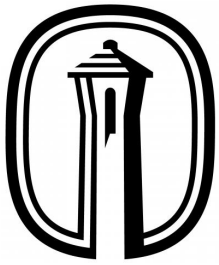
MATH 4394: Senior Project

Stability Analysis in Cournot Duopoly

TRINITY UNIVERSITY



by

Student: Lutfi Sun

Advisor: Dr. Saber Elaydi

Category: Interdisciplinary Project

May 10, 2020

Contents

Abstract

In this project, I investigate the impact of bounded rationality and asymmetrical production costs on a dynamic Cournot Duopoly Game. I find that there exists a stable Nash Equilibrium even when the firms do not have complete information about the demand curve. I show under what conditions the game has multiple equilibrium points and when the system becomes chaotic. I find that cournot duopoly with similar firms is more stable and reaches the equilibrium quantity faster.

1 Introduction

Cournot Duopoly is a competition game with two firms whose strategies are the quantities they produce of identical products. This is the benchmark for cournot competition; there are versions with multiple stages, more than two firms, and differentiable products. In this project, I work on finding nash equilibria in cournot duopoly games with multiple stages and bounded rationality.

Nash equilibrium in this context means an outcome where neither of the firms benefit from changing the quantity they produce. I look at if and how the initial conditions and production functions impact the equilibrium (ie fixed point) firms end up in and what that means in applied economics. I analyze the existence, uniqueness, and stability of nash equilibria.

I will be referring to and using game theory, difference equations, linear algebra, optimization, and real analysis. In my future studies, I am planning to introduce adaptive expectations to this model and look for relevant market data.

2 Model

2.1 Classic Cournot

In the classic Cournot game, we have two profit maximizing firms that compete over quantity of an identical product they supply in the market. Let $q_1(t)$ and $q_2(t)$ be the output (quantity supplied) of firm 1, and firm 2, respectively. Let p be the price of a unit given by demand function $p = a - b(q_1 + q_2)$ where a is the y intercept and b is the slope in $p - q$ plane. Let $c_i(q_i) = c_i q_i$ be the cost of producing q_i units of output for firms $i \in \{1, 2\}$ (ie assume constant marginal cost). Then, $pq_i = (a - b(q_i + q_j))q_i$ is the total revenue and $c_i q_i$ is the total cost of a firm supplying q_i units when its opponent supplies q_j.

- Players: $N = \{1, 2\}$

- Strategy sets: $Q_i = [0, \infty]$ for $i \in \{1, 2\}$ and firms choose quantities $q_i \in Q_i$

- Payoffs (Π for profits): For $i, j \in \{1, 2\}, \quad i \neq j$

$$\Pi_i(q_i, q_j) = \begin{cases} (a - b(q_i + q_j))\, q_i - c_i q_i & \text{if } q_i + q_j < a/b \\ -c_i q_i & \text{if } q_i + q_j \geq a/b \end{cases}$$

The maximization problem firm i faces when it believes that its opponent will produce quantity q_j^e is

$$\max \Pi_i\left(q_i, q_j^e\right) = \left(a - b(q_i + q_j^e)\right) q_i - c_i q_i$$

Recall that we can find local maximums using the first-order condition,

$$\frac{\partial \Pi_i}{\partial q_i} = a - 2bq_i - bq_j^e - c_i = 0 \tag{1}$$

.

Since the second derivative of Π_i is below zero $-2b < 0$, the first order condition gives a local maximum. We can then use equation (1) to derive the best-response function for each firm. Isolating q_i, we have:

$$BR_i\left(q_j^e\right) = \frac{a - bq_j^e - c_i}{2b}$$

This means that given the expected quantity of the opponent firm, each firm chooses quantities as follows:

$$q_1 = \frac{a - bq_2^e - c_1}{2b} \qquad q_2 = \frac{a - bq_1^e - c_2}{2b}$$

To turn Cournot Duopoly into a dynamic game, we can define the expected quantity as a function of quantity produced in previous time periods. The simplest way to do this is to assume naive expectations that is expecting what happened in the previous period to happen again. In this case, firm i will expect firm j to produce exactly what firm j has produced before, $q_j^e(t+1) = q_j(t)$. Then, we have:

$$\begin{aligned} q_i(t+1) &= BR_i(q_j(t)) \\ &= \frac{a - bq_j(t-1) - c_i}{2b} \end{aligned}$$

Now we have a system of difference equations. And, we can find its fixed points. A stable fixed point of the system corresponds to a Nash Equilibrium where firms do not have a tendency to move. Similarly the two isoclines of the system correspond to the best response functions.

To find a fixed point, we need t such that $q_i(t+1) = q_i(t)$.

$$q_1(t+1) = BR_1(q_2(t)) = BR_1(BR_2(q_1(t-1)).$$

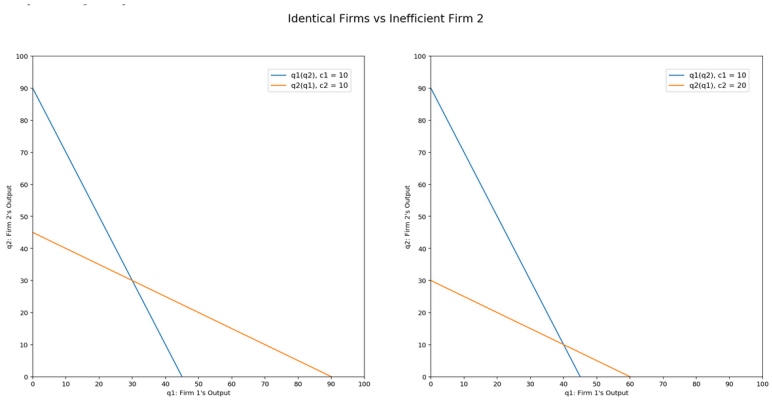


Figure 1: Cournot Isoclines (Best Response Functions)

Let q_1^* be the equilibrium quantity.

$$q_1^* = \frac{a - b(\frac{a - bq_1^* - c_1}{2b}) - c_1}{2b}$$
$$q_1^* = \frac{a + bq_1^* - c_1}{4b}$$
$$4bq_1^* = a + bq_1^* - c_1$$
$$q_1^* = \frac{a - c_1}{2b}.$$

In case of identical firms ($c_1 = c_2 = c$), the nash equilibrium quantity in Cournot Duopoly is:

$$q_1^* = q_2^* = \frac{a - c}{3b}.$$

See in figure 1 that the isoclines meet at the fixed point ie the best response functions meet at the Nash Equilibrium.

2.2 Modifying Cournot

Firms in real life do not act perfectly rational. They either lack complete information or computational ability to maximize profit. This is called bounded rationality. One way we can introduce bounded rationality is to limit firm's knowledge of the demand function. This is realistic since firms can only estimate demand by looking at how customers have reacted to changes in price and quantity in the past. When a firm tries to infer how the market will respond to its production changes by an empirical

estimate of the marginal profit, it is called local profit maximizer. We can define marginal profit as:

$$\Phi_i(t) = \left(\frac{\partial \Pi_i}{\partial q_i}\right)^{(e)} = \frac{\partial \Pi_i}{\partial q_i}\left(q_1, q_2\right), \quad i = 1, 2$$

These myopic firms increase output when they expect a positive marginal profit and decrease output when the perceived $\Phi_i(t)$ is negative (Bischi 2000). Taking derivative of the profit function:

$$\begin{aligned} \Pi_i &= (a - bq_i(t) - bq_j(t))q_i(t) - c_i q_i(t) \\ \implies \frac{\partial \Pi_i}{\partial q_i} &= a - c_i - 2bq_i(t) - bq_j(t) \end{aligned}$$

Let α be the speed of adjustment where the firm will adjust its output $\alpha q_i \frac{\partial \Pi_i}{\partial q_i}\left(q_1, q_2\right)$ as much. Our dynamical system then becomes:

$$q_1(t+1) = q_1(t) + \alpha q_1(t)\left[a - c_1 - 2bq_1(t) - bq_2(t)\right]$$

$$q_2(t+1) = q_2(t) + \alpha q_2(t)\left[a - c_2 - 2bq_2(t) - bq_1(t)\right]$$

Letting $q_i(t+1) = q_i(t) = q_i^*$, we have:

$$\begin{aligned} 0 &= \alpha q_1^*\left[a - c_1 - 2bq_1^* - bq_2^*\right] \\ 0 &= \alpha q_2^*\left[a - c_2 - 2bq_2^* - bq_1^*\right] \end{aligned}$$

For identical firms where $c = c_1 = c_2$, we find four fixed points:

$$E_0 = (0, 0)$$

$$E_1 = \left(\tfrac{a-c}{2b}, 0\right)$$

$$E_2 = \left(0, \tfrac{a-c}{2b}\right)$$

$$E_3 = \left(\tfrac{a-c}{3b}, \tfrac{a-c}{3b}\right)$$

3 Stability Analysis

3.1 Tools for Systems of Nonlinear Difference Equations

To determine the stability of a fixed point, I will use the Hartman-Grobman Theorem, which states that within a certain neighborhood around a fixed point, the dynamical system behaves linear. To locally linearize the dynamical system, we can represent it as an orbit in (Q_1, Q_2) space and approximate its trajectory using the Jacobian matrix. Let $F = (f_1, f_2) : \mathbb{R}^2_+ \to \mathbb{R}^2_+$ where

$$q_1(t+1) = f_1\left(q_1(t), q_2(t)\right)$$
$$q_2(t+1) = f_2\left(q_1(t), q_2(t)\right)$$

The matrix below is called Jacobian, and it is a linear approximation of our update rule.

$$JF\begin{pmatrix} q_1 \\ \\ q_2 \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1}{\partial \varepsilon_1} & \frac{\partial f_1}{\partial q_2} \\ \\ \frac{\partial f_2}{\partial q_1} & \frac{\partial f_2}{\partial q_2} \end{pmatrix} = \begin{pmatrix} J_{11} & J_{12} \\ \\ J_{21} & J_{22} \end{pmatrix} \tag{2}$$

Taking partial derivatives of f_1 and f_2 we have:

$$J_{11} = 1 + \alpha\left(a - c - 4bq_1 - bq_2\right)$$
$$J_{12} = -\alpha b q_1$$
$$J_{21} = -\alpha b q_2$$
$$J_{22} = 1 + \alpha\left(a - c - 4bq_2 - bq_1\right)$$

Given the quantities supplied at time t $\boldsymbol{q}(t) = (q_1(t), q_2(t))$, the linear approximation of the future quantities is $\boldsymbol{u}(t+1) = J(\boldsymbol{q}(t))$. Once we linearize the system around a fixed point, we can use the properties of linear systems to analyze stability.

Theorem 1. *Let $f : G \subset \mathbb{R}^2 \to \mathbb{R}^2$ be a C^1 map, where G is an open subset of $\mathbb{R}^2$, X^* is a fixed point of f, and $A = Jf\left(X^*\right)$. Let ρ(A) be the spectral radius of A (largest (absolute) eigen value). Then the following statements hold true:*

1. *If $\rho(A) < 1$, then X^* is asymptotically stable.*

2. *If $\rho(A) > 1$, then X^* is unstable.*

3. *If $\rho(A) = 1$, then X^* may or may not be stable.*

This theorem is from book Discrete Chaos by Saber Elaydi (Theorem 4.11).

.

One way we can make this analysis easier is to use the trace-determinant method instead of calculating eigen values. The characteristic equation of the Jacobian matrix

$$\lambda^2 - (J_{11} + J_{22})\,\lambda + (J_{11}J_{22} - J_{12}J_{21}) = 0$$

or

$$\lambda^2 - \mathrm{tr}(J)\lambda + \det(J) = 0$$

Applying the stability condition in discrete dynamics $|\lambda| < 1$ (Theorem 1), we have:

$$|\lambda| = \left| \frac{tr(J) \pm \sqrt{tr(J)^2 - 4det(J)}}{2} \right| < 1$$

Which is equivalent to $|\,\mathrm{tr}\,A| < 1 + \det A < 2$.

