A positional derivative package for Maxima

Barton Willis
University of Nebraska at Kearney
Kearney Nebraska

September 10, 2002

Introduction

Working with derivatives of unknown functions[1] can be cumbersome in Maxima. If we want, for example, the first order Taylor polynomial of $f(x+x^2)$ about $x=1$, we get

```
(c1)  taylor(f(x + x^2),x,1,1);
```
(d1)
$$f(2)+\left(\frac{d}{dx}f\left(x^2+x\right)\Big|_{x=1}\right)(x-1)+\cdots$$

To "simplify" the Taylor polynomial, we must assign a gradient to f

```
(c2)  gradef(f(x),df(x))$
(c3)  taylor(f(x+x^2),x,1,1);
```
(d3)
$$f(2)+3\,df(2)\,(x-1)+\cdots$$

This method works well for simple problems, but it is tedious for functions of several variables or high order derivatives. The positional derivative package `pdiff` gives an alternative to using `gradef` when working with derivatives of unknown functions.

[1] By *unknown function*, we mean a function that isn't bound to a formula and that has a derivative that isn't known to Maxima.

Installation

To install the positional derivative package, download `pdiff.lisp` from `www.unk.edu/acad/math/people/willisb` and place it in a directory that Maxima can find.

Usage

To use the positional derivative package, you must load it from a Maxima prompt. Assuming `pdiff.lisp` is in a directory that Maxima can find, this is done with the command

(c1) `load("pdiff.lisp")$`

Use the full pathname if Maxima can't find the file. Loading `pdiff.lisp` sets the option variable `use_pdiff` to true; when `use_diff` is true, Maxima will indicate derivatives of unknown functions positionally. To illustrate, the first three derivatives of f are

(c2) `[diff(f(x),x),diff(f(x),x,2), diff(f(x),x,3)];`

(d2)
$$\left[f_{(1)}(x), f_{(2)}(x), f_{(3)}(x)\right]$$

The subscript indicates the order of the derivative; since f is a function of one variable, the subscript has only one index. When a function has more than one variable, the subscript has an index for each variable

(c3) `[diff(f(x,y),x,0,y,1), diff(f(y,x),x,0,y,1)];`

(d3)
$$\left[f_{(0,1)}(x,y), f_{(1,0)}(y,x)\right]$$

Setting `use_pdiff` to false (either locally or globally) inhibits derivatives from begin computed positionally

(c4) `diff(f(x,x^2),x), use_pdiff : false;`

(d4)
$$\frac{d}{dx} f\left(x,x^2\right)$$

(c5) `diff(f(x,x^2),x), use_pdiff : true;`

(d5)
$$f_{(1,0)}(x,x^2) + 2x\, f_{(0,1)}(x,x^2)$$

Taylor polynomials of unknown functions can be found without using `gradef`. An example

(c6)
```
taylor(f(x+x^2),x,1,2);
```
(d6)

$$f(2)+3\,f_{(1)}(2)\,(x-1)+\frac{\left(2\,f_{(1)}(2)+9\,f_{(2)}(2)\right)\,(x-1)^2}{2}+\cdots$$

Additionally, we can verify that $y=f(x-ct)+g(x+ct)$ is a solution to a wave equation without using `gradef`

(c7)
```
y : f(x-c*t) + g(x+c*t)$
```
(c8)
```
ratsimp(diff(y,t,2) - c^2 * diff(y,x,2));
```
(d8)

$$0$$

(c9)
```
remvalue(y)$
```
Expressions involving positional derivatives can be differentiated

(c10)
```
e : diff(f(x,y),x);
```
(d10)

$$f_{(1,0)}(x,y)$$

(c11)
```
diff(e,y);
```
(d11)

$$f_{(1,1)}(x,y)$$

The chain rule is applied when needed

(c12)
```
[diff(f(x^2),x), diff(f(g(x)),x)];
```
(d12)

$$\left[2x\,f_{(1)}(x^2),g_{(1)}(x)\,f_{(1)}(g(x))\right]$$

The positional derivative package doesn't alter the way known functions are differentiated

(c13)
```
diff(exp(-x^2),x);
```
(d13)

$$-2xe^{-x^2}$$

To convert positional derivatives to standard Maxima derivatives, use `convert_to_diff`

(c14)
```
e : [diff(f(x),x), diff(f(x,y),x,1,y,1)];
```
(d14)

$$\left[f_{(1)}(x), f_{(1,1)}(x,y)\right]$$

(c15)
```
e : convert_to_diff(e);
```
(d15)

$$\left[\frac{d}{dx} f(x), \frac{d^2}{dy\,dx} f(x,y)\right]$$

To convert back to a positional derivative, use `ev` with `diff` as an argument

(c16)
```
ev(e,diff);
```
(d16)

$$\left[f_{(1)}(x), f_{(1,1)}(x,y)\right]$$

Conversion to standard derivatives sometimes requires the introduction of a dummy variable. Here's an example

(c17)
```
e : diff(f(x,y),x,1,y,1);
```
(d17)

$$f_{(1,1)}(x,y)$$

(c18)
```
e : subst(p(s),y,e);
```
(d18)

$$f_{(1,1)}(x,p(s))$$

(c19)
```
e : convert_to_diff(e);
```
(d19)

$$\left.\frac{d^2}{d\,\%x_0\,dx} f(x,\%x_0)\right|_{[\%x_0=p(s)]}$$

Dummy variables have the form ci, where i=0,1,2... and c is the value of the option variable `dummy_char`. The default value for `dummy_char` is `%x`. If a user variable conflicts with a dummy variable, the conversion process can give an incorrect value. To convert the previous expression back to a positional derivative, use `ev` with `diff` and `at` as arguments

(c20) `ev(e,diff,at);`
(d20)

$$f_{(1,1)}(x, p(s))$$

Maxima correctly evaluates expressions involving positional derivatives if a formula is later given to the function. (Thus converting an unknown function into a known one.) Here is an example; let

(c21) `e : diff(f(x^2),x);`
(d21)

$$2x\, f_{(1)}(x^2)$$

Now, give f a formula

(c22) `f(x) := x^5;`
(d22)

$$f(x) := x^5$$

and evaluate `e`

(c23) `ev(e);`
(d23)

$$10x^9$$

This result is the same as

(c24) `diff(f(x^2),x);`
(d24)

$$10x^9$$

In this calculation, Maxima first evaluates $f(x)$ to x^{10} and then does the derivative. Additionally, we can substitute a value for x before evaluating

(c25) `ev(subst(2,x,e));`
(d25)

$$5120$$

We can duplicate this with

(c26) `subst(2,x,diff(f(x^2),x));`

(d26)

$$5120$$

(c27)
```
remfunction(f);
```
(d27)

$$[f]$$

We can also evaluate a positional derivative using a local function definition

(c28)
```
e : diff(g(x),x);
```
(d28)

$$g_{(1)}(x)$$

(c29)
```
e, g(x) := sqrt(x);
```
(d29)

$$\frac{1}{2\sqrt{x}}$$

(c30)
```
e, g = sqrt;
```
(d30)

$$\frac{1}{2\sqrt{x}}$$

(c31)
```
e, g = h;
```
(d31)

$$h_{(1)}(x)$$

(c32)
```
e, g = lambda([t],t^2);
```
(d32)

$$2x$$

The `pderivop` function

If F is an atom and $i_1, i_2, \ldots i_n$ are nonnegative integers, then pderivop$(F, i_1, i_2, \ldots i_n)$, is the function that has the formula

$$\frac{\partial^{i_1+i_2+\cdots+i_n}}{\partial x_1^{i_1} \partial x_2^{i_2} \cdots \partial x_n^{i_n}} F(x_1, x_2, \ldots x_n).$$

If any of the derivative arguments are not nonnegative integers, we'll get an error

(c33)
```
pderivop(f,2,-1);
```
Each derivative order must be a nonnegative integer

The `pderivop` function can be composed with itself

(c34)
```
pderivop(pderivop(f,3,4),1,2);
```
(d34)

$$f_{(4,6)}$$

If the number of derivative arguments between two calls to `pderivop` isn't the same, Maxima gives an error

(c35)
```
pderivop(pderivop(f,3,4),1);
```
The function f expected 2 derivative argument(s), but it received 1

When `pderivop` is applied to a known function, the result is a lambda form[2]

(c37)
```
f(x) := x^2;
```
(d37)

$$f(x) := x^2$$

(c38)
```
df : pderivop(f,1);
```
(d38)

$$\lambda([Q_{1253}],2\,Q_{1253})$$

(c39)
```
apply(df,[z]);
```
(d39)

$$2z$$

(c40)
```
ddf : pderivop(f,2);
```
(d40)

$$\lambda([Q_{1254}],2)$$

(c41)
```
apply(ddf,[10]);
```
(d41)

$$2$$

(c42)
```
remfunction(f);
```

[2] If you repeat theses calculations, you may get a different prefix for the `gensym` variables.

(d42)

$$[f]$$

If the first argument to pderivop is a lambda form, the result is another lambda form

```
(c43)        f : pderivop(lambda([x],x^2),1);
```

(d43)

$$\lambda([Q_{1255}],2\,Q_{1255})$$

```
(c44)       apply(f,[a]);
```

(d44)

$$2\,a$$

```
(c45)       f : pderivop(lambda([x],x^2),2);
```

(d45)

$$\lambda([Q_{1256}],2)$$

```
(c46)       apply(f,[a]);
```

(d46)

$$2$$

```
(c47)       f : pderivop(lambda([x],x^2),3);
```

(d47)

$$\lambda([Q_{1257}],0)$$

```
(c48)       apply(f,[a]);
```

(d48)

$$0$$

```
(c49)       remvalue(f)$
```

If the first argument to pderivop isn't an atom or a lambda form, Maxima will signal an error

```
(c50)         pderivop(f+g,1);
```

Non-atom g+f used as a function

You may use `tellsimpafter` together with `pderivop` to give a value to a derivative of a function at a point; an example

```
(c51)  tellsimpafter(pderivop(f,1)(1),1)$
(c52)  tellsimpafter(pderivop(f,2)(1),2)$
(c53)  diff(f(x),x,2) + diff(f(x),x)$
(c54)  subst(1,x,%);
(d54)
```

$$3$$

This technique works for functions of several variables as well

```
(c55)  kill(rules)$
(c56)  tellsimpafter(pderivop(f,1,0)(0,0),a)$
(c57)  tellsimpafter(pderivop(f,0,1)(0,0),b)$
(c58)  sublis([x = 0, y = 0], diff(f(x,y),x) + diff(f(x,y),y));

(d58)
```

$$b+a$$

TEX-ing positional derivatives

Several option variables control how positional derivatives are converted to TEX. When the option variable `tex_uses_prime_for_derivatives` is true (default false), makes functions of one variable TEX as superscripted primes

```
(c59)  tex_uses_prime_for_derivatives : true$
(c60)  tex(makelist(diff(f(x),x,i),i,1,3))$
(d60)
```

$$\left[f'(x), f''(x), f'''(x)\right]$$

```
(c61)  tex(makelist(pderivop(f,i),i,1,3))$
```

$$\left[f', f'', f'''\right]$$

When the derivative order exceeds the value of the option variable `tex_prime_limit`, (default 3) derivatives are indicated with parenthesis delimited superscripts

```
(c62)  tex(makelist(pderivop(f,i),i,1,5)), tex_prime_limit : 0$
```

$$\left[f^{(1)}, f^{(2)}, f^{(3)}, f^{(4)}, f^{(5)}\right]$$

(c63) `tex(makelist(pderivop(f,i),i,1,5)), tex_prime_limit : 5$`

$$\left[f', f'', f''', f'''', f'''''\right]$$

The value of `tex_uses_prime_for_derivatives` doesn't change the way functions of two or more variables are converted to TEX.

(c64) `tex(pderivop(f,2,1));`

$$f_{(2,1)}$$

When the option variable `tex_uses_named_subscripts_for_derivatives` (default false) is true, a derivative with respect to the i-th argument is indicated by a subscript that is the i-th element of the option variable `tex_diff_var_names`. An example is the clearest way to describe this.

(c65) `tex_uses_named_subscripts_for_derivatives : true$`

(c66) `tex_diff_var_names;`

(d66)

$$[x,y,z]$$

(c67) `tex([pderivop(f,1,0), pderivop(f,0,1), pderivop(f,1,1), pderivop(f,2,0)]);`

$$[f_x, f_y, f_{xy}, f_{xx}]$$

(c68) `tex_diff_var_names : [a,b];`

(d68)

$$[a,b]$$

(c69) `tex([pderivop(f,1,0), pderivop(f,0,1), pderivop(f,1,1), pderivop(f,2,0)]);`

$$[f_a, f_b, f_{ab}, f_{aa}]$$

(c70)
```
tex_diff_var_names : [x,y,z];
```
(d70)

$$[x,y,z]$$

(c71)
```
tex([diff(f(x,y),x), diff(f(y,x),y)]);
```

$$[f_x(x,y), f_x(y,x)]$$

When the derivative order exceeds tt tex_prime_limit, revert to the default method for converting to TEX

(c72)
```
tex(diff(f(x,y,z),x,1,y,1,z,1)), tex_prime_limit : 4$
```

$$f_{xyz}(x,y,z)$$

(c73)
```
tex(diff(f(x,y,z),x,1,y,1,z,1)), tex_prime_limit : 1$
```

$$f_{(1,1,1)}(x,y,z)$$

A longer example

We'll use the positional derivative package to change the independent variable of the differential equation

(c74)
```
de :  4*x^2*'DIFF(y,x,2) + 4*x*'DIFF(y,x,1) + (x-1)*y = 0;
```

(d74)

$$4x^2\left(\frac{d^2}{dx^2}y\right)+4x\left(\frac{d}{dx}y\right)+(x-1)\,y=0$$

With malice aforethought, we'll assume a solution of the form $y=g(x^n)$, where n is a number. Substituting $y\rightarrow g(x^n)$ in the differential equation gives

(c75)
```
de : subst(g(x^n),y,de);
```
(d75)

$$4x^2\left(\frac{d^2}{dx^2}g\left(x^n\right)\right)+4x\left(\frac{d}{dx}g\left(x^n\right)\right)+(x-1)\,g\left(x^n\right)=0$$

(c76) `de : ev(de, diff);`
(d76)

$$4x^2\left(n^2x^{2n-2}g''(x^n)+(n-1)\,nx^{n-2}g'(x^n)\right)+4nx^ng'(x^n)+(x-1)\,g(x^n)=0$$

Now let $x\rightarrow t^{1/n}$
(c77) `de : radcan(subst(x^(1/n),x, de));`
(d77)

$$4n^2x^2g''(x)+4n^2xg'(x)+\left(x^{\frac{1}{n}}-1\right)g(x)=0$$

Setting $n\rightarrow 1/2$, we recognize that g is the order 1 Bessel equation
(c78) `subst(1/2,n, de);`
(d78)

$$x^2g''(x)+xg'(x)+\left(x^2-1\right)g(x)=0$$

Limitations

- Positional derivatives of subscripted functions are not allowed.
- Derivatives of unknown functions with symbolic orders are not computed positionally.
- The positional derivative package has only been tested under Maxima 5.6 compiled with GCL 2.4; it has not been tested with any other Maxima version.
- The `pdiff.lisp` code alters the Maxima functions `mqapply` and `sdiffgrad` Although I'm unaware of any problems associated with these altered functions, there may be some. Setting `use_pdiff` to false should restore `mqapply` and `sdiffgrad` to their original functioning.

Conclusion

The `pdiff` package provides a simple way of working with derivatives of unknown functions. If you find a bug in the package, or if you have a comment or a question, please send it to `willisb@unk.edu`.

The `pdiff` package could serve as a basis for a Maxima package differential and integral operators.