

OXFORD

INTRODUCTION TO
COMPUTATIONAL
ECONOMICS
USING FORTRAN

HANS FEHR AND FABIAN KINDERMANN

01000100 01101100 01100100 01110010 01100111
01100100 01110101 01100011 01110100 01101001
01101111 01101110 00100000 01110100 01101111
00100000 01000011 01101111 01101101 01110000
01110101 01110100 01100001 01110100 01101001
01101111 01101110 01100001 01101100 00100000
01000101 01100011 01101111 01101110 01101111
01101101 01101001 01100011 01110011 00100000
01100010 01111001 00100000 01001000 01100001
01101110 01110011 00100000 01000110 01100101
01101000 01110010 00100000 01100001 01101110
01100100 00100000 01000110 01100001 01100010
01101001 01100001 01101110 00100000 01001011
01101001 01101110 01100100 01100101 01110010
01101101 01100001 01101110 01100000 01101010

Introduction to Computational Economics Using Fortran

Introduction to Computational Economics Using Fortran

Hans Fehr

Fabian Kindermann



Great Clarendon Street, Oxford, OX2 6DP,
United Kingdom

Oxford University Press is a department of the University of Oxford.
It furthers the University's objective of excellence in research, scholarship,
and education by publishing worldwide. Oxford is a registered trade mark of
Oxford University Press in the UK and in certain other countries

© Hans Fehr and Fabian Kindermann 2018

The moral rights of the authors have been asserted

First Edition published in 2018

Impression: 1

All rights reserved. No part of this publication may be reproduced, stored in
a retrieval system, or transmitted, in any form or by any means, without the
prior permission in writing of Oxford University Press, or as expressly permitted
by law, by licence or under terms agreed with the appropriate reprographics
rights organization. Enquiries concerning reproduction outside the scope of the
above should be sent to the Rights Department, Oxford University Press, at the
address above

You must not circulate this work in any other form
and you must impose this same condition on any acquirer

Published in the United States of America by Oxford University Press
198 Madison Avenue, New York, NY 10016, United States of America

British Library Cataloguing in Publication Data

Data available

Library of Congress Control Number: 2017949307

ISBN 978-0-19-880439-0 (hbk.)
978-0-19-880440-6 (pbk.)

Printed and bound by
CPI Group (UK) Ltd, Croydon, CR0 4YY

Links to third party websites are provided by Oxford in good faith and
for information only. Oxford disclaims any responsibility for the materials
contained in any third party website referenced in this work.

■ PREFACE

Computer technology has been evolving rapidly in the last century. Although Thomas J. Watson, the first president of IBM, once famously projected that ‘there is a world market for maybe five computers’, we nowadays rely on computer work in nearly every situation of life. We write emails, book journeys via the internet, and let computers fly airplanes and conduct trains. Naturally, economics has not been unaffected by this trend. On the one hand, the steady increase in computer performance and speed allowed us to solve problems much faster than before. On the other hand, numerical methods that had been invented in the first half of the twentieth century could finally be applied to problems for which analytical solutions did not exist. Nevertheless, owing to the intensive knowledge requirements regarding programming and numerical maths that were needed to solve real economic problems on a computer, the field of computational economics emerged quite slowly. Nowadays however, where a desktop computer or laptop is a high performance machine and various tools for numerical maths are available, computational economics has become increasingly popular.

There exist many textbooks dedicated to the field of computational economics. One category of these books primarily describes the theory and implementation of numerical methods. Judd (1998) and Press et al. (2001) are popular examples of this kind. Other textbooks like Miranda and Fackler (2002), Adda and Cooper (2003), Kendrick, Mercado, and Amman (2006), or Heer and Maussner (2009) show how to apply those methods to economic problems. Most of these books, however, are dedicated to graduate students and require an above-average knowledge of economic theory, programming, and numerical maths. Very often these books are also specializing in a specific field such as international trade and development, dynamic macroeconomics, or finance. They may offer program codes but typically do not provide information on how to install and use a suitable compiler.

This is where our book tries to step in. We offer an introduction to computational economics to students at all levels of education, regardless of their prior programming experience. This book is based entirely on the programming language Fortran. To facilitate the first steps into writing your own codes, we give an introduction to this particular programming language and demonstrate how to download and use free compilers for different operating systems. Our book offers various examples from economics and finance organized in self-contained chapters that speak to the diverse backgrounds of our readers. While early chapters are accessible for undergraduate students, the level of complexity slowly increases, so that the later part of the book is well suited for graduate students at the Master and Ph.D. level. For each of the topics we consider, we first explain some theoretical background, then show how to implement the problem on the computer, and finally discuss simulation results. Since our book serves as an introduction to using

computational methods in various fields of economics and finance, we provide a list of reading material for further study at the end of each chapter. In addition, readers can work through various exercises which promote practical experience and further deepen the economic and technical insights. Therefore, after reading and working through the book, students should have no problem mastering more sophisticated and specialized textbooks and courses in computational economics at the graduate level.

This textbook contains eleven chapters which are organized in three central parts: Part I offers an introduction to programming and tools from numerical maths, Part II presents various applications for beginners, and in Part III we discuss dynamic programming problems for more advanced students and researchers. In order to make it easy for beginners to familiarize themselves with the field of computational economics, Chapter 1 gives an introduction to Fortran, the programming language used throughout the book. In Chapter 2 we discuss several numerical methods that are used frequently throughout this book. Since we do not assume that our readers have a lot of programming experience and only require standard mathematical knowledge, we only introduce the basic concepts and keep the theoretical parts quite condensed. A large set of numerical methods is provided through our toolbox. In the introductory chapters we discuss how to use this toolbox and the methods therein. Hence, after working through the first part of the book, you should have enough knowledge on programming and numerical tools to manage the remaining material in the follow-up parts. Part II comprises five chapters with easily accessible applications. Chapter 3 introduces the static general equilibrium model typically applied to issues of labour markets or international trade. In Chapter 4 we demonstrate different approaches to optimize a portfolio, to price options, and to manage credit and mortality risk. This leads in to Chapter 5, in which we discuss individual savings and investment decisions within the life-cycle framework with uncertain labour income, asset returns, and lifespan. We extend this partial equilibrium life-cycle model to its general equilibrium counterpart, the overlapping generations model, in Chapters 6 and 7. We use this model to investigate dynamic tax problems and optimal pension design in stationary and ageing societies. Finally, Part III of this book consists of four chapters with advanced applications. Chapter 8 introduces numerical solution methods for dynamic programming problems. In Chapter 9 we apply these methods to infinite horizon models of the macroeconomy and study growth, business cycles, and distributional issues. Chapter 10 focuses on advanced life-cycle labour-supply and investment problems using the dynamic programming approach. In the final Chapter 11 we extend the overlapping generations model to account for idiosyncratic earnings risk and study the design of optimal fiscal policies that balance tax distortions and public insurance provision.

While the different chapters somewhat build on one another regarding their economic themes, we tried to keep them as independent and self-contained as possible. Hence, the reader familiar with Fortran could skip Chapter 1 and directly start with Chapter 2, whereas the reader sufficiently familiar with numerical techniques could just take a quick look at Chapter 2 and directly start with studying Parts II and III. The chapters offering

applications follow the same logic and therefore could be also read independently. As already mentioned above, we encourage the reader to work through an extensive number of exercises in each chapter in order to deepen their understanding of the methods learned and to gain practical experience necessary to become a decent computational economist. Note that, especially when working on the first tasks, you will suffer a lot from programming errors and bugs. This is quite normal. Please don't be disappointed, if a program doesn't work at the first or second shot. Code can be written in several minutes. However, getting it to work can take hours or days.

This book is accompanied by a website

www.ce-fortran.com.

On this website we provide the source codes to any of the programs discussed in this book. You will also find a link to our toolbox, an instruction on how to install and use it as well as a description of the methods it provides. The website also contains up-to-date information on how to set up a Fortran compiler on your operating system and therefore to get started as quickly as possible on working through this book.

Last but not least, writing such a book is never possible without the help of others. First of all, we want to thank Oxford University Press for deciding to become our publisher. We are especially grateful to Katie Bishop, Susan Frampton, and Subramaniam Vengatakrishnan for their assistance in the production of this book. Over the years we have been working on this we have benefited from comments, suggestions, and endorsements from many colleagues, especially Ben Heijdra, Leonhard Knoll, Laurie Reijnders, Alexander Rothkopf, and Andras Simonovits. Thanks also to our former students Theresa Grimm, Maurice Hofmann, Sarah Lenz, Franziska Schlumprecht, Lorenz Schneck, Lukas Schwabe, Maximilian Stahl, and Patrick Wiesmann who provided excellent research assistance and worked on computer codes that provided the basis for certain chapters. Parts of this book were revised when Hans Fehr visited the ARC Centre of Excellence in Population Aging Research (CEPAR) at the University of New South Wales in 2016. He thanks the members of CEPAR for their hospitality during his stay.

Finally, we hope that you enjoy reading this book and that you have as much fun doing computational economics as we do.

Bonn and Würzburg in June 2017

■ CONTENTS

PART I AN INTRODUCTION TO FORTRAN 90 AND NUMERICAL METHODS

| | |
|--------------------------------------------------------|----|
| 1 Fortran 90: A simple programming language | 3 |
| 1.1 About Fortran in general | 3 |
| 1.1.1 The history of Fortran | 3 |
| 1.1.2 Why Fortran? | 4 |
| 1.1.3 The workings of high-level programming languages | 5 |
| 1.1.4 Fortran compilers for Windows, Mac, and Linux | 6 |
| 1.2 Imperative Fortran programs | 6 |
| 1.2.1 The general structure of Fortran programs | 7 |
| 1.2.2 The declaration of variables | 7 |
| 1.2.3 The basics of imperative programming | 8 |
| 1.2.4 Control flow statements | 11 |
| 1.2.5 The concept of arrays | 16 |
| 1.3 Subroutines and functions | 19 |
| 1.4 Modules and global variables | 23 |
| 1.4.1 Storing code in a module | 23 |
| 1.4.2 The concept of global variables | 25 |
| 1.5 Installing the toolbox | 27 |
| 1.6 Plotting graphs with the toolbox and GNUPlot | 28 |
| 1.6.1 Two-dimensional plotting | 28 |
| 1.6.2 Three-dimensional plotting | 31 |
| 1.7 Further reading | 34 |
| 1.8 Exercises | 35 |
| 2 Numerical solution methods | 39 |
| 2.1 Matrices, vectors, and linear equation systems | 39 |
| 2.1.1 Matrices and vectors in Fortran | 39 |
| 2.1.2 Solving linear equation systems | 40 |
| 2.2 Nonlinear equations and equation systems | 47 |
| 2.2.1 Bisection search in one dimension | 48 |
| 2.2.2 Newton's method in one dimension | 51 |
| 2.2.3 Fixed-point iteration methods | 54 |
| 2.2.4 Multidimensional nonlinear equation systems | 56 |
| 2.3 Function minimization | 60 |
| 2.3.1 The Golden-Search method | 61 |
| 2.3.2 Brent's and Powell's algorithms | 63 |
| 2.3.3 The problem of local and global minima | 67 |

x CONTENTS

| | |
|--------------------------------------------------------------|-----|
| 2.4 Numerical integration | 68 |
| 2.4.1 Summed Newton-Cotes methods | 69 |
| 2.4.2 Gaussian quadrature | 72 |
| 2.5 Random variables, distributions, and simulation | 77 |
| 2.5.1 Random variables and their distribution | 77 |
| 2.5.2 Simulating realizations of random variables | 81 |
| 2.6 Function approximation and interpolation | 85 |
| 2.6.1 Polynominal interpolation | 88 |
| 2.6.2 Piecewise polynomial interpolation | 91 |
| 2.6.3 A two-dimensional interpolation example | 95 |
| 2.7 Linear programming | 100 |
| 2.7.1 Graphical solution to linear programs in standard form | 102 |
| 2.7.2 The simplex algorithm | 103 |
| 2.8 Further reading | 105 |
| 2.9 Exercises | 106 |

PART II COMPUTATIONAL ECONOMICS FOR BEGINNERS

| | |
|--------------------------------------------------------|-----|
| 3 The static general equilibrium model | 113 |
| 3.1 The basic economy model | 113 |
| 3.1.1 The command optimum | 113 |
| 3.1.2 The market solution | 115 |
| 3.1.3 Variable labour supply | 119 |
| 3.1.4 Public sector and tax incidence analysis | 120 |
| 3.2 Extensions of the basic model | 123 |
| 3.2.1 Imperfect labour markets and unemployment policy | 123 |
| 3.2.2 Intermediate goods in production | 126 |
| 3.2.3 Open economies and international trade | 130 |
| 3.3 Further reading | 134 |
| 3.4 Exercises | 134 |
| 4 Topics in finance and risk management | 139 |
| 4.1 Mean-variance portfolio theory | 139 |
| 4.1.1 Portfolio choice with risky assets | 139 |
| 4.1.2 Introducing risk-free assets | 143 |
| 4.1.3 Short-selling constraints | 146 |
| 4.1.4 Monte Carlo minimization | 149 |
| 4.2 Option pricing theory | 151 |
| 4.2.1 The binomial approach by Cox-Ross-Rubinstein | 152 |
| 4.2.2 The Black-Scholes formula | 155 |
| 4.2.3 Numerical implementation of both approaches | 158 |
| 4.2.4 Option pricing with Monte Carlo simulation | 161 |

| | |
|----------------------------------------------------------|------------|
| 4.3 Managing credit risk with corporate bonds | 164 |
| 4.3.1 Modelling credit risk with a single corporate bond | 164 |
| 4.3.2 Credit risk in a bond portfolio | 173 |
| 4.4 Mortality risk management | 184 |
| 4.4.1 Modelling longevity risk | 184 |
| 4.4.2 Pricing and risk analysis of insurance products | 189 |
| 4.4.3 Optimization of a mortality portfolio | 196 |
| 4.5 Appendix | 198 |
| 4.6 Further reading | 200 |
| 4.7 Exercises | 201 |
| 5 The life-cycle model and intertemporal choice | 205 |
| 5.1 Why do people save? | 205 |
| 5.1.1 Optimal savings in a certain world | 205 |
| 5.1.2 Uncertain labour income and precautionary savings | 207 |
| 5.1.3 Uncertain capital and labour income | 212 |
| 5.2 Where do people save and invest? | 214 |
| 5.2.1 Uncertain capital income and portfolio choice | 214 |
| 5.2.2 Uncertain lifespan and annuity choice | 218 |
| 5.3 Further reading | 221 |
| 5.4 Exercises | 222 |
| 6 The overlapping generations model | 225 |
| 6.1 General structure and long-run equilibrium | 225 |
| 6.1.1 Demographics, behaviour and markets | 225 |
| 6.1.2 Computation of the long-run equilibrium | 229 |
| 6.1.3 Long-run analysis of policy reforms | 232 |
| 6.2 Transitional dynamics and welfare analysis | 234 |
| 6.2.1 Computation of transitional dynamics | 235 |
| 6.2.2 Generational welfare and aggregate efficiency | 240 |
| 6.2.3 Comprehensive analysis of policy reforms | 245 |
| 6.3 Further reading | 250 |
| 6.4 Exercises | 250 |
| 7 Extending the OLG model | 253 |
| 7.1 Accounting for variable labour supply | 253 |
| 7.1.1 The household decision problem | 254 |
| 7.1.2 Functional forms and numerical implementation | 255 |
| 7.1.3 Simulation results and economic interpretations | 258 |
| 7.1.4 A note on labour-augmenting technological progress | 261 |
| 7.2 Human capital and the growth process | 263 |
| 7.2.1 Education investment and externalities | 264 |
| 7.2.2 Numerical implementation and simulation | 266 |

| | |
|-------------------------------------------------------|-----|
| 7.2.3 Human-capital spillovers and endogenous growth | 270 |
| 7.2.4 Numerical implementation and simulation | 271 |
| 7.3 Longevity risk and annuitization | 274 |
| 7.3.1 The households' problem without annuity markets | 274 |
| 7.3.2 Numerical implementation and simulation | 277 |
| 7.3.3 Introducing private annuity markets | 279 |
| 7.4 Further reading | 282 |
| 7.5 Exercises | 282 |

PART III ADVANCED COMPUTATIONAL ECONOMICS

| | |
|---------------------------------------------------------------|-----|
| 8 Introduction to dynamic programming | 289 |
| 8.1 Motivation: The cake-eating problem | 289 |
| 8.1.1 The all-in-one solution | 290 |
| 8.1.2 The dynamic programming approach | 291 |
| 8.1.3 An analytical solution | 295 |
| 8.2 Numerical solution by value function iteration | 298 |
| 8.2.1 Grid search | 301 |
| 8.2.2 Optimization and interpolation | 306 |
| 8.3 Numerical solution by policy function iteration | 313 |
| 8.3.1 Root-finding and interpolation | 314 |
| 8.3.2 The method of endogenous gridpoints | 316 |
| 8.4 Further reading | 320 |
| 8.5 Exercises | 321 |
| 9 Dynamic macro I: Infinite horizon models | 323 |
| 9.1 The basic neoclassical growth model | 323 |
| 9.1.1 The model economy | 324 |
| 9.1.2 Numerical implementation | 329 |
| 9.1.3 A model with a public sector | 334 |
| 9.2 The stochastic growth model | 341 |
| 9.2.1 Modelling aggregate uncertainty | 341 |
| 9.2.2 A numerical implementation using discretized shocks | 344 |
| 9.2.3 Simulating time paths | 350 |
| 9.2.4 Speeding up the computational process | 352 |
| 9.3 The real business-cycle model | 354 |
| 9.3.1 A dynamic program with endogenous labour supply | 354 |
| 9.3.2 Numerical implementation with policy function iteration | 356 |
| 9.3.3 Comparing model results to the data | 358 |
| 9.3.4 The welfare costs of business-cycle fluctuations | 363 |
| 9.3.5 Procyclical vs. constant government expenditure | 369 |

| | | |
|-----------|------------------------------------------------------|------------|
| 9.4 | The heterogeneous agent model | 374 |
| 9.4.1 | The basic setup | 374 |
| 9.4.2 | Solving for market-clearing prices | 377 |
| 9.4.3 | Determining household policy functions | 380 |
| 9.4.4 | Aggregation of individual decisions | 385 |
| 9.4.5 | Model parametrization and simulation | 390 |
| 9.4.6 | The optimum quantity of debt | 394 |
| 9.5 | Further reading | 401 |
| 9.6 | Exercises | 401 |
| 10 | Life-cycle choices and risk | 406 |
| 10.1 | Labour supply, savings, and risky earnings | 406 |
| 10.1.1 | The baseline model | 407 |
| 10.1.2 | The role of variable labour supply | 422 |
| 10.1.3 | Female labour-force participation | 429 |
| 10.2 | Portfolio choice and retirement savings | 444 |
| 10.2.1 | A model with stocks and bonds | 444 |
| 10.2.2 | The choice to buy annuities | 469 |
| 10.2.3 | Retirement savings in tax-favoured savings vehicles | 478 |
| 10.3 | Further reading | 492 |
| 10.4 | Exercises | 493 |
| 11 | Dynamic macro II: The stochastic OLG model | 505 |
| 11.1 | General structure and long-run equilibrium | 505 |
| 11.1.1 | Demographics, behaviour, and markets | 506 |
| 11.1.2 | Numerical implementation of steady-state equilibrium | 512 |
| 11.1.3 | Model parametrization and calibration | 516 |
| 11.1.4 | The initial equilibrium | 521 |
| 11.1.5 | Long-run analysis of policy reforms | 523 |
| 11.2 | Transitional dynamics and welfare analysis | 525 |
| 11.2.1 | Computation of transitional dynamics | 525 |
| 11.2.2 | Generational welfare and aggregate efficiency | 527 |
| 11.3 | Comprehensive analysis of policy reforms | 539 |
| 11.3.1 | The optimal size of the pension system | 539 |
| 11.3.2 | The optimal progressivity of the labour-income tax | 544 |
| 11.3.3 | Should capital income be taxed? | 550 |
| 11.4 | Further reading | 556 |
| 11.5 | Exercises | 558 |
| | BIBLIOGRAPHY | 561 |
| | INDEX | 567 |

Part I

An Introduction to

Fortran 90 and Numerical

Methods

1 Fortran 90: A simple programming language

Before diving into the art of solving economic problems on a computer, we want to give a short introduction into the syntax and semantics of Fortran 90. As describing all features of the Fortran language would probably fill some hundred pages, we concentrate on the basic features that will be needed to follow the rest of this textbook. Nevertheless, there are various Fortran tutorials on the Internet that can be used as complementary literature.

1.1 About Fortran in general

1.1.1 THE HISTORY OF FORTRAN

Fortran is pretty old; it is actually considered the first known higher programming language. Going back to a proposal made by John W. Backus, an IBM programmer, in 1953, the term Fortran is derived from *The IBM Formula Translation System*. Before the release of the first Fortran compiler in April 1957, people used to use assembly languages. The introduction of a higher programming language compiler tremendously reduced the number of code lines needed to write a program. Therefore, the first release of the Fortran programming language grew pretty fast in popularity. From 1957 on, several versions followed the initial Fortran version, namely FORTRAN II and FORTRAN III in 1958, and FORTRAN IV in 1961. In 1966, the *American Standards Association* (now known as the ANSI) approved a standardized *American Standard Fortran*. The programming language defined on this standard was called FORTRAN 66. Approving an updated standard in 1977, the ANSI paved the way for a new version of Fortran known as FORTRAN 77. This version became popular in computational economics during the late 80s and early 90s. More than 13 years later, the Fortran 90 standard was released by both the *International Organization for Standardization* (ISO) and ANSI consecutively. With Fortran 90, the *fixed format* standard was exchanged by a *free format* standard and, in addition, many new features like modules, recursive procedures, derived data types, and dynamic memory allocation made the language much more flexible. From Fortran 90 on, there has only been one major revision, in 2003, which introduced object-oriented programming features into the Fortran language. However, as object-oriented

programming will not be needed and Fortran 90 is by far the more popular language, we will focus on the 1990 version in this book.

1.1.2 WHY FORTRAN?

Fortran is an imperative, procedural, high-level programming language that is designed and optimized for numerical calculations. In detail this means that: (a) a Fortran program consists of statements that will be executed consecutively (i.e. a Fortran program starts with the first line and ends with the last); (b) Fortran code that will be used several times can be stored in functions and subroutines that can be called up from other program parts; and (c) Fortran abstracts from the details of a computer, i.e. having the right compiler, Fortran code could be run on any computer independent of its hardware configuration and operating system. Fortran can also be viewed as a *general purpose programming language*. A general purpose language is a programming language that is designed for software development in many different application domains. In contrast so-called *domain-specific programming languages* are constructed to basically serve one application domain, like e.g. Matlab for matrix operations and Mathematica for symbolic mathematics.

At this point our students usually ask ‘So, why Fortran? Why not a domain-specific language or something more “fashionable” like Java?’ Let us come up with some justifying points. First, domain-specific languages might be relatively efficient in the specific domain they were created for, however, they are slow in other areas. Matlab, for example, is very good working with matrices of a regular or sparse nature, however, if you have ever tried to run some do-loops in Matlab, you will know what we mean. Second, when it comes to the more ‘fashionable’ languages like Java or C++, one has to always keep in mind that those languages are usually object-oriented and abundant. They are very good for creating software with graphical user interfaces, but not for running numerical maths code quickly and efficiently. As computational economics usually consists of at least 80 per cent numerical methods, we need a programming language that is efficient in executing do-loops and if-statements, calling functions and subroutines, and in allowing permanent storage of general codes that are frequently used. Fortran is a language that delivers all those features and much more; the only real alternatives we know would be C or a relatively new language called Julia. There is no real reason why one shouldn’t use C or Julia instead of Fortran, it’s just a matter of preference. Finally, Fortran was used a lot by engineers, numerical mathematicians, and computational economists during the 80s and 90s when a lot of optimizers and interpolation techniques were written. Fortunately, Fortran 90 is backward compatible with FORTRAN 77, i.e. all codes that are out there can easily be included in our Fortran programs.

All in all, it seems reasonable for us to have chosen a language matching all of the features necessary without any of the unnecessary extras that would make execution less efficient.

1.1.3 THE WORKINGS OF HIGH-LEVEL PROGRAMMING LANGUAGES

Low-level assembly languages that are basically used to program computers, microprocessors, etc. do not abstract from the computer's instruction set architecture. Therefore, the syntax of assembly languages depends on the specific physical or virtual computer used. In addition, assembly languages are quite complicated and even a small program requires many lines of computer code.

In opposition to that, high-level programming languages abstract from the system architecture and are therefore portable between different systems. As in all other languages, one has to learn the vocabulary, grammar, and punctuation, called the syntax, of that language. However, compared to real languages, the syntax of a high-level language can be learned within hours. In addition, the semantics, i.e. the meaning or interpretation of the different symbols used, usually follow principles that can be comprehended using pure logic.

As high-level languages abstract from the computer's instruction set architecture, one needs a *compiler*, sometimes also called *interpreter* that interprets the statements given in the program and translates them into a computer's native language, i.e. binary code. The way high-level programming works can be seen from Figure 1.1. First, the programmer has to write the source code in the respective high-level language and feed it to the compiler. The compiler then proceeds in two steps. In the *compilation* step it interprets the source code and creates a so-called object code file which consists of binary code. Binary code files usually have the suffix `.obj`. During this step, the compiler usually performs a syntax validation scan. If there are any syntactical errors, it produces error messages and stops the compilation process. In order to create an executable file, which has the suffix `.exe` in *Microsoft Windows*, the compiler now *links* the object code with all other libraries that are needed to run the program, e.g. external numerical routines. The resulting execution can now be sent to the operating system which advises the hardware to do exactly what the programmer wants.

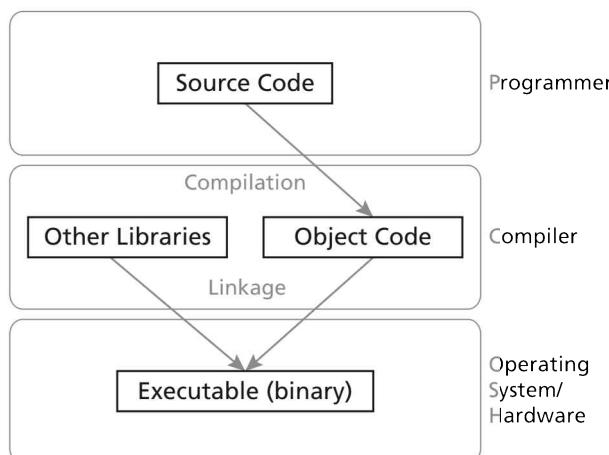


Figure 1.1 The way high-level compilers work

Program 1.1 Hello World

```
program HelloWorld
    write(*,*)'Hello World'
end program
```

1.1.4 FORTRAN COMPILERS FOR WINDOWS, MAC, AND LINUX

The website that accompanies this book

www.ce-fortran.com

suggests a freely available Fortran compiler that you can use for running the programs in this book. Just click on the menu item “Compiler” and choose your operating system. The website will guide you through the installation process. Finally, you will be advised how to run a program with your compiler. We therefore use the simplest programming example given in any computational textbook, namely the *Hello World* program shown in Program 1.1. A Fortran program always begins with the word **program** indicating that the program starts here. After that, one has to declare the program’s name, which must follow the Fortran naming conventions, i.e.

1. the name can’t have more than 31 characters
2. it must start with a letter
3. the other characters may be of any kind including symbols
4. capital letters may be used but the compiler is case insensitive
5. the name must not be a valid Fortran command.

Finally, the program ends with **end program**. In between these two statements, we can write the general program code which will be executed in an imperative way, i.e. the program starts with the first line and ends with the last. In the case of Program 1.1 there is only one statement that makes the program write Hello World in an unformatted way to the console.

1.2 Imperative Fortran programs

By knowing the basics of running a program in Fortran we can now proceed by writing the first imperative Fortran code. In order to describe the general structure of Fortran programs and give an overview of the basic features, we will restrict ourselves to imperative Fortran programs and leave the procedural component (i.e. subroutines and functions) to Section 1.3.

Program 1.2 Hello

```

program Hello

    ! declaration of variables
    implicit none
    character(len=50) :: input

    ! executable code
    write(*,*)'Please type your name:'
    read(*,*)input
    write(*,*)'Hello ',input

end program

```

1.2.1 THE GENERAL STRUCTURE OF FORTRAN PROGRAMS

In general, a Fortran program consists of a declarative part in which all variables needed for the execution are declared¹ and an executable part which gives instructions on what to do with all those variables. The executable part therefore consists of several *statements*, where each statement is usually given on one line. This is what the concept of imperative programming is basically about. The structure can easily be seen from Program 1.2, the interpretation of which is straightforward. Note, that after having written the first piece of executable code, we cannot declare a variable anymore. The statements in italic font in Program 1.2 starting with an exclamation point are comments. Those can be used to make it easier for the user to read the program, however, the compiler completely ignores them.

1.2.2 THE DECLARATION OF VARIABLES

Variables are used to store data during the execution of the program. Table 1.1 summarizes the five data types Fortran knows. Declaring the type of variable at the beginning of a program is not compulsory per se. Nevertheless, Fortran implicitly declares all variables that are used in the executable code as `integer`. This can cause severe problems when we run our program and forget to declare one variable that, for example, should be of `real*8` type. To prevent Fortran from implicit variable declaration by making declaration statements compulsory for any variable, we suggest always using the statement `implicit none` at the beginning of the declarative program part, see Program 1.2. Having specified `implicit none`, the compiler will tell us which variables are not yet declared and show an error message during the compilation step.² Program 1.3 shows some examples of variable declarations. The interpretation of the first declaration

¹ Technically speaking, the declaration of a variable induces the compiler to reserve some space in the memory that can be used to store data.

² Verify this by running Program 1.2 and deleting the line where `input` is declared. In the console window an error message will now appear when you try to execute the program.

Table 1.1 Different variable types in Fortran

| Type | Explanation |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| logical | can only take the values <code>.true.</code> or <code>.false.</code> |
| integer | can store integer data in between $-2^{31} + 1$ and $2^{31} - 1$ (4-byte integer data). |
| real*8 | stores real data according to the 8-byte data standard, i.e. in between $\sim -10^{308}$ and $\sim 10^{308}$. We suggest declaring all variables as real*8 , not real , as the former usually produces more accurate results. |
| character | stores character data. If not stated otherwise, such a variable can just store one character. The declaration statement character(len=n) produces a variable that can store a whole string of characters of length n, where n has to be a positive integer number. Strings of characters will in the following just be called string. |
| complex | are used to store complex numbers. However, this will not be important in this book. |

Program 1.3 Variable declarations

```
program VarDec
    ! declaration of variables
    implicit none
    logical :: logic
    integer :: a, b
    real*8 :: x, y1
    character :: one_char
    character(len=20) :: long_char

    real*8, parameter :: pi = 3.14d0
    integer, parameter :: n = 56

end program
```

statements should be clear from the above explanations. In addition to specifying the type of variable, we can also make it a **parameter** like in the last two statements of the program. This basically tells the compiler that the value of a variable should be fixed at a certain level and not be changed anymore. Specifying a parameter, we immediately have to declare the variables value, e.g. `pi = 3.14d0`.³

1.2.3 THE BASICS OF IMPERATIVE PROGRAMMING

After having declared the necessary variables, the first thing we would like to do is give values to those variables deemed necessary and then display these values on the console.

Reading and writing One way of assigning values to variables is making the user of the program type a value to the console. Program 1.4 explains how to do that. In this program,

³ The d0 after 3.14 tells the compiler that we are using double-precision variables. Always use d0 when declaring the value of **real*8** variables. Try to find out what happens if you don't use it.

Program 1.4 Reading and writing

```

program ReadWrite
    ! declaration of variables
    implicit none
    integer :: a
    real*8 :: x

    ! executable code
    write(*,*)"Type an integer number:"
    read(*,*)a

    write(*,*)"Type a real number:"
    read(*,*)x

    write(*,*)a, x
end program

```

we do the following: we first declare an integer variable `a` and a real variable `x`. This is our declarative program part. In the executable part, we first ask the user to type an integer number. The command `write(*,*)` makes Fortran write something to the console without having a specific format, where the phrase 'Type an integer number:' in apostrophes declares a text that should be displayed. The `read(*,*)a` statement induces the compiler to read a number from the console and assign it to the variable `a`. The first of the two stars in parentheses therefore tells the compiler from which location it should read. A * denotes the console as reading location. Reading from a file is also possible, however that will be explained later. The second star defines the format in which the number or text will be given. On the console, we always use *, which means there is no specific format. The compiler will then automatically check whether the number typed by the user is in the range of the variable we want to assign a value to, e.g. `a`.⁴ The same explanation applies to the statements concerning the real variable. Finally, we write the values of `a` and `x` to the console. Note that we can write several variables at once by just separating them with a comma. The same applies to the `read` statement.

Formatters In order to display variables in a formatted way, we use formatters. An example of how to use formatters is given in Program 1.5. This program does basically the same thing as Program 1.4, however it also prints the variables `a` and `x` in a formatted way on the console. The formatter, which is shown in apostrophes and parentheses, replaces the second star in the `write` statement, and states that we want to write an integer of maximum length three digits, two blank spaces and a real number with a maximum of 10 digits (including the decimal point), where six digits are reserved for the decimal places. Run the program and verify that this is true. There is a formatter

⁴ Verify this by running Program 1.4 and typing 12.5 when you are asked to type an integer number. The compiler will then throw up an error message.

Program 1.5 Formatters

```
program Formatter
    ! declaration of variables
    implicit none
    integer :: a
    real*8 :: x

    ! executable code
    write(*,*)'Type an integer number:'
    read(*,*)a

    write(*,*)'Type a real number:'
    read(*,*)x

    write(*,'(i3, 2x, f10.6)')a, x

end program
```

Table 1.2 Formatters for different variable types

| Formatter | Explanation |
|--------------|----------------------------------------------------------------------------------------------------------|
| i2 | value of a logical (T or F) with a total width of 2 |
| i4 | integer of a maximum of 4 digits |
| f12.4 | real of a maximum of 12 digits (including the decimal point) 4 digits are reserved for decimal places |
| a | string of arbitrary length |
| x | a blank space |

for each type of variable usually consisting of a letter indicating the type of variable that should be written and a number that defines the width of the output. For real variables, there is a second number specifying the number of decimal places. Table 1.2 summarizes common formatters. Putting a number in front of a formatter means that we want the compiler to write the same type of data several times. Multiplying a whole set of formatters can be realized by using brackets. Consider for example the following statement:

```
write(*,'(a,2x,2(f4.2,2x),a,i2)')'hello',123.456,1.544,'heLLO',12
```

The formatter tells the compiler to first write a string, then two blank spaces, twice a real number of a total of four digits with two decimal places followed by two blank spaces, again a string, and finally an integer with two digits. Note that there is one problem that arises with this statement. If we take a look at the data that should be written, not given through variables however directly typed into the written statement, we see that the first real number will be too big for its formatter. Fortran will therefore just display four star symbols indicating this overflow. Hence, the console output will be

```
hello **** 1.54 hELLO12.
```

You can verify this by writing a program that just consists of the statement above. Then change the formatting code and verify the results.

Program 1.6 Value assignment through arithmetics

```

program Arithmetics
    ! declaration of variables
    implicit none
    real*8 :: a, b, c, d

    ! executable code
    a = 6d0
    b = 2.5d0
    c = exp(b) + a**2*sqrt(b)
    d = max(a,b)*sign(b, a)/mod(9d0,5d0) + abs(c)

    write(*, '(4f10.4)') a,b,c,d

end program

```

Value assignments and calculations We can assign values to variables by stating `<variable> = <value>` as shown above. However, `<value>` does not necessarily have to be typed directly as a number, but can also be the result of some arithmetic operations. Program 1.6 illustrates how to use arithmetic to define variable values. Beneath the standard math operators `+`, `-`, `*`, `/`, and `**` (meaning to the power of), Fortran comes with several intrinsic functions like `exp`, `log`, etc. a summary of which can be found on our website. In Program 1.6, we first define four variables `a`, `b`, `c`, and `d` as of type `real*8`. In the first two lines, we assign the values 6 and 2.5 to variables `a` and `b`, respectively. Up to that point, `c` and `d` are still undefined. We then calculate:

$$c = \exp(b) + a^2 \cdot \sqrt{b} \quad \text{and} \quad d = \max(a, b) \cdot |b| \cdot \operatorname{sgn}(a)/(9 \bmod 5) + |c|.$$

Note that Fortran uses the standard PEMDAS precedence rule. In the last line of executable code we then let the compiler print the values of `a`, `b`, `c`, and `d` on the console, where every variable should have four decimal places. Run the program and compare the results to the ones you obtain by calculating the values of `c` and `d` by hand.

1.2.4 CONTROL FLOW STATEMENTS

In Section 1.2.3 we showed how to declare variables and write simple imperative codes in order to assign values. However, writing line after line of value assignments isn't the only thing we can do in Fortran. *Control flow statements* allow for the execution of conditional statements, i.e. the statement will only be executed, if a certain condition is true, and repeated statements, i.e. the same statement will be executed several times. The concept of control flow statements can easily be represented in a *control flow diagram*. Figure 1.2 shows such a diagram for a simple program. Its interpretation is quite intuitive. The program starts where the large black dot is: We first should read an integer number from the console. Then, we should check whether $i > 0$. There now are two conditional statements. If $i \leq 0$, we write an error message that says "i should be greater 0" and the

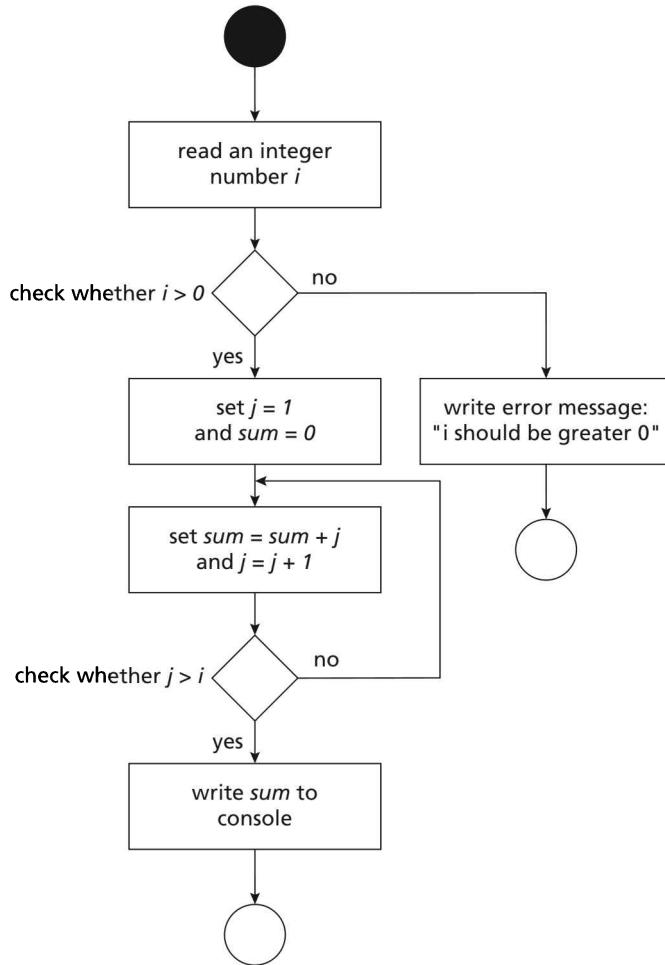


Figure 1.2 A control flow diagram

program stops afterwards, indicated by the circle. If $i > 0$ there is a series of conditional statements which basically sum up all integer numbers from 1 to i . The three statements after the first condition form what we call a *do-loop*. Using a do-loop, we can repeat the same statements, namely `sum=sum+j` and `j=j+1` several times.

