RHODES UNIVERSITY
Where leaders learn

Rhodes Business School

Simulating the economic impact emerging from the strategic decision-making of firms

A macroeconomic agent-based model

A dissertation submitted in partial fulfilment of the requirement for the degree of

Master of Business Administration

of

Rhodes University

by

A.D.J. Giovannoni
17G8858

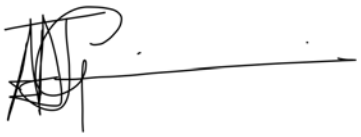
13 September 2021

Abstract

The key research problem addressed in this project is: *how does the strategic decision making of firms affect an economy?* While this is a difficult question to answer, insights may be gained through the use of computational techniques such as agent-based modelling (ABM). An agent-based model is developed that simulates micro-level economic interactions between individuals and firms in different markets, resulting in emergent macro-level features. Technological progress is an important determinant of economic growth and has been decomposed into two complementary factors: knowledge and technological sophistication. The model is used to explore the long-term, macroeconomic consequences of firms investing more heavily in knowledge development. The simulations show that a shift towards a knowledge-based economy (KBE) has an insignificant impact on GDP over the long-term. However, the shift does produce a significant increase in unemployment. The higher unemployment is shown to be particularly high for the unskilled sectors of the population. It is therefore paramount that companies embark on skills development, training and educational initiatives when following a path of technological and knowledge innovation. A transformation of the education system to one that is inclusive, focused on quality, adaptive, encourages creativity, aligned with the needs of industry, and stimulates R&D is crucial. Without a coordinated strategy between industry, academia, and government in attempts to achieve a KBE, economic growth may be constrained, while the levels of unemployment, inequality and poverty may deteriorate.

Declaration

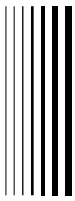
I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other university for a degree and that it represents my own work. I know the meaning of plagiarism and declare that all of the work in the thesis, save for that which is properly acknowledged, is my own.



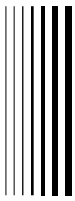
A.D.J. Giovannoni
13 September 2021

Acknowledgements

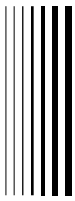
I would like to thank my supervisor Evert Knoesen for his helpful input, patience and understanding throughout the project. For proofreading various sections of the dissertation, I owe a debt of gratitude to Jeremy Baxter, Dr. Shelley Edwards, Dr. Vincent Smith, and Dr. Tracey Nowell. Their input was invaluable, especially with my many grammatical errors. (Any errors that may remain are mine, and mine alone.) They were also instrumental with providing moral support throughout, especially when my motivation wavered. I am grateful to many people who encouraged and inspired me throughout the process. Prof. Justin Jonas, Prof. Rod Walker, and Prof. Martin Hill relentlessly pushed me to get things done. My colleagues, Prof. David Roux, Dr. Jennifer Williams, Prof. Makaiko Chithambo and Anthony Sullivan provided moral support throughout. My classmates, Linda Fleming and Cornelia Blignaut, were a constant source of inspiration. I would like to thank some of my previous students, in particular Talon Myburgh, Amy Bray, and Miles Ellery for bullying me into developing a new course that required me to learn the Python programming language. My wonderful sister, Renata, opened her home to me whenever I needed a break. Finally I would like to thank my parents who have always been extremely supportive in all of my crazy endeavours.



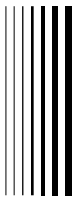
Contents



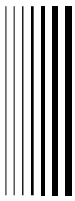
List of Figures



List of Tables



Listings



Glossary

4IR 4th industrial revolution. 4, 20

ABM agent-based modelling. 1, 21, 25, 34, 35, 101

ACE agent-based computational economics. 26

AI artificial intelligence. 18

BHAG big hairy audacious goal. 11

BPA business process automation. 17

DCS distributed control system. 17

DSGE dynamic stochastic general equilibrium. 27

ERP enterprise resource planning. 17

FDI foreign direct investment. 20

FIFO first-in, first-out. 50

GDP gross domestic product. 2, 66

GMM Gaussian Mixture Model. 64

ICT information and communications technology. 18

IS-LM investment-savings and liquidity preferences-money supply. 27

KBE knowledge-based economy. ii, 4, 18, 81, 82, 84

KEI Knowledge Economy Index. 18, 81

MABM macroeconomic agent-based modelling. 5, 26, 28, 29, 30, 31, 35, 46, 51

MES manufacturing execution system. 17

OECD Organisation for Economic Co-operation and Development. 18

OOD object-oriented design. 96, 98

OOP object-oriented programming. 23, 25, 34, 96, 97, 98

PLC programmable logic controller. 17

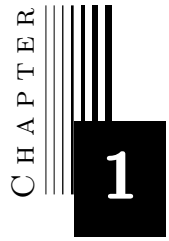
R&D research and development. 19, 82

RBT resource-based theory. 12

RBV resource-based view. 10, 13, 14, 19

SCADA supervisory control and data acquisition. 17

WEF World Economic Forum. 18



Introduction

Firms are an integral part of an economy, so that a collective strategic shift by firms in an economy could result in significant macroeconomic consequences. Such strategic shifts are often driven by external factors such as prevailing economic conditions, changes in government policy, increased competition, or technological change. Recent political uncertainty, struggling state-owned enterprises, and low business confidence has led to many South African companies reducing their investment in fixed capital, with a resulting downturn in economic growth and persistently low unemployment (Lings 2017, Stanlib 2018, Hungwe & Odhiambo 2019). South Africa is also wanting to move towards a knowledge-based economy (KBE), which will result in considerable change for many existing companies and will require new innovative companies to be created (South Africa 2011, Vadra 2017). It is important to consider the economic consequences of such a paradigm shift.

An economy is a highly complex system, comprised of a large number of agents that are constantly changing and interacting with each other in a myriad of ways. Constructing and analysing models are an attempt to understand complex real-world phenomena. This complexity often requires simplifying assumptions or considering only certain aspects of the system (Giordano et al. 2014). Microeconomics considers the properties and behaviour of each of the individual economic agents (Besanko & Braeutigam 2013), while macroeconomics attempts to understand the aggregated behaviour of an economy as a whole. Various economic modelling techniques can be used to understand, describe, or predict the behaviour of each of these aspects. One modelling technique that has gained some traction in recent years is agent-based modelling (ABM). The appeal of ABM in economic modelling is that by implementing the micro-level properties and behaviour for a large number of agents, macro-level phenomena should emerge. In principle, it provides a tool that can tangibly connect the realms of economic theory. In this research project, an agent-based macroeconomic simulation model will be developed that can be used to explore the macroeconomic consequences of a collective strategic shift by firms. In particular, as more firms in the economy make a strategic choice to embrace

technological sophistication, nurture innovation and invest in knowledge, the simulation should provide a means to predict any significant long-term changes in macroeconomic measures.

1.1 Economic growth

One of the most important measures of a country's economic output is gross domestic product (GDP). While GDP may be defined in various ways (Burda & Wyplosz 2017), it ultimately provides a measure of the productive activity within an economy. One way of defining GDP is as the sum of all the final goods and services that an economy produces in a given time period within a country. Using this definition, there are four main contributors to GDP, Y: consumption of final goods and services, C; investment, I; government spending, G; and net sales to the rest of the world, which is the difference between exports, X and imports, Z; resulting in

$$Y = C + I + G + X - Z. \tag{1.1}$$

The circular flow diagram shown in Figure 1.1 illustrates the factors that contribute to an economy, and highlights the complexity that emerges from the interplay between these contributors. The circular flow model serves as the basis for many macroeconomic agent-based models.

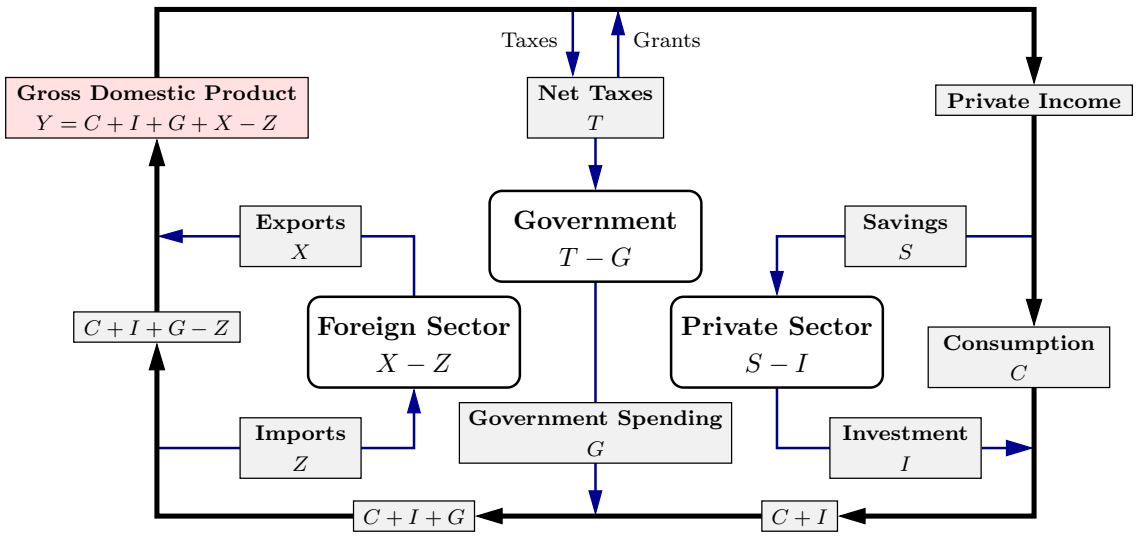


Figure 1.1: Circular flow diagram showing the flow of money within an economy. Adapted from Burda & Wyplosz (2017).

The GDP (in 2019 US$) for South Africa from 1960 till 2018 is shown in Figure 1.2. From the GDP time series it can be noted that in the long-term, GDP approximately follows an exponential growth model,

$$Y(t) = Y_0\, e^{g(t-t_0)}, \tag{1.2}$$

where Y_0 represents the initial GDP at an initial time t_0 and g is the annual growth rate. Also shown in Figure 1.2 is a fit to the South African GDP data based on equation 1.2, showing an approximate exponential growth over the long-term.

It has been observed that economic output, as measured by GDP, tends to grow over time, so it is important to understand the reasons and determinants of economic growth (Acemoglu

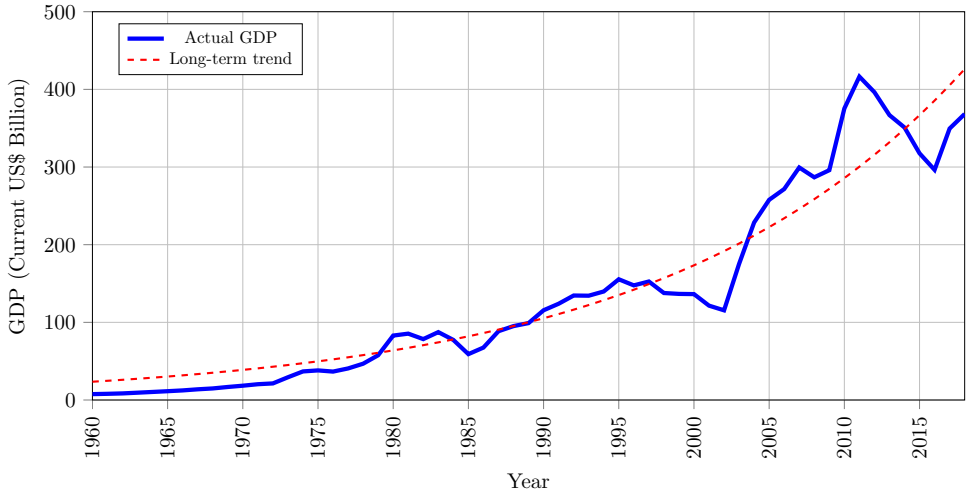


Figure 1.2: Time series of GDP (in current US$) for South Africa. Dashed line indicates fitted GDP. Raw data source: World Bank (2019)

2009). Empirical research into determinants of growth paints a complex picture. Durlauf et al. (2005) identified a large number of growth factors, including: education, demographic age characteristics, geography, levels of foreign direct investment, government expenditure, government policy, levels of inequality, health of the population, political stability, trade policy, and property rights. He & Xu (2019) assigned growth determinant variables into four categories: (a) Solow-Swan growth variables (income, capital, population growth, labour, investment), (b) demography, (c) policy and (d) institution (governance, legal, etc.). A recent study by Bruns & Ioannidis (2020) indicates that the relevance of particular growth determinants is dependent on the growth period of the economy.

There are several theories and models of economic growth (Piętak 2014). In practice, theoretical macroeconomic growth models need to consider a much smaller set of factors than those detailed by Durlauf et al. (2005). Macroeconomic textbooks, such as Burda & Wyplosz (2017), often describe three main forces driving economic growth: investment in capital, population growth, and technological progress. There are therefore three variables that are used in many macroeconomic models: *capital*, K, *labour*, L, and *technology*, T. A macroeconomic model would then consider these three variables as the aggregate factors of production, and so the macroeconomic production function would therefore be of the form (Barro et al. 2004, Pokrovskii 2014)

$$Y(t) = F(K(t), L(t), T(t)). \tag{1.3}$$

The mathematical formulation of many growth models, such as the Solow-Swan model (Solow 1956, Swan 1956), Schumpeterian growth models (Acemoglu 2009) or the Romer model (Romer 1990, Grossman & Helpman 1991, Aghion & Howitt 1992), consider these factors as dynamical variables, each of which is outlined as follows:

(a) Investment in physical capital or *capital stock*, K, is an important driver of economic growth. Here, capital stock refers to all buildings, machinery, plant, and equipment that are used to produce the output. In macroeconomic models such as the Solow-Swan model,

capital accumulation through investment is the key dynamical variable used in modelling growth (see Appendix B, pg. 90).

(b) Growth in the population size necessitates an increase in the output of goods and services: '*more people require more stuff*'. It also means that more people require an income to purchase goods and services: '*more people need jobs*'. In order to increase output to satisfy the demand arising from a growing population and to provide the disposable income that the population needs, industry would need to increase the *labour employed*, L.

(c) An important determinant of economic growth is *technological progress*, T. In the context of economic theory, the term '*technology*' refers to how the "inputs to the production process are transformed into output" (Jones 2002, pg. 80). Technological progress is achieved through invention, innovation, and the generation of knowledge and new ideas. Diaconu (2011) argued that innovation is a major determinant of growth. Innovation may come in many forms: innovation in the quality and variety of products, innovation in creating new markets, innovation in manufacturing processes that result in greater efficiencies or lower costs, and innovation in the way in which a business operates. In Schumpeterian models, economic growth is driven by process innovations to existing machines (new innovations replace old technologies), leading to improvement in efficiencies (Acemoglu 2009). In other words, growth involves creative destruction (Aghion et al. 2015). Romer (2011) extended the Solow-Swan model by incorporating the endogenous accumulation of knowledge as a contributor to economic growth.

The agent-based model that has been developed will include these three factors of production for individual firms. Of interest is the impact of investment in technology and knowledge accumulation on macroeconomic performance.

1.2 The role of technology

In recent years there has been considerable attention given to the 4^{th} industrial revolution (4IR) and a shift towards a KBE in many advanced economies. The 4IR refers to the application of computing, information, and communications technologies in new and novel ways. It is characterised by an integration of diverse technologies, blurring the distinctions between physical (such as autonomous vehicles, 3D printing, robotics, and new materials), digital (hardware, software, and communications), and biological domains (Schwab 2016, Lee et al. 2018). As the technologies become more pervasive, sophisticated and integrated, society and economies are rapidly transforming, resulting in new economic paradigms (Schwab 2016). One of the consequences of the 4IR is that it is extremely knowledge-based, demanding new competencies and skills in the labour force. This emergent economic reality, with knowledge as a new currency, is the essence of the KBE. Sum & Jessop (2013) have argued that policies that promote knowledge development are "the best route to economic progress in developing economies".

To improve efficiencies, create new markets and establish global competitiveness, many companies are embracing new and emerging technologies (WEF 2018). Achieving these goals within an economy requires significant investment in new technologies, a highly skilled labour force, and the establishment of high-technology industries (Vadra 2017). The new reality of accelerating change and innovation is placing greater demands on business leaders to formulate workforce strategies that will develop the necessary skills required for the future (WEF 2018). It also requires a paradigm shift in educational policies, systems, and curricula. Curricula need to be highly adaptive in order to respond to rapid changes. An emphasis needs to be placed on critical thinking, complex problem-solving, lifelong learning, creativity, technological design, and programming (WEF 2018, Table 4). There are therefore two important aspects that need to be considered in the modelling of firms within an economy: technology and knowledge.

1.3 Agent-based models

Agent-based modelling has recently become a useful tool for analysing a wide variety of complex systems (Epstein 2006, Siegfried 2014). In agent-based models, populations of individual agents are created within an artificial environment and allowed to interact with each other or with the environment. There are traditionally three basic ingredients in an agent-based model: *agents* as the active elements in the model, an *environment* in which the agents operate, and a *scheduler* that determines the sequence and timing of events in the simulation. A key feature is that

> "the only equations present are those used by individual agents for decision making. Different agents may have different decision rules and different information; usually, no agents have global information, and the behavioural rules involve bounded computational capacities - the agents are 'simple'." (Axtell & Epstein 2006, pg. 152)

Agents are the basic 'atomic units' that are used in the construction of an agent-based model. They can represent individuals, organisms, social groups, components, complex organisations, etc. An important advantage of macroeconomic agent-based modelling (MABM) approaches is that there is no need for assumptions at the macroeconomic level (Lengnick 2013).

Behavioural rules that describe how each type of agent makes different decisions in different contexts are explicitly incorporated in the model, providing a 'bottom-up' construction of macroeconomic phenomena. For individuals within an economy, the behavioural rules could encode consumer spending patterns (Carroll 2001, Bremer et al. 2017), changes in education (Kinsella et al. 2011), population dynamics (Grow & Van Bavel 2016), etc. Firms' behavioural rules could incorporate the dynamic capability as proposed by Zott (2003), associated employment policies, pricing decisions, technological change, etc. (Kinsella et al. 2011).

1.4 Project overview

The primary objective of the project is to implement an agent-based simulation that can be used to determine the macro-economic impact of firms strategically adopting higher levels of technological sophistication. It should demonstrate the importance and relevance of micro-level strategic decisions on macroeconomic indicators. This would facilitate a quantitative analysis of the short-, medium-, and long-term impact of macroeconomic policy decisions. The specific objectives of the project are:

(a) To design, implement and test an agent-based simulation framework that can be used to conduct macroeconomic simulations.

(b) To apply the simulation framework to investigate potential changes in the long-term macroeconomic indicators resulting from the strategic shift of firms towards greater levels of technological sophistication.

The implementation will incorporate both quantitative and qualitative modelling techniques to model the behaviour of individuals and firms in an economy, develop mathematical and computational models to produce quantitative data, and use the simulated results to make predictions, placing it within a *positivist* research paradigm (Guo & Sheffield 2006, De Vos et al. 2011).

Two simulation scenarios are considered: one scenario (referred to as the *baseline* scenario) that is used as a reference or benchmark, and a second scenario (referred to as the *explored* scenario) in which firms in the economy have made a shift towards higher investment in knowledge and higher levels of technological capabilities. The conceptual framework for how the simulations are conducted is shown in Figure 1.3. Several Monte Carlo simulations (Rubinstein & Kroese 2016) will be conducted with a set of baseline parameters, establishing baseline trends in a number of macroeconomic and other variables. Stochastic fluctuations from each Monte Carlo run are then averaged-out, giving the underlying output variables from the model. For example, Figure 1.3 shows how a mean GDP is obtained in each of the two scenarios. The baseline trend for GDP growth is then validated according to the Solow-Swan growth model. For the second explored scenario, a single model parameter is changed in order to shift the strategic focus of firms. By comparing the output from the two scenarios it is then possible to determine if there are significant shifts in the long-term behaviour of the macroeconomic variables.

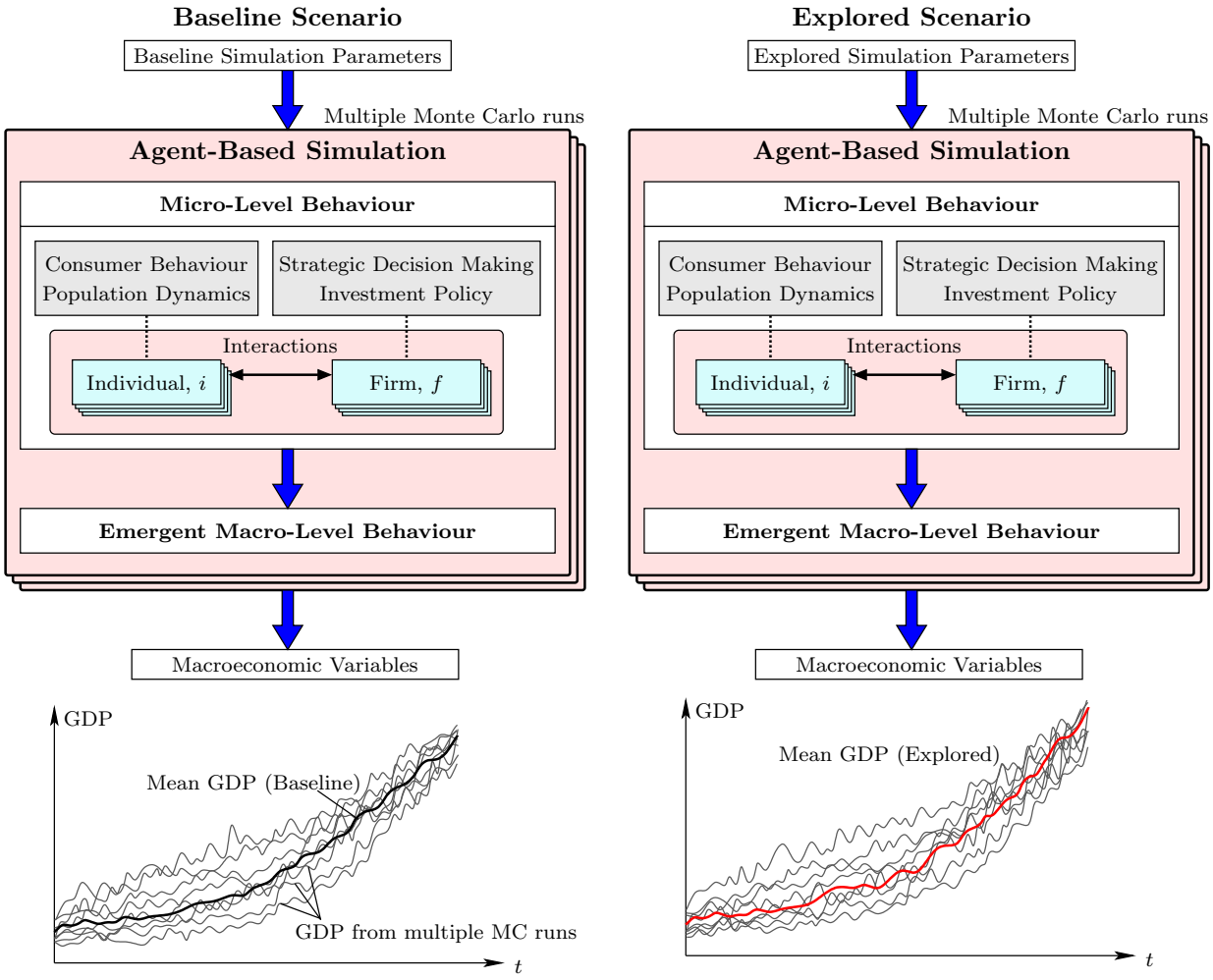


Figure 1.3: The conceptual framework for the project.

An overview of the chapters that follow is shown in Figure 1.4.

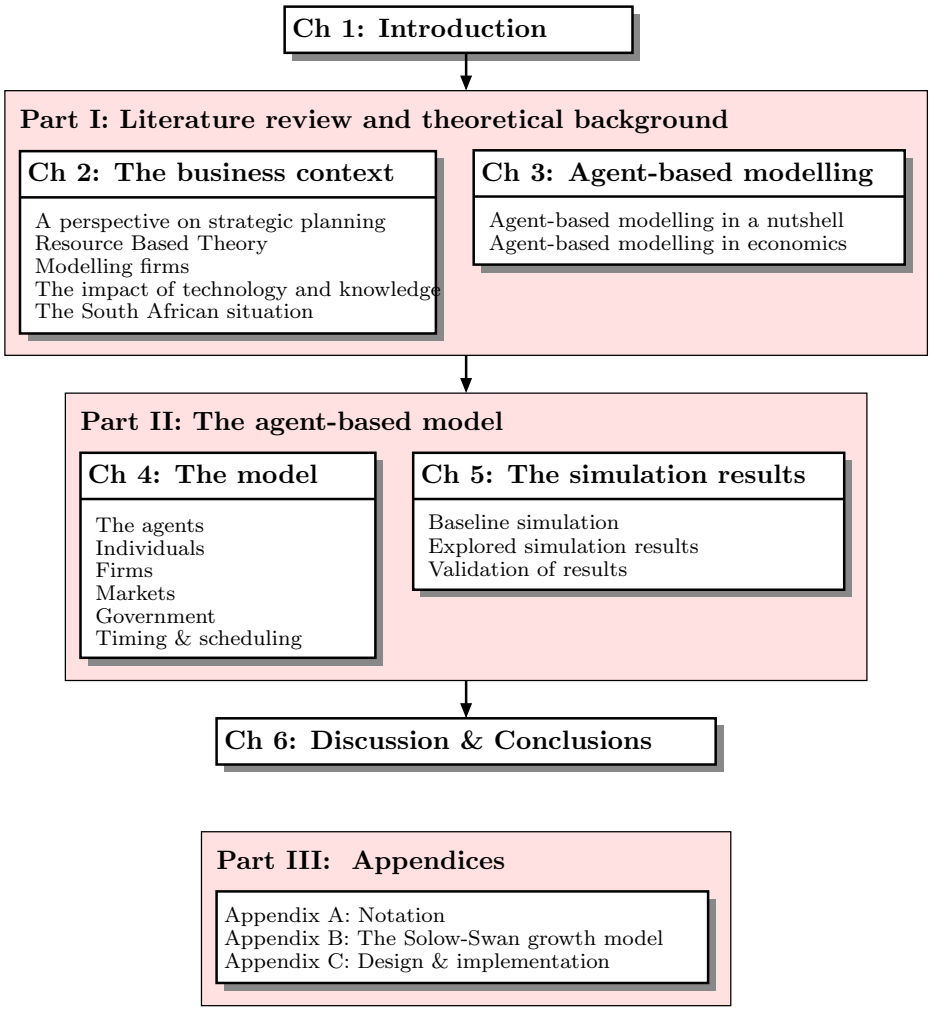
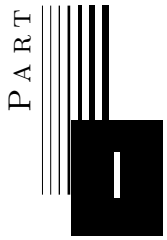


Figure 1.4: An overview of the logical flow of the chapters.



Literature review and theoretical background

CHAPTER 2

The business context

This chapter provides a brief overview of strategic planning and the role of technology within firms. Section 2.1 provides an overview of strategic planning within organisations. An important contemporary perspective of strategic planning is encompassed by the resource-based view (RBV), discussed in Section 2.2. The agent-based model developed in later chapters will require a mathematical model of a firm's production that is outlined in Section 2.3. Section 2.4 discusses the importance of technology and knowledge, and in particular the 4^{th} industrial revolution and the knowledge-based economy (KBE). Finally, some brief comments about the relevance to South Africa are provided in Section 2.5.

2.1 A perspective on strategic planning

Organisations need to think about where they are, where they want to be and how to get there. The essence of strategic planning is to provide a path between the current status and a desired future status (Butuner 2015). There are three main questions that organisations need to ask themselves, leading to various strategic planning phases (Louw & Venter 2013). The "*where are we?*" question is addressed in the *strategic analysis* phase. The "*where do we want to be?*" question addressed is in the *strategic development* phase. The "*how do we get there?*" question is answered in the *strategy implementation* phase. These phases are illustrated in Figure 2.1.

(a) **Strategic analysis**: A careful and critical analysis of the business and its operating environment needs to be undertaken. Typically an organisation would need to address many questions such as: *What are the current problems? What are we doing well? What are we doing badly? What is our market? What is happening in our market? What can be done about it? What are our strengths and weaknesses? What opportunities do we have? What threats or risks do we face?* Many strategic analysis tools can be used in this process (Fleisher & Bensoussan 2007), including: SWOT analysis, the PESTEL framework

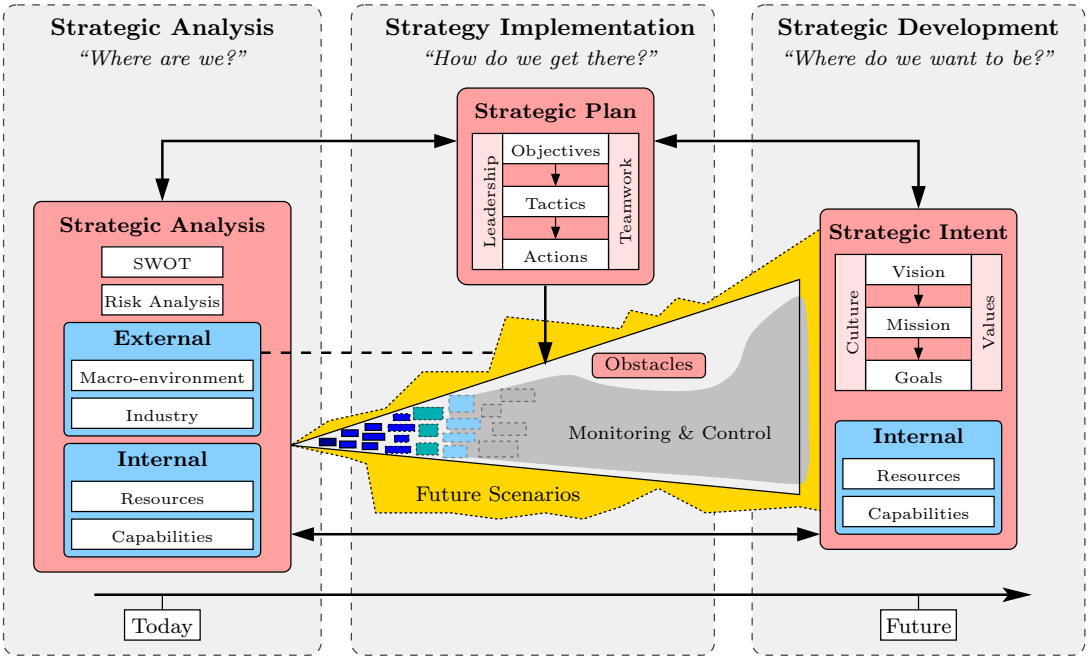


Figure 2.1: A proposed framework for strategic planning.

(Rothaermel 2016), developing a strategy canvas (Kim & Mauborgne 2015), analysing the value chain (Porter 1985), and considering Porter's 5 forces (Porter 1985, 2008).

(b) **Strategy development**: With an understanding of the current situation, an organisation can then begin to develop its strategic intent, which essentially aims to answer the question: *Where do we want to be?* As opposed to operational or functional plans that are short-term, strategic plans are usually long-term in nature, and so the strategic intent should describe aspirations for a distant future (Butuner 2015). The importance of strategic intent is illustrated by the quote by Lewis Carrol, "*If you don't know where you are going, any road will get you there.*"

Strategic intent is usually encapsulated in an organisation's vision, mission and goals (Rothaermel 2016). A *vision* articulates an organization's future aspirations, providing an answer to the question: "*What do we want to become?*" (David & David 2014). Collins & Porras (2011) suggested that a vision should consist of a vivid description of the future and be a big hairy audacious goal (BHAG). A BHAG is a clear and compelling future objective that an organization can strive for. The *mission* describes what an organisation actually does, defining the way in which the vision can be achieved, answering the question: "*What is our business?*" (David & David 2014). It establishes the goods and services the organisation plans to provide, and the markets in which it plans to operate. The *goals* are broad targets or milestones that need to be achieved in order to accomplish the mission and vision. Underpinning the future intent would be the aspired culture within the organisation and the values establishing how it would operate.

(c) **Strategy implementation**: With a clear understanding of where they are and where they want to go, it becomes necessary to formulate a plan to achieve the strategic intent. This would entail developing short-term operational plans, possibly restructuring the organisa-

tion, identifying the leadership requirements, developing processes and systems, effective communication, formulating a change management plan, analysing possible future scenarios, and implementing techniques for monitoring and control (Louw & Venter 2013). There are several tools that can assist in this process. Organisational analysis and design can be performed using the McKinsey 7S framework (Ravanfar 2015). Business processes can be formulated using business process re-engineering (Hammer & Stanton 1995). Tools such as the balanced scorecard can be used to develop approaches to measurement and control (Atkinson 2006).

In practice formulating a strategic plan should be an iterative process and organisations should go back and forth between each of the phases outlined above (Steiner 2010).

2.2 Resource Based Theory

The ability of an organisation to effectively identify, understand, and exploit any potential sources of competitive advantage are important goals of strategic management (Barney & Clark 2007). An organisation's resources include all assets, capabilities, processes, attributes, information and knowledge that are controlled by the organisation in order to implement value-creating strategies to achieve a competitive advantage. Within a resource-based view of the firm, the resources within an organisation may be categorised as *physical capital resources* (physical technology, plant and equipment, access to raw materials); *human capital resources* (the training, education, experience, judgement, intelligence and insights of employees) and *organisational capital resources* (formal reporting structures, planning, coordination and control mechanisms) (Barney 1991).

Barney & Clark (2007) describe competitive advantage in terms of a firm's ability to implement a value-creating strategy than cannot be easily adopted by competitors (both current and future). A sustained competitive advantage is a competitive advantage whose benefits cannot be easily duplicated by competitors. Strategy can be viewed in the context of resource-based theory (RBT) as "the match an organization makes between its internal resources and skills and the opportunities and risks created by its external environment" (Grant 1991). The relationship between resources, capabilities, competitive advantage and strategy in the context of strategic analysis in the resource-based view is illustrated in Figure 2.2.

Barney (1991) proposed a resource-based model incorporating four factors that are required to achieve a sustained competitive advantage: the resources must be valuable (V), rare (R), imperfectly imitable (I), and non-substitutable (N). These characteristics are often referred to as the 'VRIN' framework (Foss 2011). Valuable resources can lead to a sustained competitive advantage since they enable a firm to implement strategies that improve efficiencies and effectiveness. If the resources are homogeneously available to a large number of organisations, then each organisation can utilise the resources in a similar manner, thereby ensuring that no one organisation has a competitive advantage (even though the resource may be valuable). When the valuable resources available to an organisation are rare, then those resources have

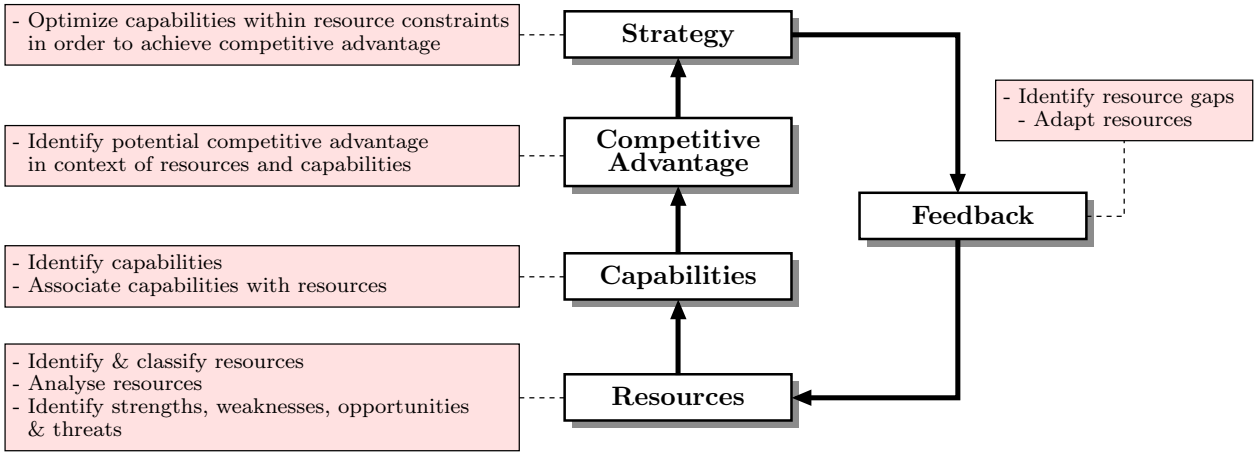


Figure 2.2: A resource-based framework for strategic analysis. (Adapted from Grant (1991))

the potential to establish a sustained competitive advantage. Barney (1991) goes on to argue that valuable, rare resources can only be a source of sustained competitive advantage if they cannot be easily obtained by other firms, i.e. are imperfectly imitable. The resources can be imperfectly imitable due to several reasons (Barney & Clark 2007):

- A firm may possess a resource as the result of a set of *unique historical conditions*. The firm may simply have been 'in the right place at the right time' allowing it to acquire the resources. A firm's performance is therefore not only dependent on industry structure but also on its unique historical circumstances. A competitor or new entrant will not have had the benefits, experience or access that may arise due to the particular history of a firm.

- There may also be a *causal ambiguity* in the link between the firm's resources and its sustained competitive advantage. Since the link is ambiguous or poorly understood, it will be difficult for competitors to replicate the strategy.

- *Complex social phenomena* (for example, organisational culture) within an organisation would also make it difficult for a competitor to duplicate how the resources (especially human and organisational capital resources) lead to a sustained competitive advantage. The ability of these resources to lead to sustained competitive advantage may be attributable to the complex social phenomena (so there would be no causal ambiguity). Knowing the causal relationship does not necessarily enable another firm to create an environment where these phenomena could be established.

The final requirement for a resource to be able to lead to a sustained competitive advantage is that it is not strategically equivalent to another resource. If a resource can easily be substituted by another resource, which may not be valuable or rare, to implement a particular strategic plan, then that resource is not a source of competitive advantage.

It is more widely recognised in the economics literature that performance is strongly influenced by the path taken by a firm, i.e. 'history matters' (Liebowitz & Margolis 1999, David 2005,

Martin & Sunley 2012). "Path dependence is having a major impact on the way in which economists model firm and market behavior" (Lockett & Thompson 2001, pg. 743). Within the RBV, the resources to a firm (its 'opportunity set') are unique to the particular firm and are acquired or developed over time (Lockett & Thompson 2001, Lockett et al. 2009). Several path-dependent aspects of firm behaviour are strategically relevant for the firm (Lockett & Thompson 2001): (a) various patterns of diversification and market entry; (b) corporate refocusing and market exit; (c) the ability of firms to innovate and exploit their *dynamic capability*; (d) the relationship between diversification and performance; and (e) the evolution of an industry with rapid technological change. Of relevance to this project is the role of knowledge, innovation and dynamic capabilities.

Competitive advantage will change over time (Helfat & Peteraf 2003, Teece 2009). The implication is that strategy continually needs to adapt (Helfat et al. 2009, Endres 2017). Dynamic capability refers to a "firm's ability to integrate, build and reconfigure internal and external competencies to address rapidly changing environments" (Zott 2003). It provides a strategic framework that can be used to explain differences in performance of firms despite being in the same industry with similar environmental conditions and analyse changes in organizational capabilities (Winter 2003, Zott 2003, Endres 2017).

Motivated by the RBV of the firm, Zott (2003) proposes a model in which a firm is characterised by its *resource configuration* at any point in time that is comprised of three variables: *product quantity* (or output produced), *product innovation rate* that encapsulates the resources allocated to product differentiation, and *process innovation rate* that incorporates the resources that a firm allocated to achieve process innovations. The resource configuration is evolved or reconfigured in four stages as illustrated in Figure 2.3: variation (through imitation/experimentation), selection, retention and competition. The dynamic capability attributes that characterise the evolutionary process are the timing of the change, the cost of the change and how the firm learns from changes. Based on the model, Zott (2003) poses further questions regarding the dynamic capability of a firm. Firstly, Zott (2003) asks if it is possible to identify the attributes of dynamic capability that impact the performance of a firm. Secondly, in what way do the attributes contribute towards the emergence of differentiated performance within an industry. The essence of the current project is to determine the macroeconomic impact of widespread changes in the resource configuration of firms in an economy.

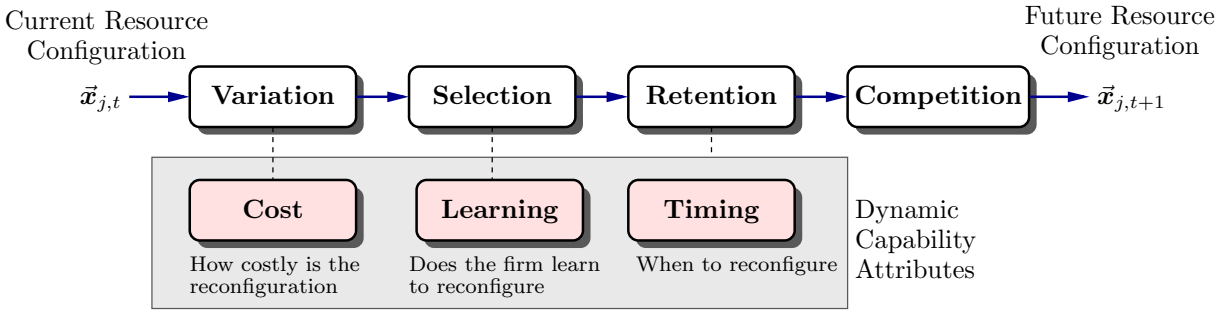


Figure 2.3: Overview of Zott's model for resource reconfiguration. (Adapted from Zott (2003))

2.3 Modelling firms

Firms play a vital role in an economy by transforming available productive resources into final goods and services. This is an oversimplification, in reality, firms are highly complex entities (Walker 2016). The productive resources (land, labour, capital equipment, natural resources, raw materials, water, electricity, etc.) can be regarded as *inputs* or *factors of production* that firms use in the production or transformation process, while the *outputs* are the various goods and services that the firm produces (Besanko & Braeutigam 2013). For simplicity, it is often assumed that a firm produces a single product. Figure 2.4 shows a system or 'black-box' perspective of a firm that converts inputs to outputs (Walker 2016).

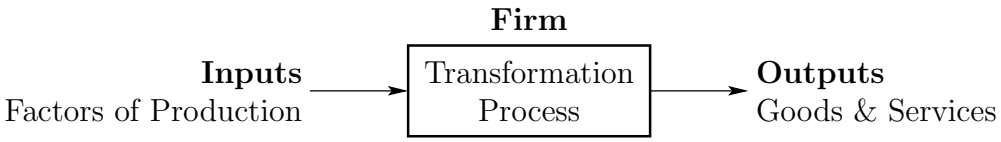


Figure 2.4: A systems view of the firm showing how the input factors of production are transformed into the output goods and services.

The mathematical relationship between the inputs and outputs of a firm are encoded in a *production function* (Heathfield & Wibe 2016, Shephard 1970). It allows one to ignore the operational details within a firm, and focus purely on the most significant factors of production. Transformations that are not 'acts of production' are not considered (Heathfield & Wibe 2016). At an abstract level, the production function encapsulates a firm's production process. This abstraction is both a blessing and a curse. On the one hand, it allows one to address many 'what?' questions: *What are the most relevant or significant factors of production? What is the relationship between each factor of production and output? What factors of production can one measure?* On the other hand, abstraction does not offer any insight into many 'why?' questions: *Why are some factors of production more important than others? Why is the output more strongly influenced by one factor compared to another? Why are some possible factors of production unimportant?* Some of these 'why' questions are important considerations in the resource-based view of the firm.

2.3.1 The production function

It would be useful to construct a mathematical model that can describe the transformation process of a firm. The production function would be used to model the output behaviour for the firm, without going into details on how the production is physically accomplished. To accomplish this task, one needs to clearly define the inputs and outputs of the system. It is assumed that each factor of production may be aggregated and quantified in some way (Heathfield & Wibe 2016). For example, a firm may have a wide variety of equipment used in its manufacturing process, but are all 'lumped together' as a quantifiable measure of equipment. This is an enormous simplification but is often a necessary step in mathematical modelling (Giordano et al. 2014).

Let us consider a firm producing a single product using n factors of production, x_i, where each factor is a positive number (i.e. $x_i \in \mathbb{R}_+$ for $i = 1 \dots n$). The firm's production process will transform the inputs, x_i, to a quantity of output, Q. It will be convenient to denote all the inputs as a vector, $\vec{\boldsymbol{x}} \equiv x_i$. Each of the n inputs of the production function is assumed to be non-negative, real numbers [1] (i.e. $\vec{\boldsymbol{x}} \in \mathbb{R}^n_+$). The output will also be a non-negative number, $Q \in \mathbb{R}_+$. Mathematically, the production function is assumed to be a smooth, infinitely differentiable function of each of the inputs, x_i. The requirement for differentiability ensures that the production function is continuous in each of the inputs, thereby ensuring that small changes in any input lead to small changes in output. A firm will also adopt technology in the production process, which we will represent by ω. The transformation process, illustrated in Figure 2.5, is encapsulated in the firms' *production function*, F, that is defined as follows. The production function specifies the *maximum* output quantity, Q, that a firm can produce as a function of its inputs, $\vec{\boldsymbol{x}}$ and its technology, ω.

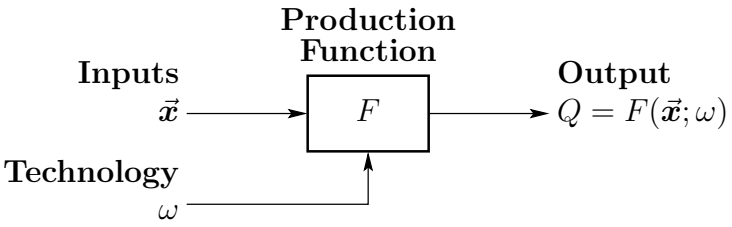


Figure 2.5: A systems view of the production function.

A commonly used production function is the *Cobb-Douglas production function*. The most general form of the Cobb-Douglas production function is given by (Shephard 1970)

$$F(\vec{\boldsymbol{x}}) := \prod_{i=1}^{n} (A_i x_i)^{\alpha_i}, \tag{2.1}$$

where A_i represents the *factor productivity* and the parameters α_i represent the *output elasticities* for the inputs x_i. The factor productivity parameters must satisfy $A_i > 0$, while α_i satisfies the condition

$$\sum_{i=1}^{n} \alpha_i \stackrel{!}{=} r, \tag{2.2}$$

for some constant r.

A special case of the general Cobb-Douglas production function considers two input factors of production, capital, K and labour, L, so that $x_i \equiv (K, L)$. The most widely used form of the two-input Cobb-Douglas production function is given by (Sørensen & Whitta-Jacobsen 2010, Romer 2011)

$$Q = F(K, AL) := K^{\alpha}(AL)^{1-\alpha} \tag{2.3}$$

where α satisfies $0 < \alpha < 1$, and A is a labour productivity parameter. This is a simplified version of the generic production function given in equation (2.1), with $x_i \equiv (K, L)$, $\alpha_i \equiv (\alpha, 1-\alpha)$ and $A_i \equiv (1, A)$. This form of production function has a constant returns to scale.

[1] Of course not all input factors are real numbers: labour is technically only supplied in discrete units. But if one regards labour in terms of time, then it may be regarded as a continuous variable.

The parameter α represents the output elasticity for capital, while $1-\alpha$ represents the output elasticity for labour. This form of the Cobb-Douglas production function is convenient since it only has two parameters, A and α.

2.4 The impact of technology and knowledge

Technology plays an important role in many contemporary economic growth models (Romer 1990, Sharipov 2015). Schumpeterian growth theory recognizes the importance of innovation in replacing old technologies through *creative destruction* (Aghion et al. 2014, 2015). In the Schumpeterian framework, improvement in production quality and efficiency as a result of process innovations is the 'engine of economic growth' (Acemoglu 2009). The Romer (1990) model for economic growth, incorporates knowledge accumulation as a dynamic endogenous variable. Cantner & Pyka (1998) argued that technological change and innovation is driven by both technological capabilities and 'know-how' or knowledge. These two interrelated aspects of technological evolution have been incorporated in the agent-based model as two separate variables (discussed in §4.3.2, pg.41): technological sophistication, τ and knowledge, W. These two variables allow for the implementation of *level* and *composition* effects in determining productivity growth, loosely aligned with the approach of Aghion et al. (2006)[2]. In the sections that follow, the history of technological change is reviewed in order to understand the current importance of technology in a firm's operational decisions. This context motivates the choice of the two aspects of technological innovation.

