University of Brasília

Department of Economics

Graduate Program in Economics

Social Cost of Carbon: A Toy Model

Ruan Perassa Coelho

MASTER'S DISSERTATION

ECONOMICS

Brasília

2025

University of Brasília
Department of Economics
Graduate Program in Economics

Custo Social do Carbono: Um Modelo Simplificado

Ruan Perassa Coelho

Advisor: Prof. Dr. Roberto de Goés Ellery Júnior

Brasília
2025

CATALOGING-IN-PUBLICATION DATA SHEET

Perassa Coelho, Ruan.

Social Cost of Carbon: A Toy Model / Ruan Perassa Coelho; advisor Roberto de Goés Ellery Júnior. -- Brasília, 2025.

110 p.

Master's Dissertation (Economics) -- University of Brasília, 2025.

1. Macroeconomia. 2. Mudanças Climáticas. 3. Custo Social do Carbono. 4. Múltiplos Agentes. 5. Jogo Estático. I. Ellery Júnior, Roberto de Goés, orient. II. Título.

University of Brasília
Department of Economics
Graduate Program in Economics

Social Cost of Carbon: A Toy Model

Ruan Perassa Coelho

Brasília, April 10th 2025:

This work is dedicated to being lucky, well,
and, most importantly, alive.

Agradecimentos

Agradeço aos professores que me orientaram durante esse processo, Tomás Martinez e Roberto Ellery, pela paciência, atenção e confiança que tiveram em mim - pequenas atitudes tiveram enormes impactos sobre minha perspectiva e atitude em relação a essa dissertação.

Agradeço às pessoas mais próximas, que tiveram de ouvir sobre, se preocuparam, se animaram, se desanimaram, mas, acima de tudo, tentaram sempre me jogar para frente dentro desse mar de ressaca que foram os meses de elaboração desse documento.

Agradeço, por fim, a mim mesmo - o melhor amigo e pior inimigo. Que todo o caminho percorrido até aqui sirva como um grande aprendizado.

“Man’s main concern is not
to gain pleasure or to avoid pain but rather
to see a meaning in his life. That is why man
is even ready to suffer, on the condition, to be
sure, that his suffering has a meaning.“
(Viktor E. Frankl)

Resumo

A ambição de limitar o aquecimento global envolve, necessariamente, a compreensão do impacto das políticas de mitigação no sistema econômico, tanto no nível nacional quanto global. Esse trabalho apresenta um estudo sobre a trajetória de emissões para diferentes limites de aumento na temperatura média global através de uma versão modificada dos modelos DICE e DICE-CJL. Além disso, a pesquisa também investiga os resultados desse modelo para jogos estáticos com dois e três agentes. O estudo indica que a ótima taxação da tonelada de gás carbono, em dólares, cresce substancialmente a medida que menores aumentos máximos de temperatura são impostos.

Palavras-chave: Macroeconomia; Mudanças Climáticas; Custo Social do Carbono; Múltiplos Agentes; Jogo Estático.

Abstract

The ambition to cap global warming involve, necessarily, the compreheension of the mitigation policies' impact onto the economic system, both at national and global levels. This work presents a study on the emissions' trajectories for different average global temperature increase limits through a modified version of DICE and DICE-CJL models. Beyond that, the research also investigates this model's results for static games with two and three agents. This study indicates that the optimal taxation of a ton of carbon gas, in dollars, grow substantially as lower temperature increase maximums are imposed.

Keywords: Macroeconomics; Climate Change; Social Cost of Carbon; Multiple Agents; Static Game.

List of figures

List of tables

Contents

1 Introduction

Human activities, principally through emissions of greenhouse gases, have unequivocally caused global warming, with global surface temperature reaching 1.1°C above 1850–1900 in 2011–2020.[...] Human-caused climate change is already affecting many weather and climate extremes in every region across the globe. (Core Writing Team, H. Lee and J. Romero (eds.), 2023)]

Anthropogenic global warming is already much present, as we've recently breached the 1.5ºC warming point above pre-industrial levels for the very first time (Copernicus Climate Change Service (C3S), 2025). Much of its catastrophic damages are now being either visualized and measured, mostly through climatic events such as El Niño/La Niña or the Indian Monsoon, or estimated through computer modelling. It's a hard task, even now that climate change is a daily reality with related jobs going from cloud physics researchers to ESG consultants. It is a recurrent subject in the news, either linked with natural disasters, such as floods and droughts, heatwaves and snowstorms, or with innovation, such as green energy, electric cars, green cities and sustainable products. Many of its consequences are still underestimated, unpredicted or unknown, especially those in human populations (i.e. starvation, migration, disease).

There is one thing the international community has, for a long time now, understood: the only way to properly address this issue is by reducing GHG emissions. Conversations about tackling the CO_2 emissions date way back, before the 1980s, and it quickly became an important matter for the scientific community, as even Carl Sagan testified before the US Congress on the urgency of this undeniable matter. However, very little has been done to really mitigate emissions at national and subnational levels, especially compared to levels we *must* achieve to cap global warming at relatively low temperatures. The IPCC AR6 synthesis report states that, with high confidence, the concentrations of carbon dioxide in the atmosphere at the present moment "are higher than at any time over at least the past two million years" and that average global surface temperature surpassed the 1850-1900 levels by 1.1ºC in the 2011-2020 period, with higher increases seen over land (1.59ºC) than over ocean (0.88ºC) (Core Writing Team, H. Lee and J. Romero (eds.), 2023).

CO_2 is not the only gas which provokes global warming: methane (CH_4), nitrous oxide (N_2O) and fluorinated gases (F-gases) are also heavy contributors to the greenhouse effect that stimulates the heating of our atmosphere. But, although their warming impact is much higher than carbon dioxide, our CO_2 emissions from fossil fuels were, in 2023,

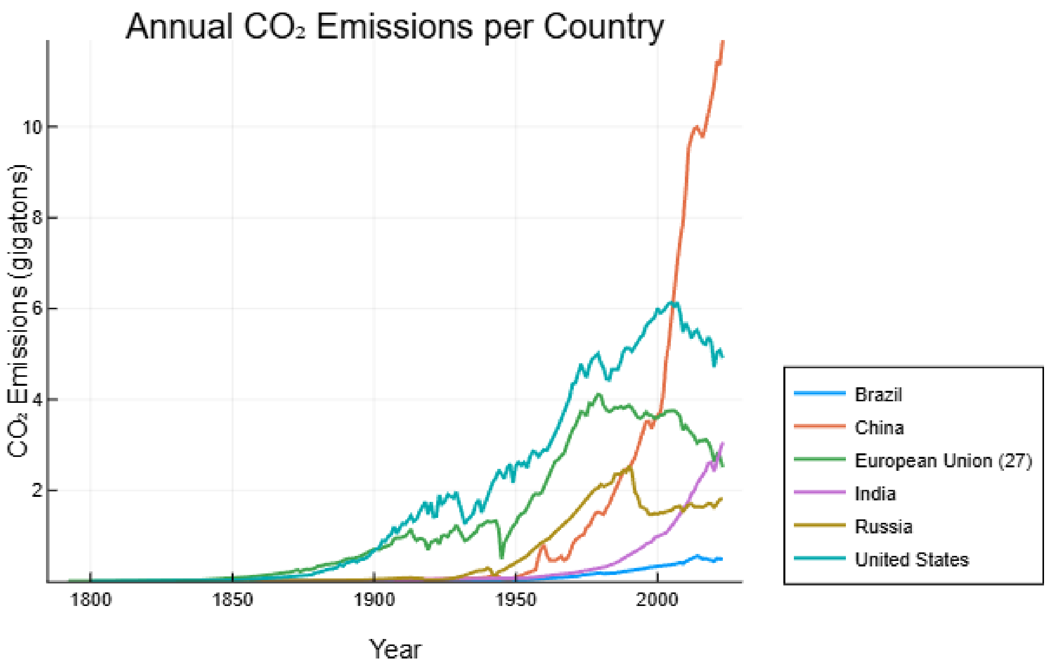

Figure 1.1 – Annual carbon dioxide emissions from fossil fuels and industry yearly by country. Data Source: Our World in Data.

73.7% of total GHG emissions - not including, therefore, changes in land use and forestry (which are the largest ones in Brazil) (Core Writing Team, H. Lee and J. Romero (eds.), 2023).

In figure 1.1, it is possible to see how countries like China, India and Russia still increase their emissions annually, while the United States and the EU, although now on the downturn side of the curve, still emit quite a lot (gross emissions). These five countries and the EU, according to the EDGAR's "GHG Emissions of All World Countries" 2024 report, were responsible for 64.2% of all global emissions in 2023, this time including LULUCF (Land Use, Land Use Change and Forestry) and, at the same year, accounted for 63.2% of global gross domestic product (Crippa *et al.*, 2024). Any possible offensive we might take against climate change must involve these nations in coordination, as they are a very large slice of the problem - this is one of the objectives of this study, to understand how some key features, like population size, growth and capital stock, determine the optimal solution to tackle human-induced global warming.

The growing trend of global emissions has been consistent from the beginning of the 21^{st} century, with only two exceptions: the 2009 global financial crisis and the COVID-19 pandemic. This trend is mainly composed by China, India and other emerging countries: from figure 1.2, one can deduce from India's graph that the time series mean for, at least, the past forty years is several points above zero. When talking about GHG per capita emissions, the picture is completely different: China and India go way down on the largest emitters' list, while much smaller countries like Qatar, Kuwait and United Arab Emirates head the list - and Brazil surpasses China and India, but remains behind the USA and Russia on per capita emission numbers.

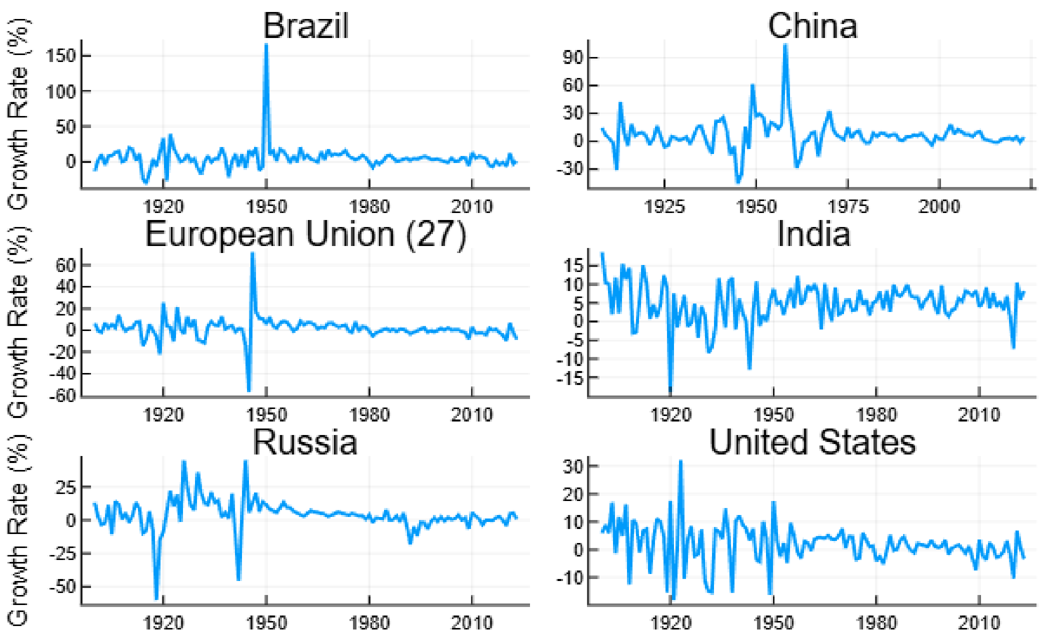

Figure 1.2 – Annual CO_2 emissions growth rate from fossil fuels and industry yearly by country. Data Source: Our World in Data.

International agreements are one solution to address climate change as an international crisis, not only at a regional or national level. This solution seems quite reasonable, as climate change fits a very specific type of problem in economics - it's an *externality* issue, which can be summed up here as an issue where the agents responsible for the damage caused are not paying for those damages. The damage in this case, of course, is spread all over the globe, hitting the hardest in low-income global south countries (Core Writing Team, H. Lee and J. Romero (eds.), 2023). So, getting emitters from all over the globe, even those that may have not emitted at all practically, to sit down together and make an agreement on how to conduct future pollutant policies does have a large potential to deal with human-induced global warming. Yet, most agreements between the largest GHG emitters have not progressed with sufficient care and willfulness - focusing solely on the more recent Paris Agreement, most countries have not met their own GHG emissions mitigation criteria, or even worse, quit the agreement altogether (as the United States of America did twice, in 2017 and in 2025). An illustrative ranking made by Climate Action Tracker (Climate Action Tracker, 2025) shows that the biggest global GHG emitters, accounting for nearly two thirds of all emissions worldwide, have their targets and actions in reducing greenhouse gases labeled as "insufficient" if they were to meet the Paris Accord 1.5ºC - also indicating that, given the policies realized and planned for the near future from these countries, temperature would likely sit between the range of 2-3ºC by the year 2100 (Climate Action Tracker, 2025). In order to observe possible future paths, scientists use models of the real world, encompassing physical and social features most of the times.

Models are very useful in understanding and simulating reality, and they also perform a significant role in the field of Economics. Ranging from quite simple and straightforward non-structural linear regressions to quite large dynamic stochastic general equilibrium models, or even dynamic games with multiple agents, there's a very important role for modelling, in all its spectrum, to understanding and doing economics. Economic models are mathematical representations of how the real economic system works and can differ in their fundamental building blocks, such as foresight (how the agents inside the model see the future), aggregation level (how agents are separated from one another), variable exogeneity (which key features of the economy come from within the model), motion laws (how do the economic variables behave along time) and time scale (how far in the future does the model try to reach). They can also have common ground between them: the class of models this work will deal with, for example, perform a kind of optimization process - meaning that these models were developed to obtain an optimal solution given certain assumptions and constraints, such as mathematical properties of utility functions, production functions and expectations. Choosing between one feature or another is paramount for constructing properly the problem itself, so it may have a solution, and for solving it - making a sparser definition of sectors or countries may have a large impact on how fast one can find a solution.

Moreover, economic models differ not only in their theoretical underpinnings but also in the way they are applied across subfields, reflecting diverse research agendas and degrees of complexity. Some frameworks focus narrowly on a single market or sector—so-called partial-equilibrium models—while others adopt a full general-equilibrium approach to capture interactions among all markets simultaneously. Likewise, models may be static snapshots or dynamic systems that trace economic evolution over time, and they can treat uncertainty either deterministically or through stochastic processes. The choice between these alternatives often hinges on the questions at hand: for instance, macroeconomists routinely employ dynamic stochastic general equilibrium (DSGE) models to analyze business cycles and policy shocks, whereas industrial organization scholars might favor game-theoretic oligopoly models to study firm behavior in specific industries. Equally important are practical considerations—data availability, computational tractability, and the desired granularity of results can tip the balance toward simpler specifications or, conversely, motivate more detailed, computationally intensive designs. By selecting the appropriate scope and structure, researchers ensure that their models remain both analytically tractable and relevant to real-world decision-making.

Among types of economic models, one that connects with climate-oriented research is the Integrated Assessment Model (IAM) class. These are a combination of modelling exercises between two (or more) fields, economic and environmental (for the majority of them, climate-related), and have been quite useful in understanding what policymakers' options are on tackling global warming and, in some manner, forecasting future trajectories given the present set of actions. This is the first objective of this dissertation: develop my own code for some traditional IAMs as a means of understanding how they work, what their strong and weak points are, and verify previous results in the literature (taken as a benchmark later in this work). As mentioned before, international agreements are important and implementing a stylized version of them within the traditional model scope is a second objective of this work. In it, two or more agents interact with each other and have to define emissions' trajectories that sum up at the end at the atmospheric level. A third and final objective is to study, within the implementation phase mostly, the impacts of different parameter values - some of them set by the original authors - on the model's output.

So, in order to lay the groundwork and present this work's model, the next sections are as follows: section 2 is a presentation of relevant literature for IAMs, where conceptual definitions and the main reference models for this work are presented; section 3 contains a brief discussion of the computational and mathematical challenges of DICE, DICE-CJL and DSICE, and the choices made to formulate the model that is solved here; section 4 presents the model itself, which is highly similar to DICE-CJL, and the algorithm used in finding its solution; section 5 presents the results for the benchmark case and some studies on its behaviour when some key parameters change; section 6 concludes this work and discuss the implications of its results to policy making and further research.

2 Model Development

2.1 Integrated Assessment Models

Let's define what are IAMs, first and foremost, as to their conceptual nature. Here's a sufficient definition on IAM from (FisherVanden; Weyant, 2020):

> [...]IAMs are tools that capture the complex interactions and interdependencies across the natural and human systems and across spatial and temporal scales for a wide range of uses, including improving the science of fine-scale impact analysis, multi-stakeholder policy making, and the development of adaptation strategies.

Integrated assessment models, in its historical essence, are mathematical (here, including statistical) representations of the economic (human) system coupled with some sort of natural (physical) system; here, authors may choose the degree of complexity of both economic and natural systems, and also how realistic is the interaction between them. Examples of human systems that have had appearances in IAMs are sectors such as energy, agriculture and aviation (Jacoby *et al.*, 2006); as for physical systems, common ones are temperature and carbon concentration feedbacks.

As the definition properly stated, one of the objectives in developing and solving coupled models like these are assisting policy makers on deliberating strategies on mitigation and adaptation from environmental issues. Many of those have already been the target for one or more Integrated assessment model, such as water polution (Chaubey *et al.*, 2021), ocean acidification (Cameron; Vial, 2019), air quality (Wang; Mueller; Gerber, 2021) and biodiversity loss (Vuuren *et al.*, 2022). Climate and economic wedded models are, however, the origin and still the most common ones in IAM literature - varying from small ones, with large time scales and sparse system definition, to huge large-scale models, such as (Cai; Judd; Lontzek, 2012; Cai; Lontzek, 2019), where the model has a large state space and it must be solved via supercomputers (it would take way too long in a normal computer!). Entering the scope of this work, I focus now solely on IAMs that deal with some sort of climate module, specially those targeting global climate change. Next, I discuss some key aspects of climate IAMs that aided me in the construction of my model.

2.1.1 Macroeconomics

Most cost-benefit IAMs (DICE and its descendants) are all built from the same macroeconomy standpoint, which reflects the importance of macroeconomy inside climate-economy coupled studies. The theoretical standpoint from this work's model is to assume a central global social planner that chooses allocations, at an aggregated level, in order to maximize social welfare while simultaneously correcting the externality problem - a straight derivation from the Ramsey-Cass-Koopmans growth model (Ramsey, 1928; Cass, 1965; Koopmans, 1965)

$$\begin{aligned} \max_{c} \quad & \int_0^{\infty} e^{-(\rho-n)t} u(c)dt \\ s.t. \quad & c = f(k) - (n+\delta)k - \dot{k} \end{aligned}$$

However, other models approach the problem from a competitive equilibrium perspective and, thus, a representative agent solves the allocation problems in face of a Pigouvian carbon tax - which equals the marginal damage of an extra ton of carbon emitted to the atmosphere. It is also true that models that deal with regional effects and policies not only exist, but are quite common and have quite an important role for large policy making institutions around the world. Examples are FUND (Climate Framework for Uncertainty, Negotiation and Distribution), which is a multiregion model with sector specialized damage functions and is a part of the IPCC reports' impact assessment; and RICE, another model developed by Nordhaus (Nordhaus; Yang, 1996), that extends the original DICE by splitting the world into multiple blocks (regions), each with idiosyncratic economic (production, abatement cost, etc) and damage functions.

2.1.2 Optimization

A very strong backbone of nearly all cost-benefit IAMs, the social planner's maximization problem provides a clear welfare criterion. Here, every ton of carbon abated is directly weighted against consumption foregone and future damages to output avoided. As the same first-order conditions can be derived from a representative-agent economy - facing the Pigouvian carbon tax -, it is ensured that policy solutions are quite robust to both social planner and competitive equilibrium framings. A direct consequence of the optimal control framework is a more normative direction in its conclusions as results imply an *optimal* tax (or price) given a set of constraints on how the economy and climate interact and stabilize for long periods. This comes with a substantial sensitivity for key parameters, such as the pure time preference and the capital elasticity of output that can alter the resulting social cost of carbon by orders of magnitude. Other sensible points for this approach are computational aspects such as the famous *curse of dimensionality* (Bellman, 1957) and setting proper boundaries on the state space so that feasibility is satisfied over it.

2.1.3 Damage Functions

Every modern IAM must translate economic activity into a physical variable that affects global climate and then feed the resulting impacts back into social welfare; this is usually done by damage functions and abatement costs. Damage functions try to translate changes in the environment that hurts society - hence, the economy - in some kind of way; these changes could be, but are not limited to, increase in the intensity and duration of droughts, rainfloods, sea-level rise and heatwaves (Core Writing Team, H. Lee and J. Romero (eds.), 2023). All the previous catastrophic events mentioned have a temperature component to its severity (Core Writing Team, H. Lee and J. Romero (eds.), 2023), therefore macroeconomists have simplified climate damages in a mathematical function only dependent on current temperature variation compared to pre-industrial levels. As mentioned in (Hassler; Krusell; Smith, 2016), bottom-up approaches also exist, where authors characterize individual types of damage and assign market prices to them, e.g. sea-level rise will likely have a large impact in real state. DICE and many other models use a quadratic functional and directly connect the damage to total output, aggregating for both types of damage and locations. Regional models sometimes have local damage functions forms (Tol; Narita; Anthoff, 2008; Hassler; Krusell; Olovsson, 2021) and, even though quadratic forms are the most common, other forms such as polynomial have been used in the past (Weitzman, 2011; Murphy; Nordhaus; Reilly, 2018). One unifying feature of all these choices is that these are heavily simplified version of a global and non-uniform phenomenom, so damages are usually subestimated. DSICE adds a stochastic *climate tipping point*, which can be described as "critical threshold at which a tiny perturbation can qualitatively alter the state or development of [the climate] system" (Cai; Lontzek, 2019), to input the case of high damage and low probability events into optimal policies.

2.1.4 Abatement Costs

The abatement factor, usually characterized as μ and called *the mitigation rate*, can be either an exogenous (in this case, one may set previously the mitigation trajectories and evaluate the impact differences from each one on economic and climate variables) or an endogenous variables. The latter is usually formulated with the mitigation rate being one the optimization controls - for DICE and many others, the social planner chooses both consumption and μ to maximize the discounted sum of utilities over time. Most cost–benefit IAMs (DICE and RICE, for example) use a simple convex polynomial with a quadratic exponent as the largest; if the optimal rate chosen is 100%, then abatement costs are fixed at a constant proportion of output (sometimes identified as the *backstop price*). This, of course, ignores heterogeneity between important sectors, such as energy, aviation and other heavy industries, and their capacity to actually adapt to greener technologies within the model's timeframe. In (McKinsey & Company Kimberly Henderson, 2020), the authors identify *abatement potentials* for a variety of sectors and regions in order to create emission pathways that a coherent with temperature targets.

2.1.5 Uncertainty

Originally, DICE and RICE are both deterministic systems. Since then, models have incorporated stochastic components both in economic and climate modules (Weitzman, 2009; Cai; Judd; Lontzek, 2012; Lemoine; Traeger, 2014). Adding random perturbations may present a challenge, specially for computational reasons: many of the IAMs, specially those similar to DICE, rely on *value function iteration* to find the optimal policies - and, without extraordinary properties that may accelerate the common algorithm, authors use interpolation or collocation methods to approximate their solutions. In (Lemoine; Traeger, 2014), authors develop a continuous-time version of DICE with a technology level that follows a standard AR1 process; DSICE (Cai; Lontzek, 2019) also implements the shock, but adds a persistency component inside the shock's law of motion and define their systems in discrete time. For both models, authors choose to use *Chebyshev polynomials* to approximate their solutions, but in the latter, authors interpolate the solution at each time step, and in the former, Jenn and Traeger use the Chebyshev polynomials to solve approximately a differential equation at the chosen nodes. In any case, it's worth to mention that adding uncertainty in models tend to make investment costs higher and mitigation efforts quicker and larger. DSICE also shows that, by adding climate tipping events such as the collapse of Atlantic Meridional Overturning Circulation (AMOC) or the Amazon Forest Dieback (AFD), carbon dioxide and other GHG should be priced or taxed way higher than they currently are.

2.1.6 Preferences and Discounting

Changing discount rates can shift dramatically the outcome of an IAM. Some previous work, such as (Cai; Judd; Lontzek, 2012; Cai; Lontzek, 2019) and (Lemoine; Traeger, 2014), evaluated how robust are their results to variations in both preference and discounting parameters. DICE and subsequent models have long used the Constant Relative Risk Aversion (CRRA) function to measure the agent's utility of consumption, while using the Ramsey equation

$$r = \rho + \eta * g$$

to tie up the discount rate r to the growth rate g. In (Stern, 2007), it is famously argued that the future consumption discount rate should almost entirely be an *ethical* decision, in contrast with Nordhaus (Nordhaus, 1994) and many other think that the rates should reflect observed consumer and market behaviours. This work will not dwell too much in this topic, as many others have already done it and it remains a sensitive, debatable matter. However, it is important to analyze some scenarios for the risk aversion parameter, as climate change damages still have a large uncertainty around them.

Among the mentioned IAMs, three are the main references of this work: DICE (Dynamic Integrated Climate-Economy), DICE-CJL (CJL are the initial letter for the authors' names, Cai, Judd and Lontzek) and DSICE (Dynamic Stochastic Integrated Climate-Economy). DICE and DICE-CJL are, essentially, the same model, the only difference being that the authors made the time frame more flexible in DICE-CJL - in DICE, the model is solved for every ten years within the time span of some centuries, whereas DICE-CJL made it possible to solve it to even fractions of a year. DSICE is an extension of a one-year DICE-CJL that incorporates stochastic shocks in the technology level, so connected to the productivity and output levels, and in the climate, through an multiplicative functional form term in the damage function and, by consequence, affecting final output directly. DICE-CJL is the model chosen for direct replication in this work and can be defined as an intermediate point between the final model and Nordhaus' DICE and thus having, essentially, the same climate and economic models as DICE. This means that DICE-CJL is also a deterministic model. All reference models can be found in the annexes.

3 A Toy Model

This model is a version of DICE-CJL with a one-year time step, essentially. Primarily, as the main reference model DSICE has a very large state space and two control variables, it suffers heavily from the *curse of dimensionality* and has to be solved by *value function iteration* - which can be computationally costly if not very optimized to run in parallel for this and the other DICE related models. However, DICE-CJL is completely deterministic and can be solved as a constrained nonlinear optimization problem via a NLP solver. This solution method makes it quite unfeasible to insert uncertainty (i.e. stochastic shocks) into the decision making as all periods are solved simultaneously. Therefore, the productivity shocks performed in DSICE are left out for further research.