If-statements and logical expressions An if-statement is needed to check whether some condition is true or false. The general syntax of an if-statement is

```

if (<condition1>) then
  <executable statements>
elseif (<condition2>) then
  <executable statements>
  [.....]
else
  <executable statements>
endif
  
```

The word **if** indicates that we want to check a certain condition. The condition has to be given in parentheses followed by the word **then**. The compiler will now check whether the condition holds and executes the code that stands within the **then** and the next control flow statement. This next statement can be of three different types. If an if-statement is followed by an **elseif** and <condition1> is false, the compiler will automatically check whether <condition2> is true. You can line up **elseif** as much as you want. The compiler will then check in hierarchical order, i.e. if the first condition is true, the statements within the first **if** and the first **elseif** will be executed and the compiler jumps to the point where **endif** ends the whole control flow structure. If the first condition is false and the second condition holds, the second set of executable statements will be executed and the compiler again jumps to the **endif** statement etc. An **else** without an **if** indicates which code fragment should be used if none of the above conditions holds, an **endif** tells the compiler where the whole if-construct ends. Of course, **elseif** and **else** statements are optional whereas the **endif** is compulsory. Program 1.7 illustrates the behaviour of if-statements. The program should be self-explanatory. Guess which condition will be true in this case. Run the program and verify it. Then change the value given to **a** and see what happens.

Conditions in the if-statement can be any kind of *logical expression*. A logical expression is an expression that is either true or false. Logical expressions can be created by using relational operators like ‘is equal to’, ‘is not equal to’, ‘is greater than’, etc. Table 1.3 gives an overview of relational operators in Fortran. Beneath the Fortran 90 relational operators, the table also shows the respective FORTRAN 77 operators which are still valid in Fortran 90. In the following, we will only use the Fortran 90 commands. Nevertheless, you can find the old operators in a lot of codes especially in older textbooks. An example for a logical expression is **a + b <= c**. Note that relational operators bind less than mathematical ones, i.e. the compiler will first evaluate all mathematical parts of logical

Program 1.7 If-statements

```
program IfStatements
    implicit none
    integer :: a

    ! initialize a
    a = 1

    ! check for the size of a
    if(a < 1)then
        write(*,'(a)')'condition 1 is true'
    elseif(a < 2)then
        write(*,'(a)')'condition 2 is true'
    elseif(a < 3)then
        write(*,'(a)')'condition 3 is true'
    else
        write(*,'(a)')'no condition is true'
    endif

end program
```

Table 1.3 Relational operators in Fortran

| Fortran 90 | FORTRAN 77 | Explanation |
|--------------------|-------------------|-----------------------------|
| <code>==</code> | <code>.EQ.</code> | is equal to |
| <code>/=</code> | <code>.NE.</code> | is not equal to |
| <code>></code> | <code>.GT.</code> | is strictly greater than |
| <code>>=</code> | <code>.GE.</code> | is greater than or equal to |
| <code><</code> | <code>.LT.</code> | is strictly lower than |
| <code><=</code> | <code>.LE.</code> | is lower than or equal to |

Table 1.4 Logical operators in Fortran

| Operator | Explanation |
|---------------------|------------------------------------------------|
| <code>.and.</code> | true if both expressions are true |
| <code>.or.</code> | true if at least one expression is true |
| <code>.eqv.</code> | true if both expressions have the same value |
| <code>.neqv.</code> | true if both expressions have different values |

Table 1.5 Results of logical connections

| a | b | <code>a.and.b</code> | <code>a.or.b</code> | <code>a.eqv.b</code> | <code>a.neqv.b</code> |
|----------------------|----------------------|----------------------|----------------------|----------------------|-----------------------|
| <code>.true.</code> | <code>.true.</code> | <code>.true.</code> | <code>.true.</code> | <code>.true.</code> | <code>.false.</code> |
| <code>.true.</code> | <code>.false.</code> | <code>.false.</code> | <code>.true.</code> | <code>.false.</code> | <code>.true.</code> |
| <code>.false.</code> | <code>.true.</code> | <code>.false.</code> | <code>.true.</code> | <code>.false.</code> | <code>.true.</code> |
| <code>.false.</code> | <code>.false.</code> | <code>.false.</code> | <code>.false.</code> | <code>.true.</code> | <code>.false.</code> |

statements and then the relational ones. Let, e.g. $a = 3$, $b = 5$, and $c = 6$. The logical expression will therefore be evaluated in the following way:

$$a + b \leq c \Rightarrow 3 + 5 \leq 6 \Rightarrow 8 \leq 6 \Rightarrow .false.$$

The value of the logical expression is therefore false for the above values of a , b , and c . Logical expressions can be linked by means of *logical operators*, a summary of which is given in Table 1.4. In this table, the operators are listed by importance for the compiler. Hence, `.and.` is the most binding operation among the logical expressions and will always be executed first. On the other hand, `.neqv.` is the less binding constraint. Note that you can make, e.g. an `.or.` more binding than an `.and.` by putting the respective expression in parentheses. The pattern therefore is the same as with adding and multiplying. The result of logical connections can be seen from Table 1.5. In addition to connecting logical expressions, we can also negate an expression by simply typing `.not.<expression>` with any kind of logical expression. Consider for example the expression

$$x \leq 2 .and. (y \leq 3 .or. z < 5) .or. .not. (y \leq 3 .and. z < 5)$$

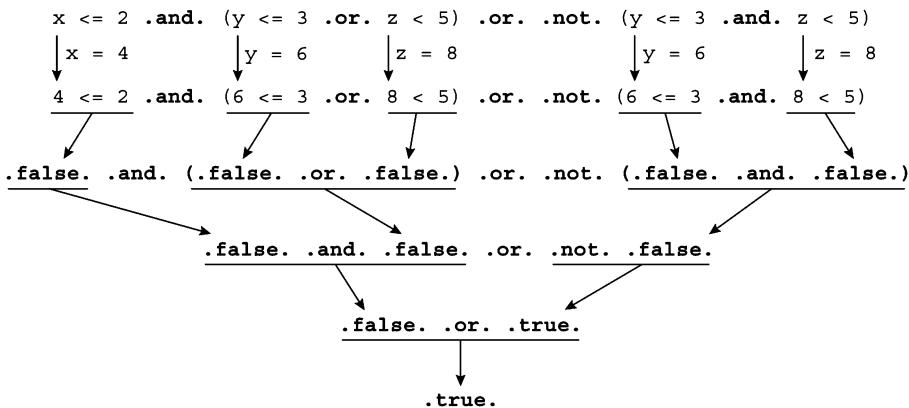


Figure 1.3 Evaluation process of logical expression

with $x = 4$, $y = 6$, and $z = 8$. The evaluation process of this expression can be seen from Figure 1.3. Verify this result by writing a program that defines the three variables x , y , and z and writes the result of the logical expression given above to the console.

Do-loops A do-loop is a construct that allows you to repeat a set of statements several times. An example of how to use do-loops is given in Program 1.8. This program shows three examples of do-loops. The first loop increments the integer variable j from 1 to 10 and writes its value to the console in every iteration. The general syntax for a do-loop that directly increments a variable is

```

do <variable> = <beginning>, <ending>, <stepsize>
    <executable statements>
enddo
  
```

The word **do** indicates that we want to start a loop. We then have to state which variable we want to increment or decrement during the loop. **<beginning>** and **<ending>** then tell the compiler what the starting value is and at which value to stop. **<stepsize>** is optional and specifies which step size to use. Finally, we write down the code that should be executed for every iteration of the loop. The **enddo** tells the compiler where the executable code of the loop ends. An example for a different step size is given in the second loop. We hereby let j take the values from 10 to 1 downwards. The last do-loop shows an alternative formulation. We hereby first initialize j at value 1. We then write a general loop without a specification statement for a variable. Within the loop we let the compiler write the value of j to the console and add 1 to j . Note that the command $j = j + 1$ simply takes the current value of j , adds one to this value and directly stores the result in j again. The if-statement in the loop checks, whether j has exceeded a value of 10. If this is the case, the exit command makes the compiler leave the current do-loop.

Program 1.8 Do-loops

```

program DoLoops

    implicit none
    integer :: j

    ! perform a do-loop for j = 1 to 10
    do j = 1, 10
        write(*,'(i3)')j
    enddo

    ! write a blank line
    write(*,*)

    ! perform a do-loop for j = 10 to 1
    do j = 10, 1, -1
        write(*,'(i3)')j
    enddo

    ! write a blank line
    write(*,*)

    ! alternative do-loop
    j = 1
    do
        write(*,'(i3)')j
        j = j + 1

        ! exit the do-loop
        if(j > 10)exit
    enddo

end program

```

The statement `if(j > 10)exit` is thereby just a shortcut for that can be used if there is just one executable statement associated with the if-statement.

```

if(j > 10)then
    exit
endif

```

Knowing the concepts of if-statements and do-loops, we can now implement the program specified by the control flow diagram in Figure 1.2. Program 1.9 shows how to do that.

1.2.5 THE CONCEPT OF ARRAYS

An array is a construct that allows storage of a whole set of data under one variable name. Arrays, like variables, are defined in the declarative part of a program. Typical examples of array declarations are

Program 1.9 Summing up in do-loops

```

program SummingUp

    implicit none
    integer :: i, j, sum

    ! read the variable
    write(*,'(a)')'Type an integer variable that is > 0'
    read(*,*)i

    ! check whether i > 0
    if(i > 0)then

        ! initialize sum at 0
        sum = 0

        ! do the summing up
        do j = 1, i
            sum = sum + j
        enddo

        ! write the result to the console
        write(*,'(a,i3,a,i10)')'The sum of 1 to ',i,' is ',sum

    else

        ! write the error message
        write(*,'(a)')'Error: i should be greater than 0'
    endif

end program

```

```

real*8 :: a(10)
integer :: b(12, 5, 16)
real*8 :: c(0:12, -3:5)

```

The numbers in parentheses, after the array's name, indicate its dimension. `a`, e.g., is an array of dimension 1 with 10 entries in the only dimension. `b` is of dimension 3 with 12, 5, and 16 elements in the three dimensions, respectively. Last but not least, `c` is an array of dimension 2 with 13 and 9 elements in the two directions. However, the index of dimension 1 starts at 0 and that of dimension 2 at -3 . The single elements of an array can be accessed in the executable code by just saying, e.g., `a(5)`, `b(1, 3, 16)`, or `c(0, -2)`. The statement `b(1, 3, 16)` then returns the value of the array at positions 1, 3, and 16 in the three dimensions, respectively. Note, that if not defined otherwise, an array's index always starts with 1. In the case of `c`, however, we made the array index start with 0 and go up to 12 in the first dimension, meaning a total size of 13.

An example of how to deal with arrays is given in Program 1.10. In the declarative part of the program, we first define two arrays of same size, the indices of which start at 0 and go up to 10, as well as an integer variable that serves as counter for our do-loops. In the first step of our executable program part we initialize the values of `x`. We do this by running a do-loop over `j` from the first to the last array element, where we let `x` be

Program 1.10 Handling arrays

```

program Arrays

    implicit none
    real*8 :: x(0:10), y(0:10)
    integer :: j

    ! initialize x and calculate y
    do j = 0, 10
        x(j) = 1d0/10d0*dble(j)
        y(j) = exp(x(j))
    enddo

    ! output table of values
    write(*,'(a)')           x          y'
    do j = 0, 10
        write(*,'(2f10.3)')x(j), y(j)
    enddo

end program

```

equidistant points on the interval $[0, 1]$.⁵ Having set x , we can go over to calculating y . We want y to be the exponential function evaluated at the different values of the array x . This again is done by means of do-looping. Finally, we let Fortran print the values of x and y to the screen.

Nevertheless, do-looping is not the only way to work with arrays. Let's consider the example given in Program 1.11. This example contains several new features. We start with introducing a parameter n that defines the size of any array in this program. Note that n must have the parameter property. If not, the declaration of arrays will not work. Next, we define three different arrays, two of dimension 1 and one of dimension 2. We initialize the array x as in Program 1.10. Next, we want the array y to have the entries of x and add 1 to each entry. We can do this again with a do-loop as seen in the program before, however there is a shortcut to that. If we use the notation $y(:) = x(:) + 1d0$, it has the same effect. This statement basically tells the compiler to take all entries of x , add 1 to each of them and store the results in y . The $(:)$ therefore indicates that we want to do something with each entry of an array. We can also perform an action with only part of an array. If we for example stated $y(1:6) = x(0:5) + 1d0$, the compiler would take the elements of x at the indices from 0 to 5, add 1 to each of them, and store them in y at the positions 1 to 6. Next we calculate the values of z by means of a separate do-loop for each of the dimensions of z . Last, we output a two-dimensional table of values for z . We therefore let the compiler print a headline that consists of the string ' $x/y |$ ' and all the values of y followed by a blank line. We then output one line for each set of numbers

⁵ In the initialization statement for x , the term **dble**(j) converts the integer value of j to a **real*8** value. This should always be done when using integers in calculating **real*8** values. See the exercise section for an explanation.

Program 1.11 Alternative array handling

```

program AlternativeArrays

    implicit none
    integer, parameter :: n = 8
    real*8 :: x(0:n), y(0:n), z(0:n, 0:n)
    integer :: j, k

    ! initialize x
    do j = 0, n
        x(j) = 1d0/dble(n)*dble(j)
    enddo

    ! give y the values of x plus 1
    y(:) = x(:) + 1d0

    ! calculate z
    do j = 0, n
        do k = 0, n
            z(j, k) = x(j)**2 + y(k)
        enddo
    enddo

    ! output table of values
    write(*,'(a,9f7.2)')'      x/y | ', y(:)
    write(*,'(a)')'           | '
    do j = 0, n
        write(*,'(f7.2,a,9f7.2)')x(j),' | ', z(j, :)
    enddo

end program

```

in z that has the same index in the first dimension. Unfortunately, we can't write n into the formatter, as the formatter is a string. Hence, we have to manually type the number 9 (which corresponds to the length of the array y), as creating a general formatter with parameter n would be far beyond the scope of this introduction.

1.3 Subroutines and functions

Subroutines and *functions* can be used to store codes that are frequently used within one program. While a subroutine just executes an imperative code, a function is a construct that, like a function in maths, receives some input value and calculates a return value.

Subroutines Program 1.12 demonstrates how to use subroutines in a Fortran program. Subroutines, as well as functions, are defined after the main program code. The part of the program which contains subroutines and functions is separated from the main code by means of the `contains` statement. We declare a subroutine by typing the keyword

Program 1.12 Subroutines

```

program Subroutines

    implicit none
    real*8 :: a, b, c, d

    a = 3d0
    b = 5d0

    ! call subroutine
    call addIt(a, b)

    ! redefine values
    c = 10d0
    d = 2d0

    ! call subroutine again
    call addIt(c, d)

! separates main program code from subroutine and functions
contains

    subroutine addIt(a, b)

        implicit none

        ! input arguments
        real*8, intent(in) :: a, b

        ! other variables
        real*8 :: c

        ! executable code
        c = a + b
        write(*, '(2(f8.2,a),f8.2)') a, ' + ', b, ' = ', c

    end subroutine

end program

```

subroutine and a name afterwards. The name, `addIt` in the case of Program 1.12, must follow the Fortran naming conventions as described in Section 1.1.4 and must neither correspond to the program name nor any name of a variable declared in the main program. After having specified the name, we can tell Fortran which communication variables the subroutine receives from the main program.⁶ In our case, we specified two input variables, `a` and `b`, the sum of which will be calculated and written to the console. Note that communication variables do not have to be simple input arguments per se, they can also be arrays or, as we will see later, functions or subroutines. If we pass on a variable from the main program and change its value within the subroutine, the change will also be adopted by the main program.⁷ The structure of a subroutine is now pretty much the

⁶ You do not need to specify any communication variable. Just type empty parentheses in this case.

⁷ We recommend always using communication variables to pass data on to a subroutine or function. Yet, it would generally be possible to use variables that are declared in the main program within a subroutine. Verify

same as that of a main program. We first have a declarative part in which we define all variables used in the subroutine, including communication variables. In addition, we can regulate whether our communication variables should be of input or output type. This is done via an `intent` statement. If a variable's intent is `in`, we are not allowed to change its value throughout the subroutine.⁸ If the intent is `out`, the variable must be given a value within the subroutine. Specifying `inout` has the same effect as not declaring any `intent`. After having declared all variables, the executable part of our subroutine specifies what to do with them. We can use any kind of executable statements we know from our main programs.

When it comes to using a subroutine in a main program, we use the keyword `call` followed by the name of the subroutine that should be called.⁹ In addition, we have to specify which variables of the main program to pass on to the subroutine, where the number and type of variables must be exactly the same as those specified in the subroutine's declaration. In our case, we use the subroutine with different variables twice.

Functions Functions are equally easy to use. Program 1.13 shows an example. Declaring a function is very similar to declaring a subroutine. However, in addition to specifying the type and intent of communications variables, we also have to state the function's return value. This is done using a type declaration with the function's name. The name will therefore serve as a regular variable throughout the executable part of the function. The value given to this variable finally is the return value of the function, which in our example corresponds to the sum of `a` and `b`. Having specified the function we now can use it in our main program. Since the function has a return value, we do not call it via a `call` statement like a subroutine, but instead assign the return value directly to a variable, `res` in our case. `res` now contains our function value, i.e. the sum of `a` and `b`. Finally, we write the result to the console.

Passing arrays to functions or subroutines Sometimes one would like to pass an array to a function or subroutine. This can be done in two ways, see Program 1.14. If we take a look at the function declaration, we see that communication variable `a` is specified like we would specify a regular array in a main program. The declaration statement therefore tells the compiler that it should expect an array of dimension 1 and length 2 to be passed to the function. In the case of `b`, we used the so-called assumed-shape statement.

this by deleting the input variables `a` and `b` from the subroutine and calling up the subroutine without input variables. How does the output change compared to Program 1.12? Note that when a variable is declared in a subroutine that has the same name as a variable in the main program, the subroutine will always use the 'local' variable, not the 'global' one from the main program.

⁸ Verify this by trying to change the value of `a` in the above subroutine. When compiling, there now should be an error in the output window saying that this is not allowed.

⁹ Subroutines do not necessarily have to be called up by a main program, but can also be used by other subroutines.

Program 1.13 Functions

```

program Functions

    implicit none
    real*8 :: a, b, res

    a = 3d0
    b = 5d0

    ! call function
    res = addIt(a,b)

    ! output
    write(*,'(2(f8.2,a),f8.2)')a,' + ',b,' = ',res

contains

    function addIt(a, b)

        implicit none

        ! input arguments
        real*8, intent(in) :: a, b

        ! function value
        real*8 :: addIt

        ! executable code
        addIt = a + b

    end function

end program

```

Typing `a:` instead of a valid integer number, we allow the length of `b` to be anything. The compiler therefore will just expect an array of dimension 1, but with an arbitrary length. However, we can refer to `b`'s length, which will be calculated on-the-fly during runtime, by typing `size(b)`.¹⁰ We consequently define our return value as an array of two dimensions, the first of which is of size 2 and the second of the same size as `b`. The executable code of our function should then be straightforward.

In the main program, we now specify three arrays. The two input arrays of dimension 1 and the array that should store the result of our function. This array must have dimension 2 with the size of `a` in dimension 1 and that of `b` in dimension 2. We then initialize the arrays `a` and `b`. There is a shortcut to do this by saying `a(:) = (/<values>/)`, where `<values>` is just a list of initialization values separated by commas. Finally, we can call our function, assign the return value to `res`, and write the result to the console.

¹⁰ If you declare a multidimensional array, you can refer to the length into any dimension by stating `size(b, <dim>)` where `<dim>` is just an integer number declaring the dimension.

Program 1.14 Passing arrays to functions

```

program ArrayFunc

    implicit none
    real*8 :: a(2), b(5), res(2,5)

    a(:) = (/3d0, 4d0/)
    b(:) = (/1d0, 2d0, 3d0, 4d0, 5d0/)

    ! call function
    res = addIt(a,b)

    ! output
    write(*,'(5f8.2/5f8.2)') res(1,:),res(2,:)

contains

    function addIt(a, b)

        implicit none

        ! input arguments
        real*8, intent(in) :: a(2), b(:)

        ! function value
        real*8 :: addIt(2, size(b))

        ! local variables
        integer :: j, k

        ! executable code
        do j = 1, 2
            do k = 1, size(b)
                addIt(j,k) = a(j) + b(k)
            enddo
        enddo

    end function

end program

```

1.4 Modules and global variables

A module is a construct that allows storage of frequently used codes and variables in a separate location. The codes and variables can then be used by different programs.

1.4.1 STORING CODE IN A MODULE

Module 1.15m gives an example of a simple module which contains a function to calculate the volume of a sphere.¹¹ The structure of a module is similar to that of a regular program,

¹¹ The respective program is called prog1_15m.f90 in our program database.

Module 1.15m Storing code in modules

```
module Volume

    implicit none

    ! module parameter
    real*8, parameter :: pi = 3.14159265358d0

    ! separates variable declarations from subroutine and functions
contains

        ! for calculating volume of a sphere
function vol(r)

            implicit none

            ! input and output variables

            real*8, intent(in) :: r
            real*8 :: vol

            ! calculation
            vol = 4d0/3d0*r**3*pi

        end function

end module
```

however, a module begins with the keyword **module** and ends with **end module**. The variable declaration part again starts with an **implicit none** statement followed by any variable declaration. In our case, we defined a parameter of type **real*8** that gives us the value of π . In contrast to a main program, a module does not contain any directly executable codes. Nevertheless, we can store functions and subroutines within a module. The statement **contains** again indicates that we want to start the part of the module where those are declared. Functions or subroutines are declared in the same way we have seen before, see Programs 1.12 and 1.13. In the case of Module 1.15m we declared a function **vol** that calculates the volume of a sphere with radius **r**. The radius is given as an input argument to the function via the **real*8** variable **r**.

We can now write a very simple program that uses the module **volume**. There are multiple ways of doing so. The easiest is to include the module code into the main program. This is done through an **include** statement, see Program 1.15. The **include** statement tells the compiler that we want to use the main program together with the module that is stored in the file **prog1_15m.f90**.¹² When we run the program, the compiler will automatically compile the module prior to compiling the program and therefore make it ready for usage. Including a module this way ensures that any change we

¹² We can also store several modules in the file **prog1_15m.f90**, see next example.

Program 1.15 Calling module code from Module 1.15m

```

include "prog01_15m.f90"

program Sphere

    ! import variables and subroutines from module Volume
    use Volume

    implicit none

    write(*, '(f12.4)') vol(1d0)

end program

```

make in the module file will immediately be recognized by the compiler.¹³ The **include** statement, however, only tells our compiler that it should compile the code we have stored within our module file. In the program `Sphere` itself we then still have to indicate that we want to use the module `Volume` by typing **use** `Volume` before any variable declaration statement. We can now access any declared variable, function, or subroutine of that module. Hence, we can write `vol(1d0)`, i.e. the volume of a sphere with radius 1, to the console.

1.4.2 THE CONCEPT OF GLOBAL VARIABLES

A global variable is a variable that is present in any part of a program, i.e. in the main program as well as any subroutine and function. Typical examples of global variables are model parameters. One way to realize this concept is to pass a variable we want to be global to a subroutine and function of our main program. However, if we have larger programs this can become messy. Therefore, we can use modules to make life easier.

Consider for example Module 1.16m, which actually consists of two separate modules. The first module `Globals` is pretty simple, as it contains no subroutines or functions. It just declares two variables, `beta` and `eta` denoting time preference and risk aversion in a simple two-period life-cycle model, where households' utility function is given by

$$u(c_1, c_2) = \frac{1}{1 - \eta} \cdot c_1^{1-\eta} + \frac{\beta}{1 - \eta} \cdot c_2^{1-\eta}. \quad (1.1)$$

This utility function is specified in the second module `UtilFunc`. The module only contains a function `utility` that takes as input variables values for c_1 and c_2 and just

¹³ An alternative way of handling the module file would be to precompile it using the *Compile* button in the *Build Toolbar*. However this way the compiler sometimes does not recognize changes that have been made within the module file.

Module 1.16m A module to store global variables

```

module Globals

    implicit none

    ! time preference
    real*8 :: beta

    ! risk aversion
    real*8 :: eta

end module

module UtilFunc

    implicit none

contains

    ! a utility function
    function utility(c1, c2)

        use Globals

        implicit none
        real*8, intent(in) :: c1, c2
        real*8 :: utility

        utility = 1d0/(1d0-eta)*c1** (1d0-eta) &
                  + beta/(1d0-eta)*c2** (1d0-eta)

    end function

end module

```

calculates the value of lifetime utility. Note that this function uses the module `Globals` and especially the parameters defined therein.¹⁴

Suppose now, we want to write a program with which we can calculate the utility function for different combinations of c_1 and c_2 that satisfy the budget constraint

$$c_1 + c_2 = 1,$$

where we assumed the present value of income and the interest rate to be 1 and 0, respectively. An example of such a program could be Program 1.16. Again we have to tell the compiler in which file we have stored our modules by means of the `include` statement. In the main program, before declaring any program specific variables, we tell the program to use the `Globals` module that contains our global variables `beta` and `eta` as well as the `UtilFunc` module that contains the utility function. We then define some

¹⁴ It is important that the module `Globals` is defined prior to the module `UtilFunc`, since the latter uses the former. If the modules were arranged in reverse order, an error statement would tell us that `UtilFunc` can not find the module `Globals`.

Program 1.16 A program that uses global variables

```

include "prog_01_16m.f90"

program CalcUtil

    use Globals
    use UtilFunc

    implicit none
    real*8 :: c1, c2, util
    integer :: j

    ! initialize parameters
    beta = 0.9d0
    eta = 2d0

    ! calculate utility for different consumption pairs
    ! between 0.3 and 0.7
    do j = 0, 20
        c1 = 0.3d0 + (0.7d0-0.3d0)/20*dble(j)
        c2 = 1d0-c1
        util = utility(c1, c2)
        write(*,'(3f10.4)')c1, c2, util
    enddo

end program

```

variables that are used later on. Afterwards, we can initialize our global model parameters `beta` and `eta`. Those parameter values will consequently be stored in the module `Globals` and then be used by the function `utility`. Finally, we calculate the utility function resulting from different combinations of `c1` and `c2` that satisfy the household budget constraint by means of the function `utility`.

1.5 Installing the toolbox

This textbook comes with a toolbox, i.e. a collection of subroutines and functions, that will frequently be used in this book mostly to solve numerical problems. These subroutines and functions will be discussed in detail in the next chapters. The toolbox and the necessary programs should automatically have been installed together with the compiler. However, you can also download and install the toolbox directly from our website

www.ce-fortran.com

Just click on the menu item *Toolbox* and the website will guide you through the installation process. This menu item is also the place to look for updates. The toolbox requires the installation of the program *GNUPlot*. *GNUPlot* is a drawing program that can be used to draw graphs on Windows, Mac, and Linux machines. The toolbox contains some

interface routines that help you with that. Two things to note on the toolbox: First, we provide the toolbox in plain text format. So if you want to take a deeper look at the subroutines provided therein feel free to do so. Second, when you install the toolbox it will be compiled with the compiler you installed on your computer. By doing so it is ensured that the compiler will always be able to access the toolbox routines without problems.

1.6 Plotting graphs with the toolbox and GNUPlot

We use program GNUPlot for plotting graphs in Fortran as Fortran itself does not provide any plotting routines. Plotting graphs is, however, very simple, as the toolbox provides respective interface routines that transfer your data into GNUPlot graphs. In this section, we demonstrate how to plot both two- and three-dimensional data.

1.6.1 TWO-DIMENSIONAL PLOTTING

Program 1.17 shows how to plot graphs with GNUPlot using the toolbox. Of course, GNUPlot has to be installed on your computer. In order to use any subroutine or function located in the toolbox, we have to include the toolbox by stating `use toolbox`. We then declare two arrays, one for our x-axis data and one for the y-axis data, respectively. We initialize x at equidistant points on the interval [0, 1] via the first do-loop. Afterwards, we calculate the y-data we want to plot, which in the first case is equal to x^2 . Note that we used the shortcut method discussed in Section 1.2.5. In order to tell the toolbox to include our x and y data in a plot, we call the subroutine `plot`. The minimum requirement for using `plot` is to provide two equal size arrays of type `real*8`, the first containing the x-axis data and the second the y-axis data. The subroutine `plot` can be called with many more arguments which we will discuss later. Having used the subroutine `plot`, our x and y data is included in the plot we want to make. In order to actually draw the data with GNUPlot we have to call the subroutine `execplot`. When we call this subroutine, a window will appear on the screen displaying our x and y data as in Figure 1.4. The execution of our main program will pause. In order to close the plot window and let the program continue, you have to click on the console window and press `RETURN`.

We can also plot several graphs into one window. This is done in the second part of the program. We therefore first calculate the y data as \sqrt{x} and write it to our next plot by calling the subroutine `plot`. This time we include one of the *optional arguments* of the subroutine `plot`. These optional arguments are used by stating their name, in our case `legend`, and by specifying a value, in our case the name of the graph that should appear in the legend of the plot, namely '`square root`'. All the optional arguments of `plot` and

Program 1.17 Plotting graphs with the toolbox

```

program Plotgraphs

use toolbox

implicit none
real*8 :: x(0:100), y(0:100)
integer :: i1

! Initialize x values
do i1 = 0, 100
    x(i1) = 1d0/100d0*dble(i1)
enddo

! Calculate plot data
y = x**2
call plot(x, y)

! execute plot program
call execplot()

! Calculate data for roots
y = x** (1d0/2d0)

! you can specify a legend entry in the plot as follows
call plot(x, y, legend='square root')

! the same for a cubic root
y = x** (1d0/3d0)
call plot(x, y, legend='cubic root')

! execute plot program and give the plot a title
call execplot(title='Roots')

! plot has many more options that are specified here
y = x** (1d0/2d0)
call plot(x, y, color='green', linewidth=3d0, marker=2, &
          markersize=0.7d0, noline=.false., legend='square root')

y = x** (1d0/3d0)
call plot(x, y, color=#5519D6, marker=5, markersize=1.2d0, &
          noline=.true., legend='cubic root')

call execplot(xlim=(/0d0, 1.1d0/), xticks=0.1d0, &
              xlabel='x-Axis', ylim=(/0d0, 1.4d0/), yticks=0.2d0, &
              ylabel='y-Axis', title='Roots', legend='rs', &
              filename='testplot', filetype='eps', output='testdata')

end program

```

`execplot` are discussed in detail below. Having included the `x` and `y` data for the square root we can proceed and calculate the data for a cubic root. We can include this data into the graph by just calling the `plot` statement again. In total the toolbox would allow you to place up to 1,000 graphs in one plot. Having added all the relevant plot data to our graph, we can again make `GNUPLOT` draw our required lines by calling `execplot`. This time we specify the optional argument `title` which will define the title of our plot. The resulting plot will look like the one in Figure 1.5.

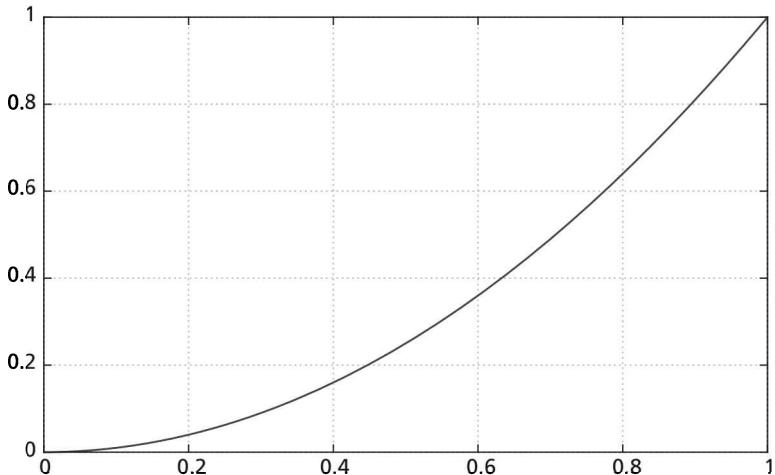


Figure 1.4 Plotting a graph with GNUPlot

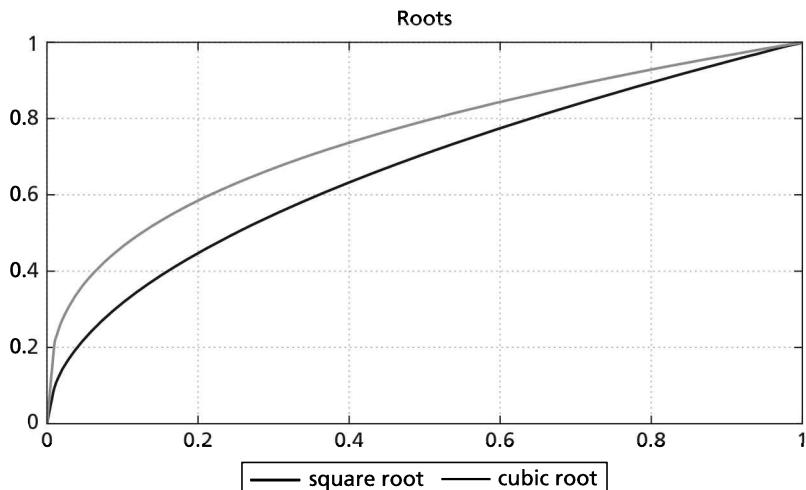


Figure 1.5 Plotting a graph with GNUPlot (2)

Note: The top line drawn in the graph is the cubic root, the bottom line is the square root

The last part of the program repeats the plot of square and cubit root, but this time it applies all optional arguments to the subroutines `plot` and `execplot`. In the subroutine `plot` you can specify which colour should be used for the graph, how thick the line should be, whether the data points you supplied should be marked with a marker and how this marked should look like. The `noline` statement tells GNUPlot whether it should use a connection line between the supplied data points or only show the markers. Finally the `legend` statement specifies the legend entry for the graph. The subroutine `execplot`

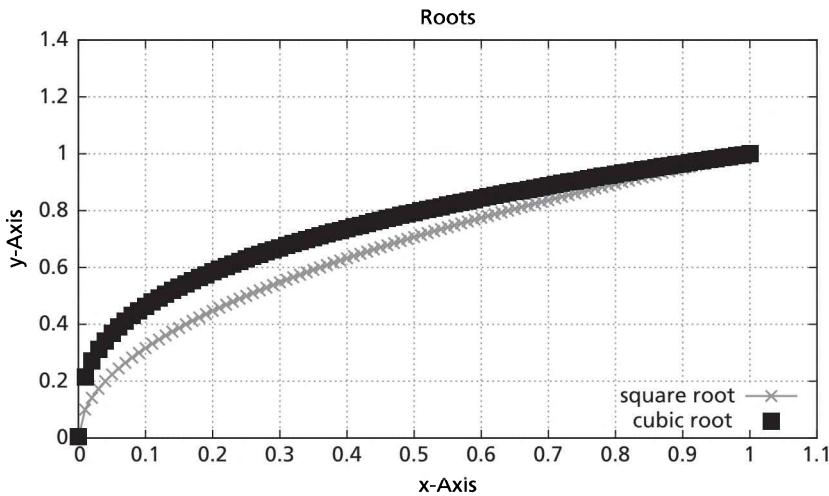


Figure 1.6 Plotting a graph with GNUPlot (3)

receives a whole lot of arguments related to the format of the plot. In addition, we can make GNUPlot export the graph in .eps or .png format. On our website we provide detailed information about all the optional arguments of `plot` and `execplot`, what kinds of values need to be supplied and what they actually do. We also show how to use GNUPlot to create histograms. The graph resulting from the last statement is shown in Figure 1.6.

1.6.2 THREE-DIMENSIONAL PLOTTING

The conventions for plotting three-dimensional data are very similar to those of two-dimensional plotting. However, in the 3D case we allow only one curve or surface per graph. Hence, we do not have to separately call a plotting routine and the subroutine that execute GNUPlot. Instead, all of this is packed together in the subroutine `plot3d`. Program 1.18 gives an example of how three-dimensional plotting works in practice with our toolbox.