2.4.1 The evolution of industrial revolutions

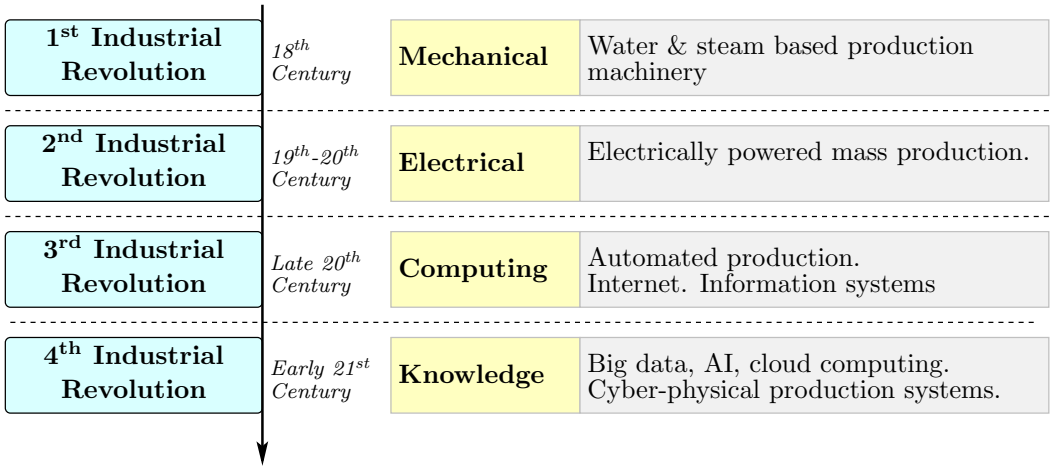


Figure 2.6: Historical summary of industrial revolutions. (Produced based on Schwab (2016))

Historically, scientific discovery and associated technological innovations served as catalysts for industrial revolutions (see Figure 2.6 for an overview). The formulation of Newtonian mechanics, calculus, new production technologies and the invention of the steam engine ushered in the mechanical era of the first industrial revolution, from around 1760 to around 1870 (Coluccia

[2]Refer to §4.3.4, pg.44 for how these effects have been implemented in the model.

2012*a*). It provided a shift from pure human labour as factors of production to the utilization of machinery to do much of the 'heavy lifting' in newly formed factories. A deeper understanding of electromagnetism (from the initial discovery by Ørsted in 1820; to the empirical results of Faraday; the mathematical work of Gauss and Ampère; and culminating in the formulation of a unified theory by Maxwell in 1865 (Griffiths 2008)) led to the electrically driven second industrial revolution from the late 19$^{\text{th}}$ century to the early 20$^{\text{th}}$ century. Electrical technology led to mass production since each machine could be independently powered and controlled by electrical motors. This enabled Ford to introduce the assembly line as an effective and efficient means of production (Coluccia 2012*b*). The 20$^{\text{th}}$ century saw many new scientific discoveries and technological developments leading to massive advances in computing capabilities. Computers changed the nature of business in many ways, resulting in higher levels of automation. For example, production processes were automated using programmable logic controllers (PLCs), supervisory control and data acquisition (SCADA) systems, distributed control systems (DCSs), robotics, etc. (Zhang 2010), while business processes were automated using manufacturing execution systems (MESs), enterprise resource planning (ERP) systems and various business process automation (BPA) systems (Kletti 2007). These systems, designed to automate many aspects of a business operation, also provided huge amounts of data. Data that could then be used in decision making.

The 4$^{\text{th}}$ industrial revolution refers to the evolution and broad application of the technologies developed during the computing revolution of the 20$^{\text{th}}$ century. As the founder of the World Economic Forum (WEF) noted: "digital technologies that have computer hardware, software and networks at their core are not new, but in a break with the third industrial revolution, they are becoming more sophisticated and integrated and are, as a result, transforming societies and the global economy." (Schwab 2016, pg. 12). In other words, digital connectivity coupled with innovations in software technology are fundamentally changing society and business. Some of these innovations include implantable technologies, big data for decision making, cloud computing, artificial intelligence (AI), 3D printing, blockchain, smart homes, the internet of things, etc. (Schwab 2016). Powell & Snellman (2004) has observed that innovation can benefit organisations and consumers in a number of ways. Reducing costs through improved productivity and higher levels of efficiencies are important benefits for a business. Innovation has led to a dramatic proliferation of new goods and services, which is attractive to consumers.

Using data-mining, AI, machine learning, and big data analytics technologies, many companies are now able to exploit their vast amounts of data towards gaining a competitive advantage (Provost & Fawcett 2013). Data and the ability to extract knowledge is becoming an important strategic resource (Smith 2020). This observation has become one of the fundamental principles of data science (Provost & Fawcett 2013).

2.4.2 Knowledge-based economies

The ability to solve complex problems by extracting useful knowledge from data is becoming an important part of a business (Provost & Fawcett 2013, Larose & Larose 2014). Knowledge

has become an important driver for economic growth in many countries (Godin 2006). The Organisation for Economic Co-operation and Development (OECD) defined a knowledge-based economy (KBE) in the following way: "The knowledge-based economy is an expression coined to describe trends in advanced economies towards greater dependence on knowledge, information and high skill levels, and the increasing need for ready access to all of these by the business and public sectors" (OECD 2005). Vadra (2017) proposed four pillars that underpin the successful cultivation of a KBE: (a) a highly educated and skilled workforce, (b) an environment that nurtures research and development, (c) a dynamic and effective information and communications technology (ICT) infrastructure, and (d) a culture that promotes knowledge development. These pillars are closely aligned with the four pillars comprising the World Bank Knowledge Economy Index (KEI) (Chen & Dahlman 2005). Furthermore, the World Economic Forum has identified several technological drivers of change for the foreseeable future: high-speed internet connectivity, AI, big data analytics and cloud-based technology (WEF 2018). Productivity growth has been facilitated by investments in information and communication technologies (Powell & Snellman 2004). Harris (2001) recognised some key features of a KBE that contrast traditional economic thinking. Knowledge is durable, can be repeatedly applied without deterioration, and can be stored and maintained at virtually zero cost. The implication is that knowledge accumulation violates the economic law of diminishing returns. There is also empirical evidence that "the contribution of information technology investments to productivity growth exceeds the contribution of other investments - but only when coupled with significant organizational changes." (Powell & Snellman 2004). Knowledge, innovation and the ability to successfully leverage technology has become an important strategic resource for organisations.

The ability of firms to innovate often leads to improved performance. Andrade et al. (2018) showed that a higher return on investment from research and development (R&D) is achieved as firms approach the technological frontier. The RBV and a firm's dynamic capability may explain why certain firms are better positioned to develop new innovative processes or products. An innovative firm tends to create an environment that fosters further innovative behaviour, resulting in 'innovatability' being an important intangible and dynamic resource for the firm (Saunila & Ukko 2014). There is an increasing need for firms to focus on the role of learning and continuous innovation (Powell & Snellman 2004). Such a firm will tend to be more flexible, perceptive, adaptable and profitable. A firm's dynamic capability, including its ability to innovate and utilise its accumulated knowledge, is therefore closely associated with the firm's ability to learn and embrace change. Once a firm embarks down a path of innovation, it results in a positive feedback loop, often resulting in superior performance.

2.5 The South African situation

South Africa is experiencing persistently poor economic growth, high levels of inequality and poverty (Akanbi 2016). Based on long-run historical data, Akanbi (2016) has demonstrated that there are several causal links between growth, poverty and inequality. It has been said that South Africa is "one of the most unequal societies in the world" (Tregenna & Tsela 2012). Income inequality may be measured using the Gini index, where a Gini index of zero means that there is perfect equality, while a Gini index closer to 1 indicates a completely unequal distribution of income. Inequality levels in South Africa have followed an upward trend since 1993, as can be seen from Figure 2.7(c) and as observed by Tregenna & Tsela (2012). South Africa currently has a Gini coefficient between 0.65 and 0.67 (Statistics South Africa 2019*a*). Some of the reasons for the poor economic performance include: decreasing levels of foreign direct investment (FDI) as can be seen in Figure 2.7(a), high levels of uncertainty (Redl 2018), a struggling education system (Reddy et al. 2015), poor performance by state-owned enterprises, an underperforming manufacturing sector (Ly 2019), and a low savings rate (Hungwe & Odhiambo 2019).

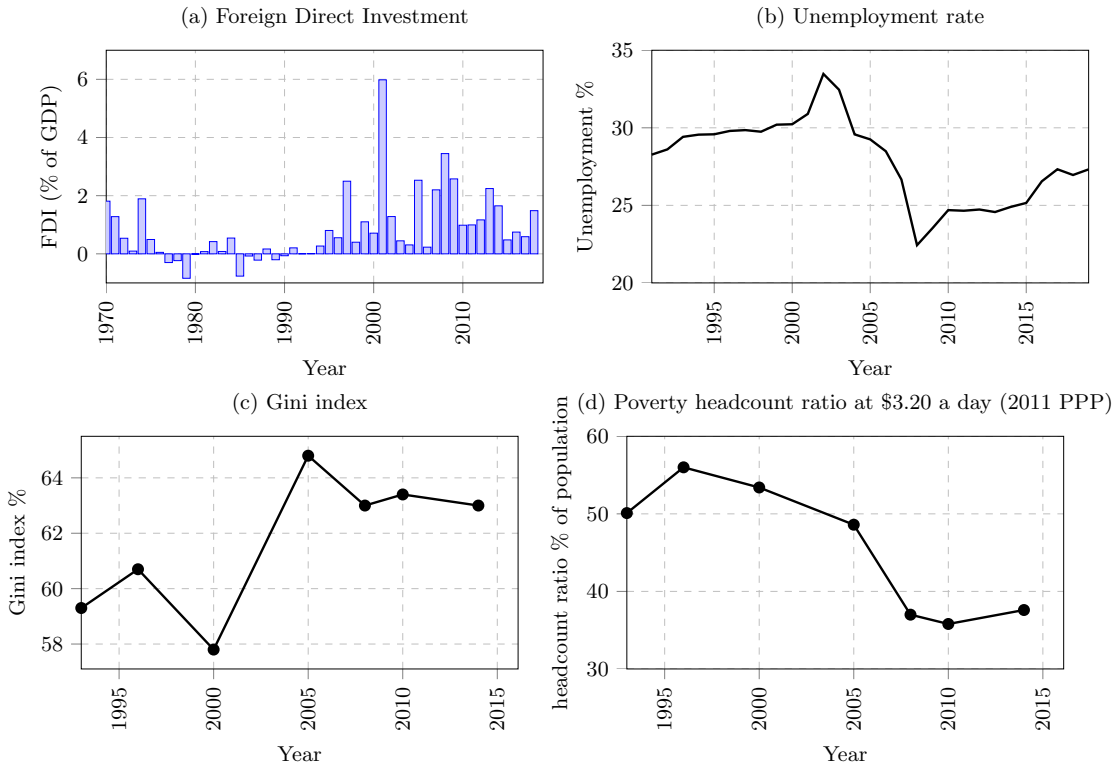


Figure 2.7: Various macroeconomic indicators for South Africa. Raw data source: World Bank (2019)

Recent media articles have noted the potential negative implications of the 4IR on South African unemployment (Biepke 2018, Falkenberg 2018). South Africa has a high unemployment rate fluctuating around 27%, shown in Figure 2.7(b), and a relatively large number of low-skilled workers (Davies & van Seventer 2020). Without significant improvements in education and skills development, the fourth industrial revolution may result in even higher levels of unemployment. The agent-based model developed in this project aims to test this hypothesis.

CHAPTER 3

Agent-based modelling

Agent-based modelling (ABM) is a computational approach for simulating the behaviour of a large number of autonomous agents within an artificial environment, where the agents may interact with each other and their environment (Siegfried 2014). It has become an increasingly useful tool in the study of complex adaptive systems, particularly in social systems including areas such as economics, sociology, geography, ecology and environmental sciences (Billari et al. 2006, Epstein 2006). Complex adaptive systems are characterised by a large number of coupled elements whose properties are interdependent (Gatti et al. 2018, Siegfried 2014).

The fundamental ingredient of ABMs is the notion of an *agent*, that can be defined as[1]: a discrete, heterogeneous, software entity or object, with a set of unique defining attributes or characteristics, that exists within some artificial environment, is capable of autonomous action governed by a set of behavioural rules, and interacts with other agents in order to meet its objectives (Siegfried 2014). Agent's are assigned certain characteristics or attributes and possess behavioural functionality that encodes their interactions with other agents as well as the environment. The environment could also be attributed with characteristics that may either be externally determined or altered based on the interactions by the agents.

The distinguishing feature of an ABM is that it is a 'bottom-up' approach to modelling in which one models the micro-level behaviour, decision-making and interactions of the various agents in the system (Grow & Van Bavel 2017). An important consequence of the ABM approach is the concept of *emergent phenomena*. The term emergence is used to characterise aggregated structures that may arise from simple rules (Delli Gatti et al. 2011). Explicit modelling of the behaviour of individual agents and interactions between agents results in emergent macro-level phenomena, as represented in Figure 3.1. Behavioural rules which are explicitly incorporated within the agent, describe how the agent makes decisions in different circumstances. Macro-level phenomena then emerge from the aggregated results of all the

[1]Note that there is no universal agreement of the definition of an agent (Macal & North 2008).

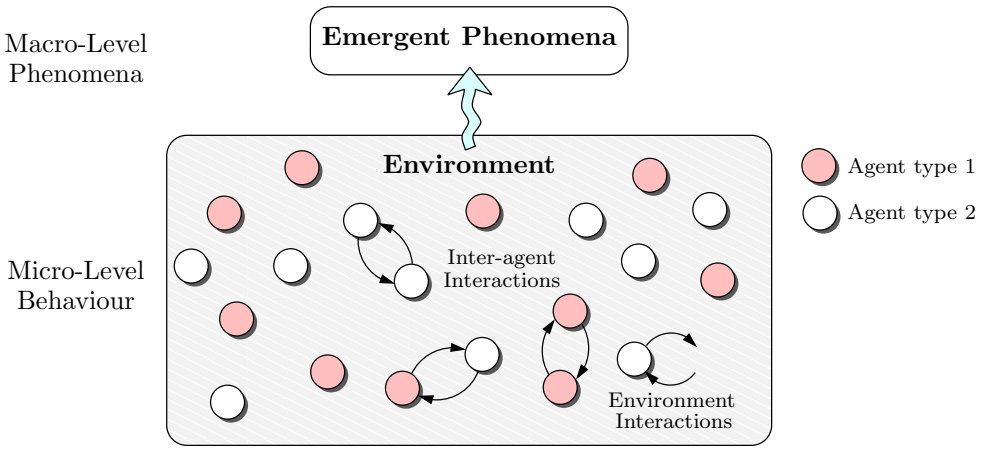


Figure 3.1: The concept of an agent-based model.

micro-level interactions between a large number of agents (Siegfried 2014). An advantage of agent-based modelling is that it provides a way to study the interplay between different scales within a system since it is possible to interrogate the relationship between micro-level behaviour and macro-level phenomena (Gatti et al. 2018). For example, macroeconomic consumption within a country is a consequence of the aggregated consumption decisions of all the households in the country. The emergence of macroscopic behaviour from microscopic interactions is an important principle in physics, where for example, the empirical laws of thermodynamics emerge from the quantum interactions of a large ensemble of particles using the theoretical framework of statistical mechanics (Pathria 1996)[2]. Agent-based modelling has therefore attracted interest in the econophysics research community (Abergel et al. 2013).

3.1 Agent-based modelling in a nutshell

The basic features and characteristics of agent-based modelling and simulation will be discussed using a simple illustrative example. There are three basic ingredients in an agent-based model: *agents* as the active elements in the model, an *environment* in which the agents operate and a *scheduler* that determines the sequence and timing of events in the simulation. A key feature is that

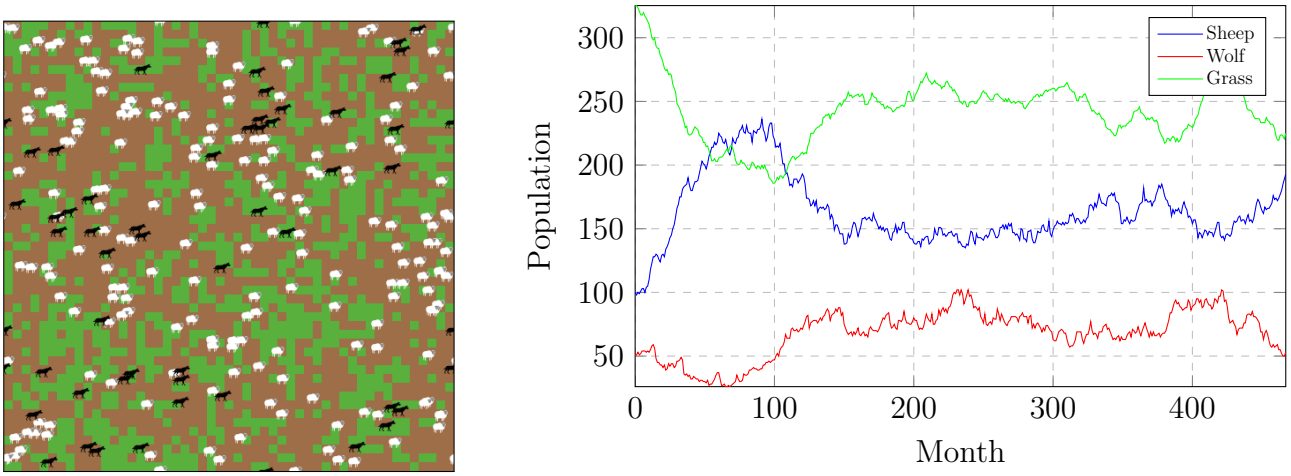
> "the only equations present are those used by individual agents for decision making. Different agents may have different decision rules and different information; usually, no agents have global information, and the behavioral rules involve bounded computational capacities - the agents are 'simple'." (Axtell & Epstein 2006, pg. 152)

3.1.1 An illustrative example

To make the concepts concrete, a simple agent-based model of the interactions between predators (wolves) and prey (sheep) will be used as an example (Wilensky 1997). The results shown

[2]Previously I worked on a research project that was related to spacetime geometry and gravity as possibly being an emergent phenomenon (Giovannoni et al. 2011).

in Figure 3.2 are from a NetLogo[3] (Wilensky 1999) example simulation. The Wolf Sheep Predation model (Wilensky 1997) explores the population dynamics of a predator-prey ecosystem. In population dynamics, the Lotka-Volterra model is the classical model that is typically used to study population dynamics of predator-prey systems (Wilensky & Reisman 2006). In the classical approach, relationships are established between changes in the population densities between the various populations, typically expressed as a system of coupled differential equations (Polking et al. 2006). In contrast to the classical approach, agent-based models are primarily concerned with specifying the behaviour of individual agents, with the population dynamics 'emerging' from the collective interactions between the agents. Within a partially grassy environment, illustrated in Figure 3.2(a), a population of wolves and sheep are created. Both sheep and wolves are allowed to randomly wonder through the environment. In order to survive, sheep need to encounter grass patches to consume, while wolves need to encounter a sheep to consume. If an agent does not feed within a period of time it will eventually starve to death. If two agents of the same type encounter each other, they produce offspring. Figure 3.2(b) shows the resulting emergent population counts based on the interaction of the wolf and sheep represented in the environment shown in Figure 3.2(a).



(a) The wolf-sheep environment.

(b) Population results from a simulation

Figure 3.2: An example NetLogo Wolf-Sheep-Grass ABM simulation Wilensky (1997).

Agents are the basic elements that are used in the construction of an agent-based model. In the example shown in Figure 3.2, there are two types of agents: wolves and sheep. The various features of the definition of an agent will now be unpacked in the context of the example. Some of these generic features may be understood by referring to Figure 3.3 (Nikolic & Kasmire 2012).

As discrete software objects, each type of agent encapsulates its own set of particular *attributes* that can be used to characterise its state at any point in time. Each type of agent is heterogeneous in the sense that it has the same set of attributes that are used to characterise it. All sheep have the same basic attributes, as do all wolves. It is also possible that different agents have common attributes. For example, both sheep and wolves may have age and

[3]'NetLogo is a multi-agent programmable modeling environment.' It allows for the easy creation of simple agent-based models. `https://ccl.northwestern.edu/netlogo/`

energy as attributes. This has practical consequences, especially when implemented using an object-oriented programming (OOP) language. Both agents can be defined as sub-classes of an abstract agent class, thereby exploiting OOP features of inheritance and polymorphism. Over time, an agent may also undergo internal state variable changes. In the wolf-sheep example, the energy would represent the state of hunger of the agent. A wolf that has not eaten for a long time would have its energy levels depleted until it may eventually die. Figure 3.3 shows that attributes are the internal properties of the agent that essentially define the state-variables of an agent.

In addition to the attributes that define the state of the agent, each type of agent would also possess a set of *behavioural rules* that determines that nature of its external interactions and how its internal state can evolve. The behavioural rules are dependent on the current state of the agent as determined by the attributes and may result in a change of the state. "Agents are characterized by simple behavioral rules (methods acting on attributes), that is stylized (algorithmic) patterns of economic behavior. Each agent may follow different ... rules due to different circumstances, i.e. different time periods, geographical areas, markets, and so on." (Delli Gatti et al. 2011, pg. 31). An important question in developing an agent-based model is *what are the characteristics and behaviours of each agent?*

Agents can interact externally with other agents in the simulation or with the environment. Various types of interactions are possible, as shown in Figure 3.3. Attributes may either be changed through interactions with other agents in the system, by interactions with the environment or by behavioural changes over time. Each agent is therefore autonomous since the evolution of the agent throughout the simulation is dependent on its state and unique external interactions, and is not governed by a central coordinating mechanism (Gatti et al. 2018).

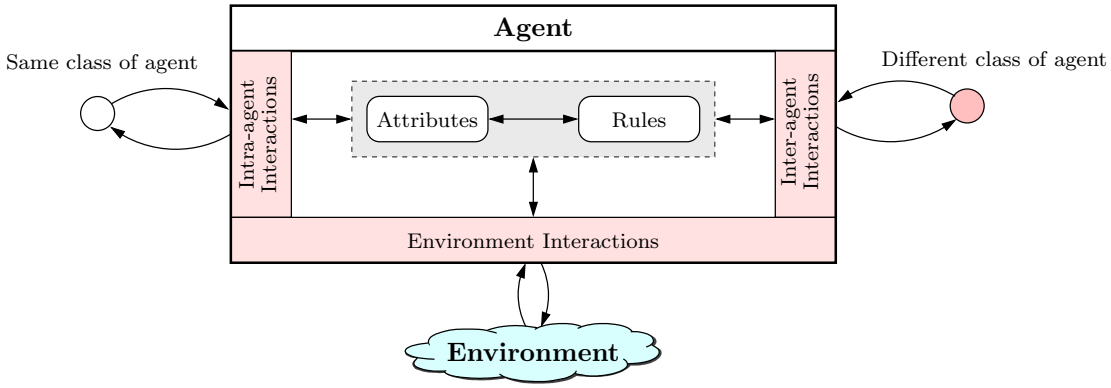


Figure 3.3: Diagrammatic representation of agents and their interactions.

Considering the wolf-sheep example, there are a number of rules that govern the behaviour of the agents (Wilensky & Reisman 2006). At each time-step of the simulation, controlled by the scheduler, each sheep and wolf moves to a new location in the environment, which results in a reduction of energy due to its movement. If a wolf encounters a sheep and is hungry (has low energy), then it will eat the sheep. This defines an *inter-agent interaction* rule. If a wolf encounters another wolf then the wolf may reproduce, based on predefined

procreation probabilities. This is an example of an *intra-agent interaction* rule. A wolf that does not encounter any sheep for a long time may eventually deplete its energy and would then die of starvation. Over time the agents would also age and would eventually die of old age. These are examples of *internal* behavioural rules. The environment consists of regions with and without grass. If a sheep has moved into a region with grass then it may graze, thereby replenishing its energy and also reducing the grass density in that region. This is an example of an *environmental interaction*.

Agents typically exist in some environment. In certain cases the environment would represent some physical space as in the wolf-sheep example. In the economic context the environment may incorporate government policy, the physical location of the agents, the types of markets, or any other exogenous parameters used in the model.

3.1.2 ABM implementation

Boero et al. (2015) note that object-oriented programming (OOP) is a natural programming paradigm for implementing agent-based models[4]. Object-oriented approaches to software design and implementation have become one of the dominant software development paradigms since the 1980s (Booch et al. 2007). Most modern, general purpose programming languages such as C++, Python, Java, C#, etc. include features for object-orientation. The object model brings together many software analysis, modelling, design, and implementation principles in a synergistic way (Booch et al. 2007). Object-orientation integrates a number of key concepts: encapsulation, abstraction, inheritance, and polymorphism.

Objects are central to the object-oriented paradigm. Booch et al. (2007) defines an *object* as a "tangible entity that exhibits some well-defined behaviour". One of the most important concepts in OOP involves the creation of *abstract data types* that are used to specify the nature of objects. In OOP an abstract data type is defined by a *class*. A class is a program blueprint that serves as a template for creating objects (Hillar 2015). Essentially a class is the predefined data type for an object that encapsulates internal state variables as attributes or fields, together with the possible behaviours of the object. In other words a class contains a collection of variables or attributes, together with functions (procedures, behaviours or actions) that can manipulate the variables. The functions or behaviours are often referred to as *methods*. An object is an *instance* of class, meaning that it is a particular implementation of the class.

There are a number of general ABM tools, libraries, and development environments available to aid implementation. Abar et al. (2017) provides a review of available tools. A popular development environment that is useful for rapid prototyping is the NetLogo framework discussed previously (Wilensky 1999). Some programming libraries include a C-library, Swarm (Minar et al. 1996); a Python library, Mesa (Masad & Kazil 2015); and a high-performance computing framework, FLAME, (Kiran et al. 2010).

[4]Refer to Appendix C.1 on page 96 for a brief overview of object-oriented design terminology and concepts.

3.2 Agent-based modelling in economics

Agent-based computational economics (ACE) is the term that is often used to describe the application of agent-based techniques to economic modelling (Gatti et al. 2018). Agent-based approaches have become a popular tool to economic modelling in a wide variety of economic applications (Boero et al. 2015). The appeal is due to the fact that simple behavioural rules determine how agents interact within a complex environment (Ashraf et al. 2017). The usefulness of agent-based economic modelling is that it provides a way to integrate theoretical models with empirical data, allows for the exploration of causal effects, provides a way to incorporate different levels of heterogeneity of the economic agents, is usually scalable, and can integrate social dynamics (Boero 2015). A major advantage of agent-based models is that there is no assumption of equilibrium conditions. "In fact, equilibrium in its traditional meaning is not a requirement in agent-based modelling, and if any type of equilibria emerges, it must be an endogenous outcome generated by the interactions among the agents." (Gatti et al. 2018, pg. 12). Generally there is no need to assume that individual agents will seek an optimal condition. It is possible to encode and test the effect of different behavioural rules (Hamill & Gilbert 2016). A criticism of agent-based models used for economic modelling is the lack of standardisation; for example in the large choice of alternative behavioural rules (Hamill & Gilbert 2016).

Agent-based modelling has been used to investigate many aspects of economics, including:

- emergent macroeconomic phenomena (Gualdi et al. 2015, Delli Gatti et al. 2005, Lengnick 2013),
- financial markets (Grilli & Tedeschi 2016),
- the impact of technology on economic growth (Napoletano et al. 2005),
- the effect of inflation on macroeconomic performance (Ashraf et al. 2014),
- the role of banks in an economy (Ashraf et al. 2017),
- the impact of economic policy decisions (Dawid et al. 2014, Riccetti et al. 2017, Bonaventura 2011),
- the impact of consumer behaviour (North et al. 2010, Tsekeris & Vogiatzoglou 2011) and
- the dynamic impact of various types of firms (Assenza et al. 2015, Ikeda et al. 2007, Ciutacu & Micu 2015).

A recent review article by Dawid & Delli Gatti (2018) summarised and compared a number of macroeconomic agent-based models. The various models incorporate some or all of the economic elements shown in Figure 3.4. Lengnick (2013) argues that there are two categories of ACE models: ones that try to accurately mimic real-world economies, and ones that are more stylized or abstract that try to develop a deeper understanding of macroeconomic theory. The simplest MABM models, such as the model by Lengnick (2013), incorporate only two types of

agents: households and firms. They only interact via two markets: the consumer goods product market and the labour market. More complex MABMs incorporate more types of agents (such as government, banks, different types of firms, etc.) that would interact via additional markets. Markets therefore serve as an important mechanism for interaction amongst economic agents.

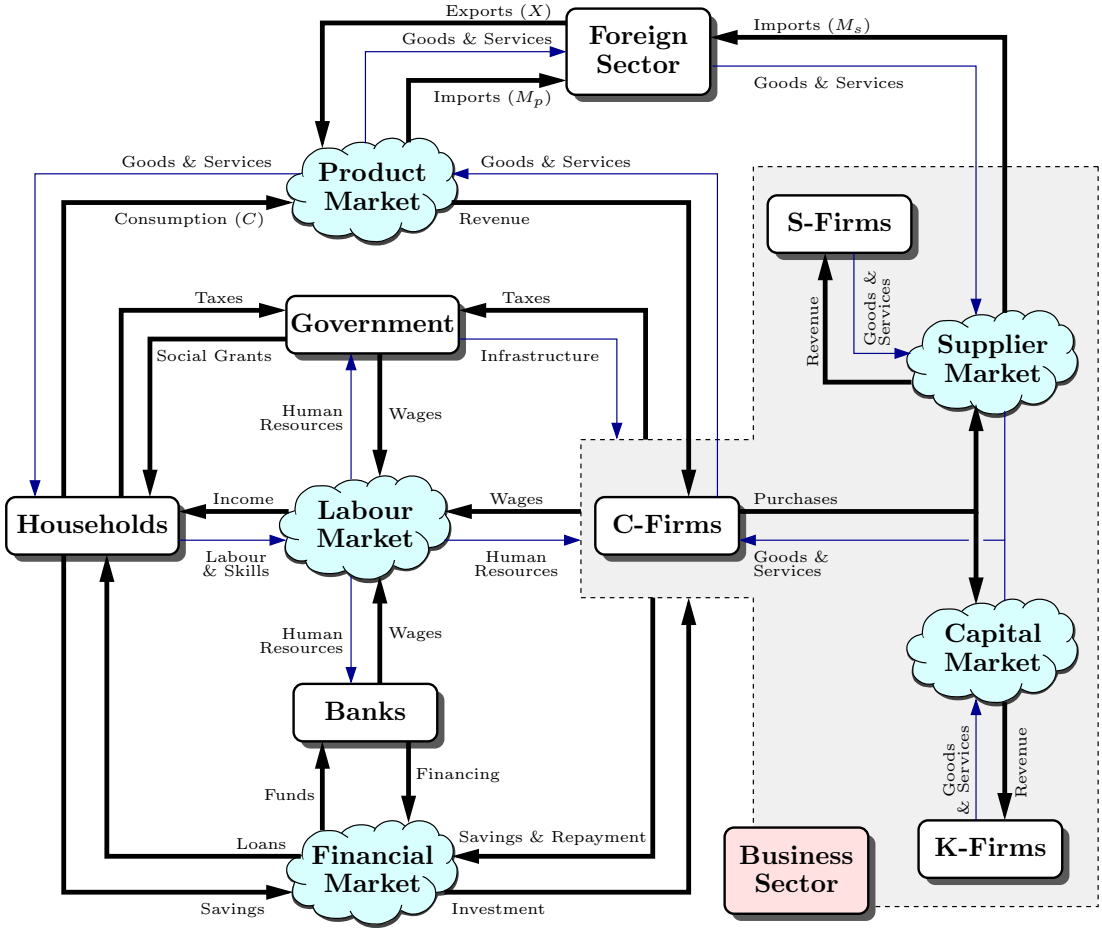


Figure 3.4: A typical economic framework used as a basis for macroeconomic agent-based models.

Agent-based approaches contrast other approaches to economic modelling such as the investment-savings and liquidity preferences-money supply (IS-LM) model (Rode 2012), various endogenous and exogenous growth models (Aghion & Howitt 1998, Romer 2011), and dynamic stochastic general equilibrium (DSGE) models (Blanchard 2018, Lengnick 2013). The IS-LM model is a simple model that essentially assumes equilibrium conditions in goods and asset markets. DSGE models are quantitative models of economic growth or business cycle fluctuations, based on microeconomic foundations that attempt to explain aggregate macroeconomic phenomena (Christiano et al. 2018). They are dynamic because they evolve with time; stochastic in that they involve random events, disturbances or imperfections; and general equilibrium refers to the model incorporating the behaviour of supply and demand for the economic system as a whole. The model is constructed from small, homogeneous models of representative agents, which include households, firms, government, etc., that are aggregated over time to produce macroeconomic effects (Costa 2016). Despite their dominant position in macroeconomic research, recently DSGE approaches have come under criticism, particularly for failing to predict the financial crisis (Blanchard 2018, Christiano et al. 2018).

3.2.1 Households

One of the essential agents in all MABM models is a *household*. Households serve as the primary consumer of goods and services in the economy as well as providing labour to firms. The behavioural rules governing households therefore focus on consumer spending patterns (Carroll 2001, Bremer et al. 2017) and how their evolving skills affect employability (Kinsella et al. 2011). In models that include a financial sector, the household savings is used to finance investment (Dawid et al. 2016). For simplicity many models fix the number of households (Assenza et al. 2015, Gualdi et al. 2015, Lengnick 2013).

The income of households is primarily through salaries (Lengnick 2013), but some models include firm dividend payouts (Dawid et al. 2016, Gualdi et al. 2015). Accumulated cash of households will increase with income and decrease with spending on consumption (Lengnick 2013). At a time t, the cash can be calculated, using

$$\text{Cash}(t) = \text{Cash}(t-1) + \text{Income}(t) - \text{Spending}(t). \tag{3.1}$$

Most MABM implementations do not allow the net cash reserve of households to go negative by constraining the spending budget. A portion of cash or income can be saved, where savings may then be used for investment in firms (Dawid et al. 2016).

A household agent would have behavioural decision-making rules associated with how it would purchase goods from one or more firm agents. Key decision-making rules for households define their consumption behaviour, which would include: establishing budgetary constraints, determining consumption preferences, and the selection of firms from which to purchase goods. Households may allocate a fixed fraction of their budget to consumption (Gualdi et al. 2015, Kinsella et al. 2011) or attempt to spend all of their available cash on consumption (Lengnick 2013). Various other approaches to budgeting by households involve optimization techniques or anticipation of current and future earnings (Dawid & Delli Gatti 2018). Where multiple products are available, product selection can be based on probabilistic choices incorporating product utility (Dawid et al. 2016). The approach of Ashraf et al. (2017) is for consumers to choose a 'consumption bundle' (a selection of goods and services) in such a way as to maximize a utility function. Households in Lengnick (2013) only purchase from a small subset of firms, establishing fixed trading relationships with those firms. In Gualdi et al. (2015), each household selects a fixed number of firms, sorts them by product price and then buys from them sequentially from lowest to highest price. Any remaining cash is then added to savings. Some agent-based models incorporate skill level as a property of the household agents. Dawid et al. (2016) and Kinsella et al. (2011) allow for the skill level or ability of a worker to evolve over time. This includes levels of education and experience gained over time. In these models, higher skill levels typically result in higher salaries.

3.2.2 Firms

The economic framework in Figure 3.4 (on pg. 27) shows the different types of firms that can be modelled in the business sector. Consumer or C-firms produce final goods and services purchased by households, capital or K-firms supply capital equipment that are purchased by the consumer firms, and supplier or S-firms provide raw materials that may be purchased by consumer firms. All of these types of firms would employ labour, pay taxes, and engage with the financial market, depending on the scope of the particular agent-based model. For simplicity, in this section we will only review the behaviour of C-firms. It is often assumed that a particular firm only produces a single product that it sells for a market adjusted price (Dawid & Delli Gatti 2018). Firms are typically characterised by the product that they manufacture and sell, the price of the product, their production capacity, the number of workers that they employ, and their financial status.

In the different MABM models, a firm's production capacity is determined in different ways, usually characterised by the factors of production and the type of production functions used. Gualdi et al. (2015) only uses labour as a factor of production. Production capacity is determined by market demand, which then establishes labour requirements. Similarly Lengnick (2013) linearly scales production capacity with labour as the only factor of production. In models without labour, production is purely a function of capital (Delli Gatti et al. 2005). Dawid et al. (2016) also incorporates a buffer stock together with anticipated demand in the production planning process. The Leontief-type production function used by Dawid et al. (2016) incorporates capital, labour, and average labour skill levels. Napoletano et al. (2005) uses a Leontief production function but without distinct labour skill levels nor stock accumulation. However, Napoletano et al. (2005) incorporates technological progress in the model by allowing for changing productivity and adaptation of capital stock in order to optimize profitability.

Firms require behavioural rules for how to adapt to their economic environment. A firm agent would typically have production and pricing adaptation rules based on the external market environment, such as high demand for its products. The production rules would result in decisions on how to adapt the firm's resources. This aligns with the resource-based view where decisions are governed by how to optimally utilize scarce resources.

The behavioural rules are used to determine a number of possible firm-level decisions such as production capacity[5], $q_f^{\max}(t)$, and product pricing, $p_{fg}(t)$, where the subscript f is used to index the firm and g is used to index the product. Some of the approaches discussed in the literature, summarised in the review by Dawid & Delli Gatti (2018), assume that each firm has information about the market demand, as well as the prevailing average market price, $\bar{p}_g^{\mathrm{M}}$, for its product, g. It can be argued that, in reality, complete information of market demand for a product would be relatively difficult for a firm to possess. Rather than using market demand as a factor used in making production or pricing decisions, a firm could rather use its own stock levels, $\Delta_f(t)$, as a signal for production and pricing adaptation rules.

[5]Note that the notation used here deviates from what is presented by Dawid & Delli Gatti (2018), but is consistent with the notation used later.

3.2.2.1 Production rules

In agent-based models, firms need behavioural rules for how to adapt their production capacity, $q_f^{\max}(t)$. In MABM research (as outlined in the review by (Dawid & Delli Gatti 2018)), many approaches are used for production adaptation; however, they typically follow the following simple heuristic:

Heuristic 3.1: Simple production adaptation heuristic

```
if market demand is high
then
  the firm must increase production capacity
else if market demand is low
then
  the firm must decrease production capacity.
```

Some MABM implementations, such as those by Dosi et al. (2010), Assenza et al. (2015) and Gualdi et al. (2015), simply scale the production capacity based on estimates for market demand, average market prices or inventories. As an example of a heuristic rule for production adaptation, the simple approach by Gualdi et al. (2015) is demonstrated in the following update rule,

$$q_f^{\max}(t+1) = \begin{cases} q_f^{\max}(1+\gamma_f^y\,\xi) & \text{if } \left[q_f^{\max}(t) = D_f(t)\right] \text{ and } \left[p_{fg}(t) > \bar{p}_g^{\mathrm{M}}(t)\right] \\ q_f^{\max}(1-\gamma_f^y\,\xi) & \text{if } \left[q_f^{\max}(t) > D_f(t)\right] \text{ and } \left[p_{fg}(t) < \bar{p}_g^{\mathrm{M}}(t)\right]. \end{cases} \tag{3.2}$$

where $D_f(t)$ refers to the total demand for the goods produced by firm f at time t, and γ_f^y is a production growth rate parameter between 0 and 1. This update rule encodes the following heuristic:

Heuristic 3.2: Production adaptation heuristic

```
# market demand is high:
if  the firm production satisfies market demand
  and the firms sales price is higher than the average market price
then
  the firm must increase production capacity
#  market demand is low:
else if the firm production exceeds market demand
  and the firms sales price is lower than the average market price
then
  the firm must decrease production capacity.
```

The decision to adjust production capacity is based on comparing its current capacity to an estimation of demand, in conjunction with its pricing relative to the market. If the firm is able to meet the market demand ($q_f^{\max} = D_f$) despite the fact that its prices are higher than the average market price ($p_{fg} > \bar{p}_g^{\mathrm{M}}$), then the firm has a strong incentive to produce more.This set of conditions signals a situation in which the market demand exceeds the available market supply capacity, and the firm should therefore increase its supply capabilities. If the production

capacity of the firm exceeds its demand despite it offering a price lower than the average market price, then the firm should reduce its production capacity. These conditions signal a market in which supply capacity exceeds demand. The amount by which the firm increases or decreases production is governed by the parameter $\gamma_f^y \in [0,1]$ and a uniformly distributed stochastic variable $\xi \sim U[0,1]$. It should be noted that this approach where the production capacity is merely scaled up or down by some factor does not explicitly adjust the factors of production. This rule relies on a firm being able to estimate market demand.

It is often more practical for a firm to use stock or inventories at the end of a sales period as the signal for adjusting production. In other words a firm can use stock as a proxy for measuring demand. The production adaptation rule can therefore take the following general form (Dawid & Delli Gatti 2018)

$$q_f^{\max}(t+1) = \begin{cases} q_f^{\max}(1+\eta_f^y) & \text{if } \left[\Delta_f(t) < \Delta_f^L\right] \text{ and } \left[p_{fg}(t) > \bar{p}_g^{\mathrm{M}}(t)\right] \\ q_f^{\max}(1-\eta_f^y) & \text{if } \left[\Delta_f(t) > \Delta_f^U\right] \text{ and } \left[p_{fg}(t) < \bar{p}_g^{\mathrm{M}}(t)\right]. \end{cases} \tag{3.3}$$

Here Δ_f^L represents the lower limit for inventories, while Δ_f^U represents the upper limit. In this model, a firm will increase production by a factor of η_f^y if there is a shortage of stock and the price is greater than the average market price. Conversely, a firm will decrease production if there is an excess of stock and the price is lower than the average market price. This rule introduces inertia in production adaptation since if the actual stock levels falls between the two limits, there will be no change in production capacity.

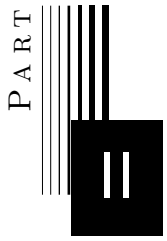
Other models make use of rules to adapt the factors of production in the firm's production function Lengnick (2013). In particular, firms can adapt production output through changing its labour content, capital and/or level of technological sophistication used in the production process. The approach used by Lengnick (2013) utilises a simple heuristic to change a firm's labour based on current stock levels, which is then used as a single factor of production.

3.2.2.2 Pricing rules

Another important parameter that a firm would need to adapt is its product price, p_{fg}, based on the prevailing market price. Dawid et al. (2016) calculates the product price by conducting consumer surveys, in which a firm collects pricing data from a small subset of consumers. A general price adaptation rule used in many MABM models is of the form (Dawid & Delli Gatti 2018)

$$p_{fg}(t+1) = \begin{cases} p_{fg}(1+\eta_f^p) & \text{if } \left[\Delta_f(t) < \Delta_f^L\right] \text{ and } \left[p_{fg}(t) < \bar{p}_g^{\mathrm{M}}(t)\right] \\ p_{fg}(1-\eta_f^p) & \text{if } \left[\Delta_f(t) > \Delta_f^U\right] \text{ and } \left[p_{fg}(t) > \bar{p}_g^{\mathrm{M}}(t)\right]. \end{cases} \tag{3.4}$$

This form is similar to the production adaptation rule in Equation 3.3. A firm will increase its price by a factor of $\eta_f^p \in [0,1]$ if there is a high market demand (signalled by a shortage of stock), and the price is lower than the average market price. A firm will reduce its price by a factor of η_f^p if there is low demand (signalled by high stock levels), and the price is higher than the average market price. Alternatively, Gualdi et al. (2015) uses a production-weighted average price for the market price.



The agent-based model

CHAPTER 4

The model

The agent-based model that has been developed is based on a simplified economy[1] consisting of various types of agents and markets as shown in Figure 4.1. The model consists of 5 types of agents: individuals, consumer firms (C-firms), a capital firm (K-firm), other countries, and government. The agents interact via three markets: a product market, a foreign market and a labour market.

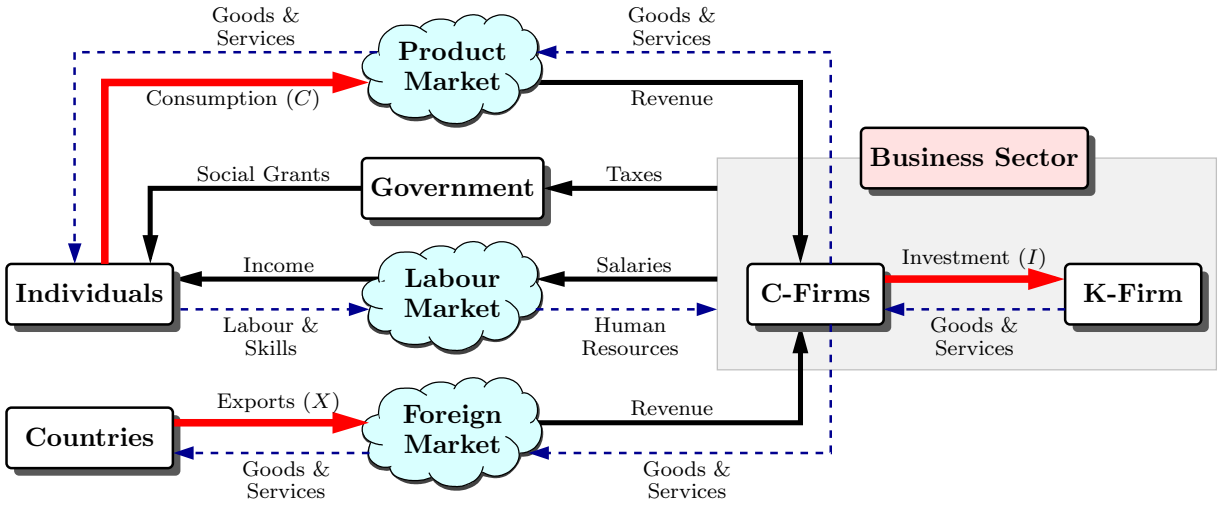


Figure 4.1: The economic framework used as the basis for the simulation. (The thick black lines show money flows through the economy, while the dashed lines show the flow of goods and services. The thick red line shows the money flows that are used to calculate GDP.)

[1] In this chapter 'the economy' refers to the agent-based simulation model.

4.1 The agents

Figure 4.2 shows the class hierarchy of the different types of agents used in the model. The primary consumers in the model are a large collection or a population of individual[2] agents[3]. Each individual agent can age, gain an education, get employed, consume goods and services, retire and eventually die. An individual procures goods and services from consumer firms in the product market, contributing towards GDP by providing *consumption*, C. The second type of consumer agent is a country, which collectively forms the world. Goods and services are bought by each country in the foreign-market, resulting in total *exports*, X. There are a fixed number of consumer firms (or C-firms) that are infinitely lived. Goods and services are produced and sold in the two consumer goods markets by the consumer firm agents. For the sake of simplicity, a single capital-goods firm (or K-firm) is created to supply capital equipment and knowledge or technology resources to the C-firms. The C-firms therefore provide *investment*, I, as the final contributor towards GDP. Both types of firms will employ a number of individual agents that are acquired through the labour market. Note that since there is a single K-firm, it is not necessary to implement a capital-goods market. The final type of agent in the model is a single government agent, which taxes firms and pays out social grants to those individuals that are unemployed. For simplicity, the role of government has been limited to these function. Note that individuals do not pay taxes in the model.

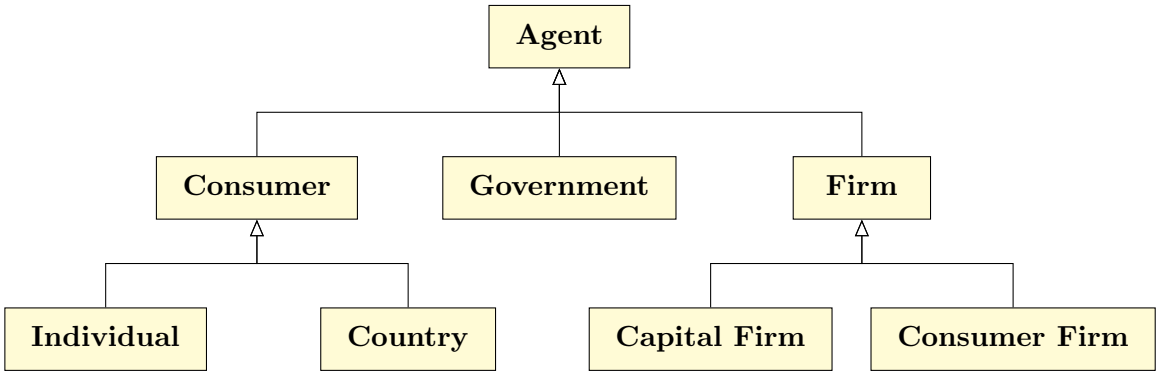


Figure 4.2: The class hierarchy for the various agents used in the model.

