Modelling the Effect of Present Bias on Cost/Benefit Analysis within Market-based and Cutoff Pollution Regulation Frameworks

University College London
Spring Term 2021
ECON0052 Project

Frank May, Ben Willert, Archie Dickinson, Max Binns, Matthew Jack-Kee
Department of Economics
University College London, UK

Abstract

We develop a model to analyse the impact of present bias on firms' decisions regarding decarbonisation in a regulatory framework with: A) a linearly-decreasing supply of 'grandfathered' pollution permits, and B) a pre-specified cutoff point whereby pollution-inefficient firms' production is discontinued. We find that firms' actions are resistant to present bias insofar as it rarely causes dynamic preference reversals, which would lead to insufficient green technology accumulation to continue operating beyond the cutoff point. We suggest that policymakers append existing permit schemes with cutoff rules in order to incentivise a present-bias-robust incentive to invest in greener technology.

1. Introduction

1.1: Objectives
Within the context of pollution generation in the economy, the benefits of increased manufacturing and output in the present can often outweigh the discounted costs of pollution in the future. Likewise, the high start-up costs of 'cleaner' production methods can remain unattractive in comparison to the discounted future benefits of such production. To tackle these social welfare inefficiencies, market-based regulations such as pollution permits have been introduced to cap emissions at known quantities. This paper first explores the concepts of time inconsistency within intertemporal choice in the literature, as well as the effectiveness of existing permit schemes.

We then introduce a model to quantify the sensitivity of firms' decision making with respect to their present bias. We model a market of multiple firms with varying levels of present bias who are regulated within a cap-and-trade pollution permit scheme. There are multiple time periods, with an eventual 'cutoff' date for environmentally inefficient production processes. Given the current application of pollution permit schemes in multiple countries, as well as many proposed eventual bans on highly polluting production methods, we believe this paper provides results that are highly applicable to current real-world situations. Our model quantitatively analyses the implications and limitations of pollution permit schemes with regard to cognitive bias.

By critically evaluating the findings of the model in conjunction with our hypotheses, we will be able to add colour to the literature-based predictions. We aim for our findings to contribute to, and spur additional discussion on the future outcomes of market-based pollution schemes, and their relevance in a framework with time inconsistent agents.

1.2: Cost-Benefit Analysis and Cognitive Bias
While typically considered distinct, by allocating the right to pollute by a certain amount to firms, pollution permit markets can be considered a Coasean bargaining system (Hanley, 2019). By endogenising the obligation to regulate pollution emissions into profit functions, a permit market outsources cost-benefit analysis (CBA) to firms, as they seek to optimise their profits subject to this new constraint. CBA is a method to evaluate the social welfare impact of a given policy or project. All costs and benefits are monetised and calculated into a single number known as the net present value (NPV), which estimates the net positive or negative welfare effect. As individual firms represent stakeholders in the Coasean bargaining system of a permit market, this is an instance of a 'bottom-up' CBA (Carolus *et al*, 2018). The characteristic of permit markets to find the lowest cost of achieving a given emission level is perfectly reflected by the nature of bottom-up CBA to find equitable distributions of costs and benefits. Furthermore, this instance of bottom-up CBA is unique, as typically they only work within a 'representative scale', which doesn't characterise the global issue of pollution and emissions.

Inconsistent intertemporal choice, namely present bias and time inconsistency, hinder the ability of agents within the economy from applying effective cost-benefit analyses. Time consistent discounting is generally assumed across agents in an economy, which fails to take into account the empirically frequent present bias (Thaler & Shefrin, 1981; Benzion *et al*, 1989). Inaccurate calculations, and more often complete naivete to cognitive bias, can result in distorted NPV estimates, which fail to represent the true social welfare impacts across multiple time periods

1.3: Permit Schemes
Permit trading mechanisms were established to allocate property rights to polluters which subsequently leads to permit trading among firms that pollute heterogeneously. A tradable pollution permits system (TPP) works by firstly creating a cap which total emissions cannot exceed (without incurring a very large fine), followed by a period where rights to pollute are distributed (either by a regulatory body or auction) to firms within the industry. Subsequently, firms are able to trade these permits so that a fair market price for pollution permits is created. One core principle of a cap-and-trade scheme is that the pollution limit (the cap) is reduced over time until there is a known cutoff point at which firms must adhere to new polluting standards, or they face production bans. The prohibition on the sale of internal combustion engine vehicles beyond 2030 in the UK demonstrates this. This incentivises longer term investment in less pollutant-intensive production methods while still allowing firms to pollute during the period leading up to the cutoff point (Hanley et al., 2019). The central principle of permit trading encourages firms that have lower marginal cost of abatement than that of permit prices to choose to invest in abatement technologies and therefore reduce emissions.

By establishing a price per unit of pollutant emitted (CO2 or SO2 for example) and hence a market for said emissions, tradable pollution permit systems effectively assign property rights to pollute to different parties. Without the establishment of a market that assigns property rights to free goods such as clean air or water quality, firms can continue to abuse the free good at the expense of the public. Tradable permit schemes were introduced by Professors Thomas Crocker and J. Dales in the 1960’s in an attempt to quantify the value of nature and the public goods that users benefited from. Although initially met with a degree of friction, cap-and-trade schemes have proven very effective in reducing total emissions due to the ability for the markets to push for the continued decrease in industry-wide emissions thanks to the market dynamics of a TPP. There are two methods of assigning permits to agents in an industry: by auctioning rights to pollute or by ‘grandfathering’ (allocating emissions permits by historical emissions levels and then allowing market dynamics between agents to form permit prices). In this latter form, all permit trading is inter-firm, and a market price is formed so that low cost-of-abatement firms sell permits to high cost-of-abatement firms.

One of the most successful pollution trading schemes is the EU Emissions Trading Scheme (ETS) which was established in 2005 as a way to significantly reduce greenhouse gas emissions produced by the 11,000 CO2 producers that are signed up to the ETS in the EU nations. The EU ETS is the cornerstone for greenhouse gas abatement in the region and played a significant role in the EU’s push to cut GHG emissions by 20% from 1990 to 2020 (Commission to the European Parliament and the Council, 2020). Permit trading schemes unsurprisingly present a number of benefits and drawbacks. Jenkins and Lamech (1992) affirm that these mechanisms allow for flexibility to allow industry growth while allowing for environmental concerns to be accounted for and promoting the general idea that pollution controls can and will be met. However, the authors also suggest that success dependence varies on the number of parties involved in the TPP and the non-existence of transaction costs so that firms are encouraged to trade permits when it is profitable to do so. Initial allocation of permits by regulatory bodies can also greatly affect firms' profits. Empirical evidence indicates that large windfall profits can be accrued from generous allocation of permits (Laing et al. 2013). Firms with more carbon efficient production methods could potentially make significantly more profit ex-post due an ability to sell permits to higher marginal profit firms. This finding is also found by our model, with the lower-polluting firms gaining relatively more from trade in permits.

1.4: Present Bias and Discounting

In our bottom-up CBA framework, firms seek to maximise NPV of all future profits. At any time t, they forecast the profits for periods t = 1, …, T, and from there, and select either investment or continuing production. However, by decentralising the policy decision, firms are now the key decision makers. Therefore, any behavioural inconsistencies present in the wider economy are more likely to manifest themselves; the most common being present bias. Moreover, it is not only firms afflicted by present bias, it affects regulatory bodies, too; even the UK Government frequently implements declining-discount rate (DDR) schedules across longer horizons within policy appraisal (Hanley, 2019).

Present bias is a behavioural concept that refers to one's tendency to seek immediate reward and avoid immediate cost (O'Donoghue & Rabin, 1999). It refutes the neo-classical argument of time consistency; that the ranking of outcome-time pairs does not change depending on when one is asked (Gurguc, 2020). With time consistency, if an individual chooses £10 today over £20 next week, she will also choose £10 in 52 weeks' time over £20 in 53, yet empirical studies frequently find this is violated (Thaler, 1981; Read & van Leeuwen, 1998). This suggests that discount rates across a shorter horizon are significantly higher than across periods further in the future.

Present biased preferences and a declining discount rate are captured effectively by a Quasi-Hyperbolic discount model (Phelps & Pollak, 1968; Laibson, 1997). Overall utility at time *t* is:

$$U_t(c_t, \dots, c_T) = u(c_t) + \beta\delta u(c_{t+1}) + \beta\delta^2 u(c_{t+2}) + \cdots + \beta\delta^T u(c_{t+T})$$

For $\beta, \delta \in \{0,1\}$

Here, the discount rate depends on the time horizon considered, as is characteristic of present biased preferences.

Although more frequently framed in terms of impatience and willpower, literature suggests present bias is also prevalent in environmental settings, given the large upfront costs and less easily discernible future benefits.

Specifically, a strong prevalence is suggested in emissions-related phenomena. Payments for Ecosystem Services (PES) are programmes that compensate farmers for forest preservation and reforestation (Clot & Stanton, 2014). Their payment structures offer key insight into the existence of present bias for reducing emissions. Pago de Servicios Ambientales (PSA) in Costa Rica presented farmers with a large upfront payment in the first year, followed by smaller annual payments. Conversely, the Bolivian Fundación Natura (FN) offered farmers equal annual payments. PSA was significantly oversubscribed, whereas FN enrolment rates were poor. The greater popularity of the Costa Rican front-loaded payment structure suggests that farmers have declining discount rates and exhibit present bias. More generally, this substantiates how environmental activities entail a large upfront cost, with benefits accruing further in the future. Duquette et al. (2011) corroborate this, highlighting how earlier adopters of US sustainable farming policy exhibited empirically lower discount rates for future benefits; they value future benefits more, and so were more likely to adopt early.

Regarding major regulatory bodies, we postulate that the existence of present bias in a market for permits may result in not following through with caps after pressure from lobbying groups. We further suggest that regulatory bodies could be more susceptible to present bias, as they have no monetary incentive, and instead are bound by re-election constraints. Mechanically, governments may base future pledges to reduce permits off a lower, longer-term discount rate, which values future benefits more strongly. However, when the time comes, the higher, short-term discount rate induces governments to assuage the demands of permit-protestors in order to seek the more proximal utility from re-election. Consequently, this dynamic preference reversal may undermine government credibility at capping pollution permits, which could push firms to invest less in green technologies and further harm social welfare.The argument is analogous for firms, who, upon decision, cannot help but apply the shorter-term discount rate and continue to produce. As such, this dynamic preference reversal inhibits green investment.

1.5: Hypotheses

Theory suggests present bias will play a role through affecting the firms' discounting stage of CBA and push back the time when firms eventually decarbonise. If firms experience strong present bias, they are likely to push back the costs of decarbonising for as long as possible, in favour of pulling forward continuing profit from their high emission technology that is already in production. Those firms with business models to favour today's profit over decarbonising will risk long term profit as they near the government's set cutoff date for pollution permits. The firms' decision here may be due to them betting against the government sticking to their emission targets and suggests that they expect the government to extend the cutoff date, incentivising the firms to continue to pollute.

However, if governments are clear with their target dates, firms may likely decide to endure the costs of decarbonising today, to experience the profits in the future, which is the opposite of what present bias may conspire to alter. Firms will observe that there is an opportunity to earn long term profit by decarbonising, both through the continued production of low emission technology as well as the trading of pollution permits to the polluting firms.

The decision made by firms depends on their view of the probability that the government will stick to their emission targets. Many predict that the government will continue to extend the dates for emission targets and thus present bias will tell them to put off decarbonising in the future. Therefore, the more present-biased companies are likely to push back the costs of decarbonising and prioritise today's earnings. In the case of car production, this will lead to fewer cars being produced in the future when pollution permits approach the cutoff date.

There are, however, some problems that arise from the reduction of the pollution permit cap. Noll (1982) suggests that as more firms move towards low emission technology, market failures may arise. "Thin markets" are likely to occur as more producers move towards low emission technology, leading to high transaction costs and noisy price signals when firms trade pollution permits. This undermines how much pollution actually costs. Fewer polluters will also lead to an uncompetitive permit market, where the polluter substantially controls the price of the permits due to their monopsony power. Noll argues that there is a design problem and that there should be a separate market for each pollution receptor point and emissions are only detected in the area they are made. This is impossible and infeasible.

2. Methodology, Model Development and Results

2.1: Qualitative objective

Our objective is to generate an environment within which there is a marketplace with a global cap on the amount of pollution that can be emitted in a given time period. This cap is then reduced linearly each year (which is known by the firms) before reaching a cutoff at a pre-specified point in time (also known by firms) after which, those who aren't sufficiently pollution-efficient are removed from the market. To quantitatively examine our theoretical predictions, we seek to emulate the intertemporal decision-making process firms must undertake within this environment and examine the extent with which present bias adversely affects their actions.

To aid in this explanation's simplicity, we will use car production as an analogous real-world market to draw practical comparisons and explain motivations for certain modelling decisions. Tesla generates a large proportion of its revenues from the sale of pollution permits under California's 'Zero Emission Vehicle' regulation and the EU's emission-trading scheme (ETS) (Tesla, 2020). Furthermore, several countries have committed to banning the production of internal combustion engine vehicles past some point in the future. As such, this should add some analogical usefulness and colour to our theoretical exposition.

2.2: Permit Market Infrastructure

The global cap of pollution permits is defined as m. We assume all *n* firms are equally sized, and there is no auction scheme to allocate permits, meaning each firm is thus apportioned *m*/*n* permits in a 'grandfathering' manner. Without any trade in permits, firm *i*'s pre-trade quantity (PTQ) is:

$$c_{i,\, pre-trade} = \frac{m}{n\,\gamma_i}$$

In subsequent time periods, the number of permits is reduced linearly. If firms want to sell pollution permits to each other, at least one firm must renege on producing some of their PTQ, and will then be able to sell the permits they would've used to allow another firm to produce above their PTQ.