There are two ways to represent three-dimensional data, depending on how the plot should look. In the first part of the program, we want to plot the function

$$z = f(x, y) = \sin(x) \cdot \cos(y)$$

on the interval $(x, y) \in [-5, 5] \times [-3, 3]$. In order to create a plot of this function, we generate some plotting data like in the previous Program 1.17. For the plotting to work correctly, we have to represent the function on a rectilinear grid. This means that we generate x-axis and y-axis data separately and evaluate the function $f(x, y)$ at each

Program 1.18 Plotting graphs with the toolbox

```

program Plotgraphs3D
[.....]
! Initialize x values
do i1 = 0, nplot1
    x(i1) = -5d0 + 10d0/dble(nplot1)*dble(i1)
enddo

! initialize y values
do i2 = 0, nplot2
    y(i2) = -3d0 + 6d0/dble(nplot2)*dble(i2)
enddo

! get z values
do i1 = 0, nplot1
    do i2 = 0, nplot2
        z(i1, i2) = sin(x(i1))*cos(y(i2))
    enddo
enddo

! call 3D plotting routine
call plot3d(x, y, z)

! call 3D plotting routine with complete configuration
call plot3d(x, y, z, color='black', linewidth=0.5d0, marker=2, &
            markersize=0.3d0, noline=.false., &
            xlim=(-6d0, 6d0/), xticks=1.0d0, xlabel='x-Axis', &
            ylim=(-3d0, 3d0/), yticks=0.5d0, ylabel='y-Axis', &
            zlim=(-1d0, 1d0/), zticks=0.2d0, zlabel='z-Axis', &
            zlevel=0d0, surf=.true., surf_color=2, &
            transparent=.true., view=(/70d0, 50d0/), &
            title='sin(x)*cos(y)', filename='testplot', &
            filetype='eps', output='testdata')

! initialize spiral data
do i1 = 0, nplot1
    c(i1) = 20d0*dble(i1)/dble(nplot1)
    a(i1) = sin(c(i1))
    b(i1) = cos(c(i1))
enddo

! plot spiral
call plot3d(a, b, c, linewidth=2d0)

end program

```

combination of data points x and y . We chose to partition the intervals $[-5, 5]$ and $[-3, 3]$ into 81 and 51 equidistant points, respectively. The x and y data are stored in arrays $x(0:nplot1)$ and $y(0:nplot2)$.¹⁵ From evaluating the function at each data-point combination, we get a two-dimensional array $z(0:nplot1, 0:nplot2)$ that contains the function values of f . We can now simply plot our function by feeding our data into the subroutine `plot3d`. This subroutine again calls `GNUPLOT` and produces a surface plot of

¹⁵ Note that we are not required to chose an equidistant partition. Instead, any arbitrary partition of the intervals would be allowed.

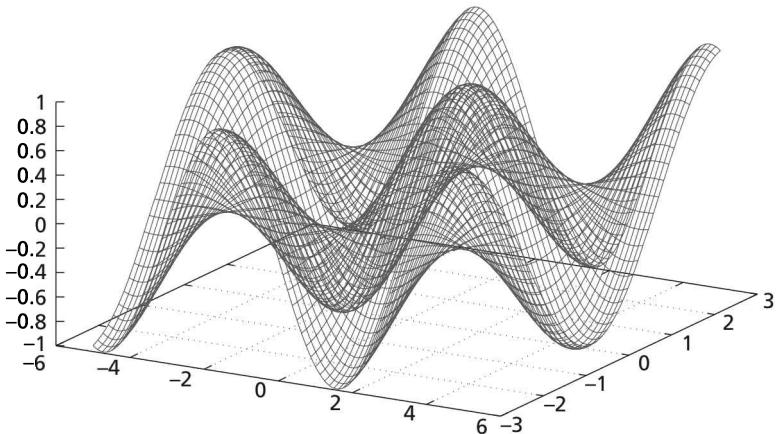


Figure 1.7 Plotting a surface with GNUPLOT

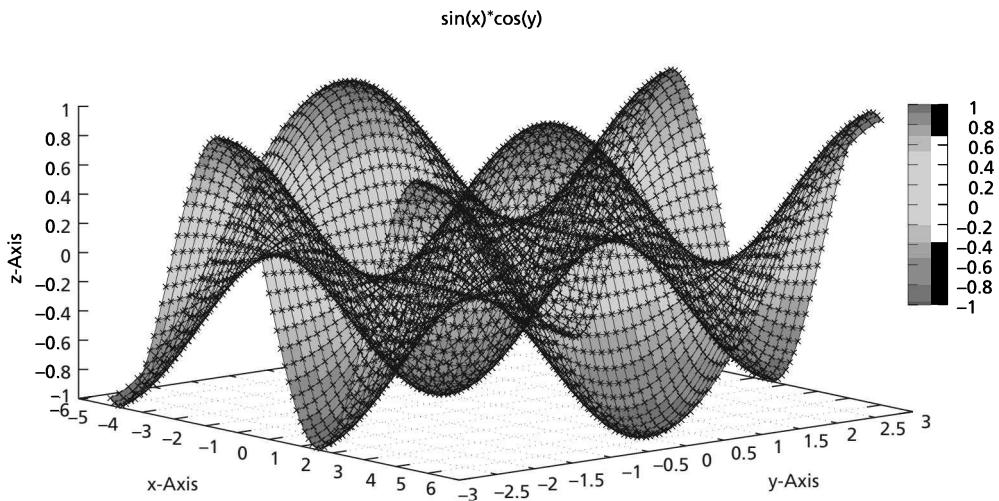


Figure 1.8 Plotting a surface with GNUPLOT (2)

the function as can be seen in Figure 1.7. The next statement calls the subroutine `plot3d` again, but this time we provide all possible input arguments by which we can alter the appearance of the plot, see again the website for details on all these input arguments. The corresponding output is shown in Figure 1.8.

Finally, GNUPLOT can not only plot the surfaces of functions, but is also able to represent simple lines in the three-dimensional space. One example of such a line is a spiral. A spiral can be represented by the data points $(a, b, c) = (\sin(c), \cos(c), c)$. We generate these data points in the last part of Program 1.18. The corresponding plot data are three arrays, `a`, `b`, and `c`, of the same length. When we feed these arrays into the subroutine `plot3d`, it generates a three-dimensional spiral as can be seen from Figure 1.9.

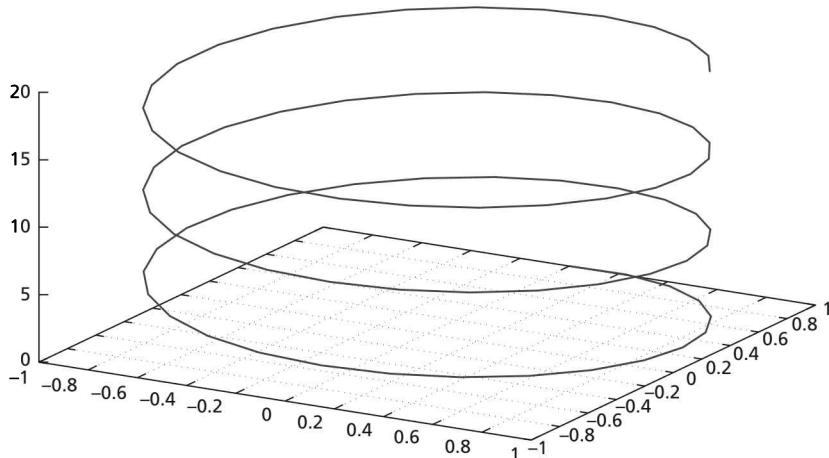


Figure 1.9 Plotting a line in 3D with GNUPlot

Note that this feature of three-dimensional plotting could also be used to generate a three-dimensional scatter plot of some data. To achieve this, we should set the `noline` option to `.true.` to avoid that GNUPlot connects data points with a line and specify a `marker` that generates a mark for each of the data-point combinations.

1.7 Further reading

The best way to learn a programming language is to read, write, and run programs immediately. Kendrick and Amman (1999) provide some guidance through the most widely used programming languages of economists. Aruoba and Fernandez-Villaverde (2015) compare some modern programming languages (including Fortran 2008) with respect to their speed and coding requirements. As it turns out, Fortran has a considerable speed advantage and the coding is fairly simple and compact. There are many good books on Fortran programming. Chivers and Sleightholme (2012) provide a very gentle introduction for complete beginners with little or no programming background, but also offer a lot of advanced material for experienced Fortran programmers who want to update their skills. Each chapter contains many example programs and a list of problems to solve. While Chapman (2004) covers almost the same material, the main advantage is the very student friendly design. Throughout the book various boxes highlight good programming practices and programming pitfalls and specific sidebars provide additional information of potential interest to the student. Students are especially motivated by various quizzes that are answered in the appendix and many exercises at the end of each chapter.

1.8 Exercises

- 1.1. (a) Write a program that reads two scalars, say x and y of type `real*8` from the console. Print an error message when the numbers are not readable. Otherwise the program should compute $x+y$, $x-y$, $x*y$ and x/y and print the answers in a readable fashion. Test your program with the numbers $x=2$ and $y=4$.
- (b) Change your program so that the two scalars x and y are of type `integer`. Test the program again with $x=2$ and $y=4$ and explain the difference.
- 1.2. Write a program that adds the numbers 55,555,553 and 10,000,001 and stores the result in variables `sum1` and `sum2` which are of type `real` and `real*8`, respectively. Print the values of `sum1` and `sum2` to the console, compare the two results and explain the difference.
- 1.3. (a) Store the value of 10^9 in a variable of type `real*8` in three ways: `10**9`, `10d0**9`, and `10**9d0`. Then repeat the same for the value 10^{10} . Print out the variable values to the console and check the result.
- (b) Extend the program and define two `real*8` variables of value 0.00000000003. The first definition ends with `d0` while in the second definition the `d0` is omitted. Print the values of the two variables with format `f30.25` and explain the difference.
- (c) Finally define two `real*8` variables of value 3.1415926535. Again, the first definition ends with `d0` while in the second definition the `d0` is omitted. Print the values of the two variables with format `f15.12` and explain the difference.
- 1.4. Write a program that evaluates the logical expression

```
x >= 3 .and. y <= 4 .and. z == 5 .or. x <= y .and. y < z
```

for $x = 4$, $y = 6$, and $z = 8$. Explain the result. Evaluate again for $x = 4$, $y = 6$, and $z = 2$ and explain.

- 1.5. The German income tax function has four brackets:

| If taxable income y is | | |
|--------------------------|-----------|----------------------------------|
| but not | more than | then income tax $T(y)$ is (in €) |
| 0 € | 8130 € | 0 |
| 8131 € | 13469 € | $(933.70x + 1400)x$ |
| 13470 € | 52881 € | $(228.74z + 2397)z + 1014$ |
| 52882 € | 250730 € | $0.42y - 8196$ |
| 250731 € | | $0.45y - 15718$ |

$$x = (y - 8130)/10000, z = (y - 13469)/10000.$$

Write a program that reads some taxable income y of type `real*8` from the console and then computes the resulting tax burden $T(y)$, the average tax rate $\frac{T(y)}{y}$, and the marginal tax rate $\frac{\partial T(y)}{\partial y}$ and prints it to the console.

- 1.6. The Fibonacci-series is defined as follows: The first two elements of the series are $a_1 = 1$ and $a_2 = 1$. Each of the following elements is computed as the sum of the two previous elements, i.e. $a_n = a_{n-1} + a_{n-2}$. The first ten elements of the Fibonacci-series are therefore

$$1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$$

- (a) Write a program that computes the n -th element of the Fibonacci-series. The main program should first read the value of an `integer` scalar n which determines the requested element a_n of the series. Then the main program should print out the result to the console using a function `fib(n)`. This function calculates a_n using a `do-loop`.
- (b) In the second step add to this program a function `biform(n)` which uses Binet's formula

$$F_n = \frac{\phi^n - (1 - \phi)^n}{\sqrt{5}} \quad \text{with } \phi = \frac{1 + \sqrt{5}}{2}$$

to compute the n -th Fibonacci number, where ϕ is the *golden ratio*.

- (c) Define a maximum number `plotmax` and compute the relative difference between the two approaches for each element up to `plotmax`. Plot this difference to the console.
- 1.7. Write a program that simulates rolling a pair of six-sided dice. The outcome we are interested in is the sum of the values the two dice show. We know that with probability $1/36$ this sum equals 2, with probability $2/36$ it equals 3, and so on. The goal of this task is to simulate this probability distribution using random number generation. Therefore define a number `iter` of times for which you simulate rolling the dice. In each iteration, use the intrinsic random number generator subroutine `random_number(x)`, which sets the argument x to a pseudo random real number x with $0 \leq x \leq 1$. Use this number to simulate the outcome of one roll of one dice. Do this twice and sum up the result to a variable `d`. To store the results, create an array `Dsum(2:12)` and add a value of 1 to the entry `d` of this array. The simulated probability distribution is then `dble(Dsum) / dble(iter)`. Print this probability distribution to the console.

Then generalize your program to rolling n dice, each with k sides. Again, plot the results for the sums between n and nk to the console.

Hint: Use the intrinsic subroutine `random_seed()` at the beginning of the program. What is the effect of this subroutine?

- 1.8. Two six-sided dices are rolled until their sum equals a value $x > 2$ or until their sum is equal to a value $y > 2$. Write a simulation program to determine the percentage of how often the game will be stopped by the first condition instead of the second condition. To generalize the simulation procedure of rolling the dice, create a subroutine `random_int(res, intl, inth)`. This subroutine should generate a random number of type `integer` between the values `intl` and `inth`. Thereby assume that each of the potential integer values $\{intl, intl+1, \dots, inth\}$ occurs with equal probability. The simulated random number should be stored in the value `res`.

Plot the percentages and the largest number of rolls before the game stops to the console. Set $x = 4$ and $y = 10$ and test the program. Then set $x = 4$ and $y = 7$. Explain your results!

Now use three dice to simulate the game. First set $x = 4$ and $y = 17$ and then set $x = 4$ and $y = 5$. Again explain your results.

- 1.9. (a) Write a function `utility(c, gamma)` that computes for an input variable `c` and an intertemporal substitution elasticity `gamma` the value of the utility function

$$u(c) = \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Function `utility(c, gamma)` should first check whether `c` is positive. In case $c < 0$ an error message should be written to the console and the program should stop (use the `stop` command).

- (b) Now write a program that tests the function `utility(c, gamma)` with alternative values for `c` and `gamma`. Then plot the function using the toolbox for alternative parameter values $\gamma \in [0.25, 0.5, 0.75, 1.25]$.

- 1.10. Write a subroutine `utility_int(a, b, u)` where the input values `a` and `b` define the endpoints of an interval $[a, b]$. Within this interval the subroutine should compute the values of the function $u(c)$ from the previous exercise at `n` nodes. The function values should be stored in the array `u(:)` which is of assumed size and will be given back to the main program. The value of `n` can be calculated from size of `u(:)`. Therefore proceed via the following steps:

- (a) At the beginning of the subroutine check whether $0 < a < b$. If this is not the case, write an error message and stop the program.
 (b) If $0 < a < b$ holds then determine `n` by setting `n = size(u)`.

- (c) Compute the array entries in $u(:)$ as $u(c_j)$ using the function `utility(c)` with a specific value for γ . Define

$$c_j = a + \frac{(j-1)}{n-1}(b-a) \quad \text{for } j = 1, \dots, n.$$

Don't forget the `db1e` command.

Now set $\gamma = 0.5$ and write a program that tests the subroutine using $a = 1$ and $b = 2$, and an array length of 11. Write your results to the console.

2 Numerical solution methods

In this chapter we develop simple methods for solving numerical problems. We start with linear equation systems, continue with nonlinear equations and finally talk about optimization, interpolation, and integration methods. Each section starts with a motivating example from economics before we discuss some of the theory and intuition behind the numerical solution method. Finally, we present some Fortran code that applies the solution technique to the economic problem.

2.1 Matrices, vectors, and linear equation systems

This section mainly addresses the issue of solving linear equation systems. As a linear equation system is usually defined by a matrix equation, we first have to talk about how to work with matrices and vectors in Fortran. After that, we will present some linear equation system solving techniques.

2.1.1 MATRICES AND VECTORS IN FORTRAN

The general structure of a matrix A and a vector b in mathematics is given by

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

A matrix consists of several columns, where a vector only has one. We call A a $m \times n$ matrix, and b a n -dimensional vector. A natural way to store matrices and vectors in Fortran is via the concept of arrays. We thereby store matrices in a two-dimensional array of lengths m and n and a vector in a one-dimensional array of length n . There are a number of intrinsic Fortran functions that were especially written for operation with matrices and vectors. A summary of these is given in Program 2.1. Most of these functions should be self-explanatory, however they are also described on our website.

Program 2.1 Matrix and vector operations

```

program matrices

    implicit none
    integer :: i, j
    real*8 :: a(4), b(4)
    real*8 :: x(2, 4), y(4, 2), z(2, 2)

    ! initialize vectors and matrices
    a = (/dble(5-i), i=1, 4/)
    b = a+4d0
    x(1, :) = (/1d0, 2d0, 3d0, 4d0/)
    x(2, :) = (/5d0, 6d0, 7d0, 8d0/)
    y = transpose(x)
    z = matmul(x, y)

    ! show results of different functions
    write(*,'(a,4f7.1/)')           vector a = ',(a(i),i=1,4)
    write(*,'(a,f7.1/)')           sum(a) = ',sum(a)
    write(*,'(a,f7.1/)')           product(a) = ',product(a)
    write(*,'(a,f7.1/)')           maxval(a) = ',maxval(a)
    write(*,'(a,i7/)')            maxloc(a) = ',maxloc(a)
    write(*,'(a,f7.1/)')           minval(a) = ',minval(a)
    write(*,'(a,i7/)')            minloc(a) = ',minloc(a)
    write(*,'(a,4f7.1/)')           cshift(a, -1) = ',cshift(a, -1)
    write(*,'(a,4f7.1/)')           eoshift(a, -1) = ',eoshift(a, -1)
    write(*,'(a,l7/)')             all(a<3d0) = ',all(a<3d0)
    write(*,'(a,l7/)')             any(a<3d0) = ',any(a<3d0)
    write(*,'(a,i7/)')            count(a<3d0) = ',count(a<3d0)
    write(*,'(a,4f7.1/)')           vector b = ',(b(i),i=1,4)
    write(*,'(a,f7.1/)')           dot_product(a,b) = ',dot_product(a,b)
    write(*,'(a,4f7.1/,20x,4f7.1/)') &
        matrix x = ',((x(i,j),j=1,4),i=1,2)
    write(*,'(a,2f7.1,3(/20x,2f7.1/))') &
        transpose(x) = ',((y(i,j),j=1,2),i=1,4)
    write(*,'(a,2f7.1/,20x,2f7.1/)') &
        matmul(x,y) = ',((z(i,j),j=1,2),i=1,2)

end program

```

2.1.2 SOLVING LINEAR EQUATION SYSTEMS

In this section we would like to show how to solve linear equation systems. There are two ways to solve these systems: by factorization or iterative methods. Both methods will be discussed in the following.

Example Consider the supply and demand functions for three goods given by

$$\begin{aligned}
 q_1^s &= -10 + p_1 & q_1^d &= 20 - p_1 - p_3 \\
 q_2^s &= 2p_2 & q_2^d &= 40 - 2p_2 - p_3 \\
 q_3^s &= -5 + p_3 & q_3^d &= 25 - p_1 - p_2 - p_3
 \end{aligned}$$

As one can see, the supply of the three goods only depends on their own price, while the demand side shows strong price interdependencies. In order to solve for the equilibrium prices of the system we set supply equal to demand $q_i^s = q_i^d$ in each market which after rearranging yields the linear equation system

$$\begin{aligned} 2p_1 + p_3 &= 30 \\ 4p_2 + p_3 &= 40 \\ p_1 + p_2 + 2p_3 &= 30. \end{aligned}$$

We can express a linear equation system in matrix notation as

$$Ax = b \quad (2.1)$$

where x defines an n -dimensional (unknown) vector and A and b define a $n \times n$ matrix and a n -dimensional vector of exogenous parameters of the system. In the above example, we obviously have $n = 3$ and

$$A = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 2 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 30 \\ 40 \\ 30 \end{bmatrix}.$$

Gaussian elimination and factorization We now want to solve for the solution x of a linear equation system. Of course, if A were a lower triangular matrix of the form

$$A = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix},$$

the elements of x could be easily derived by simple forward substitution, i.e.

$$\begin{aligned} x_1 &= b_1/a_{11} \\ x_2 &= (b_2 - a_{21}x_1)/a_{22} \\ &\vdots \\ x_n &= (b_n - a_{n1}x_1 - a_{n2}x_2 - \dots - a_{n(n-1)}x_{n(n-1)})/a_{nn}. \end{aligned}$$

Similarly, the problem can be solved by backward substitution, if A was an upper triangular matrix. However, in most cases A is not triangular. Nevertheless, if a solution

to the equation system existed, we could break up A into the product of a lower and upper triangular matrix, meaning there is a lower triangular matrix L and an upper triangular matrix U , so that A can be written as

$$A = LU.$$

If we knew these two matrices, equation (2.1) could be rearranged as follows:

$$Ax = (LU)x = L(Ux) = Ly = b.$$

Consequently, we first would determine the vector y from the lower triangular system $Ly = b$ by forward substitution and then x via $Ux = y$ using backward substitution.

Matrix decomposition In order to factorize a matrix A into the components L and U we apply the *Gaussian elimination method*. In our example we can rewrite the problem as

$$\begin{aligned} Ax &= \begin{bmatrix} 2 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 2 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_L \times \underbrace{\begin{bmatrix} 2 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 2 \end{bmatrix}}_U \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \underbrace{\begin{bmatrix} 30 \\ 40 \\ 30 \end{bmatrix}}_b. \end{aligned}$$

We now want to transform L and U in order to make them lower and upper triangular matrices. However, these transformations must not change the result x of the equation system $LUX = b$. It can be shown that subtracting the multiple of one row from another satisfies this condition. Note that when we *subtract* a multiple of row i from row j in matrix U , we have to *add* the same multiple to the same cell in matrix L which is eliminated in matrix U .

In the above example, the first step is to eliminate the cells below the diagonal in the first column of U . In order to get a zero in the first column of the last row, one has to multiply the first row of U by 0.5 and subtract it from the third. Consequently, we have to add 0.5 to the first column of the third row of L . After this step matrices L and U are transformed to

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0 & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 4 & 1 \\ 0 & 1 & 1.5 \end{bmatrix}.$$

In order to get a zero in the second column of the last row of U , we have to multiply the second row by 0.25 and subtract it from the third line. The entry in the last line and the second column of L then turns into 0.25. After this second step the matrices are

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.25 & 1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 2 & 0 & 1 \\ 0 & 4 & 1 \\ 0 & 0 & 1.25 \end{bmatrix}.$$

Now L and U have the intended triangular shape. It is easy to check that $A = LU$ still holds. Hence, the result of the equation systems $LUX = b$ and $Ax = b$ are identical. The above approach can be applied to any invertible matrix A in order to decompose it into the L and U factors. We can now solve the system $Ly = b$ for y by using forward substitution. The solution to this system is given by

$$\begin{aligned} y_1 &= 30/1 = 30, \\ y_2 &= (40 - 0 \cdot 30)/1 = 40 \quad \text{and} \\ y_3 &= (30 - 0.5 \cdot 30 - 0.25 \cdot 40)/1 = 5. \end{aligned}$$

Given the solution $y = [30 \ 40 \ 5]^T$, the linear system $Ux = y$ can then be solved using backward substitution, yielding the solution of the original linear equation, i.e.

$$\begin{aligned} x_3 &= 5/1.25 = 4, \\ x_2 &= (40 - 1 \cdot 4)/4 = 9 \quad \text{and} \\ x_1 &= (30 - 0 \cdot 9 - 1 \cdot 4)/2 = 13. \end{aligned}$$

The solution of a linear equation system via LU -decomposition is implemented in the toolbox that accompanies this book. Program 2.2 demonstrates its use. In this program, we first include the `toolbox` module and specify the matrices A, L, U and the vector b . We then initialize A and b with the respective values given in the above example. The subroutine `lu_solve` that comes with the toolbox can now solve the equation system $Ax = b$. The solution is stored in the vector `b` at the end of the subroutine. Alternatively we could solely factorize A . This can be done with the subroutine `lu_dec`. This routine receives a matrix A and stores the L and U factors in the respective variables. The output shows the same solution and factors as computed above.

The so-called L-U factorization algorithm is faster than other linear solution methods such as computing the inverse of A with determinants and then computing $A^{-1}b$ or using Cramer's rule. Although L-U factorization is one of the best general methods for solving a linear equation system, situations may arise in which alternative methods may be preferable. For example, when one has to solve a series of linear equation systems which all have the same A matrix but different b vectors, b_1, b_2, \dots, b_m it is often

Program 2.2 Linear equation-solving using the toolbox

```

program lineqsys

use toolbox

implicit none
integer :: i, j
real*8 :: A(3, 3), b(3)
real*8 :: L(3, 3), U(3, 3)

! set up matrix and vector
A(1, :) = (/ 2d0, 0d0, 1d0 /)
A(2, :) = (/ 0d0, 4d0, 1d0 /)
A(3, :) = (/ 1d0, 1d0, 2d0 /)
b       = (/ 30d0, 40d0, 30d0 /)

! solve the system
call lu_solve(A, b)

! decompose matrix
call lu_dec(A, L, U)

! output
write(*,'(a,3f7.2/)')' x = ', (b(j),j=1,3)
write(*,'(a,3f7.2/,2(5x,3f7.2/))' &
      ' L = ',((L(i,j),j=1,3),i=1,3)
write(*,'(a,3f7.2/,2(5x,3f7.2/))' &
      ' U = ',((U(i,j),j=1,3),i=1,3)

end program

```

computationally more efficient to compute and store the inverse of A and then compute the solutions $x = A^{-1}b$; by performing direct matrix vector multiplications.

Gaussian elimination can be accelerated for matrices possessing special structures. If A was symmetric positive definite, A could be expressed as the product

$$A = LL^T$$

of a lower triangular matrix L and its transpose. In this situation one can apply a special form of Gaussian elimination, the so-called *Cholesky factorization algorithm*, which requires about half of the operations of the Gaussian approach. L is called the Cholesky factor or square root of A . Given the Cholesky factor of A , the linear equation

$$Ax = LL^T x = L(L^T x) = b$$

may be solved efficiently by using forward substitution to solve $Ly = b$ and then backward substitution to solve $L^T x = y$.

Another factorization method decomposes $A = QR$, where Q is an *orthogonal* matrix and R an upper triangular matrix. An orthogonal matrix has the property $Q^{-1} = Q^T$. Hence, the solution of the equation system can easily be computed by solving the system

$$Rx = Q^T b$$

via backward induction. However, computing a QR decomposition usually is more costly and more complicated than the L-U factorization.

Matrix inversion We can also invert matrices by using L-U factorization. Since $AA^{-1} = I$, the inversion of a $n \times n$ matrix A is equivalent to successively computing the solution vectors x_i of the equation systems

$$\begin{bmatrix} a_{11} & \dots & \dots & a_{1n} \\ \vdots & & & \vdots \\ a_{n1} & \dots & \dots & a_{nn} \end{bmatrix} \times \underbrace{\begin{bmatrix} x_{1i} \\ \vdots \\ x_{ni} \end{bmatrix}}_{x_i} = \underbrace{\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}}_{e_i} \quad \text{for } i = 1, \dots, n,$$

where e_i denotes the i -th unit vector, i.e. a vector with all entries being equal to zero, but the i -th entry being equal to one. Defining a matrix in which the i -th column is equal to the solution x_i , we obtain the inverse of A , i.e.

$$A^{-1} = \begin{bmatrix} x_{11} & \dots & \dots & x_{1n} \\ \vdots & & & \vdots \\ x_{n1} & \dots & \dots & x_{nn} \end{bmatrix}.$$

A matrix inversion can be performed by means of the function `lu_invert` which also comes with the toolbox. Program 2.3 shows how to do this. After having specified A and b in this program, we compute the inverse of A with the function `lu_invert`. The result is stored in the array `Ainv`. We then compute the solution of the above equation system by multiplying `Ainv` with `b`. The result is printed on the screen.

Iterative methods The Gaussian elimination procedure computes an exact solution to the equation system $Ax = b$. However, the algorithm is quite unstable and sensitive to round-off errors. A class of more stable algorithms is the class of *iterative methods* such as the Jacobi or the Gauss-Seidel approach. In opposition to the factorization methods, both of these methods only yield approximations to the exact solution.

The basic idea of iterative methods is quite simple. Given the linear equation system $Ax = b$, one can choose any invertible matrix Q and rewrite the system as

$$Ax + Qx = b + Qx \quad \text{or}$$

$$Qx = b + (Q - A)x.$$

Program 2.3 Inversion of a matrix

```

program inversion

use toolbox

implicit none
integer :: i, j
real*8 :: A(3, 3), Ainv(3, 3), b(3)

! set up matrix and vector
A(1, :) = (/ 2d0, 0d0, 1d0/)
A(2, :) = (/ 0d0, 4d0, 1d0/)
A(3, :) = (/ 1d0, 1d0, 2d0/)
b         = (/30d0, 40d0, 30d0/)

! invert A
Ainv = lu_invert(A)

! calculate solution
b = matmul(Ainv, b)

! output
write(*,'(a,3f7.2/)')' x = ', (b(j),j=1,3)
write(*,'(a,3f7.2/,2(8x,3f7.2/))' &
      ' A^-1 = ', ((Ainv(i,j),j=1,3),i=1,3)

end program

```

We therefore obtain

$$x = Q^{-1} [b + (Q - A)x] = Q^{-1}b + (I - Q^{-1}A)x.$$

If we let Q have a simple and easily invertible form, e.g. the identity matrix, we can derive the iteration rule

$$x^{i+1} = Q^{-1}b + (I - Q^{-1}A)x^i.$$

It can be shown that, if the above iteration converges, it converges to the solution of linear equation system $Ax = b$.¹

The Jacobi and the Gauss-Seidel method are popular representatives of the iteration methods. While the Jacobi method sets Q equal to the diagonal matrix formed out of the diagonal entries of A , the Gauss-Seidel method sets Q equal to the upper triangular matrix of upper triangular elements of A . Both of these matrices are easily invertible by hand. Program 2.4 shows how iterative methods can be applied to our problem.

When A becomes large and sparse, i.e. many entries of A are equal to zero, the convergence speed of both the Jacobi and Gauss-Seidel methods becomes quite slow. Therefore one usually applies SOR-methods (successive overrelaxation), another representative of iterative methods. However, due to their higher complexity, we will not discuss these methods in detail.

¹ Convergence depends on the value of the spectral radius $\|I - Q^{-1}A\|$ which has to be smaller than 1.

Program 2.4 Jacobi iteration

```

program jacobi

implicit none
integer :: iter, i
real*8 :: A(3, 3), Dinv(3, 3), ID(3, 3), C(3, 3)
real*8 :: b(3), d(3), x(3), xold(3)

! set up matrices and vectors
A(1, :) = (/ 2d0, 0d0, 1d0 /)
A(2, :) = (/ 0d0, 4d0, 1d0 /)
A(3, :) = (/ 1d0, 1d0, 2d0 /)
b         = (/ 30d0, 40d0, 30d0 /)

ID = 0d0
Dinv = 0d0
do i = 1,3
    ID(i, i) = 1d0
    Dinv(i, i) = 1d0/A(i, i)
enddo

! calculate iteration matrix and vector
C = ID-matmul(Dinv, A)
d = matmul(Dinv, b)

! initialize xold
xold = 0d0

! start iteration
do iter = 1, 200

    x = d + matmul(C, xold)

    write(*,'(i4,f12.7)')iter, maxval(abs(x-xold))

    ! check for convergence
    if(maxval(abs(x-xold)) < 1d-6)then
        write(*,'(/a,3f12.2)')' x = ', (x(i),i=1,3)
        stop
    endif

    xold = x
enddo

write(*,'(a)')'Error: no convergence'

end program

```

2.2 Nonlinear equations and equation systems

One of the most basic numerical problems encountered in computational economics is to find the solution to a nonlinear equation or a whole system of nonlinear equations. A nonlinear equation system usually can be defined by a function

$$f(x) : \mathbb{R}^n \rightarrow \mathbb{R}^n$$

that maps an n dimensional vector x into the space \mathbb{R}^n . We call the solution to the linear equation system $f(x) = 0$ a *root* of f . The root of a nonlinear equation system usually

has no closed-form solution. Consequently, various numerical methods addressing the issue of root-finding have been invented. In this section, we introduce a very simple one-dimensional method called bisection search. As this method converges quite slowly, we show how to speed it up by using Newton's method. We then discuss some modifications of Newton's algorithm and show the very general class of fixed-point iteration methods. All these methods will, like in Section 2.1, be demonstrated with a simple example. We close the section by introducing a solver for multidimensional nonlinear equation systems.

Example Suppose the demand function in a goods market is given by

$$q^d(p) = 0.5p^{-0.2} + 0.5p^{-0.5},$$

where the first term denotes domestic demand and the second term export demand. Supply should be inelastic and given by $q^s = 2$ units. At what price p^* does the market clear? Setting supply equal to demand leads to the equation

$$0.5p^{-0.2} + 0.5p^{-0.5} = 2$$

which can be reformulated as

$$f(p) = 0.5p^{-0.2} + 0.5p^{-0.5} - 2 = 0. \quad (2.2)$$

Equation (2.2) exactly has the form $f(p) = 0$ described above. The market clearing price consequently is the solution to a nonlinear equation. Note that it is not possible to derive an analytical solution to this problem. Hence, we need a numerical method to solve for p^* .

2.2.1 BISECTION SEARCH IN ONE DIMENSION

A very intuitive and ad hoc approach to solving nonlinear equations in one dimension is the so-called *bisection search*. The basic idea behind this method is a mathematical theorem called *intermediate value theorem*. It states that if $[a, b]$ is an interval, f is a continuous function, and $f(a)$ and $f(b)$ have different signs, i.e. $f(a) \cdot f(b) < 0$, then f has a root in the interval $[a, b]$. The bisection algorithm now successively bisects the interval $[a, b]$ into intervals of equal size and tests in which interval the root of f is located by using this theorem.

Specifically, the algorithm proceeds as follows: suppose we had an interval $[a_i, b_i]$ with the property that $f(a_i) \cdot f(b_i) < 0$, i.e. f has a root in this interval. We now bisect this interval into the subintervals $[a_i, x_i]$ and $[x_i, b_i]$ with

$$x_i = \frac{a_i + b_i}{2}.$$

We can now test in which interval the root of f is located and calculate a new interval $[a_{i+1}, b_{i+1}]$ by

$$\begin{aligned} a_{i+1} &= a_i & \text{and } b_{i+1} &= x_i & \text{if } f(a_i) \cdot f(x_i) < 0, \\ a_{i+1} &= x_i & \text{and } b_{i+1} &= b_i & \text{otherwise.} \end{aligned}$$

Figure 2.1 demonstrates the approach.

Note that due to the bisection of $[a_i, b_i]$, we successively halve the interval size with every iteration step. Consequently x_i converges towards the real root $f(x^*)$ of f . We stop the iteration process if x_i has sufficiently converged, i.e. the difference between two successive values is smaller than an exogenously specified tolerance level ϵ

$$|x_{i+1} - x_i| = |x_{i+1} - a_{i+1}| = |x_{i+1} - b_{i+1}| = \frac{1}{2^{i+2}} |b_0 - a_0| < \epsilon.$$

Program 2.5 shows how to find the equilibrium price in the above example by using bisection search.

The program proceeds as follows: We first have to make an initial guess of the interval in which the actual root of f is located. We chose $a_0 = 0.05$ and $b_0 = 0.25$ in our case. We then calculate function values at the interval endpoints via (2.2) and check whether our condition for a root of f in $[a_0, b_0]$ is satisfied. If not, we write an error message to the console and abort the program. Having prepared all this, our iteration process starts by calculating x_0 . Note that we specify a maximum number of iterations in our do-loop. If there is no convergence after 200 iterations, the program will be aborted with an error message. Having calculated the new x and the respective function value, we can check

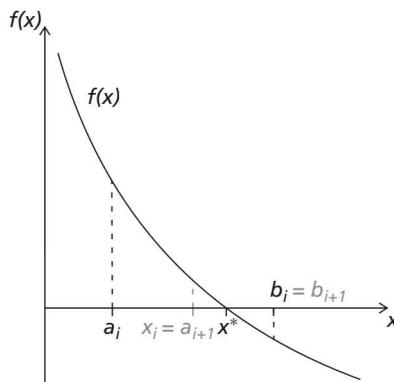


Figure 2.1 Bisection search for finding the root of a function

Program 2.5 Bisection search in one dimension

```

program bisection

implicit none
integer :: iter
real*8 :: x, a, b, fx, fa, fb

! set initial guesses and function values
a = 0.05d0
b = 0.25d0
fa = 0.5d0*a**(-0.2d0)+0.5d0*a**(-0.5d0)-2d0
fb = 0.5d0*b**(-0.2d0)+0.5d0*b**(-0.5d0)-2d0

! check whether there is a root in [a,b]
if(fa*fb >= 0d0)then
    stop 'Error: There is no root in [a,b]'
endif

! start iteration process
do iter = 1, 200

    ! calculate new bisection point and function value
    x = (a+b)/2d0
    fx = 0.5d0*x**(-0.2d0)+0.5d0*x**(-0.5d0)-2d0

    write(*,'(i4,f12.7)')iter, abs(x-a)

    ! check for convergence
    if(abs(x-a) < 1d-6)then
        write(*,'(/a,f12.7,a,f12.9)')' x = ',x,' f = ',fx
        stop
    endif

    ! calculate new interval
    if(fa*fx < 0d0)then
        b = x
        fb = fx
    else
        a = x
        fa = fx
    endif
enddo

write(*,'(a)')'Error: no convergence'

end program

```

whether the result of two successive iterations satisfy our convergence condition. If this is the case, we stop the program and display our approximation of the root of f . If not, we choose the appropriate interval and begin a new iteration.

Looking at the output of the program we find that our interval is halved in every iteration step. This is obvious from

$$|x_{i+1} - x_i| = \frac{1}{2^{i+2}} |b_0 - a_0| = \frac{1}{2} \left[\frac{1}{2^{i+1}} |b_0 - a_0| \right] = \frac{1}{2} |x_i - x_{i-1}|.$$

In general, we call convergence speed linear if

$$|x_{i+1} - x_i| < c|x_i - x_{i-1}| \quad \text{with} \quad c > 0,$$

i.e. the difference between the solutions of two successive iteration steps diminishes by a constant factor c . Hence, bisection converges linearly to the actual root of f . In Section 2.2.2 we will show how to enhance this convergence speed by using some more information about the shape of our function.

2.2.2 NEWTON'S METHOD IN ONE DIMENSION

One of the best known and most efficient methods to solve a linear equation is the so-called *Newton method*. It is based on the idea of successive linearization, which allows the replacing of a nonlinear root-finding problem with a sequence of simpler linear ones. The solutions of the simpler problems finally converge to the solution of the nonlinear one.

Starting at an arbitrary point x_0 , we can approximate f by a first-order Taylor expansion, i.e.

$$\hat{f}(x) \approx f(x_0) + f'(x_0)(x - x_0).$$

As the function value $f(x_0)$ and the derivative of f at x_0 usually are known, we can easily find the solution to the equation $\hat{f}(x) = 0$. This solution is given by

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

Figure 2.2 demonstrates the approach. We approximate the function f linearly at the point x_0 and find the solution x_1 to the approximated equation system. x_1 is now closer to the true solution x^* of the equation $f(x) = 0$ than x_0 .

