The Art of the Possible

Tools and Methods for Solving Models with Substantial Heterogeneity

Tobias Grasl

Birkbeck College, University of London

Submitted for the Degree of PhD

April 24, 2017

Abstract

Macroeconomic models with rational, heterogeneous agents offer the opportunity to study both individual and aggregate economic outcomes, and the interaction between the two. Solving such models is difficult: the non-trivial problem of solving a maximisation problem in the presence of uncertainty is complicated by the need to determine model-consistent expectations in an economy with a non-degenerate distribution of agents over states.

This thesis provides both technical and mathematical tools which aid the economist in working with such models. Chapter 1 provides an introduction to the topic and discusses the literature.

Chapter 2 presents software libraries which automate some of the steps, for example calculating expectations from policy functions. The economist can focus on the code which implements the model-specific solution to the optimisation problem. The libraries are shown to solve models far more efficiently than a comparable solution coded in Matlab.

Chapter 3 introduces a new algorithm for calculating model-consistent expectations which relies on straightforward mathematics and a guess for the distribution of agents over states. The initial guess, the distribution obtained under constant aggregate conditions, yields good results. Multiple approaches for further improvement in the forecasting function are discussed. All solutions are computed using the libraries from chapter 2, and the algorithm is also implemented as part of those libraries for use in other models.

Chapter 4 discusses a model of the labour market with matching and large, heterogeneous firms. The firms experience idiosyncratic demand shocks and adjust their size in response. Steady state solutions are computed for different values of the exogenous tax rate and the transition path demonstrates that, in contrast to the canonical matching model, employment does not adjust instantaneously to changes in market conditions.

Chapter 5 discusses some avenues for potential future research.

For Mum, Anni, Trudi, Pella & David and Gigi, who together got me here.

I would like to thank Stephen Wright for giving me the opportunity to complete this work, and for advice and encouragement along the way. I would also like to thank my colleagues Alex, George, Marco and Patrick for discussions and companionship. I am grateful to the participants of workshops and sessions at CEF 2012 Prague and CEF 2013 Vancouver. Finally, I would like to thank my examiners, Alessandro Mennuni and Wouter den Haan, for an interesting viva and comments that were both helpful and encouraging.

All the work presented in this thesis is my own

Contents

List of Tables

List of Figures

1 Introduction

An economist has a difficult job. The world she seeks to understand is complex and intractable, so she must identify assumptions which improve tractability at an acceptable cost to realism. The model she constructs proves difficult, if not impossible, to solve, so she must apply further approximations to make the mathematics work. And finally, there are no off-the-shelf tools which perform the required computations, so she must develop them herself. She can not be just a 'dismal scientist', but must be a, most likely equally dismal, mathematician and programmer to boot.

This thesis investigates theoretical and practical tools for completing the third step, computation, in the context of macroeconomic models with substantial[1] heterogeneity. The models under consideration can be thought of as 'DSGE[2] models with heterogeneity', and importantly continue to assume model-consistency in expectation formation[3,4]. An early example in this literature is Krusell and Smith (1997).

Chapter 2 presents a computational toolkit, developed as part of this thesis, which abstracts from much of the technical difficulty of solving the model and allows the researcher to focus on the economics and mathematics. Chapter 3 introduces a new algorithm for solving such models in the presence of aggregate risk caused by exogenous shocks. Chapter 4 considers a model of the labour market in which firms experience idiosyncratic demand shocks and hence differ in size. This chapter motivates the research and provides some context in the literature, while Chapter 5 concludes.

1.1 Does heterogeneity matter?

1.1.1 Dynamic Stochastic General Equilibrium Models

Macroeconomics studies the behaviour of the economy in aggregate. The fact that this behaviour is caused by the behaviour of individuals and firms participating in the economy is obvious, but, despite the well known reference to 'animal spirits' in Keynes (1936), the actual behaviour of individual economic agents was largely abstracted away in much of the business cycle theory up to the 1960s.

[1]Substantial here meaning not just a predefined, finite set of different agents, but a potentially continuous range of heterogeneity in outcomes, albeit perhaps arising from a very limited amount of input heterogeneity.

[2]Dynamic Stochastic General Equilibrium

[3]The introduction of Krusell and Smith (2006) describes the type of model considered well, though in this thesis heterogeneity need not necessarily be among consumers.

[4]This contrasts with agent-based models in the tradition of Brock and Hommes (1997), which consider heterogeneity from a different starting point, with ex-ante heterogeneity in expectations. Whilst the latter approach is becoming increasingly popular in the literature, that literature has not been considered as part of this project. That is not to say that the work might not be useful for such models: the computational tools developed are fairly general and would help in solving any model in which the solution contains policy functions defined over a large domain, and the theoretical tools are designed to find model-consistent expectations, which, arguably, should be attributed to at least some agents in any theoretical economy.

Lucas (1976) argued that the expectations of households and firms matter. They will change their behaviour in response to policy changes affecting those expectations, but these effects are missed by models that do not take account of expectations. This argument proved powerful. Much of the subsequent business cycle literature, building on the methodology[5], if not always the economic assumptions, of Kydland and Prescott (1982), explicitly considers optimising, forward-looking economic agents. Models in this literature rest on the assumption that agents of each type satisfy the conditions for *exact aggregation*; as a result, the optimisation problem need only be solved for a single, *representative* agent. The conditions under which this is true come in both theoretical and mathematical guise: the assumption of complete markets in the sense of Arrow-Debreu implies full insurance against idiosyncratic risk, so that agents[6] always have identical state; or the model may be parametrised in such a way that individual policies aggregate mathematically to a function which solves the same optimisation problem [7].

1.1.2 Investigating the Impact of Heterogeneity

The DSGE approach has come under much criticism, perhaps most damagingly for its apparent failure to provide practical tools to aid in determining monetary or fiscal policy (Mankiw, 2006). A potential weak point that was investigated from the earliest days[8] is the assumption of a representative agent: is it justified, or do models taking account of income and wealth heterogeneity reach different conclusion than those that do not? In many cases answering this question requires solving a model which includes heterogeneity[9].

One difficulty in solving models with heterogeneous agents and aggregate fluctuations is that, in theory, the entire distribution of agents over their states forms an input to each agents' optimisation problem. The problem is infinite dimensional, hence impossible to solve with

[5]Today termed dynamic general equilibrium (DSGE) modelling.

[6]In this case, households.

[7]On the consumption side, households are commonly assumed to have utility functions satisfying the Gorman form, though this assumption does not take account of liquidity constraints

[8]Ríos-Rull (1995) surveys the early literature.

[9]See İmrohoroğlu (1992) for example.

known methods. Ríos-Rull (1995) surveyed the early literature which attempts to construct macroeconomic models with heterogeneous agents. He found that these models avoided the problem of infinite-dimensionality by either imposing the distribution from the outside or abstracting from capital. Krusell and Smith (1998) introduced a method to circumvent the problem in their model: current aggregate states are sufficient to forecast future aggregates, and hence all variables which directly affect household choices, with a high degree of accuracy. They termed this finding 'approximate aggregation'. That aggregate law of motion is found recursively and jointly with the individual decision rules.

A possible interpretation of the findings of Krusell and Smith (1998), and more so the work of Aiyagari (1994) which they build on, is that heterogeneity does not significantly affect the behaviour of aggregate economic variables[10]. This interpretation is rejected by Carroll (2000), who reached this conclusion by comparing a representative consumer model with a version of Krusell and Smith (1998) calibrated to reproduce key microeconomic facts. The representative consumer model can not replicate the empirical aggregate marginal propensity to consume, whereas the version with heterogeneous households provides a much closer fit. The author pronounces a "Requiem for the Representative Consumer" and declares that '…for many purposes, the representative-consumer model should be abandoned in favour of a model that matches key microeconomic facts'. That language is relatively strong for academic economics, and the significance of the result should not be understated: in the most parsimonious business-cycle model possible, introducing a realistic degree of heterogeneity significantly affects policy- and welfare-relevant aggregate relationships.

In an extension of the model described above which adds indivisible labour supply at the individual level, An et al. (2009) demonstrated that a representative agent model can only replicate the behaviour of the economy with heterogeneous agents if household utility is allowed to be non-concave or unstable.

[10]The authors of the former paper invested some effort into finding a calibration which does produce a degree of wealth heterogeneity close to the empirical level, and were careful to document that the aggregate outcomes, particularly the covariance of consumption and output, do change by a significant margin under that calibration.

1.1.3 Developing the Techniques

The solution introduced by Krusell and Smith (1998) works by repeatedly executing two steps in succession until convergence, starting from an initial guess for the aggregate forecasting function:

1. Calculate the individual policies given the current assumption on the aggregate forecasting function

2. Update the aggregate forecasting function given the new individual policies

Since their contribution, the numerical work on solving models with substantial heterogeneity has continued.

The difficulty in computing rational expectations under aggregate uncertainty is just one of the issues arising when solving models with substantial heterogeneity. A further difference to representative agent models is that the individual agents' policies must be calculated across the entire feasible domain of their state variables, often including state boundaries. Representative agent models are generally only solved in a comparatively small neighbourhood of the steady state. Carroll (2006) introduced a straightforward and efficient method to performing this aspect of the algorithm.

Young (2010) uses a discrete representation of the wealth distribution, rather than a large number of individuals, to simulate an economy with heterogeneous households. This removes one of the potential issues encountered in the alternative approach, namely that the distribution of individual stochastic shock realisations encountered during simulation affects the outcome.

Den Haan and Rendahl (2010) update the aggregate forecasting function by approximating the individual transition rule with a (piece-wise) linear function, so that aggregation of the function equates to calculating the function value at the average capital holdings. This does away with the need for simulation altogether, but requires the introduction of an additional aggregate state variable so that the capital held by employed and unemployed agents can be tracked separately, because the individual transition rules differ between the two groups.

The approach of Reiter (2010) is conceptually close to the approach presented in Chapter 3 in that he uses a discrete representation of the steady state distribution from the model without aggregate uncertainty as a reference to perform the aggregation step. He independently adjusts the reference distribution to match each point on a grid over current aggregate states and then finds the forecast for future aggregate states at that point using fixed-point methods.

An approach that does not follow the outline above but is also related to that introduced in Chapter 3 is Reiter (2009). He first takes a discrete approximation of the steady state distribution under the assumption of no aggregate uncertainty. He then considers each of the points in this discrete approximation as a separate variable, derives a system of equations that describes the economy as a function of these variables and finds a first-order approximation of that solution using a perturbation approach.

1.2 Accompanying Code

This thesis presents software libraries and solves models using those libraries. Source code for all the software discussed, as well as the model solutions, are available online. The website Grasl (2016) (https://modelsolver.bitbucket.io/phd/) contains links and compiled packages.

2

The ModelSolver Toolkit

The process of theoretical analysis of an economy commonly involves three broad steps: constructing the model, solving the mathematical problem and computing the solution to the equations obtained. An economist is taught how models are constructed and how the mathematics is solved from her earliest days in the field. Computing the solution to an economic model is arguably the least standardised step in the process. With a few exceptions (for example Dynare, Adjemian et al., 2011; Carroll et al., 2016)[1] the code to solve a model must be

[1]The latter of these packages came to the author's attention as this thesis went to press. It shares many common aims with the ModelSolver Toolkit presented in this chapter.

hand-crafted using whichever languages and libraries the economist finds most readily accessible. This is time-intensive at an individual level and inefficient for the profession, causing duplication of effort and limiting knowledge sharing.

This chapter presents a toolkit which provides solutions to some of the computational tasks the economist has thus far had to program herself. At conception, the target audience of the toolkit (a.k.a. the author) was an economist working with macroeconomic models with substantial[2] heterogeneity. Whilst this target audience remains relevant, the tools provided are more widely applicable.

Insofar as it is written in Scala[3], a language few economist are familiar with, the toolkit is open to the criticism of being yet another different and inaccessible approach, rather than a widely usable tool. Whilst the claim that economists will (and should!) be reluctant to learn another technology has merit, it is also the case that there are no obvious better candidates. Models of the kind targeted are too diverse to allow for a completely programming-free approach, so the toolkit must provide an accessible language in which to implement the model. An economist should not be expected to learn a lower level language such as C++ or FORTRAN, and initial research for this project showed that more specialised tools such as MATLAB and Mathematica perform very poorly when faced with problems of this kind[4]. Scala has some significant points in its favour: it provides enough syntactic sugar to be easily accessible; it runs on a platform[5] that is mature, performant and free to deploy; and it is widely used outside of academia, so that both good, free development tools and budding economist-programmers are available. The examples in this chapter, as well as the code to solve the models in subsequent ones, illustrate the ease-of-use of the language in conjunction with the toolkit.

[2]Substantial here meaning not just a pre-defined, finite set of different agents, but a potentially continuous range of heterogeneity in outcomes, albeit perhaps arising from a very limited amount of input heterogeneity.

[3]Much of the underlying logic is actually implemented in Java, but Scala provides a good platform for accessing that Java code whilst using idioms more familiar to users of higher-level languages.

[4]This point was recently documented by Aruoba and Fernández-Villaverde (2014)

[5]The Java Virtual Machine

2.1 The Benchmark Model

This section introduces the model used in subsequent sections first to illustrate the concepts on which the ModelSolver Toolkit builds, and later to evaluate the toolkit. The model is a variation of the stochastic growth model (see, for example, Taylor and Uhlig, 1990).

2.1.1 The Production Technology

The economy is a production economy with competitive goods, labour and physical capital markets. Firms in the economy face the production function

$$Y_t = A_t K_t^{\alpha} (P_t \bar{l} L_t)^{1-\alpha} \tag{2.1}$$

where

$$P_t = g P_{t-1} \tag{2.2}$$

Here, Y_t is output per period, A_t is the exogenous aggregate productivity process, and P_t is labour-augmenting productivity growth with growth rate g. K_t is aggregate capital, L_t is employment and $\bar{l}$ is the time endowment per employed person.

Firms hire capital and labour to maximise profits each period. The firms' first order conditions yield a rental rate of capital, r_t, and wage per unit of time worked, w_t, of

$$r_t = \alpha A_t \left(\frac{K_t}{P_t \bar{l} L_t} \right)^{\alpha - 1} \tag{2.3}$$

$$w_t = (1-\alpha) A_t P_t \left(\frac{K_t}{P_t \bar{l} L_t} \right)^{\alpha} \tag{2.4}$$

2.1.2 Households

The economy is populated by a continuum of infinitely-lived households of measure one, indexed on the unit interval. Each household i seeks employment and, when employed, supplies

$\bar{l}$ units of labour per period. The household receives income e_t^i, which depends only on variables exogenous to its decision.

There is one asset, production capital. Households have identical utility functions and maximise expected lifetime utility subject to their budget constraint. Household i's problem is thus

$$\max_{\{c_t^i, k_{t+1}^i\}_{t=0}^{\infty}} E\left[\sum_{t=0}^{\infty}\left(\beta^t \frac{(c_t^i)^{1-\gamma}-1}{1-\gamma}\right)\right] \tag{2.5}$$

$$\text{s.t. } c_t^i + k_{t+1}^i = (1 + r_t - \delta)k_t^i + e_t^i \tag{2.6}$$

$$k_{t+1}^i \geq 0 \tag{2.7}$$

where c_t^i is consumption, k_t^i is individual capital holdings. β is the per-period discount rate, γ is the coefficient of relative risk aversion and δ the depreciation rate of physical capital.

Solving the households' maximisation problem yields first-order condition

$$\beta E\left[\left(c_{t+1}^i\right)^{-\gamma}(1 + r_{t+1} - \delta)\right] = \left(c_t^i\right)^{-\gamma} - \phi_t^i \tag{2.8}$$

where $\phi_t^i \geq 0$ is the multiplier on the borrowing constraint.

2.1.3 The State of the Economy

Each household's state at the beginning of period t is capture by two variables: its exogenous income e_t^i and its endogenous capital holdings k_t^i. There is only one stochastic state which is independent of households: A_t, the exogenous aggregate productivity process.

The state of the economy, $\mathbb{S}_t$, is therefore given by

$$\mathbb{S}_t = (\omega_t, A_t) \tag{2.9}$$

where ω_t is the distribution of households over capital and employment:

$$\omega_t = \omega_t : (0,1) \to \mathbb{R}^2_+ \equiv \{(k^i_t, e^i_t) : i \in (0,1)\} \tag{2.10}$$

2.1.4 Uncertainty and Insurance Markets

The description of the economy to this point is incomplete in that it does not specify the nature of the income process or the insurance markets available to households. The model will later be solved in a variety of configurations for both. Until they are defined, the assumption should be the most general one, namely that households may, for whatever reason, find themselves in differing states at any given point in time.

2.1.5 The Solution

Each household has a single endogenous choice: the consumption-savings decision. This choice is conditional on the state of the economy, $\mathbb{S}_t$, as well as the household's own state (k^i_t, e^i_t). It can therefore be expressed as

$$k^i_{t+1} = f(k^i_t, e^i_t, \mathbb{S}_t) \tag{2.11}$$

Given the current state of the economy $\mathbb{S}_t$, this function yields all future household wealth levels k^i_{t+1} and therefore, in conjunction with future exogenous states, the future state $\mathbb{S}_{t+1}$. Finding f thus solves the model.

2.2 Concepts and Algorithms

The ModelSolver Toolkit does not aim to introduce new concepts and algorithms, but to facilitate the use of those already documented in the literature. This section outlines the theoretical building blocks that form the basis for much of the functionality the ModelSolver provides.

2.2.1 Bounded Rationality and Approximate Aggregation

The benchmark economy has infinitely many households, which may be in different states. The state of the economy comprises the state of all its constituent households and is therefore infinite dimensional. A rational, forward-looking household includes this state in its decision making and must form expectations over its future value that are consistent[6] with the behaviour of the economy. Though the household may be capable of doing so, an economist modelling the household is as yet not provided with tools for such a task.

A fact which may simplify the necessary calculations is that a household's economic outcomes are not affected directly by every other household's choices. Instead, the consumption-savings decisions of all households jointly give rise to aggregate capital and hence the interest rate and wage. The latter two are the only endogenous variables not specific to the household that appear in its optimisation problem. This is a key aspect of a market economy, and it reduces the number of variables a household considers explicitly to a small number, albeit that forecasting those variables exactly still requires knowledge of all households' actions.

The conjecture of Krusell and Smith (1998) was that a small set of aggregate variables, including but not necessarily limited to aggregate capital, forms a sufficient set of information to forecast its own future state with high precision. Boundedly rational (see Simon, 1972) households choose to restrict the amount of information they consider to that set of aggregate variables and can take advantage of the accurate forecast to reduce their problem to a tractable one. The papers' results confirmed the authors' conjecture. They termed this feature of the economy approximate aggregation.

The ModelSolver Toolkit's principal assumption is that agents are boundedly rational and use a small number of endogenous aggregate states in their decision making, ignoring other agents. As a consequence, approximate aggregation is also assumed to hold: if the forecasts formed using the approach are not very accurate then the assumption of bounded rationality may also not be appropriate.

[6]Consistent in the sense that the probability assigned to any possible future outcome is that implied by the model (Muth, 1961).

2.2.2 Alternating Solution

The assumption of bounded rationality renders the households' problem finite-dimensional and hence tractable, but an issue of circularity remains. Solving the household optimisation problem requires, as input, a function that forecasts future aggregate capital based on current values. But this function is itself an aggregation of all the savings choices made by individual households, so that computing it requires, as input, the solution to the household optimisation.

Krusell and Smith (1998) acknowledge this circularity and integrate in into their solution algorithm. They start with an initial guess for the capital forecasting function, and then iterate over two steps until the solutions have converged:

1. Solve the household problem, using the current guess of the forecasting function as input
2. Update the guess of the forecasting function using the latest outcome of step 1.

In theoretical terms, the ModelSolver Toolkit's overall algorithm maintains this approach. It can also be replicated precisely.

2.2.3 Interleaved Iteration

The description above of the two steps of the algorithm does not explain how these steps are performed. Multiple approaches to both steps are documented in the literature, and indeed Chapter 3 proposes a new way of performing step 2. The original authors perform step 1 by value function iteration, and simulate a large number of periods in step 2 in order to generate data from which the forecasting function can be estimated. Refinements to both approaches, such as Carroll (2006) for step 1 and Den Haan and Rendahl (2010) for step 2, exist, and will be discussed in more detail below.

If one of the steps is itself recursive, a potential (compute) optimisation[7] is to refrain from forcing that step to converge fully in each iteration of the overall loop. A partially converged

[7]This optimisation is present in the code provided with Den Haan and Rendahl (2010), though it is not mentioned in the paper explicitly.

solution may already be close enough to the actual solution to improve the other step's result relative to the last iteration.

The effect of this approach is to change the balance of how many times each of the steps is executed. Its impact depends on the relative computational cost of those steps. The simulation-based solution to step 2 of Krusell and Smith (1998) is costly, so although step 1 is iterative, using this approach would not constitute an optimisation. In the algorithm of Den Haan and Rendahl (2010) step 2 is computationally trivial, so that the approach yields large gains.

The ModelSolver Toolkit performs one overall iterative loop. In each iteration of this loop it delegates to the model-specific code the decision to perform step 1 (first) and then step 2. Potential decision criteria are legion: most straightforwardly, both steps might be performed in each iteration; the number of iterations performed might be decisive; more complex criteria include measures of the degree of convergence. For example, the solution of the benchmark model in the calibration of Den Haan and Rendahl (2010) performs only step 1 in the first 100 iterations of the loop, but both step 1 and step 2 in each subsequent iteration.

2.2.4 Rectangular Grid Approximation

The ModelSolver approximates functions it is solving for using spline interpolation on a rectangular grid. In this approach, a grid of coordinates over the domain of the function is constructed and a multi-dimensional array is used to hold the function values on the grid. Since the grid is rectangular, the grid coordinate values along each dimension are constant across the other dimensions, so that the coordinate system is a list of one-dimensional arrays. The n-th member of this list holds the values of the n-th input variable (i.e. x_n) on the points of the grid traversing that dimension. Each element in the function value array represents a point on this grid, and the value held at that position is the value of the function at that point.

Continuous Variables

Most input variables to the functions are continuous variables, whereas the grid is by definition discrete. Off-grid function values are evaluated by interpolation. Currently, only linear interpolation is available, yielding a piece-wise linear approximation.

Discrete Variables

Some models use discrete variables, particularly for exogenous states. This may be due to the discrete nature of the quantity being modelled, such as ‘employment status’, or due to a modelling decision, commonly made to decrease computational cost. The latter is the case with aggregate shocks in the benchmark model, for example.

The possible values of such variables are used as the grid coordinates in that dimension. No interpolation is required for these variables.

2.3 The Numerics Library

The ModelSolver Toolkit consists of two libraries: the Numerics library and the ModelSolver library. The latter contains the functionality specific to solving models, along with algorithm implementations and other code specific to economics. It is the subject of the next section. This section presents the Numerics library.

Scala and Java are all-purpose programming languages and are not designed with numerical computation in mind. They therefore lack some programming constructs which are useful in numerical applications. The Numerics library provides these constructs. Much of the code written to solve models using the toolkit will utilise the Numerics library heavily, whilst the ModelSolver library will form the scaffolding that structures the application, but will not frequently be called directly.

The source code for the numerics library is available online (Grasl, 2011b, 2014d).

2.3.1 Multi-Threading

Numerical computations, run on modern computers that have multiple cores, can benefit greatly from multi-threading. The Numerics library takes extensive advantage of this, distributing calculations across multiple threads, where possible transparently. This applies particularly to array operations, and is one of the key reasons why the performance of the library is better relative to manual, loop-based approaches.

Updating shared resources in multi-threaded operations is problematic, and transparency can be an impediment in such cases. The documentation for specific functions below identifies those that are multi-threaded.

2.3.2 `DoubleArray`

The `DoubleArray` type is at the root of the Numerics library, both in the sense of being causal for its creation and the foundation upon which the rest of the library is built.

A `DoubleArray` is an `n`-dimensional, regular array of double values. Both languages used to construct the library already provide such a concept, but these implementations do not easily support some of the primary use cases the ModelSolver requires: applications where the number of array dimensions is not known at design-time; applying functions element-wise to all the members of one or more arrays; and performing operations on sub-arrays in selected dimensions.

More specialized languages, such as Octave, do provide "point-wise" operators which can be applied to each point in an array and even methods for applying functions element-by-element to one or more arrays. Working with arrays when the number of dimensions is unknown ex ante remains difficult. Performance and memory requirements also cause difficulties. Aruoba and Fernández-Villaverde (2014) compare the execution time of a common economic problem in a number of languages and find MATLAB and Mathematica among the slowest options.

Language designers face trade-offs when choosing how to implement arrays (Bezanson et al., 2014a). One of the choices commonly made by general purpose languages, including Java and Scala, is that the rank[8] of an array must be known when the program is compiled. This is not conducive to writing applications that may require variable rank. The ModelSolver is just such an application because the number of model variables, which determines the rank

[8]Number of dimensions

of arrays needed to represent model functions, depends on the model. The `DoubleArray` type supports working with arrays of unknown rank.

This section documents the most important features of `DoubleArray`. Comprehensive documentation is provided in the form of Javadoc and Scaladoc documentation that forms part of the toolkit.

2.3.2.1 `createArrayOfSize(n1, n2,...)`

`n1,n2,...`	`Int`, number of elements in each dimension
`returns`	A new `DoubleArray` of the requested size

Creates and returns a `DoubleArray` of the given size. Note that the size of the array is fixed at creation and cannot be changed subsequently.

2.3.2.2 Indexing

Zero-Based

`DoubleArray` uses zero-based indexing, so the first value in an array a is $a(0)$.

Arrays as Indexes

`DoubleArray` allows arrays of length unknown at compile-time and so must allow indexes of variable length. Integer arrays serve as such indexes.

Example

```
val a = createArrayOfSize(5,4,3)
val idx = Array(1,2,3)

a(idx) // Equivalent to a(1,2,3)
```

Linear Indexing

In some situations it is beneficial to be able to identify an element in an array without using its full index. Linear indexing serves this purpose. In linear indexing, a single integer is passed and can identify any point in an array. Whenever a single index is used on an array, linear indexing is applied.

In an array of n dimensions of length $D_1, D_2, \cdots, D_n$ respectively, the linear index of index $i_1, i_2, \cdots, i_n$ is given by

$$\sum_{i=1}^{n} \left(i_n \prod_{j=i+1}^{n} D_j \right) \tag{2.12}$$

Note that the order of dimensions is reversed from that in MATLAB.

Example

```
val a = createArrayOfSize(5,4,3)

a(0)  // Equivalent to a(0,0,0)
a(1)  // Equivalent to a(0,0,1)
a(20) // Equivalent to a(1,2,2)
```

Sub-Array Selection

The character $ has a special meaning within an index. If a $ is passed in any position, then the function result will be the sub-array at those index points where positive integers where passed, but with the other dimensions unrestricted.

Example

```
val a = createArrayOfSize(5,4,3)
```

```
a($,1,1)  // Returns a 1-D array of size 5
```

Returns a one-dimensional `DoubleArray` containing the elements of `a` along its first dimension at the second point in its second and third dimensions.

2.3.2.3 Basic Arithmetic Operators

The `DoubleArray` supports the basic arithmetic operations, both as element-wise operations with other `DoubleArray` instances and with `Double` instances. If the operands are both `DoubleArray` , then they must be of identical size. If one of the operands is a `Double` , the operation is again element-wise but that operand is the same for all array elements.

Example

```
a + b // Adds two arrays element-wise
a/d   // Divides each element of A by the Double d
d + a // Double can be the first operand
```

In addition, C-style compound assignment operators that modify their first operand are also supported.

Example

```
a -= b // Modifies a to contain a-b (like a = a-b)
a *= d // Modifies a to contain a*d
```

2.3.2.4 `a << (d1,d2,...)`

`d1,d2,...`	Double values to be filled into the array

Fills array `a` with the provided values, in linear indexing order. There must either be as many input values as there are elements in `a`, or one. In the latter case, each element of `a` is set to `n1`.

Example

```
val a = createArrayOfSize(2,2)

a << (1 to 4)  // a contains (1,2
               //             3,4)
```

2.3.2.5 a -> fn

`fn`	A function that maps a `Double` to another `Double`
`returns`	A `DoubleArray` off the same size as `a`

Performs an element-wise mapping of the array `a` through function `fn`, so that each element in the returned array is the result of applying the function `fn` to the value at the same position in `a`.

This operation is performed in multiple threads, and no guarantees are made which points will or will not share the same thread. Writing to shared resources in `fn` should be avoided.

Example

```
val a = createArrayOfSize(2)
a << (1,4)
a -> (ai => ai*10 + sqrt(ai))  // Returns (11,42)
```

This results in an array with elements $(11, 42)$.

2.3.2.6 a ->= fn

`fn`	A function that maps a `Double` to another `Double`
`returns`	`a`

Performs `->` and stores the result in `a`.

2.3.2.7 **`(a1 :: a2 ::. . . :: an) -> fn`**

`a1,a2,. . . ,an`	`DoubleArray` instances of the same size
`fn`	A function of n `Double` which returns an array of m `Double`
`returns`	An array of m `DoubleArray` instances of size `a1.size`

Performs and element-wise mapping of the elements at the same positions in each of the input arrays through the function `fn`.

This operation is performed in multiple threads, and no guarantees are made which points will or will not share the same thread. Writing to shared resources in `fn` should be avoided.

Example

```
a << (1, 2) // Assumes A,B,C are all 2-element DoubleArray
b << (2, 3)
c << (3, 5)

val Array(d, e) = (a :: b :: c) :-> ((ai, bi, ci) => {
  Array(( ai + bi) * ci, (ai + bi) / ci)
})
```

`d` now contains elements $(9, 25)$ and `e` $= (1, 1)$.

2.3.2.8 **`(a1::a2::. . . ::an) =-> fn`**

`a1,a2,. . . ,an`	`DoubleArray` instances of the same size
`fn`	A function of n `Double` which returns a `Double`
`returns`	`A1`

Performs `->` and stores the result in `a1`.

2.3.2.9 **`a.reduce( fn )`**

`fn`	A function of type `Iterator[Double] => Double`
`returns`	A `Double`

Returns the result of passing an iterator over the entire array, in linear-index order, to `fn`.

Example

```
val a = createArrayOfSize(10) << (1 to 10)


val res = a.reduce( (it:Iterator[Double]) => {
  var s = 0d
  for( i <- it ) s = s+i
  s
})
```

`res` now contains 55.

2.3.2.10 **`A\(d1,d1,...)`**

`d1,d2,...`	`Int` zero-based dimensions across which to apply the mapping
`returns`	See below

The returned value can be uses as the first operand in arithmetic operations, including compound assignments, `::` and `reduce`. The effect is that the subsequent operations will be applied repeatedly, once for each point in the dimensions *not* specified. If `A.size = s`, where `s` is an `Array[Int]`, then subsequent operands added with the `::` operator must have size `(s(d1), s(d2),...)`.

These subsequent operations are performed in multiple-threads. The usual proviso applies.

Example

```
val a = createArrayOfSize(2,2)
val b = createArrayOfSize(2)

a << (1 to 4)
b << (5,6)

val c = (a\(1) :: b) -> ((ai,bi) => ai + bi) //add b to a across dim 1
// c = (6,8
//      8,10)
val d = (c\(0)).reduce( (iter : Iterator[Double]) => {
  iter.reduce( (prod,ci) => prod * ci)  // Product of elements
})
```

`c` now contains `(6,8; 8,10)`, and `d` is `(48, 80)`.

2.3.3 Interpolation

The iterative methods used in solving models by grid-projection often require results that lie off the grid to be bought back onto the grid (see Carroll, 2006). This is achieved by interpolation. Because the grid can consist of 100000s of points, and these operations happen multiple times per iteration, efficient interpolation is key in keeping the time required to solve a model down.

The methods provided by this library currently all perform linear interpolation. To date this has been found to be sufficient. Alternative approaches could be implemented transparently since they do not affect the method signature. In economic problems the shape of the function, particularly its monotonicity, concavity and boundaries, are often important, so care must be taken to preserve these values where necessary.

2.3.3.1 Arrays as Function Values

A common use of `DoubleArray` in the ModelSolver Toolkit is to act as an interpolant: the array represents the values of a function $f(x_1, x_2, \cdots, x_n)$ at the points on a rectangular grid. Each array dimension d represents one of the input variables x_d, and each point i in that dimension represents a known value x_d^i of x_d. The x_d^i change monotonically in i. Determining the value of the function at off-grid points requires interpolation along each dimension. The Numerics library provides n-dimensional interpolation to support this approach.

2.3.3.2 `spec(x, a, v)`

d	`Int` The dimension to interpolate
x	`DoubleArray` The values of input variable x_d at the grid points along dimension d
v	`Double` The target value of x_d to interpolate to

Returns an `Interpolation.Specification` which specifies the parameters for a single dimension of the interpolation to be performed. This specification is used in `interp` below.

2.3.3.3 `interp(a, sp1, sp2,...)`

a	`DoubleArray` to be interpolated
sp1,sp2,...	`Interpolation.Specifications` detailing operation to be performed
returns	The result of the interpolation

Interpolates `a` according to the specifications `sp1,sp2,...`. The result will be of dimension `a.numberOfDimensions - length(sp1,sp2,...)`, with the remaining dimensions of equal size and in the same order as those dimensions of `a` not interpolated.

Example

```
val a = createArrayOfSize(5,7,3,2)
val x0 = createArrayOfSize(5) << (1,2,3,4,5)
val x2 = createArrayOfSize(3) << (3,6,9)
a << (1 to a.numberOfElements) // Fill A with some values
val b = interp(a, spec(0,x0,2.5), spec(2,x2,4))
```

After this operation, `b.size` is `(7,2)`.

2.3.3.4 Interpolating many values

Reverse-time iterative methods, such as the method of endogenous gridpoints, commonly yield functions defined at off-grid points in one dimension as the outcome of an iteration. The on-grid function values must then be determined for the entire grid before commencing the next iteration. If the function is monotonous, commonly the case in well-behaved economic problems, an optimisation is possible: the interpolation can be performed by iterating just once across both the grid and the function values in parallel, computing the result for each target value along the way. A generic interpolation mechanism, unaware of the target's monotonicity, would require that each point be treated independently.

2.3.3.5 **`interpolateFunction(srcX,srcY,d,tgtX, params)`**

`srcX`	`DoubleArray` x values of the interpolant. Must be monotonous along dimension `d`
`srcY`	`DoubleArray` y values of the interpolant, same size as `srcX`
`d`	`Int` dimension along which to interpolate
`tgtX`	`DoubleArray` x values to which to interpolate. Must be monotonous along `d`
`params`	*Optional* `Params` object with additional parameters.
`returns`	DoubleArray

Takes `srcX` and `srcY` to be input and output values respectively of a function, and uses this function as an interpolant to determine the output values at `tgtX`.

Interpolation is only performed along one dimension, `d`, and both source and target x values must be monotonous, with the same direction of monotonicity, in this dimension.

If the input function is multi-dimensional then the interpolation is effectively performed repeatedly for each point in dimensions other than `d`. Neither source nor target x values need be identical for each of these points.

`tgtX` may be either one-dimensional or have the same number of dimensions as `srcX`. In the former case, the same target x values are evaluated for each point in the non-`d` dimensions of `srcX`. In the latter case, the non-interpolated dimensions of `tgtX` must be of the same size as those of `srcX`. In both cases the size of the target along the interpolated dimension is not restricted.

The resulting array will be of the same size as `srcX` in the non-interpolated dimensions, and the same size as `tgtX` in the interpolated one.

Params

Both `interpolateFunction` and `interpolateFunctionAcross` accept an optional parameter of type `Params`. The additional options which can specified using this parameter are:

`constrained`

If `constrained` is set, then the domain of the hypothetical function being interpolated is assumed to be constrained at the lower value of the source x values. Any target x values lying below[9] this value are set to have the same y value as the first x value.

If constrained is not set, extrapolation is performed to find appropriate y values for target x values lying below the first source x value.

[9]'Below' here assumes that the interpolant is increasing. For decreasing functions, this would be 'above'. Directions in further statements are similarly assuming an increasing function and must be reversed in the opposite case.

`withDiscontinuities(discontinuities)`

`discontinuities`	a function of type `(Array[Int])=>Iterator[Double]`

The function `discontinuities` provides, for each index in the non-interpolated dimensions, an iterator over the discontinuities present in the function being interpolated.

Interpolation in the presence of discontinuities is discussed in Section 2.3.3.7.

`preserveMonotonicity`

This parameter, applicable only in the presence of discontinuities, specifies that the interpolation should not result in non-monotonicity around the discontinuities. See Section 2.3.3.7.

Example

```
val srcX = createArrayOfSize(5,2)
val srcY = createArrayOfSize(5,2)

srcX\0 << (1 to 5)
srcY\0 << (2 to (10,2)) // so y = 2x ...
srcY\1 += createArray(2,3) // ...+ 2(3) along column 0(1)

val targetX = createArrayOfSize(10,2)
targetX\0 << (5 to (50,5))

val res = interpolateFunction(srcX, srcY, 0, targetX)
// res contains y for x=(5,10,15,...,50) in both columns
```

2.3.3.6 `interpolateFunctionAcross(srcX,srcY,d,tgtX,params,d1,d2,`

`d1,d2,...`	Dimensions of tgtX across which to replicate the interpolation
`returns`	A DoubleArray

The parameters shared with `interpolateFunction` above are identical. The difference between that method and this one is that this one allows `tgtX` to be of higher dimension than `srcX`, and that the interpolation will be repeated for each grid point in those additional dimensions. These additional dimensions must be specified as `d1,d2,...` and need not be successive or be the highest dimensions of `tgtX`.

2.3.3.7 Interpolation with Discontinuities

Commonly, and as a prerequisite for applying the mathematics necessary to solve a model, functions encountered in economics are assumed to be both continuous and differentiable. Such assumptions sometimes break down. An approximate solution to the model being analysed may nonetheless be feasible provided the failure of the assumptions is limited to a manageable set of points.

