2016-04-29

William A. Stein

4/29/2016

Contents

1 Math 480: Open Source Mathematical Software

1.0.1 2016-04-29

1.0.2 William Stein

1.1 Lectures 15: Symbolic Calculus (part 3/3)

1.2 Plan:

- reminder: homework and peer grading due today at 6pm.
- start screencast
- talk for a few minutes about something
- finish up and polish your homework and peer grading, and ask questions.

1.3 1. Example involving rounding error

```
10e100 + 1 - 10e100
0.000000000000000
```

```
10^100 + 1 - 10^100
```

```
1
```

```
a = exp(100+10^(-20)) + 1 - exp(100)
show(a)
```

$$e^{\frac{10000000000000000000001}{100000000000000000000}} - e^{100} + 1$$

```
N(a)
```

```
0.000000000000000
```

```
F = RealIntervalField(200); F
```

```
Real Interval Field with 200 bits of precision
```

```
F(a)
```

```
2.6881171418161354484260761372890942647?e23
```

1.4 2. Assumptions - sometimes needed when integrating

E.g., when computing

$$\int x^n dx$$

the answer is usually easy it's $\frac{x^{n+1}}{n+1}$. However, when $n = 1$, it's $\log(x)$.

```
var('x, n')
integral(x^n, x)
```

```
(x, n)
Error in lines 2-2
Traceback (most recent call last):
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/smc_sagews/sage_server.py’’, line 905, in execute
    exec compile(block+'\n', '', 'single') in namespace, locals
  File ‘‘’’, line 1, in <module>
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/sage/misc/functional.py’’, line 664, in integral
    return x.integral(*args, **kwds)
  File ‘‘sage/symbolic/expression.pyx’’, line 11352, in
sage.symbolic.expression.Expression.integral
(/projects/sage/sage-6.10/src/build/cythonized/sage/symbolic/expression.cpp:60288)
    return integral(self, *args, **kwds)
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/sage/symbolic/integration/integral.py’’, line 759, in integrate
    return indefinite_integral(expression, v, hold=hold)
  File ‘‘sage/symbolic/function.pyx’’, line 988, in
sage.symbolic.function.BuiltinFunction.__call__
(/projects/sage/sage-6.10/src/build/cythonized/sage/symbolic/function.cpp:11343)
    res = super(BuiltinFunction, self).__call__(
  File ‘‘sage/symbolic/function.pyx’’, line 508, in sage.symbolic.function.Function.__call__
(/projects/sage/sage-6.10/src/build/cythonized/sage/symbolic/function.cpp:7211)
    res = g_function_eval2(self._serial, (<Expression>args[0])._gobj,
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/sage/symbolic/integration/integral.py’’, line 85, in _eval_
    res = integrator(f, x)
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/sage/symbolic/integration/external.py’’, line 22, in maxima_integrator
    result = maxima.sr_integral(expression,v)
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/sage/interfaces/maxima_lib.py’’, line 784, in sr_integral
    self._missing_assumption(s)
  File ‘‘/projects/sage/sage-6.10/local/lib/python2.7/site-
packages/sage/interfaces/maxima_lib.py’’, line 993, in _missing_assumption
    raise ValueError(outstr)
ValueError: Computation failed since Maxima requested additional constraints; using the
'assume' command before evaluation *may* help (example of legal syntax is 'assume(n>0)',
see `assume?` for more details)
Is n equal to -1?
```

```
forget()
assume(n==-1)
integral(x^n, x)
```

```
log(x)
```

```
forget(n==-1)
assume(n!=-1)
show(integral(x^n, x))
```

$$\frac{x^{n+1}}{n+1}$$

1.5 3. Sympy

Sympy is a Python library for symbolic calculus, which can be used independently from Sage, and is also in Sage. See http://www.sympy.org/en/index.html

Integration with Sage could be improved.

But being able to use Sympy without Sage is potentially very valuable (see Hamster). Or it can cause you to waste a lot of time (see Chris Swierczewski).

On Sage support list, here about sympy as follows:

1. I've been using Sympy, but switched to Sage since Sympy is too slow or missing something.

2. I've been using Sage to compute this integral (or series) and it's wrong! There is a bug in Maxima Use algorithm='sympy', since Sympy is right.

```
from sympy import Limit, symbols, cos
x = symbols('x')
```

```
expr = Limit((cos(x) - 1)/x, x, 0)
print expr
```

```
Limit((cos(x) - 1)/x, x, 0)
```

```
expr.doit()   # really??
```

```
0
```

```
show(expr)   # this doesn't work.  sigh.
```

```
  Limit((cos(x) - 1)/x, x, 0)
```

```
reset()  # since we overwrote x above
m = integrate(sin(x)*cos(x)*tan(x), x); show(m)      # uses maxima
```

$$\frac{1}{2}x - \frac{1}{4}\sin\left(2\,x\right)$$

```
s = integrate(sin(x)*cos(x)*tan(x), x, algorithm='sympy'); show(s)  \
    # uses sympy instead under the hood!
```

$$-\frac{1}{2}\cos\left(x\right)\sin\left(x\right) + \frac{1}{2}x$$

```
show(s-m)
```

$$-\frac{1}{2}\cos\left(x\right)\sin\left(x\right) + \frac{1}{4}\sin\left(2\,x\right)$$

```
(s - m).simplify_full()
```

```
0
```

```
bool(s==m)
```

```
True
```