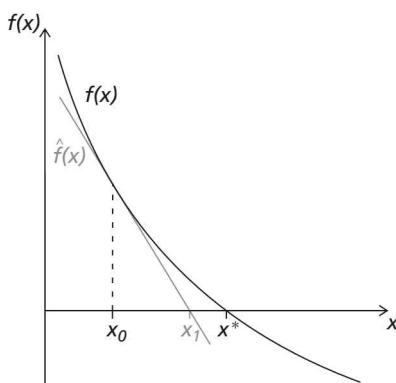


Figure 2.2 Newton's method for finding the root of a function

By defining the iteration rule

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \quad (2.3)$$

and starting with an initial guess x_0 , we therefore obtain a series of values x_0, x_1, x_2, \dots that converges towards x^* . We again stop the iteration process, if x_i has sufficiently converged, i.e.

$$|x_{i+1} - x_i| = \left| \frac{f(x_i)}{f'(x_i)} \right| < \epsilon.$$

Program 2.6 demonstrates this approach. We first have to make an initial guess x_0 . We chose 0.05 in our example. We then start the iteration process by calculating the function value and the first derivative

$$f'(p) = -0.1p^{-1.2} - 0.25p^{-1.5}.$$

Program 2.6 Newton's method in one dimension

```
program newton

implicit none
integer :: iter
real*8 :: xold, x, f, fprime

! set initial guess
xold = 0.05d0

! start iteration process
do iter = 1, 200

    ! calculate function value
    f = 0.5d0*xold**(-0.2d0)+0.5d0*xold**(-0.5d0)-2d0

    ! calculate derivative
    fprime = -0.1d0*xold**(-1.2d0)-0.25d0*xold**(-1.5d0)

    ! calculate new value
    x = xold - f/fprime

    write(*,'(i4,f12.7)')iter, abs(x-xold)

    ! check for convergence
    if(abs(x-xold) < 1d-6)then
        write(*,'(/a,f12.7,a,f12.9)')' x = ',x,'      f = ',f
        stop
    endif

    ! copy old value
    xold = x
enddo

write(*,'(a)')'Error: no convergence'

end program
```

The new value x_{i+1} can then be computed from (2.3). We finally check for convergence and stop the program, if convergence is reached, otherwise we store the value x_{i+1} and start a new iteration with it.

If we take a closer look at the output of this program, we see that, from iteration step 3 onwards, the number of zeros in our interval $|x_{i+1} - x_i|$ always doubles. Specifically, it can be shown that

$$|x_{i+1} - x_i| < c|x_i - x_{i-1}|^2$$

with a certain $c > 0$ holds, if x_i is sufficiently close to the true solution of $f(x) = 0$ and the iteration process converges. We call this convergence property quadratic convergence. This convergence speed obviously is much faster than the linear speed of the bisection search.

However, Newton's method also comes with disadvantages. One of them obviously is that f has to be continuously differentiable at least once, for quadratic convergence even twice. In addition, one has to keep in mind that Newton's algorithm not necessarily converges to the root of f . There are several properties a function can have that prevent convergence of the iteration process. If for example f has an extreme value near the root x^* it might be that the result of one iteration x_i is very close to this value. Consequently, the derivative $f'(x_i)$ gets close to zero and the calculation in (2.3) might lead to an error. Furthermore, if f has an inflection point, a situation like on the left side of Figure 2.3 might arise, where the iterated values jump from a left to a right point and back. Last, if f is not defined for every $x \in \mathbb{R}$, like e.g. $\log(x)$ in the right part of Figure 2.3, we might jump outside the defined area of f while iterating. This problem, however, can often be solved by choosing different initial guesses for the starting value of the Newton procedure.

Sometimes it is quite difficult to calculate the derivative of f analytically. If this is the case, one usually approximates the derivative by a secant, i.e.

$$f'(x_i) \approx \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}.$$

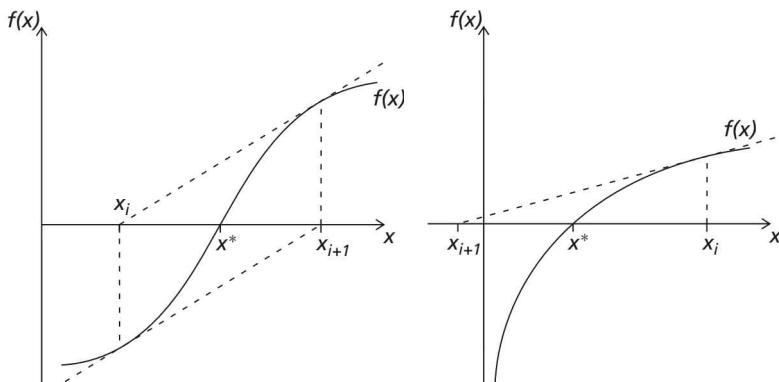


Figure 2.3 Convergence problems in Newton's algorithm

The iteration procedure in (2.3) then turns into

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i).$$

This method is called *secant method*. Note that we now need two starting values x_0 and x_1 in order to start the iteration process. It can be shown that in terms of convergence speed, the secant method iteration values follow

$$|x_{i+1} - x_i| = c|x_i - x_{i-1}|^{1.6}.$$

Consequently, convergence speed is somewhat in between linear and quadratic. However, the advantage of the secant method is that in every iteration only one function value $f(x_i)$ has to be calculated. In opposition to that, the original Newton procedure needs to calculate $f(x_i)$ and $f'(x_i)$, hence, two function values. Consequently, if we combine two steps of the secant procedure into one in order to reach comparability, we obtain the convergence speed

$$|x_{i+1} - x_i| = c|c|x_{i-1} - x_{i-2}|^{1.6}|^{1.6} = c^{2.6}|x_{i-1} - x_{i-2}|^{2.56}.$$

Therefore, the secant method in general is more efficient than Newton's algorithm. Furthermore it is easier to implement, as we do not need to calculate the derivative of f .

2.2.3 FIXED-POINT ITERATION METHODS

The last and most general class of methods we want to introduce for solving one-dimensional, nonlinear equations is the class of *fixed-point iteration* methods. This method is based on a reformulation of the nonlinear equation. Suppose we transform $f(x) = 0$ by multiplying with σ and adding x on both sides into

$$x = x + \sigma f(x) =: g(x).$$

Obviously, the solution x^* to this equation is a fixed point of $g(\cdot)$ and in addition $f(x^*) = 0$ holds if $\sigma \neq 0$. Hence, the solution to the fixed-point equation $x = g(x)$ is also the solution to the nonlinear equation $f(x) = 0$.

The fixed point iteration now is defined by

$$x_{i+1} = g(x_i) = x_i + \sigma f(x_i).$$

It can be shown that a fixed-point iteration procedure converges, if $|g'(x^*)| < 1$ and the initial guess x_0 is close enough to the root x^* of f . Convergence speed of a fixed-point iteration strongly depends on the function g , i.e. we have

Program 2.7 Fixed-point iteration in one dimension

```

program fixedpoint

implicit none
integer :: iter
real*8 :: xold, x, f, sigma

! set initial guess and chose sigma
xold = 0.05d0
sigma = 0.2d0

! start iteration process
do iter = 1, 200

    ! calculate function value
    f = 0.5d0*xold**(-0.2d0)+0.5d0*xold**(-0.5d0)-2d0

    ! calculate new value
    x = xold + sigma*f

    write(*,'(i4,f12.7)')iter, abs(x-xold)

    ! check for convergence
    if(abs(x-xold) < 1d-6)then
        write(*,'(/a,f12.7,a,f12.9)')' x = ',x,'      f = ',f
        stops
    endif

    ! copy old value
    xold = x
enddo

write(*,'(a)')'Error: no convergence'

end program

```

$$|x_{i+1} - x_i| = c|x_i - x_{i-1}|^p$$

if the first $p - 1$ derivatives of g at x^* are zero and the p th derivative is not equal to zero. As in general $g'(x^*) = 1 + \sigma f'(x^*) \neq 0$, the above fixed-point iteration will converge linearly towards the actual root of f . It is therefore quite slow, however, easy to implement. Program 2.7 shows the fixed-point iteration method for our example.

The key question in a fixed-point iteration is how to choose σ . Usually one has to try different values for σ . The rule of thumb is that the lower σ the slower the method converges. However, if σ is too large, it might happen that the method doesn't converge at all. Hence, choosing an appropriate σ is quite complicated and consequently fixed-point iteration is not used very often.

We want to close this section with a small remark. Note that Newton's method

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} =: g(x_i)$$

is also a fixed-point iteration procedure. This fixed-point iteration procedure

$$g'(x_i) = \frac{f(x_i)f''(x_i)}{[f'(x_i)]^2} = 0 \quad \text{and} \quad g''(x_i) = \frac{f''(x_i)}{f'(x_i)} \neq 0,$$

converges quadratically. Consequently, it might be possible to construct methods that converge even faster than Newton's algorithm by choosing an appropriate fixed-point iteration procedure. However, up to now, procedures that can be run with acceptable effort have only been found for very special cases of functions f .

2.2.4 MULTIDIMENSIONAL NONLINEAR EQUATION SYSTEMS

For multidimensional equation systems, solution methods become quite complicated. Therefore we don't want to discuss the theory in detail. For finding the root of a multidimensional equation system we use the so-called *Broyden's method*, which is the most popular multivariate generalization of the univariate secant method. This method is also called a *quasi-Newton method* as it approximates the Jacobi matrix of f in order to find the function's root. For demonstrating the use of Broyden's algorithm we consider another example.

Example Two firms compete in a simple Cournot duopoly with the inverse demand and the cost functions

$$P(q) = q^{-1/\eta} \quad C_k(q_k) = \frac{c_k}{2}q_k^2 \quad \text{for firm } k = 1, 2 \text{ with } q = q_1 + q_2.$$

Given the profit functions of the two firms

$$\Pi_k(q_1, q_2) = P(q_1 + q_2)q_k - C_k(q_k)$$

each firm k takes the other firm's output as given and chooses its own output level in order to solve

$$\frac{\partial \Pi_k}{\partial q_k} = f(q) = (q_1 + q_2)^{-1/\eta} - \frac{1}{\eta}(q_1 + q_2)^{-1/\eta-1}q_k - c_kq_k = 0 \text{ with } k = 1, 2.$$

The toolbox contains the subroutine `fzero` which implements Broyden's method. This function can be applied to both one- and multidimensional problems. Module 2.8m and Program 2.8 shows how to do this.

The module `globals` now contains both variables as well as a function. The variables are the structural parameters of our model and need no further explanation. The function `cournot` returns exactly the values of the two marginal profit equations. It therefore takes

Module 2.8m Multidimensional root-finding

```

module globals

    implicit none
    real*8 :: eta = 1.6d0
    real*8 :: c(2) = (/0.6d0, 0.8d0/)

contains

    ! function that defines the oligopoly equations
    function cournot(q)

        implicit none
        real*8, intent(in) :: q(:)
        real*8 :: cournot(size(q, 1)), QQ
        integer :: i

        QQ = sum(q)
        do i = 1, size(q, 1)
            cournot(i) = QQ**(-1d0/eta)-1d0/eta* &
                         QQ**(-1d0/eta-1)*q(i)-c(i)*q(i)
        enddo

    end function cournot

end module

```

Program 2.8 Multidimensional root-finding

```

include "prog02_08m.f90"

program oligopoly

    use globals
    use toolbox

    implicit none
    real*8 :: q(2)
    logical :: check

    ! initialize q
    q = 0.1d0

    ! find root
    call fzero(q, cournot, check)

    if(check) stop 'Error: fzero did not converge'

    ! output
    write(*,'(/a)')'                                Output'
    write(*,'(a,f10.4)')'Firm 1: ',q(1)
    write(*,'(a,f10.4)')'Firm 2: ',q(2)
    write(*,'(/a,f10.4)')'Price : ',(q(1)+q(2))**(-1d0/1.6d0)

end program

```

a vector $q(:)$ of arbitrary size as input, which contains two quantities q_1 and q_2 . The output of the function is again a vector of the same size as the input vector and contains the value of the marginal profit equations at the respective input quantities.

The program `oligopoly` executes the root-finding process. Beneath including the module `globals` that contains the function we want to find a root of, we also have to include the module `toolbox` which provides the subroutine `fzero`. Before we call this subroutine, we have to make an initial guess for the quantity vector q . We then let `fzero` search the root of the function `cournot`, which specifies the equations for our oligopoly quantities. In addition to the quantity vector q and the function `cournot`, we hand a logical variable `check` to the subroutine. If the iteration process of `fzero` does not converge, `check` will get the value `.true.`, else `.false.`. Note that we can only pass on functions or subroutines to `fzero` that are stored in a module. If the function `cournot` were part of the main program `oligopoly`, we would receive an error message from the compiler. Try to generate this error message on your own.

Given $\eta = 1.6$, $c_1 = 0.6$, and $c_2 = 0.8$ the program yields the solutions $q_1 = 0.8396$ and $q_2 = 0.6888$. We can now change the exogenous parameter values for η , c_1 , c_2 and check how this affects the solution.

Gauss-Seidel Iteration It may happen that `fzero` does not work. In this case various iteration algorithms serve as easy-to-implement and fast-solution alternatives. The most prominent ones are the so-called *Gauss-Jacobi* and the *Gauss-Seidel* methods which are now applied to solve the previous Cournot duopoly problem.

In order to implement an iteration process we reformulate the zero-profit conditions as

$$\begin{aligned} q_{1,i+1} &= \frac{1}{c_1} \left\{ (q_{1,i} + q_{2,i})^{-1/\eta} - \frac{1}{\eta} (q_{1,i} + q_{2,i})^{-1/\eta-1} q_{1,i} \right\} \\ q_{2,i+1} &= \frac{1}{c_2} \left\{ (q_{1,i} + q_{2,i})^{-1/\eta} - \frac{1}{\eta} (q_{1,i} + q_{2,i})^{-1/\eta-1} q_{2,i} \right\} \end{aligned}$$

where $q_{k,i}$ is the i th iterate of q_k . Starting with an initial guess $q_{k,0}$ the algorithm computes in each iteration i new values for the quantities q_k and then a new guess $q_{k,i+1}$ for the next iteration. Typically, the new guess is a linear combination of the old guess and the computed value, i.e.

$$q_{k,i+1} = \omega q_{k,i+1} + (1 - \omega) q_{k,i} \quad \text{with } 0 < \omega \leq 1.$$

If $\omega = 1$ then the computed new value of q_k is the new guess for the next iteration. If the weight ω is reduced then the new guess will depend more on the former guess and less on the computed value. As we will see, the damping factor ω is important for the speed of convergence. Typically there exists an optimal damping factor ω^* which minimizes the number of necessary iterations until sufficient convergence is achieved. The difference between the Gauss-Jacobi and the Gauss-Seidel method is only of minor importance. The former does not update any component of q until all new values of the q_k 's are computed.

Program 2.9 Gauss-Seidel iteration

```

program oligopoly2
    [.....]
    ! initialize q
    qold = 0.1d0
    damp = 0.7d0

    do iter = 1, 100

        QQ = sum(qold)
        q = 1d0/c*(QQ**(-1d0/eta) - 1d0/eta*QQ**(-1d0/eta-1d0)*qold)
        q = damp*q + (1d0-damp)*qold

        ! write to screen
        write(*,'(a, i5,2f10.4)')'Iter: ', iter, q(1), q(2)

        ! check for convergence
        if( all(abs(q-qold) < 1d-6) )then
            write(*,'(/a, 2f10.4)')' Output', q(1), q(2)
            stop
        endif

        ! update q
        qold = q
    enddo

end program

```

Table 2.1 Iteration of nonlinear equation system

| Iteration i | $\omega = 0.4$ | | $\omega = 0.7$ | | $\omega = 1$ | |
|---------------|----------------|--------|----------------|--------|--------------|--------|
| | q_1 | q_2 | q_1 | q_2 | q_1 | q_2 |
| 0 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 1 | 1.3132 | 0.9999 | 2.2232 | 1.6794 | 3.1331 | 2.3498 |
| 3 | 0.9194 | 0.7379 | 0.8282 | 0.6778 | 1.3985 | 1.1267 |
| 5 | 0.8503 | 0.6942 | 0.8393 | 0.6886 | 1.0239 | 0.8369 |
| 7 | 0.8410 | 0.6893 | 0.8396 | 0.6888 | 0.9071 | 0.7436 |
| 9 | 0.8397 | 0.6888 | 0.8396 | 0.6888 | 0.8653 | 0.7098 |
| 11 | 0.8396 | 0.6888 | | | 0.8495 | 0.6969 |
| 15 | 0.8396 | 0.6888 | | | 0.8411 | 0.6900 |
| 20 | | | | | 0.8394 | 0.6887 |
| 33 | | | | | 0.8396 | 6.8888 |

The latter uses new components as soon as the values have been computed. Consequently, we would use $q_{1,i+1}$ instead of $q_{1,i}$ in the computation of $q_{2,i+1}$ above.

Program 2.9 shows how the Cournot duopoly problem could be solved by iteration. Variable `qold` stores the initial guess while `q` is the new guess which is updated with the damping factor `damp`.

In each iteration the convergence of both quantities is checked. If convergence is not sufficient, the new guess is updated and a new iteration is started. Otherwise the program prints the results and stops. The iteration procedure is very stable and efficient. The initial guess is not really important but speed can be improved by choosing the optimal damping factor. Table 2.1 shows the results for three different values of the damping

factor ω . If we always take the computed value as the new guess, then it takes 33 iterations until sufficient convergence is reached. Lowering the damping factor initially reduces the required number of iterations down to nine at $\omega = 0.7$. If we decrease the damping factor further the number of required iterations increases again.

2.3 Function minimization

Beneath the problem of root-finding, minimizing functions constitutes a major problem in computational economics. Let

$$f(x) : \mathcal{X} \rightarrow \mathbb{R}$$

a function that maps an n -dimensional vector $x \in \mathcal{X} \subseteq \mathbb{R}^n$ into the real line \mathbb{R} . Then the corresponding minimization problem is defined as

$$\min_{x \in \mathcal{X}} f(x).$$

Note that minimization and maximization do not have to be considered separately as minimizing the function $-f(x)$ is exactly equal to its maximization. Obviously, minimization is very closely related to root-finding, as the derivative of a function becomes zero in a minimum. Consequently, if derivatives can be easily calculated, root-finding is a fair alternative to using a minimization method. However there are cases where using a minimization method is superior, e.g. if derivatives can't be calculated in a reasonable amount of time. In addition, non-differentiability of f eliminates the option of choosing a root-finding instead of a minimization algorithm. Last but not least, minimization methods often give us the opportunity to constrain the set of possible values for x . This is obviously not possible with root-finding algorithms. Figure 2.4 addresses this issue. Looking for the unconstrained minimum of f , we can see that we could use either a root-finding or a minimization algorithm. However, if we would like to find the minimum

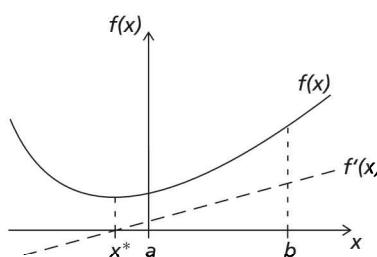


Figure 2.4 Constraint minimization

of f only on the interval $[a, b]$, root-finding will not help us as f' is positive on the whole interval. A constrained minimization method, however, could give us the actual solution a to this problem.

Example A household can consume two goods x_1 and x_2 . It values the consumption of those goods with the joint utility function

$$u(x_1, x_2) = x_1^{0.4} + (1 + x_2)^{0.5}.$$

Here x_2 acts as a luxury good, i.e. the household will only consume x_2 , if its available resources W are large enough. x_1 on the other hand is a normal good and will always be consumed. Naturally, we have to assume that $x_1, x_2 \geq 0$. With the prices for the goods being p_1 and p_2 , the household has to solve the optimization problem

$$\max_{x_1, x_2 \geq 0} x_1^{0.4} + (1 + x_2)^{0.5} \quad \text{s.t.} \quad p_1 x_1 + p_2 x_2 = W.$$

Note that there is no analytical solution to this problem.

Due to marginal utility increasing to ∞ with x_1 approaching zero, the optimal x_1 will always be strictly larger than zero. However, we might get a corner solution $x_2 = 0$. As the set of allowed values for x_2 is constrained, we will not be able to use a root-finding method to solve for the optimal choice of x_1 and x_2 . Hence, we need a minimization routine. When using a minimization algorithm with an equality constraint, it is always useful to first plug in the constraint into the utility function and therefore reduce the dimension of the optimization problem. We therefore reformulate the above problem to

$$\max_{x_2 \geq 0} \left[\frac{W - p_2 x_2}{p_1} \right]^{0.4} + (1 + x_2)^{0.5}.$$

This problem can be solved with various minimization routines.

2.3.1 THE GOLDEN-SEARCH METHOD

The *Golden-Search method* minimizes a one-dimensional function on the initially defined interval $[a, b]$. The idea behind this method is quite similar to the one of bisection search. Golden-Search, however, divides in each iteration i the interval $[a_i, b_i]$ into two subintervals by using the points $x_{i,1}$ and $x_{i,2}$ which satisfy the conditions

$$x_{i,2} - a_i = b_i - x_{i,1} \quad \text{and} \quad \frac{x_{i,1} - a_i}{b_i - a_i} = \frac{x_{i,2} - x_{i,1}}{b_i - x_{i,1}}.$$

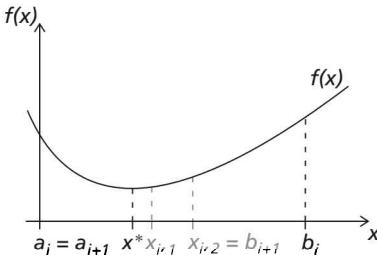


Figure 2.5 Golden search method for finding minima

Consequently, $x_{i,1}$ and $x_{i,2}$ are computed from

$$x_{i,j} = a_i + \omega_j(b_i - a_i) \quad \text{with} \quad \omega_1 = \frac{3 - \sqrt{5}}{2} \approx 0.382 \quad (2.4)$$

$$\text{and} \quad \omega_2 = 1 - \omega_1 = \frac{\sqrt{5} - 1}{2}.$$

We now compute the function values $f(x_{i,j})$ for $j = 1, 2$ and compare them. The next iteration's interval is then chosen according to

$$\begin{aligned} a_{i+1} &= a_i & \text{and} & \quad b_{i+1} = x_{i,2} & \quad \text{if} \quad f(x_{i,1}) < f(x_{i,2}), \\ a_{i+1} &= x_{i,1} & \text{and} & \quad b_{i+1} = b_i & \quad \text{otherwise.} \end{aligned}$$

The idea behind this iteration rule is quite simple. If $f(x_{i,1}) < f(x_{i,2})$, the lower values of f will be closer to $x_{i,1}$ than to $x_{i,2}$. Consequently, one chooses the interval $[a_i, x_{i,2}]$ as the new iteration interval and therefore rules out the greater values of f , see Figure 2.5.

Program 2.10 shows how to apply the Golden-Search method to the above problem. At the beginning of the program we choose values for the model parameter p_1, p_2 and W . We assume the price of the luxury good to be twice the price of the normal good and normalize the available resources $W = 1$. We then have to set the starting interval. Note that setting $a = 0$ restricts x_2 to be non-negative. b is finally initialized in a way that guarantees the consumption of good 1 to be positive for any $x_2 \in [a, b]$. In the iteration process we calculate $x_{i,1}$ and $x_{i,2}$ and the respective function values as shown in (2.4). Note that we always use the negative of the actual function value in order to assure that we maximize household's utility. Contrary to Section 2.2, we define our tolerance level as $|b_i - a_i|$. This is because we now have two values $x_{i,1}$ and $x_{i,2}$ in every iteration and don't know which one to choose as an approximation to our minimum x^* . Consequently, taking the interval width of $[a_i, b_i]$ as tolerance criterion insures that we can take any value within this interval and meet our tolerance criterion. If our criterion is satisfied, we print the minimum and the respective function value on the console. If not, we set the new iteration's interval $[a_{i+1}, b_{i+1}]$ according to the above iteration rule.

Taking a look at the output of the program, we find that $x_1 = 1$ and $x_2 = 0$ is the optimal solution to our optimization problem. This indicates that the constraint

Program 2.10 Golden-Search method in one dimension

```

program golden

implicit none
real*8, parameter :: p(2) = (/1d0, 2d0/)
real*8, parameter :: W = 1d0
real*8 :: a, b, x1, x2, f1, f2
integer :: iter

! initial interval and function values
a = 0d0
b = (W-p(1)*0.01d0)/p(2)

! start iteration process
do iter = 1, 200

    ! calculate x1 and x2 and function values
    x1 = a+(3d0-sqrt(5d0))/2d0*(b-a)
    x2 = a+(sqrt(5d0)-1d0)/2d0*(b-a)
    f1 = -(((W-p(2)*x1)/p(1))**0.4d0+(1d0+x1)**0.5d0)
    f2 = -(((W-p(2)*x2)/p(1))**0.4d0+(1d0+x2)**0.5d0)

    write(*,'(i4,f12.7)')iter, abs(b-a)

    ! check for convergence
    if(abs(b-a) < 1d-6)then
        write(*,'(/a,f12.7)')' x_1 = ',(W-p(2)*x1)/p(1)
        write(*,'(a,f12.7)')' x_2 = ',x1
        write(*,'(a,f12.7)')' u_ = ',-f1
        stop
    endif

    ! get new values
    if(f1 < f2)then
        b = x2
    else
        a = x1
    endif
enddo

end program

```

$x_2 > 0$ is actually binding for this price and resource combination. We can test this by extending the initial interval $[a, b]$ and setting $a = -0.9d0$ instead of 0. Doing this, Golden-Search returns the unconstrained optimum $x_1 \approx 1.61$ and $x_2 \approx -0.31$ of the optimization problem in our example. Note that, similarly in the bisection search method, the convergence speed of Golden-Search is linear.

2.3.2 BRENT'S AND POWELL'S ALGORITHMS

The problem with Golden-Search is its slow convergence. Therefore the function f needs to be called upon quite frequently. For well-behaved functions a more sophisticated algorithm is based on *Brent's method*. It relies on parabolic approximations \hat{f} of the actual function f . Finding the minimum of a parabola is quite easy. If the parabola is given by

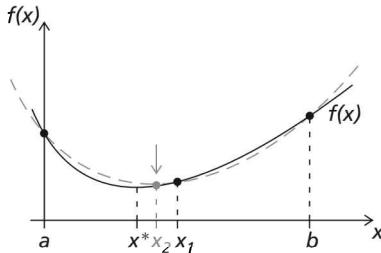


Figure 2.6 Brent's method for finding minima

$\hat{f}(x) = c_0 + c_1x + c_2x^2$, then its minimum is located at

$$x = -\frac{c_1}{2c_2}.$$

Figure 2.6 shows how Brent's method proceeds in finding a minimum. We start with the initial interval $[a, b]$ and compute the intersection point $x_1 = (a + b)/2$. We then compute the parameters c_i of a parabola that contains the three points $(a, f(a))$, $(b, f(b))$, and $(x_1, f(x_1))$ from

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & a & a^2 \\ 1 & x_1 & x_1^2 \\ 1 & b & b^2 \end{bmatrix}^{-1} \begin{bmatrix} f(a) \\ f(x_1) \\ f(b) \end{bmatrix}$$

The minimum of this parabola can be calculated and is denoted by x_2 . As x_2 lies between a and x_1 , we use the points a , x_2 , and x_1 in the next iteration step to again compute a parabola and find its minimum. The method is repeated until we have converged close enough to the true minimum x^* . Note that in every iteration step, we have to calculate only one function value, namely the value at the new point. To apply Golden-Search we needed two function values in every iteration step.

There also is a multidimensional extension of Brent's method called *Powell's algorithm*. This algorithm minimizes a multidimensional function by taking a set of n -dimensional, linearly independent direction vectors and minimizing f along the lines spanned by the n different vectors. Figure 2.7 demonstrates how Powell's algorithm works. We start at an initial guess x_1 . We then take our first optimization direction, represented by the arrow from x_1 and minimize the function along this direction. Given this minimum, we take the next direction and minimize along this one. We then again step back to the first direction etc. We iterate until we finally reach convergence. For the minimization along the different directions, we can again use Brent's method. There are certainly much more sophisticated algorithms to minimize functions that are based, for example, on higher-order approximations. Yet for the (limited) purposes of this book, Brent's method and Powell's algorithm have a couple of advantages. First of all, they do not rely on

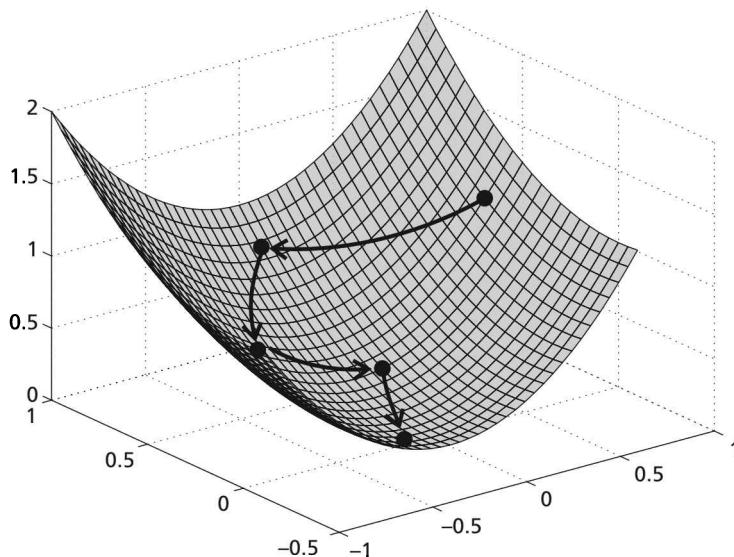


Figure 2.7 Powell's algorithm for finding minima in multiple dimensions

first or higher-order approximations of the function that should be minimized. This will be important especially in Part III, Advanced Computational Economics. Second, the algorithms are extremely stable and hardly run into computational problems. Third, they can easily deal with simple inequality constraints, which we will also exploit in several chapters including this simple example. And last but not least they are pretty minimalistic, meaning that they provide exactly the amount of computational power that we need, but do not overshoot in the sense that they can solve extremely complicated constraint optimization problems. In consequence, the algorithms do not need a lot of computing power and therefore probably solve problems faster than their more sophisticated counterparts. All in all we find these algorithms a very useful tool for the examples given in our book. Yet we encourage all readers of this book to explore the variety of minimization algorithms available in books and on the Internet on their own.

Both Brent's and Powell's methods are implemented in the subroutine `fminsearch`, which is included in the toolbox. Module 2.11m and Program 2.11 demonstrate their use on the above example.

The module `globals` again contains the structural parameters of the model, i.e. prices and the endowment of the household, as well as a function that returns the value of household's utility for any given input value x_2 . Note that we have to return the negative of the utility function, since Brent's and Powell's methods are to minimize functions. Minimizing the negative of a function obviously is equivalent to maximizing this function. Note further that if we wanted to define a multidimensional optimization problem, we would have declared the input variable x to the function as an assumed-size vector by using $x(:)$. The output value would still be a one-dimensional object.

Module 2.11m Brent and Powell for finding minima

```

module globals

    implicit none
    real*8, parameter :: p(2) = (/1d0, 2d0/)
    real*8, parameter :: W = 1d0

contains

    ! the utility function
    function utility(x)

        implicit none
        real*8, intent(in) :: x
        real*8 :: utility

        utility = -((W-p(2)*x)/p(1))**0.4d0+(1d0+x)**0.5d0

    end function

end module globals

```

Program 2.11 Brent and Powell for finding minima

```

include "prog02_11m.f90"

program brentmin

    use globals
    use toolbox

    implicit none
    real*8 :: x, f, a, b

    ! initial interval and function values
    a = 0d0
    b = (W-p(1)*0.01d0)/p(2)

    ! set starting point
    x = (a+b)/2d0

    ! call minimizing routine
    call fminsearch(x, f, a, b, utility)

    ! output
    write(*, '(/a,f12.7)')' x_1 = ', (W-p(2)*x)/p(1)
    write(*, '(/a,f12.7)')' x_2 = ', x
    write(*, '(/a,f12.7)')' u = ', -f

end program

```

Program brentmin then executes the minimization process. It therefore uses both the module globals as well as the module toolbox. The subroutine fminsearch takes five input arguments. The first is the initial guess x. After the subroutine has successfully found a minimum, the minimum point will be stored again in the variable x. f does not have to be initialized. The subroutine will store the function value in the minimum in

this scalar upon successful convergence. The next two arguments a and b are the interval bounds on which the subroutine should search for a minimum. Note that these interval borders also serve as simple inequality constraints. In our case, for example, we set the lower bound on the variable x_2 to 0. Consequently, the optimizer will restrict the choice set on the input variable x to values that are greater or equal to zero, even if a lower function value could be obtained by setting x_2 to a value lower than zero. Verify this by loosening the lower constraint on the input value x by, for example, setting $a = -1\text{d}0$. What is the result of the optimization procedure in this case? What happens when you set a at even lower values? The upper interval border in our case is determined by the fact that consumption of the first good x_1 should not be negative. The last argument given to the subroutine is the specific function we want to minimize, in our case the function utility. The result of the optimization process is finally printed on the console.

2.3.3 THE PROBLEM OF LOCAL AND GLOBAL MINIMA

Unfortunately, there is no guarantee that a minimization method will find the global minimum of a function. It might well be that one ends up with a local minimum. Figure 2.8 shows this problem with Golden-Search. We can see that f has a local minimum on the left and the global minimum on the right side of the optimization interval $[a, b]$. When we perform the first step of Golden-Search, we calculate the points x_1 and x_2 and the respective function values. For a next iteration's optimization interval, we would then choose $[a, x_1]$, as $f(x_1) < f(x_2)$. Obviously with this step we have already eliminated the chance of finding the global minimum, as after the first step we concentrate on the left-hand side of the interval $[a, b]$.

An easily implementable approach to overcome this problem is to first divide the original interval $[a, b]$ into n subintervals $[x_i, x_{i+1}]$ of equal size with $x_1 = a$ and $x_{n+1} = b$. For every subinterval, one then performs a full minimization procedure to find the minimum x_i^* of f on the interval $[x_i, x_{i+1}]$. Now we have a set of minima $\{x_i^*\}_{i=1}^n$. Our global minimum finally is the value x_i^* with the smallest function value $f(x_i^*)$.

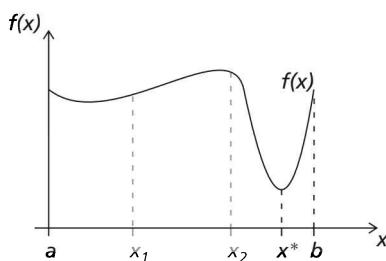


Figure 2.8 The problem of local minima

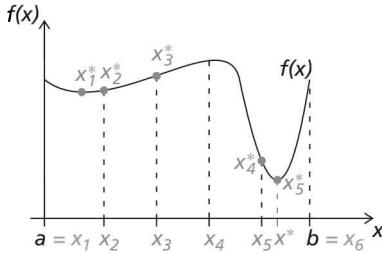


Figure 2.9 Subinterval division for solving the problem of local minima

Figure 2.9 demonstrates the approach. We used a division into five intervals in this case. The grey dots mark the local minima on the subintervals. We can see that the global minimum is also contained in this set. Hence, with this procedure, we are able to find the global minimum of a function. Note, however, that the procedure is quite costly, as we have to perform five minimizations instead of only one. Consequently it should only be used when one is sure that the problem of local optima may arise.

2.4 Numerical integration

In many economic applications it is necessary to compute the definite integral of a real-valued function f with respect to a ‘weight’ function w over an interval $[a, b]$, i.e.

$$I(f) = \int_a^b w(x)f(x) dx.$$

The weight function may be the identity function $w(x) = 1$, so that the integral represents the area under the function f in the interval. In other applications the weight function could also be the probability density function of a continuous random variable \tilde{x} with support $[a, b]$, so that $I(f)$, represents the expected value of $f(\tilde{x})$.

In the following we discuss so-called numerical *quadrature methods*. In order to approximate the above integral, quadrature methods choose a discrete set of quadrature nodes x_i and appropriate weights w_i in a way that

$$I(f) \approx \sum_{i=0}^n w_i f(x_i).$$

The integral is therefore approximated by the sum of function values at the nodes $a \leq x_0 < \dots < x_n \leq b$ and the respective quadrature weights w_i . Of course, the approximation improves with a rising number n .

Different quadrature methods only differ with respect to the chosen nodes x_i and weights w_i . In the following we will concentrate on the *Newton-Cotes* method and the *Gaussian quadrature* method. The former approximates the integrand f between nodes using low-order polynomials and adds the integrals of the polynomials to estimate the integral of f . The latter methods choose the nodes and weights in order to match specific moments (such as expected value, variance etc.) of the approximated function.

2.4.1 SUMMED NEWTON-COTES METHODS

In this section we set $w(x) = 1$. A weighted integral can then simply be computed by approximating

$$\int_a^b f(x) dx.$$

Summed Newton-Cotes formulas partition the interval $[a, b]$ into n subintervals of equal length h by computing the quadrature nodes

$$x_i := a + ih, \quad i = 0, 1, \dots, n, \quad \text{with} \quad h := \frac{b - a}{n}.$$

Next, the function f on every subinterval $[x_i, x_{i+1}]$ is approximated with a polynomial of degree k . Finally, we integrate the approximating polynomial on every subinterval and sum up all the resulting sub-areas. Figure 2.10 shows this approach for $k = 0$ and $k = 1$.