Since the ModelSolver evaluates functions by interpolation, the interpolation mechanism must be able to deal with points where continuity fails - in other words, with discontinuities. That is precisely what the parameter `withDiscontinuities` described above achieves.

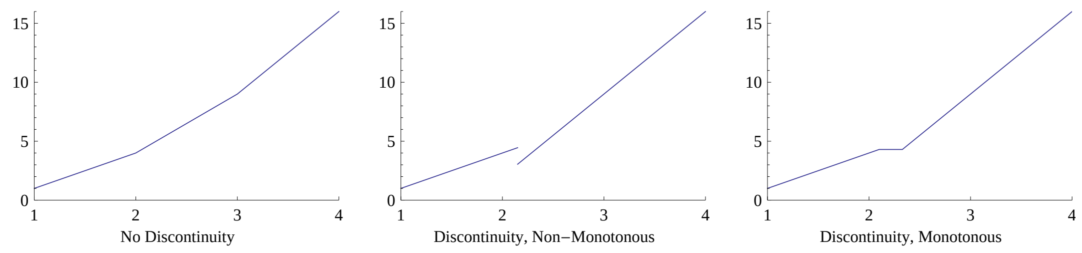

Figure 2.1: The Effective Shape of the Interpolant

The parameter identifies points along each line of interpolation[10] at which the function should not be assumed to be continuous. When the interpolation is performed, any target `x` value which lies next to such a discontinuity, with no intervening points in `srcX`, will be calculated *as if* the source function did not extend beyond the discontinuity. The resulting y value will be found by extrapolation from the part of the source function lying prior to the discontinuity.

Monotonicity

Many economic functions are assumed or required to be monotonous[11]. This monotonicity is often important order to be able to find a solution to a model. The interpolation mechanism must hence ensure that this property is preserved where necessary. In the presence of discontinuities this requires special care.

Figure 2.1 illustrates the effect of the two parameters on the function defined by `srcX=(1,2,3,4)`, `srcY = (1,4,9,16)`. On the left there are no discontinuities. The central case has a discontinuity at $x = 2.15$, but does not preserve monotonicity. The right-hand graph illustrates the same discontinuity, but with monotonicity being preserved. Note that this last case depends also on which target points prior to the discontinuity are interpolated, because the last such point determines the lowest possible y value attainable after the discontinuity.

2.4 The ModelSolver Library

The ModelSolver library builds on the Numerics library and aids a researcher in solving economic models. It performs those tasks which are generic and provides the structure within which the remaining, model-specific tasks are embedded. The source code for the ModelSolver library is available online (Grasl, 2011a, 2014c).

[10]The methods using this approach only perform 1-dimensional interpolation.
[11]Often because they are the derivative of convex or concave functions.

2.4.1 How to Solve a Model

The overall solution algorithm, shown in Algorithm 2.1, is straightforward: the configuration is set up based on information such as command line parameters; a model is configured using that configuration; state and solvers are initialised; finally, solvers are repeatedly called upon to update the state until the convergence criteria are satisfied.

Algorithm 2.1: The Overall Solution Algorithm

```
// Initialise configuration from inputs etc.
modelConfig = modelRunner.createConfig()

// Configure the model
model.setConfig( modelConfig )

// Get the initial state
state = model.initialState

// Get (configured) individual and aggregate solvers
individualSolver = model.individualSolverInstance
aggregateSolver = model.aggretaeSolverInstance

repeat
  // Perform an individual iteration
  individualSolver.performIteration( state )

  // If the model says so...
  if( model.shouldUpdateAggregates( state ) )

    // ...update aggregate forecasts
    aggregateSolver.updateAggregateTransition( state )
  end

// Stop when the convergence criterion is smaller
// than the target
until(state.getConvergenceCriterion < target)
```

This overall algorithm is executed by code provided by the ModelSolver library. The model-specific pieces of the code are provided by the user of the library in the form of subclasses or instances of the following six classes, seen as eponymous variables in the algorithm above.

- The **`ModelRunner`** is the program's entry point and initiates the process. This class handles program arguments, sets up the configuration and determines which instances of other classes should be used.
- The **`ModelConfig`** holds the configuration of the model. It is a mixture of generic fields, for example defining the variables in the model and their discretised representation, and model-specific fields such as model parameters.
- The **`State`** holds the state of the computation as the model is solved. Some of that state is imposed by the solvers below, some of it may be model-specific.
- The **`IndividualProbemSolver`** solves individual agents' optimisation problem.
- The **`AggregateProblemSolver`** finds the aggregate law of motion.
- The **`Model`** controls common functionality required by the other classes, and some decisions in the overall solution mechanism.

The remainder of this section provides more details on each of these classes, their purpose and the functionality provided by the ModelSolver library. Sample Scala code shown is that which solves the benchmark model as adapted by Krusell and Smith (1998) in the calibration of Den Haan (2010b), labelled the KS model.

2.4.2 `ModelRunner`

`ModelRunner` is the main entry point into the program. The implementation of this class defines what command line arguments the program will accept, processes those arguments to create a `Config` object and also determines which `Model` class will be used.

The library provides an implementation with a standard set of functionality to solve and simulate a model. The model-specific subclass must implement two functions: `createConfig` to create a configuration object, and `getModelClass` to identify which class holds the model itself.

Other methods can be overridden to extend the functionality or to handle special cases.

Listing 2.1: The ModelRunner Implementation for the KS Model

```
object Runner extends ModelRunner[Model, Config, State] {
 /**
  * Override the default simulation mechanism to read pre-created shock
  * sequence (from Den Haan et al 2010)
  */
 override def simulateModel(
    model: Model,
    state: State,
    periods: Int,
    burnIn: Int,
    stateDir: File,
    simPath: String): SimulationResults[_, _] = {

   // Read the data, and simulate the relevant shocks
    var zStream = getClass.getResourceAsStream("/Z_Formatted.txt")

    var shocksFromCSV = Numerics.instance().
        readFormattedCSV[Integer](zStream).asInstanceOf[IntegerArray]

    // Adjust the shocks because the numbering scheme differs from that
    // in the data
    var allShocks = createIntArrayOfSize(shocksFromCSV.size()(0), 2)
    allShocks.at(-1, 0).fill(shocksFromCSV.add(Integer.valueOf(-1)))
    val initialDist = model.initialDensity(state)

     // Get an appropriate simulator for this model
    val simulator = getSimulator((initialDist,0.asInstanceOf[Integer]))

    return simulator.simulateShocks(initialDist, allShocks, model,
      state, SimulationObserver.silent(), stateDir, simPath)
}

override def createConfig(commandLine: CommandLine): Config = {

  val config = new Config()

  // Initial Shock Levels (transient and permanent)
  config.setInitialExogenousStates(createIntArray(0,0))

  // Individual shocks (values actually not used)
  config.setIndividualExogenousStates(
         createArray(0.15, (1 - 0.15 * 0.05) / 0.95))

  // For solving, use log-distributed points of individual wealth
  config.setIndividualEndogenousStates(
         nLogSequence(200, 250, 1))

  // For Simulation, make the grid more dense and use even distribution
  config.setIndividualEndogenousStatesForSimulation(
         nLogSequence(200, 2001, 0))

  // Create a capital grid with 15 points spaced around the mean 39.85, but
      with the points closer to the mean a little
  config.setAggregateEndogenousStates(pullToMean(
         createArray(0.9 * 39.85 to (1.1 * 39.85, 15): _*), 0.2))

  // Good and bad productivity levels for agg shock
  config.setAggregateExogenousStates(createArray(.99d, 1.01d))

  // No normalising (permanent) shock
  config.setAggregateNormalisingExogenousStates(createArray(1d))
```

```
    // Markov Transition from denHaan comparison paper
    config.setExogenousStateTransiton( createArrayOfSize(2, 2, 2, 2, 1) << (
           21d / 40d, 1d / 32d, 7d / 20d, 3d / 32d,
           3d / 32d, 7d / 24d, 1d / 32d, 7d / 12d,
           7d / 180d, 1d / 480d, 301d / 360d, 59d / 480d,
           7d / 768d, 7d / 288d, 89d / 768d, 245d / 288d))


    // Indicate where the initial state should be read from
    setInitialStatePath("Solutions/KS_NA/state.mat")

    // The distribution to be used for derivative aggregation
    config.noAggRiskSteadyState = new DiscretisedDistribution(
           new File("Solutions/KS_NA/ergodicDist.mat"))

    config
  }

  override def getModelClass = classOf[Model]
}
```

2.4.3 `ModelConfig`

The ModelSolver library provides generic functionality for solving models. Each model has a distinct set of variables with distinct relationships. The `ModelConfig` bridges the gap from the specific model to the generic functionality.

Listing 2.2: The ModelConfig Implementation for the KS Model

```
/**
 * The Config class holds the configuration - some model specific fields and
 * some generic ones, set up in the Runner
 */
class Config extends ModelConfigBase {

  // Model-specific params
  var growthRate: Double = 1
  var discountRate: Double = 0.99
  var unemploymentInsuranceRate = 0.15
  var utilityFunction: UtilityFunction = new CRRAUtility(1.0)
  var productionFunction: ProductionFunction =
    new CobbDouglasProduction(.36, 1d-.36, .025)
  var noAggRiskSteadyState: DiscretisedDistribution = _

  // Expose defined variables, which have standard accessors, with meaningful
      names
  def individualCapitalLevels = getIndividualEndogenousStates()(0)
  def individualCapitalLevelsForSimulation =
      getIndividualEndogenousStatesForSimulation()(0)
  def individualShockLevels = getIndividualExogenousStates()(0)
  def aggregateCapitalLevels = getAggregateEndogenousStates()(0)
  def aggregateProductivityShocks = getAggregateExogenousStates()(0)
  def permanentAggregateShockLevels = getAggregateNormalisingExogenousStates()(0)

  // Wealth is constrained at 0
  override def isConstrained = true

  // Defined the solvers to solve individual and aggregate probelms
  override def getIndividualSolver = classOf[IM_EGM_Solver]
  override def getAggregateProblemSolver = classOf[IM_DA_Solver]
}
```

2.4.3.1 Model Variables

A key piece of functionality provided by the `ModelConfig` base class is to set up the variables that are part of the model. This information is required by other parts of the library, particularly the solvers.

Each variable is defined as a one-dimensional, monotonous array of values, which form the grid points in the dimension of that variable. In some cases, often for shocks, the assumption is that the variable is discrete and takes precisely those values; in other cases the variable is assumed to be continuous and the grid points are the points at which the function value is held, with interpolation determining intermediate points. The representation of the variable in `ModelConfig` is identical regardless of this assumption

The configuration defines the following different types of variables:

- *Individual endogenous states*: Variables if individual agents which are driven by their choice, but are determined prior to the period in which they take effect. An example is individual capital holdings $k_{i,t}$ in the benchmark model.
- *Individual exogenous states*: Variables of individual agents which are driven by exogenous processes and which are determined prior to the period in which they take effect. An example is employment status $e_{i,t}$ in the benchmark model.
- *Aggregate endogenous states*: Aggregate variables which are endogenously determined prior to the period in which they take effect. An example is aggregate capital K_t in the benchmark model.
- *Aggregate controls*: Aggregate variables which are endogenously determined in the period in which they take effect. An example is bond price in the extended model in Chapter 3.
- *Aggregate exogenous states*: Aggregate variables which are driven by exogenous processes and which are determined prior to the period in which they take effect. An example is the productivity shock in the benchmark model.
- *Aggregate exogenous normalising states*: Aggregate variables which are driven by exogenous processes, which are determined prior to the period in which they take effect and which do not affect exogenous processes once they have taken effect. An example is the permanent shock in the model of Carroll (2006).

The following discussion refers to these variable arrays as $\vec{x}, \vec{e}, \vec{X}, \vec{C}, \vec{E}$ and $\vec{N}$ respectively. Each of these is an unknown-length array of one-dimensional arrays.

Each type of variable is configured by calling an eponymous setter method, for example `setIndividualEndogenousStates(`$\vec{x}$`)`. An arbitrary number of each type of variable can be configured, but specific solvers may only support a limited number of any given type[12].

Note that two types of variable which might have been expected are missing: individual controls and individual exogenous normalising states. Individual controls can be used with the library, but they represent an agent's choices rather than determining them. Individual controls therefore never appear as an input variable into functions the solvers use, and only input variables form part of the grid and need to be configured. The models of Chapters 3 and 4 both have individual controls.

Individual exogenous normalising states might be used for variables such as permanent idiosyncratic shocks. These cause the variance of the distribution of individual exogenous states to grow indefinitely over time. Both solution and simulation of the model under the approach used by the ModelSolver library become impossible, so they are not supported.

The Grids

Functions used with the toolkit are represented as arrays which hold the value of the function at the grid points defined by its input variables, with an additional dimension at the end determining the number of output variables for multivalued function. Since each type of function has different inputs the size of the grid will also be different, but the ModelSolver defines certain rules which govern how grids are constructed.

Functions[13] arrange their inputs in the same order as the variable types are listed above. The individual policy function governing the choice of next-period individual states, for example, depends on all of the above variables, so the grid on which it is defined is of size corresponding to the lengths of $(\vec{x}, \vec{e}, \vec{X}, \vec{C}, \vec{E}, \vec{N}, length(\vec{x}))$. In the benchmark model, if the configuration has 100 points for individual capital, 2 for individual employment status, 12 for aggregate capital, 2 aggregate productivity states, then the grid for the individual policy rule is of size $(100, 2, 12, 2, 1)$. The final dimension is of length 1 because there is one individual

[12]The solvers currently provided only support a single individual endogenous state.

[13]This rule can be broken in rare cases where computational performance is strongly improved by doing so.

state, $k_{i,t}$.

2.4.3.2 Exogenous Process

`ModelConfig` also holds the transition probabilities which determine how exogenous variables evolve. The probabilities are configured by passing an array of probabilities to `setExogenousStateTransiton`. The array must have dimensions corresponding to the lengths of $(\vec{e}, \vec{E}, \vec{e}, \vec{E}, \vec{N})$, where the first two entries represent current period exogenous state and the last three next period exogenous states.

2.4.3.3 Model-Specific Configuration

Most models will also have model-specific configuration parameters which are best maintained in this class, an instance of which is shared by the solvers and the model.

2.4.4 `Model`

The `Model` implementation controls the overall process of finding the numerical solution: `initialState` provides the initial starting point and `shouldUpdateAggregates` is called each iteration after the individual decision rule has been updated and determines whether the aggregate law of motion is updated. The solution algorithm terminates when the convergence criterion, held in the `State`, becomes sufficiently small.

Both solvers have access to the `Model` class. It is therefore also a convenient location to hold variables which constitute neither configuration nor calculation state. One example is pre-computed values for commonly performed operations on input grid variables, avoiding costly repetitive calculations in each iteration.

`Model` also provides methods for constructing arrays for use as function grids, following the ordering convention described in the previous section. `createIndividualTransitionGrid` in the benchmark model would, for example, create the array of size $(100, 2, 12, 2, 1)$ described above.

Listing 2.3: Extract from the Model Implementation for the KS Model

```
class Model extends AbstractModel[Config, State]() {

  // These arrays are for convenience - they are constant and are used to
  // calculate wages and returns
  var employmentDistByProductivity: DoubleArray = _
  var aggLabourByProductivity: DoubleArray = _
  var wageRatiosbyProductivity: DoubleArray = _

  var expectedL: DoubleArray = _
  var expectedA: DoubleArray = _

  var interestGrid: DoubleArray = _

  var normalisedLiquidAssets: DoubleArray = _
  var simNormalisedLiquidAssets: DoubleArray = _

  override def initialise(): Unit = {

    super.initialise()

    // Prepare some values which do not change but are used often,
    // held in the fields of this class
    ...
  }

  // Update aggregates in each iteration
  override def shouldUpdateAggregates(state: State) = true

  // Just the mean of the capital distribution
  override def calculateAggregateStates(
      distribution: SimState,
      aggregateExogenousStates:
      IntegerArray,
      state: State) = Array(distribution.mean(
        _config.getIndividualEndogenousStatesForSimulation().get(0)))

  // Store the final distribution used for DA to start the sim with later
  override def writeAdditional(state: State, writer: NumericsWriter) {
      state.daDistributionByState(0).write(writer, "finalGradDensity")
  }

  /**
   * Called by the Toolkit when reading state, this is overridden to convert
   * non-agg-risk (NA) state into appropriate initial state with agg risk
   */
  override def readAdditional(state: State, reader: NumericsReader): Unit = {

    // If the policy that has been read has only one shock level it is an NA
    // state - need to expand to fit with-agg-risk grid
    if (state.getIndividualPolicy().size()(3) == 1) {

      val indPolicy = createIndividualTransitionGrid()
      indPolicy\(0,1,2) << state.getIndividualPolicy()($, $, $, 0, 0, 0)

      state.setIndividualPolicy(indPolicy)

      // Also need to initialise the aggregate transition, because the one
      // read is the NA (constant) transition
      initAggregateTransition(state)
    }

    // Calculate expectations from the aggregate transition
    adjustExpectedAggregates(state)
  }

  /**
    * Additional code which helps with initialisation omitted, see source
    */
  ...

  /**
    * Set up the starting state for the calculation
    */
  override def initialState(): State = {

    // Create a state object for this configuration
    val state = new State(_config)
    // State class needs these for interpolation
    state.normalisedLiquidAssets = normalisedLiquidAssets
    state.simLiquidAssets = simNormalisedLiquidAssets

    // The end of period states are just the capital levels to be carried over
    val simCapitalGrid = createSimulationGrid()

    simCapitalGrid.across(0) <<
      (_config.getIndividualEndogenousStatesForSimulation().get(0))

    state.simCapitalGrid = simCapitalGrid

    state.daDistributionByState = prepareDistributionsForGrad(
      _config.noAggRiskSteadyState)

    state
  }

  /**
   * Adjusts the expected capital level for permanent shocks (in the next
   * period) and also calculates expected prices given the expected factor
   * inputs
   */
  def afterAggregateExpectationUpdate(
      oldVal: DoubleArray,
      newVal: DoubleArray,
      state: State) {

    // Adjust for future permanent shocks
    state.getExpectedAggregateStates\2 /=
      _config.permanentAggregateShockLevels

    // Update expected prices
    val Array(w, r) =
      (state.getExpectedAggregateStates() :: expectedL :: expectedA) :-> (
        in => {
      val Array(k,l,p) = in
```

```
        Array(wage(p, k, l), grossInterest(p,k,l))
      })

    state.expectedR = r
    state.expectedW = w
  }

}
```

2.4.5 `IndividualProblemSolver`

The individual problem solution algorithm is assumed to be iterative. The method `performIteration` of this interface is called by the `Solver` to update the solution to the individual problem.

The base class is an interface leaving all the implementation to extensions. An extension of `IndividualProblemSolver` which performs the method of endogenous gridpoints (Carroll, 2006) is provided and described in Section 2.5.1.

2.4.6 `AggregateProblemSolver`

The method `updateAggregateTransition` of this interface is called by the `Solver` whenever the aggregate law of motion needs to be updated.

The base class is an interface leaving all the implementation to extensions. Extensions which perform the algorithm of Krusell and Smith (1998) and derivative aggregation (Chapter 3) are provided. A model-specific implementation employing explicit aggregation (Den Haan and Rendahl, 2010) is also provided for Chapter 3.

2.4.7 `State`

Instances of `State` hold the current state of the calculation. The interface defines methods that are required by the framework code to access and update that state. `getAggregateTransition`, for example, provides the current version of the aggregate law of motion.

Two important methods are `setIndividualError` and `setAggregateError`, which allow the solvers to update variables indicating how the solution is progressing. The overall convergence criterion is calculated from these errors. By default, the model is considered solved when both values have been set to less than $1e - 6$;

Implementations of this class are also the appropriate location for other, model-specific information that changes as the calculation progresses. The framework provides methods for writing this running state to disk.

Listing 2.4: The State Implementation for the KS Model

```
/**
 * The State holds the state of the ongoing calculation is updated by
 * Solver classes as well as model-specific code
 */
class State(config: Config) extends AbstractStateBase[Config](config)
                      with DerivAggCalcState[Config]
{
 var normalisedLiquidAssets: DoubleArray = _

 var simCapitalGrid: DoubleArray = _
 var simLiquidAssets: DoubleArray = _

 var expectedR: DoubleArray = _
 var expectedW: DoubleArray = _

 var daDistributionByState: Array[DiscretisedDistribution] = _

 override def getIndividualPolicyForSimulation() = {

  if (_individualTransitionForSimulation == null) {
   _individualTransitionForSimulation =
    interpolateFunction(normalisedLiquidAssets,
                 getIndividualPolicy(),
                 0,
                 simLiquidAssets,
                 params.constrained)
  }
```

```
    // Need to update the policy for the simulation grid, which is used by
    // derivative aggregation
    _individualTransitionForSimulation
  }

  override def getEndOfPeriodStatesForSimulation() = simCapitalGrid

  // There is only one shock, and use that one to get the
  def getDistributionForState(shockLevels: Array[Int]) =
                      daDistributionByState(shockLevels(0))
}
```

2.5 Algorithms

The classes introduced so far provide the framework for solving models with substantial heterogeneity and control the overall process. They do not assist in solving the individual or aggregate problems, nor has simulation been discussed.

This section addresses those topics. It introduces a class which performs the generic parts of the method of endogenous gridpoints of Carroll (2006) to solve individual problems, another which does the same for derivative aggregation (see Chapter 3) to solve the aggregate problem and a third which performs the algorithm of Krusell and Smith (1998).

The section also discusses the simulator, which simulates models with a distribution of agents using the algorithm of Young (2010). The concluding discussion concerns aggregate controls[14] and their handling by the ModelSolver Toolkit.

2.5.1 The Method of Endogenous Gridpoints

Some of the difficulty in solving models with heterogeneity among agents arises from the need to determine the individuals' policy functions for the range of feasible state values. Such functions are commonly found using some form of value function iteration (Young, 2010, is an

[14]Non-predetermined variables.

example), or by iterating on the Euler equation (for example Maliar et al., 2010). The former requires an expensive optimisation step in each iteration, and the latter tends to converge slowly. They are hence very time-consuming.

Carroll (2006) introduces an alternative algorithm which also iterates on the individual problem, but reverses the direction of iteration. It assumes, in each iteration, that the future individual policy function is known, and uses it to calculate the expectational part of the Euler equation conditional on future states. This in turn implies the current-period value in that equation. If the individual agent's current period states can be determined from that value analytically, then combining the three steps yields a mapping from future states to current states. Inverting this mapping results in the next candidate for the individual decision rule. The author called his method the *Method of Endogenous Gridpoints*.

The method requires the mapping from current states to current values in the Euler equation to be invertible. For models which satisfy that requirement it is significantly faster than alternative approaches. Indeed, in the comparison project of Den Haan (2010b) the two most efficient solutions to also provide relatively accurate results both use this approach. They are at least 6 times faster than other solutions.

2.5.1.1 The Mathematics

In the case of the benchmark model described in Section 2.1, the policy function sought is the household's next period capital choice. From Eq. (2.11):

$$k_{t+1} = f(k_t, e_t, K_t, A_t) \tag{2.13}$$

The method iterates on versions $f^j, j \in \mathbb{N}$, of this policy function, from an initial guess f^0. It assumes that the households' forecasting function for aggregate capital $K_{t+1} = F(K_t, A_t)$ is known.

In Section 2.1, the households' optimisation problem has already been reduced to the Euler equation, Eq. (2.8):

$$\beta E\left[(c_{t+1})^{-\gamma}(1+r_{t+1}-\delta)\right] = (c_t)^{-\gamma} - \phi_t \tag{2.14}$$

Rearranging the budget constraint and iterating it forward one period yields:

$$c_t = (1+r_t-\delta)k_t + e_t - k_{t+1} \tag{2.15}$$

$$\Rightarrow c_{t+1} = (1+r_{t+1}-\delta)k_{t+1} + e_{t+1} - k_{t+2} \tag{2.16}$$

Substituting this into the Euler equation and replacing k_{t+2} with the assumed policy function f^j then gives:

$$\beta E\left[\left(R_{t+1}k_{t+1} + e_{t+1} - f^j(k_{t+1}, e_{t+1}, K_{t+1}, A_{t+1})\right)^{-\gamma} R_{t+1}\right] =$$

$$(R_t k_t + e_t - k_{t+1})^{-\gamma} - \phi_t \tag{2.17}$$

$$\text{where } R_t \equiv (1+r_t-\delta) \tag{2.18}$$

Now consider the multiplier ϕ_t, and recall that the function to be found is f^{j+1} which gives k_{t+1} as a function of current states. Whenever $k_{t+1} > 0$ the borrowing constraint does not bind by definition, so that $\phi_t = 0$. When $k_{t+1} = 0$, the value of f^{j+1} is already known and is also 0, so ϕ_t is not required in this case. Thus ϕ_t can be safely dropped from the equation.

This allows for further rearrangement:

$$k_t = \frac{\beta}{R_t}\left(E\left[\left(R_{t+1}k_{t+1} + e_{t+1} - f^j(k_{t+1}, e_{t+1}, K_{t+1}, A_{t+1})\right)^{-\gamma} R_{t+1}\right]\right)^{-\frac{1}{\gamma}} - \frac{e_t}{R_t} \tag{2.19}$$

Finally, substitute for K_{t+1} using the assumed forecasting function:

$$k_t = \frac{\beta}{R_t}\left(E\left[v(k_{t+1}, e_{t+1}, K_t, A_t, A_{t+1})\right]\right)^{-\frac{1}{\gamma}} - \frac{e_t}{R_t} \tag{2.20}$$

where

$$v(k_{t+1}, e_{t+1}, K_t, A_t, A_{t+1}) = \tag{2.21}$$

$$\left(R_{t+1}k_{t+1} + e_{t+1} - f^j(k_{t+1}, e_{t+1}, F(K_t, A_t), A_{t+1})\right)^{-\gamma} R_{t+1} \tag{2.22}$$

Conditional on current and future states $(k_{t+1}, e_{t+1}, K_t, A_t, A_{t+1})$, $v(.)$ and hence the term inside the expectation operator can be calculated. The expectation is over the future exogenous states, whose probability distribution is known conditional on current exogenous states e_t and A_t. Thus the last set of equations yields the function

$$k_t = h(k_{t+1}; e_t, K_t, A_t) \tag{2.23}$$

where

$$h(k_{t+1}; e_t, K_t, A_t) = \frac{\beta}{R_t}\left(E\left[v(k_{t+1}, e_{t+1}, K_t, A_t, A_{t+1})\middle| e_t, A_t\right]\right)^{-\frac{1}{\gamma}} - \frac{e_t}{R_t} \tag{2.24}$$

If this function is invertible in its first parameter, then inverting it yields precisely the function sought:

$$f^{j+1}(k_t; e_t, K_t, A_t) \equiv h^{-1}(k_t; e_t, K_t, A_t) \tag{2.25}$$

Due to the borrowing constraint, the range of valid k_{t+1} does not extend below 0. If $h(0; e, K, A) = k^0 > 0$ for some (e, K, A), and under the assumption that $h(.)$ is monotonically increasing in k_{t+1}, this formula will not define $f^{j+1}(k_t, e, K, A)$ for $k_t < k^0$. But this is precisely the region where the borrowing constraint binds, so that $f^{j+1}(k_t, e, K, A) = 0 \; \forall \; k_t < k^0$.

2.5.1.2 The Algorithm

The method of endogenous gridpoints is performed as follows:

Algorithm 2.2: The Method of Endogenous Gridpoints

```
Guess $f^0$
j = 0
repeat
  foreach $(k_{t+1}, e_t, X_t, A_t)$ on the grid
    $s = 0$
    foreach $(e_{t+1}, A_{t+1})$ on the grid
      // Calculate the expected value, conditional on states
      $p = v(k_{t+1}, f^j(k_{t+1}, .), e_{t+1}, F(K_t, A_t), A_{t+1})$

      // Calculate the expectation cumulatively,
      // multiplying by the probability of each future state
      $s = s + \pi(e_t, A_t, e_{t+1}, A_{t+1})p$
    end

    // Invert to get implied current endogenous state(s)
    $k = h^{-1}(s, k_{t+1}, e_t, K_t, A_t)$

    // Store that point in the next iteration of the mapping
    $f^{j+1}(k, e_t, K_t, A_t) \equiv k_{t+1}$
  end
  $j = j + 1$

  // Stop when successive iterations are sufficiently close
```

```
    // by the chosen measure
until($||f^{j-1}, f^j|| < \epsilon_{\max}$)
```

The subclass `EGMIndividualProblemSolver` of `IndividualProblemSolver` performs those parts of the algorithm which can be automated. A model-specific extension of that class must implement `calculateFutureConditionalExpectations` for step (8), `calculateImpliedStartOfPeriodState` to perform step (16) and provide an initial guess for the individual policy in step (1). The following listing shows the implementation for solving the individual problem of the benchmark model:

Listing 2.5: Solves the benchmark model's individual problem using EGM

```
class IM_EGM_Solver(model: Model, config: Config)
  extends EGMIndividualProblemSolver[Config, State, Model](model, config) {

  // The end of period states are just the capital levels to be carried over
  setEndOfPeriodStates(_model.createIndividualVariableGrid()\(0) <<
     _config.individualCapitalLevels)

  setStartOfPeriodStates(_model.normalisedLiquidAssets)

  // This calculates the future part of the Euler equation, and is called
  // once for each possible future stochastic state and current agg state
  override def calculateFutureConditionalExpectations(
     exogenousTransition: Array[Int],
     currentAggs: Array[Int],
     state: State): DoubleArray = {

    val Array(_,currentAggStateIndex, futureIndStateIndex,
      futureAggStateIndex, futurePermShockIndex) = exogenousTransition
    val aggCapIndex = currentAggs(0)

    /* Collect the data needed for the calculation
     */
    //The expected future capital carried over for individuals in this state
    val kpp = state.getExpectedIndividualTransition()(
      currentAggStateIndex, futureAggStateIndex, futurePermShockIndex,
```

```scala
    aggCapIndex, $, futureIndStateIndex)

//Collect expected future aggregates
val R = state.expectedR(currentAggStateIndex, futureAggStateIndex,
    futurePermShockIndex, aggCapIndex, 0)
val w = state.expectedW(currentAggStateIndex, futureAggStateIndex,
    futurePermShockIndex, aggCapIndex, 0) *
    // Need to adjust the aggregate wage for the employment state
      _model.wageRatiosbyProductivity(futureIndStateIndex,
        futureAggStateIndex)

// Permanent shock
val perm = _config.permanentAggregateShockLevels(futurePermShockIndex)

/* Calculate the value of the euler condition conditional on the future
 * states
 */
//First, calculate consumption as starting capital times interest plus
// wages minus ending capital
val individualCapitalLevels: DoubleArray = _config.individualCapitalLevels

// Apply interest and normalise by permanent shocks
val kpMult = R / perm

// Calculate consumption at this point
// !!! KPP is from the function interpolated to normalised grid
// Need not take the permanent shock into account yet again!
val c = (individualCapitalLevels :: kpp) -> (
   (kp, kpp) => kp * kpMult + w - kpp * _config.growthRate)

// Need to normalize by growth and future permanent shock
val growthFactor = _config.growthRate * perm

/* Now determine the future euler value, i.e. the discounted marginal
 * utility of consumption
 */
val fce = c -> (c => R *
   _config.utilityFunction.marginalUtility(0, growthFactor * c))
```

```
    // Return that value
    fce
  }

  /**
   * Given expectations over the future euler, determine the implied current
   * liquid assets
   */
  override def calculateImpliedStartOfPeriodState(
      individualExpectations: JavaDoubleArray, state: State) = {

    // The last dimension is of size 1, so select it away to match sizes
    val impliedLiquidAssets = (individualExpectations :: _eopStates) ->
    (( futureEuler, kp ) => {

      // Invert implied marginal utility
      val c = _config.utilityFunction.inverseMarginalUtility(
                              _config.discountRate * futureEuler)

      /* USING THE GROWTH FACTOR HERE MEANS THAT kp is 'normalised' by it */
      // Liquid assets must be equal to consumption plus final capital
      c + kp * _config.growthRate
    })

    impliedLiquidAssets
  }

  /**
   * Overridden to reduce # of calcs
   */
  override def updateError(
      oldPolicy: JavaDoubleArray, newPolicy: JavaDoubleArray, state: State) {
    if(state.getPeriod % 10 == 5) {
      state.setIndividualError(
        maximumRelativeDifferenceSpecial(newPolicy(20, $), oldPolicy(20, $)))
    }
  }
}
```

2.5.2 Derivative Aggregation

Chapter 3 presents a new algorithm, Derivative Aggregation, for updating the aggregate forecasting function from the individual policy functions. Derivative aggregation is fast and produces accurate forecasts, at least in the economies investigated in this thesis. The ModelSolver library provides a class, `DerivativeAggregationSolver`, which performs much of the computation necessary. Only three model specific methods need to be implemented: `deriveAggregationByIndividualState`, `deriveAggregationByAggregateState` and `deriveIndividualTransformationByTheta`. In many cases these methods will be quite straightforward to implement. Further details are in Chapter 3. The following listing presents the code necessary to solve the incomplete markets version of the benchmark model:

Listing 2.6: Applies Derivative Aggregation to the Incomplete Markets Model

```
class IM_DA_Solver(model: Model, config: Config,
   simulator: DiscretisedDistributionSimulator)
 extends DefaultDASolver[Config, State, Model](model, config, simulator) {

 // Use a log linear rather than a linear transition
 useLogs(true)

 // These grids needs to be the size of the simulation grid
 // dF/dk = 1 (it is the mean)
 setAggregationByIndividualStateDerivative(0, 0,
    model.createSimulationGrid() << 1)

 // dk/dK = k (proportional wealth increase)
 setTransformationByIndStateDerivative(0, 0,
    model.createSimulationGrid()\(0) <<
      config.getIndividualEndogenousStatesForSimulation()(0))
}
```

2.5.3 The Algorithm of Krusell and Smith (1998)

The algorithm of Krusell and Smith was the first which allowed a model with heterogeneous, rational agents and aggregate uncertainty to be solved. It is at the root of much of the literature discussed in this thesis.

The algorithm is both straightforward to execute, and widely applicable. The drawback of the algorithm is that it requires a significant amount of simulation to solve a model, and simulation is computationally expensive. With modern computers, however, a solution to the benchmark model with incomplete markets, for example, can be found within approximately 4 minutes.

The method for updating the aggregate forecasting function given the individual policy functions is:

1. Simulate the model for a number of periods, collecting aggregate state and control values
2. Discard some of the initial periods, allowing the economy to 'settle down'
3. From the remaining data estimate forecasting functions for current controls and next-period aggregate states given current aggregate states

An implementation of this algorithm is provided by the class `KrusellSmithSolver`. An advantage of this class compared to derivative aggregation, for example, is that it does not require any additional implementation.

Some aspects of the algorithm can be configured: the number of periods to simulate and discard can be set; the forecasting function can be estimated as a linear or log-linear function; the amount of damping can be adjusted; the algorithm can be configured to reuse the same shock sequence in each iteration; and it can similarly be configured to start each iteration from the distribution which resulted from the final step of the prior iteration. The following listing shows how this solver can be used to solve the incomplete markets model:

Listing 2.7: A KrusselSmithSolver for the eponymous model

```
class IM_KS_Solver(model: Model, config: Config)
 extends KrusellSmithSolver[Config, State, Model, DiscretisedDistribution](
 /*Creating a new Simulator excludes the observers, which are not needed*/
   model, config, new DiscretisedDistributionSimulatorImpl) {
 setSimPeriods(1100)
 setDiscardPeriods(100)
 setNewWeight(.3)
 useLogs(true)
 keepDist(true)

 addTransitionListener(
   (oldVal: JavaDoubleArray,newVal: JavaDoubleArray,state: State) => {
  state.setIndividualError(1) // Ensure individual problem is solved again
  println(s"""${state.getAggregateCriterion}""")
 })
}
```

2.5.4 Simulating Heterogeneous Agent Models

Having solved a model, economists commonly wish to simulate it. In some cases, such as the incomplete markets model discussed in Section 2.6.3, interesting implications of the model cannot be obtained from the solution directly and must be revealed through simulation. In other cases, the impact of certain changes to exogenous variables is to be analysed. Many algorithms for solving models (see, for example Krusell and Smith, 1998), themselves rely in part on simulation. Any toolkit to solve such models must therefore provide for it.

Models that allow for heterogeneity among agents commonly assume a continuum of agents. As discussed in Algan et al. (2008), this assumption is crucial even in terms of the definition of aggregate state variables: it implies that the moments of individual agents' realisations of stochastic variables, conditional on aggregate stochastic states, are known from the law of large numbers. They do not need to be considered as additional aggregate states. The authors argue that a simulation procedure should replicate this behaviour. Simulation approaches which simulate individual agents may not satisfy this constraint even when a very large number of agents are simulated. In simulations other than then very shortest the proportion of agents with a given history of shocks will not be close to the value implied by the law of large numbers.

Young (2010) develops a simulation algorithm which does satisfy the constraint. In this approach, the continuous distribution of agents over their individual states is approximated by a discretised distribution[15] over a time-invariant grid. In each simulation period, the full distribution of individual shocks, conditional on the aggregate shock realisation, is applied to the population at each point on this grid. Individual choices are calculated conditional on those shocks and the population at the point is distributed to the points of the next-period distribution based on the future individual states obtained.

Since the distribution grid is discrete but individual policy functions are continuous, it will almost surely be the case that the individual future state calculated does not lie precisely on a grid point. The author's innovation is to redistribute the population of agents who choose this state to the grid points either side of it, whilst ensuring that their mean state remains at this value.

