $\mathbf{1}$, and if the quantity $\mathbf{b}$ is equal to the next higher integer $\mathbf{2}$, then $\mathbf{a=4}$. Hence to be consistent, we must answer **no** to the first question.

Let's tell Maxima to **forget** about the **assume** assignment and see what the difference is.

```
(%i17) ( forget(a>1), facts() );
(%o17)                                 []
(%i18) is( a>1 );
(%o18)                               unknown
(%i19) integrate (x^a/(x+1)^(5/2), x, 0, inf );
Is  a + 1  positive, negative, or zero?
pos;
   2 a + 2
Is ------- an integer?
      5
no;
Is  2 a - 3  positive, negative, or zero?
neg;
                                        3
(%o19)                       beta(a + 1, - - a)
                                        2
(%i20) [is( a>1 ), facts() ];
(%o20)                           [unknown, []]
```

Thus we see that omitting the initial **assume** statement causes **integrate** to ask one additional question. We also see that answering questions posed by the **integrate** dialogue script does not result in population of the **facts** list.

The Maxima **beta** function has the manual entry:

> Function: **beta(x, y)**
> The **beta** function, defined as
> **gamma(x) gamma(y)/gamma(x + y)**.

In the usual mathematics notation, the beta function can be defined in terms of the gamma function as

$$\mathbf{B}(\mathbf{r},\mathbf{s}) = \frac{\mathbf{\Gamma}(\mathbf{r})\,\mathbf{\Gamma}(\mathbf{s})}{\mathbf{\Gamma}(\mathbf{r}+\mathbf{s})} \tag{7.2}$$

for all $\mathbf{r}, \mathbf{s}$ in the complex plane.

The Maxima **gamma** function has the manual entry

> Function: **gamma(x)**
> The **gamma** function. See also **makegamma**. The variable **gammalim** controls simplification of the **gamma** function. The **Euler-Mascheroni constant** is $\%\mathbf{gamma}$.

The gamma function can be defined for complex $\mathbf{z}$ and $\mathbf{Re}(\mathbf{z}) > \mathbf{0}$ by the integral along the real $\mathbf{u}$ axis

$$\mathbf{\Gamma}(\mathbf{z}) = \int_{\mathbf{0}}^{\infty} \mathbf{u}^{\mathbf{z-1}}\,\mathbf{e}^{-\mathbf{u}}\,\mathbf{d}\,\mathbf{u} \tag{7.3}$$

and for $\mathbf{Im}(\mathbf{z}) = \mathbf{0}$ and $\mathbf{Re}(\mathbf{z}) = \mathbf{n}$ and $\mathbf{n}$ an integer greater than zero we have

$$\mathbf{\Gamma}(\mathbf{n}+\mathbf{1}) = \mathbf{n}\,! \tag{7.4}$$

How can we check the definite integral Maxima has offered? If we ask the **integrate** function for the indefinite integral, we get the "noun form", a signal of failure:

```
(%i21) i1 : integrate(x^a/(x+1)^(5/2), x );
                                /       a
                                [      x
(%o21)                          I ---------- dx
                                ]         5/2
                                / (x + 1)
(%i22) grind(i1)$
'integrate(x^a/(x+1)^(5/2),x)$
```

Just for fun, let's include the **risch** flag and see what we get:

```
(%i23) i1 : integrate(x^a/(x+1)^(5/2), x ), risch;
                          /            a log(x)
                          [          %e
(%o23)                    I -------------------------- dx
                          ]                2
                          / sqrt(x + 1) (x  + 2 x + 1)
```

We again are presented with a noun form, but the integrand has been written in a different form, in which the identity

$$\mathbf{x^A = e^{A\,\ln(x)}}$$

has been used.

We can at least make a spot check for a value of the parameter $\mathbf{a}$ in the middle of the assumed range $(\mathbf{1}, \mathbf{3/2})$, namely for $\mathbf{a = 5/4}$.

```
(%i24) beta(9/4,1/4);
                                     9  1
(%o24)                            beta(-, -)
                                     4  4
(%i25) float(%);
(%o25)                          3.090124462168952
(%i26) quad_qagi(x^(5/4)/(x+1)^(5/2),x, 0, inf);
(%o26)        [3.090124462010259, 8.6105700347616221E-10, 435, 0]
```

We have used the quadrature routine, **quad_qagi** appropriate to an infinite or semi-infinite interval. The first element of the returned list is the numerical answer, which agrees with the analytic answer.

Example 6: Automatic Change of Variable and gradef

Here is an example which illustrates Maxima's ability to make a change of variable to enable the return of an indefinite integral. The task is to evaluate the indefinite integral

$$\int \frac{\mathbf{\sin(r^2)\,dr(x)/dx}}{\mathbf{q(r)}}\,\mathbf{dx} \tag{7.5}$$

by telling Maxima that the $\mathbf{\sin(r^2)}$ in the numerator is related to $\mathbf{q(r)}$ in the denominator by the derivative:

$$\frac{\mathbf{d\,q(u)}}{\mathbf{d\,u}} = \mathbf{\sin(u^2)}. \tag{7.6}$$

We would manually rewrite the integrand (using the chain rule) as

$$\frac{\mathbf{\sin(r^2)\,dr(x)/dx}}{\mathbf{q}} = \frac{\mathbf{1}}{\mathbf{q}}\,\mathbf{(dq(r)/dr)\,(dr(x)/dx)} = \frac{\mathbf{1}}{\mathbf{q}}\,\frac{\mathbf{dq}}{\mathbf{dx}} = \frac{\mathbf{d}}{\mathbf{dx}}\,\mathbf{\ln(q)} \tag{7.7}$$

and hence obtain the indefinite integral $\mathbf{\ln(q(r(x)))}$.

Here we assign the derivative knowledge using **gradef** (as discussed in Ch. 6):

```
(%i1) gradef(q(u),sin(u^2) )$
(%i2) integrand : sin(r(x)^2)* 'diff(r(x),x ) /q( r(x) ) ;
                              d                2
                             (-- (r(x))) sin(r (x))
                              dx
(%o2)                        ----------------------
                                    q(r(x))
(%i3) integrate(integrand,x);
(%o3)                             log(q(r(x)))
(%i4) diff(%,x);
                              d                2
                             (-- (r(x))) sin(r (x))
                              dx
(%o4)                        ----------------------
                                    q(r(x))
```

Note that **integrate** pays no attention to **depend** assignments, so the briefer type of notation which **depend** allows with differentiation cannot be used with **integrate**:

```
(%i5) depends(r,x,q,r);
(%o5)                            [r(x), q(r)]
(%i6) integrand : sin(r^2)* 'diff(r,x) / q;
                                  dr      2
                                  -- sin(r )
                                  dx
(%o6)                             ----------
                                      q
(%i7) integrate(integrand,x);
                                        /
                                    2   [ dr
                                sin(r ) I -- dx
                                        ] dx
                                        /
(%o7)                           ---------------
                                       q
```

which is fatally flawed, since Maxima pulled both $\mathbf{sin(r(x)^2)}$ and $\mathbf{1/q(r(x))}$ outside the integral.

Of course, the above **depends** assignment will still allow Maxima to rewrite the derivative of **q** with respect to **x** using the chain rule:

```
(%i8) diff(q,x);
                                     dq dr
(%o8)                                -- --
                                     dr dx
```

Example 7: Integration of Rational Algebraic Functions, rat, ratsimp, and partfrac

A *rational algebraic function* can be written as a quotient of two polynomials. Consider the following function of **x**.

```
(%i1) e1 : x^2 + 3*x -2/(3*x)+110/(3*(x-3)) + 12;
                          2          2       110
(%o1)                   x  + 3 x - --- + --------- + 12
                                   3 x   3 (x - 3)
```

We can obviously find the lowest common denominator and write this as the ratio of two polynomials, using either **rat** or **ratsimp**.

```
(%i2) e11 : ratsimp(e1);
                                  4       2
                                 x  + 3 x  + 2
(%o2)                            -------------
                                      2
                                     x  - 3 x
```

Because the polynomial in the numerator is of higher degree than the polynomial in the denominator, this is called an *improper rational fraction.* Any improper rational fraction can be reduced by division to a mixed form, consisting of a sum of some polynomial and a sum of proper fractions. We can recover the "partial fraction" representation in terms of *proper rational fractions* (numerator degree less than denominator degree) by using **partfrac**`(expr, var)`.

```
(%i3) e12 : partfrac(e11,x);
                          2          2        110
(%o3)                    x  + 3 x - --- + --------- + 12
                                    3 x    3 (x - 3)
```

With this function of **x** expressed in partial fraction form, you are able to write down the indefinite integral immediately (ie., by inspection, without using Maxima). But, of course, we want to practice using Maxima!

```
(%i4) integrate(e11,x);
                                                  3      2
                  2 log(x)   110 log(x - 3)   2 x  + 9 x  + 72 x
(%o4)           - -------- + -------------- + ------------------
                     3             3                  6
(%i5) integrate(e12,x);
                                                 3      2
                  2 log(x)   110 log(x - 3)   x    3 x
(%o5)           - -------- + -------------- + -- + ---- + 12 x
                     3             3          3     2
```

Maxima has to do less work if you have already provided the partial fraction form as the integrand; otherwise, Maxima internally seeks a partial fraction form in order to do the integral.

Example 8

The next example shows that **integrate** can sometimes split the integrand into at least one piece which can be integrated, and leaves the remainder as a formal expression (using the noun form of **integrate**). This may be possible if the denominator of the integrand is a polynomial which Maxima can factor.

```
(%i6) e2: 1/(x^4 - 4*x^3 + 2*x^2 - 7*x - 4);
                                   1
(%o6)                  --------------------------
                         4      3      2
                        x  - 4 x  + 2 x  - 7 x - 4
(%i7) integrate(e2,x);
                                    /  2
                                    [ x  + 4 x + 18
                                    I ------------- dx
                                    ]  3
                       log(x - 4)   / x  + 2 x + 1
(%o7)                  ---------- - ----------------
                           73              73
(%i8) grind(%)$
log(x-4)/73-('integrate((x^2+4*x+18)/(x^3+2*x+1),x))/73$
```

```
(%i9) factor(e2);
                                    1
(%o9)                     ----------------------
                                    3
                           (x - 4) (x  + 2 x + 1)
(%i10) partfrac(e2,x);
                                      2
                             1       x  + 4 x + 18
(%o10)                   ---------- - -----------------
                         73 (x - 4)           3
                                     73 (x  + 2 x + 1)
```

We have first seen what Maxima can do with this integrand, using the **grind** function to clarify the resulting expression, and then we have used **factor** and **partfrac** to see how the split-up arises. Despite a theorem that the integral of every rational function can be expressed in terms of algebraic, logarithmic and inverse trigonometric expressions, Maxima declines to return a symbolic expression for the second, formal, piece of this integral (which is good because the exact symbolic answer is an extremely long expression).

Example 9: The logabs Parameter and log

There is a global parameter **logabs** whose default value is **false** and which affects what is returned with an indefinite integral containing logs.

```
(%i11) logabs;
(%o11)                                false
(%i12) integrate(1/x,x);
(%o12)                               log(x)
(%i13) diff(%,x);
                                       1
(%o13)                                 -
                                       x
(%i14) logabs:true$
(%i15) integrate(1/x,x);
(%o15)                          log(abs(x))
(%i16) diff(%,x);
                                       1
(%o16)                                 -
                                       x
(%i17) log(-1);
(%o17)                           log(- 1)
(%i18) float(%);
(%o18)                    3.141592653589793 %i
```

When we override the default and set **logabs** to **true**, the argument of the **log** function is wrapped with **abs**. According to the manual

> When doing indefinite integration where logs are generated, e.g. integrate(1/x,x), the answer is given in terms of log(abs(...)) if logabs is true, but in terms of log(...) if logabs is false. For definite integration, the logabs:true setting is used, because here "evaluation" of the indefinite integral at the endpoints is often needed.

7.3 Piecewise Defined Functions and integrate

We can use Maxima's **if**, **elseif**, and **else** construct to define piecewise defined functions. We first define and plot a square wave of unit height which extends from $\mathbf{1} \leq \mathbf{x} \leq \mathbf{3}$.

```
(%i1) u(x) := if x >= 1 and x <= 3 then 1 else 0$
(%i2) map('u,[0.5,1,2,3,3.5]);
(%o2)                          [0, 1, 1, 1, 0]
(%i3) (load(draw),load(qdraw))$
                qdraw(...), qdensity(...), syntax: type qdraw();
(%i4) qdraw( yr(-1,2),ex1(u(x),x,0,4,lw(5),lc(blue)) )$
```

This produces

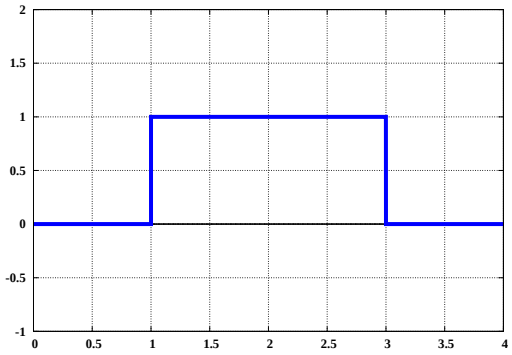

Figure 2: if $\mathbf{x} > \mathbf{1}$ and $\mathbf{x} < \mathbf{3}$ then $\mathbf{1}$ else $\mathbf{0}$

We next define a function which is $(\mathbf{x} - \mathbf{1})$ for $\mathbf{1} \leq \mathbf{x} < \mathbf{2}$ and is $(\mathbf{6} - \mathbf{2}\,\mathbf{x})$ for $\mathbf{2} \leq \mathbf{x} \leq \mathbf{3}$ and is $\mathbf{0}$ otherwise.

```
(%i5) g(x):= if x >= 1 and x < 2 then (x-1)
              elseif x >= 2 and x <= 3 then (6 - 2*x) else 0$
(%i6) map('g,[1/2,1,3/2,2,5/2,3,7/2]);
                                  1
(%o6)                       [0, 0, -, 2, 1, 0, 0]
                                  2
(%i7) qdraw( yr(-1,3),ex1(g(x),x,0,4,lw(5),lc(blue)) )$
```

which produces

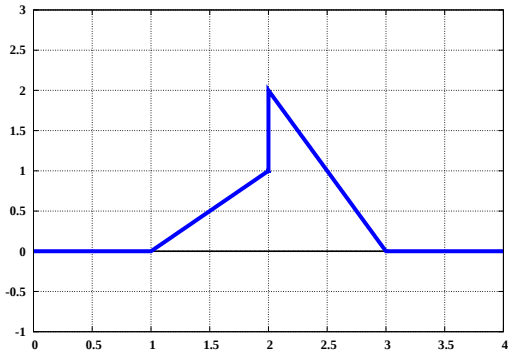

Figure 3: Example of a Piecewise Defined Function

This is not quite the figure we want, since it doesn't really show the kind of discontinuity we want to illustrate. With our definition of $\mathbf{g}(\mathbf{x})$, **draw2d** draws a blue vertical line from $(\mathbf{2}, \mathbf{1})$ to $(\mathbf{2}, \mathbf{2})$.

We can change the way $\mathbf{g(x)}$ is plotted to get the plot we want, as in

```
(%i8) block([small :1.0e-6],
         qdraw( yr(-1,3),ex1(g(x),x,0,2-small,lw(5),lc(blue)),
                  ex1(g(x),x,2+small,4,lw(5),lc(blue)) ) )$
```

which produces

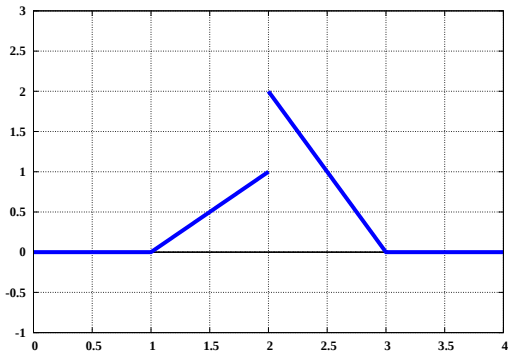

Figure 4: Example with More Care with Plot Limits

The Maxima function **integrate** cannot handle correctly a function defined with the **if**, **elseif**, and **else** constructs.

```
(%i9) integrate(g(x),x,0,4);
        4
       /
       [
(%o9) I   (if (x > 1) and (x < 2) then x - 1 elseif (x > 2) and (x < 3)
       ]
       /
        0
                                                          then 6 - 2 x else 0) dx
```

This means that for now we will have to break up the integration interval into sub-intervals by hand and add up the individual integral results.

```
(%i10) integrate(x-1,x,1,2) + integrate(6-2*x,x,2,3);
                                        3
(%o10)                                  -
                                        2
```

7.4 Area Between Curves Examples

It is easy to find the area between curves using **integrate**, but first make a plot so you can see what issues you have to worry about.