Summed rectangle rule If $k = 0$, the interpolating functions are polynomials of degree 0, i.e. constant functions. We can write any area below this constant function on the interval $[x_i, x_{i+1}]$ as

$$I_{[x_i, x_{i+1}]}^{(0)}(f) = (x_{i+1} - x_i)f(x_i) = hf(x_i),$$

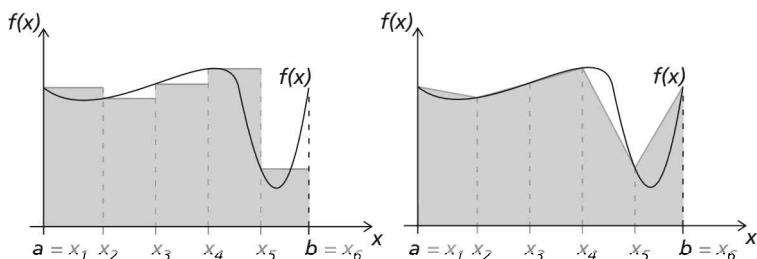


Figure 2.10 Summed Newton-Cotes formulas for $k = 0$ and $k = 1$

which is the area of a rectangle. The degree 0 formula is therefore called the *summed rectangle rule*, the explicit form of which is given by

$$I^{(0)}(f) = \sum_{i=0}^{n-1} h f(x_i).$$

The weights of the summed rectangle rule consequently are

$$w_i = h \quad \text{for } i = 0, \dots, n-1 \quad \text{and} \quad w_n = 0.$$

Summed trapezoid rule If $k = 1$, the function f in the i -th subinterval $[x_i, x_{i+1}]$ is approximated by the line segment passing through the points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$. The area under this line segment is the surface of a trapezoid, i.e.

$$I_{[x_i, x_{i+1}]}^{(1)}(f) = h f(x_i) + \frac{1}{2} h [f(x_{i+1}) - f(x_i)] = \frac{h}{2} [f(x_i) + f(x_{i+1})].$$

It is immediately clear that the *summed trapezoid rule* improves the approximation of $I(f)$ compared to the summed rectangle rule discussed above. Summing up the areas of the trapezoids across subintervals yields the approximation

$$I^{(1)}(f) = \frac{h}{2} \left\{ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + f(b) \right\}$$

of the whole integral $I(f)$. The weights of the rule therefore are

$$w_0 = w_n = \frac{h}{2} \quad \text{and} \quad w_i = h \quad \text{for } i = 1, \dots, n-1. \quad (2.5)$$

The summed rectangle and trapezoid rule are simple and robust. Hence, computation does not need much effort. In addition, the accuracy of the approximation of $I(f)$ increases with rising n . It is also clear that the trapezoid rule will exactly compute the integral of any first-order polynomial, i.e. a line.

Program 2.12 shows how to apply the summed trapezoid rule to the function $\cos(x)$. We thereby first declare all the variables needed. Special attention should be devoted to the declaration of x , w , and f . These arrays will store the quadrature nodes x_i , the weights w_i , and the function values at the nodes $f(x_i)$, respectively. We then first compute h and the nodes $x_i = a + ih$ and the weights w_i as in (2.5). With the respective function values, the approximation to the integral

$$\int_0^2 \cos(x) dx = \sin(2) - \sin(0)$$

Program 2.12 Summed trapezoid rule with $\cos(x)$

```

program NewtonCotes
    implicit none
    integer, parameter :: n = 10
    real*8, parameter :: a = 0d0, b = 2d0
    real*8 :: h, x(0:n), w(0:n), f(0:n)
    integer :: i

    ! calculate quadrature nodes
    h = (b-a)/dble(n)
    x = (/ (a + dble(i)*h, i=0,n) /)

    ! get weights
    w(0) = h/2d0
    w(n) = h/2d0
    w(1:n-1) = h

    ! calculate function values at nodes
    f = cos(x)

    ! Output numerical and analytical solution
    write(*,'(a,f10.6)')' Numerical: ',sum(w*f, 1)
    write(*,'(a,f10.6)')' Analytical: ',sin(2d0)-sin(0d0)

end program

```

is the given by `sum(w*f, 1)`. Note that the approximation of the integral is quite poor. If we increase the number of quadrature nodes n increases the accuracy, however, we need about 600 nodes in order to perfectly match numerical and analytical results on six digits.

Summed Simpson rule Finally if $k = 2$, the integrand f in the i -th subinterval $[x_i, x_{i+1}]$ is approximated by a second-order polynomial function $c_0 + c_1x + c_2x^2$. Now three graph points are required to specify the polynomial parameters c_i in the subintervals. Given the points

$$f(x_i) = c_0 + c_1x_i + c_2x_i^2 \quad (2.6)$$

$$f\left(\frac{x_i + x_{i+1}}{2}\right) = c_0 + c_1\left(\frac{x_i + x_{i+1}}{2}\right) + c_2\left(\frac{x_i + x_{i+1}}{2}\right)^2 \quad (2.7)$$

$$f(x_{i+1}) = c_0 + c_1x_{i+1} + c_2x_{i+1}^2 \quad (2.8)$$

we can approximate the integral of the true function f by the integral of the quadratic function

$$\int_{x_i}^{x_{i+1}} (c_0 + c_1x + c_2x^2) dx.$$

After substituting (2.6) to (2.8) in order to determine the polynomial parameters, one can compute the integral of the quadratic function in the subinterval $[x_i, x_{i+1}]$ as

$$I_{[x_i, x_{i+1}]}^{(2)}(f) = \frac{h}{6} \left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1}) \right].$$

Summing up the different areas on the subintervals yields

$$I^{(2)}(f) = \frac{h}{6} \left\{ f(a) + 2 \sum_{i=1}^{n-1} f(x_i) + 4 \sum_{i=0}^{n-1} f\left(\frac{x_i + x_{i+1}}{2}\right) + f(b) \right\}.$$

The summed Simpson rule is almost as simple to implement as the trapezoid rule. If the integrand is smooth, Simpson's rule yields an approximation error that with rising n falls twice as fast as that of the trapezoid rule. For this reason Simpson's rule is usually preferred to that of the trapezoid and rectangle rule. Note, however, that the trapezoid rule will often be more accurate than Simpson's rule if the integrand exhibits discontinuities in its first derivative, which can occur in economic applications exhibiting corner solutions.

Of course, summed Newton-Cotes rules also exist for higher-order piecewise polynomial approximations, but they are more difficult to work with and thus rarely used.

2.4.2 GAUSSIAN QUADRATURE

Contrary to the Newton-Cotes methods, *Gaussian quadrature* explicitly allows for non-constant weight functions $w(x)$. In addition, such methods don't try to approximate the function f , but rather determine weights and nodes such that particular moments with respect to the weighting function w are matched perfectly.

Gauss-Legendre quadrature Suppose again, for the moment, that $w(x) = 1$. The Gauss-Legendre quadrature nodes $x_i \in [a, b]$ and weights w_i are computed in a way that they satisfy the *2n + 2 moment-matching conditions*

$$\int_a^b x^k dx = \sum_{i=0}^n w_i x_i^k \quad \text{for } k = 0, 1, \dots, 2n + 1. \quad (2.9)$$

With the above conditions holding, we can write every integral over a polynomial

$$p(x) = c_0 + c_1 x + \dots + c_m x^m$$

with degree $m \leq 2n + 1$ as

$$\begin{aligned}\int_a^b p(x) dx &= c_0 \underbrace{\int_a^b 1 dx}_{=\sum_{i=0}^n w_i} + c_1 \underbrace{\int_a^b x dx}_{=\sum_{i=0}^n w_i x_i} + \dots + c_m \underbrace{\int_a^b x^m dx}_{=\sum_{i=0}^n w_i x_i^m} \\ &= \sum_{i=0}^n w_i [c_0 + c_1 x_i + \dots + c_m x_i^m] = \sum_{i=0}^n w_i p(x_i).\end{aligned}$$

Consequently, quadrature nodes and weights that satisfy the moment-matching conditions in (2.9) are able to integrate any polynomial p of degree $m \leq 2n + 1$ exactly.

Calculating the nodes and weights of the Gauss-Legendre quadrature formula is not so easy. One method is to set up the nonlinear equation system defined in (2.9). For $n = 2$, for example, we have

$$\begin{aligned}w_0 + w_1 + w_2 &= b - a = \int_a^b 1 dx, \\ w_0 x_0 + w_1 x_1 + w_2 x_2 &= \frac{1}{2} (b^2 - a^2) = \int_a^b x dx, \\ &\vdots \\ w_0 x_0^5 + w_1 x_1^5 + w_2 x_2^5 &= \frac{1}{6} (b^6 - a^6) = \int_a^b x^5 dx.\end{aligned}$$

This nonlinear equation system can be solved by a root-finding algorithm like `fzero`. However, there is a more efficient, but also less intuitive way to calculate quadrature nodes and weights of the Gauss-Legendre quadrature implemented in the subroutine `legendre` in the toolbox. Program 2.13 demonstrates how to use it. The subroutine takes two `real*8` arguments defining the left and right interval borders a and b . In addition, we have to pass two arrays of equal length to the routine. The first of these will be filled with the nodes x_i , whereas the second will be given the weights w_i . After having calculated the function values $f(x_i)$, we can again calculate the numerical approximation of the integral like in Program 2.11. Here, with 10 quadrature nodes, we already match the analytical integral value by six digits. Note that we needed about 600 nodes with the trapezoid rule.

Gauss-Hermite quadrature Next we consider a specific case with a weight function $w(x) \neq 0$. Suppose that the weight function is equal to the density function of the standard normal distribution, i.e.

$$w(x) = \frac{1}{\sqrt{2\pi}} \exp(-0.5x^2).$$

Program 2.13 Gauss-Legendre quadrature with $\cos(x)$

```

program GaussLegendre

use toolbox

implicit none
integer, parameter :: n = 10
real*8, parameter :: a = 0d0, b = 2d0
real*8 :: x(0:n), w(0:n), f(0:n)

! calculate nodes and weights
call legendre(a, b, x, w)

! calculate function values at nodes
f = cos(x)

! Output numerical and analytical solution
write(*,'(a,f10.6)')' Numerical: ',sum(w*f, 1)
write(*,'(a,f10.6)')' Analytical: ',sin(2d0)-sin(0d0)

end program

```

This results in the *Gauss-Hermite* quadrature method. If \tilde{x} is a normally distributed random variable we derive x_i and w_i from

$$E(\tilde{x}^m) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-0.5x^2) x^m dx = \sum_{i=0}^n w_i x_i^m = I^{\text{GH}}(x^m)$$

with $m \leq 2n + 1$. Consequently, with a Gauss-Hermite quadrature, we are able to perfectly match the first $2n + 1$ moments of a normally distributed random variable \tilde{x} .

An approximation procedure for normally distributed random variables is included in the toolbox. The procedure `normal_discrete(x, w, mu, sig2)` is exactly based on the Gauss-Hermite quadrature method. The subroutine receives four input arguments, the last of which are expected value and variance of the normal distribution that should be approximated. The routine then stores appropriate nodes x_i and weights w_i in the arrays `x` and `w` (that should be of same length) which can be used to compute moments of a normally distributed random variable \tilde{x} with expectation `mu` and variance `sig2`. For applying this subroutine we consider the following example:

Example Consider an agricultural commodity market, where planting decisions are based on the price expected at harvest

$$A = 0.5 + 0.5E(p), \quad (2.10)$$

with A denoting acreage supply and $E(p)$ defining expected price. After the acreage is planted, a normally distributed random yield $y \sim N(1, 0.1)$ is realized, giving rise to the quantity $q^s = Ay$ which is sold at the market clearing price $p = 3 - 2q^s$.

In order to solve this system we substitute

$$q^s = [0.5 + 0.5E(p)]y$$

and therefore

$$p = 3 - 2[0.5 + 0.5E(p)]y.$$

Taking expectations on both sides leads to

$$E(p) = 3 - 2[0.5 + 0.5E(p)]E(y)$$

and therefore $E(p) = 1$. Consequently, equilibrium acreage is $A = 1$. Finally, the equilibrium price distribution has a variance of

$$\text{Var}(p) = 4[0.5 + 0.5E(p)]^2 \text{Var}(y) = 4\text{Var}(y) = 0.4.$$

Suppose now that the government introduces a price support program which guarantees each producer a minimum price of 1. If the market price falls below this level, the government pays the producer the difference per unit produced. Consequently, the producer now receives an effective price of $\max(p, 1)$ and the expected price in (2.10) is then calculated via

$$E(p) = E[\max(3 - 2Ay, 1)]. \quad (2.11)$$

The equilibrium acreage supply finally is the supply A that fulfils (2.10) with the above price expectation. Again, this problem cannot be solved analytically.

In order to compute the solution of the above acreage problem, one has to use two subroutines from the toolbox. `normal_discrete`, on the one hand, provides the method to discretize the normally distributed random variable y . The solution to (2.10) can, on the other hand, be calculated by using the method `fzero`. Module 2.14m and Program 2.14 show how to do this.

Due to space restrictions, we do not show the whole module and program but only the basic parts. For running the program, we need a module `globals` in which we store the expectation `mu` and variance `sig2` of the normally distributed random variable `y` as well as the minimum price `minp` guaranteed by the government. In addition, the module stores the quadrature nodes `y` and weights `w` obtained by discretizing the normal distribution for `y`. Beneath these central variables, the module `globals` contains the function `market` that calculates the market equilibrium condition in (2.10). This function only gets `A` as an input. From `A` we can calculate the expected price `E(p)` by means of (2.11) as well as the quadrature nodes and weights and finally the market clearing condition.

Module 2.14m Agricultural problem

```

module globals

    implicit none
    real*8, parameter :: mu = 1d0, sig2 = 0.1d0, minp = 1d0
    integer, parameter :: n = 10
    real*8 :: y(0:n), w(0:n)

contains

    function market(A)

        implicit none
        real*8, intent(in) :: A
        real*8 :: market
        real*8 :: Ep

        ! calculate expected price
        Ep = sum(w*max(3d0-2d0*A*y, minp))

        ! get equilibrium equation
        market = A - (0.5d0+0.5d0*Ep)

    end function

end module globals

```

Program 2.14 Agricultural problem

```

include "prog02_14m.f90"

program agriculture
    [.....]
    ! discretize y
    call normal_discrete(y, w, mu, sig2)

    ! initialize variables
    A = 1d0

    ! get optimum
    call fzero(A, market, check)

    ! get expectation and variance of price
    Ep = sum(w*max(3d0-2d0*A*y, minp))
    Varp = sum(w* (max(3d0-2d0*A*y, minp) - Ep)**2)
    [.....]
end program

```

In the program itself, we first discretize the distribution for y by means of the subroutine `normal_discrete`. This routine receives the expected value and variance of the normal distribution and stores the respective nodes y and weights w in two arrays of same size. Next we set a starting guess for acreage supply. We then let `fzero` find the root of the function `market` that calculates the market equilibrium condition in (2.10). In order to show the effects of a minimum price, we first set the minimum price guarantee `minp` to a large negative value, say -100 . This large negative minimum price will never be binding,

hence, the outcome is exactly the same as in the model without a price guarantee, see the above example description. If we now set the minimum price at 1, equilibrium acreage supply increases by about 9.7 per cent. This is because the expected effective price for the producer rises from the minimum price guarantee. In addition, the uncertainty for the producers is reduced, since the variance of the price distribution decreases from 0.4 to about 0.115, which is again due to the fact that the possible price realization now has a lower limit of 1.

2.5 Random variables, distributions, and simulation

Section 2.4 showed us a way to deal with normally distributed random variables by means of Gauss-Hermite integration schemes. This section is devoted to random variables more generally: how to calculate their cumulative distributions and densities and how to simulate a series of realizations of a random variable that follows a particular distribution. To keep things as accessible as possible, we will not dive deeply into the measure-theoretic definitions of random variables, but rather choose a more intuitive route.

2.5.1 RANDOM VARIABLES AND THEIR DISTRIBUTION

We say that \tilde{x} is a random variable if it measures the outcome of some underlying experiment Ω . The outcome ω of the experiment is uncertain and follows some probability distribution P . The random variable maps each outcome ω of the experiment into something we can measure, for example, a real number, an interval on the real line, or some subset of the multidimensional space \mathbb{R}^n . As an illustration let's assume we were shooting with bow and arrows on a target. While the outcome of such a shot is most likely uncertain, it is also very complex. In fact, shooting the arrow starts a complex physical process that causes the arrow to hit the target at a certain point in space and time. When carrying out such an experiment we are, however, not interested in the underlying physical mechanics, but rather in how far our arrow is away from the centre of the target. When we assume that we get 10 points for a bull's eye, 1 for the outer ring, and 0 in case we miss the target, then a random variable \tilde{x} maps the exact physical outcome of shooting an arrow into a discrete set of points $\mathcal{X} = \{0, 1, \dots, 10\}$. What will be the measured outcome of this experiment is highly uncertain and depends on the distance between shooter and target as well as our shooting skills. We might, however, attach probabilities to each outcome. We say that $P(\tilde{x} = z)$ with $z \in \mathcal{X}$ is the probability that we obtain z points when shooting the arrow. As the measured outcome of the experiment is a discrete set, we call \tilde{x} a *discrete random variable*. The probability measure obviously needs to satisfy

$$\sum_{z \in \mathcal{X}} P(\tilde{x} = z) = 1.$$

Instead of classifying the outcome of shooting with bow and arrows according to a discrete point system, we could also measure exactly how far the point where the arrow hit the target is away from the centre. Let's assume we would measure this in centimeters (cm). In this case, our random variable \tilde{x} maps the outcome of the experiment into a non-negative number, i.e. $\mathcal{X} = \mathbb{R}_0^+$. We can associate probabilities to the event that our shot is at most z cm away from the bull's eye. The corresponding probability is $P(\tilde{x} \leq z)$. As the outcome of the experiment then is continuous, we call \tilde{x} a *continuous random variable*. Of course, since there are no negative distances and our shot has to go somewhere in the end, we have

$$P(\tilde{x} \leq z) = 0 \text{ if } z < 0 \quad \text{and} \quad P(\tilde{x} \leq \infty) = 1.$$

Cumulative distribution function The distribution of a (one-dimensional) random variable is typically described by means of a cumulative distribution function $\Phi(z)$. This function informs us about the probability that the random variable \tilde{x} takes on a value smaller than or equal to the value z . We then obviously have

$$\Phi(z) = \begin{cases} P(\tilde{x} \leq z) & \text{if } \tilde{x} \text{ is continuous, and} \\ \sum_{y \in \mathcal{X}, y \leq z} P(\tilde{x} = y) & \text{if } \tilde{x} \text{ is discrete.} \end{cases}$$

The cumulative distribution function has to satisfy the properties

$$\lim_{z \rightarrow -\infty} \Phi(z) = 0 \quad \text{and} \quad \lim_{z \rightarrow \infty} \Phi(z) = 1$$

as well as

$$\Phi(z_1) \leq \Phi(z_2) \quad \text{if} \quad z_1 \leq z_2.$$

This means that the cumulative distribution function moves from zero to one and is weakly monotonically increasing in z . Knowing the cumulative distribution function of a random variable, we can not only characterize the probability that the random variable takes on a value smaller than or equal to z , but also that the value is larger than z or that the value falls into a certain interval. To this end, we use the relationships

$$P(\tilde{x} > z) = 1 - \Phi(z) \quad \text{and} \quad P(z_l < \tilde{x} \leq z_u) = \Phi(z_u) - \Phi(z_l).$$

Probability density function When a random variable is continuous, we can also characterize its distribution by means of the probability density function. The density of a random variable can be seen as the continuous analogue to the probability distribution of

a discrete variable. It indicates in which part of the real line \mathbb{R} realizations of \tilde{x} are more or less likely to occur. The probability density function $\phi(z)$ of a continuous random variable is the derivative of the cumulative distribution function,² i.e. we can write

$$\phi(z) = \Phi'(z) \quad \text{or} \quad \Phi(z) = \int_{-\infty}^z \phi(y) dy.$$

Particular distributions implemented in the toolbox The toolbox contains functions for calculating the probability density function or probability distribution as well as the cumulative distribution function of a particular set of distributions, which we will use throughout this book. In the following we briefly describe these distributions, their probability density functions or probability distributions, as well as the parameters that govern their shape and moments. We refrain, however, from characterizing cumulative distribution functions as this is typically very complicated for most distributions. However, the cumulative distribution functions (or approximations thereof) are also implemented in the toolbox.

1. *Uniform distribution:* A random variable that is uniformly distributed on the interval $[a, b]$ has the probability density function

$$\phi(z) = \begin{cases} \frac{1}{b-a} & \text{if } z \in [a, b], \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The mean and variance of the uniform distribution are

$$E(\tilde{x}) = \frac{a+b}{2} \quad \text{and} \quad Var(\tilde{x}) = \frac{(b-a)^2}{12}.$$

2. *Normal distribution:* A normally distributed random variable has the density

$$\phi(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(z-\mu)^2}{2\sigma^2}\right)$$

with moments

$$E(\tilde{x}) = \mu \quad \text{and} \quad Var(\tilde{x}) = \sigma^2.$$

² Whenever the distribution function is differentiable.

3. *Log-normal distribution:* A random variable is log-normally distributed, when its log follows a normal distribution. The corresponding probability density function is

$$\phi(z) = \begin{cases} \frac{1}{z\sqrt{2\pi\sigma^2}} \cdot \exp\left(-\frac{(\log(z)-\mu)^2}{2\sigma^2}\right) & \text{if } z \geq 0, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Its moments are

$$E(\tilde{x}) = \exp\left(\mu + \frac{\sigma^2}{2}\right) \quad \text{and} \quad \text{Var}(\tilde{x}) = \exp(2\mu + \sigma^2) \cdot [\exp(\sigma^2) - 1].$$

4. *Gamma distribution:* The Gamma distribution is a generalization of the exponential and the χ^2 distribution. It features two parameters $\alpha > 0$ and $\beta > 0$ that govern its shape and scale. The probability density function is

$$\phi(z) = \begin{cases} \frac{\beta^\alpha z^{\alpha-1} \exp(-z\beta)}{\Gamma(\alpha)} & \text{if } z \geq 0, \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

$\Gamma(\alpha)$ denotes the Gamma function. Mean and variance of the distribution are

$$E(\tilde{x}) = \frac{\alpha}{\beta} \quad \text{and} \quad \text{Var}(\tilde{x}) = \frac{\alpha}{\beta^2}.$$

5. *Beta distribution:* The beta distribution is a continuous probability distribution on the interval $[0, 1]$ with two parameters $p, q > 0$. Its probability density function is

$$\phi(z) = \begin{cases} \frac{z^{p-1}(1-z)^{q-1}}{B(p,q)} & \text{if } z \in [0, 1], \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

$B(p, q)$ denotes the beta function. The mean and variance of the beta distribution are

$$E(\tilde{x}) = \frac{p}{p+q} \quad \text{and} \quad \text{Var}(\tilde{x}) = \frac{pq}{(p+q+1)(p+q)^2}.$$

6. *Bernoulli distribution:* The outcome of a Bernoulli-distributed random variable is discrete and can only take on two values, 0 and 1. The value of 1 occurs with probability p . The most prominent example of a Bernoulli-distributed random variable is the outcome of tossing (a fair) coin, where $p = 0.5$ and the outcome of 1 could be heads. The probability distribution of the Bernoulli distribution therefore reads

$$P(\tilde{x} = 0) = 1 - p \quad \text{and} \quad P(\tilde{x} = 1) = p.$$

Mean and variance of a Bernoulli-distributed random variable are

$$E(\tilde{x}) = p \quad \text{and} \quad \text{Var}(\tilde{x}) = p(1 - p).$$

7. *Binomial distribution:* A binomial distributed random variable is the sum of n independent Bernoulli-distributed random variables. One can think, for example, of tossing a coin n times and counting the number of heads that occurred. The corresponding probability distribution is

$$P(\tilde{x} = z) = \binom{n}{z} p^z (1 - p)^{n-z} \quad \text{for } z \in \{0, 1, \dots, n\}.$$

$\binom{n}{z}$ thereby denotes the binomial coefficient. Mean and variance of the binomial distribution are

$$E(\tilde{x}) = np \quad \text{and} \quad \text{Var}(\tilde{x}) = np(1 - p).$$

The probability density functions and cumulative distribution functions of these seven particular distributions are implemented in the toolbox in functions that end on `PDF` and `CDF`, respectively. Program 2.15 shows how these functions can be called. In the example of a uniform distribution, we first set interval bound a and b and construct an equidistant set of points around this interval. We store these points in an array z of length NN . We then successively call the functions `uniformPDF` and `uniformCDF` at the respective points $z(ii)$ with the parameters a and b . We plot the resulting probability density and cumulative distribution function by means of the plotting routines of the toolbox. We can repeat the same procedure with every of the aforementioned distributions. There are two peculiarities one has to take care of. First, when we call the distribution functions of the log-normal distribution, the input parameters `mu` and `sigma` govern the actual mean and variance of the log-normal distribution, i.e. $\text{mu} = E(\tilde{x})$ and $\text{sigma} = \text{Var}(\tilde{x})$ and not the mean and variance μ and σ^2 of the underlying normal distribution. Second, the probability distribution as well as the cumulative distribution of the Bernoulli and Binomial distribution are only defined on integer and not real numbers. Figure 2.11 shows the outcomes of the program for selected distributions.

2.5.2 SIMULATING REALIZATIONS OF RANDOM VARIABLES

In practice, it is often useful to simulate a series of independent realizations of a random variable to find out more about its properties or to use it for example in Monte Carlo simulations. In our terminology of shooting with bow and arrows, this would mean that we shoot on the same target n times and for each shot record the outcome $x_i, i = 1, \dots, n$. Assuming that we don't improve our shooting skills nor get tired in the process of doing this, $\{x_i\}_{i=1}^n$ is a series of independent draws from the underlying random variable \tilde{x} .

Program 2.15 Density and cumulative distribution functions

```

program distributions
[.....]
! uniform distribution
a = -1d0
b = 1d0
call grid_Cons_Equi(z, a-0.5d0, b+0.5d0)
do ii = 0, NN
    dens(ii) = uniformPDF(z(ii), a, b)
    dist(ii) = uniformCDF(z(ii), a, b)
enddo

call plot(z, dens, legend="Density")
call plot(z, dist, legend="Distribution")
call execplot(title="Uniform Distribution")
[.....]
! log-normal distribution
mu = 1d0
sigma = 0.25d0

call grid_Cons_Equi(z, 0d0, mu+5d0*sqrt(sigma))
do ii = 0, NN
    dens(ii) = log_normalPDF(z(ii), mu, sigma)
    dist(ii) = log_normalCDF(z(ii), mu, sigma)
enddo

call plot(z, dens, legend="Density")
call plot(z, dist, legend="Distribution")
call execplot(title="Log-Normal Distribution")
[.....]
! binomial distribution
p = 0.25d0
n = 12
do ii = 0, n
    z(ii) = dble(ii)
    dens(ii) = binomialPDF(ii, n, p)
    dist(ii) = binomialCDF(ii, n, p)
enddo
[.....]
end program

```

It is not straightforward to generate draws from random variables on a computer. Fortran provides a so-called *pseudo-random number generator* that simulates a sequence of numbers whose properties approximate those of a sequence of uniformly distributed random numbers on the interval $[0, 1]$. This random number generation algorithm is provided in the intrinsic subroutine **random_number(x)**, which fills up an array **x** with pseudo-random numbers. The resulting series of numbers is called pseudo-random, as the exact outcome of the number generation typically only depends on one or very few initial values, which is called the *seed*. For a specific value of the seed, the sequence of numbers stored in **x** is in fact deterministic. This means that when a certain seed is specified and we run a program in which we just want to simulate a series of random numbers by means of the subroutine **random_number** several times, then the series would be identical every time we run the program, see the exercise section of Chapter 1. The only

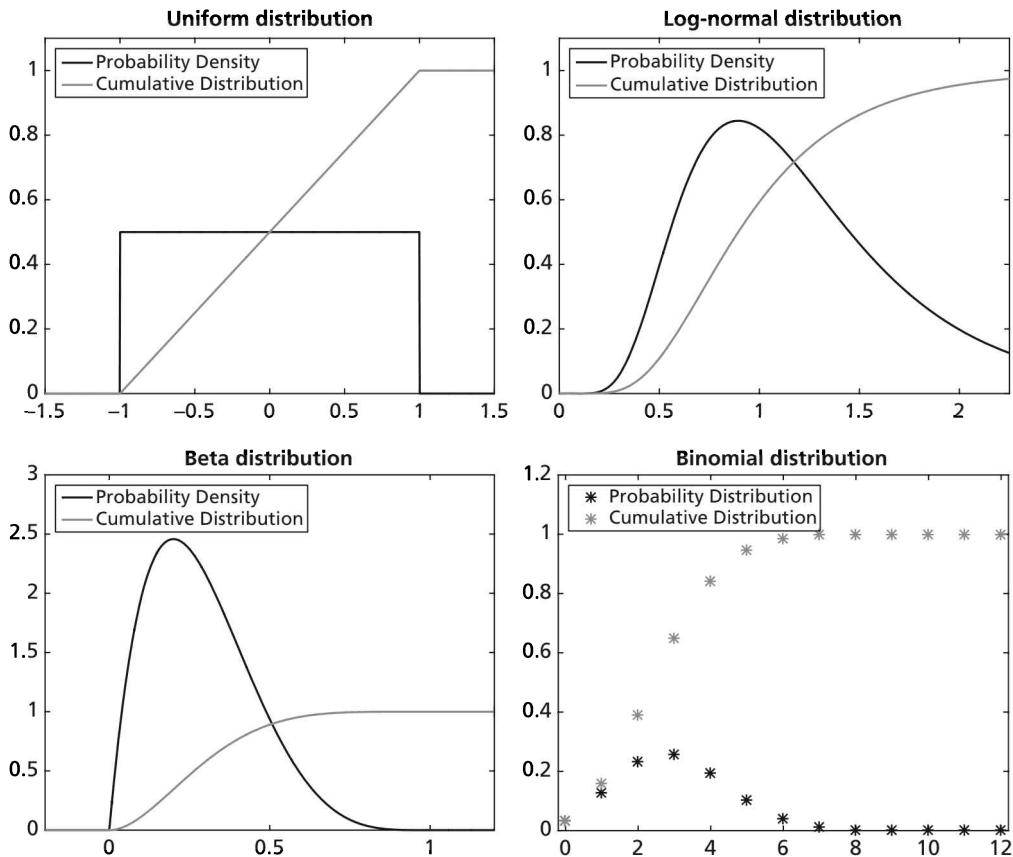


Figure 2.11 Selected density and cumulative distribution functions

way to generate a ‘new’ series of random numbers is to give the seed a new value, where this value should typically contain some form of randomness for the whole procedure to make sense.

All of this is taken care of in the toolbox. In fact, the subroutine

```
simulate_uniform(x, a, b, fixed)
```

directly exploits the intrinsic subroutine `random_number` to simulate a series of independent uniformly distributed random variables on the interval $[a, b]$, where in the most simple case we have $a = 0$ and $b = 1$, and stores the result in an array `x`. The last input argument to this subroutine `fixed` is optional. When it is omitted, the subroutine resets the seed of the random number generator according to some algorithm that is based on reading the current system time, meaning that every time the program is executed a new random seed will be set and the sequences stored in `x` will differ. If, however, the logical variable `fixed` is specified with value `.true.`, then the seed will be set to a very specific

value, which guarantees that the same ‘random’ sequence of realizations is picked every time we restart the program. This can be useful when we are trying to replicate exactly the same result over and over again. Recall that the intrinsic subroutine `random_number` only generates uniform random variable realizations $\{z_i\}_{i=1}^n$ on the interval $[0, 1]$. It is, however, easy to transform these into random variable realizations $\{x_i\}_{i=1}^n$ that are uniformly distributed on the interval $[a, b]$ using the relation

$$x_i = a + (b - a) \cdot z_i. \quad (2.12)$$

Simulating realizations of a random variable with a more complicated distribution turns out to be a bit more challenging. One would think that we could apply a very simple statistical theorem for this. In fact, we know that when $\{z_i\}_{i=1}^n$ is a series of independent realizations of a uniformly distributed random variable on the interval $[0, 1]$, then the transformed realizations

$$x_i = \Phi^{-1}(z_i)$$

follow a distribution with cumulative distribution function Φ . Φ^{-1} is the inverse of the cumulative distribution function, meaning a function that satisfies $z = \Phi^{-1}[\Phi(z)]$. Note that we intuitively used this relationship in equation (2.12). There are, however, two problems associated with this method. First, it is often impossible to analytically characterize both the cumulative distribution function Φ as well as its inverse Φ^{-1} , e.g. for the normal distribution. Second, even if we could provide an approximation of Φ^{-1} , the simulated sequence $\{x_i\}_{i=1}^n$ typically doesn’t have the best properties and its generation is not very efficient.

The toolbox provides a solution. It contains a set of subroutines that allow for the simulation of a series of random variables that follow the seven distributions discussed above. Program 2.16 shows how we can use them. The way these subroutines work is always the same. We first have to specify the respective parameters of the distribution and then call, for example, the subroutine `simulate_normal` to simulate a sequence of normally distributed random numbers with mean `mu` and variance `sigma`. The result is stored in the array `x` of length `NN`. We then use our simulated series of random number realizations to create a histogram plot of relative frequencies using the subroutine `plot_hist`. This subroutine takes as input the array `x` of random numbers as well as the number of histogram bars it should use. For the uniform and the normal distribution, we use 20 bars. For the binomial distribution, for example, it only makes sense to use $n + 1$, as there are only $n + 1$ potential realizations. In addition to the number of bars, the subroutine `plot_hist` can receive two optional arguments `left` and `right` that define the left and right endpoint of the histogram. Figure 2.12 shows the resulting histogram plots for the distributions shown in Figure 2.11. Not surprisingly, the histograms approximately resemble the shape of the respective probability density functions.

Program 2.16 Density and cumulative distribution functions

```

program simulation
[.....]
! uniform distribution
a = -1d0
b = 1d0
call simulate_uniform(x, a, b)
call plot_hist(x, 20)
call execplot(title="Uniform Distribution")

! normal distribution
mu = 1d0
sigma = 0.25d0
call simulate_normal(x, mu, sigma)
call plot_hist(x, 20, left=mu-3d0*sqrt(sigma), &
               right=mu+3d0*sqrt(sigma))
call execplot(title="Normal Distribution")
[.....]
! binomial distribution
p = 0.25d0
n = 12
call simulate_binomial(x, n, p)
call plot_hist(x, n+1, left=-0.5d0, right=dble(n)+0.5d0)
call execplot(title="Binomial Distribution")

end program

```

2.6 Function approximation and interpolation

In computational economic applications one often has to approximate an analytically intractable (or even unknown) function f with a computationally tractable one \hat{f} . The approximation of a function usually requires two preparatory steps, the first of which is to choose the analytical form of the *approximant* \hat{f} . Due to manageability reasons, one usually confines oneself to approximants that can be written as a linear combination of a set of $n + 1$ linearly independent *basis functions* $\phi_0(x), \dots, \phi_n(x)$, i.e.

$$\hat{f}(x) = \sum_{j=0}^n c_j \phi_j(x), \quad (2.13)$$

whose *basis coefficients* c_0, \dots, c_n are to be determined. The monomials $1, x, x^2, x^3, \dots$ of increasing order are a popular example for basis functions. The number n is called the *degree of interpolation*.

In the second step one has to specify which properties of the original function f the approximant \hat{f} should replicate. Since there are $n+1$ undetermined coefficients, one needs at least $n + 1$ conditions in order to pin down the approximant. The simplest conditions that could be imposed are that the approximant replicates the original function values $f(x_0), \dots, f(x_n)$ at selected *interpolation nodes* x_0, \dots, x_n . \hat{f} is then called an *interpolant*

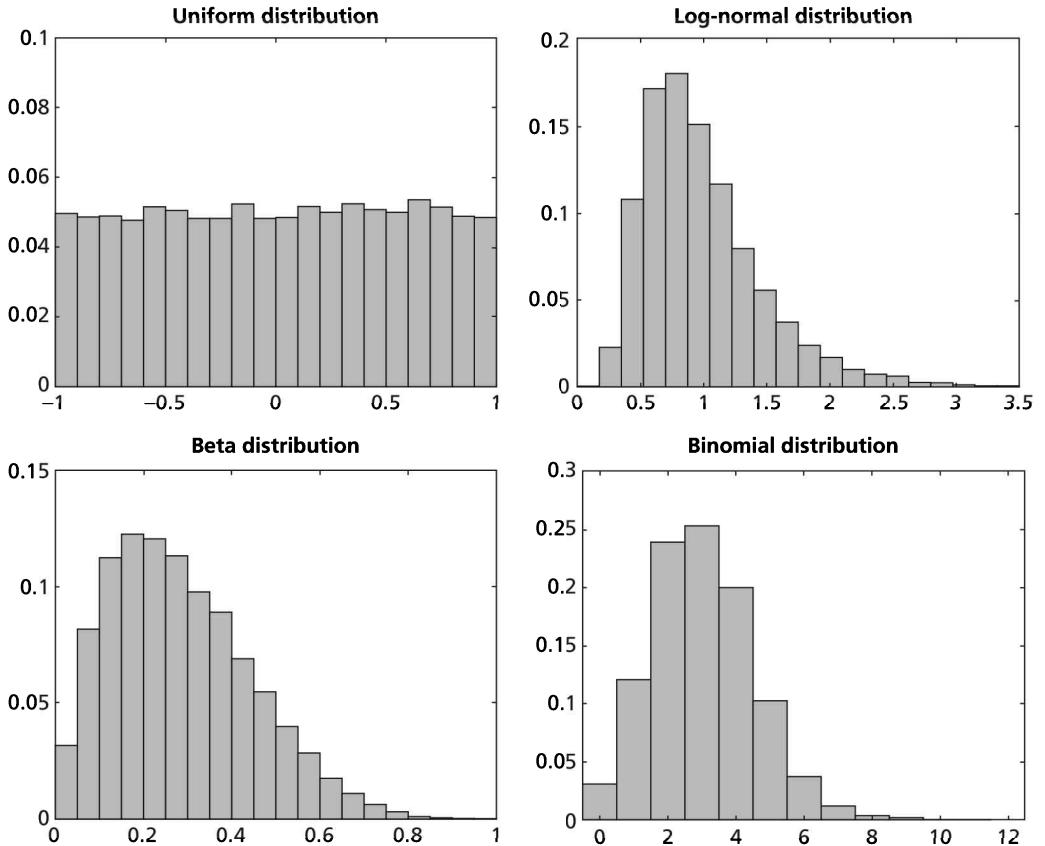


Figure 2.12 Selected density and cumulative distribution functions

of f . Given $n + 1$ interpolation nodes x_i and the $m + 1$ basis functions $\phi_i(x)$, the computation of the basis coefficients c_i reduces to solving a linear equation system

$$\sum_{j=0}^n c_j \phi_j(x_i) = f(x_i), \quad i = 0, 1, \dots, n.$$

Using matrix notation, the interpolation conditions may be written as $\Phi c = y$, where

$$\Phi = \begin{bmatrix} \phi_0(x_0) & \phi_1(x_0) & \dots & \phi_n(x_0) \\ \phi_0(x_1) & \phi_1(x_1) & \dots & \phi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(x_n) & \phi_1(x_n) & \dots & \phi_n(x_n) \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \text{ and } y = \begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

The interpolation scheme is well defined if the interpolation nodes and the basis functions are chosen such that the interpolation matrix is non singular. We then have

$$c = \Phi^{-1}y.$$

Interpolation may be viewed as a special case of the *curve-fitting problem*. The latter arises when there are fewer basis functions $\phi_j(x), j = 0, \dots, m$ than function evaluation nodes $x_i, i = 0, \dots, n$. With $n > m$ it is not possible to satisfy the interpolation conditions exactly at every node. However, we can construct a reasonable approximant by minimizing the sum of squared errors

$$\|\Phi c - y\|_2 = \sum_{i=0}^n \left(\sum_{j=0}^m c_j \phi_j(x_i) - f(x_i) \right)^2.$$

Following this strategy, we obtain the well-known least-squares approximation

$$c = (\Phi^T \Phi)^{-1} \Phi^T y,$$

which is equivalent to the interpolation equation, if $n = m$, i.e. if Φ is invertible.

