evaluate_integral

October 6, 2021

1 Integration by parts with arbitrary functions in *sympy*

Date: 2021-10-04, *Written by:* Johannes Borgqvist.

One of the last steps in order to finish the writing of the code is to be able to evaluate integrals by using integration by parts on an integral where the integrand contains an unknown function or more specifically the derivative of an unknown function. To this end, this notebook is an attempt to start writing the function(s) required to simplify these integrals. Also, the integrals should be definite where the integration is conducted with the help of a dummy variable.

More concretely, consider the following integral

$$\int_0^t \frac{\mathrm{d}f(s)}{\mathrm{d}s}e^{ks}\mathrm{d}s$$

where t is the variable, f is the unknown function, s is the dummy variable and k is an arbitrary constant. Then the integration by parts of the above integral looks as follows:

$$\begin{aligned}
\int_0^t \frac{\mathrm{d}f(s)}{\mathrm{d}s}e^{ks}\mathrm{d}s &= \left[f(s)e^{ks}\right]_0^t - k\int_0^t f(s)e^{ks}\mathrm{d}s & (1)\\
&= f(t)e^{kt} - \underbrace{f(0)}_{=C} - k\int_0^t f(s)e^{ks}\mathrm{d}s & (2)\\
\implies \int_0^t \frac{\mathrm{d}f(s)}{\mathrm{d}s}e^{ks}\mathrm{d}s &= f(t)e^{kt} - C - k\int_0^t f(s)e^{ks}\mathrm{d}s & (3)
\end{aligned}$$

where C is an arbitrary integration constant. So, we need to be able to evaluate this integral using integration by parts. So let's see if we can work with this.

[23]:
```
# Import our beloved libraries
from sympy import *
from sympy.core.function import *
# Create two arbitrary coefficients
k1 = Symbol('k1')
k2 = Symbol('k2')
# Create two arbitrary integration coefficients
C1 = Symbol('C1')
C2 = Symbol('C2')
# Create the variable which we integrate with
t = Symbol('t')
# Create the dummy variable with which we integrate with
```

```python
s = Symbol('s')
# Create two arbitrary expressions
f = symbols('f',cls=Function)
g = symbols('g',cls=Function)
# Define an expression for the integrand
integrand = diff(f(t),t)*exp(k1*t)
#integrand = Derivative(f(t),t)*exp(k*t)
# Create an integral
#integral = Integral(integrand,t,(t,0,t))
integral = integrate(integrand,(t,0,t))
# Print the integral
print("\n\tInitial Integral:\n\t\t%s\n"%(latex(integral)))
# Evaluate integral
integral = integral.doit()
# Print the evaluated integral
print("\n\tEvaluated Integral:\n\t\t%s\n"%(latex(integral)))
# Create a coefficient counter
coefficient_counter = 1
```

```
        Initial Integral:
                \int\limits_{0}^{t} e^{k_{1} t} \frac{d}{d t} f{\left(t
\right)}\, dt


        Evaluated Integral:
                \int\limits_{0}^{t} e^{k_{1} t} \frac{d}{d t} f{\left(t
\right)}\, dt
```

So it seems like the built-in "*doit*" command does not do the trick. So we need to implement something by ourselves.

$$\int\limits_{0}^{t} e^{kt} \frac{d}{dt} f(t)\, dt$$

[24]:
```python
# Okay, so can we extract all factors?
factors_in_integrand = integrand.factor().args
print("\n\tFactors in integrand:\t\t\t%s\n"%(str(factors_in_integrand)))
```

```
        Factors in integrand:                   (Derivative(f(t), t), exp(k1*t))
```

So that is brilliant, hey? Let's see if we can identify the derivatives here… Perhaps, we do not need to find the derivatives themselves, but rather we can find the undefined functions!