Example 1

We will start with a very simple example, finding the area between $\mathbf{f_1(x) = \sqrt{x}}$ and $\mathbf{f_2(x) = x^{3/2}}$. A simple plot shows that the curves cross at $\mathbf{x = 0}$ and $\mathbf{x = 1}$.

```
(%i1) (load(draw),load(qdraw) )$
                qdraw(...), qdensity(...), syntax: type qdraw();
(%i2) f1(x) := sqrt(x)$
```

```
(%i3) f2(x) := x^(3/2)$
(%i4) qdraw(   xr(-.5,1.5), yr(-.5,1.5),
         ex1(f1(x),x,0,1.5,lw(5),lc(blue) ),
         ex1(f2(x),x,0,1.5,lw(5),lc(red) ) )$
```

which produces the plot

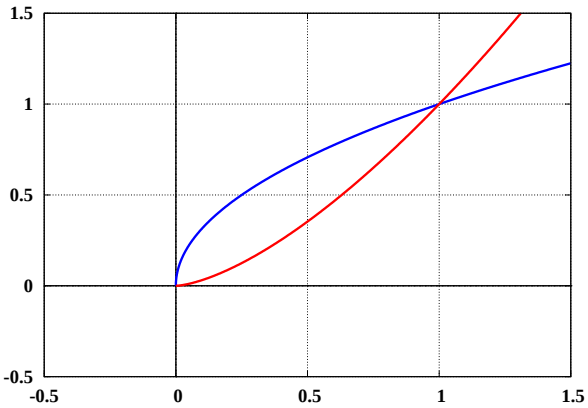

Figure 5: $\sqrt{x}$ (blue) and $x^{3/2}$ (red)

Next we redraw this simple plot, but add some shading in color to show the area. The simplest way to do this is to draw some vertical lines between the functions in the region of interest $0 \leq x \leq 1$. We can use the **qdraw** package function **line**. We first construct a list of x axis positions for the vertical lines, using values $0.01, 0.02, ...0.99$. We then construct a list `vv` of the vertical lines and merge that list with a list `qdlist` containing the rest of the plot instructions. We then use **apply** to pass this list as a set of arguments to **qdraw**.

```
(%i5) qdlist : [xr(-.5,1.5), yr(-.5,1.5),
         ex1(f1(x),x,0,1.5,lw(5),lc(blue) ) ,
         ex1(f2(x),x,0,1.5,lw(5),lc(red) ) ]$
(%i6) xv:float(makelist(i,i,1,99)/100)$
(%i7) (vv:[],for x in xv do
        vv:cons(line(x,f2(x),x,f1(x),lw(1),lc(khaki) ),vv),
        vv:reverse(vv) )$
(%i8) qdlist : append(vv,qdlist)$
(%i9) apply('qdraw, qdlist)$
```

which produces the result

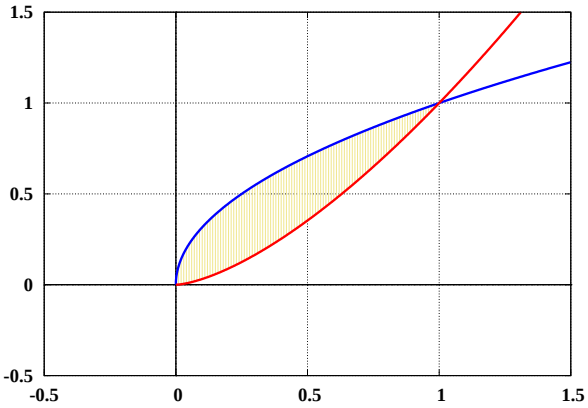

Figure 6: $\sqrt{x}$ (blue) and $x^{3/2}$ (red)

If we did not know the intersection location of the two curves, we could use **solve** or **find_root** for example.

```
(%i10) solve( f1(x) = f2(x),x );
(%o10)                          [x = 0, x = 1]
```

Once we know the interval to use for adding up the area and we know that in this interval $\mathbf{f_1(x) > f_2(x)}$, we simply sum the infinitesimal areas given by $\mathbf{(f_1(x) - f_2(x))\,dx}$ (base $\mathbf{dx}$ columns) over the interval $\mathbf{0 \le x \le 1}$.

```
(%i11) integrate(f1(x) - f2(x),x,0,1);
                                        4
(%o11)                                 ----
                                        15
```

so the area equals $\mathbf{4/15}$.

$$\int_0^1 (\sqrt{x} - x^{3/2})\,dx = 4/15 \tag{7.8}$$

Example 2

As a second example we consider two polynomial functions:
$\mathbf{f_1(x) = (3/10)\,x^5 - 3\,x^4 + 11\,x^3 - 18\,x^2 + 12\,x + 1}$
and $\mathbf{f_2(x) = -4\,x^3 + 28\,x^2 - 56\,x + 32}$. We first make a simple plot for orientation.

```
(%i1) f1(x) := (3/10)*x^5 -3*x^4 + 11*x^3 -18*x^2 + 12*x + 1$
(%i2) f2(x) := -4*x^3 + 28*x^2 -56*x + 32$
(%i3) (load(draw),load(qdraw) )$
               qdraw(...), qdensity(...), syntax: type qdraw();

(%i4) qdraw(yr(-20,20),ex1(f1(x),x,-1,5,lc(blue) ),
               ex1(f2(x),x,-1,5,lc(red)))$
```

which produces the plot

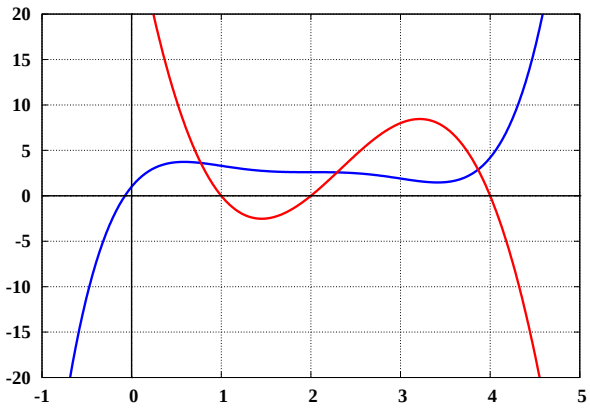

Figure 7: $\mathbf{f_1(x)}$ (blue) and $\mathbf{f_2(x)}$ (red)

Using the cursor on the plot, and working from left to right, $\mathbf{f_1}$ becomes larger than $\mathbf{f_2}$ at about $(\mathbf{0.76}, \mathbf{3.6})$, becomes less than $\mathbf{f_2}$ at about $(\mathbf{2.3}, \mathbf{2.62})$, and becomes greater than $\mathbf{f_2}$ at about $(\mathbf{3.86}, \mathbf{2.98})$.

The **solve** function is not able to produce an analytic solution, but returns a polynomial whose roots are the solutions we want.

```
(%i5) solve(f1(x) = f2(x),x);
                      5        4         3         2
(%o5)         [0 = 3 x  - 30 x  + 150 x  - 460 x  + 680 x - 310]
(%i6) grind(%)$
[0 = 3*x^5-30*x^4+150*x^3-460*x^2+680*x-310]$
```

By selecting and copying the **grind(..)** output, we can use that result to paste in the definition of a function $\mathbf{p(x)}$ which we can then use with **find_root**.

```
(%i7) p(x) := 3*x^5-30*x^4+150*x^3-460*x^2+680*x-310$
(%i8) x1 : find_root(p(x),x,.7,1);
(%o8)                          0.77205830452781
(%i9) x2 : find_root(p(x),x,2.2,2.4);
(%o9)                          2.291819210962957
(%i10) x3 : find_root(p(x),x,3.7,3.9);
(%o10)                          3.865127100061791
(%i11) map('p, [x1,x2,x3] );
(%o11)                [0.0, 0.0, 9.0949470177292824E-13]
(%i12) [y1,y2,y3] : map('f1, [x1,x2,x3] );
(%o12)     [3.613992056691179, 2.575784006305792, 2.882949345140702]
```

We have checked the solutions by seeing how close to zero $\mathbf{p(x)}$ is when $\mathbf{x}$ is one of the three roots $[\mathbf{x1}, \mathbf{x2}, \mathbf{x3}]$. We now split up the integration into the two separate regions where one or the other function is larger.

```
(%i13) ratprint:false$
(%i14) i1 : integrate(f1(x) - f2(x),x,x1,x2);
                                    41875933
(%o14)                              --------
                                    7947418
(%i15) i2 : integrate(f2(x)-f1(x),x,x2,x3);
                                    12061231
(%o15)                              --------
                                    1741444
(%i16) area : i1 + i2;
                                   30432786985
(%o16)                             -----------
                                   2495489252
(%i17) area : float(area);
(%o17)                         12.19511843643877
```

Hence the total area enclosed is about $\mathbf{12.195}$. Maxima tries to calculate exactly, replacing floating point numbers with ratios of integers, and the default is to warn the user about these replacements. Hence we have used **ratprint:false$** to turn off this warning.

7.5 Arc Length of an Ellipse

Consider an ellipse whose equation is

$$\frac{\mathbf{x^2}}{\mathbf{a^2}} + \frac{\mathbf{y^2}}{\mathbf{b^2}} = \mathbf{1}. \tag{7.9}$$

in which we assume $\mathbf{a}$ is the semi-major axis of the ellipse so $\mathbf{a > b}$. In the first quadrant ($\mathbf{0 \le x \le a}$ and $\mathbf{0 \le y \le b}$), we can solve for $\mathbf{y}$ as a function of $\mathbf{x}$:

$$\mathbf{y(x) = b\sqrt{1 - (x/a)^2}}. \tag{7.10}$$

If $\mathbf{S}$ is the arc length of the ellipse, then, by symmetry, one fourth of the arclength can be calculated using the first quadrant integral

$$\frac{\mathbf{S}}{\mathbf{4}} = \int_{\mathbf{0}}^{\mathbf{a}} \sqrt{\mathbf{1} + \left(\frac{\mathbf{dy}}{\mathbf{dx}}\right)^{\mathbf{2}}}\,\mathbf{dx} \tag{7.11}$$

We will start working with the argument of the square root, making a change of variable $\mathbf{x \to z}$, with $\mathbf{x = a\,z}$, so $\mathbf{dx = a\,dz}$, and $\mathbf{a}$ will come outside the integral from the transformation of $\mathbf{dx}$. The integration limits using the $\mathbf{z}$ variable are $\mathbf{0 \le z \le 1}$.

We will then replace the semi-minor axis $\mathbf{b}$ by an expression depending on the ellipse eccentricity $\mathbf{e}$, which has $\mathbf{0 < e \leq 1}$, and whose square is given by

$$\mathbf{e^2 = 1 - \left(\frac{b}{a}\right)^2 \leq 1} \tag{7.12}$$

(since $\mathbf{b < a}$), so

$$\mathbf{b = a\sqrt{1 - e^2}} \tag{7.13}$$

```
(%i1) assume(a>0,b>0,a>b,e>0,e<1 )$
(%i2) y:b*sqrt(1 - (x/a)^2)$
(%i3) dydx : diff(y,x)$
(%i4) e1 : 1 + dydx^2;
                                    2  2
                                   b  x
(%o4)                            ----------- + 1
                                         2
                                  4      x
                                 a  (1 - --)
                                         2
                                        a
(%i5) e2 : ( subst( [x = a*z,b = a*sqrt(1-e^2)],e1 ),ratsimp(%%) );
                                    2  2
                                   e  z  - 1
(%o5)                              ---------
                                     2
                                    z  - 1
(%i6) e3 : (-num(e2))/(-denom(e2));
                                        2  2
                                   1 - e  z
(%o6)                              ---------
                                         2
                                    1 - z
(%i7) e4 : dz*sqrt(num(e3))/sqrt(denom(e3));
                                          2  2
                                dz sqrt(1 - e  z )
(%o7)                           ------------------
                                          2
                                   sqrt(1 - z )
```

The two substitutions give us expression `e2`, and we use a desperate device to multiply the top and bottom by $(\mathbf{-1})$ to get `e3`. We then ignore the factor $\mathbf{a}$ which comes outside the integral and consider what is now inside the integral sign (with the required square root).

We now make another change of variables, with $\mathbf{z \rightarrow u}$, $\mathbf{z = \sin(u)}$, so $\mathbf{dz = \cos(u)\,du}$. The lower limit of integration $\mathbf{z = 0 = \sin(u)}$ transforms into $\mathbf{u = 0}$, and the upper limit $\mathbf{z = 1 = \sin(u)}$ transforms into $\mathbf{u = \pi/2}$.

```
(%i8) e5 : subst( [z = sin(u), dz = cos(u)*du ],e4 );
                                          2    2
                        du cos(u) sqrt(1 - e  sin (u))
(%o8)                   ------------------------------
                                       2
                              sqrt(1 - sin (u))
```

We now use **trigsimp** but help Maxima out with an **assume** statement about $\mathbf{\cos(u)}$ and $\mathbf{\sin(u)}$.

```
(%i9) assume(cos(u)>0, sin(u) >0)$
(%i10) e6 : trigsimp(e5);
                                       2    2
(%o10)                    du sqrt(1 - e  sin (u))
```

We then have

$$\frac{\mathbf{S}}{\mathbf{4}} = \mathbf{a}\int_{\mathbf{0}}^{\boldsymbol{\pi}/\mathbf{2}} \sqrt{\mathbf{1}-\mathbf{e^2}\,\mathbf{sin^2 u}}\,\mathbf{du} \tag{7.14}$$

Although **integrate** cannot return an expression for this integral in terms of elementary functions, in this form one is able to recognise the standard trigonometric form of the complete elliptic integral of the second kind, (a function tabulated numerous places and also available via Maxima).

Let

$$\mathbf{E(k)} = \mathbf{E}(\boldsymbol{\phi}=\boldsymbol{\pi}/\mathbf{2},\mathbf{k}) = \int_{\mathbf{0}}^{\boldsymbol{\pi}/\mathbf{2}} \sqrt{\mathbf{1}-\mathbf{k^2}\,\mathbf{sin^2 u}}\,\mathbf{du} \tag{7.15}$$

be the definition of the complete elliptic integral of the second kind.

The "incomplete" elliptic integral of the second kind (with two arguments) is

$$\mathbf{E}(\boldsymbol{\phi},\mathbf{k}) = \int_{\mathbf{0}}^{\boldsymbol{\phi}} \sqrt{\mathbf{1}-\mathbf{k^2}\,\mathbf{sin^2 u}}\,\mathbf{du}. \tag{7.16}$$

Hence we have for the arc length of our ellipse

$$\mathbf{S} = \mathbf{4\,a\,E(e)} = \mathbf{4\,a\,E}(\boldsymbol{\pi}/\mathbf{2},\mathbf{e}) = \mathbf{4\,a}\int_{\mathbf{0}}^{\boldsymbol{\pi}/\mathbf{2}} \sqrt{\mathbf{1}-\mathbf{e^2}\,\mathbf{sin^2 u}}\,\mathbf{du} \tag{7.17}$$

We can evaluate numerical values using **elliptic_ec**, where
$\mathbf{E(k)} = \mathbf{elliptic\_ec(k^2)}$, so $\mathbf{S} = \mathbf{4\,a\,elliptic\_ec(e^2)}$.

As a numerical example, take $\mathbf{a}=\mathbf{3}$, $\mathbf{b}=\mathbf{2}$, so $\mathbf{e^2}=\mathbf{5/9}$, and

```
(%i11) float(12*elliptic_ec(5/9));
(%o11)                         15.86543958929059
```

We can check this numerical value using **quad_qags**:

```
(%i12) first( quad_qags(12*sqrt(1 - (5/9)*sin(u)^2),u,0,%pi/2) );
(%o12)                          15.86543958929059
```

7.6 Double Integrals and the Area of an Ellipse

Maxima has no core function which will compute a symbolic double definite integral, (although it would be easy to construct one). Instead of constructing such a homemade function, to evaluate the double integral

$$\int_{\mathbf{u1}}^{\mathbf{u2}} \mathbf{du} \int_{\mathbf{v1(u)}}^{\mathbf{v2(u)}} \mathbf{dv}\,\mathbf{f(u,v)} \equiv \int_{\mathbf{u1}}^{\mathbf{u2}} \left(\int_{\mathbf{v1(u)}}^{\mathbf{v2(u)}} \mathbf{f(u,v)}\,\mathbf{dv} \right) \mathbf{du} \tag{7.18}$$

we use the Maxima code

```
integrate( integrate( f(u,v),v,v1(u),v2(u) ), u, u1,u2 )
```

in which $\mathbf{f(u,v)}$ can either be an expression depending on the variables $\mathbf{u}$ and $\mathbf{v}$, or a Maxima function, and likewise $\mathbf{v1(u)}$ and $\mathbf{v2(u)}$ can either be expressions depending on $\mathbf{u}$ or Maxima functions. Both $\mathbf{u}$ and $\mathbf{v}$ are "dummy variables", since the value of the resulting double integral does not depend on our choice of symbols for the integration variables; we could just as well use $\mathbf{x}$ and $\mathbf{y}$.