As an example, illustrated in Fig. 2.2, consider the following case in the benchmark model: assume that a set of households of measure $.1$ hold capital 1 and are all employed, that the probability of remaining employed in the next period is $.8$, the probability of unemployment by implication $.2$. The capital grid consists of 6 points, $\{0, 1, \ldots, 5\}$.

The individual policy function is such that employed households with capital 1 will choose 3.7 in the next period. The sub-population currently at 1 is therefore distributed as follows: of the $.08$ with good fortune, $.024$ are assigned to capital level 3 and $.056$ to capital level 4, ensuring that their average capital holdings remain at 3.7. Similarly, $.02$ suffer bad luck and become unemployed and are distributed to the same grid points 3 and 4 with weights $.006$ and $.014$.

Repeating this procedure for each point in the current period distribution and taking the

[15]The author calls it a 'histogram', but the simulation procedure implies point densities, so this thesis avoids the term

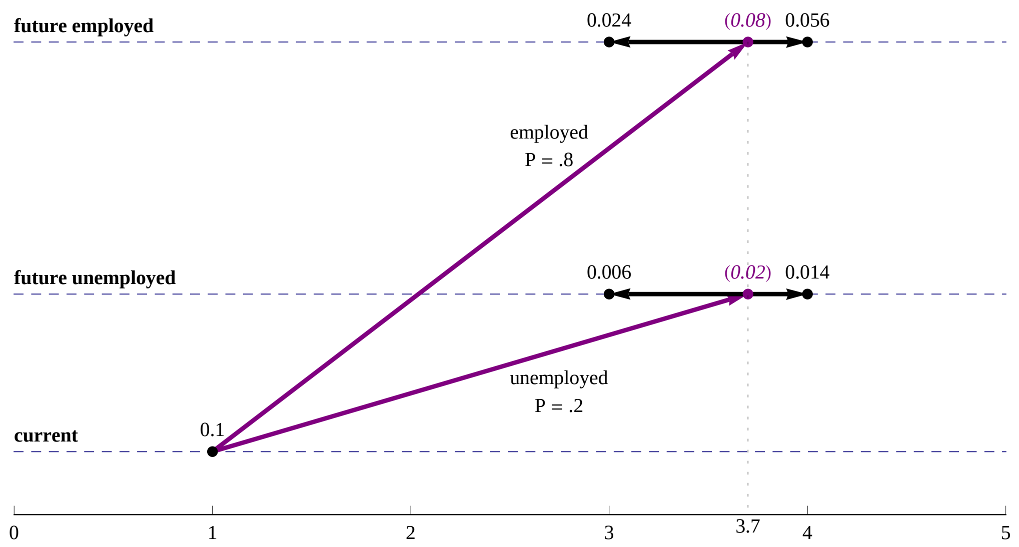

Figure 2.2: The Simulation Mechanism (Young, 2010)

sum of contributions at each point of the future period yields next period's distribution over endogenous and exogenous individual states.

2.5.4.1 Dealing with Simulation Grid Overflow

The grid on which the distribution is defined is, by necessity, finite. The mechanism as illustrated breaks down when some part of the population chooses a state that lies outside the grid. It may be possible to avoid this issue by choosing a grid that is sufficiently large, but in economies where the individual policy is such that the state may grow indefinitely, and when many periods are to be simulated, that is not possible.

Young (2010) does not address the problem[16]. Den Haan and Rendahl (2010), who also use this simulation mechanism, choose to treat such a situation as if the policy choice were between the two highest capital levels[17]. This is a numerically feasible approach, but it has the undesirable effect that the contribution from any individual point which overflows to the

[16]The code accompanying the paper may be informative, but it is somewhat difficult to follow.
[17]Echoing the approach one would take in a linear extrapolation.

penultimate point in the distribution is negative.

The approach used in the simulator provided by the ModelSolver toolkit is as follows: All parts of the population that extend beyond the grid are collected in one group[18] and assigned a common state value equal to the mean for that population. This solution effectively adds a grid point in each endogenous dimension, but where the value of that endogenous state is variable (and indeed varies along the other dimensions).

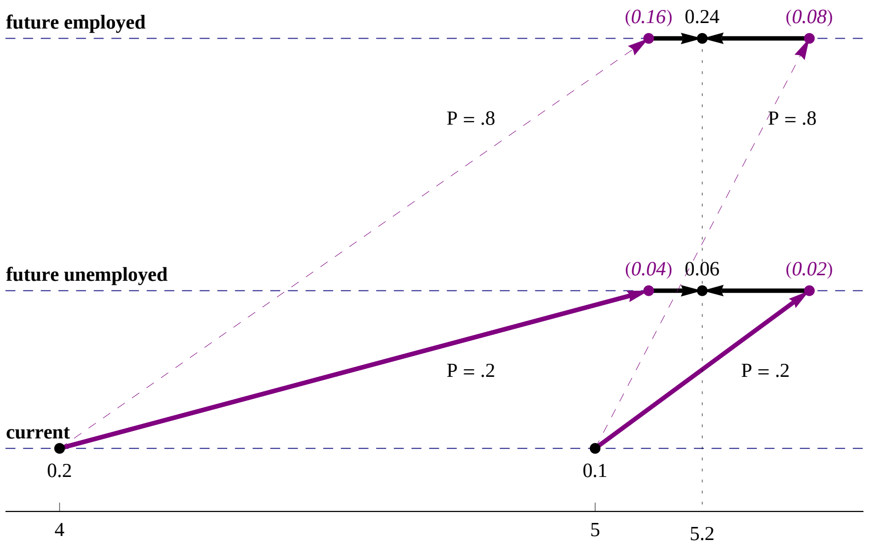

Figure 2.3: Simulation Grid Overflow

Continuing with the benchmark economy, Fig. 2.3 illustrates the overflow mechanism. Assume that in the current period .1 of the population are employed and hold capital 5, and .2, also employed, hold 4. Assume the policy function yields future wealth of 5.4 and 5.1 for the two groups.

In the next period there are then .08 future employed households with capital 5.4 from the first group, and .16 with capital 5.1 from the second group. Combining the groups gives a group of mass .24 with wealth 5.2. For the future unemployed a similar calculation yields a

[18]To be precise, one group per exogenous individual state

group at the same wealth level of weight .06.

2.5.5 Aggregate Controls

Non-state aggregate variables, referred to as aggregate controls in this thesis, required careful handling. Two related difficulties arise, both during simulation of the economy. First, aggregate control values are determined in equilibrium in the period in which they affect individual choices, thus presenting a chicken and egg problem: individual choices determine aggregate controls determine individual choices. Second, the equilibrium may not be directly represented by the variable value which affects individual choice: in the model of Krusell and Smith (1997), for example, the variable affecting individual choice is bond *price* but the equilibrium condition states that net bond *demand* is zero.

Addressing the first point, Ríos-Rull (1997) presents two different approaches. The first constructs a forecasting function which forecasts the control value conditional on aggregate states[19], and assumes this forecast value is the realized equilibrium value during solution of the individual problem and simulation. The aggregate controls obtained are thus forecasts and not necessarily the correct value which follows from individual choices. The alternative solves the individual problem conditional on aggregate controls, calculates individual choices conditional on aggregate controls in each simulation period, and then determines the market clearing control value using the equilibrium condition. This approach determines market clearing values more accurately because it solves the equilibrium problem each period, but at the cost of more computation. The ModelSolver library uses the latter approach.

The second problem is addressed below.

2.5.5.1 Definition of Equilibrium Values

Aggregate controls are determined in equilibrium. It may be the case that the individual control[20] and the aggregate control are not directly related. This is the cases in the model of

[19]This is analogous to the forecasts of future aggregate states.

[20]Though it is not configured since it is not a ‘grid variable’, an individual control can be calculated and used as part of the solution.

Krusell and Smith (1997): the individual control is ‘individual bond holdings’ which aggregates to ‘net bond holdings’, which in turn must be 0 in equilibrium. The aggregate control used in the individual decision rule, however, is ‘bond price’, since that is the variable that individual households observe when making their decision. The ModelSolver library refers to the equilibrium variable as the ‘determinant’.

This introduces two (sets of) variables,

C_t	the aggregate control variables (bond price)
$E_t^C \equiv E^C(\omega_t, C_t, A_t)$	the aggregated individual controls (net bond demand, the ‘determinant’)

The individual policy function is then

$$f_x \equiv f_x(x_{i,t}, a_{i,t}, X_t, C_t, A_t)$$

where $x_{i,t}$ are individual endogenous states, $a_{i,t}$ individual exogenous states, X_t aggregate states and A_t aggregate exogenous states.

Solving for equilibrium also requires knowledge of the target of the equilibrium variable, which is expressed as a function of the control variable:

$$T_t^C \equiv T^C(C_t)$$

so that the equilibrium condition is

$$E^C(\omega_t, C_t, A_t) = T^C(C_t) \tag{2.26}$$

This latter definition allows for some flexibility: in the case of the model with bonds the function is identically 0, $T_t^C \equiv 0$, since net bond holdings must be 0.

As an alternative example, consider the case where C_t is aggregate labour supply. In that case, E_t^C would be realised aggregate labour supply, and the two values would have to be equal (i.e. the agents observe the labour supply, which is also the aggregated outcome of their individual labour supply decisions). Hence $T_t^C \equiv C_t$.

The `ModelConfig` implementation provides method `setControlTargets` which allowed $T^C(C_t)$ to be defined. Determinant values are only needed during simulation and are calculated by the implementation of `calculateControlDeterminants` in the model-specific subclass of `Model`.

2.6 Evaluating the ModelSolver Toolkit

The ModelSolver Toolkit utilises numerical methods to find approximate solutions which satisfy model equations. The use of numerical methods raises the question of accuracy. This question cannot be answered definitively because the numerical methods used are not fully specified, depending as they do on the algorithms employed, which the toolkit does not prescribe.

This section approaches the question obliquely: rather than evaluating the accuracy of a solution directly it compares it to solutions generated by existing tools, using models and algorithms that have already been documented in the literature. In this way, both the numerical result and the speed of execution can be compared.

Three different calibrations of the benchmark model are evaluated, increasing in complexity. The solutions obtained match their respective benchmarks closely, whilst the ModelSolver performs relatively poorly at solving the less complex[21] problems but outperforms the alternative by a factor of six for the most complex calibration.

The source code which calculates these solutions, along with scripts which evaluate the results, are available online (Grasl, 2014a).

[21]The main reason is that the ModelSolver assumes the problems as complex.

2.6.1 The Deterministic Solution with Identical Households

The model is first solved under the assumptions that there is no uncertainty, that all exogenous processes are constant and that all households are identical.

Assume, w.l.o.g., that $A_t \equiv 1$. The further assumption that all households are employed is also justified, since the only alternative of universal unemployment is not interesting. Hence $L_t \equiv 1$.

In this case total income is $w_t\bar{l}$ and, since households are identical, it is shared equally between all agents. Hence

$$e_t = w_t\bar{l} \tag{2.27}$$

2.6.1.1 The Analytical Solution

Solving Eq. (2.2) backwards yields

$$P_t = g^t P_0 \tag{2.28}$$

for some initial productivity P_0.

Assume that a balanced growth path exists s.t. $k_t = k_0 g^t$ and $c_t = c_0 g^t$, and substitute these into Eq. (2.8):

$$(c_0 g^t)^{-\gamma} = \beta(c_0 g^{t+1})^{-\gamma}(1 + r_{t+1} - \delta) \tag{2.29}$$

and, by Eq. (2.3),

$$(c_0 g^t)^{-\gamma} = \beta(c_0 g^{t+1})^{-\gamma}(1 + \alpha \left(\frac{k_0 g^{t+1}}{P_0 g^{t+1} \bar{l}}\right)^{\alpha-1} - \delta) \tag{2.30}$$

$$\Rightarrow \qquad g^{\gamma} = \beta(1 + \alpha \left(\frac{k_0}{P_0 \bar{l}}\right)^{\alpha-1} - \delta) \tag{2.31}$$

$$\Rightarrow \qquad k_0 = P_0 \bar{l} \left(\frac{g^{\gamma} - \beta(1-\delta)}{\alpha\beta}\right)^{\frac{1}{\alpha-1}} \tag{2.32}$$

2.6.1.2 The ModelSolver Solution

The first step toward finding k_0 using the ModelSolver computes a numerical approximation to the individual savings decision over a range of input values. Since all households are identical at all times, household wealth is equal to aggregate capital holdings. The law of motion of aggregate capital is then just the individual policy function. k_0 can be found as its fixed point.

The individual policy function is approximated on a grid over the input variables. During the computation, individual and aggregate capital are considered as separate variables, despite having to be identical in the solution. This is to facilitate reuse of the same code as the more complex calibrations below, though it also allows the hypothetical impact of either variable on the decision to be analysed in isolation.

There are 11 points in both the individual and aggregate capital dimensions, spaced evenly around the analytical steady state. The policy function is found over this range of values using the method of endogenous gridpoints (Carroll, 2006), and the fixed point of that policy function where $k = K$ is steady state capital.

2.6.1.3 Results

α	β	γ	δ	G	P_0	$\bar{l}$	k_0
$\frac{1}{3}$	0.98	$\frac{1}{2}$	0.025	$1.025^{\frac{1}{4}}$	1	1	17.98

Table 2.1: Parameters of the Economy

Table 2.1 shows a possible parametrisation of the problem described, along with the implied value of steady state capital in the deterministic case, calculated using Eq. (2.32).

With the grid as described, the ModelSolver finds the correct solution to within $1e-15$. This indicates that the ModelSolver is capable of finding a good solution, though the accuracy is also partially a result of ex ante knowledge of the correct solution, since one of the grid points is in fact the correct solution. Perturbing the grid from this position by 1.01% causes the accuracy of the solution found to drop to $4e-7$.

2.6.2 Complete Markets with Aggregate Uncertainty

The second calibration adds aggregate uncertainty, but maintains homogeneity among households. Aggregate uncertainty is introduced by allowing A_t to vary following an exogenous stochastic process. The process is set up, following King and Rebelo (1999), as an $AR(1)$ process in logs:

$$A_t = \mathrm{e}^{\epsilon_t} \tag{2.33}$$

$$\text{where} \qquad \epsilon_t = \rho\epsilon_{t-1} + \xi_t \tag{2.34}$$

The parameter values are shown in Table 2.2.

ρ	σ_ξ	σ_ϵ([22])
0.979	0.0072	0.0353

Table 2.2: Parameters of the aggregate stochastic process

The ModelSolver solves the model on a grid of discrete points in the state space. It hence requires a discrete, rather than a continuous, stochastic process. Tauchen (1986) first demonstrated how to convert the latter to the former, and Kopecky and Suen (2010) introduced a refined algorithm suitable for highly persistent stochastic processes. The solution presented

[22]Note that σ_ϵ is not a free parameter but is implied by the other parameters.

here uses the latter approach with 21 points in the aggregate exogenous state dimension.

The number of points in the individual and aggregate capital dimensions is increased to 31, with the range increased to 4 from the deterministic steady state in either direction. These points are more tightly spaced around that steady state to provide the greatest grid density in the area where the economy is likely to spend most of its time.

As a benchmark for comparison, the model is also solved in Dynare using both first and second-order approximations.

2.6.2.1 Results

Figure 2.4 plots the laws of motion for aggregate capital in this economy generated by the ModelSolver and the Dynare second-order approximation. Both are shown as percentage deviations from the first-order approximation since the linear component would otherwise dominate and obscure the differences. The plot on the left shows future aggregate capital as a function of the exogenous productivity when the current capital is at the deterministic steady state level. On the right, K_{t+1} is plotted as a function of K_t when productivity is at its mean value 1.

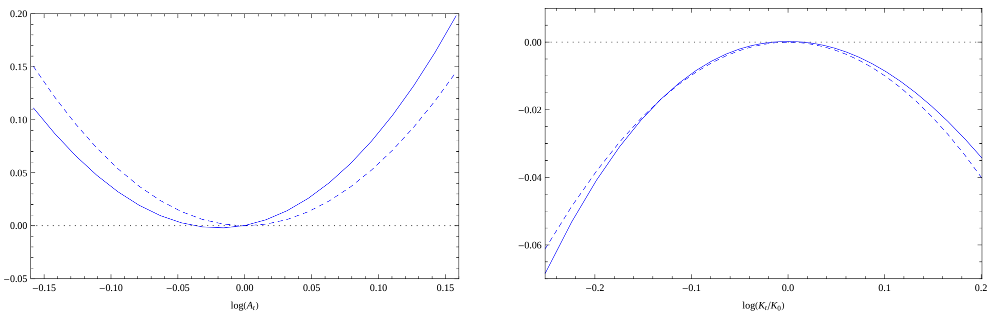

Figure 2.4: t+1-Period Aggregate Capital
Solver (solid) and Dynare 2nd-order (dashed)
As deviation from Dynare linear solution

The graphs suggest that the two solutions are very similar. A numerical comparison confirms this impression: On the aggregate grid used by the ModelSolver the point-by-point relative difference between the values of K_{t+1} produced by the two approaches is everywhere less than $.056\%$. The two solution approaches are quite different, so the similarity of the results is remarkable.

One interesting finding concerns precautionary savings: the model of this section differs from that of the last section only in that it has uncertainty, so that households have a precautionary savings motive in addition to the desire to smooth consumption. The precautionary savings motive implies lower consumption given the same state of the economy, which in turn implies greater savings and hence a higher future capital level. But the consumption-savings decision is the only endogenous choice in this model, so the effect of precautionary savings should be to unambiguously raise the fixed point of the capital transition function at the mean level of productivity.

The fixed point of the aggregate law of motion for capital at productivity 1 is $.002\%$ higher than the deterministic steady state in the ModelSolver solution. Dynare finds it to be $.0004\%$ lower.

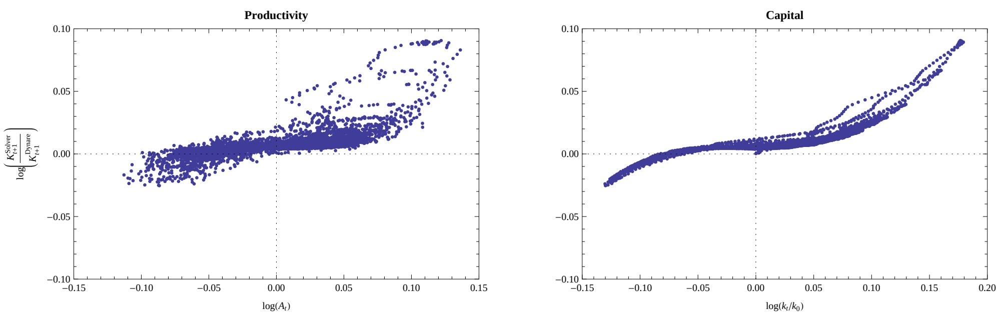

Figure 2.5: Scatter plot of $(t+1)$-period Difference between the Dynare and ModelSolver simulation results against t-period Productivity and Capital respectively

A further comparison of the two solutions is provided by simulating an identical series of shocks from an identical starting point using each one. The simulation covers 10000 periods. Figure 2.5 plots the differences in the aggregate capital produced against the prior period's shock (left) and aggregate capital. The mean relative difference is just $.0072\%$, with the maximum below $.1\%$. The graphs illustrate that the errors increase with distance from steady state.

2.6.3 Incomplete Markets with Aggregate Uncertainty

In addition to the aggregate uncertainty described above, households may be exposed to uninsurable idiosyncratic income risk. The model of Krusell and Smith (1998) extends the benchmark to include idiosyncratic employment shocks which, by assumption, households can not insure against due to the absence of complete markets. This section solves that model in the calibration of Den Haan (2010b).

Households can be either unemployed or employed in each period. All employed households earn the same wage and all unemployed households receive identical unemployment benefits, at replacement rate μ of the employed households' income. The government runs a balanced budget and finances unemployment benefit by taxing labour income at rate τ_t.

The income process for individual households is hence dependent on their employment state $a_t^i \in \{0,1\}$, where $a_t^i = 1$ for an employed household and $a_t^i = 0$ for an unemployed household in period t.

Household income is then

$$e_t^i = [(1-\tau_t)a_t^i + \mu(1-a_t^i)]\bar{l}w_t \tag{2.35}$$

The per-period income tax rate τ_t is chosen to ensure that tax receipts exactly cover unemployment benefits. If the unemployment rate is u_t, this implies

$$(1-u_t)(\tau_t\bar{l}w_t) = u_t(\mu\bar{l}w_t) \tag{2.36}$$

$$\tau_t = \mu\frac{u_t}{1-u_t} \tag{2.37}$$

Aggregate productivity also follows a two-state Markov process, with productivity levels $\{.99, 1.01\}$. Unemployment is exogenous and is tied to the productivity process, with unemployment levels of 10% and 4% at low and high productivity respectively.

The joint stochastic process governing the transition of aggregate productivity between states, and of households between employment and unemployment, is chosen as follows: each aggregate productivity level has mean duration of 8 periods; unemployment has a mean duration of 2.5 periods when productivity is low, and 1.5 periods when productivity is high. The resulting probability transition matrix is shown in Table 2.3.

	A_{t+1}	$1-\xi$		$1+\xi$	
	a^i_{t+1}	0	1	0	1
A_t	a^i_t				
$1-\xi$	0	0.525	0.35	0.03125	0.09375
	1	0.038889	0.836111	0.002083	0.122917
$1+\xi$	0	0.09375	0.03125	0.291667	0.583333
	1	0.009115	0.115885	0.024306	0.850694

Table 2.3: Transition Probabilities in the Baseline Case (Source: Den Haan et al., 2010)

The replacement rate of unemployment benefit is set to .15 and the labour endowment per worker is chosen so that aggregate labour supply is 1 in the bad state, implying $\bar{l} = 1/0.9$

2.6.3.1 Results

To evaluate the performance and accuracy of the ModelSolver in such a setting, the model is solved using the Explicit Aggregation algorithm of Den Haan and Rendahl (2010). Their original implementation in MATLAB serves as a benchmark against a re-implementation using the ModelSolver Toolkit.

Run	Matlab	ModelSolver
1	1235	209
2	1334	215
3	1383	197
Average	1317	207

Table 2.4: Explicit Aggregation Solution, Solution Times (seconds)

The model is solved three times using each implementation, on the same machine and with no other applications running. Both solutions are configured to run in a single thread.

The calibration solved is $\gamma = 1$, starting from a common initial guess.

Table 2.4 shows the times taken for all runs. The average time for the Matlab solution is 1317 seconds, with 207 seconds required by the ModelSolver-based version. Both solutions were found in 422 steps.

	Mean Difference	Maximum Difference
Individual Policy	2.2E-8	8.8E-8
Aggregate Forecast	7.2E-9	2.5E-8

Table 2.5: Explicit Aggregation Solution, Relative Differences
Matlab vs ModelSolver

The resulting individual policy functions and aggregate capital forecasting rules are almost identical. Table 2.5 shows the point-by-point comparison across the two functions' grids. The maximum errors for both lie below 1E-7. The threshold for ending the iterative process is 1E-6, so that either solution would be accepted by the other implementation as a legitimate one relative to its own.

2.7 Future Work

2.7.1 Additional Algorithms

Section 2.5 introduced the algorithms provided by the ModelSolver Toolkit. Though these algorithms are applicable to a variety of problems, as demonstrated in this thesis, they do require models to satisfy certain constraints which may discount some models of interest. One clear route for enhancing the toolkit is to provide implementations for additional algorithm, such as value function iteration.

2.7.2 Function Objects

The ModelSolver uses `DoubleArray` objects extensively to represent mathematical functions. The arrays hold the function values and thus define the range of the function, but provide no information regarding the domain. The latter, along with related information such as the location of discontinuities, is implicit, and must be maintained by the calling code. This is inefficient, makes the code harder to work with and can give rise to errors. The inefficiency is compounded by the fact that many models have multiple functions with the same domain, and where good behaviour breaks down at the same points.

The Numerics library class `Function` captures all this information. Objects of this type provide many useful operations on functions, for example numeric differentiation. They are used internally by the ModelSolver Toolkit, but are not yet exchanged with the model code. Using this class more widely would improve the usability of the toolkit significantly, especially for models that are not quite as well-behaved as the one considered in this chapter.

2.7.3 Porting to Other Technologies

Both the Numerics library and the ModelSolver Toolkit have been in parallel development since the inception of this project. After evaluating the suitability of both MATLAB and Mathematica to performing the task, and finding them unsuitable in different ways, Java and, later, Scala where chosen as the implementation languages mostly due to the absence of other obvious candidates, coupled with the author's existing knowledge of these languages.

The `DoubleArray` class in particular provides a relatively specialised and, hence, unique way of working with functions approximated on grids. There is, however, no obvious impediment to replicating this class in other languages. Such replication is not of benefit in itself, but would be worthwhile if a greater number of interested economists were familiar with the target platform.

The *Julia* programming language (see Bezanson et al., 2014b), published subsequent to commencement of this project, is gaining traction in the community of computational economists. It might provide a more suitable platform for the ModelSolver Toolkit.

2.8 Conclusion

This chapter presented two libraries, the Numerics library and the ModelSolver library, which together form the ModelSolver Toolkit. The Numerics library provides a class `DoubleArray` for working with large, multi-dimensional arrays of `double` values, and many associated operations and utilities which make working with such arrays straightforward. One key use of `DoubleArray` is to hold values of functions defined by interpolation over a grid. The ModelSolver library uses this paradigm extensively to provide implementations of the common parts of algorithms used in solving models with substantial heterogeneity.

The toolkit was used to solve three different calibrations of the stochastic growth model, including the version first introduced by Krusell and Smith (1998). These solutions, provided in the code accompanying the thesis, demonstrate the flexibility of the toolkit. The solutions were compared to ones obtained from existing tools and were found to be accurate. In the most complex case, the ModelSolver outperformed an equivalent solution in MATLAB by a factor of 6.

3

Finding the Forecasting Function by Derivative Aggregation

Any researcher interested in questions concerning the interaction of aggregate and individual outcomes in the economy will, at some point, want to solve models that include both aggregate and individual uncertainty. The latter gives rise to heterogeneity, which in turn implies that the full state of the economy is a very high-dimensional structure. Models of this type do not usually have analytical solutions. Standard perturbation-based methods are prohibitively expensive with such high-dimensional problems, so other computational approaches are necessary. Such methods have been available since at least the pioneering contribution of Krusell and Smith (1998). There have been many refinements to the available algorithms since, but the solution of models with anything other than the most rudimentary structures remains computationally expensive and slow.

This chapter presents a novel algorithm, termed ‘Derivative Aggregation’, for finding a forecasting function for the aggregate variables in such models. The algorithm calculates derivatives of the forecasting function by using directional derivatives along curves in the set of feasible distributions. Each curve captures the way in which the distribution changes as one of the current period aggregates changes. The derivatives of the forecasting function are found by aggregating derivatives of the individual policy functions along the curves.

The algorithm is presented in a very abstract form, and is in principle applicable to any economy with a clear relationship between individual and aggregate states, but where individual agents’ choices do not directly affect those of others. This chapter will argue both that the mathematics underlying derivative aggregation describes why approximate aggregation[1] holds in some cases, and that in those cases where it does not hold the approach is nonetheless a reasonable way to determine a forecast rule that might be used by boundedly rational agents.

To demonstrate the high level of accuracy and low computational cost of the approach it is first used to solve the model described in Den Haan et al. (2010), allowing comparison with other approaches. The solution is found more efficiently and with accuracy comparable to the Explicit Aggregation (XPA) algorithm of Den Haan and Rendahl (2010). The latter was found to be one of the best in terms of accuracy and performance in Den Haan (2010b).

The algorithm also extends to more complex scenarios without substantial adjustment, and at reasonable additional computational overhead. This is demonstrated by solving the model of Krusell and Smith (1997), which includes an additional asset, a one period bond.

[1]As defined in Krusell and Smith (1998) to mean that the aggregate forecasting rule predicts aggregates ‘almost perfectly’

All the solutions developed for this chapter use the ModelSolver toolkit introduced in Chapter 2. The toolkit includes a solver that automates the algorithm described here, and which requires only very straightforward code to utilise. The code for the toolkit and the models are available online.

The remainder of this chapter is structured as follows: Section 3.1 outlines some key assumptions adopted throughout this chapter. Section 3.2 motivates the algorithm by considering a simplified form of the benchmark model, solves the benchmark model and assesses the solution. Section 3.3 explores some approaches to further improving the accuracy of forecasts.

Section 3.4 presents the mathematics of derivative aggregation for generic models in detail. Section 3.5 solves the extended model with one period bonds. Section 3.6 discusses the approach, and Section 3.7 concludes.

To ease presentation most of the mathematical and computational details are relegated to appendices, rather than interrupting the flow of the main body: Section A.1 contains proofs of theorems and derivations, Appendix A.2 demonstrates the extension of the approach to higher-order approximations, while Appendices A.3 and A.4 discuss the individual and aggregate solutions to the model with bonds.

The source code which calculates all model solutions discussed in this chapter, along with scripts which evaluate the results, are available online (Grasl, 2014a).

3.1 Assumptions

This chapter adopts all of the assumptions and concepts introduced in Section 2.2. The most significant of these assumptions is *bounded rationality*. This assumption in turn leads to the application of the *alternating solution* approach.

The two alternating steps of the approach are to solve the household problem and to update the forecasting function. Derivative aggregation, the method presented here, addresses the latter. It does not change the approach for solving the household problem, which is solved using the Method of Endogenous Gridpoints (Carroll, 2006) in all models discussed in this chapter.

3.1.1 The Individual Policy Function

In the benchmark model presented in Section 2.1, a household must forecast future aggregate capital in order to form expectations over the future prices that affect its decision. Under the assumption of bounded rationality, it forms this forecast using only the current aggregate variables, productivity A_t and capital K_t, as inputs. By implication its decision is conditional on these variables, rather than the full state of the economy S_t. The household's policy function, which determines its future capital holdings $k_{i,t+1}$, takes the form

$$k_{i,t+1} = f(k_{i,t}, e_{i,t}, A_t, K_t) \tag{3.1}$$

This policy function is assumed known when the Derivative Aggregation step is performed. It is further assumed to be differentiable in the continuous variables k_t and K_t almost everywhere.

3.1.2 The Set of Feasible Distributions Ω

The economy is populated by a continuum of households of measure 1, indexed by i. These households have identical preferences and, at any given point in time, differ only in their employment state $e_{i,t} \in \{0, 1\}$, where 0 means unemployed and 1 means employed[2], and wealth $k_{i,t}$. The distribution of households over the two variables at time t, ω_t, can be written as

$$\omega_t = \omega_t : [0,1] \to \mathbb{R} \times \{0,1\} \equiv \{(k_{i,t}, e_{i,t}) : i \in [0,1]\} \tag{3.2}$$

[2]Employment state is exogenous and hence has little impact on the algorithm, which is concerned with how endogenous states develop. Much of the rest of the chapter ignores employment state and is implicitly conditional on it.

Given the constraints of the economy, not all possible distributions can be realised. Let Ω be the set of feasible distributions. Assume that this set is a convex subset of $\{\omega : [0,1] \to \mathbb{R} \times \{0,1\}\}$, and that all $\omega \in \Omega$ have finite variance.

3.2 Solving the Benchmark Model

The benchmark model solved in this chapter is the final model solved in the previous chapter: the stochastic growth model under the assumption of incomplete markets and idiosyncratic employment shocks, first solved by Krusell and Smith (1998). This chapter continues to use the parametrisation of Den Haan et al. (2010). Later sections of the paper will go on to solve extensions of this model.

3.2.1 The Problem

The task at hand is to to determine a function which forecasts future aggregate capital K_{t+1} given current capital K_t, conditional on a realisation of productivity A_t.

Household savings are the source of all capital, so aggregate capital in any period is the sum of individual capital holdings:

$$K_t = K(\omega_t) = \int_0^1 k_{i,t}\, di \tag{3.3}$$

The level of aggregate capital in the next period, K_{t+1}, depends on the distribution ω_{t+1} in the next period. Given the current distribution ω_t, ω_{t+1} is found by application of the household policy function $f(.)$ to each household in ω_t. Therefore K_{t+1} is given by

$$K_{t+1}(\omega_t) \equiv K(\omega_{t+1}) = \int_0^1 f(k_{i,t}, e_{i,t}, A_t, K_t)\, di \tag{3.4}$$

Find an approximate rule for the future value K_{t+1} in terms of K_t near a given point $(K_t^\star, A_t^\star)$

for which the distribution $\omega^\star \in \Omega$, is known.

3.2.2 A Straightforward Approximation

As outlined above, aggregate capital K_t depends on the underlying distribution of individual households over wealth. There are many distributions which yield any given value of K_t. Without further assumptions it is likely that not all of these distributions will give rise to the same value of K_{t+1}.

Putting aside this observation for the moment, *assume* that there were an exact but unknown relationship $K_{t+1} = H(K_t)$[3]. Then a common approach to approximating it would be to choose a Taylor polynomial $P(K_t)$, for example a first order expansion:

$$H(K_t) \approx P(K_t) \equiv H(K_t^\star) + \left.\frac{dH}{dK_t}\right|_{K_t^\star} (K_t - K_t^\star) \tag{3.5}$$

The household wealth levels $k_{i,t}^\star$, their employment states $e_{i,t}^\star$ and the function f are known by assumption. $H(K_t^\star) = K_{t+1}(\omega^\star)$, so by Eq. (3.4)

$$H(K_t^\star) = \int_0^1 f(k_{i,t}^\star, e_{i,t}^\star, A_t^\star, K_t^\star)\, di \tag{3.6}$$

This implies

$$\left.\frac{dH}{dK_t}\right|_{K_t^\star} = \left.\frac{d}{dK_t}\right|_{K_t^\star} \left(\int_0^1 f(k_{i,t}^\star, e_{i,t}^\star, A_t^\star, K_t^\star)\, di\right) \tag{3.7}$$

and, dropping the point of differentiation for notational ease:

$$\frac{dP}{dK_t} = \frac{dH}{dK_t} = \int_0^1 \left(\frac{\partial f}{\partial k_{i,t}} \frac{dk_{i,t}}{dK_t} + \frac{\partial f}{\partial K_t}\right) di \tag{3.8}$$

Note that $\frac{de_{i,t}}{dK_t}$ and $\frac{dA_t}{dK_t}$ are 0 due to exogeneity, and $\frac{dK_t}{dK_t} = 1$.

f is known and, by assumption, differentiable in both $k_{i,t}$ and K_t, so the partial derivatives

[3]Only the relationship of K_{t+1} to K_t is of interest at this point, so the explicit dependence of H on A_t is ignored. The remainder of this section is conditional on a particular realisation of A_t.

can be calculated. But what of $\frac{dk_{i,t}}{dK_t}$?

As defined in Eq. (3.3), K_t is the mean of the $k_{i,t}$. The causal direction of change is from $k_{i,t}$ to K_t, contrary to the partial derivative above. This makes the derivative counter-intuitive, but does not cause problems mathematically.

More problematically, there are many underlying changes in the distribution ω_t, and hence the individual $k_{i,t}$, which could cause the same change in K_t. For that reason, $\frac{dk_{i,t}}{dK_t}$ is in general not well defined. The assumption of a function H which yields a unique and correct K_{t+1} for any K_t breaks down.

That H does not exist was known in advance. That is why the task at hand is to find an approximate, not an exact, forecasting function. Equation (3.8) points to a possible approach: identify a 'representative' set of $\frac{dk_{i,t}}{dK_t}$, substitute them into the equation and calculate the gradient.

Derivative aggregation proposes to perform the first part of this approach by constructing a $C \subset \Omega$ such that, within C, the derivatives $\frac{dk_{i,t}}{dK_t}$ are well defined, and such that the polynomial P obtained from using those derivatives in the equation is not just a good approximation for H within C, but also provides a good forecast of K_{t+1} obtained given *any* $\omega_t \in \Omega$. Then P is the forecasting function sought. The subset C is necessarily a curve, and it is constructed below, after some further discussions regarding Eq. (3.8).

3.2.3 Exact Aggregation

There are two special cases when $\frac{dH}{dK_t}$ in Eq. (3.8) is uniquely identified by the model:

First, it may be the case that both partial derivatives take the same value for all $k_{i,t}$. Since the integral over the $\frac{dk_{i,t}}{dK_t}$ must be 1 by Eq. (3.3), $\frac{dH}{dK_t}$ would simply be the sum of the two partial derivatives. In this situation, a representative agent model *with the same decision rules as the disaggregated model* is a perfect representation of the economy, since all households react the same regardless of their individual state.

Second, it may be the case that for any K_t, there is a unique distribution $\omega \in \Omega$ which yields K_t. The $k^{\star}_{i,t}$ and the $\frac{dk_{i,t}}{dK_t}$ are then uniquely identified, so $\frac{dH}{dK_t}$ is too. A unique and precise forecasting rule exists. It need not generally be the case, however, that this forecasting rule is identical to the one derived from a representative-agent model with the same assumptions on individual behaviour. Thus the economy has an exact aggregate analogue, but a micro-founded model in which the representative agent has the same policy function as individuals in the disaggregated model may not be accurate.

3.2.4 Approximate Aggregation

Relax the assumptions of identical partial derivatives or a unique distribution for a given K_t a little and consider the findings of Carroll (2000) in this context. The author considers two models:

The benchmark Krusell & Smith model has aggregate behaviour close to that of a representative agent model, but wealth is too concentrated around its mean value relative to empirical observations. Aggregate co-movements do not match the data. This approximates the case of identical partial derivatives above: because wealth is closely distributed around the mean, the partial derivatives of individual decisions are close to their mean, so approximate aggregation holds and matches a representative agent model with those same partial derivatives quite closely.

The extended model yields a wealth distribution which matches the empirically observed one, including a significant proportion of households with very low wealth. Aggregate economic behaviour no longer matches that of an identically-calibrated representative agent model because poor households have marginal propensities to consume - the partial derivatives - far from the mean value. Approximate aggregation still holds because the wealth distribution only has a narrow range of realisations for any given K_t, approximating the second case above.