This chapter provides an overview of some of the technical aspects of the implementation of the agent-based model. It incorporates many features of ABMs as discussed previously as well as design aspects utilising OOP. A summary of symbols and notation that will be used is provided in Appendix A. Not all aspects of the model are described here, however full implementation details are provided in Appendix C.

[2]See §A.1, pg.87 for details on notation.

[3]For reasons that will become apparent in §4.2, pg.35, the primary consumers in the economy are referred to as *individuals* rather than *households*.

4.2 Individuals

The primary consumers in the economy are individual agents. Individuals also serve as the workforce for firms. The collection of all individual agents forms the population. The simulation is initialised with $\hat{N}_{\mathrm{P}} = 2{,}500$ individuals. Each individual agent is characterised by an age, a level of education, an economic status, a demand for various goods and services, available cash and accumulated wealth.

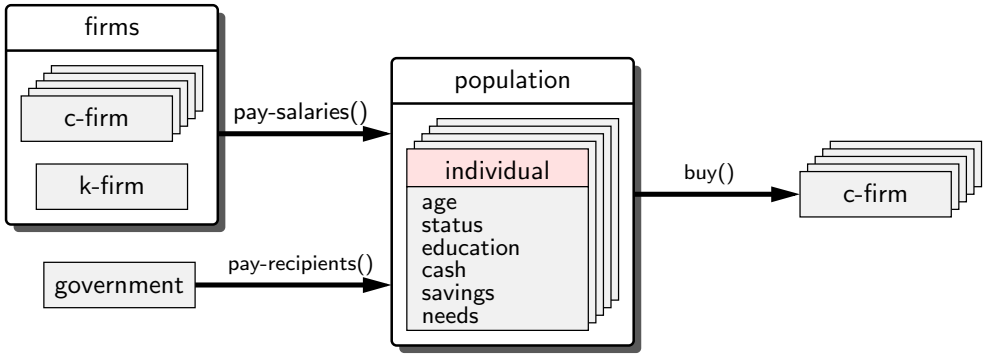


Figure 4.3: An overview of the behaviour and properties of individual agents.

The ABM model presented here differs from many of the other ABM models discussed in Chapter 3. The main difference is the individual agent's ability to evolve over time. The number of individual agents within the population will grow at a predefined growth rate, so that the number of individuals in the economy is not constant as is the case with many other MABM models (Assenza et al. 2015, Gualdi et al. 2015, Lengnick 2013). In addition, each individual will undergo a life-cycle, by being born, age and eventually die. It is for this reason that the consumers are referred to as 'individuals' and not 'households'. Every individual is characterised by several variables and parameters, including

- age (see §4.2.1, pg.36),
- education (see §4.2.2, pg.37),
- status (see §4.2.2, pg.37 and §4.2.3, pg.38),
- needs or demand for products (see §4.2.4, pg.39 and §4.4, pg.53),
- cash and savings.

Individual agents will have an amount of cash representing their disposable income. Income received either from salaries or social grants will increase the cash available. A fixed percentage of cash, dependent on the level of education, will be allocated to savings. After a person has retired they will live off their accumulated savings. A number of other properties required for the product and labour markets are also implemented (see §4.4, pg.53 for details).

4.2.1 Age of individuals

Each individual agent in the population is created with an age variable. The age distribution of individuals in the population is initialised in such a way as to follow the general shape of the South African age distribution curve. The population distribution is approximated by a half-normal distribution function, and so the population of individual agents is initialised with randomly drawn ages that approximately follow a half-normal distribution. The initial distribution of ages in a typical simulation run is shown in Figure 4.4, together with a fitted half-normal distribution curve and the appropriately scaled actual distribution of ages in South Africa.

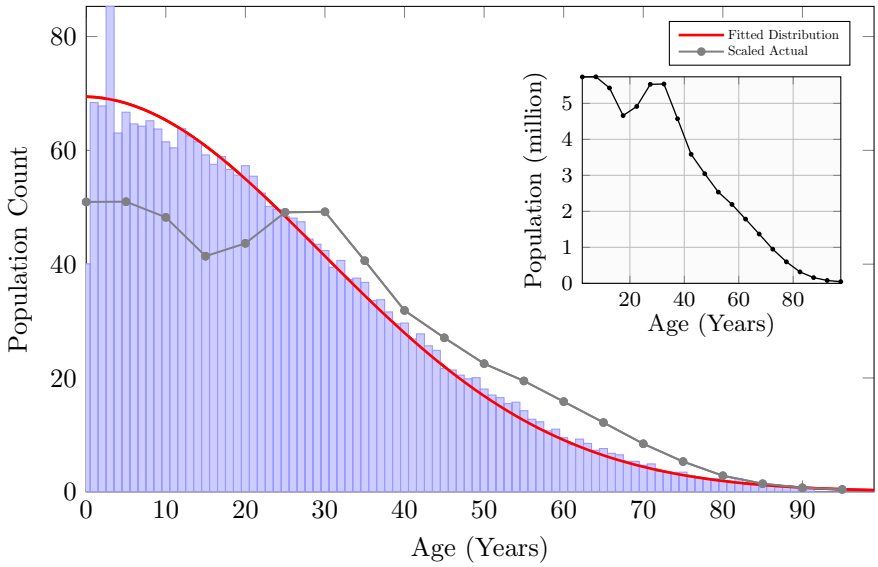


Figure 4.4: Simulation initial population distribution. The insert shows actual South African population age distribution. (Statistics South Africa 2019*b*, Table 11).

Two parameters are used to regulate population growth: the population birth rate $\hat{g}_{\mathrm{B}} = 2.75\%$ per annum, and the overall population growth rate $\hat{g}_{\mathrm{P}} = 1.5\%$ per annum, motivated by recent South African population statistics (Statistics South Africa 2019*b*). The birth rate is used to calculate the number of new births during a year, while the difference in growth rates is used to determine how many individuals will perish over the year. Individual agents are selected to die so that a half-normal population distribution is maintained over time. This approach contrasts the approach of Kinsella et al. (2011) where the agents all live till the age of 75. When the old agent dies a new agent is created, inheriting all the wealth of the parent.

4.2.2 Status of individuals

There are two properties that characterise the role an individual plays in the economy: an education variable or attribute, referred to as individual•education, and a status variableor attribute, referred to as individual•status. The individual•education attribute defines the individual's level of education in a similar way as by Kinsella et al. (2011). The possible values for the individual•education attribute are obtained from the *Education* set, where[4]:

$$\textit{Education} := \left\{ \begin{array}{l} \textit{Unskilled} \text{ (no qualification)}, \\ \textit{Matric} \text{ (semi-skilled)}, \\ \textit{Bachelors} \text{ degree (skilled)}, \\ \textit{Postgraduate} \text{ degree (highly skilled)}. \end{array} \right\} \tag{4.1}$$

Note that the education/skill level terms will be used interchangeably.

As individuals age their role in the economy is determined by the individual•status attribute. When born, they are classified as a *Child*, and after entering the schooling system as *Studying*. They progress through the education system, possibly gaining higher levels of qualification. Upon exiting the education system with a particular level of education, the individual is classified as *Unemployed* and enters the labour market. If a firm decides to employ an individual, the individual•status attribute is set to *Employed*. The individual will then work until retirement. The possible values for the individual•status attribute are obtained from the *Status* set, where[5]:

$$\textit{Status} := \Big\{ \textit{Child}, \quad \textit{School}, \quad \textit{Studying}, \quad \textit{Unemployed}, \quad \textit{Employed}, \quad \textit{Retired} \Big\} \tag{4.2}$$

The various status possibilities are shown in Figure 4.5, together with a status categorisation scheme detailed in Section 4.2.3 on page 38. The categorisation scheme is used to classify particular groups of individuals based on their role in the economy.

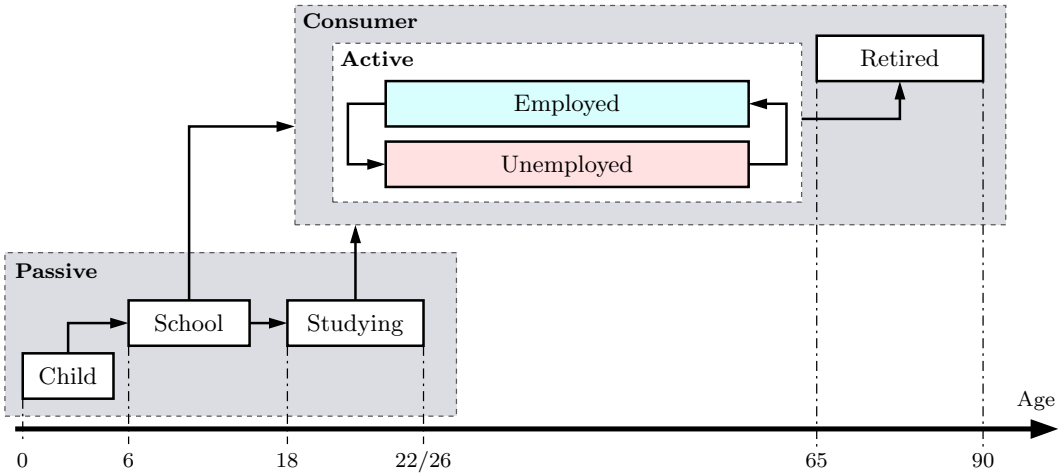


Figure 4.5: A diagrammatic representation of the evolution of the status attribute for individuals.

The change in status of an individual is a function of the age of the individual, random choices or decisions that the individual makes, and whether or not they are employed. The choices

[4]The *Education* set is implemented as an instance of a structured dictionary class, **EducationConstants**.
[5]The Status set is implemented as an instance of a structured dictionary class, **StatusConstants**.

or decisions are based on *progression probability* parameters, denoted by $\mathcal{P}$(decision), with values given in Table C.2 on page 100. At various times when a progression choice needs to be made, a uniformly distributed random number, $r \sim U[0, 1]$, is generated. If r is less than the progression probabilities for that decision, $\mathcal{P}$(decision), then that choice is made. In this way, a decision or choice is only made $100\times$ $\mathcal{P}$(decision)% of the time. For example, if an individual has finished school and obtained a matric qualification (with the individual•education assigned a value of *Matric*), there is a progression probability, $\mathcal{P}$(entering tertiary) = 0.4, that the individual will enter higher education, upon which the individual•status attribute is assigned the value *Studying*. The remaining 60% of those individuals with matric would then enter the labour force as *active* members of the population. The complete set of rules describing the change in status and education of individuals are provided in Appendix C.5 on page 141. The outcome of the progression outlined above is that between the ages of 18 and 26, individuals enter the potential workforce with an individual•status attribute equal to *Unemployed* and an individual•education attribute equal to a value depending on how far the individual managed to progress through the education system.

4.2.3 Population category tree

The population of individual agents are stored in a hierarchical *tree* data structure[6], as shown in Figure 4.6. Each individual agent is assigned to a *leaf* node on the tree, based on their status.

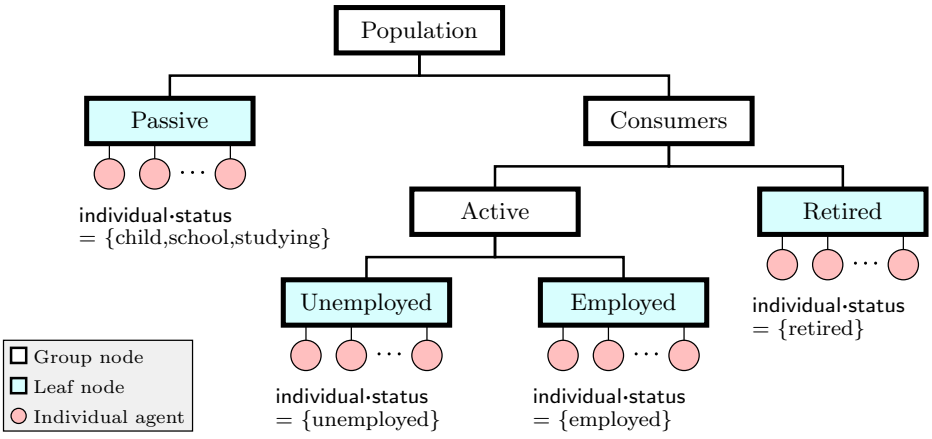


Figure 4.6: The population tree data structure used for categorising individuals.

The motivation for using a tree structure is that it provides a convenient mechanism to select and allocate population groups to the various markets by extracting appropriate subtrees. For example, the subtree of all individuals that are *unemployed* is used as the collection of sellers in the labour market. The *consumers* subtree is used as the collection of buyers in the product market. The *active* subtree of the population refers to those individuals that are employable or are already employed. When calculating unemployment rates it is necessary to consider all of the agents in the *active* subtree, where it is assumed that all unemployed individuals are seeking employment.

[6]The standard Python tree library (**treelib**) was used. The library provides tools for selecting a subtree from the overall tree structure.

4.2.4 Behavioural rules

Individuals require decision-making rules for their consumption behaviour. The individuals that are categorised as consumers receive various sources of income: the employed receive salaries, the unemployed receive social grants and the retired use their accumulated life savings or wealth. Each month, the income received is added to the individual•cash variable. After receiving a salary, employed individuals will save a percentage of their income adding to their accumulated wealth. Any remaining cash is then available to purchase goods from the product market. Details of how markets function is described in Section 4.4 on page 53.

Each month, the consumer individuals will decide on the basket of goods required, analogous to the 'consumption bundle' of Ashraf et al. (2017). The various categories of consumers will each demand a different mix of products, defined as input parameters to the simulation shown in Table C.2 on page 100. Unemployed and retired people will have a goods basket dominated by essential goods, while employed individuals will have a mix of goods that progressively includes more luxury goods at higher levels of education. In the market shopping algorithm, consumers will calculate their needs on a monthly basis and are stored in the individual•needs basket array variable. Consumers will attempt to spend all their cash based on the product proportions in determining their individual•needs basket.

4.3 Firms

The second major class of agents in the economic model are firms that produce consumer goods, referred to as C-firms. A fixed number of C-firms are created and are infinitely lived. The number of firms is scaled based on the initial population size. For the sake of simplicity it is assumed that each C-firm in the economy produces a single product. A single capital producing firm, or K-firm, provides the necessary capital equipment or knowledge infrastructure to the consumer firms. Both types of firms will employ individuals, earn revenue, and pay taxes. Thus the only costs incurred by firms are labour costs and taxation. There are no material costs in the model. It is assumed that salaries paid to individuals are constant.

It is convenient to implement an abstract **Firm** class which will provide the generic functionality of all firms. Details on the design and implementation of the various types of firms are provided in Appendix C.

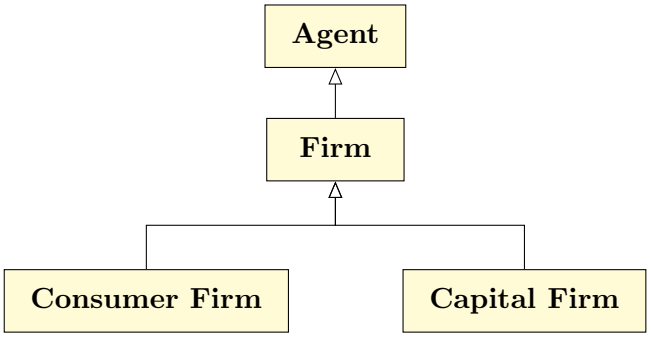


Figure 4.7: The class hierarchy for the firm agents.

C-firms serve several roles in the economy. Firstly, as *producers* they provide the goods to be purchased by individuals. Production is determined by a firm's *production function*, which encapsulates how the firm's resources result in the output of goods and services (see §4.3.4, pg.44). Capital and labour are the primary factors of production. However, technological capabilities and accumulated knowledge are important aspects that need to be incorporated in the production function (Cantner & Pyka 1998). Secondly, within the market model, firms serve as sellers in the product market (see §4.4, pg.53). Thirdly, as *employers* firms hire individuals as their labour force, acting as buyers in the labour market. Finally, C-firms *invest* accumulated wealth to increase production capacity. Therefore each C-firm is defined by several variables and parameters, including

- product (see §4.3.1, pg.41),
- product sales price (see §4.3.5.5, pg.51),
- employees (see §4.3.3, pg.42),
- capital, knowledge and technological sophistication (see §4.3.2, pg.41 and §4.3.4, pg.44).

4.3.1 Products

Each firm will produce a single product, defined by the parameter firm•product chosen from a set of available products referred to as the *Goods* set. This implies that there are multiple products in the economic model, but for simplicity, this has been restricted to only two products. The firm•product parameter is therefore limited to values from the *Goods* set, where[7]:

$$\textit{Goods} := \left\{ \textit{Essential} \text{ goods}, \quad \textit{Luxury} \text{ goods} \right\}. \tag{4.3}$$

The *Goods* set establishes various sectors in the consumer goods market, where each sector is defined by the particular type of product. Each product sector is assumed to have a similar market structure in terms of product homogeneity, pricing, technological sophistication, labour requirements, productivities, etc.

The firm has a sales price, p_{fg} for the product that it sells, initialised with a value $p_{fg} \approx \hat{P}_{\mathrm{G}}[g]$, where $\hat{P}_{\mathrm{G}}$ is a predefined simulation parameter (see Table C.4, pg. 100). Over time the sales price will change according to market driven pricing adaptation rules (see §4.3.5.5, pg.51).

4.3.2 Knowledge and technological sophistication

Two variables are used to define different yet complementary aspects of the technological capabilities of a firm: *knowledge* and *technological sophistication*.

- **Knowledge:** The accumulated knowledge of a firm, incorporating 'know-how', systems, procedures, associated infrastructure, etc., is captured by the knowledge variable, W. Knowledge is treated as an asset, which means that the knowledge variable is measured in currency units. Firms can therefore invest in knowledge in a similar way as investing in capital. A primary difference between capital and knowledge is that knowledge does not depreciate over time. Knowledge is used in the production function for the firm (see §4.3.4, pg.44).
- **Technological sophistication:** The technological capabilities of a firm are represented by the technological sophistication variable τ. Each firm is initialised with a level of technological sophistication, $\tau \sim U[0, 1]$, that grows slowly over time. The lower limit $\tau \approx 0$ represents a firm with low technological capabilities; while $\tau = 1$ represents the *technological frontier* for the industry (Andrade et al. 2018). Technological sophistication is used to establish the skills composition of employees within the firm, discussed in the following section.

Technological progress highlighted in the introduction as an important determinant of economic growth, has been decomposed into these two complementary factors. The role that these variables play within a firm is illustrated in Figure 4.8 and will be described in the sections that follow.

[7]The *Goods* set is implemented as an instance of a structured dictionary class, **GoodsConstants**.

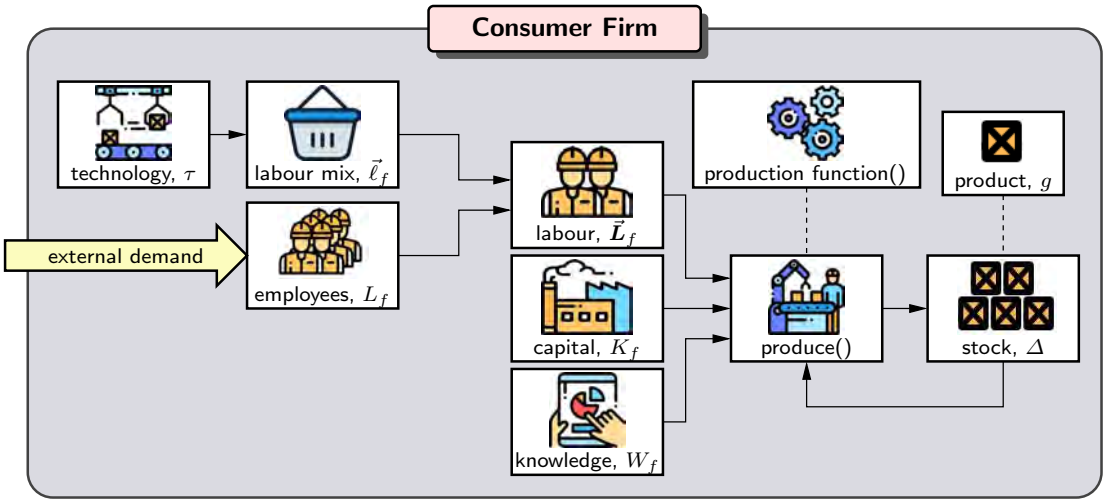


Figure 4.8: An overview of the C-firm properties that contribute towards production.

4.3.3 Employees

A firm agent will employ several individual agents with a suitable mix of skills. Within the **Firm** class the firm•employees variable is implemented as a collection of individual agents. There are two aspects of labour that a firm needs to consider: the total number of employees, and the capabilities of the employees (Bechet 2008). In the model, it is assumed that the total number of staff is determined by production requirements (see §4.3.4, pg.44), while the required capabilities of staff is determined by the technological sophistication of the firm. A C-firm would need to adapt its production capacity in response to market demand, which means that the total number of employees, L_f, would need to adapt to market conditions (see §4.3.5.2, pg.48). The required number of employees in a K-firm will scale with its revenue.

A more technologically sophisticated firm will require a workforce with higher levels of education. The staff capabilities or skills composition of the firm is described by a *qualifications mix* of employees, represented by the array $\vec{\ell}_f$. It is assumed that the qualifications mix is determined by the technological sophistication of the firm, τ. As a firm becomes more technologically sophisticated its labour composition needs to evolve, requiring progressively fewer unskilled workers and more highly skilled employees. At each level of education e, a continuous function $\ell_{fe}(\tau)$ with allowed values between 0 and 1, defines the proportion of the workforce required at that level[8]. Here e represents index values from the *Education* set discussed in Section 4.2.2 on page 37. The required mix of qualifications for labour is structured as an array $\vec{\ell}_f$ whose length is the size of the *Education* set and where each element is a function of the firm's technological sophistication,

$$\vec{\ell}_f \equiv \ell_{fe}(\tau) = (\ell_{f0}(\tau)\ \ell_{f1}(\tau)\ \ell_{f2}(\tau)\ \ell_{f3}(\tau))\,. \tag{4.4}$$

Since each of the array components, $\ell_{fe}(\tau)$, represents the required proportion of a firm's

[8]The approach used is inspired by fuzzy membership functions from the field of fuzzy set theory (Siler & Buckley 2005).

workforce with a qualification e, they must sum to 1

$$\sum_{e=0}^{3} \ell_{fe}(\tau) \stackrel{!}{=} 1, \quad \text{for all values of } \tau. \tag{4.5}$$

For example, if $\vec{\ell}_f = (0.4, 0.3, 0.2, 0.1)$, then 40% of the workforce must be unskilled, 30% must be semi-skilled, 20% must be skilled, and 10% must be highly skilled.

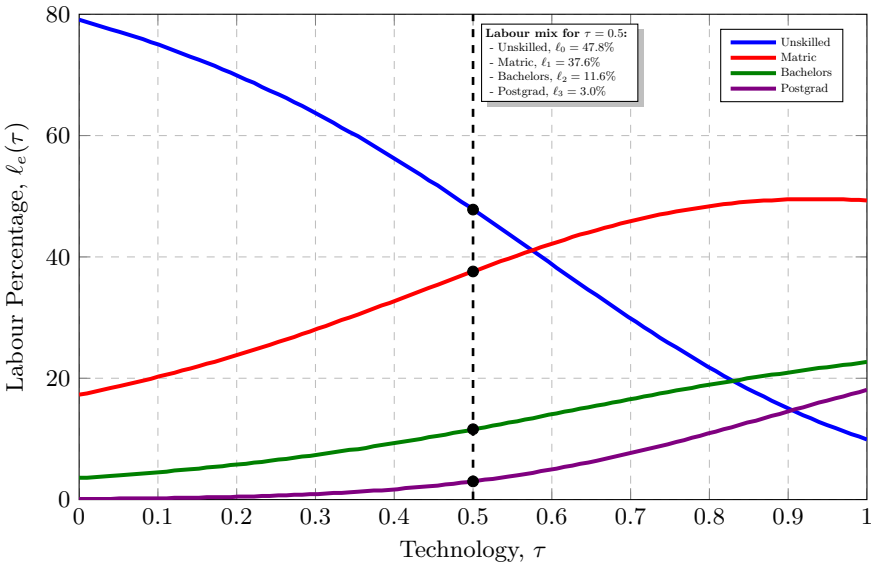


Figure 4.9: Qualifications mix functions used in the simulation to determine the labour composition based on technological sophistication.

The various functions used for $\ell_{fe}(\tau)$ are shown in Figure 4.9. The functional forms used show that as τ increases there is a steady decrease in the requirement for unskilled labour, while skilled labour becomes more desirable. The figure shows the labour mix percentages for $\tau = 0.5$. The required labour for a firm is then obtained as the number of employees multiplied by the qualifications-mix array, resulting in the *labour composition array*,

$$\vec{\boldsymbol{L}}_f(\tau_f) = L_f \, \vec{\ell}_f(\tau_f). \tag{4.6}$$

The total labour composition array is used as the basis for various baskets used in the labour market (see §4.4, pg.53). For example, if $\vec{\ell}_f = (0.4, 0.3, 0.2, 0.1)$ and the number of employees is $L_f = 50$, then the labour composition of the firm is 20 unskilled workers, 15 semi-skilled, 10 skilled, and 5 highly skilled. The firm•requirements basket is the desired labour composition, the firm•possess basket is the actual employee composition, and the firm•needs basket is the array of labour that needs to be changed. The firm•requirements basket is adjusted based on technological sophistication and product demand, and each month prior to shopping in the labour market, the firm will calculate its firm•needs basket, using

$$\text{firm•needs} = \text{firm•requirements} - \text{firm•possess}. \tag{4.7}$$

Strategic labour adjustments will modify firm•requirements as described in Section 4.3.5.2 on page 48. The firm•needs is used to calculate aggregate demand in the labour market.

4.3.4 Production

The production capacity for each firm is assumed to depend on the total labour mix of employees, $\vec{\boldsymbol{L}}_f$, the available capital, K_f, and the level of accumulated knowledge, W_f, giving rise to a production function

$$q_f = F_f(K_f, \vec{\boldsymbol{L}}_f, W_f). \tag{4.8}$$

The firm capital, measured in currency units, represents the total investment in plant and equipment. Capital depreciates over time at a depreciation rate $\hat{d}$ =0.01 % per month.

Each C-firm is assumed to follow a Cobb-Douglas type production function[9] that is used to calculate the production capacity of a firm. The production function used in the model is given by

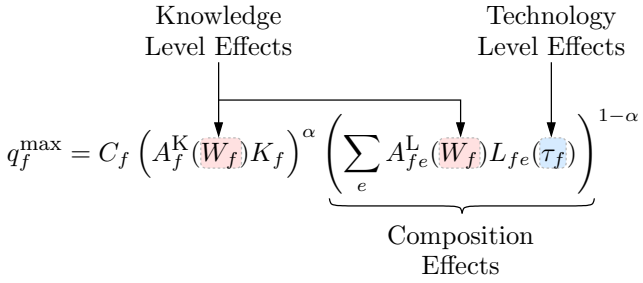
$$q_f^{\max} = C_f \left(A_f^{\mathrm{K}}(W_f)K_f\right)^{\alpha_f} \left(\sum_e A_{fe}^{\mathrm{L}}(W_f)L_{fe}(\tau_f)\right)^{1-\alpha_f}, \tag{4.9}$$

where C_f is an overall production scaling parameter, A_f^{K} is the capital productivity, A_{fe}^{L} are the labour productivities for each education level, and α_f is the output elasticity for capital[10]. The production function provides the maximum potential output for the firm. The actual production in a month, $q_f(t)$, is dependent on the remaining stock from the previous month, $\Delta_f(t-1)$, that is

$$q_f(t) = q_f^{\max}(t) - \Delta_f(t-1). \tag{4.10}$$

4.3.4.1 Composition and level effects

The modelled components of technological progress, knowledge W and technological sophistication τ, affect the firm production function in different ways. The key idea is that knowledge influences productivity, while technological sophistication influences labour composition. The form of the production function incorporates both *level* and *composition* effects, identified as follows:



The composition effects are achieved through the various labour mix array components combined with the associated labour productivities. There are two level effects: a *technology level effect* that influences the labour composition as discussed in the previous section, and a *knowledge level effect* that impacts productivities. The capital and labour productivities are assumed

[9] See review of Cobb-Douglas production functions in §2.3.1, pg.15.
[10] See §2.3.1, pg.15.

to be functions of knowledge. For simplicity a linear relationship is assumed between knowledge, W_f, and each of the productivity parameters A_f^{K} and A_{fe}^{L}. The relationships used for the capital and labour productivities used in the model are shown in Figures 4.10 and 4.11 respectively[11].

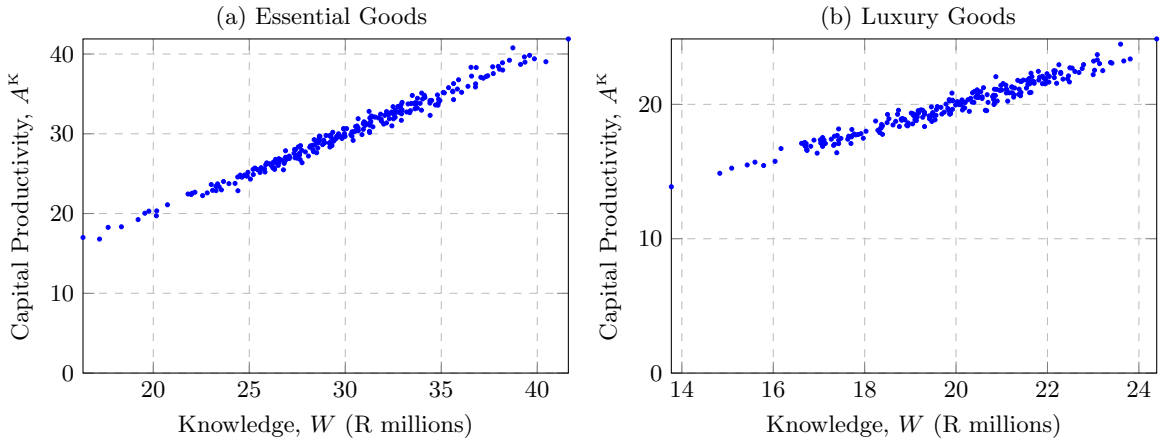


Figure 4.10: Firm capital productivity as a function of knowledge for each product sector.

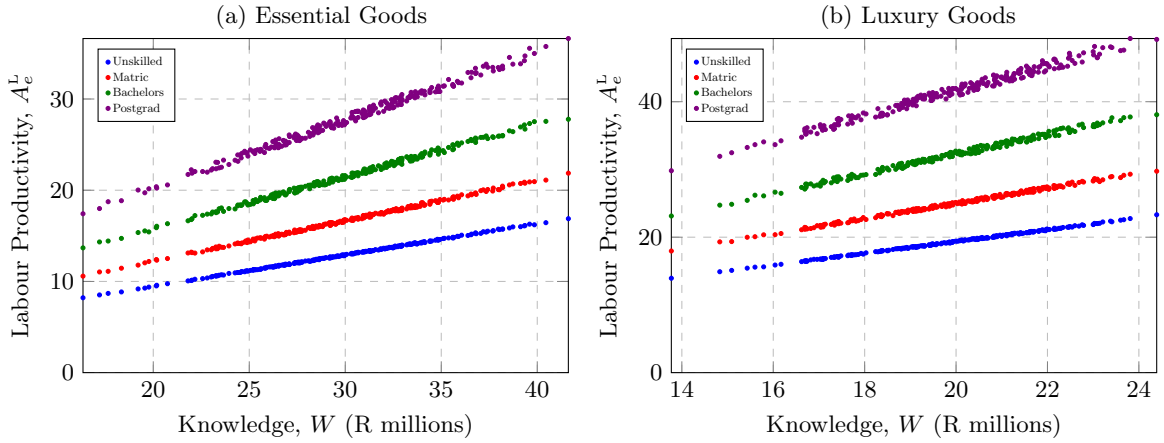


Figure 4.11: Firm labour productivities as a function of knowledge for each product sector and different education levels.

4.3.4.2 Initialisation

Before creating the C-firms, total demand for each good is estimated based on the active population in the economy and the estimated demand for exports. From the estimated total demand, the total requirement for production is calculated for each sector. The production is then split between the firms in the sector, together with a normally distributed variation, giving an initial production capacity for each firm, $q_{f0}^{\max}$. Initial values for technological sophistication and knowledge are drawn from a uniform distribution. Based on the firm's initial capacity and total labour requirements, the firm's initial labour count L_f is estimated, which is then used to establish the production function. When created, each firm's production function is initialised

[11] In this section, to illustrate the relationship between the variables and parameters, a large number of firms were created and the various parameters or variables were then extracted as shown. The relationships are shown for each product sector.

with its own linear functions for the various productivities. The linear functions are similar for each of the two goods sectors, as can be seen in Figures 4.10 and 4.11. The production function is initialised to output the initial capacity, $q_{f0}^{\max}$, by adjusting the scaling parameter, C_f, and the capital K_f. Figure 4.12 shows the average production function curves based on a large number of typical firms. Figure 4.13 shows the production function curves for different values of labour and knowledge as a function of capital, K.

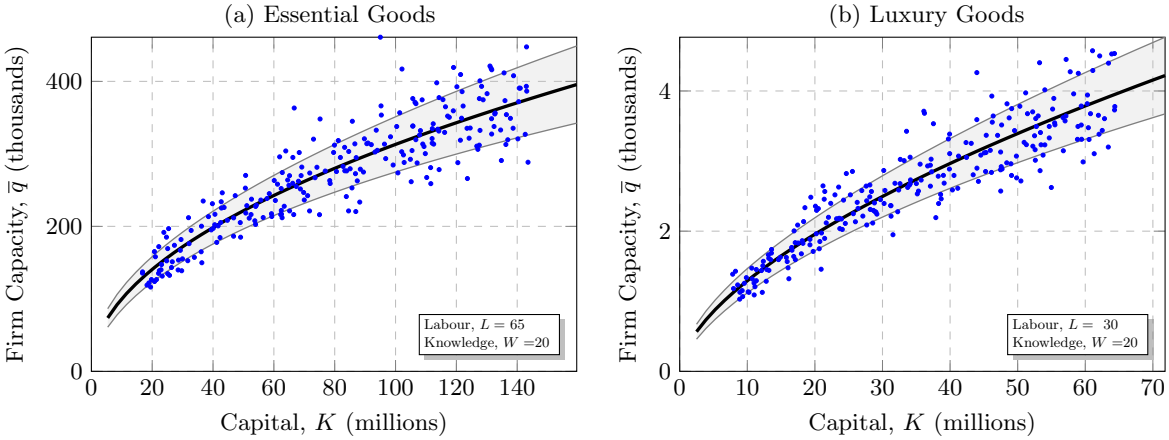


Figure 4.12: Production functions used for the consumer firms in the economy. The data points are samples for a number of firms in each sector showing the variation in production capacities of individual firms.

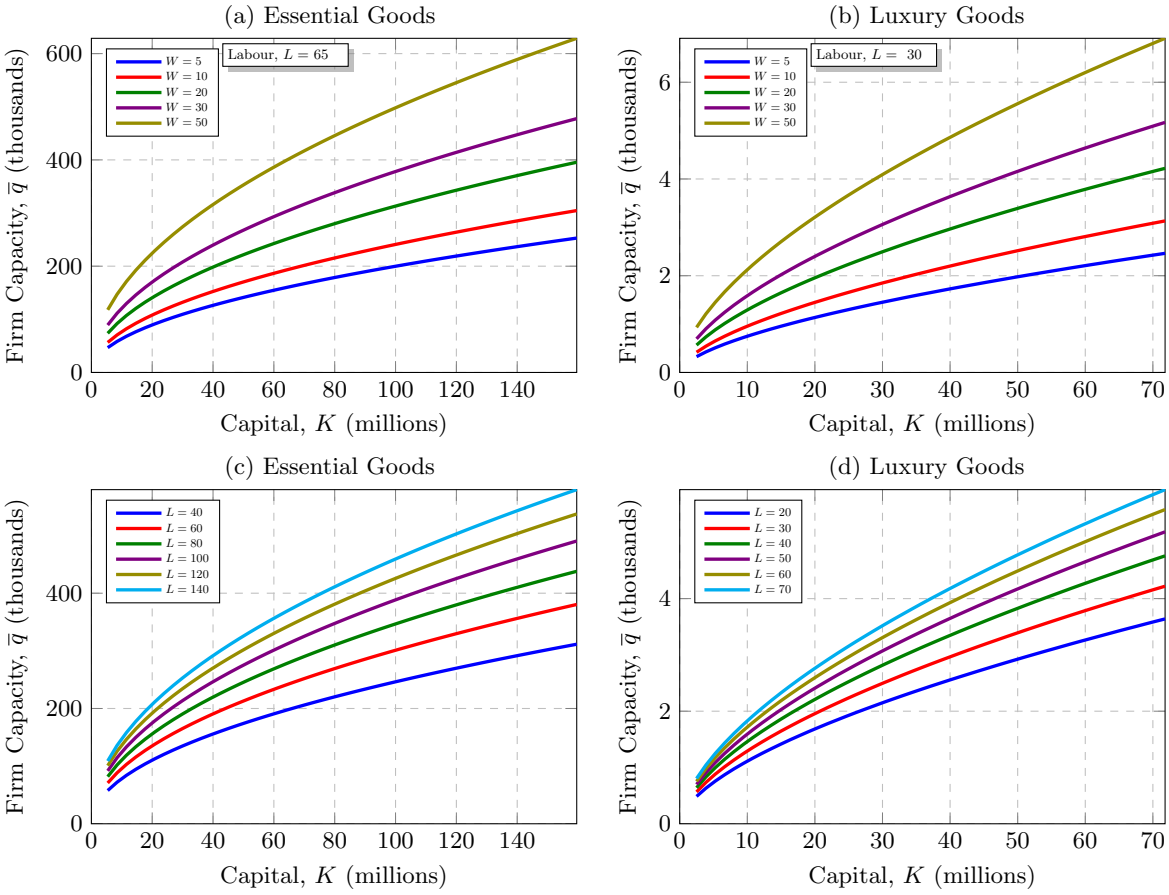


Figure 4.13: Production functions for the various sectors showing the dependence on knowledge, W, and total labour, L.

4.3.5 Firm Choices

Based on prevailing market conditions a C-firm may adjust its production capacity, $q_f^{\max}$, and its sales price of the product, p_{fg}. Various approaches to production and pricing decisions used in existing MABM models was discussed in §3.2.2, pg.29. The approach to changing a firm's production capacity, motivated by the approach of Lengnick (2013), is to adapt the factors of production, using stock as a measure of demand (Dawid & Delli Gatti 2018). These adaptations can occur at different frequencies for each firm. Pricing adjustments can be made with relative ease and so can occur fairly frequently. The ability to change production capacity would take more time, hence less frequently. Long-term decisions are made every few years and result in significant changes to the factors of production: labour, capital or knowledge. These decisions are more strategic in nature and typically would require knowledge of the prevailing market conditions and require long-term history of market performance. The subsections that follow outline how these pricing and production adaptation decisions are implemented.

4.3.5.1 Relative stock

Firms will have different production capacities and operate in different market sectors, so using current stock is problematic as an absolute measure of demand. It is convenient to define a firm's *relative stock level* as the ratio of the stock to the production capacity,

$$\delta_f(t) := \frac{\Delta_f(t)}{q_f^{\max}(t)}. \tag{4.11}$$

This measure of stock is useful within the model since it is a normalised variable, taking on a value between 0 and 1. Here $\delta_f = 0$ represents a stock-out scenario signalling high demand, while $\delta_f = 1$ represents an extreme condition of zero sales signalling low demand. The firm's *average relative stock level* is the N-point moving average of the relative stock, $\langle\delta\rangle$, defined by

$$\langle\delta\rangle_f(t) := \frac{1}{M_f}\sum_{\tau=1}^{M_f}\delta_f(t-\tau). \tag{4.12}$$

The moving average is taken over the previous M_f months (initialised as a random number $M_f \sim U[6,12]$) and is a firm-level variable that is a useful long-term measure of demand. The averaging ensures that decisions are made based on persistent market conditions, ensuring that decisions are not made based on possibly anomalous conditions for a particular month. The condition $\langle\delta\rangle_f = 0$ represents a persistent stock-out scenario over the past M_f months, signalling a high demand for the product in the market over time. The condition $\langle\delta\rangle_f = 1$ represents an extreme case of continued zero sales over the previous M_f months.

4.3.5.2 Labour adaptation

Firms would adapt their production capacity based on stock levels that serve as signals of market demand, illustrated by the Heuristic 3.1 on page 30. Periodically, firms will make production changing decisions based on the relative stock moving average, $\langle\delta\rangle$, that is the firm's measure of demand. Figure 4.14 illustrates the adaptation rule that is used for labour changes. Various stock level trigger parameters[12], namely $\hat{\vec{\boldsymbol{\delta}}}_g \equiv \left(\delta_g^{LL}, \delta_g^{L}, \delta_g^{H}, \delta_g^{HH}\right)$, are used to trigger an increase or decrease in required labour by predefined amounts $\hat{\Delta\vec{\boldsymbol{L}}}_g \equiv \left(\Delta L_g^{++}, \Delta L_g^{+}, \Delta L_g^{-}, \Delta L_g^{--}\right)$, for each goods sector g. The essence of the rule shown in Figure 4.14 is to calculate the labour change,

$$\Delta L(\langle\delta\rangle) = \begin{cases} \Delta L_g^{++} & \text{if } \langle\delta\rangle < \delta_g^{LL} \\ \Delta L_g^{+} & \text{if } \delta^{LL} \leqslant \langle\delta\rangle < \delta_g^{L} \\ -\Delta L_g^{-} & \text{if } \delta^{H} \leqslant \langle\delta\rangle < \delta_g^{HH} \\ -\Delta L_g^{--} & \text{if } \langle\delta\rangle \geqslant \delta^{HH} \end{cases} \tag{4.13}$$

so that the new labour count is $L_{\text{new}} = L + \Delta L$. High demand, $\langle\delta\rangle < \delta^{LL}$, will result in an increase in labour by an amount L_g^{++}, while low demand, $\langle\delta\rangle \geqslant \delta^{HH}$, will result in an decrease in labour by an amount L_g^{--},

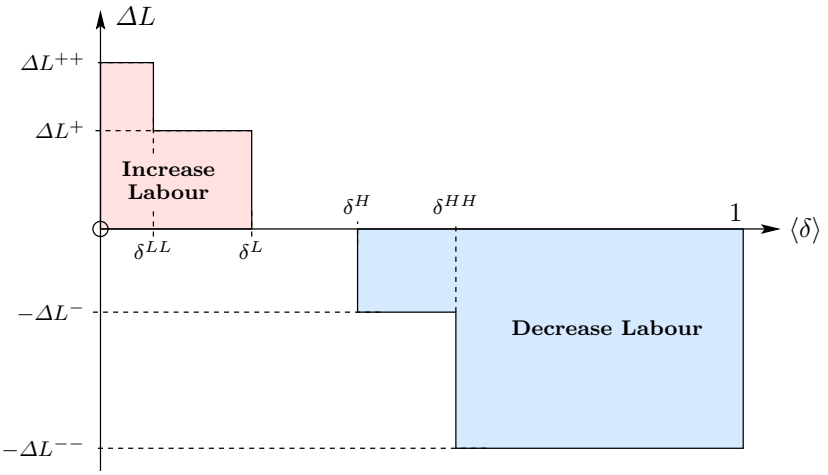


Figure 4.14: Labour change adaptation rule.

[12]Refer to Table C.3 in Appendix §C, pg.96 for the parameter values used in the simulation.

4.3.5.3 Strategic decision making

Each firm is initialised with an investment period of between 1 to 3 years in which the firm would make a strategic decision on how to invest its savings. Firms can make significant changes to their labour force, make large-scale investments in new capital or embark on knowledge development programmes. In the model, the strategic choice is made by selecting one of the three factors of production to change over the next few years. This choice is regulated by a single parameter, the *capital-labour investment probability*, $\hat{\xi}$.

In the *baseline* simulation, $\hat{\xi} = 0.4$, which means that there is an equal probability of 0.4 of either investing in capital or expanding labour; while the probability of investing in knowledge is 0.2. In other words, for the baseline simulations, 40% of the time the firm will choose to adjust labour, 40% of the time the firm will invest in capital, and 20% of the time knowledge investments will be made. For the explored simulations, $\hat{\xi} = 0.35$, means that 35% of the time the firm will choose to change its labour, 35% of the time the firm will invest in capital, and 30% of the time the firm will invest in knowledge. Adjusting this single simulation parameter results in a shift in the overall trend of how firms choose to invest for the future. Changing the simulation parameter from $\hat{\xi} = 0.4$ to $\hat{\xi} = 0.35$ is the way in which the firms in the economic model are made to shift towards a knowledge-based paradigm.

The strategic decision making rule is illustrated in Figure 4.15. Labour changes are based on the rule illustrated in Figure 4.14. Changes to capital and knowledge are based on continuous adjustment functions between minimum and maximum settings for each firm. Details on the adjustment functions used can be found in Appendix C.4.1 on page 139. When a firm decides to make large investments in either capital or knowledge, it will also require additional labour. For example, if a firm decides to increase its capital by 10%, it will also require additional, say, 5% increase in labour. All changes to the factors of production are measured in currency units, so the firm will only make the long-term investment if it has sufficient savings. Since there is no banking sector in the model, all financing of the investment is assumed to be from the firm.

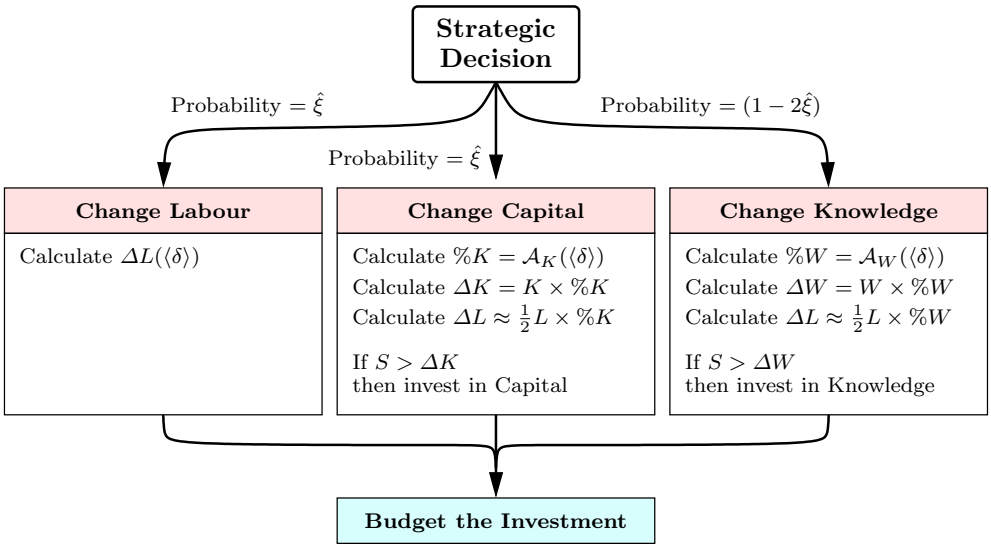


Figure 4.15: Long-term adaptation decision making process.

4.3.5.4 Budgeting the investment

When a decision is made to invest in labour, capital or knowledge, the firm determines the actual monthly amounts to spend on the investment. The outcome of the adaptation rules discussed above are desired percentage changes in either capital or knowledge together with an investment period or *term*. The next decisions to be made are the actual investment amounts, whether the firm can afford the desired investment and how to spread the investment out over time. In short, the firm needs to make *budgeting decisions*. The budgeting process described below is illustrated based on capital investments. An identical process is used for knowledge investments.

The adaptation rules will result in a desired percentage change in capital, $\%K_f$, from which the actual monetary value of the investment, ΔK_f, can be determined. If the firm has sufficient accumulated savings, S_f, that exceeds the proposed investment amount, that is $S_f > \Delta K_f$, then the firm may proceed with the investment.