Example 1: Area of a Unit Square

The area of a unit square (the sides have length $\mathbf{1}$) is:

$$\int_{\mathbf{0}}^{\mathbf{1}} \mathbf{dx} \int_{\mathbf{0}}^{\mathbf{1}} \mathbf{dy} \equiv \int_{\mathbf{0}}^{\mathbf{1}} \left(\int_{\mathbf{0}}^{\mathbf{1}} \mathbf{dy} \right) \mathbf{dx} \tag{7.19}$$

which is done in Maxima as:

```
(%i1) integrate( integrate(1,y,0,1), x,0,1 );
(%o1)                                 1
```

Example 2: Area of an Ellipse

We seek the area of the ellipse such that points $(\mathbf{x}, \mathbf{y})$ on the boundary must satisfy:

$$\frac{\mathbf{x^2}}{\mathbf{a^2}} + \frac{\mathbf{y^2}}{\mathbf{b^2}} = \mathbf{1}. \tag{7.20}$$

The area will be four times the area of the first quadrant. The "first quadrant" refers to the region $\mathbf{0} \leq \mathbf{x} \leq \mathbf{a}$ and $\mathbf{0} \leq \mathbf{y} \leq \mathbf{b}$. For a given value of $\mathbf{y} > \mathbf{0}$, the region inside the arc of the ellipse in the first quadrant is determined by $\mathbf{0} \leq \mathbf{x} \leq \mathbf{x_{max}}$, where $\mathbf{x_{max}} = (\mathbf{a/b}) \sqrt{\mathbf{b^2 - y^2}}$. For a given value of $\mathbf{x} > \mathbf{0}$, the region inside the arc of the ellipse in the first quadrant is determined by $\mathbf{0} \leq \mathbf{y} \leq \mathbf{y_{max}}$, where $\mathbf{y_{max}} = (\mathbf{b/a}) \sqrt{\mathbf{a^2 - x^2}}$.

One way to find the area of the first quadrant of this ellipse is to sum the values of $\mathbf{dA} = \mathbf{dx\,dy}$ by "rows", fixing $\mathbf{y}$ and summing over $\mathbf{x}$ from $\mathbf{0}$ to $\mathbf{x_{max}}$ (which depends on the $\mathbf{y}$ chosen). That first sum over the $\mathbf{x}$ coordinate accumulates the area of that row, located at $\mathbf{y}$ and having width $\mathbf{dy}$. To get the total area (of the first quadrant) we then sum over rows, by letting $\mathbf{y}$ vary from $\mathbf{0}$ to $\mathbf{b}$.
This method corresponds to the formula

$$\frac{\mathbf{A}}{\mathbf{4}} = \int_{\mathbf{0}}^{\mathbf{b}} \left(\int_{\mathbf{0}}^{(\mathbf{a/b}) \sqrt{\mathbf{b^2 - y^2}}} \mathbf{dx} \right) \mathbf{dy}. \tag{7.21}$$

Here we calculate the first quadrant area using this method of summing over the area of each row.

```
(%i1) facts();
(%o1)                                 []
(%i2) assume(a > 0, b > 0, x > 0, x < a, y > 0,y < b )$
(%i3) facts();
(%o3)             [a > 0, b > 0, x > 0, a > x, y > 0, b > y]
(%i4) [xmax : (a/b)*sqrt(b^2 - y^2),ymax : (b/a)*sqrt(a^2-x^2)]$
(%i5) integrate( integrate( 1,x,0,xmax), y,0,b );
                                    %pi a b
(%o5)                               -------
                                       4
```

which implies that the area interior to the complete ellipse is $\pi\,\mathbf{a}\,\mathbf{b}$.

Note that we have tried to be "overly helpful" to Maxima's **integrate** function by constructing an "assume list" with everything we can think of about the variables and parameters in this problem. The main advantage of doing this is to reduce the number of questions which **integrate** decides it has to ask the user.

You might think that **integrate** would not ask for the sign of $(\mathbf{y} - \mathbf{b})$, or the sign of $(\mathbf{x} - \mathbf{a})$, since it should infer that sign from the integration limits. However, the **integrate** algorithm is "super cautious" in trying to never present you with a wrong answer. The general philosophy is that the user should be willing to work with **integrate** to assure a correct answer, and if that involves answering questions, then so be it.

The second way to find the area of the first quadrant of this ellipse is to sum the values of $\mathbf{dA = dx\,dy}$ by "columns", fixing $\mathbf{x}$ and summing over $\mathbf{y}$ from $\mathbf{0}$ to $\mathbf{y_{max}}$ (which depends on the $\mathbf{x}$ chosen). That first sum over the $\mathbf{y}$ coordinate accumulates the area of that column, located at $\mathbf{x}$ and having width $\mathbf{dx}$. To get the total area (of the first quadrant) we then sum over columns, by letting $\mathbf{x}$ vary from $\mathbf{0}$ to $\mathbf{a}$.

This method corresponds to the formula

$$\frac{\mathbf{A}}{\mathbf{4}} = \int_{\mathbf{0}}^{\mathbf{a}} \left(\int_{\mathbf{0}}^{(\mathbf{b/a})\sqrt{\mathbf{a^2 - x^2}}} \mathbf{dy} \right) \mathbf{dx} \tag{7.22}$$

and is implemented by

```
(%i6) integrate( integrate( 1,y,0,ymax), x,0,a );
                                      %pi a b
(%o6)                                 -------
                                         4
```

Example 3: Moment of Inertia for Rotation about the x-axis

We next calculate the moment of inertia for rotation of an elliptical lamina (having semi-axes $\mathbf{a}, \mathbf{b}$) about the $\mathbf{x}$ axis. We will call this quantity $\mathbf{I_x}$. Each small element of area $\mathbf{dA = dx\,dy}$ has a mass $\mathbf{dm}$ given by $\boldsymbol{\sigma}\,\mathbf{dA}$, where $\boldsymbol{\sigma}$ is the mass per unit area, which we assume is a constant independent of where we are on the lamina, and which we can express in terms of the total mass $\mathbf{m}$ of the elliptical lamina, and the total area $\mathbf{A = \pi\,a\,b}$ as

$$\boldsymbol{\sigma} = \frac{\textbf{total mass}}{\textbf{total area}} = \frac{\mathbf{m}}{\boldsymbol{\pi}\,\mathbf{a}\,\mathbf{b}} \tag{7.23}$$

Each element of mass $\mathbf{dm}$ contributes an amount $\mathbf{y^2\,dm}$ to the moment of inertia $\mathbf{I_x}$, where $\mathbf{y}$ is the distance of the mass element from the $\mathbf{x}$ axis. The complete value of $\mathbf{I_x}$ is then found by summing this quantity over the whole elliptical laminate, or because of the symmetry, by summing this quantity over the first quadrant and multiplying by $\mathbf{4}$.

$$\mathbf{I_x} = \int\int_{\textbf{ellipse}} \mathbf{y^2\,dm} = \int\int_{\textbf{ellipse}} \mathbf{y^2}\,\boldsymbol{\sigma}\,\mathbf{dx\,dy} = \mathbf{4}\,\boldsymbol{\sigma} \int\int_{\textbf{first quadrant}} \mathbf{y^2\,dx\,dy} \tag{7.24}$$

Here we use the method of summing over rows:

```
(%i7) sigma : m/(%pi*a*b)$
(%i8) 4*sigma*integrate( integrate(y^2,x,0,xmax),y,0,b );
                                       2
                                      b  m
(%o8)                                 ----
                                       4
```

Hence we have derived $\mathbf{I_x = m\,b^2/4}$ for the moment of inertia of an elliptical laminate having semi-axes $\mathbf{a}, \mathbf{b}$ for rotation about the $\mathbf{x}$ axis.

Finally, let's use **forget** to remove our list of assumptions, to show you the type of questions you might encounter in using **integrate** in this kind of problem if you don't help Maxima out. We try calculating the area of the first quadrant, using the method of summing over rows, without any assumptions provided:

```
(%i9) forget(a > 0, b > 0, x > 0, x < a, y > 0,y < b )$
(%i10) facts();
(%o10)                                []
(%i11) integrate( integrate(1,x,0,xmax),y,0,b);
Is  (y - b) (y + b)  positive, negative, or zero?
n;
```

```
     2    2
Is  b  - y   positive or zero?
p;
Is  b  positive, negative, or zero?
p;
                                    %pi a b
(%o11)                              -------
                                       4
```

and if we just tell Maxima that `b` is positive:

```
(%i12) assume(b>0)$
(%i13) integrate( integrate(1,x,0,xmax),y,0,b);
Is  (y - b) (y + b)  positive, negative, or zero?
n;
     2    2
Is  b  - y   positive or zero?
p;
                                    %pi a b
(%o13)                              -------
                                       4
```

This gives you some feeling for the value of being over-helpful with **integrate**.

7.7 Triple Integrals: Volume and Moment of Inertia of an Solid Ellipsoid

Consider an ellipsoid with semi-axes $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ such that $-\mathbf{a} \le \mathbf{x} \le \mathbf{a}$, $-\mathbf{b} \le \mathbf{y} \le \mathbf{b}$, and $-\mathbf{c} \le \mathbf{z} \le \mathbf{c}$. We also assume here that $\mathbf{a} > \mathbf{b} > \mathbf{c}$. Points $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ which live on the surface of this ellipsoid satisfy the equation of the surface

$$\frac{\mathbf{x^2}}{\mathbf{a^2}} + \frac{\mathbf{y^2}}{\mathbf{b^2}} + \frac{\mathbf{z^2}}{\mathbf{c^2}} = \mathbf{1}. \tag{7.25}$$

Volume

The volume of this ellipsoid will be $\mathbf{8}$ times the volume of the first octant, defined by $\mathbf{0} \le \mathbf{x} \le \mathbf{a}$, $\mathbf{0} \le \mathbf{y} \le \mathbf{b}$, and $\mathbf{0} \le \mathbf{z} \le \mathbf{c}$. To determine the volume of the first octant we pick fixed values of $(\mathbf{x}, \mathbf{y})$ somewhere in the first octant, and sum the elementary volume $\mathbf{dV}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \mathbf{dx\,dy\,dz}$ over all accessible values of $\mathbf{z}$ in this first octant, $\mathbf{0} \le \mathbf{z} \le \mathbf{c}\sqrt{\mathbf{1} - (\mathbf{x/a})^\mathbf{2} - (\mathbf{y/b})^\mathbf{2}}$. The result will still be proportional to $\mathbf{dx\,dy}$, and we sum the z-direction cylinders over the entire range of $\mathbf{y}$ (accessible in the first octant), again holding $\mathbf{x}$ fixed as before, so for given $\mathbf{x}$, we have $\mathbf{0} \le \mathbf{y} \le \mathbf{b}\sqrt{\mathbf{1} - (\mathbf{x/a})^\mathbf{2}}$. We now have a result proportional to $\mathbf{dx}$ (a sheet of thickness $\mathbf{dx}$ whose normal is in the direction of the $\mathbf{x}$ axis) which we sum over all values of $\mathbf{x}$ accessible in the first octant (with no restrictions): $\mathbf{0} \le \mathbf{x} \le \mathbf{a}$. Thus we need to evaluate the triple integral

$$\frac{\mathbf{V}}{\mathbf{8}} = \int_\mathbf{0}^\mathbf{a} \mathbf{dx} \int_\mathbf{0}^{\mathbf{ymax(x)}} \mathbf{dy} \int_\mathbf{0}^{\mathbf{zmax(x,y)}} \mathbf{dz} \equiv \int_\mathbf{0}^\mathbf{a} \left[\int_\mathbf{0}^{\mathbf{ymax(x)}} \left(\int_\mathbf{0}^{\mathbf{zmax(x,y)}} \mathbf{dz} \right) \mathbf{dy} \right] \mathbf{dx} \tag{7.26}$$

Here we call on **integrate** to do these integrals.

```
(%i1) assume(a>0,b>0,c>0,a>b,a>c,b>c)$
(%i2) assume(x>0,x<a,y>0,y<b,z>0,z<c)$
(%i3) zmax:c*sqrt(1-x^2/a^2-y^2/b^2)$
(%i4) ymax:b*sqrt(1-x^2/a^2)$
```

```
(%i5) integrate( integrate( integrate(1,z,0,zmax),y,0,ymax),x,0,a );
     2  2    2  2    2  2
Is  a  y  + b  x  - a  b   positive, negative, or zero?

n;
                                    %pi a b c
(%o5)                               ---------
                                        6
(%i6) vol : 8*%;
                                   4 %pi a b c
(%o6)                              -----------
                                        3
```

To answer the sign question posed by Maxima, we can look at the special case $\mathbf{y = 0}$ and note that since $\mathbf{x^2 \leq a^2}$, the expression will almost always be negative (ie., except for points of zero measure). Hence the expression is negative for all points interior to the surface of the ellipsoid. We thus have the result that the volume of an ellipsoid having semi-axes $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is given by

$$\mathbf{V = \frac{4\pi}{3} a\, b\, c} \tag{7.27}$$

We can now remove the assumption $\mathbf{a > b > c}$, since what we call the $\mathbf{x}$ axis is up to us and we could have chosen any of the principal axis directions of the ellipsoid the $\mathbf{x}$ axis.

Of course we will get the correct answer if we integrate over the total volume:

```
(%i7) [zmin:-zmax,ymin:-ymax]$
(%i8) integrate( integrate( integrate(1,z,zmin,zmax),y,ymin,ymax),x,-a,a );
     2  2    2  2    2  2
Is  a  y  + b  x  - a  b   positive, negative, or zero?

n;
                                   4 %pi a b c
(%o8)                              -----------
                                        3
```

Moment of Inertia

If each small volume element $\mathbf{dV = dx\, dy\, dz}$ has a mass given by $\mathbf{dm = \rho\, dV}$, where $\boldsymbol{\rho}$ is the mass density, and if $\boldsymbol{\rho}$ is a constant, then it is easy to calculate the moment of inertia $\mathbf{I_3}$ for rotation of the ellipsoid about the $\mathbf{z}$ axis:

$$\mathbf{I_3 = \int\int\int (x^2 + y^2)\, dm = \rho \int\int\int (x^2 + y^2)\, dx\, dy\, dz} \tag{7.28}$$

where the integration is over the volume of the ellipsoid. The constant mass density $\boldsymbol{\rho}$ is the mass of the ellipsoid divided by its volume.

```
(%i9) rho:m/vol;
                                        3 m
(%o9)                               -----------
                                    4 %pi a b c
(%i10) i3:rho*integrate(integrate(integrate(x^2+y^2,z,zmin,zmax),
                        y,ymin,ymax),x,-a,a );
     2  2    2  2    2  2
Is  a  y  + b  x  - a  b   positive, negative, or zero?

n;
```

```
                                  5  3           7
                           (8 %pi a  b  + 8 %pi a  b) m
(%o10)                     ----------------------------
                                       5
                                40 %pi a  b
(%i11) ratsimp(i3);
                                   2    2
                                 (b  + a ) m
(%o11)                           -----------
                                      5
```

Hence the moment of inertia for the rotation about the $\mathbf{z}$ axis of a solid ellipsoid having mass $\mathbf{m}$, and semi-axes $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ is:

$$\mathbf{I_3 = m\,(a^2 + b^2)/5}. \tag{7.29}$$

7.8 Derivative of a Definite Integral with Respect to a Parameter

Consider a definite integral in which the dummy integration variable is $\mathbf{x}$ and the integrand $\mathbf{f(x, y)}$ is a function of both $\mathbf{x}$ and some parameter $\mathbf{y}$. We also assume that the limits of integration $(\mathbf{a}, \mathbf{b})$ are also possibly functions of the parameter $\mathbf{y}$. You can find the proof of the following result (which assumes some mild restrictions on the funtions $\mathbf{f(x, y)}$, $\mathbf{a(y)}$ and $\mathbf{b(y)}$) in calculus texts:

$$\frac{\mathbf{d}}{\mathbf{dy}} \int_{\mathbf{a(y)}}^{\mathbf{b(y)}} \mathbf{f(x, y)}\,\mathbf{dx} = \int_{\mathbf{a}}^{\mathbf{b}} \frac{\boldsymbol{\partial}\,\mathbf{f(x, y)}}{\boldsymbol{\partial}\,\mathbf{y}}\,\mathbf{dy} + \mathbf{f(b(y), y)}\,\frac{\mathbf{d\,b}}{\mathbf{d\,y}} - \mathbf{f(a(y), y)}\,\frac{\mathbf{d\,a}}{\mathbf{d\,y}} \tag{7.30}$$

Here we ask Maxima for this result for arbitrary functions:

```
(%i1) expr : 'integrate(f(x,y),x,a(y),b(y) );
                                  b(y)
                                 /
                                 [
(%o1)                            I      f(x, y) dx
                                 ]
                                 /
                                  a(y)
(%i2) diff(expr,y);
                                                                  b(y)
                                                                 /
                    d                              d             [       d
(%o2) f(b(y), y) (-- (b(y))) - f(a(y), y) (-- (a(y))) + I      (-- (f(x, y))) dx
                    dy                             dy          ]       dy
                                                               /
                                                                a(y)
```

and we see that Maxima assumes we have enough smoothness in the functions involved to write down the formal answer in the correct form.

