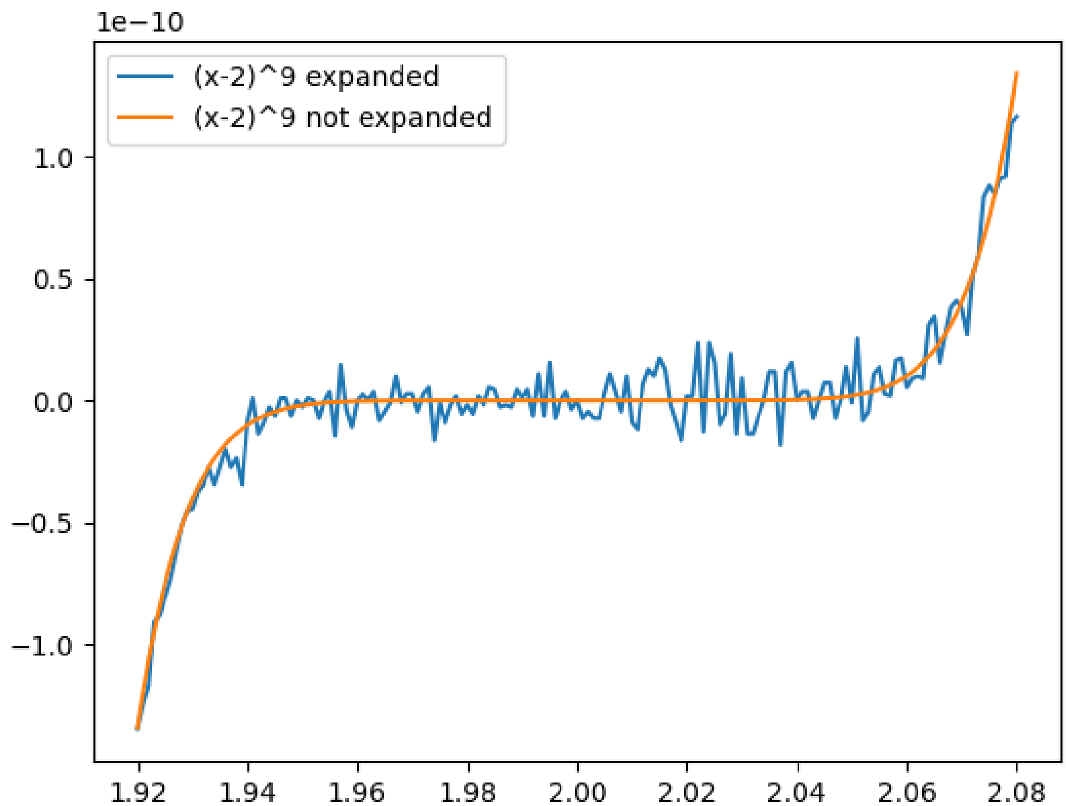
1. i. and ii.



   iii. The difference is that the expanded version has relatively large fluctuations for each increment of x while the non expanded version is smooth. This is because the non expanded version has to add and subtract very large numbers so it loses digits at the end, and since the result is very close to 0 those lost digits have a significant effect. The non expanded version is correct.

2. i. Multiply by the conjugate (sqrt(x+1)+1)/(sqrt(x+1)+1). This will result in x/(sqrt(x+1)+1) which will properly evaluate to near 0 for x close to 0.

   ii. Multiply by the conjugate (sin(x)+sin(y))/(sin(x)+sin(y)). This results in ((sin(x))^2 - (sin(y))^2)/(sin(x)+sin(y)) which will properly evaluate to near 0 when x is close to y.

   iii. Using L'Hospital's rule, the limit of this as x approaches 0 is the same as the limit of sin(x)/cos(x) as x approaches 0 which is 0.

3. P2(x) about x = 0: f(0) + f'(0)(x-0) + f''(0)(x-0)^2/2! = 1 + x - x^2/2

   a. P2(0.5) = 1 + 0.5 + 0.5^2/2 = 1.125
      f(0.5) = (1 + 0.5 + 0.5^3)(cos(0.5)) = 1.121

Actual relative error = 0.004/1.121 = 0.00357
Upper bound for error = (max(f’’’(x)[0,0.5])(abs(0.5-0)^3)/3! = 0.0627
The upper bound is much larger than the absolute error

b. error(x) = (max(f’’’(x)[0,x])(abs(x-0)^3)/3!

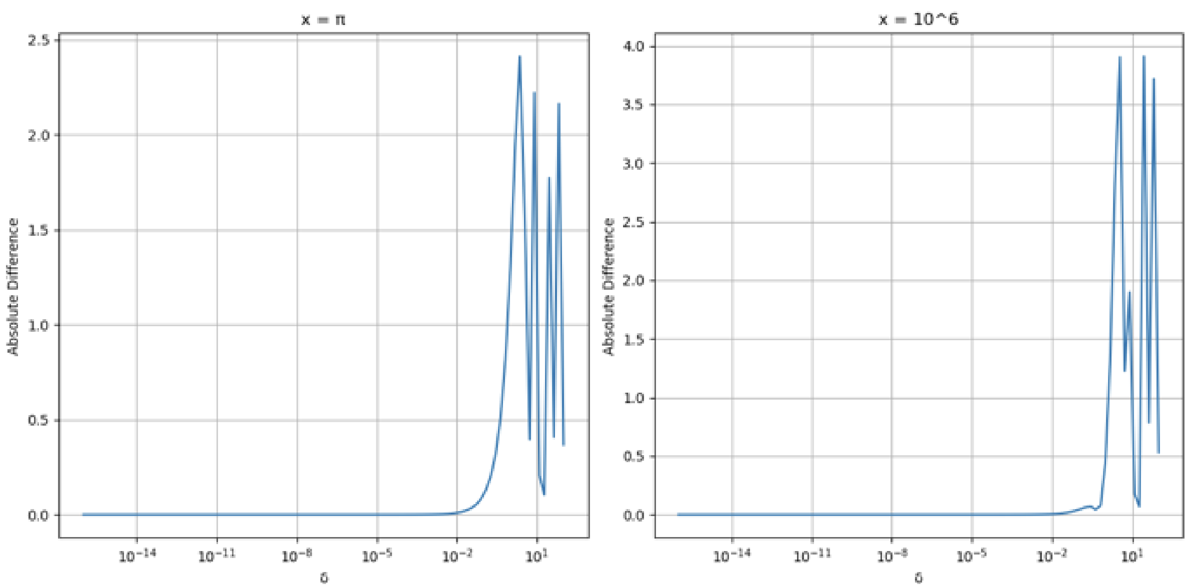
c. integral(f(x))[0,1] = 1.396
integral(P2(x))[0,1] = x + x^2/2 - x^3/6 = 1.333

d. Relative error of integral = 0.063/1.396 = 0.0451

4. a. R1: (-56 - sqrt(-56^2-4))/2 = three decimals: -55.982 full: -55.98213716
R2: (-56 + sqrt(-56^2-4))/2 = three decimals: -0.018 full: -0.0178628407
Relative error of R1 = 1.372 * 10^-4 / 55.98213716 = 2.45 * 10^-6
Relative error of R2 = 1.372 * 10^-4 / -0.0178628407 = 0.00768

b. By Vieta’s formula, R1 + R2 = -b/a. Since R2 has a much larger error, we can calculate R1 and use this formula to get R2. R2 = 55.98213716 - (56/1) = -0.01786284. The other relation is R1*R2 = c/a.

5. a. Relative error is very large when x is very small
b.



c. The algorithm I would use is approximation = delta * derivative + (delta**2 / 2) * (-np.sin(x + np.random.uniform(0, delta)))