HW 3

Kevin Lin

1/30/2026

1

Let $X \sim \text{Unif}[-1,1], Z \sim \mathbb{N}(0,1), Y = XZ^2$, independently.

(i) We can find the CEF $E[Y|X]$ as follows:

$$\begin{aligned} E[Y|X] &= E[XZ^2|X] \\ &= XE[Z^2|X] \\ &= XE[Z^2] \\ &= X \cdot \text{Var}(Z) \\ &= X \cdot 1 \\ &= X \end{aligned}$$

The conditional variance $\text{Var}(Y|X)$ is:

$$\begin{aligned} \text{Var}(Y|X) &= \text{Var}(XZ^2|X) \\ &= X^2\text{Var}(Z^2|X) \\ &= X^2\text{Var}(Z^2) \\ &= X^2(E[Z^4] - (E[Z^2])^2) \\ &= X^2(3-1) \\ &= 2X^2 \end{aligned}$$

(ii) We can find the best linear projection of the CEF $\beta_0 + \beta_1 X$ as follows:

$$\begin{aligned}
\beta_1 &= \frac{\mathrm{Cov}(Y, X)}{\mathrm{Var}(X)} \\
&= \frac{E[YX] - E[Y]E[X]}{\mathrm{Var}(X)} \\
&= \frac{E[X^2 Z^2] - 0 \cdot 0}{\mathrm{Var}(X)} \\
&= \frac{E[X^2]E[Z^2]}{\mathrm{Var}(X)} \\
&= \frac{\frac{1}{3} \cdot 1}{\frac{1}{3}} \\
&= 1 \\
\beta_0 &= E[Y] - \beta_1 E[X] \\
&= 0 - 1 \cdot 0 \\
&= 0
\end{aligned}$$

Given that the CEF is $E[Y|X] = X$, the linear projection perfectly matches the CEF.

(iii) If we observe n i.i.d samples of (X_i, Y_i), the asymptotic variance of the fitted coefficients $\hat{\beta}_0, \hat{\beta}_1$ are given by:

$$\begin{aligned}
AV(\hat{\beta}) &= (E[XX'])^{-1} E[\mathrm{Var}(Y|X)XX'](E[XX'])^{-1} \\
&= \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} E[\mathrm{Var}(Y|X)] & E[X\mathrm{Var}(Y|X)] \\ E[X\mathrm{Var}(Y|X)] & E[X^2\mathrm{Var}(Y|X)] \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \\
&= \begin{pmatrix} E[\mathrm{Var}(Y|X)] & 3E[X\mathrm{Var}(Y|X)] \\ 3E[X\mathrm{Var}(Y|X)] & 9E[X^2\mathrm{Var}(Y|X)] \end{pmatrix} \\
&= \begin{pmatrix} E[2X^2] & 3E[X \cdot 2X^2] \\ 3E[X \cdot 2X^2] & 9E[X^2 \cdot 2X^2] \end{pmatrix} \\
&= \begin{pmatrix} 2/3 & 0 \\ 0 & 18/5 \end{pmatrix}
\end{aligned}$$

Consequently with n i.i.d samples, the variances are:

$$\begin{aligned}
AV(\hat{\beta}_0) &= \frac{2}{3n} \\
AV(\hat{\beta}_1) &= \frac{18}{5n}
\end{aligned}$$

(iv) Under the standard assumptions, we would assume that $Var(Y|X) = \sigma^2$, which is a constant. Consequently, the asymptotic variance of the fitted coefficients would be:

$$\begin{aligned} AV(\hat{\beta}) &= (E[XX'])^{-1}E[\mathrm{Var}(Y|X)XX'](E[XX'])^{-1} \\ &= \sigma^2(E[XX'])^{-1} \\ &= \sigma^2 \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \end{aligned}$$

We can estimate σ^2 with $E[\mathrm{Var}(Y|X)]$ as follows:

$$\begin{aligned} \sigma^2 &\approx E[\mathrm{Var}(Y|X)] \\ &= E[2X^2] \\ &= \frac{2}{3} \end{aligned}$$

Thus, the asymptotic variances would be:

$$\begin{aligned} AV(\hat{\beta}_0) &\approx \frac{2}{3n} \\ AV(\hat{\beta}_1) &\approx \frac{2}{n} \end{aligned}$$

The discrepancy in $AV(\hat{\beta}_1)$ arises because the standard assumption of homoskedasticity is violated in this case since $\mathrm{Var}(Y|X)$ is not constant. The variance $\mathrm{Var}(Y|X)$ grows quadratically with X, leading to a larger asymptotic variance which affects OLS disproportionately. Interestingly, by chance, the estimate for $AV(\hat{\beta}_0)$ remains accurate due to the symmetry of X around 0, which causes the heteroskedasticity to have a negligible effect on the intercept term.