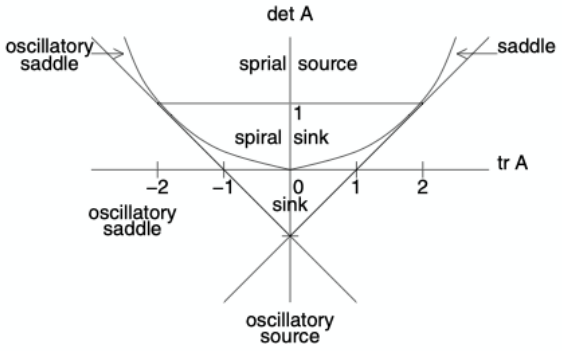


Figure 2: Stability on The Trace Determinant Plane (Elaydi, 2007)

3.2 Stability Results and Economics Interpretation

3.2.1 Extinction Point: $(0, 0)$

For the neighborhood around the extinction point, we have the Jacobian:

$$J(0,0) = \begin{pmatrix} 1+\alpha(a-c) & 0 \\ 0 & 1+\alpha(a-c) \end{pmatrix}$$

Since we have the matrix in upper triangular form, the eigenvalues are the entries in the diagonal: $\lambda = 1 + \alpha(a - c)$. Then, the fixed point $(0, 0)$ is unstable when $\alpha(a-c) > 0$. Positive α means that the firms try to adjust their productions according to increase their profits. The parameter a is the intercept of the demand curve. We can interpret it as the maximum willingness to pay in the market. So $(a - c) > 0$ means that there are buyers in the market who are willing to pay a price per unit that is above the production cost c. Therefore, our condition suggests that there will be a market for a given product as long as there are firms that respond to market incentives and buyers who are willing to pay above production costs.

3.2.2 Exclusion (Monopoly) Point: $(\frac{a-c}{2b}, 0)$

The point $(\frac{a-c}{2b}, 0)$ is where firm 1 dominates the market and firm 2 runs out of business. The Jacobian is

$$J\left(\tfrac{a-c}{2b}, 0\right) = \begin{pmatrix} 1-\alpha(a-c) & -\frac{\alpha(a-c)}{2} \\ 0 & 1+\frac{\alpha(a-c)}{2} \end{pmatrix}$$

The two eigen values are

$$\begin{aligned} \lambda_1 &= 1 + \frac{\alpha(a-c)}{2} \\ \lambda_2 &= 1 - \alpha(a-c) \end{aligned} \tag{3}$$

For $\alpha(a - c) > 2$, both eigenvalues would exceed 1, and the point would be unstable. For $0 < \alpha(a - c) < 2$, λ_1 would be above 1 and λ_2 would be below which implies a saddle point. Then, the extinction point is unstable for $\alpha(a - c) > 2$ and a saddle point when $0 < \alpha(a - c) < 2$.

.

There are a few lessons we can draw here. For a demand curve slightly above cost, $0 < \alpha(a-c) < 2$, a monopoly can temporarily exist but the market will eventually draw other firms in. Looking at the eigenvectors, we can see that v_1 is in horizontal and v_2 is in vertical direction in the $q_1 - q_2$ plane. Then, the monopoly saddle point attracts from the extinction point and repels towards coexistence (duopoly) points. For larger values of demand, $\alpha(a-c) > 2$, a monopoly is unstable. Here, I only analyze firm 1's dominance. The findings can apply to firm 2 for the fixed point $(0, \frac{a-c}{2b})$.

3.2.3 Coexistence (Cournot Duopoly) Point: $\left(\frac{a-c}{3b}, \frac{a-c}{3b}\right)$

The fixed point $\left(\frac{a-c}{3b}, \frac{a-c}{3b}\right)$ is where both firms supply a share of the quantity demanded, thus a duopoly. The jacobian is

$$J\left(\tfrac{a-c}{3b}, \tfrac{a-c}{3b}\right) = \begin{pmatrix} 1 - \frac{2}{3}\alpha(a-c) & -\frac{1}{3}\alpha(a-c) \\ -\frac{1}{3}\alpha(a-c) & 1 - \frac{2}{3}\alpha(a-c) \end{pmatrix}$$

with eigenvalues

$$\lambda_1 = 1 - \alpha(a-c), \quad \lambda_2 = 1 - \frac{\alpha(a-c)}{3}.$$

Therefore, the local stability condition for E_3 is $0 < \alpha(a-c) < 2$. We see that for reasonable levels of demand above the production cost, the Cournot duopoly fixed point is asymptotically stable. In other words, it is a Nash equilibrium as firms do not have a tendency to change their production levels once they are on this point.

4 Numerical Analysis

Now, we can test and visualize the dynamics we have found. To do this, I use python. First, I look at the dynamics of the game by iterating from an initial point with identical firms. The initial point I pick is $(0.01, 03)$ where firm 2 dominates the market. I want keep $0 < \alpha(a - c) < 2$ where there is a stable Nash equilibrium according to our conditions.

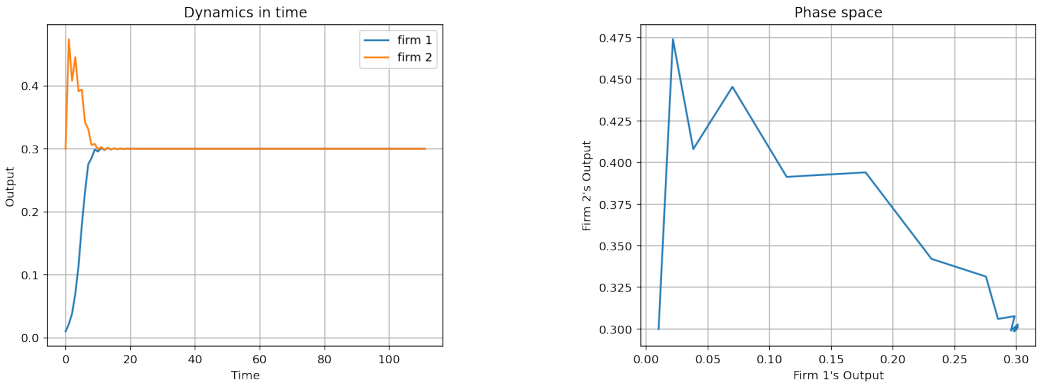


Figure 3: 111 iterations with $\alpha = 2, a = 1.9, b = 1$, and $c = 1$

In Figure 3, we see that the system reaches equilibrium point in less than 20 iterations. The quiver plot in figure 4 shows that the extinction point is a source and both monopoly points are saddle. First drawn towards a monopoly, the market settles in on the Cournot Nash Equilibrium.

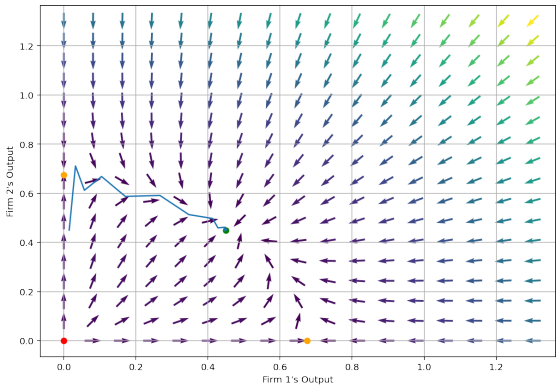


Figure 4: Quiver plot where $\alpha = 2, a = 1.9, b = 1$, and $c = 1$

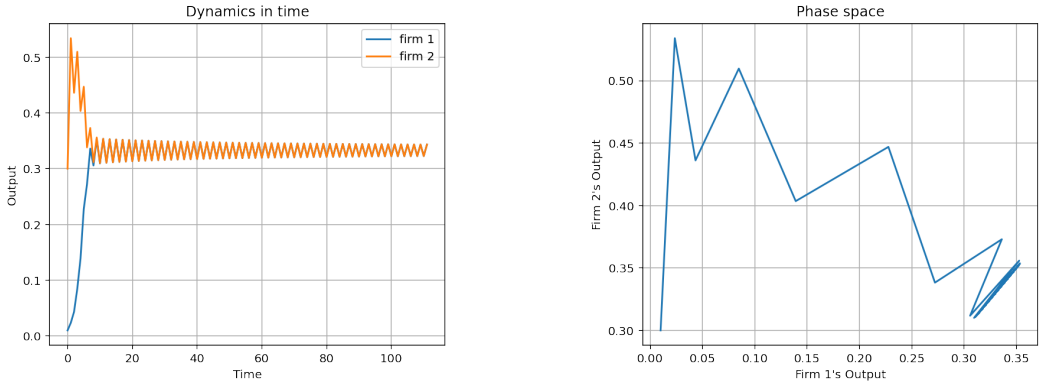


Figure 5: 111 iterations with $\alpha = 2, a = 2, b = 1$, and $c = 1$

.

Secondly, I want to see what would happen in the border where $\alpha(a - c) = 2$. I let $\alpha = 2, a = 2, b = 1$, and $c = 1$. We see in Figure 5 that the system reaches near an equilibrium quantity and keeps oscillating around it.

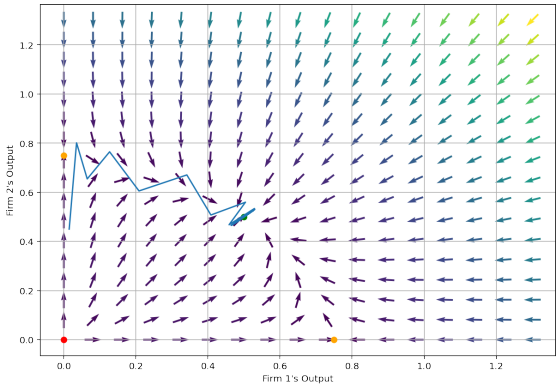


Figure 6: Quiver plot where q_1 with $\alpha = 2, b = 1$, and $c = 1$

I also analyze how changes in demand would impact the market. I let $\alpha = 2, b = 1$, $c = 1$ and play with the demand curve by changing values of a. In figure 7, I iterated the dynamical system for a hundred times with four different demand curves. As I increased demand, the system got more unstable. Remember that most of our conditions for stability depend on $\alpha(a - c)$.

Impact of Demand on Output Over Time

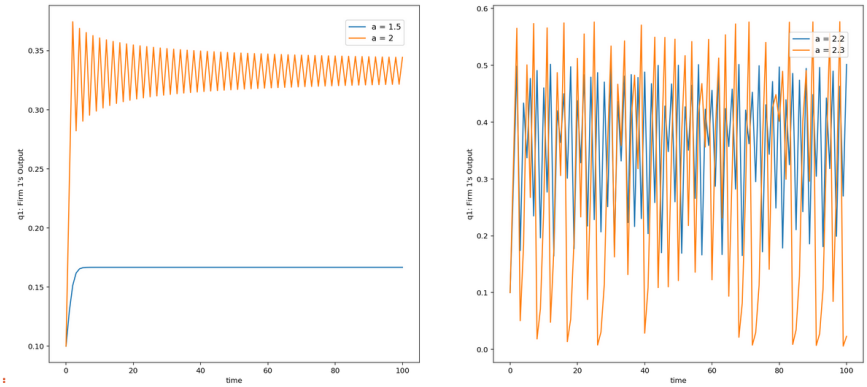


Figure 7: 100 iterations of q_1 with $\alpha = 2, b = 1$, and $c = 1$

The graph on the right shows that after $\alpha(a - c)$ passes 2, the system no longer has a single point it approaches. Furthermore, we see that when the demand intercept is large, the market becomes more volatile. Perhaps, this insight can be related to the heterogeneity of customers as the demand curve covers more combinations of willingness to pay for larger values of a.

