JOURNAL OF ECONOMICS TEACHING

Integrating Advertising into Duopoly Simulations for Intermediate Microeconomics Courses Using Python

This paper proposes incorporating advertising as a decision variable in price competition models of duopolies in undergraduate microeconomic theory courses. We use Python, which allows students to explore various scenarios with minimal coding and focus on fostering an intuitive understanding. This saves classroom time from solving complex models that would be difficult to solve algebraically. We demonstrate the approach using a baseline price competition model with advertising. We then showcase how Python helps explore variations in factors such as the price elasticity of demand, the advertising elasticity of demand, and marginal cost. We believe that this activity allows students to gain a deeper understanding of market competition with advertising, which reflects a crucial aspect of real-world competition.

Masanori Kuroki[†]

†Arkansas Tech University

1. Introduction

In an undergraduate intermediate microeconomic theory course, simultaneous duoply problems are usually two firms setting the quantity or price of their products. This paper proposes adding advertising as a decision variable in a price competition model, in which firms can influence demand through advertising, and using the model as an "advanced" oligopoly optimization problem in an intermediate microeconomic theory course. Given that advertising is one of the three choice variables in the microeconomics profit-maximization question (Chamberlin, 1993), and given the prevalence of advertising in duopolies (e.g., CocaCola vs. Pepsi, Visa vs. Mastercard), we believe that students will benefit from this exercise and recommend that this activity be used as an extension after teaching the standard textbook Cournot and Bertrand models. This exercise is also suitable for upper-level electives such as industrial organization, game theory, and managerial economics.

However, solving these models algebraically with traditional methods (i.e. paper-and-pencil and/or hand-calculator) can be demanding. In a price competition duopoly model with advertising, there will be four first-order conditions (with respect to price and advertising for each firm), and the simultaneous solution of these first-order conditions can be quite complex. This paper addresses this challenge by using Python, a free and versatile programming language. We present Python simulations that solve complex models with concise code, allowing exploration of various scenarios. This approach offers students a twofold benefit: they gain experience with realistic business situations and learn Python, a popular language in tech and data analysis. Pedagogical research finds that complementing traditional lectures using simulations to solve the numerical exercise helps students to understand the implications and insights of economic models (e.g. Kuroki, 2021; Gilbert, Koska, & Oladi, 2023). Additionally, the paper addresses the advantages of using Google Colab, a free and web-based Python environment, for beginners.

One estimate suggests that the total advertising expenditure in the United States exceeded to be $345 billion in 2023 (IBISWorld, 2024), and researchers find advertising increases sales and profitability (Eng & Keh, 2007). Given the size of spending on advertising, the interplay of advertising and inter-firm competition seems to still be an active area of research among microeconomic theorists (e.g. Hwang, Kim, & Boleslavsky, 2019; Liu, 2022). As for teaching undergraduate students, several intermediate microeconomics textbooks already use advertising as a choice variable for monopolies. For example, Pindyck and Rubinfeld's *Microeconomics* (2013) states that "[p]ricing is important for a firm, but most firms with market power have another important decision to make: how much to advertise." (p. 429), with the following end-of-chapter question (p. 438):

> Consider a firm with monopoly power that faces the demand curve
>
> $P = 100 - 3Q + 4A^{1/2}$
>
> and has the total cost function
>
> $C = 4Q^2 + 10Q + A$
>
> where A is the level of advertising expenditures, and P and Q are price and output. Find the values of A, Q, and P that maximize the firm's profit.

Similarly, see the following question from Nicholson and Snyder's (2007, p. 515) textbook Microeconomic Theory: Suppose a monopoly market has a demand function in which the quantity demanded depends not only on market price (*P*) but also on the amount of advertising the firm does (*A*, measured in dollars). The specific form of this function is

$$Q = (20 - P)(1 + 0.1A - 0{,}01A^2)$$

The monopolistic firm's cost function is given by

$$C = 10Q + 15 + A$$

a. Suppose there is no advertising $A = 0$. What output will the profit-maximizing firm choose? What market price will this yield? What will be the monopoly's profits?

b. Now let the firm also choose its optimal level of advertising expenditure. In this situation, what output level will be chosen? What price will this yield? What will the level of advertising be? What are the firm's profits in this case?

However, existing undergraduate-level textbooks seem to lack integration of advertising as a decision variable for oligopolies.[3] To address this gap, we have developed a price and advertising competition model for duopoly, where each firm chooses both price and advertising levels simultaneously. Using advertising in oligopoly models can help students understand how firms strategically use advertising to compete with each other. Advertising is a key tool for differentiating products, building brand loyalty, and capturing market share. Including advertising in oligopoly models can also help students analyze real-world advertising strategies and market behavior in sectors like airlines and cell phone carriers.

This paper makes several contributions. First, we provide a discussion of simple price competition models with advertising and combine the standard textbook oligopoly model with advertising. Second, we demonstrate that the models can be solved using Python with several lines of code despite the complexity of the first-order conditions. Finally, we allow the model parameters to change so that students can see how firms' profit-maximizing prices and advertising expenditures change.

2. Price and Advertising Competition Duopoly Models

We introduce the price and advertising competition duopoly model that serves as the foundation for our simulations. The baseline demand functions that include the impact of advertising and price on the quantity demanded for the two firms are:

$$q_1 = a - bp_1 + cp_2 + dA_1^f - eA_2^g$$
$$q_2 = a - bp_2 + cp_1 + dA_2^f - eA_1^g$$

where p is unit price, q is quantity sold, and A is advertising expenditure. b is the own-price effect and c is the cross-price effect. We assume that $a > 0$, $b > 0$, $c > 0$, and $b > c$. They imply

[1] This observation is based on our review of eight undergraduate textbooks: Banerjee (2015), Emerson (2019), Jehle and Reny (2011), Nicholson and Snyder (2007), Perloff (2017), Pindyck and Rubinfeld (2013), Varian (2014), Wetzstein (2013). Notably, Nicholson and Snyder (2007) present a Bertrand competition model with advertising (p. 533), but advertising is treated as a fixed endowment, not as a choice variable.

that good 1 is a substitute for good 2 and vice versa, and they are differentiated goods and thus imperfect substitutes, as the own-price effect is stronger than the cross-price effect. In our model, we make the following assumptions to simplify the numerical simulation process.

1. Each firm knows how its quantity demanded depends on both its price and its advertising expenditures through sufficient market research.
2. Each firm knows how its quantity demanded is affected by its competitor's price and its competitor's advertising expenditures.
3. Advertising is assumed to increase the firm's demand, and advertising by one firm has a negative effect on demand for the rival firm's product.
4. Diminishing marginal returns of advertising: As a firm increases advertising spending, the additional gain in quantity demanded will eventually become smaller and smaller.
5. Advertising and price do not depend on each other in the demand function, as shown above in the monopoly problem from Pindyck and Rubinfeld (2013). This indicates that an increase in advertising affects the intercept, not the slope, of the demand curve in the price-quantity space.[3]

Similarly, d is the own-advertising effect and e is the demand effect of its rival's advertising. The exponents f and g are assumed to be positive and less than 1 to reflect the diminishing return. We will set $d > e$, to make the numerical simulation tractable. The usual profit maximizing objective is assumed where each firm aims to maximize profits:

$$\pi_1 = p_1 q_1 - wq_1 - A_1$$
$$\pi_2 = p_2 q_2 - wq_2 - A_2$$

where w is the unit variable cost and A represents the cost of advertising. Variable costs are assumed to be linear and symmetric (i.e. a constant marginal cost of production), and fixed costs are assumed to be zero. The profit functions include the cost of advertising. To find the equilibrium, one needs to solve four first-order conditions: $\partial\pi_1/\partial p_1 = 0$, $\partial\pi_1/\partial A_1 = 0$, $\partial\pi_2/\partial p_2 = 0$, $\partial\pi_2/\partial A_2 = 0$. Solving for the profit-maximizing p_1, p_2, A_1, and A_2 will be complex and demanding to do by hand. While the algebra and calculus of the model are not impossible to do by hand, time is limited in the classroom, and some students may not have a strong math background to solve partial derivatives or systems of linear equations. Thus, we turn to simulations using Python and first find the "baseline" solution to the problem above, using a numerical example constructed to be tractable. Then we change some parameters to see how profit-maximizing price, advertising, and profits change.

[3] The monopoly problem from Nicholson and Snyder (2007), which is shown above, may be more consistent with the real world, as advertising often affects the price elasticity of demand. As Pindyck and Rubinfeld (2013) point out, increased awareness through advertising can have two main effects on price elasticity of demand. On the one hand, if advertising exposes a wider range of consumers to a product, customers might become more price-sensitive overall. This is because new customers may have more readily available substitutes and be more likely to switch to them if the price rises. On the other hand, advertising can also be used to create a distinct image, emotional connection, or brand identity for a product. This can make consumers less sensitive to price changes, as they perceive the product to offer unique value beyond just its basic function.

3. Why Python?

Python is a general-purpose programming language and is now one of the most popular programming languages. Incorporating Python into the microeconomic theory curriculum offers a valuable opportunity. Students gain exposure to powerful skills with applications across diverse disciplines.

This exposure can act as a gateway, potentially sparking a passion for coding and igniting their interest in exploring Python's capabilities for various activities beyond the classroom. Furthermore, economists are increasingly drawn to Python's versatility. It not only allows them to code their economic models but also provides many features necessary for data analysis and econometrics.

For both instructors and students new to Python, microeconomics offers a practical entry point for learning the language. Python allows users to solve most optimization problems in microeconomics with just several lines of code, with a library called *SymPy* (short for Symbolic Mathematics in Python), which is a computer algebra system capable of symbolic manipulation of mathematical formulas and equations. Ultimately, Python can equip economics majors with powerful tools for model building and provides a deeper understanding of the impact of changing variables. Finally, Python is a free and open-source language, making it a perfect fit for educational institutions with policies promoting open educational resources (OER) to reduce costs for students and departments. While established tools like Mathematica may be popular in academic mathematics, they are typically not free.

To our knowledge, there are two pedagogical papers on using Python for a microeconomic theory course at the undergraduate level: (1) Kuroki (2021) shows how to use Python to solve optimization problems, such as utility maximization and profit maximization, in an undergraduate microeconomic theory course, and (2) Luedtke (2023) shows how to develop a computer algorithm using Python to identify Nash equilibria for game theory.[3] It is well-known that computer programming skills can make students marketable after they graduate and both papers emphasize that integrating Python into undergraduate economics courses is a good way for students to learn valuable computer programming skills with little or no pre-existing programming knowledge. Of course, the major drawback of Python is that most students lack prior experience with Python, requiring instructors to introduce a new system during class time. While being introduced to Python for the first time can be overwhelming, students typically adapt to using Python for this purpose quickly as we use only four Python functions.

A. How Can I Start Using Python?

One popular option is installing Anaconda, a free and open-source distribution of Python that comes pre-loaded with tools for scientific computing. However, some instructors might hesitate to require students to install software on their personal computers. Additionally, students who use Chromebooks cannot install traditional software. A great solution to these concerns is Google Colab, which is a web-based Integrated Development Environment (IDE) for Python. An IDE offers a user-friendly interface for writing and running Python code. Since Google Colab is web-based, there is no need for installation. Students only need a Google account.

[3] Other examples of using Python to teach undergraduate economics students are Jenkins's (2022) macroeconomics and Kuroki's (2023) econometrics.

To use Google Colab, simply go to https://colab.research.google.com and click "New Notebook". Users need to be logged in with their Google account. To execute a code, select it with a click and then either press the play button to the left of the code or use the keyboard shortcut "Command/Ctrl + Enter".

B. Why Not Excel?

Excel is used extensively to teach microeconomics to undergraduates and is also widely used outside of the classroom. Price competition models with advertising can be solved using Excel with the Solver add-in (which solves optimization problems). However, since reaching Nash equilibrium requires repeated use of Solver, recording a "macro" (which can automate tasks that you want to perform repeatedly) and using Visual Basic for Applications (VBA), a programming language used in Excel, will most likely be necessary. While learning how to use Solver is straightforward, using macros can be challenging for many students. Since many students know Excel, they might wonder if it can be used for this exercise. We recommend instructors explain why Python is a better fit for this exercise.

4. Simulations Using Python

The materials provided here are intended for instructors who teach upper-level undergraduate microeconomic theory. All students are expected to have completed at least one calculus course and be familiar with the traditional duopoly model before this exercise. We recommend instructors follow these steps:

1. Solve the provided duopoly model without advertising by hand.
2. Use Python to demonstrate that this same solution can be achieved computationally.[4]
3. Introduce the duopoly model with advertising and discuss the steps required to solve it manually (e.g., taking partial derivatives with respect to price and advertising for each firm). Emphasize the complexity involved in solving by hand.
4. Finally, introduce Python as a tool to solve these complex algebraic problems.

Since most students are not familiar with Python, we recommend instructors first show the code on a projector and have students type along. This allows students to get comfortable with the code structure. Once they are more confident, instructors can ask students how they would modify the code for different scenarios by changing parameters.

First, consider the following price competition among firms that produce differentiated but substitutable products. Thus, the quantity demanded for firm 1 and firm 2 is a function of both the price the firm establishes and the price established by their rival. To simplify the algebra, the slopes and intercepts are parameterized. Suppose that the firms have the following demand curves:

[4] Instructors need to discuss and review each line of code in class. Students frequently encounter error messages, often arising from typos like case sensitivity or missing punctuation (parentheses, commas). This meticulous nature of programming languages can surprise students. It is helpful to remind them that even minor errors can cause issues. This emphasizes the importance of attention to detail, a valuable skill in programming and beyond.

$$q_1 = 400 - 4p_1 + 2p_2$$
$$q_2 = 240 - 3p_2 + 1.5p_1$$

For simplicity, assume that both firms have zero variable and fixed costs for their products so that all cost terms disappear. Their profit functions are:

$$\pi_1 = p_1 q_1 = p_1(400 - 4p_1 + 2p_2)$$
$$\pi_2 = p_2 q_2 = p_2(240 - 3p_2 + 1.5p_1)$$

Taking the first derivative with respect to prices, i.e. $\partial\pi / \partial p_1 = 0$ and $\partial\pi / \partial p_2 = 0$, the reaction functions are:

$$400 - 8p_1 + 2p_2 = 0$$
$$\rightarrow p_1 = 50 + 0.25p_2$$
$$240 - 6p_2 + 1.5p_1 = 0$$
$$\rightarrow p_2 = 40 + 0.25p_1$$

Solving for p_1 and p_2:

$$p_1 = 50 + 0.25(40 + 0.25p_1) = 60 + 0.0625p_1$$
$$\rightarrow 0.9375p_1 = 60$$
$$\rightarrow p_1 = 64$$
$$\rightarrow p_2 = 40 + 0.25(64) = 56$$

Thus, the Nash equilibrium is firm 1 charging $64 and firm 2 charging $56. Solving this price competition model algebraically should be fairly feasible for most students. Once instructors have demonstrated solving the traditional duopoly model without advertising by hand, they can introduce the Python approach.

In Python, an equal sign (=) is called an assignment operator, which assigns value to a variable. The left side of the assignment operator is a variable, and the right side of the assignment operator is a value. To do algebra and calculus, we use *SymPy*, which provides essential functions for this simulation: defining variables (*symbols()*), taking derivatives (*diff()*), solving equations (*solve()*), and substituting values (*subs()*). Figure 1 shows the code and the solution of the model as output. To use *SymPy*'s symbolic math capabilities, the line "from sympy import *" needs to be included at the start of each Python session. This line makes *SymPy*'s tools available for the code to work with symbolic expressions. Note that a pound sign (#) is used for comments that are not executed by Python. Also, note that the *print* function is used in conjunction with the *subs* function to show the quantity and profit for each firm using the profit-maximizing prices. "\n" in the print function produces a line break in the text to make the output more readable.

Figure 1. Python code for a simple price competition model

```
from sympy import * # This allows us to use all the functions of SymPy.
p1, p2= symbols('p1, p2') # Set p1 and p2 as variables.
q1 = 400 - 4*p1 + 2*p2 # Create the demand function for firm 1.
q2 = 240 - 3*p2 + 1.5*p1 # Create the demand function for firm 2.
c1, c2 = 0, 0 # Set costs for both firms zero.
profit1, profit2 = p1*q1 - c1, p2*q2 - c2 # Create profit functions for both firms.
problem = diff(profit1, p1), diff(profit2, p2) # Create first-order conditions.
solution = solve(problem) # Solve systems of equations.
print(solution, '\n',
      'q1 =', q1.subs(solution), '\n',
      'q2 =', q2.subs(solution), '\n',
      'profit1 =', profit1.subs(solution), '\n',
      'profit2 =', profit2.subs(solution))
```

```
{p1: 64.0000000000000, p2: 56.0000000000000}
 q1 =  256.000000000000
 q2 =  168.000000000000
 profit1 =  16384.0000000000
 profit2 =  9408.00000000000
```

If we add advertising as a choice variable to a price competition model, the simultaneous solution of four first-order conditions can be quite complex. Consider the following price competition model that assumes that products are symmetrically differentiated, i.e., cross-price and cross-advertising effects are not equal to own-price and own-advertising effects. This means that the effect of price and advertising on demand is the same for both firms, but the effect of the competitor's price/advertising is weaker.:

$$q_1 = 100 - 2p_1 + 1.5p_2 + 2.5A_1^{0.5} - A_2^{0.5}$$
$$q2 = 100 - 2p_2 + 1.5p^1 + 2.5A_2^{0.5} - A_1^{0.5}$$

Both firms have a marginal cost of $30 per unit and zero fixed costs.

$$\pi_1 = p_1 q_1 - 30q_1 - A_1$$
$$\pi_2 = p_2 q_2 - 30q_2 - A_2$$

If students are asked to solve for the equilibrium prices and advertising expenditures, algebra will be involved. After taking the first derivative for each variable and each firm, students will need to solve the following four equations and four unknown variables (p_1, p_2, A_1, A_2):

$$p_1 = 40 + 0.625A_1^{0.5} - 0.25A_2^{0.5} + 0.375p_2$$
$$p_2 = 40 + 0.625A_2^{0.5} - 0.25A_1^{2} + 0.375p_1$$
$$A_1 = (1.25p_1 - 37.5)^2$$
$$A_2 = (1.25p_2 - 37.5)^2$$

If Python is used for the problem above, the code will be the following (see Figure 2):

Figure 2. Python code for a price competition model with advertising

```
p1, p2, A1, A2 = symbols('p1, p2, A1, A2')
q1 = 100 - 2*p1 + 1.5*p2 + 2.5*A1**0.5 - A2**0.5
q2 = 100 - 2*p2 + 1.5*p1 + 2.5*A2**0.5 - A1**0.5
c1 = 30 * q1 + A1
c2 = 30 * q2 + A2
profit1, profit2 = p1*q1 - c1, p2*q2 - c2
problem = diff(profit1, p1), diff(profit2, p2), diff(profit1, A1), diff(profit2, A2)
solution = solve(problem)
print(solution, '\n',
      'q1 =', q1.subs(*solution), '\n',
      'q2 =', q2.subs(*solution), '\n',
      'profit1 =', profit1.subs(*solution), '\n',
      'profit2 =', profit2.subs(*solution))
```

```
[{A1: 28900.0000000000, A2: 28900.0000000000, p1: 166.000000000000, p2: 166.000000000000}]
 q1 = 272.000000000000
 q2 = 272.000000000000
 profit1 = 8092.00000000000
 profit2 = 8092.00000000000
```

Note that raising a number to a power is "**", two asterisks, in Python, not "^". Also, note one asterisk (*) in front of "solution" in the *sub* function. This is because Python returns the solution as a list (values in square brackets []), which needs to be "unpacked" with an asterisk to be used in the *sub* function. For the purpose of this exercise, it should be mentioned briefly but it is not necessary to be discussed in detail, as this is a little technical for beginner Python users.

Some instructors may prefer that students derive the first-order conditions first to obtain the four equations and then use Python to solve the system of the equations to find the equilibrium. This can be easily done also with the *solve* function as shown in Figure 3. Note that there are two possible solutions and the second solution is the same as the one in Figure 2.

Figure 3. Solving four equations and four unknown variables

```
P1, P2, A1, A2 = symbols('P1, P2, A1, A2')
eq1 = P1 - (40 + 0.625*A1**0.5 - 0.25*A2**0.5 + 0.375*P2)
eq2 = P2 - (40 + 0.625*A2**0.5 - 0.25*A1**0.5 + 0.375*P2)
eq3 = A1 - (1.25*P1 - 37.5)**2
eq4 = A2 - (1.25*P2 - 37.5)**2
solve([eq1, eq2, eq3, eq4], [P1, P2, A1, A2])
```

```
[(115.158878504673, 26.1869158878505, 11331.3040440213, 22.7181413223862),
 (166.000000000000, 166.000000000000, 28900.0000000000, 28900.0000000000)]
```

5. Further Exercises

The baseline model above imposes symmetry, and thus both firms charge the same price and spend the same amount on advertising. However, several potential economic questions can be examined within this framework. One advantage of using Python is that it adapts instantly to changes in the model parameters and thus allows students to play with simulations and see how market outcomes like price, output levels, and firm profits change. In other words, modifying parameters with Python moves beyond static analysis and equips students with a dynamic and interactive learning experience. Students can actively explore "what-if" scenarios, fostering a deeper understanding of the interplay between various factors that influence outcomes and gaining valuable economic intuition. Here we outline a few of them.

A. Changes in the Degree of Differentiation and the Price Elasticities of Demand

If the demand is not very price elastic, i.e. the price elasticity of demand is small, the firm should be able to charge a higher price than its rival. What about advertising? As pointed out by Pindyck and Rubinfeld (2013), the firm should advertise more because a smaller elasticity of demand implies a larger markup of price over marginal cost, i.e. the marginal profit from each extra unit sold is higher. Since advertising can increase the quantity demanded, it makes sense to sell more by advertising more. Using Python, we reduce the slope of p_1 in the demand function q_1, thereby making the demand for q_1 more inelastic (see Figure 4). As expected, firm 1 spends more on advertising, charges a higher price than firm 2, and earns higher profits.

Figure 4. Firm 1 has a lower price elasticity of demand

```
p1, p2, A1, A2 = symbols('p1, p2, A1, A2')
q1 = 100 - 1.9*p1 + 1.5*p2 + 2.5*A1**0.5 - A2**0.5
q2 = 100 - 2*p2 + 1.5*p1 + 2.5*A2**0.5 - A1**0.5
c1 = 30 * q1 + A1
c2 = 30 * q2 + A2
profit1, profit2 = p1*q1 - c1, p2*q2 - c2
problem = diff(profit1, p1), diff(profit2, p2), diff(profit1, A1), diff(profit2, A2)
solution = solve(problem)
print(*solution, '\n',
      'q1 =', q1.subs(*solution), '\n',
      'q2 =', q2.subs(*solution), '\n',
      'profit1 =', profit1.subs(*solution), '\n',
      'profit2 =', profit2.subs(*solution))
```

```
{A1: 54076.8880641434, A2: 35295.0702006232, p1: 216.035502958580, p2: 180.295857988166}
 q1 = 353.467455621302
 q2 = 300.591715976331
 profit1 = 11680.6078218550
 profit2 = 9882.61965617449
```

B. Changes in the Advertising Elasticity of Demand

Similarly, what happens if the effect of advertising becomes larger? Here we increase the slope of A_1 in the demand function q_1 (see Figure 5). As expected, if demand is very sens-

tive to advertising, firm 1 spends more on advertising and earns higher profits than firm 2. It also charges a higher price than its rival because higher advertising shifts the demand curve to the right. Recall that an increase in advertising affects the intercept of the demand function.

Figure 5. Firm 1 has a higher price elasticity of demand

```
p1, p2, A1, A2 = symbols('p1, p2, A1, A2')
q1 = 100 - 2*p1 + 1.5*p2 + 2.6*A1**0.5 - A2**0.5
q2 = 100 - 2*p2 + 1.5*p1 + 2.5*A2**0.5 - A1**0.5
c1 = 30 * q1 + A1
c2 = 30 * q2 + A2
profit1, profit2 = p1*q1 - c1, p2*q2 - c2
problem = diff(profit1, p1), diff(profit2, p2), diff(profit1, A1), diff(profit2, A2)
solution = solve(problem)
print(*solution, '\n',
      'q1 =', q1.subs(*solution), '\n',
      'q2 =', q2.subs(*solution), '\n',
      'profit1 =', profit1.subs(*solution), '\n',
      'profit2 =', profit2.subs(*solution))
```

```
{A1: 63711.3974851194, A2: 31294.8671699863, p1: 224.162436548223, p2: 171.522842639594}
 q1 = 388.324873096447
 q2 = 283.045685279188
 profit1 = 11686.7060475663
 profit2 = 8762.56280759620
```

C. Changes in the Marginal Cost

Marginal costs can be lowered, for example, if the government subsidizes the production or if firms invest in research and development to create new technology. In the code below, shown in Figure 6, firm 1's marginal cost is reduced. Not surprisingly, firm 1's profit and production are greater than firm 2's. Also, firm 1's price and advertising increases. To understand why firm 1's advertising increases when its marginal cost decreases, recall that a lower marginal cost implies a larger markup of price over marginal cost, making the marginal profit from each extra unit sold higher. Therefore, advertising more to sell more units makes sense. Firm 1's higher price is somewhat counterintuitive since its lower marginal cost should allow it to undercut its rival by lowering the price. To understand why firm 1's price increases when its marginal cost decreases, again we need to keep in mind that higher advertising shifts the demand curve (and its marginal revenue curve) to the right. Thus, it is conceivable that lower marginal costs lead to higher prices when combined with advertising.

Figure 6. Firm 1 has a lower marginal cost

```
p1, p2, A1, A2 = symbols('p1, p2, A1, A2')
q1 = 100 - 2*p1 + 1.5*p2 + 2*A1**0.5 - A2**0.5
q2 = 100 - 2*p2 + 1.5*p1 + 2*A2**0.5 - A1**0.5
c1 = 20 * q1 + A1
c2 = 30 * q2 + A2
profit1, profit2 = p1*q1 - c1, p2*q2 - c2
problem = diff(profit1, P1), diff(profit2, P2), diff(profit1, A1), diff(profit2, A2)
profit1, profit2 = p1*q1 - c1, p2*q2 - c2
problem = diff(profit1, p1), diff(profit2, p2), diff(profit1, A1), diff(profit2, A2)
solution = solve(problem)
print(*solution, '\n',
      'q1 =', q1.subs(*solution), '\n',
      'q2 =', q2.subs(*solution), '\n',
      'profit1 =', profit1.subs(*solution), '\n',
      'profit2 =', profit2.subs(*solution))
```

```
{A1: 4268.44444444444, A2: 2635.11111111111, p1: 85.3333333333333, p2: 81.3333333333333}
 q1 = 130.666666666667
 q2 = 102.666666666667
 profit1 = 4268.44444444444
```

6. Concluding Remarks

Traditional textbooks lack integration of advertising into oligopoly models. We bridge this gap by introducing a price and advertising competition model. Including advertising elements in price competition models for duopolies enhances the models' real-world relevance. However, the resulting models involve complex algebra, potentially consuming valuable classroom time dedicated to solving them. To address this, we propose using Python simulations. This approach fosters an intuitive grasp through simulations, circumventing the need for complex algebraic manipulations. Importantly, it exposes students to Python programming, a valuable skill in today's data-driven world. We believe this activity allows students to gain a deeper understanding of market competition with advertising, reflecting a crucial aspect of real-world competition.

References

Banerjee, S. 2015. *Intermediate microeconomics: A tool-building approach*. Routledge. Emerson, P. M. 2019. *Intermediate microeconomics*. Online textbook. https://open.umn.edu/ opentextbooks/textbooks/intermediate-microeconomics

Chamberlin, E. 1993. *The theory of monopolistic competition*. Harvard University Press.

Eng, L. L., & Keh, H. T. 2007. The effects of advertising and brand value on future operating and market performance. *Journal of Advertising*, *36*, 91–100. DOI: 10.2753/JOA0091-3367360407

Gilbert, J., Koska, O. A., & Oladi, R. 2023. Building and using nonlinear simulations in Excel with an application to the specific factors model. *Southern Economic Journal*, 1–24. DOI: 10.1002/soej.12628

Hwang, I., Kim, K, & Boleslavsky, R. 2019. *Competitive advertising and pricing*. Yale University Working Paper. https://economics.yale.edu/sites/default/files/cap_06052019.pdf

Jehle, G. A., & Reny, P. J. 2011. *Advanced microeconomic theory* (3rd ed.). Pearson.

IBISWorld. 2024. *Total advertising expenditure*. Available at https://www.ibisworld.com/us/bed/total-advertising-expenditure/4118/

Jenkins, B. C. 2022. A Python-based undergraduate course in computational macroeconomics. *The Journal of Economic Education*, *53*(2), 126–140. DOI: 10.1080/00220485.2022.2038322

Kuroki M. 2021. Using Python and Google Colab to teach undergraduate microeconomic theory. *International Review of Economics Education*, *38*, 100225. DOI: 10.1016/j.iree.2021.100225

Kuroki, M. 2023. Integrating data science into an econometrics course with a Kaggle competition. *The Journal of Economic Education*, *54*(4), 364–378. DOI: 10.1080/00220485.2023.2220695

Liu, X 2022. Competitive pricing and advertising with spillover. *Journal of Mathematical Economics*, *101*, 102660. DOI: 10.1016/j.jmateco.2022.102660

Luedtke, A. O. 2023 Teaching Nash equilibrium with Python. *The Journal of Economic Education*, *54*(2), 177-183. DOI: 10.1080/00220485.2023.2168813

Nicholson, W. J., & Snyder, C. M. 2007. *Microeconomic theory* (11th ed.). Thomson South-Western.

Perloff, J. M. 2017. *Microeconomics: Theory and applications with calculus* (4th ed.). Pearson.

Pindyck, R. S., & Rubinfeld, D. L. 2013. *Microeconomics* (8th ed.). Pearson Education.

Varian, H. R. 2014. *Intermediate microeconomics: A modern approach* (9th ed.). W. W. Norton & Company.

Wetzstein, M. E. 2013. *Microeconomic theory: Concepts and connections* (2nd ed.). Routledge.