[25]:
```
# The derivative term
derivative_term = 0
# Loop over our factors in the integrand
# and see if any of them contain a derivative
for factor in factors_in_integrand:
    # Find the arbitrary functions
    a_f = list(factor.atoms(AppliedUndef))
    if len(a_f)>0:
        derivative_term = factor
# Print the derivative term
print("\t\tThe derivative term is:\n\t\t\t%s\n"%(latex(derivative_term)))
# Primitive function
primitive_function = Integral(derivative_term,t).doit()
# Print the primitive function
print("\t\tThe primitive function is:\n\t\t\t%s\n"%(latex(primitive_function)))
```

```
                The derivative term is:
                        \frac{d}{d t} f{\left(t \right)}

                The primitive function is:
                        f{\left(t \right)}
```

Ok, so now we are onto something. Let's put this to the test. Let's say we have two expressions

$$\int_0^t \frac{\mathrm{d}f(s)}{\mathrm{d}s}e^{k_1 s}\mathrm{d}s + \int_0^t \frac{\mathrm{d}f(s)}{\mathrm{d}s}e^{k_2 s}\mathrm{d}s = f(t)(e^{k_1 t}+e^{k_2 t}) - 2C_1 - k_1\int_0^t f(s)e^{k_1 s}\mathrm{d}s - k_2\int_0^t f(s)e^{k_2 s}\mathrm{d}s, \tag{4}$$

$$\int_0^t \frac{\mathrm{d}f(s)}{\mathrm{d}s}e^{k_1 s}\mathrm{d}s + \int_0^t \frac{\mathrm{d}g(s)}{\mathrm{d}s}e^{k_2 s}\mathrm{d}s = f(t)e^{k_1 t}+g(t)e^{k_2 t} - C_1 - C_2 - k_1\int_0^t f(s)e^{k_1 s}\mathrm{d}s - k_2\int_0^t g(s)e^{k_2 s}\mathrm{d}s. \tag{5}$$

$$\tag{6}$$

Then, we want to write a function which re-writes an integral expressions where a derivative is located in the integrands and then introduces one arbitrary coefficient in the first case and a second arbitrary integral in the second case.

First we need to see if it can identify the number of arbitrary functions in a composite integrand.

[26]:
```
integrand_composite_1 = integrand + diff(g(t),t)*exp(k2*t)
integrand_composite_2 = integrand + diff(f(t),t)*exp(k2*t)
print("\n\t\tComposite integrand 1:\n\t\t\t%s\n"%(latex(integrand_composite_1)))
print("\n\t\tComposite integrand 2:\n\t\t\t%s\n"%(latex(integrand_composite_2)))
arb_funcs_1 = integrand_composite_1.atoms(AppliedUndef)
arb_funcs_2 = integrand_composite_2.atoms(AppliedUndef)

#print(integrand_composite_1.args)
#print(integrand_composite_2.args)
```

```
print("\n\t\tArbitrary functions in 1:\n\t\t\t%s\n"%(latex(arb_funcs_1)))
print("\n\t\tArbitrary functions in 2:\n\t\t\t%s\n"%(latex(arb_funcs_2)))

print("\n\t\tTerms in integrand 1:\n\t\t\t%s"%(latex(integrand_composite_1.
↪args)))
print("\n\t\tTerms in integrand 2:\n\t\t\t%s"%(latex(integrand_composite_2.
↪args)))
```

```
                Composite integrand 1:
                        e^{k_{1} t} \frac{d}{d t} f{\left(t \right)} + e^{k_{2}
t} \frac{d}{d t} g{\left(t \right)}


                Composite integrand 2:
                        e^{k_{1} t} \frac{d}{d t} f{\left(t \right)} + e^{k_{2}
t} \frac{d}{d t} f{\left(t \right)}


                Arbitrary functions in 1:
                        \left\{f{\left(t \right)}, g{\left(t \right)}\right\}


                Arbitrary functions in 2:
                        \left\{f{\left(t \right)}\right\}


                Terms in integrand 1:
                        \left( e^{k_{1} t} \frac{d}{d t} f{\left(t \right)}, \
e^{k_{2} t} \frac{d}{d t} g{\left(t \right)}\right)

                Terms in integrand 2:
                        \left( e^{k_{1} t} \frac{d}{d t} f{\left(t \right)}, \
e^{k_{2} t} \frac{d}{d t} f{\left(t \right)}\right)
```

Ok, let's see if we might be able to write a function here. We'll need three things 1. "function_list" which is a list of arbitrary functions, 2. "constant_list" which is a list of arbitrary constants coupled to the arbitrary functions, 3. "integrand_vector" which is a list of the integrand that are to be evaluated using integration by parts.