Impact of Demand on Equilibrium Output (Identical Firms)

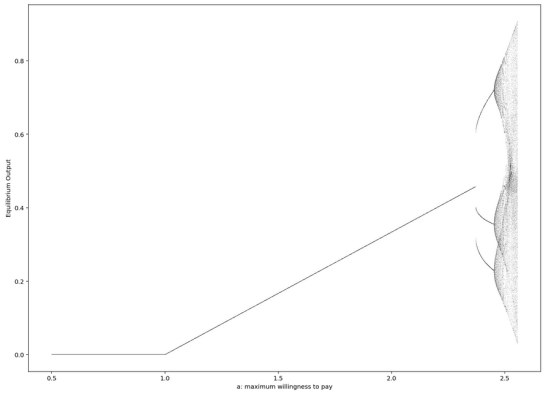


Figure 8: Bifurcation diagram for q_1 with $\alpha = 2, b = 1$, and $c = 1$

In agreement with our conditions from the stability analysis, figure 3 shows that the extinction point becomes unstable after a exceeds c, and a market for the product emerges. The equilibrium quantity trifurcates at around $a = 2.37$, and the system becomes chaotic for values of a larger than 2.45.

Tripling of Equilibrium Output in a Market with Identical Firms

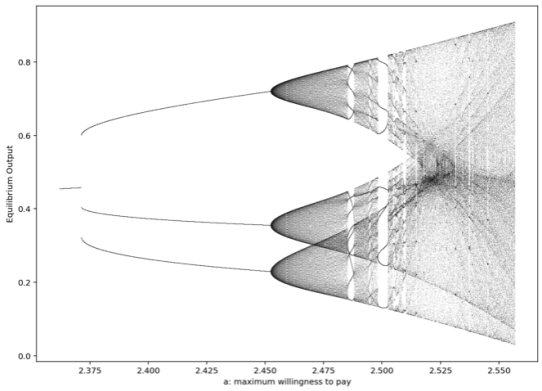


Figure 9: Bifurcation diagram (zoomed in) for q_1 with $\alpha = 2, b = 1$, and $c = 1$

When we zoom in the diagram, we can see more clearly the sudden change from one nash equlibrium to three nash equilibria. This further explains the behavior we see in Figure 7.

Cournot Duopoly with Asymmetrical Firms

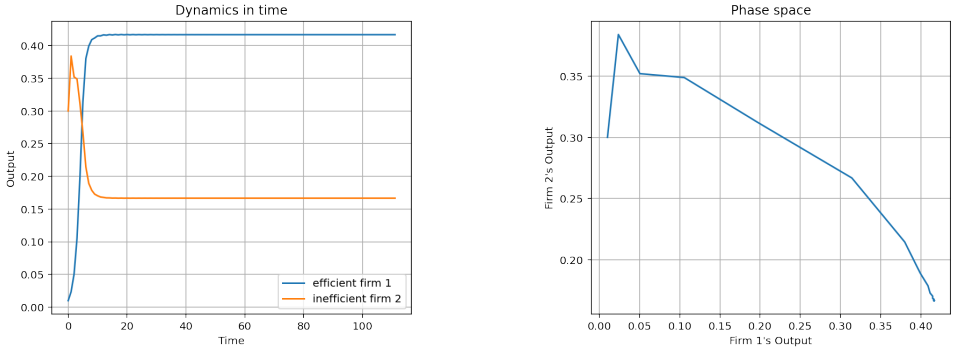


Figure 10: 111 iterations with $\alpha = 2, a = 2, b = 1, c_1 = 1$, and $c_2 = 1.25$

Another scenario we can study is when one firm is more efficient than the other (lower cost of production). Figure 10 shows that the more efficient firm eventually takes over most of the market. In Figure 11, we see the same phenomenon in a quiverplot. The market forces push the firms to a point where efficient firm dominates the market.

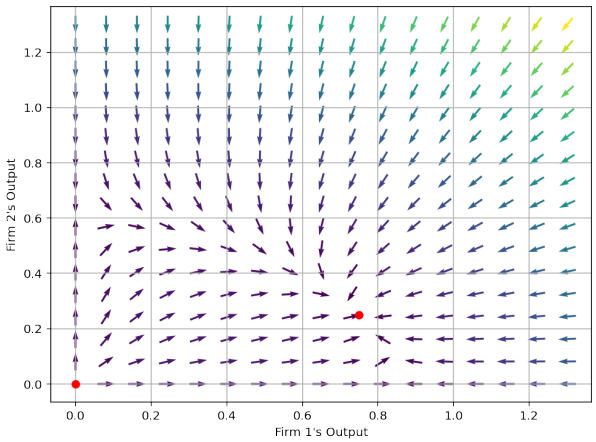


Figure 11: Quiverplot $\alpha = 2, a = 2, b = 1, c_1 = 1$, and $c_2 = 1.25$

If we widen the gap between cost functions and reduce demand, the nash equilibrium merges with the monopoly fixed point (Figure 12). This shows that in our dynamics, monopoly can be a permanent point if one firm has a substantial advantage in production and able to meet all the quantity demanded.

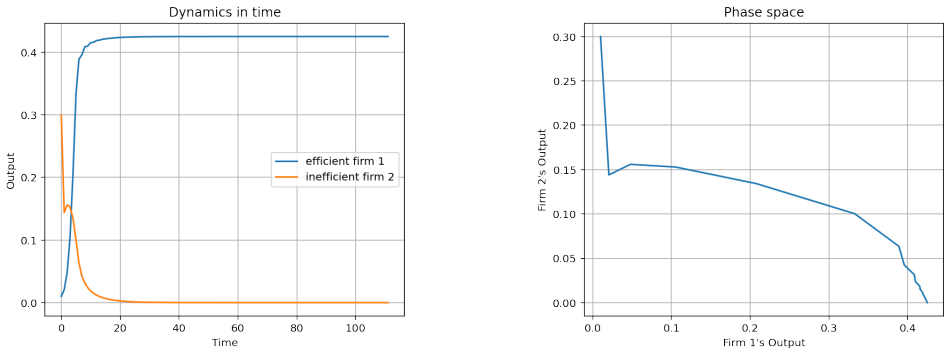


Figure 12: Monopoly $\alpha = 2, a = 1.6, b = 1, c_1 = 0.75$, and $c_2 = 1.25$

As we increase the demand in a market with asymmetrical firms, the system starts to destabilize. As seen in Figure 13, the equilibrium point bifurcates around $a = 1.5$ and system becomes chaotic much earlier than it does for a market with identical firms, $a = 2.37$. If this insight applies to real life, we can say that there is more room for volatility in markets with dissimilar firms.

Impact of Demand on Equilibrium Output (Asymmetrical Firms)

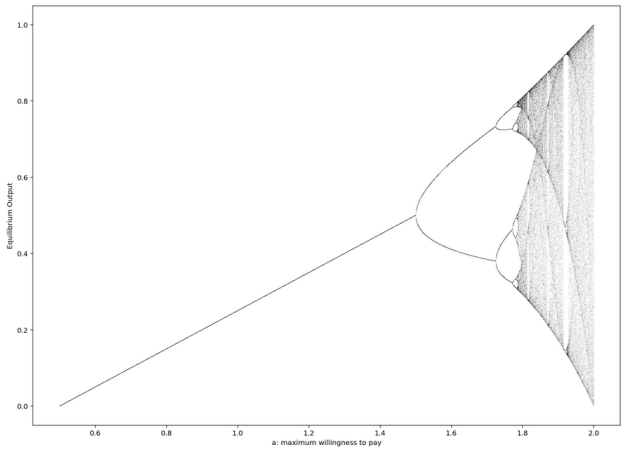


Figure 13: Bifurcation diagram for q_1 with $\alpha = 2, b = 1$, $c_1 = 0.5$, and $c_2 = 1.5$

I increased the asymmetry and demand to show this doubling in Figure 13. We see that each firm has its own cycle. The efficient firm (firm 1) eventually dominates the market, but its production is more volatile than the inefficient firm. I think this is because $\alpha(a - c_1)$ exceeds the critical value 2 more than $\alpha(a - c_2)$ does (since c_2 is higher).

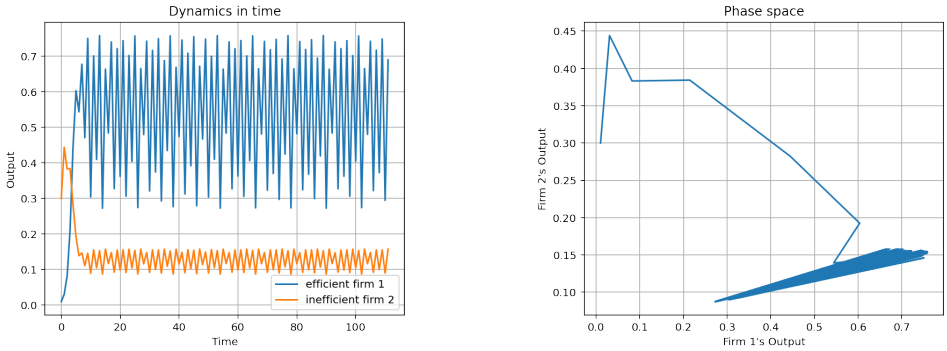


Figure 14: $\alpha = 2, a = 2.1, b = 1, c_1 = 0.75$, and $c_2 = 1.25$

To go into the more chaotic region of the bifurcation diagram, I increase the demand to 3.1 and introduce another kind of asymmetry between firms by varying their adjustment speeds. Figure 15 shows that these dynamics make it possible for the inefficient firm to perform better than the efficient firm about one fourth of the times.

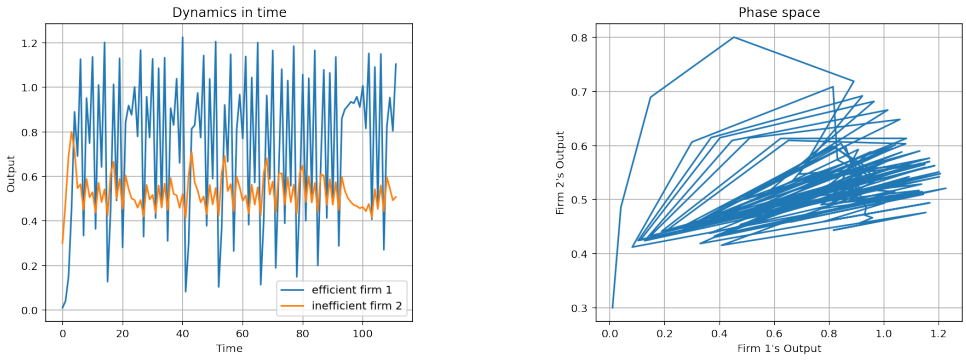


Figure 15: $\alpha_1 = 1.5, \alpha_2 = 0.5, a = 3.1, b = 1, c_1 = 0.75$, and $c_2 = 1.25$

I also analyzed the impact of changes in the firms' speed of adjustment to the market incentives, α, on equilibrium output. Each graph in Figure 16 shows a market with two firms that have different adjustment speeds.