Example 1

We next display a simple example and begin with the simplest case, which is that the limits of integration do not depend on the parameter $\mathbf{y}$.

```
(%i3) expr : 'integrate(x^2 + 2*x*y, x,a,b);
                                b
                               /
                               [             2
(%o3)                          I  (2 x y + x ) dx
                               ]
                               /
                                a
(%i4) diff(expr,y);
                                       b
                                      /
                                      [
(%o4)                              2  I  x dx
                                      ]
                                      /
                                       a
(%i5) (ev(%,nouns), ratsimp(%%) );
                                      2    2
(%o5)                                 b  - a
(%i6) ev(expr,nouns);
                              2       3       2       3
                           3 b  y + b     3 a  y + a
(%o6)                      ----------- - -----------
                                3               3
(%i7) diff(%,y);
                                      2    2
(%o7)                                 b  - a
```

In the last two steps, we have verified the result by first doing the original integral and then taking the derivative.

Example 2

As a second example, we use an arbitrary upper limit $\mathbf{b(y)}$, and then evaluate the resulting derivative expression for $\mathbf{b(y) = y^2}$.

```
(%i1) expr : 'integrate(x^2 + 2*x*y, x,a,b(y) );
                               b(y)
                              /
                              [                2
(%o1)                         I      (2 x y + x ) dx
                              ]
                              /
                               a
(%i2) diff(expr,y);
                                                          b(y)
                                                         /
                    2                    d               [
(%o2)             (b (y) + 2 y b(y)) (-- (b(y))) + 2 I      x dx
                                       dy              ]
                                                       /
                                                        a
(%i3) ( ev(%,nouns,b(y)=y^2 ), expand(%%) );
                                   5      4    2
(%o3)                           2 y  + 5 y  - a
```

```
(%i4) ev(expr,nouns);
                          3            2          2      3
                         b (y) + 3 y b (y)    3 a  y + a
(%o4)                    ----------------- - -----------
                                 3                3
(%i5) diff(%,y);
              2     d                          d                 2
          3 b (y) (-- (b(y))) + 6 y b(y) (-- (b(y))) + 3 b (y)
                    dy                       dy                        2
(%o5)     ------------------------------------------------------ - a
                                     3
(%i6) ( ev(%,nouns,b(y)=y^2), (expand(%%) );
                                   5      4    2
(%o6)                             2 y  + 5 y  - a
```

We have again done the differentiation two ways as a check on consistency.

Example 3

An easy example is to derive

$$\frac{\mathbf{d}}{\mathbf{d\,t}}\int_{\mathbf{t}}^{\mathbf{t^2}}(\mathbf{2\,x+t})\,\mathbf{d\,x}=\mathbf{4\,t^3+3\,t^2-4\,t} \tag{7.31}$$

```
(%i7) expr : 'integrate(2*x + t,x,t,t^2);
                                  2
                                 t
                                /
                                [
(%o7)                          I   (2 x + t) dx
                                ]
                                /
                                 t
(%i8) (diff(expr,t),expand(%%) );
                                    3      2
(%o8)                           4 t  + 3 t  - 4 t
```

Example 4

Here is an example which shows a common use of the differentiation of an integral with respect to a parameter. The integral

$$\mathbf{f_1(a,w)}=\int_{\mathbf{0}}^{\infty}\mathbf{x\,e^{-a\,x}\cos(w\,x)\,dx} \tag{7.32}$$

can be done by Maxima, if we tell Maxima that $\mathbf{a>0}$ and $\mathbf{w>0}$ and if we ignore a **ind** ("indefinite") term which Maxima adds to the result due to a bug in Maxima's **limit** function.

```
(%i1) assume( a > 0, w > 0 )$
(%i2) integrate(x*exp(-a*x)*cos(w*x),x,0,inf);
                                        2    2
                                       w  - a
(%o2)                        ind - -----------------
                                     4       2  2    4
                                    w  + 2 a  w  + a
```

But if we could not find this result directly as above, we could find the result by setting $\mathbf{f_1(a,w) = -\partial f_2(a,w)/(\partial a)}$, where

$$\mathbf{f_2(a,w) = \int_0^\infty e^{-ax}\cos(w\,x)\,dx} \tag{7.33}$$

since the differentiation will result in the integrand being multiplied by the factor $(-\mathbf{x})$ and produce the negative of the integral of interest.

```
(%i3) i1 : 'integrate(exp(-a*x)*cos(w*x),x,0,inf)$
(%i4) i2 : ev(i1,nouns);
                                      a
(%o4)                               -------
                                     2    2
                                    w  + a
(%i5) di2 : (diff(i2,a), ratsimp(%%) );
                                     2    2
                                    w  - a
(%o5)                         -----------------
                               4      2  2    4
                              w  + 2 a  w  + a
(%i6) result : (-1)*(diff(i1,a) = di2 );
                inf
               /                                    2    2
               [         - a x                     w  - a
(%o6)          I     x %e      cos(w x) dx = - -----------------
               ]                                  4      2  2    4
               /                                 w  + 2 a  w  + a
                0
```

which reproduces the Maxima result without the buggy "`ind`". We suspect the **limit** function because Maxima can do the **indefinite** integral of the harder integral we want:

```
(%i7) indef:integrate(x*exp(-a*x)*cos(w*x),x );
         - a x     3    2
(%o7) (%e      (((w  + a  w) x + 2 a w) sin(w x)
                          2    3       2    2                4      2  2    4
                   + ((- a w  - a ) x + w  - a ) cos(w x)))/(w  + 2 a  w  + a )
(%i8) limit(indef,x,0,plus);
                                     2    2
                                    w  - a
(%o8)                         -----------------
                               4      2  2    4
                              w  + 2 a  w  + a
(%i9) limit(indef,x,inf);
(%o9)                                ind
(%i10) limit(exp(-a*x),x,inf);
(%o10)                               0
(%i11) limit(exp(-a*x)*cos(w*x),x,inf);
(%o11)                               0
(%i12) limit(x*exp(-a*x)*cos(w*x),x,inf);
(%o12)                               0
```

It is not clear why **limit** has a problem with `indef`, given that the expression is strongly damped by the $\mathbf{e^{-ax}}$ factor.

7.9 Integration by Parts

Suppose $\mathbf{f(x)}$ and $\mathbf{g(x)}$ are two continuously differentiable functions in the interval of interest. Then the **integration by parts rule** states that given an interval with endpoints $(\mathbf{a,b})$, (and of course assuming the

derivatives exist) one has

$$\int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f}(\mathbf{x})\,\mathbf{g}'(\mathbf{x})\,\mathbf{dx} = [\mathbf{f}(\mathbf{x})\,\mathbf{g}(\mathbf{x})]_{\mathbf{a}}^{\mathbf{b}} - \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f}'(\mathbf{x})\,\mathbf{g}(\mathbf{x})\,\mathbf{dx} \tag{7.34}$$

where the prime indicates differentiation. This result follows from the product rule of differentiation. This rule is often stated in the context of indefinite integrals as

$$\int \mathbf{f}(\mathbf{x})\,\mathbf{g}'(\mathbf{x})\,\mathbf{dx} = \mathbf{f}(\mathbf{x})\,\mathbf{g}(\mathbf{x}) - \int \mathbf{f}'(\mathbf{x})\,\mathbf{g}(\mathbf{x})\,\mathbf{dx} \tag{7.35}$$

or in an even shorter form, with $\mathbf{u} = \mathbf{f}(\mathbf{x})$, $\mathbf{du} = \mathbf{f}'(\mathbf{x})\,\mathbf{dx}$, $\mathbf{v} = \mathbf{g}(\mathbf{x})$, and $\mathbf{dv} = \mathbf{g}'(\mathbf{x})\,\mathbf{dx}$, as

$$\int \mathbf{u}\,\mathbf{dv} = \mathbf{u}\,\mathbf{v} - \int \mathbf{v}\,\mathbf{du} \tag{7.36}$$

In practice, we are confronted with an integral whose integrand can be viewed as the product of two factors, which we will call $\mathbf{f}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$:

$$\int \mathbf{f}(\mathbf{x})\,\mathbf{h}(\mathbf{x})\,\mathbf{dx} \tag{7.37}$$

and we wish to use integration by parts to get an integral involving the derivative of the first factor, $\mathbf{f}(\mathbf{x})$, which will hopefully result in a simpler integral. We then identify $\mathbf{h}(\mathbf{x}) = \mathbf{g}'(\mathbf{x})$ and solve this equation for $\mathbf{g}(\mathbf{x})$ (by integrating; this is also a choice based on the ease of integrating the second factor $\mathbf{h}(\mathbf{x})$ in the given integral). Having $\mathbf{g}(\mathbf{x})$ in hand we can then write out the result using the indefinite integral integration by parts rule above. We can formalize this process for an *indefinite integral* with the Maxima code:

```
(%i1) iparts(f,h,var):= block([g ],
         g : integrate(h,var),
         f*g - 'integrate(g*diff(f,var),var ) )$
```

Let's practice with the integral $\int \mathbf{x^2}\,\mathbf{sin}(\mathbf{x})\,\mathbf{dx}$, in which $\mathbf{f}(\mathbf{x}) = \mathbf{x^2}$ and $\mathbf{h}(\mathbf{x}) = \mathbf{sin}(\mathbf{x})$, so we need to be able to integrate $\mathbf{sin}(\boldsymbol{x})$ and want to transfer a derivative on to $\mathbf{x^2}$, which will reduce the first factor to $\mathbf{2\,x}$. Notice that it is usually easier to work with "Maxima expressions" rather than with "Maxima functions" in a problem like this.

```
(%i2) iparts(x^2,sin(x),x);
                              /
                              [                  2
(%o2)                    2 I x cos(x) dx - x  cos(x)
                              ]
                              /
(%i3) (ev(%,nouns),expand(%%) );
                                        2
(%o3)                 2 x sin(x) - x  cos(x) + 2 cos(x)
(%i4) collectterms(%,cos(x) );
                                             2
(%o4)                   2 x sin(x) + (2 - x ) cos(x)
```

If we were not using Maxima, but doing everything by hand, we would use two integrations by parts (in succession) to remove the factor $\mathbf{x^2}$ entirely, reducing the original problem to simply knowing the integrals of $\mathbf{sin}(\boldsymbol{x})$ and $\mathbf{cos}(\boldsymbol{x})$. Of course, with an integral as simple as this example, there is no need to help Maxima out by integrating by parts.

```
(%i5) integrate(x^2*sin(x),x);
                                             2
(%o5)                   2 x sin(x) + (2 - x ) cos(x)
```

We can write a similar Maxima function to transform *definite integrals* via integration by parts.

```
(%i6) idefparts(f,h,var,v1,v2):= block([g ],
         g : integrate(h,var),
           'subst(v2,var, f*g)  - 'subst(v1,var, f*g ) -
               'integrate(g*diff(f,var),var,v1,v2 ) );
(%o6) idefparts(f, h, var, v1, v2) :=
block([g], g : integrate(h, var), 'substitute(v2, var, f g)
                                   v2
                                  /
                                  [
 - 'substitute(v1, var, f g) - I   g diff(f, var) dvar)
                                  ]
                                  /
                                   v1
(%i7) idefparts(x^2,sin(x),x,0,1);
          1
         /
         [                                        2
(%o7) 2 I  x cos(x) dx + substitute(1, x, - x  cos(x))
         ]
         /
          0
                                                                  2
                                                    - substitute(0, x, - x  cos(x))
(%i8) (ev(%,nouns),expand(%%) );
(%o8)                        2 sin(1) + cos(1) - 2
(%i9) integrate(x^2*sin(x),x,0,1);
(%o9)                        2 sin(1) + cos(1) - 2
```

7.10 Change of Variable and changevar(..)

Many integrals can be evaluated most easily by making a change of variable of integration. A simple example is:

$$\int \mathbf{2\,x\,(x^2+1)^3\,dx} = \int \mathbf{(x^2+1)^3\,d\,(x^2+1)} = \int \mathbf{u^3\,du} = \mathbf{u^4/4} = \mathbf{(x^2+1)^4/4} \tag{7.38}$$

There is a function in Maxima, called **changevar** which will help you change variables in a one-dimensional integral (either indefinite or definite). However, this function is buggy at present and it is safer to do the change of variables "by hand".

> Function: **changevar(integral-expression, g(x,u), u, x)**
> Makes the change of variable in a given *noun form* **integrate** expression such that the old variable of integration is $\mathbf{x}$, the new variable of integration is $\mathbf{u}$, and $\mathbf{x}$ and $\mathbf{u}$ are related by the equation $\mathbf{g(x,u) = 0}$.

Example 1

Here we use this Maxima function on the simple indefinite integral $\int \mathbf{2\,x\,(x^2+1)^3\,dx}$ we have just done "by hand":

```
(%i1) expr : 'integrate(2*x*(x^2+1)^3,x);
                                /
                                [      2      3
(%o1)                         2 I x (x  + 1)  dx
                                ]
                                /
(%i2) changevar(expr,x^2+1-u,u,x);
                                    /
                                    [  3
(%o2)                               I u  du
                                    ]
                                    /
(%i3) ev(%, nouns);
                                       4
                                      u
(%o3)                                 --
                                      4
(%i4) ratsubst(x^2+1,u,%);
                           8       6       4       2
                          x  + 4 x  + 6 x  + 4 x  + 1
(%o4)                     ---------------------------
                                      4
(%i5) subst(x^2+1,u,%o3);
                                    2      4
                                   (x  + 1)
(%o5)                              ---------
                                       4
(%i6) subst(u=(x^2+1),%o3);
                                    2      4
                                   (x  + 1)
(%o6)                              ---------
                                       4
```

The original indefinite integral is a function of $\mathbf{x}$, which we obtain by replacing $\mathbf{u}$ by its equivalent as a function of $\mathbf{x}$. We have shown three ways to make this replacement to get a function of $\mathbf{x}$, using **subst** and **ratsubst**.

Example 2

As a second example, we use a change of variables to find $\int \left[(\mathbf{x+2})/\sqrt{\mathbf{x+1}}\right]\,\mathbf{dx}$.

```
(%i7) expr : 'integrate( (x+2)/sqrt(x+1),x);
                                /
                                [    x + 2
(%o7)                           I ----------- dx
                                ] sqrt(x + 1)
                                /
```

```
(%i8) changevar(expr,u - sqrt(x+1),u,x);
Is  u  positive, negative, or zero?

pos;
                                  /
                                  [      2
(%o8)                             I (2 u  + 2) du
                                  ]
                                  /
(%i9) ev(%, nouns);
                                       3
                                    2 u
(%o9)                               ---- + 2 u
                                     3
(%i10) subst(u = sqrt(x+1),%);
                                    3/2
                          2 (x + 1)
(%o10)                    ------------ + 2 sqrt(x + 1)
                               3
```

Of course Maxima can perform the original integral in these two examples without any help, as in:

```
(%i11) integrate((x+2)/sqrt(x+1),x);
                                  3/2
                             (x + 1)
(%o11)                    2 (---------- + sqrt(x + 1))
                                  3
```

However, there are occasionally cases in which you can help Maxima find an integral (for which Maxima can only return the noun form) by first making your own change of variables and then letting Maxima try again.

Example 3

Here we use **changevar** with a *definite* integral, using the same integrand as in the previous example. For a definite integral, the variable of integration is a ”dummy variable”, and the result is not a function of that dummy variable, so there is no issue about replacing the new integration variable **u** by the original variable **x** in the result.

```
(%i12) expr : 'integrate( (x+2)/sqrt(x+1),x,0,1);
                                  1
                                 /
                                 [     x + 2
(%o12)                           I  ----------- dx
                                 ]  sqrt(x + 1)
                                 /
                                  0
(%i13) changevar(expr,u - sqrt(x+1),u,x);
Is  u  positive, negative, or zero?

pos;
                                sqrt(2)
                               /
                               [             2
(%o13)                         I         (2 u  + 2) du
                               ]
                               /
                                1
(%i14) ev(%, nouns);
                                   10 sqrt(2)   8
(%o14)                             ---------- - -
                                       3        3
```

Example 4

We next discuss an example which shows that one needs to pay attention to the possible introduction of obvious sign errors when using **changevar**. The example is the evaluation of the definite integral $\int_0^1 \mathbf{e^{y^2}}\,\mathbf{dy}$, in which $\mathbf{y}$ is a real variable. Since the integrand is a positive (real) number over the interval $\mathbf{0} < \mathbf{y} \leq \mathbf{1}$, the definite integral must be a positive (real) number. The answer returned directly by Maxima's **integrate** function is:

```
(%i15) i1:integrate (exp (y^2),y,0,1);
                              sqrt(%pi) %i erf(%i)
(%o15)                      - --------------------
                                        2
```

Maxima's symbol **erf(z)** represents the error function $\mathbf{Erf(z)}$. We have discussed the Maxima function **erf(x)** for real $\mathbf{x}$ in Example 4 in Sec.(7.2). Here we have a definite integral result returned in terms of `erf(%i)`, which is the error function with a pure imaginary agument. At the time of writing this section, using Maxima version 5.17.1, intensive work is being done to strengthen the abilities to evaluate special functions for complex numerical arguments. At this time, we need to exert extra effort to turn this symbolic result into a floating point number, since we have the present behavior:

```
(%i16) erfilist:[erf(%i),erf(%i*1.0)];
(%o16)      [erf(%i), 1.650425758797543 %i + 3.231006984618108 1E-16]
(%i17) fpprec:100$
(%i18) map('bfloat,erfilist);
(%o18) [erf(%i), 1.650425758797543052125433860055636614561080932617187 5b0 %i + \
3.231006984618108065163177816255110324974642228325338866312677055248059332 3707\
58056640625b-16]
(%i19) map('realpart,erfilist);
(%o19)                      [0, 3.231006984618108 1E-16]
(%i20) map('imagpart,erfilist);
(%o20)                  [- %i erf(%i), 1.650425758797543]
```

Thus we need to replace `erf(%i)` by `erf(%i*1.0)` to turn `i1` into a numerical value.

```
(%i21) i2:subst(%i*1.0,%i,i1);
(%o21) - 0.5 sqrt(%pi) %i (1.650425758797543 %i + 3.231006984618108 1E-16)
(%i22) i3:bfloat (expand (i2));
(%o22) 1.462651745907181764868695352520057210951879265828646745064176000807 1408\
472584549847007715283379499 48b0 - 2.863405386094492514349272973670212113472663\
771667665920503055325088531196242919146050092449869801336b-16 %i
```

Neglecting the floating point error, we then have `i1 = 1.463` approximately.