Equation (3.8) makes it clear that if both the underlying distributions which yield a given K_t and the partial derivatives of the individual policy rules vary widely then a good approximation will not be possible. Assuming that the approximation is possible, the equation implies that finding it requires finding representative values of the derivatives $\frac{dk_{i,t}}{dK_t}$.

3.2.5 A Graphical Illustration with Two Households

Consider the model with the number of households restricted to 2, indexed by $\{1, 2\}$. Aggregate capital is given by $K_t = k_{1,t} + k_{2,t}$. The objective is to find a first order-polynomial in K_t, $P(K_t)$, which yields a good approximation of the future aggregate capital K_{t+1} for any current state of the economy.

Figure 3.1 illustrates the proposed procedure. There are only two households, so the endogenous state of the economy can be represented as a point in the two-dimensional k_1, k_2 plane, shown in the top left graph. The shaded area is the feasible set of distributions Ω.

The first step is to identify a differentiable curve[4] $C(\theta)$ in Ω which approximately delineates its centre, and along which K_t is strictly monotonic. $\theta \in [\underline{\theta}, \bar{\theta}]$ indexes the curve, but has not economic meaning.

Each θ identifies a point on the curve. Hence $C(\theta)$ defines a pair of coordinate functions $\{k_{1,t}(\theta), k_{2,t}(\theta)\}$. Aggregating the two yields $K_t(\theta) = k_{1,t}(\theta) + k_{2,t}(\theta)$. This mapping is strictly monotonous by assumption, and can thus be inverted to $\theta(K_t)$, shown at the top right.

By assumption, the households' policy rule is known. $k_{i,t+1}(\theta) = f(k_{i,t}(\theta), .)$ can thus be calculated, and gives rise to $K_{t+1}(\theta) = f(k_{1,t}(\theta), .) + f(k_{2,t}(\theta), .)$, illustrated at bottom left.

Finally, the mapping $\theta(K_t)$ can be substituted into the mapping $K_{t+1}(\theta)$ to yield a function $K_{t+1} = \tilde{H}(K_t) \equiv K_{t+1}(\theta(K_t))$, which is represented in the bottom right quadrant.

How does $\tilde{H}$ relate to H in Eq. (3.8)? The functions are on the same domain and range, so that $\tilde{H}$ can replace H in the equation. The inverted function $\theta(K_t)$ can also be substituted into the coordinate functions to construct mappings $\{k_{1,t}(K_t), k_{2,t}(K_t)\}$. All the functions derived from the curve C in this section inherit its assumed differentiability. Along the curve C, therefore, $\frac{dk_{i,t}}{dK_t}$ are well-defined:

[4]See Simon and Blume (1994), page 313, for a formal definition of a curve. A curve is a one-dimensional, connected set which can be indexed by a real number.

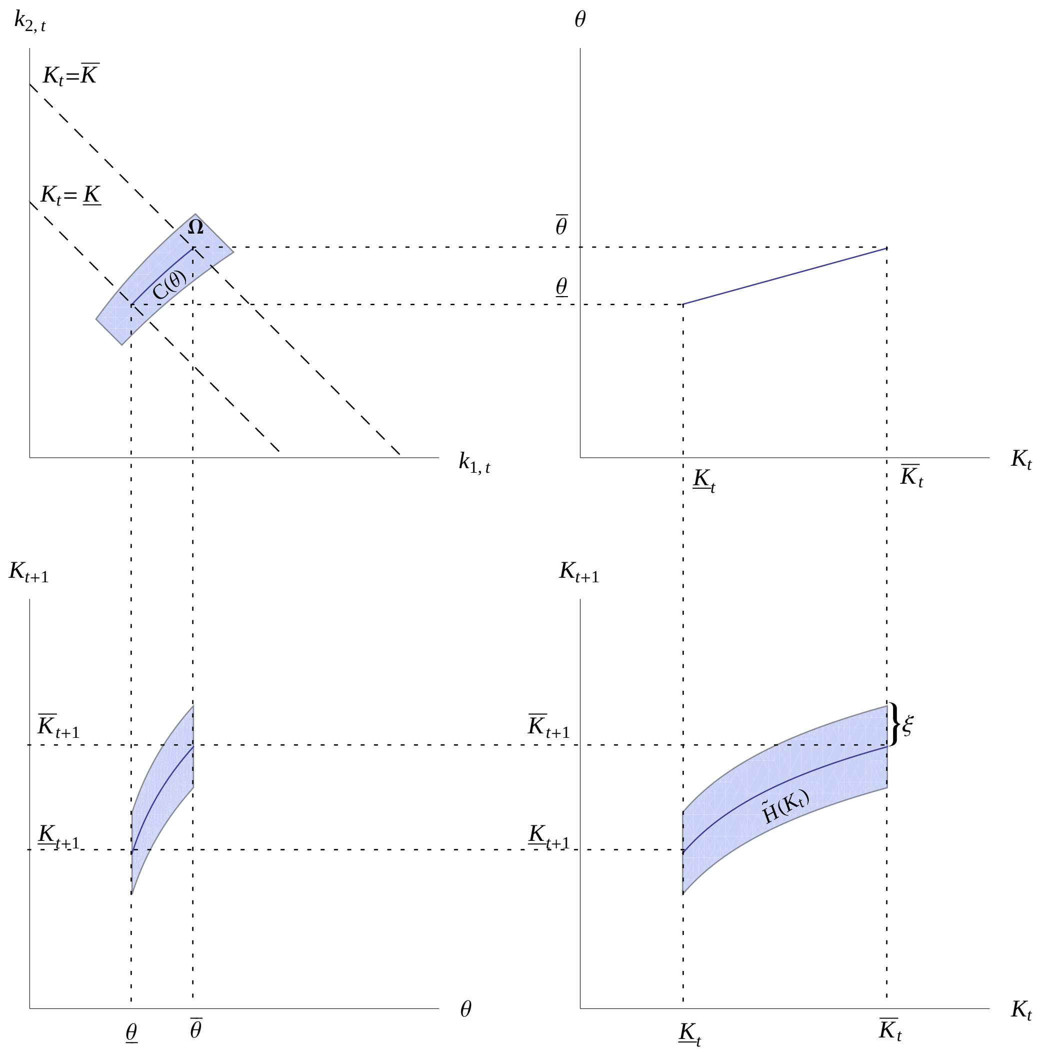

Figure 3.1: An Illustration of the Two-Household Solution
The curve $C(\theta)$ runs along the centre of Ω in the direction of increasing K_t. Each θ along the curve identifies a unique K_t (top right) and a K_{t+1} (bottom left). Since the former relation is monotonic a mapping $\tilde{H}$ from K_t to K_{t+1} results.

$$\frac{dk_{i,t}}{dK_t} = \frac{dk_{i,t}}{d\theta}\frac{d\theta}{dK_t} = \frac{\frac{dk_{i,t}}{d\theta}}{\frac{dK_t}{d\theta}} \tag{3.9}$$

The derivative in Eq. (3.8) can be calculated:

$$\frac{dP}{dK_t} = \frac{\frac{\partial f}{\partial k_{1,t}}\frac{dk_{1,t}}{d\theta} + \frac{\partial f}{\partial k_{2,t}}\frac{dk_{2,t}}{d\theta}}{\frac{dK_t}{d\theta}} + \left.\frac{\partial f}{\partial K_t}\right|_{k_{1,t}} + \left.\frac{\partial f}{\partial K_t}\right|_{k_{2,t}} \tag{3.10}$$

It is now straightforward to construct the Taylor polynomial $P(K_t)$. This polynomial is the approximate forecasting function sought.

3.2.6 Returning to the Full Distribution

The illustrated approach extends readily to the model with a continuum of households, rather than two. $C(\theta)$ becomes a curve in an infinite-dimensional space.

Before returning to the details, consider what this curve represents: in an informal way, a point on the curve is the most likely distribution to arise which aggregates to its capital level K_t. For two very close levels of aggregate capital, and again informally, these distributions are likely to be very similar[5], perhaps resulting from a different aggregate shock some relatively large number of periods in the past.

Since i has no economic meaning, assume without loss of generality that, in each distribution along C, i orders the households by wealth. Each coordinate function then specifies how a particular percentile of the wealth distribution changes along C.

Further assume that if $k_{i,t} = k_{j,t}$ at one point along the curve then this identity also holds at other points along the curve. This assumption may seem innocuous at first glance since households have identical preferences, but it is in fact almost certainly not literally true: for example, as θ and hence K_t increases one would expect some, but not all, households at the

[5]Similar in the sense that probability masses and probability densities, are close at all points.

borrowing limit to become unconstrained. Nonetheless, it is a useful simplifying assumption because it allows the curve to be identified in two steps:

1. Identify a point ω_1 on the curve and, without loss of generality, assume that this point corresponds to $C(1)$ (i.e. $\omega_1 = C(1)$).
2. Define the curve through a function $T : \mathbb{R}^2 \to \mathbb{R}$ s.t $k_{i,t}(\theta) \equiv T(k_{i,t}(1), \theta)$.

3.2.7 Choosing a Curve

The curve C has only been discussed in informal terms, leaving the question of its selection open. Derivative aggregation does not provide a precise answer to this question: selecting the curve is a heuristic activity. Later sections do discuss some methods for discovering better curves, but a researcher applying derivative aggregation must form at least an initial guess based on his understanding of the model at hand.

With that in mind, consider the two steps introduced in the last section in the context of the benchmark model.

3.2.7.1 The Reference Distribution

Step 1. calls for selection of a point which lies somewhere along the centre of the unknown set Ω. In other words, a distribution that is average. The exogenous process for aggregate productivity implies equal probability for high and low productivity periods. An average distribution results from an economic history which includes a roughly equal number of each level of productivity, especially in more recent periods.

Absent a solution to the model, such a distribution can not be constructed directly. On the other hand, a model with a similar structure of idiosyncratic shocks but no aggregate uncertainty can be solved with known methods, since it does not require that an aggregate forecasting function be found. Individual households face greater risk from idiosyncratic shocks than aggregate shocks in the benchmark model. Abstracting from the latter should not change their behaviour to a significant degree. The steady-state distribution of that model should be quite similar to the average distribution sought.

This distribution is the initial choice for ω_1[6].

3.2.7.2 The Transformation

Consider that the economy is a standard production economy. A small variation in current aggregate capital K_t is the result of a small variation in the history of economic shocks. Both aggregate shocks and capital affect households through factor prices: the wage and the interest rage. A slightly higher K_t means that, all other things being equal, all households received slightly more income historically. The assumption that all households should presently also have slightly more wealth is therefore not unreasonable.

Higher wages affect all households equally, whereas higher interest rates provide more additional income for those households with higher wealth. More wealthy households also already have higher consumption and, given the standard utility function exhibiting diminishing marginal utility, can be expected to spend less of any additional income on immediate consumption. Thus the further assumption that the increase in household wealth resulting from a marginally better history of aggregate shocks should increase with the wealth held under the lower-shock scenario is also justified.

Based on this reasoning, assume that the increase in wealth is proportional to the wealth held, so that

$$T(k_{i,t}, \theta) = \theta k_{i,t} \tag{3.11}$$

The calculation only requires $dT/d\theta$ rather than T itself, so in this case the input provided is

[6]Since the number of unemployed differs between the two aggregate states in the benchmark model, the no-aggregate-risk model used has a level of unemployment lying between the two. The reference distributions - one for each aggregate state - are constructed by adjusting the unemployment rate in the steady state solution of this model.

$$\frac{dT(k_{i,t},\theta)}{d\theta} = k_{i,t} \tag{3.12}$$

Recall that

$$K_t = \int_0^1 k_{i,t}\ \mathrm{d}i \qquad \Rightarrow \qquad \frac{dK_t}{d\theta} = \int_0^1 k_{i,t}\ \mathrm{d}i = K_t \tag{3.13}$$

The derivative of the polynomial is then:

$$\frac{dK_{t+1}}{dK_t} = \frac{dP}{dK_t} = \frac{1}{K_t}\int_0^1 \frac{\partial f}{\partial k_{i,t}} k_{i,t}\ \mathrm{d}i + \int_0^1 \left.\frac{\partial f}{\partial K_t}\right|_{k_{i,t}} \mathrm{d}i \tag{3.14}$$

The next section evaluates the solution of the model obtained by repeatedly updating, until convergence, the aggregate forecasting function using the formula above to calculate its gradient.

3.2.8 Assessing Performance and Accuracy

Den Haan (2010b) compared different solution methodologies for heterogeneous agent models. He applied them to a particular calibration of the benchmark model and compared the solution time and a number of metrics to evaluate the relative accuracy of solutions. He found the Explicit Aggregation approach of Den Haan and Rendahl (2010) to provide the best combination of speed and accuracy. This section presents the results from solving the model by Derivative Aggregation under the same calibration[7] and compares a number of metrics to the Explicit Aggregation solution.

[7]The calibration of the model is documented in Section 2.6.3.

3.2.8.1 Solution Time

To perform the comparison under similar conditions the Explicit Aggregation algorithm was implemented using the ModelSolver Toolkit, and the solutions were computed using the same individual capital levels, individual shock levels and aggregate shock levels. The aggregate capital levels differ because Explicit Aggregation requires that they be separated into two variables, the capital held by employed and unemployed agents, whereas Derivative Aggregation uses only one variable for aggregate capital. The calculations were performed on the same machine with no other user processes running.

Computing the solution using Explicit Aggregation required approximately 89 seconds. The Derivative Aggregation approach took around 13 seconds.

The computation of the aggregate forecasting function under Explicit Aggregation in this model is trivial: at each point of the aggregate variable grid, the individual policy is evaluated for an employed and an unemployed household under the assumption that the households have wealth levels equal to the average for their employment status. Updating the forecasting function under Derivative Aggregation is computationally more complex. The difference in performance hence stems from two sources: first, the size of the grid is smaller using Derivative Aggregation because it does not require two aggregate variables. Second, the solution converges in fewer steps, requiring just 225 for Derivative Aggregation compared to 423 for Explicit Aggregation.

3.2.8.2 Aggregate Forecast Accuracy

		10000 period ahead (%)			*1 period ahead (%)*			
	Time (s)	Bias	Mean	Max	Bias	Mean	Max	R^2
XPA	89	0.065	0.119	0.383	0.003	0.008	0.049	99.9992
DA	13	-0.054	0.077	0.391	-0.002	0.003	0.019	99.9998

Table 3.1: Compute Time and Forecast Accuracy
(XPA) Explicit Aggregation
(DA) Derivative Aggregation

Both Derivative Aggregation and Explicit Aggregation are methods for computing the aggregate forecasting function[8]. The metrics of greatest interest therefore concern the accuracy of aggregate forecasts produced.

To evaluate the forecasting functions, an identical sequence of 10000 shocks is simulated using the individual policy rules produced by each solution. Subsequently, aggregate variables are forecast over the long-term and the short-term: the long-term forecast predicts a sequence of 10000 aggregate variables forecast from the starting point of the simulation, under the assumption of the shock sequence simulated; the short-term forecast predicts the aggregates for each period based on the simulated aggregates from the prior period, and is a sequence of 10000 one-period-ahead forecasts.

Table 3.1 presents the summary statistics of the forecast errors from the two solutions. On average, both the long-term and short-term forecast errors resulting from Derivative Aggregation are approximately 40% smaller than those obtained from Explicit Aggregation. Both solutions show a bias in their forecasts which is relatively large relative to the average error. Under the metric used by Krusell and Smith (1998) and many subsequent authors[9], the R^2 of short-term forecast errors, both solutions perform well with values in excess of 0.99999.

The long-term forecast errors from both solutions around the point of their maximum[10] under Derivative Aggregation are shown in Fig. 3.2. The error of the Explicit Aggregation solution is also relatively large at the maximum point. The errors decrease again rapidly. The tendency of the values from Derivative Aggregation to lie below those of Explicit Aggregation in this region is a reflection of the opposing bias of the two time series.

Figure 3.3 shows a scatter plot of the short-term forecast errors against the capital levels from which those forecasts were formed. Derivative Aggregation, in common with all other approaches discussed, constructs a short-term forecasting function, so considering these errors may yield insights into improving the solution. The plots illustrate that there is still a

[8]Strictly speaking they are methods for computing the aggregate forecasting function implied by an individual policy function, but used iteratively in conjunction with a method for calculating the individual policy function they are used to compute the forecasting function

[9]This metric is found to be an inadequate measure of the accuracy of the forecasting function in Den Haan (2010a)

[10]All maxima and means discussed refer to absolute numbers.

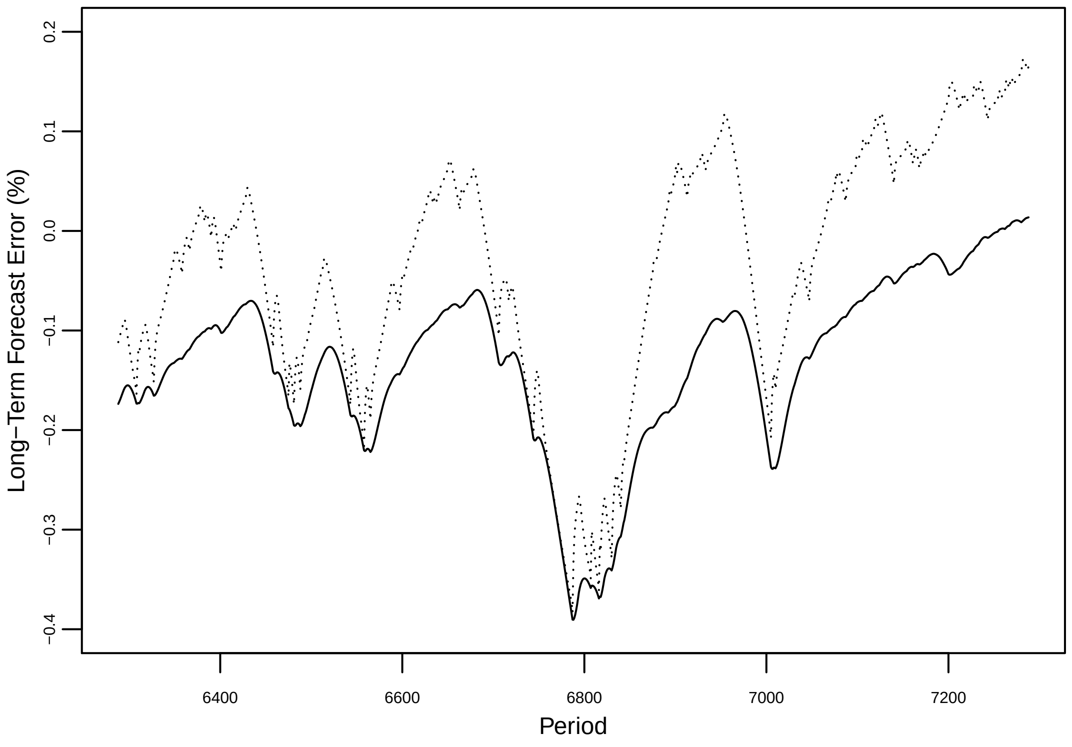

Figure 3.2: Relative Errors in the Region Around the Maximum Error for DA (solid) and XPA (dotted)

linear component remaining in the error, so the gradient was not calculated perfectly. Both an inaccurate reference distribution and a poorly chosen transformation could contribute to such an error.

The point at which the approximation is calculated is shown on the graphs as the large dot. This point is neither in the middle of the range of K_t, nor is the forecast at that point very accurate. This aspect of the solution is most likely due to a poorly chosen reference distribution.

Later sections investigate approaches for improving both the reference distribution and the transformation.

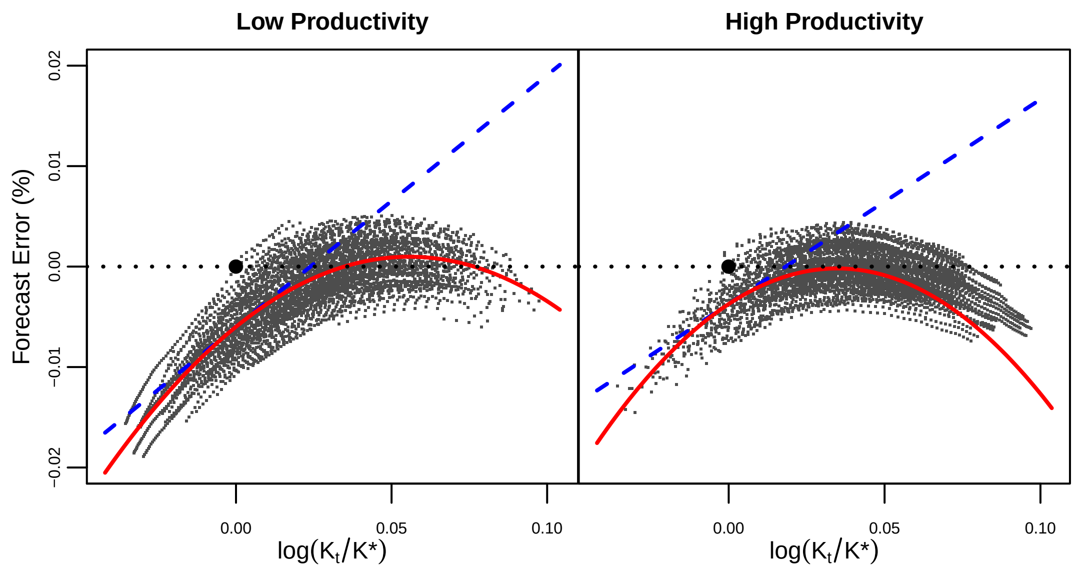

Figure 3.3: One-Period-Ahead Forecast Errors (%), Derivative Aggregation
The solid and dashed lines show a quadratic curve fitted to the errors and its linear component. The large dot indicates the point at which the approximation was taken.

3.2.8.3 The Impact on Individual Outcomes

One interpretation of bounded rationality is utilitarian: households choose to restrict their information set because doing so imposes very little cost. This interpretation requires that the decisions households make using the approximate forecasting function are close to those optimal under a true, full-information solution.

Den Haan (2010b) introduces the *dynamic Euler-equation accuracy test* to assess whether or not this condition is met. The data for this test is calculated during the simulation described above. For a single household, two alternative time-series of consumption and, by implication, capital holdings are generated: first, the time series implied by the individual policy function obtained as part of the solution; second, a time-series obtained by deriving consumption each period from the Euler equation when expected consumption in the next period is calculated based on the expected aggregates from the forecasting function in the next two periods[11].

Table 3.2 presents summary statistics of the difference between the time series for the

[11]A more complete description is in Section A.5.

	k			c		
	Bias	Mean	Max	Bias	Mean	Max
XPA	-0.052	0.052	0.109	-0.006	0.006	0.102
DA	-0.052	0.052	0.107	-0.006	0.006	0.106

Table 3.2: Dynamic Euler Equation Errors (%)
(XPA) Explicit Aggregation
(DA) Derivative Aggregation

two solutions. The metrics are almost identical, reflecting the shared approach to solving the individual problem. The mean difference in consumption over the full length of the simulation is only $.006\%$, with the maximum difference at $.1\%$. The households decisions using the approximate forecasting function are indeed close to optimal.

3.3 Some Experiments to Improve the Solution

The results presented in the previous section compare well with those obtained from alternative approaches. Nonetheless, the analysis of forecast errors indicates that there is potential for improvement, particularly in the heuristic choices made: the reference distribution is not central in the range of K_t and does not produce a K_{t+1} average for its K_t, and the forecast errors have a remaining first-order component which is large relative to the spread of errors. In other words, the curve C implied by the two choices does not lie in the middle of Ω.

A secondary consideration is that the errors also show a 2nd-order component, which suggests that a first-order solution may not be optimal.

This section presents some ideas for improving the choices for the reference distribution and the transformation, and goes on to evaluate solutions that apply those ideas as well as a second-order approximation.

3.3.1 Discovering a Reference Distribution by Simulation

Simulating an economy with heterogeneous agents is computationally expensive. The results from Den Haan (2010b) indicate that simulation-based solutions are much less efficient than those which avoid it, thus the absence of simulation in the approach presented so far is partly responsible for its efficiency.

On the other hand, the assumed reference distribution is not optimal and finding a new one requires some simulation. The first experiment introduces a small amount of simulation aimed at improving the distribution. As already stated, the aim is to produce an average distribution. In this economy each aggregate state occurs with equal probability[12]. The exogenous process for aggregate productivity has a mean duration of 8 periods in each state. When the economy is in a particular state, it is likely to have been in that state for a few periods.

Following this reasoning, the reference distributions used are updated as follows in each iteration: the economy is simulated for 48 periods, starting from the reference distribution of the last iteration, and alternating between aggregate states every 8 periods. The reference distribution used to compute the derivatives for each aggregate state is sampled after four periods in that state toward the end of the simulation. The simulation in each iteration continues from the final state of the prior iteration.

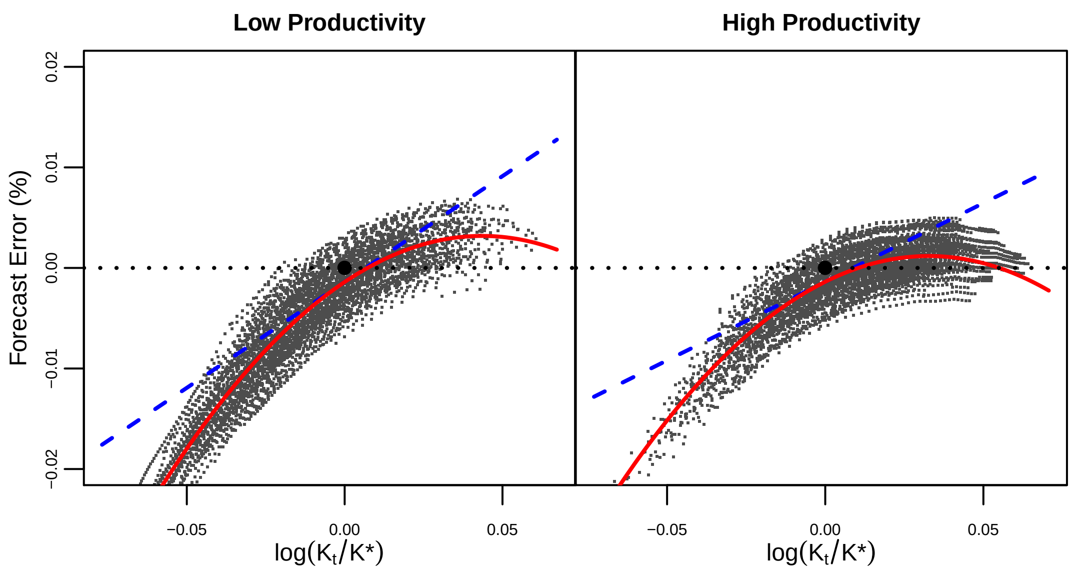

Figure 3.4: One-Period-Ahead Forecast Errors (%), Derivative Aggregation with Reference Distribution Discovery by Simulation

[12]Unconditionally.

Figure 3.4 plots the forecast errors from simulating the resulting solution. The aim of finding an average distribution has been achieved: the point at which the approximation is taken has indeed moved to a more central location in the range of K_t and produces a forecast representative for its level of that variable.

Conversely, the aim of decreasing forecast errors has not been satisfied. The errors at low levels of K_t have increased in magnitude. Table 3.3 shows that the average forecast error has also increased. This increase is accompanied by a corresponding increase in the bias.

The poorly-chosen reference distribution in the baseline solution was offsetting the effects of the imperfect gradient and the absence of a quadratic approximation. Improving the choice of distribution has counter-productive effects on accuracy.

		10000 period ahead (%)			*1 period ahead (%)*			
	Time (s)	Bias	Mean	Max	Bias	Mean	Max	R^2
(1)	13	-0.054	0.077	0.391	-0.002	0.003	0.019	99.9998
(2)	57	-0.101	0.128	0.641	-0.004	0.005	0.028	99.9993

Table 3.3: Compute Time and Forecast Accuracy
(1) Derivative Aggregation
(2) DA with Distribution Improvement

3.3.2 Discovering a Transformation by Comparing Distributions

The purpose of the transformation T is to capture the relative rates of change of the wealth of different individual households as aggregate capital changes. During the simulation described in the last experiment a number of distributions are generated with different but quite close levels of aggregate capital. By comparing those distributions a new transformation can be constructed.

The procedure works as follows: Consider two distributions $\omega^A = \{k^A_{i,t} : i \in [0,1]\}$, and $\omega^B = \{k^B_{i,t} : i \in [0,1]\}$. To construct an equivalence between the indexes, assume without loss of generality that i orders the households in each distribution by wealth. Further assume that household i in distribution ω^A is the same as household i in distribution ω^B, but after a slightly different history. Finally, assume that the change in θ when moving from ω^A

to ω^B is 1. This assumption also imposes no loss of generality since θ is scale-free. Then $dT(k_{i,t}, \theta)/d\theta \approx k_{i,t}^B - k_{i,t}^A$ at ω_A.

Figure 3.5 plots the derivative obtained by updating it in each iteration using this approach. The initial assumption of a function linear in $k_{i,t}$ with intercept 0 was somewhat off the mark: at low $k_{i,t}$ the function is strongly curved, and extrapolating from the broadly linear section at higher $k_{i,t}$ implies a non-zero intercept.

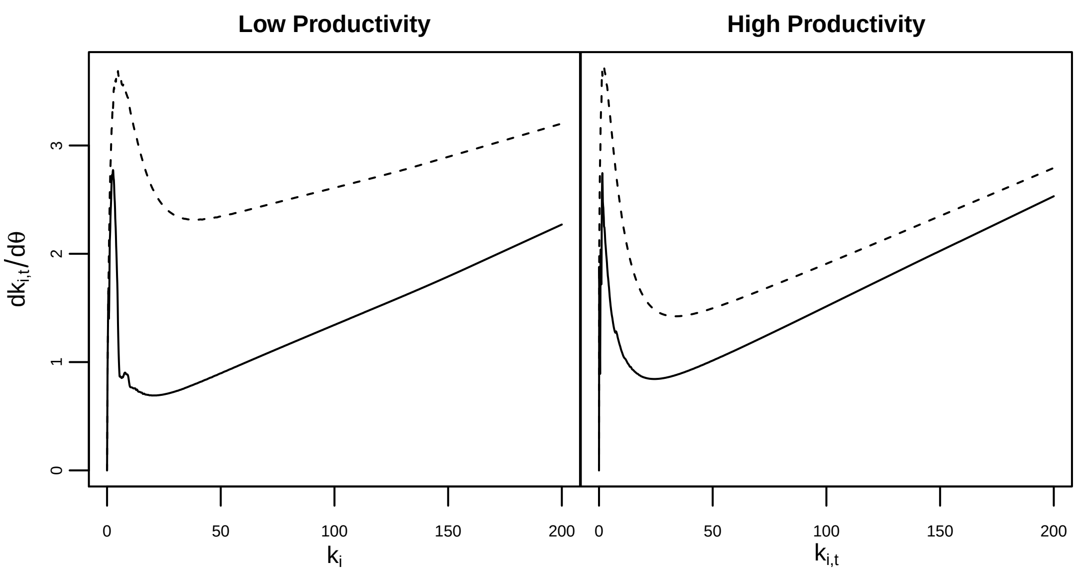

Figure 3.5: The Derivative of the Transformation, $\frac{dT}{d\theta}$, for Employed (*solid*) and Unemployed (*dashed*) Households

At $k_{i,t}$ greater than approximately 150 the function is determined by linear extrapolation from lower levels because the numerical method used to obtain the derivative breaks down at those wealth levels, where the density of households is both very low and lumpy. Since the proportion of the population in this region is less than $.01\%$ this imposition likely has very little effect.

Figure 3.6 plots the forecast errors from simulating the solution obtained using that derivative. The point of approximation remains central. In comparison to the corresponding plots from the previous experiments (Figs. 3.3 and 3.4) the linear component of the curve fitted to the errors has a visually lower gradient, suggesting that the first-order derivative has been captured more accurately.

Table 3.4 shows the summary statistics, in comparison to those from the baseline experiment. The mean error is elevated and the bias is substantially higher, though both have im-

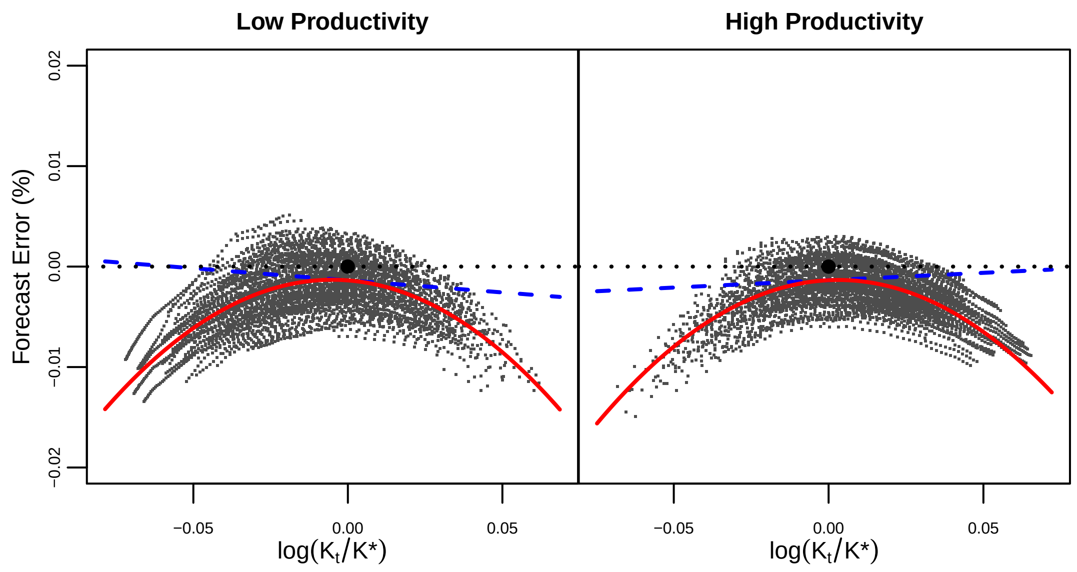

Figure 3.6: One-Period-Ahead Forecast Errors (%), Derivative Aggregation with Reference Distribution and Transformation Discovery by Simulation

		10000 period ahead (%)			*1 period ahead (%)*			
	Time (s)	Bias	Mean	Max	Bias	Mean	Max	R^2
(1)	13	-0.054	0.077	0.391	-0.002	0.003	0.019	99.9998
(2)	70	-0.077	0.084	0.285	-0.003	0.003	0.015	99.9998

Table 3.4: Compute Time and Forecast Accuracy
(1) Derivative Aggregation
(2) DA with Distribution & Transformation Improvement

proved compared to the previous experiment. The plot of errors suggests that the bias results from the second-order component of the errors.

3.3.3 A Second-Order Approximation

The experiments above demonstrate that much of the remaining forecast inaccuracy stems from second-order effects. Derivative Aggregation can be applied to find a polynomial approximation of any order. The theoretical section which follows presents the mathematics for the second order approach.

The second order solution calculation assumes $d^2T(.)/d\theta^2$ to be 0 everywhere, so that the transformation remains linear. The numerical approach used for transformation discovery of the first order does not extend well to second-order approximation. Other approaches remain to be investigated.

Fig. 3.7 presents the forecast errors which result from simulating the solution of the model using both distribution and transformation discovery and a second order approach. Visually, the second order component is less pronounced relative to the last experiment, and the errors appear more closely clustered around 0. Other aspects of the plot remain unaffected.

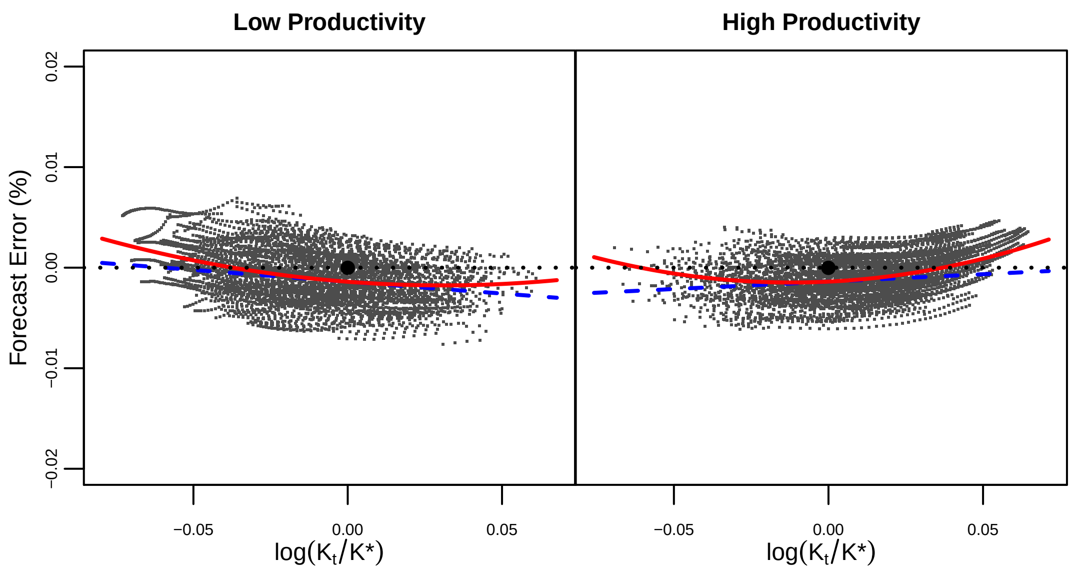

Figure 3.7: One-Period-Ahead Forecast Errors (%), Derivative Aggregation with 2nd-Order Approximation and Transformation Improvement

The metrics in Table 3.5 confirm this visual impression. The mean error is approximately 25% lower than in the baseline experiment, the bias has decreased by a similar absolute amount, and the R^2 has acquired an extra 9 to read 0.999999.

The table also highlights the cost: computation time increases by a factor of around 5.5 even with the small amount of simulation added to the solution approach. Each additional improvement in solution accuracy comes at the cost of further compute time, although the incremental cost of the second-order approximation itself is small.

		10000 period ahead (%)			*1 period ahead (%)*			
	Time (s)	Bias	Mean	Max	Bias	Mean	Max	R^2
(1)	13	-0.054	0.077	0.391	-0.002	0.003	0.019	99.9998
(2)	57	-0.101	0.128	0.641	-0.004	0.005	0.028	99.9993
(3)	70	-0.077	0.084	0.285	-0.003	0.003	0.015	99.9998
(4)	74	-0.023	0.053	0.156	-0.001	0.002	0.008	99.9999

Table 3.5: Compute Time and Forecast Accuracy, Four experiments:
(1) Baseline Derivative Aggregation (DA)
(2) DA with Reference Distribution Discovery
(3) DA with Reference Distribution and Transformation Discovery
(4) 2nd-Order DA with Reference Distribution and Transformation Discovery

3.4 Generalising the Approach

The approach used above to solve the benchmark model is not specific to that model. It is applicable to a wide range of models having heterogeneous agents wishing to form, and act according to, model-consistent expectations about aggregate quantities. This section first briefly outlines the generic form of models to which Derivative Aggregation can be applied, and then presents the mathematical formulae for computing the aggregate forecasting functions. These formulae are completely generic and can be implemented without knowledge of the model. An implementation of `AggregateProblemSolver` which performs 1st or 2nd order Derivative Aggregation is provided as part of the ModelSolver Toolkit presented in Chapter 2.

3.4.1 A Brief Digression on Notation

This section outlines the notation used to discuss models in the following sections.

An individual i in the economy may have both endogenous and exogenous states, identified by x_i and a_i:

$$x_i = (x_i^1, x_i^2, \ldots, x_i^{n_x}) \tag{3.15}$$

$$a_i = (a_i^1, a_i^2, \ldots, a_i^{n_a}) \tag{3.16}$$

A *distribution* ω describes the overall state of the economy, and is a mapping of individual agents, identified by their index, to their state variables. In the case of a continuum of agents:

$$\omega : [0,1] \to \mathbb{R}^{n_a+n_x} \equiv \{(x_i, a_i) : i \in [0,1]\} \tag{3.17}$$

Note that it is common to represent such a distribution instead as a probability density function over the individual state space. Distributions as used here may, however, have non-zero point probabilities, for instance at constraint boundaries. The notation above supports that, and the mathematics used in this paper is easier to present in this form.

The economies discussed may have additional *aggregate* exogenous states A:

$$A = (A^1, A^2, \dots, A^{n_A}) \tag{3.18}$$

Finally, the models will also specify aggregate endogenous variables, which are scalar functions of the underlying distribution ω.

$$X = (X^1, X^2, \dots, X^{n_X}) \tag{3.19}$$

where

$$X^j \equiv X^j(\omega) \equiv \int_0^1 F^j(x_i, X^{-j})\,\mathrm{di} \tag{3.20}$$

$$\text{for some} \quad F^j : \mathbb{R}^{n_x} \times \mathbb{R}^{j-1} \to \mathbb{R} \tag{3.21}$$

Endogenous aggregates are defined in this way to allow for dependencies between aggregates that are non-circular, accepting for example centred moments[13].

[13]There may also be an invertible scalar function g applied to the whole integral, so that $X^j(\omega) =$

Note that all variables, aggregate or individual, may be indexed by the discrete time t to indicate which period they apply to.

3.4.2 A Generic Model

Consider an economy populated by a continuum[14] of decision-making agents of measure 1[15], indexed by $i \in [0,1]$. Time is discrete and indexed by $t \in \{0,1,2,\cdots\}$. Economic activity continues for a number of periods, and is subject to exogenous shocks both at the aggregate and individual level. Agents are forward looking, taking potential future outcomes into consideration when making decisions in the current period. All problems can be expressed and solved recursively, so that only the two period form need be considered.

Agents must decide, one period in advance, on their vectors of private endogenous state variables $x_{i,t+1}$ given their current endogenous and exogenous state vectors $x_{i,t}$ and $a_{i,t}$. Conditional on these states, agents have identical preferences[16].

Further assume that no agent is directly affected by any other agents' private variables or decisions, but that agents do take into account vectors of exogenous and endogenous aggregate variables A_t and X_t respectively[17].

Finally assume that there is a known, time-invariant stochastic process that governs the joint transition of all individual and aggregate exogenous variables, and that the distribution of agents over their states always has finite variance.

$g\left(\int_0^1 F^j(x_i, X^{-j})\,\mathrm{d}i\right)$. Presenting this function introduces additional complexity without adding to the argument, so it is omitted throughout. Allowing such a function caters, for example, for log-linear approximations in the aggregates.

[14]The method presented could also be applied to a countably infinite or a finite set of agents, but this paper focuses only on the case of a continuum to simplify the exposition, particularly where integrals are used.

[15]In principle the measure could be any size and vary over time, but restricting it to 1 simplifies the exposition here without losing any insight.

[16]The exogenous states, as defined, allow for agents to have permanently different states and hence for ex-ante as well as ex-post heterogeneity, so that this conditional identity does not constrain the model in any way.

[17]There are no constraining relationships between the cardinality of any of the vectors. There may, for example, be more aggregate than individual endogenous variables.

3.4.3 The Forecasting Function

In an economy as described, individual agents follow a common, time-invariant policy function $f_x(.)$ which determines their future endogenous state given their current state and the state of the aggregate economy:

$$x_{i,t+1} = f_x(x_{i,t}, a_{i,t}, X_t, A_t) \tag{3.22}$$

Agents are forward looking, so this rule takes possible future outcomes into account, and agents must forecast all variables which determine those outcomes. As before, agents are assumed to be boundedly rational: they forecast future aggregate states based on the restricted information set of current aggregate states[18]. They are further assumed to seek a polynomial form of this function. The forecasting function sought therefore takes the form

$$X_{t+1} = P(X_t, A_t) \tag{3.23}$$

where P is a polynomial in X_t. If A_t are continuous variables then P is also assumed to be a polynomial in those variables, otherwise P is determined conditional on each possible value of A_t.

3.4.4 The Overall Approach

As demonstrated for the case of a single aggregate variable in Section 3.2, this approach determines an n^{th}-order polynomial forecasting function by aggregating the derivatives of the individual policy rules along a curve. In the more general case with n_X aggregate states, the domain of the polynomial P is n_X-dimensional and the algorithm thus requires identification

[18]Aggregate states may include variables that are not directly relevant economically but help with forecasting, such as moments of the individual state distribution.

of an n_X-dimensional manifold[19] within Ω.

The overall approach is as follows:

1. Identify a point $\omega^\star$, the reference distribution, approximately in the centre of Ω.
2. Identify a set of transformations $T = \{T_k : 1 \leq k \leq n_X\}$, under each of which only one of the current aggregate state variables changes. The curves mapped out by these transformations effectively form the coordinate system of the manifold.
3. Calculate the derivatives of the polynomial P using the formulae derived in the next section.
4. Construct the polynomial from the derivatives and the current and future aggregate variables implied by the reference distribution.

3.4.5 The Mathematics

Proposition 1 details how the first order derivatives of P may be calculated:

Proposition 1. *Let $T_k(x, \theta_k)$, $T_k : \mathbb{R}^{n_x} \times \mathbb{R} \to \mathbb{R}^{n_x}$, be a parametrised transformation of the individual endogenous variables. Denote by $\mathbb{T}_k(\omega, \theta_k) = \{(T_k(x_i, \theta_k), a_i) : (x_i, a_i) \in \omega\}$ the transformed distribution which results from applying $T_k(., \theta_k)$ to each individual in the distribution ω.*

Assume that

1. *X^k changes under $\mathbb{T}_k(., \theta_k)$, but all other aggregates do not:*

$$X^l\left(\mathbb{T}_k(\omega, \theta_k)\right) = X^l(\omega) \quad (\forall \omega \in \Omega, \theta_k \in \mathbb{R}) \qquad \Leftrightarrow \qquad l \neq k$$

2. *$T_k(x, \theta_k)$ is continuously differentiable in θ_k for all $x \in \mathbb{R}^{n_x}$ and θ_k in some interval that includes 1, at least as a boundary*

[19]A manifold is a topological space which locally resembles a Euclidean space at all points. In this cases, the curves sought map precisely to the axes of the corresponding n_x-dimensional Euclidean space.

3. *$T_k(., 1)$ is the identity:*

$$T_k(x, 1) = x \quad \forall x \in \mathbb{R}^n_x$$

and, by implication,

$$\mathbb{T}_k(\omega, 1) = \omega \quad \forall \omega \in \Omega$$

Then, for any distribution ω, as that distribution is transformed by $\mathbb{T}_k(., \theta_k)$ for θ_k moving away from 1, the derivative of the future aggregate state variable X^j_{t+1} with respect to the current aggregate state variable X^k_t is given by:

$$\begin{aligned}\frac{dX^j_{t+1}}{d\tilde{X}^k_t} = &\sum_{s=1}^{n_x} \int_0^1 \left(\frac{\partial F^j(.)}{\partial x^s} \left[\sum_{r=1}^{n_x} \frac{\partial f^s_x}{\partial \tilde{x}^r_{i,t}} \frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t} + \frac{\partial f^s_x}{\partial \tilde{X}^k_t} \right] \right) \mathrm{di} \\ &+ \sum_{s=1}^{j-1} \int_0^1 \left(\frac{\partial F^j}{\partial X^s} \frac{dX^s_{t+1}}{d\tilde{X}^k_t} \right) \mathrm{di}\end{aligned} \tag{3.24}$$

where

$$\frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t} = \frac{\dfrac{\partial T^r_k(x_{i,t}, 1)}{\partial \theta_k}}{\displaystyle\sum_{s=1}^{n_x} \int_0^1 \frac{\partial F^k(x_{l,t}, X^{-k})}{\partial x^s} \frac{\partial T^s_k(x_{l,t}, 1)}{\partial \theta_k}\, dl} \tag{3.25}$$

and

$$\tilde{x}_{i,t} \equiv T_k(x_{i,t}, \theta_k) \tag{3.26}$$

$$\tilde{X}^k_t \equiv X^k_t(\mathbb{T}_k(\omega, \theta_k)) \tag{3.27}$$

The proof is in Section A.1.

Note that all the partial derivatives in both equations of the proposition are of the aggregation functions F^j, the transformations T_j and the individual transition rule f_x, and the aggregation is performed over the distribution ω. The first two functions and the distribution are parameters of the model and the solution approach, and f_x is assumed known at the time of calculation. The calculation of these partial derivatives and the subsequent integration are straightforward numerical computations.

3.4.6 Exogenous Aggregates

In many models, among them the examples in this paper, exogenous aggregates take discrete values. In those cases, the derivatives with respect to endogenous aggregates can be performed conditional on the value of the exogenous variables.

With continuous exogenous aggregates, calculating derivatives of next-period endogenous states with respect to the exogenous ones is straightforward, since a change in an exogenous state does not necessitate a change in the underlying distribution. There is no need to identify a transformation along which to calculate derivatives, and the only terms remaining in Eq. (3.24) are those including derivatives with respect to the aggregate.

3.4.7 Higher-Order Approximations

Proposition 1 outlined the formulae for first order approximations using derivative aggregation. Both the proposition and the proof rely solely on elementary manipulation of derivatives, and the approach can readily be extended to higher order approximations. The resulting formulae are lengthy, and deriving problem-specific solutions from the aggregation functions and transformations can also be cumbersome, but the higher order approach does not make the computation significantly more difficult. Section 3.3 demonstrated the benefits of a second-order solution. The general formulae for second-order approximations are derived in Appendix A.2, along with the specific solutions for the benchmark model.

3.5 Adding a One-Period Bond

The benchmark model is a relatively basic model of incomplete markets and heterogeneous agents in that it allows investment in only one asset, risky capital. It is useful as a starting point to demonstrate the methodology, particularly since there is a broad literature of existing solutions that can be used for performance and accuracy comparisons.

This section adds a second asset, risk-free bonds, to the economy, and allows agents to trade in these bonds among themselves. The structure of the model follows Krusell and Smith (1997), but the model continues to use the calibration described above for all variables shared by the two models.

3.5.1 The Extended Model

The production-side of the extended model economy is identical to that of the benchmark model. The sole addition is that households may exchange one-period, risk-free bonds, providing a second channel for risk mitigation that counters aggregate uncertainty. Following Krusell and Smith (1997), households continue to be constrained to non-negative capital holdings and are also subject to a constraint $\underline{b}$ on borrowing in bonds, which prevents them from building up arbitrary levels of debt.

This model hence adds a further constraint to the household problem, and also introduces the requirement to find the equilibrium market-clearing bond price in each period, so that net demand for bonds is 0.

3.5.1.1 The Household Problem

Household i's problem is

$$\max_{\{c_t^i, k_{t+1}^i, b_{t+1}^i\}_{t=0}^{\infty}} E\left[\sum_{t=0}^{\infty}\left(\beta^t \frac{(c_t^i)^{1-\gamma}-1}{1-\gamma}\right)\right] \tag{3.28}$$

$$\text{s.t. } c_t^i + k_{t+1}^i + p_t b_{t+1}^i = (1+r_t-\delta)k_t^i + b_t^i + [(1-\tau_t)\bar{l}a_t^i + \mu(1-a_t^i)]w_t \tag{3.29}$$

$$b_{t+1}^i \geq -\underline{\mathrm{b}} \tag{3.30}$$

$$k_{t+1}^i \geq 0 \tag{3.31}$$

where b_t^i is the face value of bonds carried over from period $t-1$ to period t by agent i and p_t is the market-clearing bond price in period t.

Solving the households' maximisation problem yields two first-order conditions

$$\beta E\left[\left(c_{t+1}^i\right)^{-\gamma}(1+r_{t+1}-\delta)\right] = \left(c_t^i\right)^{-\gamma} - \phi_{k,t}^i \tag{3.32}$$

$$\beta E\left[\left(c_{t+1}^i\right)^{-\gamma}\right] = \left(c_t^i\right)^{-\gamma} p_t - \phi_{b,t}^i \tag{3.33}$$

where $\phi_{k,t}^i, \phi_{b,t}^i \geq 0$ are the multipliers on the capital and bond borrowing constraints respectively.

3.5.2 Solving the model

The addition of the second asset, bonds, adds significantly to the difficulty of solving the model. This added complexity arises both when solving the individual problem and when solving the aggregate problem. In both cases an additional variable, bond holdings and the bond price respectively, are introduced. In the case of the individual problem the method of endogenous gridpoints, used in the benchmark model, which is very efficient and hence contributes to the rapid solution of the problem, becomes more difficult to apply.

3.5.2.1 The Additional Variables

As explained in Krusell and Smith (1997), the addition of bond holdings as an individual variable need not lead to an additional individual state variable. Since both assets are liquid and incur no transaction costs, a household is indifferent as to the composition of its portfolio at the beginning of the period and only its overall wealth level affects its decision. Thus there remains only one individual state variable, which is now individual total net wealth after interest, and the composition of the portfolio becomes a control variable which must be found but which does not appear as an input to the individual policy function.

Bond price is considered to be an aggregate control. The approach to working with aggregate controls has already been discussed in Chapter 2. The specific changes to the derivative aggregation mechanism are detailed in Appendix A.4.

3.5.2.2 Representing the Bond Price

An additional complication arises from having both aggregate capital, which affects the expected return on capital, and the bond price on the grid. The Euler conditions of the individual problem, Eqs. (3.32) and (3.33), illustrate that the relationship between these two variables is constrained. Since the solution approach attempts to solve the individual problem for each possible combination of values of the aggregate variables on the grid, these combinations must satisfy the constraint. If capital K_t and the bond price p_t were included on the grid as independent variables the constraint would be violated in some cases, for example when capital is very high, implying a low return on capital, but the bond price is very low, implying a high return on bonds. To avoid this problem the bond price itself is not added to the grid. Instead, the position of the bond price in the interval between its theoretical minimum and maximum conditional on the expected returns to capital is used. Risk aversion implies that the risk-free return on bonds cannot exceed the risky unconditional expected return on capital, whilst individual rationality implies that it cannot be below the worst-case return on capital.

Thus the additional aggregate grid variable is

$$s_t \in (0,1) \qquad s.t. \qquad p_t = s_t E_t[R_{t+1}] + (1-s_t)E_t[R_{t+1}|A_{t+1} = .99] \tag{3.34}$$

3.5.2.3 Solving the Individual Problem

Carroll (2006)'s method of endogenous gridpoints can no longer be applied without modification to this model. The solution here uses an extension of that method which has not previously been documented in the literature. That extension utilises the Euler conditions Eqs. (3.32) and (3.33), along with the fact that the aggregate productivity process only allows for two possible states in each period. The method is explained in detail in Appendix A.3.

3.5.3 Calibration

For all variables shared with the benchmark model the same calibration is maintained. The only additional exogenous value is the individual borrowing limit, $\underline{b}$, for which the value of 2.4 is taken from Krusell and Smith (1997).

3.5.4 The Distribution for Derivative Aggregation

The steady-state distribution of the model with no aggregate risk does not yield a good solution in this case. The mean error of the aggregate capital forecast is 1%. The bias is $.9\%$, indicating that the point at which the approximation is taken is not close to the distributions realised in the final simulation.

I adjusted the algorithm to simulate 24 periods during each iteration of derivative aggregation, alternating between 8 periods, the expected duration, in each state. In this way, as the solution converges, the simulations performed are using a more accurate approximation of the final individual transition rules, and so the distribution used converges to a point that lies within Ω.

3.5.5 The Solution

The solution is found in approximately 12 minutes. Whilst this is a significant increase on the benchmark model, much of the additional computation time is explained by two factors: firstly the additional simulations necessary to find the reference distribution, which consume approximately half of the entire time, and secondly the addition of the extra dimension, bond price, to the individual problem. The grid over which the solution is calculated has 9 points in that dimension, directly increasing the number of computations required by that factor. The rest of the increase is caused by the additional calculations required to solve for bond holdings and capital in the individual problem. Notwithstanding the increase in computation time, the solution is found sufficiently quickly to apply this methodology even when the model must be solved many times to estimate its parameters, for example.

In terms of accuracy, the algorithm continues to deliver extremely good results. Table 3.6 presents the statistics for the aggregate forecast. The minimum and maximum forecast errors for the aggregate capital level over a 10000 period simulation are qualitatively identical to those of the benchmark model. Bond prices are predicted with even greater precision.

An interesting outcome regarding bond prices is that the aggregate variable chosen to represent it is not forecast very well either by derivative aggregation or by a rule estimated on the simulation results. The choice of the aggregate non-state variable, which makes the bond price conditional on aggregate capital via expected returns, captures almost all of the variation of the bond price by aggregate capital level. The forecasting rule, containing only capital, can therefore not predict the aggregate variable very well. Nonetheless, the bond price, which actually affects individual choices, is forecast very accurately.

The statistics for the individual Euler equation error in Table 3.7 show that the mean error has increased by a factor of around 2 relative to the benchmark model. The maximum errors have increased dramatically, but in the case of the most important variable, consumption, this is the result of an extreme outlier. The second highest error is 1.4%, and over the 10000 period simulation, only 20 values are higher than the maximum of the benchmark model. Given the greater complexity of the model, the increased mean error is to be expected.

	10000 period ahead (%)			*1 period ahead (%)*			
	Bias	Mean	Max	Bias	Mean	Max	R^2
K	-0.061	0.074	0.279	-0.002	0.003	0.015	99.9998
p	-0.001	0.002	0.006	-0.000	0.000	0.000	99.9999

Table 3.6: Forecast Accuracy, Incomplete Markets with Bonds

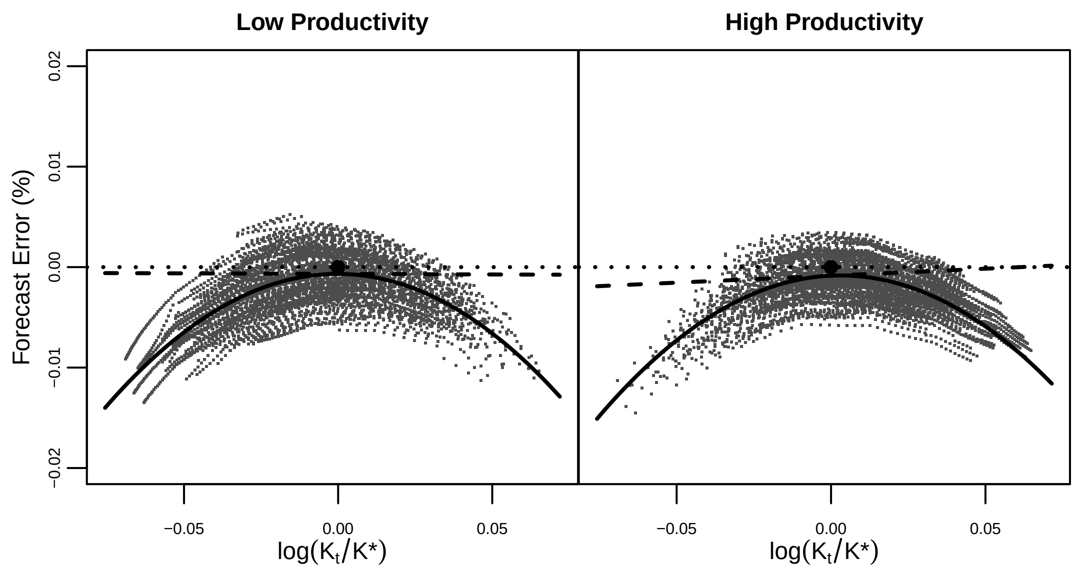

Figure 3.8: One-Period-Ahead Forecast Errors (%), Incomplete Markets with Bonds

	Bias	Mean	Max
k	-0.002	0.035	1.275
b	0.006	0.081	11.600
c	0.001	0.007	16.338

Table 3.7: Dynamic Euler Equation Errors (%), Incomplete Markets with Bonds

3.6 Discussion

3.6.1 Derivative Aggregation and Approximate Aggregation

Section 3.2 demonstrated how the forecasting rules for a model exhibiting approximate aggregation can be found by derivative aggregation, conditional on the existence of an appropriate curve in the space of distributions. The examples solved show that such curves do exist and can be identified, at least in those models.

The section also demonstrated that the mathematics underlying derivative aggregation gives some insight into when approximate, or indeed exact, aggregation do or do not hold. This is a point which would benefit from further research. The mathematics laid out here, in conjunction with an understanding of the microeconomic influences in a model, may very well provide a great deal of insight in advance as to how accurate an aggregate solution to a model is likely to be.

3.6.2 When Approximate Aggregation Does Not Hold

When approximate aggregation does not hold, at least with a small error[20], there is no function $H(.)$ such that $H(X_t)$ yields an 'almost perfect' description of X_{t+1}. By implication, there is no solution approach that will produce highly accurate forecast rules based only on aggregate variables. Since aggregates are fairly broadly defined to cover almost any integrated function over the individual states, the only remaining option that will deliver a rational-expectations solution is to include the full distribution as a state variable in the individual problem. Such problems can generally not be solved.

Young (2010) argues that forecast rules found under approximate aggregation are sufficiently accurate to consider the solutions fully, rather than boundedly, rational. Reversing the direction of this argument, it implies that economies where approximate aggregation does not hold are exactly those where a boundedly-rational individual decision rule must suffice.

As shown in earlier sections, the failure of approximate aggregation implies that there is no consistent $dx_{i,t}/dX_t$. The distribution of individual states can change in various ways, yet cause the same change in the aggregate state. Economic models do not tend to produce arbitrary changes, however, and provided the approach is mathematically feasible, it should still be possible to find a suitable set of aggregates and transformations which capture the

[20]Under the definition provided approximate aggregation will always hold for some ξ, but only small values of ξ are really of interest.

principal directions of variation. Derivative aggregation is still a reasonable approach to find as good a forecast as possible.

3.6.3 Finding Appropriate Curves

The success of derivative aggregation relies on the ability to identify suitable curves in the set of possible distributions along which the aggregation can be performed. In all the cases presented here, the curves used were fairly straightforward choices. Experiments with other, radically different choices have, however, shown that the choice of curve strongly affects the accuracy of the results. It may not always be the case that good curves are easy to identify, particularly when using aggregates more complex than the mean of individual states. Moreover, it is not clear whether any particular choice is optimal or not.

The choice of curves, or alternatively the reference distribution $\omega^\star$ and the transformations T, are a topic for further research. The curves should delineate the most representative set of distributions among those that occur in the model. Since the distributions are generated by the individual policy function f_x, the main ingredient required to understand the curves is present in each iteration. There may be better algorithmic approaches to updating the curves than that already presented.

It is also true that understanding the relationship between aggregate and individual state changes is in large part the reason for solving such models. Assuming some knowledge of those changes in advance, as is necessary to identify the curves, may seem counter-productive. However, the microeconomic foundations of the model do, at least, yield some insights, as was demonstrated in the sample models. Models without aggregate risk, or indeed with 'rule-of-thumb' aggregate forecasting rules, are much easier to solve, and experiments with both can improve understanding.

3.6.4 Other Limitations

The approach presented in this chapter does have some restrictions and they have not been discussed in detail. Many are shared by other approaches. Primarily, the individual policy must be differentiable in both individual and aggregate states. Models where this function is discontinuous or non-differentiable can not be solved using derivative aggregation.

The chapter has also not focused on limitations caused by the iterative nature of the overall algorithm. There is no guarantee that the solution will converge, and, as is the case with most numerical approaches, choosing appropriate initial conditions can be both crucial to success and difficult.

3.6.5 Implementation

One benefit of derivative aggregation is that it is a very 'mechanical' process that requires few inputs. The solutions presented in this chapter were all computed using the `DerivativeAggregationSolver` which is part of the toolkit introduced in Chapter 2. Very little additional programming is necessary: the solver only requires the derivatives of the transformations, $dT/d\theta$, and the derivatives of the aggregations, dF/dx and dF/dX, as inputs. The source code for both models and toolkit is available on-line.

3.7 Conclusion

This chapter has presented an algorithm, referred to as Derivative Aggregation, for finding an approximate forecasting rule for aggregate variables in models with heterogeneous agents and aggregate uncertainty. The approach was shown to yield an accurate solution whenever approximate aggregation holds, but this chapter has also argued that it can find a boundedly-rational forecasting rule when the model does not have a very accurate aggregate approximation.

The key insight is that since aggregate states are most commonly integrals, or functions of integrals, over individual states, derivatives of and with respect to the former can be determined from the latter. Perhaps more than other algorithms, Derivative Aggregation emphasises the nature of the relationships between individual action and aggregate outcomes. In principle it is suitable for a wide range of models. The primary requirements are that the problem can be expressed recursively, and that individual agents must not directly affect each other, as in a network model for example. In common with prior approaches, the accuracy of the solution also depends crucially on how well a parsimonious set of variables can forecast future aggregate states.

The article went on to solve both the model of Krusell and Smith (1998) and its extension in Krusell and Smith (1997). The solution to the baseline model demonstrated that the approach is quick and provides accurate solutions, and is superior in both dimensions to Explicit Aggregation (see Den Haan and Rendahl, 2010). The latter was used for as a benchmark since it was found to be one of the best in the comparison project of Den Haan (2010b).

The second model adds a one-period bond as a second asset. It demonstrated that the approach continues to perform well with additional aggregate variables.

The toolkit presented in Chapter 2 provides a class for applying derivative aggregation.

4

Matching with Heterogeneous Firms

4.1 Introduction

Firms differ in size. One measure of this difference is the number of workers a firm employs. This measure is imperfectly correlated with other measures of size such as revenue, profit or market capitalisation[1], but the distribution found under all measures share a common qualitative character: they are closely approximated by the Zipf distribution (Axtell, 2001).

[1]For example in January 2016, Apple Inc. was the largest US firm by market capitalisation ($540 Bn) and had 110,000 employees. The largest employer was Walmart with 2,200,000 employees and a market capitalisation of $220 Bn. (Source: Google Finance)

The widely used real business cycle model (see, for example, King and Rebelo, 1999) abstracts from this difference in firm sizes by assuming a representative firm. Even the model of Mortensen and Pissarides (1994), explicitly constructed to study the dynamics of employment, assumes that firms are all of identical size: 1. This model can not reproduce some of the key business cycle facts related to employment (see Shimer, 2005). But firm size matters when it comes to posting vacancies: a firm's target size depends primarily on market conditions, so other things being equal a smaller firm will post more vacancies; and non-linear vacancy posting costs will cause the number of vacancies posted to appear in the firm's optimality conditions, further affecting firm behaviour.

This chapter departs from the assumption of a representative firm by considering an economy populated by a continuum of firms which may differ in the number of workers they employ. These firms also suffer idiosyncratic demand shocks.

The model discussed is solved using the ModelSolver Toolkit presented in Chapter 2. All source code is available online (Grasl, 2014b).

4.2 Related Literature

There is a long history of papers considering economies with firm size heterogeneity. Hopenhayn (1992), for example, considers long run equilibrium in a model of an industry with heterogeneous firms subject to idiosyncratic productivity shocks. Wages and prices are exogenous to each firm and the labour market is frictionless, so firms have no endogenous state. In this environment, the author is able to prove existence and uniqueness of a static equilibrium. The model in this paper differs in that labour costs and prices are endogenous to the firm, and labour market frictions mean that firm size is an endogenous state.

There is a more recent, growing literature which consider theoretical models of frictional labour markets in a setting with firm-size heterogeneity. The general framework allows for a rich degree of variation in models: wages can be determined by bargaining or posted by firms; firms may be able to lay off workers or only experience exogenous quits; firms may be able to exit the market and new firms enter; and the cost of any size adjustment itself allows for a large number of different specifications.

Felbermayr et al. (2011) introduce search frictions into a trade model in which firms are subject to differing, but constant, productivity levels. Firm employment is an endogenous state, but the exogenous state does not change. The paper focuses on long-run equilibrium, and the time-invariant state of each firm allows for a closed form solution to the model. The paper shares with the model in this chapter the nature of competition in the product market: both adopt the specification of Dixit and Stiglitz (1977).

Acemoglu and Hawkins (2014) consider a continuous-time model populated by discretely-sized firms with heterogeneous, but constant, productivity. Both job separation and exit are exogenous. Vacancy-posting costs are quadratic in the number of vacancies posted and wages are bargained continuously. Their analysis finds that the model generates a greater degree of persistence in unemployment and market tightness than the benchmark Mortensen-Pissarides model. The model studied here uses a similar cost structure but adds time-variation in firms' exogenous state, endogenous redundancies and exit.

Elsby and Michaels (2013) study a discrete-time model with linear vacancy costs, cost-free redundancies and continuous wage bargaining. There is no firm entry or exit. They are able to derive analytical expressions for a number of variables of interest, including the wage and the adjustment of the employment distribution between periods. This tractability is at least in part due to the absence of redundancy costs and exogenous job separation. Their numerical analysis finds that the model exhibits persistence in market tightness despite the linear cost structure. They also find that they can account for empirical observations on firm-size distribution and growth rates, though the required firm productivity shocks are Pareto-distributed, and variation in idiosyncratic productivity also has a permanent component.

Fujita and Nakajima (2016) consider a similar economy, adding job-to-job transitions but abstracting from workers' bargaining power. They demonstrate that it then accounts for the cyclical properties of both job flows and worker flows.

In settings with directed search, where wages are posted by firms competing for applicants rather than determined by bargaining, both Kaas and Kircher (2015) and Schaal (2012) consider multi-worker firms. The setting provides greater tractability because the wages of existing employees need not be renegotiated, removing much of the dependency of individual decisions on the firm-size distribution.

4.3 Model

4.3.1 In Relation to Literature

The model constructed in this chapter builds on the literature outlined above by combining aspects of the different models. The most significant departure from the majority of the literature on search friction with heterogeneous firms is the use of monopolistic competition and idiosyncratic demand shocks, rather than decreasing returns to scale production technology and productivity shock. The former specification is common in the wider literature on firm heterogeneity (see, for example, Luttmer, 2007).

The model is most closely related to Elsby and Michaels (2013), adding entry and exit and making redundancies costly. The message of Carroll (2000), albeit in a very different setting, is that meaningful results about the impact of heterogeneity on a model economy require an empirically accurate distribution of agents; the message of Axtell (2001) is that most firms are of size 0. A study of firm-size heterogeneity must consider entry and exit. Cooper et al. (2007) find that redundancy costs are empirically an equally good fit for the data as hiring costs.

A further important difference is the structure of remuneration: much of the literature on wage bargaining in search models, including Elsby and Michaels (2013) and Acemoglu and Hawkins (2014), use the set-up of Stole and Zwiebel (1996) in which wages are determined by continuous negotiation over marginal gains. With endogenous redundancies such renegotiations are possible only if redundancies are costless; costly redundancies imply that the expected marginal profit of a firm laying off workers is negative, so the wage bargain leads to a negative pay-off for workers. Whilst this may be theoretically feasible in an environment with commitment, experiments conducted whilst writing this chapter suggest that the resulting model cannot be solved. The model here follows the literature in determining the remuneration of new hires through Nash bargaining over the expected marginal surplus, but changes the timing of payments: the workers' share of the expected surplus is paid upon recruitment as a sign-up bonus. Per-period wages are kept at a level which leaves the worker indifferent between working and unemployment[2]. This set-up does not directly affect firms' vacancy posting behaviour since bargaining is still over the same value[3], but does affect the firing decision through the lower wage.

4.3.2 A Brief Preview

Time is discrete. The economy is populated by a measure m of firms and a unit measure of people. Each firm employs workers as its only input and uses a common production technology. People not employed by any firm receive benefits and search for employment, whilst employed people are compensated for their labour. There is a government which taxes firm production at source.

Though the production technology used by all firms is identical, they produce differentiated goods subject to differing levels of demand. This firm-specific demand level may also change over time. In the absence of either money or a storage technology, firms sell their entire after-tax output each period, and each person spends their entire wage on an appropriate consumption bundle given their income and prices. All people, as well as the government, share identical preferences over the consumption goods. Each person aims to maximise lifetime income.

Each period, a fixed proportion of employees leave each firm for unspecified reasons. In addition, firms aim to maximise discounted expected profits and may adjust the size of their workforce in pursuit of that aim. If they wish to grow they must post costly vacancies which are matched to unemployed people searching for work. If they wish to reduce their size they

[2]The wage paid to workers by a firm making redundancies in Elsby and Michaels (2013) has the same property, which is a direct result of the assumption of cost-free redundancies.

[3]Firms are risk neutral, so the fact that uncertainty over future outcomes has been internalised by the firm does not concern it.

may make some of their employees redundant, which also incurs costs. A firm which chooses to make all its employees redundant exits the market.

New firms may enter the economy but must pay a fixed entry fee to do so. Entering firms are of size 0 and the level of demand for their goods is only established once they have entered the market.

4.3.3 Firms

There is a continuum of firms of measure m, indexed by $i \in (0, m)$, each of which produces a unique good, also simply labelled i.

4.3.3.1 Production

Firms use a single input, labour, and share a constant returns to scale production technology. In each period t, firm i uses its employees, $l_{i,t}$, to produce output subject to the production function

$$y_{i,t} = X l_{i,t} \tag{4.1}$$

where X is a constant aggregate productivity factor. There is no storage technology, so firms wish to sell their entire produce each period. The firm-specific demand curve, derived in the next section, supports this transaction at price $p_{i,t}$.

4.3.3.2 Labour Costs

Following the common approach in matching models (see Pissarides, 2000, Chapter 1) firms and newly-employed workers engage in generalised Nash bargaining to determine the new employee's remuneration. As Shimer (2005) points out, this approach does not determine the distribution of wage payments over the duration of employment, only their total expected value when the employment relationship is entered. In the current model, firms' ability to lay off workers means that the widely adopted convention of continuously renegotiated wages is problematic[4]. Instead, workers' remuneration is split into two parts: firms pay all their workers the common wage w_t, which is chosen so that employees are indifferent between working and unemployment. In addition, new hires receive a sign-on bonus $b_{i,t}$, which is paid in their first period of employment and chosen through Nash bargaining.

The reasons for this structure were discussed above: in the presence of costly redundancies, continuously negotiated wages may lead to worker utility below that of unemployment. Experiments showed that such a specification cannot be solved.

The payment structure is interesting in that it transfers all the risk from workers, traditionally considered risk averse, to the firm, traditionally considered risk neutral. Empirically, on the other hand, length of employment with a firm has a mild positive effect on a workers remuneration (Williams, 2009).

The firm is also subject to taxation on output at a time varying proportional rate τ_t. This is paid directly in the firm's goods.

4.3.3.3 Firm Size Adjustment

A firm starts each period with a predetermined size $l_{i,t}$. All of these workers participate in the production process during the period. A fixed proportion s of the workers then leave the firm for reasons beyond its control. In addition, the firm may choose to make $r_{i,t}$ workers redundant, at cost $c^r(r_{i,t}, l_{i,t})$.

A firm may also attract additional workers for the next period by posting vacancies v_t at cost $c^v(v_{i,t}, l_{i,t})$. The actual number of new workers the firm will hire as a result of these vacancies is $q(\theta_t)v_{i,t}$, where θ_t is market tightness and $q(\theta_t)$ the probability of a vacancy being matched, as specified below. $q(\theta_t)$ depends only on aggregate variables and is taken as given by the firm.

Both redundancy and vacancy posting costs are assumed to be strictly increasing, concave

[4]The problems arise from two sources: continuously renegotiated wages in the presence of possible negative surpluses imply that employees would be better off quitting rather than staying; and with redundancies the probability of losing a job is endogenous and not necessarily monotonous in firm size, leading to non-well-behaved value functions.

and 0 at 0:

$$c^k(0, .) = 0 \qquad k \in \{v, r\} \tag{4.2}$$

$$c_1^k(x, .) > 0 \qquad k \in \{v, r\}, \forall x > 0 \tag{4.3}$$

$$c_{11}^k(x, .) \geq 0 \qquad k \in \{v, r\}, \forall x \geq 0 \tag{4.4}$$

Taking exogenous leavers and the firm's choices into account, it's workforce evolves according to the law of motion

$$l_{i,t+1} = (1 - s)l_{i,t} - r_{i,t} + q(\theta_t)v_{i,t} \tag{4.5}$$

4.3.3.4 Profit

Each firm maximises expected lifetime profits, where future profits are discounted at rate δ. Its value function in period t can therefore be expressed as

$$\begin{aligned} F(l_{i,t}, x_{i,t}; A_t) = \max_{l_{i,t+1}, v_{i,t}, r_{i,t}} & (p_{i,t}X(1 - \tau_t) - w_t)l_{i,t} - c_r(r_{i,t}, l_{i,t}) - c_v(v_{i,t}, l_{i,t}) \\ & - \delta v_{i,t}q(\theta_t)b_{i,t+1} + \delta E\left[F(l_{i,t+1}, x_{i,t+1}; A_{t+1})\right] \qquad (4.6) \\ \text{s.t.} \quad & l_{i,t+1} = (1 - s)l_{i,t} - r_{i,t} + q(\theta_t)v_{i,t} \qquad (4.7) \end{aligned}$$

where A_t is a vector of aggregate variables which affect current or future market conditions and are hence taken into account by the firm.

Note that, though the bonus $b_{i,t+1}$ for new employees is paid in the period in which they start working, it is determined in the period when the match is made and is thus known one period in advance. Structuring the bonus payments as above in the value function avoids the need for an extra state variable.

4.3.3.5 Entry and Exit

Entry

New firms are free to enter the market, but this entry incurs start-up costs C_s. The new firm enters the market with no employees and with an unknown demand level. The level of demand for the firm's goods is realised once the entry decision has been made and is drawn according to the steady-state density function of the Markov process for that variable, conditional on the aggregate state. The free entry condition requires that the cost C_s is equal to the expected value of the entering firm.

Exit

Firms can also exit the market, and will do so if it is in their interest. A firm must pay the redundancy cost for its entire workforce if it chooses to exit. This cost must be balanced against the value of remaining in the market, and means that some firms will remain in the market despite their value being negative.

4.3.4 People

There is a continuum of people of measure 1. Each person aims to maximise lifetime income and spends all such income each period on an optimal consumption bundle given prices and preferences, which are shared by all people. These preferences are of the Dixit-Stiglitz (Dixit and Stiglitz, 1977) form and hence admit to aggregation (see Acemoglu, 2008, Chapter 5). Consumption can thus be considered in aggregate.

4.3.4.1 Consumption

Consumption behaviour is similar to that described in Benhabib et al. (2015): demand for particular goods varies exogenously both across firms and across time. The representative consumer's problem is

$$\max_{\{c_{i,t}\}_{i=0}^{1}} \left(\int_0^1 \epsilon_{i,t} (c_{i,t} + \phi)^{\frac{\gamma-1}{\gamma}} \, \mathrm{d}i \right)^{\frac{\gamma}{\gamma-1}} \quad s.t. \qquad I_t = \int_0^1 c_{i,t} p_{i,t} \, \mathrm{d}i \tag{4.8}$$

where the variables are

$\epsilon_{i,t}$	Demand level for good i at time t
$c_{i,t}$	Consumption of good i at time t
$p_{i,t}$	Price of good i at time t
I_t	Total income of the economy at time t

$\epsilon_{i,t}$ is exogenous.

The solution to this problem, derived in Section B.1, is

$$p_{i,t} = \Phi_t \epsilon_{i,t} \left(\frac{C_t}{c_{i,t} + \phi} \right)^{\frac{1}{\gamma}} \tag{4.9}$$

where

$$C_t = \left(\int_0^1 \epsilon_{i,t} (c_{i,t} + \phi)^{\frac{\gamma-1}{\gamma}} \, \mathrm{d}i \right)^{\frac{\gamma}{\gamma-1}} \tag{4.10}$$

$$P_t = \int_0^1 p_{i,t} \, \mathrm{d}i \tag{4.11}$$

$$\Phi_t = \frac{I_t + \phi P_t}{C_t} \tag{4.12}$$

There is no money in this economy. All transactions are transfers of units of production, the value of which is proportional to Φ_t by Eq. (4.9). This also applies to I_t and P_t, so that Φ_t is simply a scaling factor (see 4.12). Without loss of generality assume $\Phi_t = 1 \ \forall t$.

Note that the parameter ϕ is necessary to allow for firm exit: if $\phi = 0$ then the price for any good, given in equation Eq. (4.9), tends to positive infinity as its supply drops toward 0. Thus no firm would ever exit the market, however low its demand shock.

Each person aims to maximise her lifetime income subject to the discount rate δ. She can either be employed or unemployed in any period, but this status is exogenous.

4.3.4.2 Unemployment

An unemployed person is in the labour market and receives unemployment benefit z per period. She is matched to a posted vacancy with probability $f(\theta_t)$, which depends only on aggregates and is taken as given by the individual. She is equally likely to be matched to any posted vacancy, so that the expected value of finding a job this period is

$$\mathfrak{W}_t = \int_0^1 \left(\frac{v_{j,t}}{V_t} E[N_{j,t+1}] \right) \mathrm{d}j \tag{4.13}$$

where $v_{j,t}$ are the vacancies posted by firm j, $V_t \equiv \int_0^m v_{j,t}\, \mathrm{d}j$ is the total number of vacancies posted, and $N_{j,t+1}$ is the value of finding a job with firm j.

The value of being unemployed in period t, $\mathfrak{U}_t$, is therefore

$$\mathfrak{U}_t = z + \delta E \left[f(\theta_t)\mathfrak{W}_t + (1 - f(\theta_t))\mathfrak{U}_{t+1} \right] \tag{4.14}$$

4.3.4.3 Finding a Job

An unemployed person who is matched with a vacancy at firm i in period t will be paid the sign-up bonus $b_{i,t+1}$ in period $t+1$. In addition, she will receive the same remuneration as any other employee of the firm from that period onward. The expected value of this remuneration, defined in the next section, is $E[W_{i,t+1}]$. Thus the value of finding that job, $N_{i,t}$, is

$$N_{i,t+1} = b_{i,t+1} + E[W_{i,t+1}] \tag{4.15}$$

4.3.4.4 Employment

An employee of firm i in period t earns wage w_t and will continue to work at that firm in the next period with probability $(1 - s - \frac{r_{i,t}}{l_{i,t}})$. With probability $(s + \frac{r_{i,t}}{l_{i,t}})$, the employee and the firm will separate at the outset of period $t + 1$. The former employee will join the pool of job-seekers in the labour market[5]. The wage is chosen so that the employee is indifferent between employment and unemployment, thus having no incentive to quit.

The value of being employed by firm i in period t, $W_{i,t}$, can therefore be expressed as[6]

$$W_{i,t} = w_t + \delta E\left[\left(1 - s - \frac{r_{i,t}}{l_{i,t}}\right) W_{i,t+1} + \left(s + \frac{r_{i,t}}{l_{i,t}}\right) \mathfrak{U}_{t+1}\right] \tag{4.16}$$

The presence of $r_{i,t}$ in this equation[7] is the reason that the model is not well-behaved in the presence of costly redundancies and continuous wage renegotiation. When a firm makes costly redundancies it chooses a size at which its future marginal value is negative, equal in size to the marginal cost of redundancies. Therefore $E[W_{i,t+1} - \mathfrak{U}_{i,t+1}]$ would also be negative, leaving a positive product of $r_{i,t}$ in the equation above. The surplus of being employed at a firm laying workers off is increasing in the proportion of lay-offs.

Elsby and Michaels (2013) avoid this conundrum by making redundancies costless, so that $E[W_{i,t+1} - \mathfrak{U}_{t+1}] = 0$ when redundancies are made, and $r_{i,t}$ drops out of this equation. By changing the timing of payments, the specification in this chapter achieves the same effect.

Since the wage is chosen so that the employee is indifferent between staying in employment and unemployment, $W_{i,t} = \mathfrak{U}_t$. Therefore $\mathfrak{U}_{t+1} = W_{i,t+1}$, so the above reduces to

$$\begin{aligned}
& & W_{i,t} &= w_t + \delta E[\mathfrak{U}_{i,t+1}] && (4.17)\\
&\Rightarrow & \mathfrak{U}_t &= w_t + \delta E[\mathfrak{U}_{i,t+1}] && (4.18)\\
\text{Eq. (4.14)} &\Rightarrow & z + \delta E\left[f(\theta_t)\mathfrak{W}_t + (1 - f(\theta_t))\mathfrak{U}_{t+1}\right] &= w_t + \delta E[\mathfrak{U}_{i,t+1}] && (4.19)\\
&\Rightarrow & w_t &= z + \delta f(\theta_t) E\left[\mathfrak{W}_t - \mathfrak{U}_{t+1}\right] && (4.20)
\end{aligned}$$

The expected value of a new job in general, $\mathfrak{W}_t$, can now be further simplified:

[5] In this set-up a worker separating from a firm spends at least one period unemployed. This may be acceptable in the Hagedorn and Manovskii (2008) specification, which uses a weekly period, but would be less good in a Shimer (2005) calibration, which is using quarterly periods.

[6] Individual values are of course also conditional on the aggregate state, but for ease of exposition this is not shown here

[7] To be precise, the equivalent equation, in which the wage is also firm-specific.

$$\mathfrak{W}_t = \int_0^m \left(\frac{v_{j,t}}{V_t} E[N_{j,t+1}] \right) \mathrm{d}j \tag{4.21}$$

$$\text{Eq. (4.15)} \Rightarrow \qquad \mathfrak{W}_t = \int_0^m \frac{v_{j,t}}{V_t} (b_{j,t+1} + E[W_{j,t+1}]) \mathrm{d}j \tag{4.22}$$

$$W_{j,t+1} = \mathfrak{U}_{t+1} \Rightarrow \qquad \mathfrak{W}_t = \frac{1}{V_t} \int_0^m v_{j,t} b_{j,t+1} \mathrm{d}j + \frac{E[\mathfrak{U}_{t+1}])}{V_t} \int_0^m v_{j,t} \mathrm{d}j \tag{4.23}$$

$$V_t \equiv \int_0^m v_{j,t} \mathrm{d}j \Rightarrow \qquad \mathfrak{W}_t = \frac{1}{V_t} \int_0^m v_{j,t} b_{j,t+1} \mathrm{d}j + E[\mathfrak{U}_{t+1}] \tag{4.24}$$

Substituting this back into the wage equation Eq. (4.20):

$$w_t = z + \delta \frac{f(\theta_t)}{V_t} \int_0^1 v_{j,t} b_{j,t+1} \mathrm{d}j \tag{4.25}$$

4.3.5 The Employment Relationship

4.3.5.1 Aggregate Labour Market Variables

So far the discussion has concerned individual actors in the economy: firms and people. Aggregating the individual variables discussed gives rise to the aggregate labour market variables:

$$\text{Aggregate Employment } L_t: \qquad L_t = \int_0^m l_{i,t} \mathrm{d}i \tag{4.26}$$

$$\text{Unemployment } U_t: \qquad U_t = 1 - L_t \tag{4.27}$$

$$\text{Aggregate Vacancies } V_t: \qquad V_t = \int_0^m v_{i,t} \mathrm{d}i \tag{4.28}$$

An additional derived variable is useful as a labour market indicator:

$$\text{Market Tightness } \theta_t: \qquad \theta_t = \frac{V_t}{U_t} \tag{4.29}$$

4.3.5.2 Matching

Unemployed people are matched to vacancies using the matching function introduced by den Haan et al. (2000). The number of matches M_t is given by

$$M_t \equiv M(U_t, V_t) = \frac{U_t V_t}{(U_t^{\iota} + V_t^{\iota})^{\frac{1}{\iota}}} \tag{4.30}$$

$$\Rightarrow \qquad M_t = \frac{U_t V_t}{U_t \left(1 + \left(\frac{V_t}{U_t}\right)^{\iota}\right)^{\frac{1}{\iota}}} \tag{4.31}$$

$$\Rightarrow \qquad M_t = \frac{V_t}{(1 + \theta_t^{\iota})^{\frac{1}{\iota}}} \tag{4.32}$$

This gives rise to the probabilities of a vacancy being matched and of finding a job:

$$\text{Probability of a vacancy being matched:} \qquad q(\theta_t) \equiv \frac{M_t}{V_t} = (1 + \theta_t^{\iota})^{-\frac{1}{\iota}} \tag{4.33}$$

$$\text{Probability of finding a job:} \qquad f(\theta_t) \equiv \frac{M_t}{U_t} = \theta_t (1 + \theta_t^{\iota})^{-\frac{1}{\iota}} \tag{4.34}$$

4.3.5.3 Timing of Events

The sequence of events each period is:

1. Firm and, if applicable, aggregate shocks are realized
2. Firms post vacancies
3. Firms engage in production
4. The government collects taxes
5. Firms pay workers, government pays unemployment benefits and goods are exchanged to form consumption bundles
6. Workers are matched to vacancies and redundancies are announced
7. Bonuses for the next period are negotiated

The final step is important: bonuses are negotiated one period in advance and before next-period shocks are known.

4.3.5.4 Bargaining

The sign-up bonuses are determined through generalized Nash bargaining over the joint surplus between each individual new employee and her employer. The employee has bargaining power β.

Firm Surplus

The surplus $S^f_{i,t+1}$ which accrues to the firm from the marginal new employee has three components: the expected benefit to the firm of that employee in the next period, which is the firm's expected marginal value; the sign-up bonus that must be paid to the employee; and the change in the sign-up bonus due to the $q(\theta_t)v_{i,t}$ other new employees as a result of the additional one:

$$S^f_{i,t} = \delta(E\left[F_1(l_{t+1},.)\right] - (b_{i,t+1} + \frac{\partial b_{i,t+1}}{\partial l_{i,t+1}} q(\theta_t)v_{i,t})) \tag{4.35}$$

New Employees' Surplus

The new employee's surplus, $S^w_{i,t+1}$, is the value of having found the job over unemployment:

$$S^w_{i,t} = \delta(N_{i,t} - E[U_{t+1}]) \tag{4.36}$$

$$\text{Eq. (4.15)} \Rightarrow \qquad S^w_{i,t} = \delta(b_{i,t+1} + \delta E[W_{i,t+1}] - \delta E[U_{t+1}]) \tag{4.37}$$

$$W_{i,t+1} = U_{t+1} \Rightarrow \qquad S^w_{i,t} = \delta b_{i,t+1} \tag{4.38}$$

Surplus Splitting

The standard outcome of the Nash bargain means that the surpluses will satisfy

$$\beta S^f_{i,t} = (1-\beta)S^w_{i,t} \tag{4.39}$$

$$4.38,4.35 \Rightarrow \quad \beta\left(E[F_1(l_{t+1},.)] - (b_{i,t} + \frac{\partial b_{i,t+1}}{\partial l_{i,t+1}} q(\theta_t)v_{i,t})\right) = (1-\beta)b_{i,t+1} \tag{4.40}$$

$$\Rightarrow \quad b_{i,t+1} + \beta \frac{\partial b_{i,t+1}}{\partial l_{i,t+1}} q(\theta_t)v_{i,t} = \beta E[F_1(l_{t+1},.)] \tag{4.41}$$

4.4 The Solution

4.4.1 The Firms' Problem

The firms' problem is

$$F(l_{i,t}, x_{i,t}, w_{i,t}; A_t) = \max_{l_{i,t+1}, v_{i,t}, r_{i,t}} (p_{i,t}X(1-\tau_t) - w_t)l_{i,t} - c^v(v_{i,t}, l_{i,t}) - c^r(r_{i,t}, l_{i,t}) -$$
$$- \delta v_{i,t} q(\theta_t) b_{i,t+1} + \delta E\left[F(l_{i,t+1}, x_{i,t+1}; A_{t+1})\right] \tag{4.42}$$
$$\text{s.t.} \quad l_{i,t+1} = (1-s)l_{i,t} + q(\theta_t)v_{i,t} - r_{i,t} \tag{4.43}$$
$$v_{i,t} \geq 0, r_{i,t} \geq 0, l_{i,t+1} \geq 0 \tag{4.44}$$

where the price $p_{i,t}$ is endogenous and may vary with $l_{i,t}$.

The Lagrangian for this problem is

$$\mathcal{L} = (p_{i,t}X(1-\tau_t) - w_t)l_{i,t} - c^v(v_{i,t}, l_{i,t}) - c^r(r_{i,t}, l_{i,t}) - \delta v_{i,t} q(\theta_t) b_{i,t+1}$$
$$+ \delta E\left[F(l_{i,t+1}, p_{i,t+1}; A_{t+1})\right] + \lambda_{i,t}((1-s)l_{i,t} + q(\theta_t)v_{i,t} - r_{i,t} - l_{i,t+1}) \tag{4.45}$$

The first order conditions with respect to $v_{i,t}$ are:

$$\begin{aligned} v_{i,t} &\geq 0, \\ -c_1^v(v_{i,t}, l_{i,t}) - \delta q(\theta_t)\left(b_{i,t+1} + v_{i,t}\frac{\partial b_{i,t}}{\partial v_{i,t}}\right) &+ \lambda_{i,t} q(\theta_t) \leq 0, \\ v_{i,t}(-c_1^v(v_{i,t}, l_{i,t}) - \delta q(\theta_t)\left(b_{i,t+1} + v_{i,t}\frac{\partial b_{i,t}}{\partial v_{i,t}}\right) &+ \lambda_{i,t} q(\theta_t)) = 0 \\ \Rightarrow v_{i,t} &= 0 \end{aligned}$$

or

$$\lambda_{i,t} - \delta\left(b_{i,t+1} + v_{i,t}\frac{\partial b_{i,t+1}}{\partial v_{i,t}}\right) = \frac{c_1^v(v_{i,t}, l_{i,t})}{q(\theta_t)} \tag{4.46}$$

The first order conditions with respect to $r_{i,t}$ are:

$$r_{i,t} \geq 0, \qquad -c_1^r(r_{i,t}, l_{i,t}) - \lambda_{i,t} \leq 0, \qquad r_{i,t}(-c_1^r(r_{i,t}, l_{i,t}) - \lambda_{i,t} q(\theta_t)) = 0$$

$$\Rightarrow r_{i,t} = 0$$

or

$$\lambda_{i,t} = -c_1^r(r_{i,t}, l_{i,t}) \tag{4.47}$$

The first order conditions with respect to $l_{i,t+1}$ are:

$$l_{i,t+1} \geq 0, \qquad \delta E[F_1(l_{i,t+1}, .)] - \lambda_{i,t} \leq 0, \qquad l_{i,t+1}(\delta E[F_1(l_{i,t+1}, .)] - \lambda_{i,t}) = 0$$

$$\Rightarrow l_{i,t+1} = 0$$

or

$$\lambda_{i,t} = \delta E[F_1(l_{i,t+1}, .)] \tag{4.48}$$

The envelope theorem for constrained maximisation implies that

$$F_1(l_{i,t}, p_{i,t}; A_t) = \frac{\partial \mathcal{L}}{\partial l_{i,t}} = X(1-\tau_t)\left(\frac{\partial p_{i,t}}{\partial l_{i,t}} l_{i,t} + p_{i,t}\right) - w_t$$
$$-c_2^v(v_{i,t}, l_{i,t}) - c_2^r(r_{i,t}, l_{i,t}) + (1-s)\lambda_{i,t} \quad (4.49)$$

Iterating this forward one period yields

$$F_1(l_{i,t+1}, p_{i,t+1}; A_t) = X(1-\tau_t)\left(\frac{\partial p_{i,t+1}}{\partial l_{i,t+1}} l_{i,t+1} + p_{i,t+1}\right) - w_t$$
$$-c_2^v(v_{i,t+1}, l_{i,t+1}) - c_2^r(r_{i,t+1}, l_{i,t+1}) + (1-s)\lambda_{i,t+1}$$
$$(4.50)$$

4.4.1.1 Some immediate consequences

Assume that $c_1^v(v,.) \geq 0, c_1^r(r,.) > 0$ and that $b_{i,t+1} + v_{i,t}\frac{\partial b_{i,t}}{\partial v_{i,t}} > 0$.

Then

From Eq. (4.46): $$v_{i,t} > 0 \Rightarrow \lambda_{i,t} > 0 \quad (4.51)$$

From Eq. (4.47): $$r_{i,t} > 0 \Rightarrow \lambda_{i,t} < 0 \quad (4.52)$$

Hence

$$v_{i,t} r_{i,t} = 0 \quad (4.53)$$

A firm does not both make employees redundant and post vacancies.

4.4.1.2 The Exit Condition

As stated in Section 4.3, a firm that chooses to exit the market must pay the redundancy cost for its entire workforce. A consequence of this requirement is that the value calculation and hence the marginal conditions of the firm do not change, further implying that first-order conditions must still be satisfied. In other words, exit will occur if the optimal choice of future firm size is 0.

Consider the first-order conditions for $l_{i,t+1}$ and $r_{i,t}$ for the exiting firm:

$$(4.48) \Rightarrow \qquad \lambda_{i,t} \geq \delta E[F_1(0,.)] \tag{4.54}$$

$$(4.47) \Rightarrow \qquad \lambda_{i,t} = -c_1^r((1-s)l_{i,t}, l_{i,t}) \tag{4.55}$$

$$\Rightarrow \qquad c_1^r((1-s)l_{i,t}, l_{i,t}) \leq -E[F_1(0,.)] \tag{4.56}$$

In conjunction they imply that a firm will exit if the marginal cost of doing so is less than the negative expected marginal value of a size 0 firm, conditional on current firm-exogenous variables. Convexity of $c^r(.,.)$ further implies that if the conditions above hold for a particular $l_{i,t}^*$, then they will hold for any $l_{i,t} < l_{i,t}^*$, again conditional on exogenous states.

In summary, for any given level of firm demand, all firms smaller than some threshold value will exit, and all larger firms will not. If the threshold is 0, none of the firms at that demand level exit. This matches the findings of Samaniego (2006).

4.4.2 The Sign-Up Bonus

Surplus splitting yielded equation Eq. (4.41) for the sign-up bonus:

$$b_{i,t+1} + \beta v_{i,t} q(\theta_t) \frac{\partial b_{i,t+1}}{\partial l_{i,t+1}} = \beta E[F_1(l_{t+1}, .)] \tag{4.57}$$

Substituting Eq. (4.48) into Eq. (4.46) yields:

$$\delta E[F_1(l_{i,t+1}, .)] - \delta \left(b_{i,t+1} + v_{i,t} \frac{\partial b_{i,t+1}}{\partial v_{i,t}} \right) = \frac{c_1^v(v_{i,t}, l_{i,t})}{q(\theta_t)} \tag{4.58}$$

$$\Rightarrow \qquad b_{i,t+1} + v_{i,t} \frac{\partial b_{i,t+1}}{\partial v_{i,t}} = E[F_1(l_{i,t+1}, .)] - \frac{c_1^v(v_{i,t}, l_{i,t})}{\delta q(\theta_t)} \tag{4.59}$$

But $\frac{dl_{i,t+1}}{dv_{i,t}} = q(\theta_t)$, so this becomes

$$b_{i,t+1} + v_{i,t}q(\theta_t)\frac{\partial b_{i,t+1}}{\partial l_{i,t+1}} = E[F_1(l_{i,t+1}, .)] - \frac{c_1^v(v_{i,t}, l_{i,t})}{\delta q(\theta_t)} \tag{4.60}$$

Combining this with the Eq. (4.57) yields

$$(1-\beta)b_{i,t+1} = \beta\frac{c_1^v(v_{i,t}, l_{i,t})}{\delta q(\theta_t)} \tag{4.61}$$

$$\Rightarrow \qquad b_{i,t+1} = \frac{\beta}{(1-\beta)\delta q(\theta_t)}c_1^v(v_{i,t}, l_{i,t}) \tag{4.62}$$

4.4.3 Size Adjustment Costs

4.4.3.1 A Brief Thought Experiment

Assume that both $c^v(v, l)$ and $c^r(r, l)$ are linear in their first arguments, and do not change with the second. The first observation is that the sign-up bonus, Eq. (4.62), is independent of the firm's decision. It takes a common value across all firms.

Considering equations Eqs. (4.46) and (4.47) further reveals that there is then at most one value of $\lambda_{i,t}$ which can satisfy each constraint for any given combination of firm-exogenous states. $\lambda_{i,t}$ indicates the expected marginal firm value and depends only on $l_{i,t+1}$ and the firm-exogenous state. For any given level of exogenous states, firms below a certain size threshold would all post vacancies to attain that size, and firms above a second, higher, threshold would choose redundancies to attain that. Firms in-between can satisfy neither constraint and will not adjust their size endogenously, shrinking only by the exogenous mechanism.

Considering only vacancies, Acemoglu and Hawkins (2014) called the resulting firm size dynamics 'bang bang': as soon as a firm's target size changes as a result of an exogenous shock the firm adjusts its size to the optimal one. Since almost all firms adjust instantly, aggregate employment also adjusts very rapidly. As Fujita and Ramey (2007) discuss, this behaviour does not match empirical observations. Kaas and Kircher (2015) also discuss non-linear recruitment costs, and how they might arise for example through time spend recruiting which is lost to production.

4.4.3.2 Quadratic Costs

Acemoglu and Hawkins (2014) instead choose a quadratic form for the vacancy posting cost. This chapter extends the assumption to redundancy costs.

Assumption 1 (Quadratic Costs)**.** *Both vacancy posting and redundancy costs are quadratic:*

$$c^v(v,l) = \frac{1}{2}\nu_v v^2 \tag{4.63}$$
$$c^r(r,l) = \frac{1}{2}\nu_r r^2 \tag{4.64}$$

Under this assumption,

$$c_1^v(v,l) = \nu_v v \tag{4.65}$$
$$c_1^r(r,l) = \nu_r r \tag{4.66}$$

Substituting the former into Eq. (4.62), the sign-up bonus a firm must pay its newly hired employees becomes:

$$b_{i,t+1} = \frac{\beta\nu_v}{(1-\beta)\delta q(\theta_t)} v_{i,t} \tag{4.67}$$

This in turn can be substituted into the wage equation Eq. (4.25) to give

$$w_t = z + \delta\frac{\beta\nu_v}{(1-\beta)\delta q(\theta_t)}\frac{f(\theta_t)}{V_t}\int_0^1 v_{j,t}^2 \,\mathrm{d}j \tag{4.68}$$
$$\Rightarrow \qquad w_t = z + \frac{\beta\nu_v}{1-\beta}\frac{\mathfrak{V}_t^2}{U_t} \tag{4.69}$$

where

$$\mathfrak{V}_t^2 = \int_0^1 v_{j,t}^2 \, \mathrm{d}j \tag{4.70}$$

The quadratic costs and the sign-up bonus Eq. (4.67) can also be substituted into the first order condition for $v_{i,t}$, Eq. (4.46):

$$\lambda_{i,t} - \delta \left(b_{i,t+1} + v_{i,t} \frac{\partial b_{i,t+1}}{\partial v_{i,t}} \right) = \frac{c_1^v(v_{i,t}, l_{i,t})}{q(\theta_t)} \tag{4.71}$$

$$\Rightarrow \quad \lambda_{i,t} - \delta \left(\frac{\beta \nu_v}{(1-\beta)\delta q(\theta_t)} v_{i,t} + v_{i,t} \frac{\beta \nu_v}{(1-\beta)\delta q(\theta_t)} \right) = \frac{\nu_v v_{i,t}}{q(\theta_t)} \tag{4.72}$$

$$\Rightarrow \quad \lambda_{i,t} = v_{i,t} \frac{\nu_v}{q(\theta_t)} \left(1 + \frac{2\beta}{1-\beta} \right) \tag{4.73}$$

$$\Rightarrow \quad v_{i,t} = \frac{1-\beta}{\nu_v(1+\beta)} q(\theta_t) \lambda_{i,t} \tag{4.74}$$

Similarly for, $r_{i,t}$:

$$r_{i,t} = -\frac{\lambda_{i,t}}{\nu_r} \tag{4.75}$$

The non-linear costs mean that one of the first order conditions always binds. This facilitates application of the reverse-time recursive solution approach: the expectation of a firm's future choice of vacancies or redundancies determines the expectation of the marginal value it expects two periods ahead, so that expectations of marginal values do not need to be calculated independently.

4.5 Computing the Solution

The solution is calculated using the framework introduced in Chapter 2. This section outlines the variables used and calculations performed.

4.5.1 The Variable Grid

The full variable grid has the following variables, further described in Table 4.1:

$(l_{i,t}, \epsilon_{i,t}, L_t, Y_t, V_t, \mathfrak{V}_t^2, \tau_t)$

Variable	Aggregation	Description
Individual Endogenous States		
$l_{i,t}$		Firm size
Individual Exogenous States		
$\epsilon_{m,t}$		Firm-specific demand
Aggregate Endogenous States		
L_t	$\int_0^m l_{i,t} \, \mathrm{d}i$	Employment
Y_t	$\left(\int_0^m \epsilon_{i,t}(Xl_{i,t} + \phi)^{\frac{\gamma-1}{\gamma}} \, \mathrm{d}i\right)^{\frac{\gamma}{\gamma-1}}$	Output, which equals consumption
Aggregate Controls		
V_t	$\int_0^m v_{i,t} \, \mathrm{d}i$	Total vacancies
$\mathfrak{V}_t^2$	$\int_0^m v_{i,t}^2 \, \mathrm{d}i$	Second moment of vacancies
Aggregate Exogenous States		
τ_t		Rate of tax on output

Table 4.1: Grid Variables

Additional variables of interest which do not constitute a grid dimension are the firm's control variables, firm vacancies $v_{i,t}$ and firm redundancies $r_{i,t}$, and the measure of entering firms, E_t.

4.5.2 The Firms' Problem

The individual agents of interest are firms. The firms' problem is amenable to solution by the Method of Endogenous Gridpoints (Carroll, 2006), so the solution extends `EGMIndividualProblemSolver`.

4.5.2.1 Computing the Expectations

The first step of each iteration is to compute the expectation of the firms discounted marginal value, conditional on given realisations of the exogenous states and aggregate variables, and an assumption of firms' policy functions.

$$F_1(l_{i,t+1}, \epsilon_{i,t+1}; A_t) = X(1-\tau_t)\left(\frac{\partial p_{i,t+1}}{\partial l_{i,t+1}} l_{i,t+1} + p_{i,t+1}\right) - w_t + (1-s)\lambda_{i,t+1} \tag{4.76}$$

Since a firm's entire output, $Xl_{i,t+1}$, is consumed each period the prices must be calculated from the demand curve at that point:

$$\text{Eq. (4.9)} \Rightarrow \qquad p_{i,t} = \epsilon_{i,t}\left(\frac{C_t}{c_{i,t}+\phi}\right)^{\frac{1}{\gamma}} \tag{4.77}$$

$$c_{i,t} = Xl_{i,t} \Rightarrow \qquad p_{i,t} = \epsilon_{i,t}\left(\frac{C_t}{Xl_{i,t}+\phi}\right)^{\frac{1}{\gamma}} \tag{4.78}$$

$$\Rightarrow \qquad p_{i,t+1} = \epsilon_{i,t+1}\left(\frac{C_{t+1}}{Xl_{i,t+1}+\phi}\right)^{\frac{1}{\gamma}} \tag{4.79}$$

$$\Rightarrow \qquad \frac{\partial p_{i,t+1}}{\partial l_{i,t+1}} l_{i,t+1} + p_{i,t+1} = \epsilon_{i,t+1} C_{t+1}^{\frac{1}{\gamma}} \left(Xl_{i,t+1}+\phi\right)^{-\frac{1}{\gamma}-1} \left((Xl_{i,t+1}+\phi) - \frac{1}{\gamma} Xl_{i,t+1}\right) \tag{4.80}$$

$$\Rightarrow \qquad \frac{\partial p_{i,t+1}}{\partial l_{i,t+1}} l_{i,t+1} + p_{i,t+1} = \epsilon_{i,t+1} C_{t+1}^{\frac{1}{\gamma}} \left(Xl_{i,t+1}+\phi\right)^{-\frac{1}{\gamma}-1} \left(\left(1-\frac{1}{\gamma}\right) Xl_{i,t+1} + \phi\right) \tag{4.81}$$

Let $\mathfrak{l}(l_{t+1}) \equiv (Xl_{i,t+1} + \phi \left(\left(1-\frac{1}{\gamma}\right) Xl_{i,t+1} + \phi\right)$. This function depends only on the individual expected states, which are on-grid and so do not change from iteration to iteration. Then

$$\frac{\partial p_{i,t+1}}{\partial l_{i,t+1}} l_{i,t+1} + p_{i,t+1} = \epsilon_{i,t+1} C_{t+1}^{\frac{1}{\gamma}} \mathfrak{l}(l_{t+1}) \tag{4.82}$$

Also note that an assumed version of the firms' policy function exists, so that there is a function $l_{i,t+2}(l_{i,t+1}, .)$ where the other parameters are exogenous to the firm. Given this, the implied endogenous change undertaken by the firm can be calculated:

$$\Delta(l_{i,t+1},.) = l_{i,t+2}(l_{i,t+1},.) - (1-s)l_{i,t+1} \tag{4.83}$$

Using this, the first order conditions Eqs. (4.74) and (4.75) can be used to determine $\lambda_{i,t+1}$:

$$\lambda(l_{i,t+1},.) = \begin{cases} \frac{1+\beta}{1-\beta}\frac{\nu_v}{q(\theta_t)^2}\Delta(l_{i,t+1},.) & \text{if } \Delta(l_{i,t+1},.) > 0 \\ -\nu_r\Delta(l_{i,t+1}) & \text{otherwise} \end{cases} \tag{4.84}$$

From these functions, $F_1(l_{i,t+1},.)$ can easily be calculated:

$$F_1(l_{i,t+1},\epsilon_{i,t+1};A_t) = X(1-\tau_t)\epsilon_{i,t+1}C_{t+1}^{\frac{1}{\gamma}}\mathfrak{l}(l_{t+1}) - w_t + (1-s)\lambda(l_{i,t+1},.) \tag{4.85}$$

4.5.2.2 Computing Implied $l_{i,t}$

From the expectations above, which are conditional on future outcomes, the framework automatically calculates the expectations conditional only on current exogenous individual and aggregate states. The next step is to determine the current state, $l_{i,t}$, implied by those expectations, which were calculated at $l_{i,t+1}$.

The process is straightforward: as shown above, a positive expected marginal value implies that a firm will post vacancies but no redundancies, and a negative one the inverse. Given the expected discounted marginal value $\lambda_{i,t}(l_{i,t+1},.)$, the chosen size adjustment and hence the implied current state $l_{i,t}$ follows from the first order conditions:

$$l_{i,t} = \frac{1}{1-s}\left(l_{i,t+1} + \lambda_{i,t}(l_{i,t+1},.) \times \begin{cases} -\frac{1-\beta}{1+\beta}\frac{q(\theta_t)^2}{\nu_v} & \text{if } \lambda_{i,t}(l_{i,t+1},.) > 0 \\ \frac{1}{\nu_r} & \text{otherwise} \end{cases}\right) \tag{4.86}$$

The implied $l_{i,t}$ may be less than 0 if the expected marginal value of a firm is sufficiently positive. As when the individual savings problem is solved using this method, the negative values do not represent a problem: it simply means that, all other variables being equal, a firm of size 0 would choose an $l_{i,t+1}$ greater than the value a negative $l_{i,t}$ was obtained from. The subsequent interpolation, performed by the framework, will correctly handle this case.

4.5.3 The Steady State With No Aggregate Uncertainty

This economy presents the conundrum common to economies with individual agents, aggregate variables and model-consistent expectations, already encountered in Chapters 2 and 3: the firms choices depend on expectations of future aggregate variables, which in turn depend on firms choices.

The method used to calculate such a steady state in earlier chapters can not be used here: the models in those chapters had no non-state aggregate variables[8].

In the absence of aggregate uncertainty, because the number of firms is sufficient for the law of large numbers to apply, a reasonable expectation is that the economy should converge to a steady state in which the firm-size distribution, and consequently all aggregate variables, are constant. This section discusses how that constant state is determined.

4.5.3.1 Assumption of Current Aggregate Variables

A naive approach to finding such a steady state might proceed as follows:

1. Guess values for aggregate variables
2. Solve the individual problem assuming those values for all periods
3. Simulate the resulting policies until the distribution converges
4. Compare the resulting aggregates to the guess

[8]For the model with bonds this is only true in the absence of aggregate uncertainty since bonds and equity are equivalent

5. Update the guess and repeat until they match

This approach does not work. Since constant aggregate values are assumed when calculating the individual policies, constant policies are used during simulation: firms' actions do not react to realised market conditions. The absence of this self-correcting mechanism means that the simulated economy often behaves unrealistically, for example achieving employment of 0 or greater than 1.

In this economy, firm actions are sensitive to θ_t and there is a hard upper bound of 1 on employment. The proposed solution method does not allow for these features, so aggregates tend to diverge and, often, overshoot full employment.

As an extreme but illustration example, consider the case where the chosen assumption is that $L_t = 0 \ \forall t$. The firms assume low market tightness, a high probability of a vacancy being matched and hence post many vacancies. During simulation L_t rises, but firms continue to operate under the illusion that it is 0 and keep posting many vacancies, and under a reasonable calibration, the realised L_t eventually exceeds 1 - more people are employed than exist.

4.5.3.2 Allowing for Feedback

To avoid this problem, the second and third steps of the solution method are adjusted:

1. Guess values for aggregate variables

2. Solve the individual problem assuming those values for all periods, *but allowing for variation in L_t in the current period around that assumed state*

3. Simulate the resulting policies until the distribution converges, *choosing the firm policy conditional on the realised value of L_t in each simulation period*

4. Compare the resulting aggregates to the guess

5. Update the guess and repeat until they match

A numerical Newton-method is used to determine the fixed point at which assumed and realised aggregates are within the chosen margin of error.

4.5.4 Computing the Transition Path

At a high level, the perfect-foresight transition path is determined as follows:

1. Choose the steady-state policies of the end calibration as the initial guess for individual policies
2. Starting from the steady-state distribution of the starting calibration, simulate using the policies until convergence, collecting realised aggregates each period
3. Solve backwards for the policies in each simulated period, starting from the end calibration steady-state policies
4. The generated sequence of policies is the new guess; if it is sufficiently close to the old guess, stop; otherwise, repeat from step 2.

This approach assumes that the final, realised steady state will be the one found during the steady-state computation for the end calibration. In other words, that the starting point does not matter. The assumption is found to be correct to a high degree of numerical accuracy.

During the computation it also transpired that allowing the number of periods solved to decrease between iterations causes instability. The algorithm was adjusted so that the number of periods solved can never decrease, even in the event that a steady state is found more quickly.

4.6 Parameters

The model's parameters, along with the values used in the numerical exercises that follow, are shown in Table 4.2.

The parameters common to the models in this paper and that of Hagedorn and Manovskii (2008) are taken from that paper, largely because the starting point for this project was the literature deriving from Shimer (2005), of which the former contribution is part. The shared parameters are δ, β, ι, s and z.

Parameter	Value	Parameter	Value
δ	$0.99^{\frac{1}{12}}$	ν^v	6
β	.052	ν^r	4
z	0.955	σ	.085
γ	5	ρ	.99
ϕ	.001	ι	.407
τ	42.48%	s	.0081

Table 4.2: Parameter values used

The tax rate τ in the benchmark model is chosen to approximate total government revenues in advanced economies. The OECD average value for this measure was 42.48% in 2014 (OECD, 2014).

The other parameter values are chosen somewhat arbitrarily to yield a model which can be solved and has a non-trivial solution. The main aims were to achieve a relatively wide distribution of firm sizes and some level of entry and exit in steady state.

γ and ϕ are parameters in the utility function which control how strongly prices react to changes in production and, consequently, demand levels. The elasticity of substitution[9] γ must be relatively high, otherwise consumers are very price sensitive and even large demand shocks for individual goods would only yield small additional production. ϕ is set relatively small so as to stay 'close' to a solution without ϕ, whilst still allowing firms with poor demand to choose 0 production, and hence exit.

The values of the size adjustment cost parameters, ν_v and ν_r, were found by repeated upward adjustment to prevent the largest firms from growing beyond the maximum grid size.

The exogenous process for idiosyncratic demand is calibrated as follows:

$$\log(\epsilon_{i,t}) = \rho \log(\epsilon_{i,t-1}) + \xi_{i,t} \qquad \xi_{i,t} \overset{i.i.d.}{\sim} N(0, \sigma) \tag{4.87}$$

The auto-correlation ρ and standard deviation of innovations σ are chosen to yield a some-

[9]Whilst by a strict definition of consumption γ is not precisely the elasticity of substitution, defining consumption of good i as $\hat{c}_i = c_i + \phi$ restores that interpretation (see Acemoglu, 2008).

what persistent process and a sizable spread in firm sizes. This continuous process is approximated by 11 discrete states using the Rouwenhorst method (see Kopecky and Suen, 2010).

It should be noted that all these parameters strongly affect the number of vacancies and redundancies firms post. The first two affect the benefit derived from size adjustment, the second two the cost of size adjustment and the final pair the uncertainty over optimal size. There are therefore strong interactions between the effects of these parameters. A full calibration exercise to uncover these relationships and, more importantly, the effects of the parameters on other economic measures, is beyond the scope of this chapter. The computational effort required is considerable, since it would involve yet another layer of numeric root-finding or optimisation.

4.7 Results

4.7.1 Aggregate Outcomes

Table 4.3 presents the main economic aggregates in steady state in the benchmark calibration.

Variable	Value	Variable	Value
L	0.893	$f(\theta)$	0.168
U	0.107	$\mathfrak{V}$	0.028
Y	2.467	Exiting Firms	0.010
V	0.091	Entry Tax	98.623
R	0.011	Mean Value	140.700
θ	0.851	w	1.042
$q(\theta)$	0.197	Mean Bonus	0.530

Table 4.3: Aggregate Variables in Steady State, Benchmark

The market tightness θ is 0.851, somewhat higher than the value of 0.634 targeted by the calibration of Hagedorn and Manovskii (2008)[10]. As a consequence, the probability of finding a job, $f(\theta)$, is also higher (0.168 vs. 0.139) and the probability of filling a vacancy, $q(\theta)$, lower

[10]The target value was also calculated by the authors from quarterly job-finding and vacancy-filling rates, despite the model period being 1/12th of a quarter. The actual target value for θ should be $.139/.266 = .523$ when weekly probabilities are used.

(0.197 vs. 0.226). Nonetheless, given the possible range of the numbers the fact that they are so close without calibration is remarkable.

The value used for the exogenous separation rate $s = 0.0081$ was chosen to match the empirically observed total separation rate. The endogenous redundancy rate $R = 0.011$ adds to the these separations and, by extension, this economy has a far higher separation rate than is seen in the data. Choosing a lower s would most likely result in a higher R since exogenous separation allows firms to be less active in making people redundant. Addressing this mismatch must, therefore, be part of a wider calibration effort.

Employee remuneration is also interesting: the wage w is 1.042 and the average sign-up bonus paid is 0.530. The average probability of an employee losing her job, $s + R$, is 0.019, so a new employee will expect to receive wages for 52 periods[11] on average. The total remuneration derived from wages far exceeds that of the average sign-up bonus, which represents the entire surplus derived from having a job. This is a result of the high ratio of z to productivity, coupled with low worker bargaining power.

4.7.2 The Distribution of Firms

Figure 4.1 plots the distributions of firms over exogenous demand levels and endogenous firm size.

The distribution by demand illustrates the effect of entry and exit: only firms at the lower end of the demand spectrum exit the market, but entering firms draw demand levels distributed in accordance with the steady state of the Markov process for idiosyncratic demand. Low-demand firms are displaced by new firms, some of which experience high demand. As a result the economy has a higher proportion of high demand firms relative to the Markov process.

The distribution by size, on the other hand, is strongly skewed toward 0, with a very thin tail toward the top end. The distribution is not smooth, which is a result of the discretised shock distribution[12]. Tables 4.4 and 4.5 show the estimated parameters and standard errors

[11] A mere coincidence.

[12] Section B.2 presents the distributions of the model with an identical calibration but 51 levels in the demand distribution, rather than 11. It is smoother.

of the size distribution under the assumption of the distribution being log-normal or Pareto respectively. The log-normal distribution is a better fit, with an AIC of 3095 compared to 5694 for the Pareto model, contradicting empirical results for the firm size distribution (see Axtell, 2001).

Restricting the estimation to the tail of the distribution, the fit of the Pareto distribution improves relative to that of the log-normal one. For firms greater than 0.5 in size, the Pareto distribution has a marginally better AIC than the log-normal, 5656 vs. 5674, though the Pareto parameter of 1.92 is no longer close to the empirically observed value of 1.27 (Axtell, 2001).

	Estimate	Std. Error
m	-1.03	0.03
sd	1.47	0.02

Table 4.4: Estimated Parameters of Size Distribution as a Log-Normal Distribution

	Estimate	Std. Error
α	1.23	0.01

Table 4.5: Estimated Parameters of Size Distribution as a Power-Law Distribution

4.7.3 Firms' Policies and Outcomes

Figure 4.2 plots firms vacancy-posting and redundancy policies, firm value and the firms' sign-up bonus against firm size.

4.7.4 A Change in Tax Policy

4.7.4.1 The Steady-State After Adjustment

This section investigates the effect of a change in tax policy: the government raises the tax rate by 1% to 43.48%. The government also adjusts the market entry fee to maintain a measure 1 of firms throughout[13].

[13]This aspect of the experiment is by necessity rather than choice: dealing with variable market entry is difficult and beyond the scope of this project. This point is discussed further in Section 4.8.

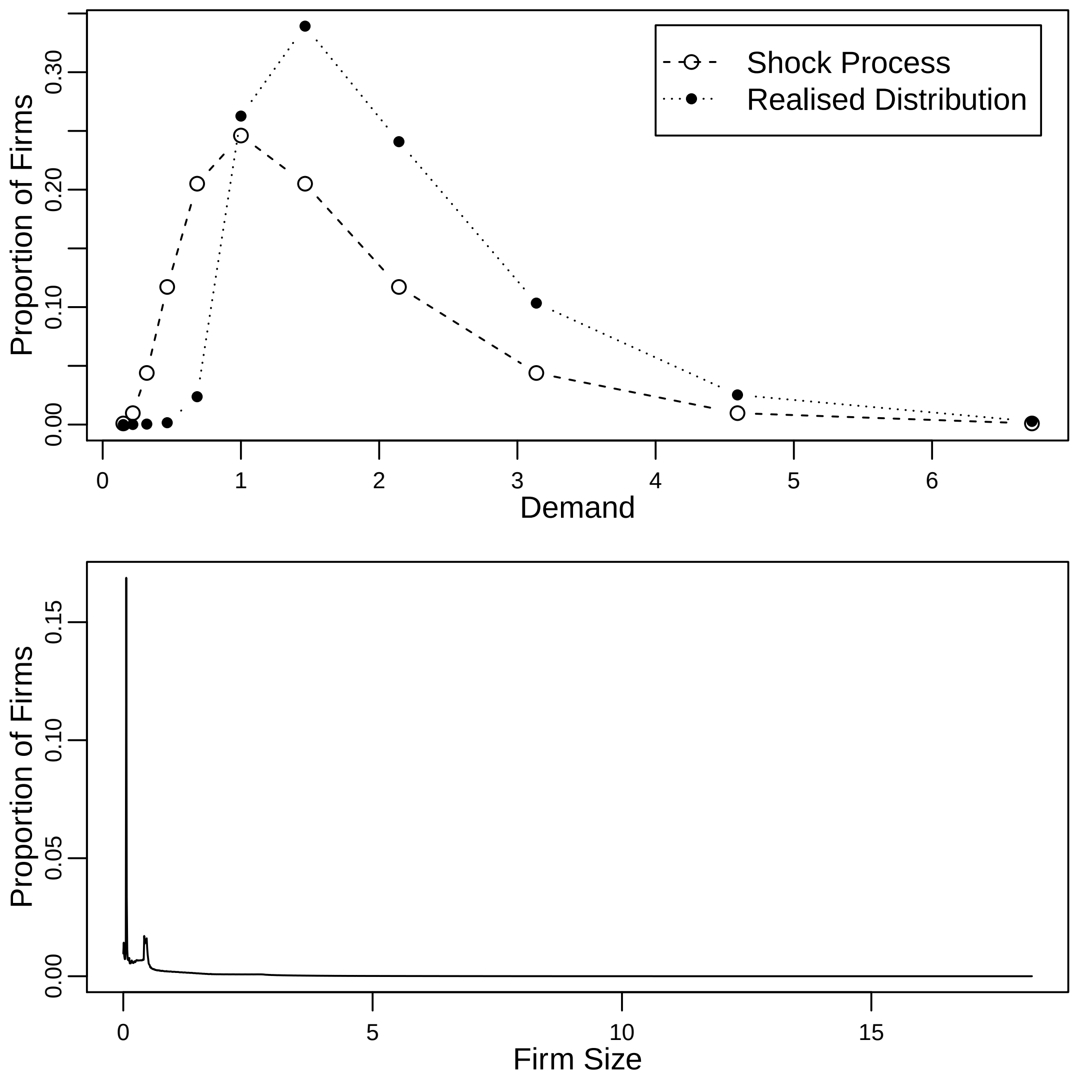

Figure 4.1: The Distribution of Firms by Level of Demand (top) and Size

The top graph illustrates the effect of exit and entry: though the shock process has a log-normally distributed steady state, after exit and entry there are more firms toward the top end of the distribution. Despite this, the firm size distribution is heavily skewed toward 0.

Note: The largest firms is of size 18.22, but the tail is very thin.

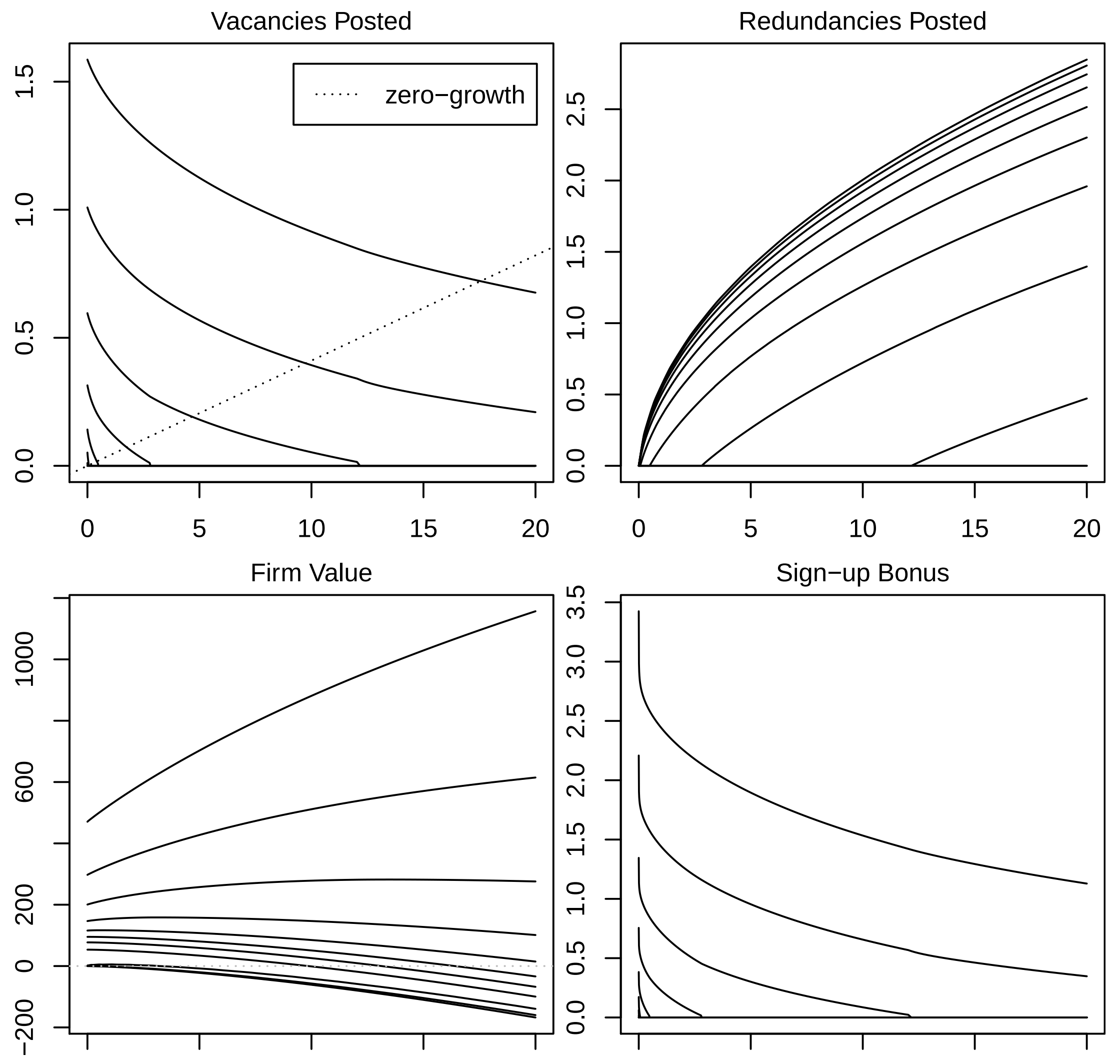

Figure 4.2: Firm-Specific Policies and Outcomes, By Size
The graphs are plotted for each of the levels of demand a firm can experience. For redundancies, the order from top to bottom is from least to greatest demand; for the other variables the situation is reversed.

First, consider the new steady state, presented in Table 4.6. The 1% increase in the tax on firm output has reduced employment by 0.7%, mainly through its impact on vacancies posted, V, which have decreased from 0.091 to 0.089. Both R and the proportion of firms exiting the economy each period have not changed significantly.

The value of firms has decreased on average as a result of the need to dedicate more output to taxation. Since the same also applies for newly entered firms, the entry tax charged to maintain a steady population of firms of measure 1 has also decreased, from 98.6 to 96.0.

Variable	Value	Variable	Value
L	0.886	$f(\theta)$	0.160
U	0.114	$\mathfrak{V}$	0.027
Y	2.466	Exiting Firms	0.010
V	0.089	Entry Tax	95.961
R	0.011	Mean Value	136.983
θ	0.781	w	1.034
$q(\theta)$	0.205	Mean Bonus	0.506

Table 4.6: Aggregate Variables in Steady State, High-Tax Scenario

4.7.4.2 The Transition Path

Figure 4.3 displays the development of the main aggregate variables during the first 30 periods after the government changes its policy. V and, as a consequence, θ, are jump variables and react strongly to the initial shock. Both variables continue to change thereafter, but now in opposite directions as the gradual adjustment of the firm size distribution causes U to adjust more rapidly than V after the initial shock.

This economy replicates the persistence of market tightness observed in Acemoglu and Hawkins (2014), but it does so even in this scenario where the measure of firms is constant. In addition θ is strongly pro-cyclical since U and V adjust in opposite directions in response to the shock.

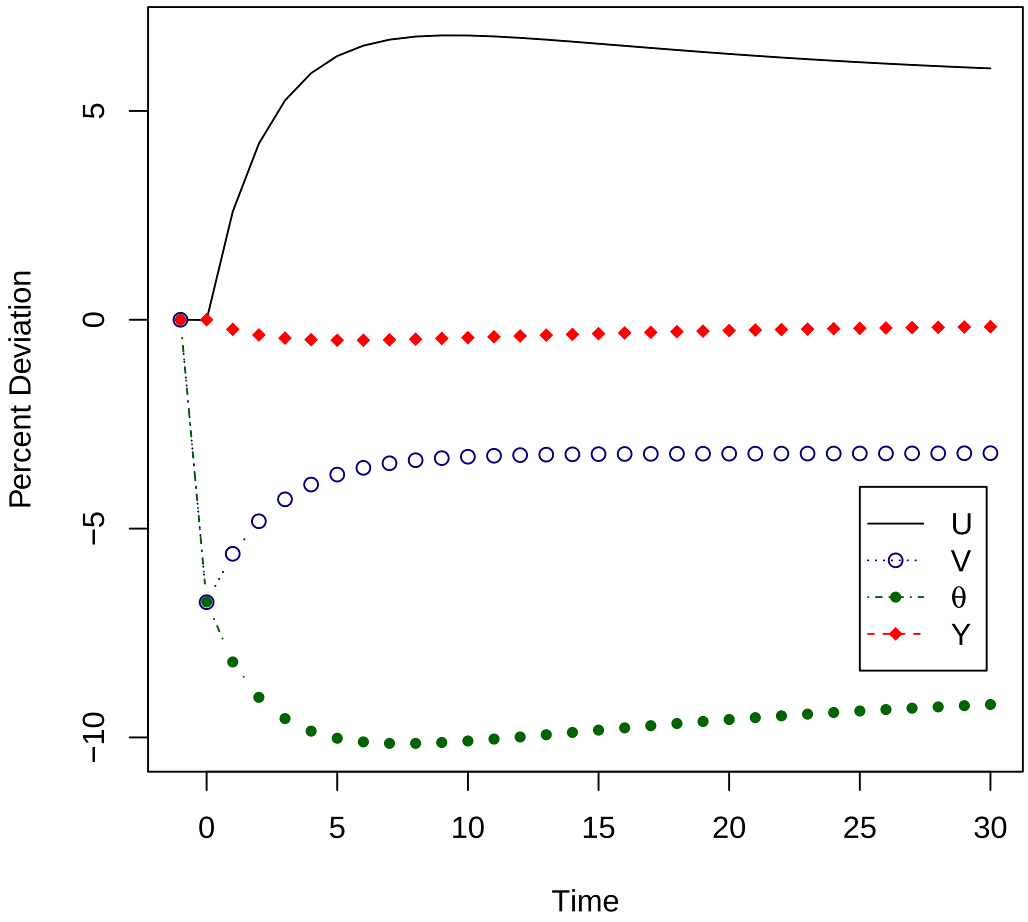

Figure 4.3: Aggregates During The Transition from Low to High Tax Regime

4.8 Discussion And Further Work

This chapter has presented a generalization of the standard random search model of unemployment in which firms experience differing levels of demand and, as a result, differ in size. Firms can engage in both costly hiring and firing, incumbent firms can exit the market and new firms can enter.

Some of the contribution of this work is technical: it demonstrates that the toolkit introduced in Chapter 2 can be used to solve models with heterogeneity in which multiple aggregate variables influence individual agents' behaviour. It has also used the Method of Endogenous Gridpoints (Carroll, 2006) in a setting other than that for which it was first developed.

On the economic side, the model introduces a novel approach to surplus splitting in an economy with Nash bargaining: the employee's share of the surplus is paid as a sign-up bonus and wages are set to make the employee indifferent between remaining in work and becoming unemployed. This approach allows the firms' problem to be solved with both endogenous redundancies and non-trivial surplus sharing, which is not possible under a continuously renegotiated wage.

4.8.1 Entry and Aggregate Shocks

Two interesting aspects of the model economy have not been fully studied in this chapter: firm entry and aggregate shocks. The exclusion of both aspects stems from the same issue: at present, the level of entry and vacancies can not be jointly resolved in equilibrium.

Entry should be resolved via its impact on vacancies: the more new firms enter, the more vacancies they will post, which is expected to decrease the value of each entering firm due to greater competition. An equilibrium is reached when entering firms' value matches the entry cost. This mechanism fails because, in the current implementation, firm value barely varies in aggregate vacancies, and the variation it does have is in the 'wrong' direction: firm value is rising in vacancies so the more firms enter the more valuable they get, leading to unlimited entry.

One cause of this issue is that expectations do not vary in aggregate vacancies. By the terminology of this paper, vacancies are an ‘aggregate control’. The default[14] behaviour of the ModelSolver Toolkit is that expectations do not depend on aggregate controls. This behaviour is a logical extension of individual behaviour to aggregates: just as a firm, given its current state, determines both its current controls and its future state from them, so current aggregate states determine the level of current aggregate controls and future aggregate states. It is also an assumption widely followed in the literature since at least Krusell and Smith (1997). Unlike the bond price and aggregate capital in that model, however, aggregate vacancies directly[15] affect future aggregate employment. Expectations must hence vary by the number of aggregate vacancies.

The solution approach for finding the steady state without aggregate risk, outlined earlier, avoids the problem by assuming that vacancies are in equilibrium at a pre-determined level, and that entry will always match exit. It then adjusts the assumed level of vacancies repeatedly until the first assumption is met, and forces the second assumption to be met by setting the entry cost at the appropriate level.

Both calculating a transition path with a constant entry cost and a variable total measure of firms, and simulating a model which allows for aggregate uncertainty, require vacancies and entry to be determined in equilibrium. To perform these calculations, expectations must depend on vacancies.

The toolkit has been extended to allow for aggregate controls to affect expectations in some processes, for example in the method of endogenous gridpoints. This work is not complete and bears higher computational cost and many potential pitfalls with it. First experiments suggest it does address the problem.

[14]Until recently, only.

[15]In the sense of appearing in the algebraic equation which determines future aggregate employment given current aggregate variables.

4.8.2 Further Work

A key further step is to perform a full calibration exercise. Parameters in the model were taken from the literature on economies that are not precisely equivalent, or chosen somewhat arbitrarily. As a result, some aspects of the solution do not match stylized facts of the economy well: most obviously, $s + R$ is too high relative to the observed separation rate.

5 Discussion

This thesis has considered theoretical macroeconomic models with substantial heterogeneity. It has introduced a software library which helps to solve such models, a new technique for solving such models, and finally a new instance of such a model which aims to address questions concerning the interaction of firm size heterogeneity with frictions in the labour market and wage bargaining. All three areas offer significant scope for future work.

5.1 The ModelSolver Toolkit

5.1.1 Computational Performance

The experiments in Chapter 2 demonstrated that the ModelSolver Toolkit computes a model solution quickly relative to hand-crafted Matlab code.

One reason for this comparatively high level of performance is that the underlying technology, the Java VM, is faster than Matlab (Aruoba and Fernández-Villaverde, 2014).

A further reason is that the toolkit was designed explicitly to take advantage of multiple cores when running array operations. This parallelisation automatically follows array dimensions, which many economic problems can benefit from. A common feature of solving economic models is that functions are constrained or expected to be well-behaved in certain inputs, which translate to dimensions in the ModelSolver toolkit. When an operation is performed along such a dimension, the toolkit allows the operation to be run for each point in the other dimensions in parallel. In the method of endogenous gridpoints, the calculation of the expectational part of the Euler equation is performed in this way, along the dimension of the individual state. Because all aggregates and exogenous variables are equal for an agent in any of those states, these only need to be determined once in each parallel call. Since the constraints apply along this dimension they can also be easily considered under this approach.

5.1.2 Ease of Use

The real test of a model-solving technology is now 'how fast does it run' but 'how quickly can a model be implemented'. Researchers are likely to spend many hours writing the solution before they run it.

The Scala programming language offers a powerful combination of features: it runs on a fast platform which executes compiled code, but if offers rich language features for expressing domain-specific concepts in an intuitive way. The ModelSolver Toolkit takes advantage of these features to allow economists to work with familiar concepts, such as multi-dimensional arrays, using familiar notation. There is scope for further improvement, as outlined below.

It is difficult to compare ease of use directly. The lines of code that need to be written is similar to the hand-crafted Matlab solution: both the minimal incomplete markets example and the code written for Den Haan and Rendahl (2010) have around 360 lines of functional code. Many of the lines in the ModelSolver version are 'boilerplate' such as class or method headers. These models are very simple, and the hand-coded solution is likely to grow by more than the ModelSolver one as the model gets more complex. Implementing the solution for Chapter 4 was quite quick once the appropriate mathematics had been found. Given the relatively large number of variables, all the variations in their behaviour[1], a manual solution would have been time-consuming .

5.1.3 Comparable Solutions

5.1.3.1 HARK

There is an existing open-source, researcher-led effort to build a toolkit for solving economic models in the form of HARK (Carroll et al., 2016). As stated in the introduction, this work has only recently come to my attention. Its goals, essentially identical to those of the ModelSolver Toolkit, are stated in the manual:

> *...But modellers whose questions require explicit structural modelling involving non-trivial kinds of heterogeneity (that is, heterogeneity that cannot simply be aggregated away) are mostly still stuck in the bad old days.*
>
> *The ultimate goal of the HARK project is to fix these problems.*

HARK and the ModelSolver Toolkit share some similarities: the separation between individual and aggregate problems, and the provision of tools such as interpolators for assistance in solving the problems. Both also require a significant amount of programming at present. HARK uses python and builds on the numpy and scipy libraries, both of which are popular within the scientific community.

[1]An aggregate state, a control with contemporaneous impact and a control with impact only via expectations.

Based on the sample models provided it appears that HARK is not quite as capable as the ModelSolver Toolkit as it stands. On the other hand, it has been presented at conferences and thus received the first exposure to the wider community of researchers. Some collaboration and exchange of ideas may be fruitful.

5.1.4 Potential Improvements

5.1.4.1 Greater Flexibility

Models with heterogeneous agents vary to a great degree, not only in the types of agents they may consider but also in the techniques necessary to solve them. The model of Chapter 4 can not yet be solved under the assumption of aggregate uncertainty using the ModelSolver Toolkit, highlighting the fact that a single approach is not applicable to all such models. Extending the toolkit to cover more cases will make it useful to a greater number of researchers and, more importantly, encourage a greater number of researchers to consider heterogeneity in their research. One way of performing such extensions is to provide additional solvers which work better in scenarios where, for example, the method of endogenous gridpoints can not be applied.

Any such extension should be driven by 'exogenous shocks' in the form of interest from, and perhaps contributions by, the research community.

5.1.4.2 A Modelling Language

An aspect of Dynare (Adjemian et al., 2011) which contributes to its ease of use is the fact that little programming is necessary. Models are specified in a custom language designed for that purpose and which leans heavily on mathematical notation.

A zero-programming approach to solving heterogeneous agent models is perhaps not feasible due to wide variety of such models. A well-designed modelling language, used to express the individual solution equations and the aggregation functions, may nonetheless be a powerful additional tool in the toolkit. The equations solving all the models in this thesis contain relatively simple algebra, and a consistent meaning of symbols such as that laid out in Section 3.4.1 would allow those equations to be expressed in a common notation. Programming may only be necessary for handling special cases, such as the separation of bond and capital holdings in Section 3.5.

5.1.4.3 Graphical Processors

Certain aspects of scientific computing have benefited greatly from the ability to attain a very high level of parallelisation using graphical processors. This technology has been used in economics (Aldrich et al., 2011). The tools presented here do not use this approach. The complexity of solution algorithms may render an effective, general model-solving toolkit using graphical processors, which have strong restrictions on how input data is presented, impossible to implement. The approach is worth investigating, however, especially if a modelling language is used and the translation to runnable code hides the complexity of working with graphical processors.

5.2 Derivative Aggregation

Chapter 3 introduced the method of Derivative Aggregation for updating the aggregate forecasting function in a model with heterogeneous agents and aggregate uncertainty. It was shown to deliver a function with forecasting accuracy comparable to other solutions in the literature, but requiring less compute time.

5.2.1 Efficiency

The method is efficient in the sense that it requires little compute time relative to the canonical approach of Krusell and Smith (1998). The primary reason for this efficiency is that it requires very little simulation: the Krusell-Smith approach requires around 600 simulation steps in each iteration of updating the forecasting function, Derivative Aggregation just 2. Simulation is a very expensive step in models with heterogeneous agents, so avoiding it reduces the required compute time dramatically.

5.2.2 Few Constraints

The method places few constraints on models. The principal constraint is that there should be a clear relationship between individual and aggregate variables which can be expressed in the form presented in Section 3.4.1. This lack of constraints is the reason it performs well relative to Den Haan and Rendahl (2010): in that approach, individual and aggregate variables must be related in a specific way which, in the sample model, requires the addition of a second aggregate state variable, increasing the number of dimensions and hence the computational complexity.

The Krusell-Smith approach imposes even fewer constraints: only that there be a relatively stable relationship between current aggregate states and both current aggregate controls and future aggregate states. There may be models amenable to solution by the Krusell-Smith algorithm to which Derivative Aggregation can not be applied.

5.2.3 Theoretical Insight

The key equation of derivative aggregation is Eq. (3.8), restated here:

$$\frac{dK_{t+1}}{dK_t} = \int_0^1 \left(\frac{\partial f}{\partial k_{i,t}} \frac{dk_{i,t}}{dK_t} + \frac{\partial f}{\partial K_t} \right) di \tag{5.1}$$

It specifies how the derivative of the aggregate forecasting function is calculated from the derivatives of the individual policy function and a function approximating the change in contemporaneous individual states as the aggregate state changes.

The equation also demonstrates when exact or approximate aggregation hold. Exact aggregation requires either that both partial derivatives are identical across all agents i, or that $\frac{dk_{i,t}}{dK_t}$ is unique. The former will occur exactly only in the case of representative agent or a trivial policy function, the later implies that for each level of K_t there is exactly one underlying distribution of $k_{i,t}$. Approximate aggregation holds when the derivative $\frac{dk_{i,t}}{dK_t}$ does not vary widely, implying that the set of realised distributions of the economy is relatively tight around the mean for each possible aggregate state value.

Further contemplation of these cases may yield valuable insights. There is also scope for further research into whether the relationship can be applied to yield insights into which models allow approximate aggregation and which do not.

5.3 Matching, Bargaining and Heterogeneous Agents

Chapter 4 offers the clearest scope for further research. The first goal is to address the problem of calculating firm entry in equilibrium. This would enable solution of both a transition path with endogenous entry, and a version of the model with aggregate uncertainty. An analysis of the interaction of job creation and destruction, firm entry and exit and employment in this economy should follow.

The impact of modelling choices, particularly the structure of remuneration and of size adjustment costs, should be subjected to further analysis. What other choices are possible whilst still allowing the model to be solved? What is the impact of these alternatives?

Finally, the model should be brought to the data. There is a wealth of both aggregate and firm-level data regarding hiring and firing, and investigating the fit of both sets to the model, and particularly how they interact, is an interesting future project.

Bibliography

Acemoglu, D. (2008). *Introduction to modern economic growth*. Princeton University Press.

Acemoglu, D. and Hawkins, W. B. (2014). Search with multi-worker firms. *Theoretical Economics*, 9(3):583–628.

Adjemian, S., Bastani, H., Juillard, M., Mihoubi, F., Perendia, G., Ratto, M., and Villemot, S. (2011). Dynare: Reference manual, version 4. Technical report, Dynare Working Papers 1, CEPREMAP.

Aiyagari, S. R. (1994). Uninsured idiosyncratic risk and aggregate saving. *The Quarterly Journal of Economics*, pages 659–684.

Aldrich, E. M., Fernández-Villaverde, J., Gallant, A. R., and Rubio-Ramírez, J. F. (2011). Tapping the supercomputer under your desk: Solving dynamic equilibrium models with graphics processors. *Journal of Economic Dynamics and Control*, 35(3):386–393.

Algan, Y., Allais, O., and Den Haan, W. J. (2008). Solving heterogeneous-agent models with parameterized cross-sectional distributions. *Journal of Economic Dynamics and Control*, 32(3):875–908.

An, S., Chang, Y., and Kim, S.-B. (2009). Can a representative-agent model represent a heterogeneous-agent economy. *American Economic Journal: Macroeconomics*, pages 29–54.

Aruoba, S. B. and Fernández-Villaverde, J. (2014). A Comparison of Programming Languages in Economics. Technical report, National Bureau of Economic Research.

Axtell, R. L. (2001). Zipf distribution of us firm sizes. *Science*, 293(5536):1818–1820.

Benhabib, J., Wang, P., and Wen, Y. (2015). Sentiments and aggregate demand fluctuations. *Econometrica*, 83(2):549–585.

Bezanson, J., Chen, J., Karpinski, S., Shah, V., and Edelman, A. (2014a). Array operators using multiple dispatch: A design methodology for array implementations in dynamic languages. In *Proceedings of ACM SIGPLAN International Workshop on Libraries, Languages, and Compilers for Array Programming*, page 56. ACM.

Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B. (2014b). Julia: A fresh approach to numerical computing. *CoRR*, abs/1411.1607.

Brock, W. A. and Hommes, C. H. (1997). A rational route to randomness. *Econometrica: Journal of the Econometric Society*, pages 1059–1095.

Carroll, C. D. (2000). Requiem for the representative consumer? aggregate implications of microeconomic consumption behavior. *The American Economic Review*, 90(2):110–115.

Carroll, C. D. (2006). The method of endogenous gridpoints for solving dynamic stochastic optimization problems. *Economics letters*, 91(3):312–320.

Carroll, C. D., Kaufman, A. M., Low, D. C., Palmer, N. M., and White, M. N. (2016). Hark: Heterogeneous agents resources and toolkit.

Cooper, R., Haltiwanger, J., and Willis, J. L. (2007). Search frictions: Matching aggregate and establishment observations. *Journal of Monetary Economics*, 54:56–78.

Den Haan, W. J. (2010a). Assessing the accuracy of the aggregate law of motion in models with heterogeneous agents. *Journal of Economic Dynamics and Control*, 34(1):79–99.

Den Haan, W. J. (2010b). Comparison of solutions to the incomplete markets model with aggregate uncertainty. *Journal of Economic Dynamics and Control*, 34(1):4–27.

Den Haan, W. J., Judd, K. L., and Juillard, M. (2010). Computational suite of models with heterogeneous agents: Incomplete markets and aggregate uncertainty. *Journal of Economic Dynamics and Control*, 34(1):1–3.

den Haan, W. J., Ramey, G., and Watson, J. (2000). Job destruction and propagation of shocks. *American Economic Review*, 90(3):482–498.

Den Haan, W. J. and Rendahl, P. (2010). Solving the incomplete markets model with aggregate uncertainty using explicit aggregation. *Journal of Economic Dynamics and Control*, 34(1):69–78.

Dixit, A. K. and Stiglitz, J. E. (1977). Monopolistic competition and optimum product diversity. *The American Economic Review*, 67(3):297–308.

Elsby, M. W. and Michaels, R. (2013). Marginal jobs, heterogeneous firms, and unemployment flows. *American Economic Journal: Macroeconomics*, 5(1):1–48.

Felbermayr, G., Prat, J., and Schmerer, H.-J. (2011). Globalization and labor market outcomes: wage bargaining, search frictions, and firm heterogeneity. *Journal of Economic Theory*, 146(1):39–73.

Fujita, S. and Nakajima, M. (2016). Worker flows and job flows: A quantitative investigation. *Review of Economic Dynamics*.

Fujita, S. and Ramey, G. (2007). Job matching and propagation. *Journal of Economic dynamics and control*, 31(11):3671–3698.

Grasl, T. (2011a). The modelsolver library. https://bitbucket.org/modelsolver/modelsolver.

Grasl, T. (2011b). The numerics library. https://bitbucket.org/modelsolver/numerics.

Grasl, T. (2014a). Incomplete marketes - scala. https://bitbucket.org/modelsolver/incomplete-markets-scala.

Grasl, T. (2014b). Matching model. https://bitbucket.org/modelsolver/matching-model.

Grasl, T. (2014c). Modelsolver-scala. https://bitbucket.org/modelsolver/modelsolver-scala.

Grasl, T. (2014d). Numerics-scala. https://bitbucket.org/modelsolver/numerics-scala.

Grasl, T. (2016). The art of the possible - accompanying code. https://modelsolver.bitbucket.io/phd/.

Hagedorn, M. and Manovskii, I. (2008). The cyclical behavior of equilibrium unemployment and vacancies revisited. *The American Economic Review*, 98(4):1692–1706.

Hopenhayn, H. A. (1992). Entry, exit, and firm dynamics in long run equilibrium. *Econometrica: Journal of the Econometric Society*, pages 1127–1150.

İmrohoroğlu, A. (1992). The welfare cost of inflation under imperfect insurance. *Journal of Economic Dynamics and Control*, 16(1):79–91.

Kaas, L. and Kircher, P. (2015). Efficient firm dynamics in a frictional labor market. *The American Economic Review*, 105(10):3030–3060.

Keynes, J. M. (1936). *The General theory of employment, interest and money*. Palgrave Macmillan.

King, R. G. and Rebelo, S. T. (1999). Resuscitating real business cycles. *Handbook of macroeconomics*, 1:927–1007.

Kopecky, K. A. and Suen, R. M. H. (2010). Finite state markov-chain approximations to highly persistent processes. *Review of Economic Dynamics*, 13(3):701–714.

Krusell, P. and Smith, A. A. (1997). Income and wealth heterogeneity, portfolio choice, and equilibrium asset returns. *Macroeconomic dynamics*, 1(02):387–422.

Krusell, P. and Smith, A. A. (1998). Income and wealth heterogeneity in the macroeconomy. *Journal of Political Economy*, 106(5):867–896.

Krusell, P. and Smith, A. A. (2006). Quantitative macroeconomic models with heterogeneous agents. *Econometric Society Monographs*, 41:298.

Kydland, F. E. and Prescott, E. C. (1982). Time to build and aggregate fluctuations. *Econometrica: Journal of the Econometric Society*, pages 1345–1370.

Lucas, R. E. (1976). Econometric policy evaluation: A critique. In *Carnegie-Rochester conference series on public policy*, volume 1, pages 19–46. North-Holland.

Luttmer, E. G. (2007). Selection, growth, and the size distribution of firms. *The Quarterly Journal of Economics*, pages 1103–1144.

Maliar, L., Maliar, S., and Valli, F. (2010). Solving the incomplete markets model with aggregate uncertainty using the Krusell–Smith algorithm. *Journal of Economic Dynamics and Control*, 34(1):42–49.

Mankiw, N. G. (2006). The macroeconomist as scientist and engineer. *The Journal of Economic Perspectives*, 20(4):29–46.

Mortensen, D. T. and Pissarides, C. A. (1994). Job creation and job destruction in the theory of unemployment. *The review of economic studies*, 61(3):397–415.

Muth, J. F. (1961). Rational expectations and the theory of price movements. *Econometrica: Journal of the Econometric Society*, pages 315–335.

OECD (2014). General government revenue. http://http://www.oecd-ilibrary.org/governance/general-government-revenue/indicator/english_b68b04ae-en. Accessed: 2016-07-03.

Pissarides, C. A. (2000). *Equilibrium unemployment theory*. MIT press.

Reiter, M. (2009). Solving heterogeneous-agent models by projection and perturbation. *Journal of Economic Dynamics and Control*, 33(3):649–665.

Reiter, M. (2010). Solving the incomplete markets model with aggregate uncertainty by backward induction. *Journal of Economic Dynamics and Control*, 34(1):28–35.

Ríos-Rull, J.-V. (1995). Models with heterogeneous agents. *Frontiers of business cycle research*, pages 98–125.

Ríos-Rull, J.-V. (1997). *Computation of equilibria in heterogeneous agent models.* Federal reserve bank.

Samaniego, R. M. (2006). Do firing costs affect the incidence of firm bankruptcy? *Macroeconomic Dynamics*, 10(04):467–501.

Schaal, E. (2012). Uncertainty, productivity and unemployment in the great recession.

Shimer, R. (2005). The cyclical behavior of equilibrium unemployment and vacancies. *American economic review*, pages 25–49.

Simon, C. P. and Blume, L. (1994). *Mathematics for economists*, volume 7. Norton New York.

Simon, H. A. (1972). Theories of bounded rationality. *Decision and organization*, 1(1):161–176.

Stole, L. A. and Zwiebel, J. (1996). Intra-firm bargaining under non-binding contracts. *The Review of Economic Studies*, 63(3):375–410.

Tauchen, G. (1986). Finite state markov-chain approximations to univariate and vector autoregressions. *Economics letters*, 20(2):177–181.

Taylor, J. B. and Uhlig, H. (1990). Solving nonlinear stochastic growth models: A comparison of alternative solution methods. *Journal of Business & Economic Statistics*, 8(1):1–17.

Williams, N. (2009). Seniority, experience, and wages in the uk. *Labour Economics*, 16(3):272–283.

Young, E. R. (2010). Solving the incomplete markets model with aggregate uncertainty using the Krusell–Smith algorithm and non-stochastic simulations. *Journal of Economic Dynamics and Control*, 34(1):36–41.

Appendices

A

Derivative Aggregation

A.1 Mathematical Appendix

A.1.1 Proof of Proposition 1

The proofs of the two equations of the theorem proceed independently:

Proof of 3.24.

From Eq. (3.20):

$$\begin{aligned} X^j_{t+1} &= \int F^j(x_{i,t+1}, X^{-j}_{t+1}) \text{ di} \\ &= \int F^j\left(f_x(x_{i,t}, a_{i,t}, X_t, A_t), X^{-j}_{t+1}\right) \text{ di} \end{aligned} \tag{A.1}$$

So if the distribution is transformed according to $\mathbb{T}_k$ in period t, and using $\tilde{X}^l_{t+1}$ to denote next period aggregate variables that result from applying the transition in this period:

$$\begin{aligned} \tilde{X}^j_{t+1} &= \int F^j\left(f_x(\tilde{x}_{i,t}, a_{i,t}, \tilde{X}_t, A_t), \tilde{X}^{-j}_{t+1}\right) \text{ di} \\ &= \int F^j\left(f_x(T_k(x_{i,t}, \theta_k), a_{i,t}, \tilde{X}_t, A_t), \tilde{X}^{-j}_{t+1}\right) \text{ di} \end{aligned} \tag{A.2}$$

Taking the derivative w.r.t θ_k

$$\begin{aligned} \frac{d\tilde{X}^j_{t+1}}{d\theta_k} &= \frac{d}{d\theta_k}\int F^j\left(f_x(T_k(x_{i,t}, \theta_k), a_{i,t}, \tilde{X}_t, A_t), \tilde{X}^{-j}_{t+1}\right) di \\ &= \int \left[\sum_{s=1}^{n_x}\left(\frac{\partial F^j(.)}{\partial x^s}\frac{df^s_x(T(x_{i,t}, \theta_k), a_{i,t}, \tilde{X}_t, A_t)}{d\theta_k}\right) + \sum_{s=1}^{j-1}\left(\frac{\partial F^j(.)}{\partial X^s}\frac{d\tilde{X}^s_{t+1}}{d\theta_k}\right)\right] di \\ &= \sum_{s=1}^{n_x}\int\left(\frac{\partial F^j(.)}{\partial x^s}\left[\sum_{r=1}^{n_x}\frac{\partial f^s_x(.)}{\partial x^r}\frac{d\tilde{x}^r_{i,t}}{d\theta_k} + \frac{\partial f^s_x(.)}{\partial X^k}\frac{d\tilde{X}^k_t}{d\theta_k}\right]\right) \text{di} + \sum_{s=1}^{j-1}\int\left(\frac{\partial F^j(.)}{\partial X^s}\frac{d\tilde{X}^s_{t+1}}{d\theta_k}\right) \text{di} \end{aligned} \tag{A.3}$$

By the chain rule,

$$\frac{d\tilde{X}^j_{t+1}}{d\tilde{X}^k_t} = \frac{d\tilde{X}^j_{t+1}}{d\theta_k}\frac{d\theta_k}{d\tilde{X}^k_t}$$

so that

$$\frac{d\tilde{X}^j_{t+1}}{d\tilde{X}^k_t} = \sum_{s=1}^{n_x}\int\left(\frac{\partial F^j(.)}{\partial x^s}\left[\sum_{r=1}^{n_x}\frac{\partial f^s_x(.)}{\partial x^r}\frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t} + \frac{\partial f^s_x(.)}{\partial X^k}\right]\right) \text{di} + \sum_{s=1}^{j-1}\int\left(\frac{\partial F^j(.)}{\partial X^s}\frac{d\tilde{X}^s_{t+1}}{d\tilde{X}^k_t}\right) \text{di} \tag{A.4}$$

where there is only one derivative of f_x with respect to aggregates since $\mathbb{T}_k$ leaves aggregates other than k unchanged by condition (1) of the theorem.

But $\mathbb{T}_k(\omega, 1) = \omega$ by condition (3), so evaluating the derivative at $\theta_k = 1$ yields the desired result. □

Proof of 3.25.

Under the transformation, by definition

$$\frac{d\tilde{x}^r_{i,t}}{d\theta_k} = \frac{\partial}{\partial\theta_k}\left(T^r_k(x_{i,t}, \theta_k)\right)$$

Also note that, since other aggregates are invariant under $\mathbb{T}_k$,the transformed $\tilde{X}^k_t$ is

$$\tilde{X}^k_t = \int F^k(T_k(x_{l,t}, \theta_k), X^{-k})\ \mathrm{dl}$$

so that:

$$\begin{aligned}
\frac{d\tilde{X}^k_t}{d\theta_k} &= \frac{d}{d\theta_k}\left(\int F^k(T_k(x_{l,t}, \theta_k), X^{-k})\ \mathrm{dl}\right) \\
&= \int \frac{d}{d\theta_k}\left(F^k(T_k(x_{l,t}, \theta_k), X^{-k})\right)\ dl \\
&= \int \sum_{s=1}^{n_x}\left(\left(\frac{\partial}{\partial x^s}F^k(T_k(x_{l,t}, \theta_k), X^{-k})\right)\left(\frac{d}{d\theta_k}T^s_k(x_{l,t}, \theta_k)\right)\right)\ \mathrm{dl} \\
&= \sum_{s=1}^{n_x}\int \frac{\partial F^k(T_k(x_{l,t}, \theta_k), X^{-k})}{\partial x^s}\frac{dT^s_k(x_{l,t}, \theta_k)}{d\theta_k}\ \mathrm{dl}
\end{aligned}$$

Combining the two derivatives results in:

$$\begin{aligned}
\frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t} &= \frac{d\tilde{x}^r_{i,t}}{d\theta_k}\frac{d\theta_k}{d\tilde{X}^k_t} \\
&= \frac{d\tilde{x}^r_{i,t}}{d\theta_k}\left(\frac{d\tilde{X}^k_t}{d\theta_k}\right)^{-1} \\
&= \frac{\partial T_k(x_{i,t},\theta_k)}{\partial\theta_k}\left(\sum_{s=1}^{n_x}\int \frac{\partial F^k(T_k(x_{l,t},\theta_k))}{\partial x^s}\frac{\partial T^s_k(x_{l,t},\theta_k)}{\partial\theta_k}\,\mathrm{dl}\right)^{-1} \\
&= \frac{\frac{\partial T^k(x_{i,t},\theta_k)}{\partial\theta_k}}{\sum_{s=1}^{n_x}\int\left(\frac{\partial F^k(T_k(x_{l,t},\theta_k))}{\partial x^s}\frac{\partial T^s_k(x_{l,t},\theta_k)}{\partial\theta_k}\right)\,\mathrm{dl}}
\end{aligned}$$

But $\mathbb{T}_k(\omega,1)=\omega$ by condition (3) of the theorem, so evaluating the derivative at $\theta_k=1$ yields the derivative at ω, the untransformed distribution □

□

A.2 Extension to 2nd derivative

The derivative aggregation can be extended to second and higher order derivatives. This section demonstrates the second derivative. In order to calculate a cross partial derivative w.r.t. different aggregates it is necessary to consider both relevant transformations being applied simultaneously. For straight double derivatives this is not necessary. The mathematics presented below is identical in both cases

For this section, define $\tilde{x}$ and $\tilde{X}^k$ to mean:

$$\begin{aligned}
\tilde{x} &= T_h(T_k(x,\theta_k),\theta_h) \\
\tilde{X^k} &= \int_0^1 F^k(T_h(T_k(x_i,\theta_k),\theta_h),a,X^{-k})di \\
\tilde{X^h} &= \int_0^1 F^h(T_h(T_k(x_i,\theta_k),\theta_h),a,X^{-h})di
\end{aligned}$$

Note that, because T_k, T_h are identity transformations when their parameters are 1, and all derivatives are determined at that value, these definitions do not change the value of $\tilde{X}^k, \tilde{X}^h$ or their first derivatives from that in prior sections. Only when the second derivative is needed to they come into play.

This implies

$$\begin{aligned}
\tilde{X}^j_{t+1} &= \int F^j\left(f_x(\tilde{x}_{i,t},a_{i,t},\tilde{X}_t,A_t),\tilde{X}^{-j}_{t+1}\right)\ \mathrm{di} \\
&= \int F^j\left(f_x(T_h(T_k(x_{i,t},\theta_k),\theta_h),a_{i,t},\tilde{X}_t,A_t),\tilde{X}^{-j}_{t+1}\right)\ \mathrm{di}
\end{aligned}$$

so that:

$$\begin{aligned}
\frac{d^2\tilde{X}^j_{t+1}}{d\tilde{X}^h_t d\tilde{X}^k_t} = \frac{d}{d\tilde{X}^h_t}&\left(\sum_{s=1}^{n_x}\int\left(\frac{\partial F^j(.)}{\partial x^s}\left[\sum_{r=1}^{n_x}\frac{\partial f_x^s(.)}{\partial x^r}\frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t}+\frac{\partial f_x^s(.)}{\partial X^k}\right]\right)\mathrm{di}\right. \\
&\left.+\sum_{s=1}^{j-1}\int\left(\frac{\partial F^j(.)}{\partial X^s}\frac{d\tilde{X}^s_{t+1}}{d\tilde{X}^k_t}\right)\mathrm{di}\right)
\end{aligned}$$

Consider the derivative with respect to $\tilde{X}^h_t$ of each term above:

$$\begin{aligned}
\frac{d}{d\tilde{X}^h_t}\left(\frac{\partial F^j(.)}{\partial x^s}\right) &= \frac{d}{d\tilde{X}^h_t}\left(\frac{\partial F^j(f_x(\tilde{x}_{i,t},a_{i,t},\tilde{X}_t,A_t),\tilde{X}^{-j}_{t+1})}{\partial x^s}\right) \\
&= \sum_{u=1}^{n_x}\left(\frac{\partial^2 F^j(.)}{\partial x^s\partial x^u}\left[\sum_{r=1}^{n_x}\frac{\partial f_x^u(.)}{\partial x^r}\frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^h_t}+\frac{\partial f_x^u(.)}{\partial X^h}\right]\right)+\sum_{u=1}^{j-1}\left(\frac{\partial^2 F^j(.)}{\partial x^s\partial X^u}\frac{d\tilde{X}^u_{t+1}}{d\tilde{X}^h_t}\right)
\end{aligned}$$

where there is only one derivative with respect to aggregates because T_h leaves other aggs unchanged.

$$
\begin{aligned}
\frac{d}{d\tilde{X}_t^h}\left(\frac{\partial f_x^s(.)}{\partial x^r}\right) &= \frac{d}{d\tilde{X}_t^h}\left(\frac{\partial f_x^s(\tilde{x}_{i,t}, a_{i,t}, \tilde{X}_t, A_t)}{\partial x^r}\right) \\
&= \sum_{u=1}^{n_x}\left(\frac{\partial^2 f_x^s(.)}{\partial x^r \partial x^u}\frac{d\tilde{x}_{i,t}^u}{d\tilde{X}_t^h}\right) + \frac{\partial^2 f_x^s(.)}{\partial x^r \partial X^h} \\
\frac{d}{d\tilde{X}_t^h}\left(\frac{d\tilde{x}_{i,t}^r}{d\tilde{X}_t^k}\right) &= \frac{d^2\tilde{x}_{i,t}^r}{d\tilde{X}_t^h d\tilde{X}_t^k} \\
\frac{d}{d\tilde{X}_t^h}\left(\frac{\partial f_x^s(.)}{\partial X^k}\right) &= \frac{d}{d\tilde{X}_t^h}\left(\frac{\partial f_x^s(\tilde{x}_{i,t}, a_{i,t}, \tilde{X}_t, A_t)}{\partial X^k}\right) \\
&= \sum_{u=1}^{n_x}\left(\frac{\partial^2 f_x^s(.)}{\partial X^k \partial x^u}\frac{d\tilde{x}_{i,t}^u}{d\tilde{X}_t^h}\right) + \frac{\partial^2 f_x^s(.)}{\partial X^k \partial X^h} \\
\frac{d}{d\tilde{X}_t^h}\left(\frac{\partial F^j(.)}{\partial X^s}\right) &= \frac{d}{d\tilde{X}_t^h}\left(\frac{\partial F^j(f_x(\tilde{x}_{i,t}, a_{i,t}, \tilde{X}_t, A_t), \tilde{X}_{t+1}^{-j})}{\partial X^s}\right) \\
&= \sum_{u=1}^{n_x}\left(\frac{\partial^2 F^j(.)}{\partial X^s \partial x^u}\left[\sum_{r=1}^{n_x}\frac{\partial f_x^u(.)}{\partial x^r}\frac{d\tilde{x}_{i,t}^r}{d\tilde{X}_t^h} + \frac{\partial f_x^u(.)}{\partial X^h}\right]\right) + \sum_{u=1}^{j-1}\int\left(\frac{\partial^2 F^j(.)}{\partial X^s \partial X^u}\frac{d\tilde{X}_{t+1}^u}{d\tilde{X}_t^h}\right) \mathrm{di} \\
\frac{d}{d\tilde{X}_t^h}\left(\frac{d\tilde{X}_{t+1}^s}{d\tilde{X}_t^k}\right) &= \frac{d^2\tilde{X}_{t+1}^s}{d\tilde{X}_t^h d\tilde{X}_t^k}
\end{aligned}
$$

Combining the terms yields a lengthy and complicated equation involving a number of sums, without adding any substance to the argument. If the method needs to be applied, it is likely more straightforward to calculate the sums individually and then apply the formula for the overall derivative above.

As before, all the partial derivatives in the equations above are of functions f_x and F^k, which are known at the time of the calculation. The only quantities present that have not yet been calculated are $\frac{d^2\tilde{X}_{t+1}^s}{d\tilde{X}_t^h d\tilde{X}_t^k}$, which is recursively provided by this calculation, and $\frac{d^2\tilde{x}_{i,t}^r}{d\tilde{X}_t^h d\tilde{X}_t^k}$, which is derived below.

A.2.1 Deriving $\frac{d^2\tilde{x}^r_{i,t}}{d\tilde{X}^h_t d\tilde{X}^k_t}$

$$\frac{d^2\tilde{x}^r_{i,t}}{d\tilde{X}^h_t d\tilde{X}^k_t} = \frac{d}{d\tilde{X}^h_t}\left(\frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t}\right) \tag{A.5}$$

$$= \frac{d}{d\tilde{X}^h_t}\left(\frac{d\tilde{x}^r_{i,t}}{d\theta_k}\frac{d\theta_k}{d\tilde{X}^k_t}\right) \tag{A.6}$$

$$= \frac{d}{d\tilde{X}^h_t}\left(\frac{d\tilde{x}^r_{i,t}}{d\theta_k}\right)\frac{d\theta_k}{d\tilde{X}^k_t} + \frac{d\tilde{x}^r_{i,t}}{d\theta_k}\frac{d}{d\tilde{X}^h_t}\left(\frac{d\theta_k}{d\tilde{X}^k_t}\right) \text{ by the product rule:} \tag{A.7}$$

$$= \frac{d\theta_h}{d\tilde{X}^h_t}\frac{d}{d\theta_h}\left(\frac{d\tilde{x}^r_{i,t}}{d\theta_k}\right)\frac{d\theta_k}{d\tilde{X}^k_t} + \frac{d\tilde{x}^r_{i,t}}{d\theta_k}\frac{d\theta_h}{d\tilde{X}^h_t}\frac{d}{d\theta_h}\left(\frac{d\theta_k}{d\tilde{X}^k_t}\right) \tag{A.8}$$

$$= \frac{d^2\tilde{x}^r_{i,t}}{d\theta_h\, d\theta_k}\frac{d\theta_h}{d\tilde{X}^h_t}\frac{d\theta_k}{d\tilde{X}^k_t} + \frac{d\tilde{x}^r_{i,t}}{d\theta_k}\frac{d\theta_h}{d\tilde{X}^h_t}\frac{d}{d\theta_h}\left(\left[\frac{d\tilde{X}^k_t}{d\theta_k}\right]^{-1}\right) \tag{A.9}$$

$$= \frac{d^2\tilde{x}^r_{i,t}}{d\theta_h\, d\theta_k}\frac{d\theta_h}{d\tilde{X}^h_t}\frac{d\theta_k}{d\tilde{X}^k_t} - \frac{d\tilde{x}^r_{i,t}}{d\theta_k}\frac{d\theta_h}{d\tilde{X}^h_t}\left(\frac{d\tilde{X}^k_t}{d\theta_k}\right)^{-2}\frac{d^2\tilde{X}^k_t}{d\theta_h\, d\theta_k} \tag{A.10}$$

$$= \left(\frac{d\tilde{X}^h_t}{d\theta_h}\frac{d\tilde{X}^k_t}{d\theta_k}\right)^{-1}\left(\frac{d^2\tilde{x}^r_{i,t}}{d\theta_h\, d\theta_k} - \frac{d\tilde{x}^r_{i,t}}{d\tilde{X}^k_t}\frac{d^2\tilde{X}^k_t}{d\theta_h\, d\theta_k}\right) \tag{A.11}$$

The first term of this sum is straightforward to calculate, since $\tilde{x}$ is defined in terms of the two transformations containing θ_h, θ_k, and the second two multiplicands have already been calculated in the course of obtaining the first derivatives.

The second additive term depends more closely on the specific aggregates, transformations and values of k and h.

When $k = h$, it is the double derivative of $\tilde{X}_k$ with respect to its parameter θ_k, and should be relatively straightforward to calculate.

When $k \neq h$, it is the cross partial derivative of $\tilde{X}^k$ w.r.t to both parameters θ_h and θ_k. Though the first condition of Proposition 1 requires that the change in θ_h does not affect $\tilde{X}_k$, it could affect the value that θ_k must take to achieve a given $\tilde{X}_k$, and by extension the rate of change of the latter with respect to the former. Absence of X_h in the formula for the previously obtained first derivative $\frac{d\tilde{X}^k}{d\theta_k}$ does not imply that transforming the distribution according to T_h does not affect the value, because other terms may be affected. Hence, this term may or may not be zero.

A.3 Solving the Individual Problem with Bonds

Ignoring constraint multipliers, Eqs. (3.32) and (3.33) together imply

$$E_t\left[\left(c^i_{t+1}\right)^{-\gamma}(1+r_{t+1}-\delta)\right] = E_t\left[\left(c^i_{t+1}\right)^{-\gamma}\right]\frac{1}{p_t} \tag{A.12}$$

The aggregate productivity process allows only possible values, $A_t \in \{A_-, A_+\}$ where A_- is the bad state and A_+ the good state. Denoting by $\pi_{-,t+1}, \pi_{+,t+1}$ the probability of being in a good or bad state respectively in the next period, and by $E_{-,t}, E_{+,t}$ the corresponding current period expectations conditional on those outcomes, Eq. (A.12) yields

$$\begin{aligned}
&\pi_{-,t+1}E_{-,t}\left[\left(c^i_{t+1}\right)^{-\gamma}\right]R_{-,t+1} + \pi_{+,t+1}E_{+,t}\left[\left(c^i_{t+1}\right)^{-\gamma}\right]R_{+,t+1} \\
&\qquad\qquad = \pi_{-,t+1}E_{-,t}\left[\left(c^i_{t+1}\right)^{-\gamma}\right]\frac{1}{p_t} + \pi_{+,t+1}E_{+,t}\left[\left(c^i_{t+1}\right)^{-\gamma}\right]\frac{1}{p_t} \\
&\Rightarrow E_{+,t}\left[\left(c^i_{t+1}\right)^{-\gamma}\right] = E_{-,t}\left[\left(c^i_{t+1}\right)^{-\gamma}\right]\frac{\pi_{-,t+1}}{\pi_{+,t+1}}\frac{R_{-,t+1}-\frac{1}{p}}{\frac{1}{p}-R_{+,t+1}}
\end{aligned} \tag{A.13}$$

The solution approach utilises this equation as follows:

1. As in the first step of the method of endogenous gridpoints, future marginal utility of consumption is calculated conditional on realised future aggregate shocks, both individual and aggregate
2. Summing across possible individual states, multiplied by their probabilities, gives the expectations $E_{+,t}$ and $E_{-,t}$ above as functions of household wealth x_{t+1} *after interest payments* on the grid
3. Inverting one of the functions allows identification of $x_{+,t+1}$, the wealth held in the good state, as a function of $x_{-,t+1}$, the wealth held in the bad state.

4. Wealth is carried over as a combination of capital and bonds, and hence equates to

$$x_{t+1} = R_{t+1} k_{t+1} + b_{t+1} \tag{A.14}$$

which implies

$$x_{+,t+1} = R_{+,t+1} k_{t+1} + b_{t+1} \tag{A.15}$$

$$x_{-,t+1} = R_{-,t+1} k_{t+1} + b_{t+1} \tag{A.16}$$

so that

$$k_{t+1} = \frac{x_{+,t+1} - x_{-,t+1}}{R_{+,t+1} - R_{-,t+1}} \tag{A.17}$$

From this we can identify k_{t+1} and b_{t+1} corresponding to future wealth holdings. If they both satisfy the constraint, this step is complete. Otherwise, adjust the values to satisfy the constraint.

5. These functions map capital and bond holdings to the exogenous grid of future wealth after interest. But these variables also determine the current wealth not consumed, and together with the standard inter-temporal Euler condition, which yields current consumption, identify the endogenous grid of current wealth levels that correspond to expected future wealth levels.

6. Finally, having identified the current wealth corresponding to each future current wealth level, and the corresponding split between capital and bonds, the algorithm has arrived at k_{t+1} and b_{t+1} as functions of x_t.

A.3.1 Impact on Simulation Methodology

It should be noted that this approach yields policies for capital and bonds carried over at the end of the period in terms of household wealth after interest held at the beginning of the period. Because, during simulation, the wealth after interest in $t+1$ can only be calculated knowing R_{t+1}, simulation is much more onerous in this model than in the benchmark model. Rather than directly interpolating the household wealth transition function, it is necessary to interpolate the capital and bond policies, calculate future aggregate capital from the former, use that to determine the future interest rate and then calculate future household wealth levels after interest by multiplying the capital with realised returns and adding bonds.

A.4 Dealing with Aggregate Controls

Aggregate controls in the model do not affect the mathematics of the algorithm directly, since they can be substituted out of the equations as is common in other approaches. As outlined in Chapter 2, however, the algorithm in this paper solves the individual problem conditional also on the value of the non-state aggregate variable, bond price in this case. One consequence of this decision is that the implementation of the algorithm must take that variable into account when performing derivative aggregation.

A.4.1 Performing Derivative Aggregation

The dependence of the individual transition function on the aggregate controls means that, when the aggregate derivatives are calculated using the formulae derived in Section 3.4, the resulting derivatives will be specific to a particular value of the aggregate controls. The derivatives then need to be adjusted for the dependence of the aggregate controls on the aggregate states.

To simplify exposition, this section presents the mathematics for a single state and control. It extends naturally to the case with multiple instance of both variables.

$$A_{t+1} = H(A_t, C_t) \qquad \text{for some (unknown) function H} \tag{A.18}$$

$$\Rightarrow \frac{dA_{t+1}}{dA_t} = \frac{\partial H}{\partial A} + \frac{\partial H}{\partial C}\frac{dC_t}{dA_t} \tag{A.19}$$

$\frac{\partial H}{\partial A}$ is the value calculated for $\frac{dA_{t+1}}{dA_t}$ in the case with no controls, and the second term is the adjustment for the presence of controls. The first multiplicand is the aggregation of the derivatives of the individual transition as the aggregate control changes.

To derive the second multiplicand, consider that C_t in this case is the equilibrium value. Taking the derivative of Eq. (2.26) by A_t yields:

$$\frac{\partial E^C}{\partial \omega}\frac{d\omega_t}{dA_t} + \frac{\partial E^C}{\partial A} + \frac{\partial E^C}{\partial C}\frac{dC_t}{dA_t} = \frac{dT^C}{dC}\frac{dC_t}{dA_t} \tag{A.20}$$

$$\Rightarrow \frac{dC_t}{dA_t} = \frac{\frac{\partial E^C}{\partial \omega}\frac{d\omega_t}{dA_t} + \frac{\partial E^C}{\partial A}}{\frac{dT^C}{dC} - \frac{\partial E^C}{\partial C}} \tag{A.21}$$

Here, $\frac{dw_t}{dA_t}$ has already been calculated to perform derivative aggregation without controls, the derivatives of E^C are aggregations of the derivatives of the individual control policy functions, aggregated over ω_t, and T^C is a known function so that the derivative can easily be calculated.

Substituting Eq. (A.21) into Eq. (A.19) yields the full formula for calculating the derivative of the future aggregate with respect to the current aggregate when control variables are present.

Moreover, the result of Eq. (A.21) is also used to calculate the expected future control values from the expected future aggregate values.

A.5 Metrics

The metrics uses to assess how closely the agents' forecasts fit actual outcomes are taken from Den Haan (2010b).

A.5.1 Aggregate Error

The first metric performs two simulations: first, a distribution of individuals is simulated over the 10000 periods, and the aggregates arrived at are calculated. Then the same sequence of shocks is used to generate a time series of aggregates implied purely by the aggregate forecast rule provided in the solution, starting from the same aggregate state. This second time series is effectively what the agents in period 1 would forecast up to 10000 periods ahead, given a known shock sequence, whilst the first time series provides the actual outcomes. The mean and maximum percentage difference between these time series provide a good indication of how well the aggregate transition rule forecasts the economy over long time horizons.

A.5.2 Dynamic Euler Equation Error

The second test, generating the *Dynamic Euler Equation Error*, is performed in conjunction with the first simulation described above. During this simulation, a sample individual is simulated using two methods: first, using the individual transition rule from the solution. Then by using that rule only to calculate the conditional expectation of future consumption, and deriving current consumption and hence future wealth directly from the Euler equation and the budget constraint. The two time series are again compared. This test indicates how well the individual decision rule derived matches the actual decisions that agents would take if they solved the problem in each period.

B

Matching

B.1 Solving the Consumption Problem

The representative family's consumption problem is

$$\max_{\{c_{i,t}\}_{i=0}^{1}} \left(\int_0^1 \epsilon_{i,t}(c_{i,t} + \phi)^{\frac{\gamma-1}{\gamma}} di \right)^{\frac{\gamma}{\gamma-1}} s.t. \qquad I_t = \int_0^1 c_{i,t} p_{i,t} di \tag{B.1}$$

The Lagrangian for this problem is

$$\mathcal{L} = \left(\int_0^1 \epsilon_{i,t}(c_{i,t}+\phi)^{\frac{\gamma-1}{\gamma}} di\right)^{\frac{\gamma}{\gamma-1}} + \lambda_t \left(I_t - \int_0^1 c_{i,t}p_{i,t} di\right) \tag{B.2}$$

yielding first order conditions with respect to each $c_{i,t}$ of

$$\frac{d\mathcal{L}}{dc_{i,t}} = 0 \Rightarrow \frac{\gamma}{\gamma-1}\left(\int_0^1 \epsilon_{i,t}(c_{i,t}+\phi)^{\frac{\gamma-1}{\gamma}} di\right)^{\frac{1}{\gamma-1}} \epsilon_{i,t}\frac{\gamma-1}{\gamma}(c_{i,t}+\phi)^{-\frac{1}{\gamma}} - \lambda_t p_{i,t} = 0 \tag{B.3}$$

Letting

$$C_t = \left(\int_0^1 \epsilon_{i,t}(c_{i,t}+\phi)^{\frac{\gamma-1}{\gamma}} di\right)^{\frac{\gamma}{\gamma-1}} \tag{B.4}$$

this reduces to

$$p_{i,t} = \lambda_t^{-1}\epsilon_{i,t}\left(\frac{C_t}{c_{i,t}+\phi}\right)^{\frac{1}{\gamma}} \tag{B.5}$$

But then:

$$(c_{i,t}+\phi)p_{i,t} = \lambda_t^{-1}\epsilon_{i,t}C_t^{\frac{1}{\gamma}}(c_{i,t}+\phi)^{1-\frac{1}{\gamma}} \tag{B.6}$$

$$\Rightarrow \int_0^1 (c_{i,t}+\phi)p_{i,t} \text{ di} = \int_0^1 \lambda_t^{-1}\epsilon_{i,t}C_t^{\frac{1}{\gamma}}(c_{i,t}+\phi)^{1-\frac{1}{\gamma}} \text{ di} \tag{B.7}$$

$$\Rightarrow I_t + \phi\int_0^1 p_{i,t} \text{ di} = \lambda_t^{-1}C_t^{\frac{1}{\gamma}}\int_0^1 \epsilon_{i,t}(c_{i,t}+\phi)^{1-\frac{1}{\gamma}} \text{ di} \tag{B.8}$$

$$\Rightarrow I_t + \phi P_t = \lambda_t^{-1}C_t^{\frac{1}{\gamma}}C_t^{\frac{\gamma-1}{\gamma}} \tag{B.9}$$

$$\text{where } P_t = \int_0^1 p_{i,t} \text{ di} \tag{B.10}$$

$$\Rightarrow \lambda_t^{-1} = \frac{I_t + \phi P_t}{C_t} \tag{B.11}$$

Substituting this into Eq. (B.5) leads to

$$p_{i,t} = \frac{I_t + \phi P_t}{C_t} \epsilon_{i,t} \left(\frac{C_t}{c_{i,t} + \phi} \right)^{\frac{1}{\gamma}} \tag{B.12}$$

B.2 A 'Wider' Shock Distribution

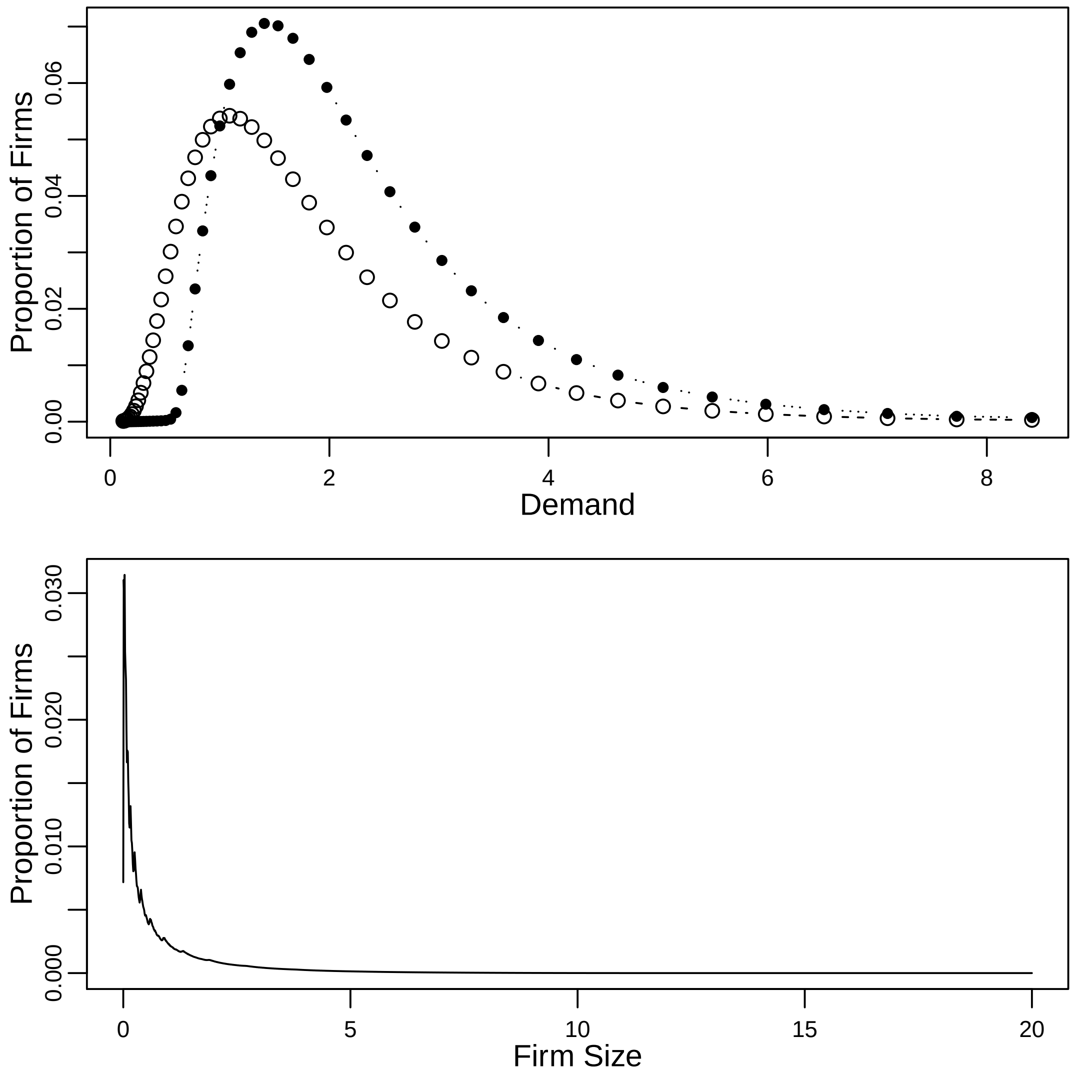

Figure B.1: Firm Distributions with 51 Demand Levels