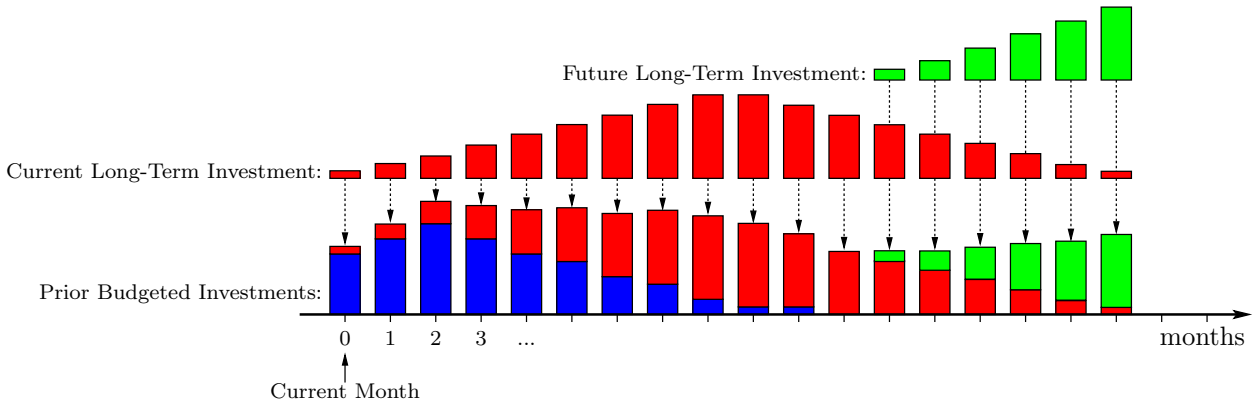


Figure 4.16: Firm budgeting process over various time periods.

The investment budget is stored as a first-in, first-out (FIFO) queue of investment amounts[13]. The budgeting over time is represented in Figure 4.16. For the long-term investments, the firm's investment amount is allocated over a period of somewhere between 18 and 48 months, and is added to the queue. The investment is budgeted following a triangular investment profile, where the investment amount is spread out in incremental amounts until half the period, and then in progressively small amounts up to the full investment period. Each month the firm will retrieve the next investment amount for the current month (the 0^{th} entry in the queue), deduct that amount from the savings, and purchase that amount from the K-firm.

[13]Implementation using a **FIFO** class described in the software documentation in Appendix §C.3, pg.101

4.3.5.5 Pricing adaptation

Firms will change their prices based on their supply capabilities, demand characteristics and their prices relative to prevailing market prices. Previously it was argued that the average relative stock level, $\langle\delta\rangle_f$, is a convenient measure of demand. If the demand is low, signalled by a high value of $\langle\delta\rangle_f$, the firm should reduce its price; while a high demand, corresponding to low values of $\langle\delta\rangle_f$, means that a firm can increase its price. A firm would also need to compare its product price, p_{fg}, to the average market price, $\bar{p}_g^{\mathrm{M}}$. It is convenient to define a *market price ratio* for the firm, Π_{fg}^{M}, as the ratio of the firms price to the average market price,

$$\Pi_{fg}^{\mathrm{M}} = \frac{p_{fg}}{\bar{p}_g^{\mathrm{M}}} \tag{4.14}$$

If the firm's price is higher than the average market price, $\Pi_{fg}^{\mathrm{M}} > 1$, then the firm should decrease its price; while if the price is lower than the average market price, $\Pi_{fg}^{\mathrm{M}} < 1$, then the firm should increase its price.

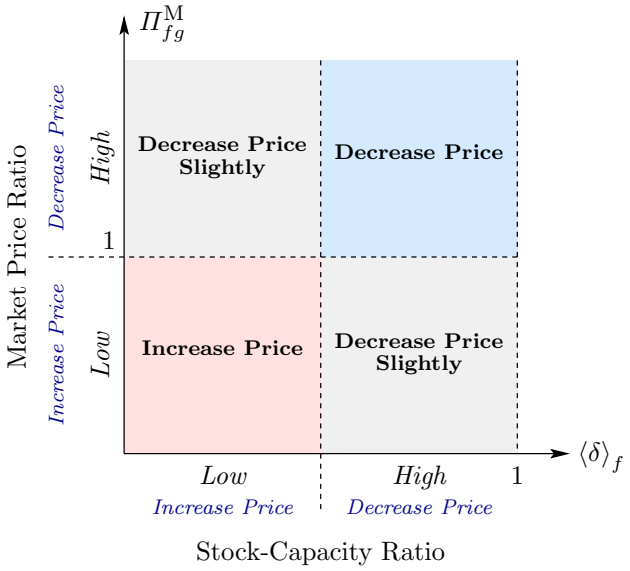


Figure 4.17: Pricing adaptation rule

The combined heuristics for the pricing decisions based on relative stock, $\langle\delta\rangle_f$, and the market price ratio, Π_{fg}^{M}, are summarised in Figure 4.17. This pricing heuristic is similar to that given in equation 3.4 on page 31 used in many MABM models. As shown in the diagram, there are two scenarios where the rules conflict: high demand (signalled by a low $\langle\delta\rangle_f$) with a high price (with $\Pi_{fg}^{\mathrm{M}} > 1$) and low demand (signalled by a high $\langle\delta\rangle_f$) with low price ($\Pi_{fg}^{\mathrm{M}} < 1$).

- A low $\langle\delta\rangle_f$ with high Π_{fg}^{M} scenario is favourable for the firm, meaning that despite having higher than average prices, the firm is still able to sell its product. In this case, the firm should leave the price unchanged, or reduce the price by a very small amount in order to remain competitive.

- The high $\langle\delta\rangle_f$ with low Π_{fg}^{M} case is an ambiguous scenario for the firm. Despite having a low price, the firm is still incapable of selling all its products. In this case, the firm would only slightly decrease its price in an attempt to sell more products in the market.

The heuristic outlined above was implemented by means of a *price adjustment surface* for each product, shown in Figure 4.18. Each surface provides a continuous function, $\Delta p_{fg}\left(\langle\delta\rangle_f, \Pi^{\mathrm{M}}_{fg}\right)$, giving the price adjustment percentage. The function for the surface was constructed by combining two sigmoid functions.

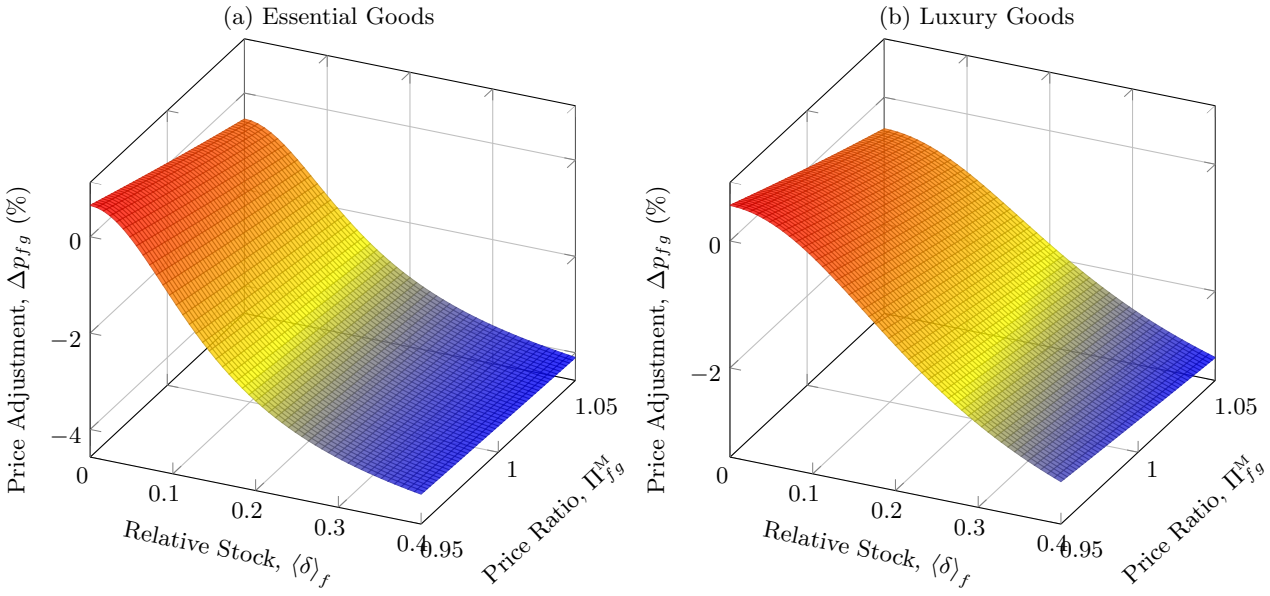


Figure 4.18: Firm pricing adaptation surfaces.

A continuous version of the heuristic pricing adaptation rule shown in Figure 4.17 is obtained by projecting the pricing adaptation surfaces onto the plane. The resulting projections are shown in Figure 4.19, together with a zero price adjustment contour separating the regions of price increase and price decrease.

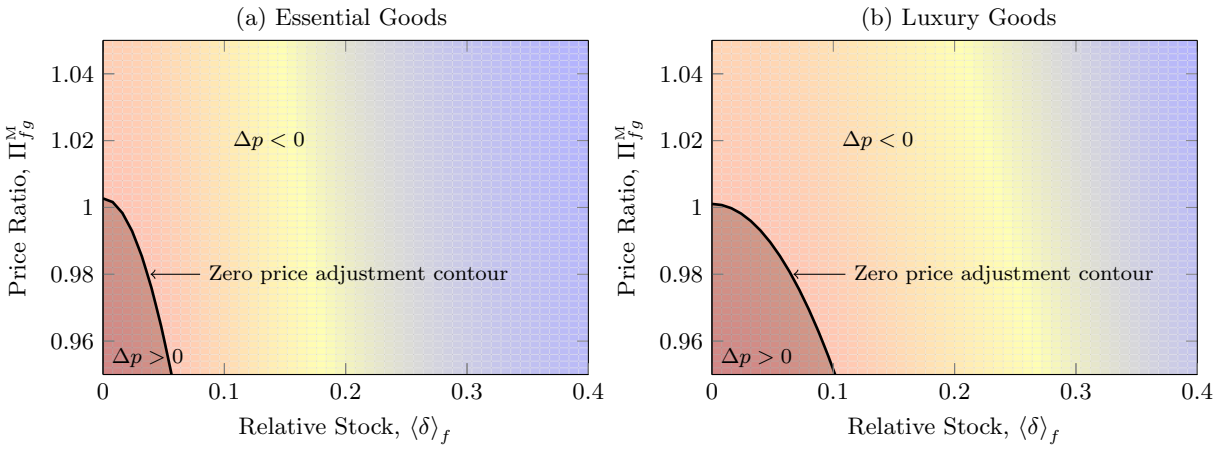


Figure 4.19: Projection of firm pricing adaptation surfaces showing the zero price adjustment contour. There are two regions: a shaded price increasing region, $\Delta p > 0$, and a price decreasing region, $\Delta p < 0$.

4.4 Markets

In the model, markets are the mediating mechanism by which the various economic agents interact with each other. There are three types of markets used in the model shown in Figure 4.1 on page 33: a product market, a labour market and a foreign market. An abstract **Market** class was implemented as a base class for the three market types as shown in the class hierarchy in Figure 4.20. The **Market** class provides the generic properties of a market, which consists of a collection of buyer agents who will attempt to purchase an assortment of products; and seller agents that are each able to supply a particular product.

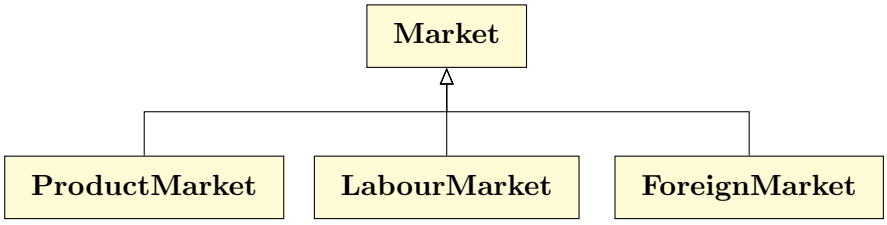


Figure 4.20: The class hierarchy for the various markets.

For example, individual agents will purchase goods and services from firms via the product_market object; country agents will purchase goods and services from firms via the foreign_market object; while firms will purchase or employ labour via the labour_market object. In all of these cases, a market is composed of a number of buyers and sellers, each being different groups of agents in the economy. For the markets considered in the model, Table 4.1 summarises the various subsets of agents and their roles in the respective markets.

Table 4.1: Summary of the features of each of the markets used in the model.

Class	Market Object	Buyer Agents	Seller Agents	Products
ProductMarket	product_market	Consumer Individuals	C-Firms	Goods
ForeignMarket	foreign_market	Countries	C-Firms	Goods
LabourMarket	labour_market	All Firms	Unemployed Individuals	Education

The process by which a buyer purchases desired products from the available seller agents is implemented in a market•shopping() algorithm, explained in detail within the software documentation in Appendix C. Figure 4.21 provides a schematic representation of the primary interfaces between a buyer agent and a seller agent required by the algorithm.

We introduce the concept of a *basket* analogous to the 'consumption bundle' used by Ashraf et al. (2017). Each buyer agent will have a basket of 'products' that they already possess and a basket of 'products' that they need. A buyer agent will generate a desired buyer•needs basket of 'products', when instructed by the buyer•calculate_needs() method. The preferences, consumption patterns or consumer behaviour of the buyer agents are encoded within this method. In essence the needs basket establishes product demand. This demand is determined in different ways for each type of buyer in the different markets.

Product market: For individual agents in the product_market, the quantities of goods in the individual•needs basket is calculated based on preset *status consumption parameters* and

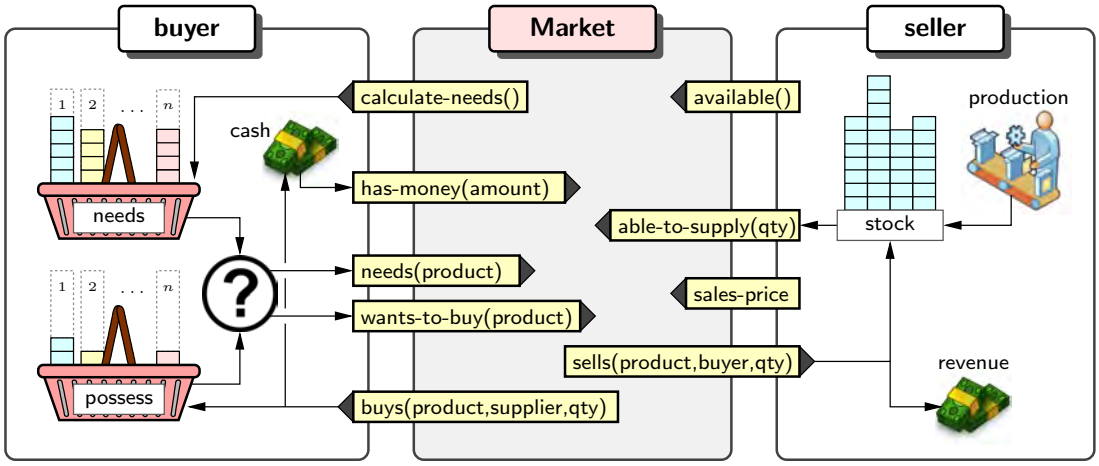


Figure 4.21: Schematic diagram showing the interfaces between buyer and seller agents required for the **Market** class.

cash available. As a buyer in the product market, each individual will require different quantities of products taken from the *Goods* set. The *status consumption parameter* is the default, base basket of goods for the various categories of individual agents based on status and education (see Table C.2 on page 100 for the default parameter values). For example, the status consumption parameter for an unskilled individual is given by [1000, 5]. This status consumption parameter is then scaled based on disposable cash available to the individual and by a random variation. So a particular unskilled individuals needs basket could then be equal to [2478 12], that describes the individuals need for 2478 *Essential* items and 12 *Luxury* items.

Labour market: For firm agents in the labour_market, the quantities of labour at each education level in the needs basket is determined by the labour composition vector, calculated based on the firm's technological sophistication (see §4.3.3, pg.42). As a buyer in the labour market, a firm will require a certain number of employees at different skill levels as the 'products' taken from the Education set in the needs basket. When shopping in the market, a buyer will add purchased 'products' to its possess basket with the aim of attaining a situation where the possess basket equals the needs basket.

For each 'product', which may be either a goods or an education level, depending on the market, only a subset of suppliers will be able to supply that product. Only *Essential* C-firms will be able to supply *Essential* goods; while only those individuals with a degree will be able to supply a *Bachelors* level of education as a potential employee. Furthermore, a typical customer for a product would not go to every supplier of that product, but will choose to purchase from a small group of potential suppliers. We have defined a souk as a marketplace comprising of a small subset of the seller agents from which a buyer will attempt to purchase a particular product from its needs basket. A buyer will take a number of trips to a souk in an attempt to satisfy all its needs.

4.5 Government

The government agent in the economy serves two simple roles. Firstly, it provides monthly social grants to those individuals that are unemployed. It is assumed that a fixed social grant amount of R4,000 is paid to unemployed individuals. The social grant amount was chosen to be about a third of the salary of the lowest paid worker.

Secondly, government obtains revenue only through taxation of firms. The corporate tax rate, r_{G}, is adjusted annually at the beginning of each year in order to ensure that it has sufficient funds to pay the social grants. Changes to the tax rate are based on the total grant payments, $P^{\mathrm{G}}_{\mathrm{tot}}(t)$, and the total net profit of firms, $N_{\mathrm{tot}}(t)$. Computationally, the change in tax rate can be calculated using (see Appendix §C.4.2, pg.139 for the derivation)

$$\Delta r_{\mathrm{G}} \approx \frac{1}{N_{\mathrm{tot}}} \frac{dP^{\mathrm{G}}_{\mathrm{tot}}}{dt} - \frac{r_{\mathrm{G}}}{N_{\mathrm{tot}}} \frac{dN_{\mathrm{tot}}}{dt} \tag{4.15}$$

where the derivatives are performed using a standard numerical 3-point backward-difference formula. All that is required is to store the previous 3-year's values for the net profit and total grant payments in order to estimate the derivatives. In effect equation (4.15) uses historical data to forecast changes in the total grant payments, $P^{\mathrm{G}}_{\mathrm{tot}}(t)$, as well as the changes in the total net profit of firms $N_{\mathrm{tot}}(t)$, and use this information to adjust the taxation rate.

4.6 Timing & scheduling

Each time step in the simulation is taken to be a calendar month. During each month there are several events that take place, some at the beginning of the month, some during the month, and some at the end of the month. At the beginning of each year, the required changes to the population is calculated in order to satisfy the population growth requirements. This results in a list of people who are randomly chosen to die every month and an estimate of the number of new births each month (see §4.2.1, pg.36 for details). In addition, government will recalculate the corporate tax rate to ensure that it has sufficient funds to pay the social grants.

4.6.1 Beginning of the month

At the beginning of every month, the following events take place.

1. **Population Adjustment:** The population of individuals is adjusted on a monthly basis. This method will perform the following tasks:
   (a) If it is the individual's birthday month, the individual's age is increased by one. Having aged a year, the status of an individual may also change and would need to be updated. If the individual is at school or studying, their status may change based on the probabilities of progressing (see §4.2.2, pg.37.) A test is performed to check if the individual has reached retirement age, and if employed, will be removed from the employer's list of employees.
   (b) New individuals are 'born', based on the estimated number of new births calculated at the beginning of the year.
   (c) Individuals that were chosen to die at the beginning of the year are then killed off. When killing an individual, if the individual is employed, they are then removed from the employer's list of employees. The accumulated wealth of the individual is redistributed as an inheritance to a small, randomly chosen group of individuals.
2. **Government update:** As a result of the population readjustment and possible changes to status of individuals, the government is updated with the list of unemployed individuals that require social grant payments.
3. **Payments to individuals:** Individuals would receive income from a number of possible sources. Firms would pay salaries to all the employees. All unemployed individuals would receive a social grant from government. All retired individuals would transfer a small amount from their savings to their available cash.
4. **Labour market:** There are a number of tasks that take place in the labour market:
   (a) Since the population status may have changed, for example through new entrants from the schooling or tertiary education system, deaths, etc., the list of sellers in the labour market is updated.

(b) For similar reasons, the demand and capacity of the labour market is recalculated, using the methods discussed in the generic market class (see §4.4, pg.53).

5. **Product and foreign markets:**

   (a) The product market is updated with the collection of consumer agents, required after the population adjustment.

   (b) Demand is calculated in each of the product and foreign markets based on the needs of individuals and countries.

   (c) Each consumer firm in the product market will produce its capacity of products and place them into stock available for purchase during the month. The firm's stock and capacity attributes are updated based on its particular production function.

4.6.2 During the month

During each month, goods are purchased in the two markets based on the shopping algorithm outlined in Section 4.4 on page 53.

1. **Product market shopping:** The population would go shopping in the product market, based on the shopping algorithm with consumers as the buyers and firms as the sellers.

2. **Foreign market shopping:** The countries making up the rest of the world would go shopping in the product market, with the countries as the buyers and firms as the sellers.

4.6.3 End of the month

At the end of each month, there are several tasks that are performed:

1. **Firms adjust capacity & pricing:** Each consumer firm would potentially adjust its production capacity and product pricing. This may result in changes to labour requirements, capital investment or knowledge investment. (see §4.3.5, pg.47 for details.) C-firms spend their monthly budgeted investment amount as income to the K-firm. Capital depreciation is calculated for each of the consumer firms using a fixed depreciation rate, $\hat{d}$ =0.01 % per month.

2. **Labour market shopping:** At the end of the month, all firms having reassessed their labour requirements, would go shopping in the labour market for new employees. The population tree is updated in order to re-categorise individuals.

3. **Individuals & firms save:** A proportion of remaining cash that individuals and firms have at the end of the month is transferred to savings.

CHAPTER 5

The simulation results

This chapter provides the results of the various simulations[1]. Simulations were run over a time of 50 years (or 600 months), with $\hat{N}_{\mathrm{MC}} = 20$ Monte Carlo runs. The underlying output variables from the model are obtained by averaging over all the Monte Carlo runs thereby eliminating stochastic fluctuations arising from each simulation. The Monte Carlo runs also provided uncertainty bounds on the variables. The uncertainty bounds were calculated as a standard error ($e = \sigma/\sqrt{N_{\mathrm{MC}}}$), shown with some of the results as applicable. A burn-in of 3 years was used before data was recorded to ignore any initial transients in the simulation.

The simulations were initialised with a seed population of $\hat{N}_{\mathrm{P}} = 2{,}500$ individuals, an initial unemployment rate of $\hat{r}_{\mathrm{U0}} = 25.0\%$, and a population growth rate of $\hat{g}_{\mathrm{P}} = 1.5\%$ per annum. Only one K-firm was used, 15 firms were created in the essential goods sector, and 10 firms in the luxury goods sector.

Two simulation scenarios were considered: a baseline simulation and an 'explored' simulation with higher investments in knowledge and technology. The baseline simulation with the *capital-labour investment probability* parameter $\hat{\xi} = 0.4$, provides the benchmark results which can be used to compare against. Results from the explored simulation with the capital-labour investment probability parameter adjusted to 0.35 is then compared to the baseline results. A model validation check is performed by fitting the simulation results to the Solow-Swan growth model.

[1]Details of all the simulation parameters is provided in Appendix C.2.

5.1 Baseline simulation

5.1.1 Population results

5.1.1.1 Population growth

Figure 5.1 shows the exponential growth rate of the population, starting from an average initial population of 2616, growing at an average growth rate of $\tilde{g}_{\mathrm{P}} = 1.47\%$ per annum (aligned with the simulation parameter for growth of $\hat{g}_{\mathrm{P}} = 1.5\%$ per annum.) The initial population count is slightly higher than the seed population parameter, $\hat{N}_{\mathrm{P}} = 2{,}500$, due to the initial burn-in. The counts shown are the averages over all the Monte Carlo simulation runs for the baseline scenario. Also shown is the population of active and employed individuals, together with their growth rates. The growth in employed individuals is roughly equal to the population growth rate, indicating long-term stability in unemployment.

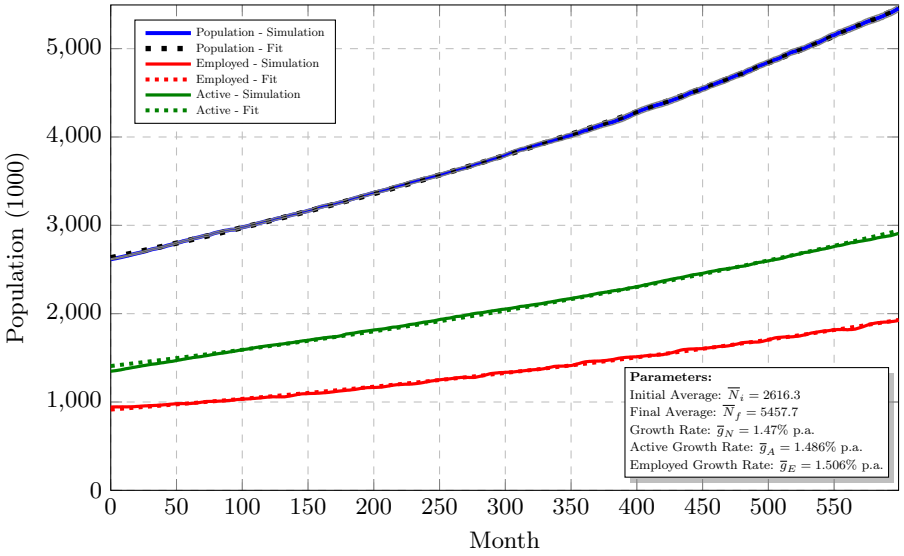


Figure 5.1: Time series of population growth and total number of employed individuals.

5.1.1.2 Population distributions

The initial and final population age distributions are shown in Figure 5.2. The simulation has approximately maintained the half-normal population distribution throughout the simulation runs.

5.1.1.3 Population education & status

The education and status proportions within the population are largely unchanged over time since the educational throughput probability parameters (discussed in §4.2.2, pg.37) are fixed. Figure 5.3 shows the changes in the average of all individual status variables over time. The long-term average proportions of the status and education variables for the population are shown in the pie-charts in Figure 5.4. The proportion of unskilled labour remains approximately constant at an average of 55.26% of the active population.

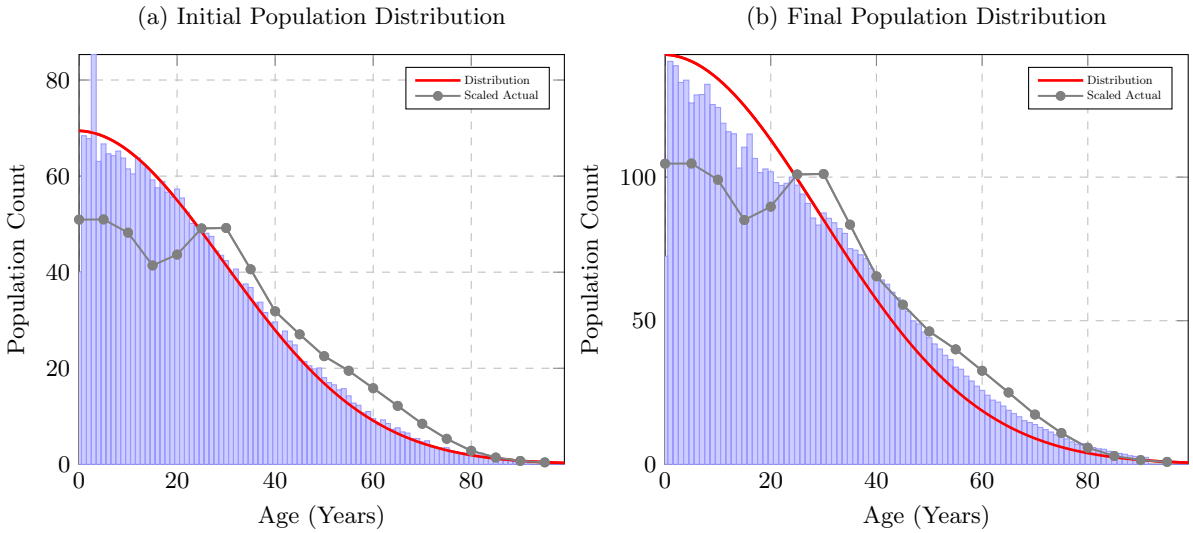


Figure 5.2: Initial and final age distribution of the population from the baseline simulation. The scaled actual data points are based on South African population data (Statistics South Africa 2019*b*, Table 11).

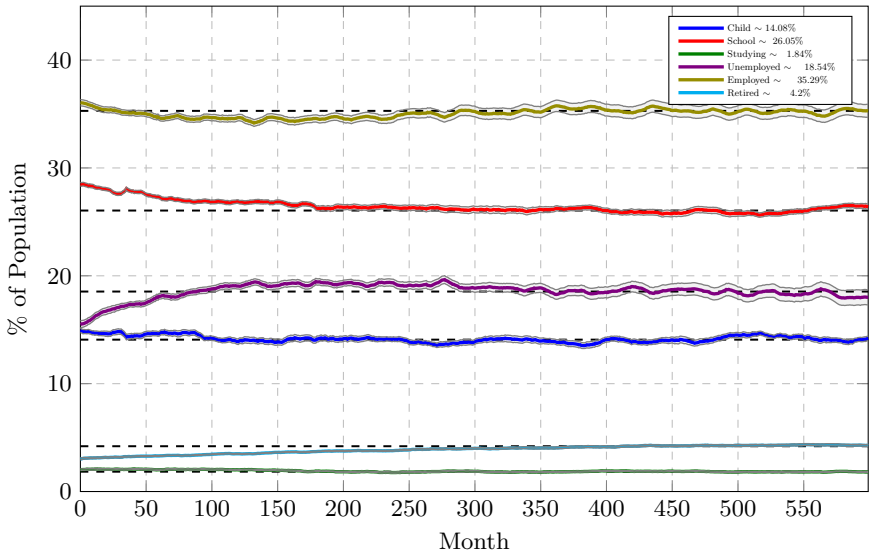


Figure 5.3: Time series of the population status given as a percentage of the total population.

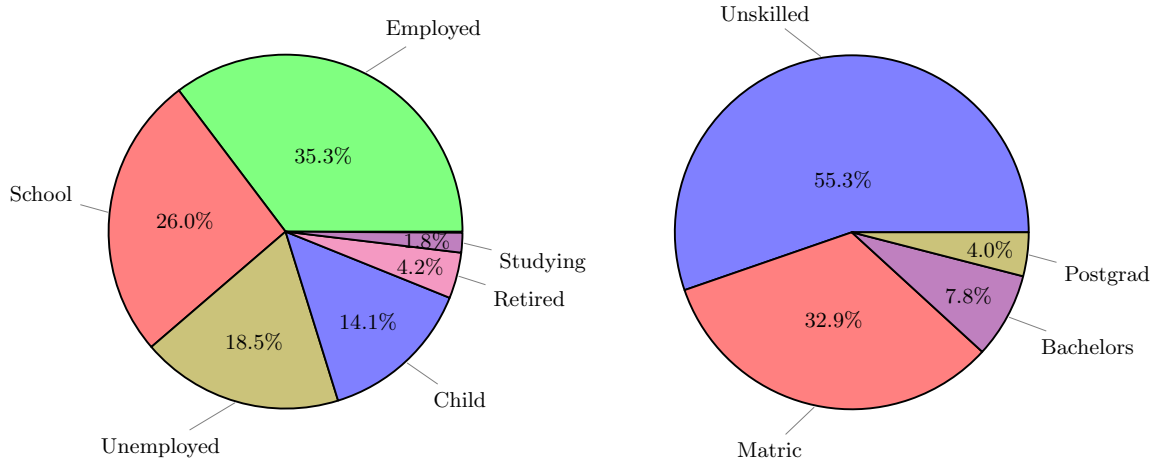


Figure 5.4: Pie charts showing the long-term average population breakdown by status and education level.

5.1.2 Firms

5.1.2.1 Labour composition

The labour composition for all firms is shown in Figure 5.5. We note that unskilled and semi-skilled (matric) labour forms the largest portion of the workforce. For comparison, the number of employed individuals in the population is also shown. For the lower-skilled portions of the population, the number of unemployed individuals is increasing, possibly indicating a higher unemployment rate. By contrast, the more highly skilled sectors of the population show a decrease in the number of unemployed individuals.

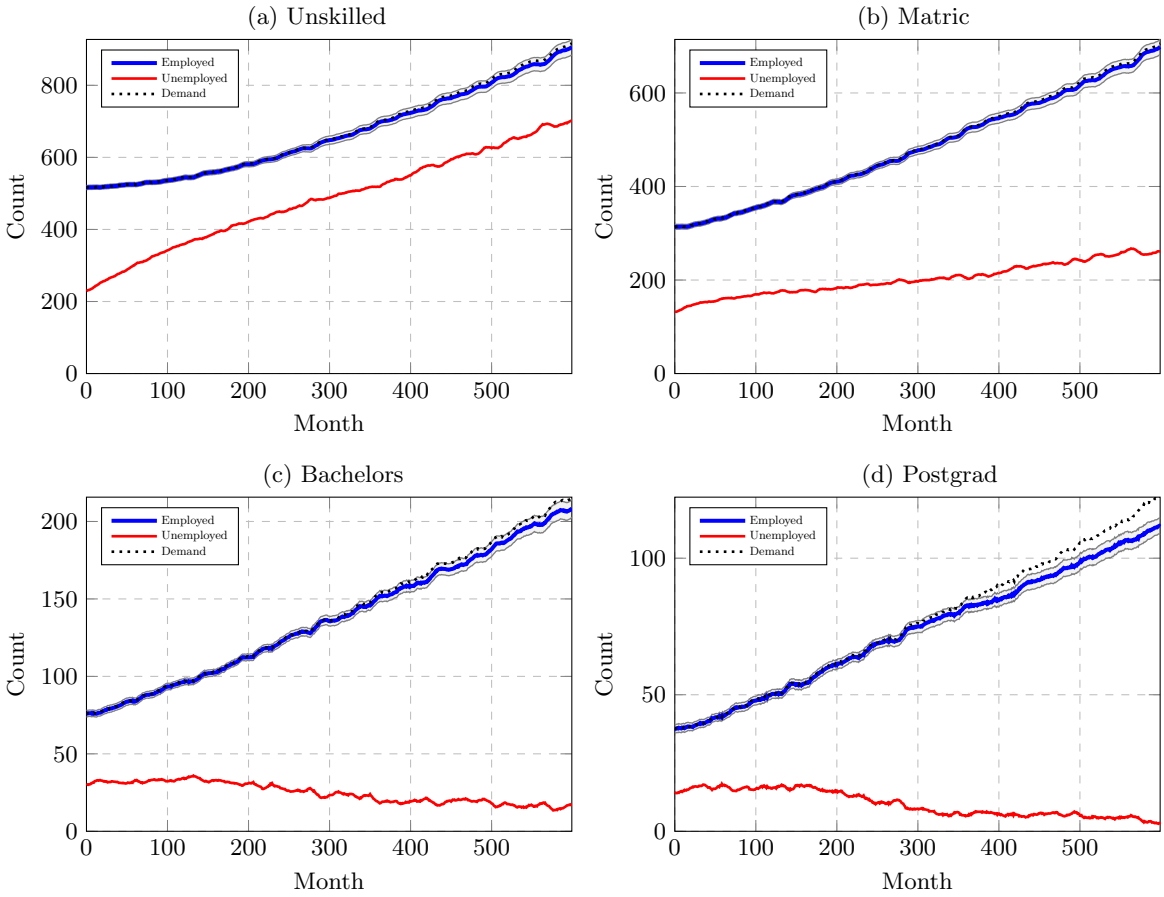


Figure 5.5: Time series of firm labour by education level from the baseline simulation.

5.1.2.2 Factors of production

The evolution of the other factors of production are shown in Figure 5.6, giving the Monte Carlo average over all firms of the capital, K, and knowledge, W, for each of the goods sectors. Since the capital-labour investment probability parameter $\hat{\xi} = 0.4$, in the baseline simulations, more emphasis is placed on investment in capital than in knowledge. Hence the capital growth rate is higher than for knowledge. For comparison, the growth in technological sophistication, τ, is also shown.

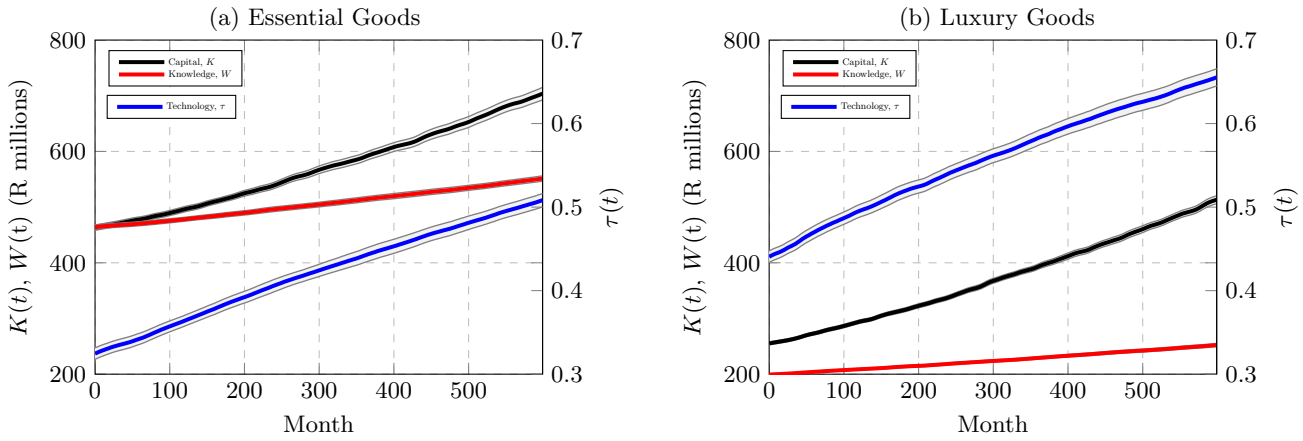


Figure 5.6: Time series of firm factors of production from the baseline simulation.

The increase in capital and knowledge over time is the result of firm investments, shown in Figure 5.7. As expected, the investment in capital for all firms is considerably higher than the investment in knowledge, due to the capital-labour investment probability parameter prioritizing capital investment over knowledge investment by firms.

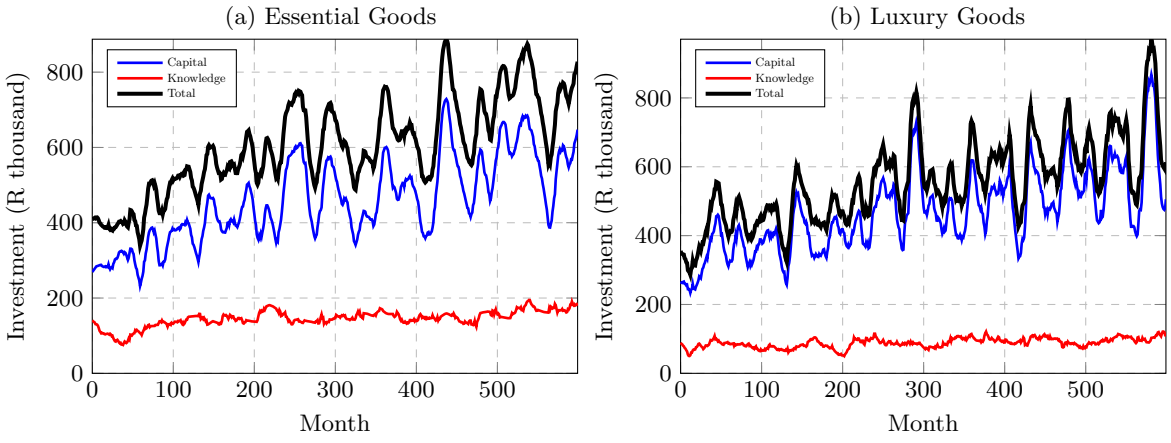


Figure 5.7: Time series of firm investment in capital and knowledge from the baseline simulation.

5.1.2.3 Firm output

Figure 5.8 shows the total volumes for the firms in each product sector, including total production capacity, actual firm output, consumption volume, exports volume, and total stock levels.

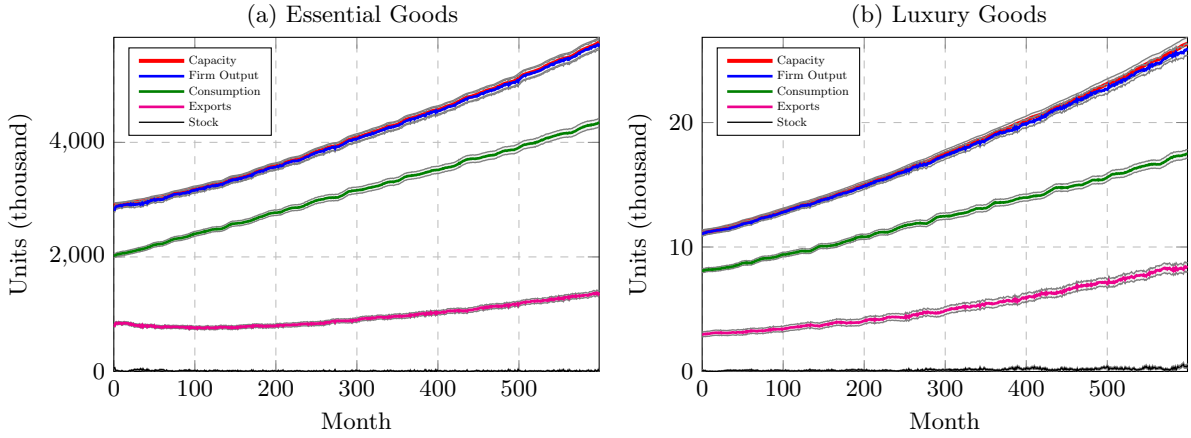


Figure 5.8: Time series of total firm volumes from the baseline simulation.

The average growth rates for production capacity, consumption and exports for the two sectors are summarised in Table 5.1. The growth in production capacity for the essential goods sector is slightly lower than the average population growth rate of 1.47% p.a. The slightly higher production capacity growth in the luxury goods sector is possibly due to higher exports. The low growth rates in consumption for both sectors are due to high unemployment (see §5.1.5.2, pg.68) and the resulting lower disposable income for individuals. The simulations have highlighted an important consequence of increased unemployment in an economy: increasing unemployment leads to lower consumption and hence lower production by firms.

Table 5.1: Summary of production/output growth rates for firms in the baseline simulation.

	Sector	
Description	**Essential**	**Luxury**
Average growth rate for production capacity	1.422% p.a.	1.738% p.a.
Average growth rate for consumption	1.461% p.a.	1.524% p.a.
Average growth rate for exports	1.315% p.a.	2.219% p.a.

5.1.3 Product market

The change in prices of the goods in product market is shown in Figure 5.9, together with the demand (consumer and export) and firm total supply capacities.

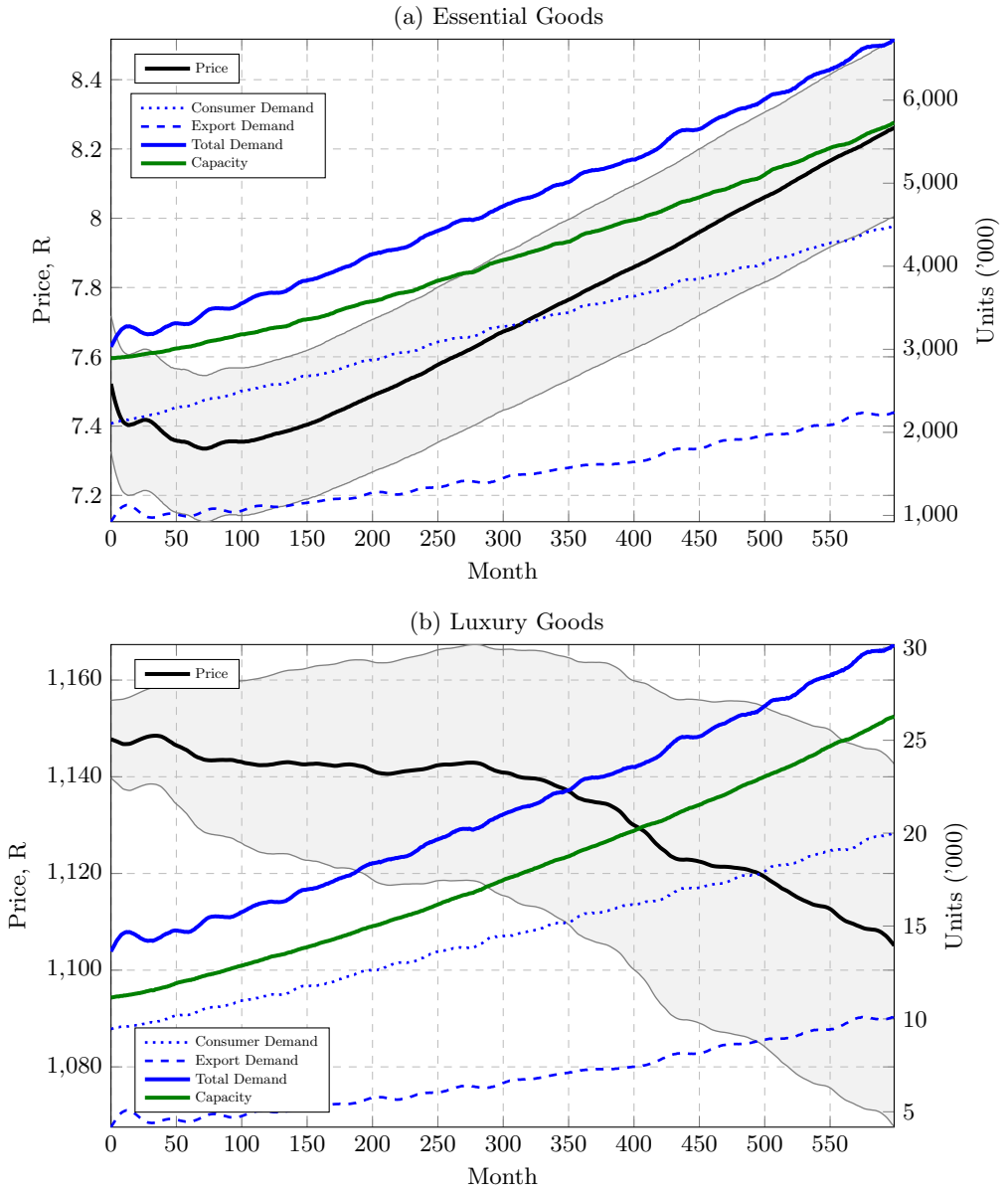


Figure 5.9: Product market prices.

The change in pricing for two goods sectors seems to indicate different demand and supply behaviour. Yet, in both sectors the aggregate demand is higher than the firm supply capacities, which should result in price increases for both sectors. It was necessary to investigate this apparent contradiction. An advantage of agent-based modelling is that it is possible to investigate 'under the hood' of the simulation and observe all the variables of all agents. The application of the firm pricing adaptation rules (presented in Section 4.3.5.5 on page 51) needs to be investigated in order to understand the reasons for the general decrease in the market price for the luxury goods sector.

Figure 5.10 shows the average relative stock level, $\langle \delta \rangle_f$, and the market price ratio , Π^{M}_{fg}, data points for the pricing factors of all the Monte Carlo runs in the baseline simulation. The average

relative stock level is what each firm uses as its measure of market demand. The red contour represents the zero-price adjustment contour from the surfaces shown in Figure 4.18 (on page 52), with the shaded red region below the curve indicating where prices would increase. The data points for the firms were allocated to various clusters using a Gaussian Mixture Model (GMM) machine learning clustering algorithm (Géron 2019).

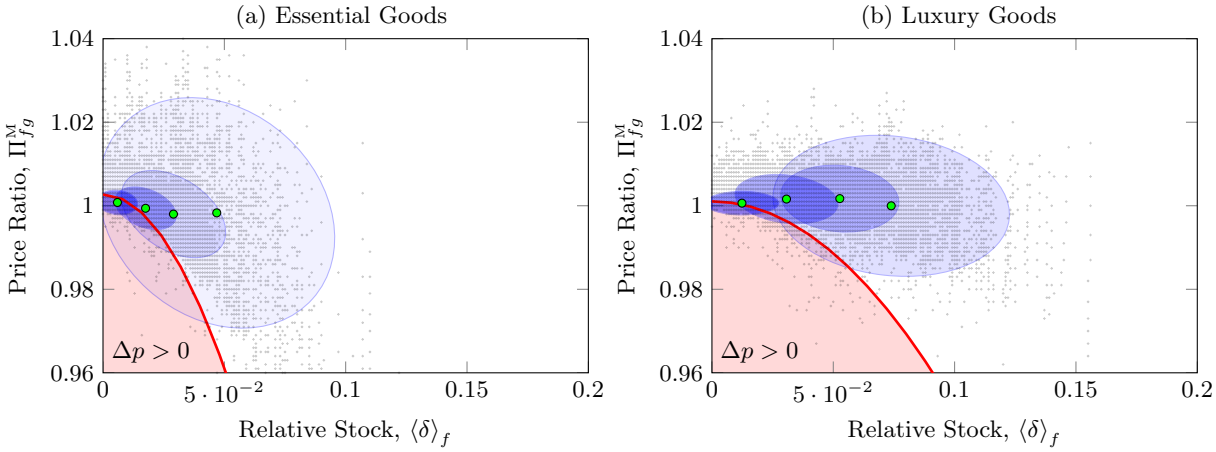


Figure 5.10: Firm pricing adaptation from all baseline simulation data. Thick red line shows the zero-price adjustment contour, with the red shaded region showing where pricing increases. Pricing adaptation clusters are shown as blue shaded regions, with cluster centers in green.