2.3: Firms

Firms are identified through a heterogenous γ, which parameterises the quantity of pollution emitted per unit of production. A firm with a high γ will pollute more per unit of production versus a firm with a low γ. For a given sample of firms, their γ parameters are drawn from a normal distribution truncated at {0, infinite}.

$$\gamma \sim N(\mu, \sigma),$$
$$\gamma \in \{0, \infty\}$$

This constrains firms from depolluting in production, which in exceptional cases might not be an accurate representation of the real world (Brewdog, 2020). However for the purposes of our model, this is crucial in ensuring well-behaved profit functions and non-negative optimal production quantities.

2.3: Profit Function

To systematically determine optimal points of production, we make direct use of a simple *post-trade* profit function of the following form:

$$\pi_{post-trade}(c, \gamma, \lambda, p, m, n) = \gamma(100 + \lambda)c^{\left(\frac{1}{1+e^{-(\gamma-0.5)}}\right)} + p\gamma\left(\frac{m}{n\gamma} - c\right)$$

p = permit price
n = number of firms
λ = accumulated investment

The first term of which is a *pre-trade* profit function whereby firms can only produce and sell cars, and cannot choose to renege on production in order to sell permits. The pre-trade profit function is shown in fig.1.

$$\pi_{pre-trade}(c,\ \gamma,\ \lambda) = \gamma(100 + \lambda)c^{\left(\frac{1}{1+e^{-(\gamma-0.5)}}\right)}$$

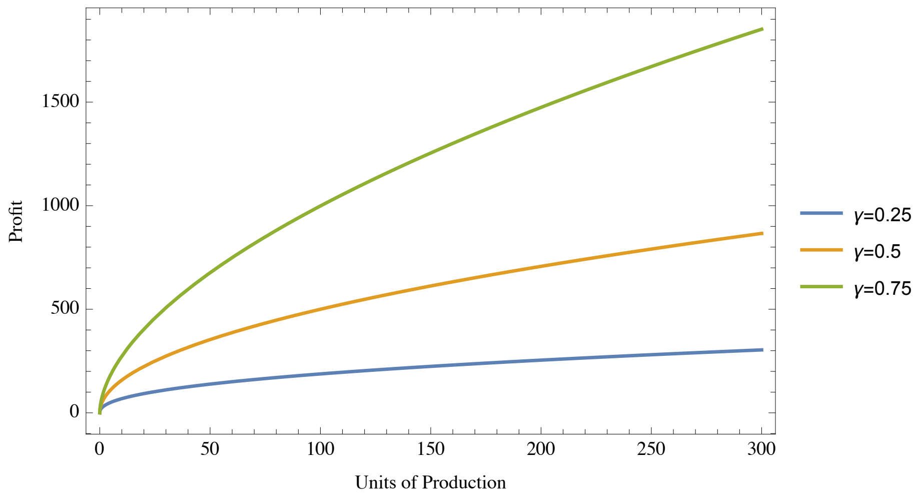


Figure 1

And the second term is the profit earned from selling permits resulting from choosing not to produce a given quantity of cars. The firm will decide on their optimal number of units of production, which will cost $p\gamma$ for every unit the firm exceeds their pre-trade quantity by, or will give the same as profit for every unit the firm gives up relative to their pre-trade quantity. The profit surface post-trade looks like the following:

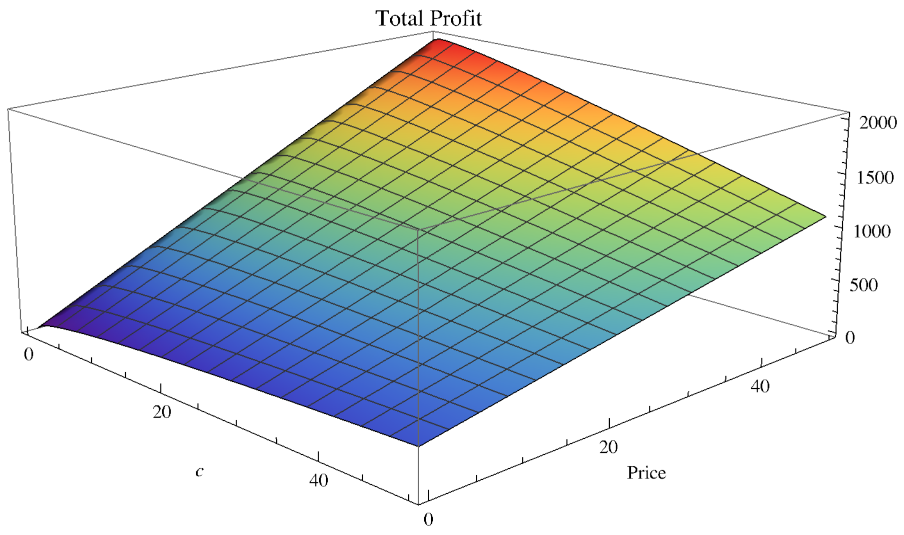


Figure 2

There are several features of this function that are important to ensure the general model runs smoothly and produces realistic results:

Exponent dependent on γ
This takes into account the idea that variable costs will generally be higher in companies with a less pollution-efficient production process, which we assume is passed directly into profits. It also allows for heterogeneous exponents, meaning each optimal production point will be unique.

Exponent always below 1
This causes diminishing marginal profit, convex profit functions, making optimisation easier. This is performed through the use of a transformed sigmoid function such that its first derivative's peak occurs at $\gamma = 0.5$.

γ product in coefficient term
Since variable costs are already partly explained by the exponent taking γ as a function, having a γ product in the coefficient of the permit profit might seem unnecessary. However, without this feature, lower γ firms end up buying permits and high γ firms selling permits, which contravenes

common sense. For a mathematical derivation, see Appendix 1.

2.4: Supply and Demand in the Permit Market
In our environment at pre-trade quantities, marginal profit for low-γ firms is very low, while marginal profit for high-γ firms is very high. It is therefore clearly beneficial for the low-γ firms to produce fewer cars and sell the resulting permits to the high-γ firms. To calculate an equilibrium price for which trade can occur, we set up market supply and demand curves, the solution of which will clear the permit market.

These curves are produced through the sum of each of the individual firm's willingness to supply and demand. Individual supplies and demands are calculated by determining each firms' optimal point of production. This is done by differentiating the profit function with respect to c and rearranging to find c as a function of p:

$$\partial_c \pi_{post-trade} = \partial_c \pi_{pre-trade} - p\gamma = 0$$

There is no need to use an explicit Lagrangian, as the global permit constraint is already implicitly included in the post-trade profit function (in the form of the PTQ).

Once this is done for each firm, supply and demand functions can be calculated as the difference between the optimal production point and the PTQ as functions of p. Max[f(p), 0] operators are used to ensure negative demand/supplies are not calculated. At a given price, market supply and demand is simply the sum of the individual supply and demand functions. As usual, equilibrium prices are calculated where market supply == market demand. For a 25 firm market with 1000 permits, no accumulated investment and γ distributed with mean 0.5 and standard deviation 0.15, an example of an equilibrium would be:

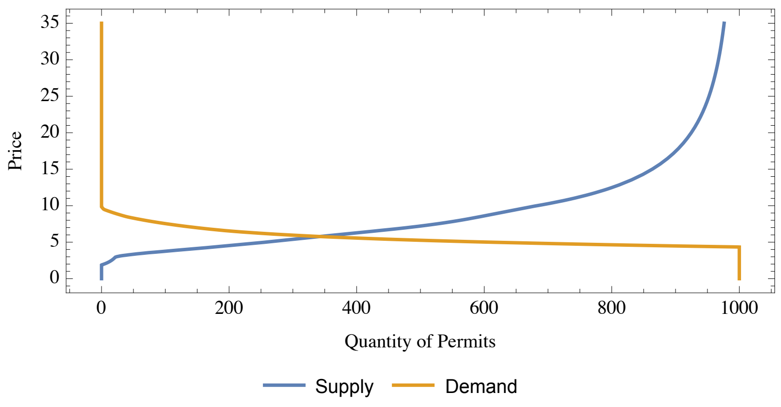


Figure 3

There is a clear asymptote in supply at m = 1000, which is as should be, as the maximum number of permits that can be supplied is constrained by the global permit supply. Demand is artificially cut to 0 when m=1000 (it would tend towards infinite as p tends towards 0, which would ruin the scale of the graph). The equilibrium price was £5.77. For each individual firm, we can see what their individual positions in the market were:

Firm #	Pollution-eff	Q(Pre-trade)	Q(Post-trade)	Total permits required	Permits bought/sold	Pre-trade profit	Post-trade Profit	Post / pre profit ratio	Beta
1	0.734084	54.4897	169.867	124.697	84.6969	683.994	801.092	1.1712	0.99271
2	0.729825	54.8077	167.098	121.952	81.9523	679.374	790.925	1.1642	0.694314
3	0.289965	137.948	40.7462	11.815	-28.185	263.185	315.281	1.19795	0.0703155
4	0.391419	102.192	53.9722	21.1257	-18.8743	349.026	367.113	1.05182	0.908433
5	0.305123	131.095	42.4328	12.9472	-27.0528	275.693	321.963	1.16783	0.618401
6	0.513622	77.8783	78.2248	40.178	0.177972	460.037	460.038	1.	0.369975
7	0.406911	98.3017	56.4561	22.9726	-17.0274	362.611	376.732	1.03894	0.615432
8	0.383747	104.235	52.794	20.2595	-19.7405	342.349	362.54	1.05898	0.811454
9	0.486735	82.1802	71.8557	34.9747	-5.02531	434.837	435.818	1.00225	0.721655
10	0.184701	216.566	31.144	5.75233	-34.2477	178.514	276.623	1.54958	0.36365
11	0.459329	87.0836	66.0254	30.3274	-9.67263	409.609	413.545	1.00961	0.733157
12	0.2966	134.862	41.4736	12.3011	-27.6989	268.647	318.166	1.18433	0.464487
13	0.371041	107.805	50.9142	18.8913	-21.1087	331.36	355.229	1.07203	0.0040433
14	0.677094	59.076	137.017	92.7736	52.7736	623.127	680.034	1.09133	0.779068
15	0.331295	120.738	45.5639	15.0951	-24.9049	297.536	334.303	1.12357	0.62227
16	0.749619	53.3604	180.459	135.275	95.2753	700.94	839.916	1.19827	0.90876
17	0.310138	128.975	43.0108	13.3393	-26.6607	279.854	324.246	1.15863	0.717751
18	0.472823	84.5983	68.8177	32.5386	-7.46144	421.973	424.225	1.00534	0.457624
19	0.344893	115.978	47.3091	16.3166	-23.6834	309.017	341.149	1.10398	0.884902
20	0.345	115.942	47.323	16.3264	-23.6736	309.108	341.204	1.10384	0.430179
21	0.533137	75.0277	83.2999	44.4102	4.41021	478.611	479.266	1.00137	0.375122
22	0.504279	79.3212	75.9334	38.2916	-1.7084	451.229	451.337	1.00024	0.49331
23	0.322757	123.932	44.5106	14.3661	-25.6339	290.375	330.16	1.13701	0.41642
24	0.568487	70.3622	93.5977	53.2091	13.2091	512.871	518.11	1.01022	0.454946
25	0.553567	72.2587	89.0662	49.3041	9.30409	498.313	501.043	1.00548	0.327935

Clearly every market participant is better off ex-post than they were ex-ante- an essential result, as market participants who are made worse off by trade would simply not trade in the first place. The participants with γ furthest away from the sample median were made best-off as a result of trade. This makes logical sense, as their optimal quantities of production will be farthest away from their PTQs, meaning they have the most to gain by engaging in trade. Furthermore, those with small gammas received the lowest profits.

This method of permit allocation is far more preferable than other methods, for example, looking at the minimum price that firms are willing to transact at by equating pre and post trade profits. This is because it allows for firms to sell only some of their permits and produce using the rest, which is far more realistic than a simple 'all-or-nothing' approach to selling permits. However, there isn't any matching of buyers and sellers, simply a price calculated which when re-inputted into each firm's profit function yields optimal production points that coalesce to a total permit requirement that is equal to the cap. This isn't the way that markets work practically, and is definitely a shortcoming of the model.

2.5: 'Solve' Command
Unfortunately, Mathematica's "Solve" and "NSolve" command aren't capable of solving our supply = demand equation, so we needed to find a way to numerically solve for p. We did this by incrementally iterating through prices and calculating supply and demand value at each of these prices. Once supply exceeded demand, we knew we'd passed the equilibrium, and so linear interpolated between the points before and after the equilibrium to give a good estimation of the true equilibrium price.

2.6: Multiple Time Periods and Firms' Decision Structure
The number of time periods parameterised represents the amount of time before the cutoff. When specifying the number of time periods, one also needs to decide on starting permits and ending permits. This will then automatically calculate the amount that the permit supply will reduce by each period.

At every time period, we give each firm a choice. The firm can either invest in decarbonising, or can invest in their existing production techniques. If a firm invests in their production technique, they increase their accumulated investment parameter, λ, by a parameterised amount (usually 10), increasing the coefficient on their profit function. This is supposed to be interpreted as reducing the variable costs in production. As the investment parameter increases linearly, each subsequent round of investment will have a diminishing impact on profitability relative to the previous increase, which is also realistic. If a firm instead chooses to decarbonise, they reduce their γ by a parameterised amount, usually 0.1. By doing this, the firm also resets their accumulated investment to 0. This is rationalised by the notion that investment and research into a previous technology is irrelevant and unhelpful if firms then choose to produce using a different technology altogether. Negligible research into old-fashioned internal combustion engines would improve the efficiency of the working of lithium-ion EV drive-trains.