Impact of Adjustment Speed on Output Over Time

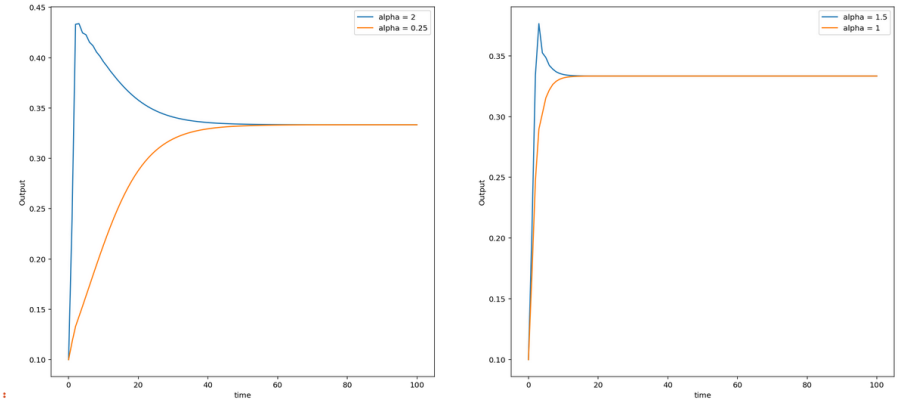


Figure 16: Liberal vs Conservative Companies

I call the firm with higher α liberal and lower α conservative. The graph on the left has two firms with drastically different adjustment speeds, $\alpha_1 = 2, \alpha_2 = 0.25$. We see that when firms have similar adjustment speeds, the market reaches equilibrium faster.

Impact of Adjustment Speed on Equilibrium Output

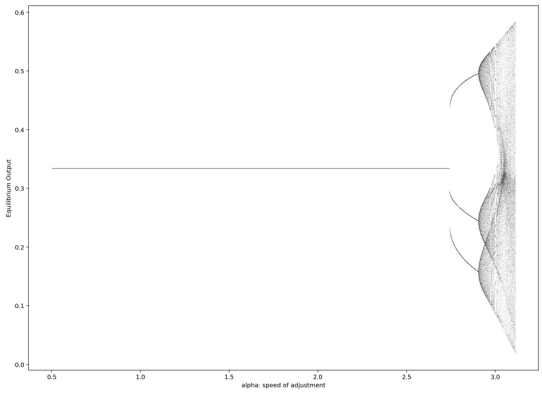


Figure 17: Bifurcation diagram for q_1 with $a = 2, b = 1$, $c = 1$

Lastly, I keep everything except cost constant. We see a diagram that looks like the symmetry of Figure 8 where we changed demand keeping cost constant. This makes sense as our key value $\alpha(a - c)$ includes the demand intercept and cost with opposite signs. Furthermore, we find a and c together as $a - c$ in all our Jacobian matrices and stability conditions.

Impact of Production Cost on Equilibrium Output (Identical Firms)

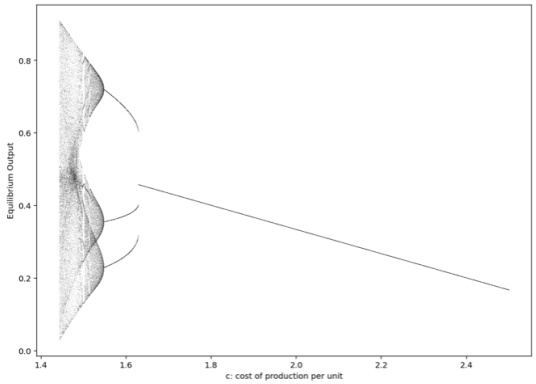


Figure 18: Bifurcation diagram for q_1 with respect to c where $\alpha = 2, a = 3, b = 1$

5 Conclusions and Next Steps

5.1 Some Takeaways

- $(0,0)$. When $\alpha(a-c) > 0$, the extinction fixed point is unstable $\implies$ as long as there is someone willing to pay above the cost of a product, there will arrive firm to provide the service.
- $(\frac{a-c}{2b}, 0)$. When $0 < \alpha(a-c) < 2$, the monopoly fixed point is a saddle point $\implies$ when the demand is close to the cost of production and one firm is more efficient, there is room for a monopoly.
- A cournot with similar firms (true both for production cost and adjustment speed) is more stable and reaches the equilibrium quantity faster.

5.2 Next Steps

- Adaptive Expectations:

$$q_1(t+1) = q_1(t) + \alpha q_1(t) \frac{\partial \Pi_1}{\partial q_1}\left(q_1(t), q_2(t)\right)$$

- Positive Externalities:

$$r_1\left(q_2\right) = \mu_1 q_2\left(1 - q_2\right)$$
$$r_2\left(q_1\right) = \mu_2 q_1\left(1 - q_1\right)$$

- Understand and find critical curves, chaotic attractors, and basins of attractions.

6 References

1. Bischi, Gian Italo and Michael Kopel. (2001). "Equilibrium selection in a nonlinear duopoly game with adaptive expectations ". The Journal of Economic Behavior Organization. Vol. 46 (2001) 73–100.

2. Bischi G.I., Naimzada A. (2000). "Global Analysis of a Dynamic Duopoly Game with Bounded Rationality". In: Filar J.A., Gaitsgory V., Mizukami K. (eds) Advances in Dynamic Games and Applications. Annals of the International Society of Dynamic Games, vol 5. Birkhäuser, Boston, MA

3. Elaydi, N Saber. (2007). "Discrete Chaos, Second Edition". Chapman & Hall/CRC.

4. Ueda, Masahiko. (2019). "Effect of Information Asymmetry in Cournot Duopoly Game with Bounded Rationality." Applied Mathematics and Computation 362: 124535. Crossref. Web.

A Python Code

Dynamics in Time

May 10, 2020

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

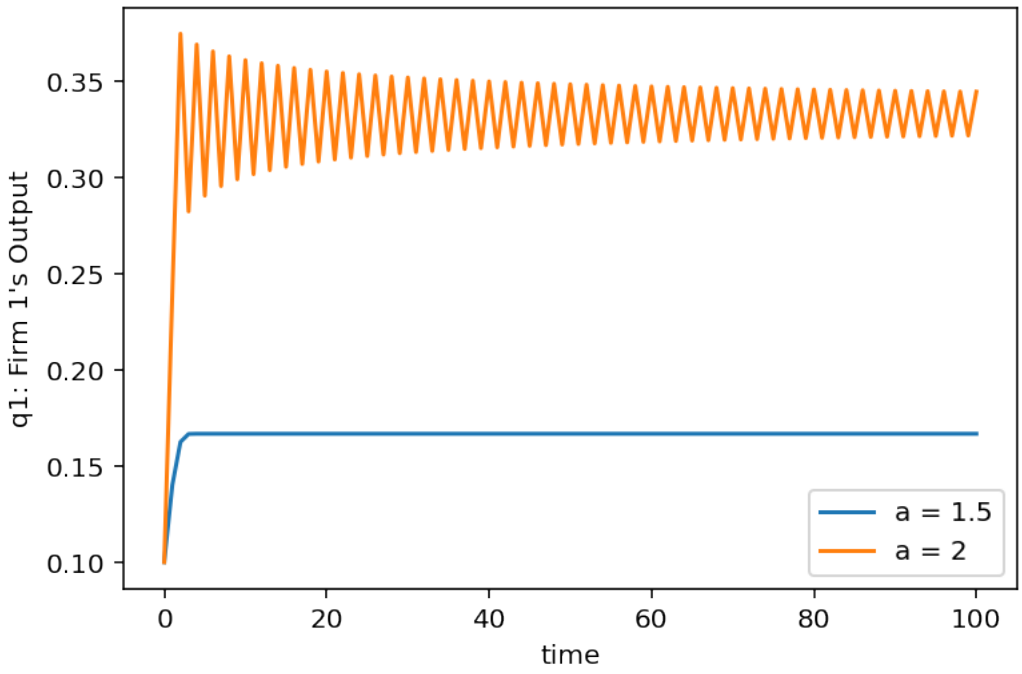
[2]:
```
def myopic(x,y, alpha1,alpha2, a,b,c, n):
    xs=[x]
    ys=[y]
    for i in range(n):
        xs.append(xs[i] + alpha1*xs[i]*(a-c-2*b*xs[i]-b*ys[i]))
        ys.append(ys[i] + alpha2*ys[i]*(a-c-2*b*ys[i]-b*xs[i]))
    return np.array([xs, ys])
```

[3]:
```
x = np.linspace(0, 100, 101)
a15 = myopic(0.1,0.1,     2,2,    1.5,1,1,    100)
a20 = myopic(0.1,0.1,     2,2,    2.0,1,1,    100)
a22 = myopic(0.1,0.1,     2,2,    2.2,1,1,    100)
a23 = myopic(0.1,0.1,     2,2,    2.3,1,1,    100)
a25 = myopic(0.1,0.1,     2,2,    2.5,1,1,    100)
a30 = myopic(0.1,0.1,     2,2,    3.0,1,1,    100)
```

[4]:
```
plt.plot(x, a15[0], label ="a = 1.5")
plt.plot(x, a20[0], label ="a = 2")
plt.xlabel('time')
plt.ylabel('q1: Firm 1\'s Output')
plt.legend()
```

[4]:
```
<matplotlib.legend.Legend at 0x7f072c6d7b70>
```

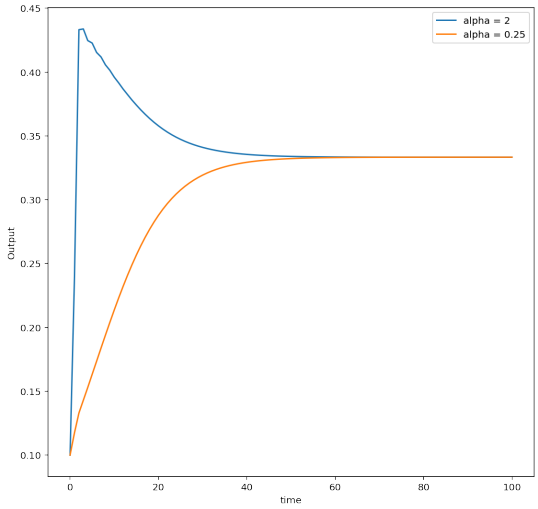
[4]:

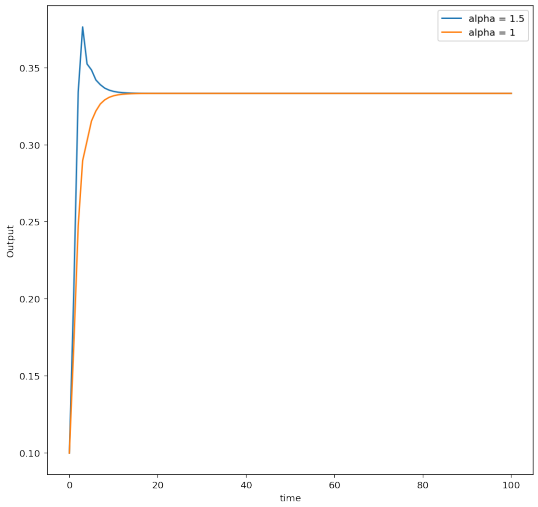


[5]:
```
alp25 = myopic(0.1,0.1,    2.0,0.25,    2,1,1,   100)
alp15 = myopic(0.1,0.1,    1.5,1.0,     2,1,1,   100)

plt.figure(figsize=(20, 9))

plt.subplot(121)
plt.plot(x, alp25[0], label ="alpha = 2")
plt.plot(x, alp25[1], label ="alpha = 0.25")
plt.xlabel('time')
plt.ylabel('Output')
plt.legend()

plt.subplot(122)
plt.plot(x, alp15[0], label ="alpha = 1.5")
plt.plot(x, alp15[1], label ="alpha = 1")
plt.xlabel('time')
plt.ylabel('Output')
plt.legend()
```

[5]: <matplotlib.legend.Legend at 0x7f072c1d1208>

[5]:

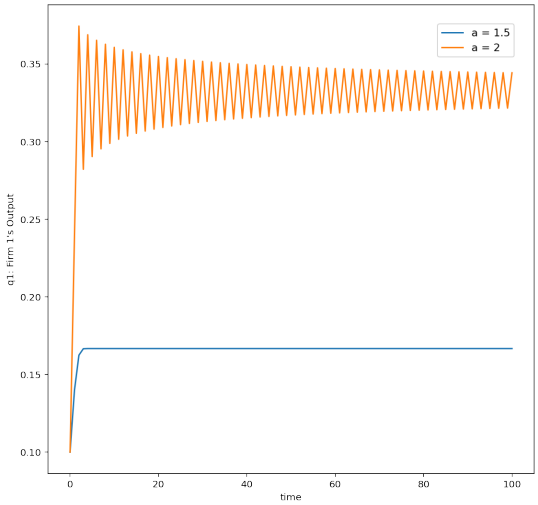


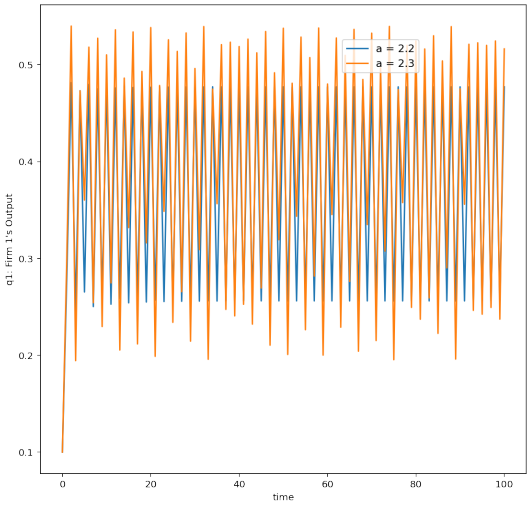


[6]:
```
plt.figure(figsize=(20, 9))

plt.subplot(121)
plt.plot(x, a15[0], label ="a = 1.5")
plt.plot(x, a20[0], label ="a = 2")
plt.xlabel('time')
plt.ylabel('q1: Firm 1\'s Output')
plt.legend(bbox_to_anchor=(0.97, 0.97), prop={'size': 11})

plt.subplot(122)
plt.plot(x, a22[0], label ="a = 2.2")
plt.plot(x, a23[0], label ="a = 2.3")
plt.xlabel('time')
plt.ylabel('q1: Firm 1\'s Output')
plt.legend(bbox_to_anchor=(0.79, 0.89), prop={'size': 11})
```

[6]: <matplotlib.legend.Legend at 0x7f072c0b3cc0>

[6]:





[0]:

[0]:

Phase Space Diagrams

May 10, 2020

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import sympy as sm
```

[3]:
```
def myopic(q1,q2, alpha1,alpha2, a,b,c1,c2, n):
    xs=[q1]
    ys=[q2]
    for i in range(n):
        xs.append(xs[i] + alpha1*xs[i]*(a-c1-2*b*xs[i]-b*ys[i]))
        ys.append(ys[i] + alpha2*ys[i]*(a-c2-2*b*ys[i]-b*xs[i]))
    return np.array([xs, ys])
```

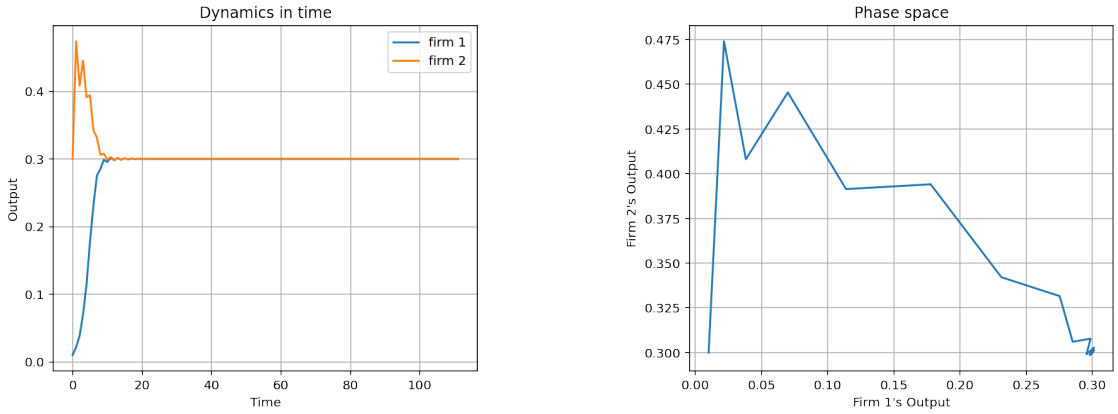
[4]:
```
def myop_map(x,y, alpha1,alpha2, a,b,c1,c2, n):

    return x + alpha1*x*(a-c1-2*b*x-b*y),  y + alpha2*y*(a-c2-2*b*y-b*x)
```

[3]:
```
iv1    = 0.01
iv2    = 0.3
alpha1 = 2
alpha2 = 2
a      = 1.9
b      = 1
c1     = 1
c2     = 1
n=111

time   = np.linspace(0, 100, 1001)

q1s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[1]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)
```

```
ax1.plot(q1s, label='firm 1')
ax1.plot(q2s, label='firm 2')
ax1.set_title("Dynamics in time")
ax1.set_xlabel("Time")
ax1.set_ylabel("Output")
ax1.grid()
ax1.legend(loc='best')

ax2.plot(q1s, q2s)
ax2.set_xlabel("Firm 1's Output")
ax2.set_ylabel("Firm 2's Output")
ax2.set_title("Phase space")
ax2.grid()
print("iv1 = ",  iv1, ", iv2 = ", iv2, ", alpha1 = ", alpha1, ", alpha2 = ",␣
 ↪alpha2, ", a = ", a, ", b = ", b, ", c1 = ", c1, ", c2 = ", c2)
```

```
iv1 =  0.01 , iv2 =  0.3 , alpha1 =  2 , alpha2 =  2 , a =  1.9 , b =  1 , c1 =
1 , c2 =  1
```

[3]:



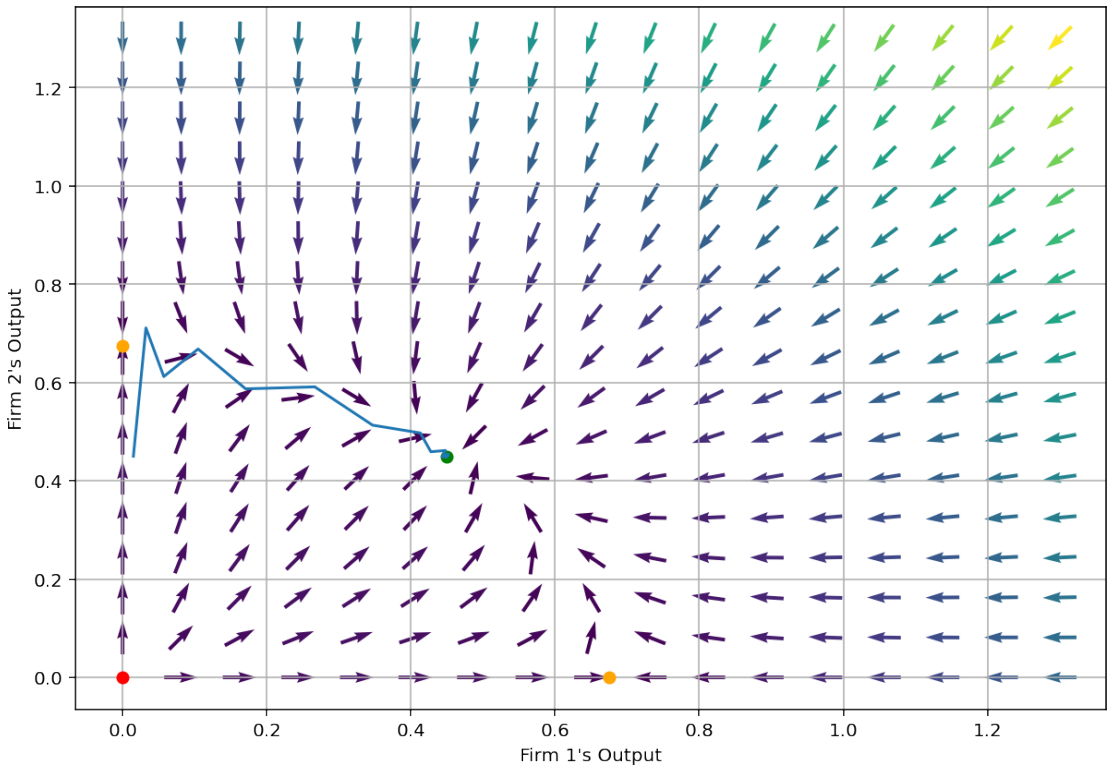
[13]:

```
fig2 = plt.figure(figsize=(10,7))
ax4 = fig2.add_subplot(1,1,1)

alpha1 = 2
alpha2 = 2
a      = 1.9
b      = 1
c1     = 1
c2     = 1

x = np.linspace(0, 1.3, 17)
y = np.linspace(0, 1.3, 17)

X1 , Y1  = np.meshgrid(x, y)
```

```
DX1, DY1 = myop_map(X1, Y1, alpha1,alpha2, a,b,c1,c2, n)
M = (np.hypot(DX1, DY1))
M[ M == 0] = 1.
DX1 /= M
DY1 /= M

ax4.plot(0,                 0,                   color='red', marker='o')
ax4.plot(1.5*(a-c1)/(2*b), 0,                    color='orange', marker='o')
ax4.plot(0,                 1.5*(a-c1)/(2*b),    color='orange', marker='o')
ax4.plot(1.5*(a-c1)/(3*b), 1.5*(a-c1)/(3*b),    color='green', marker='o')

q1s    = 1.5 * myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = 1.5 * myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[1]
ax4.plot(q1s, q2s)

ax4.quiver(X1, Y1, DX1, DY1, M, pivot='mid')
ax4.grid()
ax4.set_xlabel("Firm 1's Output")
ax4.set_ylabel("Firm 2's Output")
```

[13]: Text(0, 0.5, "Firm 2's Output")

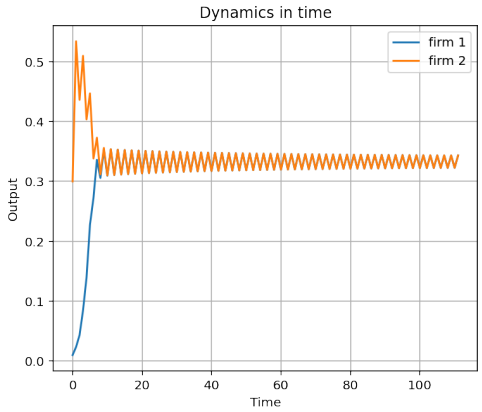
[13]:

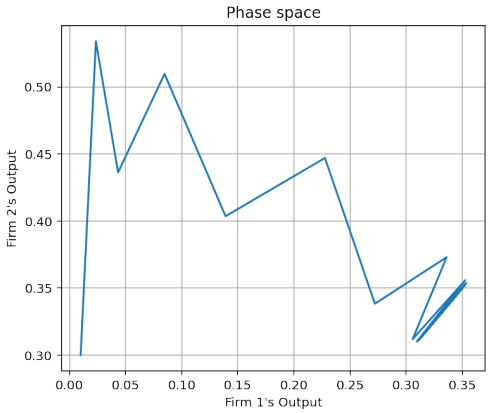


[14]:
```
iv1    = 0.01
iv2    = 0.3
alpha1 = 2
alpha2 = 2
a      = 2
b      = 1
c1     = 1
c2     = 1
n=111

time   = np.linspace(0, 100, 1001)

q1s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[1]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

ax1.plot(q1s, label='firm 1')
ax1.plot(q2s, label='firm 2')
ax1.set_title("Dynamics in time")
ax1.set_xlabel("Time")
ax1.set_ylabel("Output")
ax1.grid()
ax1.legend(loc='best')

ax2.plot(q1s, q2s)
ax2.set_xlabel("Firm 1's Output")
ax2.set_ylabel("Firm 2's Output")
ax2.set_title("Phase space")
ax2.grid()
print("iv1 = ",  iv1, ", iv2 = ", iv2, ", alpha1 = ", alpha1, ", alpha2 = ",␣
 ↪alpha2, ", a = ", a, ", b = ", b, ", c1 = ", c1, ", c2 = ", c2)
```

