notebook001

March 20, 2025

1 Funkcja f(x) = x * ln(x)^4

[121]:
```
import sympy as sp

sp.init_printing()
```

[122]:
```
x = sp.symbols('x')
definition = x * (sp.log(x) ** 4)
```

1.1 Zadanie 1

Wyznacz dziedzinę funkcji i asymptoty, jeśli są.

[123]:
```
domain = sp.calculus.util.continuous_domain(definition, x, sp.Reals)
asymptote_to_infinite = sp.limit(definition, x, sp.oo)
asymptote_to_negative_infinite = sp.limit(definition, x, -sp.oo)
asymptote_horizontal = sp.limit(definition, x, 0)
```

[124]:
```
print(f'Dziedzina funkcji: {domain}')
print(f'Asymptota pionowa: {asymptote_to_infinite}')
print(f'Asymptota pozioma: {asymptote_horizontal}')
```

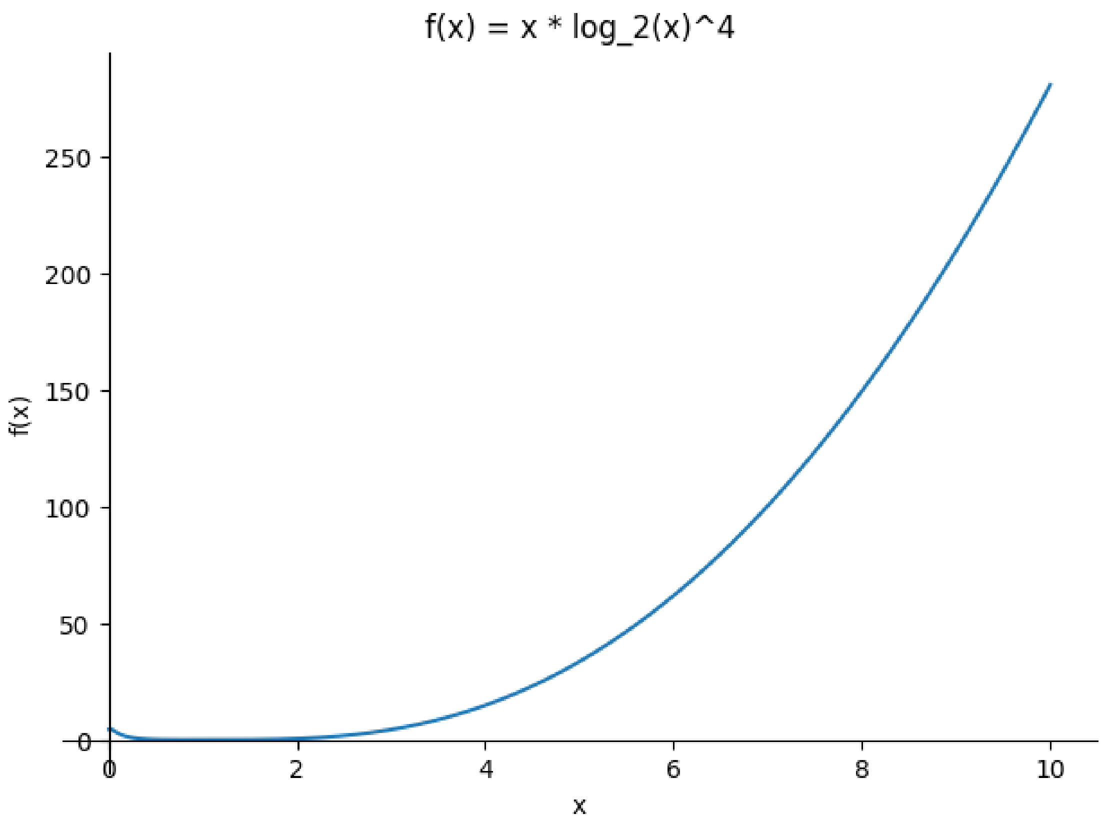
```
Dziedzina funkcji: Interval.open(0, oo)
Asymptota pionowa: oo
Asymptota pozioma: 0
```

1.2 Zadanie 2

Narysuj wykres funkcji.

[125]:
```
plot_title = 'f(x) = x * log_2(x)^4'
x_label = 'x'
y_label = 'f(x)'
step = 0.01
end_value = 10

sp.plot(definition, (x, step, end_value), title=plot_title, xlabel=x_label,␣
 ↪ylabel=y_label)
```



[125]: <sympy.plotting.backends.matplotlibbackend.matplotlib.MatplotlibBackend at 0x73323bc371a0>

1.3 Zadanie 3

Obliczyć pochodne f(k), k = 1, 2, 3, 4 a następnie obliczyć wartości tych pochodnych w podanych punktach

[126]:
```python
first_derivative = sp.diff(definition)
second_derivative = sp.diff(first_derivative)
third_derivative = sp.diff(second_derivative)
fourth_derivative = sp.diff(third_derivative)
```

[127]:
```python
print(f'Pierwsza pochodna (1.11) {first_derivative.subs(x, 1.11).evalf()}')
print(f'Druga pochodna (2.22) {second_derivative.subs(x, 2.22).evalf()}')
print(f'Trzecie pochodna (3.33) {third_derivative.subs(x, 3.33).evalf()}')
print(f'Czwarta pochodna (4.44) {fourth_derivative.subs(x, 4.44).evalf()}')
```

```
Pierwsza pochodna (1.11) 0.00466495920995858
Druga pochodna (2.22) 4.35185927358923
Trzecie pochodna (3.33) 1.97565552576735
Czwarta pochodna (4.44) -0.545166487330982
```

1.4 Zadanie 4

Wyznaczyć przedziały monotoniczności funkcji.

[128]:
```
increasing_intervals = sp.solve(first_derivative > 0, x)
decreasing_intervals = sp.solve(first_derivative < 0, x)

print(f'Przedziały, na których funkcja jest rosnąca: {increasing_intervals}')
print(f'Przedziały, na których funkcja jest malejąca: {decreasing_intervals}')
```

```
Przedziały, na których funkcja jest rosnąca: (1 < x) | ((0 < x) & (x < exp(-4)))
Przedziały, na których funkcja jest malejąca: (x < 1) & (exp(-4) < x)
```

1.5 Zadanie 5

Kiedy funkcja osiąga ekstrema?

[129]:
```
derivative_zeroes = sp.solve(first_derivative, x)

min_value = definition.subs(x, derivative_zeroes[0]).evalf()
max_value = definition.subs(x, derivative_zeroes[0]).evalf()
min_for = derivative_zeroes[0]
max_for = derivative_zeroes[0]

for zero in derivative_zeroes:
    result = definition.subs(x, zero).evalf()
    if min_value > result:
        min_value = result
        min_for = zero

    if max_value < result:
        max_value = result
        max_for = zero

print(f'Maksimum dla {max_for}: {max_value}')
print(f'Minimum dla {min_for}: {min_value}')
```

```
Maksimum dla exp(-4): 4.68880355551595
Minimum dla 1: 0
```

1.6 Zadanie 6

Wyznaczyć przedziały wypukłości i wklęsłości wykresu funkcji

[130]:
```
potential_inflection_points = sp.solve(second_derivative, x)
potential_inflection_points = sorted(potential_inflection_points, key=lambda p:␣
↪float(p.evalf()))
print("Punkty przegięcia:", potential_inflection_points)
```

```
intervals = [(0, potential_inflection_points[0]),␣
↪(potential_inflection_points[0], potential_inflection_points[1]),
             (potential_inflection_points[1], float('inf'))]
print(f"Przedziały: {intervals}")


def get_test_point(interval):
    start, end = interval
    if end == float('inf'):
        return start + 1  # Dla nieskończoności punkt przesunięty o 1
    else:
        return (start + end) / 2


test_points = [get_test_point(interval) for interval in intervals]
print(f"Punkty testowe: {test_points} \n")
```

```
Punkty przegięcia: [exp(-3), 1]
Przedziały: [(0, exp(-3)), (exp(-3), 1), (1, inf)]
Punkty testowe: [exp(-3)/2, exp(-3)/2 + 1/2, 2]
```

[131]:
```
print("\nOdpowiedź:")
for i, interval in enumerate(intervals):
    value = second_derivative.subs(x, test_points[i]).evalf()
    if value > 0:
        print(f"Przedział {interval}: wypukła (f'' > 0)")
    elif value < 0:
        print(f"Przedział {interval}: wklęsła (f'' < 0)")
    else:
        print(f"Przedział {interval}: f'' = 0 (sprawdź dokładniej)")
```

```
Odpowiedź:
Przedział (0, exp(-3)): wklęsła (f'' < 0)
Przedział (exp(-3), 1): wypukła (f'' > 0)
Przedział (1, inf): wypukła (f'' > 0)
```

1.7 Zadanie 7

Wykres funkcji f ma punkty przegięcia

Punkty przegięcia to miejsce w którym druga pochodna funkcji wynosi 0 lub jest niezdefiniowana, a dodatkowo następuje zmiana znaku.

[132]:
```
for i, point in enumerate(test_points):
    value = second_derivative.subs(x, point).evalf()
    sign = '+' if value > 0 else '-' if value < 0 else '0'
    print(f"Przedział {intervals[i]}: f''({point}) = {value}, znak: {sign}")
```

```
Przedział (0, exp(-3)): f''(exp(-3)/2) = -1519.12015235506, znak: -
Przedział (exp(-3), 1): f''(exp(-3)/2 + 1/2) = 7.45739790882108, znak: +
Przedział (1, inf): f''(2) = 3.54876738748707, znak: +
```

[133]:
```
print(f'Zmiana znaku następuje w {test_points[0]}, co oznacza, że występuje tu␣
 ↪punkt przegięcia.')
```

```
Zmiana znaku następuje w exp(-3)/2, co oznacza, że występuje tu punkt
przegięcia.
```

1.8 Zadanie 8

Pole obszaru D ograniczonego wykresem funkcji y = f(x) i prostymi y = 0, x = 1, x = 7

f(x) > 0 dla x > 0

[134]:
```
# Całka oznaczona od 1 do 7
definite_integral = sp.integrate(definition, (x, 1, 7))
print("Pole obszaru D:", definite_integral)

# Wartość numeryczna
pole_numeric = definite_integral.evalf()
print("Pole obszaru D (numerycznie):", pole_numeric)
```

```
Pole obszaru D: -49*log(7)**3 - 147*log(7)/2 + 36 + 147*log(7)**2/2 +
49*log(7)**4/2
Pole obszaru D (numerycznie): 161.523723002576
```

1.9 Zadanie 9

Długość łuku L określanego wzorem L : y = f(x), x należącego do przedziału [1, 7] wynosi

[135]:
```
integrand = sp.sqrt(1 + first_derivative ** 2)
length = sp.integrate(integrand, (x, 1, 7))
length_numeric = length.evalf()
print("Długość łuku L:", length_numeric)
```

```
Długość łuku L: 101.331959288273
```

1.10 Zadanie 10

Objętość bryły Ωx powstałej w wyniku obrotu figury T : 0 <= y <= f(x), 1 <= x <= 7 wokół osi Ox wynosi

[136]:
```
integrand_solid = definition ** 2
definite_integral_solid = sp.integrate(integrand_solid, (x, 1, 7))
volume = sp.pi * definite_integral_solid
volume_numeric = volume.evalf()

print("Objętość bryły Ωx:", volume_numeric)
```

```
Objętość bryły Ωx: 29829.7214762937
```