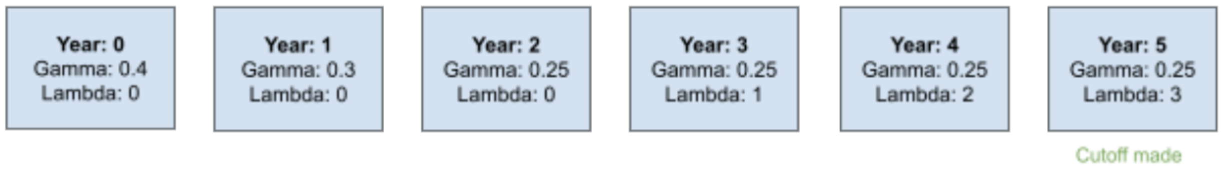
2.7: Cost-Benefit Analysis: Forecast Period
In order to make a decision on what to do at each time period, firms employ CBA across decision-space, and choose the action with the highest NPV. By enforcing accumulated investment to reset when firms first decarbonise, we are also significantly reducing the search space of actions. This is because once a firm begins to decarbonise, it is *never* optimal for them to stop and build up accumulated investment before they reach the cutoff point. This is because the opportunity cost of investing after beginning the decarbonising process (but before reaching the cutoff) will be investing at a higher γ value (earlier on), which will yield a higher profit return, both because it is performed at a higher γ, and also because it is performed closer to the present, meaning it is discounted less.

To calculate the NPV, we simply iterate through each remaining time period, and calculate the profit made in each period. We calculate the profit in each hypothetical scenario where the decarbonisation process is started in every time period, decreasing γ by 0.1 (or whatever the chosen value is) until the cutoff is reached. Once the cutoff is reached, the firm simply invests in accumulating λ. The firm will not reduce their γ below the cutoff, as it is irrational as it will earn them less profits.

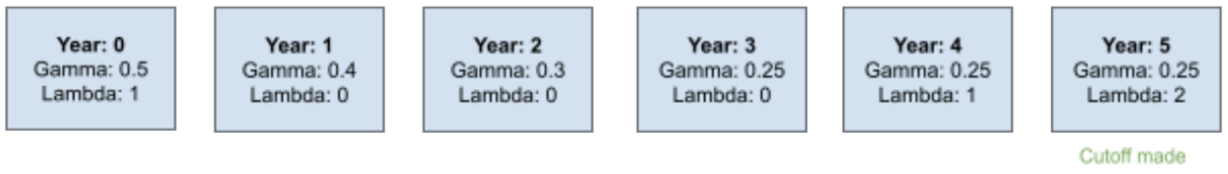
A visualisation of the scenarios a given firm will consider every time period:

Starting $\gamma = 0.5$
Number of years to cutoff = 5
Cutoff = 0.25

Begin decarbonisation in: Year 0 (present)



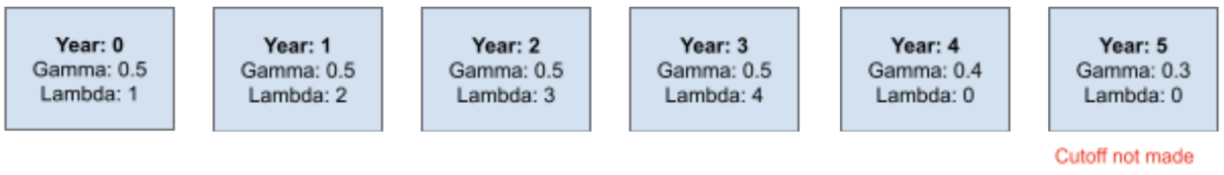
Begin decarbonisation in: year 1



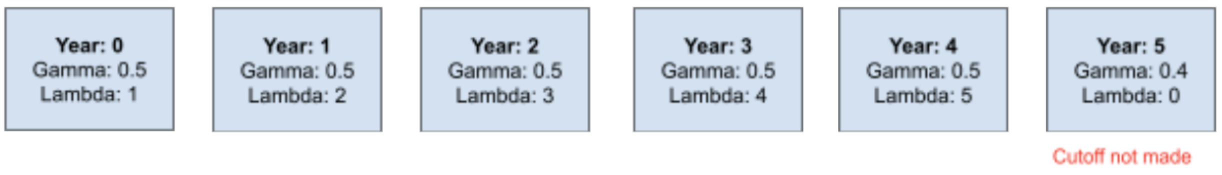
.

.

Begin decarbonisation in: Year 4



Begin decarbonisation in: Year 5



Expected profits are calculated for each of these parameter values at each time period and discounted back to the present. The firms are assumed to know each others' γ values at present time (equivalent to information disclosure of plc's). This allows them to forecast expected permit prices for every future period, where the linear eduction of permit supply is known. This means optimal consumption points are easily solvable and thus expected profits also easily solvable.

2.8: Cost-Benefit Analysis: Post-Cutoff Period
If a firm successfully makes it past the cutoff period, we calculate a terminal value expected profit. This is done by assuming an expected growth rate of profits post-cutoff which is applied to the final profit made during the forecast period. The formula for the terminal value is:

$$\frac{E[\pi_T](1+g)}{\rho-g}$$

which is simply the value of the final forecasted profit growing at rate g for infinite time periods discounted at rate ρ. This is appended to the forecast period *only* if the firm has a gamma below/at the cutoff point. For the derivation of the formula, see appendix 2. In finance, DCF valuation methods that employ terminal values are criticised for apportioning the majority of a firm’s value to this terminal value, which relies on assuming that firms exist for an infinite amount of time into the future, which is unrealistic. However since we aren’t valuing companies, but modelling a decision-making process well, whereby no executives make decisions on the basis that their business will fail at a specified point in time in the future, we believe terminal value’s inclusion is justified.

2.9: Cost-Benefit Analysis: Discount Rates
The rate with which future cash flows are discounted to present-day is of central importance in our model, and is what we are using to find an answer to our question, which is how present bias affects the decision-making process for the firms in our model. We test two related discounting methods:

Exponential Discounting (ED):
Proposed by Samuelson (1937), all intertemporal considerations are boiled into one parameter:

$$NPV = \sum_{t=0}^{T} \delta^t \pi_t$$

$$\text{where } \delta = \frac{1}{1+\rho}$$

Quasi-Hyperbolic Discounting (QHD):
Proposed by Laibson (1997), any future cash flow is discounted using an exponentially discounted factor consistent with our ED scheme above multiplied by another factor, β, which varies between 0 and 1. The discount coefficient is as follows:

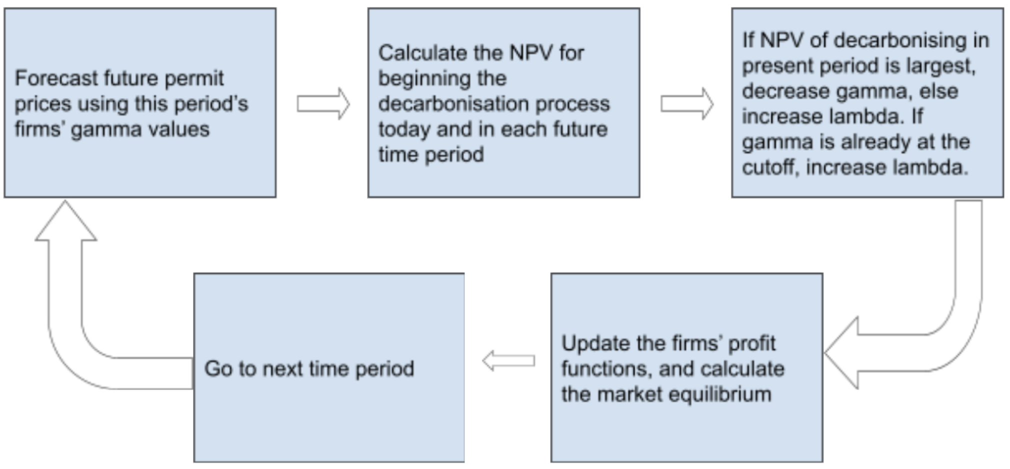
$$D(\tau) = \begin{cases} 1 & if\,\tau = 0 \\ \beta\delta^t & if\,\tau > 0 \end{cases}$$

$$0 \leq \beta \leq 1$$

Where τ represents the number of time periods in the future.

2.10: Cost-Benefit Analysis: Decision Making
Now, each firm is capable of forecasting each scenario: working out the expected profits of each scenario in each time period, discounting those profits to a present value and summing them together to form an NPV of each possible decision. The firms' decision rule is simply to look at the array of NPVs, and if the NPV of decarbonising *today* is highest, then they begin the decarbonisation process.

2.11: Process
Drawing together what we have already explained, the process that firms go through is as following:


Forecast future permit prices using this period's firms' gamma values
Calculate the NPV for beginning the decarbonisation process today and in each future time period
If NPV of decarbonising in present period is largest, decrease gamma, else increase lambda. If gamma is already at the cutoff, increase lambda.
Update the firms' profit functions, and calculate the market equilibrium
Go to next time period

2.12: Results

We first looked at the effect of present bias where all firms have the same degree of present bias. By maintaining the same list of firms and only changing their β values, it allows us to examine the *ceteris paribus* effect of present bias on decarbonisation schedules.

Due to the nature of the model (and in the real world), firms are interdependent in their decisions to decarbonise. If a sample is populated by a disproportionately large number of high-γ firms, there is a larger payoff to decarbonising, as market demand will be higher, causing a higher permit price. The opposite is also true of a disproportionately large number of low-γ firms. Because of this, we also randomly distributed beta across samples of 25 firms each. By aggregating data across several samples, we should be able to randomise out any unrepresentative samples. As such, we generated 20 samples of 25 firms. This is also more representative of the real world, where present bias won't be identical across all firms in an industry.

It is worth noting that pollution levels will unambiguously be at whatever the permit supply is for any given time period. Instead, we are looking at the economic effects of an environmental policy.

In our environment, an adverse event will be a dynamic preference reversal leading to a firm not reaching the cutoff point and being ejected from the market. The qualitative rationale is that it would cause job losses and consumers would suffer in the preventable removal of choice of goods.

Standardised parameter values
Across our samples, we used the following parameter values:

$$\gamma \sim N(0.5,\, 0.15)$$

$$\beta \sim U(0,1)$$

$$\rho = 0.125$$

$$\text{Lamdba step per unit of investment } = 10$$

$$\text{Decrease in Gamma per step} = 0.1$$

$$\text{Initial permit supply } = 1000$$

$$\text{Final permit supply} = 250$$

$$\text{Number of time periods} = 10$$

$$\text{Cutoff Gamma} = 0.15$$

$$g = 0.03$$

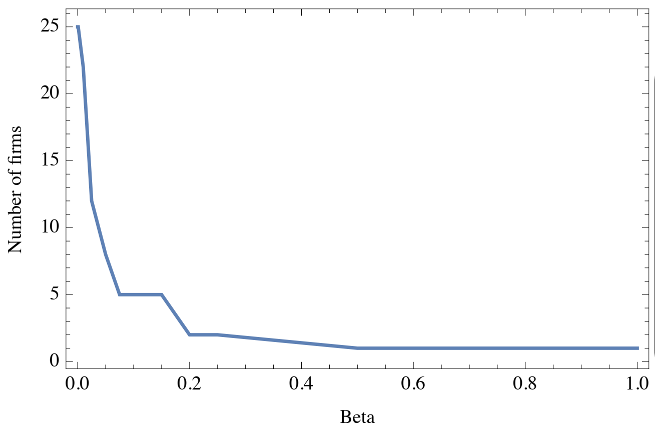
We chose our ρ discount rate to be similar to a weighted average cost of capital for the car industry (PwC, 2020). This is because in finance, the WACC is typically used as the discount factor for which valuation and forward-looking decision-making is done upon. A growth rate of 3% is chosen for the same reason.

Did firms decarbonise?

In the vast majority of cases, they do. From our 500-firm sample, 448 firms made the cutoff point.

Does present bias cause dynamic preference reversals?

We found present bias **did not** have a significant effect on the decision of when to decarbonise. Plotting the number of firms that do not make the cutoff γ value against different values of β for a given, randomly chosen 25-firm sample:



Beta	⌗ firms missing cutoff
1	1
0.5	1
0.25	2
0.2	2
0.15	5
0.1	5
0.075	5
0.05	8
0.025	12
0.01	22
0.001	25
0	25

Figure 4

By solely varying β, we can see at what values of β that firms begin reversing their preferences dynamically. At $\beta = 1$, this is the equivalent of firms employing EDU, a time-consistent preference relationship, where only one firm decides to not decarbonise. However, as we start to vary β, it is only once it takes very small values (corresponding to extremely high present bias) that preferences begin to dynamically reverse across large numbers of the firms. For context, a β value of 0.05 means agents prefer utility/profits today **20x more** than they do in the future once exponential discounting is accounted for, and it is around this region where firms properly start to decarbonise.