2

(i) See attached Jupyter notebook for code and plots.

(ii) The sandwich bands are noticeably wider in each set of pointwise and simultaneous bands. This is because the sandwich bands account for potential heteroskedasticity in the error terms, which increases the uncertainty in the estimates. In contrast, the standard bands assume homoskedasticity, which can lead to narrower bands that may underestimate the true variability of the estimates. In general as well, the Bonferroni simultaneous bands are wider than the pointwise bands, since they account for the multiple comparisons problem by adjusting the confidence level for each individual interval to maintain an overall confidence. We can also see that the confidence bands are much wider on the right, as due to our data generating following $y = 2^x$, the variance of Y increases exponentially with X. This leads to greater uncertainty in the estimates as the OLS extrapolates badly at larger values of x.

3

(i) See attached Jupyter notebook for code and plots. Immediately we see that coverage is terrible because the true regression line is non-linear, while OLS is fitting a linear model. Consequently, the confidence bands will rarely cover the true regression line at all X_i. Similar to problem 2, we see that the sandwich bands are wider than the standard bands, and the simultaneous bands are wider than the pointwise bands. Likewise, we see that coverage gets worse as X increases due to the exponential increase in variance of Y.

(ii) See attached Jupyter notebook for code. We see that none of the pointwise confidence bands contain the true CEF at all X_i, while only 3 out of 1000 simultaneous confidence bands contain the true CEF at all X_i. This poor coverage is due to the fact that the true CEF is non-linear, while OLS is fitting a linear model, as explained in (i). Mathematically, the confidence bands assume that estimator $-$ target $=$ error, however because we have a x^3 term in the true CEF, the bias is ommitted from the equation, leading to poor coverage. Even as n increases and the sampling error decreases, the bias remains, leading to persistent undercoverage. This even occurs with the sandwich bonds, as robustness only handles heteroskedasticity, not model misspecification and bias.

(iii) In a pointwise coverage, we can see that this statistic is much better, however no CI reaches the nominal 95% coverage. This is because while pointwise coverage is easier to achieve than simultaneous coverage, the bias from model misspecification still leads to undercoverage. Even as n increases, the bias remains, leading to persistent undercoverage. The Bonferroni bands only improve coverage because they are wider due to the higher variance.