The alternative to adding stochastic shocks chosen here is to add more countries or regions to the model, making it a multi-agent version of DICE-CJL (and DICE, indirectly). The interaction between the agents is, forcibly, described by an open-loop game: the starting agent chooses her optimal trajectory given a guess for the other players' trajectories; then, come the other players' rounds, where each one at a time has the same opportunity to optimize its own choices based, now, on the trajectories already defined by the first player. After all players have done their path selection, next rounds come until some threshold has been achieved.

Games of two and three agents have been performed, where new calibration was done to match Germany's (for both two and three agents cases) and China's (only three agents game) economic data. In both games, the climate system is shared by all countries and, obviously, affected by each one's emissions.

3.1 Climate

Beginning with the climate module, two variables are used in the temperature block and three variables in the carbon concentration block, making a total of five climate variables to have their time series set within each player's turn. With respect to the carbon concentration, the system's dynamics are described by the motion law (in matrix form)

$$\boldsymbol{M}_{t+1} = \boldsymbol{\Phi}_M \boldsymbol{M}_t + (E_t, 0)^T$$

where $\boldsymbol{M}_t = \left(M_{AT,t}, M_{OC,t}, M_{LO,t}\right)$ is the three-dimensional vector for carbon concentrations, in order, at the atmosphere, at the upper oceanic layer and at the lower oceanic layer. Clearly, all emissions go directly to the atmospheric layer of the system, being partially transferred to the upper ocean with a two year delay. Total emissions are divided into two distinct paths: a land (hence, environmental) component $E_{land,t}$ and a human component $E_{ind,t}$

$$E_t = E_{land,t} + E_{ind,t}$$

The parameter matrix $\mathbf{\Phi}_M$ has the following structure

$$\mathbf{\Phi}_M = \begin{bmatrix} 1-\varphi_{12} & \varphi_{21} \\ \varphi_{12} & 1-\varphi_{21} \end{bmatrix}$$

The parameters' values for this carbon diffusion matrix have all been calibrated by authors on (Cai; Lontzek, 2019) on historical data and the original paper have all the important information on the matter. For the temperature side of climate, a similar linear dynamical system is constructed:

$$\boldsymbol{T}_{t+1} = \mathbf{\Phi}_T \boldsymbol{T}_t + (\xi_1 \mathcal{F}(M_{AT,t}), 0)^T$$

in which

$$\mathbf{\Phi}_T = \begin{bmatrix} 1-\phi_{21}-\xi_2 & \phi_{21} \\ \phi_{12} & 1-\phi_{12} \end{bmatrix}$$

In a parallel fashion, $\mathbf{\Phi}_T$ can be interpreted as the heat diffusion rate matrix between Earth's layers, and the $\mathcal{F}(M_{AT,t})$ is the *radiative forcing* function - dependent only on the CO_2 concentration at the atmosphere. As stated in (Cai; Lontzek, 2019), ξ_2 is "the rate of cooling arising from infrared radiation to space" and, hence, represents a constant exogenous loss of atmospheric temperature for the complete climate system. Global warming, as an increase in global average T_{AT}, derives from the radiative forcing term, which is defined as

$$\mathcal{F}(M_{AT,t}) = \eta \cdot log_2\left(\frac{M_{AT,t}}{M^*_{AT}}\right) + F_{EX,t}$$

where M^*_{AT} is the preindustrial atmospheric carbon concentration. Again in the temperature half of the climate module, the oceanic temperature has a fixed trajectory derived from the deterministic, with the complete state space, DICE solution. Here, again, any stochastic shock that $M_{AT,t}$ may suffer (due to a transfer from gross output via emissions) has an impact on $T_{AT,t+1}$ via radiative forcing; then, at $t+1$, this impact is shared with $M_{OC,t+1}$ as heat diffusion occurs between geolayers.

3.2 Economy

One social planner overlooks the entire closed economy, with no government. She maximizes the sum of discounted utility over an infinite (approximated here by 600 years, as done both in DICE and DICE-CJL) periods of time.

$$\max_{\{c_t,\mu_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(C_t, L_t)$$

where c_t is the per capita average consumption - hence, the aggregated total consumption is $C_t = c_t * L_t$ for the population L_t, measured in millions. As mentioned before, the time step is one year, so every variable described here is of annual frequency. Population trend over time is

$$L_t = L_{ss} - (L_{ss} - L_0) \cdot e^{-dL\cdot(t-1)}$$

The annual gross output per country (or globally, in the one-agent case) has the Cobb-Douglas production function shape:

$$f(A_t, K_t, L_t) = A_t K_t^{\alpha} L_t^{1-\alpha}$$

where K_t is the capital stock (in trillions of US dollars) and A_t is the deterministic productivity time series, as set by the expression

$$A_t = A_0 \exp\left(\frac{\alpha_1(1-e^{-\alpha_2 t})}{\alpha_2}\right)$$

The equation above is extracted from the DSICE model (different from both DICE and DICE-CJL) and its parameters α_1 and α_2 denote, respectively, the 2005 growth rate and the decline rate of α_1. The output Y_t represents the gross product already discounted by the climate damage - due to higher levels of atmospheric temperature in comparison to pre-industrial time - via $\Omega(T_{AT,t})$, hence

$$Y_t \equiv \Omega(T_{AT,t}) f(A_t, K_t, L_t)$$

where

$$\Omega(T_{AT,t}) = \frac{1}{1 + \pi_1 \cdot T_{AT,t} + \pi_2 \cdot T_{AT,t}^2}$$

Damage functions, as mentioned before, have been a fairly discussed topic among some well-established authors within the climate IAMs discussion (Nordhaus, 1994; Weitzman, 2009). In this brief work, I will use only the most common choice to describe mathematically global average damage from the temperature anomaly. However, another useful format that applies a simple but quite potent transformation, proposed by (Weitzman, 2009), uses a quadratic polynomial, but within an exponential function, as shown below,

$$\Omega_{exp}(T_{AT,t}) = \frac{1}{e^{1+\pi_1 \cdot T_{AT,t} + \pi_2 \cdot T^2_{AT,t}}}$$

which can substantially modify the main economic variables' outputs in comparison to the former damage structure. Putting into numbers, an increase in 1ºC would mean a gross product 0.36 lower in the exponential case than in the simple quadratic one.

Raises in temperature derive from rises in carbon concentration, which have both natural and human contributions. Anthropogenic emissions (here, *ind* indicates industrial) are defined as, in billions of metric tons of carbon dioxide,

$$E_{ind,t} = \sigma_t(1-\mu_t)f(A_t, K_t, L_t)$$

Here, σ_t is the carbon intensity of output and is described mathematically as

$$\sigma_t = \sigma_0 \exp\left(\frac{-0.0073(1-e^{-0.003t})}{0.003}\right)$$

The capital motion law and the national equilibrium equality are defined, respectively, as

$$K_{t+1} = (1-\delta) \cdot K_t + I_t$$

$$Y_t = C_t + \Psi_t + I_t$$

In the second expression above, the term Ψ_t is the abatement cost of mitigation level μ_t and is expressed as a fraction of the gross product (post-damage) in the format

$$\Psi_t = \theta_{t,1}\mu_t^{\theta_2} Y_t$$

in which the multiplier $\theta_{t,1}$ is the mitigation cost coefficient, described by the following equation

$$\theta_{t,1} = \frac{1.17\sigma_t(1+e^{-0.005t})}{2\theta_2}$$

The benchmark one agent model, used as reference for all the other variations presented in the results section, will follow the parametrization common for DICE, DICE-CJL and DSICE - both economic and climate parameters are included. All parameters already presented in numerical form in these expressions above will be maintained throughout the work.

To conclude this section, I define the *social cost of carbon* both theoretically and mathematically. From (Nordhaus, 2008),

> Another key concept in the economics of climate change is the "carbon price," or, more precisely, the price that is attached to emissions of carbon dioxide. One version of a carbon price is the "social cost of carbon." This measures the cost of carbon emissions. More precisely, it is the present value of additional economic damages now and in the future caused by an additional ton of carbon emissions.

Therefore, the social cost of carbon is the marginal loss in "economic damages" - which could be consumption or capital, for this model it's the same - for an extra ton of carbon dioxide released into the atmosphere by industrial activity. This marginal loss is measured in a monetary fashion, usually US dollars. One parallel way to look at this cost is as the relative *shadow price* (a rate of substitution) between carbon concentration in the atmosphere and capital (or consumption) for a given period.

$$SCC = -1000 * \frac{\frac{\partial \mathcal{L}}{\partial M_{AT,t}}}{\frac{\partial \mathcal{L}}{\partial C_t}}$$

The equation below shows the version of SCC I use in this work, where $\mathcal{L}$ is the Lagrangian of the whole 600-period optimization problem. Carbon concentration is measured in billions of CO_2 tons and consumption in trillions of 2005 US\$, making it necessary to adjust it by a thousand.

3.3 Parameterization

Below are the values for all the parameters and variables defined to set the Julia's computational version of the work presented here. Most values are derived from the original (Nordhaus, 2008) and some are borrowed from (Cai; Lontzek, 2019) - the reason for taking some values from the latter work is that the model developed, in some minimal sense, here employs a lot of the functional format that DSICE does.

The variables for the alternative social planners were chosen by me, and their choice was rather experimentalist. Knowing the DICE's functions for productivity and abatement cost, which have essentially the same mathematical structure with different parametrization, I chose to tweak them and aim to build different behaviours in face of the same set of controls and states. Comparison between the resulting altered functions is presented below.

Symbol	Value	Description
ψ	[0.5, 1.5, 2]	Intertemporal elasticity of substitution
β	0.985	Utility time discount factor
α	0.3	Capital elasticity of gross output
α_1	0.0092	Productivity growth rate
α_2	0.001	Rate of decay for the productivity growth rate
δ	0.1	Capital depreciation rate per year
A_0	0.0272	Productivity trend inital value (trend level)
L_0	6514.0	Population at initial time (in millions)
K_0	137.0	Capital stock at initial time (in $ trillions)
π_1	0.0	Climate damage factor
π_2	0.0028388	Climate damage factor
θ_2	2.8	Abatement cost parameter
σ_0	0.13418	Carbon intensity of output initial value (level)
$d\sigma$	-0.00730	Drift rate of decarbonization
σ_{dcy}	0.003	Decay rate of adjustment

Table 3.1 – One Agent Model Economic Parameters - Benchmark

Symbol	Value	Description
$M_{AT,0}$	808.9	Initial atmospheric carbon concentration
$M_{UO,0}$	1255	Initial upper ocean's carbon concentration
$M_{LO,0}$	18365	Initial lower ocean's carbon concentration
φ_{12}	0.019	Diffusion rate from atmosphere to upper ocean
φ_{23}	0.0054	Diffusion rate from upper to lower ocean
φ_{21}	0.01	Diffusion rate from upper ocean to atmosphere
φ_{32}	0.00034	Diffusion rate from lower to upper ocean
$T_{AT,0}$	0.7307	Initial atmospheric temperature
$T_{OC,0}$	0.0068	Initial oceanic temperature
ξ_1	0.037	Atmospheric temperature rate of change due to radiative forcing
ξ_2	0.047	Atmospheric temperature decrease rate due to infrared radiation to space
ϕ_{12}	0.01	Heat diffusion rate from atmosphere to ocean
ϕ_{21}	0.0048	Heat diffusion rate from ocean to atmosphere
η	3.8	Radiative forcing parameter
M_{AT}^*	596.4	Preindustrial atmospheric carbon concentration level

Table 3.2 – One Agent Model Climate Parameters - All Models

As this is a highly stylized model, the purpose of the handpicked parameters is to evaluate a global scenario of different generic (in some degree) agents rather than to make it as close as possible to the real world's economic system. Therefore, to make a computational thought experiment where I can evaluate how integrating different actors alters traditional

DICE results.

To avoid repetition, here I present only the adapted values for the three regions presented in the one agent model scenario and, later, integrated into the three agent model. The first region repeats almost all parameters from the benchmark case. Initial capital stock data for these regions were obtained from the Penn World Table version 10.0, where the first region's value is the USA's 2005 capital stock, the third region's value is Germany's, and the second region was chosen to sit somewhere in between these other two. In the two and three agent model, I use these values to estimate each region's proportion to the global capital stock parameter value defined in DICE, so values for these multiagent versions are higher.

Population was also selected in a stylized fashion: each region was given a fraction of the total world's population loosely based on some countries' data, such as the USA and Germany. The important information is that: the first region, which has higher costs for mitigating and a higher productivity growth rate, has also the largest population of all three - followed by the third region, which has the least carbon intense economy, and then the second region (again, remaining in the middle of the other two regions when it comes to carbon intensity of output).

Symbol	**Value**	**Description**
α	0.3	Capital elasticity of gross output
α_1	0.0092	Productivity growth rate
α_2	0.001	Rate of decay for the productivity growth rate
A_0	0.0272	Productivity trend inital value (trend level)
L_0	2791.714	Population at initial time (in millions)
L_{ss}	3685.714	Steady-state population (in millions)
K_0	56.0489	Capital stock at initial time (in $ trillions)
σ_0	0.13418	Carbon intensity of output initial value (level)
$d\sigma$	-0.00730	Drift rate of decarbonization
σ_{dcy}	0.003	Decay rate of adjustment

Table 3.3 – One Agent Model Economic Parameters - First region

Below are the plots for the carbon intensity of output σ_t, population L_t and productivity A_t for all three regions. The first region, over time and over other regions, is always on top; following it, region two and three, with the exception of population levels, where the third region was calibrated to be denser.

Symbol	Value	Description
α	0.26	Capital elasticity of gross output
α_1	0.0075	Productivity growth rate
α_2	0.0011	Rate of decay for the productivity growth rate
A_0	0.0295	Productivity trend inital value (trend level)
L_0	1395.857	Population at initial time (in millions)
L_{ss}	1842.857	Steady-state population (in millions)
K_0	16.430	Capital stock at initial time (in \$ trillions)
σ_0	0.10	Carbon intensity of output initial value (level)
$d\sigma$	-0.0080	Drift rate of decarbonization
σ_{dcy}	0.0035	Decay rate of adjustment

Table 3.4 – One Agent Model Economic Parameters - Second region

Symbol	Value	Description
α	0.28	Capital elasticity of gross output
α_1	0.0055	Productivity growth rate
α_2	0.0013	Rate of decay for the productivity growth rate
A_0	0.0332	Productivity trend inital value (trend level)
L_0	2326.428	Population at initial time (in millions)
L_{ss}	3071.428	Steady-state population (in millions)
K_0	35.2	Capital stock at initial time (in \$ trillions)
σ_0	0.0885	Carbon intensity of output initial value (level)
$d\sigma$	-0.0082	Drift rate of decarbonization
σ_{dcy}	0.0032	Decay rate of adjustment

Table 3.5 – One Agent Model Economic Parameters - Third region

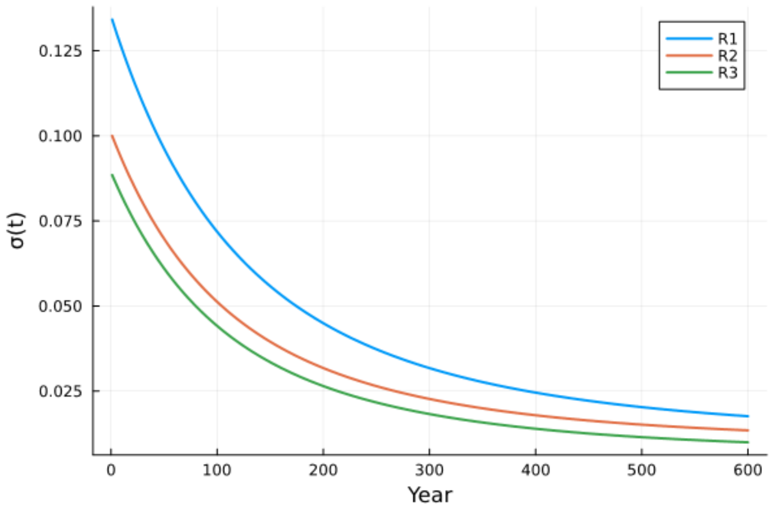

Figure 3.1 – Carbon intensity for all three regions over time.

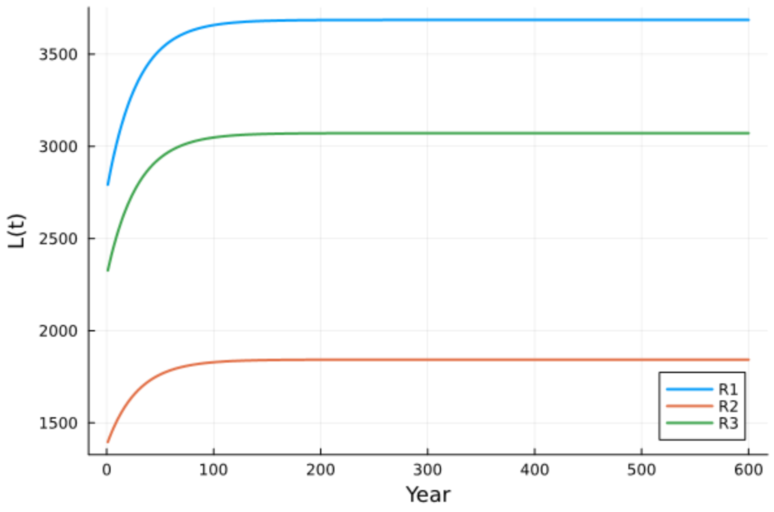

Figure 3.2 – Population for all three regions over time.

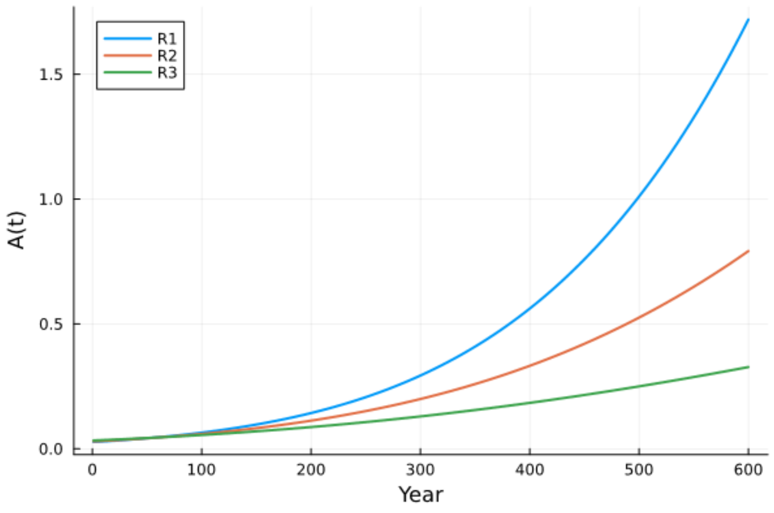

Figure 3.3 – Productvitity for all three regions over time.

4 Results

This section will be divided into three separate sections. In the first, I'll show the benchmark case of DICE-CJL, using the parametrization and structure from DICE with the modifications made in (Cai; Judd; Lontzek, 2012). It is important to add that one little modification was performed that links the model here to DSICE: I modified the utility function so it has the intertemporal elasticity of substitution exponent as in 4. In this manner, the γ parameter within this work is the inverse of the traditional DICE's version.

$$u(C_t, L_t) = \frac{\left(\frac{C_t}{L_t}\right)^{(1-1/\gamma)}}{1-\frac{1}{\gamma}} \cdot L_t$$

The one-agent model is simply the repetition of a close-to-identical DICE-CJL; two and three are expansions from the former, where agents will interact with each other in order to find an idiosyncratic economic equilibrium strategy while sharing the climate system.

The second and third sections show the outputs from solving the model with two and three agents, respectively. There are no complex rules for the two and three-agent models. For example, once the two agents' program starts, the first player has a guess on the other regions' best emissions responses and solves the optimization problem with the following modified constraint:

$$M_{AT}(t+1) \quad = \quad \phi_{11}M_{AT} + (1-\phi_{11})M_{UO} + E_{Region1}(t) + E_{Region2}(t) + E_{land}(t)$$

Other variables don't matter directly for decision making as the only link with the climate is carbon emissions and, therefore, each player can independently choose their respective mitigation rates and consumption. At the end of a round, the next round's emissions guess for each player is updated with a linear combination of the most recent and the old trajectories (alike the Gauss-Siebel method for matrices and linear systems). The process is repeated until a convergence threshold is met.

There is one caveat for the three-agent models: the order in which each region plays each round is random; therefore, it's possible that the two last agents to play may do so with the most recent version of the first player's optimal emissions trajectory.

Interestingly, one could see this recursive game as a recurrent Climate Change Conference, such as COP21 and COP25, where countries, separated into two or three blocks representing the whole world, choose how they will choose the next six centuries' emissions path and effectively keep these paths with complete honesty. At the end of each round, a mediator takes the values given by all players and, individually, does a ever-decreasing ponderate average of them until the differences between two rounds' decisions, for all players, are close enough.

4.1 One-Agent Model

Primary results are shown below for the benchmark case - all the parameters can be found in the appendix **??**.

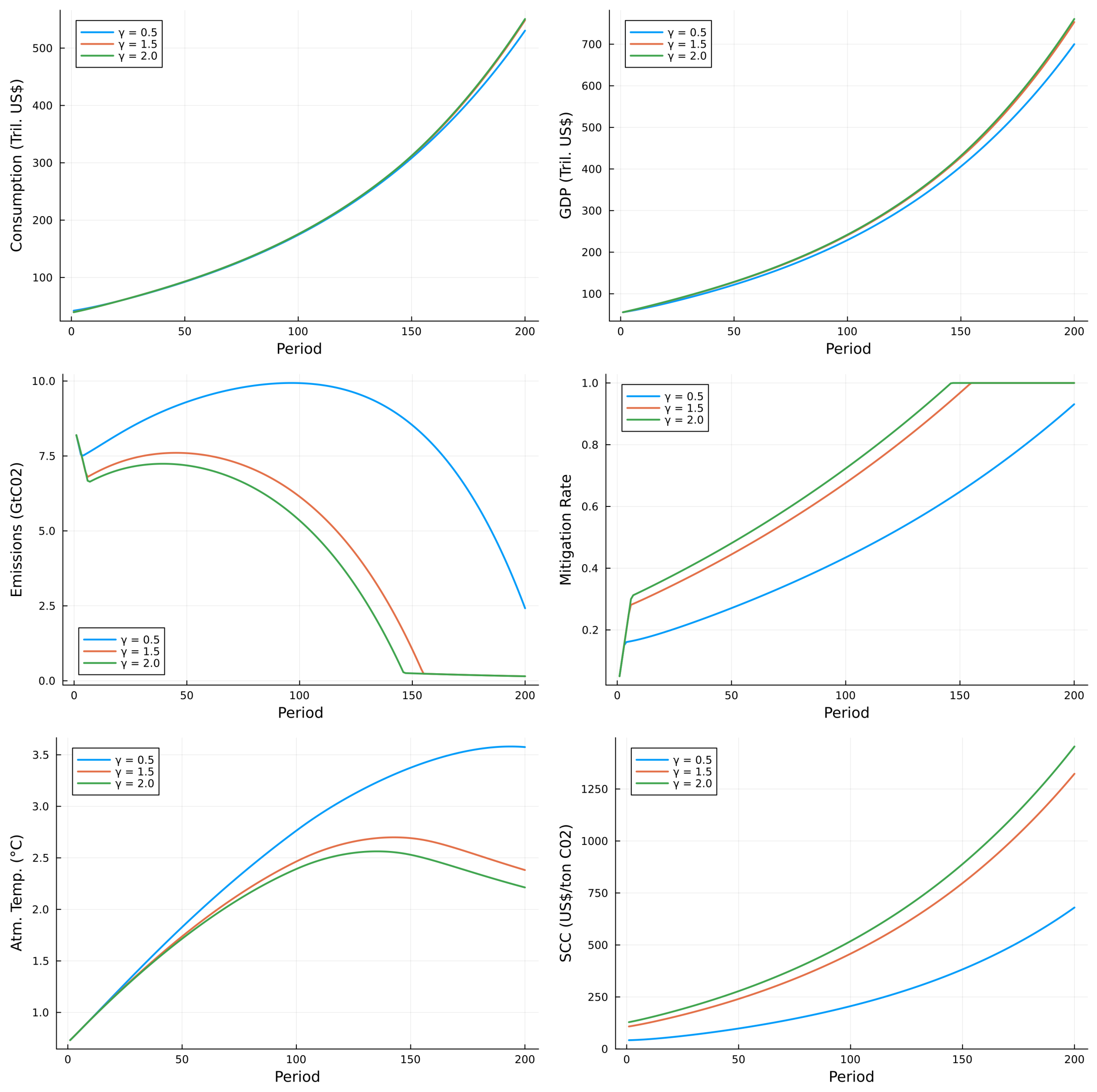

Figure 4.1 – Economic and climate results for the benchmark case - one-agent, DICE/DSICE standard parametrization.

Having in mind the results above, let's now look at the behaviour of the almos exact same system, but with different IES. As the parameter γ rises for the scheme A calibration, capital and GDP also slightly go up, while emissions lower substantially - as expected, once gamma represents the IES and makes the social planner abdicate more current consumption in exchange for future consumption. Consumption on itself is not sensitive to changes in the intertemporal elasticity, leaving all the trade-off necessities to the mitigation rate. It is possible to visualize in figure 4.2 that the social planner chooses to reduce CO_2 emissions almost entirely at the expense of capital, therefore forcing the same optimal consumption trend over time throughout the IES range: γ goes up, so the same agent now is less fond of trading consumption at the current moment for higher damages way down the line at the same time it still prefers to keep the same levels of historic consumption; hence, it depletes a little more the current capital stock - i.e. increases fundamentally savings, although a light rise in investment is also noticed - to preserve future levels of consumption concurrently mitigating global warming.

At the year 2205, temperature surpasses the $3°C$ level when the traditional intertemporal elasticity value is set. Instead, when greater values are imposed on the agent's IES, there's an approximately $0.5°C$ fall at the same year for atmospheric temperature altogether with rather lower carbon concentrations in the atmosphere - which start to decrease earlier as the natural process inserted into the model by the climate motion laws.