This is also reflected in our large-sample aggregated data with heterogeneous beta values. Plotting beta and gamma:

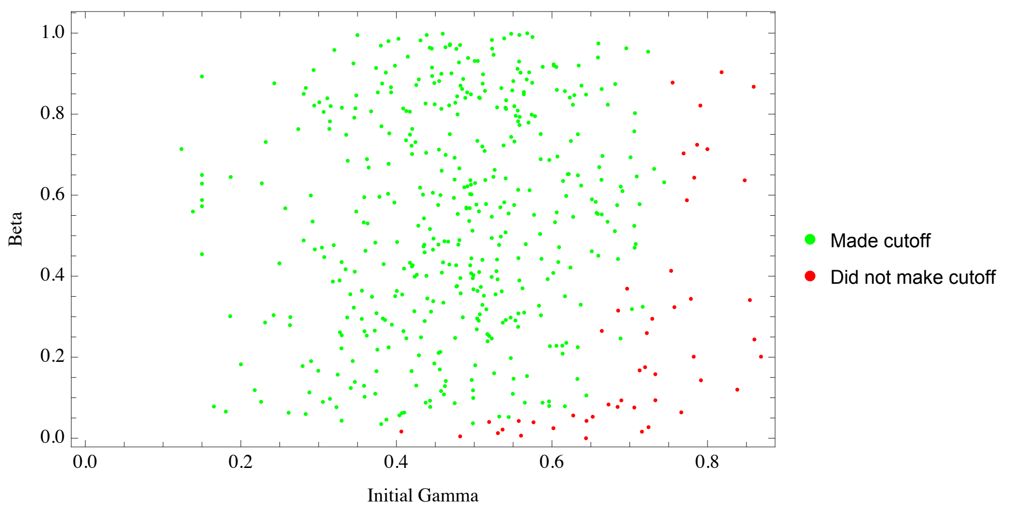


Figure 5

We can see how it is only at very low β values that firms who might have previously made the cutoff do not make the cutoff. This also provides an intuitive sense insofar as it is the high-gamma, low-beta firms who tend not to make the cutoff. This would be because they might be far away from the cutoff point to begin with, and the high profits they make at this high gamma simply don't make it worthwhile to decarbonise. This might be equivalent, for example, to a Mustang. The amount of investment they need to make in decarbonising and the opportunity cost of doing so simply doesn't make it worthwhile, even with the knowledge of an impending cutoff.

We found that the reason for this was the significant profits that were attributed to the terminal value. Firms realised that missing the cutoff point was extremely costly, and even though they might be present biased, the magnitude of the prospective cost in the future versus the magnitude of the potential benefit today massively outweighed most firms' present bias.

This is visualised well by a sample of 25 firms' γ evolutions. It is 12 periods long in this case because the first period is initial, and the second is the present:

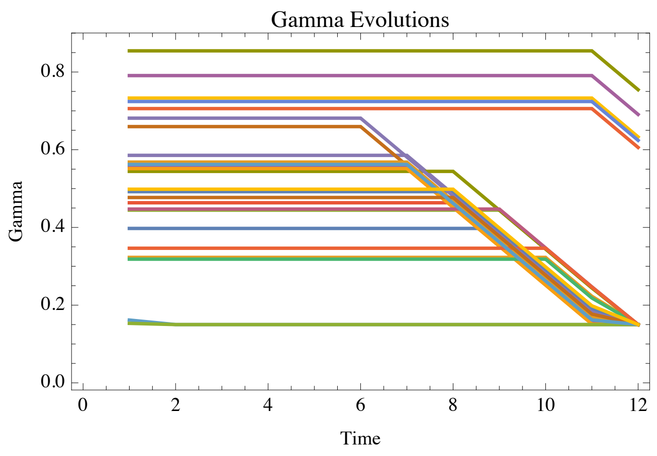


Figure 6

The vast majority of firms time their decarbonisation so that they make it just in time for the cutoff; they are only incentivised to begin decarbonisation when the threat of missing the cutoff is present. It is also worth noting that a mechanical feature of the code is that all firms γ invest in the last period, meaning the reduction in γ from the high-γ firms isn't anything of note.

3. Discussion & Conclusions

3.1: Findings

In most cases, the impact of present bias is fairly predictable. Children eat one marshmallow, adults delay Christmas shopping and ECON0052 students leave their project until the last minute. Prior to generating results from our various scenarios, we assumed the same would be true for present-biased firms in a permit-market environment. The tendency of green investment to involve delayed profits and high up-front investment fits the stereotypical scenario where present-biased agents tend to push back such costs and bring forward benefits. Combine this with a regulatory environment criticised for not incentivising firms to develop cleaner technology (Hanley, 2019), we felt confident that firms would miss the cutoff once pressure from present bias began to build. This said, we did also caveat with a suggestion that a firmly implemented cutoff date would likely have an incentivising effect on firms to decarbonise.

As expected, we did see present bias adversely affect firms. This was reflected both when holding all else equal (fig. 5) and also in aggregated random samples (fig. 6). What was notable, however, was how robust firms' decarbonisation decisions were to the present bias. This was because of the very large value of the terminal cash flow, which outweighed present bias in the majority of cases. It was only once non-present cash flows were discounted by an additional factor of 13x-15x did substantial numbers of preferences begin to reverse. Even though present bias is definitely prevalent in the shareholder value maximisation ethos that has recently afflicted the corporate world, it is a stretch to suggest that firms will frequently exhibit present bias of this magnitude.

Our analysis also provided validation for the criticism of permit markets' inability to incentivise investment into decarbonising technologies. As seen in fig. 7, in the vast majority of cases, decarbonisation happens as a response to the impending cutoff, seen by a tendency to make it 'just-in-time'.

3.2: Strengths and Weaknesses of Model/Approach
In analysing the validity of our model's conclusions, we must critically assess the potential shortcomings in the structure of our model. It is worth starting off by assuming our model is accurate, before exploring the sign, magnitude and provenance of biases coming from the model's imperfect representation of the real world.

Potentially inaccurate price forecasts
The model hinges on firms making decisions regarding expectations of future profits. A key process in this forecasting necessity of firms is the price they expect permits to sell at in the future. If they predict permit prices to be higher in the future, they will be more incentivised to begin decarbonisation. In our model, firms make their price forecasts based on what the permit market equilibrium will clear at with the present γ values of firms under decreasing permit supply. The inability of firms to forecast other firms' behaviour means they underestimate the price that they think permits will sell at in the future. This leads to firms being more inclined to resist decarbonising, which suggests even fewer firms wouldn't decarbonise than is indicated in our model.

Demand for cars assumed fixed
In our model, we assume that demand for cars is fixed across any value of γ, and that it is unchanged. What might be more realistic is to increase the profit margins on lower values of γ as time progresses. This would mimic the broad shift observed in the real world of a preference for electric vehicles. More generally, it would make sense for pollution-efficient firms to have a profit margin boost accounted for by end-consumers' demand for pollution-efficient firms. This means that again, firms are biased against decarbonising, as they don't recognise the potential demand-side benefits of a natural structural break in consumer preferences for pollution-efficiency.

Firms assumed to be the same size
In our model, all firms are equally sized. This means they all receive the same share of pollution permits, and also ignores the possibility of retaining profits. What happens in reality is larger firms are apportioned more permits, meaning they are able to produce more and earn more profits. Furthermore, the larger, more profitable firms with poor pollution efficiency will have larger reserves of profits, meaning they would be able to invest in reducing their γ value at a faster rate or increasing their lambda value at a higher rate. Missing this functionality means the profitability of being a larger, more pollution-inefficient firm might be understated.

No randomness
Arguably the biggest difference between the real world and our model is the deterministic manner in which the environment evolves. This is in stark contrast to the real world, which is constantly rocked by random events.

Hidden Bugs
While every effort has gone into eliminating inadvertent coding errors, it is a possibility worth explicitly stating that bugs might still exist in our model. This shouldn't have large consequences, as in all cases our results were practically explainable in the context of the input parameters. However we still think it is important and feel it obligatory to explicitly mention this possibility.

3.3: Further Investigation and Policy Implications
An important aspect we didn't get around to modelling was the notion of present-biased governments not following through with the original planned cutoff date. We would expect this to have significant effects on the number of firms who make it past the cutoff. One could model firms' expectations of the cutoff date as an asymmetric distribution whose probability density is maximised at the original cutoff date, before decreasing as the date becomes further into the future. Firms would then calculate NPVs with reference to their expected cutoff date.

There were also several practical policy takeaways that are made from our results. Firstly, by combining a permit scheme with a cutoff point, we present a potential policy combination where each constituent compliments the other's weakness. Permit markets are denounced for their lack-of incentives to develop decarbonising technologies. Cutoff points often create large, unprepared-for shocks. However we have found that by combining the two, we have created an environment in which pollution emission levels reduce in a predictable, controlled way, and investment in green technology is incentivised and effectuated. This, however, is contingent on preventing market failures within the permit market. Our results saw cases of thin markets (Noll, 1982), with monopsonies developing, particularly in the latter time periods. While in our model this isn't consequential (as there is no bargaining power functionality), it is likely to be in real-world markets. The good news is that the monopsony buyers are almost invariably firms who aren't destined to make it past the cutoff, meaning potentially problematic thin markets have expiry dates.

Secondly, this study highlighted the proposed policy-mix's robustness to present bias. It is widely accepted that present bias is abundant in the world. Instead of designing policy that conveniently ignores it to preserve theoretical elegance, we should instead focus on implementing policy that is resistant to its effects.

4. Appendix

Appendix 1:

Consider a marketplace composed of two firms, A and B such that Gamma A < Gamma B. The optimisation problem which yields profit-maximising quantities is as such:

$$\partial_c \pi_{post-trade}(c,\ \gamma,\ \lambda,\ p,\ m,\ n) = \gamma(100\ + \lambda)\frac{c^{\frac{1}{1+e^{(0.5-\gamma)}}-1}}{1+e^{(0.5-\gamma)}} - p\gamma$$

Optimising:

$$(100\ + \lambda)\frac{c^{\frac{1}{1+e^{(0.5-\gamma)}}-1}}{1+e^{(0.5-\gamma)}} = p$$

$$'c = \left(\frac{\frac{\lambda}{e^{0.5\ -\gamma}+1.} + \frac{100.}{e^{0.5\ -\gamma}+1.}}{p}\right)^{\frac{1}{1.-\frac{1.}{e^{0.5\ -\gamma}+1.}}}$$

Which is increasing in gamma.

For the hypothetical post-trade profit function without the gamma product on the coefficient in front of the pre-trade segment the same is done:

$$\partial_c \pi_{post-trade}(c,\ \gamma,\ \lambda,\ p,\ m,\ n) = (100\ + \lambda)\frac{c^{\frac{1}{1+e^{(0.5-\gamma)}}-1}}{1+e^{(0.5-\gamma)}} - p\gamma$$

,

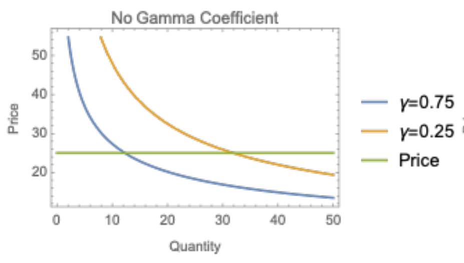
$$\frac{(100\ + \lambda)\frac{c^{\frac{1}{1+e^{(0.5-\gamma)}}-1}}{1+e^{(0.5-\gamma)}}}{\gamma} = p$$

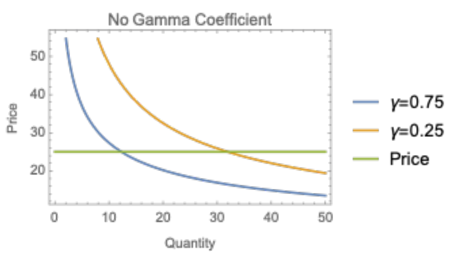
,

$$c = \left(\frac{\frac{\lambda}{E^{0.5\ -\gamma}+1.} + \frac{100.}{E^{0.5\ -\gamma}+1.}}{\gamma \mathrm{p}}\right)^{\frac{1}{1.-\frac{1.}{E^{0.5\ -\gamma}+1.}}}$$

Which is decreasing in gamma.

To visualise, we can plot this optimisation problem for the firms for both profit functions, where Gamma A=0.25, Gamma B = 0.75 and for a given price = 25:





We can clearly see the lower gamma having a higher optimal point of production, which is nonsensical in almost all practical cases. As a result, using a gamma coefficient ensures the optimal quantity of production is decreasing in gamma, which is what we want to reflect.

Appendix 2:

The sum to infinity of cash flows at time T growing at rate g discounted by rate Rho:

$$\frac{\pi_T \times (1+g)}{(1+\rho)} + \frac{\pi_{T+1} \times (1+g)}{(1+\rho)^2} + \frac{\pi_{T+2} \times (1+g)}{(1+\rho)^3} + \ldots$$

$$= \frac{\pi_T \times (1+g)}{(1+\rho)} + \frac{\pi_T \times (1+g)^2}{(1+\rho)^2} + \frac{\pi_T \times (1+g)^3}{(1+\rho)^3} + \ldots$$

$$= \pi_T \left(\frac{(1+g)}{(1+\rho)} + \frac{(1+g)^2}{(1+\rho)^2} + \frac{(1+g)^3}{(1+\rho)^3} + \ldots \right)$$

$$= \sum_{t=1}^{\infty} \pi_T \left(\frac{1+g}{1+\rho} \right)^t$$

$$= \sum_{t=0}^{\infty} \pi_T \left(\frac{1+g}{1+\rho} \right) \times \left(\frac{1+g}{1+\rho} \right)^t$$

This is a geometric sum, with a known formula for its sum to infinity.

$$\text{Sum of geometric sequence : } \sum_{0}^{\infty} ar^k = \frac{a}{1-r}$$

$$\text{Where } r < |1|$$

$$\therefore \frac{1+g}{1+\rho} < |1| \implies g < \rho$$

Our previous formula for the sum follows:

$$\sum_{t=0}^{\infty} \pi_T \left(\frac{1+g}{1+\rho} \right) \times \left(\frac{1+g}{1+\rho} \right)^t = \frac{\pi_T \left(\frac{1+g}{1+\rho} \right)}{1 - \left(\frac{1+g}{1+\rho} \right)}$$

$$= \frac{\pi_T \times (1+g)}{\rho - g}$$

5. Literature used

Department for Transport, 2020. *Government takes historic step towards net-zero with end of sale of new petrol and diesel cars by 2030*. [online] GOV.UK. Available from: https://www.gov.uk/government/news/government-takes-historic-step-towards-net-zero-with-end-of-sale-of-new-petrol-

and-diesel-cars-by-2030 [Accessed 12 April, 2021].

