```python
# Finding_Minima_Cubic.py

import sympy as sp

def find_cubic_minimum(a, b, c, d, sigma_a, sigma_b, sigma_c, sigma_d):
    # Define symbols
    x = sp.Symbol('x')

    # Define the cubic function
    cubic = a * x**3 + b * x**2 + c * x + d

    # Compute the derivative
    derivative = sp.diff(cubic, x)

    # Solve for critical points
    critical_points = sp.solve(derivative, x)

    # Evaluate the cubic at the critical points to find the y values
    y_values = [cubic.subs(x, cp) for cp in critical_points]

    # Determine the minimum y value and its corresponding x value (minimum point)
    min_y = min(y_values)
    min_x = critical_points[y_values.index(min_y)]

    # Uncertainty propagation
    sigma_y = sp.sqrt((sigma_a**2 + (sigma_b * min_x)**2 + (sigma_c * min_x**2)**2).evalf())
    sigma_x = sigma_b * (sigma_b * min_x)**2 / (2 * sp.sqrt(a * (a * min_x**2 + b * min_x + c)))

    return (min_x, min_y), (sigma_x, sigma_y)

# Example usage:
a = 2.6448e-12
b = -2.7265e-10
c = 9.2552e-09
d =  -1.0215e-07
sigma_a = -3.4909e-10
sigma_b = -3.1985e-09
```

```
sigma_c = -1.5194e-07
sigma_d = 7.3801e-11

min_point, uncertainties = find_cubic_minimum(a, b, c, d, sigma_a, sigma_b, sigma_c, sigma_d)
print("Minimum point (x, y):", min_point)
print("Uncertainties (delta_x, delta_y):", uncertainties)
```