The other two parameterization schemes show similar results, but not quite the same. In the second region parameterization scheme, productivity remains with the same historical trajectory, population levels are lower and the carbon intensity of output σ_t has major modifications: not only is the initial value lower (meaning a lower overall carbon intensity level), but the rate of growth over time is lower (more negative) and the speed of convergence to the long-run carbon intensity is greater. The combination of a lower growth rate with a greater convergence velocity, in this scenario, gives a lower $d\sigma/\sigma dcy$ ratio, which should incline the carbon intensity to grow more moderately; however, as σdcy assumes larger positive values, the time-dependent component is driven faster to the long-run value. Sigma's effect can be seen in figure 3.1. For the total aggregated result, figure 4.3 shows that the maximum values reached for the 200-year period are all below the first region's largest values, no matter which variable is in focus.

The third region's results follow the same pattern as region 2, when in comparison to region 1. Maximum values for all variables are smaller than their counterparts in the other two regions. Emissions show a curious result: their decay is slower, but for the traditional IES value (again, here represented by $\gamma = 0.5$, the decay is faster and it starts sooner than in the other regions. Mitigation rates over time are especially low, passing over the 50% line only after more than 150 periods while the other two regions were closer to the 100% mark at this point in time.

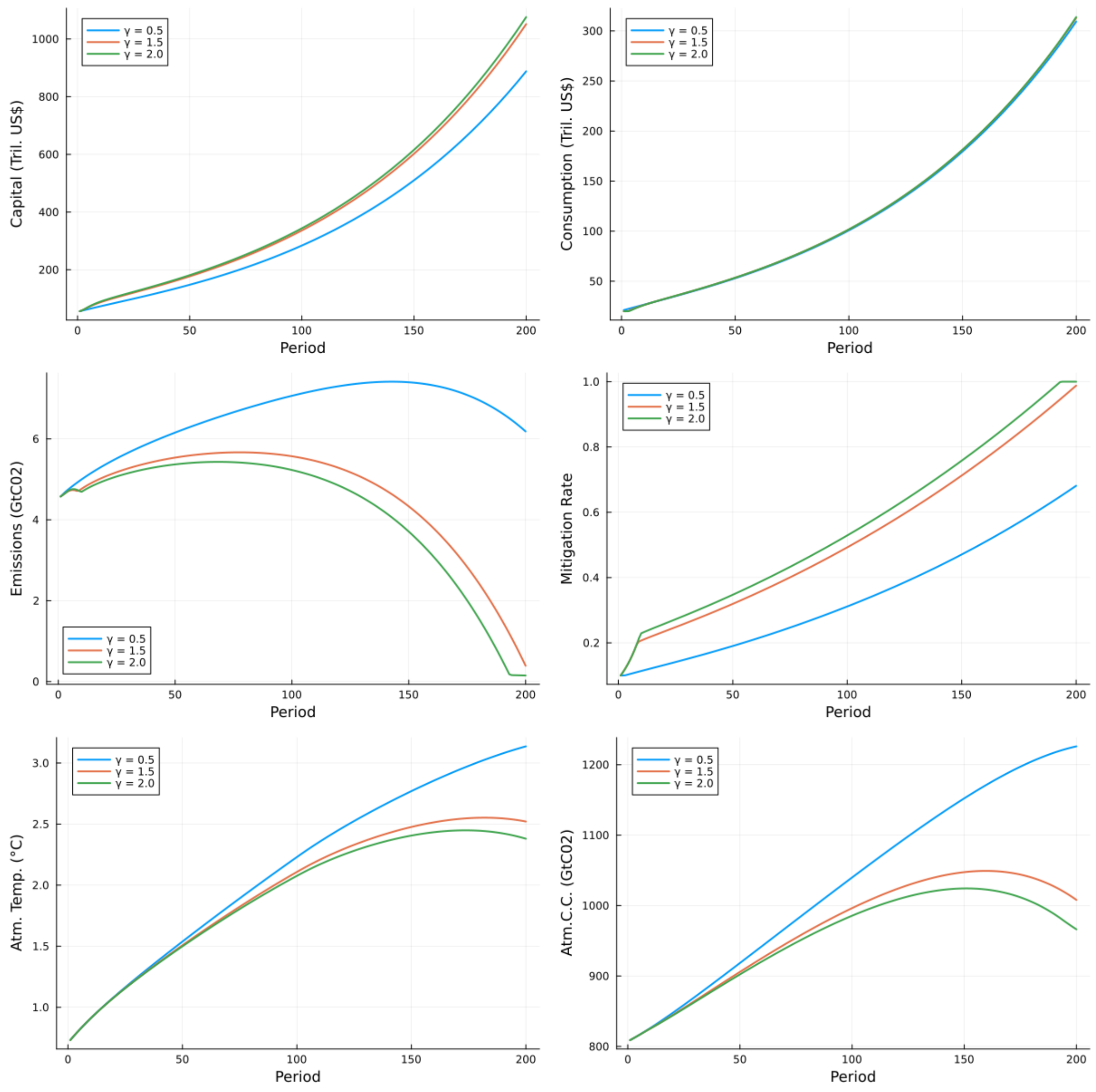

Figure 4.2 – Economic and climate variables for the first scheme of parametrization. Trajectories in blue, orange and green represent the optimal paths under $\gamma = 0.5$, $\gamma = 1.5$ and $\gamma = 2.0$, respectivelly. Capital and consumption are displayed in trillions of US$; emissions and atmospheric carbon concentration are measured in billions of CO_2 tons; and temperature is measured in degrees Celsius (deviation from pre-industrial levels).

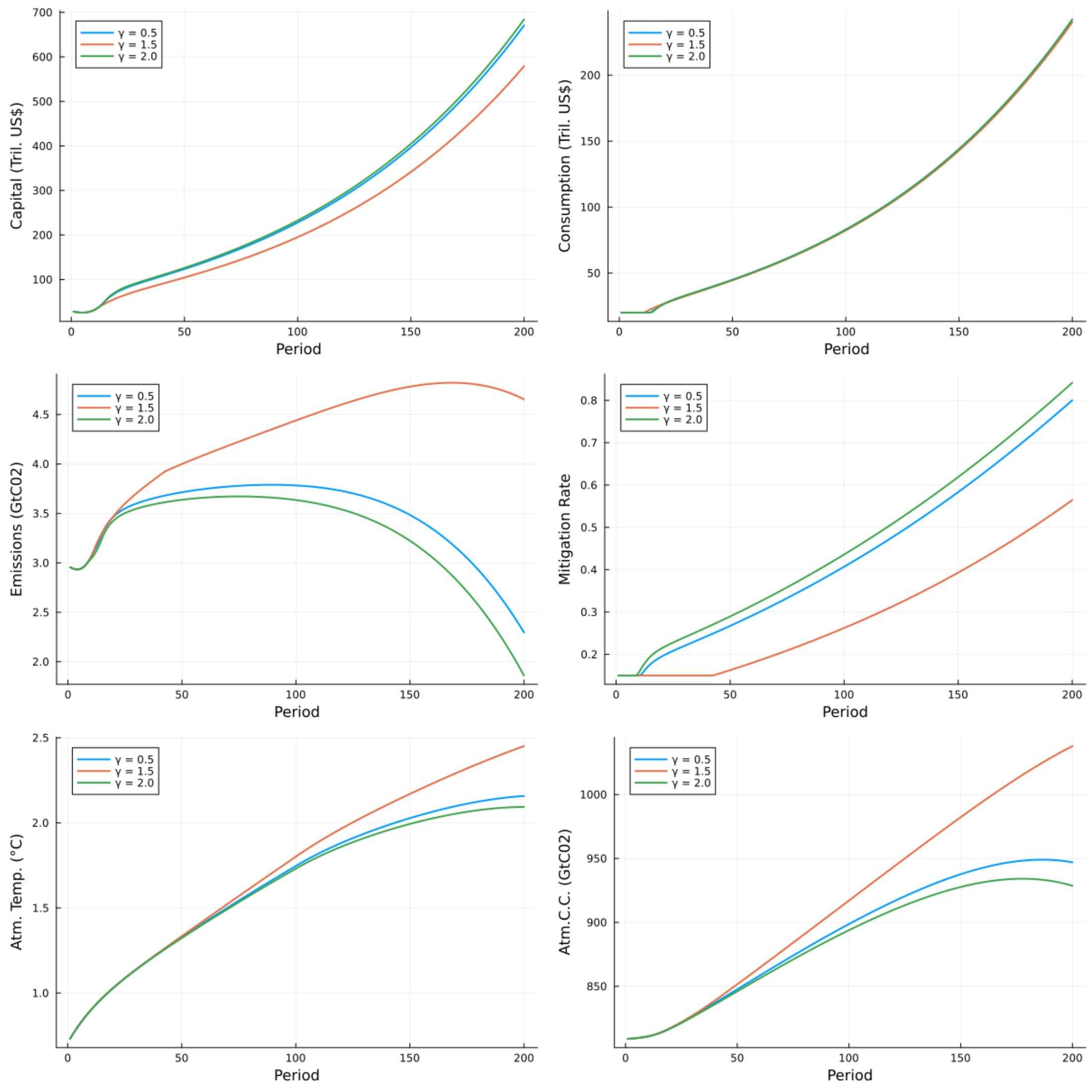

Figure 4.3 – Economic and climate variables for the second scheme of parametrization. Trajectories in blue, orange and green represent the optimal paths under $\gamma = 0.5$, $\gamma = 1.5$ and $\gamma = 2.0$, respectivelly.

These results are important to show that there are some unanimous key dynamic features from the model's structure: capital and consumption both increase over the first 200 periods; emissions tend to increase over the first periods, with this period being extended or compressed based upon the IES chosen, and then decrease (sometimes to zero); and atmospheric temperature and carbon concentration seem to grow and then stabilize during the same period (for higher IES values, even a slight decrease is perceived).

4.2 Two-Agents Model

For demonstration of the two-agents model, I've chosen to display only two of the three regions previously defined to interact with each other. The model does allow swap-

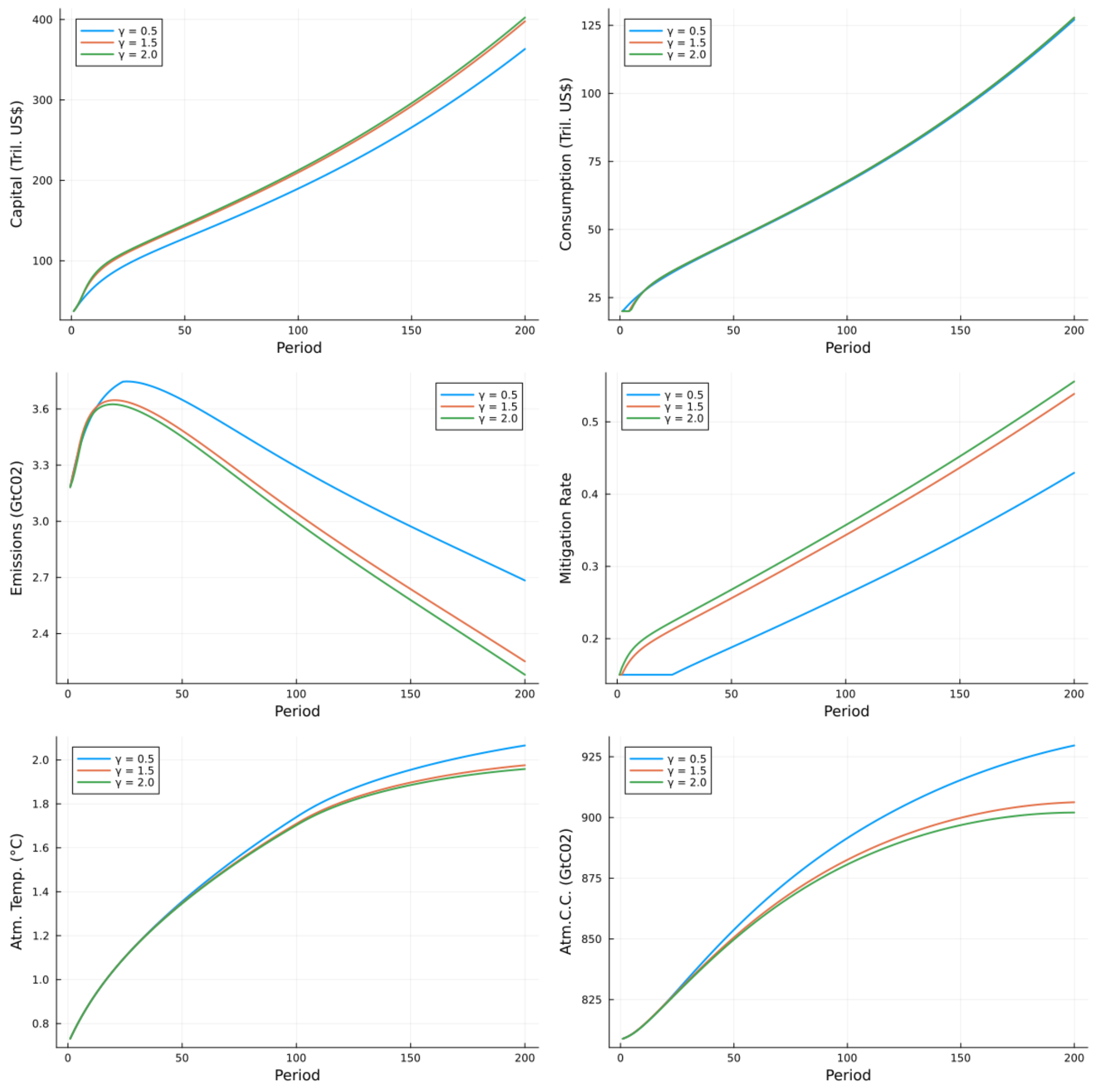

Figure 4.4 – Economic and climate variables for the third scheme of parametrization. Trajectories in blue, orange and green represent the optimal paths under $\gamma = 0.5$, $\gamma = 1.5$ and $\gamma = 2.0$, respectivelly.

ping regions deliberately, as only parametrization is unique for each of them - all defining expressions and bounds are essentially the same.

4.2.1 3ºC Case

Results for the two-agents model with a 3ºC cap on average global surface temperature increase from preindustrial times are shown in 4.5. First and foremost, it can be seen that the atmospheric temperature levels do not reach the maximum value imposed; therefore, this restriction is non-binding as the optimal solution still lies within boundaries. Each country has its own social cost of carbon, as they perceive marginal utilities differently for their respective emissions and capital trajectories. Notably, region 1 takes the leading role in mitigation, reaching 100% mitigation rates shortly after the period 150, while region 2 is closer to 40% abatement for the same period. Still, and this is repetitive in the two-agents model framework, region 1 manages to increase substantially its GDP throughout the 200-period time span, even more than the one-agent benchmark case. Region 2 extends its non-zero (or far-from-zero) emissions for a longer time - so once region 1's emissions are tangent to zero, the climate system relies almost solely upon region 2's climate policy not to overshoot.

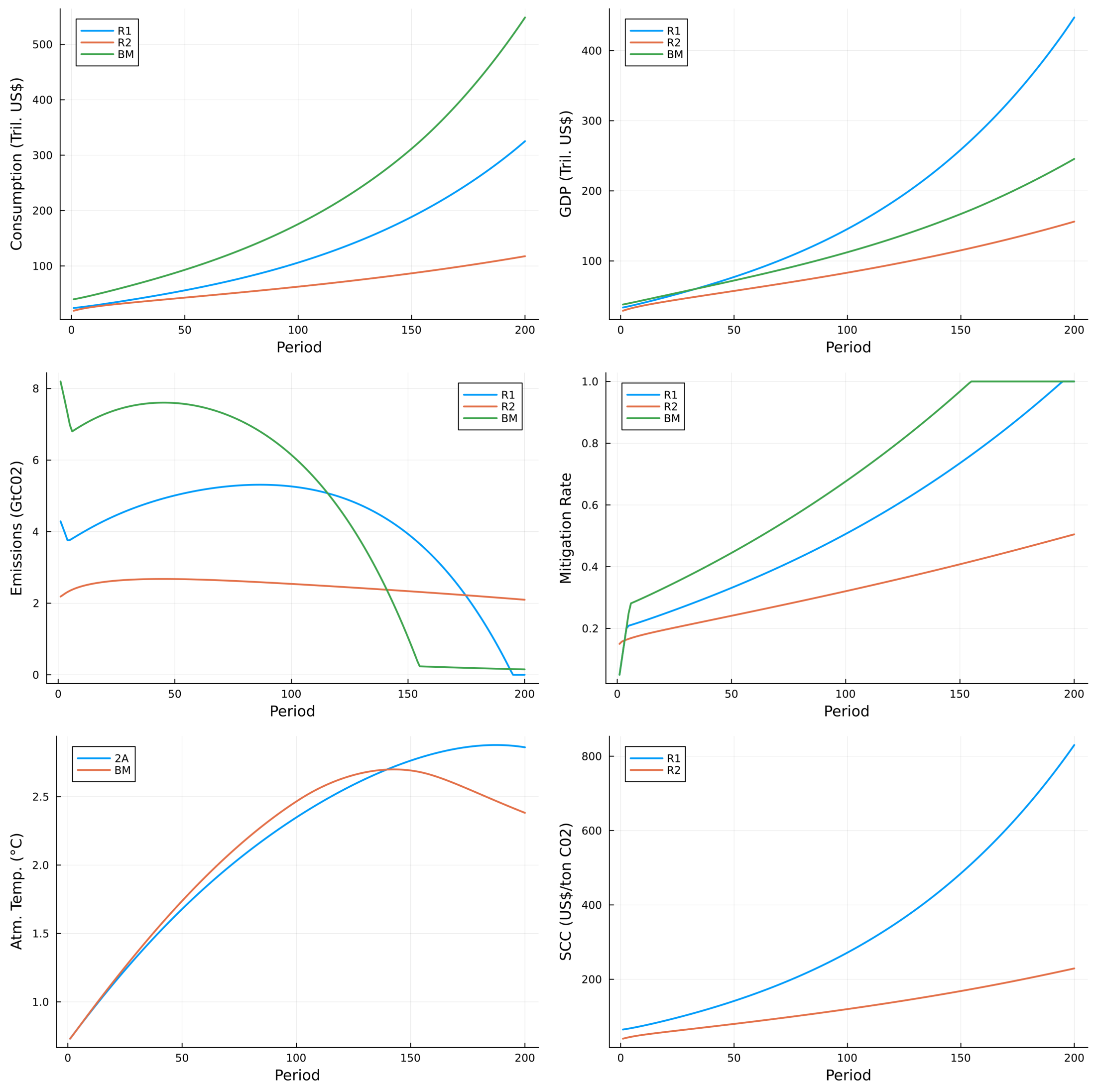

Figure 4.5 – Economic and climate variables for the two-agents model results with a 2ºC temperature anomaly cap. Trajectories in blue, orange and green represent the optimal paths under $\gamma = 0.5$, $\gamma = 1.5$ and $\gamma = 2.0$.

It is possible to interpret from the emissions plot that total emissions start not so prominent as the benchmark's analogue, but it catches up somewhere between the periods 50 and 100; after the latter period, total emissions exceed those same results for the benchmark's case and sustains the gap between them for a long time (region 2's emissions only come close zero around the year 2400).

4.2.2 2ºC Case

When limiting global warming at the 2ºC maximum threshold, the bigger and demographically denser region takes a front role in mitigating its own emissions pretty quickly. While consumption levels are below those seen in the benchmark trajectory, region 1 obtains a GDP time series that is above that of the benchmark case - region 2 doesn't, even though it mitigates and consumes far less. As one social planner assumes that the emissions path is set by the other, having no bargaining power on what it should or shouldn't be, these results do seem as the only equilibrium this game has (although no formal proof is presented, several simulations within the same practical aspects always output the same result).

Interestingly, as region 1 aggressively mitigates its emissions during the first 100 years, the surface temperature approximates the fixed maximum temperature with less speed than in other scenarios. Social cost of carbon, on the contrary, rises sharply for region 1, breaching the $US\$5000$ line even before the year 150 - much more than any other verified case.

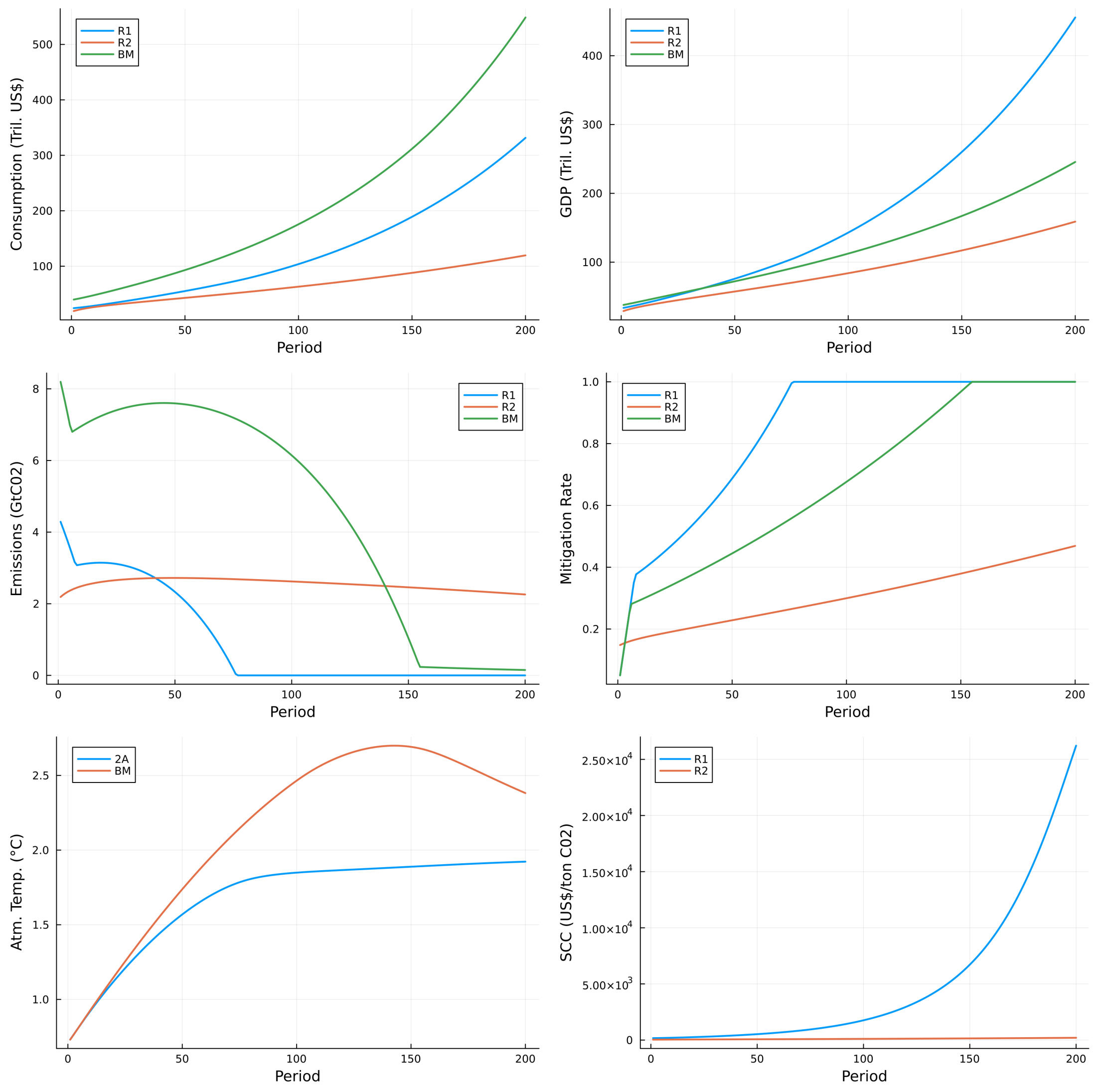

Figure 4.6 – Economic and climate variables for the two-agents model results with a 2ºC temperature anomaly cap. Trajectories in blue, orange and green represent the optimal paths under $\gamma = 0.5$, $\gamma = 1.5$ and $\gamma = 2.0$.

4.3 Three-Agents Model

The results for the three-agents model are below, separated into three main cases. Each case is, basically, set upon a limit to average global surface temperature. These limits are 1.5ºC, 2.0ºC and 3.0ºC.

Before beginning the dissection of each case's results, one thing must be indicated: the order in which planners solve this problem is randomized for each run, but the first player has a disadvantage of setting his trajectory with only the best guess of the other two players' responses - and the same follows for the second player's choice with respect to the third player's best guess. Their choice is made either with an educated guess (based on their one-agent model's solution) or the previous iteration's answer, while the second and, especially, the third agent's optimization process is based on the most recent response of the other players. For instance, if the first player decides, based on the others' best response guesses, to mitigate substantially as the other players haven't done so and temperatures are causing too great a damage, he gets stuck on this high abatement path. Randomizing the order in which the three regions solve the problem is a beginning, but it does not deal completely with the issue - the first player, at the first round, can either decide to mitigate too little or too much, and this can have a great impact on all three regions' trajectories.

However, mitigation rates' and emission trajectories' levels are quite similar among the different optimizations performed, especially when looking at the general dynamic of the system - such as the time period span for emissions growth and decline. Therefore, first I present an example case for the trajectories - one among the huge number of possible scenarios. Then, I show the average trajectories from 100 simulations, and the results from the climate system once these paths are enforced into the dynamics.

4.3.1 3.0ºC Case

So, as previously said, this is an example path for the three-agents 3ºC limit warming model. Consumption per region is lower than the benchmark case, as expected, and region 1 - which has a greater initial and steady-state population, greater initial capital stock and faster productivity growth - shows both greater levels of consumption and GDP (directly influenced by capital stock).

Mitigation rates stay below the benchmark threshold for almost all the trajectory period (except for the initial period, where I forced the benchmark case to start at $\mu = 0.0$). One region (in this example, region 1) depicts a sharper upward inclination, somewhat stabilizing near the end of the reference period below the maximum mitigation rate limit; both other regions stay around the 50% line, indicating that they leave the gross CO_2 mitigation for region 1 and set more modest targets, given that the temperature target has already been achieved.