We can confirm this numerical value using the quadrature routine **quad_qags**:

```
(%i23) quad_qags(exp(y^2),y,0,1);
(%o23)        [1.462651745907182, 1.6238696453143376E-14, 21, 0]
```

Let's ask Maxima to change variables from $\mathbf{y}$ to $\mathbf{u} = \mathbf{y^2}$ in the following way:

```
(%i24) expr : 'integrate(exp(y^2),y,0,1);
                                  1
                                 /
                                 [    2
                                 [   y
(%o24)                           I  %e   dy
                                 ]
                                 /
                                  0
```

```
(%i25) changevar(expr,y^2-u,u,y);
                                    1
                                   /       u
                                   [     %e
                                   I   ------- du
                                   ]   sqrt(u)
                                   /
                                    0
(%o25)                          - -------------
                                          2
(%i26) ev(%, nouns);
                                sqrt(%pi) %i erf(%i)
(%o26)                          --------------------
                                         2
```

which is the negative of the correct result. Evidently, Maxima uses **solve**$(\mathbf{y^2 = u}, \mathbf{y})$ to find $\mathbf{y(u)}$ and even though there are two solutions $\mathbf{y = \pm\sqrt{u}}$, Maxima picks the wrong solution without asking the user a clarifying question.

We need to force Maxima to use the correct relation between $\mathbf{y}$ and $\mathbf{u}$, as in:

```
(%i27) changevar(expr,y-sqrt(u),u,y);
Is  y  positive, negative, or zero?

pos;
                                    1
                                   /       u
                                   [     %e
                                   I   ------- du
                                   ]   sqrt(u)
                                   /
                                    0
(%o27)                             -------------
                                          2
(%i28) ev(%, nouns);
                                  sqrt(%pi) %i erf(%i)
(%o28)                          - --------------------
                                          2
```

which is now the correct result with the correct sign.

Example 5

We now discuss an example of a change of variable in which **changevar** produces the wrong overall sign, even though we try to be very careful. We consider the indefinite integral $\int \left(\mathbf{x}/\sqrt{\mathbf{x^2 - 4}} \right) \, \mathbf{dx}$, which **integrate** returns as:

```
(%i1) integrate(x/sqrt(x^2-4),x);
                                        2
(%o1)                             sqrt(x  - 4)
```

Now consider the change of variable $\mathbf{x \rightarrow t}$ with $\mathbf{x = 2/\cos(t)}$.

We will show first the **changevar** route (with its error) and then how to do the change of variables "by hand", but with Maxima'a assistance. Here we begin with assumptions about the variables involved.

```
(%i2) assume(x > 2, t > 0, t < 1.5, cos(t) > 0, sin(t) > 0 );
(%o2)           [x > 2, t > 0, t < 1.5, cos(t) > 0, sin(t) > 0]
```

```
(%i3) nix : 'integrate(x/sqrt(x^2-4),x);
                                /
                                [      x
(%o3)                           I ------------ dx
                                ]       2
                                / sqrt(x  - 4)
(%i4) nixt : changevar(nix,x-2/cos(t), t, x) ;
                    /
                    [                   sin(t)
(%o4)         - 2 %i I ------------------------------------------ dt
                    ]                      2
                    / sqrt(cos(t) - 1) cos (t) sqrt(cos(t) + 1)
(%i5) nixt : rootscontract(nixt);
                             /
                             [          sin(t)
(%o5)                 - 2 %i I ------------------------ dt
                             ]    2             2
                             / cos (t) sqrt(cos (t) - 1)
(%i6) nixt : scanmap('trigsimp,nixt);
                                 /
                                 [    1
(%o6)                          - 2 I ------- dt
                                 ]    2
                                 / cos (t)
(%i7) ev(nixt,nouns);
(%o7)                             - 2 tan(t)
```

Since we have assumed $\mathbf{t > 0}$, we have $\mathbf{\tan(t) > 0}$, so **changevar** is telling us the indefinite integral is a negative number for the range of $\mathbf{t}$ assumed.

Since we are asking for an indefinite integral, and we want the result in terms of the original variable $\mathbf{x}$, we would need to do some more work on this answer, maintaing the assumptions we have made. We will do that work after we have repeated this change of variable, doing it "by hand".

We work on the product $\mathbf{f(x)\,dx}$:

```
(%i8) ix : subst(x=2/cos(t),x/sqrt(x^2 - 4) )* diff(2/cos(t));
                               4 sin(t) del(t)
(%o8)                     ------------------------
                                  4            3
                          sqrt(------- - 4) cos (t)
                                  2
                               cos (t)
(%i9) ix : trigsimp(ix);
                                   2 del(t)
(%o9)                            - -----------
                                     2
                                   sin (t) - 1
(%i10) ix : ratsubst(1,cos(t)^2+sin(t)^2,ix);
                                   2 del(t)
(%o10)                             --------
                                     2
                                   cos (t)
(%i11) integrate(coeff(ix,del(t) ) ,t);
(%o11)                             2 tan(t)
```

which is the result **changevar** should have produced.

Now let's show how we can get back to the result produced by **integrate**.

```
(%i12) subst(tan(t)= sqrt(sec(t)^2-1),2*tan(t) );
                                        2
(%o12)                          2 sqrt(sec (t) - 1)
(%i13) subst(sec(t)=1/cos(t),%);
                                        1
(%o13)                          2 sqrt(------- - 1)
                                        2
                                     cos (t)
(%i14) subst(cos(t)=2/x,%);
                                        2
                                       x
(%o14)                            2 sqrt(-- - 1)
                                        4
(%i15) ratsimp(%);
                                        2
(%o15)                             sqrt(x  - 4)
```

7.11 Fourier Series Expansion of a Function

7.11.1 Expansion of a Function over $(-\pi,\ \pi)$

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

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} + \sum_{\mathbf{n=1}}^{\infty} [\mathbf{a_n}\,\mathbf{cos(n\,x)} + \mathbf{b_n}\,\mathbf{sin(n\,x)}] \tag{7.39}$$

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

$$\mathbf{a_n} = \frac{\mathbf{1}}{\boldsymbol{\pi}} \int_{-\boldsymbol{\pi}}^{\boldsymbol{\pi}} \mathbf{f(y)}\,\mathbf{cos(n\,y)}\,\mathbf{dy}, \quad \mathbf{b_n} = \frac{\mathbf{1}}{\boldsymbol{\pi}} \int_{-\boldsymbol{\pi}}^{\boldsymbol{\pi}} \mathbf{f(y)}\,\mathbf{sin(n\,y)}\,\mathbf{dy}. \tag{7.40}$$

(For a derivation of these equations see Sec.7.11.8 and Eqs.(7.58) and (7.59).)

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

The first term of the expansion

$$\frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} = \frac{\mathbf{1}}{\mathbf{2\,\pi}} \int_{-\boldsymbol{\pi}}^{\boldsymbol{\pi}} \mathbf{f(y)}\,\mathbf{dy} = \langle \mathbf{f(x)} \rangle \tag{7.41}$$

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

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

If you are trying to find a fourier expansion for a function or expression **f(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 Tools.

7.11.2 Fourier Expansion of $\mathbf{f(x) = x}$ over $(-\pi, \pi)$

We first use the coefficient formulas Eq.(7.40) to find the Fourier expansions of this simple linear function $\mathbf{f(x) = x}$ by hand.

Because the average value of **f(x)** over the domain $(-\pi, \pi)$ is zero, $\mathbf{a_0 = 0}$. Because **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 $\mathbf{2\sin(x)}$ which retains only the $\mathbf{n = 1}$ term in the expansion, the two term approximation $\mathbf{2\sin(x) - \sin(2x)}$, which includes the $\mathbf{n = 1}$, **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 $\mathbf{f(x) = x}$ 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 $\mathbf{f(x) = x}$ (see the figure on the next page).

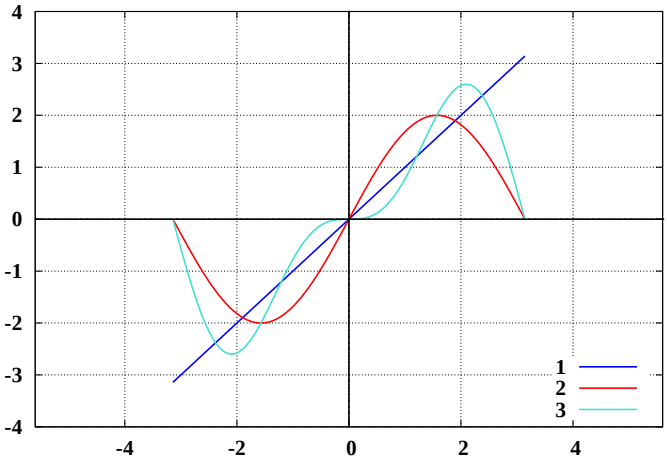

Figure 8: One and Two Term Approximations

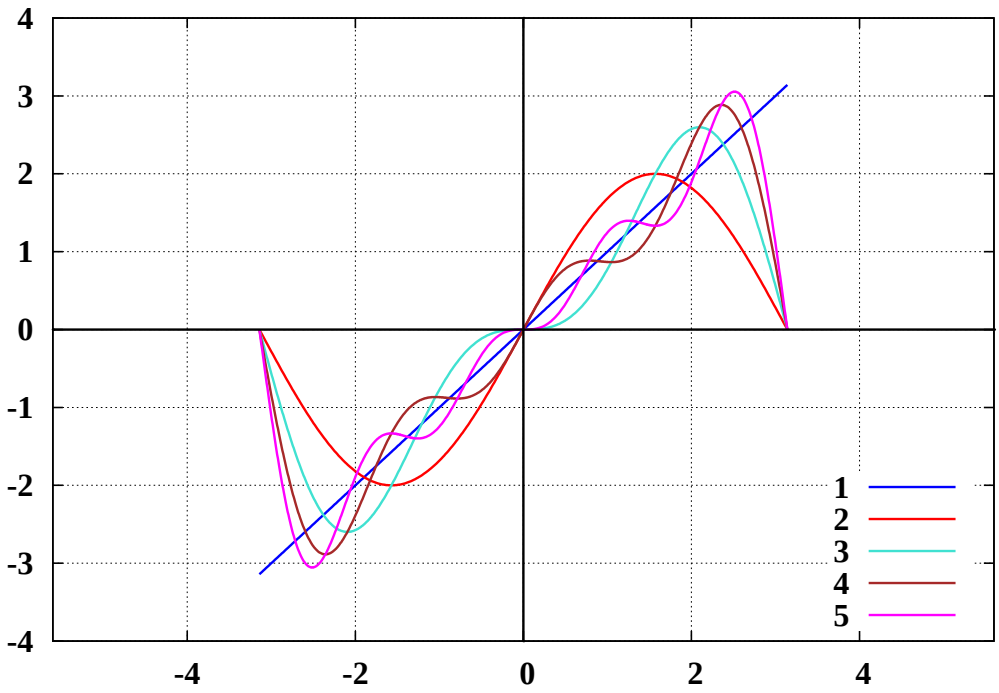

Figure 9: 1, 2, 3, and 4 Term Approximations

7.11.3 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 $\mathbf{n}$ 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 $\mathbf{n}$ to be a local variable, and then have the statement (inside the **block**) **assume**$(\mathbf{n} > \mathbf{0})$, 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 $\mathbf{cos}(\mathbf{n}\,\pi\,\mathbf{x}/\mathbf{p})$ or $\mathbf{sin}(\mathbf{n}\,\pi\,\mathbf{x}/\mathbf{p})$, (where $\mathbf{p}$ will be π for our expansion domain here).

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 $\mathbf{a_0}$ and $\mathbf{a_n}$, with the second meant for $\mathbf{n} = \mathbf{1}, \mathbf{2}, \ldots$, and there is never an output for $\mathbf{b_0}$. 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 $\mathbf{n} > \mathbf{0}$ becomes a global fact.

Also note that the **fourie.mac** package does not define a Maxima function $\mathbf{b(n)}$, although you could using:

```
(%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 $\mathbf{f(x) = x}$, **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.

7.11.4 Expansion Over $(-\mathbf{p}, \mathbf{p})$

The Fourier series expansion of a function defined over the interval $-\mathbf{p} \le \mathbf{x} \le \mathbf{p}$ (and whose Fourier expansion will represent a function which has a period $\mathbf{2\,p}$) can be found from the expansion over the interval $(-\boldsymbol{\pi}, \boldsymbol{\pi})$ which we have been using above by a simple change of variables in the integrals which appear in Eqs.(7.40) and (7.39).

However, we will simply use the results derived in Sec.7.11.8, and written down in Eqns (7.56) and (7.57).

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} + \sum_{\mathbf{n=1}}^{\infty}\left[\mathbf{a_n}\cos\left(\frac{\boldsymbol{\pi}\,\mathbf{n\,x}}{\mathbf{p}}\right) + \mathbf{b_n}\sin\left(\frac{\boldsymbol{\pi}\,\mathbf{n\,x}}{\mathbf{p}}\right)\right] \tag{7.42}$$

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

$$\mathbf{a_n} = \frac{\mathbf{1}}{\mathbf{p}}\int_{\mathbf{-p}}^{\mathbf{p}} \mathbf{f(y)}\cos\left(\frac{\boldsymbol{\pi}\,\mathbf{n\,y}}{\mathbf{p}}\right)\,\mathbf{dy}, \quad \mathbf{b_n} = \frac{\mathbf{1}}{\mathbf{p}}\int_{\mathbf{-p}}^{\mathbf{p}} \mathbf{f(y)}\sin\left(\frac{\boldsymbol{\pi}\,\mathbf{n\,y}}{\mathbf{p}}\right)\,\mathbf{dy}. \tag{7.43}$$

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

Our $\mathbf{a_0}$ is defined by

$$\mathbf{a_0} = \frac{\mathbf{1}}{\mathbf{p}}\int_{\mathbf{-p}}^{\mathbf{p}} \mathbf{f(x)}\,\mathbf{dx} \tag{7.44}$$

The **fourie.mac** package's definition of *their* $\mathbf{a_0}$ is

$$[\mathbf{a_0}]_{\mathbf{fourie.mac}} = \frac{\mathbf{1}}{\mathbf{2\,p}}\int_{\mathbf{-p}}^{\mathbf{p}} \mathbf{f(x)}\,\mathbf{dx} \tag{7.45}$$

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

7.11.5 Fourier Series Expansion of $|\mathbf{x}|$

We define the function to have period $\mathbf{4}$ with $\mathbf{f(x)} = |\mathbf{x}|$ for $\mathbf{-2 \le x \le 2}$. This function is an even function of $\mathbf{x}$ so the $\mathbf{sin}$ coefficients $\mathbf{b_n}$ are all zero. The average value of $|\mathbf{x}|$ over the domain $(\mathbf{-2}, \mathbf{2})$ is greater than zero so we willl have a non-zero coefficient $\mathbf{a_0}$ 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: $-\mathbf{2} \leq \mathbf{x} \leq \mathbf{0}$, in which $|\mathbf{x}| = -\mathbf{x}$, and the interval $\mathbf{0} \leq \mathbf{x} \leq \mathbf{2}$ in which $|\mathbf{x}| = \mathbf{x}$. We will use the formula Eq. (7.43) for the coefficients $\mathbf{a_n}$ (note that $\mathbf{a_n} = \mathbf{0}$ if $\mathbf{n}$ is even,) and the expansion formula Eq. (7.42).