The pricing adaptation clusters are represented by the blue-shaded regions in Figure 5.10, together with the cluster centers. The general shape and positioning of the pricing adaptation clusters are different for the two sectors. The essential goods sector has pricing clusters that tend to be closer to the pricing-increase region than for the luxury pricing clusters. The most densely populated clusters (indicated with the blue shading intensity) represent the predominant pricing adaptation factors. For the essential goods sector, the dominant cluster is centered slightly inside the pricing increase region, while for the luxury goods sector, the dominant cluster is centered outside the pricing increase region. Stated differently, based on the positioning of the dominant cluster, the firms in the essential goods sector experience relatively high demand, which will tend to drive the price up, as shown in shown in Figure 5.11(a). Firms in the luxury goods sector experience relatively low demand, driving the prices down as shown in shown Figure 5.11(b).

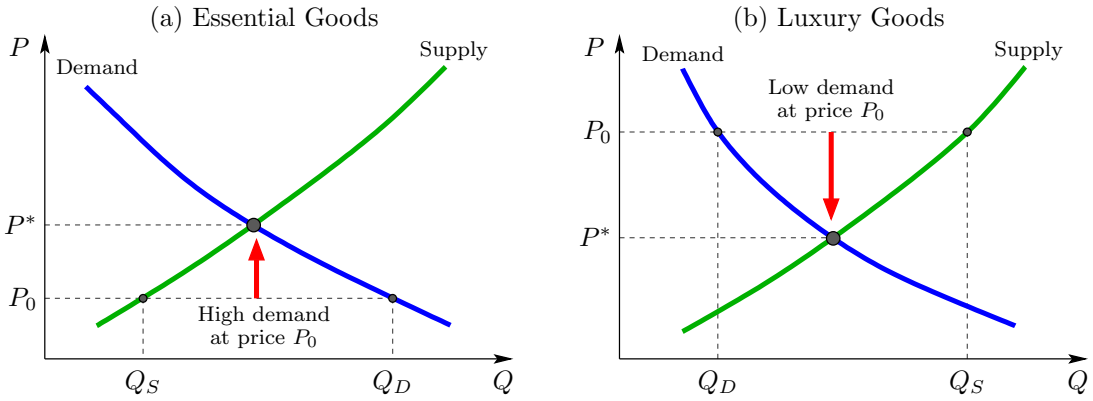


Figure 5.11: Supply and demand curves used to explain the pricing behaviour in the two goods markets.

5.1.4 Government

Government pays a fixed social grant amount of R4,000 to unemployed individuals. The government payments of social grants, shown in Figure 5.12, is determined by the total number of unemployed individuals in the population. As expected, Figure 5.12 shows that the total social grants is closely correlated with the unemployment count.

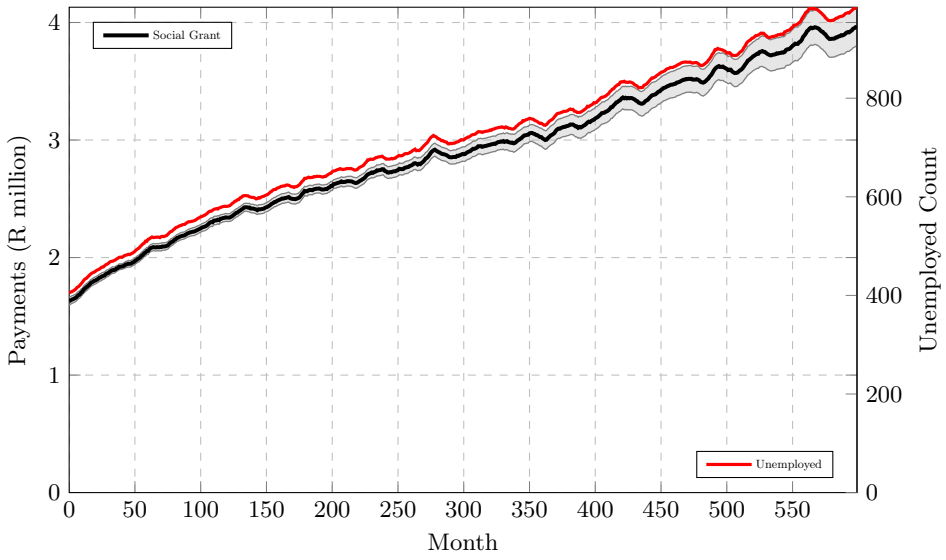


Figure 5.12: Total government payment of social grants to the unemployed. For comparison, the total count of unemployed individuals is also shown.

The simulation was initialised with an initial corporate tax rate of 25.0%. The trend in the corporate tax rate, $r_{\mathrm{G}}(t)$, shown in Figure 5.13, loosely follows the unemployment rate. This is to be expected due to the assumption that all government income is to be allocated to the payment of social grants with the tax rate adjustment performed in such a way as to compensate for forecast changes in the social grant payment (see §4.5, pg.55).

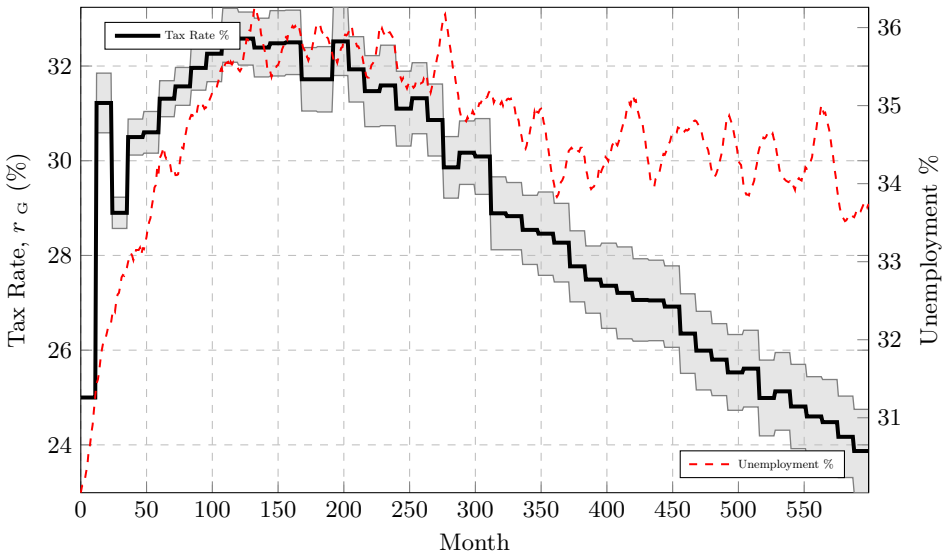


Figure 5.13: The company tax rate, $r_{\mathrm{G}}(t)$. For comparison, the unemployment rate is also shown.

5.1.5 Macroeconomic results

5.1.5.1 GDP

Figure 5.14 shows the baseline simulation results for GDP, Y, together with its components, consumption, C, investment, I and exports, X. The initial baseline average GDP for the first year was $\tilde{Y}_i$ = R34.72 million; while at the end of the simulation runs, the final baseline average GDP was $\tilde{Y}_f$ = R76.9 million. Over the period, the average annual GDP growth rate was $\tilde{g}_Y$ = 1.67% per annum; while population average growth rate was found to $\tilde{g}_N$ = 1.47% per annum. GDP growth in the model seems to be influenced by the population growth rate. The growth rates for the other components of GDP are also shown. There is a relatively low growth in investment, $\tilde{g}_I$ = 1.11% per annum.

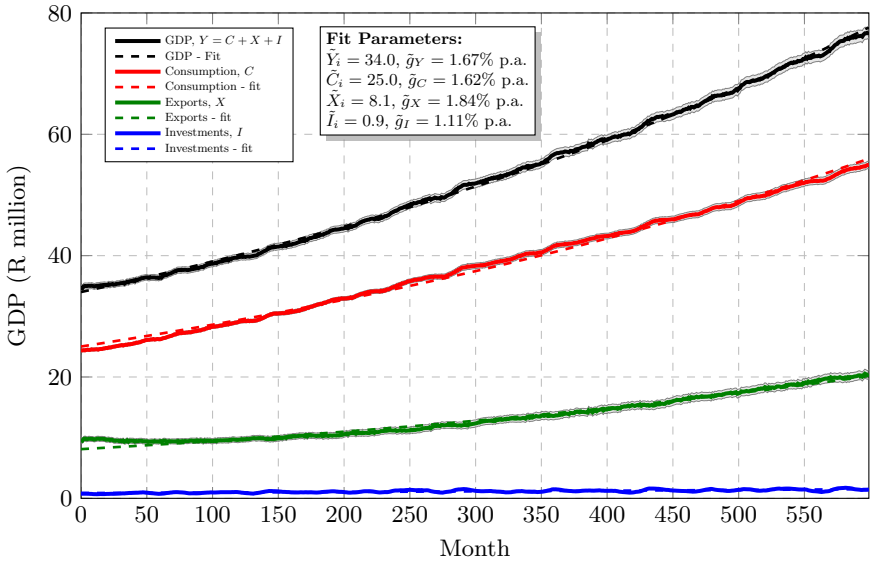


Figure 5.14: Time series of GDP from the baseline simulation. Dashed lines indicate fitted GDP growth rate.

5.1.5.2 Unemployment

The unemployment rate over time is shown in Figure 5.15, together with the unemployment rates for the various skill levels in the population. The unemployment rate remains fairly constant with a long-term average of 34.44%.

The trends show lower levels of unemployment for higher levels of skills. Even though the labour composition within firms is dominated by unskilled workers (as was shown in Figure 5.5), unemployment is still highest for the unskilled sector of the population. Unemployment rates for the more highly skilled sectors of the population decreases over time.

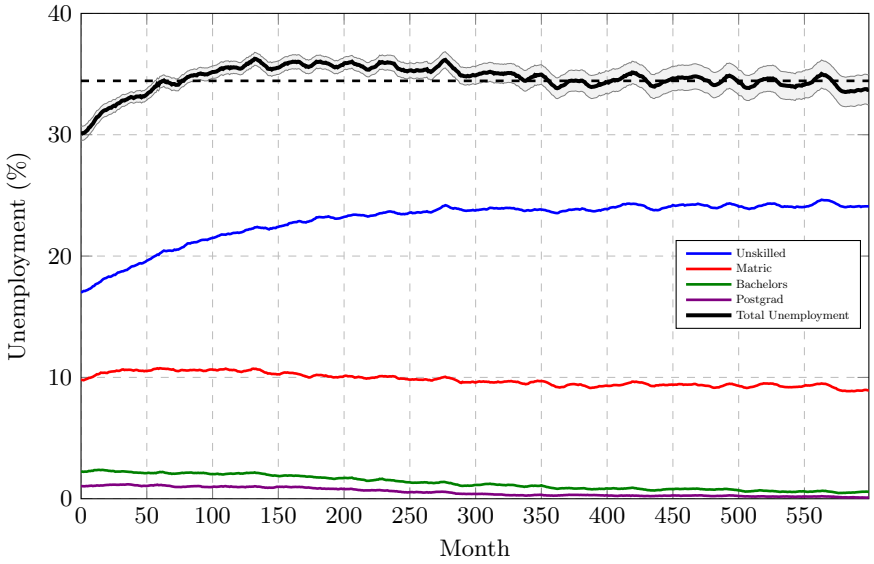


Figure 5.15: Time series of unemployment from the baseline simulation.

5.2 Explored simulation results

5.2.1 Population

The initial and final population distributions for the two simulation scenarios are shown in Figure 5.16, showing no major differences between the scenarios. All aspects of the population remain largely unaffected by the change in the scenarios.

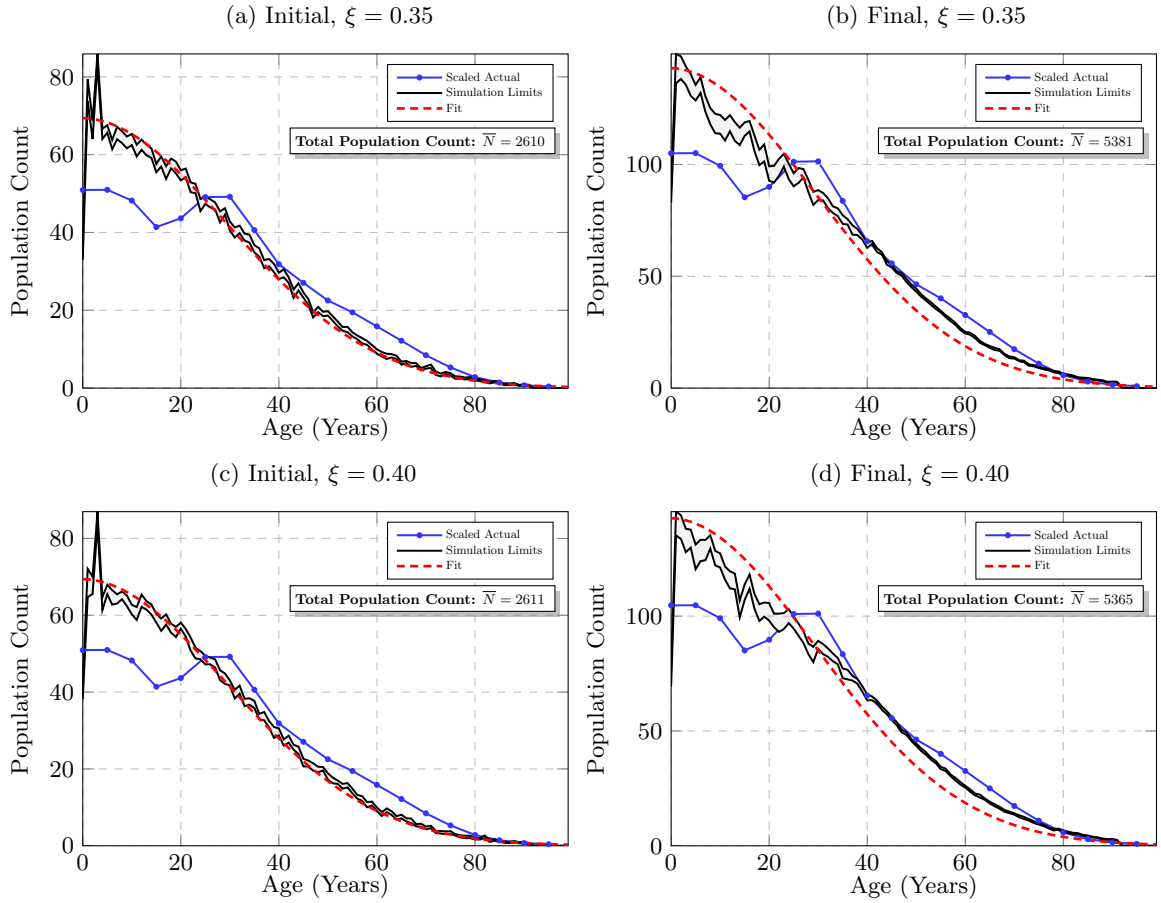


Figure 5.16: Initial and final age distribution of the population for the explored parameter values. The scaled actual data points are based on South African population data (Statistics South Africa 2019*b*, Table 11).

5.2.2 Firms

5.2.2.1 Labour composition

A comparison of the total labour counts in Figure 5.17, shows that there is a lower total labour force across both consumer goods sectors in the explored scenario.

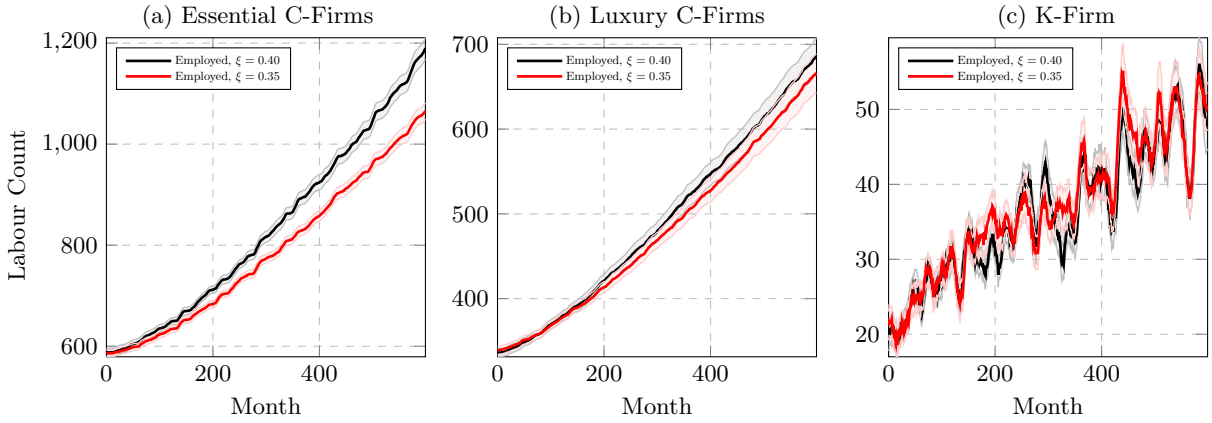


Figure 5.17: A comparison of the firm labour for all sectors from the simulation scenarios. Shaded regions indicate uncertainty bounds.

Plots of the labour composition (as a percentage of the total labour force) of all firms between the two scenarios are shown in Figure 5.18. In both scenarios, the labour composition of unskilled workers steadily declines over time, semi-skilled (matric) worker composition remains fairly constant in the long-term; while the composition of skilled (both bachelors and postgraduate) workers increases over time. As could be expected, the unskilled labour composition is lower in the explored scenario since fewer unskilled workers would be required in a more technologically sophisticated economy. The significant deviation between the two scenarios appears in the composition of the more highly skilled workers (skilled/bachelors and highly-skilled/postgrad). The explored scenario shows a significant increase in the labour composition of the more highly skilled workers. These results demonstrate that as firms strategically shift towards higher levels of technological sophistication over time, their workforce would need to be more highly skilled.

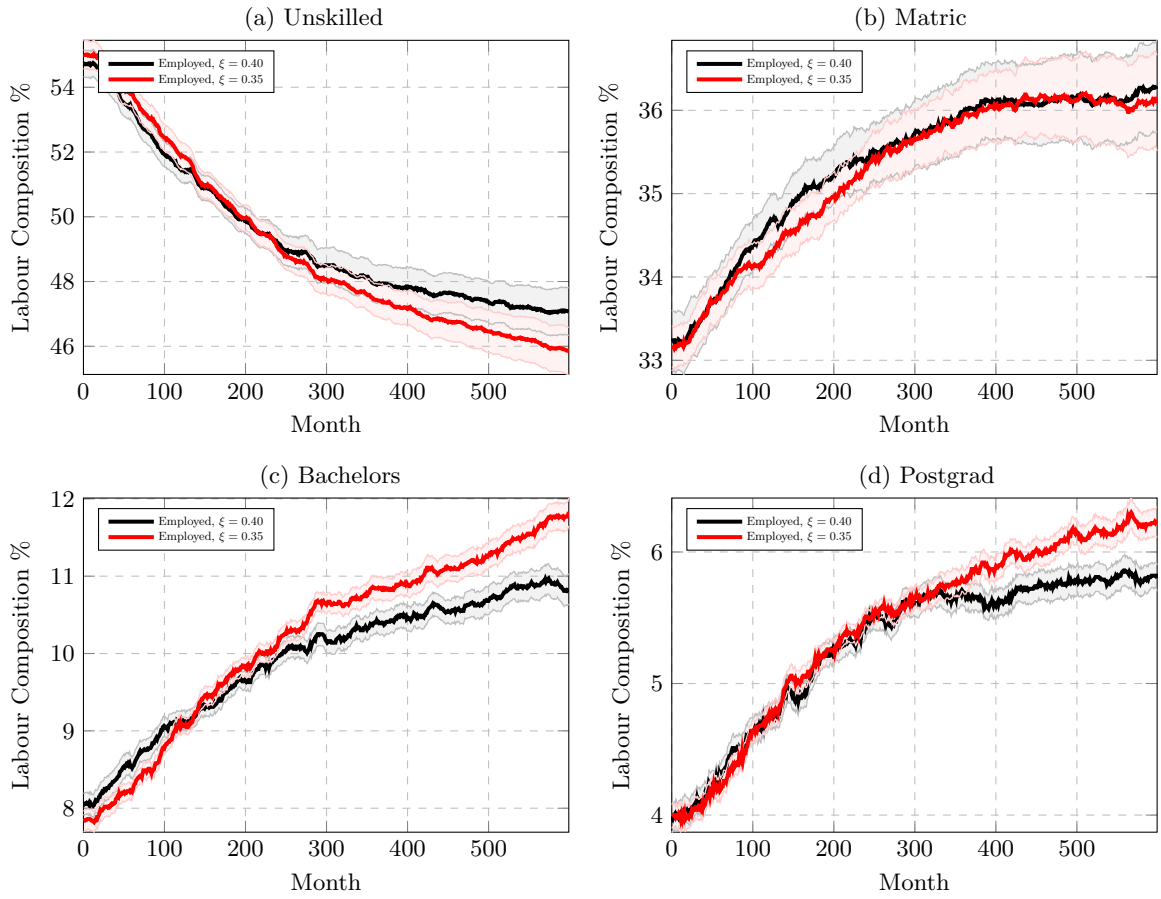


Figure 5.18: A comparison of the total labour composition by education for the simulation scenarios. Shaded regions indicate uncertainty bounds.

5.2.2.2 Factors of production

To compare the impact on all the factors of production, all the factors were normalised to their initial values, therefore showing relative changes. The results for capital, labour, knowledge and technology changes over time for the two simulation scenarios are shown in Figure 5.19. As expected, the baseline simulations show higher growth in capital and labour, since the *capital-labour investment probability* parameter, $\hat{\xi}$, is larger than in the explored simulation. The larger aggregate investment in knowledge and technology is also evident in the explored simulations, representing an overall shift towards a knowledge-based economy.

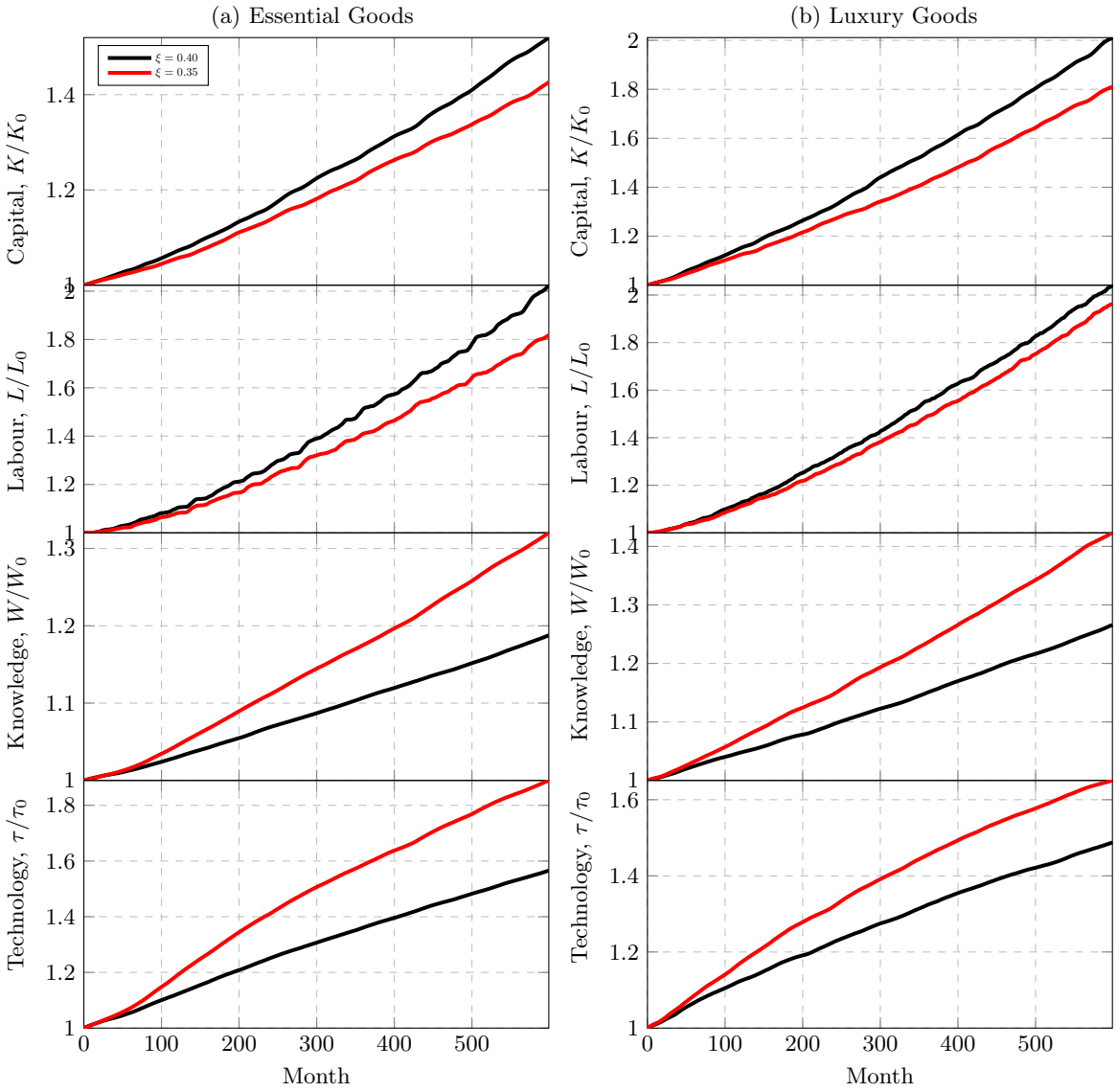


Figure 5.19: Relative changes in the factors of production for the two simulation scenarios.

5.2.2.3 Labour and capital productivity

In the firm production function given in Equation 4.9 on page 44, there are two types of productivity parameters used: a capital productivity, A_f^{K}, and labour productivity parameters, A_{fe}^{L}. The effect on production capacities for the two scenarios may be understood by considering the plots shown in Figures 5.20 and 5.21. The plots show the relative change in output corresponding to relative changes in labour and capital. The relationship between aggregate production capacity and total labour shown in Figure 5.20, where both variables have been normalised to their initial values. The slope of the curves provide a measure of the aggregate efficiencies or productivities in the economy. For the explored scenario, aggregate labour productivity and capital productivity is shown to improve relative to the baseline scenario.

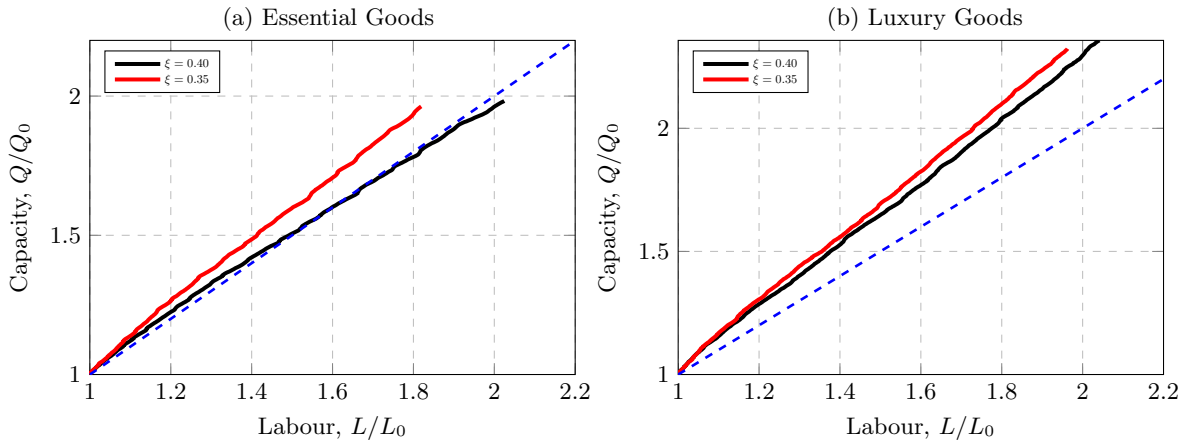


Figure 5.20: Change in capacity vs labour for the two simulation scenarios. Data is normalised to the initial values, Q_0 and L_0.

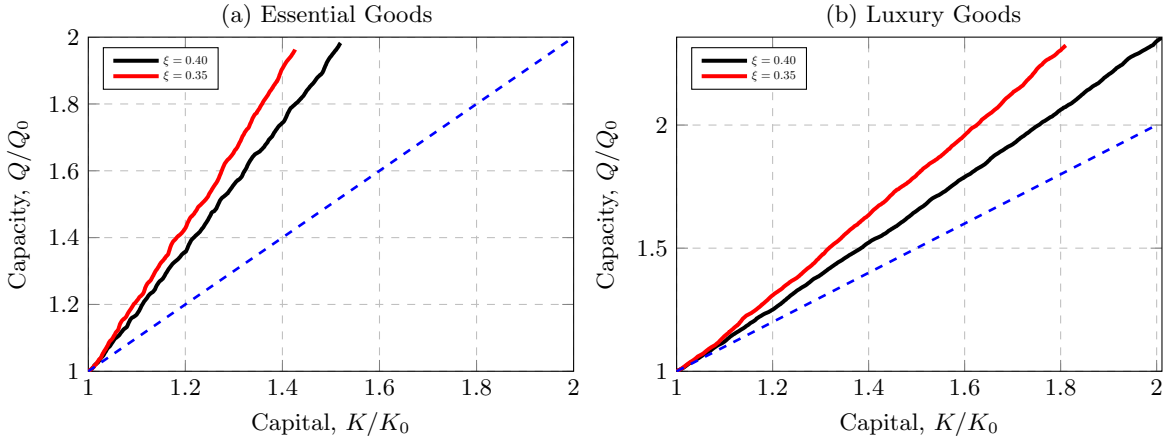


Figure 5.21: Change in capacity vs capital for the two simulation scenarios. Data is normalised to the initial values, Q_0 and K_0.

5.2.3 Macroeconomic results

5.2.3.1 GDP

Comparing the change in GDP, shown in Figure 5.22, we note a slight decrease in the long-term GDP from the baseline simulation to the explored simulation. The decrease is the result of slightly lower consumption, with exports and investment showing little change. The lower consumption is a consequence of changes in unemployment, shown in the following section. As highlighted earlier, GDP in the model is influenced by the population growth rate, which was unchanged between the two scenarios.

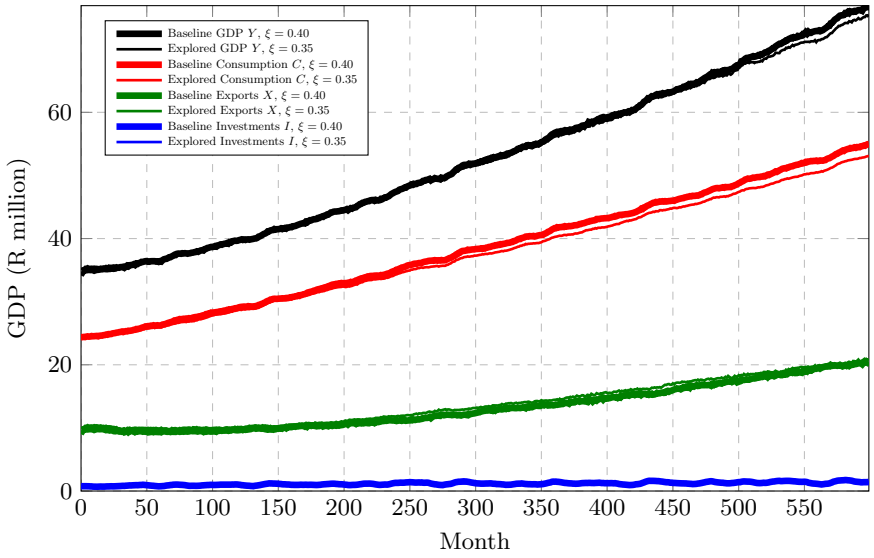


Figure 5.22: Comparison of GDP for the two simulation scenarios.

5.2.3.2 Unemployment

Figure 5.23 shows the unemployment rate for the two scenarios. In the long-term, there is significantly higher unemployment in the explored scenario. The higher levels of unemployment impacts the spending patterns of the population. A larger number of consumer's with low disposable income, would result in lower aggregate consumption and hence a lower GDP.

Comparing the change in unemployment for the different skill levels, shown in Figure 5.24, it can be seen that there are significant increases in unemployment for the unskilled and semi-skilled education levels of the population. For the more highly skilled sectors of the population, unemployment steadily declines, with lower levels of unemployment long-term for the explored scenario. This shows the need for a more highly skilled workforce in the long-term when there is a shift towards higher levels of technological sophistication in the economy.

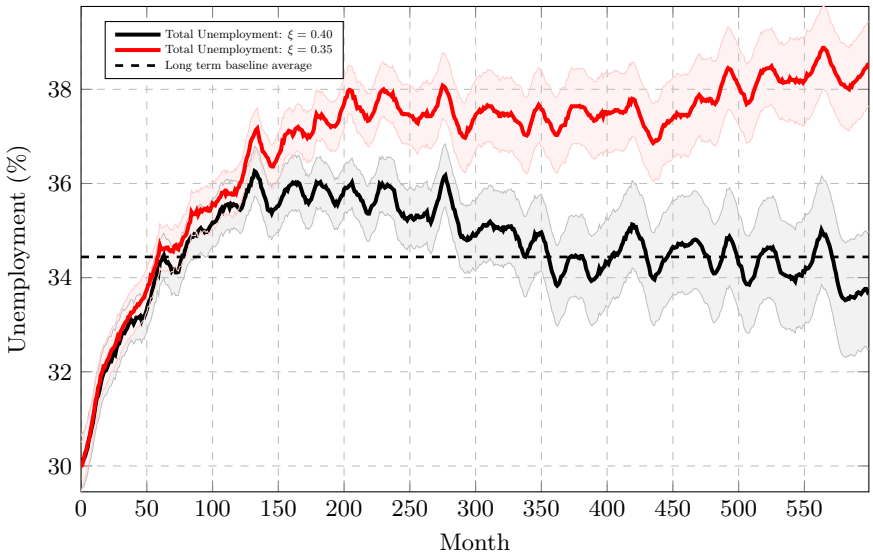


Figure 5.23: Comparison of the total unemployment for the difference simulation scenarios. Shaded regions indicate uncertainty bounds.

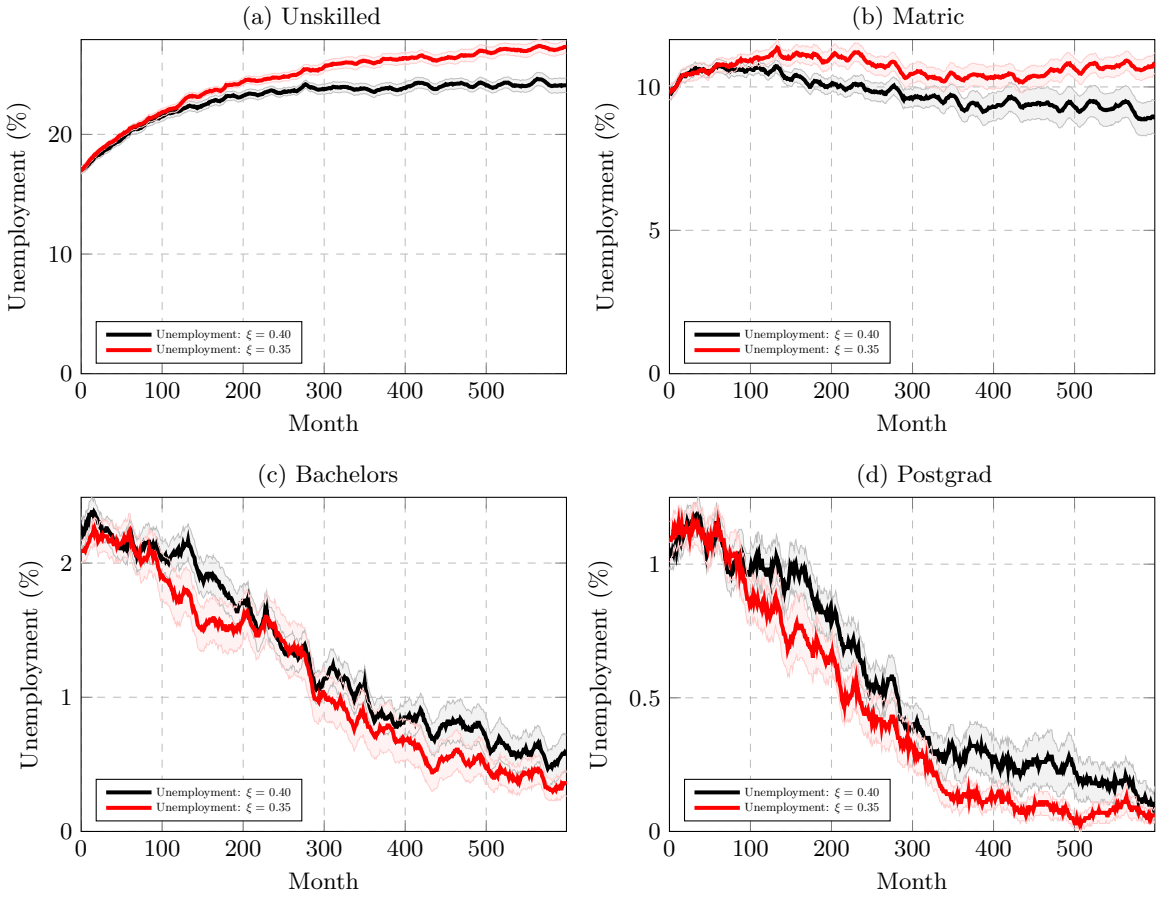


Figure 5.24: Comparison of unemployment for the simulation scenarios and different education levels. Shaded regions indicate uncertainty bounds.

5.2.3.3 Inequality

It was also possible to calculate the Gini index for the population, as shown in Figure 5.25. What is evident is the relatively high levels of inequality in both scenarios, with no significant change in the Gini index in the long-term. According to Statistics South Africa (2019*a*), South Africa currently has a Gini coefficient between 0.65 and 0.67, which is relatively close to the long-term values obtained in the simulations.

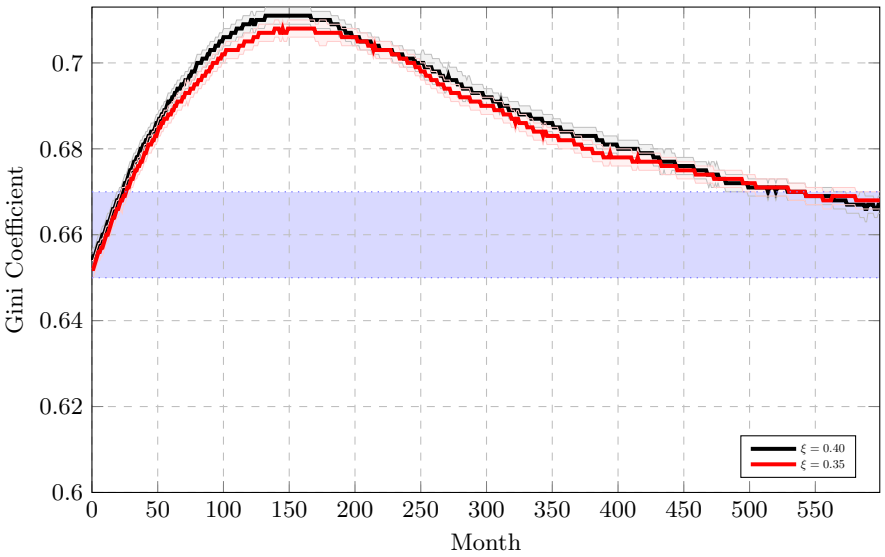


Figure 5.25: The Gini coefficient obtained from the simulations. The blue shaded region shows the range of Gini coefficients for South Africa (Statistics South Africa 2019*a*).

5.3 Validation of results

The results obtained can be validated according to established macroeconomic models.

5.3.1 Solow-Swan growth model

The exogenous, neoclassical Solow-Swan model serves as a simple macroeconomic model for understanding economic growth (Sørensen & Whitta-Jacobsen 2010, Romer 2011). In Appendix B, the solution to the Solow-Swan model is obtained as

$$Y(t) = Y_0 k_0^{-\alpha} e^{(g+n)t} \left[\frac{s}{\delta + g + n} \left(1 - e^{-(\delta+g+n)(1-\alpha)t}\right) + k_0^{1-\alpha} e^{-(\delta+g+n)(1-\alpha)t} \right]^{\frac{\alpha}{1-\alpha}}, \qquad (5.1)$$

where, Y_0 is the initial GDP; k_0 is the initial capital per unit of effective labour; α is the aggregate *output elasticity for capital* used in a 2-input Cobb-Douglas production function; δ is the capital depreciation rate; s is the savings rate; g is the labour productivity growth rate; and n is the labour growth rate. In validating the simulations to the model, the Solow-Swan solution given in (5.1) was fitted to the GDP data for the two simulation scenarios shown in Figure 5.26. The fits were performed using a standard Python library function: scipy.optimize•curve_fit().

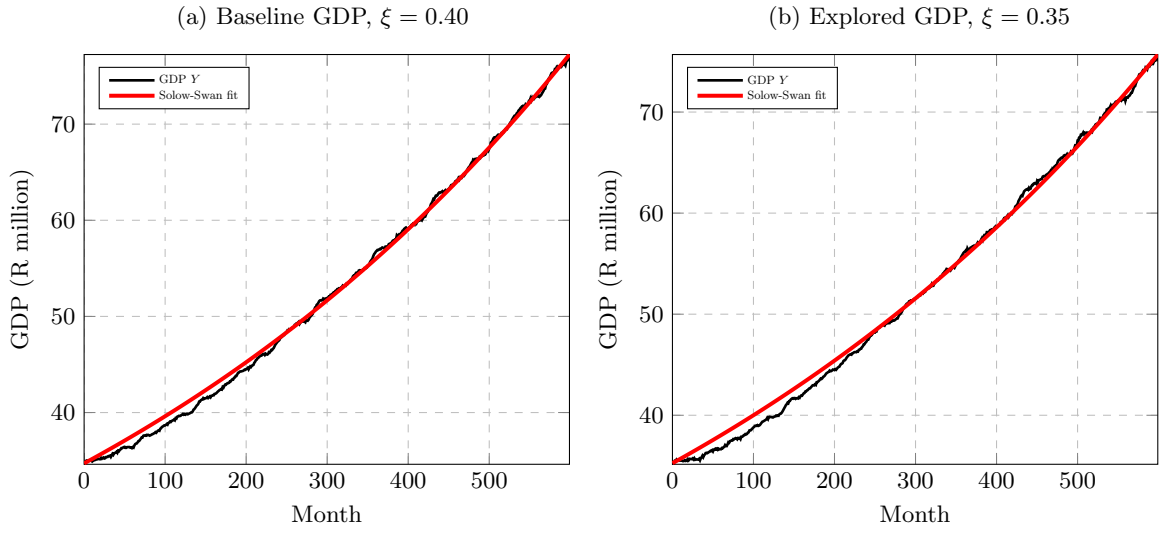


Figure 5.26: Comparison of GDP to the Solow-Swan fitted solutions.

In performing the fit, some of the seed parameter values could be estimated from the other simulation results. The initial GDP was simply obtained as the value of GDP in the first month. The labour growth rate, n, was obtained by performing an exponential fit to the total labour force, elements of which were shown in Figure 5.17. As described in Appendix B.1.2, the savings rate in the Solow-Swan model relates the capital investment to GDP, $I(t) = sY(t)$. Using the capital investment results (such as those shown in Figure 5.7), together with the GDP data, it was possible to calculate the average savings rate, $\bar{s}$ over the duration of the simulation run, which was used as the base parameter for the fit. The average savings rate obtained for the baseline simulation was found to be $\bar{s}_b = 1.77\% \pm 0.011\%$; while for the explored simulation it was found to be $\bar{s}_e = 1.48\% \pm 0.009\%$. These values indicate a lower overall investment in

capital in the explored scenario since there was a greater investment in knowledge in that case. The other seed parameter values for the fit were chosen based on 'educated' guesses and some trial and error. (For example, α was chosen to be equal to 0.5.) Furthermore, the parameters were also suitably constrained in the fitting process. (For example, α was constrained to lie between 0 and 1, while s was constrained to lie within one standard deviation of the mean value $\bar{s}$.)

A fit to the baseline simulation results produced the following model parameters:

$$Y_0 = 34.7,\ k_0 = 0.07,\ \alpha = 0.499,\ \delta = 0.0\%,\ s = 1.78\%,\ g = 0.132\%\text{p.m.},\ n = 0.12\%\text{p.m.}$$

Fitting the explored simulation results produced the model parameters:

$$Y_0 = 35.3,\ k_0 = 0.06,\ \alpha = 0.503,\ \delta = 0.0013\%,\ s = 1.49\%,\ g = 0.137\%\text{p.m.},\ n = 0.11\%\text{p.m.}$$

A number of observations can be made from these parameters:

(a) The savings rate is reduced from 1.78% in the baseline simulation to 1.49% in the explored simulation. This highlights a decrease in capital investment in the explored simulation scenario, which was the result of the re-allocation of investment towards knowledge development within the firms.

(b) A relevant difference between the fit parameters is the increase in the aggregate labour productivity growth rate from 0.132% p.m. in the baseline simulation to 0.137% p.m. in the explored simulation. Recall that the knowledge level effects in the production function of of an individual firm (§4.3.4.1, pg.44) were such that an increase in the firm's investment in knowledge would result in an increase in the labour and capital productivities. The explored simulation results showed a relative increase in total knowledge investment, as was shown in Figure 5.19, which would result in an overall increase in the productivities for all firms. Increases in aggregate labour and capital productivities relative to the baseline scenario were shown in Section 5.2.2.3 on page 73. This increase in the productivities is manifested in the higher aggregate labour productivity growth rate parameter, g, obtained in the Solow-Swan model fit.

The change in parameters of the Solow-Swan growth model therefore confirms the change in behaviour of the agent-based model between the two simulation scenarios.

5.3.2 Okun's law

Okun's law predicts a relationship between changes in GDP and unemployment (Sørensen & Whitta-Jacobsen 2010, Romer 2011, Burda & Wyplosz 2017). The premise is that higher unemployment suppresses GDP growth. A linear growth form of Okun's law may be written as (Abel et al. 2016)

$$\frac{\Delta Y}{Y} = k + c\,\Delta u, \tag{5.2}$$

where $\frac{\Delta Y}{Y}$ is the percentage change in growth of GDP, Δu is the change in actual unemployment, k is the intercept at $\Delta u = 0$, and c is the slope relating changes in unemployment to changes in GDP. Historical data showed that there is a tendency for increases in GDP to be accompanied by decreases in unemployment (Sørensen & Whitta-Jacobsen 2010). This means that Okun's law postulates that the slope parameter, c, must be negative.

Figure 5.27 shows the monthly percentage change in GDP versus the change in unemployment for the two simulation scenarios. The data points show the annual changes in the Monte Carlo averages from the simulations. A linear regression was performed on the data from each of the two scenarios, with the regression lines shown in the figure. In both cases a negative slope was obtained, with $c = -1.195$ in the baseline scenario, and $c = -1.112$ in the explored scenario, confirming the essential postulate of Okun's law. In the model, a 1% increase in unemployment results in a 1.195% decrease in GDP for the baseline scenario, and a slightly smaller 1.112% decrease in GDP for the explored scenario. These values are of the same order of magnitude as those obtained from a recent study on South African data (Mazorodze & Siddiq 2018).