Figure 2.13 shows the difference between an interpolant $\hat{f}^{INT}(x)$ and curve-fitting approximant $\hat{f}^{APP}(x)$ for four interpolation nodes.

In general, any approximation scheme should satisfy the following criteria:

1. The goodness of fit should rise by increasing the number of basis functions and nodes.
2. Basis coefficients have to be computable quickly and accurately.
3. The approximant \hat{f} should be easy to work with, i.e. basis functions ϕ_j should be simple and relatively costless to evaluate, differentiate, and integrate.

In the following we distinguish between *polynomial* and *piecewise polynomial* interpolation. The former applies basis functions, which are non-zero over the entire domain of the function being approximated. The latter uses basis functions that are non-zero over subintervals of the approximation domain.

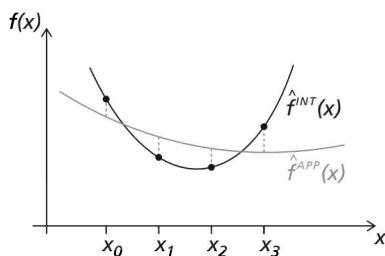


Figure 2.13 Interpolation vs. approximation

2.6.1 POLYNOMINAL INTERPOLATION

According to the Weierstrass Theorem, any continuous function f defined on a bounded interval $[a, b]$ of the real line can be approximated to any degree of accuracy using a polynomial. This theorem provides a strong motivation for using polynomials to approximate continuous functions. Suppose there are $i = 0, \dots, n$ nodes x_i and interpolation data $y_i = f(x_i)$. Then the polynominal function

$$p(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n$$

of degree n can exactly replicate the data at each node. From $n + 1$ nodes we can derive $n + 1$ equations

$$\begin{aligned} c_0 + c_1x_0 + c_2x_0^2 + \dots + c_nx_0^n &= f(x_0) \\ c_0 + c_1x_1 + c_2x_1^2 + \dots + c_nx_1^n &= f(x_1) \\ &\vdots \\ c_0 + c_1x_n + c_2x_n^2 + \dots + c_nx_n^n &= f(x_n). \end{aligned}$$

In matrix notation we obtain

$$\Phi = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}, \quad c = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \quad \text{and} \quad y = \begin{bmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_n) \end{bmatrix},$$

where the linear equation system $\Phi c = y$ could be solved by LU-decomposition. Φ in this context is called the *Vandermonde matrix*.

Having already specified the basis functions, we still have to chose the interpolation nodes x_i , $i = 0, \dots, n$. In general, we could use any arbitrary set of points $x_i \in [a, b]$. However, if we want to interpolate a function f accurately, we should pay some more attention to the choice of nodes as it heavily influences the goodness of fit of our interpolation scheme. Of course, the simplest approach would be to select n evenly spaced (or equidistant) interpolation nodes

$$x_i = a + \frac{i}{n}(b - a), \quad i = 0, \dots, n.$$

In practice, however, polynomial interpolation at evenly spaced nodes often comes with problems as the Vandermonde matrix becomes ill conditioned and close to singular. Therefore the calculation of coefficients becomes prone to errors. There are functions for which the goodness of fit of polynomial interpolants with equidistant nodes rapidly

deteriorates, rather than improves, with the degree of approximation n rising. The classical example is Runge's function

$$f(x) = \frac{1}{1 + 25x^2}.$$

The main reasons for the errors with evenly spaced nodes are problems at the endpoints of the interpolation interval. There, the interpolating polynomials tend to oscillate. However, this problem can be solved by choosing the nodes so that they are more closely spaced near the endpoints and less in the centre. This is the underlying principle for the construction of so-called *Chebyshev nodes*

$$x_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{n-i+0.5}{n+1}\pi\right), \quad i = 0, \dots, n.$$

They are symmetric on $[a, b]$, but their distance steadily decreases when moving closer to the endpoints. Figure 2.14 shows Chebyshev nodes for $n = 7$. Chebyshev-node polynomial interpolants possess some strong theoretical properties with respect to the approximation error in between the nodes, but this will not be of any interest in the following.

Polynomial interpolation is implemented in the toolbox in the subroutine `poly_interp1`. In addition, there are two grid constructors for both equidistant and Chebyshev nodes, respectively. Program 2.17 shows how to apply these subroutines in order to interpolate Runge's function. First we set up an evenly spaced grid of 1,000 nodes on the domain $[-1, 1]$ which will be used for plotting our interpolation results. We also calculate the real values of Runge's function at these nodes. We then start constructing equidistant interpolation nodes. The subroutine `grid_Cons_Equi` therefore receives a one-dimensional array of arbitrary length as well as the left and right interpolation interval endpoints. It fills up the array with a set of evenly spaced points on the specified interval. We then also calculate Runge's function at the interpolation nodes. The interpolating polynomial can be evaluated at the grid `xplot` using the subroutine `poly_interp1`. It receives as input the nodes at which to evaluate the polynomial as well as the interpolation data (x_i, y_i) . Finally, we plot the error the interpolant makes in predicting the values in between the interpolation nodes, i.e. the difference between the polynomial and the real function values at the points in `xplot`. We repeat the procedure using Chebyshev nodes from the grid constructor `grid_Cons_Cheb`. The resulting plot can be seen in Figure 2.15. We can see that the approximation with equidistant nodes is

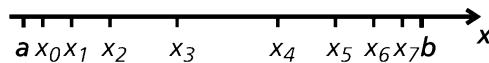


Figure 2.14 Chebyshev nodes

Program 2.17 Polynomial interpolation of Runge's function

```

program polynomials

use toolbox

implicit none
integer, parameter :: n = 10, nplot = 1000
real*8 :: xi(0:n), yi(0:n)
real*8 :: xplot(0:nplot), yplot(0:nplot), yreal(0:nplot)

! get equidistant plot nodes and Runge's function
call grid_Cons_Equi(xplot, -1d0, 1d0)
yreal = 1d0/(1d0+25*xplot**2)
call plot(xplot, yreal, legend='Original')

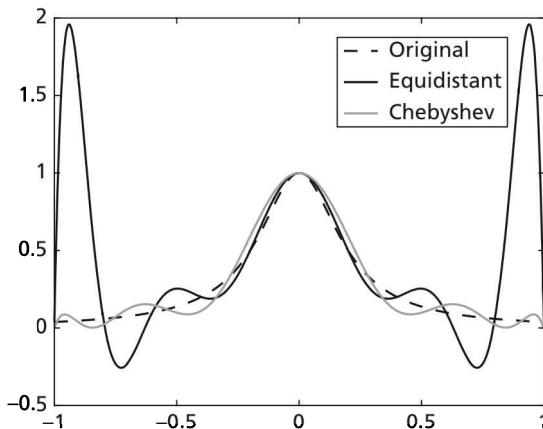
! equidistant polynomial interpolation
call grid_Cons_Equi(xi, -1d0, 1d0)
yi = 1d0/(1d0+25*xi**2)
yplot = poly_interp(xplot, xi, yi)
call plot(xplot, yplot, legend='Equidistant')

! Chebyshev polynomial interpolation
call grid_Cons_Cheb(xi, -1d0, 1d0)
yi = 1d0/(1d0+25*xi**2)
yplot = poly_interp(xplot, xi, yi)
call plot(xplot, yplot, legend='Chebyshev')

! execute plot
call execplot()

end program

```

**Figure 2.15** Equidistant vs. Chebyshev nodes

quite inaccurate compared to the one with Chebyshev nodes. If we now increased the number of interpolation data and therefore the degree of the interpolating polynomial, we would find that the error for equidistant nodes rapidly increases, whereas the one for Chebyshev nodes decreases.

2.6.2 PIECEWISE POLYNOMIAL INTERPOLATION

Compared to fitting the function f on the entire domain $[a, b]$ by one polynomial, it might be easier to only define interpolating polynomials on subintervals $[x_i, x_{i+1}]$. This approach is called *piecewise polynomial interpolation*. Global polynomial interpolation, as described in Sections 2.6.1, uses all the information stored in the data points (x_i, y_i) , $i = 0, \dots, n$ to interpolate the function f . A change in the data point (x_0, y_0) , for example, can therefore cause an adjustment in the shape of the interpolant \hat{f} even on an interval $[x_i, x_{i+1}]$ far away from this point. In contrast, piecewise polynomial interpolation only makes use of local information about the function f around an interval $[x_i, x_{i+1}]$. It therefore proves more stable towards changes in interpolation data that happen far away from such an interval. More importantly, however, calculating piecewise polynomial interpolants usually turns out much simpler.

In general, an order k interpolating piecewise polynomial function \hat{f} consists of n polynomials of degree $k \geq 1$, each defined on one of the n intervals $[x_i, x_{i+1}]$, $i = 0, \dots, n - 1$. It satisfies the interpolation conditions $\hat{f}(x_i) = f(x_i)$ at the interpolation nodes and is continuous over the whole domain $[a, b]$. However, in contrast with an interpolating polynomial, an order- k piecewise polynomial function is only $k - 1$ times continuously differentiable on the whole interval $[a, b]$, i.e. it is not as smooth as a polynomial. Note that the problem of non-differentiability only arises at the interpolation nodes, where the polynomial pieces are spliced together.

In the following we concentrate on two specific classes of piecewise polynomial interpolants, which are employed widely in practice. A first-order or *piecewise linear* function is a series of line segments that is linked at the interpolation nodes to form a continuous function. A third-order function or *cubic spline* is a series of cubic polynomial segments spliced together to form a twice continuously differentiable function. Figure 2.16 compares piecewise linear $\hat{f}^{\text{LIN}}(x)$ and cubic spline $\hat{f}^{\text{SPL}}(x)$ interpolation for $n = 4$ breakpoints.

Piecewise linear functions are particularly easy to construct and evaluate in practice, which explains their widespread popularity. Assume an arbitrary set of interpolation

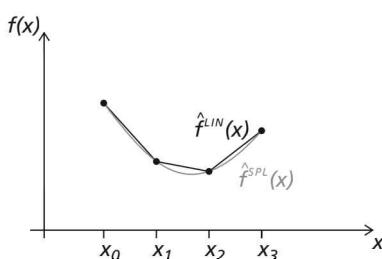


Figure 2.16 Piecewise linear interpolation vs. cubic splines

data $(x_i, f(x_i))$ is given. In the interval $[x_i, x_{i+1}]$ we define the approximating linear function f_i as

$$\hat{f}_i(x) = c_{0,i} + c_{1,i}x,$$

where the coefficients can be computed from the two interpolation conditions

$$\begin{aligned} f(x_i) &= c_{0,i} + c_{1,i}x_i \\ f(x_{i+1}) &= c_{0,i} + c_{1,i}x_{i+1}. \end{aligned}$$

We can show that the interpolating function can be written as

$$\hat{f}_i(x) = \frac{(x_{i+1} - x)f(x_i) + (x - x_i)f(x_{i+1})}{x_{i+1} - x_i} \quad \text{if } x \in [x_i, x_{i+1}]$$

which further simplifies to

$$\begin{aligned} \hat{f}_i(x) &= \varphi \cdot f(x_i) + (1 - \varphi) \cdot f(x_{i+1}) \\ \text{with } \varphi &= \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad \text{if } x \in [x_i, x_{i+1}]. \end{aligned} \quad (2.14)$$

Piecewise linear functions are attractive for their simplicity, however due to not being differentiable over the whole domain they have some shortcomings which limit their application in many computational economic problems. For example, when derivatives are required in order to find extrema of a function using Newton-type methods, piecewise linear functions are not a good choice. In such a case, cubic splines offer a better alternative. They retain much of the stability and flexibility of piecewise linear functions, but possess continuous first and second derivatives. Consequently they typically produce adequate approximations for both the function and its first and second derivatives.

The $n - 1$ piecewise third-order polynomials of a cubic spline on the interval $[x_i, x_{i+1}]$ are

$$\hat{f}_i(x) = c_{0,i} + c_{1,i}x + c_{2,i}x^2 + c_{3,i}x^3, \quad i = 0, \dots, n - 1.$$

Note that we now have $4n$ coefficients to pin down our cubic spline. The required equations are derived from the following conditions:

1. The polynomials have to replicate the function values at the interpolation nodes ($2n$ equations)

$$\begin{aligned} c_{0,i} + c_{1,i}x_i + c_{2,i}x_i^2 + c_{3,i}x_i^3 &= f(x_i), & i &= 0, \dots, n - 1, \\ c_{0,i} + c_{1,i}x_{i+1} + c_{2,i}x_{i+1}^2 + c_{3,i}x_{i+1}^3 &= f(x_{i+1}), & i &= 0, \dots, n - 1. \end{aligned}$$

2. First derivatives of two consecutive polynomials at the inner interpolation nodes have to be equal ($n - 1$ equations)

$$c_{1,i} + 2c_{2,i}x_i + 3c_{3,i}x_i^2 = c_{1,i+1} + 2c_{2,i+1}x_i + 3c_{3,i+1}x_i^2, \quad i = 1, \dots, n - 1.$$

3. Second derivatives of two consecutive polynomials at the inner interpolation nodes have to be equal ($n - 1$ equations)

$$2c_{2,i} + 6c_{3,i}x_i = 2c_{2,i+1} + 6c_{3,i+1}x_i, \quad i = 1, \dots, n - 1.$$

All in all this makes $4n - 2$ equations in order to compute $4n$ coefficients $c_{0,i}, c_{1,i}, c_{2,i}, c_{3,i}$, $i = 0, \dots, n - 1$. The two missing equations can be specified in different ways. Typically one applies the condition for a so-called *natural spline*, which requires that the second derivative is zero at the endpoints of the domain, i.e.

$$\begin{aligned} 2c_{2,1} + 6c_{3,1}x_0 &= 0 \\ 2c_{2,n} + 6c_{3,n}x_n &= 0. \end{aligned}$$

This implies the spline function to be a straight line outside of the domain $[a, b]$.

Note that the above illustration of a spline function is useful for understanding how piecewise third-order polynomial interpolation actually works. In practice, however, calculating $4n$ coefficients would not be very efficient both in terms of computational time and computer memory usage. Yet, one can show that by defining proper basis functions $\phi_j(x)$, one can write any cubic spline function similarly to (2.13) as

$$\hat{f}(x) = \sum_{j=0}^{n+2} c_j \phi_j(x).$$

A cubic spline is therefore exactly defined by $n + 3$ coefficients c_j . These coefficients can be pinned down by the $n + 1$ interpolation conditions $\hat{f}(x_i) = f(x_i)$, $i = 0, \dots, n$ together with two additional conditions at the endpoints, see the discussion above.³ While this sounds like quite complicated maths, the toolbox will in the end take care of the calculation of coefficients c_j . The only thing we need to remember is that in order to define a spline function, $n + 3$ coefficients are needed.

Technically we could choose our interpolation nodes quite arbitrarily on the domain. Indeed, the ability to distribute breakpoints unevenly over the domain adds considerably to the flexibility of cubic splines, allowing them to accurately approximate a wide range of functions. However, there is a very simple algorithm for computing the coefficients of

³ At this point it becomes immediately clear why two additional equations are needed to define an interpolating spline function.

Program 2.18 Piecewise linear and cubic spline interpolation

```

program piecewise_pol

use toolbox

implicit none
integer, parameter :: n = 10, nplot = 1000
real*8 :: xi(0:n), yi(0:n)
real*8 :: xplot(0:nplot), yplot(0:nplot), yreal(0:nplot)
real*8 :: varphi, coeff(n+3)
integer :: ix, il, ir

! get nodes and data for interpolation
call grid_Cons_Equi(xi, -1d0, 1d0)
yi = 1d0/(1d0+25*xi**2)

! get nodes and data for plotting
call grid_Cons_Equi(xplot, -1d0, 1d0)
yreal = 1d0/(1d0+25*xplot**2)
call plot(xplot, yreal, legend='Original')

! piecewise linear interpolation
do ix = 0, nplot
    call linint_Equi(xplot(ix), -1d0, 1d0, n, il, ir, varphi)
    yplot(ix) = varphi*yi(il) + (1d0-varphi)*yi(ir)
enddo
call plot(xplot, yplot, legend='Piecewise linear')

! cubic spline interpolation
call spline_interp(yi, coeff)
do ix = 0, nplot
    yplot(ix) = spline_eval(xplot(ix), coeff, -1d0, 1d0)
enddo
call plot(xplot, yplot, legend='Cubic spline')

! execute plot
call execplot(xlim=(-1d0,1d0))

end program

```

splines when interpolation nodes are equidistant, which is implemented in the toolbox. In addition, the toolbox provides a routine for linear interpolation on equidistant nodes.⁴

Program 2.18 shows how to apply both methods to Runge's function. We first create the interpolation nodes ξ_i using the construction subroutine `grid_Cons_Equi` for equidistant nodes and store the values of Runge's functions in the interpolation data y_i . Linear interpolation on equidistant nodes can be performed using the subroutine `linint_Equi`. This subroutine receives as input arguments the point $xplot(ix)$ at which we want to interpolate Runge's function as well as the specification of the grid of nodes we are working on, i.e. the left and right interval bound as well as the number of interpolation nodes n . In addition, we hand over two variables `il` and `ir` of type `integer` as well as a variable `varphi` of type `real*8`. In these arrays the subroutine `linint_Equi` stores the indexes of the left and right interpolation node x_i and x_{i+1} as well as the interpolation share φ , so that

⁴ The toolbox in fact also contains routines for linear interpolation on Chebyshev nodes (`linint_Cheb`) and nodes with growing distance (`linint_Grow`), which can be used analogously.

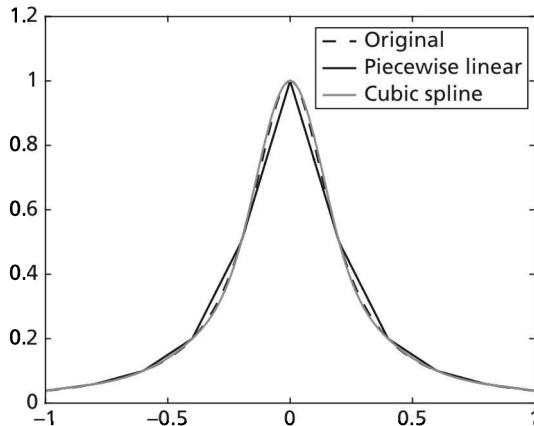


Figure 2.17 Piecewise linear interpolation and cubic splines for Runge's function

the interpolated function value can be computed exactly from (2.14). Spline interpolation on equidistant grids is implemented in the subroutine `spline_interp` and the function `spline_eval`. The former calculates the $n+3$ spline coefficients c_j , the latter evaluates the interpolating cubic spline function at some arbitrary point x on the interpolation interval $[a, b]$. The subroutine `spline_interp(yi, coeff)` receives as input the interpolation data yi , and stores the corresponding spline coefficients in an array `coeffs` of length $n+3$. The function `spline_eval(xplot(ix), coeff, -1d0, 1d0)` evaluates the cubic spline function at point `xplot(ix)`, given the coefficients computed before as well as the left and right interval border of the interpolation domain. Note that the function `spline_eval` already assumes that the nodes x_i are equally spaced on the domain $[-1, 1]$. We finally plot Runge's functions and its piecewise polynomial interpolants for all our 1,001 equally spaced nodes on $[-1, 1]$. Figure 2.17 shows the results.

2.6.3 A TWO-DIMENSIONAL INTERPOLATION EXAMPLE

The routines provided above can be extended to handle multidimensional interpolation problems. However, once we want to interpolate functions that exist on a multidimensional space, we have to make some assumptions regarding the structure of the interpolation nodes. It is most convenient to assume that these nodes form a so-called *rectilinear grid*, that is the cells defined by the interpolation nodes are rectangular cuboids.⁵ Let's take as example the two-dimensional function

$$f(x, z) = x^2 + \sqrt{z}.$$

⁵ There are also interpolation methods for irregular sets of nodes. However, such methods require a lot more sophistication than the rather simple methods presented here.

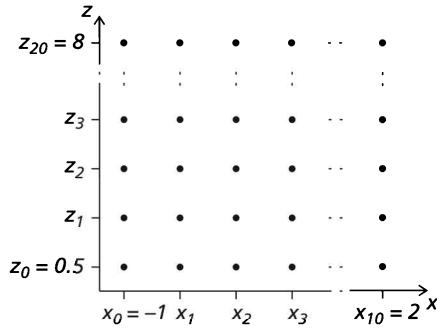


Figure 2.18 Rectilinear grid of nodes for multidimensional interpolation

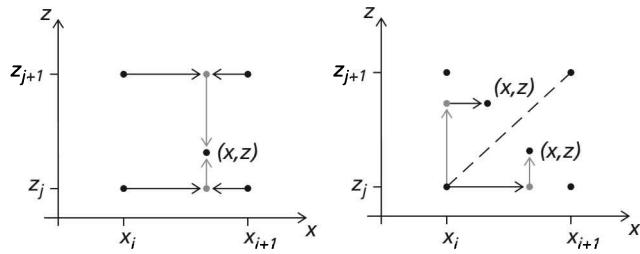


Figure 2.19 Two-dimensional linear interpolation

This function should be interpolated for values of $x \in [-1, 2]$ and $z \in [0.5, 8]$. The easiest way to set up a rectilinear grid of nodes is to construct separate (equidistant) grids along the x and the z dimension and just use their tensor product. Figure 2.18 illustrates this idea. We therefore used $n_x = 10$ and $n_z = 20$ nodes along the x and the z dimension, respectively. The advantage of this rectilinear grid is that gridpoints follow some kind of order and that we can still index them using integer values. Note that the spaces between the single nodes do not have to be identical, nor do they necessarily have to be equidistant. The distance between two nodes could, for example, also be growing or declining along the grid.

Bilinear interpolation On such a simple rectilinear grid structure there are in fact two ways to perform linear interpolation. Figure 2.19 illustrates their mechanics. We thereby assume the point (x, z) at which we want to interpolate the function $f(x, z)$ to lie within the interval $[x_i, x_{i+1}]$ along the x -dimension and the interval $[z_j, z_{j+1}]$ along the z -dimension. The first approach shown in the left panel of the figure is called *bilinear interpolation*. Its main idea is to successively use linear interpolation along the different dimensions of the function f . For both values z_j and z_{j+1} , we therefore interpolate the function f along the x -dimension to obtain

$$\begin{aligned}\hat{f}_z(z_j) &= \varphi_x f(x_i, z_j) + (1 - \varphi_x) f(x_{i+1}, z_j) \quad \text{and} \\ \hat{f}_z(z_{j+1}) &= \varphi_x f(x_i, z_{j+1}) + (1 - \varphi_x) f(x_{i+1}, z_{j+1}).\end{aligned}$$

To determine the actual function value of the interpolant, we then interpolate the values of f_z along the z -dimension and obtain

$$\hat{f}_{i,j}(x, z) = \varphi_z \hat{f}_z(z_j) + (1 - \varphi_z) \hat{f}_z(z_{j+1}),$$

with

$$\varphi_x = \frac{x_{i+1} - x}{x_{i+1} - x_i} \quad \text{and} \quad \varphi_z = \frac{z_{j+1} - z}{z_{j+1} - z_j}.$$

Following this logic, we can conveniently write the piecewise linear interpolating function of $f(x, z)$ as

$$\begin{aligned} \hat{f}_{i,j}(x, z) &= \varphi_x \cdot \varphi_z \cdot f(x_i, z_j) + (1 - \varphi_x) \cdot \varphi_z \cdot f(x_{i+1}, z_j) \\ &\quad + \varphi_x \cdot (1 - \varphi_z) \cdot f(x_i, z_{j+1}) + (1 - \varphi_x) \cdot (1 - \varphi_z) \cdot f(x_{i+1}, z_{j+1}) \end{aligned}$$

if $x \in [x_i, x_{i+1}]$ and $z \in [z_j, z_{j+1}]$.⁶

While a bilinear interpolant is also very easily computed in higher dimensions, bilinear interpolation is not exactly the generalization of linear interpolation in multiple dimensions. Recalling our knowledge from linear algebra, we know that the analogue to a line in two-dimensional space is a hyperplane in multidimensional space, or simply a plane when we are thinking about three dimensions. A plane, however, is defined by three and not four points. Hence, a bilinear interpolant is not a plane, but rather a polynomial of degree two. One can see this by rearranging the above definition of $\hat{f}_{i,j}(x, z)$ to

$$\hat{f}_{i,j}(x, z) = c_0 + c_1 x + c_2 z + c_3 xz \tag{2.15}$$

with suitable coefficients c_0 , c_1 , c_2 and c_3 . Hence, the interpolant $\hat{f}_{i,j}(x, z)$ contains an interaction term of x and z , which is typical for a two-dimensional quadratic polynomial, but not for a plane. One can criticize bilinear interpolation for not being a real linear interpolation scheme, nor being a fully quadratic interpolation scheme either, since a complete quadratic interpolation scheme would also feature quadratic terms $c_4 x^2$ and $c_5 z^2$ along both interpolation dimensions in (2.15). Because of its restricted quadratic nature, a bilinear interpolant can exhibit some unintended curvature in the inner part of the rectangle on which it interpolates the function f . It might therefore be useful to use an interpolation scheme that actually works with hyperplanes.

⁶ Note that this kind of successive dimension reduction can be easily extended to more than two dimensions.

Multidimensional linear interpolation As just discussed, we only need three points to actually define a plane in three-dimensional space. To implement a true linear interpolation scheme for the function $f(x, z)$, we can therefore follow the logic of the interpolation scheme illustrated in the right panel of Figure 2.19. After knowing in which rectangle the point (x, z) is located, we have to select into which of the triangles that are marked by the diagonal of the rectangle this point actually falls. Note that we can express any point in the square marked by the corner points (x_i, z_j) and (x_{i+1}, z_{j+1}) as

$$\begin{aligned} \begin{bmatrix} x \\ z \end{bmatrix} &= \begin{bmatrix} x_i \\ z_j \end{bmatrix} + (1 - \varphi_x) \left(\begin{bmatrix} x_{i+1} \\ z_j \end{bmatrix} - \begin{bmatrix} x_i \\ z_j \end{bmatrix} \right) + (1 - \varphi_z) \left(\begin{bmatrix} x_{i+1} \\ z_{j+1} \end{bmatrix} - \begin{bmatrix} x_i \\ z_j \end{bmatrix} \right) \\ &= \varphi_x \begin{bmatrix} x_i \\ z_j \end{bmatrix} + (\varphi_z - \varphi_x) \begin{bmatrix} x_{i+1} \\ z_j \end{bmatrix} + (1 - \varphi_z) \begin{bmatrix} x_{i+1} \\ z_{j+1} \end{bmatrix} \end{aligned}$$

with $0 \leq \varphi_x, \varphi_z \leq 1$. The diagonal of the rectangle is then defined by $\varphi_x = \varphi_z$ and a point (x, z) falls into the lower triangle whenever $1 - \varphi_z \leq 1 - \varphi_x$ or better $\varphi_x \leq \varphi_z$. The same logic can be applied for the upper triangle, where we would move along the z -dimension first, see Figure 2.19. Consequently, we can write the true two-dimensional linear interpolant of f as

$$\hat{f}_{i,j}(x, z) = \begin{cases} \varphi_x f(x_i, z_j) + (\varphi_z - \varphi_x) f(x_{i+1}, z_j) \\ \quad + (1 - \varphi_z) f(x_{i+1}, z_{j+1}) & \text{if } \varphi_x \leq \varphi_z \quad \text{and} \\ \varphi_z f(x_i, z_j) + (\varphi_x - \varphi_z) f(x_i, z_{j+1}) \\ \quad + (1 - \varphi_x) f(x_{i+1}, z_{j+1}) & \text{otherwise,} \end{cases}$$

where φ_x and φ_z are defined in exactly the same way as under the bilinear interpolation scheme.

Program 2.19 shows how we can implement the two-dimensional bilinear and linear interpolation schemes. In this program we first set up our interpolation data. We use the equidistant grid constructor function in both the x and the z dimension with the parameters as specified above. We then construct the interpolation data by calculating the function value $f(x_i, z_j)$ at every gridpoint of the rectilinear grid. Next we set up another equidistant rectilinear grid on the same interval (but with many more points `nerr`) and store it in the arrays `xerr` and `zerr` (not shown in the program). We use these points to evaluate the error the different interpolation routines make in between the data nodes. To do this, we calculate for each value combination of `xerr` and `zerr` the rectangle corners (x_i, z_j) and (x_{i+1}, z_{j+1}) as well as the interpolation shares φ_x and φ_z . We then evaluate the bilinear interpolant at each node, and store the result in an array `y_bi`. The true linear interpolant that is defined on triangles can be calculated from the same interpolation data as described above.

Program 2.19 Two-dimensional interpolation

```

program multi_interp
[....]
integer, parameter :: nx = 10, nz = 20, nerr = 1000
real*8, parameter :: x_l = -1d0, x_r = 2d0
real*8, parameter :: z_l = 0.5d0, z_r = 8d0
[.....]

! get nodes and data for interpolation
call grid_Cons_Equi(xi, x_l, x_r)
call grid_Cons_Equi(zi, z_l, z_r)
do ix = 0, nx
    do iz = 0, nz
        yi(ix, iz) = xi(ix)**2 + sqrt(zi(iz))
    enddo
enddo
[.....]
! piecewise linear interpolation
do ix = 0, nerr
    do iz = 0, nerr
        call linint_Equi(xerr(ix), x_l, x_r, &
                           nx, ixl, ixr, varphix)
        call linint_Equi(zerr(iz), z_l, z_r, &
                           nz, izl, izr, varphiz)

        ! bilinear
        Y_bi(ix, iz) = varphix*varphiz*yi(ixl, izl) &
                       + (1d0-varphix)*varphiz*yi(ixr, izl) &
                       + varphix*(1d0-varphiz)*yi(ixl, izr) &
                       + (1d0-varphix)*(1d0-varphiz)*yi(ixr, izr)

        ! on triangles
        if(varphix <= varphiz)then
            y_tr(ix, iz) = varphix*yi(ixl, izl) &
                           + (varphiz-varphix)*yi(ixr, izl) &
                           + (1d0-varphiz)*yi(ixr, izr)
        else
            y_tr(ix, iz) = varphiz*yi(ixl, izl) &
                           + (varphix-varphiz)*yi(ixl, izr) &
                           + (1d0-varphix)*yi(ixr, izr)
        endif
    enddo
enddo
[.....]
! cubic spline interpolation
call spline_interp(yi, coeff)
do ix = 0, nerr
    do iz = 0, nerr
        Y_sp(ix, iz) = spline_eval((/xerr(ix), zerr(iz)/), &
                                    coeff, (/x_l, z_l/), (/x_r, z_r/))
    enddo
enddo
[.....]
end program

```

Multidimensional spline interpolation Last but not least, when the interpolation nodes form a rectilinear grid, we can also use multidimensional spline interpolation. The subroutine `spline_interp` and the function `spline_eval` therefore are called up in a similar way as in the previous Program 2.18. We first interpolate the data

y_i using the subroutine `spline_interp`. The two-dimensional cubic spline now is defined by an array of coefficients of size `coeff(nx+3, nz+3)`, i.e. the array has the same number of dimensions as y_i but two more entries along each dimension. We then evaluate the spline function defined by `coeff` at each node given by `xerr` and `zerr` using the function `spline_eval`. This function takes as input a point where the spline should be evaluated, the defining array of coefficients as well as the left and right interval points of the equidistant rectilinear grid of interpolation nodes. To evaluate how well the three schemes interpolate the function f in between the interpolation nodes we eventually calculate the maximum absolute relative difference of the interpolation error, e.g. `maxval(abs(y_bi/yreal-1d0))`, and write it to the console (not shown here). When running the program we first of all find that—for this particular function—spline interpolation is much more accurate than linear interpolation. In fact the error made by the linear interpolation routines is about a factor 25 larger than the error made by multidimensional spline interpolation. The bilinear and linear interpolation schemes, while a little different in shape, generate exactly the same interpolation error.

2.7 Linear programming

In planning production processes and in several logistic problems one usually finds oneself confronted with solving a *linear program*. A linear program in *standard form* is given by the optimization problem

$$\min_{x_1, \dots, x_n} c_1 x_1 + c_2 x_2 + \dots + c_n x_n \quad \text{with} \quad x_i \geq 0, \quad i = 1, \dots, n$$

with the m inequality constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &\leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n &\leq b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &\leq b_m. \end{aligned}$$

With the matrices and vectors

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \quad c = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \quad \text{and} \quad x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix},$$

we can write

$$\min_x c^T x \quad \text{s.t.} \quad x \geq 0 \quad \text{and} \quad Ax \leq b.$$

Example Consider a company that produces two goods x_1 and x_2 on one type of machine. Due to storage capacity, the production is limited to 100 pieces altogether. The production of the two goods requires time and raw material. Good 1 thereby is a time-and cost-intensive production good which requires four hours of production time and raw material worth € 20 per piece. Good 2 is less intensive and can therefore be produced within one hour and with raw material worth € 10. The company owns eight machines which can run 20 hours a day. The remaining time has to be spent on maintenance. In addition, overall costs of raw material should be limited to €1,100 per day. The two goods can be sold on the market for €120 and €40, respectively. What is the optimal production of goods per day for the company?

Table 2.2 summarizes the problem. The goal of the company obviously is to maximize its profits $120x_1 + 40x_2$ subject to the given production constraints. This problem can be written as a linear program in standard form

$$\min_{x_1, x_2} -120x_1 - 40x_2, \quad x_1, x_2 \geq 0$$

subject to the constraints

$$\begin{aligned} x_1 + x_2 &\leq 100 \\ 4x_1 + x_2 &\leq 160 \\ 20x_1 + 10x_2 &\leq 1100. \end{aligned}$$

Note that we again just minimize the negative of the objective function in order to maximize it. The same approach was taken in Section 2.3 that dealt with the minimization of nonlinear functions.

Table 2.2 Linear programming example

| | Type 1 | Type 2 | Restriction |
|----------------|--------|--------|-------------|
| Pieces per day | 1 | 1 | 100 |
| Time input | 4 | 1 | 160 |
| Raw material | 20 | 10 | 1,100 |
| Market price | 120 | 40 | ? |

2.7.1 GRAPHICAL SOLUTION TO LINEAR PROGRAMS IN STANDARD FORM

There is an intuitive graphical solution method to linear programs in standard form. Figure 2.20 shows how this solution looks. We first have to draw the three inequality constraints. All x_1 - x_2 combinations that satisfy such a constraint lie below the line on which equality holds. In our example, these equality lines can be computed as

$$\begin{aligned}x_2 &= 100 - x_1 \\x_2 &= 160 - 4x_1 \\x_2 &= 110 - 2x_1.\end{aligned}$$

They are depicted as solid, dashed, and dash-dotted grey lines in the left part of the figure, respectively. The area, in which all of the three inequalities hold simultaneously, i.e. where x_1 and x_2 lie below all of the three lines, is the grey shaded area. Note that also $x_1, x_2 \geq 0$ has to hold. The grey shaded area therefore represents our feasible set of x_1 - x_2 combinations.

Next, we look at the iso-profit lines of the company. An iso-profit line is the set of all x_1 - x_2 combinations leading to the same profit $\bar{\Pi}$, i.e.

$$x_2 = \frac{\bar{\Pi}}{40} - 3x_1.$$

The iso-profit line for $\bar{\Pi} = 2400$ is the black line in the right part of Figure 2.18. The further the line is to the right the higher the y -intercept and the higher the company's profits are. We therefore take the iso-profit line and make a parallel shift to the right. We can do this up to the point where we touch the border of our feasible solution set. The point marked in black, where the iso-profit line exactly touches this set, is then

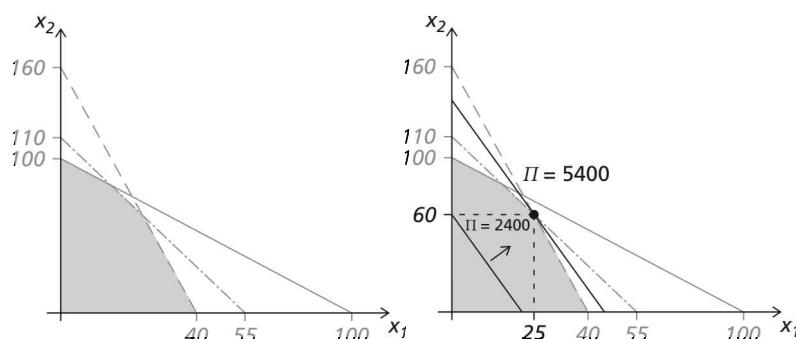


Figure 2.20 Graphical solution to linear programs in standard form

our optimal solution, as it guarantees the highest profit, given that x_1 and x_2 satisfy the three inequality conditions. Note that our optimum is defined by the intersection point between the dashed and the dash-dotted line, i.e. constraints two and three hold with equality. We therefore can compute our optimum via calculating the intersection point between the two equality lines, i.e. setting

$$\begin{aligned} 160 - 4x_1 &= 110 - 2x_1 \\ \Rightarrow x_1^* &= 25 \quad \text{and} \quad x_2^* = 60. \end{aligned}$$

Note that, due to $x_1^* + x_2^* = 85 < 100$, constraint one is not binding. The maximum profit the company makes finally is

$$\Pi^* = 120x_1^* + 40x_2^* = 5400.$$

2.7.2 THE SIMPLEX ALGORITHM

The graphical solution to a linear program is quite intuitive and easy to handle if the program is given in standard form and there are only two choice variables. However, in practice, the number of variables and constraints is usually quite high. In addition, beneath inequality constraints we can also have equality constraints in our program. We therefore define the *canonical form* of a linear program as

$$\min_x c^T x \quad \text{s.t.} \quad x \geq 0 \quad \text{and} \quad Ax = b.$$

There is a quite efficient and often-used algorithm for solving linear programs in canonical form called the *simplex algorithm* invented by George Danzig in 1947. The idea behind this algorithm is as follows: From the above figure we can already guess that the solution to a linear program must be a so-called *corner* of the set of feasible solutions. Corners, in the above problem, are points where two equality lines (including the axes) intersect. The simplex algorithm therefore basically checks all corners of the set of feasible solutions and chooses the corner point that minimizes the objective function.