hw3_code

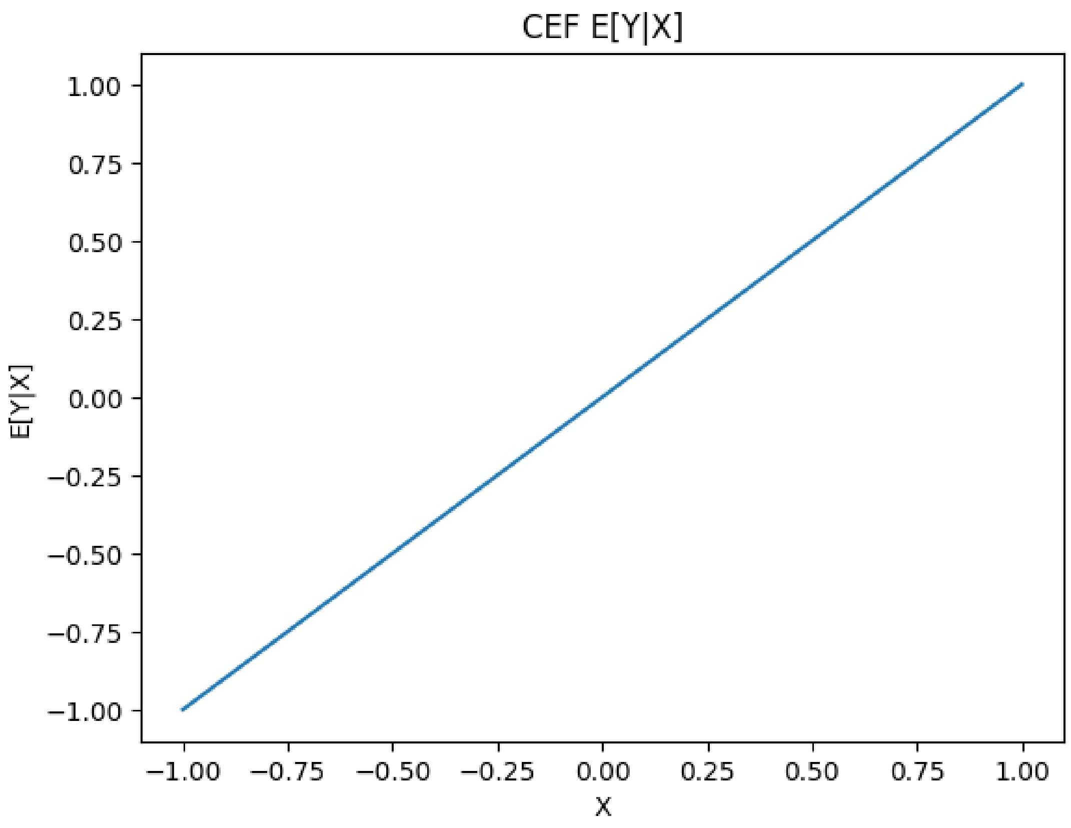
January 27, 2026

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

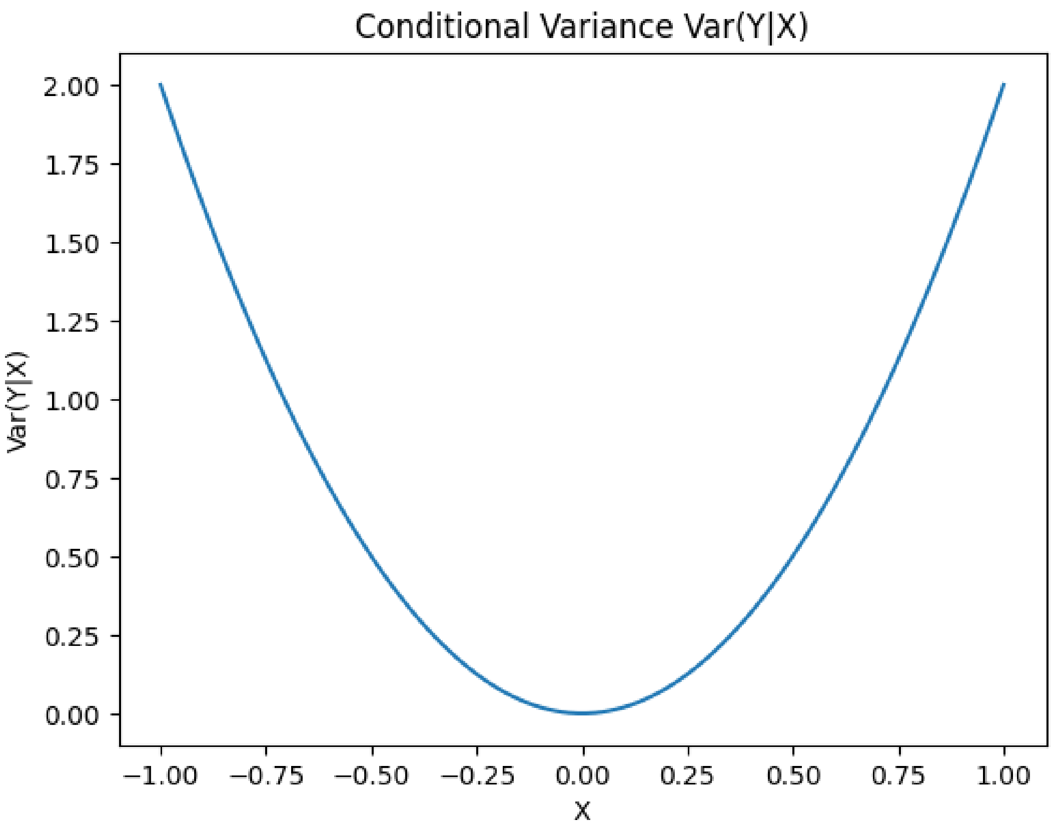
1 1

1.1 i

[ ]:
```
plt.plot(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100))
plt.title("CEF E[Y|X]")
plt.xlabel("X")
plt.ylabel("E[Y|X]")
plt.show()
```

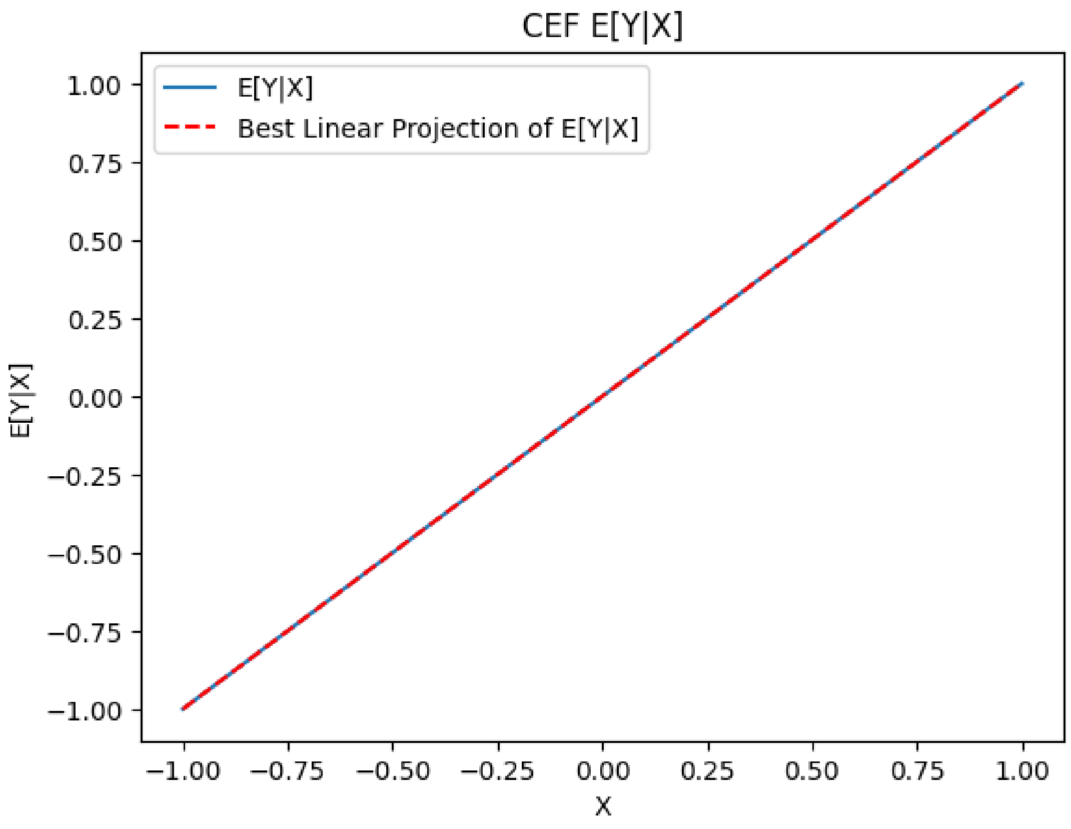


```
[4]: plt.plot(np.linspace(-1, 1, 100), 2 * np.linspace(-1, 1, 100)**2)
     plt.title("Conditional Variance Var(Y|X)")
     plt.xlabel("X")
     plt.ylabel("Var(Y|X)")
     plt.show()
```



1.2 ii

```
[6]: plt.plot(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100), label='E[Y|X]')
     plt.plot(np.linspace(-1, 1, 100), np.linspace(-1, 1, 100), label='Best Linear␣
       ↪Projection of E[Y|X]', color="red", linestyle="dashed")
     plt.title("CEF E[Y|X]")
     plt.xlabel("X")
     plt.ylabel("E[Y|X]")
     plt.legend()
     plt.show()
```

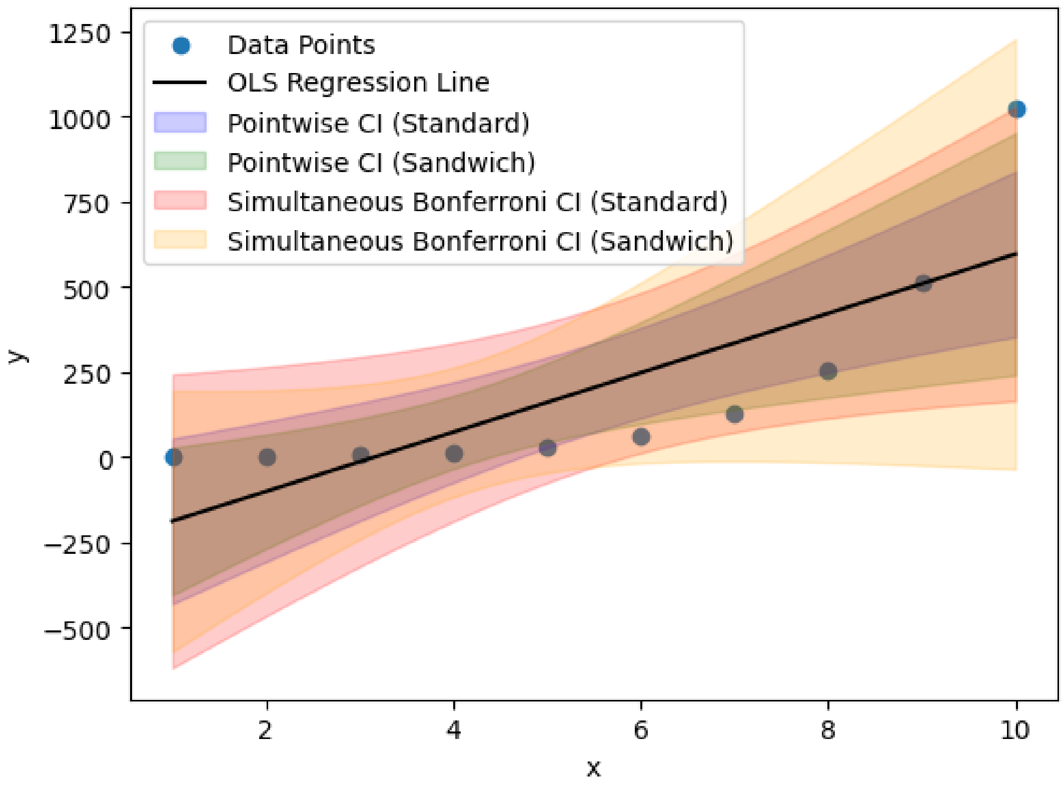


2 2

3 i

```
[ ]: import statsmodels.api as sm
     from scipy.stats import norm

     def confBand(x, y, conf):
         x = np.asarray(x)
         y = np.asarray(y)

         X = sm.add_constant(x)
         model = sm.OLS(y, X).fit()

         xg = np.linspace(min(x), max(x), 100)
         Xg = sm.add_constant(xg)
         yhat_g = model.predict(Xg)

         V_std = model.cov_params()
```

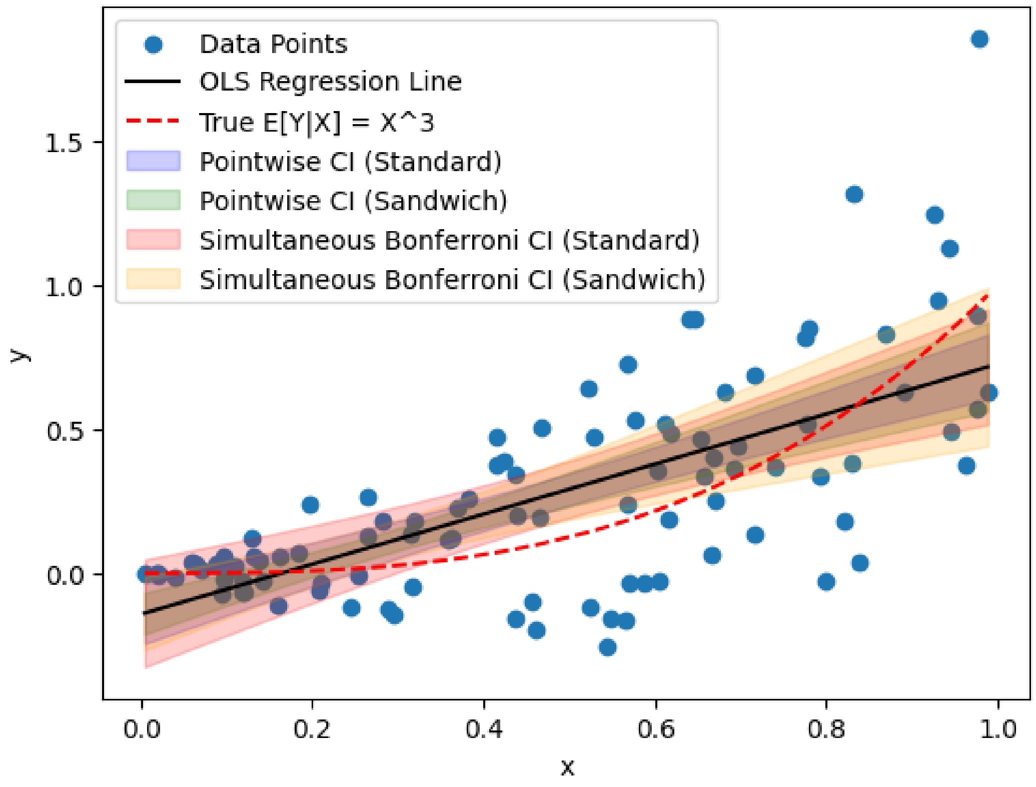
```
    V_rob = model.get_robustcov_results(cov_type='HC1').cov_params()

    se_std = np.sqrt(np.sum(Xg @ V_std * Xg, axis=1))
    se_rob = np.sqrt(np.sum(Xg @ V_rob * Xg, axis=1))

    z_point = norm.ppf(1 - (1 - conf) / 2)
    z_bonf = norm.ppf(1 - (1 - conf) / (2 * len(xg)))

    pw_std_u = yhat_g + z_point * se_std
    pw_std_l = yhat_g - z_point * se_std

    pw_rob_u = yhat_g + z_point * se_rob
    pw_rob_l = yhat_g - z_point * se_rob

    bonf_std_u = yhat_g + z_bonf * se_std
    bonf_std_l = yhat_g - z_bonf * se_std

    bonf_rob_u = yhat_g + z_bonf * se_rob
    bonf_rob_l = yhat_g - z_bonf * se_rob

    plt.scatter(x, y, label='Data Points')
    plt.plot(xg, yhat_g, color='black', label='OLS Regression Line')

    plt.fill_between(xg, pw_std_l, pw_std_u, color='blue', alpha=0.2,␣
  ↪label='Pointwise CI (Standard)')
    plt.fill_between(xg, pw_rob_l, pw_rob_u, color='green', alpha=0.2,␣
  ↪label='Pointwise CI (Sandwich)')
    plt.fill_between(xg, bonf_std_l, bonf_std_u, color='red', alpha=0.2,␣
  ↪label='Simultaneous Bonferroni CI (Standard)')
    plt.fill_between(xg, bonf_rob_l, bonf_rob_u, color='orange', alpha=0.2,␣
  ↪label='Simultaneous Bonferroni CI (Sandwich)')

    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.show()

n = 10
x = np.arange(1, n + 1)
y = 2 ** x

confBand(x, y, 0.95)
```



4 3

4.1 i

[12]:
```
np.random.seed(0)

N = 1000
n = 100
conf = 0.95

pw_std_cov = 0
pw_rob_cov = 0
bonf_std_cov = 0
bonf_rob_cov = 0

pw_std_cov_i = 0
pw_rob_cov_i = 0
bonf_std_cov_i = 0
bonf_rob_cov_i = 0

for i in range(N):
    X = np.random.uniform(0, 1, n)
    Z = np.random.normal(0, 1, n)
    Y = X**3 + 0.5 * X * Z

    grid = np.sort(X)
    true_ef = grid**3

    x_data = np.asarray(X)
    y_data = np.asarray(Y)

    X_model = sm.add_constant(x_data)
    model = sm.OLS(y_data, X_model).fit()

    Xg = sm.add_constant(grid)
    yhat_g = model.predict(Xg)

    V_std = model.cov_params()
    V_rob = model.get_robustcov_results(cov_type='HC1').cov_params()

    se_std = np.sqrt(np.sum(Xg @ V_std * Xg, axis=1))
    se_rob = np.sqrt(np.sum(Xg @ V_rob * Xg, axis=1))

    z_point = norm.ppf(1 - (1 - conf) / 2)
    z_bonf = norm.ppf(1 - (1 - conf) / (2 * len(grid)))

    pw_std_u = yhat_g + z_point * se_std
    pw_std_l = yhat_g - z_point * se_std

    if np.all((pw_std_l <= true_ef) & (true_ef <= pw_std_u)):
        pw_std_cov += 1

    pw_std_cov_i += np.mean((pw_std_l <= true_ef) & (true_ef <= pw_std_u))

    pw_rob_u = yhat_g + z_point * se_rob
    pw_rob_l = yhat_g - z_point * se_rob

    if np.all((pw_rob_l <= true_ef) & (true_ef <= pw_rob_u)):
        pw_rob_cov += 1

    pw_rob_cov_i += np.mean((pw_rob_l <= true_ef) & (true_ef <= pw_rob_u))

    bonf_std_u = yhat_g + z_bonf * se_std
    bonf_std_l = yhat_g - z_bonf * se_std

    if np.all((bonf_std_l <= true_ef) & (true_ef <= bonf_std_u)):
        bonf_std_cov += 1
    bonf_std_cov_i += np.mean((bonf_std_l <= true_ef) & (true_ef <= bonf_std_u))

    bonf_rob_u = yhat_g + z_bonf * se_rob
    bonf_rob_l = yhat_g - z_bonf * se_rob

    if np.all((bonf_rob_l <= true_ef) & (true_ef <= bonf_rob_u)):
        bonf_rob_cov += 1

    bonf_rob_cov_i += np.mean((bonf_rob_l <= true_ef) & (true_ef <= bonf_rob_u))

    if i == 0:
        plt.scatter(x_data, y_data, label='Data Points')
        plt.plot(grid, yhat_g, color='black', label='OLS Regression Line')
        plt.plot(grid, true_ef, color='red', linestyle='--', label='True E[Y|X]␣
↪= X^3')

        plt.fill_between(grid, pw_std_l, pw_std_u, color='blue', alpha=0.2,␣
↪label='Pointwise CI (Standard)')
        plt.fill_between(grid, pw_rob_l, pw_rob_u, color='green', alpha=0.2,␣
↪label='Pointwise CI (Sandwich)')
        plt.fill_between(grid, bonf_std_l, bonf_std_u, color='red', alpha=0.2,␣
↪label='Simultaneous Bonferroni CI (Standard)')
        plt.fill_between(grid, bonf_rob_l, bonf_rob_u, color='orange', alpha=0.
↪2, label='Simultaneous Bonferroni CI (Sandwich)')

        plt.xlabel('x')
        plt.ylabel('y')
        plt.legend()
        plt.show()

print("Empirical Coverage Probabilities:")
print(f"Pointwise CI (Standard): {pw_std_cov / N:.3f}")
print(f"Pointwise CI (Sandwich): {pw_rob_cov / N:.3f}")
print(f"Simultaneous Bonferroni CI (Standard): {bonf_std_cov / N:.3f}")
print(f"Simultaneous Bonferroni CI (Sandwich): {bonf_rob_cov / N:.3f}")
print()
print("Average Pointwise Coverage:")
print(f"Pointwise CI (Standard): {pw_std_cov_i / N:.3f}")
print(f"Pointwise CI (Sandwich): {pw_rob_cov_i / N:.3f}")
print(f"Simultaneous Bonferroni CI (Standard): {bonf_std_cov_i / N:.3f}")
print(f"Simultaneous Bonferroni CI (Sandwich): {bonf_rob_cov_i / N:.3f}")
```



```
Empirical Coverage Probabilities:
Pointwise CI (Standard): 0.000
Pointwise CI (Sandwich): 0.000
Simultaneous Bonferroni CI (Standard): 0.003
Simultaneous Bonferroni CI (Sandwich): 0.003

Average Pointwise Coverage:
Pointwise CI (Standard): 0.420
Pointwise CI (Sandwich): 0.381
Simultaneous Bonferroni CI (Standard): 0.760
Simultaneous Bonferroni CI (Sandwich): 0.706
```

LLM Usage: All work was done by myself in VSCode with GitHub Copilot integration. The integration "provides code suggestions, explanations, and automated implementations based on natural language prompts and existing code context," and also offers autonomous coding and an in-IDE chat interface that is able to interact with the current codebase. Only the Copilot provided automatic inline suggestions for both LaTex and Python in `.tex` and `.ipynb` Jupyter notebook files respectively were taken into account / used.