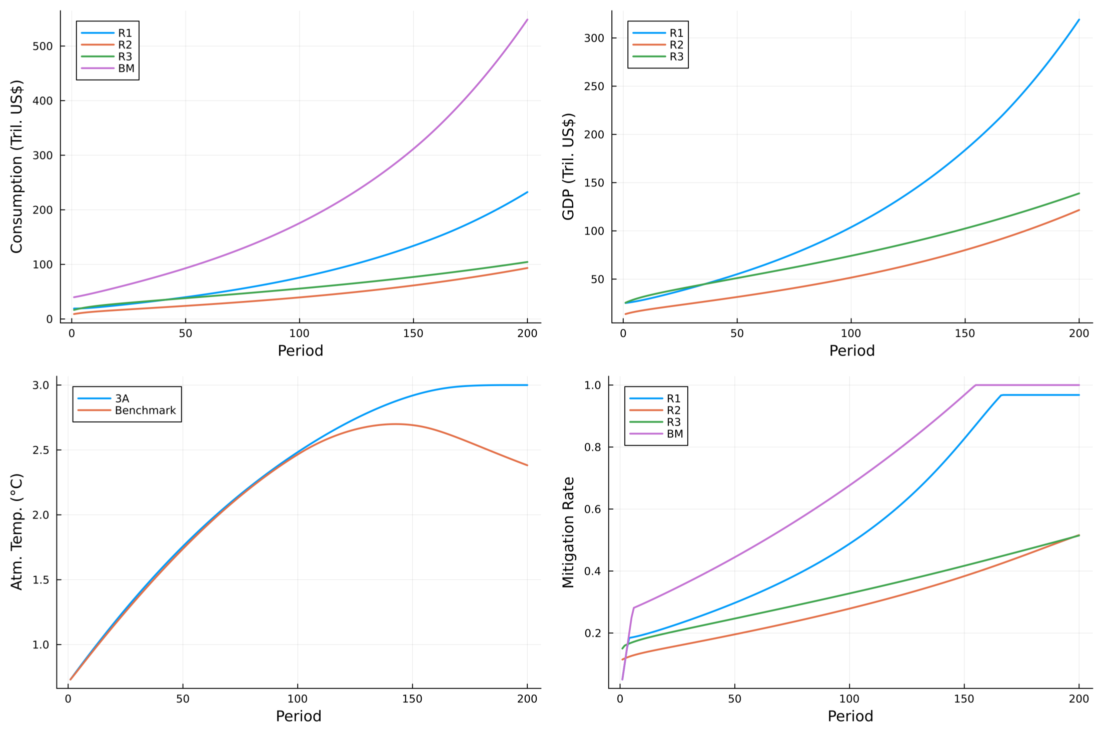

Figure 4.7 – Consumption, GDP, atmospheric temperature (above preindustrial levels) and mitigation rate values for the first 200 year period (from top left to bottom right in reading order, respectively). On the top left and the bottom right corners, an extra purple line is added: the one-agent benchmark model result, for comparison.

Individual emissions stay below the benchmark path, however, cumulative emissions stay at or above the benchmark values - this reflects the higher atmospheric carbon concentration levels and the lower social cost of carbon trajectories. The SCC are all individuals, so each planner sets their own national price, so to speak, in order to enforce the emissions path they agreed on with the other two planners (it is important to remember that, in this model, the social cost of carbon is a byproduct of the optimization process, not a direct tool used by the agents to mitigate; understanding it as the optimal Pigouvian tax on carbon emissions is part of the interpretation of the results). It remains below the benchmark case line for all individuals - as the temperature target is fairly high and closer to a business-as-usual policy scenario temperature result.

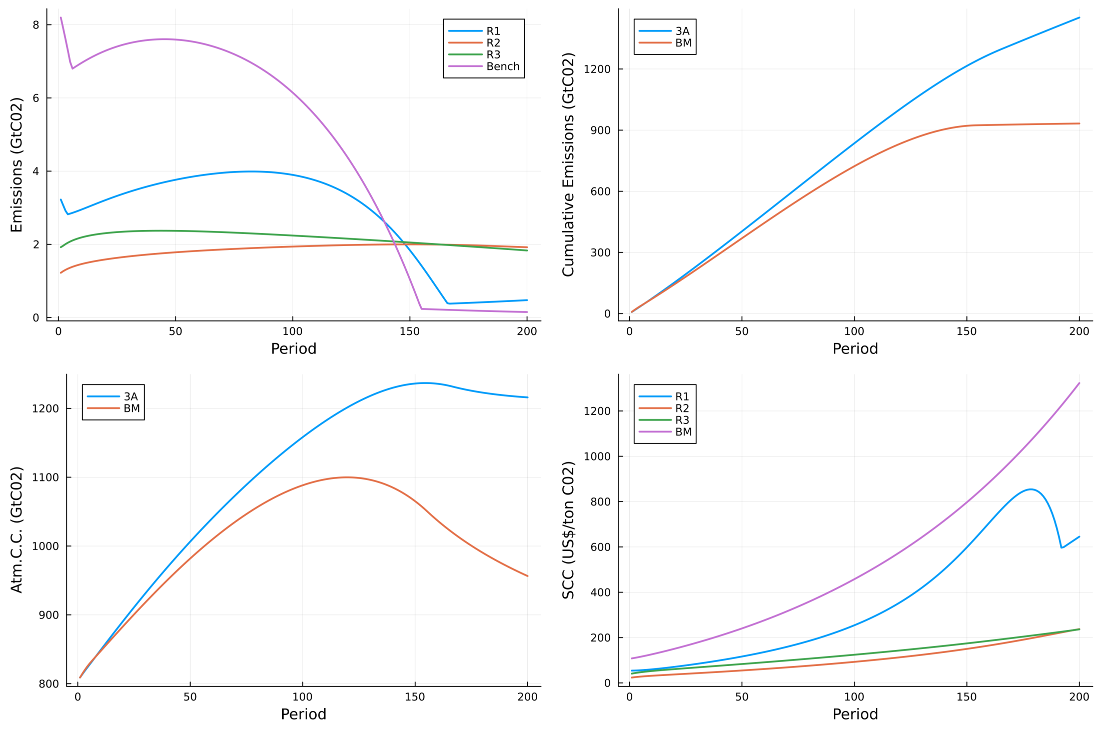

Figure 4.8 – Individual carbon emissions, cumulative carbon emissions, atmospheric carbon concentration and the social cost of carbon for the first 200 year period (from top left to bottom right in reading order, respectively). The one-agent model benchmark series is added to the individual carbon emissions panel for comparison.

4.3.2 2.0ºC Case

For the 2.0ºC temperature anomaly limit case, it's possible to see that, around the year 100, temperature stabilizes at the maximum permitted level. This is also shortly followed by the atmospheric carbon concentration hitting its peak at nearly 1000 $GtCO_2$ and then decreasing over time. Cumulative emissions, although still increasing in the referenced time span, remain all the way below the benchmark no-target levels.

It's interesting to notice that both capital stock and consumption trajectories are almost identical to those obtained by the 3ºC target system. Social planners can successfully mitigate enough while still remaining very close to optimal capital and consumption trajectories due to relatively low damages in the 200-year period. Although abatement costs rise, a fine equilibrium is met between the monetary necessities to mitigate faster and the resulting damages of a lower temperature trajectory. It's possible that this equilibrium would get harder to achieve with more extreme damage functions.

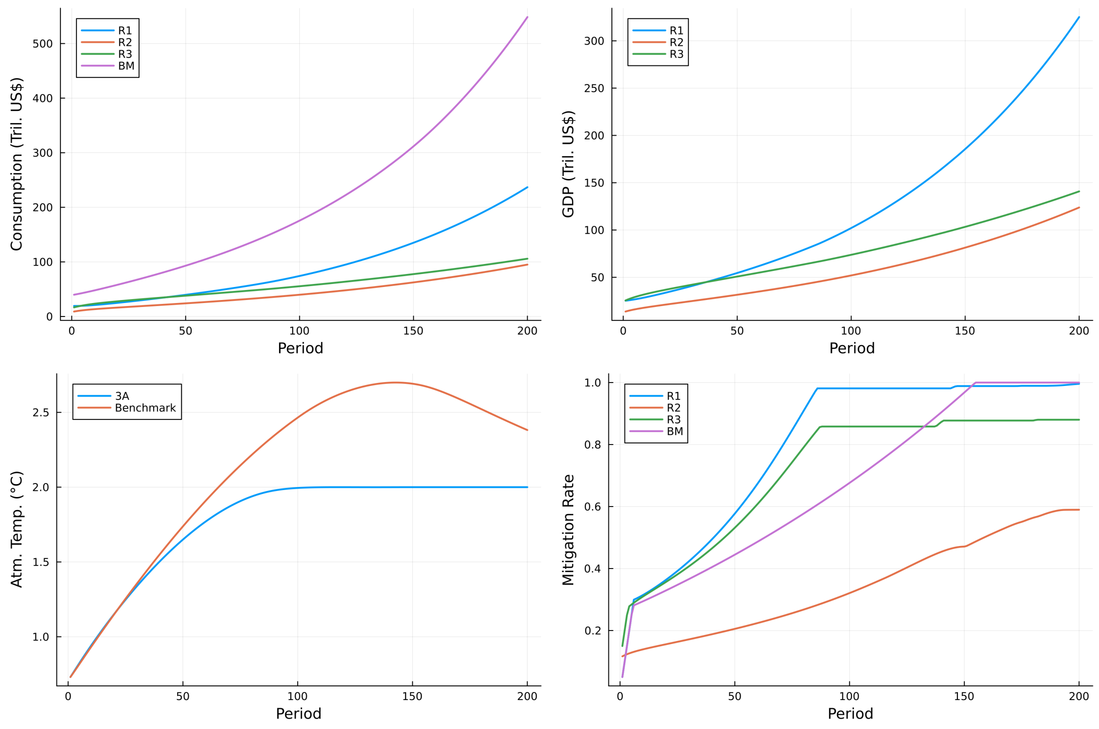

Figure 4.9 – Consumption, GDP, atmospheric temperature (above preindustrial levels) and mitigation rate values for the first 200 year period (from top left to bottom right in reading order, respectively). On the top left and the bottom right corners, an extra purple line is added: the one-agent benchmark model result, for comparison.

One other important figure is the social cost of carbon plot in 4.10: costs of emitting an extra ton of carbon dioxide into the atmosphere rise sharply and in a more aggressive manner than in the 3ºC case. When hitting the year 100, the SCC starts a fast decline as the temperature limit is reached and carbon concentration in the atmosphere starts to slowly diffuse into other layers. This means that the shadow price of emissions drops persistently - as the temperature cap is binding to the optimization - and the shadow price of capital rises with abatement costs reaching the highest values up until this period (mitigation rates, seen in 4.9, hit a high percentage level and then only slowly increase). After that, the optimal tax then surges as the emissions component dominates while the capital component stabilizes in order to respect the temperature cap.

Comparing to the presented benchmark values, regions 1 and 3 indirectly set their social costs significantly higher: at the year 2055, SCCs for these regions are close to $US\$500$ while benchmark values hang around $US\$250$. Region 2, however, stays below the benchmark line for more than the first 50 periods and then surpasses it to reach its peak, with a little delay with respect to the other two regions.

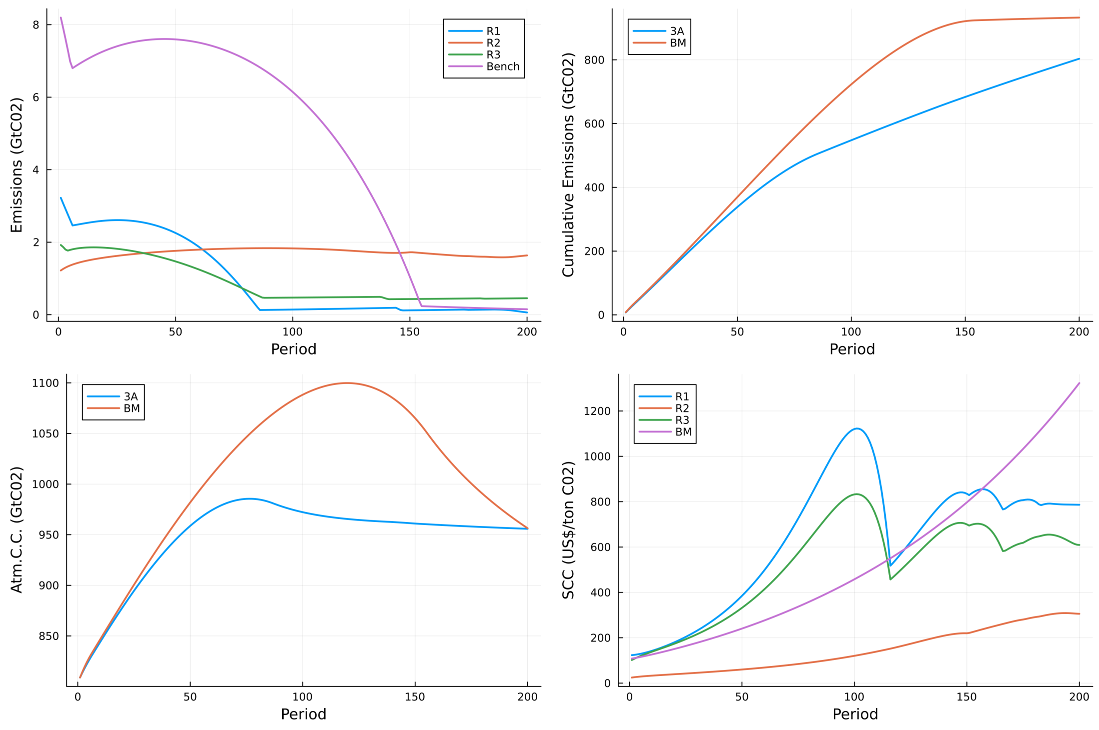

Figure 4.10 – Individual carbon emissions, cumulative carbon emissions, atmospheric carbon concentration and the social cost of carbon for the first 200 year period (from top left to bottom right in reading order, respectively). The one-agent model benchmark series is added to the individual carbon emissions panel for comparison.

4.3.3 1.5ºC Case

As a continuation to diminishing the maximum temperature rise, the system mostly amplifies what had already started doing in the 2ºC case. Now, for instance, more regions are significantly above the benchmark case, indicating that higher efforts are performed to maintain the highest possible average temperature stable at low levels. Consumption and GDP, by the same instrument seen in the previous results, have their trajectory almost identical between the last two cases and the current.

Emissions drop by some amount, in all regions, while the social cost of carbon responds by reaching far greater values: one region touches the $US\$3000$ line before hitting bottom, while two at some point evaluate an extra ton of emissions in $US\$1000$ - a higher price not only even in comparison to almost all past results for this model, but also at the practiced prices currently in most carbon taxes and carbon markets.

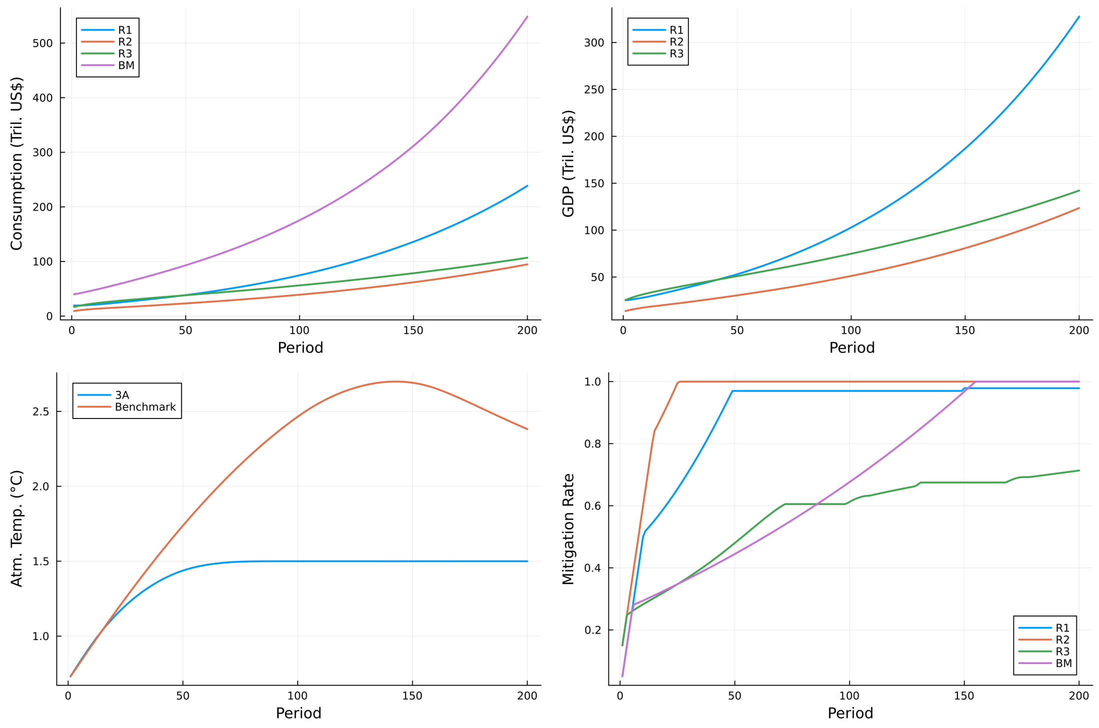

Figure 4.11 – Consumption, GDP, atmospheric temperature (above preindustrial levels) and mitigation rate values for the first 200 year period (from top left to bottom right in reading order, respectively). On the top left and the bottom right corners, an extra purple line is added: the one-agent benchmark model result, for comparison.

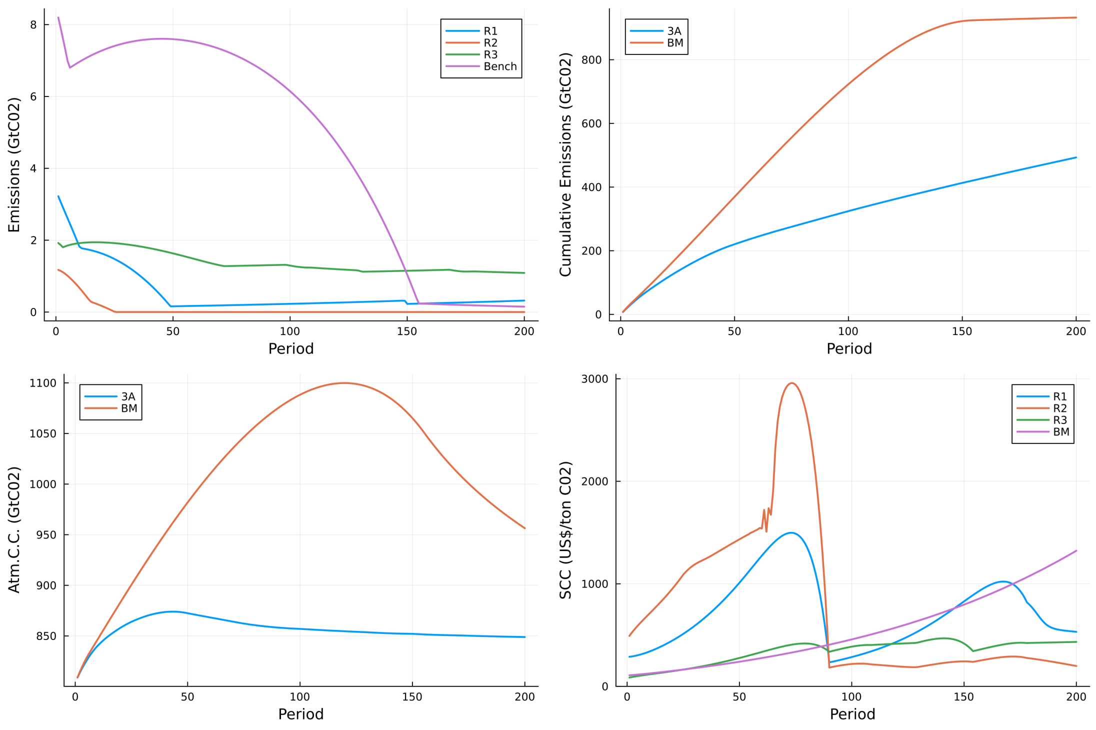

Figure 4.12 – Individual carbon emissions, cumulative carbon emissions, atmospheric carbon concentration and the social cost of carbon for the first 200 year period (from top left to bottom right in reading order, respectively). The one-agent model benchmark series is added to the individual carbon emissions panel for comparison.

4.3.4 Average Results for the 2ºC Case

These are the results from optimizing several times (one hundred, to be specific), then taking the average mitigation rate path for each social planner and simulating the complete climate and economy coupled systems with capital trajectory from the main result shown before (as already mentioned, capital and consumption are almost intact from one run to the other, so it has been understood that any capital path within the same parametrization is sufficient to simulate upon).

As the plots below demonstrated, results seem quite consistent, but with one major difference: as each social planner chooses an average path for mitigation, they no longer collectively see the 2ºC upper bound - as a result, the average global temperature slides off the upper bound.

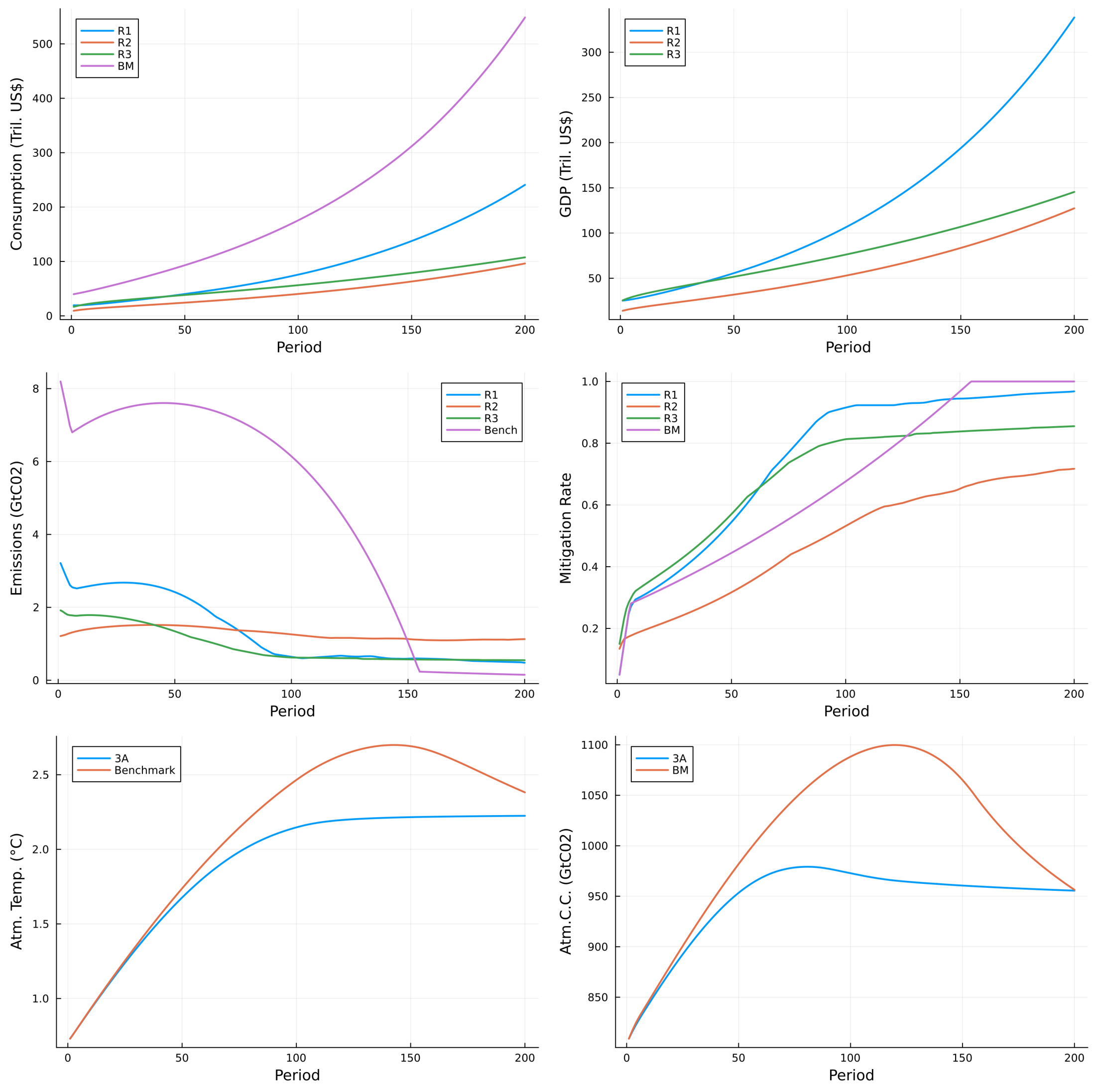

Figure 4.13 – Results for simulating the complete system with individual average (over 100 optimizations) mitigation rate path.

5 Concluding Remarks

The analysis performed here illustrated, briefly, the functioning of integrated assessment models in macroeconomics, looking into more traditional models such as DICE, and more recent ones like DICE-CJL and DSICE. By reproducing the benchmark case - where no temperature target is set, but the social planner feels the effect of polluting through direct damages to her country's GDP -, I could understand and modify the model in order to observe its behaviour with different parameters and number of agents.

Some facts can be extracted from the results presented here: first, I've observed that the intertemporal elasticity of substitution can be a decisive factor in the output of models with such structure, making emissions lower substantially as the IES increases - in other words, as social planners get more balanced between future and current consumption, making the consumption trajectory grow slower over time than with smaller γ (the symbol representing IES in this work).

Second fact is that parametrization is key to determine the long-term behaviour of the agent, with great attention being given to the productivity function and to the carbon intensity of output function. These two have been altered by changing parameters' values (those originally set by (Nordhaus, 1992) and (Cai; Lontzek, 2019)), allowing the visualization of how making output less carbon intense and initial capital smaller results in agents setting their emissions significantly lower, but not necessarily increasing their mitigation rate per period faster.

For multiple agents, the strategic dimension of the problem becomes evident. Each planner optimizes their own path while reacting to—yet influencing—the decisions of the others. This interdependence adds a layer of complexity that has substantial effects on the final equilibrium trajectories. One of the central facts from this section is that the sequencing of decisions, especially in decentralized optimization settings, can produce lasting effects on mitigation strategies. Even when the optimization algorithm randomizes the order in which planners act, the agent that moves first tends to face a structural disadvantage, as it lacks accurate information about the other agents' choices. This often leads to overly aggressive or quite sluggish mitigation, which in turn shapes the reaction paths of the remaining agents.

Despite this asymmetry, the results show a general robustness in the long-term dynamics of the system: when multiple agents interact with each other, aggregated results tend to surpass one-global-agent levels of the two major climate components, atmospheric temperature and carbon concentration. When met with hard restrictions on maximum global warming allowed - exogenously, maybe via global agreements -, mitigation rate levels increase and emissions per region have to drop, maintaining temperature stable at its peak. The social cost of carbon, that here is interpreted as the optimal Pigouvian tax to charge an extra ton of emissions released by industrial activity, is not constant but dynamic and non-linear, signifying that social planners would not only have to charge quite substantial prices (US\$1000 before 2100 in some cases) but also raise and lower those prices for intercalated periods of time.