[27]:
```
function_list = [f, g]
constant_list = [C1, C2]
integrand_list = [integrand_composite_1, integrand_composite_2]
```

[41]:
```
def integration_by_parts(function_list,constant_list,integrand_list,variable):
    # At the end, we want to return the correct integrals
    integral_list = []
    # Loop through the integrands
```

```
    for integrand in integrand_list:
        # Allocate a temporary integral
        temp_int = 0
        # Split this into further parts and solve these
        for sub_integrand in integrand.args:
            # Now, firstly we isolate the factors
            factors_in_subintegrand = sub_integrand.factor().args
            # The derivative term
            derivative_term = 0
            # Loop over our factors in the integrand
            # and see if any of them contain a derivative
            for factor in factors_in_subintegrand:
                # Find the arbitrary functions
                a_f = list(factor.atoms(AppliedUndef))
                if len(a_f)>0:
                    derivative_term = factor
            # The other func is the coefficient of the derivative term
            other_func = sub_integrand.coeff(derivative_term)
            # Calculate the primitive function
            primitive_function = Integral(derivative_term,variable).doit()
            # Let's add the three terms to our temporary integral
            temp_int += primitive_function*other_func
            # Loop over integration constants as well
            for f_i in range(len(function_list)):
                if function_list[f_i](variable)==primitive_function:
                    temp_int += -constant_list[f_i]*other_func.subs(variable,0)
            # Lastly, we add the integral term
            new_integrand = primitive_function*Derivative(other_func,variable).
 ↪doit()
            temp_int += -Integral(new_integrand,(variable,0,variable))
        if temp_int != 0:
            temp_int = temp_int.doit()
        # Add the evaluated integrals to our integral list
        integral_list.append(temp_int)
    # Return the integral list
    return integral_list
```

[42]:
```
# Let's test our function
integral_list =␣
 ↪integration_by_parts(function_list,constant_list,integrand_list,t)
# Print the results of our calculations
print("\\begin{align*}")
# Loop over them and print them one by one
for index in range(len(integrand_list)):
   ␣
 ↪print("%s&=%s\\\\"%(latex(Integral(integrand_list[index],(t,0,t))),latex(integral_list[inde
print("\\end{align*}")
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-42-6e16a0757a1f> in <module>
      1 # Let's test our function
----> 2 integral_list =␣
 ↪integration_by_parts(function_list,constant_list,integrand_list,t)
      3 # Print the results of our calculations
      4 print("\\begin{align*}")
      5 # Loop over them and print them one by one

<ipython-input-41-cb4aad50ec0d> in integration_by_parts(function_list,␣
 ↪constant_list, integrand_list, variable)
      9         for sub_integrand in integrand.args:
     10             # Now, firstly we isolate the factors
---> 11             factors_in_subintegrand = sub_integrand.factor().args
     12             # The derivative term
     13             derivative_term = 0

AttributeError: 'Tuple' object has no attribute 'factor'
```

The results of our calculations are:

$$\int\limits_0^t \left(e^{k_1 t}\frac{d}{dt}f(t) + e^{k_2 t}\frac{d}{dt}g(t)\right)\,dt = -C_1 - C_2 - k_1\int\limits_0^t f(t)e^{k_1 t}\,dt - k_2\int\limits_0^t g(t)e^{k_2 t}\,dt + f(t)e^{k_1 t} + g(t)e^{k_2 t}$$

$$\int\limits_0^t \left(e^{k_1 t}\frac{d}{dt}f(t) + e^{k_2 t}\frac{d}{dt}f(t)\right)\,dt = -2C_1 - k_1\int\limits_0^t f(t)e^{k_1 t}\,dt - k_2\int\limits_0^t f(t)e^{k_2 t}\,dt + f(t)e^{k_1 t} + f(t)e^{k_2 t}$$

And I mean this looks marvelous does it not? We've actually implemented the integration by parts correctly!

Now, I discovered a problem where the function above does not do the trick. It is the case when the composite function looks as follows:

$$\frac{\mathrm{d}f}{\mathrm{d}t}$$

which is to say that there is no other function present in the expression. So we really need to take this into account as well. Let's play around a little bit with expressions like this and then we can se what comes of it.

[30]:
```
# Define a new integrand
integrand_non_composite_1 = Derivative(f(t),t)
# Print the args, hey?
print("The arguments of %s:"%(str(integrand_non_composite_1)))
```

```
print(integrand_non_composite_1.args)
# Define a new integrand
integrand_non_composite_2 = k1*Derivative(g(t),t)
# Print the args, hey?
print("The arguments of %s:"%(str(integrand_non_composite_2)))
print(integrand_non_composite_2.args)
# Print the args, hey?
print("The arguments of %s:"%(str(integrand_composite_1)))
print(integrand_composite_1.args)
# Print the args, hey?
print("The arguments of %s:"%(str(integrand_composite_1.args[0])))
print(integrand_composite_1.args[0].args)
```

```
The arguments of Derivative(f(t), t):
(f(t), (t, 1))
The arguments of k1*Derivative(g(t), t):
(k1, Derivative(g(t), t))
The arguments of exp(k1*t)*Derivative(f(t), t) + exp(k2*t)*Derivative(g(t), t):
(exp(k1*t)*Derivative(f(t), t), exp(k2*t)*Derivative(g(t), t))
The arguments of exp(k1*t)*Derivative(f(t), t):
(Derivative(f(t), t), exp(k1*t))
```

Now, it all of a sudden came to me. Instead of using the function args which is not reliable, we should use the function coeff which is reliable! So below follows a better version of the function at hand.

[68]:
```
def␣
↪integration_by_parts(function_list,constant_list,integrand_list,variable,dummy):
↪
    # At the end, we want to return the correct integrals
    integral_list = []
    # Loop through the integrands
    for integrand in integrand_list:
        # Allocate a temporary integral
        temp_int = 0
        # Loop through the functions and find the coefficients in the integrand
        for func_index in range(len(function_list)):
            # Extract the function at hand
            func = function_list[func_index]
            # If we have no coefficients, we'll just move on
            if integrand.coeff(Derivative(func(variable),variable)) == 0:
                continue
            else:# Otherwise, we extract the other function in the integrand
                # Extract the coefficient which is the "other function" in the␣
↪integrand
                other_function = integrand.
↪coeff(Derivative(func(variable),variable))
            # Now, we have two options:
```

```
            # 1. The coefficient is a constant (e.g. 1, 5 or k),
            # 2. The coefficient is a function meaning that we have to use␣
↪integration
            # by parts.
            # Alternative 1: Constant
            if len(list(other_function.atoms(Function)))==0:
                # Just solve the integral
                temp_int += (other_function*func(variable)) -␣
↪(other_function*constant_list[func_index])
            else:# Alternative 2: Integration by parts
                # The boundary term in the integration by parts
                temp_int += (other_function*func(variable)) - (other_function.
↪subs(variable,0)*constant_list[func_index])
                # The integral in the integraiton by parts
                new_integrand =␣
↪(func(variable)*Derivative(other_function,variable).doit()).
↪subs(variable,dummy)
                temp_int += -Integral(new_integrand,(dummy,0,variable))
        # Evaluate the integrand if possible
        if temp_int != 0:
            temp_int = temp_int.doit()
        # Add the evaluated integrals to our integral list
        integral_list.append(temp_int)
    # Return the integral list
    return integral_list
```