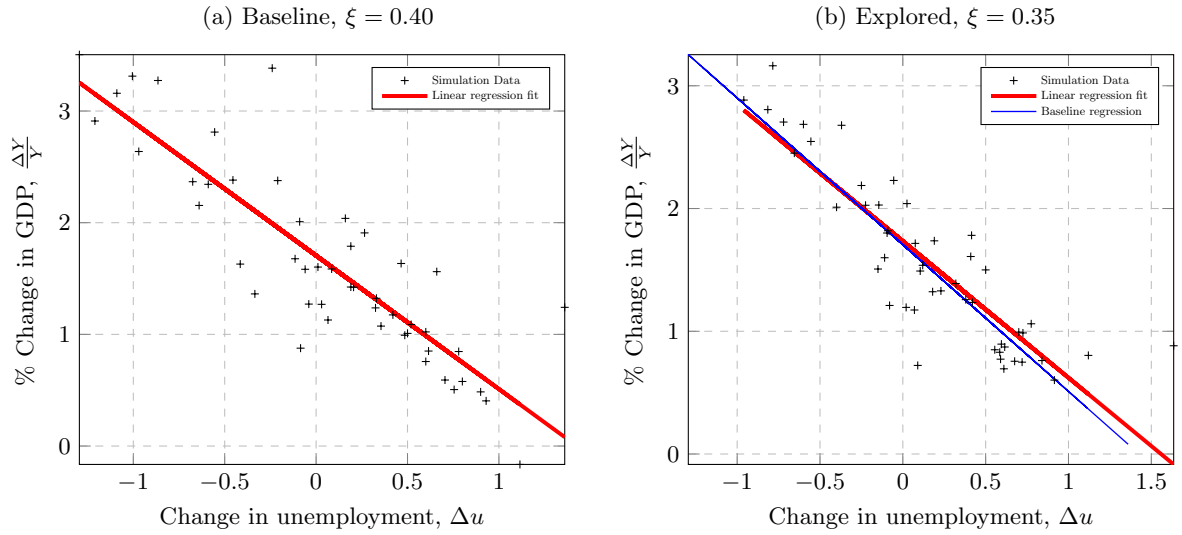


Figure 5.27: Comparison of annual % change in GDP to change in unemployment for the two simulation scenarios, showing Okun's law.

CHAPTER 6

Discussion & Conclusions

An agent-based model was developed in order to simulate and investigate the macroeconomic consequences emerging from the strategic decisions of firms to invest in knowledge development. Technological progress through innovation and knowledge development are important drivers of economic growth (Diaconu 2011, Asongu & Andrés 2020). Two complementary aspects of technological progress were used in the model: technological sophistication and investment in knowledge. Two simulation scenarios were constructed, one used as a baseline or reference scenario, and an explored scenario in which there was a higher tendency by firms to invest in knowledge development.

The simulation results showed no significant change in GDP as firms in the model economy invested more heavily in knowledge. However, the greater investment in knowledge development resulted in significantly higher levels of unemployment, particularly for unskilled workers. The unemployment rates for the more highly skilled sectors of the population decreased over time as firms became more technologically sophisticated. The results were validated by fitting the simulation data to the Solow-Swan growth model and through the confirmation of Okun's law relating GDP growth and unemployment.

6.1 Implications

The results from the model highlight several implications for business and macroeconomic policy decisions, especially in a South African context.

6.1.1 A holistic approach

The simulations only considered the consequences resulting from a change in a single parameter: investment in knowledge. The composition of skills within the population was unchanged as firms invested more heavily in knowledge. This resulted in stagnant economic growth and higher levels of unemployment. Attempting to leverage the advantages of the 4^{th} industrial revolution (4IR) and the knowledge-based economy (KBE) without increasing the skills composition of the population will ultimately constrain long-term future growth opportunities.

An aggregate measure of the level of development of a KBE is provided by the World Bank Knowledge Economy Index (KEI), (Chen & Dahlman 2005). The KEI incorporates the four pillars of the knowledge economy: an educated and skilled labour force; an effective system to stimulate innovation; a modern infrastructure for information and communication technology; and suitable economic incentives, together with sound economic policies (Chen & Dahlman 2005, Asongu & Andrés 2020). Sustained investment in these four pillars is a necessary condition to successfully become a KBE. The important consequence of the four pillars' framework is that no single economic stakeholder can be the driver for knowledge based development. A coordinated strategy between academia, industry and government is imperative (Etzkowitz & Leydesdorff 1995, Leydesdorff 2012). Academic curricula need to be aligned with the needs of industry. Government policies need to facilitate human capital development; be aligned with the needs of industry and educational institutions; allow for the efficient allocation of resources; stimulate creativity; encourage collaboration and integration; promote research and development; and incentivise innovation and knowledge development. (Chen & Dahlman 2005, Asongu 2017). Companies need to engage more closely with academic institutions and implement skill development programmes.

The simulations show that unemployment is highest for the unskilled sector of the population, despite being the largest labour composition within firms. Furthermore, a shift towards a knowledge-based economy resulted in even higher levels of unemployment, particularly within the unskilled and semi-skilled labour force. An important consideration is that the education throughput rates were the same for all simulations, so the population composition of skills was unchanged. Furthermore, it was found that in the long-term, inequality will remain relatively unchanged when moving towards a KBE, *ceteris paribus*. The results reinforce the observation that education inequality will lead to higher levels of poverty and income or wealth inequality (Akanbi 2016).

Implication: *Failure to adopt a coordinated strategy between industry, academia, and government in attempts to achieve a KBE may result in constrained long-term economic growth and deteriorating levels of unemployment, inequality and poverty.*

6.1.2 Education & skills development

A largely unskilled population arising from poor education throughput in the model resulted in higher levels of unemployment when moving towards a KBE. In South Africa, despite modest improvements in the education system, there are significant challenges that remain. A recent report by the Centre for Risk Analysis highlighted several of these challenges, including poor throughput rates in the basic education system, high failure rates, large disparities in the attainment of educational qualifications between different racial groups, poor infrastructure in schools, etc. (CRA 2018). "The education system represents the single greatest obstacle to socio-economic advancement in South Africa." (CRA 2018, pg.3) Three issues were identified as being particularly problematic. First, was the generally poor quality of mathematics education in schools. This is significant because mathematics forms the foundation for just about all technological fields of study. "A good maths pass in matric is in all probability the most important marker in determining whether a young person will enter the middle classes" (CRA 2018, pg.3). Secondly, a low rate of participation in tertiary education by black people was observed. Without a significant improvement in tertiary education participation, poverty alleviation and inequality will persist. Finally, a very high school drop-out rate, with approximately half the children entering the schooling system actually finishing school. These dire issues were the basis for the educational throughput parameters used in the model.

Investment in human capital, as well as the necessary infrastructure to support high-technology industries, is essential in order to develop a KBE (Blankley & Booyens 2010). Perhaps the most critical component required to develop the capacity for a KBE is a highly skilled population (Blankley & Booyens 2010, Vadra 2017). Some important aspects of developing a highly skilled population include: increasing the quality of the education system, balancing the needs of a general education with a technical education, increasing R&D, and promoting a culture of lifelong learning (Asongu & Odhiambo 2020). In part, this would require a more flexible education system that can adapt to developing the necessary skills required by a more technologically sophisticated economy (Lee et al. 2018). Companies wanting to shift towards higher levels of technological sophistication and knowledge utilisation should invest in human capital development and engage more closely with educational institutions to align curricula with their needs.

Implication: *A necessary condition for a KBE is the transformation of the education system to one that is inclusive, focused on quality, adaptive, encourages creativity, aligned with the needs of industry, and stimulates R&D.*

6.1.3 The efficiency incentive

An economic incentive for firms to invest in knowledge development is to improve efficiencies. Higher levels of productivity and efficiency can be achieved from technological progress through knowledge development (Perelman 1995, Powell & Snellman 2004). The agent-based model resulted in higher aggregate capital and labour productivities in the economy. When comparing the simulation results to the Solow-Swan growth model, it was found that an increase in investment in knowledge and technology results in an increase in the aggregate Solow-Swan labour productivity parameter. Firms would undoubtedly embrace productivity-improving initiatives in order to increase profitability and operating efficiencies.

Implication: *Improved efficiencies provide an economic incentive for companies to invest in knowledge development.*

6.1.4 To KBE, or not to KBE, that is the question

"A knowledge-based economy will not necessarily ensure national economic prosperity, improved health and well-being, ecological sustainability and reduced inequalities." (Blankley & Booyens 2010). In other words, a knowledge-based economy is *not* a silver bullet that can be used to solve broad socio-economic problems of poor economic growth, unemployment, inequality and poverty. It would seem that the simulation results show that a country like South Africa should not embrace the KBE paradigm. What the simulation does show is that embracing a knowledge economy without changing anything else would be problematic in the long-run. A holistic, multi-faceted approach is required.

There is also another dimension that must be considered: global competitiveness. New opportunities for growth and development in developing countries have been created by the globalisation of technology (Vadra 2017). The model assumed that global demand would steadily increase regardless of anything else. In reality, not adopting global trends in technological progress could leave a country behind due to lower levels of productivity.

6.2 Alternative scenarios

With the simulation framework used, it would be possible to investigate alternative scenarios by adjusting other parameters in the model. Some possibilities are highlighted as follows:

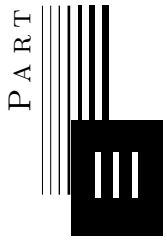
- The educational throughput probability parameters (discussed in §4.2.2, pg.37) could be changed in order to investigate the impact of changes in the education system. Weybright et al. (2017) has described the South African school dropout rate as a national crisis. Within the model, the proportion of individuals finishing school can be adjusted to assess the macroeconomic the impact of even marginal improvements in throughput rates.
- A fixed amount for social grants (see §4.5, pg.55) has been used and by using alternative social grant amounts, the broader economic impact can be investigated.
- The growth rate parameter for the foreign market could be adjusted to assess the impact of lower global competitiveness if a KBE is not adopted.

6.3 Recommendations for improvements

There are several possible improvements that could be made to the model presented. These are briefly itemized as follows:

- The model can be extended to incorporate more than two products in the consumer product market.
- Alternative production functions could be used for different firms, either within the various product sectors, or between different product sectors.
- Make the number of firms in the economy dynamic, allowing underperforming firms to close and new firms to be created in a market with high demand.
- Incorporate a more sophisticated mechanism for selecting both consumer and firm buying preferences.
- Allow for salary increases.
- Introduce income tax on employed individuals.
- Use a production-weighted average price for the market price of each good (Gualdi et al. 2015).
- Make government an employer and expand the role of government in the economy.
- Introduce materials supplier firms, with an associated market for raw materials. They would also employ workers through the labour market. This would result in a modification to the calculation of a firm's costs and resulting profitability.

- Modify the implementation to incorporate multiple capital and knowledge producing firms. This would result in an additional market for capital goods and knowledge/technology.
- In validating the model, it may be informative to compare the simulation results to alternative growth models, such as the Romer model (Romer 1990, 2011), the Ramsey-Cass-Koopmans growth model (Romer 2011, Ch. 2), or a Schumpeterian growth model (Acemoglu 2009, Ch. 14).
- Introduce 'shocks' in the economy. For example, removing a country in the foreign market.
- Allow for both short- and long-term adjustments in labour.
- It would be necessary to parallelise the existing code in order to reduce the simulation time.



Appendices

Notation

A.1 Software objects

All software elements are identified in the text using a sans-serif font.

- Constants are identified using capitalised, italic font. For example, the set of education values is written as: *Education*.
- Instances of objects are identified using lower-case. An instance of an individual is written as: individual; an instance of a firm is written as: firm; an instance of the product market is written as: product_market.
- Attributes of an object are referenced using dot-referencing. The age attribute of an individual is written as individual•age.
- Classes are identified using bold capitalisation. The individuals class is written as **Individual**. Therefore an individual is an instance of the class **Individual**; and product_market is an instance of **ProductMarket**.
- Methods are written in a number of ways, all using lower-case and identified with bracketting, "()". A generic method is written as: calculate_needs(). A particular object's method uses dot-referencing and is written as: individual•calculate_needs(). A particular object's method with parameters is identified as: individual•has_money(amount).

A.2 Indexing

Index	Description
f	Firm index
i	Individual index
g	Product. See §4.3.1, pg.41
e	Level of education. See §4.2.2, pg.37

A.3 Firms

Firms	
Variable	**Description**
p_{fg}	Product g sales price of firm f
$\bar{p}_g^{\mathrm{M}}$	The average market price for a product g
Π_{fg}^{M}	Market price ratio for the firm - see equation 4.14, pg.51
q_f	Actual production by firm f
$q_f^{\max}$	Production capacity of firm f
Q	Aggregate actual production by all firms
$Q^{\max}$	Aggregate production capacity of all firms
Δ_f	Stock of firm f
δ_f	Relative stock of firm f - see equation 4.11, pg. 47
$\langle\delta\rangle_f$	N-point moving average of relative stock of firm f - see equation 4.12, pg. 47
M_f	Number of months for moving average of stock - see equation 4.12, pg. 47
N^{F}	The total number of consumer firms
N_g^{F}	The total number of firms producing a product g
τ_f	Technological sophistication of the firm
K_f	Capital of firm f
W_f	Knowledge of firm f
C_f	Overall production scaling parameter in Cobb-Douglas production function
A_f^{K}	Capital productivity in Cobb-Douglas production function
A_{fe}^{L}	Labour productivities for each education level in Cobb-Douglas production function
α_f	Output elasticity for capital in Cobb-Douglas production function
$\vec{\ell}_f$	Skills composition or qualification mix as proportions
L_f	The total number of employees in firm f
$\vec{\boldsymbol{L}}_f$	Labour mix vector of employees for firm f
L_{fe}	Labour mix vector component for firm f with education level e
S_f	Savings (or accumulated wealth) of firm f
R_f	Monthly revenue of firm f

Note: In places where the firm index, f, is implied, it will be dropped. For example, the firm technological sophistication τ_f, will sometimes be simply written as τ.

A.4 Mathematical notation

Some of the mathematical notation used:

$\equiv$	Means "is equivalent to"
$:=$	Means "is defined as"
$\overset{!}{=}$	Means "must equal"
$\sim$	Means "is distributed as"
$\approx$	Means "approximately equal to"

- Δx: Means a change in a variable x.
- $\tilde{x}$: Denoted the average value of the variable x obtained over all Monte Carlo simulation runs.
- $\hat{x}$: Denotes x as a simulation parameter.
- $x \sim U[a, b]$: Means x is a random variable with a value drawn from a uniform distribution between a and b.
- $\mathcal{P}$(decision): The probability, between 0 and 1, of a particular decision being made.
- $\langle x \rangle$: The N-point moving average value of a variable $x(t)$ is denoted by

$$\langle x(t) \rangle_N := \frac{1}{N} \sum_{n=1}^{N} x(t-n) \tag{A.1}$$

A.5 Government variables

Index	Description
r_{G}	Corporate tax rate
$P_{\mathrm{tot}}^{\mathrm{G}}$	Total grant payments
N_{tot}	Total net profit of all firms.

A.6 Economic variables

Index	Description
Y	GDP
C	Consumption
X	Exports
I	Investment

APPENDIX B

The Solow-Swan growth model

This appendix provides an overview of the Solow-Swan growth model (Solow 1956, Swan 1956). For a variable, $x(t)$, that changes with time, the growth of x is defined as

$$g_x(t) := \frac{\dot{x}(t)}{x(t)}, \tag{B.1}$$

where $\dot{x}(t)$ is the instantaneous rate of change (or time derivative) of x. For example, considering a constant population growth rate, g, then

$$\frac{\dot{N}}{N} = g. \tag{B.2}$$

This results in exponential population growth[1],

$$N(t) = N_0 e^{g(t-t_0)} \tag{B.3}$$

where N_0 is the initial population at time t_0. Performing an exponential fit to the South African population data (as shown in Figure B.1) results in $g \approx 2.239\%$ per annum. This is in fairly good agreement with the average annual population growth rate, $\bar{a} \approx 2.076\%$.

The exogenous, neoclassical Solow-Swan model serves as a simple baseline model for understanding economic growth (Romer 2011). Here we will consider a continuous-time version of the Solow-Swan model, described by (Romer 2011), as opposed to the discrete-time version, described by (Sørensen & Whitta-Jacobsen 2010).

Before describing the model, it is important to state the assumptions of the Solow-Swan model.

[1]The ordinary differential equation, $\dot{x} = gx$, can be separated and integrated,

$$\int_{x_0}^{x} \frac{1}{x} dx = \int_{t0}^{t} g dt \quad \Rightarrow \quad \ln(x)|_{x_0}^{x} = g(t - t_0) \quad \Rightarrow \quad x(t) = x_0 e^{g(t-t_0)}$$

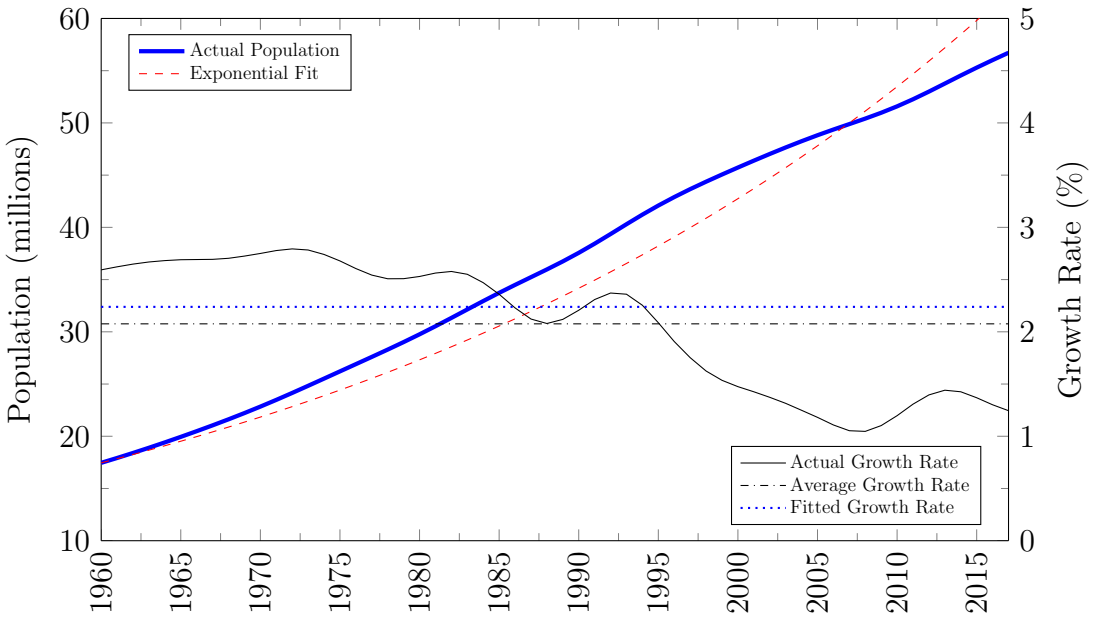


Figure B.1: Time series of population for South Africa. Dashed lines indicate fitted population growth, see text. Raw data source: *World Bank Open Data* (n.d.)

- There are four key variables that are used in the model: output, $Y(t)$; capital, $K(t)$; labour, $L(t)$ and a knowledge or labour productivity variable, $A(t)$.
- There is a single good in the economy, that is produced with only two factors of production: capital, $K(t)$ and labour, $L(t)$. This implies the existence of a generic global production function

$$Y(t) = F(K(t), A(t)L(t)). \tag{B.4}$$

  The combination of labour with productivity, $A(t)L(t)$, is referred to the *effective labour*.
- The production exhibits constant returns to scale with respect to the two factors of production.
- Changes to capital stock is only determined by new investments and depreciation.

For concreteness, the Cobb-Douglas production function is commonly used in the Solow-Swan model, and we will consider the following form

$$Y(t) = K(t)^{\alpha}(A(t)L(t))^{1-\alpha} \tag{B.5}$$

where α is the *output elasticity for capital*, satisfying $0 < \alpha < 1$, $\beta = 1 - \alpha$ is the *output elasticity for labour* and $A(t)$ is a *labour productivity* variable. By construction, since the sum of the capital and labour elasticities equals 1 (i.e. $\alpha + \beta = 1$), this form of the production function exhibits constant returns to scale.

The output is split between consumption and investment (or savings). If we introduce an exogenous savings rate parameter, s, then the investment in new capital is given by $I(t) = sY(t)$. Hence consumption is given by $C(t) = (1 - s)Y(t)$.

We can calculate the output and capital stock per unit of effective labour, $A(t)L(t)$. Dividing both sides of the Cobb-Douglas production function by $A(t)L(t)$, we obtain

$$\frac{Y(t)}{A(t)L(t)} = K(t)^{\alpha}\frac{A(t)^{1-\alpha}L(t)^{1-\alpha}}{A(t)L(t)} = \frac{K(t)^{\alpha}}{(A(t)L(t))^{\alpha}} \tag{B.6}$$

If we define the output, $y(t)$, and capital, $k(t)$, per unit of effective labour as

$$y(t) := \frac{Y(t)}{A(t)L(t)} \qquad \text{and} \qquad k(t) := \frac{K(t)}{A(t)L(t)} \tag{B.7}$$

then the output is simply expressed in terms of the capital

$$y(t) = k(t)^{\alpha}. \tag{B.8}$$

Therefore, in order to determine the output, $y(t)$, we would need to understand the evolution of the capital, $k(t)$.

B.1 Evolution equations

B.1.1 Labour and productivity

The labour productivity is assumed to change at a constant rate, g,

$$\dot{A}(t) = gA(t), \tag{B.9}$$

while labour is also assumed to grow at a constant rate, n,

$$\dot{L}(t) = nL(t). \tag{B.10}$$

Both g and n are exogenous, constant parameters in the model. With the constant rates n and g, productivity and labour both grow exponentially

$$A(t) = A_0 e^{gt} \qquad \text{and} \qquad L(t) = L_0 e^{nt}, \tag{B.11}$$

where A_0 and L_0 are the initial values at time $t = 0$. From the definition of the output per unit of effective labour, (B.7), we have $Y(t) = A(t)L(t)y(t)$ and then using the relationship (B.8), we obtain

$$Y(t) = A(t)L(t)k(t)^{\alpha}. \tag{B.12}$$

Taking logarithms of both sides (and again using the notation $\tilde{y} \equiv \ln y$), one obtains

$$\tilde{Y}(t) = (\tilde{A}_0 + \tilde{L}_0) + (g+n)t + \alpha\tilde{k} \tag{B.13}$$

and if one lumps the two initial conditions together by defining $\tilde{C}_0 := \tilde{A}_0 + \tilde{L}_0$, one has

$$\boxed{\tilde{Y}(t) = \tilde{C}_0 + (g+n)t + \alpha\tilde{k}(t).} \tag{B.14}$$

B.1.2 Capital

Increases in capital stock are determined from new investments, based on an investment (or savings) rate, s. Investment in new capital is given by $I(t) = sY(t)$. Decreases on capital stock are assumed to be only due to depreciation at a rate, δ. The depreciation in capital stock is therefore $\delta K(t)$. Both the investment rate, s, and the depreciation rate, δ, are exogenous parameters in the model. We are assuming that these are the only contributing factors contributing towards changing the level of capital in the economy. The resulting dynamic equation for the capital stock is therefore

$$\dot{K} = sY(t) - \delta K(t). \tag{B.15}$$

This equation is sometimes referred to as the *capital accumulation equation* (Sørensen & Whitta-Jacobsen 2010). We now need to determine the dynamics of the capital per unit of effective labour[2], $k(t)$. From the definition of $k(t)$ we obtain using rules for differentiation (dropping showing the time dependence for clarity)

$$\dot{k}(t) = \frac{d}{dt}\left(KA^{-1}L^{-1}\right) = \frac{\dot{K}}{AL} - \frac{K}{AL}\frac{\dot{A}}{A} - \frac{K}{AL}\frac{\dot{L}}{L} \tag{B.16}$$

We can now substitute in from (B.9), (B.10) and (B.15) as well as the definition of $k(t)$,

$$\dot{k}(t) = \frac{sY - \delta K}{AL} - gk(t) - nk(t) = s\frac{Y}{AL} - \delta\frac{K}{AL} - gk(t) - nk(t) \tag{B.17}$$

Recognising that $y = Y/AL = k^\alpha$ in (B.8), we finally obtain the evolution equation

$$\boxed{\dot{k}(t) = sk(t)^\alpha - (\delta + g + n)k(t).} \tag{B.18}$$

This is the key dynamic equation for the Solow-Swan model. The rate of change of capital is determined from two terms in the equation. The first is through investment, s. The second term is composed of three exogenous parameters, $\delta + g + n$, is referred to as the *breakeven investment*. Writing $b := \delta + g + n$, then the capital evolution equation can be written as

$$\dot{k}(t) = sk(t)^\alpha - bk(t). \tag{B.19}$$

If we assume that the initial capital stock at time $t = 0$ is given by k_0, i.e.

$$k_0 = \frac{K_0}{A_0 L_0}, \tag{B.20}$$

then the solution to the capital evolution equation is given by (see the following page)

$$\boxed{k(t) = \left[\frac{s}{b}\left(1 - e^{-b(1-\alpha)t}\right) + k_0^{1-\alpha}e^{-b(1-\alpha)t}\right]^{\frac{1}{1-\alpha}}.} \tag{B.21}$$

[2]For convenience it will often be simply referred to as capital

Solution to the capital equation: The differential equation for the evolution of the capital, written as

$$\dot{k} + bk = sk^{\alpha}. \tag{B.22}$$

This is of the form of a *Bernoulli equation*, which is an equation of the form

$$y' + P(x)y = Q(x)y^n, \quad \text{where} \quad n \neq 0, 1 \tag{B.23}$$

so that a textbook approach to solving the equation can be used (Zwillinger 1998). Let's make a substitution

$$x(t) = k^{1-\alpha}, \tag{B.24}$$

with a derivative given by

$$\dot{x} = (1-\alpha)\dot{k}k^{\alpha}. \tag{B.25}$$

Substituting into the differential equation (B.22) results in

$$\dot{x} + b(1-\alpha) = s(1-\alpha). \tag{B.26}$$

For notational convenience, we will express the substitution in terms of the labour elasticity, $\beta = 1 - \alpha$,

$$x(t) = k(t)^{\beta}, \quad \text{so that} \quad k(t) = x(t)^{1/\beta}, \tag{B.27}$$

and the resulting equation for $x(t)$ becomes

$$\dot{x} + b\beta = s\beta. \tag{B.28}$$

This is a first-order linear ordinary differential equation, with a general solution given by

$$x(t) = e^{-z}\left(\int e^{z} s\beta \, dt + C\right), \quad \text{with} \quad z = \int b\beta \, dt, \tag{B.29}$$

where C is an integration constant determined from initial conditions. Since b, s and β are constants, the integrals can be easily performed, resulting in the solution,

$$x(t) = \frac{s}{b} + Ce^{-b\beta t}, \tag{B.30}$$

and so one obtains

$$k(t) = \left(\frac{s}{b} + Ce^{-b\beta t}\right)^{1/\beta}. \tag{B.31}$$

If at time $t = 0$, the level of capital stock is, k_0, then we can determine the integration constant, C, as

$$C = k_0^{\beta} - \frac{s}{b}. \tag{B.32}$$

The analytic solution is therefore given by

$$k(t) = \left[\frac{s}{b} + \left(k_0^{\beta} - \frac{s}{b}\right)e^{-b\beta t}\right]^{1/\beta}. \tag{B.33}$$

This can be re-arranged as

$$k(t) = \left[\frac{s}{b}\left(1 - e^{-b\beta t}\right) + k_0^{\beta} e^{-b\beta t}\right]^{1/\beta}. \tag{B.34}$$

B.2 The Solow-Swan solution

Knowing the solution to the capital evolution equation means that we are now in a position to calculate the output per unit of effective labour from (B.8)

$$\boxed{y(t) = \left[\frac{s}{b}\left(1 - e^{-b(1-\alpha)t}\right) + k_0^{1-\alpha} e^{-b(1-\alpha)t}\right]^{\frac{\alpha}{1-\alpha}}.} \tag{B.35}$$

We can now solve for the actual output, lumping the two constants L_0 and A_0 into a single constant

$$Y(t) = C_0\, e^{(g+n)t}\left[\frac{s}{b}\left(1 - e^{-b(1-\alpha)t}\right) + k_0^{1-\alpha} e^{-b(1-\alpha)t}\right]^{\frac{\alpha}{1-\alpha}}. \tag{B.36}$$

If the initial condition is given by $Y_0 = Y(0)$, then

$$Y_0 = C_0\,\left(k_0^{1-\alpha}\right)^{\frac{\alpha}{1-\alpha}} = C_0 k_0^{\alpha}, \tag{B.37}$$

so that

$$C_0 = Y_0 k_0^{-\alpha}, \tag{B.38}$$

resulting in the solution to the Solow-Swan model

$$Y(t) = Y_0 k_0^{-\alpha}\, e^{(g+n)t}\left[\frac{s}{b}\left(1 - e^{-b(1-\alpha)t}\right) + k_0^{1-\alpha} e^{-b(1-\alpha)t}\right]^{\frac{\alpha}{1-\alpha}}. \tag{B.39}$$

If we re-introduce the original model parameters, then the solution for the economic output is finally given by

$$\boxed{Y(t) = Y_0 k_0^{-\alpha}\, e^{(g+n)t}\left[\frac{s}{\delta+g+n}\left(1 - e^{-(\delta+g+n)(1-\alpha)t}\right) + k_0^{1-\alpha} e^{-(\delta+g+n)(1-\alpha)t}\right]^{\frac{\alpha}{1-\alpha}}.} \tag{B.40}$$

For reference, the parameters in the Solow-Swan model are summarised in Table B.1. The explicit solution allows an easy fit to the data obtained in the agent-based simulations, described in §5.3.1, pg.77.

Table B.1: Summary of Solow-Swan parameters

Parameter	**Description**
α	Cobb-Douglas production function output elasticity for capital.
δ	Capital depreciation rate
s	Savings rate
g	Labour productivity growth rate
n	Labour growth rate
Y_0	Initial GDP
k_0	Initial capital per unit of effective labour

Design & implementation

This appendix provides details on the design and implementation of the model presented. Design concepts and terminology central to object-oriented design (OOD) and object-oriented programming (OOP) have been used (Booch et al. 2007, Hillar 2015).

C.1 Object-oriented design in a nutshell

Object-oriented approaches to software design and implementation has become one of the dominant software development paradigms since the 1980s (Booch et al. 2007). Most modern, general purpose programming languages (C++, Python, Java, C#, etc.) include features for object-orientation. The object model brings together many software analysis, modelling, design and implementation principles in a synergistic way (Booch et al. 2007). Object-oriented programming is a natural way to implement agent-based models (Boero et al. 2015). This section provides a concise overview of object-oriented design (OOD). The object-oriented paradigm incorporates several important characteristics, including data abstraction, inheritance, encapsulation, and polymorphism. These features are outlined in the sections that follow.

C.1.1 Abstraction, classes & objects

Objects are central to the object-oriented paradigm. Booch et al. (2007) defines an *object* as a "tangible entity that exhibits some well-defined behaviour." A particular firm in an economy would be an object, as would be each individuals, government, etc. One of the most important concepts in OOP involves the creation of *abstract data types* that are used to specify the nature of objects representing something from the real-world. In OOP, an abstract data type is defined by a *class*. Classes are used to abstract behaviour. A class is a template or blueprint for creating objects and incorporates internal state variables (attributes or fields), as well as the possible behaviours of the object (Hillar 2015). In other words, class contains a collection of variables

(attributes) together with functions (procedures, behaviours or actions) that can manipulate the variables. The functions or behaviours are often referred to as *methods*. An object is an *instance* of class, meaning that it is a particular implementation of the class.

These ideas are best illustrated by means of an example. Suppose we have a class called **Firm** (as represented in Figure C.1) that we use to model firms in our economy[1]. It serves as the template from which all firms used in the simulation are then created. Any object instance that is created based on this template would have the attributes, employees, product, cash, etc. as shown in the diagram. At an abstract level, these attributes are the 'things that define what it means to be a firm' in the model. An important part of the software engineering design process involves deciding on the relevant attributes that are needed to uniquely define the class.

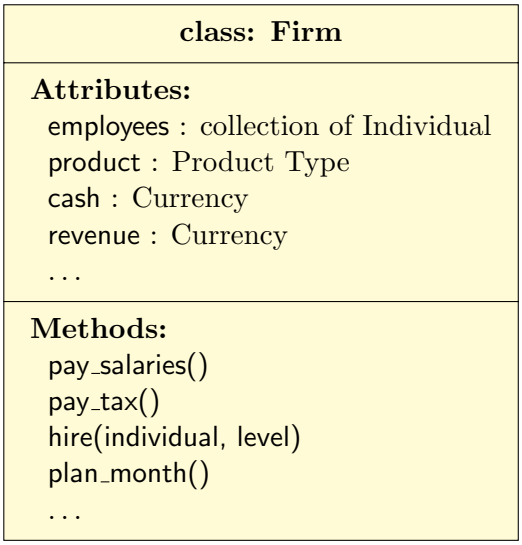


Figure C.1: An example definition for a class '**Firm**'.

A particular firm, say 'XYZ Manufacturers', would be an instance (an object) that is based on the template class **Firm**. When the object XYZ is created (or, using OOP terminology, instantiated), the attribute variables are initialized with initial values. So XYZ may be created with 15 employees, producing chairs, with starting cash of R 19,230.00, etc. All of these attributes are unique to the XYZ object.

C.1.2 Encapsulation

The paradigm shift in OOP is that the data and the methods to manipulate the data are tightly integrated within the class definition. The data attributes that define the properties of an object are hidden or *encapsulated* within the object. The value of the attribute should only be changed by the methods of the object itself.

In the example shown in Figure C.1, the collection employees is an attribute of a **Firm**. The employees variable of an firm should not be directly accessible by an outside entity. The hiring

[1]Note that this is not the complete class definition. The actual class definition is given in the software documentation in the following section.

or firing of staff is a purely internal decision within the firm and should not be accessed by external objects. Changing internal attributes should therefore be done via operations acting on the object - via methods. A firm such as 'XYZ Manufacturers' may only increase its employees when the hire() method is called. The hire(individual, level) method would then update the employees variable by adding the individual to the employees collection.

C.1.3 Inheritance

Another powerful feature of OOP is the ability to inherit all the features (both attributes and methods) from a more abstract class. The parent class is referred to as the *base* or *superclass*, while the child class is said to be a *subclass*. It is often useful to create a more general, abstract class, and then establish a relationship between the classes. Usually, one creates a '*is a*' relationship between two classes.

As an example that will be used in the development of our agent-based model, suppose we have already seen the generic, abstract **Firm** class. In the model, there are two types of firms: a consumer firm and a capital firm, as shown in Figure C.2. In terms of the language used, one would say that the class **Consumer Firm** 'is a' **Firm**. The class **Firm** is the base class, while **Consumer Firm** is the subclass. All of the attributes and methods defined by the class **Firm** are then automatically inherited by the **Consumer Firm** and **Capital Firm** classes, and don't need to be redefined. Therefore common features don't need to be implemented separately.

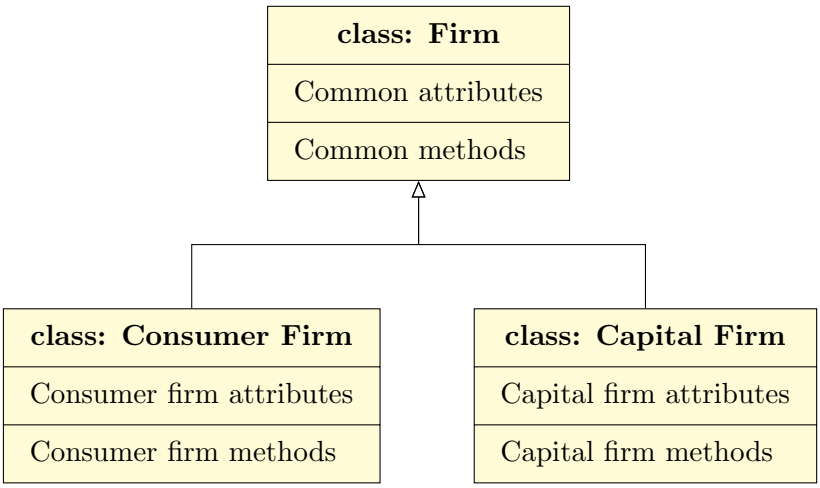


Figure C.2: An example of a class inheritance.

An important aspect of OOD is the design of the *class hierarchy*. By carefully considering and designing the class structures in a particular problem, a problem can be made considerably easier. The effort is in the design process and wherever possible, requires one to move functionality to higher levels of abstraction. In the model developed, we have an assortment of different types of objects that would interact: individuals, firms, government, etc. These various objects are related by the class hierarchy shown in Figure C.3. Inheritance therefore provides another level of abstraction in the OOD process.

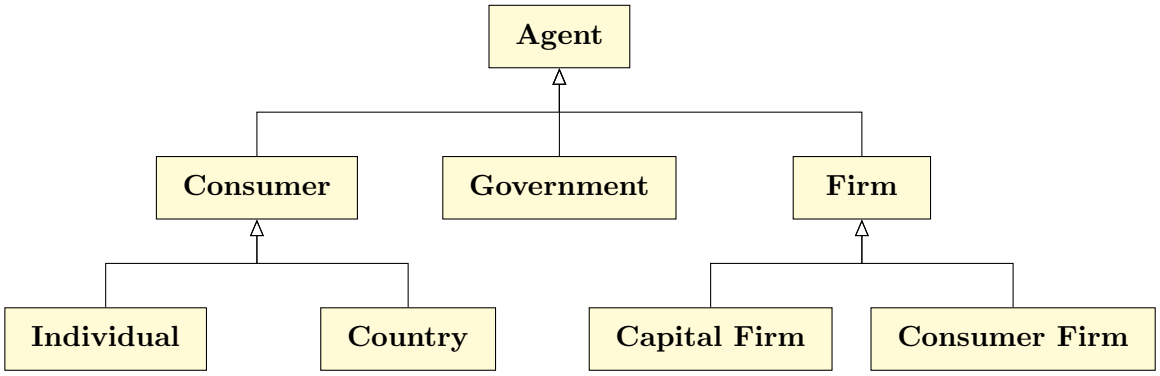


Figure C.3: The class diagram for the agents in the model.

C.1.4 Polymorphism

The last important characteristic of OOP that we will discuss, *polymorphism*, is a consequence of inheritance. The key idea of polymorphism is that various types of objects can have the same interface, provided by methods. What this means is that each class in a class hierarchy branch would have the same method name, but with the implementation being different for each subclass.

Consider the method plan_month() provided by the **Firm** class shown in Figure C.2. Each of the subclasses can override the base class implementation of the method, so that the **Consumer Firm** and **Capital Firm** classes would each have different implementations of the plan_month() method. The consumer firm method, cfirm•plan_month(), would do completely different things compared to the capital firm method, kfirm•plan_month(). The advantage of polymorphism is that if additional types of firms are added, the interfaces to the new firm classes would be completely consistent with the existing classes.

C.2 Simulation parameters

The simulation is initialized with parameters shown in the following tables.

Table C.1: Initial simulation parameter settings

Par.	Description	Value
$\hat{t}_{\max}$	Simulation duration	600 months
$\hat{N}_{\mathrm{MC}}$	Number of Monte Carlo runs	20
$\hat{\xi}$	Capital-labour investment probability. §4.3.5.3, pg.49	0.4, 0.35

Table C.2: Initial simulation parameter settings for the population or individuals

Par.	Description	Value
$\hat{N}_{\mathrm{P}}$	Initial population size	2,500
$\hat{g}_{\mathrm{P}}$	Population annual growth rate	1.5% p.a.
$\hat{g}_{\mathrm{B}}$	Population annual birth rate	2.75% p.a.
$\hat{r}_{\mathrm{U0}}$	Initial unemployment rate	25.0%
$\hat{\vec{r}}_{\mathrm{IS}}$	Individual savings rate (by education)	[2, 4, 8, 10] %
Progression Probability Parameters		
$\hat{p}_{\mathrm{E0}}$	Progression probability of individuals finishing school	0.62
$\hat{p}_{\mathrm{E1}}$	Progression probability of individuals getting matric	0.75
$\hat{p}_{\mathrm{E2}}$	Progression probability of individuals entering tertiary education	0.4
$\hat{p}_{\mathrm{E3}}$	Progression probability of individuals finishing tertiary education	0.75
$\hat{p}_{\mathrm{E4}}$	Progression probability of individuals entering graduate school	0.4
$\hat{p}_{\mathrm{E5}}$	Progression probability of individuals finishing graduate school	0.95
Default Status Consumption Parameters		
$\hat{\vec{\boldsymbol{d}}}_0$	Status consumption parameter - unskilled	[1000, 5]
$\hat{\vec{\boldsymbol{d}}}_1$	Status consumption parameter - matric	[1000, 15]
$\hat{\vec{\boldsymbol{d}}}_2$	Status consumption parameter - bachelors	[1000, 25]
$\hat{\vec{\boldsymbol{d}}}_3$	Status consumption parameter - postgrad	[1000, 50]
$\hat{\vec{\boldsymbol{d}}}_4$	Status consumption parameter - unemployed	[1000, 1]
$\hat{\vec{\boldsymbol{d}}}_5$	Status consumption parameter - retired	[1000, 2]

Table C.3: Initial simulation parameter settings for firms

Par.	Description	Value
$\hat{N}_{\mathrm{F}}$	Number of firms per 1000 population (by goods)	[6, 4]
$\hat{\vec{\boldsymbol{s}}}_{g\mathrm{F}}$	Firm savings rate (by goods)	[0.25, 0.75] %
$\hat{d}$	Capital depreciation rate per month	0.01 %/p.m.
Long-Term Adjustment Parameters		
$\hat{\vec{\boldsymbol{\delta}}}_0$	Stock-capacity ratio long-term labour trigger levels (%) - essential	[0.2, 7.0, 10.0, 20.0]
$\hat{\vec{\boldsymbol{\delta}}}_1$	Stock-capacity ratio long-term labour trigger levels (%) - luxury	[0.5, 5.0, 12.5, 17.0]
$\hat{\Delta\vec{\boldsymbol{L}}}_0$	Long-term labour changes - essential	[12, 7, 0, -6, -10]
$\hat{\Delta\vec{\boldsymbol{L}}}_1$	Long-term labour changes - luxury	[6, 4, 0, -3, -5]

Table C.4: Initial simulation parameter settings for markets

Par.	Description	Value
$\hat{P}_{\mathrm{G}}$	Base product prices (by goods)	R [10, 1200]
$\hat{P}_{\mathrm{L}}$	Base labour prices (by education)	R [12500, 25000, 60000, 120000]

Table C.5: Initial simulation parameter settings for government

Par.	Description	Value
$\hat{S}_{\mathrm{G}}$	Government grant amount	R 4,000 p.m.
$\hat{r}_{\mathrm{G}}$	Government tax rate	25.0%

Table C.6: Initial simulation parameter settings for the world

Par.	Description	Value
$\hat{N}_{\mathrm{X}}$	World size / number of countries	6
$\hat{g}_{\mathrm{X}}$	World annual growth rate	1.8% p.a.
$\hat{\boldsymbol{D}}_{\mathrm{X}}$	World default product basket quantities	[31000, 140]

C.3 Software implementation & documentation

The following pages provide the software documentation for the implementation of the model. For each of the classes used in the simulation, the class hierarchy is provided, together with the class attributes and methods. Where appropriate, information is provided on the algorithms or calculations performed. Note that the documentation has been auto-generated from comments within the source code using *doxygen* (`https://www.doxygen.nl/index.html`).

The implementation was inspired by various aspects of the Python ABM library, Mesa (Masad & Kazil 2015). Some of the main Python libraries that were used include:

- a generic simulation library **pypet**,
- a tree data structure library **treelib**,
- a mathematical library **numpy**,
- a scientific computing library **scipy** and
- a plotting library **matplotlib**.

The graphs were generated using the LaTeX PGFPlots package.

Economic Simulation

4

Generated by Doxygen 1.8.20

Chapter 1

Class Index

Here are the classes, structs, unions and interfaces with brief descriptions:

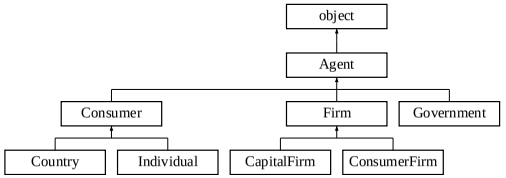
Chapter 2

Class Documentation

2.1 Agent Class Reference

The abstract base class from which all other agent classes are derived.

Inheritance diagram for Agent:



Public Member Functions

- def **__init__** (self, unique_id, parameters, capacity=0)
- def **capacity** (self)
- def **capacity** (self, value)

Public Attributes

- **unique_id**
- **name**
- **parameters**

2.1.1 Detailed Description

The abstract base class from which all other agent classes are derived.

All agents are sub-classes of the abstract class **Agent**. There are two types of agents that act as consumers **Individual** and **Country**. There are two types of firms: **CapitalFirm** and **ConsumerFirm**. This design would make it relatively simple to add new types of consumers or firms.

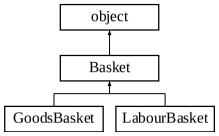
The documentation for this class was generated from the following file:

- ABM/AgentAbstract.py

2.2 Basket Class Reference

The abstract base basket class for various types of baskets used by the **Market** class.

Inheritance diagram for Basket:



Public Member Functions

- def __init__ (self, size, name='Basket', issupply=False, integer_items=False)
  Constructor method to initialize the basket.
- def is_empty (self)
  A property method that indicates if the basket has no items in it.
- def empty (self)
  A method to empty the current basket.
- def randomize (self, *args)
  Initialize a random quantity of items in the basket for testing.
- def normalize (self)
  A method to normalize the items in the basket so that the sum of the items equals 1.
- def sum (self)
  A method that return the total quantity of items in the basket.
- def __getitem__ (self, i)
  A Dunder or 'magic' method that is used to reference a basket like a list.
- def __setitem__ (self, idx, value)
  A Dunder or 'magic' method that is used to set a basket item in the same way as a list.
- def __add__ (self, other_basket)
  A Dunder or 'magic' method that allows the addition of baskets.
- def __sub__ (self, other_basket)
  A Dunder or 'magic' method that is allows the subtraction of baskets.
- def __div__ (self, other_basket)
  A Dunder or 'magic' method that is allows the division of baskets.

- def __truediv__ (self, other_basket)
  A Dunder or 'magic' method that is allows the division of baskets.
- def __iadd__ (self, other)
  A Dunder or 'magic' method that allows the in place addition of baskets.
- def **value** (self)
- def __str__ (self)
  A Dunder or 'magic' method used for printing the items of a basket.

Public Attributes

- **name**
- **size**
- **items**
- **integer_items**
- **is_supply**

2.2.1 Detailed Description

The abstract base basket class for various types of baskets used by the **Market** class.

A basket is a list of 'products' (either from the GoodsConstants or EducationConstants sets) that a buyer agent will use to store its needs. Each buyer agent will have a basket of 'products' that they already possess and abasket of 'products' that they need.

There are two types of baskets used that are sub-classes of this base class.

- a GoodsBasket used for products in the ProductMarket; and
- a LabourBasket used for products in the "LabourMarket".
- a LabourBasket used for products in the LabourMarket.
- a LabourBasket used for products in the LabourMarket .

2.2.2 Constructor & Destructor Documentation

2.2.2.1 __init__()

```
def __init__ (
            self,
            size,
            name = 'Basket',
            issupply = False,
            integer_items = False )
```

Constructor method to initialize the basket.

Each basket consists of a list of items. It is possible to have integer or real-valued items in a basket. The class provides a set of Dunder methods to allow treating a basket as a list.

Reimplemented in LabourBasket, and GoodsBasket.

2.2.3 Member Function Documentation

2.2.3.1 __add__()

```
def __add__ (
              self,
              other_basket )
```

A Dunder or 'magic' method that allows the addition of baskets.

For instances, **basket1 = Basket(N)** and **basket2 = Basket(N)**, one can add the two basket using

$$\textbf{basket3} = \textbf{basket1} + \textbf{basket2}.$$

2.2.3.2 __div__()

```
def __div__ (
              self,
              other_basket )
```

A Dunder or 'magic' method that is allows the division of baskets.

For instances, **basket1 = Basket(N)** and **basket2 = Basket(N)**, one can divide the two basket using

$$\textbf{basket3} = \textbf{basket1}/\textbf{basket2}.$$

2.2.3.3 __getitem__()

```
def __getitem__ (
              self,
              i )
```

A Dunder or 'magic' method that is used to reference a basket like a list.

For an instance, **basket = Basket(N)**, one can obtain the ith element in the basket using **basket[i]** rather than **basket.↩ items[i]**.

2.2.3.4 __iadd__()

```
def __iadd__ (
              self,
              other )
```

A Dunder or 'magic' method that allows the in place addition of baskets.

For instances: **basket1 = Basket(N)** and **basket2 = Basket(N)**, one can write

basket1+=**basket2**.

The method will add the items of **basket2** to **basket1**, updating the values of **basket1**.

2.2.3.5 __setitem__()

```
def __setitem__ (
              self,
              idx,
              value )
```

A Dunder or 'magic' method that is used to set a basket item in the same way as a list.

For an instance, **basket = Basket(N)**, one can set the ith element in the basket using **basket[i] = value**.

2.2.3.6 __sub__()

```
def __sub__ (
              self,
              other_basket )
```

A Dunder or 'magic' method that is allows the subtraction of baskets.

For instances, **basket1 = Basket(N)** and **basket2 = Basket(N)**, one can subtract the two basket using

$$\textbf{basket3} = \textbf{basket1} - \textbf{basket2}.$$

Any negative values are set to zero since it is assumed that only positive quantities are allowed.

2.2.3.7 __truediv__()

```
def __truediv__ (
              self,
              other_basket )
```

A Dunder or 'magic' method that is allows the division of baskets.

For instances, **basket1 = Basket(N)** and **basket2 = Basket(N)**, one can divide the two basket using

$$\textbf{basket3} = \textbf{basket1}/\textbf{basket2}.$$

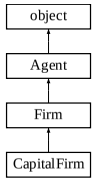
The documentation for this class was generated from the following file:

- ABM/MarketBasketModel.py

2.3 CapitalFirm Class Reference

This is the class for capital firms (or K-firms).

Inheritance diagram for CapitalFirm:



Public Member Functions

- def **__init__** (self, unique_id, parameters, initial_labour)
- def pay (self, amount)

  Increase revenue by the amount.
- def plan_month (self)

  Capital firm monthly plannig.
- def activity (self)

  The activity attribute is not used by firms.
- def **save** (self)
- def force_buy (self, level, individual)

  A method used to force the hiring of labour during initialisation.
- def reset_requirements (self, investments)

  A method to reset the needs and requirements after initialisation and creation of firms.

Public Attributes

- **total_salary**
- **need_to_buy**
- **possess**
- **basket**
- **purchase_prices**
- **wealth**
- **cash**
- **revenue**
- **propensity_to_save**
- **knowledge**
- **technological_sophistication**
- **requirements**
- **revenue_history**
- **count_change_labour**

Static Public Attributes

- string **TYPE** = 'capital'

2.3.1 Detailed Description

This is the class for capital firms (or K-firms).

Capital firms provide capital equipment to consumer firms.

2.3.2 Member Function Documentation

2.3.2.1 activity()

```
def activity (
            self )
```

The activity attribute is not used by firms.

Reimplemented from Firm.

2.3.2.2 plan_month()

```
def plan_month (
            self )
```

Capital firm monthly plannig.

The only requirement for a capital firm is to plan its labour requirements. The labour requirements based on historical revenues. A forecast salary budget is calculated and then based on the average labour costs, the number of employees is calculated. More sophisticated forecasting methods could pssibly be used.

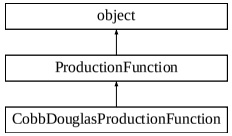
Reimplemented from Firm.

The documentation for this class was generated from the following file:

- ABM/AgentModelFirm.py

2.4 CobbDouglasProductionFunction Class Reference

Inheritance diagram for CobbDouglasProductionFunction:



Public Member Functions

- def **__init__** (self, product, knowledge)
- def **capital_productivity** (self, knowledge)
- def **productivities** (self, knowledge)
- def **set_initial_capacity** (self, initial_capacity, initial_labour, initial_knowledge)
- def **adjust_scale** (self, factor)
- def **__call__** (self, K, L, W)

Public Attributes

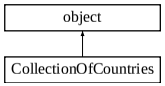
- **product_index**
- **alpha**
- **capital_rate**
- **productivities_MAX**
- **productivities_MIN**
- **base_capital_productivity**
- **base_knowledge**
- **C**

The documentation for this class was generated from the following file:

- ABM/ProductionFunctionModels.py

2.5 CollectionOfCountries Class Reference

Inheritance diagram for CollectionOfCountries:



Public Member Functions

- def **__init__** (self, parameters, population_scale_factor)
- def **initialize_tree** (self)
- def **add_country** (self)
- def **__iter__** (self)
- def **calculate_needs** (self, step)
- def **total_exports** (self)
- def **array_volume** (self)
- def **array_needs** (self)
- def **array_consumption** (self)

Public Attributes

- **number_countries**
- **population_scale_factor**

Static Public Attributes

- **agent** = None
- **parameters** = None

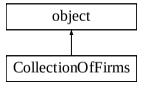
The documentation for this class was generated from the following file:

- ABM/AbstractCollectionCountries.py

2.6 CollectionOfFirms Class Reference

The collection of all firms in the economy.

Inheritance diagram for CollectionOfFirms:



Public Member Functions

- def **__init__** (self, parameters, population_scale_factor, labour, estimated_demand)
- def **tree** (self)
- def **subtree** (self, nid)
- def __iter__ (self, sector=None)

  Iterator method that return all firms in a sector.
- def calculate_needs (self)

  Method that instructs all firms to calculate their needs.
- def adjust_production_capacity (self, step, month, year)

  Method that iterates through all consumer firms to adjust their production capacity.
- def initialize_tree (self)

  Method to intialise the firm tree.
- def create_firms (self, population_scale_factor, labour, estimated_demand)

  The method used to create all the firms in the economy.
- def size (self)

  Property method that returns the total number of firms in the tree.
- def record_history (self)

  Method that insrtucts each firm to record history.
- def pay_taxes (self, government)

  Method that iterated through all firms and instructs them to pay taxes to government.
- def determine_savings (self)

  Method that iterates through all consumer firms and instructs them to save.
- def depreciate_capital (self)

  Method that iterated through all consumer firms and instructs them to depreciate capital.
- def plan_month_and_pay_salaries (self)

  Method that iterates through all firms and instructs them to pay salaries and plan their next month.

Public Attributes

- **parameters**
- **annual_revenue_history**
- **annual_salary_history**
- **agent**
- **firm_counts**
- **capital_firm**
- **salaries_paid**

2.6.1 Detailed Description

The collection of all firms in the economy.

The collection of firms is stored as a Tree (from the treelib library). The tree is split into two branches for capital and consumer firms. For simplicity, a single capital firm is used (hence there is no need to implement a capital market).

2.6.2 Member Function Documentation

2.6.2.1 __iter__()

```
def __iter__ (
              self,
              sector = None )
```

Iterator method that return all firms in a sector.

This method allows the iteration of particular firm sector.

2.6.2.2 create_firms()

```
def create_firms (
              self,
              population_scale_factor,
              labour,
              estimated_demand )
```

The method used to create all the firms in the economy.

Market demand is estimated based on the intial population and export demand. Estimates the required labour and production capacity for each firm in order to satisfy demand. Labour fromt the population that were assigned as initially employed are assigned to firms.

2.6.2.3 initialize_tree()

```
def initialize_tree (
              self )
```

Method to intialise the firm tree.

Firms are split into Consumer and Capital firms. The Consumer firms are then split into brances based on the products they sell.

2.6.2.4 plan_month_and_pay_salaries()

```
def plan_month_and_pay_salaries (
              self )
```

Method that iterates through all firms and instructs them to pay salaries and plan their next month.

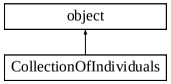
The documentation for this class was generated from the following file:

- ABM/AbstractCollectionFirms.py

2.7 CollectionOfIndividuals Class Reference

The collection class for the population of individual agents.

Inheritance diagram for CollectionOfIndividuals:



Public Member Functions

- def __init__ (self, parameters)
  Constructor to create the population of individual agents.
- def **tree** (self)
- def **subtree** (self, nid)
- def reset_cash (self)
  Resets the cash of all individuals in the population.
- def reallocate_status (self)
  Method to reallocate the status od all the individual agents in the population.
- def add_person (self, age=None)
  Method to create a new Individual agent, and assign it within the tree.
- def initialize_agents (self, parameters)
  Method to initialise the population tree, The population tree is created with the various categorisation nodes.
- def calculate_annual_population_changes (self)

Method that is called at the beginning of the year to determine the changes to the population.

- def monthly_population_adjustment (self, month)

  Method that is called monthly to adjust the population counts.
- def kill_individual (self, person)

  The method to kill an individual.
- def determine_savings (self)

  Method for all active members of the population to save.
- def pay_pensions (self)

  A method for all retired memeber of the poulation to obtain pension.
- def size (self)

  Returns the size of the population.
- def __getitem__ (self, status)

  Get method that returns a list all individuals based on the status.
- def __iter__ (self, status=None)

  Iterator method that return all individuals based on the status.
- def __getattr__ (self, name)

  Method to retriece a list of an attribute from all individuals.
- def calculate_needs (self, market_prices)

  Method used by the Market shopping calgorithm for all individuals to determine their needs.
- def count (self, status=None)

  Method to return the number of individuals in the population with a given status.
- def count_employed (self)

  Property method that returns the number of employed individuals.
- def count_unemployed (self)

  Property method that returns the number of unemployed individuals.
- def count_active (self)

  Property method that returns the number of active individuals.
- def rate_unemployment (self)

  Property method that calculated the unemployment rate.

Public Attributes

- **initial_numbers_to_employ**
- **birth_array**
- **geriatric_array**
- **death_array**
- **change_array**
- **possible_fatalities**

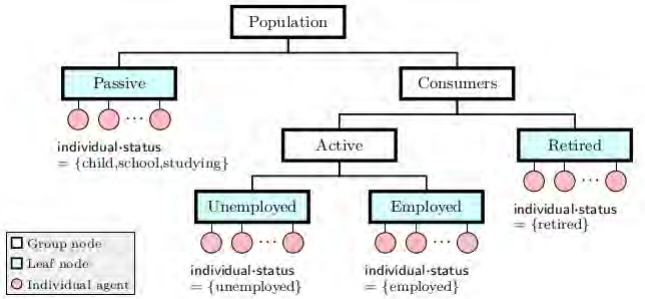
Static Public Attributes

- **agent** = None
- **parameters** = None

2.7.1 Detailed Description

The collection class for the population of individual agents.

All agents are stored as a tree structure, based on categories as shown:



2.7.2 Constructor & Destructor Documentation

2.7.2.1 __init__()

```
def __init__ (
            self,
            parameters )
```

Constructor to create the population of individual agents.

Calls the initialize_agents() method.

2.7.3 Member Function Documentation

2.7.3.1 __iter__()

```
def __iter__ (
            self,
            status = None )
```

Iterator method that return all individuals based on the status.

This method allows the iteration of particular population sectors.

2.7.3.2 calculate_annual_population_changes()

```
def calculate_annual_population_changes (
            self )
```

Method that is called at the beginning of the year to determine the changes to the population.

Population changes are determined based on the simulation parameters for birth rate and population growth rate. A half-normal target distribution is used for the population distribution of the new count. A death array for all the ages is obtained as the difference between the actual ditribution and the desired new distribution.

- Produces a births array that contains monthly number of individuals to be created for each month.
- Produces a death array that contains the number of individuals that need to die each month.
- Produces an array of randomly selected names of individuals (possible fatalities) that fall within the ages that could possibly be killed.

2.7.3.3 calculate_needs()

```
def calculate_needs (
            self,
            market_prices )
```

Method used by the Market shopping calgorithm for all individuals to determine their needs.

For all consumer members of the population, the individual.calculate_needs method is called.

2.7.3.4 count_active()

```
def count_active (
            self )
```

Property method that returns the number of active individuals.

2.7.3.5 count_unemployed()

```
def count_unemployed (
            self )
```

Property method that returns the number of unemployed individuals.

2.7.3.6 determine_savings()

```
def determine_savings (
            self )
```

Method for all active members of the population to save.

Calls the individual.save method for all active members of the population.

2.7.3.7 initialize_agents()

```
def initialize_agents (
            self,
            parameters )
```

Method to initialise the population tree, The population tree is created with the various categorisation nodes.

- Based on the simulation parameter for population size, new individual agents are created by calling the add_person() method.
- Using the simulation parameter for unemployment, employable (active) members of the population are then assigned the status of employed. This is a potential employement, since in the firm initialisation, firms will attempt to employ those individuals that are allocated as employed.
- The population tree is then corrected by calling the reallocate_status method.

2.7.3.8 kill_individual()

```
def kill_individual (
            self,
            person )
```

The method to kill an individual.

When killing an individual, the wealth of the individual is distributed between 1-4 other individuals who inherit their accumulated wealth. The individual is then removed from the population tree.

2.7.3.9 monthly_population_adjustment()

```
def monthly_population_adjustment (
            self,
            month )
```

Method that is called monthly to adjust the population counts.

Individuals ae aged a year, depening on their birthday by calling the individual.age_a_year() method.

- Based on the new births in the births_array, new individual agents are "born".
- If individual reacheds the maximum age, they are killed by calling the kill_individual() method.
- The number, N of individuals to be killed in the month is calculated.
- A randomly selected list of N individuals from the list possible_fatalities is chosen and then killed.

2.7.3.10 pay_pensions()

```
def pay_pensions (
             self )
```

A method for all retired memeber of the poulation to obtain pension.

Calls the individual.get_pension method for all retired members of the population.

2.7.3.11 rate_unemployment()

```
def rate_unemployment (
             self )
```

Property method that calculated the unemployment rate.

Uses the count_employed and count_unemployed methods.

2.7.3.12 reallocate_status()

```
def reallocate_status (
             self )
```

Method to reallocate the status od all the individual agents in the population.

The method iterates through all the individuals in the tree, and then based on the individual.status variable, moves the node within the population tree.

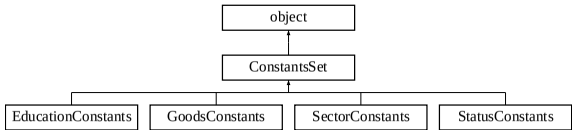
The documentation for this class was generated from the following file:

- ABM/AbstractCollectionIndividuals.py

2.8 ConstantsSet Class Reference

The class **ConstantsSet** provides an abstract container class for several structured constants used in the simulation.

Inheritance diagram for ConstantsSet:



Public Member Functions

- def __init__ (self)
  Initialize the set.
- def __getitem__ (self, code)
  Returns the value of the constant.
- def __iter__ (self)
  An iterator over all the values in the set.
- def size (self)
  The number of constants in the set.
- def str (self, index)
  Returns the string value of the index.
- def list (self)
  Returns a list of all the values of the constants in the set.
- def value (self, thename)
  Returns the value corresponding to a string description of the constant.
- def random (self)
  Returns a random value from the set.

Public Attributes

- **N**

2.8.1 Detailed Description

The class **ConstantsSet** provides an abstract container class for several structured constants used in the simulation.

The class provides a way to contain a set of constants. For example, all the status values that are used in the economy are implemented as a subclass of the **ConstantsSet** class. The abstract class provides a consistent framework for accessing the values and properties of the constants. Each subclass must implement the following attributes:

Attribute	Data Type	Description
DESCRIPTION	**dictionary**	A Python key-value pair dictionary
RANGE	array of **integer**	The range of possible values allowed to index the constant
INDEX	**dictionary**	A Python key-value pair dictionary returning the constant index

These various classes provide implementations of the **Goods** set, $\mathcal{G}$; the **Status** set, $\mathcal{S}$; and the **Education** set, $\mathcal{E}$.

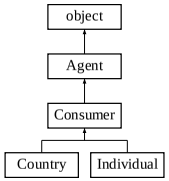
The documentation for this class was generated from the following file:

- Constants.py

2.9 Consumer Class Reference

An abstract consumber agent class from which all consumer classes are derived.

Inheritance diagram for Consumer:



Public Member Functions

- def **__init__** (self, unique_id, parameters, capacity=0)

Additional Inherited Members

2.9.1 Detailed Description

An abstract consumber agent class from which all consumer classes are derived.

All consumer agents are sub-classes of the abstract class **Consumer**.

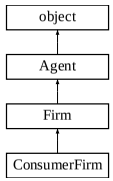
The documentation for this class was generated from the following file:

- ABM/AgentAbstract.py

2.10 ConsumerFirm Class Reference

This is the class for consumer firms (or C-firms).

Inheritance diagram for ConsumerFirm:



Public Member Functions

- def product_name (self)
  A property method that returns the name of the product sold by the firm.
- def __init__ (self, unique_id, parameters, product, initial_capacity, initial_labour)
  Constructor method to create a consumer firm.
- def initialize_baskets (self)
  Method to initialize the various baskets for the firm.
- def reset_requirements (self, investments)
  A method to reset the needs and requirements after initialisation and creation of firms.
- def save (self)
  Method the save a percentage of cash and add to the wealth.
- def **set_capital** (self, capital)
- def add_employee (self)
  Method to add an employee the the firm.
- def activity (self)
  Method that returns the activity index to determine how active the firm is in number of trips to the souk.
- def adjust_pricing (self, market_price)
  Adjust the firms product pricing.
- def mean_stock_history (self)
  Method to return the moving average of the stock.
- def mean_capacity_history (self)
  Method to return the moving average of the production capacity.
- def record_history (self)
  Method to recods the stock and stock capacity into the FIFO queues.
- def depreciate_capital (self)
  Method the depreciate the firms capital based on the depreeciation parameter.
- def invest (self, firm)
  Method to invest from the budgetes investment queue.
- def invest_knowledge (self, percent, term, labour)
  Method used to calculate and budget the invest in knowledge.
- def invest_capital (self, percent, term, labour)
  Method used to calculate and budget the invest in capital.
- def move_wealth (self, amount)
  A method to retriev cash from savings.
- def adjust_capacity (self, step, month, year)
  The method used to adjust production capacity by changing the factors of production.
- def long_term_adjustment (self)
  The long-term production adjustment method.
- def plan_month (self)
  Method called at the beginning of each month.
- def labour (self)
  Property method that returns the labour composition of the firm.
- def labour_prodctivities (self)
  Property method that returns the labour productivities.
- def capacity (self)
  Property method that returns the capacity of the firm.
- def produce (self)
  Method to calculate the required production for the month.
- def sells (self, product, buyer, quantity_bought)
  Method used by the product market when selling the product.
- def able_to_supply (self, quantity)
  Method used by the product market if the firm is able to supply the quantity of the product.

- def available (self)

  Property method that is true if the firm has available stock to sell.
- def produces (self, product)

  Returns true of the firm sells the product.

Public Attributes

- **wealth**
- **cash**
- **revenue**
- **price_adjustment_percent**
- **product**
- **item**
- **knowledge**
- **technological_sophistication**
- **tech_growth_time_constant**
- **investment_queue_capital**
- **investment_queue_knowledge**
- **investment_queue_labour**
- **mass_retrenchment_step**
- **propensity_to_save**
- **investment**
- **investment_capital**
- **investment_knowledge**
- **production**
- **stock**
- **sales_price**
- **price_adjustment_frequency_parameter**
- **adjustment_month**
- **adjustment_year**
- **CAPITAL_MAX**
- **CAPITAL_MIN**
- **KNOWLEDGE_MAX**
- **KNOWLEDGE_MIN**
- **NoMonthsHistory**
- **market_consumption_history**
- **market_capacity_history**
- **capacity_long_term_history**
- **stock_long_term_history**
- **capacity_history**
- **stock_history**
- **revenue_history**
- **salary_history**
- **capital_responsiveness**
- **knowledge_responsiveness**
- **initial_capacity**
- **initial_labour_count**
- **count_change_capital**
- **count_change_labour**
- **count_change_knowledge**
- **f**
- **requirements**
- **capital**
- **possess**
- **need_to_buy**

- **basket**
- **purchase_prices**
- **market_price_ratio**
- **delta**
- **DELTA**
- **volume**

Static Public Attributes

- string **TYPE** = 'consumer'

2.10.1 Detailed Description

This is the class for consumer firms (or C-firms).

Consumer firms provide consumer goods to the various types of consumer agents.

2.10.2 Constructor & Destructor Documentation

2.10.2.1 __init__()

```
def __init__ (
            self,
            unique_id,
            parameters,
            product,
            initial_capacity,
            initial_labour )
```

Constructor method to create a consumer firm.

The method initializes all the internal state variables for the firm.

2.10.3 Member Function Documentation

2.10.3.1 able_to_supply()

```
def able_to_supply (
            self,
            quantity )
```

Method used by the product market if the firm is able to supply the quantity of the product.

Returns the quantity of the proeduct that the firm is able to supply.

2.10.3.2 activity()

```
def activity (
              self )
```

Method that returns the activity index to determine how active the firm is in number of trips to the souk.

Reimplemented from Firm.

2.10.3.3 adjust_capacity()

```
def adjust_capacity (
              self,
              step,
              month,
              year )
```

The method used to adjust production capacity by changing the factors of production.

There are two possible adjustments that can take place:

- a short-term adjustment (short_term_adjustment()) which is evaluated monthly, and
- a long-term adjustment (long_term_adjustment()) which will take place every few years.
  This method will call the appropriate fuction based on the timing when the appropriate adjutment is required.

2.10.3.4 adjust_pricing()

```
def adjust_pricing (
              self,
              market_price )
```

Adjust the firms product pricing.

The product pricing is determined using the PricingAdjustment function based on:

- The moving average stock capacity ratio, delta, and
- The market-price ratio.

2.10.3.5 initialize_baskets()

```
def initialize_baskets (
              self )
```

Method to initialize the various baskets for the firm.

Initializes the following LabourBasket baskets:

- The possess basket
- The requirements basket
- The needs basket.
- The purchase prices basket of labour costs for each level.

2.10.3.6 invest()

```
def invest (
              self,
              firm )
```

Method to invest from the budgetes investment queue.

Capital investments are paid to the K-firm. Knowledge investment is a reallocation to the knowlede asset. Technological sophistication is increased when investing.

2.10.3.7 invest_capital()

```
def invest_capital (
              self,
              percent,
              term,
              labour )
```

Method used to calculate and budget the invest in capital.

2.10.3.8 long_term_adjustment()

```
def long_term_adjustment (
              self )
```

The long-term production adjustment method.

Based on the ξ parameter, the investment is made in either knowledge, capital or labour.

2.10.3.9 plan_month()

```
def plan_month (
            self )
```

Method called at the beginning of each month.

Resets the monthly revenue, volumes and investment.

Reimplemented from Firm.

2.10.3.10 produce()

```
def produce (
            self )
```

Method to calculate the required production for the month.

Actual production is the difference between capacity and available stock.

2.10.3.11 reset_requirements()

```
def reset_requirements (
            self,
            investments )
```

A method to reset the needs and requirements after initialisation and creation of firms.

The method re-adjusts the parameters of the firms initial production function.

2.10.3.12 sells()

```
def sells (
            self,
            product,
            buyer,
            quantity_bought )
```

Method used by the product market when selling the product.

When selling the quantify of the product,

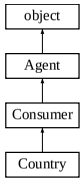
- stock levels are adjusted
- revenue and cash is increased by the sales amount
- sales volume for the month is increased.

The documentation for this class was generated from the following file:

- ABM/AgentModelFirm.py

2.11 Country Class Reference

Inheritance diagram for Country:



Public Member Functions

- def **__init__** (self, unique_id, parameters)
- def **has_money** (self, cheapest_price=0)
- def **activity** (self)
- def **preferences** (self)
- def **needs** (self, product)
- def **wants_to_buy** (self, product)
- def **buys** (self, product, firm, quantity, show_details=False)
- def **calculate_needs** (self, step)
- def **total_consumption** (self)

Public Attributes

- **size**
- **growth_rate**
- **frequency**
- **need_to_buy**
- **basket**
- **consumption**
- **purchase_prices**

The documentation for this class was generated from the following file:

- ABM/AgentModelCountry.py

2.12 Economy Class Reference

The **Economy** class is the class that comprises the elements of the agent-based model.

Inheritance diagram for Economy:

Public Member Functions

- def __init__ (self, trajectory)

  Constructor to initialize the economy.
- def initialize_population (self)

  *Initialize the population as a collection of individuals using the **CollectionOfIndividuals** class.*
- def initialize_world (self)

  *Initialize the world or foreign consumers as a collection of countries using the **CollectionOfCountries** class.*
- def initialize_government (self)

  *Initialize the government agent using the **Government** class.*
- def initialise_firms (self)

  *Initialize the business sector as a collection of firms using the **CollectionOfFirms** class.*
- def initialise_product_market (self)

  Initialize the product market.
- def initialise_foreign_market (self)

  Initialize the foreign market.
- def initialise_labour_market (self)

  Initialize the labour market.
- def step_beginning_of_year (self, step, month, year)

  Activities performed at the beginning of the year.
- def step_beginning_of_month (self, step, month, year)

  Activities performed at the beginning of each month.
- def step_during_month (self, step, month, year)

  Activities performed during each month.
- def step_end_of_month (self, step, month, year)

  Activities performed at the beginning of each month.

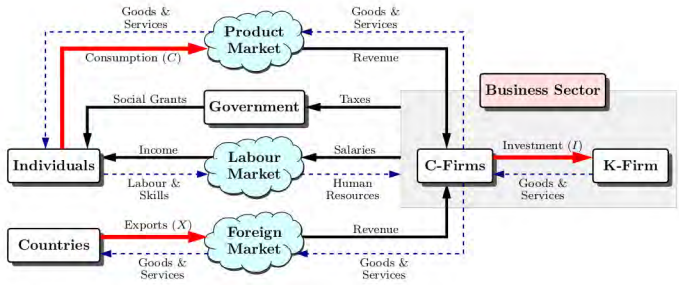
Public Attributes

- **trajectory**
- **data**
- **population**
- **world**
- **government**
- **industry**
- **product_market**
- **foreign_market**
- **labour_market**
- **total_tax**

2.12.1 Detailed Description

The **Economy** class is the class that comprises the elements of the agent-based model.

The economy is comprised of collections of various types of agents and markets through which they interact.



The agent objects in the economy are:

- **population:** A collection of **individual** agents. Type: CollectionOfIndividuals.
- **industry:** A collection of both **consumer** and **capital** firms in the economy. Type: CollectionOfFirms.
- **world:** A collection of **country** agents comprising the foreign market. Type: CollectionOfCountries.
- **government:** The **government** agent. Type: Government.

There are three types of market objects:

- **product_market:** A market for goods and services. Type: ProductMarket
- **foreign_market:** A market for goods and services. Type: ForeignMarket
- **labour_market:** The market for labour. Type: LabourMarket

In addition, there are two other important objets:

- **trajectory:** The pypet simulation trajectory containing all simulation parameters.
- **data:** The data collection object where all relevant simulation variables are recorded for subsequent analysis.

2.12.2 Constructor & Destructor Documentation

2.12.2.1 __init__()

```
def __init__ (
              self,
              trajectory )
```

Constructor to initialize the economy.

The economy is created by the following steps:

- Initialize the population. **initialize_population()**
- Initialize the foreign market. **initialize_world()**
- Estimate export sector demand. **world.calculate_needs(0)**
- Initialize the firms. **initialise_firms()**
- Initialize the government agent. **initialize_government()**
- Estimate the initial demand of the population. **population.calculate_needs()**
- Initialize the various markets in the economy:
  - initialise_product_market()
  - initialise_foreign_market()
  - initialise_labour_market()

2.12.3 Member Function Documentation

2.12.3.1 initialise_firms()

```
def initialise_firms (
              self )
```

Initialize the business sector as a collection of firms using the **CollectionOfFirms** class.

The population and world needs (ie. demand) is estimated and supplied to the **CollectionOfFirms** class. The **CollectionOfFirms** class is also supplied with the population subtree of the employed population (desired) so that they can be allocated to firms. Since not all individuals will be employed, the population status is corrected using **population.reallocate_status()**.

2.12.3.2 initialise_foreign_market()

```
def initialise_foreign_market (
              self )
```

Initialize the foreign market.

After initialising the foreign market, with the world tree as the buyers and C-firms as the sellers in the market. After creating the market, the capacity and prices are initialised

2.12.3.3 initialise_labour_market()

```
def initialise_labour_market (
              self )
```

Initialize the labour market.

After initialising the labour market, with the collection of firms as the buyers and the unemployed population subtree as the sellers in the market. After creating the market, the capacity and prices are initialised

2.12.3.4 initialise_product_market()

```
def initialise_product_market (
              self )
```

Initialize the product market.

After initialising the product market, with the consumer population subtree as the buyers and C-firms as the sellers in the market. After creating the market, the capacity and prices are initialised.

2.12.3.5 initialize_government()

```
def initialize_government (
              self )
```

Initialize the government agent using the **Government** class.

2.12.3.6 initialize_world()

```
def initialize_world (
              self )
```

Initialize the world or foreign consumers as a collection of countries using the **CollectionOfCountries** class.

The number of countries in the world will scale with the size of the population.

2.12.3.7 step_beginning_of_month()

```
def step_beginning_of_month (
            self,
            step,
            month,
            year )
```

Activities performed at the beginning of each month.

At the beginning of each month, the following activities will take place:

- The population count is adjusted (including births and deaths.)
- The government updates its census with the active_population and the grant recipients.
- The product market is updated with the latest consumer buyers subtree (since individuals may have died).
- Pensions are paid to the retired individuals in the population.
- Firms will reset production/revenues and pay salaries to employees.
- Government plays social grant to the recipients.
- The population will save a proportion of income.
- The demand of the product and foreign market is determined.
- The capacity and production needs for the month in the product market is evaluated.

2.12.3.8 step_beginning_of_year()

```
def step_beginning_of_year (
            self,
            step,
            month,
            year )
```

Activities performed at the beginning of the year.

At the beginning of each year, the following activities will take place:

- The population changes are calculated. Performed by the method: **population.calculate_annual_population↩ _changes()**.
- The government plans the year, which involves recalculating the tax rate and resetting the tax income.

2.12.3.9 step_during_month()

```
def step_during_month (
            self,
            step,
            month,
            year )
```

Activities performed during each month.

During each month, the following activities will take place:

- Consumers purchase goods from the product market.
- Countried purchase goods from the foreign market.

2.12.3.10 step_end_of_month()

```
def step_end_of_month (
            self,
            step,
            month,
            year )
```

Activities performed at the beginning of each month.

At the beginning of each month, the following activities will take place:

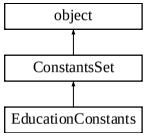
The documentation for this class was generated from the following file:

- EconomicModel.py

2.13 EducationConstants Class Reference

The constant set of all levels of education of individuals in the economy.

Inheritance diagram for EducationConstants:



Static Public Attributes

- int **NONE** = 30
- int **MATRIC** = 31
- int **BACHELORS** = 32
- int **POSTGRAD** = 33
- list **RANGE** = [30,31,32,33]
- string **NAME** = 'Education'
- dictionary **DESCRIPTION**
- dictionary **INDEX**

Additional Inherited Members

2.13.1 Detailed Description

The constant set of all levels of education of individuals in the economy.

The constant **EDUCATION** is provided as the structured constant instance for $\mathcal{E}$.

2.13.2 Member Data Documentation

2.13.2.1 DESCRIPTION

```
dictionary DESCRIPTION  [static]
```

Initial value:

```
=    {NONE:      'Unskilled',
                 MATRIC:     'Matric',
                 BACHELORS:  'Bachelors',
                 POSTGRAD:   'Postgrad'
                 }
```

2.13.2.2 INDEX

```
dictionary INDEX  [static]
```

Initial value:

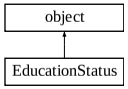
```
=    {NONE:      0,
                 MATRIC:     1,
                 BACHELORS:  2,
                 POSTGRAD:   3
                 }
```

The documentation for this class was generated from the following file:

- Constants.py

2.14 EducationStatus Class Reference

Inheritance diagram for EducationStatus:



Public Member Functions

- def **__init__** (self)
- def **index** (self)
- def **set** (self, newvalue)
- def **__str__** (self)
- def **HasNone** (self)
- def **HasMatric** (self)
- def **HasBachelors** (self)
- def **HasPostgrad** (self)

Static Public Attributes

- **value** = None

The documentation for this class was generated from the following file:

- ABM/IndividualStatusModel.py

2.15 Employee Class Reference

A simple class containing the details of an employee employed by a firm.

Inheritance diagram for Employee:

Public Member Functions

- def **__init__** (self, individual, salary)

Public Attributes

- **individual**
- **salary**

2.15.1 Detailed Description

A simple class containing the details of an employee employed by a firm.

The documentation for this class was generated from the following file:

- ABM/AgentModelFirm.py

2.16 FIFO Class Reference

An extension of the Queue class from the queue package.

Inheritance diagram for FIFO:

Public Member Functions

- def **put** (self, item, block=True, timeout=None)
- def **__getitem__** (self, n)
- def **__getattr__** (self, name)
- def **mean** (self, name)
- def **sum** (self)
- def **array_mean** (self)
- def **array** (self)
- def **regression** (self, name)

2.16.1 Detailed Description

An extension of the Queue class from the queue package.

Provides extensions of the first-in, first-out queue (FIFO) from the **queue** class. Each object in the queue is assumed to have a value-attribute. Additional methods are implemented to calculate the mean and sum of the queue entries. Also provides a method to return the array of values.

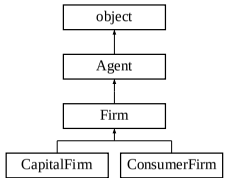
The documentation for this class was generated from the following file:

- ABM/Utilities.py

2.17 Firm Class Reference

The Firm class is an abstract base class from which actual firms are derived.

Inheritance diagram for Firm:



Public Member Functions

- def __init__ (self, unique_id, parameters)
  Initialize a generic firm.
- def pay_tax (self, government)
  Method for a firm to pay tax to government based on its net profit.
- def employee_count (self)
  Property method to return the number of employees, L_f.
- def update_employee_count (self)
  Method to update the possess basket for labour.
- def remove_employee (self, individual, Death=False)
  Method to remove an individual from the employee list.
- def pay_salaries (self)
  Method to pay salaries for all employees.
- def average_salary (self)
  A property method that returns the average salary of the employees in the firm.
- def salary_bill (self)
  A property method that returns the total salary bill for the firm.
- def hire (self, individual, level)
  A method to hire an individual at a particular education level.
- def fire (self, level)
  Method to fire a random individual at a given education level.
- def check_employees (self, population)
  A test method to check the employee list to the population.
- def plan_month (self)
  An abstract firm method that will be overridden by children classes.
- def change_labour (self, labour_change, term=None)
  A method that is used to change the firms demand for labour.

Firm-Labour-Market-Interfaces

These methods are used by the LabourMarket class, where a firm acts as a buyer in the market.

Since the firm behaves as a buyer, it must implement the buyer interfaces described in Buyer Interfaces In the labour market, the Basket of "products" is the labour composition array.

- def preferences (self)
  Property method that returns an empty set of preferences since it is not used in the labour market.

- def has_money (self, cheapest_price=None)
  A method to check if the firm has sufficient disposable income to cover the salary for the next 6 months.
- def update_baskets (self)
  The basket of labour is updated based on employee counts.
- def reset_labour_requirements (self)
  Clears the requirements basket.
- def calculate_needs (self, step=0)
  Determines the labour needs for the market based on requirements and the number of people already employed.
- def buys (self, level, individual, quantity, show_details=False)
  *Method that instructs the **firm** to ``purchase" (ie employ) an individual from the labour market.*
- def force_buy (self, level, individual)
  This method forces the firm to hire an individual, regardless of demand.
- def has_satisfied_all_needs (self)
  Checks if all employment needs of a firm have been satisfied.
- def wants_to_buy (self, level)
  Method that returns the number of employees at an education level that the firm wishes to employ.
- def needs (self, level)
  Method that is used to determine if the firm has a need for the employees at the eduction level.
- def activity (self)
  The activity attribute is not used by firms.
- def array_employees (self)
  Property method that returns an education array of the number of employed individuals.
- def array_employee_requirements (self)
  Property method that returns an education array of the number of required employed individuals.
- def array_employee_need_to_buy (self)
  Property method that returns an array of education requirements.
- def employee_requirements (self)
  Property method that returns the total requirements for labour.
- def employee_need_to_buy (self)
  Property method that returns the total needs for labour.
- def employee_possess (self)
  Property method that returns the total existing labour count.

Public Attributes

- employees
  A firm's employees is an ObjectList of Employee objects.
- requirements
  ***Labour market attritube** - The total labour requirements for the firm.*
- need_to_buy
  ***Labour market attribute** - The labour demand for the firm.*
- possess
  ***Labour market attribute** - The actual labour composition of the firm.*
- basket
  ***Labour market attribute** - The labour that has been 'purchased' in the labour market in a month.*
- **purchase_prices**
- **tax**
- **cash**
- **total_salary**

2.17.1 Detailed Description

The Firm class is an abstract base class from which actual firms are derived.

The abstract, generic properties of all firms in the business sector are implemented at this level. The two types of firms, C-firms and K-firms, are implemented as instances of the classes **ConsumerFirm** and **CapitalFirm**, which are both sub-classes of the abstract **Firm** class.

At the **Firm}** level, firms are agents that employ individuals and pay taxes. All firms will therefore have the following variables:

- revenue: The currency variable for the monthly income of the firm.
- cash: The currency variable for all the available funds on hand.

The labour market interfaces for all the children firms are provided in the Firm class. There are 3 variables used in establishing the labour demand:

- **requirements** : The total educational-labour requirements for the firm by education level.
- **possess** : The currently employed labour by education level.
- **need-to-buy** : The number of employees that are needed in order to satisfy the requirements.

The relationship between the 3 variables is:

$$\text{need-to-buy} = \text{requirements} - \text{possess}$$

2.17.2 Constructor & Destructor Documentation

2.17.2.1 __init__()

```
def __init__ (
            self,
            unique_id,
            parameters )
```

Initialize a generic firm.

The firm attributes are initialised with empty/null values.

2.17.3 Member Function Documentation

2.17.3.1 activity()

```
def activity (
              self )
```

The activity attribute is not used by firms.

Reimplemented in ConsumerFirm, and CapitalFirm.

2.17.3.2 array_employee_requirements()

```
def array_employee_requirements (
              self )
```

Property method that returns an education array of the number of required employed individuals.

2.17.3.3 buys()

```
def buys (
              self,
              level,
              individual,
              quantity,
              show_details = False )
```

Method that instructs the **firm** to ``purchase'' (ie employ) an individual from the labour market.

The 'product' is the level of education of the individual. If there is a need at the level (ie a demand for the education level, then the individual will be hired and the basket adjusted.)

2.17.3.4 calculate_needs()

```
def calculate_needs (
              self,
              step = 0 )
```

Determines the labour needs for the market based on requirements and the number of people already employed.

2.17.3.5 change_labour()

```
def change_labour (
              self,
              labour_change,
              term = None )
```

A method that is used to change the firms demand for labour.

The new total labour count is calculated (**NewLabour**). A lower bound of 5 employees is maintained. There are two scenarios for a non-zero change: either an increase of decrease in labour counts.

- **Labour Increase:** The majority of the time (~85%), the new labour is distribute in order to maintain the firm's labour mix based on technological sophistication. The rest of the time, labour is randomly distributed across education levels.
- ** Labour Decrease:** People are selected at random to be fired, at any level.

2.17.3.6 employee_possess()

```
def employee_possess (
              self )
```

Property method that returns the total existing labour count.

Should be equivalent to employee_count().

2.17.3.7 fire()

```
def fire (
              self,
              level )
```

Method to fire a random individual at a given education level.

The individual is removed from the firm list of employees, and is informed that they have been fired.

2.17.3.8 has_satisfied_all_needs()

```
def has_satisfied_all_needs (
              self )
```

Checks if all employment needs of a firm have been satisfied.

This is only used in the initialization of firms and not in the normal simulation.

2.17.3.9 hire()

```
def hire (
              self,
              individual,
              level )
```

A method to hire an individual at a particular education level.

The individual is employed at approximately their asking salary. (In future, this should undergo some form of negotiation.) The individual is appended to the firm list of employees, and is informed that they have been employed at the salary.

2.17.3.10 needs()

```
def needs (
              self,
              level )
```

Method that is used to determine if the firm has a need for the employees at the eduction level.

2.17.3.11 pay_salaries()

```
def pay_salaries (
              self )
```

Method to pay salaries for all employees.

The method loops over the collection of all employees and pays each employee the salary. It requires that each employee has a **get_salary()** method. The method returns the total salary paid to all the employees. The cash of the firm is reduced by each salary payment.

2.17.3.12 pay_tax()

```
def pay_tax (
              self,
              government )
```

Method for a firm to pay tax to government based on its net profit.

The net profit is calculated as the difference between revenue and salaries paid. Tax is only paid if the net profit is greater than zero, and is calculated based on the government tax rate. The cash reserved for the firm are reduced by the tax paid.

2.17.3.13 remove_employee()

```
def remove_employee (
              self,
              individual,
              Death = False )
```

Method to remove an individual from the employee list.

If the individual is an employee of the firm, then the individual is removed from the employee list. The employee count is updated after removal.

2.17.3.14 salary_bill()

```
def salary_bill (
              self )
```

A property method that returns the total salary bill for the firm.

2.17.3.15 update_employee_count()

```
def update_employee_count (
              self )
```

Method to update the possess basket for labour.

The education value of each employee is used to determine the possess basket of qualifications. This possess basket is the labour composition array, $\vec{L}_f$.

2.17.4 Member Data Documentation

2.17.4.1 employees

```
employees
```

A firm's employees is an ObjectList of Employee objects.

2.17.4.2 need_to_buy

```
need_to_buy
```

Labour market attribute - The labour demand for the firm.

This is the current demand basket at differnt education levels.

2.17.4.3 possess

`possess`

Labour market attribute - The actual labour composition of the firm.

This is the labour composition array basket.

2.17.4.4 requirements

`requirements`

Labour market attritube - The total labour requirements for the firm.

Of type **LabourBasket**. This is the total desired labour basket defining the desired labour composition array.

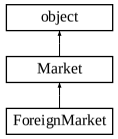
The documentation for this class was generated from the following file:

- ABM/AgentModelFirm.py

2.18 ForeignMarket Class Reference

This is the class to implement a foreign market for goods between countries and firms.

Inheritance diagram for ForeignMarket:



Public Member Functions

- def **__init__** (self, parameters, buyers, sellers)

Public Attributes

- **parameters**

2.18.1 Detailed Description

This is the class to implement a foreign market for goods between countries and firms.

In a foreign market, the buyers are a collection of **Country** agents, while the sellers are a collection of **ConsumerFirm** agents. The products in the market is a constant set of goods implemented as a **GoodsConstants** set.

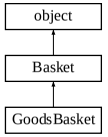
The documentation for this class was generated from the following file:

- ABM/AbstractMarket.py

2.19 GoodsBasket Class Reference

The basket class that is used to store Goods.

Inheritance diagram for GoodsBasket:



Public Member Functions

- def __init__ (self, name='Goods Basket', issupply=False, items=None, integer_items=False)

  Constructor method to initialize the basket.

Public Attributes

- **description**
- **items**

Static Public Attributes

- dictionary **description** = {}

2.19.1 Detailed Description

The basket class that is used to store Goods.

2.19.2 Constructor & Destructor Documentation

2.19.2.1 __init__()

```
def __init__ (
              self,
              size = 'Goods Basket',
              name = False,
              issupply = None,
              integer_items = False )
```

Constructor method to initialize the basket.

Each basket consists of a list of items. It is possible to have integer or real-valued items in a basket. The class provides a set of Dunder methods to allow treating a basket as a list.

Reimplemented from Basket.

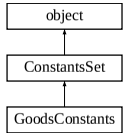
The documentation for this class was generated from the following file:

- ABM/MarketBasketModel.py

2.20 GoodsConstants Class Reference

The constant set of all goods in the economy.

Inheritance diagram for GoodsConstants:



Public Member Functions

- def random (self)

  Returns a random value from the set.

Static Public Attributes

- int **ESSENTIAL** = 20
- int **LUXURY** = 21
- list **RANGE** = [20, 21]
- string **NAME** = 'Goods'
- dictionary **DESCRIPTION**
- dictionary **INDEX**

Additional Inherited Members

2.20.1 Detailed Description

The constant set of all goods in the economy.

The constant **GOODS** is provided as the structured constant instance for $\mathcal{G}$.

2.20.2 Member Data Documentation

2.20.2.1 DESCRIPTION

```
dictionary DESCRIPTION  [static]
```

Initial value:

```
=    {ESSENTIAL:     'Essential',
                     LUXURY   :     'Luxury'}
```

2.20.2.2 INDEX

```
dictionary INDEX  [static]
```

Initial value:

```
=    {ESSENTIAL:     0,
                     LUXURY   :     1}
```

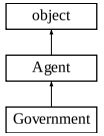
The documentation for this class was generated from the following file:

- Constants.py

2.21 Government Class Reference

The government class.

Inheritance diagram for Government:



Public Member Functions

- def **__init__** (self, unique_id, parameters, active_population, recipients)
- def **update_census** (self, active_population, recipients)
- def **collect_tax** (self, amount)
- def **plan_year** (self, year, firms)
- def **pay_recipients** (self)

Public Attributes

- **tax_rate**
- **population**
- **grant**
- **grant_maximum**
- **payments**
- **number_recipients_paid**
- **cash**
- **annual_payments**
- **annual_tax_received**
- **number_annual_payments**
- **annual_payment_history**
- **annual_profit_history**

Static Public Attributes

- **recipients** = None

2.21.1 Detailed Description

The government class.

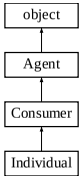
The documentation for this class was generated from the following file:

- ABM/AgentModelGovernment.py

2.22 Individual Class Reference

The Individual class provides the behaviour and properties of all individual agents in the economy.

Inheritance diagram for Individual:



Public Member Functions

- def **__init__** (self, unique_id, parameters, age=None)
- def **capacity** (self)
- def **capacity** (self, value)
- def **calculate_pension** (self)
- def **reset_cash** (self)
- def **save** (self)
- def **update_education_consequences** (self)
- def **stock** (self)
- def **has_stock** (self, quantity)
- def **employ** (self, firm, salary)
- def **fired** (self)
- def **StatusString** (self)
- def **is_active** (self)
- def **is_retired** (self)
- def **is_employed** (self)
- def **is_unemployed** (self)
- def **remove_from_firm** (self, Death=False)
- def age_a_year (self, month)

  Method to ncrease the age of an individual if it is their birthday.

Individual-Labour-Market-Interfaces

The agent's labour market seller interfaces.

These methods are used by the LabourMarket class or by other object to provide income to the individual.

- def **produce** (self)
- def **available** (self)
- def **get_initial_salary** (self)
- def **get_salary** (self, salary)
- def **get_pension** (self)
- def **get_social_grant** (self, grant)
- def **able_to_supply** (self, quantity)
- def **sells** (self, education, firm, quantity_bought)
- def **education_index** (self)

Individual-Product-Market-Interfaces

The agent's product market interfaces.

These methods are used by the ProductMarket class, or to interogate the activity in the product market.

- def activity (self)

  Method that returns the activity index to determine how active the individual is in number of trips to the souk.
- def preferences (self)

  Method that returns buyer preferences.
- def **total_consumption** (self)
- def has_money (self, cheapest_price=0)

  Method that indicates the individual has enough cash to cover the given price.
- def calculate_needs (self, market_prices)

  Method to calculate an individual's demand for goods.
- def buys (self, product, firm, quantity)

  Method where the individual buys a quantity of a product from a firm.
- def wants_to_buy (self, product)

  Methon that indicates the quantity of a product that an individual wants to purchase.
- def unsatisfied_needs (self)

  Method to indicate that there are unsatisfied needs.
- def needs (self, product)

  Method to indicate that there are needs for each pf the products.

Public Attributes

- **age**
- **status**
- **employer**
- **savings_rate**
- **revenue**
- **cash**
- **wealth**
- **product**
- **item**
- **sales_price**
- **salary**
- **birthday_month**
- **consumption**
- **pension**

Individual-Product-Market-Variables

The agent's product market variables.

- **need_to_buy**
- **basket**

2.22.1 Detailed Description

The Individual class provides the behaviour and properties of all individual agents in the economy.

2.22.2 Member Function Documentation

2.22.2.1 age_a_year()

```
def age_a_year (
              self,
              month )
```

Method to ncrease the age of an individual if it is their birthday.

The method will also check if the person has reached retirement age. If the person has retired, their status is updated to Retired and they are removed as an employee from their employer.

2.22.2.2 buys()

```
def buys (
              self,
              product,
              firm,
              quantity )
```

Method where the individual buys a quantity of a product from a firm.

If the individual has enough cash, then the individual will but the quantity of the product from the firm. The basket is updated and cash is reduced.

2.22.2.3 calculate_needs()

```
def calculate_needs (
              self,
              market_prices )
```

Method to calculate an individual's demand for goods.

The needs basket is updated.

2.22.2.4 needs()

```
def needs (
              self,
              product )
```

Method to indicate that there are needs for each pf the products.

Returns an basket-type array of boolean values.

2.22.2.5 preferences()

```
def preferences (
              self )
```

Method that returns buyer preferences.

Higher levels of education would demand more luxury type goods.

2.22.2.6 unsatisfied_needs()

```
def unsatisfied_needs (
              self )
```

Method to indicate that there are unsatisfied needs.

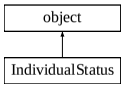
The method returns the difference between the what the individual needs and what is currently in the basket.

The documentation for this class was generated from the following file:

- ABM/AgentModelIndividual.py

2.23 IndividualStatus Class Reference

Inheritance diagram for IndividualStatus:



Public Member Functions

- def **__init__** (self, parameters, age)
- def **set** (self, newvalue)
- def **update** (self, age)
- def **__str__** (self)
- def **index** (self)
- def **IsChild** (self)
- def **IsSchool** (self)
- def **IsStudying** (self)
- def **is_unemployed** (self)
- def **is_employed** (self)
- def **is_retired** (self)
- def **is_active** (self)

Public Attributes

- **parameters**
- **education**
- **value**

2.23.1 Detailed Description

```
*************** METHODS ***************
```

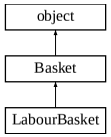
The documentation for this class was generated from the following file:

- ABM/IndividualStatusModel.py

2.24 LabourBasket Class Reference

The basket class that is used to store Education.

Inheritance diagram for LabourBasket:



Public Member Functions

- def __init__ (self, name='Labour Basket', issupply=False, items=None, integer_items=False)

  Constructor method to initialize the basket.

Public Attributes

- **description**
- **items**

Static Public Attributes

- dictionary **description** = {}

2.24.1 Detailed Description

The basket class that is used to store Education.

2.24.2 Constructor & Destructor Documentation

2.24.2.1 __init__()

```
def __init__ (
            self,
            size = 'Labour Basket',
            name = False,
            issupply = None,
            integer_items = False )
```

Constructor method to initialize the basket.

Each basket consists of a list of items. It is possible to have integer or real-valued items in a basket. The class provides a set of Dunder methods to allow treating a basket as a list.

Reimplemented from Basket.

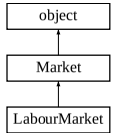
The documentation for this class was generated from the following file:

- ABM/MarketBasketModel.py

2.25 LabourMarket Class Reference

This is the class to implement a labour market between firms and individuals.

Inheritance diagram for LabourMarket:



Public Member Functions

- def **__init__** (self, parameters, buyers, sellers)

Public Attributes

- **parameters**

2.25.1 Detailed Description

This is the class to implement a labour market between firms and individuals.

In a labour market, the buyers are a collection of **Firm** agents, while the sellers are a collection of **Individual** agents. The products in the market is a constant set of goods implemented as a **EducationConstants** set.

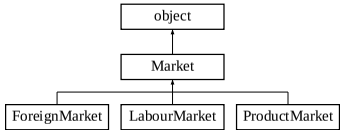
The documentation for this class was generated from the following file:

- ABM/AbstractMarket.py

2.26 Market Class Reference

This is an abstract class from which the various actual market classes are derived.

Inheritance diagram for Market:



Public Member Functions

- def __init__ (self, name, parameters, buyers, sellers, products, prices)
  Constructor for generic market.
- def product_name (self, product_id)
  Returns the product name based on the product identifier.
- def seller_stock (self)
  Returns the total stock basket for all the sellers in the market.
- def supplier_production (self)
  An instruction to all the sellers in the market to produce their goods.
- def calculate_capacity (self)
  An instruction to all the sellers in the market to determine their production capacity.
- def calculate_prices (self)
  A method to calculate the market, demand and supply prices in the market.
- def calculate_market_prices (self)
  A method to calculate the average market prices in the market.

- def update_buyers (self, buyers)
  Method to update the collection of buyers.
- def update_sellers (self, sellers)
  Method to update the collection of sellers.
- def number_buyers (self)
  Property method that returns the number of buyers in the market.
- def number_sellers (self)
  Property method that returns the number of selelrs in the market.
- def shopping (self, step)
  The main shopping algorithm that implements the mechanism of how buyers interact with sellers.

Public Attributes

- **parameters**
- **name**
- products
  The set of products that are bought/sold in the market.
- **NumProducts**
- **dictionary**
- **prices**
- **market_prices**
- **number_suppliers**
- buyers
  The collection of buyers in the market.
- sellers
  The collection of sellers in the market.
- **demand**
- **sales**
- **capacity**
- **volume**

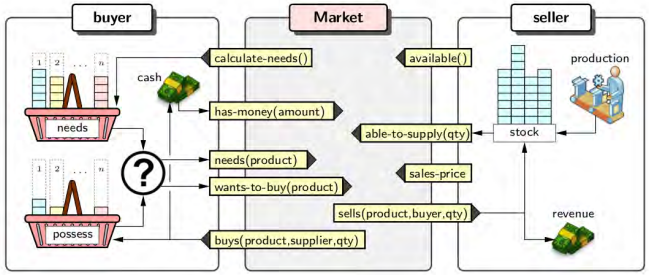
2.26.1 Detailed Description

This is an abstract class from which the various actual market classes are derived.

The abstract class **Market** is the class that facilitates the interactions between the various economic agents. The main purpose of the **Market** class is the provision of the **shopping** algorithm. The following three markets used in the simulation are derived from the abstract **Market** class:

- ProductMarket
- LabourMarket
- ForeignMarket The major difference between these different types of markets are who the buyers and sellers are.

The various market interfaces are summarised in the following diagram, and described in the sections that follow.



2.26.1.1 Buyer Interfaces

When created, a market is initialized with a collection of **buyers** agents. In the definition of each **buyers** agent class, there are predefined interfaces that will be required in the **market.shopping** algorithm. The interfaces would require the implementation of various attributes and methods within the respective classes. The **Market** class requires that the **buyer** agents contains the following attributes:

Attribute	Data Type	Description
need_to_buy	**Basket**	A Goods basket of product requirements.
purchase_prices	**Basket**	A Goods basket of product prices the buyer is willing to pay.
activity	**Integer**	Index that can be used for frequency of trips to the souk.
		Attribute that indicates the level of activity of the **buyer** agent.
		Returns the **buyer** level of activity in the market.
		The activity is used to determine the number of trips to a souk.
		In the case of an **individual** agent, the activity is given by the education level, so that individuals with higher levels of education could take more trips to the souk in the product market, since they have more disposable income.
		This attribute is not relevant for **firm** agents, and is set to 0.
preferences	list of **Integer**	An array og Goods indices that are the first items to be purchased when shopping.

The **Market** class requires that the **buyer** agents implement the following methods:

Method	Description
calculate-needs()	Instruction to the buyer agent to calculate their needs in the market.
	This method prompts the buyer agent to determine the basket of products that it requires.
	For example, for **individual** agents, the needs will be determined by whether or not the individual is employed, the education level of the individual, and the disposable income available resulting in a **needs** basket of goods.
	For a **firm** agent, this method results in a **needs** basket of labour requirements.
	The **need_to_buy** basket establishes the demand in the market.
	This method should be called prior to the shopping algorithm.
has-money(amount)	A method to determine if the **buyer** has sufficient funds to purchase a product for the given amount in the market.

Method	Description
	Returns **True**/**False** based on if it has enough cash for the amount.
	If no amount is provided, the method will simply check if the agent has some disposable cash available.
needs(product)	This method will return a **True**/**False** value based on whether or not the **buyer** agent still requires the product.
	Returns **True**/**False** if it needs the product.
	The **buyer** agent will compare its **needs** with its current purchases, and it it has not satisfied its requirements for the product, will return a **True** value.
wants-to-buy(product)	This method will return the quantity of the product that the **buyer** wishes to purchase.
	Returns the quantity of the product it wishes to purchase.
	The **buyer** will determine this from its **needs** and how much it has already purchased.
buys(product,supplier,qty)	This method instructs the **buyer** to now purchase the quantity of the product from the **supplier**.
	Buyer agent buys the quantity of the product from the supplier.
	The **buyer** will add the quantify to its **possess** of goods that it has purchased.

2.26.1.2 Seller Interfaces

The **Market** class requires that the **seller** agents contains the following attributes:

Attribute	Data Type	Description
product	**Integer**	The product that the seller wants to sell.
sales-price	**Real**	The sales price for the product.
capacity	**Integer**	The number of items the seller is able to sell.

The **Market** class requires that the **seller** agents implement the following methods:

Method	Description
available()	A method to determine of the seller is available to supply the market.
	If the seller is a firm, then the availability is determined from available stock.
	If the seller is an individual, then the availability is based on if the individual is unemployed.
able-to-supply(qty)	Returns the quantity of a product the seller is capable of providing.
sells(product,buyer,qty)	Seller agent sells the quantity of the product to the buyer.

2.26.1.3 The shopping algorithm

The main feature of the abstract **Market** class is the generic **shopping** algorithm. The algorithm describes the process of how a buyer agent will purchase its desired goods from the available sellers. The algorithm requires that certain methods are implemented in in the buyer and seller classes, described in the previous sections.

The algorithm is described as follows:

Heuristic 2.1: Shopping Algorithm

```
product_buyers ← randomised list of buyer agents
product_sellers ← product indexed list of seller agents

foreach buyer in product_buyers:
  buyer.calculate_needs()

  if buyer.has_money():
    number_trips ← trips_to_souk[buyer.activity()]
```

```
    foreach trip in number_trips:
      product ← randomly chosen product from market.products
      cheapest ← souk.cheapest_price(product)

      if buyer.needs(product) and buyer.has_money(cheapest):
          qty_to_buy ← buyer.wants_to_buy(product)
          supplier ← souk.choose_seller(product)

          if (supplier.available()) and (qty_to_buy > 0):
              qty_available ← supplier.able_to_supply(qty_to_buy)

              if qty_available > 0:
                  qty_bought ← buyer.buys(product,supplier,qty_available)

                  if qty_bought > 0:

                      if not(supplier.available()):
                          souk.remove_seller(product,supplier)
```

2.26.2 Constructor & Destructor Documentation

2.26.2.1 __init__()

```
def __init__ (
              self,
              name,
              parameters,
              buyers,
              sellers,
              products,
              prices )
```

Constructor for generic market.

2.26.3 Member Function Documentation

2.26.3.1 shopping()

```
def shopping (
              self,
              step )
```

The main shopping algorithm that implements the mechanism of how buyers interact with sellers.

For details on the shopping algorithm see The shopping algorithm.

2.26.4 Member Data Documentation

2.26.4.1 buyers

`buyers`

The collection of buyers in the market.

The buyers must be implemented as a Tree (from the treelib library). Each leaf in the tree would be an **Agent** object that implements the interface methods described in Buyer Interfaces.

2.26.4.2 products

`products`

The set of products that are bought/sold in the market.

The products should be a subclass of a ConstantsSet.

2.26.4.3 sellers

`sellers`

The collection of sellers in the market.

The sellers must be implemented as a Tree (from the treelib library). Each leaf in the tree would be an **Agent** object that implements the interface methods described in Seller Interfaces.

The documentation for this class was generated from the following file:

- ABM/AbstractMarket.py

2.27 ObjectList Class Reference

An extension of the UserList class provided in the collections package.

Inheritance diagram for ObjectList:

Public Member Functions

- def **len** (self)
- def **__nonzero__** (self)
- def **__str__** (self)
- def **__iter__** (self)
- def **__call__** (self, *args, **kwargs)
- def **__getattr__** (self, name)
- def **size** (self)

2.27.1 Detailed Description

An extension of the UserList class provided in the collections package.

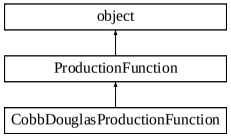
This class is almost exactly like a regular list of Nodes (actually it can hold any object), with one important difference. If you try to get an attribute from this list, it will return that attribute from every item in the list.

The documentation for this class was generated from the following file:

- ABM/Utilities.py

2.28 ProductionFunction Class Reference

Inheritance diagram for ProductionFunction:



Public Member Functions

- def **__init__** (self, name='Production Function')
- def **__call__** (self, K, L)

Public Attributes

- **name**

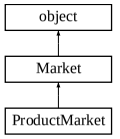
The documentation for this class was generated from the following file:

- ABM/ProductionFunctionModels.py

2.29 ProductMarket Class Reference

This is the class to implement a product market between individuals and firms.

Inheritance diagram for ProductMarket:



Public Member Functions

- def **__init__** (self, parameters, buyers, sellers)
- def adjust_product_pricing (self)

  The product market provides a mechanism for pricing adaptation.

Public Attributes

- **parameters**

2.29.1 Detailed Description

This is the class to implement a product market between individuals and firms.

In a product market, the buyers are a collection of **Individual** agents, while the sellers are a collection of **ConsumerFirm** agents. The products in the market is a constant set of goods implemented as a **GoodsConstants** set.

2.29.2 Member Function Documentation

2.29.2.1 adjust_product_pricing()

```
def adjust_product_pricing (
            self )
```

The product market provides a mechanism for pricing adaptation.

The sellers (i.e. firms) must also implement an **adjust_pricing()** method to adapt its prices to the prevailing market conditions. Each firm is provided with the average market, supply and demand prices.

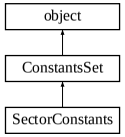
The documentation for this class was generated from the following file:

- ABM/AbstractMarket.py

2.30 SectorConstants Class Reference

The constant set of all sectors in the economy.

Inheritance diagram for SectorConstants:



Static Public Attributes

- int **ESSENTIAL** = 20
- int **LUXURY** = 21
- int **CAPITAL** = 22
- list **RANGE** = [20, 21,22]
- string **NAME** = 'Sectors'
- dictionary **DESCRIPTION**
- dictionary **INDEX**

Additional Inherited Members

2.30.1 Detailed Description

The constant set of all sectors in the economy.

The constant **SECTORS** is provided as the structured constant instance.

2.30.2 Member Data Documentation

2.30.2.1 DESCRIPTION

```
dictionary DESCRIPTION  [static]
```

Initial value:

```
=    {ESSENTIAL:     'Essential',
                     LUXURY   :     'Luxury',
                     CAPITAL  :     'Capital'}
```


2.30.2.2 INDEX

```
dictionary INDEX  [static]
```

Initial value:

```
=    {ESSENTIAL:     0,
                     LUXURY   :     1,
                     CAPITAL  :     2}
```

The documentation for this class was generated from the following file:

- Constants.py

2.31 Souk Class Reference

Inheritance diagram for Souk:

Public Member Functions

- def **__init__** (self, parameters, product_sellers, products, sizes)
- def **cheapest_price** (self, item)
- def **number_shops** (self)
- def **choose_seller** (self, item)
- def **remove_seller** (self, item, seller)

Public Attributes

- **probabilities**
- **souk_sizes**
- **products**
- **shops**
- **cheapest**

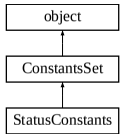
The documentation for this class was generated from the following file:

- ABM/SoukModel.py

2.32 StatusConstants Class Reference

The constant set of all status values for individuals in the economy.

Inheritance diagram for StatusConstants:



Public Member Functions

- def **isconsumer** (self, index)
- def **isemployed** (self, index)

Static Public Attributes

- int **CHILD** = 0
- int **SCHOOL** = 1
- int **STUDYING** = 2
- int **UNEMPLOYED** = 3
- int **EMPLOYED** = 4
- int **RETIRED** = 5
- list **RANGE_PASSIVE** = [0,1,2,5]
- list **RANGE_ACTIVE** = [3,4]
- list **RANGE** = [0,1,2,3,4,5]
- string **NAME** = 'Individual Status'
- dictionary **DESCRIPTION**
- dictionary **INDEX**

Additional Inherited Members

2.32.1 Detailed Description

The constant set of all status values for individuals in the economy.

The constant **STATUS** is provided as the structured constant instance for $\mathcal{S}$.

2.32.2 Member Data Documentation

2.32.2.1 DESCRIPTION

```
dictionary DESCRIPTION  [static]
```

Initial value:

```
= {
                        CHILD:      'Child',
                        SCHOOL:     'School',
                        STUDYING:   'Studying',
                        UNEMPLOYED: 'Unemployed',
                        EMPLOYED:   'Employed',
                        RETIRED:    'Retired'}
```

2.32.2.2 INDEX

```
dictionary INDEX  [static]
```

Initial value:

```
=    {CHILD:         0,
                      SCHOOL:        1,
                      STUDYING:      2,
                      UNEMPLOYED:    3,
                      EMPLOYED:      4,
                      RETIRED:       5 }
```

The documentation for this class was generated from the following file:

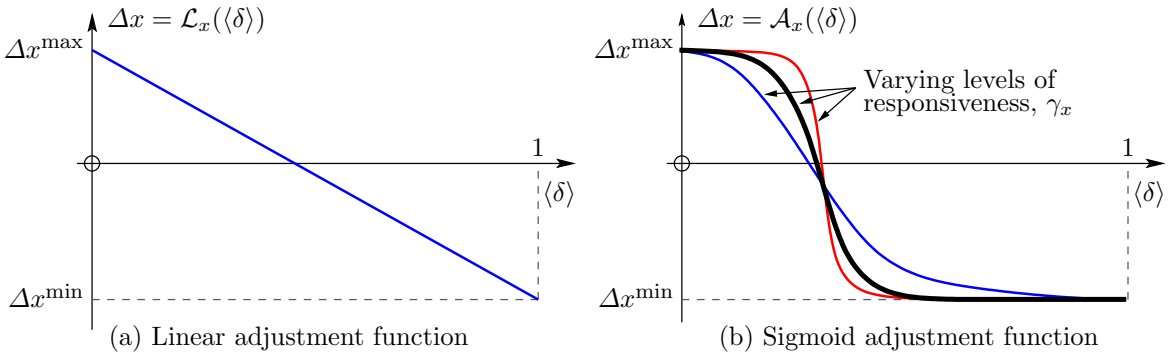
- Constants.py

Class Documentation
68
Generated by Doxygen

C.4 Miscellaneous design details

C.4.1 Adjustment functions

This section outlines the way in which adjustment functions used in the simulation. When using the average relative stock level, $\langle\delta\rangle$, a variable x (for example capital, K_f) may need to change by an amount, Δx. An *adjustment function* will provide the dependence of the change, Δx, on the stock, $\langle\delta\rangle$. The simplest way in which Δx can be determined is using a *linear adjustment function*, $\mathcal{L}_x$, as shown in Figure C.4(a). The function $\mathcal{L}_x()$ is parameterized by the minimum and maximum values, $\Delta x^{\min}$, $\Delta x^{\max}$.



(a) Linear adjustment function (b) Sigmoid adjustment function

Figure C.4: Various adjustment functions, used in the simulation.

A second way in which Δx can be determined is using a *sigmoid adjustment function*, $\mathcal{A}_x$, as shown in Figure C.4(b). This adjustment function 'softens' a heuristic of the form,

$$\boxed{\begin{array}{l}\text{if } \langle\delta\rangle \text{ is LOW, then increase } x \text{ by } \Delta x \approx \Delta x^{\max} \\ \text{if } \langle\delta\rangle \text{ is HIGH, then decrease } x \text{ by } \Delta x \approx \Delta x^{\min},\end{array}} \tag{C.1}$$

by allowing a continuous set of values for the change, Δx. The change would be given by $\Delta x = \mathcal{A}_x(\langle\delta\rangle)$, where the function $\mathcal{A}_x()$ is parameterized by minimum and maximum values, $\Delta x^{\min}$, $\Delta x^{\max}$, and a responsiveness parameter, γ_x. For low value of the relative stock ($\langle\delta\rangle \to 0$), the change will tend towards the maximum value, $\Delta x^{\max}$; while for high values of the relative stock ($\langle\delta\rangle \to 1$), the change will tend towards the minimum value, $\Delta x^{\min}$; with a continuous range of values in-between.

C.4.2 Government tax rate adjustment

Changes to the tax rate is calculated in the following way. Government will calculate the total net profit of firms $N_{\text{tot}}(t)$. The revenue obtained over the year is equal to the tax rate multiplied by the firm total revenue, $R(t) = r_{\text{G}}(t)N_{\text{tot}}(t)$, while the total social grant payments made is $P_{\text{tot}}(t)$, so that the excess government income is given by

$$E(t) = R(t) - P_{\text{tot}}(t) = r_{\text{G}}(t)N_{\text{tot}}(t) - P_{\text{tot}}(t). \tag{C.2}$$

In order to minimize the tax burden, one can require that the rate of change of excess income must equal zero, which implies that the time derivative must vanish,

$$\frac{dE(t)}{dt} \stackrel{!}{=} 0. \tag{C.3}$$

This gives the condition

$$\frac{dr_{\mathrm{G}}(t)}{dt} N_{\mathrm{tot}}(t) + r_{\mathrm{G}}(t) \frac{dN_{\mathrm{tot}}(t)}{dt} - \frac{dP_{\mathrm{tot}}(t)}{dt} \stackrel{!}{=} 0. \tag{C.4}$$

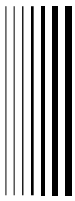
from which we obtain the rate of change of taxation

$$\frac{dr_{\mathrm{G}}(t)}{dt} = \frac{1}{N_{\mathrm{tot}}(t)} \frac{dP_{\mathrm{tot}}(t)}{dt} - \frac{r_{\mathrm{G}}(t)}{N_{\mathrm{tot}}(t)} \frac{dN_{\mathrm{tot}}(t)}{dt}. \tag{C.5}$$

C.5 Individual status changing rules

The following points describe the basic rules on how an individuals status and education changes over time. The *progression probability* parameters, $\hat{p}_{\mathrm{E}i}$ used in the simulation are given in Table C.2, pg.100.

- When a new individual is 'born', it is created with an age equal to 0 and the individual•status variable will be assigned the value *Child*, while the individual•education variable will be assigned a value of *Unskilled*.
- At the age of 6, the individual will then enter the schooling system, with individual•status assigned the value *School* until the age of 18.
- Whether or not the individual actually finishes school is based on the progression probability $\mathcal{P}(\text{finishing school}) = 0.62$, capturing a school dropout rate of 38%.
- When a individual does dropout of the schooling system, the individual•status variable is assigned the value *Unemployed*, and becomes an *active* member of the economy.
- If the individual does finish school, then there is a probability that they will pass matric, $\mathcal{P}(\text{passing matric}) = 0.75$, which results in the individual•education variable being assigned a value of *Matric* (ie. the individual is then classified as semi-skilled).
- Those 25% of individuals that do not pass matric, would then have the individual•status variable is assigned the value *Unemployed*, with the individual•education variable remaining at *None* (ie. the individual is then classified as unskilled).
- There is a progression probability $\mathcal{P}(\text{entering tertiary}) = 0.4$ that the individual that passed matric will enter higher education, upon which the individual•status variable is assigned the value *Studying*.
- The progression probability of graduating with a tertiary education, $\mathcal{P}(\text{passing tertiary}) = 0.75$, implies that 25% of individuals drop out of higher education. Those that drop out result in the individual•status variable assigned the value *Unemployed* and the individual•education variable remaining unchanged at a value of *Matric* (ie. the individual is classified as semi-skilled).
- The 75% that do pass tertiary education may continue studying, with a progression probability $\mathcal{P}(\text{entering postgraduate}) = 0.4$.
- Alternatively, they would enter the pool of the unemployed, with the individual•status variable assigned the value *Unemployed* and the individual•education variable assigned a value *Bachelors*, so that they are classified as being skilled.
- The progression probability of graduating with a postgraduate qualification, $\mathcal{P}(\text{passing postgraduate}) = 0.95$, implies that the vast majority of individuals will graduate. Upon completion, once again the individual•status variable assigned the value *Unemployed* and the individual•education variable is assigned the value *Postgraduate* (ie. are highly-skilled).



References

Abar, S., Theodoropoulos, G., Lemarinier, P. & O'Hare, G. (2017), 'Agent Based Modelling and Simulation tools: A review of the state-of-art software', *Computer Science Review* **24**, 13–33.

Abel, A., Bernanke, B. & Croushore, D. (2016), *Macroeconomics, Global Edition*, Pearson Education Limited.
URL: *https://books.google.co.za/books?id=mB3aDAAAQBAJ*

Abergel, F., Aoyama, H., Chakrabarti, B. K., Chakraborti, A. & Ghosh, A., eds (2013), *Econophysics of Agent-Based Models*, New Economic Windows, Springer International Publishing.

Acemoglu, D. (2009), *Introduction to Modern Economic Growth*, Princeton University Press.

Aghion, P., Akcigit, U. & Howitt, P. (2014), 'What Do We Learn From Schumpeterian Growth Theory?', *Handbook of Economic Growth* **2**, 515–563.

Aghion, P., Akcigit, U. & Howitt, P. (2015), 'The Schumpeterian Growth Paradigm', *Annual Review of Economic* **7**(1), 557–575.

Aghion, P. & Howitt, P. (1992), 'A Model of Growth Through Creative Destruction', *Econometrica* **60**(2), 323–351.

Aghion, P. & Howitt, P. (1998), *Endogenous Growth Theory*, MIT Press.

Aghion, P., Meghir, C. & Vandenbussche, J. (2006), 'Growth , Distance to Frontier and Composition of Human Capital', *Journal of Economic Growth* .

Akanbi, O. A. (2016), 'The growth, poverty and inequality nexus in South Africa: Cointegration and causality analysis', *Development Southern Africa* **33**(2), 166–185.

Andrade, L. R., Cardenas, L. Q., Lopes, F. D., Oliveira, F. P. & Fernandes, K. C. (2018), 'The impact of R \& D investments on performance of firms in different degrees of proximity to the technological frontier', *Economics Bulletin* **38**(2), 1156–1170.

Ashraf, Q., Gershman, B. & Howitt, P. (2014), 'How inflation affects macroeconomic performance: An agent-based computational investigation', *Macroeconomic Dynamics* **20**(2), 558–581.

Ashraf, Q., Gershman, B. & Howitt, P. (2017), 'Banks, market organization, and macroeconomic performance: An agent-based computational analysis', *Journal of Economic Behavior and Organization* **135**, 143–180.

Asongu, S. A. (2017), 'The comparative economics of knowledge economy in africa: policy benchmarks, syndromes, and implications', *Journal of the Knowledge Economy* **8**(2), 596–637.

Asongu, S. A. & Andrés, A. R. (2020), 'Trajectories of knowledge economy in ssa and mena countries', *Technology in Society* **63**, 101119.

Asongu, S. A. & Odhiambo, N. M. (2020), 'Building knowledge-based economies in africa: A systematic review of policies and strategies', *Journal of the Knowledge Economy* **11**, 1538–1555.

Assenza, T., Delli Gatti, D. & Grazzini, J. (2015), 'Emergent dynamics of a macroeconomic agent based model with capital and credit', *Journal of Economic Dynamics and Control* **50**, 5–28.

Atkinson, H. (2006), 'Strategy implementation: a role for the balanced scorecard?', *Management decision* .

Axtell, R. L. & Epstein, J. M. (2006), Coordination in transient social networks: An agent-based computational model of the timing of retirement, *in* J. M. Epstein, ed., 'Generative social science: Studies in agent-based computational modeling', chapter 7.

Barney, J. (1991), 'Firm Resources and Sustained Competitive Advantage', *Journal of Management* **17**(1), 99–120.

Barney, J. B. & Clark, D. N. (2007), *Resource-based theory: Creating and sustaining competitive advantage*, Oxford University Press.

Barro, R. J., Barro, R. J., Sala-i Martin, X. & Sala-i Martin, X. I. (2004), *Economic Growth*, 2nd edn, McGraw-Hill.

Bechet, T. (2008), *Strategic Staffing: A Comprehensive System for Effective Workforce Planning*, BusinessPro collection, American Management Association.

Besanko, D. & Braeutigam, R. (2013), *Microeconomics*, 5th edn, Wiley Global Education.

Biepke, N. (2018), 'SA , jobs , and the 4th industrial revolution'.

Billari, F. C., Fent, T., Prskawetz, A. & Scheffran, J. (2006), Agent-Based Computational Modelling: An Introduction, *in* F. C. Billari, T. Fent, A. Prskawetz & J. Scheffran, eds,

'Agent-Based Computational Modelling Applications in Demography, Social, Economic and Environmental Sciences', Physica-Verlag.

Blanchard, O. (2018), 'On the future of macroeconomic models', *Oxford Review of Economic Policy* **34**(1-2), 43–54.

Blankley, W. O. & Booyens, I. (2010), 'Building a knowledge economy in South Africa', *South African Journal of Science* **106**(11-12), 1–6.

Boero, R. (2015), A Few Good Reasons to Favor Agent-based Modeling in Economic Analyses, *in* R. Boero, M. Morini, M. Sonnessa & P. Terna, eds, 'Agent-based Models of the Economy', Springer, chapter 1, pp. 3–9.

Boero, R., Morini, M., Sonnessa, M. & Terna, P., eds (2015), *Agent-based Models of the Economy: From Theories to Applications*, Palgrave Macmillan UK.

Bonaventura, L. (2011), 'Enforcement of regulation, irregular sector, and firm performance: A computational agent-based model', *Netnomics: Economic Research and Electronic Networking* **12**(2), 99–113.

Booch, G., Maksimchuk, R., Engle, M., Conallen, J., Houston, K. & D, B. Y. P. (2007), *Object-Oriented Analysis and Design with Applications*, Addison-Wesley Object Technology Series, Pearson Education.

Bremer, L., Heitmann, M. & Schreiner, T. F. (2017), 'When and how to infer heuristic consideration set rules of consumers', *International Journal of Research in Marketing* **34**(2), 516–535.

Bruns, S. B. & Ioannidis, J. P. (2020), 'Determinants of economic growth : Different time different answer?', *Journal of Macroeconomics* **63**, 103185.

Burda, M. & Wyplosz, C. (2017), *Macroeconomics: A European Text*, 6th edn, Oxford University Press.

Butuner, H. (2015), *Systematic Strategic Planning: A Comprehensive Framework for Implementation, Control, and Evaluation*, CRC Press.

Cantner, U. & Pyka, A. (1998), 'Technological evolution - an analysis within the knowledge-based approach', *Structural Change and Economic Dynamics* **9**(85-107).

Carroll, C. D. (2001), 'A Theory of the Consumption Function, With and Without Liquidity Constraints', *Journal of Economic Perspectives* **15**(3), 23–45.

Chen, D. H. & Dahlman, C. J. (2005), 'The knowledge economy, the kam methodology and world bank operations', *World Bank Institute Working Paper* (37256).

Christiano, L. J., Eichenbaum, M. S. & Trabandt, M. (2018), 'On DSGE Models', *The Journal of Economic Perspectives* **32**(3), 113–140.

Ciutacu, I. & Micu, L.-A. (2015), 'The Firm, Part Of The Economic System: Reasons For Exiting A Market-An Agent-Based Modeling Approach', *Revista Economica* **67**.

Collins, J. & Porras, J. (2011), *Built to Last: Successful Habits of Visionary Companies*, Harper Business.

Coluccia, D. (2012*a*), The First Industrial Revolution (c1760-c1870), *in* G. Zanda, ed., 'Corporate Management in a Knowledge-Based Economy', Springer, chapter Ch3., pp. 41–51.

Coluccia, D. (2012*b*), The Second Industrial Revolution (late 1800s and early 1900s), *in* G. Zanda, ed., 'Corporate Management in a Knowledge-Based Economy', Springer, pp. 52–64.

Costa, C. J. (2016), *Understanding DSGE Models: Theory and Applications*, Vernon Series in Economic Methodology, Vernon Art and Science Incorporated.

CRA (2018), Education the single greatest obstacle tosocio-economic advancement in South Africa, Technical Report April, Centre for Risk Analysis.

David, F. & David, F. (2014), *Strategic Management: Concepts and Cases, Global Edition*, Always learning, Pearson Education, Limited.

David, P. A. (2005), 'Path dependence in economic processes: implications for policy analysis in dynamical systems contexts', *The evolutionary foundations of economics* pp. 151–194.

Davies, R. H. & van Seventer, D. (2020), *Labour market polarization in South Africa: A decomposition analysis*, number 2020/17, WIDER Working Paper.

Dawid, H. & Delli Gatti, D. (2018), Agent-Based Macroeconomics, *in* C. Hommes & B. LeBaron, eds, 'Handbook of Computational Economics - Volume 4', North Holland, chapter 2.

Dawid, H., Gemkow, S., Harting, P., van der Hoog, S. & Neugart, M. (2014), 'Agent-Based Macroeconomic Modeling and Policy Analysis: The Eurace@Unibi Model', *Handbook on Computational Economics and Finance* **01**.

Dawid, H., Harting, P., van der Hoog, S. & Neugart, M. (2016), A Heterogeneous Agent Macroeconomic Model for Policy Evaluation : Improving Transparency and Reproducibility.

De Vos, A., Strydom, H., Fouché, C. & Delport, C. (2011), *Research at Grass Roots: For the Social Sciences and Human Service Professions*, Van Schaik Publishers.

Delli Gatti, D., Desiderio, S., Gaffeo, E., Cirillo, P. & Gallegati, M. (2011), *Macroeconomics from the Bottom-up*, Vol. 1, Springer Science \& Business Media.

Delli Gatti, D., Di Guilmi, C., Gaffeo, E., Giulioni, G., Gallegati, M. & Palestrini, A. (2005), 'A new approach to business fluctuations: Heterogeneous interacting agents, scaling laws and financial fragility', *Journal of Economic Behavior and Organization* **56**(4 SPEC. ISS.), 489–512.

Diaconu, M. (2011), 'Technological Innovation: Concept, Process , Typology and Implications in the Economy', *Theoretical & Applied Economics* **18**(10), 127–144.

Dosi, G., Fagiolo, G. & Roventini, A. (2010), 'Schumpeter meeting Keynes: A policy-friendly model of endogenous growth and business cycles', *Journal of Economic Dynamics and Control* **34**(9), 1748–1767.

Durlauf, S. N., Johnson, P. A. & Temple, J. R. W. (2005), Chapter 8 Growth Econometrics, *in* P. Aghion & S. N. Durlauf, eds, 'Handbook of Economic Growth', Elsevier, pp. 555–677.

Endres, H. (2017), *Adaptability Through Dynamic Capabilities: How Management Can Recognize Opportunities and Threats*, Springer Fachmedien Wiesbaden.

Epstein, J. M. (2006), *Generative social science: Studies in agent-based computational modeling*, Princeton University Press.

Etzkowitz, H. & Leydesdorff, L. (1995), 'The Triple Helix–University-industry-government relations: A laboratory for knowledge based economic development', *EASST review* **14**(1), 14–19.

Falkenberg, R. (2018), 'For South Africa To Thrive In The Fourth Industrial Revolution, We Need A Learning Revolution'.

Fleisher, C. S. & Bensoussan, B. E. (2007), *Business and Competitive Analysis: Effective Application of New and Classic Methods*, Financial Times Press.

Foss, N. J. (2011), 'Why micro-foundations for resource-based theory are needed and what they may look like', *Journal of Management* **37**(5), 1413–1428.

Gatti, D. D., Fagiolo, G., Gallegati, M., Richiardi, M. & Russo, A. (2018), *Agent-Based Models in Economics: A Toolkit*, Cambridge University Press.

Géron, A. (2019), *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'Reilly Media.

Giordano, F., Fox, W. P. & Horton, S. (2014), *A first course in mathematical modeling*, 5th edn, Brooks/Cole.

Giovannoni, D., Murugan, J. & Prinsloo, A. (2011), 'The Giant graviton on AdS4xCP3 - another step towards the emergence of geometry', *Journal of High Energy Physics* **12**, 3.

Godin, B. (2006), 'The knowledge-based economy: Conceptual framework or buzzword?', *Journal of Technology Transfer* **31**(1), 17–30.

Grant, R. M. (1991), 'The Resource-Based Theory of Competitive Advantage: Implications for Strategy Formulation', *California Management Review* **33**(3), 114–135.

Griffiths, D. J. (2008), *Introduction to Electrodynamics*, Pearson international edition, 3rd edn, Pearson Education.

Grilli, R. & Tedeschi, G. (2016), Modeling Financial Markets in an Agent-Based Framework, *in* A. Caiani, A. Russo, A. Palestrini & M. Gallegati, eds, 'Economics with Heterogeneous Interacting Agents: A Practical Guide to Agent-Based Modelling', Springer, pp. 103–155.

Grossman, G. M. & Helpman, E. (1991), *Innovation and Growth in the Global Economy*, M.I.T Press Readings in Economics, MIT Press.

Grow, A. & Van Bavel, J. (2016), *Agent-Based Modelling in Population Studies: Concepts, Methods, and Applications*, The Springer Series on Demographic Methods and Population Analysis, Springer International Publishing.

Grow, A. & Van Bavel, J. (2017), Agent-Based Modelling as a Tool to Advance Evolutionary Population Theory, *in* A. Grow & J. Van Bavel, eds, 'Agent-Based Modelling in Population Studies', Springer International Publishing, chapter 1.

Gualdi, S., Tarzia, M., Zamponi, F. & Bouchaud, J. P. (2015), 'Tipping points in macroeconomic agent-based models', *Journal of Economic Dynamics and Control* **50**, 29–61.

Guo, Z. & Sheffield, J. (2006), A Paradigmatic and Methodological Examination of KM Research : 2000 to, *in* 'Proceedings of the 39th Hawaii International Conference on System Sciences', IEEE.

Hamill, L. & Gilbert, N. (2016), *Agent-Based Modelling in Economics*, Wiley.

Hammer, M. & Stanton, S. (1995), *The Reengineering Revolution: The Handbook*, Harper-Collins.

Harris, R. G. (2001), 'The knowledge-based economy : intellectual origins and new economic perspectives', *International journal of management reviews* **3**(1), 21–40.

He, Q. & Xu, B. (2019), 'Determinants of economic growth : A varying-coefficient path identification approach', *Journal of Business Research* **101**, 811–818.

Heathfield, D. F. & Wibe, S. (2016), *An Introduction to Cost and Production Functions*, Macmillan Education, Limited.

Helfat, C. E., Finkelstein, S., Mitchell, W., Peteraf, M., Singh, H., Teece, D. & Winter, S. G. (2009), *Dynamic capabilities: Understanding strategic change in organizations*, John Wiley \& Sons.

Helfat, C. E. & Peteraf, M. A. (2003), 'The dynamic resource-based view: Capability lifecycles', *Strategic Management Journal* **24**(10 SPEC ISS.), 997–1010.

Hillar, G. C. (2015), *Learning Object-Oriented Programming*, Packt Publishing.

Hungwe, M. & Odhiambo, N. M. (2019), 'Savings and investment dynamics in south africa', *Acta Universitatis Danubius. Œconomica* **15**(3).

Ikeda, Y., Souma, W., Aoyama, H., Iyetomi, H., Fujiwara, Y. & Kaizoji, T. (2007), 'Quantitative agent-based firm dynamics simulation with parameters estimated by financial and transaction data analysis', *Physica A: Statistical Mechanics and its Applications* **375**(2), 651–667.

Jones, C. I. (2002), *Introduction to Economic Growth*, 2nd edn, W.W. Norton.

Kim, W. C. & Mauborgne, R. (2015), *Blue Ocean Strategy, Expanded Edition: How to Create Uncontested Market Space and Make the Competition Irrelevant*, Harvard Business School Press / W. Chan Kim, Harvard Business Review Press.

Kinsella, S., Greiff, M. & Nell, E. J. (2011), 'Income distribution in a stock-flow consistent model with education and technological change', *Eastern Economic Journal* **37**(1), 134–149.

Kiran, M., Richmond, P., Holcombe, M., Chin, L. S., Worth, D. & Greenough, C. (2010), FLAME : Simulating Large Populations of Agents on Parallel Hardware Architectures, *in* 'Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems - Volume 1', pp. 1633–1636.

Kletti, J. (2007), *Manufacturing Execution System - MES*, Springer Berlin Heidelberg.

Larose, D. & Larose, C. (2014), *Discovering Knowledge in Data: An Introduction to Data Mining*, Wiley Series on Methods and Applications in Data Mining, Wiley.

Lee, M., Yun, J. J., Pyka, A., Won, D., Kodama, F., Schiuma, G., Park, H., Jeon, J., Id, K. P. & Jung, K. (2018), 'How to Respond to the Fourth Industrial Revolution , or the Second Information Technology Revolution? Dynamic New Combinations between Technology , Market , and Society through Open Innovation', *Journal of Open Innovation: Technology, Market, and Complexity* **4**(3).

Lengnick, M. (2013), 'Agent-based macroeconomics: A baseline model', *Journal of Economic Behavior and Organization* **86**, 102–120.

Leydesdorff, L. (2012), 'The triple helix of university-industry-government relations'.

Liebowitz, S. J. & Margolis, S. E. (1999), 'Path dependence', *Encyclopedia of law and economics* .

Lings, K. (2017), 'SA is experiencing an investment recession'.
URL: *https://www.moneyweb.co.za/moneyweb-opinion/south-africa-is-experiencing-an-investment-recession/*

Lockett, A. & Thompson, S. (2001), 'The resource-based view and economics', *Journal of Management* **27**(6), 723–754.

Lockett, A., Thompson, S. & Morgenstern, U. (2009), 'The development of the resource-based view of the firm: A critical appraisal', *International journal of management reviews* **11**(1), 9–28.

Louw, L. & Venter, P. (2013), *Strategic Management: Developing Sustainability in Southern Africa*, Oxford University Press Southern Africa.

Ly, M. M. (2019), South Africa's Economic Slowdown and Its Policy Options, Technical Report April, Policy Center for the New South.

Macal, C. M. & North, M. J. (2008), Agent-based modeling and simulation: ABMS examples, *in* 'Proceedings - Winter Simulation Conference', pp. 101–112.

Martin, R. & Sunley, P. (2012), 'The place of path dependence in an evolutionary perspective on the economic landscape'.

Masad, D. & Kazil, J. (2015), Mesa : An Agent-Based Modeling Framework, *in* 'Proceeding of the 14th Python in science Conference', number Scipy, pp. 51–58.

Mazorodze, B. T. & Siddiq, N. (2018), 'On the unemployment output relation in South Africa: a non-linear ARDL approach', *Journal of Economics and Behavioral Studies* **10**(5 (J)), 167–178.

Minar, N., Burkhart, R., Langton, C. & Askenazi, M. (1996), The Swarm Simulation System : A Toolkit for Building Multi-agent Simulations.

Napoletano, M., Delli Gatti, D. & Fagiolo, G. (2005), Weird ties? Growth, cycles and firm dynamics in an agent-based model with financial-market imperfections.

Nikolic, I. & Kasmire, J. (2012), *Theory*, Agent-Based Social Systems, Springer Netherlands.

North, M. J., Macal, C. M., Aubin, J. S., Thimmapuram, P., Bragen, M., Hahn, J., Karr, J., Brigham, N., Lacy, M. E. & Hampton, D. (2010), 'Multiscale agent-based consumer market modeling', *Complexity* **15**(5), 37–47.

OECD (2005), Oslo Manual: Guidelines for collecting and interpreting innovation data, Technical report, Organisation for Economic Co-operation and Development.

Pathria, R. K. (1996), *Statistical Mechanics*, 2nd edn, Elsevier Science.

Perelman, S. (1995), 'R&D, technological progress and efficiency change in industrial activities', *Review of Income and Wealth* **41**(3), 349–366.

Piętak, Ł. (2014), 'Review Of Theories And Models Of Economic Growth', *Comparative Economic Research* **17**(1).

Pokrovskii, V. N. (2014), 'Endogenous Technical Progress in the Theory of Economic Growth', *ISRN Economics* (ID928121).

Polking, J., Boggess, A. & Arnold, D. (2006), *Differential Equations: With Boundary Value Problems*, 2nd edn, Pearson Education.

Porter, M. E. (1985), *Competitive advantage : creating and sustaining superior performance*, Free Press.

Porter, M. E. (2008), 'The five competitive forces that shape strategy', *Harvard Business Review* **86**(1), 78.

Powell, W. W. & Snellman, K. (2004), 'The knowledge economy', *Annual Review of Sociology* **30**, 199–220.

Provost, F. & Fawcett, T. (2013), *Data Science for Business: What you need to know about data mining and data-analytic thinking*, "O'Reilly Media, Inc.".

Ravanfar, M. M. (2015), 'Analyzing organizational structure based on 7s model of mckinsey', *Global Journal of Management and Business Research* .

Reddy, V., Zuze, T., Visser, M., Winnaar, L., Juan, A., Prinsloo, C.H.Arends, F., Rogers, S. & F Arends, S. R. (2015), *Beyond Benchmarks*, Human Sciences Research Council.

Redl, C. (2018), 'Macroeconomic Uncertainty in South Africa', *South African Journal of Economics* **86**(3), 361–380.

Riccetti, L., Russo, A. & Gallegati, M. (2017), 'Financial Regulation and Endogenous Macroeconomic Crises', *Macroeconomic Dynamics* pp. 1–35.

Rode, S. (2012), *Advanced Macroeconomics*, Bookboon.

Romer, D. (2011), *Advanced Macroeconomics*, 4th edn, McGraw-Hill Education.

Romer, P. M. (1990), 'Endogenous Technological Change', *Journal of Political Economy* **98**(5).

Rothaermel, F. (2016), *Strategic Management*, 3 edn, McGraw-Hill Education.

Rubinstein, R. & Kroese, D. (2016), *Simulation and the Monte Carlo Method*, Wiley Series in Probability and Statistics, Wiley.

Saunila, M. & Ukko, J. (2014), 'Intangible aspects of innovation capability in SMEs: Impacts of size and industry', *Journal of Engineering and Technology Management - JET-M* **33**, 32–46.

Schwab, K. (2016), *The Fourth Industrial Revolution*, World Economic Forum.

Sharipov, I. (2015), 'Contemporary economic growth models and theories: A literature review', *CES Working Papers* **VII**(3), 759–773.

Shephard, R. (1970), *Theory of Cost and Production Functions*, Princeton University Press.

Siegfried, R. (2014), *Modeling and Simulation of Complex Systems: A Framework for Efficient Agent-Based Modeling and Simulation*, Springer Fachmedien Wiesbaden.

Siler, W. & Buckley, J. (2005), *Fuzzy Expert Systems and Fuzzy Reasoning*, Wiley.

Smith, S. S. (2020), Data As An Asset, *in* 'Blockchain, Artificial Intelligence and Financial Services', Springer, pp. 213–239.

Solow, R. M. (1956), 'A Contribution to the Theory of Economic Growth', *The Quarterly Journal of Economics* **70**(1), 65–94.

Sørensen, P. B. & Whitta-Jacobsen, H. J. (2010), *Introducing Advanced Macroeconomics: Growth and Business Cycles*, 2nd edn, McGraw-Hill Higher Education.

South Africa (2011), National Development Plan 2030. Our future - make it work, Technical report, National Planning Commission.

Stanlib (2018), The Weekly Focus: A Market and Economic Update, Technical Report 19 February, Stanlib.

Statistics South Africa (2019*a*), Inequality Trends in South Africa: A multidimensional diagnostic of inequality, Technical report, StatsSA.

Statistics South Africa (2019*b*), Mid-year population estimates, Technical report, StatsSA.

Steiner, G. (2010), *Strategic Planning*, Free Press.

Sum, N.-l. & Jessop, B. (2013), 'Competitiveness, the Knowledge-Based Economy and Higher Education', *Journal of the Knowledge Economy* **4**(1), 24–44.

Swan, T. W. (1956), 'Economic growth and capital accumulation', *Economic record* **32**(2), 334–361.

Teece, D. (2009), *Dynamic Capabilities and Strategic Management: Organizing for Innovation and Growth*, OUP Oxford.

Tregenna, F. & Tsela, M. (2012), 'Inequality in South Africa: The distribution of income, expenditure and earnings', *Development Southern Africa* **29**(1), 35–61.

Tsekeris, T. & Vogiatzoglou, K. (2011), 'Spatial agent-based modeling of household and firm location with endogenous transport costs', *NETNOMICS: Economic Research and Electronic Networking* **12**(2), 77–98.

Vadra, R. (2017), 'Knowledge economy in brics: a case of south africa', *Journal of the Knowledge Economy* **8**(4), 1229–1240.

Walker, P. (2016), *The Theory of the Firm: An overview of the economic mainstream*, Routledge Studies in the History of Economics, Taylor & Francis.

WEF (2018), The Future of Jobs, Technical report, World Economic Forum.

Weybright, E. H., Caldwell, L. L., Xie, H., Wegner, L. & Smith, E. A. (2017), 'Predicting secondary school dropout among south african adolescents: A survival analysis approach', *South African journal of education* **37**(2).

Wilensky, U. (1997), 'NetLogo Wolf Sheep Predation model.'.
URL: *http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation*

Wilensky, U. (1999), 'NetLogo'.
URL: *http://ccl.northwestern.edu/netlogo/*

Wilensky, U. & Reisman, K. (2006), 'Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories—An Embodied Modeling Approach', *Cognition and Instruction* **24**(2), 171–209.

Winter, S. G. (2003), 'Understanding dynamic capabilities', *Strategic Management Journal* **24**(10 SPEC ISS.), 991–995.

World Bank (2019), 'World Bank GDP Data'.
URL: *https://data.worldbank.org/country/south-africa*

World Bank Open Data (n.d.).
URL: *http://data.worldbank.org/*

Zhang, P. (2010), *Advanced Industrial Control Technology*, Elsevier Science.

Zott, C. (2003), 'Dynamic capabilities and the emergence of intraindustry differential firm performance: Insights from a simulation study', *Strategic Management Journal* **24**(2), 97–125.

Zwillinger, D. (1998), *Handbook of Differential Equations*, number v. 1, Elsevier Science.