5.1 Future Work

As DSICE is a stochastic extension of DICE-CJL (and, per transitivity, DICE), a similar, but even simpler, stochastic extension could be performed with the model presented here. The challenge is, as I have faced upon trying and ultimately giving up, that the problem cannot be solved through a NLP-solver alone, and must be inserted into a VFI algorithm. The difficulty comes in two colors: first, the value function iteration process requires that the state space is discretized and the value function is then computed per node in the state space grid per period - this falls under the *curse of dimensionality* issue, and computational time grows in an exponential manner as more nodes or state variables are added; second, each period uses an approximation to the next period's value function (most commonly within the reference models, Chebyshev polynomial approximation) and, as the algorithm goes backward in time, error propagation becomes a real problem, especially when solutions lay on the extremes - boundary solutions are often the case for mitigation rates, for example.

Another possible derivation for this model is the implementation of a dynamic recursive game, where trajectories are not set for all periods at once, but rather agents play each period based solely on past information. This could mean a more realistic policy function, even more for the short term, and more insightful stylized facts derived from the implementation of the model. Also, adding more players is quite a direct and obvious choice for branching out conceptually this version of DICE-CJL; however, the parametrization is quite picky and it shouldn't allow much diversity in this field (for instance, Ipopt.jl abruptly stops finding feasible solutions when tweaking productivity parameters below or above certain thresholds). In addition, inserting more agents into the game makes it harder and more time-consuming to solve, and these two features have to be always on the mind of an integrated assessment modeler.

Even within the conceptual framework of the model presented here, scenarios like uneven temperature targets and economic interconnection between players could be interesting add-ons with more simplicity in implementation and interpretation than previous ones. Hence, DICE and its derivatives still have a lot to offer to our comprehension of long-term climate policy, its possible impacts on the economy and how current policies lack both the strength and the speed most results show to be necessary. I'm thankful to have had the opportunity to contribute to this topic and hope I can further these investigations in the future.

References

BARRAGE, L.; NORDHAUS, W. D. **Policies, Projections, and the Social Cost of Carbon: Results from the DICE-2023 Model**. [*S.l.*], 2023. Cit. on p. 58.

BELLMAN, R. **Dynamic Programming**. Princeton, NJ: Princeton University Press, 1957. Cit. on p. 21.

CAI, Y.; JUDD, K. L.; LONTZEK, T. S. **DSICE: A Dynamic Stochastic Integrated Model of Climate and Economy**. Stanford, CA, 2012. Disponível em: https://ssrn.com/abstract=1992674. Cit. on pp. 19, 22, 33 e 62.

CAI, Y.; LONTZEK, T. S. The social cost of carbon with economic and climate risks. **Journal of Political Economy**, v. 127, n. 6, p. 2684–2734, 2019. DOI 10.1086/701890. Cit. on pp. 19, 21, 22, 25, 28, 50 e 63.

CAMERON, D.; VIAL, A. **An Integrated Assessment Model for Helping the United States Sea Scallop Fishery Plan Ahead for Ocean Acidification and Warming**. Silver Spring, MD, 2019. (NOAA Coastal Science Report, 2019-10). Disponível em: https://repository.library.noaa.gov/view/noaa/22747. Cit. on p. 19.

CASS, D. A. Optimum growth in an aggregative model of capital accumulation. **The Review of Economic Studies**, v. 32, n. 3, p. 233–240, 1965. DOI 10.2307/2296520. Cit. on p. 20.

CHAUBEY, I.; KRUEGER, T.; SRINIVASAN, R.; ARNOLD, J. G.; HAY, L. E. An integrated assessment model for valuing water quality changes in the united states. **Environmental Modelling & Software**, v. 142, p. 105041, 2021. DOI 10.1016/j.envsoft.2021.105041. Cit. on p. 19.

Climate Action Tracker. **Climate Action Tracker**. 2025. https://climateactiontracker.org/. Accessed: 19 July 2025. Cit. on pp. 16 e 17.

Copernicus Climate Change Service (C3S). Global climate highlights 2024. **Copernicus Climate Change Service**, January 2025. Published 10 January 2025. Cit. on p. 14.

Core Writing Team, H. Lee and J. Romero (eds.). Climate change 2023: Synthesis report. contribution of working groups i, ii and iii to the sixth assessment report of the intergovernmental panel on climate change. **IPCC**, Geneva, Switzerland, p. 35–115, 2023. DOI 10.59327/IPCC/AR6-9789291691647. Cit. on pp. 14, 15, 16 e 21.

CRIPPA, M.; GUIZZARDI, D.; PAGANI, F.; BANJA, M.; MUNTEAN, M.; SCHAAF, E.; MONFORTI-FERRARIO, F.; BECKER, W.; QUADRELLI, R.; MARTIN, A. R.; TAGHAVI-MOHARAMLI, P.; KöYKKä, J.; GRASSI, G.; ROSSI, S.; MELO, J.; OOM, D.; BRANCO, A.; SAN-MIGUEL, J.; MANCA, G.; PISONI, E.; VIGNATI, E.; PEKAR, F.

GHG emissions of all world countries. **Publications Office of the European Union**, Luxembourg, n. JRC138862, 2024. DOI 10.2760/4002897. Cit. on p. 15.

DUNNING, I.; HUCHETTE, J.; LUBIN, M. Jump: A modeling language for mathematical optimization. **SIAM Review**, SIAM, v. 59, n. 2, p. 295–320, 2017. DOI 10.1137/15M1020575. Cit. on p. 65.

FISHERVANDEN, K.; WEYANT, J. The evolution of integrated assessment: Developing the next generation of useinspired integrated assessment tools. **Annual Review of Resource Economics**, v. 12, n. 1, p. 471–487, October 2020. DOI 10.1146/annurev-resource-110119-030314. Cit. on p. 19.

HASSLER, J.; KRUSELL, P.; OLOVSSON, C. Suboptimal climate policy: Climate change uncertainty spillover in the macroeconomy. **Journal of the European Economic Association**, v. 19, n. 6, p. 2895–2928, 2021. DOI 10.1093/jeea/jvab048. Cit. on p. 21.

HASSLER, J.; KRUSELL, P.; SMITH, A. Environmental macroeconomics. *In*: **Handbook of Macroeconomics**. Elsevier, 2016. v. 2, cap. Chapter 24, p. 1893–2008. Disponível em: https://EconPapers.repec.org/RePEc:eee:macchp:v2-1893. Cit. on p. 21.

JACOBY, H. D.; REILLY, J. M.; MCFARLAND, J. R.; PALTSEV, S. Technology and technical change in the MIT EPPA model. **Energy Economics**, v. 28, n. 5–6, p. 610–631, 2006. DOI 10.1016/j.eneco.2006.05.014. Cit. on p. 19.

KOOPMANS, T. C. On the concept of optimal economic growth. *In*: FERBER, R. (Ed.). **The Econometric Approach to Development Planning**. Amsterdam: North-Holland Publishing Company, 1965. p. 225–287. Cit. on p. 20.

LEMOINE, D. M.; TRAEGER, C. P. Watch your step: Optimal policy in a tipping climate. **American Economic Journal: Economic Policy**, American Economic Association, v. 6, n. 1, p. 137–166, 2014. DOI 10.1257/pol.6.1.137. Cit. on p. 22.

McKinsey & Company KIMBERLY HENDERSON, M. B. C. **Climate Math: What a 1.5-Degree Pathway Would Take**. 2020. https://www.mckinsey.com/capabilities/sustainability/our-insights/climate-math-what-a-1-point-5-degree-pathway-would-take. Accessed April 30, 2020; McKinsey Quarterly. Cit. on p. 22.

MURPHY, D. J.; NORDHAUS, W. D.; REILLY, J. As bad as it gets: How climate change damage functions affect the social cost of carbon. **Environmental and Resource Economics**, Springer, v. 72, n. 4, p. 983–1004, 2018. DOI 10.1007/s10640-018-0219-y. Cit. on p. 21.

NORDHAUS, W. D. An optimal transition path for controlling greenhouse gases. **Science**, American Association for the Advancement of Science, v. 258, n. 5086, p. 1315–1319, 1992. Cit. on pp. 50 e 58.

NORDHAUS, W. D. **Managing the Global Commons: The Economics of Climate Change**. Cambridge, MA: MIT Press, 1994. Cit. on pp. 23 e 26.

NORDHAUS, W. D. **A Question of Balance: Weighing the Options on Global Warming Policies**. New Haven, CT: Yale University Press, 2008. ISBN 9780300137484. Cit. on pp. 28, 60 e 66.

NORDHAUS, W. D. Estimates of the social cost of carbon: Concepts and results from the dice-2013r model and alternative approaches. **Journal of the Association of Environmental and Resource Economists**, University of Chicago Press, v. 1, n. 1/2, p. 273–312, 2014. Cit. on pp. 58 e 59.

NORDHAUS, W. D. Revisiting the social cost of carbon. **Proceedings of the National Academy of Sciences**, National Academy of Sciences, v. 114, n. 7, p. 1518–1523, 2017. Cit. on p. 58.

NORDHAUS, W. D.; YANG, Z. A regional dynamic general-equilibrium model of alternative climate-change strategies. **The American Economic Review**, American Economic Association, v. 86, n. 4, p. 741–765, 1996. Cit. on p. 20.

RAMSEY, F. P. A mathematical theory of saving. **The Economic Journal**, v. 38, n. 152, p. 543–559, 1928. DOI 10.2307/2224092. Cit. on p. 20.

STERN, N. H. **The Economics of Climate Change: The Stern Review**. Cambridge, UK: Cambridge University Press, 2007. Report commissioned by HM Treasury, UK; originally released 30 October 2006. Cit. on pp. 23 e 59.

TOL, R. S. J.; NARITA, D.; ANTHOFF, D. **Damage Costs of Climate Change through Intensification of Tropical Cyclone Activities: An Application of FUND**. [*S.l.*], 2008. Cit. on p. 21.

VUUREN, D. P. van; KOK, M. T. J.; GIROD, B.; LUCAS, P. L.; VRIES, H. J. M. de; BOUWMAN, L. A. F. Use of integrated assessment models to analyse sustainable development goals, circular economy, and biodiversity. **Environmental Science & Policy**, v. 140, p. 269–281, 2022. DOI 10.1016/j.envsci.2022.03.013. Cit. on p. 19.

WÄCHTER, A.; BIEGLER, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. **Mathematical Programming**, Springer, v. 106, n. 1, p. 25–57, 2006. DOI 10.1007/s10107-004-0559-y. Cit. on p. 65.

WANG, X.; MUELLER, N. D.; GERBER, J. S. Multi-scale analysis of nitrogen loss mitigation in the u.s. corn belt using an integrated economic–ecological framework. **Proceedings of the National Academy of Sciences**, v. 118, n. 32, p. e2024912118, 2021. DOI 10.1073/pnas.2024912118. Cit. on p. 19.

WEITZMAN, M. L. On modeling and interpreting the economics of catastrophic climate change. **Review of Economics and Statistics**, MIT Press, v. 91, n. 1, p. 1–19, 2009. DOI 10.1162/rest.91.1.1. Cit. on pp. 22, 26 e 27.

WEITZMAN, M. L. Ghg targets as insurance against catastrophic climate damages. *In*: HAHN, R.; ULPH, A. (Ed.). **Climate Change and Common Sense: Essays in Honour of Tom Schelling**. Oxford, UK: Oxford University Press, 2011. p. 76–92. Cit. on p. 21.

Appendices

Appendix A – Reference Models

A.1 DICE

Nordhaus introduced DICE in his seminal paper “An Optimal Transition Path for Controlling Greenhouse Gases” (Nordhaus, 1992), published in the Journal of Public Economics in 1992. This was the first fully integrated, intertemporal general-equilibrium model linking a neoclassical growth economy to a simple climate–carbon cycle and damage function. In this original version, the world is treated as a single region, output follows a Cobb–Douglas production function, emissions are proportional to output, abatement costs are quadratic in the abatement rate, and damages are a simple quadratic function of temperature above preindustrial levels, as previously mentioned.

In 1994, Nordhaus published a book called "Managing the Global Commons: The Economics of Climate Change", which presented an expanded discussion of DICE’s structure, parameter calibration and numerical implementation. Through the late 1990s, Nordhaus released updated spreadsheet implementations that recalibrated socio-economic baselines (population, productivity) and climate parameters to match emerging IPCC guidance. These updates refined damage functions, abatement cost curves, and time-step discretization.

Much later, in 2013, Nordhaus released the DICE-2013R (Nordhaus, 2014), which further refined the damage function (drawing on newer impact literature) and explored a range of discount-rate scenarios—including comparisons to the Stern Review’s low pure-time-preference rate. It is important to note that, even though a lot of model features have developed over the years, this version and many others that came before and after are still deterministic - including the more recent versions of DICE-2016 (Nordhaus, 2017) and DICE-2023 (Barrage; Nordhaus, 2023). Now, I’ll present the DICE-2013 equations and solutions. Although an older version of DICE, it’s very well known and basically all of its structure remained in more recent models.

A.1.1 Economic System

As all DICE versions and DICE related models, first and foremost, there is a social welfare function defined as the sum of discounted utilities, with population at time t as a weight, of per capita consumption over the pre-defined number of periods. Hence,

$$W = \sum_{t=1}^{T_{max}} R(t) \cdot u(c(t), L(t))$$

where W represents the social welfare, $R(t) = (1-\rho)^{-t}$ is the future discount factor given the pure rate of social time preference ρ and $u(\cdot,\cdot)$ is the CRRA utility function

$$u(c(t), L(t)) = L(t) \cdot \frac{c(t)^{1-\gamma}}{1-\gamma} \qquad .$$

The parameter γ is interpreted, in (Nordhaus, 2014), as the "generational inequality aversion". In more traditional words, it is the elasticity of marginal utility to consumption. For example, if the parameter is close to 0, then agents find future consumption a close substitute for current consumption and vice versa (a low risk aversion, one may say). Debate on where the parameter's value should reside is large, with a quite substantial contribution in (Stern, 2007).

For economic aggregates, there's the net output $Q(t)$ defined as

$$Q(t) = \Omega(t)[1-\Lambda(t)]Y(t)$$

with $\Omega(t)$ and $\Lambda(t)$ being the damage and abatement-cost functions, respectively. Especifically in DICE-2013, the damage function is defined

$$\Omega(t) = \frac{D(t)}{1+D(t)}$$

and

$$D(t) = \psi_1 T_{AT}(t) + \psi_2 [T_{AT}(t)]^2$$

Here, T_{AT} is the global average atmospheric temperature. Additionally, the abatement cost function has the format

$$\Lambda(t) = \Psi(t)\theta_1(t)\mu(t)^{\theta_2}$$

for a given policy participation rate $\Psi(t)$. The gross output function $Y(t)$ follows a standard Cobb-Douglas production function with diminishing returns to scale:

$$Y(t) = A(t) \cdot K(t)^{\alpha} \cdot L(t)^{1-\alpha}$$

The net output respects the aggregated macroequilibrium identity (without government) $Q(t) = C(t) + I(t)$, with $I(t)$ being gross investment. The economic system is connected, other than temperature, by the aggregated CO_2 global emissions:

$$E(t) = \sigma(t)[1 - \mu(t)]Y(t) + E_{land}(t)$$

A.1.2 Climate System

The climate system is composed of two connected subsystems, the temperature and the carbon concentration feedback cycles. The radiative forcing (in simple terms, the net energy that flows in and out through the Earth's atmosphere) is a link between those and is described as

$$\mathcal{F}(t) = \eta \cdot log_2\big(M_{AT}(t)/M_{AT}(1750)\big) + F_{ex}(t)$$

in which the term $F_{ex}(t)$ represents the non-carbon-related, hence exogenous, forcings of Earth's system (including other GHG emissions). Global average temperature is divided between surface (atmospheric) temperature and ocean (in here, lower ocean) temperature, and is described by the following two equations

$$T_{AT}(t+1) = \hat{\xi}_1 T_{AT}(t) + \hat{\xi}_2 T_{AT}(t) + \xi_1 \cdot \mathcal{F}(t) \tag{A.1}$$

$$\tag{A.2}$$

$$T_(LO)(t) = \hat{\xi}_3 T_{AT}(t) + (1 - \hat{\xi}_3) T_{AT}(t) \tag{A.3}$$

Here, the parameters are rewritten just to represent the equation in a cleaner way, also closer to the posterior versions of reference. Then, there is the carbon concentration cycle with three layers: atmospheric, upper oceans and lower oceans.

$$M_{AT}(t+1) \quad = \quad \phi_{11} M_{AT} + (1 - \phi_{11}) M_{UO} + E(t) \tag{A.4}$$

$$\tag{A.5}$$

$$M_{UO}(t+1) \quad = \quad \phi_{21} M_{AT} + \phi_{22} M_{UO} + \phi_{23} M_{LO} \tag{A.6}$$

$$\tag{A.7}$$

$$M_{UO}(t+1) \quad = \quad \phi_{31} M_{UO} + (1 - \phi_{32}) M_{LO} \tag{A.8}$$

A.2 DICE-CJL

A.2.1 Economic System

As stated in (Nordhaus, 2008) and is commonly seen in neoclassical models, the utility function selected is a Constant Relative Risk Aversion (CRRA) type of function. Also known as the isoelastic function and a member of the Von Neumann-Morgenstern utility function family (as it satisfies completeness, transitivity and continuity), the function applied in both DICE and this work's models is of the form:

$$u(C_t, L_t) = \frac{\left(\frac{C_t}{L_t}\right)^{1-\gamma}}{1-\gamma} * L_t$$

Therefore, the social planner will choose a series $\{c_t\}_{t=1}^{T}$ that satisfies

$$\max_{\{c_t\}_{t=0}^{T}} \sum_{t=0}^{T} R(t) * u(C_t, L_t)$$

where $R(t) = (1+\rho)^{-t}$ and ρ, known as the *pure rate of social time preference*, set in this version of the model to 0.015 - resulting in a $\beta = 0.985$ (utility discount factor described by $\beta = (1+\rho)^{-1}$. One important feature of DICE-2007, and also other versions, is that it's set to work with 10 years time steps. This choice makes it mandatory to elaborate all dynamic transitions for a decade - most of the variables, such as GDP and capital stock, have annual data.
DICE is a dynamic constraint optimization problem, so here are the economic constraints and expressions of the model. First, the standard Cobb-Douglas

$$f(K_t, L_t, A_t) = A_t \cdot K_t^{\alpha} \cdot L_t^{1-\alpha}$$

where both productivity and population (or labor) trends at time t are exogenous - productivity is determined by an initial value and a declining growth rate

$$A_t = A_0 \cdot \exp\left(\frac{\alpha_1(1-e^{-\alpha_2 t})}{\alpha_2}\right)$$

whereas population is determined by an initial value, a constant growth rate and a steady-state value.

$$L_t = 6514e^{-0.35t} + 8600(1-e^{-0.35t})$$

The output function is then used to compute the GDP (in the DICE-2007 model, this is gross world product).

$$Y_t = \Omega(T_{AT,t}) \cdot (1-\Lambda_t) \cdot f(K_t, L_t, A_t)$$

where

$$\Omega(T_{AT,t}) = \frac{1}{1 + \pi_1 \cdot T_{AT,t} + \pi_2 \cdot T_{AT,t}^2}$$

is the damage function, which penalizes the gross product for any temperature deviation from the pre-1900 period (in ºC), and

$$\Lambda_t = \psi_t^{1-\theta_2} \cdot \theta_{1,t} \cdot \mu_t^{\theta_2}$$

is the abatement cost function, composed of the participation rate ψ_t (which translates mathematically the participation of countries or regions in the mitigation target), the mitigation cost coefficient $\theta_{1,t}$, the emissions' mitigation rate μ_t and a cost adjustment parameter θ_2.
Annual industrial carbon emissions are also proportional to the mitigation rate μ_t as the following expression shows

$$E_{Ind,t} = \sigma_t \cdot (1 - \mu_t) \cdot f(K_t, L_t, A_t)$$

Notice that only industrial emissions are accounted for in this model, hence no Land Use, Land-Use Change and Forestry (in short, LULUCF) - leaving out the case of countries that have a large sum or even the majority of their emissions through LULUCF (such as the case for Brazil). At equation A.2.1, the series of parameters σ_t can be seen as a technology factor (Cai; Judd; Lontzek, 2012) and is determined by

$$\sigma_t = \sigma_0 \cdot exp(-0.0073(1 - e^{-0.003t})/0.003)$$

At last, the model has the following constraints in the format of a very simple aggregated output function and capital motion law

$$\begin{aligned} Y_t &= C_t + I_t \\ K_{t+1} &= (1 - \delta) \cdot K_t + I_t \end{aligned}$$

A.2.2 Climate System

There are five variables that compose the entire climate state space - all of them endogenous: two of them are mean global temperature, the atmospheric temperature and the oceanic temperature; the other three are related to carbon concentration, one for the atmosphere (connected to temperature rises), one for the upper ocean (surface) and the last for the lower ocean (deep). The carbon concentration variables evolve over time as in

$$\mathbf{M}_{t+1} = \mathbf{\Phi}_{\boldsymbol{M}} * \mathbf{M}_t + (E_t, 0, 0)^T$$

where

$$\mathbf{\Phi_M} = \begin{bmatrix} 1-\varphi_{12} & \varphi_{21} & 0 \\ \varphi_{12} & 1-\varphi_{21}-\varphi_{23} & \varphi_{32} \\ 0 & \varphi_{23} & 1-\varphi_{32} \end{bmatrix}$$

and

$$E_t = E_{land,t} + E_{ind,t}$$

Here, $E_{land,t}$ is a deterministic function of time. Elements of the Φ_M indicate the rate of diffusion from one layer to the other. On the temperature side,

$$\mathbf{T}_{t+1} = \mathbf{\Phi_T} * \mathbf{T}_t + (\xi_1 \mathcal{F}_t(M_{AT,t}, 0)^T$$

where the radiative forcing follows the same expression, fundamentally, used in DICE

$$\mathcal{F}(M_{AT,t}) = \eta \cdot log_2\big(M_{AT,t}/M_{AT,0}\big) + F_{EX,t}$$

All parameter values for the climate system are identical to those used in DSICE (Cai; Lontzek, 2019).

Appendix B – Computational Aspects

B.1 Julia Solvers

JuMP.jl (Julia for Mathematical Programming) is a powerful optimization framework developed specifically for the Julia language and it is the main tool utilized in this work to solve the giant DICE model - traditionally solved in GAMS or FORTRAN. What makes JuMP special is that it runs natively in Julia, which means it doesn't rely on external software like GAMS, MATLAB, R or even Python, sometimes. Another huge advantage, one derived from Julia as a whole, is that everything happens in the same place: model building, compiling, and running — all directly in Julia, avoiding what developers usually call the "two-language problem". This problem arises specially in scientific programming where, in order to solve a fairly complex problem, researchers write prototypes in code that is pretty fast to both learn and prototype; but, as problems get larger and more intricate, quite slow when running, and then have to glue it to a faster language code - like C, C++ and FORTRAN. However, these former languages are quite hard to learn and difficult to prototype (need compilation, type checking has to be precise), resulting in difficult debugging and computational speed bottlenecks from slower languages. Julia allows fine code optimization still within the language (as even the JIT Julia compiler is written in Julia), while also being quite easy to understand, replicate and scale.

One of JuMP's strengths is how close its syntax is to the kind of math scientists, in general, are used to. For example, when we declare a variable in a constrained dynamic problem like DICE, we can do something like:

```
@variable(model, 0     K[t=1:T]     K_{max})
```

That single line not only sets the bounds for capital over time, but also automatically creates the underlying sparse matrix structure that optimization solvers need. One of Julia's strengths is that it can interpret the structure of your model before actually running it (metaprogramming) by the use of abstract syntax trees — allowing JuMP to break down constraints, understand their mathematical form, and apply automatic differentiation under the hood. This way, even in complex nonlinear problems, it computes exact gradients and Hessians for you, with no need to code derivatives by hand — which is especially useful in climate-economic models like DICE, where second-order precision really matters due to complex, non-linear damage functions.

When it comes to solving, JuMP is solver-agnostic, which means it can connect to different solvers through Julia's MathOptInterface. For most large nonlinear models — such as DICE — a common and solid choice is Ipopt.jl, a Julia wrapper for the well-known Ipopt solver (Wächter; Biegler, 2006). Julia takes full advantage of its underlying performance tools. Linear algebra operations, for instance, rely on BLAS (Basic Linear Algebra Subprograms) under the hood, which are heavily optimized and often multithreaded. And because Julia is designed for type stability — where variables keep predictable types during execution — the compiler can generate highly efficient machine code, minimizing memory allocations and speeding up numerical routines that are common in large-scale dynamic optimization problems.

Ipopt itself is packed with smart techniques that help deal with the kind of problems we face in macro-climate modeling. For instance, it adapts its barrier parameters when getting stuck in non-convex regions — as hard constraints on emissions are set in order to explore extreme case scenarios such as Business-as-usual (BAU). It also uses a filter line-search method to get around situations where constraints are badly behaved or infeasible (Wächter; Biegler, 2006), representing robustness when models are pushed to their limits — like trying to test different mitigation paths under deep uncertainty or boundary optimal controls (all-or-nothing responses).

In terms of performance, JuMP combined with Ipopt has shown to be way faster than older tools, with some benchmarks showing it running 3 to 5 times faster than AMPL for similar-sized models (Dunning; Huchette; Lubin, 2017). That's largely due to Julia-specific advantages, such as avoiding unnecessary data copying, using multithreading to speed up derivative calculations, and parallelizing the linear algebra steps used to solve the Karush-Kuhn-Tucker (KKT) systems that appear in constrained optimization problems. Also, tools like BenchmarkTools.jl help track down performance issues in long simulations - being especially useful when performing GPU computing with CUDA.jl, for example.

B.2 DICE Applicability

The DICE model stands out as one of the most widely used tools to assess the long-run impacts of climate change and carbon mitigation policy. It couples a standard economic growth model with simplified climate dynamics, creating a structure that, although quite simple in theory, becomes rapidly challenging to solve in practice — especially when one accounts for non-linearities, non-convexities, and long time horizons. Through the Julia ecosystem, particularly the combination of JuMP.jl and Ipopt.jl, implementing and solving DICE can become much more straightforward and computationally efficient - which, in reality, means a lot of time saved from waiting optimization rounds. JuMP allows us to write the model in a way that closely mimics its mathematical formulation. Variables like capital, consumption, and emissions can be declared and constrained over time with minimal syntax. This makes it easier to keep the model both readable and aligned with its theoretical underpinnings.