```
iv1 =  0.01 , iv2 =  0.3 , alpha1 =  2 , alpha2 =  2 , a =  2 , b =  1 , c1 =  1
, c2 =  1
```

[14]:



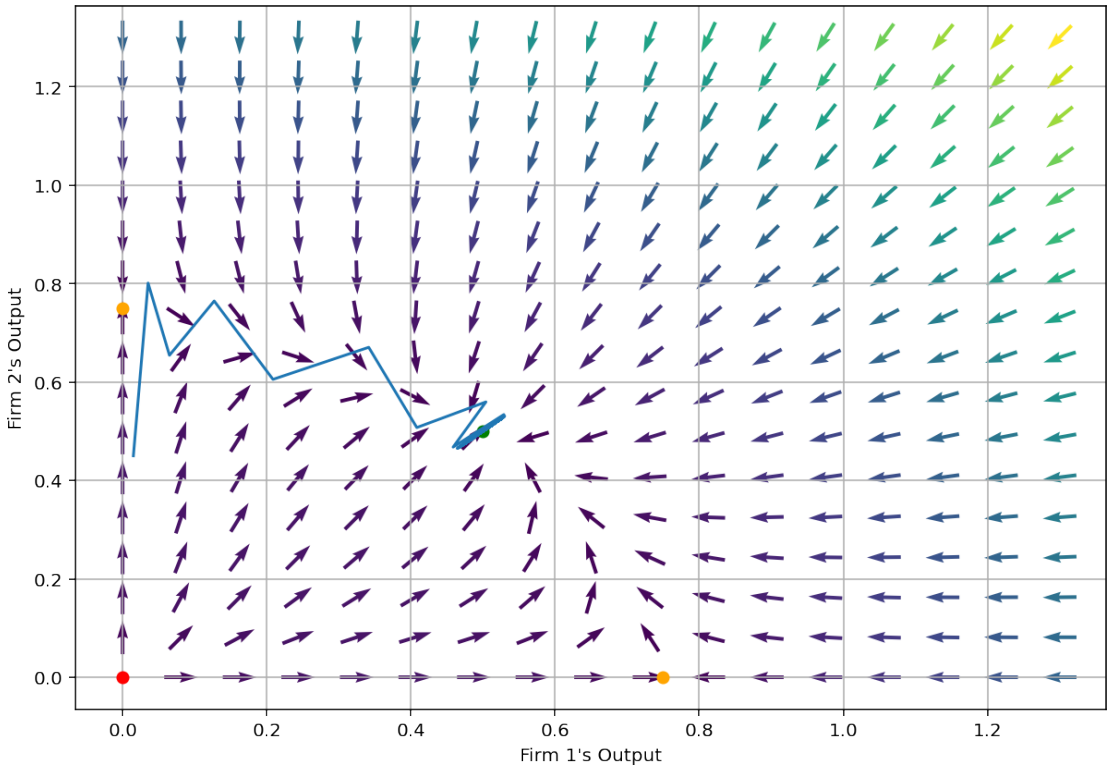


[16]:
```
fig2 = plt.figure(figsize=(10,7))
ax4 = fig2.add_subplot(1,1,1)

alpha1 = 2
alpha2 = 2
a      = 2
b      = 1
c1     = 1
c2     = 1

x = np.linspace(0, 1.3, 17)
y = np.linspace(0, 1.3, 17)

X1 , Y1  = np.meshgrid(x, y)
DX1, DY1 = myop_map(X1, Y1, alpha1,alpha2, a,b,c1,c2, n)
M = (np.hypot(DX1, DY1))
M[ M == 0] = 1.
DX1 /= M
DY1 /= M

ax4.plot(0,                 0,                  color='red', marker='o')
ax4.plot(1.5*(a-c1)/(2*b), 0,                  color='orange', marker='o')
ax4.plot(0,                 1.5*(a-c1)/(2*b),  color='orange', marker='o')
ax4.plot(1.5*(a-c1)/(3*b), 1.5*(a-c1)/(3*b),  color='green', marker='o')

q1s    = 1.5 * myopic(iv1,iv2,    alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = 1.5 * myopic(iv1,iv2,    alpha1,alpha2,    a,b,c1,c2,    n)[1]
ax4.plot(q1s, q2s)

ax4.quiver(X1, Y1, DX1, DY1, M, pivot='mid')
ax4.grid()
ax4.set_xlabel("Firm 1's Output")
```

```
ax4.set_ylabel("Firm 2's Output")
```

[16]:
```
Text(0, 0.5, "Firm 2's Output")
```

[16]:

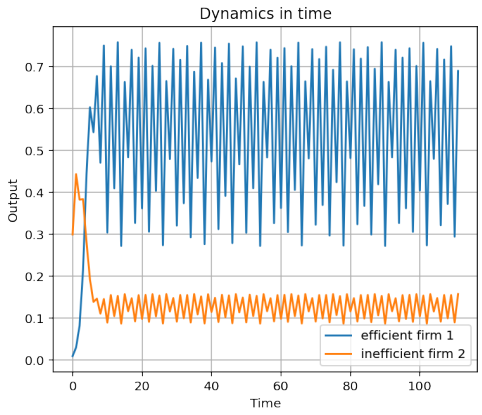


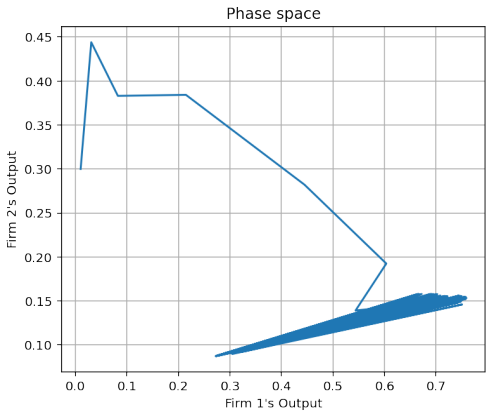
[9]:
```
iv1    = 0.01
iv2    = 0.3
alpha1 = 2
alpha2 = 2
a      = 2.1
b      = 1
c1     = 0.75
c2     = 1.25
n=111

time   = np.linspace(0, 100, 1001)

q1s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[1]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
```

```
ax2 = fig.add_subplot(1,2,2)

ax1.plot(q1s, label='efficient firm 1')
ax1.plot(q2s, label='inefficient firm 2')
ax1.set_title("Dynamics in time")
ax1.set_xlabel("Time")
ax1.set_ylabel("Output")
ax1.grid()
ax1.legend(loc='best')

ax2.plot(q1s, q2s)
ax2.set_xlabel("Firm 1's Output")
ax2.set_ylabel("Firm 2's Output")
ax2.set_title("Phase space")
ax2.grid()
print("iv1 = ",  iv1, ", iv2 = ", iv2, ", alpha1 = ", alpha1, ", alpha2 = ",␣
 ↪alpha2, ", a = ", a, ", b = ", b, ", c1 = ", c1, ", c2 = ", c2)
```

```
iv1 =  0.01 , iv2 =  0.3 , alpha1 =  2 , alpha2 =  2 , a =  2.1 , b =  1 , c1 =
0.75 , c2 =  1.25
```

[9]:





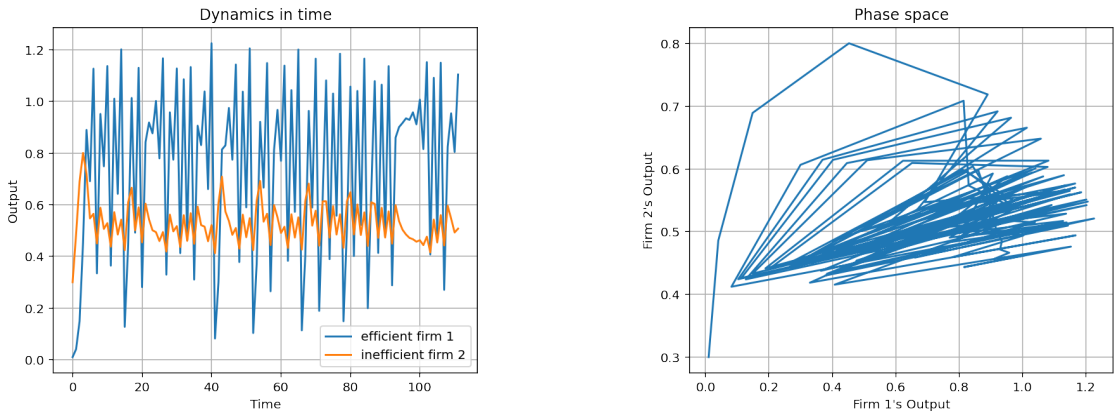
[8]:

```
iv1    = 0.01
iv2    = 0.3
alpha1 = 1.5
alpha2 = 0.5
a      = 3.1
b      = 1
c1     = 0.75
c2     = 1.25
n=111

time   = np.linspace(0, 100, 1001)
```

```
q1s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[1]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

ax1.plot(q1s, label='efficient firm 1')
ax1.plot(q2s, label='inefficient firm 2')
ax1.set_title("Dynamics in time")
ax1.set_xlabel("Time")
ax1.set_ylabel("Output")
ax1.grid()
ax1.legend(loc='best')

ax2.plot(q1s, q2s)
ax2.set_xlabel("Firm 1's Output")
ax2.set_ylabel("Firm 2's Output")
ax2.set_title("Phase space")
ax2.grid()
print("iv1 = ",  iv1, ", iv2 = ", iv2, ", alpha1 = ", alpha1, ", alpha2 = ",␣
 ↪alpha2, ", a = ", a, ", b = ", b, ", c1 = ", c1, ", c2 = ", c2)
```

```
iv1 =  0.01 , iv2 =  0.3 , alpha1 =  1.5 , alpha2 =  0.5 , a =  3.1 , b =  1 ,
c1 =  0.75 , c2 =  1.25
```

[8]:



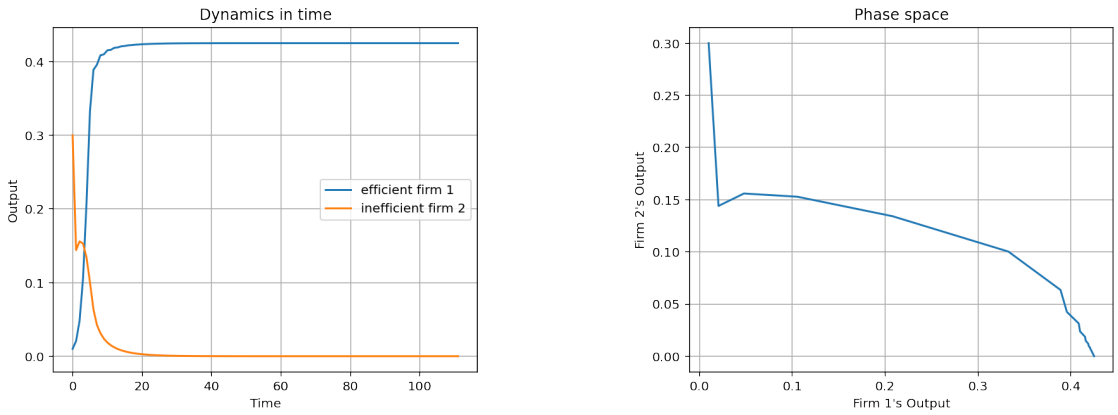
[19]:

```
iv1    = 0.01
iv2    = 0.3
alpha1 = 2
alpha2 = 2
a      = 1.6
b      = 1
```

```
c1      = 0.75
c2      = 1.25
n=111

time    = np.linspace(0, 100, 1001)

q1s     = myopic(iv1,iv2,      alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s     = myopic(iv1,iv2,      alpha1,alpha2,    a,b,c1,c2,    n)[1]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

ax1.plot(q1s, label='efficient firm 1')
ax1.plot(q2s, label='inefficient firm 2')
ax1.set_title("Dynamics in time")
ax1.set_xlabel("Time")
ax1.set_ylabel("Output")
ax1.grid()
ax1.legend(loc='best')

ax2.plot(q1s, q2s)
ax2.set_xlabel("Firm 1's Output")
ax2.set_ylabel("Firm 2's Output")
ax2.set_title("Phase space")
ax2.grid()
print("iv1 = ",  iv1, ", iv2 = ", iv2, ", alpha1 = ", alpha1, ", alpha2 = ",␣
 ↪alpha2, ", a = ", a, ", b = ", b, ", c1 = ", c1, ", c2 = ", c2)
```

```
iv1 =  0.01 , iv2 =  0.3 , alpha1 =  2 , alpha2 =  2 , a =  1.6 , b =  1 , c1 =
0.75 , c2 =  1.25
```

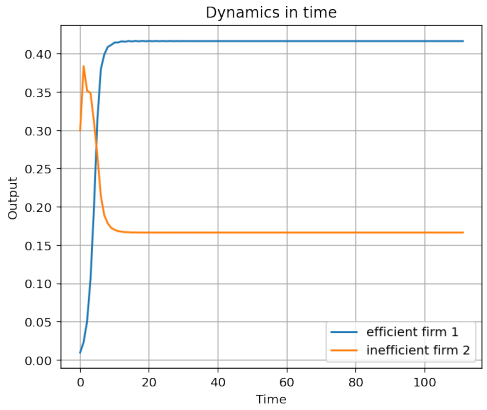
[19]:

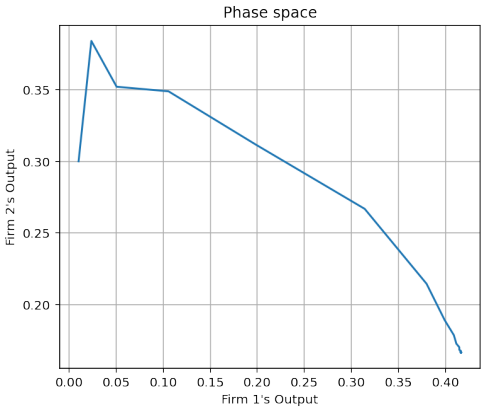


[9]:
```
iv1    = 0.01
iv2    = 0.3
alpha1 = 2
alpha2 = 2
a      = 2
b      = 1
c1     = 1
c2     = 1.25
n=111

time   = np.linspace(0, 100, 1001)

q1s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[0]
q2s    = myopic(iv1,iv2,     alpha1,alpha2,    a,b,c1,c2,    n)[1]

fig = plt.figure(figsize=(15,5))
fig.subplots_adjust(wspace = 0.5, hspace = 0.3)
ax1 = fig.add_subplot(1,2,1)
ax2 = fig.add_subplot(1,2,2)

ax1.plot(q1s, label='efficient firm 1')
ax1.plot(q2s, label='inefficient firm 2')
ax1.set_title("Dynamics in time")
ax1.set_xlabel("Time")
ax1.set_ylabel("Output")
ax1.grid()
ax1.legend(loc='best')

ax2.plot(q1s, q2s)
ax2.set_xlabel("Firm 1's Output")
ax2.set_ylabel("Firm 2's Output")
ax2.set_title("Phase space")
ax2.grid()
print("iv1 = ",  iv1, ", iv2 = ", iv2, ", alpha1 = ", alpha1, ", alpha2 = ",␣
 ↪alpha2, ", a = ", a, ", b = ", b, ", c1 = ", c1, ", c2 = ", c2)
```

```
iv1 =  0.01 , iv2 =  0.3 , alpha1 =  2 , alpha2 =  2 , a =  2 , b =  1 , c1 =  1
, c2 =  1.25
```

[9]:

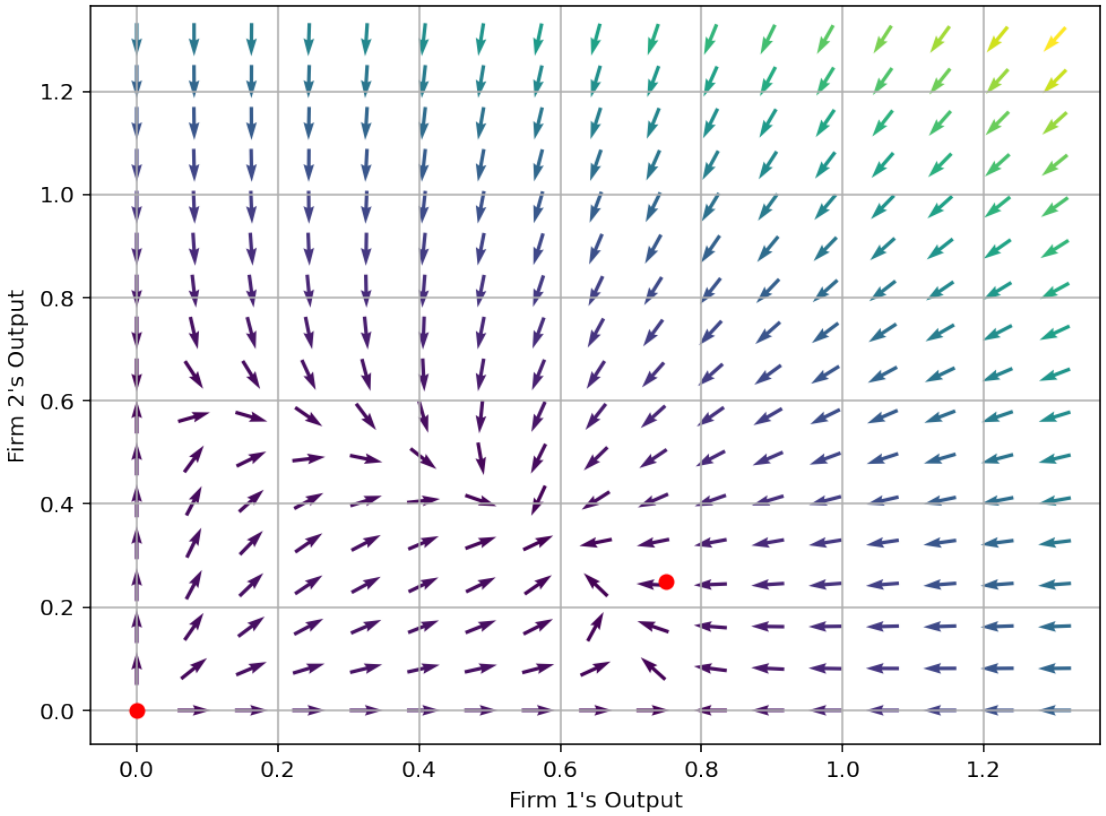




[10]:
```
fig3 = plt.figure(figsize=(8,6))
ax4 = fig3.add_subplot(1,1,1)

alpha1 = 2
alpha2 = 2
a      = 2
b      = 1
c1     = 1
c2     = 1.

x = np.linspace(0, 1.3, 17)
y = np.linspace(0, 1.3, 17)

X1 , Y1  = np.meshgrid(x, y)
DX1, DY1 = myop_map(X1, Y1, alpha1,alpha2, a,b,c1,c2, n)
M = (np.hypot(DX1, DY1))
M[ M == 0] = 1.
DX1 /= M
DY1 /= M

ax4.plot(0,                0,                    color='red', marker='o')
ax4.plot(0.75,             0.25,                 color='red', marker='o')

ax4.quiver(X1, Y1, DX1, DY1, M, pivot='mid')
ax4.grid()
ax4.set_xlabel("Firm 1's Output")
ax4.set_ylabel("Firm 2's Output")
```

[10]: Text(0, 0.5, "Firm 2's Output")

[10]:



[0]:

[0]:

Bifurcation Diagrams

May 10, 2020

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

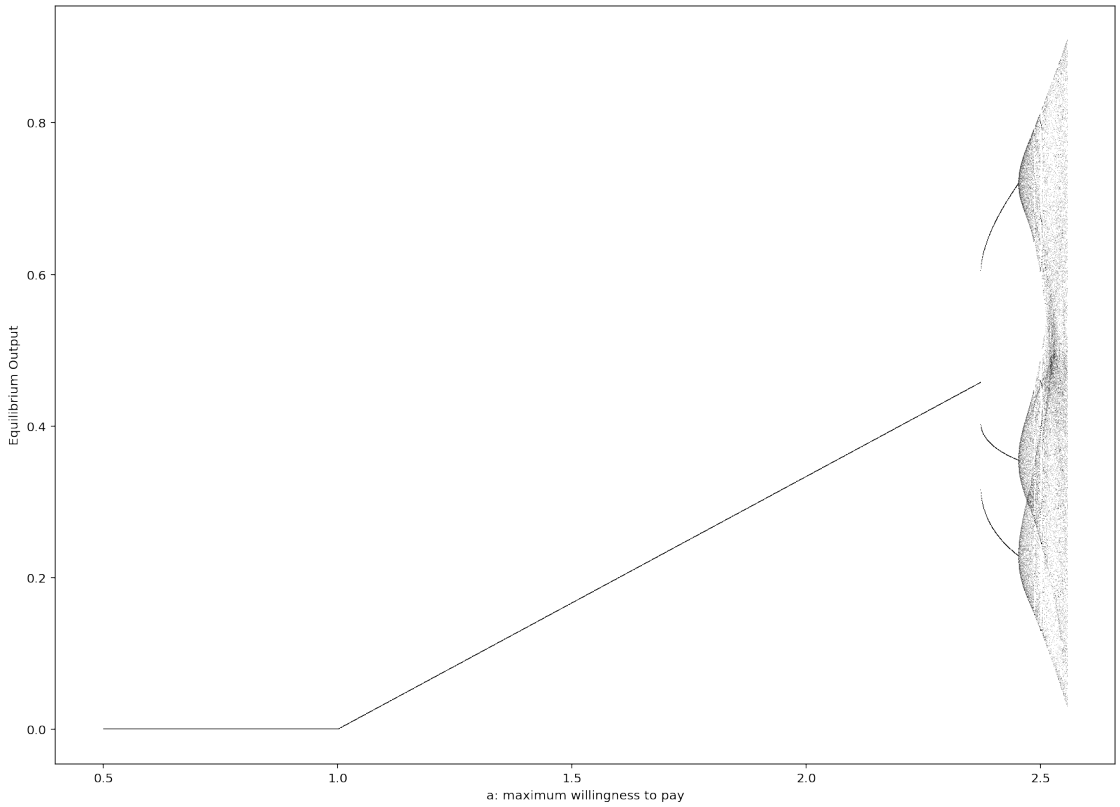
[3]:
```
n = 10000
a = np.linspace(.5, 4.0, n)
iterations = 10000
last = 100
x = 1e-5 * np.ones(n)
y = 1e-5 * np.ones(n)

def myop_map(x,y, alpha1,alpha2, a,b,c, n):

    return x + alpha1*x*(a-c-2*b*x-b*y),  y + alpha2*y*(a-c-2*b*y-b*x)
```

[4]:
```
plt.figure(figsize=(15, 11))
for i in range(iterations):
    x = myop_map(x,y, 2,2, a,1,1, n)[0]
    y = myop_map(x,y, 2,2, a,1,1, n)[1]

    if i >= (iterations - last):
        plt.plot(a, x, ',k', alpha=0.25)

plt.xlabel('a: maximum willingness to pay')
plt.ylabel('Equilibrium Output')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
overflow encountered in multiply
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in add
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in subtract
```

[4]:
```
Text(0, 0.5, 'Equilibrium Output')
```

[4]:



[5]:
```
plt.figure(figsize=(15, 11))
alpha1 = 2
alpha2 = 2

a = np.linspace(2.362, 2.7, n)
b = 1
c = 1

plt.figure(figsize=(11, 8))
for i in range(iterations):
    x = myop_map(x,y, alpha1,alpha2, a,b,c, n)[0]
    y = myop_map(x,y, alpha1,alpha2, a,b,c, n)[1]

    if i >= (iterations - last):
        plt.plot(a, x, ',k', alpha=0.25)

plt.xlabel('a: maximum willingness to pay')
plt.ylabel('Equilibrium Output')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
overflow encountered in multiply
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
```

```
invalid value encountered in subtract
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in add
```

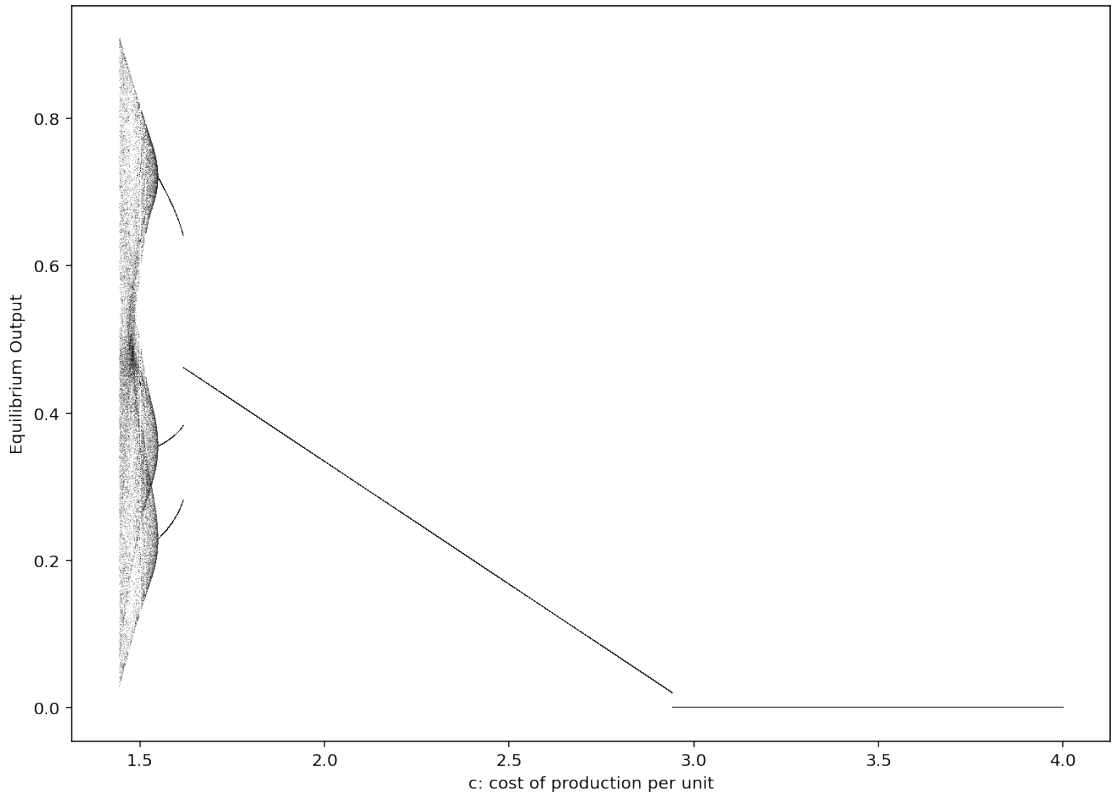
[4]:
```
plt.figure(figsize=(15, 11))
alpha1 = 2
alpha2 = 2
n = 10000
a = 3
b = 1
c = np.linspace(0.1, 4, n)

plt.figure(figsize=(11, 8))
for i in range(iterations):
    x = myop_map(x,y, alpha1,alpha2, a,b,c, n)[0]
    y = myop_map(x,y, alpha1,alpha2, a,b,c, n)[1]

    if i >= (iterations - last):
        plt.plot(c, x, ',k', alpha=0.25)

plt.xlabel('c: cost of production per unit')
plt.ylabel('Equilibrium Output')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
overflow encountered in multiply
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in add
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in subtract
```

[4]:
```
Text(0, 0.5, 'Equilibrium Output')
```

[4]:
```
<Figure size 1080x792 with 0 Axes>
```

[4]:

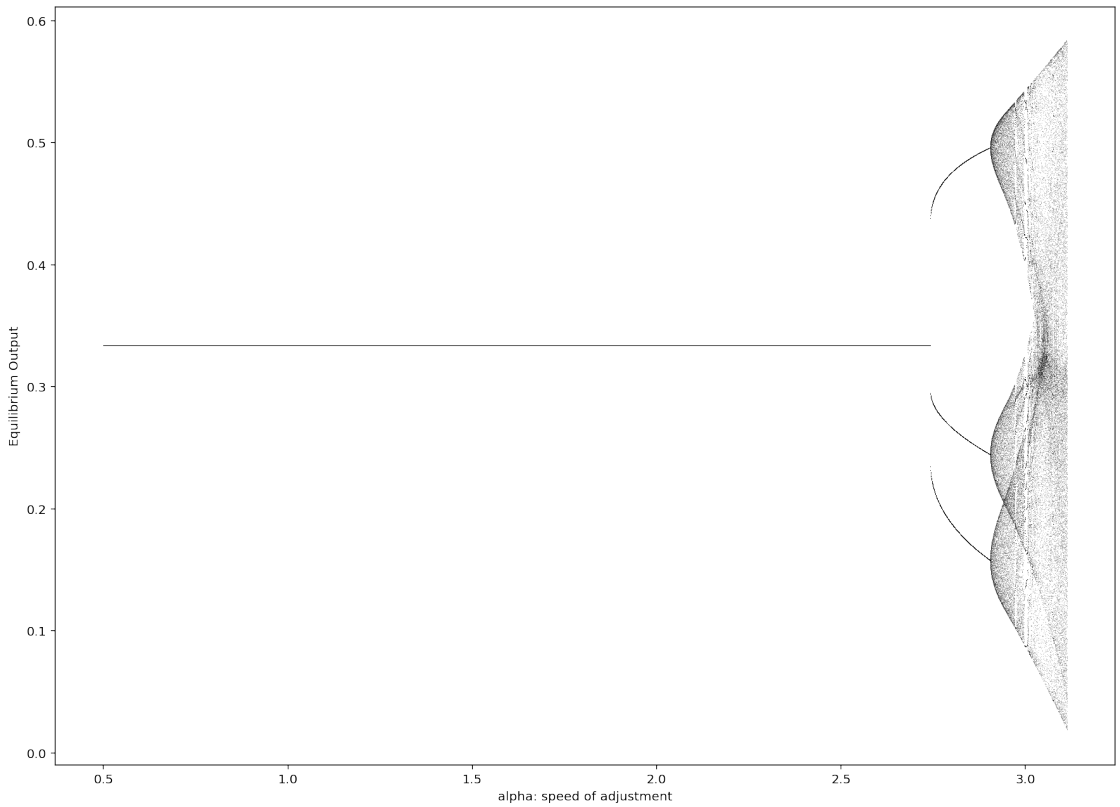


[8]:
```
plt.figure(figsize=(15, 11))
alpha = np.linspace(.5, 4.0, n)
x = 1e-5 * np.ones(n)
y = 1e-5 * np.ones(n)

for i in range(iterations):
    x = myop_map(x,y, alpha,alpha, 2,1,1, n)[0]
    y = myop_map(x,y, alpha,alpha, 2,1,1, n)[1]

    if i >= (iterations - last):
        plt.plot(alpha, x, ',k', alpha=.25)

plt.xlabel('alpha: speed of adjustment')
plt.ylabel('Equilibrium Output')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
overflow encountered in multiply
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in subtract
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in add
```

[8]: Text(0, 0.5, 'Equilibrium Output')

[8]:



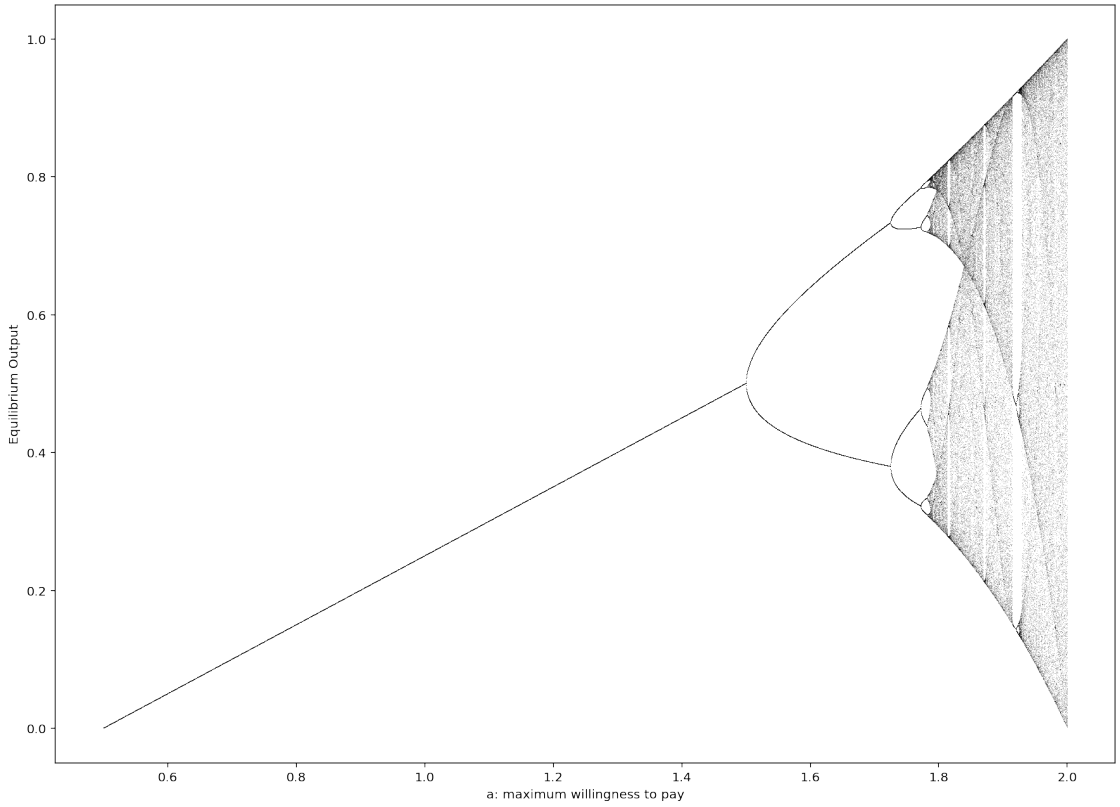
[9]:
```
plt.figure(figsize=(15, 11))
alpha1 = 2
alpha2 = 0.25

a = np.linspace(0.5, 3, n)
b = 1
c1 = 0.5
c2 = 1.5

x = 1e-5 * np.ones(n)
y = 1e-5 * np.ones(n)

for i in range(iterations):
    x = myop_map(x,y, alpha1,alpha1, a,b,c1, n)[0]
    y = myop_map(x,y, alpha1,alpha1, a,b,c2, n)[1]

    if i >= (iterations - last):
        plt.plot(a, x, ',k', alpha=.25)
```

```
plt.xlabel('a: maximum willingness to pay')
plt.ylabel('Equilibrium Output')
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
overflow encountered in multiply
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in add
/usr/local/lib/python3.6/dist-packages/ipykernel/__main__.py:10: RuntimeWarning:
invalid value encountered in subtract
```

[9]:
```
Text(0, 0.5, 'Equilibrium Output')
```

[9]:



[0]:

[0]:

[0]: