This assignment is due **August 9, 2024**. You may work in teams of up to four people. If you work in teams, make sure you include everyone's name on the assignment and there is only one submission per team. Please submit one html or pdf file. Show all your work. Use as few Python packages as possible to complete the coding portion of this assignment. Email your solutions to **charles.tutoring@gmail.com**.

1. The matrix

$$P = \begin{pmatrix} 0.84 & 0.13 & 0.03 & 0 \\ 0.10 & 0.77 & 0.09 & 0.04 \\ 0.10 & 0.20 & 0.65 & 0.05 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

denotes the one-year transition probabilities for firms with investment grade, speculative, junk, and defaulted unsecured debt. The rows represent the current states, and the columns represent the states after one year. For example, the probability that a firm with investment grade debt transitions to a firm whose debt is considered junk after one year is 3%, while the probability that a firm whose debt is considered junk transitions to a firm with speculative debt is 20%. Notice that we are supposing firms do not transition out of default.

(a) Compute P^2. Interpret your results. Are there any assumptions attached to your conclusion?

(b) Find the eigenvalues and eigenvectors of P.

(c) Use (b) to diagonalize P.

(d) Analytically compute $\lim_{n\to\infty} P^n$ using (c).

2. You have graduated from the MFE program and landed yourself a job as a quantitative trading analyst in a pod shop! Through careful analysis, you determine that the best way to model the market impact of trades is via a function proportional to $x^{3/2}$, where x represents the size of the transaction as a percent. However, your shop uses quadratic programming for portfolio construction. Hence, to integrate a market impact model into the quadratic program, the model must be a quadratic polynomial. As a result, your goal is to approximate $x^{3/2}$ using a quadratic polynomial. Your pod shop does not make any trades larger than 1% of its portfolio as a matter of policy. Therefore, if $\hat{f}$ is your approximation, you are interested

in reducing the error

$$\int_0^1 \left[x^{3/2} - \hat{f}(x)\right]^2 \, dx.$$

You want to find the orthogonal projection of $x^{3/2}$ onto the subspace of quadratic polynomials. Since you are interested in reducing the square error, use the inner product

$$\langle f, g \rangle = \int_0^1 f(x)g(x) \, dx.$$

(a) Orthogonalize the basis $(1, x, x^2)$.

(b) Use your result from (a) to project $x^{3/2}$ onto the subspace of polynomials of order at most two.

(c) To access model efficacy, also use a second-degree Taylor polynomial centered at $x = 1/2$ to approximate $x^{3/2}$.

(d) Graph $x^{3/2}$ as well as your results from (b) and (c) on one plot via `plot` in `matplotlib.pyplot`. Make sure to add a legend.

(e) Compute the norm between $x^{3/2}$ and your results from (b) and (c).

x_i	-1	0	1	2	3
y_i	3.02	2.18	4.08	-0.03	-1.08

3. Consider the function f whose xy-coordinates are in the table above. We want to approximate f via a function of the form $\alpha + \beta x$.

(a) Define

$$\mathcal{L}_2(\alpha, \beta) = \sum_{i=1}^{5} (y_i - \alpha - \beta x_i)^2 .$$

Analytically minimize $\mathcal{L}_2$ with respect to α and β to find the optimal coefficients. Remember to check whether your solution is a minimum or a maximum. Since $\mathcal{L}_2$ is convex a local minimum is a global minimum.

(b) Use `minimize` in `scipy.optimize` to check your result from (a).

(c) Your results in (a) and (b) minimized the square error, which is highly sensitive to outliers. Another

approach is to minimize the absolute error. Use `minimize` to find α and β that minimize

$$\mathcal{L}_1(\alpha, \beta) = \sum_{i=1}^{5} |y_i - \alpha - \beta x_i| .$$

Because $\mathcal{L}_1$ is hard to optimize, the optimizer may fail to converge. As long as your coefficients are reasonable, your solution is acceptable.

(d) Use `scatter` to graph the points in the table. Also, use `plot` to graph the lines corresponding to the coefficients you found in (a) and (c) on the same figure. Make sure to add a legend so that it is easy to interpret your results.

4. A convertible bond is a debt instrument that gives the owner the right, but not the obligation, to convert their debt into a predetermined number of shares of common stock. For this problem, we will suppose that the owner has the right to convert their convertible bond to one share of common stock.

Supposing the underlying stock does not pay dividends, the owner may only convert their shares to common stock at maturity, and the time until maturity is a multiple of 0.5, then–under the Black-Scholes assumptions–the value of the convertible bond is

$$P = S\Phi(d_1) - 100e^{-rT}\Phi(d_2) + 100e^{-rT} + \sum_{k=1}^{2T} \frac{c}{2} e^{-rk/2},$$

where S is the value of one share of common stock, c is the coupon rate paid semiannually, r is the continuously compounded discount rate, T is the time until maturity, σ is the volatility of returns of the underlying common stock,

$$d_1 = \frac{\ln\left(\frac{S}{100}\right) + \left(r + \frac{\sigma^2}{2}\right) T}{\sigma\sqrt{T}} \qquad \text{and} \qquad d_2 = d_1 - \sigma\sqrt{T}.$$

The function Φ is the standard normal cdf, which can be accessed in Python via `norm.cdf` within `scipy.stats`.

(a) Write a Python function that gives P as a function of c, r, S, T, and σ. Evaluate it when $c = 5\%$, $r = 8\%$, $S = 65$, $T = 5$, and $\sigma = 35\%$. Make sure that r and σ are written in decimal form when using the pricing formula. The formula above assumes c is not converted to a decimal.

(b) Use your function in part (a) to compute the numerical partial derivatives of P with respect to S and r. As before, evaluate it when $c = 5\%$, $r = 8\%$, $S = 65$, $T = 5$, and $\sigma = 35\%$.

(c) Unlike options where we normally suppose there is no credit risk, the discount rate r for a convertible bond reflects the credit of the corresponding firm. Credit risk is closely related to the stock price. When the stock price goes up, r goes down because it is easier for the firm to raise capital. In contrast, when the stock price goes down, r goes up. After careful analysis, you discover that if the stock goes from S_0 to S_1, the discount rate goes from r_0 to

$$r_1 \approx \left(\frac{S_0}{S_1}\right)^{0.15} r_0.$$

Write a Python function that returns r_1 given S_0, S_1 and r_0.

(d) Compute the numerical partial derivative of your function in (c) with respect to S_1. Suppose $S_0 = S_1 = 65$ and $r_0 = 8\%$.

(e) In part (b), you computed the partial derivative of the convertible bond with respect to S. However, as mentioned previously, when the equity changes we also expect the discount rate to change. Use your results from (b) and (d) to find the total derivative of the convertible bond price with respect to the value of the underlying equity. In this case, the total derivative is

$$\frac{dP}{dS} = \frac{\partial P}{\partial S} + \frac{\partial P}{\partial r}\frac{\partial r}{\partial S}.$$

Note: we are ignoring the effect of a change in σ due to a change in the stock price.