Noll, R. (1982). *Implementing Marketable Emissions Permits* [online]. The American Economic Review, 72(2), 120-124. Available from: http://www.jstor.org/stable/1802316 [Accessed 12 April, 2021]

Hanley, N., Shogren, J., & White, B. (2019). *Introduction to environmental economics (3rd ed.)* [textbook]. OUP Oxford. (33 – 38)

European Commission. 2020. *Report on the functioning of the European carbon market*. [online]. Available from: https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52020DC0740&from=EN [Accessed 12 April, 2021]

Jenkins, G., Lamech R. (1992). *Market Based Incentive Instruments for Pollution Control. Development Discussion* Papers 1992-02, JDI Executive Programs [online].
Available from: https://cri-world.com/publications/qed_dp_99.pdf [Accessed 12 April, 2021]

Laing, T. Sato, M., Grubb, M., Comberti, C. (2013). A*ssessing the effectiveness of the EU Emissions Trading System. Centre for Climate Change Economics and Policy* [online].
Available from: https://www.lse.ac.uk/granthaminstitute/wp-content/uploads/2014/02/WP106-effectiveness-eu-emissions-trading-system.pdf [Accessed 12 April, 2021]

Clot, S. and Stanton, 2014. *Present Bias in Payments for Ecosystem Services: Insights from a Behavioural Experiment in Uganda*. [online] Ideas.repec.org. Available at: <https://ideas.repec.org/p/lam/wpaper/14-03.html> [Accessed 12 April, 2021].

Duquette, E., Higgins, N. and Horowitz, J., 2011. *Farmer Discount Rates: Experimental Evidence*. American Journal of Agricultural Economics, 94(2), pp.451-456.

Gurguc, Z., 2020. *4.1 Intertemporal Choice - Standard Models. ECON0040*. UCL Department of Economics, Lecture 4.

Laibson, D., 1997. *Golden Eggs and Hyperbolic Discounting*. The Quarterly Journal of Economics, 112(2), pp.443-478.

O'Donoghue, T. and Rabin, M., 1999. *Doing It Now or Later*. American Economic Review, 89(1), pp.103-124.

Phelps, E. and Pollak, R., 1968. *On Second-Best National Saving and Game-Equilibrium Growth*. The Review of Economic Studies, 35(2), p.185.

Read, D. and van Leeuwen, B., 1998. *Predicting Hunger: The Effects of Appetite and Delay on Choice*. Organizational Behavior and Human Decision Processes, 76(2), pp.189-205.

Thaler, R., 1981. *Some empirical evidence on dynamic inconsistency*. Economics Letters, 8(3), pp.201-207.

Benhabib, J., Bisin, A. and Schotter, A. 2010. *Present-bias, quasi-hyperbolic discounting, and fixed costs*. Games and Economic Behavior, 69(2), pp.205-223.

Samuelson, P., 1937. *A Note on Measurement of Utility*. The Review of Economic Studies, 4(2), p.155.

Brewdog, 2020, *MAKE EARTH GREAT AGAIN: BREWDOG'S PUNKED-UP PLAN*, Available from: https://d1fnkk8n0t8a0e.cloudfront.net/docs/-Make-Earth-Great-Again_4.pdf [Accessed 12th April, 2021]

Tesla, 2020, *Tesla Annual Report 2019,* Available from: https://tesla-cdn.thron.com/static/9QDEHY_2019-tesla-inc-press-release-annual-report_JGVR0P.pdf [Accessed: 12th April, 2021]

PwC. Cost of Capital Report. https://www.pwc.co.nz/pdfs/2019pdfs/cost-of-capital-report-1.pdf. [Accessed: 12th April, 2021]

6. Code

Figure Commands

```
In[675]:= (*Figure 1*)
          a = Profit[c, 0.25, 100];
          b = Profit[c, 0.5, 100];
          d = Profit[c, 0.75, 100];
          Plot[{a, b, d}, {c, 0, 300}, Frame → True,
           FrameLabel → {"Units of Production", "Profit"},
           PlotLegends → {"γ=0.25", "γ=0.5", "γ=0.75"}]

In[679]:= (*Figure 2*)
          plot3dvar = Profit[c, 0.5, 100] + 0.5 Price (1000 / (25 × 0.5) - c);
          Plot3D[plot3dvar, {c, 0, 50}, {Price, 0, 50}, ColorFunction → "Rainbow",
           AxesLabel → Automatic, PlotLabel → "Total Profit"]

          (*Figure 3*)
          Print[ParametricPlot[{{TotalSupplyFunctionTest[p], p},
               {InitialDemandFunctionTest[p], p}}, {p, 0, 35}, PlotRange → Full,
             AspectRatio → 1 / 2.5, Frame → True, FrameLabel → {"Quantity of Permits", "Price"},
             PlotLegends → {"Supply", "Demand"}]];

In[681]:= (*Figure 4*)
          NumberOfFirmsWhoMissCutoff[input_, beta0_] := Module[{inp = input, beta = beta0},
            firmlist = inp[[6]][[11]][[2 ;; 26, 2]];
            InputArray = ConstantArray[0, 25];
            For[i = 1, i ≤ 25, i++, InputArray[[i]] = If[firmlist[[i]] == 0.15, 0, 1]];
            {beta, Total[InputArray]}
           ]
          Number1 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta1, 1];
          Number2 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta05, 0.5];
          Number3 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta025, 0.25];
          Number4 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta02, 0.2];
          Number5 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta015, 0.15];
          Number6 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta01, 0.1];
          Number7 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta0075, 0.075];
          Number8 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta005, 0.05];
          Number9 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta0025, 0.025];
          Number10 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta001, 0.01];
          Number11 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta0001, 0.001];
          Number12 = NumberOfFirmsWhoMissCutoff[TimeEvoHomogenousBeta0, 0];
          {{"Beta", "# firms missing cutoff"}, Number1, Number2, Number3, Number4,
            Number5, Number6, Number7, Number8, Number9, Number10, Number11, Number12} //
           MatrixForm
          ListPlot[{Number1, Number2, Number3, Number4, Number5, Number6, Number7,
            Number8, Number9, Number10, Number11, Number12}, PlotRange → Full,
           Joined → True, Frame → True, FrameLabel → {"Beta", "Number of firms"}]
```

```
In[696]:= (*Figure 5*)
          ListTester = ConstantArray[0, {500, 3}];
          For[j = 0, j ≤ 19, j++,
            DataCase1 = ToExpression["databetasample" <> ToString[j + 1]][[6]][[11]];
            DataCase2 = ToExpression["databetasample" <> ToString[j + 1]][[6]][[1]];
            For[i = 2, i ≤ 26, i++,
             ListTester[[(25 j) + i - 1, 1]] = DataCase2[[i, 2]];
             ListTester[[(25 j) + i - 1, 2]] = DataCase1[[i, 10]];
             ListTester[[(25 j) + i - 1, 3]] = If[DataCase1[[i, 2]] == 0.15, 0, 1];
            ];
           ];
          InterData = SortBy[ListTester, Last];
          DidntDecarbonise = ConstantArray[0, {48, 2}];
          DidDecarbonise = ConstantArray[0, {452, 2}];
          For[i = 1, i ≤ 452, i++,
            DidDecarbonise[[i, 1]] = InterData[[i, 1]];
            DidDecarbonise[[i, 2]] = InterData[[i, 2]];
           ];
          For[i = 1, i ≤ 48, i++,
            DidntDecarbonise[[i, 1]] = InterData[[i + 452, 1]];
            DidntDecarbonise[[i, 2]] = InterData[[i + 452, 2]];
           ];
          ListPlot[{DidDecarbonise, DidntDecarbonise}, PlotStyle → {Green, Red},
           Frame → True, FrameLabel → {"Initial Gamma", "Beta"},
           PlotLegends → {"Made cutoff", "Did not make cutoff"}]

In[704]:= (*Figure 6*)
          betasample1[[All, 2]] // MatrixForm
          databetasample1[[1]] // MatrixForm
          GammaEvolutionData = ConstantArray[0, {25, 12}];
          GammaEvolutionData[[All, 1]] = betasample1[[All, 2]];
          GammaEvolutionData[[All, 2 ;; 12]] = databetasample1[[1]];
          GammaEvolutionData
          ListPlot[GammaEvolutionData, Joined → True, Frame -> True,
           FrameLabel → {"Time", "Gamma"}, PlotLabel → "Gamma Evolutions"]
```

Model

```
In[579]:= Profit[c_, γ_, Co_] := Co γ (c^(1/(1+e^-(γ-0.5))))

In[580]:= PreTradeQ[m_, n_, gamma_] := m / (n gamma)
```

In[581]:=
```
(*Our function to generate a list of firms takes the number of firms,
the details of the gamma distribution and the initial profit coefficient as parameters
*)
FirmList[n0_, GammaMean0_, GammaSD0_ , ProfitCoefficient0_] :=
 Module[{n = n0, GammaMean = GammaMean0, GammaSD = GammaSD0,
   ProfitCoefficient = ProfitCoefficient0},
  (*Generate an empty list, which we later fill in with the relevant information*)
  list = ConstantArray[0, {n, 4}];
  (*Create a For loop to iterate over the empty array we want to fill in*)
  For[i = 1, i ≤ n, i++,
   list[[i, 1]] = i;
   list[[i, 2]] =
    RandomVariate[TruncatedDistribution[{0, 1000},
      NormalDistribution[GammaMean, GammaSD]]];
   (*We list each firms' profit functions within the firm list,
   as this allows easy updating with the relevant accumulated investment amount*)
   list[[i, 3]] = Profit[c, list[[i, 2]], ProfitCoefficient];
   (*Initialise accumulated investment to 0*)
   list[[i, 4]] = 0
  ];
  list
 ]
```

