1 abs_integrate

1.1 Introduction to abs_integrate

The package `abs_integrate` extends Maxima's integration code to some integrands that involve the absolute value, max, min, signum, or unit step functions. For integrands of the form $p(x)|q(x)|$, where p is a polynomial and q is a polynomial that `factor` is able to factor into a product of linear or constant terms, the `abs_integrate` package determines an antiderivative that is continuous on the entire real line. Additionally, for an integrand that involves one or more parameters, the function `conditional_integrate` tries to determine an antiderivative that is valid for all parameter values.

Examples To use the `abs_integrate` package, you'll first need to load it:

```
(%i1) load("abs_integrate.mac")$

(%i2) integrate(abs(x),x);
(%o2) (x*abs(x))/2
```

To convert (%o2) into an expression involving the signum function, apply `convert_to_signum`; thus

```
(%i3) convert_to_signum(%);
(%o3) (x^2*signum(x))/2
```

When the integrand has the form $p(x)|x-c1||x-c2|...|x-cn|$, where $p(x)$ is a polynomial and $c1, c2, ..., cn$ are constants, the `abs_integrate` package returns an antiderivative that is valid on the entire real line; thus *without* making assumptions on a and b; for example

```
(%i4) factor(convert_to_signum(integrate(abs((x-a)*(x-b)),x,a,b)));
(%o4) ((b-a)^3*signum(b-a)^2)/6
```

Additionally, `abs_integrate` is able to find antiderivatives of some integrands involving `max, min, signum`, and `unit_step`; examples:

```
(%i5) integrate(max(x,x^2),x);
(%o5) (signum(x-1)*((2*x^3-3*x^2)/12+1/12)+1/12)*signum(x)+
      (x^3/3+x^2/2)/2

(%i6) integrate(signum(x) - signum(1-x),x);
(%o6) abs(x)+abs(x-1)
```

A plot indicates that indeed (%o5) and (%o6) are continuous at zero and at one.

For definite integrals with numerical integration limits (including both minus and plus infinity), the `abs_integrate` package converts the integrand to signum form and then it tries to subdivide the integration region so that the integrand simplifies to a non-signum expression on each subinterval; for example

```
(%i1) integrate(1 / (1 + abs(x-5)),x,-5,6);
(%o1) log(11)+log(2)
```

Finally, `abs_integrate` is able to determine antiderivatives of *some* functions of the form $F(x, |x-a|)$; examples

```
(%i2) integrate(1/(1 + abs(x)),x);
(%o2) ((signum(x)+1)*log(x+1))/2-(log(1-x)*(1-signum(x)))/2

(%i3) integrate(cos(x + abs(x)),x);
(%o3) ((signum(x)+1)*sin(2*x)-2*x*signum(x)+2*x)/4
```

Barton Willis (Professor of Mathematics, University of Nebraska at Kearney) wrote the `abs_integrate` package and its English language user documentation. This documentation also describes the `partition` package for integration. Richard Fateman wrote `partition`. Additional documentation for `partition` is located at https://people.eecs.berkeley.edu/~fateman/papers/partition.pdf.

To use `load(abs_integrate)`

1.2 Definitions for abs_integrate

`extra_integration_methods` [Option]

Default value: `['signum_int, 'abs_integrate_use_if]`

The list `extra_integration_methods` is a list of functions for integration. When `integrate` is unable to find an antiderivative, Maxima uses the methods in `extra_integration_methods` to attempt to determine an antiderivative.

Each function `f` in `extra_integration_methods` should have the form `f(integrand, variable)`. The function `f` may either return `false` to indicate failure, or it may return an expression involving an integration noun form. The integration methods are tried from the first to the last member of `extra_integration_methods`; when no method returns an expression that does not involve an integration noun form, the value of the integral is the last value that does not fail (or a pure noun form if all methods fail).

When the function `abs_integrate_use_if` is successful, it returns a conditional expression; for example

```
(%i2) integrate(1/(1 + abs(x+1) + abs(x-1)),x);
(%o2) %if(-(x+1)>0,-log(1-2*x)/2+log(3)-2/3,
          %if(-(x-1)>0,x/3+log(3)/2-1/3,log(2*x+1)/2))

(%i3) integrate(exp(-abs(x-1) - abs(x)),x);
(%o3) %if(-x>0,%e^(2*x-1)/2-2*%e^(-1),
          %if(-(x-1)>0,%e^(-1)*x-(3*%e^(-1))/2,-%e^(1-2*x)/2))
```

For definite integration, these conditional expressions can cause trouble:

```
(%i4) integrate(exp(-abs(x-1) - abs(x)),x, minf,inf);
(%o4) 'limit(%if(-x > 0,(%e^-1*(%e^(2*x)-4))/2,
                 %if(-(x-1) > 0,(%e^-1*(2*x-3))/2,-%e^(1-2*x)/2)),
                 x,inf,minus)
-'limit(%if(-x > 0,(%e^-1*(%e^(2*x)-4))/2,
            %if(-(x-1) > 0,(%e^-1*(2*x-3))/2,-%e^(1-2*x)/2)),x,
            minf,plus)
```

For such definite integrals, try disallowing the method `abs_integrate_use_if`:

```
(%i9) integrate(exp(-abs(x-1) - abs(x)),x, minf,inf),
                  extra_integration_methods : ['signum_int];
(%o9) 2*%e^(-1)
```

Related options *extra_definite_integration_methods*

To use `load(abs_integrate)`

`extra_definite_integration_methods` [Option]

Default value: `['abs_defint]`

The list `extra_definite_integration_methods` is a list of extra functions for *definite* integration. When `integrate` is unable to find a definite integral, Maxima uses the methods in `extra_definite_integration_methods` to attempt to determine an antiderivative.

Each function `f` in `extra_definite_integration_methods` should have the form `f(integrand, variable, lo, hi)`, where `lo` and `hi` are the lower and upper limits of integration, respectively. The function `f` may either return `false` to indicate failure, or it may return an expression involving an integration noun form. The integration methods are tried from the first to the last member of `extra_definite_integration_methods`; when no method returns an expression that does not involve an integration noun form, the value of the integral is the last value that does not fail (or a pure noun form if all methods fail).

Related options *extra_integration_methods*

To use `load(abs_integrate)`

`intfudu` (*e*, *x*) [Function]

This function uses the derivative divides rule for integrands of the form $f(w(x)) * diff(w(x), x)$ When `infudu` is unable to find an antiderivative, it returns false.

```
(%i1) intfudu(cos(x^2) * x,x);
(%o1) sin(x^2)/2

(%i3) intfudu(x * sqrt(1+x^2),x);
(%o3) (x^2+1)^(3/2)/3

(%i4) intfudu(x * sqrt(1 + x^4),x);
(%o4) false
```

For the last example, the derivative divides rule fails, so `intfudu` returns false.

A hashed array `intable` contains the antiderivative data. To append a fact to the hash table, say $integrate(f) = g$, do this:

```
(%i1) intable[f] : lambda([u],  [g(u),diff(u,%voi)]);
(%o1) lambda([u],[g(u),diff(u,%voi)])

(%i2) intfudu(f(z),z);
(%o2) g(z)

(%i3) intfudu(f(w(x)) * diff(w(x),x),x);
(%o3) g(w(x))
```

An alternative to calling `intfudu` directly is to use the `extra_integration_methods` mechanism; an example:

```
(%i1) load("abs_integrate.mac")$
(%i2) load(basic)$
(%i3) load("partition.mac")$

(%i4) integrate(bessel_j(1,x^2) * x,x);
(%o4) integrate(bessel_j(1,x^2)*x,x)

(%i5) push('intfudu, extra_integration_methods)$

(%i6) integrate(bessel_j(1,x^2) * x,x);
(%o6) -bessel_j(0,x^2)/2
```

To use `load(partition)`

Additional documentation
https://people.eecs.berkeley.edu/~fateman/papers/partition.pdf

Related functions *intfugudu*

`intfugudu` (*e*, *x*) [Function]

This function uses the derivative divides rule for integrands of the form $f(w(x)) * g(w(x)) * diff(w(x), x)$. When `infudu` is unable to find an antiderivative, it returns false.

```
(%i1) diff(jacobi_sn(x,2/3),x);
(%o1) jacobi_cn(x,2/3)*jacobi_dn(x,2/3)

(%i2) intfugudu(%,x);
(%o2) jacobi_sn(x,2/3)

(%i3) diff(jacobi_dn(x^2,a),x);
(%o3) -2*a*x*jacobi_cn(x^2,a)*jacobi_sn(x^2,a)

(%i4) intfugudu(%,x);
(%o4) jacobi_dn(x^2,a)
```

For a method for automatically calling `infugudu` from `integrate`, see the documentation for `intfudu`.

To use `load(partition)`

Additional documentation
https://people.eecs.berkeley.edu/~fateman/papers/partition.pdf

Related functions *intfudu*

`signum_to_abs` (*e*) [Function]

This function replaces subexpressions of the form $qsignum(q)$ by $abs(q)$. Before it does these substitutions, it replaces subexpressions of the form $signum(p)*signum(q)$ by $signum(p*q)$; examples:

```
(%i1) map('signum_to_abs, [x * signum(x),
```

```
                                  x * y * signum(x)* signum(y)/2]);
(%o1) [abs(x),(abs(x)*abs(y))/2]
```

To use `load(abs_integrate)`

`conditional_integrate` (*e*, *x*) [Function]

For an integrand with one or more parameters, this function tries to determine an antiderivative that is valid for all parameter values. When successful, this function returns a conditional expression for the antiderivative.

```
(%i1) conditional_integrate(cos(m*x),x);
(%o1) %if(m#0,sin(m*x)/m,x)

(%i2) conditional_integrate(cos(m*x)*cos(x),x);
(%o2) %if((m-1 # 0) %and (m+1 # 0),
          ((m-1)*sin((m+1)*x)+((-m)-1)*sin((1-m)*x))/(2*m^2-2),
          (sin(2*x)+2*x)/4)

(%i3) sublis([m=6],%);
(%o3) -(5*cos(7*x)+7*cos(5*x))/70

(%i4) conditional_integrate(exp(a*x^2+b*x),x);
(%o4) %if(a # 0,
           -(sqrt(%pi)*%e^-(b^2/(4*a))*erf((2*a*x+b)/(2*sqrt(-a))))
            /(2*sqrt(-a)),%if(b # 0,%e^(b*x)/b,x))
```

`convert_to_signum` (*e*) [Function]

This function replaces subexpressions of the form $abs(q)$, $unit\_step(q)$, $min(q1, q2, ..., qn)$ and $max(q1, q2, ..., qn)$ by equivalent *signum* terms.

```
(%i1) map('convert_to_signum, [abs(x), unit_step(x),
                               max(a,2), min(a,2)]);
(%o1) [x*signum(x),(signum(x)*(signum(x)+1))/2,
       (a+signum(a-2)*(a-2)+2)/2,
       (a-signum(a-2)*(a-2)+2)/2]
```

To convert `unit_step` to signum form, the function `convert_to_signum` uses $unit\_step(x) = (1 + signum(x))/2$.

To use `load(abs_integrate)`

Related functions *signum_to_abs*

Appendix A Function and variable index