```
(%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 $|\mathbf{x}|$, represented by **abs(x)**, which appears in the color blue. We see that the expansion out to $\mathbf{n = 5}$ provides a close fit to $|\mathbf{x}|$. Here is that comparison:

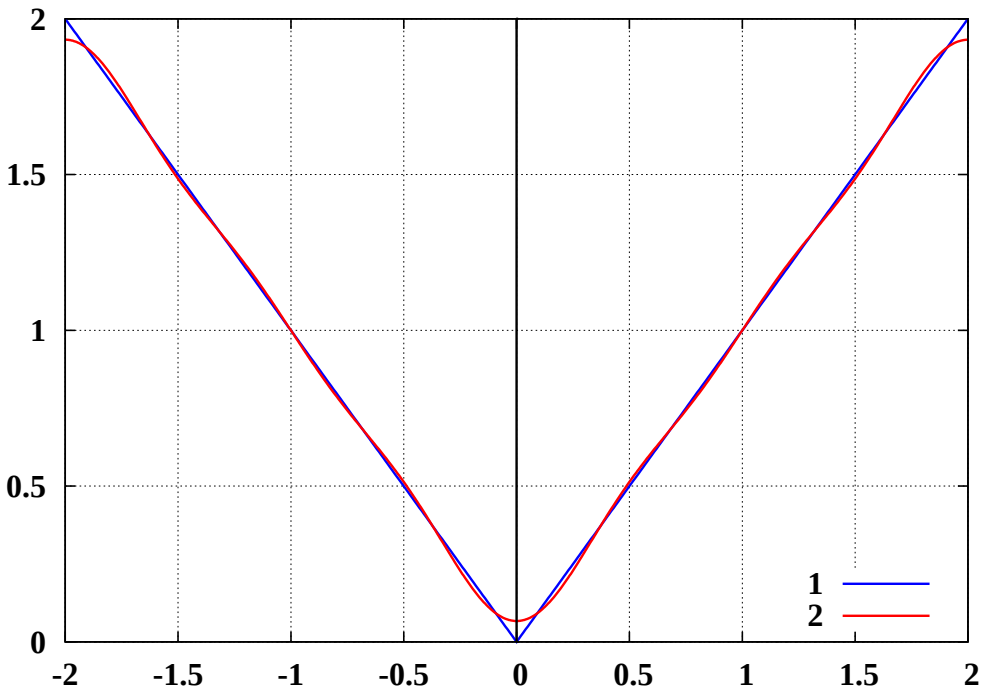

Figure 10: $\mathbf{n = 5}$ Approximation to $|\mathbf{x}|$

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 $(\mathbf{a_0})_{\mathbf{fourie}} = \mathbf{1} = \frac{\mathbf{1}}{\mathbf{2}}\,(\mathbf{a_0})_{\mathbf{ourdef}}$, (see Eq. (7.45)) so that fourie.mac's expansion starts off with the term $\mathbf{a_0}$ rather than $\frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0}$. Of course, the actual end results look the same, with the first term in this example being $\mathbf{1}$, which is the average value of $|\mathbf{x}|$ over the domain $(-\mathbf{2}, \mathbf{2})$.

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".

7.11.6 Fourier Series Expansion of a Rectangular Pulse

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

```
(%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 $\mathbf{3/2}$ above the **x** axis and with a width of **2** units over the interval $\mathbf{-1 \leq x \leq 1}$. Over the rest of the domain $\mathbf{-2 \leq x \leq 2}$, **f(x)** is defined to be zero. Although we can

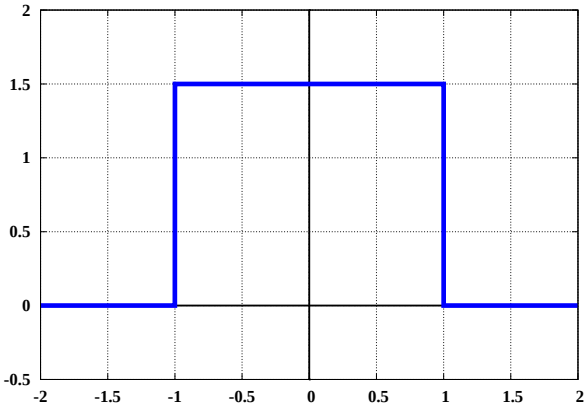

Figure 11: Rectangular Pulse of Height $3/2$

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 ($\mathbf{f(-x) = f(x)}$), so only the $\mathbf{\cos(n\,\pi\,x/2)}$ terms will contribute, so we only need to calculate the $\mathbf{a_n}$ 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 $\mathbf{n} > \mathbf{0}$, $\mathbf{a_n} = \mathbf{0}$ for $\mathbf{n}$ even, and the non-zero coefficients have $\mathbf{n} = \mathbf{1}, \mathbf{3}, \mathbf{5}, \mathbf{7}, \ldots$. Hence we only get a better approximation if we increase $\mathbf{n_{max}}$ by $\mathbf{2}$ 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.

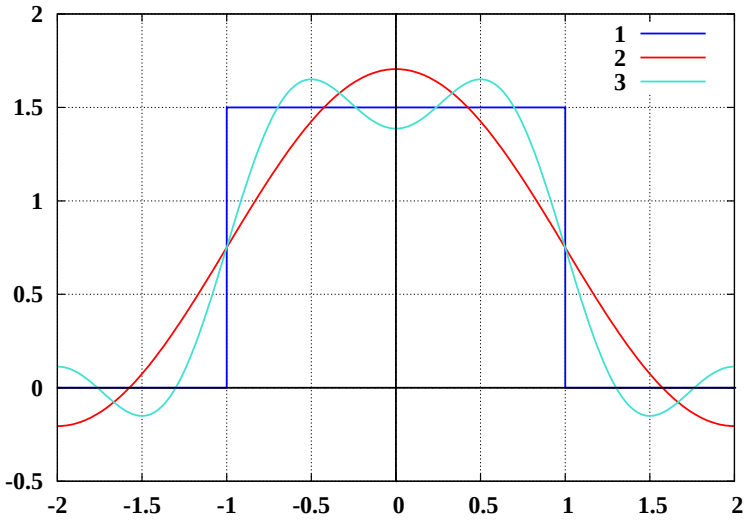

Figure 12: $\mathbf{nmax} = \mathbf{1}, \mathbf{3}$ Approx. to Rectangular Pulse

Then a plot showing the `fs(11)` approximation:

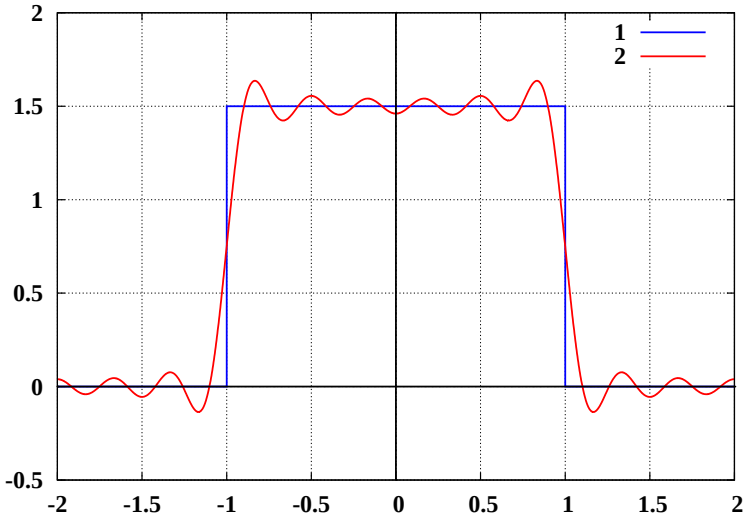

Figure 13: $\mathbf{nmax} = \mathbf{11}$ Approx. to Rectangular Pulse

7.11.7 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 $-10 \leq x < -5$, $f = 0$, and for $-5 \leq x < 0$, $f = -5$, and for $0 \leq x \leq 5$, $f = 5$, and for $5 < x \leq 10$, $f = 0$. 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

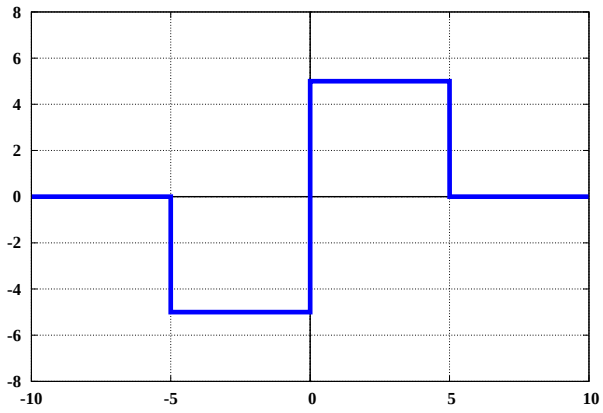

Figure 14: Two Element Pulse with Period 20

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 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

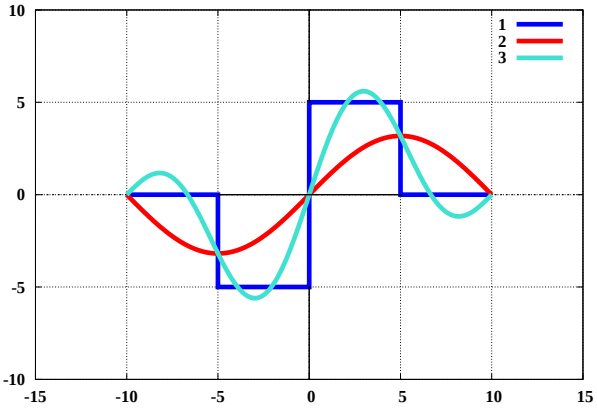

Figure 15: Two Lowest Order Approximations

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 ) )$
```

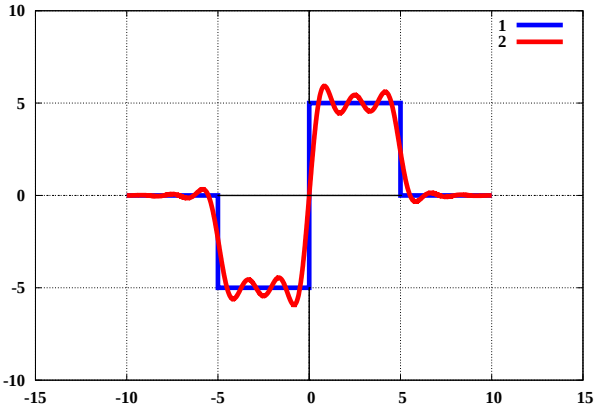

Figure 16: $\mathbf{nmax} = \mathbf{11}$ Approx. to Two Element Pulse

Whether you are working with a periodic function or not, the Fourier series expansion always represents a periodic function for all $\mathbf{x}$. For this example, the period is $\mathbf{20}$, 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

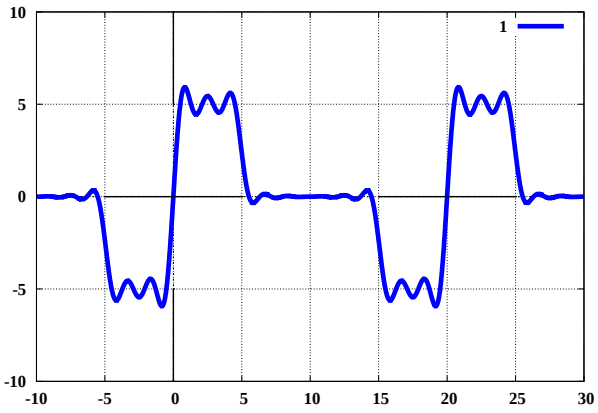

Figure 17: $\mathbf{nmax = 11}$ Approx. Drawn for Two Periods

7.11.8 Exponential Form of a Fourier Series

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

$$\left\{\mathbf{exp}\left(\frac{\mathbf{2\,\pi\,i\,n\,x}}{\mathbf{b-a}}\right)\right\} \qquad (\mathbf{n = 0,\ \pm 1,\ \pm 2,\ \cdots}) \tag{7.46}$$

on the interval $(\mathbf{a},\ \mathbf{b})$. Define the function $\boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x})$ as

$$\boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x}) = \mathbf{exp}\left(\frac{\mathbf{-2\,\pi\,i\,n\,x}}{\mathbf{b-a}}\right). \tag{7.47}$$

This function has the properties

$$\boldsymbol{\phi}_{\mathbf{-n}}(\mathbf{x}) = \boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x})^{*}, \qquad \boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x})\,\boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x})^{*} = \mathbf{1}. \tag{7.48}$$

in which the asterisk indicates the complex conjugate.

The (Hermitian) orthogonality of the $\{\boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x})\}$ over $(\mathbf{a},\ \mathbf{b})$ is expressed by

$$\int_{\mathbf{a}}^{\mathbf{b}} \boldsymbol{\phi}_{\mathbf{n}}(\mathbf{x})\,\boldsymbol{\phi}_{\mathbf{m}}(\mathbf{x})^{*}\,\mathbf{dx} = (\mathbf{b-a})\,\boldsymbol{\delta}_{\mathbf{n\,m}}, \tag{7.49}$$

in which the Kronecker delta symbol is

$$\boldsymbol{\delta}_{\mathbf{n\,m}} = \left\{ \begin{array}{lll} \mathbf{1} & \text{for} & \mathbf{n = m} \\ \mathbf{0} & \text{for} & \mathbf{n \neq m.} \end{array} \right\} \tag{7.50}$$

Eq.(7.49) is clearly true for $\mathbf{n = m}$, using Eq.(7.48). For $\mathbf{m \neq n}$, we can simplify this integral by using the exponential product law $\mathbf{e^{A}\,e^{B} = e^{A+B}}$, letting $\mathbf{r = m - n \neq 0}$, and changing the variable of integration $\mathbf{x \rightarrow y}$ via $\mathbf{x = (b-a)\,y + a}$.

The resulting integral in terms of $\mathbf{y}$ will then be over the interval $(0,1)$. The differential $\mathbf{dx} \rightarrow (\mathbf{b}-\mathbf{a})\,\mathbf{dy}$ and $(\mathbf{b}-\mathbf{a})$ comes outside the integral. Also, inside the exponential, $\mathbf{x}/(\mathbf{b}-\mathbf{a}) \rightarrow \mathbf{a}/(\mathbf{b}-\mathbf{a}) + \mathbf{y}$, and we can take outside an exponential function of a constant. Thus the integral is proportional to $\int_0^1 \mathbf{exp}(\mathbf{2\,\pi\,i\,r\,y})\,\mathbf{dy}$, which is proportional to $\mathbf{exp}(\mathbf{2\,\pi\,i\,r}) - \mathbf{1} = \mathbf{exp}(\mathbf{2\,\pi\,i})^{\mathbf{r}} - \mathbf{1} = \mathbf{0}$.

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

$$\mathbf{f(x)} = \sum_{\mathbf{n}=-\infty}^{\infty} \mathbf{C_n}\,\boldsymbol{\phi_n}(\mathbf{x}). \tag{7.51}$$

To find the $\{\mathbf{C_n}\}$, we multiply both sides of Eq.(7.51) by $\boldsymbol{\phi_m}(\mathbf{x})^*\,\mathbf{dx}$, integrate both sides over the interval $(\mathbf{a},\,\mathbf{b})$, and use orthogonality, Eq.(7.49).

$$\begin{aligned}
\int_a^b \mathbf{f(x)}\,\boldsymbol{\phi_m}(\mathbf{x})^*\,\mathbf{dx} &= \sum_{\mathbf{n}=-\infty}^{\infty} \mathbf{C_n} \int_a^b \boldsymbol{\phi_n}(\mathbf{x})\,\boldsymbol{\phi_m}(\mathbf{x})^*\,\mathbf{dx} \\
&= \sum_{\mathbf{n}=-\infty}^{\infty} \mathbf{C_n}\,(\mathbf{b}-\mathbf{a})\,\boldsymbol{\delta_{n\,m}} \\
&= (\mathbf{b}-\mathbf{a})\,\mathbf{C_m}.
\end{aligned}$$

Hence we have for the coefficients

$$\mathbf{C_n} = \frac{\mathbf{1}}{(\mathbf{b}-\mathbf{a})} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(x)}\,\boldsymbol{\phi_n}(\mathbf{x})^*\,\mathbf{dx}. \tag{7.52}$$

Inserting these coefficients into Eq.(7.51) we can write

$$\begin{aligned}
\mathbf{f(x)} &= \sum_{\mathbf{n}=-\infty}^{\infty} \mathbf{C_n}\,\boldsymbol{\phi_n}(\mathbf{x}) \\
&= \sum_{-\infty}^{\infty} \mathbf{exp}\left(\frac{\mathbf{-2\,\pi\,i\,n\,x}}{\mathbf{b}-\mathbf{a}}\right) \frac{\mathbf{1}}{\mathbf{b}-\mathbf{a}} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)}\,\mathbf{exp}\left(\frac{\mathbf{2\,\pi\,i\,n\,y}}{\mathbf{b}-\mathbf{a}}\right)\,\mathbf{dy} \\
&= \frac{\mathbf{1}}{\mathbf{b}-\mathbf{a}} \sum_{-\infty}^{\infty} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)}\,\mathbf{exp}\left(\frac{\mathbf{2\,\pi\,i\,n\,(y-x)}}{\mathbf{b}-\mathbf{a}}\right)\,\mathbf{dy}.
\end{aligned}$$

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

$$\begin{aligned}
\mathbf{f(x)} &= \frac{\mathbf{1}}{\mathbf{b}-\mathbf{a}} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)}\,\mathbf{dy} + \frac{\mathbf{1}}{\mathbf{b}-\mathbf{a}} \sum_{\mathbf{n}=\mathbf{1}}^{\infty} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)}\,\left[\mathbf{exp}\left(\frac{\mathbf{2\,\pi\,i\,n\,(y-x)}}{\mathbf{b}-\mathbf{a}}\right) + \mathbf{exp}\left(\frac{\mathbf{-2\,\pi\,i\,n\,(y-x)}}{\mathbf{b}-\mathbf{a}}\right)\right]\,\mathbf{dy} \\
&= \frac{\mathbf{1}}{\mathbf{b}-\mathbf{a}} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)}\,\mathbf{dy} + \frac{\mathbf{2}}{\mathbf{b}-\mathbf{a}} \sum_{\mathbf{n}=\mathbf{1}}^{\infty} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)}\,\mathbf{cos}\left(\frac{\mathbf{2\,\pi\,n\,(y-x)}}{\mathbf{b}-\mathbf{a}}\right)\,\mathbf{dy}
\end{aligned}$$

Using the trig identity $\mathbf{cos(A-B) = cosA\,cosB + sinA\,sinB}$, 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 $\mathbf{x}$.