[69]:
```
# Define the necessary functions, constants and integrand
function_list = [f, g]
constant_list = [C1, C2]
integrand_list = [integrand_composite_1,␣
↪integrand_composite_2,integrand_non_composite_1,integrand_non_composite_2,6*integrand_non_c
# Define a dummy variable
s = Symbol('s')
# Let's test the newest version of our function
integral_list =␣
↪integration_by_parts(function_list,constant_list,integrand_list,t,s)
# Print the results of our calculations
print("\\begin{align*}")
# Loop over them and print them one by one
for index in range(len(integrand_list)):
   ␣
↪print("%s&=%s\\\\"%(latex(Integral(integrand_list[index],(t,0,t))),latex(integral_list[inde
print("\\end{align*}")
```

```
\begin{align*}
\int\limits_{0}^{t} \left(e^{k_{1} t} \frac{d}{d t} f{\left(t \right)} +
e^{k_{2} t} \frac{d}{d t} g{\left(t \right)}\right)\, dt&=- C_{1} - C_{2} -
```

```
k_{1} \int\limits_{0}^{t} f{\left(s \right)} e^{k_{1} s}\, ds - k_{2}
\int\limits_{0}^{t} g{\left(s \right)} e^{k_{2} s}\, ds + f{\left(t \right)}
e^{k_{1} t} + g{\left(t \right)} e^{k_{2} t}\\
\int\limits_{0}^{t} \left(e^{k_{1} t} \frac{d}{d t} f{\left(t \right)} +
e^{k_{2} t} \frac{d}{d t} f{\left(t \right)}\right)\, dt&=- 2 C_{1} +
\left(e^{k_{1} t} + e^{k_{2} t}\right) f{\left(t \right)} - \int\limits_{0}^{t}
\left(k_{1} e^{k_{1} s} + k_{2} e^{k_{2} s}\right) f{\left(s \right)}\, ds\\
\int\limits_{0}^{t} \frac{d}{d t} f{\left(t \right)}\, dt&=- C_{1} + f{\left(t
\right)}\\
\int\limits_{0}^{t} k_{1} \frac{d}{d t} g{\left(t \right)}\, dt&=- C_{2} k_{1} +
k_{1} g{\left(t \right)}\\
\int\limits_{0}^{t} \left(- 3 k_{1} \frac{d}{d t} g{\left(t \right)} + 6
\frac{d}{d t} f{\left(t \right)}\right)\, dt&=- 6 C_{1} + 3 C_{2} k_{1} - 3
k_{1} g{\left(t \right)} + 6 f{\left(t \right)}\\
\end{align*}
```

The final test, innit?

$$\begin{aligned}
\int\limits_{0}^{t} \left(e^{k_{1} t} \frac{d}{d t} f{\left(t \right)} + e^{k_{2} t} \frac{d}{d t} g{\left(t \right)}\right)\, dt&=- C_{1} - C_{2} - k_{1} \int\limits_{0}^{t} f{\left(s \right)} e^{k_{1} s}\, ds - k_{2} \int\limits_{0}^{t} g{\left(s \right)} e^{k_{2} s}\, ds + f{\left(t \right)} e^{k_{1} t} + g{\left(t \right)} e^{k_{2} t}\\
\int\limits_{0}^{t} \left(e^{k_{1} t} \frac{d}{d t} f{\left(t \right)} + e^{k_{2} t} \frac{d}{d t} f{\left(t \right)}\right)\, dt&=- 2 C_{1} + \left(e^{k_{1} t} + e^{k_{2} t}\right) f{\left(t \right)} - \int\limits_{0}^{t} \left(k_{1} e^{k_{1} s} + k_{2} e^{k_{2} s}\right) f{\left(s \right)}\, ds\\
\int\limits_{0}^{t} \frac{d}{d t} f{\left(t \right)}\, dt&=- C_{1} + f{\left(t \right)}\\
\int\limits_{0}^{t} k_{1} \frac{d}{d t} g{\left(t \right)}\, dt&=- C_{2} k_{1} + k_{1} g{\left(t \right)}\\
\int\limits_{0}^{t} \left(- 3 k_{1} \frac{d}{d t} g{\left(t \right)} + 6 \frac{d}{d t} f{\left(t \right)}\right)\, dt&=- 6 C_{1} + 3 C_{2} k_{1} - 3 k_{1} g{\left(t \right)} + 6 f{\left(t \right)}
\end{aligned}$$

[ ]: