HW06 Derivative Rules

October 19, 2024

0.1 Derivative Rules Homework

Christopher Taylor Activity 1

a) $15x^4 - 20x$

b) $(60x^{11})(\pi - \pi^2 x^4) + (5x^{12} + 2)(-4\pi^2 x^3)$

c) $\frac{1}{2}u^{-1/2} + 9u^{-4} + 14u^6$

d) m

e) $0.5cosx + \frac{3}{2}x^{-1/3}$

f) $\frac{(5x^{12}+2)(-4\pi^2 x^3)-(\pi-\pi^2 x^4)(60x^{11})}{(5x^{12}+2)^2}$

g) $\frac{1}{\sqrt{x}} + \frac{1}{2x\sqrt{x}}$

h) $(sec^2 z)(sinz - 5) + tanz(cosz)$

i) $\frac{(x^2)(cosx)-(sinx)(2x)}{(x^2)^2}$

j) $2xe^x + x^2 e^x$

k) $-sinx + e^x$

l) $sec^2 x$

m) $e^x lnx + \frac{e^x}{x}$

n) $\frac{(10+sinx)(2^x lnx)-(2^x)(cosx)}{(10+sinx)^2}$

o) $cos(e^x cosx)(e^x cosx - e^x sinx)$

p) $\frac{6}{5}e^{cost}((-sint)(t^{-1/3}) - \frac{1}{3}t^{-4/3})$

q) $\frac{2x+e^x+xe^x}{x^2+xe^x}$

r) $\frac{(7\sqrt{x}+5)(10x+\frac{1}{x})-(5x^2+lnx)(3.5x^{-1/2})}{(7\sqrt{x}+5)^2}$

Activity 2

a) f'(2)=2 , g'(2)=-1 ; 2 + (-1) = 1

b) f'(2)=2 , g'(2)=-1 ; 5(2)-2(-1) -> 10 - (-2) = 12

c) f(2)=3 , g(2)=4 ; (3)(4) = 12

d) f(2)=3 , g(2)=4 ; (3)/(4) = 3/4

e) f(2)=3 , g(3)=2 ; so this is 2

f) g(2)=4 , $\sqrt{4} = 2$

g) t=2 , f(2)=3 ; $(2^2)(3) = 12$

h) f(2)=3 , g(2)=4 , $(3)^2 + (4)^2 = 9 + 16 = 25$

i) f(2)=3 , $\frac{1}{3}$

j) t=2, f(3(2) - $(g(1+2))^2) = f(6-(g(3))^2) = f(6-(2)^2) = f(6-4) = f(2) = 3$

k) we would need to know what f'(4) equals

l) $\sim\frac{3}{4}$

[15]:
```python
# Activity 3

import sympy as sp

# defining symbol for the functions
x = sp.Symbol("x")

# basic derivative rules in a dictionary
basic_derivs = {
    "x": lambda: sp.diff(x, x),
    "sin(x)": lambda: sp.diff(sp.sin(x), x),
    "cos(x)": lambda: sp.diff(sp.cos(x), x),
    "e^x": lambda: sp.diff(sp.exp(x), x),
    "ln(x)": lambda: sp.diff(sp.ln(x), x),
    "c": lambda: sp.diff(sp.Symbol("c"), x)
}

# function to compute derivative of basic function or expression
def deriv(exp):
    # conversion of string expression to sympy expression
    if isinstance(exp, str):
        exp = sp.sympify(exp)

    # checking for functions in dictionary
    if exp in basic_derivs:
        return basic_derivs[exp]()

    # if expression is polynomial or any other form, compute its derivative␣
↪with sympy
    try:
        result = sp.diff(exp, x)
        return result
    except:
```

```python
        raise ValueError(f"Expression {exp} is not recognized.")

# algebraic combination rules:
def add_deriv(f, g):
    return deriv(f) + deriv(g)

def sub_deriv(f, g):
    return deriv(f) - deriv(g)

def mult_deriv(f, g):
    #conversion from strings into sympy expressions
    f = sp.sympify(f)
    g = sp.sympify(g)

    fp = deriv(f)
    gp = deriv(g)
    return fp * g + f * gp

def div_deriv(f, g):
    #conversion from strings into sympy expressions
    f = sp.sympify(f)
    g = sp.sympify(g)

    fp = deriv(f)
    gp = deriv(g)
    return (g * fp - f * gp) / g**2

# defining chain rule
def chain(f, g):
    f = sp.sympify(f)
    g = sp.sympify(g)
    gp = sp.diff(g, x)
    fp = deriv(f)
    return gp.subs(x, f) * fp
```

[16]:
```python
# testing the above functions for proper/correct execution
f = "2*x**2 + 3*x + 1"
g = "sin(x)"

# basic function derivatives:
print("Derivative of f:", deriv(f))
print("Derivative of g:", deriv(g))

# algebraic combinations of derivative functions
print("Sum of derivatives:", add_deriv(f, g))
print("Product of derivatives:", mult_deriv(f, g))
print("Quotient of derivatives:", div_deriv(f, g))
```

```python
# chain rule testing
print("Chain of derivatives:", chain(f, g))
```

```
Derivative of f: 4*x + 3
Derivative of g: cos(x)
Sum of derivatives: 4*x + cos(x) + 3
Product of derivatives: (4*x + 3)*sin(x) + (2*x**2 + 3*x + 1)*cos(x)
Quotient of derivatives: ((4*x + 3)*sin(x) - (2*x**2 + 3*x +
1)*cos(x))/sin(x)**2
Chain of derivatives: (4*x + 3)*cos(2*x**2 + 3*x + 1)
```

[ ]:

```python

```