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} + \sum_{\mathbf{n}=\mathbf{1}}^{\infty} \left[\mathbf{a_n}\,\mathbf{cos}\left(\frac{\mathbf{2\,\pi\,n\,x}}{\mathbf{b}-\mathbf{a}}\right) + \mathbf{b_n}\,\mathbf{sin}\left(\frac{\mathbf{2\,\pi\,n\,x}}{\mathbf{b}-\mathbf{a}}\right)\right] \tag{7.53}$$

The coefficients are given by the integrals

$$\mathbf{a_n} = \frac{\mathbf{2}}{\mathbf{b-a}} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)} \cos\left(\frac{\mathbf{2\,\pi\,n\,y}}{\mathbf{b-a}}\right) \mathbf{dy}, \quad \mathbf{b_n} = \frac{\mathbf{2}}{\mathbf{b-a}} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)} \sin\left(\frac{\mathbf{2\,\pi\,n\,y}}{\mathbf{b-a}}\right) \mathbf{dy}. \tag{7.54}$$

The expansion starts with the term

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} + \cdots = \frac{\mathbf{1}}{\mathbf{b-a}} \int_{\mathbf{a}}^{\mathbf{b}} \mathbf{f(y)} + \cdots \tag{7.55}$$

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

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

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} + \sum_{\mathbf{n=1}}^{\infty} \left[\mathbf{a_n} \cos\left(\frac{\mathbf{\pi\,n\,x}}{\mathbf{p}}\right) + \mathbf{b_n} \sin\left(\frac{\mathbf{\pi\,n\,x}}{\mathbf{p}}\right)\right] \tag{7.56}$$

with coefficients

$$\mathbf{a_n} = \frac{\mathbf{1}}{\mathbf{p}} \int_{\mathbf{-p}}^{\mathbf{p}} \mathbf{f(y)} \cos\left(\frac{\mathbf{\pi\,n\,y}}{\mathbf{p}}\right) \mathbf{dy}, \quad \mathbf{b_n} = \frac{\mathbf{1}}{\mathbf{p}} \int_{\mathbf{-p}}^{\mathbf{p}} \mathbf{f(y)} \sin\left(\frac{\mathbf{\pi\,n\,y}}{\mathbf{p}}\right) \mathbf{dy}. \tag{7.57}$$

If we specialize further to $\mathbf{p} = \boldsymbol{\pi}$, the appropriate equations are

$$\mathbf{f(x)} = \frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{a_0} + \sum_{\mathbf{n=1}}^{\infty} \left[\mathbf{a_n} \cos(\mathbf{n\,x}) + \mathbf{b_n} \sin(\mathbf{n\,x})\right] \tag{7.58}$$

with coefficients

$$\mathbf{a_n} = \frac{\mathbf{1}}{\boldsymbol{\pi}} \int_{\mathbf{-\pi}}^{\boldsymbol{\pi}} \mathbf{f(y)} \cos(\mathbf{n\,y})\,\mathbf{dy}, \quad \mathbf{b_n} = \frac{\mathbf{1}}{\boldsymbol{\pi}} \int_{\mathbf{-\pi}}^{\boldsymbol{\pi}} \mathbf{f(y)} \sin(\mathbf{n\,y})\,\mathbf{dy}. \tag{7.59}$$

7.12 Fourier Integral Transform Pairs

7.12.1 Fourier Cosine Integrals and fourintcos(..)

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

$$\mathbf{F_C(f, \omega)} = \frac{\mathbf{2}}{\boldsymbol{\pi}} \int_{\mathbf{0}}^{\infty} \cos(\boldsymbol{\omega}\,\mathbf{x})\,\mathbf{f(x)}\,\mathbf{dx} \tag{7.60}$$

The given function $\mathbf{f(x)}$ can then be written as an integral over positive values of $\boldsymbol{\omega}$:

$$\mathbf{f(x)} = \int_{\mathbf{0}}^{\infty} \mathbf{F_C(f, \omega)}\ \cos(\boldsymbol{\omega}\,\mathbf{x})\,\mathbf{d\omega} \tag{7.61}$$

The two equations (7.60) and (7.61) are an example of a "Fourier transform pair", which include conventions about where to place the factor of $2/\boldsymbol{\pi}$.

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

```
(%i1) f:sin(x)*exp(-x)$
(%i3) integrate(f*cos(w*x),x,0,inf);
Is  w  positive, negative, or zero?
p;
Is  w - 1  positive, negative, or zero?
p;
                                      2
                                     w  - 2
(%o3)                              - ------
                                      4
                                     w  + 4
(%i4) fcw : (2/%pi)*%;
                                       2
                                  2 (w  - 2)
(%o4)                           - ------------
                                          4
                                  %pi (w  + 4)
(%i5) integrate(fcw*cos(w*x),w,0,inf);
Is  x  positive, negative, or zero?
p;
                                    - x
(%o5)                             %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 $\mathbf{a_z}$ which contains the Fourier cosine integral $\mathbf{F_C(f,\, z)}$, with the letter "z" being the package convention for what we called "w" ($\boldsymbol{\omega}$),

```
(%i6) load(fourie);
(%o6) C:/PROGRA~1/MAXIMA~1.1/share/maxima/5.16.1/share/calculus/fourie.mac
(%i7) fourintcos(f,x);
Is  z - 1  positive, negative, or zero?

p;
     2
Is  z  - 2 z + 2  positive or negative?

p;
                                                  2
                                      2          z
                                 2 (------ - ------)
                                      4          4
                                     z  + 4     z  + 4
(%t7)                      a  = --------------------
                            z            %pi

(%o7)                                 [%t7]
(%i8) az : ratsimp(rhs(%t7));
                                       2
                                  2 z  - 4
(%o8)                           - --------------
                                        4
                                %pi z  + 4 %pi
```

```
(%i9) (2/%pi)*ratsimp(%pi*az/2);
                                          2
                                       2 (z  - 2)
(%o9)                               - ------------
                                             4
                                      %pi (z  + 4)
```

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

7.12.2 Fourier Sine Integrals and fourintsin(..)

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

$$\mathbf{F_S(f,\omega) = \frac{2}{\pi} \int_0^\infty \sin(\omega\, x)\, f(x)\, dx} \tag{7.62}$$

The given function $\mathbf{f(x)}$ can then be written as an integral over positive values of $\boldsymbol{\omega}$:

$$\mathbf{f(x) = \int_0^\infty F_S(f,\omega)\, \sin(\omega\, x)\, d\omega} \tag{7.63}$$

The two equations (7.62) and (7.63) are another "Fourier transform pair", which include conventions about where to place the factor of $2/\boldsymbol{\pi}$.

Here is a simple example. We let the function be $\mathbf{f(x) = \cos(x)\, e^{-x}}$, defined for $\mathbf{x \geq 0}$. Let the letter "w" be used to stand for $\boldsymbol{\omega}$. 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 $\boldsymbol{\omega}$ gives back the original function.

```
(%i1) f:cos(x)*exp(-x)$
(%i2) assume(w>0)$
(%i3) integrate(f*sin(w*x),x,0,inf);
Is  w - 1  positive, negative, or zero?
p;
                                        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 $\mathbf{b_z}$ which contains the Fourier sine integral $\mathbf{F_S(f,\, z)}$, with the letter "z" being the package convention for what we called "w" ($\boldsymbol{\omega}$).

```
(%i6) load(fourie);
(%o6) C:/PROGRA~1/MAXIMA~1.1/share/maxima/5.16.1/share/calculus/fourie.mac
```

```
(%i7) facts();
(%o7)                                [w > 0]
(%i8) (forget(w>0),facts());
(%o8)                                  []
(%i9) fourintsin(f,x);
Is  z - 1  positive, negative, or zero?

p;
     2
Is  z  - 2 z + 2  positive or negative?

p;
                                          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.

7.12.3 Exponential Fourier Integrals and fourint(..)

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

$$\mathbf{F_{Exp}(f,\omega)} = \frac{\mathbf{1}}{\mathbf{2\,\pi}} \int_{-\infty}^{\infty} \mathbf{f(x) e^{i\omega x}\, dx} \tag{7.64}$$

The given function $\mathbf{f(x)}$ can then be written as an integral over both positive and negative values of $\boldsymbol{\omega}$:

$$\mathbf{f(x)} = \int_{-\infty}^{\infty} \mathbf{F_{Exp}(f,\omega)\, e^{-i\omega x}\, d\omega} \tag{7.65}$$

The two equations (7.64) and (7.65) are another "Fourier transform pair", which include conventions about where to place the factor of $\mathbf{2\,\pi}$ as well as which member has the minus sign in the exponent.

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

$$\mathbf{F_{Exp}(f,-\omega) = F_{Exp}(f,\omega)} \tag{7.66}$$

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

$$\mathbf{F_{Exp}(f,\omega)} = \frac{\mathbf{1}}{\mathbf{2}}\, \mathbf{F_C(f,\omega)}. \tag{7.67}$$

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

$$\mathbf{F_{Exp}(f,-\omega) = -F_{Exp}(f,\omega)} \tag{7.68}$$

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

$$\mathbf{F_{Exp}(f,\omega) = \frac{i}{2}\, F_S(f,\omega).} \tag{7.69}$$

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

$$\mathbf{f(x) \equiv f_e(x) + f_o(x) = \frac{1}{2}\,(f(x) + f(-x) + \frac{1}{2}\,(f(x) - f(-x)} \tag{7.70}$$

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

$$\mathbf{F_{Exp}(f,\omega) \equiv F_{Exp}(f_e + f_o, \omega) = \frac{1}{2}\, F_C(f_e,\omega) + \frac{i}{2}\, F_S(f_o,\omega)} \tag{7.71}$$

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

$$\mathbf{F_{Exp}(f,z) = \frac{1}{2}\, a_z + \frac{i}{2}\, b_z} \tag{7.72}$$

7.12.4 Example 1: Even Function

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

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

$$\mathbf{F_{Exp}}(\cos(\mathbf{x})\,\mathbf{exp}(-|\mathbf{x}|),\,\boldsymbol{\omega}) = \frac{\boldsymbol{\omega^2}+\mathbf{2}}{\boldsymbol{\pi}\,(\boldsymbol{\omega^4}+\mathbf{4})} \tag{7.73}$$

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

```
(%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 $\mathbf{x} \geq \mathbf{0}$ and confirm that the required exponential Fourier transform in this case is correctly given by $\frac{\mathbf{1}}{\mathbf{2}}\,\mathbf{F_C}(\mathbf{f},\boldsymbol{\omega})$.

```
(%i9) i3:integrate(cos(x)*exp(-x)*cos(w*x),x,0,inf);
Is  w  positive, negative, or zero?
p;
Is  w - 1  positive, negative, or zero?
p;
                                      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 $\mathbf{F_C}(\mathbf{f},\boldsymbol{\omega})$, and one half of that value is the required exponential Fourier transform.

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

```
(%i11) load(fourie);
(%o11) C:/PROGRA~1/MAXIMA~1.1/share/maxima/5.16.1/share/calculus/fourie.mac
(%i12) fourint(cos(x)*exp(-abs(x)),x);
Is  z - 1  positive, negative, or zero?
p;
     2
Is  z  - 2 z + 2  positive or negative?
p;
```

```
                                        2
                                       z         2
                                  2 (------ + ------)
                                      4         4
                                     z  + 4    z  + 4
(%t12)                      a  = --------------------
                             z            %pi

(%t13)                                b  = 0
                                       z

(%o13)                          [%t12, %t13]
(%i14) i4:ratsimp(rhs(%t12));
                                        2
                                     2 z  + 4
(%o14)                              --------------
                                        4
                                    %pi z  + 4 %pi
(%i15) (2/%pi)*ratsimp(%pi*i4/2);
                                        2
                                   2 (z  + 2)
(%o15)                             ------------
                                         4
                                   %pi (z  + 4)
```

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

7.12.5 Example 2: Odd Function

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

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

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

$$\mathbf{F_{Exp}}(\sin(\mathbf{x})\,\mathbf{exp}(-|\mathbf{x}|),\,\boldsymbol{\omega}) = \frac{\mathbf{2\,i\,\omega}}{\boldsymbol{\pi}\,(\boldsymbol{\omega}^{\mathbf{4}}+\mathbf{4})} \tag{7.74}$$

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

```
(%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 $\mathbf{x} \geq \mathbf{0}$ and confirm that the required exponential Fourier transform in this case is correctly given by $\frac{\mathbf{i}}{\mathbf{2}}\,\mathbf{F_S}(\mathbf{f},\boldsymbol{\omega})$.

```
(%i9) i3:integrate(sin(x)*exp(-x)*sin(w*x),x,0,inf);
Is  w  positive, negative, or zero?
p;
Is  w - 1  positive, negative, or zero?
p;
                                    2 w
(%o9)                               ------
                                     4
                                    w  + 4
(%i10) (2/%pi)*i3;
                                     4 w
(%o10)                           ------------
                                       4
                                 %pi (w  + 4)
```

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

Note the apparent bug in which **integrate** asks if `w` is positive, despite the **assume** database content.

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

```
(%i1) load(fourie);
(%o1) C:/PROGRA~1/MAXIMA~1.1/share/maxima/5.16.1/share/calculus/fourie.mac
(%i2) fourint(sin(x)*exp(-abs(x)),x);
(%t2)                                a  = 0
                                      z

Is  z - 1  positive, negative, or zero?
p;
```

```
     2
Is  z  - 2 z + 2  positive or negative?
p;
                                        4 z
(%t3)                           b  = ------------
                                 z          4
                                      %pi (z  + 4)

(%o3)                              [%t2, %t3]
```

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

7.12.6 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 $\mathbf{f(x) = \cos^2(x-1)\,e^{-|x|}}$ which is defined for $\mathbf{-\infty < x < \infty}$ and is neither even nor odd.

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

```
(%i1) ( assume(w>0), facts());
(%o1)                                [w > 0]
(%i2) i1:integrate(exp(%i*w*x)*cos(x-1)^2*exp(x),x,minf,0);
     2
Is  w  - 4 w + 5  positive or negative?
p;
Is  w - 2  positive, negative, or zero?
p;
                                  4                                    2
(%o2) ((2 sin(2) + cos(2) + 1) w  + (- 8 sin(2) + 6 cos(2) - 6) w  - 10 sin(2)
                        6       4       2
 + 5 cos(2) + 25)/(2 w  - 10 w  + 38 w  + 50)
                       5                                   3
 - (%i ((cos(2) + 1) w  + (- 4 sin(2) - 2 cos(2) - 6) w
                                          6       4       2
 + (- 4 sin(2) - 3 cos(2) + 25) w))/(2 w  - 10 w  + 38 w  + 50)
(%i3) i2:integrate(exp(%i*w*x)*cos(x-1)^2*exp(-x),x,0,inf);
     2
Is  w  - 4 w + 5  positive or negative?
p;
Is  w - 2  positive, negative, or zero?
p;
                           5                                   3
(%o3) (%i ((cos(2) + 1) w  + (4 sin(2) - 2 cos(2) - 6) w
                                       6       4       2
 + (4 sin(2) - 3 cos(2) + 25) w))/(2 w  - 10 w  + 38 w  + 50)
                                 4                                      2
 - ((2 sin(2) - cos(2) - 1) w  + (- 8 sin(2) - 6 cos(2) + 6) w  - 10 sin(2)
                        6       4       2
 - 5 cos(2) - 25)/(2 w  - 10 w  + 38 w  + 50)
(%i4) i12:ratsimp(i1+i2);
                       4                 3                     2
(%o4) ((cos(2) + 1) w  + 4 %i sin(2) w  + (6 cos(2) - 6) w  + 4 %i sin(2) w
                                                          6       4       2
                                       + 5 cos(2) + 25)/(w  - 5 w  + 19 w  + 25)
```

```
(%i5) i12: rectform(i12);
                    4                    2
      (cos(2) + 1) w  + (6 cos(2) - 6) w  + 5 cos(2) + 25
(%o5) ---------------------------------------------------
                     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
(%i6) i12i:ratsimp( imagpart(i12) );
                                  4 sin(2) w
(%o6)                           --------------
                                 4      2
                                w  - 6 w  + 25
(%i7) i12i:i12i/(2*%pi);
                                  2 sin(2) w
(%o7)                          --------------------
                                     4      2
                               %pi (w  - 6 w  + 25)
(%i8) i12r: realpart(i12)/(2*%pi);
                              4                    2
               (cos(2) + 1) w  + (6 cos(2) - 6) w  + 5 cos(2) + 25
(%o8)          ---------------------------------------------------
                                    6      4       2
                           2 %pi (w  - 5 w  + 19 w  + 25)
(%i9) iexp:i12r + %i*i12i;
                    4                    2
      (cos(2) + 1) w  + (6 cos(2) - 6) w  + 5 cos(2) + 25
(%o9) ---------------------------------------------------
                       6      4       2
                2 %pi (w  - 5 w  + 19 w  + 25)
                                                      2 %i sin(2) w
                                                + --------------------
                                                     4      2
                                                %pi (w  - 6 w  + 25)
```

The expression `iexp` is the desired exponential Fourier transform. We can now calculate the inverse Fourier transform.

```
(%i10) integrate(exp(-%i*w*x)*iexp,w,minf,inf);
Is  x  positive, negative, or zero?
p;
                  - x
                %e    (sin(2) sin(2 x) + cos(2) cos(2 x) + 1)
(%o10)          ---------------------------------------------
                                    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**).

```
(%i11) cos(x-1)^2;
                                   2
(%o11)                          cos (x - 1)
(%i12) trigreduce(%);
                             cos(2 (x - 1)) + 1
(%o12)                       -----------------
                                     2
```

```
(%i13) ratsimp(%);
                                cos(2 x - 2) + 1
(%o13)                          ----------------
                                       2
(%i14) trigexpand(%);
                     sin(2) sin(2 x) + cos(2) cos(2 x) + 1
(%o14)               -------------------------------------
                                    2
```

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

```
(%i15) trigreduce(%);
                                cos(2 x - 2) + 1
(%o15)                          ----------------
                                       2
(%i16) factor(%);
                               cos(2 (x - 1)) + 1
(%o16)                         ------------------
                                       2
(%i17) trigexpand(%);
                             2              2
                        - sin (x - 1) + cos (x - 1) + 1
(%o17)                  -------------------------------
                                     2
(%i18) trigsimp(%);
                                    2
(%o18)                           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.

```
(%i19) load(fourie);
(%o19) C:/PROGRA~1/MAXIMA~1.1/share/maxima/5.16.1/share/calculus/fourie.mac
(%i20) fourint(cos(x-1)^2*exp(-abs(x)),x);
     2
Is  z  - 4 z + 5  positive or negative?

p;
Is  z - 2  positive, negative, or zero?

p;
                                4                     2
                     (cos(2) + 1) z  + (6 cos(2) - 6) z  + 5 cos(2) + 25
(%t20)       a   = ------------------------------------------------------
              z                        6       4       2
                              %pi (z  - 5 z  + 19 z  + 25)

                                        4 sin(2) z
(%t21)                          b   = ------------------
                                 z          4      2
                                      %pi (z  - 6 z  + 25)

(%o21)                               [%t70, %t71]
```

```
(%i22) az : rhs(%t70);
                            4                     2
                 (cos(2) + 1) z  + (6 cos(2) - 6) z  + 5 cos(2) + 25
(%o22)           ---------------------------------------------------
                                  6       4       2
                           %pi (z  - 5 z  + 19 z  + 25)
(%i23) bz : rhs(%t71);
                                     4 sin(2) z
(%o23)                           --------------------
                                       4      2
                                 %pi (z  - 6 z  + 25)
(%i24) iexp_f : az/2 + %i*bz/2;
                      4                    2
       (cos(2) + 1) z  + (6 cos(2) - 6) z  + 5 cos(2) + 25
(%o24) ----------------------------------------------------
                          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`:

```
(%i25) subst(z=w,iexp_f) - iexp;
(%o25)                                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 $\mathbf{F_{Exp}(f,z) = a_z/2 + (i\,b_z)/2}$.

7.12.7 Dirac Delta Function $\boldsymbol{\delta(x)}$

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

$$\boldsymbol{\delta(-x) = \delta(x)} \quad \text{even function} \tag{7.75}$$

$$\boldsymbol{\delta(x) = 0} \quad \text{for} \quad \boldsymbol{x \neq 0} \tag{7.76}$$

$$\boldsymbol{\int_{-a}^{b} \delta(x)\,dx = 1} \quad \text{for} \quad \boldsymbol{a,\, b > 0} \tag{7.77}$$

These equations imply

$$\boldsymbol{\delta(y - x) = \delta(x - y)} \tag{7.78}$$

and

$$\boldsymbol{\int f(y)\,\delta(y - x)\,dy = f(x)} \tag{7.79}$$

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

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

$$\boldsymbol{\int_{-a}^{b} \delta(c\,x)\,dx = \frac{1}{c} \int_{-c\,a}^{c\,b} \delta(y)\,dy = \frac{1}{c}} \tag{7.80}$$

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

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

$$\int_{-\mathbf{a}}^{\mathbf{b}} \boldsymbol{\delta}(\mathbf{c}\,\mathbf{x})\,\mathbf{dx} = \frac{\mathbf{1}}{\mathbf{c}} \int_{-\mathbf{c}\,\mathbf{a}}^{\mathbf{c}\,\mathbf{b}} \boldsymbol{\delta}(\mathbf{y})\,\mathbf{dy} = \frac{\mathbf{1}}{\mathbf{c}} \int_{|\mathbf{c}|\,\mathbf{a}}^{-|\mathbf{c}|\,\mathbf{b}} \boldsymbol{\delta}(\mathbf{y})\,\mathbf{dy} = -\frac{\mathbf{1}}{\mathbf{c}} \int_{-|\mathbf{c}|\,\mathbf{b}}^{|\mathbf{c}|\,\mathbf{a}} \boldsymbol{\delta}(\mathbf{y})\,\mathbf{dy} = \frac{\mathbf{1}}{|\mathbf{c}|} \tag{7.81}$$

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

$$\int \boldsymbol{\delta}(\mathbf{c}\,\mathbf{x})\,\mathbf{dx} = \int \frac{\boldsymbol{\delta}(\mathbf{x})}{|\mathbf{c}|}\,\mathbf{dx} \tag{7.82}$$

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

$$\boldsymbol{\delta}(\mathbf{c}\,\mathbf{x}) = \frac{\boldsymbol{\delta}(\mathbf{x})}{|\mathbf{c}|} \tag{7.83}$$

We can find a sometimes useful representation of the Dirac delta function by using the exponential Fourier transform pair Eqs.(7.64) and (7.65). If we use $\mathbf{y}$ as the (dummy) variable of integration in Eq.(7.64), and insert into Eq.(7.65), we get (interchanging the order of integration)

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

Hence we can write

$$\boldsymbol{\delta}(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{2}\,\boldsymbol{\pi}} \int_{-\infty}^{\infty} \mathbf{e}^{\mathbf{i}\,\boldsymbol{\omega}\,\mathbf{x}}\,\mathbf{d}\boldsymbol{\omega} \tag{7.84}$$

One use of such a representation of the one dimensional Dirac delta function is to derive in a different way the result of Eq.(7.83). From Eq.(7.84) we have

$$\boldsymbol{\delta}(\mathbf{c}\,\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{2}\,\boldsymbol{\pi}} \int_{-\infty}^{\infty} \mathbf{e}^{\mathbf{i}\,\boldsymbol{\omega}\,\mathbf{c}\,\mathbf{x}}\,\mathbf{d}\boldsymbol{\omega} \tag{7.85}$$

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

$$\boldsymbol{\delta}(\mathbf{c}\,\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{c}} \left(\frac{\mathbf{1}}{\mathbf{2}\,\boldsymbol{\pi}} \int_{-\infty}^{\infty} \mathbf{e}^{\mathbf{i}\,\mathbf{y}\,\mathbf{x}}\,\mathbf{dy} \right) = \frac{\mathbf{1}}{\mathbf{c}}\,\boldsymbol{\delta}(\mathbf{x}) \tag{7.86}$$

If, on the other hand, $\mathbf{c} < \mathbf{0}$, $\mathbf{c} = -|\mathbf{c}|$ in Eq.(7.85), and we make the same change of variables, when $\boldsymbol{\omega} = -\infty$, $\mathbf{y} = \infty$, and when $\boldsymbol{\omega} = \infty, \mathbf{y} = -\infty$. Hence, for this case,

$$\boldsymbol{\delta}(\mathbf{c}\,\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{c}} \left(\frac{\mathbf{1}}{\mathbf{2}\,\boldsymbol{\pi}} \int_{\infty}^{-\infty} \mathbf{e}^{\mathbf{i}\,\mathbf{y}\,\mathbf{x}}\,\mathbf{dy} \right) = \frac{\mathbf{1}}{\mathbf{c}} \left(-\frac{\mathbf{1}}{\mathbf{2}\,\boldsymbol{\pi}} \int_{-\infty}^{\infty} \mathbf{e}^{\mathbf{i}\,\mathbf{y}\,\mathbf{x}}\,\mathbf{dy} \right) = \frac{\mathbf{1}}{|\mathbf{c}|}\,\boldsymbol{\delta}(\mathbf{x}). \tag{7.87}$$

Combining the two cases leads us again to Eq.(7.83).

Although the representation Eq.(7.84) 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.(7.84) by finite limits, we can work with the representation

$$\boldsymbol{\delta}(\mathbf{x}) = \lim_{\boldsymbol{L\to\infty}} \mathbf{d_L}(\mathbf{x}) \tag{7.88}$$

where

$$\mathbf{d_L}(\mathbf{x}) = \frac{\mathbf{1}}{\mathbf{2\,\pi}} \int_{\mathbf{-L}}^{\mathbf{L}} \mathbf{e^{i\,x\,y}}\,\mathbf{dy} = \frac{\mathbf{sin(x\,L)}}{\boldsymbol{\pi}\,\mathbf{x}} \tag{7.89}$$

which can then be used as

$$\int \mathbf{f(x)}\,\boldsymbol{\delta}(\mathbf{x})\,\mathbf{dx} = \lim_{\boldsymbol{L\to\infty}} \int \mathbf{f(x)}\,\mathbf{d_L(x)}\,\mathbf{dx} = \lim_{\boldsymbol{L\to\infty}} \int \mathbf{f(x)}\,\frac{\mathbf{sin(x\,L)}}{\boldsymbol{\pi}\,\mathbf{x}}. \tag{7.90}$$

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 $\mathbf{x}$ 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

$$\mathbf{d_\epsilon(x)} = \left\{ \begin{array}{ccc} 1/(2\,\boldsymbol{\epsilon}) & \text{for} & -\boldsymbol{\epsilon} < \mathbf{x} < \boldsymbol{\epsilon} \\ 0 & \text{for} & \mathbf{x} \le -\boldsymbol{\epsilon} \quad \text{or} \quad \mathbf{x} \ge \boldsymbol{\epsilon} \end{array} \right\} \tag{7.91}$$

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

Then

$$\mathbf{d_\epsilon(y - x)} = \left\{ \begin{array}{ccc} 1/(2\,\boldsymbol{\epsilon}) & \text{for} & (\mathbf{x} - \boldsymbol{\epsilon}) < \mathbf{y} < (\mathbf{x} + \boldsymbol{\epsilon}) \\ 0 & \text{for} & \mathbf{y} \le (\mathbf{x} - \boldsymbol{\epsilon}) \quad \text{or} \quad \mathbf{y} \ge (\mathbf{x} + \boldsymbol{\epsilon}) \end{array} \right\} \tag{7.92}$$

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

$$\int \mathbf{f(y)}\,\boldsymbol{\delta}(\mathbf{y - x})\,\mathbf{dy} = \lim_{\boldsymbol{\epsilon\to 0}} \int \mathbf{f(y)}\,\mathbf{d_\epsilon(y - x)}\,\mathbf{dy} = \lim_{\boldsymbol{\epsilon\to 0}} \frac{\mathbf{1}}{\mathbf{2\,\epsilon}} \int_{\mathbf{x-\epsilon}}^{\mathbf{x+\epsilon}} \mathbf{f(y)}\,\mathbf{dy}. \tag{7.93}$$

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

However, if $\mathbf{f(y)}$ 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

$$\lim_{\boldsymbol{\epsilon\to 0}} \frac{\mathbf{1}}{\mathbf{2\,\epsilon}}\,\mathbf{f}(\hat{\mathbf{y}})\,(\mathbf{2\,\epsilon}) = \lim_{\boldsymbol{\epsilon\to 0}} \mathbf{f}(\hat{\mathbf{y}})). \tag{7.94}$$

where $(\mathbf{x} - \boldsymbol{\epsilon}) < \hat{\mathbf{y}} < (\mathbf{x} + \boldsymbol{\epsilon})$

As $\boldsymbol{\epsilon} \to \mathbf{0},\ \hat{\mathbf{y}} \to \mathbf{x}$, and $\mathbf{f}(\hat{\mathbf{y}}) \to \mathbf{f(x)}$, which recovers Eq. (7.79), and which we repeat here for emphasis.

$$\int \mathbf{f(y)}\,\boldsymbol{\delta}(\mathbf{y - x})\,\mathbf{dy} = \mathbf{f(x)} \tag{7.95}$$

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

7.12.8 Laplace Transform of the Delta Function Using a Limit Method

Use of Eq. (7.95) allows us to immediately write down the Laplace transform of $\boldsymbol{\delta}(\mathbf{t}-\mathbf{t_0})$ for $\mathbf{t_0}>\mathbf{0}$, the integral $\int_{\mathbf{0}}^{\infty}\mathbf{e}^{-\mathbf{st}}\,\boldsymbol{\delta}(\mathbf{t}-\mathbf{t_0})\,\mathbf{dt}$, since the function $\mathbf{e}^{-\mathbf{st}}$ 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.

$$\mathcal{L}\{\boldsymbol{\delta}(\mathbf{t}-\mathbf{t_0})\}\equiv\int_{\mathbf{0}}^{\infty}\mathbf{e}^{-\mathbf{st}}\,\boldsymbol{\delta}(\mathbf{t}-\mathbf{t_0})\,\mathbf{dt}=\mathbf{e}^{-\mathbf{st_0}} \tag{7.96}$$

provided $\mathbf{s}$ 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.(7.96) as $\mathbf{t_0}\rightarrow\mathbf{0}^{+}$ then gives us the Laplace transform of $\boldsymbol{\delta}(\mathbf{t})$.

$$\mathcal{L}\{\boldsymbol{\delta}(\mathbf{t})\}\equiv\int_{\mathbf{0}}^{\infty}\mathbf{e}^{-\mathbf{st}}\,\boldsymbol{\delta}(\mathbf{t})\,\mathbf{dt}=\mathbf{1} \tag{7.97}$$

7.13 Laplace Transform and Inverse Transform Integrals

7.13.1 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

$$\int_{\mathbf{0}}^{\infty}\mathbf{e}^{-\mathbf{st}}\,\mathbf{f}(\mathbf{t})\,\mathbf{d\,t}=\mathcal{L}\{\mathbf{f}(\mathbf{t})\}=\mathbf{F}(\mathbf{s}) \tag{7.98}$$

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

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 $\mathbf{s}$ 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(1/s,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(1/s^2,s,t);
(%o7)                                 t
```

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

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

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

```
(%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
```

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

Here is an example of an expression involving a Bessel function which **specint** can handle, but **laplace** cannot. We also see that **ilt** cannot find the inverse Laplace transform.

```
(%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);
(%o22)       laplace(bessel_j(1, 2 sqrt(a) sqrt(t)) sqrt(t), t, 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
```

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

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

```
(%i25) assume(s>0)$
(%i26) laplace(erf(sqrt(t)),t,s);
(%o26)                     laplace(erf(sqrt(t)), t, s)
(%i27) specint(exp(-s*t)*erf(sqrt(t)),t);
Is  s  positive, negative, or zero?
p;
                                      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)
```

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

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);
Is  s  positive, negative, or zero?
p;
                                      - s t
(%o32)                     specint(%e      erf(t), t)
```

7.13.2 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 7.12.7 and 7.12.8. Here are three examples of using **delta** with **laplace**.

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

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

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

(for $\mathbf{a > 0}$):

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

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

(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

$$\int_0^\infty f(t)\,\delta(t-a)\,dt = f(a) \tag{7.99}$$

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

7.13.3 The Inverse Laplace Transform: ilt(..), residue(..)

Inverse Laplace Transform: ilt(...)

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

$$f(t > 0) = \frac{1}{2\,\pi\,i}\int_{\sigma-i\,\infty}^{\sigma+i\,\infty} e^{s\,t}\,F(s)\,ds \tag{7.100}$$

where the contour of integration is a vertical line in the complex $s = x + i\,y$ 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 $e^{s\,t} = e^{i\,t\,y}\,e^{t\,x}$. Because $t > 0$ the integrand is heavily damped by the factor $e^{x\,t}$ when $x \to -\infty$, 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.(7.100) is then cancelled by the factor $1/(2\,\pi\,i)$ which appears in the definition (Eq.(7.100)) 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**.

residue(..)

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** here is called **fst**, and we used **partfrac** on **fst** to exhibit the poles of this integrand. We see that there is one simple pole at $\mathbf{s = -2}$ and one pole of order **2** located at $\mathbf{s = -1}$. To use the Maxima function **residue**, we need to tell Maxima that $\mathbf{t > 0}$.

```
(%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.