The DICE model's objective — maximizing intertemporal social welfare — includes a logarithmic utility function and discounting, as well as damages caused by rising temperatures. The fact that damages are usually expressed as a quadratic polynomial function of atmospheric temperature is less of a computational burden than the fundamentally large state space, composed of five climate dynamic variables and capital. The climate module is an important feature for assessing the long-run social cost of carbon, as it relies heavily on the atmospheric carbon concentration trajectories over centuries. Temperature and carbon concentration motion laws are embedded into the model as time-recursive constraints, preserving the structure of the original DICE continuous time logic while operating in discrete time (which is already the framework for DICE-CJL).

Solving DICE-CJL with preserved original calibrated parameters is straightforward, the solution is spat out on the seconds range with clear confirmation from Ipopt that a locally optimal solution has been found. Nonetheless, one of this work's goal, initially, was to recalibrate the abatement costs and productivity parameters to countries where climate policy and economic development has been brought up in a different manner than in the USA (most of these parameters are derived from U.S. data and studies upon those, as explicited in (Nordhaus, 2008). The tests on parameter calibration were performed using, primarily, Germany's, Japan's and China's economic and demographic data. Most results came out incoherent, wrong or caused the solver to crash in some way.

To solve the issue of parametrization, I decided to perform small deviations from the standard DICE's model and check how results would alter. Therefore, I will present the final choice for parameters below as well as indicate the reasoning behind it - which, similar to the original DICE, will be rather argumentative than qualitative.

Annexes

Annex A – Julia Codes

Codes to run all one, two and three-agent models are shown below. Some plotting code was suppressed, but it can be easily reproduced based on the images.

A.1 One-Agent Model

```
###############################################################################
# DICE-CJL in Julia/JuMP (Annual steps for 600 years)
###############################################################################

using JuMP
using Ipopt
using Printf
using Plots
using CSV, DataFrames

###############################################################################
# 1) Set the time horizon: T = 1..601 (to have 600 steps)
###############################################################################
TMAX = 601
const T_index = 1:TMAX

###############################################################################
# 2) Parameters (scalars) matching GAMS
###############################################################################
# Preferences
const           = 1.5        # Elasticity of marginal utility
const           = 0.985      # Rate of social time preference

function setParams(sch::String)
    if sch == "BM"
        # Total Factor Productivity (TFP)
        global A0       = 0.02722     # Initial total factor
            productivity
        global  1        = 0.0092      # Initial growth rate for TFP
        global  2        = 0.001       # Decline rate of TFP growth

        # Emissions
        global  0        = 0.13418     # Base level for carbon
            intensity of output
        global d         = -0.00730    # Initial growth of sigma
        global  dcy      = 0.003       # Carbon intensity of output
            decay rate

        global           = 0.3         # Output elasticity of capital
        global K0       = 137.0       # Initial capital, trillion 2005
            USdollars
        global MIU0    = 0.05

        # Population Factor
        global popF = 1

    elseif sch == "SCH1"
        # Total Factor Productivity (TFP)
        global A0      = 0.02722    # Initial total factor productivity
        global  1      = 0.0092     # Initial growth rate for TFP
        global  2      = 0.001      # Decline rate of TFP growth

        # Emissions
        global  0      = 0.13418    # Base level for carbon intensity of output
        global  d      = -0.00730   # Initial growth of sigma
        global  dcy    = 0.003      # Carbon intensity of output decay rate

        global         = 0.3        # Output elasticity of capital
        global K0      = 56.049     # Initial capital, trillion 2005 USdollars
        global MIU0    = 0.1

        # Population Factor
        global popF = 4/7

    elseif sch == "SCH2"
        # Total Factor Productivity (TFP)
        global A0      = 0.02722     # Higher initial productivity (efficient economy)
        global  1      = 0.0092      # Slower initial TFP growth (mature economy)
        global  2      = 0.001       # Slightly faster decay of TFP growth

        # Emissions
        global  0      = 0.10       # Lower carbon intensity (cleaner energy mix)
        global  d      = -0.0080    # Stronger initial decline (aggressive decarbonization)
        global  dcy    = 0.004      # Faster decay rate (tech adoption)

        # Production Function
        global         = 0.26       # Lower capital elasticity (service-oriented)
        global K0      = 28.025     # From PWT data (trillion 2005 US dollars)
        global MIU0    = 0.15

        # Population Factor
        global popF = 2.5/7

    elseif sch == "SCH3"
        # Total Factor Productivity (TFP)
        global A0       = 0.0332     # High initial productivity (technologically advanced)
        global  1        = 0.0055      # Slow TFP growth (mature economy)
        global  2        = 0.0013       # Moderate decay rate

        # Emissions
        global  0        = 0.0885      # Low carbon intensity (energy-efficient)
        global  d        = -0.0082      # Strong decarbonization (policy commitment)
        global  dcy      = 0.0032      # Fast decay rate (tech innovation)

        # Production Function
        global           = 0.28        # Capital elasticity (service-oriented)
        global K0       = 37.3        # Initial capital (trillion 2005 USdollars, OECD 2005)
        global MIU0     = 0.15

        # Population Factor
        global popF = 3/7

    else
        error("Input must be one of the following:
                - BM
                - SCH1
                - SCH2
                - SCH3
        ")
    end

    return nothing
end

setParams("SCH1")

# Population & Technology
const L0        = popF*6514.0 # 2005 population (millions)
const dL        = 0.035       # Growth rate of population
const Lss       = popF*8600.0 # Asymptotic population
const            = 0.10        # Depreciation rate

const Eland0    = 1.1         # Carbon emissions from land in 2005 (GtC/yr)

# Carbon cycle
const MAT0      = 808.9       # Initial value for atmospheric carbon stock (GtC)
```

```
const MUO0     = 1255.0      # Initial value for upper ocean carbon stock (GtC)
const MLO0     = 18365.0     # Initial value for lower ocean carbon stock (GtC)

# Carbon-cycle transition matrix
const  _12      = 0.019
const  _21      = 0.01
const  _23      = 0.0054
const  _32      = 0.00034
const  _11      = 1.0 -  _12
const  _22      = 1.0 -  _21  -  _23
const  _33      = 1.0 -  _32
const  1        = 0.037       # Total radiative forcing level parameter
const  2        = 0.047       # Rate, at time t, of atmospheric temperature decrease due to infrared radiation to space
const  _12      = 0.010
const  _21      = 0.0048

# Climate model
const TOC0     = 0.0068
const TAT0     = 0.7307
const C4       = 0.0048
const           = 3.8

# Climate damages
const A1       = 0.0
const A2       = 0.0028388

# Abatement cost
const EXPCOST2 = 2.8
const PBACK    = 1.17
const BACKRAT  = 2.0
const GBACK    = 0.005
const LIMMIU   = 1.0

################################################################################
# 3) Derived Functions for time-varying parameters
################################################################################
"""
population(t) = Lss - (Lss - L(0))*exp(-dL*(t-1))
"""
function population(t::Int)
    return Lss - (Lss - L0)*exp(-dL*(t-1))
end

"""
tfp(t) = A0*exp( 1 *(1 - exp(- 2 *(t-1))) /  2  )
"""
function tfp(t::Int)
    return A0 * exp( 1 *(1.0 - exp(- 2 *(t-1))) /  2  )
end

```

```
    """
    sigmaFunc(t) =  0 *exp(  d  *(1 - exp(- dcy *(t-1))) /   dcy  )
    """
    function sigmaFunc(t::Int)
        return  0  * exp(  d  *(1.0 - exp(- dcy *(t-1))) /   dcy  )
    end

    """
    forcingOther(t) from FEX0 to FEX1 for first 100 yrs, then 0.36 after that
    """
    function forcingOther(t::Int)
        if (t-1) <= 100
            return -0.06 + 0.0036*(t-1)
        else
            return 0.3
        end
    end

    """
    eTree(t) = Eland0*exp(-0.01*(t-1))
    """
    function eTree(t::Int)
        return Eland0 * exp(-0.01*(t-1))
    end

    """
    cost1(t) = cost1(t) = (PBACK*sigma(t)*(1+exp(-GBACK*(t-1)))/EXPCOST2)*((BACKRAT-1)/BACKRAT)
    """
    function cost1(t::Int)
        return (PBACK * sigmaFunc(t) * (1.0 + exp(-GBACK*(t-1))) / EXPCOST2) * ((BACKRAT - 1.0) / BACKRAT)
    end

    """
    rr(t) = exp(-B_PRSTP*(t-1))
    Discount factor portion for period t
    """
    function rr(t::Int)
        return   ^(t-1)
    end

    function damage(T::Any)
        return (1 + A1*T + A2*(T^2))
    end


    ###############################################################################
    # 4) BUILD THE JuMP MODEL
    ###############################################################################
    model = Model(Ipopt.Optimizer)

    ################################################################################
    # 5) Declare Variables
    ################################################################################
    @variables(model, begin
        # Capital
        0 <= K[t in T_index]
        # Carbon concentrations
        MAT[t in T_index] >= 10
        MU[t in T_index]  >= 100
        ML[t in T_index]  >= 1000

        # Temperatures
        0 <= TATM[t in T_index] <= 4
        -1 <= TOCEAN[t in T_index] <= 20

        # Consumption
        20 <= C[t in T_index]

        # Emission control rate
        0 <= MIU[t in T_index] <= LIMMIU

        # Emissions
        E[t in T_index] >= 0
    end);

    # Single variable for the objective:
    @variable(model, UTILITY)

    ################################################################################
    # 6) Initial Conditions
    ################################################################################
    @constraint(model, Kinit,      K[1]    == K0);
    @constraint(model, MATinit,    MAT[1]  == MAT0);
    @constraint(model, MUinit,     MU[1]   == MUO0);
    @constraint(model, MLinit,     ML[1]   == MLO0);
    @constraint(model, TATMinit,   TATM[1] == TAT0);
    @constraint(model, TOCEANinit, TOCEAN[1] == TOC0);

    @constraint(model, MIUinit, MIU[1] == MIU0);
    @constraint(model, MIUfinal, MIU[end] == LIMMIU);
    #@constraint(model, MIUtrend[t in 100:(TMAX-1)], MIU[t] == LIMMIU);

    @constraint(model, MIUtrendLB[t in 1:(TMAX-1)],
        MIU[t+1] >= MIU[t]
    );

    @constraint(model, MIUtrendUB[t in 1:(TMAX-1)],
        MIU[t+1] <= MIU[t] + 0.05
    );

    ################################################################################
    # 7) Main Model Equations for t = 1..600
    ################################################################################
    # 7.1) Capital transition
    @constraint(model, capitalLaw[t in 1:(TMAX-1)],
        K[t+1] == (1 -   )*K[t] + (
            (1 - cost1(t)*(MIU[t]^EXPCOST2)) *
            (tfp(t)*(population(t)^(1 -   ))*(K[t]^  )) /
            damage(TATM[t])
        ) - C[t]
    );

    # 7.2) Carbon cycle
    @constraint(model, carbonLaw[t in 1:(TMAX-1)],
        MAT[t+1] ==  _11 *MAT[t] +  _21 *MU[t] + E[t]
    );
    @constraint(model, [t in 1:(TMAX-1)],
        ML[t+1]  ==  _33 *ML[t] +  _23 *MU[t]
    );
    @constraint(model, [t in 1:(TMAX-1)],
        MU[t+1]  ==  _12 *MAT[t] +  _22 *MU[t] +  _32 *ML[t]
    );

    # 7.3) Climate equations
    @constraint(model, [t in 1:(TMAX-1)],
        TATM[t+1] == (1- _21 - 2 )*TATM[t] +  _21  * TOCEAN[t] +  1  *
           (  *( log2(MAT[t]/596.4 )) +
                    forcingOther(t))
    );

    @constraint(model, [t in 1:(TMAX-1)],
        TOCEAN[t+1] == TOCEAN[t] +  _12 *(TATM[t] - TOCEAN[t])
    );

    # 7.4) Emissions
    @constraint(model, [t in 1:(TMAX-1)],
        E[t] .== sigmaFunc(t)* (1 - MIU[t]) * (tfp(t)*population(t)^(1 -
             )*K[t]^  ) +
                eTree(t)
    );

    ################################################################################
    # 8) Objective Function: Sum of discounted utilities  (t=1..600)
    ################################################################################

    @expression(model, periodUtilitySum,
        sum(
            ( ( C[t]/population(t) )^(1 - 1/   ) / (1 - 1/   ) ) *
               (rr(t)*population(t))
            for t in 1:(TMAX-1)  # i.e. 1..600
        )
    );

    # Let UTILITY = sum
    @constraint(model, defineUTILITY,
        UTILITY == periodUtilitySum
    );

    # Maximize UTILITY.
    @objective(model, Max, UTILITY)

    ################################################################################
    # 9) Solver Settings and Solve
    ################################################################################

    set_optimizer_attribute(model, "max_iter", 5_000)
    set_optimizer_attribute(model, "tol", 1e-8)

    set_optimizer_attribute(model, "print_level", 5)

    optimize!(model)
    stat = termination_status(model)

    # SCC storage vector
    SCC = zeros(TMAX-1);

    for t in 1:(TMAX-1)
        lambda_mat = shadow_price(carbonLaw[t])
        lambda_cap = shadow_price(capitalLaw[t])
        # standard formula:
        SCC[t] = (lambda_mat / lambda_cap)*1000  # in $/tC
    end

    ################################################################################
    # 10) Print/Inspect Some Results
    ################################################################################

    # Create a time grid over TMAX periods
    tgrid = 1:TMAX-1

    # Extract solution paths from the JuMP model
    K_sol    = [value(K[t]) for t in tgrid];
    C_sol    = [value(C[t]) for t in tgrid];
    MIU_sol  = [value(MIU[t]) for t in tgrid];
    E_sol    = [value(E[t]) for t in tgrid];
    MAT_sol  = [value(MAT[t]) for t in tgrid];
    MU_sol   = [value(MU[t])  for t in tgrid];
    ML_sol   = [value(ML[t])  for t in tgrid];
    TATM_sol = [value(TATM[t]) for t in tgrid];
    TOC_sol  = [value(TOCEAN[t]) for t in tgrid];

    resultDICE = (K = K_sol, C = C_sol, MIU = MIU_sol,
                  E = E_sol, MAT = MAT_sol, MU = MU_sol,
                  ML = ML_sol, TATM = TATM_sol, TOC = TOC_sol,
                  SCC = SCC
    );
```