In[582]:=
```
(*Exactly the same as before,
except with an additional column which includes each firms' beta value*)
FirmListWithBetas[n0_, GammaMean0_, GammaSD0_ , ProfitCoefficient0_] :=
 Module[{n = n0, GammaMean = GammaMean0, GammaSD = GammaSD0,
   ProfitCoefficient = ProfitCoefficient0},
  list = ConstantArray[0, {n, 5}];
  For[i = 1, i ≤ n, i++,
   list[[i, 1]] = i;
   list[[i, 2]] =
    RandomVariate[TruncatedDistribution[{0, 1000},
      NormalDistribution[GammaMean, GammaSD]]];
   list[[i, 3]] = Profit[c, list[[i, 2]], ProfitCoefficient];
   list[[i, 4]] = 0;
   list[[i, 5]] = RandomVariate[UniformDistribution[]];
  ];
  list
 ]
```

```
In[583]:= Supply[firmlist0_, m0_] := Module[{firmlist = firmlist0, m = m0},
            (*For each firm in the sample,
            calculating the profit derivatives for us to subsequently solve for optimal
             quantities*)
            DerivativeList = Table[∂c firmlist[[i, 3]], {i, 1, Length[firmlist]}];
            (*Individual supplies all calculated in one line to minimise the number
              of tables that need to be created- when combined with inner-functions,
            this causes the overall 'Demand' function to run very slowly. Use of Max
              functions crucial in ensuring we get non-negative values*)
            IndividualSupply =
             Function[p,
              Table[firmlist[[i, 2]]
                 Max[m / (Length[firmlist] * firmlist[[i, 2]]) -
                   Solve[DerivativeList[[i]] == p firmlist[[i, 2]], c][[1, 1, 2]], 0],
                {i, 1, Length[firmlist]}]];
            (*Total supply is simply given as a sum of all the individual supply functions*)
            TotalSupply = Function[p, Total[IndividualSupply[p]]];
            {TotalSupply, IndividualSupply}
           ]

In[584]:= Demand[firmlist0_, m0_, initialPermits0_] :=
           Module[{firmlist = firmlist0, m = m0, init = initialPermits0},
            (*For each firm in the sample,
            calculating the profit derivatives for us to subsequently solve*)
            DerivativeList = Table[∂c firmlist[[i, 3]], {i, 1, Length[firmlist]}];
            (*Individual demands all calculated in one line to minimise the number of
              tables that need to be created- when combined with inner-functions,
            this causes the overall 'Demand' function to run very slowly*)
            IndividualDemand =
             Function[p,
              Table[firmlist[[i, 2]]
                 Max[Solve[DerivativeList[[i]] == p firmlist[[i, 2]], c][[1, 1, 2]] -
                   m / (Length[firmlist] * firmlist[[i, 2]]), 0], {i, 1, Length[firmlist]}]];
            (*Total demand is simply given as a sum of all the individual demand functions,
            and capped at the maximum number of permits in systemn to prevent plotting
             infinite values (as demand when p → 0 approaches infinite*)
            TotalDemand = Function[p, Min[Total[IndividualDemand[p]], m]];
            InitialTotalDemand = Function[p, Min[Total[IndividualDemand[p]], init]];
            {TotalDemand, IndividualDemand, InitialTotalDemand}
           ]
```

```
In[585]:= (*Function necessary to solve for the equilibrium permit price. For a given price,
          supply exceeds demand as soon as the price exceeds the equilibrium price. As such,
          we iterate through prices and check to see whether this is the case.*)
          GetSolution[supply0_, demand0_, MaxPrice0_, StepSize0_] :=
           Module[{supply = supply0, demand = demand0, MP = MaxPrice0, SS = StepSize0},
            ComparisonTable = ConstantArray[0, {Round[MP / SS], 2}];
            (*Iterate through prices, with each iteration being equal to a step size*)
            For[i = 1, i ≤ (MP / SS), i++,
             ComparisonTable[[i, 1]] = SS i;
             ComparisonTable[[i, 2]] = supply[SS i] - demand[SS i];
             (*If the difference between supply and demand is positive,
             take the previous price iteration,
             and linearly interpolate with the current price iteration to find an
              estimation of the equilibrium price. Once this is the case, break the For loop.*)
             If[ComparisonTable[[i, 2]] > 0,
              EquilibriumPrice =
               (SS Abs[ComparisonTable[[i - 1, 2]]]) /
                 (Abs[ComparisonTable[[i, 2]]] + Abs[ComparisonTable[[i - 1, 2]]]) +
                ComparisonTable[[i - 1, 1]];
              Break[], Null]
            ];
            EquilibriumPrice
           ]

In[ ]:=
In[587]:= (*Generates a table to look at the fate of individual firms in a given permit
            equilibrium
          *)
          ProfitSolver[Firms0_, PermitPrice_, m0_] :=
           Module[{firmlist = Firms0, price = PermitPrice, m = m0},
            SolutionList = ConstantArray[0, {Length[firmlist] + 1, 9}];
            SolutionList[[1, 1]] = "Firm #";
            SolutionList[[1, 2]] = "Pollution-eff";
            SolutionList[[1, 3]] = "Q(Pre-trade)";
            SolutionList[[1, 4]] = "Q(Post-trade)";
            SolutionList[[1, 5]] = "Total permits required";
            SolutionList[[1, 6]] = "Permits bought/sold";
            SolutionList[[1, 7]] = "Pre-trade profit";
            SolutionList[[1, 8]] = "Post-trade Profit";
            SolutionList[[1, 9]] = "Post / pre profit ratio";
            For[i = 2, i ≤ Length[firmlist] + 1, i++,
             SolutionList[[i, 1]] = i - 1;
             SolutionList[[i, 2]] = firmlist[[i - 1, 2]];
             SolutionList[[i, 3]] = PreTradeQ[m, Length[firmlist], firmlist[[i - 1, 2]]];
             SolutionList[[i, 4]] =
              Solve[
                ∂c (firmlist[[i - 1, 3]] +
                     ((price * firmlist[[i - 1, 2]])
                       ((PreTradeQ[m, Length[firmlist], firmlist[[i - 1, 2]]] - c)))) == 0, c][[
                1, 1, 2]];
             SolutionList[[i, 5]] = SolutionList[[i, 4]] × firmlist[[i - 1, 2]];
             SolutionList[[i, 6]] = SolutionList[[i, 5]] - (m / Length[firmlist]);
             SolutionList[[i, 7]] = firmlist[[i - 1, 3]] /. c → SolutionList[[i, 3]];
             SolutionList[[i, 8]] = (firmlist[[i - 1, 3]] /. c → SolutionList[[i, 4]]) +
               (price * firmlist[[i - 1, 2]]
                 (PreTradeQ[m, Length[firmlist], firmlist[[i - 1, 2]]] - SolutionList[[i, 4]]));
             SolutionList[[i, 9]] = SolutionList[[i, 8]] / SolutionList[[i, 7]];
            ];
            ;
            TP = Total[Table[SolutionList[[i, 5]], {i, 2, Length[firmlist] + 1}]];
            {SolutionList, TP}
           ]
```

```
In[588]:= (*Exactly the same as before, except now with a column for beta values included*)
          ProfitSolverWithBetas[Firms0_, PermitPrice_, m0_] :=
           Module[{firmlist = Firms0, price = PermitPrice, m = m0},
            SolutionList = ConstantArray[0, {Length[firmlist] + 1, 10}];
            SolutionList[[1, 1]] = "Firm #";
            SolutionList[[1, 2]] = "Pollution-eff";
            SolutionList[[1, 3]] = "Q(Pre-trade)";
            SolutionList[[1, 4]] = "Q(Post-trade)";
            SolutionList[[1, 5]] = "Total permits required";
            SolutionList[[1, 6]] = "Permits bought/sold";
            SolutionList[[1, 7]] = "Pre-trade profit";
            SolutionList[[1, 8]] = "Post-trade Profit";
            SolutionList[[1, 9]] = "Post / pre profit ratio";
            SolutionList[[1, 10]] = "Beta";
            For[i = 2, i ≤ Length[firmlist] + 1, i++,
             SolutionList[[i, 1]] = i - 1;
             SolutionList[[i, 2]] = firmlist[[i - 1, 2]];
             SolutionList[[i, 3]] = PreTradeQ[m, Length[firmlist], firmlist[[i - 1, 2]]];
             SolutionList[[i, 4]] =
              Solve[
                ∂c (firmlist[[i - 1, 3]] +
                    ((price * firmlist[[i - 1, 2]])
                      ( (PreTradeQ[ m, Length[firmlist], firmlist[[i - 1, 2]]] - c)))) == 0, c][[
               1, 1, 2]];
             SolutionList[[i, 5]] = SolutionList[[i, 4]] × firmlist[[i - 1, 2]];
             SolutionList[[i, 6]] = SolutionList[[i, 5]] - (m / Length[firmlist]);
             SolutionList[[i, 7]] = firmlist[[i - 1, 3]] /. c → SolutionList[[i, 3]];
             SolutionList[[i, 8]] = (firmlist[[i - 1, 3]] /. c → SolutionList[[i, 4]]) +
               (price * firmlist[[i - 1, 2]]
                 (PreTradeQ[ m, Length[firmlist], firmlist[[i - 1, 2]]] - SolutionList[[i, 4]]));
             SolutionList[[i, 9]] = SolutionList[[i, 8]] / SolutionList[[i, 7]];
             SolutionList[[i, 10]] = firmlist[[i - 1, 5]];
            ];
            ;
            TP = Total[Table[SolutionList[[i, 5]], {i, 2, Length[firmlist] + 1}]];
            {SolutionList, TP}
           ]
```

```
In[589]:= (*Command to run each of the previously defined functions in order,
          which allow us to generate equilibrium simulations with a single click*)
          GetPermitEquilibrium[m0_, initialPermits0_, MaxPrice0_, StepSize0_, firms0_] :=
           Module[{m = m0, InitialPermits = initialPermits0, MaxPrice = MaxPrice0,
             StepSize = StepSize0, inputfirms = firms0},
            (*First generate supply and demand functions*)
            SupplyOutput = Supply[inputfirms, m];
            TotalSupplyFunction = SupplyOutput[[1]];
            IndividualSupplyFunctions = SupplyOutput[[2]];
            DemandOutput = Demand[inputfirms, m, InitialPermits];
            TotalDemandFunction = DemandOutput[[1]];
            IndividualDemandFunctions = DemandOutput[[2]];
            InitialDemandFunction = DemandOutput[[3]];
            (*Now calculate the equilibrium price using the iterative solve function*)
            price = GetSolution[TotalSupplyFunction, TotalDemandFunction, 1000, 0.25];
            (*Input this price into the equilibrium solver to generate the equilibrium state*)
            ProfitOutput = ProfitSolver[inputfirms, price, m];
            FirmTable = ProfitOutput[[1]];
            (*Outputs the number of permits required by the firms' optimal production quantities,
            and if it equals the input permit supply,
            we know that the optimal values of quantities etc. are valid. If this doesn'
             t debugging commences!*)
            PermitTotal = ProfitOutput[[2]];
            (*Create a list of all the relevant outputs*)
            {FirmTable, PermitTotal, price, TotalSupplyFunction, TotalDemandFunction,
             InitialDemandFunction}
           ]

In[590]:= (*Same as above except with heterogenous beta functionality*)
          GetPermitEquilibriumWithBetas[m0_, initialPermits0_, MaxPrice0_, StepSize0_,
            firms0_] :=
           Module[{m = m0, InitialPermits = initialPermits0, MaxPrice = MaxPrice0,
             StepSize = StepSize0, inputfirms = firms0},
            SupplyOutput = Supply[inputfirms, m];
            TotalSupplyFunction = SupplyOutput[[1]];
            IndividualSupplyFunctions = SupplyOutput[[2]];
            DemandOutput = Demand[inputfirms, m, InitialPermits];
            TotalDemandFunction = DemandOutput[[1]];
            IndividualDemandFunctions = DemandOutput[[2]];
            InitialDemandFunction = DemandOutput[[3]];
            price = GetSolution[TotalSupplyFunction, TotalDemandFunction, 1000, 0.25];
            ProfitOutput = ProfitSolverWithBetas[inputfirms, price, m];
            FirmTable = ProfitOutput[[1]];
            PermitTotal = ProfitOutput[[2]];
            {FirmTable, PermitTotal, price, TotalSupplyFunction, TotalDemandFunction,
             InitialDemandFunction}
           ]

In[591]:= firms = FirmListWithBetas[25, 0.5, 0.15, 100];
```

```
In[592]:= (*Testing out the functions for equilibriums*)
          TestOutput = GetPermitEquilibriumWithBetas[1000, 1000, 300, 1, firms];
          FirmTableTest = TestOutput[[1]];
          TPTest = TestOutput[[2]];
          PriceTest = TestOutput[[3]];
          TotalSupplyFunctionTest = TestOutput[[4]];
          TotalDemandFunctionTest = TestOutput[[5]];
          InitialDemandFunctionTest = TestOutput[[6]];
          Print["Total permits in system= ", TPTest];
          Print["Price: ", PriceTest];
          Print[MatrixForm[FirmTableTest]];
```

```
Total permits in system= 999.285

Price: 6.04033
```

```
Firm # Pollution-eff Q(Pre-trade) Q(Post-trade) Total permits required Permits bought/sold Pre-trade profit Post-trade Profit Post / pre profit ratio    Beta
  1      0.45106       88.68        59.017           26.6202             -13.3798          402.087          410.473            1.02086           0.0422932
  2      0.386735      103.43       48.9384          18.9262             -21.0738          344.945          369.644            1.0716            0.681543
  3      0.448863      89.1141      58.631           26.3173             -13.6827          400.095          408.919            1.02205           0.416044
  4      0.482248      82.9448      64.8624          31.2798             -8.72022          430.676          433.937            1.00757           0.842861
  5      0.644443      62.0691      110.567          71.2539              31.2539          589.186          614.124            1.04233           0.00667527
  6      0.633912      63.1002      106.558          67.5487              27.5487          578.384          598.492            1.03476           0.906145
  7      0.498184      80.2917      68.1341          33.9433             -6.05672          445.512          447.014            1.00337           0.0548314
  8      0.606948      65.9035      97.0925          58.9301              18.9301          551.05           561.466            1.0189            0.653672
  9      0.49653       80.5591      67.7849          33.6572             -6.34277          443.965          445.621            1.00373           0.64665
 10      0.411776      97.1401      52.5776          21.6502             -18.3498          366.906          384.449            1.04781           0.284772
 11      0.396498      100.883      50.3175          19.9508             -20.0492          353.465          375.264            1.06167           0.215677
 12      0.389122      102.795      49.271           19.1725             -20.8275          347.024          371.001            1.06909           0.28212
