Sympy and Code Generation

Mark Dewing

Goal: Need derivatives

- Need function value, first and second derivatives for a parameterized family of functions

$$f = x^i y^j z^k \exp(\alpha(x^2 + y^2 + z^2))$$

- Calculating and entering derivatives by hand is tedious and error-prone
- More generally – loss of domain information

Solution: Use Sympy + code generator

```
  f = sympy.sympify('x**i * exp(x**2)')
  df2 = sympy.diff(f,'x')
  df = df2.subs('i',0)

  deriv_stmt = sympy_to_py.convert([(“d”,df)])

  template_data = """\
def %name%(x):
    %derivative%
    return d
"""

  template = parse_python.parse(template_data)       # uses pyparsing

  lang_py.replace_template_node(template,'name','example')
  lang_py.replace_template_node(template,'derivative',deriv_stmt)

  print template.to_string()
```

Output:

```
def example(x):
  d = 2 * x * math.exp(x**2)
  return d
```

Observations / Future Possibilities

- I use Python because I care about performance
  - Actually, program generator written in python
    - Pare down to exact code needed for a particular problem
    - Generated program could also be in C++ or Fortran
- Correctness
- Better looking output
  - could also convert to MathML

More information

- Slides will be available at markdewing.com
- sympy http://code.google.com/p/sympy/

  Symbolic mathematics in python
- quameon http://quameon.sourceforge.net/

  Scientific app where this is developed and used

  Example code:

  https://quameon.svn.sourceforge.net/svnroot/quameon/trunk/codegen/simple_example

  Actual usage:

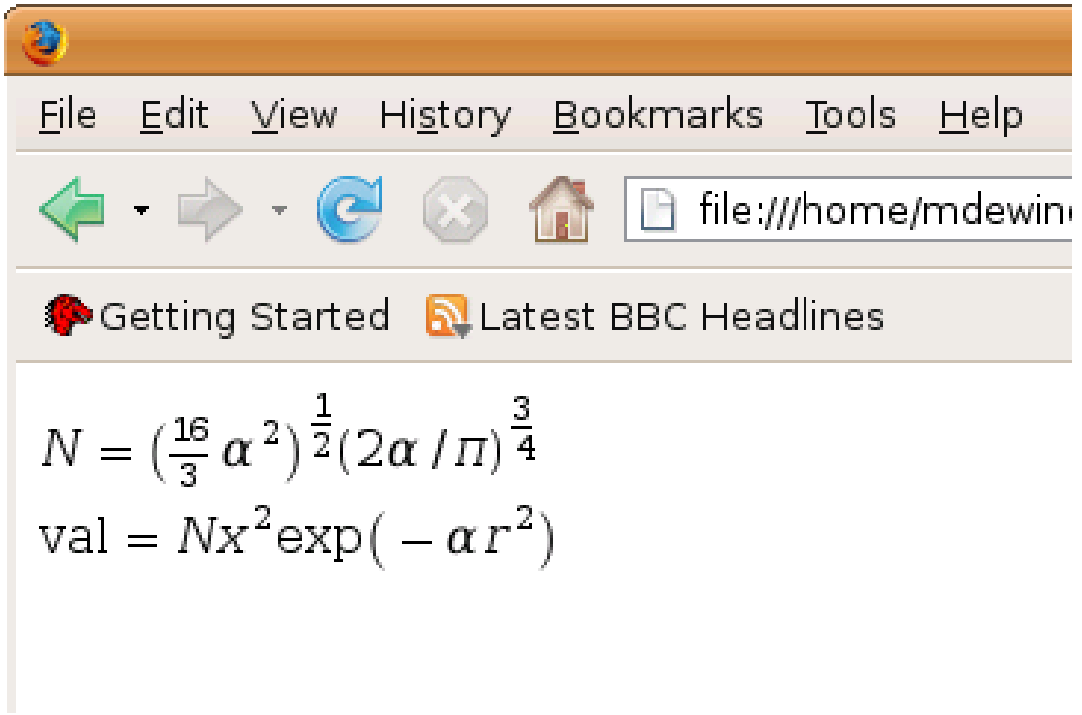
  https://quameon.svn.sourceforge.net/svnroot/quameon/trunk/codegen/primitive_gaussian
- pyparsing http://pyparsing.wikispaces.com/

  Used to parse templates

Backup

Presentation MathML

Display in browser



Solution: Use Sympy + code generator

- Compute derivatives and specialize with sympy
- Generate python code

Code generator:

```
deriv_stmt = sympy_to_py.convert([(“d”,sdf)])
template = parse_python.parse(template_data)      # uses pyparsing
lang_py.replace_template_node(template,'name','example')
lang_py.replace_template_node(template,'derivative',deriv_stmt)
print template.to_string()
```

Sympy:

```
f = sympify('x**i * exp(x**2)')
df = diff(f,x)
sdf = df.subs('i',0)
```

Python template:

```
function %name%:
   %derivative%
   return d
```

Output:

```
def example(x):
  d = 2 * x * math.exp(x**2)
  return d
```