A.2 Two-Agent Model

```
################################################################################
# TwoAgentOpenLoopDICE.jl
#
# A demonstration of a 2-agent open-loop Nash approach using your DICE-CJL code
# as a base. I do:
#   - Two sets of (K, C, MIU) => K1[t], C1[t], MIU1[t],  K2[t], C2[t], MIU2[t]
#   - One shared climate system (MAT, MU, ML, TATM, TOCEAN)
#   - Region1's model references Region2's emissions E2[t], Region2's references E1[t]
#   - Iterate (E2_current, solve region1) and (E1_current, solve region2)
#     for all t, storing final arrays once converged.
# The final solution arrays can be plotted afterwards.
################################################################################

using JuMP
using Ipopt
using Printf
using CSV, DataFrames
using Plots

################################################################################
# 1) Model horizon and parameters
################################################################################
const TMAX = 601   # i.e. 600 periods
const T_index = 1:TMAX

# Preferences
const     = 1.5
const     = 0.985

# Population & Technology
const L0  = 6514.0
const dL  = 0.035
const Lss = 8600.0

const L0sp1   = 0.6*L0
const L0sp2   = 0.4*L0
const Lsssp1  = 0.6*Lss
const Lsssp2  = 0.4*Lss

const       = 0.10
const  sp1     = 0.30
const  sp2     = 0.28

const A0R1   = 0.02722
const A0R2   = 0.0332

const  1R1    = 0.0092
```

```
    const   1R2    = 0.0055

    const   2R1    = 0.001
    const   2R2    = 0.0013

    # Data extracted from the PWT data tool
    K0 = 137.0

    sch1K0 = 56.0489
    sch2K0 = 35.2

    K0_total = sch1K0 + sch2K0

    const K0R1  = (sch1K0/K0_total)*K0
    const K0R2  = (sch2K0/K0_total)*K0

    # Emissions
    const   0R1     = 0.13418
    const   0R2     = 0.0885

    const d  R1    = -0.0073
    const d  R2    = -0.0082

    const   dcyR1   = 0.003
    const   dcyR2   = 0.0032

    const Eland0 = 1.1

    # Carbon cycle
    const MAT0   = 808.9
    const MUO0   = 1255.0
    const MLO0   = 18365.0

    const   _12     = 0.019
    const   _21     = 0.01
    const   _23     = 0.0054
    const   _32     = 0.00034
    const   _11     = 1.0 -   _12
    const   _22     = 1.0 -   _21   -   _23
    const   _33     = 1.0 -   _32
    const   _12     = 0.010
    const   _21     = 0.0048

    const   1       = 0.037
    const   2       = 0.047
    const TOC0   = 0.0068
    const TAT0   = 0.7307
    const C4     = 0.0048
    const FCO22X = 3.8

    # Damages
    const A1   = 0.0
    const A2   = 0.0028388

    # Abatement cost
    const EXPCOST2 = 2.8
    const PBACK    = 1.17
    const BACKRAT  = 2.0
    const GBACK    = 0.005
    const LIMMIU   = 1.0

    const        = 1e-8

    ##############################################################################
    # 2) Time-varying functions
    ##############################################################################
    """
    population(t) = Lss - (Lss - L0)*exp(-dL*(t-1))
    """
    function populationR1(t::Int)
        return Lsssp1 - (Lsssp1 - L0sp1)*exp(-dL*(t-1))
    end

    function populationR2(t::Int)
        return Lsssp2 - (Lsssp2 - L0sp2)*exp(-dL*(t-1))
    end

    """
    tfp(t) = A0*exp(  1 *(1 - exp(- 2 *(t-1))) /  2  )
    """
    function tfpR1(t::Int)
        # deterministic part of TFP
        return A0R1 * exp( 1R1 *(1.0 - exp(- 2R1 *(t-1))) /  2R1  )
    end

    function tfpR2(t::Int)
        # deterministic part of TFP
        return A0R2 * exp(  1R2 *(1.0 - exp(- 2R2 *(t-1))) /  2R2  )
    end

    """
    sigmaFunc(t) =  0 *exp( d  *(1 - exp(- dcy *(t-1))) /  dcy  )
    """
    function sigmaFuncR1(t::Int)
        return  0R1  * exp( d R1*(1.0 - exp(- dcyR1 *(t-1))) /  dcyR1  )
    end

    function sigmaFuncR2(t::Int)
        return  0R2  * exp( d R2*(1.0 - exp(- dcyR2 *(t-1))) /  dcyR2  )
    end

    """
    eTree(t) = Eland0*exp(-0.01*(t-1))
    """
    function eTree(t::Int)
        return Eland0 * exp(-0.01*(t-1))
    end

    """
    cost1(t) for abatement
    """
    function cost1(t::Int)
        localTerm = (PBACK * sigmaFunc(t) * (1.0 + exp(-GBACK*(t-1))) / EXPCOST2)
        return localTerm * ((BACKRAT - 1.0) / BACKRAT)
    end

    function cost1R1(t::Int)
        localTerm = (PBACK * sigmaFuncR1(t) * (1.0 + exp(-GBACK*(t-1))) / EXPCOST2)
        return localTerm * ((BACKRAT - 1.0) / BACKRAT)
    end

    function cost1R2(t::Int)
        localTerm = (PBACK * sigmaFuncR2(t) * (1.0 + exp(-GBACK*(t-1))) / EXPCOST2)
        return localTerm * ((BACKRAT - 1.0) / BACKRAT)
    end

    """
    rr(t) =   ^(t-1)
    """
    function rr(t::Int)
        return   ^(t-1)
    end

    function damage(T::Any)
        return (1 + A1*T + A2*(T^2))
    end

    function forcingOther(t::Int)
        if (t-1) <= 100
            return -0.06 + 0.0036*(t-1)
        else
            return 0.3
        end
    end

    function buildInitialGuess(Region::String)
        if Region == "R1"
            data = CSV.read("D://Mestrado//DICECJL//Results//1 Agent//resultUSA1dot5.csv", DataFrame)

            DICE = vcat(data, data[end:end, :])
            return NamedTuple((
                :K => DICE.K,
                :C => DICE.C,
                :MIU => DICE.MIU,
                :E => DICE.E,
                :MAT => DICE.MAT,
                :MU => DICE.MU,
                :ML => DICE.ML,
                :TATM => DICE.TATM,
                :TOC => DICE.TOC
            ));
        elseif Region == "R2"
            data = CSV.read("D://Mestrado//DICECJL//Results//1
               Agent//resultGER1dot5.csv", DataFrame)

            DICE = vcat(data, data[end:end, :])
            return NamedTuple((
                :K => DICE.K,
                :C => DICE.C,
                :MIU => DICE.MIU,
                :E => DICE.E,
                :MAT => DICE.MAT,
                :MU => DICE.MU,
                :ML => DICE.ML,
                :TATM => DICE.TATM,
                :TOC => DICE.TOC
            ));
        else
            error("You must enter either R1 or R2!")
        end
    end
    ##############################################################################
    # 3) Build Region1's model (open-loop) given E2(t)
    ##############################################################################
    """
    solveRegion1(E2_guess) -> returns arrays:
      K1[t], C1[t], MIU1[t], E1[t], plus climate states MAT, MU, ML, TATM,
         TOCEAN

    Open-loop approach: region1 solves one big model from t=1..600,
    treating E2_guess[t] as exogenous for the entire horizon.
    """

    function solveRegion1(E2_guess::Vector{Float64}, oldSolR1::NamedTuple)
        modelR1 = Model(Ipopt.Optimizer)

        @variables(modelR1, begin
            # Capital
            K1[t in T_index]  >= 0
            # Carbon concentrations
            MAT[t in T_index] >= 10
            MU[t in T_index]  >= 100
            ML[t in T_index]  >= 1000

            # Temperatures
            #0 <= TATM[t in T_index] <= 20
            -1 <= TOCEAN[t in T_index] <= 20

            # Consumption
            0.5 <= C1[t in T_index]

            # Emission control rate
            0 <= MIU1[t in T_index] <= LIMMIU

            # Emissions
            E1[t in T_index] >= 0
        end);

        @variable(modelR1, TATM[t in 1:TMAX],
                  lower_bound = 0.0,
                  upper_bound = 2.0
        )

            # Warm start by setting initial guesses from oldSolR1
        if oldSolR1 !== nothing
            for t in T_index
                set_start_value(K1[t], oldSolR1.K[t])
                set_start_value(C1[t], oldSolR1.C[t])
                set_start_value(MIU1[t], oldSolR1.MIU[t])
                set_start_value(E1[t], oldSolR1.E[t])
            end
        end

        # initial conditions
        @constraint(modelR1, K1_init,      K1[1] == K0R1)
        @constraint(modelR1, MAT_init,     MAT[1] == MAT0)
        @constraint(modelR1, MU_init,      MU[1]  == MUO0)
        @constraint(modelR1, ML_init,      ML[1]  == MLO0)
        @constraint(modelR1, TATM_init,    TATM[1]== TAT0)
        @constraint(modelR1, TOCEAN_init,  TOCEAN[1] == TOC0)

        @constraint(modelR1, MIUinit, MIU1[1] <= 0.05);
        #@constraint(modelR1, MIUfinal, MIU1[end] == LIMMIU);

        @constraint(modelR1, [t in 1:(TMAX-1)],
                    MIU1[t+1] >= MIU1[t]
        )

        @constraint(modelR1, [t in 1:(TMAX-1)],
                MIU1[t+1] <= MIU1[t] + 0.05
        )

        # capital transitions

        @constraint(modelR1, capitalLaw[t in 1:(TMAX-1)],
            K1[t+1] == (1 -    )*K1[t] + (
                (1 - cost1R1(t)*((MIU1[t])^EXPCOST2)) *
                (tfpR1(t)*(populationR1(t)^(1 -  sp1  ))*(K1[t]^ sp1  )) /
                damage(TATM[t])
            ) - C1[t]
        )


        # region1's own emissions

        @constraint(modelR1, [t in 1:(TMAX-1)],
            E1[t] == (sigmaFuncR1(t)*(1 - (MIU1[t])) *
                tfpR1(t)*(populationR1(t)^(1 -  sp1 ))*(K1[t]^ sp1 ))
        )


        # carbon cycle & climate eqns using E1[t] + E2_guess[t]

        @constraint(modelR1, carbonLaw[t in 1:(TMAX-1)],
            MAT[t+1] ==  _11 *MAT[t] +  _21 *MU[t] + E1[t] + E2_guess[t] +
                eTree(t)
        )
        @constraint(modelR1, [t in 1:(TMAX-1)],
            MU[t+1] ==  _12 *MAT[t] +  _22 *MU[t] +  _32 *ML[t]
        )
        @constraint(modelR1, [t in 1:(TMAX-1)],
            ML[t+1] ==  _33 *ML[t] +  _23 *MU[t]
        )
        @constraint(modelR1, [t in 1:(TMAX-1)],
            TATM[t+1] == (1- _21 - 2 )*TATM[t] +
                             _21 *TOCEAN[t] +
                             1 * (FCO22X*( log2(MAT[t]/596.4))+
                    forcingOther(t))
        )
        @constraint(modelR1, [t in 1:(TMAX-1)],
            TOCEAN[t+1] == TOCEAN[t] +  _12 *(TATM[t] - TOCEAN[t])
        )


        # objective = sum_{t=1..600} utility
        @variable(modelR1, UTILITY_R1)
        @expression(modelR1, periodUtilR1,
            sum(
                ( (C1[t]/populationR1(t))^(1-1/   ) / (1-1/   ) ) *
                    (rr(t)*populationR1(t))
                for t in 1:(TMAX-1)
            )
        )
        @constraint(modelR1, defineUtilityR1, UTILITY_R1 == periodUtilR1)
        @objective(modelR1, Max, UTILITY_R1)

        set_optimizer_attribute(modelR1, "max_iter", 10_000)
        set_optimizer_attribute(modelR1, "tol", 1e-6)
        set_silent(modelR1)

        optimize!(modelR1)

        if termination_status(modelR1) !== (LOCALLY_SOLVED)
            println("Termination status:", termination_status(modelR1))
        end

        # extract solution
        K1_sol = [value(K1[t])   for t in T_index]
        C1_sol = [value(C1[t])   for t in T_index]
        M1_sol = [value(MIU1[t]) for t in T_index]
        E1_sol = [value(E1[t])   for t in T_index]

        MAT_sol= [value(MAT[t])   for t in T_index]
        MU_sol = [value(MU[t])    for t in T_index]
        ML_sol = [value(ML[t])    for t in T_index]
        TATM_sol=[value(TATM[t])  for t in T_index]
        TOC_sol=[value(TOCEAN[t]) for t in T_index]

        SCC = zeros(TMAX-1)
        for t in 1:(TMAX-1)
            lambda_mat = shadow_price(carbonLaw[t])
            lambda_cap = shadow_price(capitalLaw[t])
            # standard formula:
            SCC[t] = (lambda_mat / lambda_cap)*1000  # in $/tC
        end

        println("Solved Region 1!
        Termination_status:", termination_status(modelR1))

        objVal = objective_value(modelR1)
        modelR1 = nothing
        GC.gc(true)

        return (
          K = K1_sol, C = C1_sol, MIU = M1_sol, E = E1_sol,
          MAT = MAT_sol, MU = MU_sol, ML = ML_sol, TATM = TATM_sol, TOC =
              TOC_sol,
          objR1 = objVal, SCC1 = SCC
        )
    end


    ################################################################################
    # 4) Build Region2's model (open-loop) given E1(t)
    ################################################################################
    """
    solveRegion2(E1_guess) => returns arrays:
      K2[t], C2[t], MIU2[t], E2[t], plus climate states
    """

    function solveRegion2(E1_guess::Vector{Float64}, oldSolR2::NamedTuple)
        modelR2 = Model(Ipopt.Optimizer)

        @variables(modelR2, begin
            # Capital
            K2[t in T_index] >= 0
            # Carbon concentrations
            MAT[t in T_index] >= 10
            MU[t in T_index]  >= 100
            ML[t in T_index]  >= 1000

            # Temperatures
            #0 <= TATM[t in T_index] <= 20
            -1 <= TOCEAN[t in T_index] <= 20

            # Consumption
            0.5 <= C2[t in T_index]

            # Emission control rate
            0 <= MIU2[t in T_index] <= LIMMIU

            # Emissions
            E2[t in T_index] >= 0
        end)

        @variable(modelR2, TATM[t in 1:TMAX],
                  lower_bound = 0.0,
                  upper_bound = 2.0
        )

            # Warm start from oldSolR2
        if oldSolR2 !== nothing
            for t in T_index
                set_start_value(K2[t], oldSolR2.K[t])
                set_start_value(C2[t], oldSolR2.C[t])
                set_start_value(MIU2[t], oldSolR2.MIU[t])
                set_start_value(E2[t], oldSolR2.E[t])

            end
        end

        # initial
        @constraint(modelR2, K2_init,     K2[1] == K0R2)
        @constraint(modelR2, MAT_init,    MAT[1] == MAT0)
        @constraint(modelR2, MU_init,     MU[1]  == MUO0)
        @constraint(modelR2, ML_init,     ML[1]  == MLO0)
        @constraint(modelR2, TATM_init,   TATM[1]== TAT0)
        @constraint(modelR2, TOCEAN_init, TOCEAN[1] == TOC0)

        @constraint(modelR2, MIU_init, MIU2[1] <= 0.15)
        #@constraint(modelR2, MIUfinal, MIU2[end] == 1.0);

        @constraint(modelR2, [t in 1:(TMAX-1)],
                    MIU2[t+1] >= MIU2[t]
        )

        @constraint(modelR2, [t in 1:(TMAX-1)],
                    MIU2[t+1] <= MIU2[t] + 0.05
        )

        # capital transitions

        @constraint(modelR2, capitalLaw[t in 1:(TMAX-1)],
            K2[t+1] == (1 -   )*K2[t] + (
                (1 - cost1R2(t)*(MIU2[t]^EXPCOST2)) *
                (tfpR2(t)*(populationR2(t)^(1 -  sp2 ))*(K2[t]^ sp2 )) /
                damage(TATM[t])
            ) - C2[t]
        )


        # region2's own emissions

        @constraint(modelR2,  [t in 1:(TMAX-1)],
            E2[t] == (sigmaFuncR2(t)*(1 - MIU2[t]) *
            tfpR2(t)*(populationR2(t)^(1 -  sp2 ))*(K2[t]^ sp2 ))
        )


        # carbon cycle & climate eqns with E1_guess[t] + E2[t]
        @constraint(modelR2, carbonLaw[t in 1:(TMAX-1)],
            MAT[t+1] ==  _11 *MAT[t] +  _21 *MU[t] + E1_guess[t] + E2[t] +
                eTree(t)
        )
        @constraint(modelR2, [t in 1:(TMAX-1)],
            MU[t+1] ==  _12 *MAT[t] +  _22 *MU[t] +  _32 *ML[t]
        )
        @constraint(modelR2, [t in 1:(TMAX-1)],
            ML[t+1] ==  _33 *ML[t] +  _23 *MU[t]
        )
        @constraint(modelR2, [t in 1:(TMAX-1)],
            TATM[t+1] == (1- _21 - 2 )*TATM[t] +
                             _21 *TOCEAN[t] +
                             1 * (FCO22X*( log2(MAT[t]/596.4))+
                    forcingOther(t))
        )
        @constraint(modelR2, [t in 1:(TMAX-1)],
            TOCEAN[t+1] == TOCEAN[t] + C4*(TATM[t] - TOCEAN[t])
        )


        # objective
        @variable(modelR2, UTILITY_R2)
        @expression(modelR2, periodUtilR2,
            sum(
                ( ( C2[t]/populationR2(t) )^(1-1/   ) / (1-1/   ) ) *
                    (rr(t)*populationR2(t)) # removed *populationR2(t)
                for t in 1:(TMAX-1)
            )
        )
        @constraint(modelR2, defineUtilityR2, UTILITY_R2 == periodUtilR2)
        @objective(modelR2, Max, UTILITY_R2)
        set_optimizer_attribute(modelR2, "max_iter", 10_000)
        set_optimizer_attribute(modelR2, "tol", 1e-6)

        set_silent(modelR2)

        optimize!(modelR2)

        if termination_status(modelR2) !== (LOCALLY_SOLVED)
            println("Termination status:", termination_status(modelR2))
        end

        # extract
        K2_sol = [value(K2[t])   for t in T_index]
        C2_sol = [value(C2[t])   for t in T_index]
        M2_sol = [value(MIU2[t]) for t in T_index]
        E2_sol = [value(E2[t])   for t in T_index]

        MAT_sol= [value(MAT[t])   for t in T_index]
        MU_sol = [value(MU[t])    for t in T_index]
        ML_sol = [value(ML[t])    for t in T_index]
        TATM_sol=[value(TATM[t])  for t in T_index]
        TOC_sol=[value(TOCEAN[t]) for t in T_index]

        SCC = zeros(TMAX-1)
        for t in 1:(TMAX-1)
            lambda_mat = shadow_price(carbonLaw[t])
            lambda_cap = shadow_price(capitalLaw[t])
            # standard formula:
            SCC[t] = (lambda_mat / lambda_cap)*1000  # in $/tC
        end

        println("Solved Region 2!
        Termination_status:", termination_status(modelR2))

        objVal = objective_value(modelR2)
        modelR2 = nothing
        GC.gc(true)
        return (
            K = K2_sol, C = C2_sol, MIU = M2_sol, E = E2_sol,
          MAT = MAT_sol, MU = MU_sol, ML = ML_sol, TATM = TATM_sol, TOC =
             TOC_sol,
          objR2 = objVal, SCC2 = SCC
          )
    end

    ################################################################################
    # 5) Outer iteration for open-loop Nash
    ################################################################################
    """
    solveTwoAgentOpenLoop(; maxIter=50, tol=1e-3)
    returns a NamedTuple with final arrays:
      K1, C1, M1, E1, K2, C2, M2, E2,
      plus the climate states MAT, MU, ML, TATM, TOCEAN

    Algorithm:
     - Start with E2_current guess
     - solve region1 => E1(t)
     - solve region2 => E2(t)
     - do a  G a u s s Seidel  partial update
     - repeat until converge
    Finally returns the last solution from region1 and region2.
    """
    function solveTwoAgentOpenLoop(; maxIter=150, tol=1e-4)
        bestR1 = nothing
        bestR2 = nothing

        dataR1 = buildInitialGuess("R1")
        dataR2 = buildInitialGuess("R2")

        # 1) initial guess for E2
        #E1_current = ones(TMAX-1)
        #E2_current = ones(TMAX-1)
        bestR1 = dataR1
        bestR2 = dataR2

        E1_current = dataR1.E #; push!(E1_guess, 0.0)
        E2_current = dataR2.E #; push!(E2_current, 0.0)

        for iter in 1:maxIter
            @printf("\n===== Iteration %d =====\n", iter)

            println("=== SOLVING REGION 2 ===")
            # region2 best response to E1_sol
            @time solR2 = solveRegion2(E1_current, bestR2)
            E2_sol = solR2.E
            bestR2 = solR2

            println("=== SOLVING REGION 1 ===")
            # region1 best response to E2_current
            @time solR1 = solveRegion1(E2_current, bestR1)
            E1_sol = solR1.E
            bestR1 = solR1

            println("=== CHECK CONVERGENCE ===")
            # check difference in E2
            diffE1 = maximum(abs.(E1_sol .- E1_current))
            diffE2 = maximum(abs.(E2_sol .- E2_current))
            @printf("   objR1=%.3f   objR2=%.3f  diffE1=%.6f
                diffE2=%.6f\n", solR1.objR1, solR2.objR2, diffE1, diffE2)

            #bestR1 = (K1_sol, C1_sol, M1_sol, E1_sol, MAT1_sol, MU1_sol,
                ML1_sol, TATM1_sol, TOC1_sol, SCC1)
            #bestR2 = (K2_sol, C2_sol, M2_sol, E2_sol, MAT2_sol, MU2_sol,
                ML2_sol, TATM2_sol, TOC2_sol, SCC2)

            if (diffE1 < tol) && (diffE2 < tol)
```

```
                println("Converged after $iter iterations.")
                break
            end

            #oldSolR1 = bestR1
            #oldSolR2 = bestR2

               = 0.2
            if diffE1     1.0
                # partial update for E2
                E1_current .=    .* E1_sol .+ (1-  ) .* E1_current
            else
                E1_current .= 0.4 .* E1_sol .+ 0.6 .* E1_current
            end

            if diffE2     1.0
                # partial update for E2
                E2_current .=    .* E2_sol .+ (1-  ) .* E2_current
            else
                E2_current .= 0.4 .* E2_sol .+ 0.6 .* E2_current
            end
        end

        # unify the final arrays from region1 + region2
        # we take region1's climate states as the final reference, or region2's.
        # We'll just pass them as well for reference.

        return (
            region1 = bestR1,   # (K1, C1, M1, E1, MAT, MU, ML, TATM, TOC)
            region2 = bestR2    # (K2, C2, M2, E2, ...)
        )
    end

    ################################################################################
    # 6) Solve and output
    ################################################################################
    function main()
        finalSol = solveTwoAgentOpenLoop()
        println("\n=== DONE. We have final solution arrays. ===")

        region1Sol = finalSol.region1
        region2Sol = finalSol.region2

        # Unpack region1
        K1_sol   = region1Sol[1]
        C1_sol   = region1Sol[2]
        M1_sol   = region1Sol[3]
        E1_sol   = region1Sol[4]
        SCC1     = region1Sol[11]

        # region2
        K2_sol   = region2Sol[1]
        C2_sol   = region2Sol[2]
        M2_sol   = region2Sol[3]
        E2_sol   = region2Sol[4]
        SCC2     = region2Sol[11]

        # climate from region2's perspective in region2Sol[5..9]

        TATM_sol = (region1Sol[8] .+ region2Sol[8]) ./ 2
        MAT_sol  = region1Sol[5]  # climate from region1's perspective
        MU_sol   = region1Sol[6]
        ML_sol   = region1Sol[7]
        TATM1_sol = region1Sol[8]
        TATM2_sol = region2Sol[8]
        TOC_sol  = region1Sol[9]

        # We can now do any post-processing or plotting. For now, let's
           just show a few final values:
        println("At final iteration, for region1, K1[TMAX]= ",
           K1_sol[end], " M1[TMAX]= ", M1_sol[end])
        println("At final iteration, for region2, K2[TMAX]= ",
           K2_sol[end], " M2[TMAX]= ", M2_sol[end])
        return (K1_sol = K1_sol, C1_sol = C1_sol, M1_sol = M1_sol,
                E1_sol = E1_sol, K2_sol = K2_sol, C2_sol = C2_sol,
                M2_sol = M2_sol, E2_sol = E2_sol, TATM_sol = TATM_sol,
                MAT_sol = MAT_sol, MU_sol = MU_sol, ML_sol = ML_sol,
                TOC_sol = TOC_sol, TATM1_sol = TATM1_sol, TATM2_sol =
                   TATM2_sol,
                SCC1 = SCC1, SCC2 = SCC2)
    end

    finalSol = main()


    ###############################################################################
    # 7) Results
    ###############################################################################
    tgrid = 1:600

    function local_damageFactor(x::T...) where {T<:Real}
        @assert length(x) == 1 "local_damageFactor expects one argument"
        t_val = x[1]
        return one(t_val) / (one(t_val) + A1*t_val + A2*(t_val^2))
    end

    GDP1_sol = [ local_damageFactor(finalSol.TATM_sol[t]) * (1 -
       cost1R1(t)*(finalSol.M1_sol[t]^EXPCOST2)) *
                (tfpR1(t)*(populationR1(t)^(1 -
                    sp1 ))*(finalSol.K1_sol[t]^ sp1 ))
                for t in tgrid ]

    GDP2_sol = [ local_damageFactor(finalSol.TATM_sol[t]) * (1 -
       cost1R2(t)*(finalSol.M2_sol[t]^EXPCOST2)) *
```

```
            (tfpR2(t)*(populationR2(t)^(1 - sp2 ))*(finalSol.K2_sol[t]^ sp2 ))
            for t in tgrid ]

finalSolDF = DataFrame(
    K1 = finalSol.K1_sol[tgrid], C1 = finalSol.C1_sol[tgrid], M1 = finalSol.M1_sol[tgrid], E1 = finalSol.E1_sol[tgrid],
    K2 = finalSol.K2_sol[tgrid], C2 = finalSol.C2_sol[tgrid], M2 = finalSol.M2_sol[tgrid], E2 = finalSol.E2_sol[tgrid],
    GDP1 = GDP1_sol[tgrid], GDP2 = GDP2_sol[tgrid],
    TAT = finalSol.TATM_sol[tgrid], TOC = finalSol.TOC_sol[tgrid],
    MAT = finalSol.MAT_sol[tgrid], MU = finalSol.MU_sol[tgrid], ML = finalSol.ML_sol[tgrid],
    SCC1 = finalSol.SCC1[tgrid], SCC2 = finalSol.SCC2[tgrid]
)
```

A.3 Three-Agent Model

```
  """
###############################################################################
# ThreeAgentOpenLoopDICE.jl
#
# A demonstration of a 3-agent open-loop Nash approach using your DICE-CJL code
# as a base. I do:
#   - Three sets of (K, C, MIU)
#   - One shared climate system (MAT, MU, ML, TATM, TOCEAN)
#   - Region i's model reference the other two regions' emissions, for i in [1,2,3]
#   - Iterate (Solve Region i for Ej_current, for j =/= i) for all t, storing final arrays once converged.
# The final solution arrays are plotted afterwards.
###############################################################################
"""

using JuMP
using Ipopt
using Printf
using CSV, DataFrames
using Random, Statistics
using Plots

"""
#################################################################################
#################################################################################
##################################### 1) MODEL PARAMETERS #####################################
#################################################################################
#################################################################################
"""

```

```
    ################################################################################
    # 1) Model horizon and parameters
    ################################################################################
    const TMAX = 601   # i.e. 600 periods
    const T_index = 1:TMAX

    # Preferences
    const     = 1.5
    const     = 0.985

    # Population & Technology
    const L0  = 6514.0
    const dL  = 0.035
    const Lss = 8600.0

    const L0R1  = (3/7)*L0
    const L0R2  = (1.5/7)*L0
    const L0R3  = (2.5/7)*L0

    const LssR1 = (3/7)*Lss
    const LssR2 = (1.5/7)*Lss
    const LssR3 = (2.5/7)*Lss

    const dLR1  = 0.035
    const dLR2  = 0.035
    const dLR3  = 0.035

    const A0    = 0.02722

    const A0R1  = 0.02722
    const A0R2  = 0.0295
    const A0R3  = 0.0332

    const  1R1    = 0.0092
    const  1R2    = 0.0075
    const  1R3    = 0.0055

    const  2R1    = 0.001
    const  2R2    = 0.0011
    const  2R3    = 0.0013

    const         = 0.10

    const  R1   = 0.30
    const  R2   = 0.26
    const  R3   = 0.28

    K0 = 137.0

    # Data extracted from the PWT data tool
    sch1K0 = 56.0489
    sch2K0 = 16.430
    sch3K0 = 35.2

    K0_total = sch1K0 + sch2K0 + sch3K0


    const K0R1  = (sch1K0/K0_total)*K0 #56.049
    const K0R2  = (sch2K0/K0_total)*K0 #28.025
    const K0R3  = (sch3K0/K0_total)*K0 #37.3

    const  0R1     = 0.13418
    const  0R2     = 0.10
    const  0R3     = 0.0885

    const d  R1    = -0.00730
    const d  R2    = -0.0080
    const d  R3    = -0.0082

    const  dcyR1    = 0.003
    const  dcyR2    = 0.0035
    const  dcyR3    = 0.0032

    const Eland0 = 1.1

    # Damages
    const A1   = 0.0
    const A2   = 0.0028388

    # Carbon cycle
    const MAT0   = 808.9
    const MUO0   = 1255.0
    const MLO0   = 18365.0

    const  _12    = 0.019
    const  _21    = 0.01
    const  _23    = 0.0054
    const  _32    = 0.00034
    const  _11    = 1.0 -  _12
    const  _22    = 1.0 -  _21  -  _23
    const  _33    = 1.0 -  _32
    const  _12    = 0.010
    const  _21    = 0.0048

    const  1      = 0.037
    const  2      = 0.047
    const TOC0   = 0.0068
    const TAT0   = 0.7307
    const C4     = 0.0048
    const FCO22X = 3.8

    # Damages
    const  1    = 0.0
    const  2    = 0.0028388

    # Abatement cost
    const EXPCOST2 = 2.8
    const PBACK    = 1.17
    const BACKRAT  = 2.0
    const GBACK    = 0.005
    const LIMMIU   = 1.0

    # Jacobi partial update parameter
    const boundLo = 0.80
    const boundHi = 1.40

    """
    ##############################################################################
    ##############################################################################
    ################################# 2) TIME-VARYING FUNCTIONS
        #################################
    ##############################################################################
    ##############################################################################
    """
    # Region 1
    function populationR1(t::Int)
        return L0R1*exp(-dLR1*(t-1)) + LssR1*(1-exp(-dLR1*(t-1)))
    end

    function tfpDetR1(t::Int)
        return A0R1 * exp( 1R1 *(1.0 - exp(- 2R1 *(t-1))) /  2R1  )
    end

    function sigmaFuncR1(t::Int)
        return  0R1  * exp( d  R1*(1.0 - exp(- dcyR1 *(t-1))) /  dcyR1  )
    end

    function costR1(t::Int)
        localTerm = (PBACK * sigmaFuncR1(t) * (1.0 + exp(-GBACK*(t-1))) /
            EXPCOST2)
        return localTerm * ((BACKRAT - 1.0) / BACKRAT)
    end


    # Region 2
    function populationR2(t::Int)
        return L0R2*exp(-dLR2*(t-1)) + LssR2*(1-exp(-dLR2*(t-1)))
    end

    function tfpDetR2(t::Int)
        return A0R2 * exp(  1R2 *(1.0 - exp(- 2R2 *(t-1))) /  2R2  )
    end

    function sigmaFuncR2(t::Int)
        return  0R2  * exp( d  R2*(1.0 - exp(- dcyR2 *(t-1))) /  dcyR2  )
    end

    function costR2(t::Int)
```

```
        localTerm = (PBACK * sigmaFuncR2(t) * (1.0 + exp(-GBACK*(t-1))) /
            EXPCOST2)
        return localTerm * ((BACKRAT - 1.0) / BACKRAT)
    end


    # Region 3
    function populationR3(t::Int)
        return L0R3*exp(-dLR3*(t-1)) + LssR3*(1-exp(-dLR3*(t-1)))
    end

    function tfpDetR3(t::Int)
        return A0R3 * exp(   1R3  *(1.0 - exp(-  2R3  *(t-1))) /   2R3   )
    end

    function sigmaFuncR3(t::Int)
        return   0R3   * exp(  d  R3*(1.0 - exp(-  dcyR3  *(t-1))) /   dcyR3   )
    end

    function costR3(t::Int)
        localTerm = (PBACK * sigmaFuncR3(t) * (1.0 + exp(-GBACK*(t-1))) /
            EXPCOST2)
        return localTerm * ((BACKRAT - 1.0) / BACKRAT)
    end


    # General
    function eTree(t::Int)
        return Eland0 * exp(-0.01*(t-1))
    end

    function rr(t::Int)
        return   ^(t-1)
    end

    function damage(T::Any)
        return (1 +   1 *T +   2 *(T^2))
    end

    function forcingOther(t::Int)
        if (t-1) <= 100
            return -0.06 + 0.0036*(t-1)
        else
            return 0.3
        end
    end


    function buildInitialGuess(Region::String)
        if Region == "R1"
            data = CSV.read("D://Mestrado//DICECJL//Results//1
                Agent//resultUSA1dot5.csv", DataFrame)

            DICE = vcat(data, data[end:end, :])
            return NamedTuple((
                :K => DICE.K,
                :C => DICE.C,
                :MIU => DICE.MIU,
                :E => DICE.E,
                :MAT => DICE.MAT,
                :MU => DICE.MU,
                :ML => DICE.ML,
                :TATM => DICE.TATM,
                :TOC => DICE.TOC
            ));
        elseif Region == "R2"
            data = CSV.read("D://Mestrado//DICECJL//Results//1
                Agent//resultGER1dot5.csv", DataFrame)

            DICE = vcat(data, data[end:end, :])
            return NamedTuple((
                :K => DICE.K,
                :C => DICE.C,
                :MIU => DICE.MIU,
                :E => DICE.E,
                :MAT => DICE.MAT,
                :MU => DICE.MU,
                :ML => DICE.ML,
                :TATM => DICE.TATM,
                :TOC => DICE.TOC
            ));
        elseif Region == "R3"
            data = CSV.read("D://Mestrado//DICECJL//Results//1
                Agent//resultJPN1dot5.csv", DataFrame)

            DICE = vcat(data, data[end:end, :])
            return NamedTuple((
                :K => DICE.K,
                :C => DICE.C,
                :MIU => DICE.MIU,
                :E => DICE.E,
                :MAT => DICE.MAT,
                :MU => DICE.MU,
                :ML => DICE.ML,
                :TATM => DICE.TATM,
                :TOC => DICE.TOC
            ));
        else
            error("You must enter either R1, R2 or R3!")
        end
    end

    function solve_climate(E_total::Vector{Float64})
        # Initialize climate variables
        MAT = zeros(TMAX)
        MU = zeros(TMAX)
        ML = zeros(TMAX)
        TATM = zeros(TMAX)
        TOCEAN = zeros(TMAX)

        # Set initial conditions (same as in regional models)
        MAT[1] = MAT0
        MU[1] = MUO0
        ML[1] = MLO0
        TATM[1] = TAT0
        TOCEAN[1] = TOC0

        # Carbon cycle and temperature evolution
        for t in 1:(TMAX-1)
            # Carbon cycle (MAT[t] uses E_total[t] from current period)
            MAT[t+1] =  _11  * MAT[t] +  _21  * MU[t] + E_total[t]
            MU[t+1] =  _12  * MAT[t] +  _22  * MU[t] +  _32  * ML[t]
            ML[t+1] =  _33  * ML[t] +  _23  * MU[t]

            # Temperature evolution (uses MAT[t] from current period)
            TATM[t+1] = (1 -  _21  -  2 ) * TATM[t] +
                         _21  * TOCEAN[t] +
                         1  * (FCO22X * log2(MAT[t] / 596.4))

            TOCEAN[t+1] = TOCEAN[t] +  _12  * (TATM[t] - TOCEAN[t])
        end

        # Return in same structure as original climMod
        return (TAT = TATM, TOC = TOCEAN, MAT = MAT, MUO = MU, MLO = ML)
    end

    """
    ##########################################################################################
    ##########################################################################################
    #################################### 3) REGIONS' SOLVERS ####################################
    ##########################################################################################
    ##########################################################################################
    """

    function solveRegion1(E2_guess::Vector{Float64}, E3_guess::Vector{Float64},
                            oldSolR1::NamedTuple = nothing)

        modelR1 = Model(Ipopt.Optimizer)
        set_optimizer_attribute(modelR1, "print_level", 0)
        #set_optimizer_attribute(modelR1, "nlp_scaling_method", "gradient-based")

        @variables(modelR1, begin
            0 <= K1[t in 1:TMAX]
            0.5 <= C1[t in 1:TMAX]
            0 <= MIU1[t in 1:TMAX] <= LIMMIU
            0 <= E1[t in 1:TMAX]
```

```

            MAT[t in 1:TMAX] >= 10
            MU[t in 1:TMAX]  >= 100
            ML[t in 1:TMAX]  >= 1000
            #0 <= TATM[t in 1:TMAX] <= 20
            -1 <= TOCEAN[t in 1:TMAX] <= 20
        end)

        @variable(modelR1, TATM[t in 1:TMAX],
                lower_bound = 0.0, #(1-fct)*cM.TAT[t],
                upper_bound = 2.0 #(1+fct)*cM.TAT[t]
        )

        # Warm start by setting initial guesses from oldSolR1
        if oldSolR1 !== nothing
            for t in T_index
                set_start_value(K1[t], oldSolR1.K[t])
                set_start_value(C1[t], oldSolR1.C[t])
                set_start_value(MIU1[t], oldSolR1.MIU[t])
                set_start_value(E1[t], oldSolR1.E[t])
            end
        end


        # initial conditions
        @constraint(modelR1, K1_init,      K1[1] == K0R1)
        @constraint(modelR1, MAT_init,     MAT[1] == MAT0)
        @constraint(modelR1, MU_init,      MU[1]  == MUO0)
        @constraint(modelR1, ML_init,      ML[1]  == MLO0)
        @constraint(modelR1, TATM_init,    TATM[1]== TAT0)
        @constraint(modelR1, TOCEAN_init,  TOCEAN[1] == TOC0)

        @constraint(modelR1, MIUinit, MIU1[1] <= 0.05)
        #@constraint(modelR1, MIU1final,  MIU1[TMAX] == LIMMIU)

        @constraint(modelR1, [t in 1:(TMAX-1)],
                    MIU1[t+1] >= MIU1[t]
        )

        @constraint(modelR1, [t in 1:(TMAX-1)],
                MIU1[t+1] <= MIU1[t] + 0.05
        )

        # capital transitions
        @constraint(modelR1, capitalLaw[t in 1:(TMAX-1)],
            K1[t+1] == (1 -   )*K1[t] + (
                (1 - costR1(t)*(MIU1[t]^EXPCOST2)) *
                (tfpDetR1(t)*(populationR1(t)^(1 -  R1 ))*(K1[t]^ R1 )) /
                   (damage(TATM[t]))
            ) - C1[t]
            )

        # region1's own emissions
        @constraint(modelR1, emissions[t in 1:(TMAX-1)],
             E1[t] == sigmaFuncR1(t)*(1 - MIU1[t]) * (
                 tfpDetR1(t)*(populationR1(t)^(1 -  R1 ))*(K1[t]^ R1 )
            )
        )

        @constraint(modelR1, carbonLaw[t in 1:(TMAX-1)],
            MAT[t+1] ==  _11 *MAT[t] +  _21 *MU[t] + E1[t] + E2_guess[t] +
                E3_guess[t] + eTree(t)
        )

        # carbon cycle & climate eqns using E1[t] + E2_guess[t]
        for t in 1:(TMAX-1)
            @constraint(modelR1,
                MU[t+1] ==  _12 *MAT[t] +  _22 *MU[t] +  _32 *ML[t]
            )
            @constraint(modelR1,
                ML[t+1] ==  _33 *ML[t] +  _23 *MU[t]
            )
            @constraint(modelR1,
                TATM[t+1] == (1- _21 - 2 )*TATM[t] +
                              _21 *TOCEAN[t] +
                              1 * (FCO22X*( log2(MAT[t]/596.4))+
                    forcingOther(t))
            )
            @constraint(modelR1,
                TOCEAN[t+1] == TOCEAN[t] +  _12 *(TATM[t] - TOCEAN[t])
            )
        end

        # objective = sum_{t=1..600} utility
        @variable(modelR1, UTILITY_R1)
        @expression(modelR1, periodUtilR1,
            sum(
                (((C1[t]/populationR1(t))^(1-1/  ))/(1-1/  )) *
                    (rr(t)*populationR1(t))
                for t in 1:(TMAX-1)
            )
        )
        @constraint(modelR1, defineUtilityR1, UTILITY_R1 == periodUtilR1)
        @objective(modelR1, Max, UTILITY_R1)

        set_optimizer_attribute(modelR1, "max_iter", 10_000)
        set_optimizer_attribute(modelR1, "tol", 1e-6)

        println("Starting R1 optimization...")

        optimize!(modelR1)

        # extract solution
        K1_sol = [value(K1[t])   for t in T_index]
        C1_sol = [value(C1[t])   for t in T_index]
        MIU1_sol = [value(MIU1[t]) for t in T_index]
        E1_sol = [value(E1[t])   for t in T_index]

        MAT_sol= [value(MAT[t])   for t in T_index]
        MU_sol = [value(MU[t])    for t in T_index]
        ML_sol = [value(ML[t])    for t in T_index]
        TATM_sol=[value(TATM[t])  for t in T_index]
        TOC_sol=[value(TOCEAN[t]) for t in T_index]

        SCC = zeros(TMAX-1)
        for t in 1:(TMAX-1)
            lambda_mat = shadow_price(emissions[t])
            lambda_cap = shadow_price(capitalLaw[t])
            # standard formula:
            SCC[t] = (lambda_mat / lambda_cap)*1000  # in $/tC
        end

        println("Solved Region 1!
        Termination_status:", termination_status(modelR1))

        objR1 = objective_value(modelR1)
        modelR1 = nothing
        GC.gc(true)

        return (
          K = K1_sol, C = C1_sol, MIU = MIU1_sol, E = E1_sol,
          MAT = MAT_sol, MU = MU_sol, ML = ML_sol, TATM = TATM_sol, TOC =
              TOC_sol,
          objR1 = objR1, SCC = SCC
        )
    end

    function solveRegion2(E1_guess::Vector{Float64},
        E3_guess::Vector{Float64},
                              oldSolR2::NamedTuple = nothing)
        modelR2 = Model(Ipopt.Optimizer)
        set_optimizer_attribute(modelR2, "print_level", 0)
        #set_optimizer_attribute(modelR2, "nlp_scaling_method",
            "gradient-based")

        @variables(modelR2, begin
            0 <= K2[t in 1:TMAX]
            0.5 <= C2[t in 1:TMAX]
            0 <= MIU2[t in 1:TMAX] <= LIMMIU
            0 <= E2[t in 1:TMAX]

            MAT[t in 1:TMAX] >= 10
            MU[t in 1:TMAX]  >= 100
            ML[t in 1:TMAX]  >= 1000
            #0 <= TATM[t in 1:TMAX] <= 20
            -1 <= TOCEAN[t in 1:TMAX] <= 20
        end)

        @variable(modelR2, TATM[t in 1:TMAX],
                lower_bound = 0.0, #(1-fct)*cM.TAT[t],
                upper_bound = 2.0 #(1+fct)*cM.TAT[t]
        )

        """
            @variable(modelR2, TOCEAN[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.TOC[t],
                    upper_bound = (1+fct)*cM.TOC[t]
            )

            @variable(modelR2, MAT[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.MAT[t],
                    upper_bound = (1+fct)*cM.MAT[t]
            )

            @variable(modelR2, MU[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.MUO[t],
                    upper_bound = (1+fct)*cM.MUO[t]
            )

            @variable(modelR2, ML[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.MLO[t],
                    upper_bound = (1+fct)*cM.MLO[t]
            )
        """

        # Warm start from oldSolR2
        if oldSolR2 !== nothing
            for t in T_index
                set_start_value(K2[t], oldSolR2.K[t])
                set_start_value(C2[t], oldSolR2.C[t])
                set_start_value(MIU2[t], oldSolR2.MIU[t])
                set_start_value(E2[t], oldSolR2.E[t])
            end
        end


        # init
        @constraint(modelR2, K2_init,     K2[1] == K0R2)
        @constraint(modelR2, MAT_init,    MAT[1] == MAT0)
        @constraint(modelR2, MU_init,     MU[1]  == MUO0)
        @constraint(modelR2, ML_init,     ML[1]  == MLO0)
        @constraint(modelR2, TATM_init,   TATM[1]== TAT0)
        @constraint(modelR2, TOCEAN_init, TOCEAN[1] == TOC0)

        @constraint(modelR2, MIU_init, MIU2[1] <= 0.15)
        #@constraint(modelR2, MIU2final[t in 100:(TMAX-1)],  MIU2[t] == 1.0)

        @constraint(modelR2, [t in 1:(TMAX-1)],
                    MIU2[t+1] >= MIU2[t]
        )

        @constraint(modelR2, [t in 1:(TMAX-1)],
                    MIU2[t+1] <= MIU2[t] + 0.05
        )

        # capital transitions

        @constraint(modelR2, capitalLaw[t in 1:(TMAX-1)],
            K2[t+1] == (1 -   )*K2[t] + (
                (1 - costR2(t)*(MIU2[t]^EXPCOST2)) *
                (tfpDetR2(t)*(populationR2(t)^(1 -  R2 ))*(K2[t]^ R2 )) /
                   (damage(TATM[t]))
            ) - C2[t]
        )


        # region1's own emissions
        @constraint(modelR2, emissions[t in 1:(TMAX-1)],
            E2[t] == sigmaFuncR2(t)*(1 - MIU2[t]) * (
                tfpDetR2(t)*(populationR2(t)^(1 -  R2 ))*(K2[t]^ R2 )
            )
        )

        @constraint(modelR2, carbonLaw[t in 1:(TMAX-1)],
        MAT[t+1] ==  _11 *MAT[t] +  _21 *MU[t] + E2[t] + E1_guess[t] +
           E3_guess[t] + eTree(t)
        )


        # carbon cycle & climate eqns using E1[t] + E2_guess[t]
        for t in 1:(TMAX-1)
            @constraint(modelR2,
                MU[t+1] ==  _12 *MAT[t] +  _22 *MU[t] +  _32 *ML[t]
            )
            @constraint(modelR2,
                ML[t+1] ==  _33 *ML[t] +  _23 *MU[t]
            )
            @constraint(modelR2,
                TATM[t+1] == (1- _21 - 2 )*TATM[t] +
                              _21 *TOCEAN[t] +
                              1 * (FCO22X*( log2(MAT[t]/596.4))+
                    forcingOther(t))
            )
            @constraint(modelR2,
                TOCEAN[t+1] == TOCEAN[t] +  _12 *(TATM[t] - TOCEAN[t])
            )
        end

        # objective = sum_{t=1..600} utility
        @variable(modelR2, UTILITY_R2)
        @expression(modelR2, periodUtilR2,
            sum(
                (((C2[t]/populationR2(t))^(1-1/  ))/(1-1/  )) *
                   (rr(t)*populationR2(t))
                for t in 1:(TMAX-1)
            )
        )
        @constraint(modelR2, defineUtilityR2, UTILITY_R2 == periodUtilR2)
        @objective(modelR2, Max, UTILITY_R2)

        set_optimizer_attribute(modelR2, "max_iter", 10_000)
        set_optimizer_attribute(modelR2, "tol", 1e-6)

        println("Starting R2 optimization...")

        optimize!(modelR2);

        K2_sol = [value(K2[t]) for t in 1:TMAX]
        C2_sol = [value(C2[t]) for t in 1:TMAX]
        MIU2_sol = [value(MIU2[t]) for t in 1:TMAX]
        E2_sol = [value(E2[t])  for t in 1:TMAX]
        MAT_sol= [value(MAT[t]) for t in 1:TMAX]
        MU_sol = [value(MU[t])  for t in 1:TMAX]
        ML_sol = [value(ML[t])  for t in 1:TMAX]
        TATM_sol=[value(TATM[t]) for t in 1:TMAX]
        TOC_sol=[value(TOCEAN[t]) for t in 1:TMAX]

        SCC = zeros(TMAX-1)
        for t in 1:(TMAX-1)
            lambda_mat = shadow_price(emissions[t])
            lambda_cap = shadow_price(capitalLaw[t])
            # standard formula:
            SCC[t] = (lambda_mat / lambda_cap)*1000  # in $/tC
        end

        println("Solved Region 2!
        Termination_status:", termination_status(modelR2))

        objR2 = objective_value(modelR2)
        modelR2 = nothing
        GC.gc(true)

        return (
          K = K2_sol, C = C2_sol, MIU = MIU2_sol, E = E2_sol,
          MAT = MAT_sol, MU = MU_sol, ML = ML_sol, TATM = TATM_sol, TOC =
              TOC_sol,
          objR2 = objR2, SCC = SCC
        )
    end

    function solveRegion3(E1_guess::Vector{Float64},
        E2_guess::Vector{Float64},
                             oldSolR3::NamedTuple = nothing)
        modelR3 = Model(Ipopt.Optimizer)
        set_optimizer_attribute(modelR3, "print_level", 0)
        #set_optimizer_attribute(modelR3, "nlp_scaling_method",
            "gradient-based")

        @variables(modelR3, begin
            0 <= K3[t in 1:TMAX]
            0.5 <= C3[t in 1:TMAX]
            0 <= MIU3[t in 1:TMAX] <= LIMMIU
            0 <= E3[t in 1:TMAX]

            MAT[t in 1:TMAX] >= 10
            MU[t in 1:TMAX]  >= 100
            ML[t in 1:TMAX]  >= 1000
            #0 <= TATM[t in 1:TMAX] <= 20
            -1 <= TOCEAN[t in 1:TMAX] <= 20
        end)


        @variable(modelR3, TATM[t in 1:TMAX],
                lower_bound = 0.0, #(1-fct)*cM.TAT[t],
                upper_bound = 2.0 #(1+fct)*cM.TAT[t]
        )

        """
            @variable(modelR3, TOCEAN[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.TOC[t],
                    upper_bound = (1+fct)*cM.TOC[t]
            )

            @variable(modelR3, MAT[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.MAT[t],
                    upper_bound = (1+fct)*cM.MAT[t]
            )

            @variable(modelR3, MU[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.MUO[t],
                    upper_bound = (1+fct)*cM.MUO[t]
            )

            @variable(modelR3, ML[t in 1:TMAX],
                    lower_bound = (1-fct)*cM.MLO[t],
                    upper_bound = (1+fct)*cM.MLO[t]
            )
        """

        # Warm start from oldSolR2
        if oldSolR3 !== nothing
            for t in T_index
                set_start_value(K3[t], oldSolR3.K[t])
                set_start_value(C3[t], oldSolR3.C[t])
                set_start_value(MIU3[t], oldSolR3.MIU[t])
                set_start_value(E3[t], oldSolR3.E[t])
            end
        end

        # init
        @constraint(modelR3, K3_init,     K3[1] == K0R3)
        @constraint(modelR3, MAT_init,    MAT[1] == MAT0)
        @constraint(modelR3, MU_init,     MU[1]  == MUO0)
        @constraint(modelR3, ML_init,     ML[1]  == MLO0)
        @constraint(modelR3, TATM_init,   TATM[1]== TAT0)
        @constraint(modelR3, TOCEAN_init, TOCEAN[1] == TOC0)

        @constraint(modelR3, MIU3_init,    MIU3[1] <= 0.15)
        #@constraint(modelR3, MIU3final[t in 100:(TMAX-1)],  MIU3[t] ==
           1.0)

        @constraint(modelR3, [t in 1:(TMAX-1)],
                    MIU3[t+1] >= MIU3[t]
        );

        for t in 1:(TMAX-1)
            @constraint(modelR3,
                MIU3[t+1] <= MIU3[t] + 0.05)
        end


        # capital transitions
        @constraint(modelR3, capitalLaw[t in 1:(TMAX-1)],
            K3[t+1] == (1 -   )*K3[t] + (
                (1 - costR3(t)*(MIU3[t]^EXPCOST2)) *
                (tfpDetR3(t)*(populationR3(t)^(1 -  R3 ))*(K3[t]^ R3 )) /
                   (damage(TATM[t]))
                ) - C3[t]
        )


        # region1's own emissions
        @constraint(modelR3, emissions[t in 1:(TMAX-1)],
            E3[t] == sigmaFuncR3(t)*(1 - MIU3[t]) * (
                tfpDetR3(t)*(populationR3(t)^(1 -  R3 ))*(K3[t]^ R3 )
            )
        )

        @constraint(modelR3, carbonLaw[t in 1:(TMAX-1)],
            MAT[t+1] ==  _11 *MAT[t] +  _21 *MU[t] + E3[t] + E1_guess[t] +
               E2_guess[t] + eTree(t)
        )

        # carbon cycle & climate eqns using E1[t] + E2_guess[t]
        for t in 1:(TMAX-1)
            @constraint(modelR3,
                MU[t+1] ==  _12 *MAT[t] +  _22 *MU[t] +  _32 *ML[t]
            )
            @constraint(modelR3,
                ML[t+1] ==  _33 *ML[t] +  _23 *MU[t]
            )
            @constraint(modelR3,
                TATM[t+1] == (1- _21 - 2 )*TATM[t] +
                              _21 *TOCEAN[t] +
                              1 * (FCO22X*( log2(MAT[t]/596.4))+
                    forcingOther(t))
            )
            @constraint(modelR3,
                TOCEAN[t+1] == TOCEAN[t] +  _12 *(TATM[t] - TOCEAN[t])
            )
        end

        # objective = sum_{t=1..600} utility
        @variable(modelR3, UTILITY_R3)
        @expression(modelR3, periodUtilR3,
            sum(
                (((C3[t]/populationR3(t))^(1-1/  ))/(1-1/  )) *
                   (rr(t)*populationR3(t))
                for t in 1:(TMAX-1)
            )
        )
        @constraint(modelR3, defineUtilityR3, UTILITY_R3 == periodUtilR3)
        @objective(modelR3, Max, UTILITY_R3)

        set_optimizer_attribute(modelR3, ”max_iter”, 10_000)
        set_optimizer_attribute(modelR3, ”tol”, 1e-6)

        println(”Starting R3 optimization...”)

        optimize!(modelR3);

        K3_sol = [value(K3[t]) for t in 1:TMAX]
        C3_sol = [value(C3[t]) for t in 1:TMAX]
        MIU3_sol = [value(MIU3[t]) for t in 1:TMAX]
        E3_sol = [value(E3[t])  for t in 1:TMAX]
        MAT_sol= [value(MAT[t]) for t in 1:TMAX]
        MU_sol = [value(MU[t])  for t in 1:TMAX]
        ML_sol = [value(ML[t])  for t in 1:TMAX]
        TATM_sol=[value(TATM[t]) for t in 1:TMAX]
        TOC_sol=[value(TOCEAN[t]) for t in 1:TMAX]

        SCC = zeros(TMAX-1)
        for t in 1:(TMAX-1)
            lambda_mat = shadow_price(emissions[t])
            lambda_cap = shadow_price(capitalLaw[t])
            # standard formula:
            SCC[t] = (lambda_mat / lambda_cap)*1000  # in $/tC
        end

        println(”Solved Region 3!
        Termination_status:”, termination_status(modelR3))

        objR3 = objective_value(modelR3)
        modelR3 = nothing
        GC.gc(true)

```

```
    return (
      K = K3_sol, C = C3_sol, MIU = MIU3_sol, E = E3_sol,
      MAT = MAT_sol, MU = MU_sol, ML = ML_sol, TATM = TATM_sol, TOC = TOC_sol,
      objR3 = objR3, SCC = SCC
    )
end


"""
##################################################################################
##################################################################################
############################ 4) OUTER ITERATION FOR OPEN-LOOP NASH ############################
##################################################################################
##################################################################################
"""

function solveTAOL3AgentsRandomOrder(; maxIter=150, tol=1e-4)
    path = "D://Mestrado//DICECJL//Results//1 Agent//BM//"
    filename = "resultBenchmark1dot5.csv"
    df = CSV.read(path * filename, DataFrame)

    # Climate module
    climMod = (TAT = df.TATM, TOC = df.TOC,
                MAT = df.MAT, MUO = df.MU, MLO = df.ML)

    push!(climMod.TAT, 0.95*climMod.TAT[end])
    push!(climMod.TOC, 0.95*climMod.TOC[end])
    push!(climMod.MAT, 0.95*climMod.MAT[end])
    push!(climMod.MUO, 0.95*climMod.MUO[end])
    push!(climMod.MLO, 0.95*climMod.MLO[end])

    global fct = 0.75

    # 1) initial solutions
    oldSolR1 = buildInitialGuess("R1")  # from your code
    oldSolR2 = buildInitialGuess("R2")
    oldSolR3 = buildInitialGuess("R3")

    # current emission paths
    E1_current = oldSolR1.E # zeros(TMAX)
    E2_current = oldSolR2.E
    E3_current = oldSolR3.E

    for iter in 1:maxIter
        println("===== Iteration $iter (Random  Gauss Seidel ) =====")

        # We shuffle the array [1,2,3] each iteration
        order = shuffle!([1,2,3])  # e.g. it might become [2,1,3] or [3,2,1], etc.

        # store new E1,E2,E3 in local arrays. Start them as the old
        E1_new = copy(E1_current)
        E2_new = copy(E2_current)
        E3_new = copy(E3_current)

        # likewise, we have local solutions for each region if we want
        localSolR1 = oldSolR1
        localSolR2 = oldSolR2
        localSolR3 = oldSolR3

        # We'll go through the order array. For each region in that order,
        # we call the corresponding solveRegion function, passing the *latest* guesses
        # for the other regions' E(t).

        for r in order
            if r == 1
                # region1 sees E2_new, E3_new (the latest versions from whomever moved earlier)
                # or if region2 or region3 haven't moved yet this iteration, we still pass old
                # but you can keep it simpler: pass E2_current, E3_current, if you prefer
                @time solR1 = solveRegion1(E2_new, E3_new, localSolR1)
                # update E1_new
                E1_new = solR1.E
                localSolR1 = solR1
            elseif r == 2
                # region2 sees E1_new, E3_new
                @time solR2 = solveRegion2(E1_new, E3_new, localSolR2)
                E2_new = solR2.E
                localSolR2 = solR2
            else # r==3
                # region3 sees E1_new, E2_new
                @time solR3 = solveRegion3(E1_new, E2_new, localSolR3)
                E3_new = solR3.E
                localSolR3 = solR3
            end
        end

        # After all 3 have solved in random sequence, measure diffs from old iteration
        diffE1 = maximum(abs.(E1_new .- E1_current))
        diffE2 = maximum(abs.(E2_new .- E2_current))
        diffE3 = maximum(abs.(E3_new .- E3_current))
        println("   diffE1=$(diffE1), diffE2=$(diffE2), diffE3=$(diffE3)")

        #    = (diffE1 + diffE2 + diffE3) / 6
        #    = 0.5/log(iter)
           = 0.35

        if diffE2     1.0
```

```
            # partial update for E2
            E1_current .=    .* E1_new .+ (1-   ) .* E1_current
            #E2_current .= 0.4 .* E2_sol .+ 0.6 .* E2_current
        else
            E1_current .= 0.4 .* E1_new .+ 0.6 .* E1_current
        end
        if diffE3     1.0
            # partial update for E2
            E2_current .=    .* E2_new .+ (1-   ) .* E2_current
            #E2_current .= 0.4 .* E2_sol .+ 0.6 .* E2_current
        else
            E2_current .= 0.4 .* E2_new .+ 0.6 .* E2_current
        end

        if diffE1     1.0
            # partial update for E3
            E3_current .=    .* E3_new .+ (1-   ) .* E3_current
            #E2_current .= 0.4 .* E2_sol .+ 0.6 .* E2_current
        else
            E3_current .= 0.4 .* E3_new .+ 0.6 .* E3_current
        end

        # also store localSol back to oldSol if you want warm start next iteration
        oldSolR1 = localSolR1
        oldSolR2 = localSolR2
        oldSolR3 = localSolR3

        if max(diffE1, diffE2, diffE3) < tol
            println("Converged after $iter iterations (Random G a u s s Seidel ).")
            break
        end
    end

    Etot = oldSolR1.E .+ oldSolR2.E .+ oldSolR3.E .+ eTree.(1:TMAX)
    climMod = solve_climate(Etot)

    # finalize
    return (region1=oldSolR1, region2=oldSolR2, region3=oldSolR3, climMod = climMod)
end


"""
##############################################################################
##############################################################################
########################################## 5) MAIN ##########################################
##############################################################################
##############################################################################
"""

```

```
    function main()
        println(" R1  = $ R1;     R2  = $ R2;     R3  = $ R3 ")

        finalSol = solveTAOL3AgentsRandomOrder()#maxIter=10, tol=1e-3)
        println("\n=== DONE. We have final solution arrays. ===")

        region1Sol = finalSol.region1
        region2Sol = finalSol.region2
        region3Sol = finalSol.region3
        climMod    = finalSol.climMod

        # Climate
        MAT_sol  = climMod.MAT
        MU_sol   = climMod.MUO
        ML_sol   = climMod.MLO
        TATM_sol = climMod.TAT
        TOC_sol  = climMod.TOC

        TATM_solR1 = region1Sol[8]
        TATM_solR2 = region2Sol[8]
        TATM_solR3 = region3Sol[8]

        # Unpack region1
        K1_sol   = region1Sol[1]
        C1_sol   = region1Sol[2]
        M1_sol   = region1Sol[3]
        E1_sol   = region1Sol[4]
        SCC1     = region1Sol[end]

        # Unpack region2
        K2_sol   = region2Sol[1]
        C2_sol   = region2Sol[2]
        M2_sol   = region2Sol[3]
        E2_sol   = region2Sol[4]
        SCC2     = region2Sol[end]

        # Unpack region2
        K3_sol   = region3Sol[1]
        C3_sol   = region3Sol[2]
        M3_sol   = region3Sol[3]
        E3_sol   = region3Sol[4]
        SCC3     = region3Sol[end]

        # Objective values
        objR1 = region1Sol[10]
        objR2 = region2Sol[10]
        objR3 = region3Sol[10]

        return (K1_sol = K1_sol, C1_sol = C1_sol, M1_sol = M1_sol,
                E1_sol = E1_sol, K2_sol = K2_sol, C2_sol = C2_sol,
                M2_sol = M2_sol, E2_sol = E2_sol, K3_sol = K3_sol,
                C3_sol = C3_sol, M3_sol = M3_sol, E3_sol = E3_sol,
                TATM_sol = TATM_sol, MAT_sol = MAT_sol, MU_sol = MU_sol,
                ML_sol = ML_sol, TOC_sol = TOC_sol,
                TATM_solR1 = TATM_solR1, TATM_solR2 = TATM_solR2,
                    TATM_solR3 = TATM_solR3,
                SCC1 = SCC1, SCC2 = SCC2, SCC3 = SCC3,
                objR1 = objR1, objR2 = objR2, objR3 = objR3
                )
    end

    result = main()

    result_df = DataFrame(
        K1 =  result.K1_sol[tgrid], K2 =  result.K2_sol[tgrid], K3 =
            result.K3_sol[tgrid],
        M1 =  result.M1_sol[tgrid], M2 =  result.M2_sol[tgrid], M3 =
            result.M3_sol[tgrid],
        E1 =  result.E1_sol[tgrid], E2 =  result.E2_sol[tgrid], E3 =
            result.E3_sol[tgrid],
        C1 =  result.C1_sol[tgrid], C2 =  result.C2_sol[tgrid], C3 =
            result.C3_sol[tgrid],
        TATM_sol = result.TATM_sol[tgrid], MAT_sol =
            result.MAT_sol[tgrid], MU_sol = result.MU_sol[tgrid],
        ML_sol = result.ML_sol[tgrid], TOC_sol = result.TOC_sol[tgrid],
        TATM_solR1 = result.TATM_solR1[tgrid], TATM_solR2 =
            result.TATM_solR2[tgrid], TATM_solR3 = result.TATM_solR3[tgrid],
        SCC1 = result.SCC1[tgrid], SCC2 = result.SCC2[tgrid], SCC3 =
            result.SCC3[tgrid],
        objR1 = result.objR1, objR2 = result.objR2, objR3 = result.objR3
    );
```

UnB

UnB