 13      0.413187      96.8085      52.7927          21.8133             -18.1867          368.154          385.322            1.04663           0.945545
 14      0.296737      134.8        38.2612          11.3535             -28.6465          268.76           325.649            1.21167           0.17202
 15      0.579983      68.9676      88.6566          51.4193              11.4193          524.183          528.328            1.00791           0.850159
 16      0.384146      104.127      48.5809          18.6622             -21.3378          342.695          368.185            1.07438           0.493691
 17      0.651078      61.4366      113.189          73.6946              33.6946          596.027          624.336            1.0475            0.717557
 18      0.483499      82.7302      65.1119          31.4815             -8.51847          431.834          434.936            1.00718           0.857599
 19      0.596526      67.0549      93.7174          55.9049              15.9049          540.611          548.225            1.01408           0.734754
 20      0.173163      230.997      28.0422          4.85587             -35.1441          169.367          282.283            1.66669           0.96247
 21      0.528684      75.6595      75.0028          39.6528             -0.347175         474.352          474.356            1.00001           0.223537
 22      0.403317      99.1775      51.3105          20.6944             -19.3056          359.448          379.303            1.05524           0.508238
 23      0.658617      60.7334      116.263          76.5725              36.5725          603.836          636.295            1.05376           0.510282
 24      0.740243      54.0363      157.213          116.376              76.3755          690.694          794.436            1.1502            0.503673
 25      0.564535      70.8548      84.237           47.5547              7.5547           509.             510.907            1.00375           0.647982
```

```
In[602]:= (*Calculating expected permit price equilibriums in the future*)
          GetPermitPriceEvolution[time_, mZero_, mFinal0_, numberYears0_, Firms0_] :=
           Module[{T = time, m0 = mZero, mFinal = mFinal0, numberYears = numberYears0,
             firmlist = Firms0},
            MarketForecast = ConstantArray[0, {(numberYears - T) + 1, 3}];
            (*Initial permits need to be consistent with what time period it is. For example in t=
             0, m=1000, but in t=1, intitail m = 925*)
            InitialPermits = m0 - T ((m0 - mFinal) / numberYears);
            For[t = 1, t ≤ (numberYears - T) + 1, t++,
             MarketForecast[[t, 1]] = t - 1;
             MarketForecast[[t, 2]] = InitialPermits - ((m0 - mFinal) / (numberYears)) (t - 1);
             (*Permit equilibriums calculated for each of the remaining time periods,
             with the appropriate supply of permits,
             at the current level of gammas. Equivalent is changing the above example
              equilibrium's parameter value from 500 to whatever,
             and caluclating the resulting price for each of the remaining time periods*)
             MarketForecast[[t, 3]] =
              GetPermitEquilibriumWithBetas[MarketForecast[[t, 2]], m0, 100, 0.2, firmlist][[3]];
            ];
            MarketForecast
           ]

In[603]:= (*This is definitely the most complicated function in the overall model. Its
           objective is to output a list net present values for each scenario of
```

```
 decarbonising in each potential future time period. It then outputs the
 point in time where decarbonising gives the largest NPV,
as well as the list of NPVs coming from decarbonising in each remaining time
 period. It needs to be able to handle variable time remaining until cutoffs,
as we iterate through this function later at differen time periods*)
GetSingleFirmNPVList[PermitPriceForecasts_, firm0_, GammaStep0_, ProfitStep0_,
   InitialTimePeriod0_, RhoDiscountRate0_, BetaRate0_, PostCutoffGrowthRate_,
   CutoffGamma_, NumberYears0_] :=
  Module[{PriceForecastsFull = PermitPriceForecasts, firmlist = firm0,
     GammaStep = GammaStep0, ProfitStep = ProfitStep0, T = InitialTimePeriod0,
     RhoDiscountRate = RhoDiscountRate0, BetaRate = BetaRate0, g = PostCutoffGrowthRate,
     Cutoff = CutoffGamma, NumberYears = NumberYears0},
    (*Intialise key parameter values and information arrays*)
    DiscountCoefficient = 1/(1 + RhoDiscountRate);
    PriceForecasts = Transpose[PriceForecastsFull][[3]];
    PermitForecasts = Transpose[PriceForecastsFull][[2]];
    NPVList = ConstantArray[0, {Length[PriceForecasts], 2}];
    (*Begin iterating through each hypothetical start date for decarbonisation*)
    For[TimeWhenGammaInvest = 0, TimeWhenGammaInvest ≤ Length[PriceForecasts] - 1 ,
     TimeWhenGammaInvest++,
     GammaForecast = firmlist[[2]];
     YearlyProfitForecast = ConstantArray[0, {Length[PriceForecasts], 2}];
     AccumulatedInvestmentForecast = 0;
     (*Begin for loop that goes over each time evolution for one hypothetical
      decarbonisation date*)
     For[i = 0, i ≤ Length[PriceForecasts] - 1, i++,
      PermitsPerFirm = PermitForecasts[[i + 1]] / Length[firms];
      GammaNew = GammaForecast;
      (*Begins logic
       chain: if the time period is before the loop-specified decarbonisation start-date,
      increase accumulated investment. If its at or after the decarbonisation start,
      decrease gamma and set accumulated investment to 0. If gamma is already
       at the cutoff, increase accumulated investment.*)
      If[i < TimeWhenGammaInvest,
       AccumulatedInvestmentForecast = AccumulatedInvestmentForecast + 1;
       ,
       If[GammaForecast > Cutoff,
         AccumulatedInvestmentForecast = 0;
         GammaNew = GammaNew - GammaStep;
         If[GammaNew < Cutoff, GammaNew = Cutoff, Null];
         ,
         AccumulatedInvestmentForecast = AccumulatedInvestmentForecast + 1
        ];
      ];
      (*Update the hypothetical gamma value for this time period*)
      GammaForecast = GammaNew;
      (*Calculate the expected profit from having this gamma at this time
       period with this level of accumulated investment*)
      OptimalProduction =
        Solve[
          (∂c Profit[c, GammaForecast,
              100 + (AccumulatedInvestmentForecast * ProfitStep)]) ==
           PriceForecasts[[i + 1]] GammaForecast, c][[1, 1, 2]];
       YearlyProfitForecast[[i + 1, 1]] =
        Profit[OptimalProduction, GammaForecast,
          (100 + (AccumulatedInvestmentForecast * ProfitStep))] +
         PriceForecasts[[i + 1]] GammaForecast
          ((PermitsPerFirm / GammaForecast) - OptimalProduction);
       (*Discount the calculated profit from this time period*)
       YearlyProfitForecast[[i + 1, 2]] =
        YearlyProfitForecast[[i + 1, 1]] * If[i == 0, 1, BetaRate DiscountCoefficient^i];
       (*If the gamma is at or below the cutoff, calculate a terminal profit*)
       YearlyProfitForecast[[Length[PriceForecasts], 1]] =
        If[GammaForecast ≤ Cutoff,
         ((YearlyProfitForecast[[Length[PriceForecasts], 1]] (1 + g)) / (RhoDiscountRate - g)),
         0];
       (*Discount the terminal profit*)
       YearlyProfitForecast[[Length[PriceForecasts], 2]] =
        YearlyProfitForecast[[Length[PriceForecasts], 1]] *
         If[i == 0, DiscountCoefficient^(Length[PriceForecasts]-1),
          BetaRate DiscountCoefficient^(Length[PriceForecasts]-1)];
      ];
      (*Update the NPV list we plan to output*)
      NPVList[[TimeWhenGammaInvest + 1, 1]] = TimeWhenGammaInvest;
      NPVList[[TimeWhenGammaInvest + 1, 2]] = Total[Transpose[YearlyProfitForecast][[2]]];
     ];
     (*Extract the date in the present/future with the highest net present value*)
     PlannedBeginDecarbonisation =
      NPVList[[Ordering[Transpose[NPVList][[2]], -1][[1]], 1]];
     {PlannedBeginDecarbonisation, NPVList}
    ]
```

```
In[604]:= (*Exactly the same but with heterogenous beta functionality *)
          GetSingleFirmNPVListWithBeta[PermitPriceForecasts_, firm0_, GammaStep0_,
            ProfitStep0_, InitialTimePeriod0_, RhoDiscountRate0_, BetaRate0_,
            PostCutoffGrowthRate_, CutoffGamma_, NumberYears0_] :=
           Module[{PriceForecastsFull = PermitPriceForecasts, firmlist = firm0,
              GammaStep = GammaStep0, ProfitStep = ProfitStep0, T = InitialTimePeriod0,
              RhoDiscountRate = RhoDiscountRate0, BetaRate = BetaRate0, g = PostCutoffGrowthRate,
              Cutoff = CutoffGamma, NumberYears = NumberYears0},
             DiscountCoefficient = 1 / (1 + RhoDiscountRate);
             PriceForecasts = Transpose[PriceForecastsFull][[3]];
             PermitForecasts = Transpose[PriceForecastsFull][[2]];
             NPVList = ConstantArray[0, {Length[PriceForecasts], 2}];
             For[TimeWhenGammaInvest = 0, TimeWhenGammaInvest ≤ Length[PriceForecasts] - 1 ,
              TimeWhenGammaInvest++,
              GammaForecast = firmlist[[2]];
```

```
   YearlyProfitForecast = ConstantArray[0, {Length[PriceForecasts], 2}];
   AccumulatedInvestmentForecast = 0;
   For[i = 0, i ≤ Length[PriceForecasts] - 1, i++,
    PermitsPerFirm = PermitForecasts[[i + 1]] / Length[firms];
    GammaNew = GammaForecast;
    If[i < TimeWhenGammaInvest,
     AccumulatedInvestmentForecast = AccumulatedInvestmentForecast + 1;
     ,
     If[GammaForecast > Cutoff,
       AccumulatedInvestmentForecast = 0;
       GammaNew = GammaNew - GammaStep;
       If[GammaNew < Cutoff, GammaNew = Cutoff, Null];
       ,
       AccumulatedInvestmentForecast = AccumulatedInvestmentForecast + 1
      ];
    ];
    GammaForecast = GammaNew;
    OptimalProduction =
     Solve[
       (∂c Profit[c, GammaForecast,
           100 + (AccumulatedInvestmentForecast * ProfitStep)]) ==
        PriceForecasts[[i + 1]] GammaForecast, c][[1, 1, 2]];
    YearlyProfitForecast[[i + 1, 1]] =
     Profit[OptimalProduction, GammaForecast,
       (100 + (AccumulatedInvestmentForecast * ProfitStep))] +
      PriceForecasts[[i + 1]] GammaForecast
       ((PermitsPerFirm / GammaForecast) - OptimalProduction);
    YearlyProfitForecast[[i + 1, 2]] =
     YearlyProfitForecast[[i + 1, 1]] * If[i == 0, 1, firmlist[[5]] DiscountCoefficient^i];
    YearlyProfitForecast[[Length[PriceForecasts], 1]] =
     If[GammaForecast ≤ Cutoff,
      ((YearlyProfitForecast[[Length[PriceForecasts], 1]] (1 + g)) / (RhoDiscountRate - g)),
      0];
    YearlyProfitForecast[[Length[PriceForecasts], 2]] =
     YearlyProfitForecast[[Length[PriceForecasts], 1]] *
      If[i == 0, DiscountCoefficient^(Length[PriceForecasts]-1),
       firmlist[[5]] DiscountCoefficient^(Length[PriceForecasts]-1)];
   ];
   NPVList[[TimeWhenGammaInvest + 1, 1]] = TimeWhenGammaInvest;
   NPVList[[TimeWhenGammaInvest + 1, 2]] = Total[Transpose[YearlyProfitForecast][[2]]];
  ];
  PlannedBeginDecarbonisation =
   NPVList[[Ordering[Transpose[NPVList][[2]], -1][[1]], 1]];
  {PlannedBeginDecarbonisation, NPVList}
 ]
```

```
In[605]:= (*Creates a function that outputs the list of profits in each future time
 period for a given decarbonisation date. This allows us to check that the
 output of our NPV list is correct,
and that firms are making the right decisions as given by the rules we want*)
GetYearlyForecastList[ChosenTime_, PermitPriceForecasts_, firm0_, GammaStep0_,
  ProfitStep0_, InitialTimePeriod0_, RhoDiscountRate0_, BetaRate0_,
  PostCutoffGrowthRate_, CutoffGamma_] :=
 Module[{Time = ChosenTime, PriceForecastsFull = PermitPriceForecasts,
   firmlist = firm0, GammaStep = GammaStep0, ProfitStep = ProfitStep0,
   InitialT = InitialTimePeriod0, RhoDiscountRate = RhoDiscountRate0,
   BetaRate = BetaRate0, g = PostCutoffGrowthRate, Cutoff = CutoffGamma},
  DiscountCoefficient = 1 / (1 + RhoDiscountRate);
  PriceForecasts = Transpose[PriceForecastsFull][[3]];
  PermitForecast = Transpose[PriceForecastsFull][[2]];
  OutputList = ConstantArray[0, 11];
  For[TimeWhenGammaInvest = 0, TimeWhenGammaInvest ≤ Length[PriceForecasts] - 1,
   TimeWhenGammaInvest++,
   GammaForecast = firmlist[[2]];
   YearlyProfitForecast  = ConstantArray[0, {Length[PriceForecasts], 2}];
   ElementList = ConstantArray[0, {Length[PriceForecasts] + 1, 2}];
   AccumulatedInvestmentForecast  = 0;
   For[i = 0, i ≤ Length[PriceForecasts] - 1, i++,
    PermitsPerFirm = PermitForecast[[i + 1]] / Length[firms];
    GammaNew = GammaForecast;
    If[i < TimeWhenGammaInvest,
     AccumulatedInvestmentForecast = AccumulatedInvestmentForecast + 1;
     ,
     If[GammaForecast > Cutoff,
       AccumulatedInvestmentForecast = 0;
       GammaNew = GammaNew - GammaStep;
       If[GammaNew  < Cutoff, GammaNew = Cutoff, Null];
       ,
       AccumulatedInvestmentForecast = AccumulatedInvestmentForecast + 1
      ];
    ];
    GammaForecast = GammaNew;
    OptimalProduction =
     Solve[
       (∂c Profit[c, GammaForecast,
           100 + (AccumulatedInvestmentForecast * ProfitStep)]) ==
        PriceForecasts[[i + 1]] GammaForecast, c][[1, 1, 2]];
    YearlyProfitForecast[[i + 1, 1]] =
     Profit[OptimalProduction, GammaForecast,
       (100 + (AccumulatedInvestmentForecast * ProfitStep))] +
      PriceForecasts[[i + 1]] GammaForecast
       ((PermitsPerFirm / GammaForecast) - OptimalProduction);
    YearlyProfitForecast[[i + 1, 2]] =
     YearlyProfitForecast[[i + 1, 1]] * If[i == 0, 1, BetaRate DiscountCoefficient^i];
    YearlyProfitForecast[[Length[PriceForecasts], 1]] =
     If[GammaForecast ≤ Cutoff,
        ((YearlyProfitForecast[[Length[PriceForecasts], 1]] (1 + g)) / (RhoDiscountRate - g)),
        0];
       YearlyProfitForecast[[Length[PriceForecasts], 2]] =
        YearlyProfitForecast[[Length[PriceForecasts], 1]] *
         If[i == 0, DiscountCoefficient^Length[PriceForecasts]-1,
          BetaRate DiscountCoefficient^Length[PriceForecasts]-1];
       ElementList[[i + 1, 1]] = YearlyProfitForecast[[i + 1, 1]];
       ElementList[[i + 1, 2]] = YearlyProfitForecast[[i + 1, 2]];
      ];
      ElementList[[Length[PriceForecasts] + 1, 1]] =
       Total[Transpose[YearlyProfitForecast][[1]]];
      ElementList[[Length[PriceForecasts] + 1, 2]] =
       Total[Transpose[YearlyProfitForecast][[2]]];
      OutputList[[TimeWhenGammaInvest]] = ElementList
     ];
     OutputList[[Time]]
    ]
```

In[606]:=
```
(*Simply runs a for loop,
inputting each firm in the sample into the NPVlist generator to get NPV lists
 for every firm in the sample*)
GetAllFirmNPV[firmlist0_, PriceForecasts0_ , ProfitStepSize0_, GammaStepSize0_,
   PresentBias0_, RhoDiscountRate0_, PostCutoffGrowth_, initialTime_, CutoffGamma_] :=
  Module[{firmlist = firmlist0, PriceForecasts = PriceForecasts0,
    PiStepSize = ProfitStepSize0, GammaStepSize = GammaStepSize0, Beta = PresentBias0,
    Rho = RhoDiscountRate0, g = PostCutoffGrowth, time = initialTime,
    Cutoff = CutoffGamma},
   DecarbonisationScheduleArray =
    Table[{i, firmlist[[i, 2]],
      GetSingleFirmNPVList[PriceForecasts, firmlist[[i]], GammaStepSize,
        PiStepSize, t, Rho, Beta , g, Cutoff, 10][[1]]}, {i, 1, Length[firms]}];
   DecarbonisationScheduleArray
  ];
```

In[607]:=
```
(*Same as above except with heterogenous beta functionality*)
GetAllFirmNPVWithBetas[firmlist0_, PriceForecasts0_ , ProfitStepSize0_,
   GammaStepSize0_, PresentBias0_, RhoDiscountRate0_, PostCutoffGrowth_,
   initialTime_, CutoffGamma_] :=
  Module[{firmlist = firmlist0, PriceForecasts = PriceForecasts0,
    PiStepSize = ProfitStepSize0, GammaStepSize = GammaStepSize0, Beta = PresentBias0,
    Rho = RhoDiscountRate0, g = PostCutoffGrowth, time = initialTime,
    Cutoff = CutoffGamma},
   DecarbonisationScheduleArray =
    Table[{i, firmlist[[i, 2]],
      GetSingleFirmNPVListWithBeta[PriceForecasts, firmlist[[i]], GammaStepSize,
        PiStepSize, t, Rho, Beta , g, Cutoff, 10][[1]]}, {i, 1, Length[firms]}];
   DecarbonisationScheduleArray
  ];
```

```
In[608]:= (*Implements changes that firms decide is best for them*)
          GetUpdatedFirmList[AllFirmDecarbonisationChoice0_, firms0_, GammaStep0_,
            ProfitStep0_, GammaCutoff0_, DefaultProfitCoefficient0_] :=
           Module[{DecarbChoice = AllFirmDecarbonisationChoice0, firmlist = firms0,
             GammaStep = GammaStep0, ProfitStep = ProfitStep0, GammaCutoff = GammaCutoff0,
             DefaultProfitCoefficient = DefaultProfitCoefficient0},
            NewFirms = firmlist;
            For[i = 1, i ≤ Length[firmlist], i++,
             (*If the firm has highest NPV today,
             then begin the decarbonisation process today*)
             If[DecarbChoice[[i, 3]] == 0,
              (*Caps the lowest gamma possible as the cutoff point*)
              NewFirms[[i, 2]] = Max[firmlist[[i, 2]] - GammaStep, 0.15];
              NewFirms[[i, 4]] = 0;
              ,
              (*If necessary increases the amount of accumulated investment*)
              NewFirms[[i, 4]] = firmlist[[i, 4]] + 1;
             ];
             NewFirms[[i, 3]] = Profit[c, NewFirms[[i, 2]],
               DefaultProfitCoefficient + (ProfitStep * NewFirms[[i, 4]])];
            ];
            NewFirms
           ]
```

```
In[609]:= (*Ties in all of our functions together and runs a simulation across all the
           time periods*)
          GetFullTimeEvolution[firmlist0_, InitialProfitCoefficient0_, NumberYears0_,
             InitialPermitSupply0_, FinalPermitSupply0_, ProfitStepSize0_, GammaStepSize0_,
             PresentBiasCoefficient0_, DiscountingFactor0_, PostCutoffGrowthRate0_,
             CutoffGamma0_] :=
            Module[{firmlist = firmlist0, InitialProfitCoefficient = InitialProfitCoefficient0,
              NumberYears = NumberYears0, InitialPermitSupply = InitialPermitSupply0 ,
              FinalPermitSupply = FinalPermitSupply0, ProfitStepSize = ProfitStepSize0,
              GammaStepSize = GammaStepSize0, PresentBiasCoefficient = PresentBiasCoefficient0,
              DiscountingFactor = DiscountingFactor0, PostCutoffGrowthRate = PostCutoffGrowthRate0,
              CutoffGamma = CutoffGamma0},
             (*Initialises arrays*)
             GammaEvolution = ConstantArray[0, { Length[firmlist], (NumberYears + 1)}];
             AccumulatedInvestmentEvolution =
              ConstantArray[0, {Length[firmlist], NumberYears + 1}];
             FirmListTensor = ConstantArray[0, NumberYears + 1];
             ProfitEvolution = ConstantArray[0, {Length[firmlist], NumberYears + 1}];
             QuantityEvolution = ConstantArray[0, {Length[firmlist], NumberYears + 1}];
             PermitPriceEvo = ConstantArray[0, NumberYears + 1];
             NewFirms = firmlist;
             For[j = 0, j ≤ NumberYears, j++,
              (*Follows the process steps as set out in the main body of the work*)
              PermitPriceEvolution = GetPermitPriceEvolution[j, InitialPermitSupply,
                FinalPermitSupply, NumberYears, NewFirms];
              InvestmentSchedule = GetAllFirmNPV[NewFirms, PermitPriceEvolution ,
                ProfitStepSize, GammaStepSize, PresentBiasCoefficient, DiscountingFactor,
                PostCutoffGrowthRate, TimeMarker, CutoffGamma];
              UpdatedFirmsList = GetUpdatedFirmList[InvestmentSchedule, NewFirms,
                GammaStepSize, ProfitStepSize, CutoffGamma, InitialProfitCoefficient];
              NewFirms = UpdatedFirmsList;
              YearlyResults = GetPermitEquilibrium[PermitPriceEvolution[[1, 2]],
                InitialPermitSupply, 150, 0.1, NewFirms];
              GammaEvolution[[All, j + 1]] = NewFirms[[All, 2]];
              AccumulatedInvestmentEvolution[[All, j + 1]] = NewFirms[[All, 4]];
              FirmListTensor[[j + 1]] = YearlyResults[[1]];
              ProfitEvolution[[All, j + 1]] =
               FirmListTensor[[j + 1]][[2 ;; Length[NewFirms] + 1, 8]];
              QuantityEvolution[[All, j + 1]] =
               FirmListTensor[[j + 1]][[2 ;; Length[NewFirms] + 1, 4]];
              PermitPriceEvo[[j + 1]] = YearlyResults[[3]];
             ];
             {GammaEvolution, AccumulatedInvestmentEvolution, ProfitEvolution,
              QuantityEvolution, PermitPriceEvo, FirmListTensor}
            ]
```

In[610]:=
```
(*Heterogenous betas*)
GetFullTimeEvolutionWithBetas[firmlist0_, InitialProfitCoefficient0_,
  NumberYears0_, InitialPermitSupply0_, FinalPermitSupply0_, ProfitStepSize0_,
  GammaStepSize0_, PresentBiasCoefficient0_, DiscountingFactor0_,
  PostCutoffGrowthRate0_, CutoffGamma0_] :=
 Module[{firmlist = firmlist0, InitialProfitCoefficient = InitialProfitCoefficient0,
   NumberYears = NumberYears0, InitialPermitSupply = InitialPermitSupply0 ,
   FinalPermitSupply = FinalPermitSupply0, ProfitStepSize = ProfitStepSize0,
   GammaStepSize = GammaStepSize0, PresentBiasCoefficient = PresentBiasCoefficient0,
   DiscountingFactor = DiscountingFactor0, PostCutoffGrowthRate = PostCutoffGrowthRate0,
   CutoffGamma = CutoffGamma0},
  GammaEvolution = ConstantArray[0, { Length[firmlist], (NumberYears + 1)}];
  AccumulatedInvestmentEvolution =
   ConstantArray[0, {Length[firmlist], NumberYears + 1}];
  FirmListTensor = ConstantArray[0, NumberYears + 1];
  ProfitEvolution = ConstantArray[0, {Length[firmlist], NumberYears + 1}];
  QuantityEvolution = ConstantArray[0, {Length[firmlist], NumberYears + 1}];
  PermitPriceEvo = ConstantArray[0, NumberYears + 1];
  NewFirms = firmlist;
  For[j = 0, j ≤ NumberYears, j++,
   PermitPriceEvolution = GetPermitPriceEvolution[j, InitialPermitSupply,
     FinalPermitSupply, NumberYears, NewFirms];
   InvestmentSchedule = GetAllFirmNPVWithBetas[NewFirms, PermitPriceEvolution ,
     ProfitStepSize, GammaStepSize, PresentBiasCoefficient, DiscountingFactor,
     PostCutoffGrowthRate, TimeMarker, CutoffGamma];
   UpdatedFirmsList = GetUpdatedFirmList[InvestmentSchedule, NewFirms,
     GammaStepSize, ProfitStepSize, CutoffGamma, InitialProfitCoefficient];
   NewFirms = UpdatedFirmsList;
   YearlyResults = GetPermitEquilibriumWithBetas[PermitPriceEvolution[[1, 2]],
     InitialPermitSupply, 150, 0.1, NewFirms];
   GammaEvolution[[All, j + 1]] = NewFirms[[All, 2]];
   AccumulatedInvestmentEvolution[[All, j + 1]] = NewFirms[[All, 4]];
   FirmListTensor[[j + 1]] = YearlyResults[[1]];
   ProfitEvolution[[All, j + 1]] =
    FirmListTensor[[j + 1]][[2 ;; Length[NewFirms] + 1, 8]];
   QuantityEvolution[[All, j + 1]] =
    FirmListTensor[[j + 1]][[2 ;; Length[NewFirms] + 1, 4]];
   PermitPriceEvo[[j + 1]] = YearlyResults[[3]];
  ];
  {GammaEvolution, AccumulatedInvestmentEvolution, ProfitEvolution,
   QuantityEvolution, PermitPriceEvo, FirmListTensor}
 ]

(*Generating output*)
```

In[662]:=
```
firms = FirmList[1, 0.5, 0.15, 100];
```

```
In[663]:= TimeEvoHomogenousBeta1 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 1,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta05 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.5,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta025 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.25,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta02 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.2,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta015 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.15,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta01 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.1,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta0075 = GetFullTimeEvolution[firms, 100, 10, 1000, 250,
             10, 0.1, 0.075,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta005 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.05,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta0025 = GetFullTimeEvolution[firms, 100, 10, 1000, 250,
             10, 0.1, 0.025,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta001 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0.01,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta0001 = GetFullTimeEvolution[firms, 100, 10, 1000, 250,
             10, 0.1, 0.001,  0.125, 0.03, 0.15];
          TimeEvoHomogenousBeta0 = GetFullTimeEvolution[firms, 100, 10, 1000, 250, 10,
             0.1, 0,  0.125, 0.03, 0.15];

In[•]:= betasample1 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample2 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample3 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample4 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample5 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample6 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample7 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample8 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample9 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample10 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample11 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample12 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample13 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample14 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample15 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample16 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample17 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample18 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample19 = FirmListWithBetas[25, 0.5, 0.15, 100];
        betasample20 = FirmListWithBetas[25, 0.5, 0.15, 100];
```

```
In[•]:= databetasample1 = GetFullTimeEvolutionWithBetas[betasample1, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample2 = GetFullTimeEvolutionWithBetas[betasample2, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample3 = GetFullTimeEvolutionWithBetas[betasample3, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample4 = GetFullTimeEvolutionWithBetas[betasample4, 100, 10, 1000,
          250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample5 = GetFullTimeEvolutionWithBetas[betasample5, 100, 10, 1000,
          250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample6 = GetFullTimeEvolutionWithBetas[betasample6, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample7 = GetFullTimeEvolutionWithBetas[betasample7, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample8 = GetFullTimeEvolutionWithBetas[betasample8, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample9 = GetFullTimeEvolutionWithBetas[betasample9, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample10 = GetFullTimeEvolutionWithBetas[betasample10, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample11 = GetFullTimeEvolutionWithBetas[betasample11, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample12 = GetFullTimeEvolutionWithBetas[betasample12, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample13 = GetFullTimeEvolutionWithBetas[betasample13, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample14 = GetFullTimeEvolutionWithBetas[betasample14, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample15 = GetFullTimeEvolutionWithBetas[betasample15, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample16 = GetFullTimeEvolutionWithBetas[betasample16, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample17 = GetFullTimeEvolutionWithBetas[betasample17, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample18 = GetFullTimeEvolutionWithBetas[betasample18, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample19 = GetFullTimeEvolutionWithBetas[betasample19, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
        databetasample20 = GetFullTimeEvolutionWithBetas[betasample20, 100, 10, 1000,
           250, 10, 0.1, 1, 0.125, 0.03, 0.15];
```