The problem with the simplex algorithm is that it only applies to linear programs in canonical form, not in standard form. In general, however, linear programs will not only consist of equality constraints. Nevertheless, any linear program with inequality constraints can be turned into a canonical form problem via the following transformation: Assume the general linear program

$$\min_{x_1, \dots, x_n} c_1 x_1 + c_2 x_2 + \dots + c_n x_n$$

with the set of constraints

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n &\leq b_1 \\ &\vdots \\ a_{k1}x_1 + a_{k2}x_2 + \dots + a_{kn}x_n &\leq b_k \\ a_{k+1,1}x_1 + a_{k+1,2}x_2 + \dots + a_{k+1,n}x_n &= b_{k+1} \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

with k inequality constraints. For simplicity we've dropped out greater-than-or-equality constraints. Note, however, that we can turn any of those constraints into a lower-than-or-equality constraint by just multiplying the whole constraint with -1 . In the above linear program, we can now introduce so-called *slack variables* $y_1, y_2, \dots, y_k \geq 0$. If we reformulate the program as

$$\min_{x_1, \dots, x_n, y_1, \dots, y_k} c_1x_1 + c_2x_2 + \dots + c_nx_n$$

with

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n + y_1 &= b_1 \\ &\vdots \\ a_{k1}x_1 + a_{k2}x_2 + \dots + a_{kn}x_n + y_k &= b_k \\ a_{k+1,1}x_1 + a_{k+1,2}x_2 + \dots + a_{k+1,n}x_n &= b_{k+1} \\ a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

we obtain a linear program in canonical form to which we can apply the simplex algorithm. Note that the slack variables do not change the objective function. If a slack variable y_i in the optimum is strictly greater than zero, this indicates that the respective constraint is not binding, like constraint one in the above example.

The simplex algorithm is implemented in subroutine `solve_lin` in the toolbox. Program 2.20 demonstrates how to use it. First we have to define the parameters of the linear program. The constraints are represented by a matrix `A` and a vector `b` as shown above. Note that the constraints have to be sorted in a certain way. The first constraints have to be the lower-than-or-equality constraints. Afterwards we have the greater-than-or-equality constraints and finally the strict equalities. The subroutine `solve_lin` then receives the variable vector `x` which does not have to be initialized as it is a return-only vector, the coefficients `c` which have to be negative in order to maximize profits, the matrix and vector representing the constraints, and three integers indicating how many lower-than-or-equality, greater-than-or-equality, and strict equality constraints the problem has. Obviously, the three integers have to add up to the total number of constraints. The routine will then take all three constraints as lower-than-or-equality constraints. After having applied the simplex algorithm to the problem, `solve_lin` stores the result in the

Program 2.20 Simplex algorithm for linear programs

```

program simplex_alg

use toolbox

implicit none
real*8 :: c(2), x(2), A(3, 2), b(3)
integer :: j

! set up matrix, target vector and coefficients
A(1, :) = (/ 1d0, 1d0/)
A(2, :) = (/ 4d0, 1d0/)
A(3, :) = (/20d0, 10d0/)

b(:) = (/100d0, 160d0, 1100d0/)

c(:) = (/ -120d0, -40d0/)

! solve linear program
call solve_lin(x, c, A, b, 3, 0, 0)

! output
write(*,'(/a,2f10.2)')'      x = ',(x(j), j=1,2)
write(*,'(/a,f10.2)')' Cons 1 = ', b(1)-sum(A(1, :)*x)
write(*,'(a,f10.2)')' Cons 2 = ', b(2)-sum(A(2, :)*x)
write(*,'(a,f10.2)')' Cons 3 = ', b(3)-sum(A(3, :)*x)

end program

```

vector x . Note that the routine will add slack variables automatically in order to transform the given linear program into standard form. The values of slack variables will not be returned. However, we can calculate them by hand using the given constraints as can be seen from the output part of the program.

2.8 Further reading

There are many textbooks on numerical methods in mathematical science, however, they are often hard to understand for economists who are not trained in rigorous mathematics. Demmel (1997) as well as Stoer and Bulirsch (2002) provide an introduction to numerical analysis written for advanced undergraduates and for those beginning graduate courses in applied scientific disciplines. In both books the material of this chapter is presented in much more detail with extended proofs, examples, and descriptions of practical algorithms for computation. Dadkhah (2011) also is a useful textbook for undergraduates with many examples which are solved numerically using Matlab, Maple, or Excel. Judd (1998) as well as Miranda and Fackler (2002) are more advanced textbooks for graduate students. Finally, Press et al. (2001) contains procedures and functions written in Fortran that can be applied to numerical problems.

2.9 Exercises

2.1. Consider the matrix

$$A = \begin{bmatrix} 1 & 5 & 2 & 3 \\ 1 & 6 & 8 & 6 \\ 1 & 6 & 11 & 2 \\ 1 & 7 & 17 & 4 \end{bmatrix}.$$

- (a) Compute the matrices L and U applying the Gaussian elimination method by hand. Check your results using the subroutine `lu_dec(A, L, U)` from the toolbox. Finally, inspect the matrix product of L and U .
- (b) Solve the linear equation system

$$Ax = b, \quad \text{with } b = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

using the matrix decomposition from the previous exercise. Check your results using subroutine `lu_solve(A, b)` from the toolbox. Again inspect the matrix product of A and x .

2.2. Consider the following intertemporal household problem: The utility function of the household is given by

$$U(c_1, c_2) = \frac{c_1^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \frac{c_2^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

with c_1 and c_2 denoting consumption in the first and the second period, respectively. γ is the intertemporal elasticity of substitution and β defines the time discount factor. The household receives labour income w in the first period and does not work in the second period. Consequently, the budget constraint is

$$c_1 + \frac{c_2}{1+r} = w,$$

where r defines the interest rate.

- (a) Define the Lagrangian for this specific optimization problem and derive the first-order conditions with respect to c_1 , c_2 , and λ . Solve the equation system analytically using parameter values $\gamma = 0.5$, $\beta = 1$, $r = 0$, and $w = 1$.

- (b) Solve the equation system resulting from a) using function `fzero` from the toolbox. Print the results and compare the numerical results with the analytical solutions.
- (c) Solve the household problem using the subroutine `fminsearch` and compare the results.
- 2.3. Write a program which computes the global minimum of the function $f(x) = x \cos(x^2)$ on the interval $[0, 5]$ using *Golden Search*. Proceed using the following steps:
- Write a function `minimize(a, b)` of type `real*8`, which computes the local minimum of $f(x)$ using *Golden Search* on the interval $[a, b]$.
 - Next split up the interval $[0, 5]$ in n subintervals $[x_i, x_{i+1}]$ of identical length and compute for each interval the local minimum using the function `minimize`.
 - Finally, your program has to sort out the global minimum from the set of computed local minima using the function `minloc(array, 1)`.
- Test your program with different values for n . How many subintervals are necessary to find the global minimum?
- 2.4. Consider the following intertemporal household optimization problem: The utility function of the household is given by

$$U(c_1, c_2, c_3) = \sum_{i=1}^3 \beta^{i-1} u(c_i) \quad \text{with} \quad u(c_i) = \frac{c_i^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}},$$

where c_i defines consumption in period i of life and β denotes the time discount rate. Assume that the household receives labour income w in the first two periods and consumes all savings in the third period, so that the budget constraint reads

$$\sum_{i=1}^3 \frac{c_i}{(1+r)^{i-1}} = \sum_{i=1}^2 \frac{w}{(1+r)^{i-1}},$$

where r defines the interest rate.

Solve for the optimal consumption levels using the subroutine `fminsearch` from the toolbox. Proceed using the following steps:

- Substitute the budget constraint in the utility function so that it depends only on c_2 and c_3 .
- Minimize the function $-\tilde{U}(c_2, c_3)$ in order to get the values c_2^* and c_3^* .
- Finally, derive c_1^* from the budget constraint.

Use the same parameter values as in Exercise 2 to test your program. Then increase the interest rate and the wage rate separately. Explain your results in economic terms. What will happen when wages are different in both periods? What happens when you alter β ?

- 2.5. Write a program which approximates the integral of $\int_a^b f(x)dx$ using the trapezoid rule, the Simpson rule, and the Gauss-Legendre quadrature method.

- (a) Assume $f(x) = \exp(-x)$.
- (b) Now assume $f(x) = |x|^{0.5}$.

Test your program in the interval $[a; b] = [-1, 1]$ with $n = 10, 20$, and 30 nodes and compare the results with the analytical calculation. What are the differences?

- 2.6. Consider the following inverse demand function

$$p(d) = \frac{4}{(d+1)^2}$$

for a good d . Assume that the price of the good falls from $p = 3$ to $p = 1$. Compute the relative change in the consumer surplus using the trapezoid rule, the Simpson rule, and the Gauss-Legendre quadrature method with $n = 10$.

- 2.7. The government has increased tax rates on labour income during the last years substantially, but the revenues have evolved as follows:

| Year | τ | Tax revenue (in bn.) |
|------|--------|----------------------|
| 2012 | 37 | 198.875 |
| 2013 | 42 | 199.500 |
| 2014 | 45 | 196.875 |

You should help the government to find the revenue maximizing tax rate. Therefore write a program that interpolates the tax function $T(\tau)$ given the data from the above table using polynominal interpolation. Print the resulting tax function in the interval $[35; 45]$ and compute the revenue-maximizing tax rate as well as the corresponding tax revenue.

Why is it not possible to use the subroutines for piecewise linear or cubic spline interpolation here?

- 2.8. Write a program which plots the function $\cos(x)$ on the interval $[0; 2\pi]$ using linear interpolation. In order to do this split up the total interval into $n + 1$ equidistant nodes $x_i, i = 0, \dots, n$ and compute in each subinterval $[x_{i-1}; x_i]$ the linear equation $f(x) = m_i \cdot x + t_i$ with

$$m_i = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \quad \text{and} \quad t_i = f(x_{i-1}) - m_i \cdot x_{i-1},$$

where x_{i-1} and x_i denote the lower and upper bound of the respective subinterval i . Then draw n graphs that show the linear approximation of the cosine function. Compare your linear interpolation results with the true cosine function for $n = 10, 20$, and 30 . Now approximate the function using the piecewise linear

interpolation subroutine `linint_Equi` as well as the spline approach and compare the results with the true cosine function.

- 2.9. Consider a Cournot oligopoly market with m identical companies. Given the demand curve $D(P) = P^{-\eta}$, each company produces output $q(P)$ by equating marginal benefits and marginal costs. Marginal benefits are given by $P + q \frac{dp}{dq}$ and marginal costs are assumed to be given by $c(q) = \alpha \sqrt{q} + q^2$. Due to the Cournot assumption each firm expects its competitors to not react to our changes in output. Consequently, we have $\frac{dp}{dq} = \frac{1}{D'(P)}$.

- Set up an equidistant price grid $P(0:N)$ on the interval $[0.1; 3.0]$ using subroutine `grid_Cons_Equi`, initialize spline coefficients `coeff_q=0` and derive the respective individual quantities $q(0:N)$ at each gridpoint using function `fzero`.
- Interpolate the individual supply function q using function `spline_interp` with the coefficients `coeff_q`. Use `spline_eval` to calculate supply in between the points specified in $P(0:N)$. Print market demand and the supply curve.
- Compute the equilibrium price and quantity using function `fzero` as well as the interpolated aggregate supply function.
- Change the number m of companies as well as the price elasticity η and α . Explain your results in economic terms.

The initial parametrization is $\alpha = 1$, $\eta = 1.5$, $m = 3$, $N = 10$, $NP = 1000$.

- 2.10. Now consider a local newspaper which is a monopoly in a region. It serves two distinct customer groups: readers and advertisers. The price of the newspaper p_R is irrelevant for the advertiser, as are the advertising prices p_A for the reader. However, since a higher newspaper price induces lower sales of newspapers, demand for ads will typically fall. On the other hand it is not clear how the demand for newspapers is affected by higher or lower advertising. In order to find out the optimal price combination (p_R^*, p_A^*) that maximizes profits, the managers of the newspaper vary the two prices and observe the following profits $G(p_R, p_A)$:

| | | p_A | | | |
|-------|------|-------|------|-------|-------|
| | | 0.5 | 4.5 | 8.5 | 12.5 |
| p_R | 0.5 | 11.5 | 70.9 | 98.3 | 93.7 |
| | 4.5 | 31.1 | 82.5 | 101.9 | 89.3 |
| | 8.5 | 18.7 | 62.1 | 73.5 | 52.9 |
| | 12.5 | -25.7 | 9.7 | 13.1 | -15.5 |

- Given this information, construct a two-dimensional spline approximation of the profit function. What is the optimal price combination (p_R^*, p_A^*) and the resulting optimal profit $G(p_R^*, p_A^*)$? Hint: Set up two equidistant price grids $p_R(0:3)$ and $p_A(0:3)$ on the interval $[0.5; 12.5]$ using subroutine `grid_Cons_Equi`. Given profits $G(0:3, 0:3)$ the spline coefficients `coeff_G`

can be derived using subroutine `spline_interp(G, coeff_G)`. Next, evaluate the profit function `Gplot(0:Nplot, 0:Nplot)` using function `spline_eval` at each grid point. Finally, find the location of the maximum profit using function `maxloc`.

- (b) Marginal costs for both newspapers and advertisements are constant at $c = 0.1$. In addition, the managers have derived the ‘true’ demand functions

$$x_R = 10 - p_R \quad \text{and} \quad x_A = 20 - p_A - 0.5p_R$$

for newspapers and advertisements, respectively. Compute the profit-maximizing price combination using either `fminsearch` or `fzero` and compare with the approximated solution.

- (c) Use $N_{\text{plot}} = 100, 1000, 10000$ and compare the approximation error.
 2.11. Three gravel-pits A_1, A_2 and A_3 store 11 tons, 13 tons, and 10 tons of gravel, respectively. The gravel is used at four building sites B_1, B_2, B_3 , and B_4 . B_1 orders 5 tons, B_2 7 tons, B_3 13 tons, and B_4 6 tons of gravel. The transport cost of one ton of gravel from pit A_i to the building site B_j are displayed in the following table.

| | B_1 | B_2 | B_3 | B_4 |
|-------|-------|-------|-------|-------|
| A_1 | 10 | 70 | 100 | 80 |
| A_2 | 130 | 90 | 120 | 110 |
| A_3 | 50 | 30 | 80 | 10 |

Minimize the total cost of transport from gravel pits to the building sites using a simplex algorithm.

Part II

Computational Economics

for Beginners

3 The static general equilibrium model

Our first application is the static general equilibrium model. The model is called *static* since intertemporal aspects, such as savings and investment, are excluded by assumption. Consequently the capital stock of the economy is constant and we concentrate on the intersectoral allocation of resources and labour-leisure choices. In this chapter, we start the discussion with the most simple model structure and then successively introduce additional complexities, like government activity, intermediate goods production, or international trade.

3.1 The basic economy model

The most simple model structure comprises a closed economy with a representative consumer who supplies labour and capital in fixed amounts which are used by firms to produce two consumption goods. Section 3.1.1 develops the so-called ‘command optimum’ using this basic model. This is the allocation that would be chosen by a social planner who knows endowments, technologies, and preferences. After that we compare the command optimum to the allocation in a market economy. Then we introduce variable labour supply and government activities.

3.1.1 THE COMMAND OPTIMUM

The representative household supplies his endowment of capital \bar{K} and labour \bar{L} to the firms which use these inputs to produce output Y_1 and Y_2 of goods 1 and 2. Household consumption is denoted by X_1 and X_2 respectively. The economic problem is to allocate the scarce factor resources K and L to the two different types of firms in order to produce an output combination which maximizes the utility of the representative household. The latter is called an *efficient allocation*. In order to find the efficient allocation we first have to specify preferences and technologies. In order to simplify the analysis we assume Cobb-Douglas preferences and technologies, i.e.

$$\begin{aligned}U(X_1, X_2) &= X_1^\alpha X_2^{1-\alpha} \\Y_i &= L_i^{\beta_i} K_i^{1-\beta_i} \quad i = 1, 2.\end{aligned}$$

In order to derive the command optimum we formulate the allocation problem as a constrained maximization problem. As the representative household's utility is to be maximized, the objective function is the utility function and the constraints are technologies and resource endowments, i.e.

$$\max_{X_1, X_2} X_1^\alpha X_2^{1-\alpha} \quad \text{s.t.} \quad X_1 = L_1^{\beta_1} K_1^{1-\beta_1}, \quad X_2 = L_2^{\beta_2} K_2^{1-\beta_2}$$

$$\bar{L} = L_1 + L_2 \quad \text{and} \quad \bar{K} = K_1 + K_2.$$

The above formulation already assumes that all produced quantities are consumed, i.e. $X_i = Y_i$, $i = 1, 2$. The last assumption makes sense economically, since otherwise some quantities would be wasted. For the formal derivation of the command optimum the constraints are substituted into the objective function so that the optimization problem becomes

$$\max_{L_1, K_1} U(L_1, K_1) = \left[L_1^{\beta_1} K_1^{1-\beta_1} \right]^\alpha \left[(\bar{L} - L_1)^{\beta_2} (\bar{K} - K_1)^{1-\beta_2} \right]^{1-\alpha}.$$

Figure 3.1 shows the graphical solution to the command optimum. Given factor endowments and technologies, it is possible to derive an efficiency locus in the factor space box on the left side where two production isoquants have the same slope. Since each point on the efficiency locus defines a specific output combination, it can be transferred to the transformation curve in the product space on the right-hand side. The command optimum then is the point on the transformation curve where the utility of the representative consumer is maximized.

For the numerical solution we specify endowments $\bar{K} = 10$ and $\bar{L} = 20$ as well as preference and technology parameters $\alpha = 0.3$, $\beta_1 = 0.3$, $\beta_2 = 0.6$. Program 3.1 then computes the solution to the problem using the minimization routine `fminsearch`. We thereby use a module `globals` that stores our model parameters as well as the `real*8`

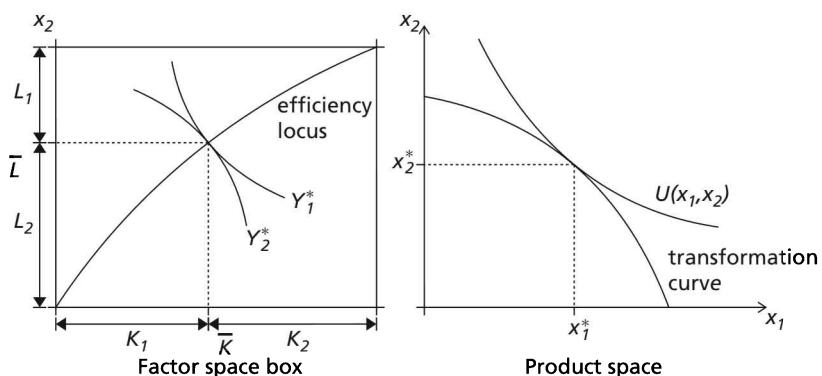


Figure 3.1 The command optimum

Program 3.1 Planner solution to the static equilibrium model

```

program planner

use globals
use toolbox

implicit none
real*8 :: x(2), L(2), K(2), Y(2), fret

! initial guess
x(:) = 5d0

! minimization routine
call fminsearch(x, fret, (/0d0, 0d0/), (/10d0, 20d0/), utility)

! solution
K(1) = x(1)
L(1) = x(2)
K(2) = Kbar - K(1)
L(2) = Lbar - L(1)
Y    = L**beta*K** (1d0-beta)

! output
[.....]
end program

```

function `utility` that returns the value of the utility function depending on L_1 and K_1 . Due to space restrictions, however, we will show neither of them here.

The solution for the command optimum is

$$\begin{array}{lll} L_1 = 3.53 & K_1 = 4.29 & X_1 = 4.04 \\ L_2 = 16.47 & K_2 = 5.71 & X_2 = 10.78. \end{array}$$

The next step could be to change technology or preference parameters in order to check how it affects the allocation. However, we proceed in a different direction and compute the allocation of a market economy.

3.1.2 THE MARKET SOLUTION

In order to derive the command optimum it was assumed that some central planner knows the technologies, preferences, and resource endowments. Of course, in reality there is no such central planner. Usually households¹ know their own endowments and preferences and firms know their own applied technologies, however not all agents

¹ Throughout this book we use the words household, consumer, agent, worker, and sometimes also individual synonymously to refer to a decision unit in the household sector of an economy, regardless of its composition.

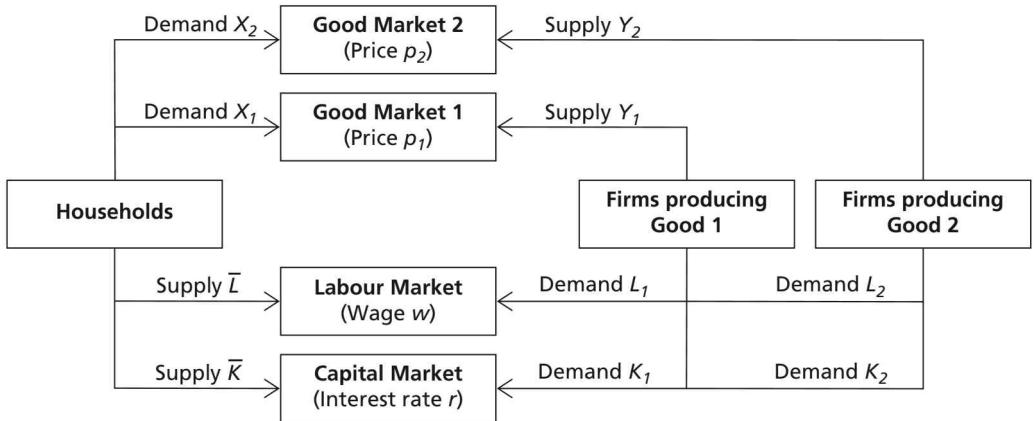


Figure 3.2 Market system in the static general equilibrium model

know everything about the endowments, preferences, and technologies of their respective competitors. At first sight one could expect that the resource allocation in such a situation is quite random and therefore very inefficient. However, as this section demonstrates, the interaction of supply and demand on goods and factor markets leads to an efficient allocation of resources despite the fact that decisions of economic agents are decentralized. The driving force behind this surprising result is the price system, which guarantees an efficient information exchange.

In the present model, two goods and two production factors are exchanged between firms and households. We therefore need four markets. With respect to prices one has to distinguish between consumer prices p_i and producer prices q_i for the two consumer goods and wage w and interest rate r for labour and capital inputs, respectively. We assume perfect competition, i.e. prices are the same for all consumers and firms in the model and firms do not make profits. Figure 3.2 shows the market system of our economy.

Given consumer and factor prices, the representative household maximizes utility by choosing consumption demand X_i and keeping in mind the budget constraint

$$\max_{X_1, X_2} X_1^\alpha X_2^{1-\alpha} \quad \text{s.t.} \quad p_1 X_1 + p_2 X_2 = w \bar{L} + r \bar{K} := \bar{Y}.$$

Taking derivatives with respect to X_1 and X_2 , we can calculate demand functions as

$$X_1 = \frac{\alpha}{p_1} \bar{Y} \quad \text{and} \quad X_2 = \frac{1-\alpha}{p_2} \bar{Y}. \quad (3.1)$$

In the production sector firms maximize their profits Π_i given their technology constraint as well as producer and factor prices

$$\max_{L_i, K_i} \Pi_i = q_i Y_i - w L_i - r K_i \quad \text{s.t.} \quad Y_i = L_i^{\beta_i} K_i^{1-\beta_i}.$$

Again taking derivatives yields the input demand functions for firms producing good i

$$L_i = \frac{\beta_i}{w} q_i Y_i \quad \text{and} \quad K_i = \frac{1-\beta_i}{r} q_i Y_i. \quad (3.2)$$

Since the simple economy abstracts from government activities, producer prices and consumer prices are identical, i.e. $q_i = p_i$. In the market economy, goods and factor prices adjust until supply equals demand in all markets, i.e.

$$X_i = Y_i, \quad i = 1, 2, \quad L_1 + L_2 = \bar{L} \quad \text{and} \quad K_1 + K_2 = \bar{K}. \quad (3.3)$$

Since demand functions are homogenous of degree zero, only relative prices matter for the equilibrium. Without loss of generality, it is possible to normalize the price of the first good (i.e. $q_1 = p_1 = 1$) and vary the remaining three prices p_2, w, r . Consequently, one only needs to consider three markets. When choosing three markets out of four, one has to make sure that all four prices are used in the equations. This requirement is satisfied with two goods markets and the labour market. After substituting factor demand functions (3.2) and consumer demand functions (3.1) and rearranging the goods and factor market equilibrium conditions (3.3), we obtain

$$\frac{1}{p_1} - \left(\frac{\beta_1}{w} \right)^{\beta_1} \left(\frac{1-\beta_1}{r} \right)^{1-\beta_1} = 0 \quad (3.4)$$

$$\frac{1}{p_2} - \left(\frac{\beta_2}{w} \right)^{\beta_2} \left(\frac{1-\beta_2}{r} \right)^{1-\beta_2} = 0 \quad (3.5)$$

$$\frac{\beta_1}{w} \alpha \bar{Y} + \frac{\beta_2}{w} (1-\alpha) \bar{Y} - \bar{L} = 0 \quad (3.6)$$

Note that despite the fact that all four prices p_1, p_2, w, r are considered, the capital market seems to be neglected. However, implicitly, it is also in equilibrium, since combining the budget constraint of the household $p_1 X_1 + p_2 X_2 = w \bar{L} + r \bar{K}$ and the zero-profit condition of firms $p_i Y_i = w L_i + r K_i$ yields

$$p_1 (X_1 - Y_1) + p_2 (X_2 - Y_2) = w (\bar{L} - L_1 - L_2) + r (\bar{K} - K_1 - K_2).$$

This specific feature of our system is called *Walras' law*. It says that if $n - 1$ markets out of n markets are in equilibrium, then the last market is automatically balanced. It would be no problem to substitute the capital market and get rid of the labour market equilibrium. However, in this special case it is not possible to substitute one of the two goods markets,

Program 3.2 Market solution to the static equilibrium model

```
program market1
[....]
! initial guess
x(:) = 0.5d0

! find market equilibrium
call fzero(x, markets, check)

! check whether fzero converged
if(check)then
    write(*, '(a/)''Error in fzero !!!'
    stop
endif
[....]
end program
```

Module 3.2m Function to determine the market solution

```
function markets(x)
[....]
! copy prices
p(1) = 1d0
p(2) = x(1)
w = x(2)
r = x(3)

! calculate total income
Ybar = w*Lbar+r*Kbar

! get market equations
markets(1) = 1d0/p(1) - (beta(1)/w)**beta(1)* &
((1d0-beta(1))/r)**(1d0-beta(1))
markets(2) = 1d0/p(2) - (beta(2)/w)**beta(2)* &
((1d0-beta(2))/r)**(1d0-beta(2))
markets(3) = beta(1)*alpha*Ybar/w+beta(2)*(1-alpha)*Ybar/w-Lbar

end function
```

since only there the goods prices are used. For the computation of equilibrium prices in our market system (3.4) to (3.6), Program 3.2 uses the root-finding algorithm `fzero`. This routine calculates the root of the market equations using Broyden's method. The market equations are provided in the function `markets`, which receives a three-dimensional price vector. This function is stored in Module 3.2m. The entries in the price vector `x` are price for good 2 p_2 , wage w , and interest rate r . Having copied the values of the vector to the respective variables, the function calculates (3.4) to (3.6) and returns the respective values. Having determined the market prices p_2 , w , and r , the program checks whether `fzero` has actually converged. If this is the case, we can calculate all economic variables of our model via (3.1), (3.2), and the production technology.

Comparing the results from market and planner solutions, we find that they yield exactly the same quantities. Given the normalization $p_1 = 1$, the remaining prices are $p_2 = 0.87$, $w = 0.34$, and $r = 0.66$.

3.1.3 VARIABLE LABOUR SUPPLY

Up until now it has been assumed that households supply a fixed quantity of labour \bar{L} on the market. In reality, however, they have to decide when to enter and leave the labour market (extensive labour supply) and how much they want to work during a specific time span (intensive labour supply). Consequently, it is useful to introduce a labour-supply decision on the household side. In the most simple case, leisure consumption is introduced as an additional consumption good. The household then maximizes utility by choosing consumption demand X_i and leisure demand ℓ , keeping in mind the budget and time constraint

$$\max_{X_1, X_2, \ell} X_1^{\alpha_1} X_2^{\alpha_2} \ell^{1-\alpha_1-\alpha_2} \quad \text{s.t.} \quad p_1 X_1 + p_2 X_2 + w\ell = w\bar{T} + r\bar{K} := \bar{Y},$$

where \bar{T} denotes the time endowment of the household. As before we derive the demand functions

$$X_1 = \frac{\alpha_1}{p_1} \bar{Y}, \quad X_2 = \frac{\alpha_2}{p_2} \bar{Y} \quad \text{and} \quad \ell = \frac{1 - \alpha_1 - \alpha_2}{w} \bar{Y}.$$

The labour-supply decision has only a small impact on the market equilibrium conditions. The two goods markets (3.4) and (3.5) remain unchanged. Only the new labour market equilibrium condition $L_1 + L_2 = \bar{T} - \ell$ changes into

$$\frac{\beta_1}{w} \alpha_1 \bar{Y} + \frac{\beta_2}{w} \alpha_2 \bar{Y} + \frac{1 - \alpha_1 - \alpha_2}{w} \bar{Y} - \bar{T} = 0. \quad (3.7)$$

The function we want to set equal to zero therefore changes to the one shown in Module 3.3m. We thereby set $\bar{T} = 30$ and the new preference parameters $\alpha_1 = 0.3$ and $\alpha_2 = 0.4$.

Module 3.3m Model with variable labour supply

```

function markets(x)
    [.....]
    ! copy prices
    p(1) = 1d0
    p(2) = x(1)
    w    = x(2)
    r    = x(3)

    ! calculate total income and consumer prices
    Ybar = w*Tbar+r*Kbar

    ! get market equations
    markets(1) = 1d0/p(1)-(beta(1)/w)**beta(1)* &
                ((1d0-beta(1))/r)**(1d0-beta(1))
    markets(2) = 1d0/p(2)-(beta(2)/w)**beta(2)* &
                ((1d0-beta(2))/r)**(1d0-beta(2))
    markets(3) = beta(1)*alpha(1)*Ybar/w+beta(2)*alpha(2)*Ybar/w+ &
                (1d0-alpha(1)-alpha(2))*Ybar/w-Tbar

end function

```

3.1.4 PUBLIC SECTOR AND TAX INCIDENCE ANALYSIS

Now we would like to introduce government activity into our model. The government provides an exogenous level of the public good G and levies consumption and income taxes in order to finance it. For simplicity it is assumed that the public good is equivalent with Good 1, but it would be no problem to specify a separate production technology. Consequently, one has to distinguish between consumer and producer prices as well as between gross and net factor prices

$$p_i = q_i(1 + \tau_i), \quad w^n = w(1 - \tau_w) \quad \text{and} \quad r^n = r(1 - \tau_r),$$

where τ_i denote consumption tax rates and τ_w, τ_r define wage and interest income tax rates, respectively. Figure 3.3 shows the model with government activity.

The budget constraint of consumers is now defined by

$$p_1 X_1 + p_2 X_2 + w^n \ell = w^n \bar{T} + r^n \bar{K} := \bar{Y}^n$$

yielding the new consumption and leisure demand functions

$$X_1 = \frac{\alpha_1}{p_1} \bar{Y}^n, \quad X_2 = \frac{\alpha_2}{p_2} \bar{Y}^n \quad \text{and} \quad \ell = \frac{1 - \alpha_1 - \alpha_2}{w^n} \bar{Y}^n. \quad (3.8)$$

On the producer side, factor demand functions (3.2) do not change. Since the government demands a share of the first good, the market equilibrium condition in the market for

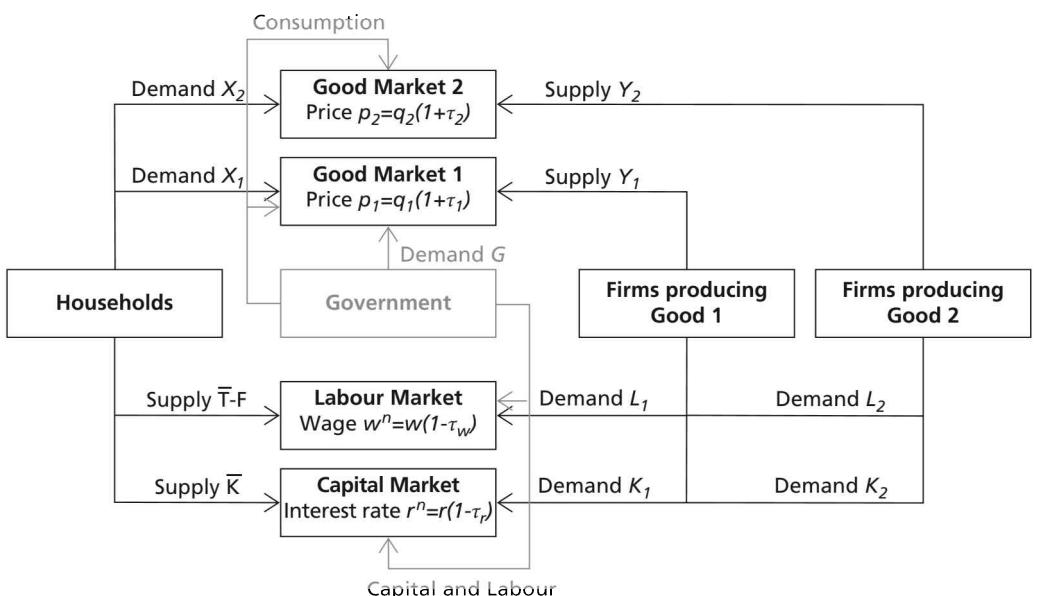


Figure 3.3 Static general equilibrium with government

consumption Good 1 changes to $X_1 + G = Y_1$. The equilibrium condition on the second goods market remains unchanged, but one has to keep in mind the difference between producer and consumer prices. Consequently, the two goods market equilibrium conditions (3.4) and (3.5) change to

$$\frac{\alpha_1 \bar{Y}^n}{p_1} + G - \left(\frac{\beta_1}{w} \right)^{\beta_1} \left(\frac{1 - \beta_1}{r} \right)^{1-\beta_1} q_1 \left(\frac{\alpha_1 \bar{Y}^n}{p_1} + G \right) = 0$$

$$1 - \left(\frac{\beta_2}{w} \right)^{\beta_2} \left(\frac{1 - \beta_2}{r} \right)^{1-\beta_2} q_2 = 0.$$

The labour market equilibrium condition (3.7) now changes to

$$\frac{\beta_1}{w} q_1 \left(\frac{\alpha_1 \bar{Y}^n}{p_1} + G \right) + \frac{\beta_2}{w} q_2 \frac{\alpha_2 \bar{Y}^n}{p_2} + \frac{1 - \alpha_1 - \alpha_2}{w^n} \bar{Y}^n - \bar{T} = 0.$$

Finally, in equilibrium, all government outlays have to be financed by taxes, i.e.

$$q_1 G = \sum_{i=1}^2 \tau_i q_i X_i + \tau_w w (\bar{T} - \ell) + \tau_r r \bar{K},$$

which yields—after substitution—the government budget equilibrium condition

$$q_1 G - \sum_{i=1}^2 \frac{\tau_i}{1 + \tau_i} \alpha_i \bar{Y}^n - \tau_w w \left(\bar{T} - \frac{1 - \alpha_1 - \alpha_2}{w^n} \bar{Y}^n \right) - \tau_r r \bar{K} = 0,$$

where always one tax rate is endogenous and the remaining three other tax rates are exogenous. Given G and the pre-specified exogenous tax rates, it is possible to solve the system of four equations (three markets and the government constraint) and four unknowns (p_2 , w , r , and the endogenous tax rate). The function we want to set to zero is shown in Module 3.4m.

In order to get an intuition of welfare effects of different tax structures in the above model, Table 3.1 reports the results of some simulations with alternative tax rates. In each simulation, some tax rates are fixed and at least one tax rate is endogenously adjusted in order to raise the required tax revenue to finance the public good. The first line reports the first-best optimum where only lump-sum taxes are levied and the tax system does not distort individual decisions. As one can see in the next line, the first-best solution can also be achieved with very high consumption taxes to finance public goods and subsidize labour supply. The next simulation considers a pure income tax system. Since taxes now distort labour supply, leisure consumption increases dramatically, which drives up wages, so that the household substitutes towards the less labour-intensive Good 1. As a consequence, welfare in the last column declines compared to the first-best

Module 3.4m Model with government activity

```

function markets(x)
    [.....]
    ! copy producer prices and taxes
    q(1)      = 1d0
    q(2)      = x(1)
    w         = x(2)
    r         = x(3)
    taur     = x(4)

    ! calculate consumer prices and total income
    p          = q*(1d0+tauc)
    wn        = w*(1d0-tauw)
    rn        = r*(1d0-taur)
    Ybarn = wn*Tbar+rн*Kbar

    ! get market equations
    markets(1) = alpha(1)*Ybarn/p(1)+G-(beta(1)/w)**beta(1)* &
                ((1d0-beta(1))/r)**(1d0-beta(1))*q(1)*(alpha(1)*Ybarn/p(1)+G)
    markets(2) = 1d0/p(2)-(beta(2)/w)**beta(2)* &
                ((1d0-beta(2))/r)**(1d0-beta(2))*q(2)/p(2)
    markets(3) = beta(1)/w*q(1)*(alpha(1)*Ybarn/p(1)+G) + &
                beta(2)/w*q(2)*alpha(2)*Ybarn/p(2) +
                (1d0-alpha(1)-alpha(2))*Ybarn/wn-Tbar
    markets(4) = q(1)*G-tauc(1)/(1d0+tauc(1))*alpha(1)*Ybarn- &
                tauc(2)/(1d0+tauc(2))*alpha(2)*Ybarn-&
                tauw*w*(Tbar-(1d0-alpha(1)-alpha(2))/wn*Ybarn)-taur*r*Kbar

end function

```

Table 3.1 General equilibrium with different tax structures in the static model

| τ_1 | τ_2 | τ_w | τ_r | w | r | q_2 | X_1 | X_2 | ℓ | U |
|----------|----------|----------|----------|------|------|-------|-------|-------|--------|------|
| 0.00 | 0.00 | 0.00 | 0.43 | 0.31 | 0.69 | 0.83 | 3.93 | 6.31 | 12.88 | 6.78 |
| 0.76 | 0.76 | -0.76 | 0.00 | 0.31 | 0.69 | 0.83 | 3.93 | 6.31 | 12.88 | 6.78 |
| 0.00 | 0.00 | 0.26 | 0.26 | 0.33 | 0.67 | 0.86 | 3.72 | 5.75 | 15.02 | 6.73 |
| 0.35 | 0.35 | 0.00 | 0.00 | 0.33 | 0.67 | 0.86 | 3.72 | 5.75 | 15.02 | 6.73 |
| 0.18 | 0.18 | 0.00 | 0.20 | 0.32 | 0.68 | 0.85 | 3.81 | 5.99 | 14.10 | 6.76 |
| 0.50 | 0.25 | 0.00 | 0.00 | 0.34 | 0.66 | 0.87 | 3.38 | 6.20 | 14.84 | 6.72 |

$$q_1 G = 3, \bar{T} = 30, \bar{K} = 10.$$

allocation. The following simulation demonstrates that a consumption tax system with a uniform tax rate could be equivalent to an income tax. As shown in the following line, welfare increases if the tax burden is shifted towards the lump-sum tax base. Finally, the last simulation of Table 3.1 considers a differentiated consumption tax where welfare decreases the most, since the tax system distorts the consumption-leisure choice and the optimal consumption structure.

This should suffice to get a first idea how this so-called *differential tax incidence* analysis works in numerical general equilibrium models. Of course, one could also vary the level of public good consumption G and compute the resulting welfare effects for different tax structures with a *budget incidence* analysis. In this case, however, utility from public consumption has to be explicitly specified.

3.2 Extensions of the basic model

Of course, the basic model structure could be extended in numerous directions. This section focuses on three specific limitations of the basic model. Subsection 3.2.1 demonstrates how to deal with labour market imperfections. Then we introduce input-output relations by distinguishing between final and intermediate goods. Finally, we consider international trade in a two-country model.

3.2.1 IMPERFECT LABOUR MARKETS AND UNEMPLOYMENT POLICY

Up to now we have abstracted from unemployment issues by assuming full employment and a perfect labour market. The introduction of unemployment in the present model is quite simple through the specification of a so-called ‘wage curve’, which simply postulates a negative relationship between the net real wage rate and the unemployment rate. Such a relationship could be derived as well from trade union models or from efficiency wage models. A very simple specification of the wage curve is a log-linear function such as:

$$\log\left(\frac{w^n}{P}\right) = \gamma_1 - \gamma_2 \log(ur)$$

where $w^n = w(1 - \tau_w - \tau_u)$ now defines the wage rate net of labour taxes and unemployment contributions τ_u and P denotes a consumer price index (which includes consumption taxes). γ_1 is a scale parameter which also reflects institutional features such as real unemployment benefits and $\gamma_2 > 0$ indicates the elasticity of the real net wage with respect to the unemployment rate ur .

Figure 3.4 illustrates the wage curve in the traditional labour market diagram. The vertical axis shows the real wage rate and the horizontal axis shows labour demand and supply. The intersection of the (inverse) labour demand function L and the labour-supply function L^S specifies the full employment equilibrium at the real wage $(w^n/P)_0$ and employment L_0 . However, due to institutional imperfections wages are determined by the wage curve and not the labour-supply curve. Consequently, the equilibrium wage rate $(w^n/P)_1$ lies above the market clearing wage which results in an unemployment rate of $ur = (L_1^S - L_1)/L_1^S$.

From the individual perspective, unemployment risk makes income uncertain. Therefore, the unemployment rate is used to compute expected income. The budget constraint changes to

$$\begin{aligned} p_1 X_1 + p_2 X_2 &= w^n L^S (1 - ur) + P \bar{B} L^S ur + r^n \bar{K} \\ &= (\bar{T} - \ell) \underbrace{[w^n (1 - ur) + P \bar{B} ur]}_{\tilde{w}^n} + r^n \bar{K} \end{aligned}$$

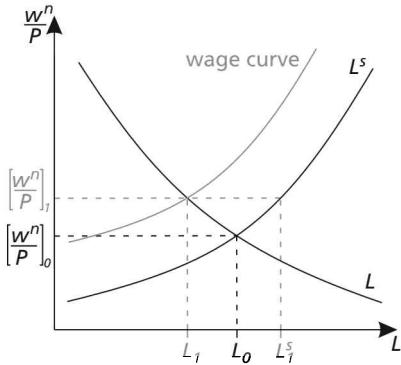


Figure 3.4 Wage curve and unemployment

where \bar{B} denotes the nominal unemployment benefit and \tilde{w}^n defines the expected net wage. After arranging we get

$$p_1 X_1 + p_2 X_2 + \tilde{w}^n \ell = \tilde{w}^n \bar{T} + r^n \bar{K} := \tilde{Y}^n.$$

Of course, the demand functions (3.8) do not change, one only has to replace \tilde{w}^n and \tilde{Y}^n . The labour market equation (3.7) takes into account that only a fraction of labour supply is employed and changes to

$$\frac{\beta_1}{w} \alpha_1 \tilde{Y}^n + \frac{\beta_2}{w} \alpha_2 \tilde{Y}^n + (1 - ur) \left(\frac{1 - \alpha_1 - \alpha_2}{\tilde{w}^n} \tilde{Y}^n - \bar{T} \right) = 0.$$

Of course, we still have to compute the unemployment contribution rate. Since we assume a separate budget for unemployment insurance, contributions of employed households have to match the benefits for the unemployed exactly. Consequently, we get

$$\tau_u = \frac{P \bar{B} u r}{w(1 - ur)}.$$

Compared to the previous model in Module 3.4m we have now five equations (three markets, the government constraint, and the wage curve) and the unemployment rate ur as an additional unknown. The functions which are now set to zero are shown in Module 3.5m.

In order to get an intuition of the unemployment effects of different tax structures in the above model, Table 3.2 reports the results of some simulations with alternative tax rates.

We fix the nominal unemployment benefit level \bar{B} , set some tax rates to zero, and derive the required revenue from the remaining tax. The first line can be directly

Module 3.5m Model with unemployment

```

function markets(x)
[.....]
! copy producer prices and taxes
q(1)      = 1d0
q(2)      = x(1)
w         = x(2)
r         = x(3)
tauw     = x(4)
taur     = x(4)
ur       = x(5)

! calculate consumer prices and total income
p        = q*(1d0+tauc)
PX      = p(1)*alpha(1)/sum(alpha)+p(2)*alpha(2)/sum(alpha)
tauu    = PX*B*ur/(w*(1-ur))
wn      = w*(1d0-tauw-tauu)*(1-ur)+PX*B*ur
rn      = r*(1d0-taur)
Ybarn   = wn*Tbar+rn*Kbar

! get market equations
markets(1) = alpha(1)*Ybarn/p(1)+G-(beta(1)/w)**beta(1)* &
((1d0-beta(1))/r)**(1d0-beta(1))*q(1)*(alpha(1)*Ybarn/p(1)+G)
markets(2) = 1d0-(beta(2)/w)**beta(2)*((1d0-beta(2))/r)** &
(1d0-beta(2))*q(2)
markets(3) = beta(1)/w*q(1)*(alpha(1)*Ybarn/p(1)+G)+ &
beta(2)/w*q(2)*alpha(2)*Ybarn/p(2)+(1-ur)* &
((1d0-alpha(1)-alpha(2))*Ybarn/wn-Tbar)
markets(4) = q(1)*G-tauc(1)/(1d0+tauc(1))*alpha(1)*Ybarn- &
tauc(2)/(1d0+tauc(2))*alpha(2)*Ybarn- tauw*w*(1-ur)* &
(Tbar-(1d0-alpha(1)-alpha(2))/wn*Ybarn)-taur*r*Kbar
markets(5) = log(w*(1-tauw-tauu)/PX)-gamma(1)+gamma(2)*log(ur)

end function markets

```

Table 3.2 General equilibrium with different tax structures and unemployment

| \bar{B} | τ | τ_w | τ_r | τ_u | w | P | ur | X_1 | X_2 | ℓ | U |
|-----------|--------|----------|----------|----------|------|------|------|-------|-------|--------|------|
| 0.1 | 0.00 | 0.00 | 0.44 | 0.02 | 0.32 | 0.91 | 0.06 | 3.84 | 6.06 | 12.85 | 6.62 |
| 0.1 | 0.00 | 0.26 | 0.26 | 0.03 | 0.36 | 0.94 | 0.09 | 3.57 | 5.36 | 15.04 | 6.47 |
| 0.1 | 0.36 | 0.00 | 0.00 | 0.04 | 0.36 | 1.27 | 0.09 | 3.57 | 5.36 | 15.04 | 6.47 |
| 0.2 | 0.00 | 0.27 | 0.27 | 0.06 | 0.36 | 0.94 | 0.10 | 3.56 | 5.33 | 15.05 | 6.45 |

In all simulations we assume $q_1 G = 3$, $\gamma_1 = -2.5$, and $\gamma_2 = 0.5$.

compared to the first-best optimum in the previous Table 3.1. With lump-sum taxes and an unemployment benefit which amounts to roughly one third of the net wage, the unemployment rate is 6 per cent and the respective contribution rate is 2 per cent. The consumption price index is computed as a weighted average of the two consumer prices, i.e.

$$P = \frac{\alpha_1}{\alpha_1 + \alpha_2} p_1 + \frac{\alpha_2}{\alpha_1 + \alpha_2} p_2.$$

Due to unemployment income and consumption is much lower than in the respective simulation of Table 3.1 which in turn explains the lower welfare level. As one can see in the next line, a pure income tax system increases unemployment significantly. Taxes now distort labour supply so that leisure consumption rises, but also the unemployment rate rises due to the wage-curve dynamics. As a consequence, welfare is significantly reduced. The next line demonstrates that the equivalence between income and consumption taxes still (roughly) remains in the imperfect labour market model. Of course, if we specify a different wage curve, a different price index, or a different unemployment benefit system, this equivalence would break down. Finally, in the last line we increase the nominal benefit level \bar{B} which directly increases the contribution rate which in turn drives up the unemployment rate. As a consequence, welfare is further reduced.

3.2.2 INTERMEDIATE GOODS IN PRODUCTION

Up to now it has been assumed that a firm's production is only used for final consumption. In reality, however, firms also produce intermediate inputs for other firms. Consequently, output demand of a specific sector not only depends on final consumption demands but also on the output of all other sectors. Final demand changes in one sector may trigger output effects in all other sectors due to the changes in the demand for intermediate inputs. In order to account for this interdependence of production sectors, the production structure of an economy is modelled by an input-output table, where X_{ij} defines input supplies from sector i to sector j of the economy. Table 3.3 displays the structure of such a stylized IO-Table for the two-sector economy.

Any IO-Table consists of three parts, the intermediate input table in the center, the final demand table on the right side, and the primary factor input table on the bottom. The lines of the intermediate input and the final demand tables define the goods market equilibrium conditions which now change to

$$Y_1 = X_{11} + X_{12} + X_1 + G$$

$$Y_2 = X_{21} + X_{22} + X_2.$$

The columns of the intermediate and primary factor input tables define the zero-profit conditions of the firms, i.e. $q_i Y_i = q_i X_{ii} + q_j X_{ji} + wL_i + rK_i$. In our example we abstract from production taxes so that intermediate input prices are producer prices.

Table 3.3 General structure of an IO-Table in a closed economy

| | | | | |
|--------------|--------------|-----------|-----------|-----------|
| $q_1 X_{11}$ | $q_1 X_{12}$ | $q_1 X_1$ | $q_1 G$ | $q_1 Y_1$ |
| $q_2 X_{21}$ | $q_2 X_{22}$ | $q_2 X_2$ | | $q_2 Y_2$ |
| wL_1 | wL_2 | | | |
| rK_1 | rK_2 | | | |
| | | $q_1 Y_1$ | $q_2 Y_2$ | |

For the production sector we assume that output is produced with the input of intermediate goods X_{ij} and a technology $L_i^{\beta_i} K_i^{1-\beta_i}$ for primary inputs. For simplicity, we assume a Leontief overall production function

$$Y_i = \min \left\{ \frac{L_i^{\beta_i} K_i^{1-\beta_i}}{a_{0i}}, \frac{X_{ii}}{a_{ii}}, \frac{X_{ji}}{a_{ji}} \right\}.$$

The advantage of such a production technology is that the fraction of primary and intermediate good inputs in production does not depend on prices, hence it is always constant. We therefore have

$$\frac{L_i^{\beta_i} K_i^{1-\beta_i}}{Y_i} = a_{0i}, \quad \frac{X_{ii}}{Y_i} = a_{ii} \quad \text{and} \quad \frac{X_{ji}}{Y_i} = a_{ji}.$$

Consequently, given gross factor prices w and r , firms only choose the optimal combination of primary input factors via minimizing costs, i.e. the firms' problem is given by

$$\min_{L_i, K_i} wL_i + rK_i \quad \text{s.t.} \quad Y_i = \frac{L_i^{\beta_i} K_i^{1-\beta_i}}{a_{0i}}.$$

Taking first-order conditions of the resulting Lagrangian, we obtain the optimal primary factor input relation

$$\frac{K_i}{L_i} = \frac{1 - \beta_i}{\beta_i} \frac{w}{r}.$$

Substituting this relation into the technology constraint $Y_i = \frac{L_i^{\beta_i} K_i^{1-\beta_i}}{a_{0i}}$, we obtain the optimal primary factor input shares in production

$$k_i = \frac{K_i}{Y_i} = a_{0i} \left[\frac{1 - \beta_i}{\beta_i} \frac{w}{r} \right]^{\beta_i} \quad \text{and} \quad l_i = \frac{L_i}{Y_i} = a_{0i} \left[\frac{\beta_i}{1 - \beta_i} \frac{r}{w} \right]^{1-\beta_i}. \quad (3.9)$$

Dividing the zero-profit condition of the firms

$$q_i Y_i = q_i X_{ii} + q_j X_{ji} + wL_i + rK_i$$

by output Y_i and rearranging it therefore yields

$$(1 - a_{ii})q_i - a_{ji}q_j = wl_i + rk_i, \quad i = 1, 2.$$

This is a linear equation system which we can write in matrix notation as

$$\begin{bmatrix} 1 - a_{11} & -a_{21} \\ -a_{12} & 1 - a_{22} \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} wl_1 + rk_1 \\ wl_2 + rk_2 \end{bmatrix} \quad (3.10)$$

Given producer prices and tax rates, one can compute consumer prices $p_i = q_i(1 + \tau_i)$, net factor prices $w^n = w(1 - \tau_w)$ and $r^n = r(1 - \tau_r)$, net income \bar{Y}^n , and consumer demands X_i, F from equation (3.8). The goods market equilibrium conditions can also be written in matrix notation as

$$\begin{bmatrix} 1 - a_{11} & -a_{12} \\ -a_{21} & 1 - a_{22} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} X_1 + G \\ X_2 \end{bmatrix}. \quad (3.11)$$

We use this equation system to derive output levels Y_1 and Y_2 . Note that the coefficient matrix on the left-hand side now differs from the one in the zero-profit condition above. Given output quantities, factor demands are computed from (3.9). Finally, the factor market equilibrium conditions and the government budget constraint can be checked for equality.

Summing up, the solution to the model with intermediate goods can be computed in the following steps:

1. Given factor prices w, r as well as tax rates τ_1, τ_2, τ_w , and τ_r , calculate k_i and l_i from (3.9).
2. Determine produces prices q_i via (3.10).
3. Calculate consumer prices p_i and net factor prices w^n and r^n in order to derive demands X_1, X_2 , and ℓ from (3.8).
4. Having calculated demands, determine output levels via (3.11).
5. Again use (3.9) to compute K_i and L_i .
6. Check all remaining markets and the government budget for clearance, i.e.

$$\begin{aligned} L_1 + L_2 + \ell - \bar{T} &= 0 \\ K_1 + K_2 - \bar{K} &= 0 \\ q_1 G - \sum_{i=1}^2 \tau_i q_i X_i - \tau_w w(L_1 + L_2) - \tau_r r \bar{K} &= 0. \end{aligned}$$

If this is not the case, adjust w, r , and the endogenous tax rate (e.g. via using `fzero`) in order to find market-clearing prices and the government budget-clearing tax rate.

Module 3.6m shows the function needed for the computation of the intermediate input model for the input coefficients: $a_{01}=a_{02}=0.2$, $a_{11}=a_{22}=0$, $a_{12}=0.3$, and $a_{21}=0.2$. The first major difference between this program and the ones before is that we now define all economic variables in the module `globals`. The reason for this is that now, in contrast with the previous versions, we already compute all those variables in the function `markets`. We can therefore use the computations in the function and take the resulting variables for producing the program's output in order to avoid double computation. Note that we now also have to use the toolbox to compute prices q_i and output levels Y_i from the linear equation system.

Module 3.6m Model with intermediate goods

```

function markets(x)

  use toolbox
  [.....]
  ! copy producer prices and taxes
  w      = 1d0
  r      = x(1)
  tauc(1) = x(2)
  tauc(2) = tauc(1)

  ! 1. calculate K/Y and L/Y
  ky = a0*((1d0-beta)/beta*w/r)**beta
  ly = a0*(beta/(1d0-beta)*r/w)**(1d0-beta)

  ! 2. determine producer prices
  q = w*ly+r*ky
  call lu_solve(ID-transpose(a), q)

  ! 3. consumer prices and demands
  p = q*(1d0+tauc)
  wn = w*(1d0-tauw)
  rn = r*(1d0-taur)
  Ybarn = wn*Tbar+rn*Kbar
  Xd = alpha/p*Ybarn
  ell = (1d0-alpha(1)-alpha(2))/wn*Ybarn

  ! 4. determine output levels
  Y(1) = Xd(1)+G
  Y(2) = Xd(2)
  call lu_solve(ID-a, Y)

  ! 5. compute K and L
  K = ky*Y
  L = ly*Y

  ! 6. check markets and budget
  markets(1) = K(1)+K(2)-Kbar
  markets(2) = q(1)*G-sum(tauc*q*Xd)-tauw*w*(Tbar-ell)-taur*r*Kbar

end function

```

The function now takes only a two-dimensional array as input. As we now determine both producer prices q_1 and q_2 via a linear equation system, we cannot normalize the producer price of the first consumption good. Therefore, we chose to normalize the gross wage w to 1.² Then, only the interest rate and the government consumption tax rate is endogenous. We thereby assume that the consumption tax is uniform across different consumption goods and there is no income tax. We then follow the six steps described above in order to calculate the capital market equilibrium condition and the government's budget constraint. Note that we use `lu_solve` for calculating both producer prices and output levels. Recall that this routine gets two input arguments `A` and `b` which are the

² Without normalization any arbitrary starting point of prices and tax rates would yield the same quantities but arbitrary equilibrium prices.

Table 3.4 Initial (with $\bar{K} = 10$) and final (with $\bar{K} = 8$) structure of the economy

| | | | | | | |
|---------------------------------------|---------------------------------|----------------|---------------------------------------|---------------------------------|----------------|----------------|
| 0.00 5.46 5.47 0.00 18.00 | 13.50 2.27 21.23 23.47 | 21.23 23.47 | 0.00 5.65 5.36 0.00 17.88 | 13.41 2.64 21.69 23.24 | 21.69 23.24 | 21.69 23.24 |
| 4.73 11.03 | 10.81 7.20 | | 4.90 11.43 | 10.56 7.04 | | |
| 21.23 | 23.47 | | 21.69 | 23.24 | | |

$w = 1.00, r = 1.82, q_1 = 0.76, q_2 = 0.97$
 $\tau_1 = \tau_2 = 0.07, p_1 = 0.81, p_2 = 1.04$
 $U = 16.99$

$w = 1.00, r = 2.31, q_1 = 0.88, q_2 = 1.09$
 $\tau_1 = \tau_2 = 0.08, p_1 = 0.95, p_2 = 1.18$
 $U = 15.51$

matrix and the vector defining the linear equation system $Ax = b$. `lu_solve` then computes the solution x to the system and stores it in the vector b .

Rather than only normalizing wages, in practice the prices of the initial equilibrium are normalized to $q_i = w = r = 1$, so that the observed nominal values in the IO-Table reflect physical quantities from where one can compute the preference and technology parameters. The latter procedure is called *calibration of parameters*. The numerical example we show here does not calibrate the preference and technology parameters from an observed equilibrium allocation. Instead parameters are specified exogenously and the resulting equilibrium allocation is computed as described above. The left part of Table 3.4 reports the initial equilibrium for the parameter and endowment specification described above. We chose $G = 3$ as in the simulation in Section 3.1.4.

Deriving leisure consumption, tax revenues, and the welfare level is not a problem. As in Section 3.1.4 one could now alter the tax structure and compute the resulting new equilibrium structure of the economy. Alternatively one could also adjust factor endowments. Assume, for example, that capital inputs in production give rise to carbon emissions E and that there is a linear relationship $E_i = \kappa K_i$ for emissions in each production sector. Consequently, the government could introduce a trading scheme for carbon emissions and reduce the quantity of emissions by 20 per cent. In our model such a policy would simply reduce the endowment of capital from $\bar{K} = 10$ to $\bar{K} = 8$. The right part of Table 3.4 shows how such a policy would change the structure of the economy. Of course, due to the reduced endowment, the price of capital would increase, inducing the intended reduction in demand. As a consequence, producer prices increase in both sectors reducing the private demand for final goods. Due to higher producer prices, consumption tax rates would also have to increase from 7 to 8 per cent.

3.2.3 OPEN ECONOMIES AND INTERNATIONAL TRADE

Up until now we have only considered a closed economy model. In order to analyse consequences of international trade in goods and production factors, we have to open

the economy for imports and exports. In principle there are two options to model that. If the home country is small compared to that of other countries then policies of the home country do not influence international goods and factor prices and it is not necessary to consider the foreign country explicitly in the numerical model. For the small open economy (SMOPEC), prices of the traded goods and factors are dictated by the international market and the domestic production sector and private demands adjust to these prices. However, if the home economy is large compared to other countries, so that domestic polices do affect international prices, one has to consider the foreign countries explicitly. In the following, we focus on the most simple case with two large countries, where home country variables are denoted by the superscript A and foreign variables are denoted by the superscript B . For simplicity we assume that both countries have identical preferences and technologies. They only differ with respect to their endowments. Let's assume that the home country A (i.e. an industrialized country of the north) is endowed with qualified labour, but is short of capital (or natural resources), while the foreign country B (i.e. an oil-exporting country of the south) is endowed with natural resources and short of qualified labour. More specifically we assume

$$\bar{T}^A = 30, \quad \bar{K}^A = 10 \quad \text{and} \quad \bar{T}^B = 10, \quad \bar{K}^B = 30.$$

As before, it is assumed that governments have to provide public goods $G = 3$ financed by uniform consumption taxes. Table 3.5 then shows the autarky equilibrium for both countries without any trade. It should not be surprising that—given the specific normalization $w^A = w^B = 1$ —the equilibrium interest rate is much higher in the home country than in the foreign country. Consequently the producer price of capital-intensive Good 1 is relatively high in the home country so demand shifts towards labour-intensive Good 2. Given our endowments, technologies, and preferences, the autarky welfare level in the home country is higher than in the foreign country.

Next, we assume that Good 2 and capital are traded internationally, while Good 1 and labour cannot be traded. In equilibrium, on the one hand, there now are two domestic

Table 3.5 Autarky equilibrium for two-country model

| | | | | | | | | | |
|-------|-------|-------|------|-------|-------|------|------|------|------|
| 0.00 | 5.46 | 13.50 | 2.27 | 21.23 | 0.00 | 1.29 | 4.41 | 0.55 | 6.25 |
| 5.47 | 0.00 | 18.00 | | 23.47 | 2.41 | 0.00 | 5.88 | | 8.29 |
| 4.73 | 10.81 | | | | 1.15 | 4.20 | | | |
| 11.03 | 7.20 | | | | 11.43 | 7.04 | | | |
| 21.23 | 23.47 | | | | 6.25 | 8.29 | | | |

$$\begin{aligned} w &= 1.00, r = 1.82, q_1 = 0.76, q_2 = 0.97 \\ \tau_1 &= \tau_2 = 0.07, p_1 = 0.81, p_2 = 1.04 \\ U &= 16.99 \end{aligned}$$

$$\begin{aligned} w &= 1.00, r = 0.18, q_1 = 0.18, q_2 = 0.35 \\ \tau_1 &= \tau_2 = 0.05, p_1 = 0.19, p_2 = 0.37 \\ U &= 12.70 \end{aligned}$$

goods markets for Good 1 and two domestic labour markets in both countries. On the other hand, there are two international markets for Good 2 and capital

$$\begin{aligned} X_{21}^A + X_{22}^A + X_2^A + X_{21}^B + X_{22}^B + X_2^B &= Y_2^A + Y_2^B \\ K_1^A + K_2^A + K_1^B + K_2^B &= \bar{K}^A + \bar{K}^B \end{aligned}$$

With respect to goods and factor prices we have to distinguish three factor prices w^A , w^B , and r which determine the producer prices q_1^A , q_1^B , and q_2 . We normalize $w^A = 1$ and specify w^B and r in order to solve for the producer prices q_1^A and q_2 of the home country via (3.11). For the foreign country, we then have

$$(1 - a_{11})q_1^B - a_{21}q_2 = w^B l_1^B + rk_1^B$$

from the zero-profit condition. As q_2 is the same for both countries it is easy to solve for q_1^B . In order to derive producer prices, consumer prices, and final demand we follow the approach of the closed economy. However, when it comes to computing output quantities Y_i^A , Y_i^B one has to supplement the three goods markets by one factor market which is the labour market of the home country in our example. Therefore, we have

$$\begin{bmatrix} 1 - a_{11} & -a_{12} & 0 & 0 \\ -a_{21} & 1 - a_{22} & -a_{21} & 1 - a_{22} \\ 0 & 0 & 1 - a_{11} & -a_{12} \\ l_1^A & l_2^A & 0 & 0 \end{bmatrix} \begin{bmatrix} Y_1^A \\ Y_2^A \\ Y_1^B \\ Y_2^B \end{bmatrix} = \begin{bmatrix} X_1^A + G \\ X_2^A + X_2^B \\ X_1^B + G \\ \bar{T}^A - \ell^A \end{bmatrix}$$

We then again proceed as we did in the closed economy case. Given output quantities we can compute factor demands and check the equilibrium on factor markets and the two public budgets. Module 3.7m shows parts of the function we use for computing the equilibrium allocation. Table 3.6 shows the equilibrium for both countries when they are opened up for trade.

As one would expect, home country A imports capital from the foreign country and exports Good 2. In equilibrium the trade balance has to be in equilibrium for every country, which implies

$$q_2 \left(Y_2^h - X_{21}^h - X_{22}^h - X_2^h \right) = r \left(\bar{K}^h - K_1^h - K_2^h \right), \quad h = A, B. \quad (3.12)$$

As one would expect, both countries benefit from trade, i.e. the respective utilities are higher than in the autarky equilibrium. Note that although we endogenously determine the wage in country B , we get the same normalized wage as in country A . This reflects the so-called ‘factor price equalization theorem’ which states that trade in goods

Module 3.7m Model with international trade

```

function markets(x)
[.....]
! 1. calculate K/Y and L/Y
[.....]

! 2. determine producer prices
q(:,1) = w(1)*ly(:,1)+r*ky(:,1)
call lu_solve(ID-transpose(a), q(:,1))
q(1,2) = (a(2,1)*q(2,1)+w(2)*ly(1,2)+r*ky(1,2))/(1d0-a(1,1))
q(2,2) = q(2,1)

! 3. consumer prices and demands
[.....]

! 4. determine output levels
vec(1) = Xd(1,1)+G
vec(2) = Xd(2,1)+Xd(2,2)
vec(3) = Xd(1,2)+G
vec(4) = Tbar(1)-ell(1)

mat(1, :) = (/1d0-a(1,1), -a(1,2), 0d0, 0d0/)
mat(2, :) = (/-a(2,1), 1d0-a(2,2), -a(2,1), 1d0-a(2,2)/)
mat(3, :) = (/0d0, 0d0, 1d0-a(1,1), -a(1,2)/)
mat(4, :) = (/ly(1,1), ly(2,1), 0d0, 0d0/)

call lu_solve(mat, vec)
Y(1,1) = vec(1)
Y(2,1) = vec(2)
Y(1,2) = vec(3)
Y(2,2) = vec(4)

! 5. compute K and L
[.....]

! 6. check markets and budget
markets(1) = L(1,2)+L(2,2)-(Tbar(2)-ell(2))
markets(2) = sum(K)-sum(Kbar)
markets(3) = q(1,1)*G-sum(tauc(:,1)*q(:,1)*Xd(:,1))-&
tauw(1)*w(1)*(Tbar(1)-ell(1))-taur(1)*r*Kbar(1)
markets(4) = q(1,2)*G-sum(tauc(:,2)*q(:,2)*Xd(:,2))-&
tauw(2)*w(2)*(Tbar(2)-ell(2))-taur(2)*r*Kbar(2)

end function

```

Table 3.6 Trade equilibrium for a two-country model

| | | | | | | | | | | |
|-------|-------|-------|------|-------|-------|------|------|-------|--------|------|
| 0.00 | 6.10 | 10.26 | 1.10 | 0.00 | 17.46 | 0.00 | 7.71 | 1.10 | 0.00 | 8.82 |
| 5.54 | 0.00 | 13.67 | | 13.05 | 32.26 | 2.80 | 0.00 | 10.28 | -13.05 | 0.03 |
| 3.58 | 15.70 | | | | | 1.81 | 0.01 | | | |
| 8.34 | 10.46 | | | | | 4.21 | 0.01 | | | |
| 17.46 | 32.26 | | | | | 8.82 | 0.03 | | | |

$$\begin{aligned}
w^A &= 1.00, r = 0.58, q_1^A = 0.37, q_2 = 0.39 \\
\tau_1^A &= \tau_2^A = 0.05, p_1^A = 0.38, p_2^A = 0.41 \\
U^A &= 23.02
\end{aligned}$$

$$\begin{aligned}
w^B &= 1.00, q_1^B = 0.37 \\
\tau_1^B &= \tau_2^B = 0.06, p_1^B = 0.39, p_2^B = 0.41 \\
U^B &= 17.38
\end{aligned}$$

can substitute for trade in production factors. Finally, since we apply the destination principle of goods taxation, the producer prices of the traded Good 2 are the same in both countries and consumer prices may differ. As a policy reform we could analyse a switch to the origin principle, where output is taxed and no border adjustment is allowed so that consumer prices of traded goods are identical and producer prices differ.

3.3 Further reading

The pioneering computable general equilibrium model (CGE) was the Norwegian multi-sectoral growth model developed by Johansen (1960). Since then, many CGE models have been built to analyse, among others, taxation, international trade as well as environmental and labour market issues. Shoven and Whalley (1984, 1992) provide a survey of this literature with mostly static models. Fehr, Rosenberg, and Wiegard (1995) develop a multi-country CGE model in order to quantify the effects of indirect tax harmonization within the European Union. Nowadays many CGE models are solved with the General Algebraic Modeling System (GAMS), which is a high-level modelling system for mathematical programming problems, see Rosenthal (2012) or Rutherford (1999). Hosoe, Gasawa, and Hashimoto (2010), and Cardenete, Guerra, and Sancho (2012) offer a detailed introduction to CGE modeling with GAMS. Burfischer (2011) introduces the Global Trade Analysis Project (GTAP), a CGE model that is widely used to analyse global events today.

3.4 Exercises

- 3.1. Show that the Cobb-Douglas function is a special case of the constant elasticity of substitution (CES) function, meaning that

$$\lim_{\epsilon \rightarrow 1} [\alpha X^{1-1/\epsilon} + (1-\alpha)Y^{1-1/\epsilon}]^{\frac{1}{1-1/\epsilon}} = X^\alpha Y^{1-\alpha}.$$

ϵ is the elasticity of substitution between X and Y and α a share parameter.

- 3.2. Now assume that the utility function in the basic economy model with a fixed labour supply is

$$U(X_1, X_2) = \left[\alpha^{\frac{1}{v}} X_1^\mu + (1-\alpha)^{\frac{1}{v}} X_2^\mu \right]^{\frac{1}{\mu}} \quad \text{with } \mu = 1 - \frac{1}{v}.$$

- a) Show that the demand functions are

$$X_1 = \frac{\alpha \bar{Y}}{p_1^\nu P} \quad \text{and} \quad X_2 = \frac{(1 - \alpha) \bar{Y}}{p_2^\nu P}.$$

with $P = \alpha p_1^{1-\nu} + (1 - \alpha)p_2^{1-\nu}$ as price index for the composite good.

- b) How do the market equilibrium conditions (3.4) to (3.6) change? Set $\nu = 0.5$ and implement these changes in Program 3.2. Now increase the elasticity of substitution to $\nu = 1.5$ and explain the changes in the initial equilibrium in economic terms.

3.3. Next assume a two-stage utility function

$$U(C, \ell) = \left[(1 - \alpha)^{\frac{1}{\nu}} C^\mu + \alpha^{\frac{1}{\nu}} \ell^\mu \right]^{\frac{1}{\mu}} \quad \text{with } \mu = 1 - \frac{1}{\nu}$$

and

$$C(X_1, X_2) = \left[\sum_{i=1}^2 \alpha_i^{\frac{1}{\nu_x}} X_i^{\mu_x} \right]^{\frac{1}{\mu_x}} \quad \text{with } \mu_x = 1 - \frac{1}{\nu_x}$$

on the household side in the basic economy model.

- a) Show that the demand functions are

$$C = \frac{(1 - \alpha) \bar{Y}}{P^\nu \Omega}, \quad \ell = \frac{\alpha \bar{Y}}{w^\nu \Omega} \quad \text{and} \quad X_i = \frac{\alpha_i Y_D}{p_i^{\nu_x} P}.$$

$\Omega = (1 - \alpha)P^{1-\nu} + \alpha w^{1-\nu}$ and $P = \sum_{i=1}^2 \alpha_i p_i^{1-\nu_x}$ define the respective price indices while $\bar{Y} = w\bar{T} + r\bar{K} = PC + w\ell$ and $Y_D = \bar{Y} - w\ell = wL + r\bar{K}$ are total and disposable income, respectively. Consequently, we again have $\bar{T} = L + \ell$ as total time endowment.

- b) How do the market equilibrium conditions (3.4) to (3.6) change? Set $\nu = \nu_x = 0.5$ and implement these changes in Program 3.3. Now increase the elasticity of substitution successively to $\nu = 0.5$, $\nu_x = 1.5$, and $\nu = \nu_x = 1.5$ and explain the changes in the initial equilibrium in economic terms.

3.4. Next consider the two-stage CES utility function with variable labour supply from the previous exercise, but in addition assume the following CES production functions

$$Y_i = \left[\beta_i L_i^{\rho_i} + (1 - \beta_i) K_i^{\rho_i} \right]^{\frac{1}{\rho_i}} \quad \text{with } \rho_i = 1 - \frac{1}{\sigma_i}$$

as the elasticity of substitution between labour and capital in production of good i .

- a) Show that the per-unit labour and capital demand functions $l_i(w, r)$ and $k_i(w, r)$ are

$$l_i(w, r) = \left\{ \beta_i + (1 - \beta_i) \left[\frac{\beta_i r}{(1 - \beta_i)w} \right]^{1-\sigma_i} \right\}^{-1/\rho_i}$$

$$k_i(w, r) = \left\{ (1 - \beta_i) + \beta_i \left[\frac{(1 - \beta_i)w}{\beta_i r} \right]^{1-\sigma_i} \right\}^{-1/\rho_i}$$

- b) Set $\nu = \nu_x = \sigma_i = 0.5$ and implement these changes numerically.

Hint: In order to solve this model numerically, it is useful to proceed as in the model with intermediate goods. Therefore, normalize $w = 1$ and search for the equilibrium interest rate r that balances the capital market. In order to do this, first compute per-unit factor demands, then the producer (and in this case consumer) prices, then the market demand (which in this case is output). Given output and per-unit factor demands, it is possible to derive capital demand in both sectors.

- c) Now increase the elasticity of substitution between capital and labour in the production function from $\sigma_i = 0.5$ to $\sigma_i = 1.5$ and explain the observed changes in the initial equilibrium in economic terms.

- 3.5. Introduce a government sector with consumption, labour and capital-income taxes into the model of the previous exercise.

Run the same simulations as in Table 3.1. Why can you not get the same equivalence results?

- 3.6. Use the model with intermediate goods and government activity (Program 3.6). Assume that the household optimizes a two-stage CES utility function of the form

$$\max_{C, \ell} U(C, \ell) = \left[(1 - \alpha)^{\frac{1}{\nu}} C^\mu + \alpha^{\frac{1}{\nu}} \ell^\mu \right]^{\frac{1}{\mu}} \quad \text{s.t.} \quad PC + w^n \ell = \bar{Y}^n$$

as well as

$$\max_{X_1, X_2} C(X_1, X_2) = \left[\alpha_x^{\frac{1}{\nu_x}} X_1^{\mu_x} + \alpha_x^{\frac{1}{\nu_x}} X_2^{\mu_x} \right]^{\frac{1}{\mu_x}} \quad \text{s.t.} \quad p_1 X_1 + p_2 X_2 = Y_D.$$

As before, ν and ν_x define the elasticities of substitution between the consumption aggregate C and leisure ℓ and between different consumption goods, respectively. In addition, the production function now should be

$$a_{0i} Y_i = \left[\beta_i L_i^{1-\frac{1}{\sigma_i}} + (1 - \beta_i) K_i^{1-\frac{1}{\sigma_i}} \right]^{\frac{1}{1-\frac{1}{\sigma_i}}},$$

where σ_i defines the elasticity of substitution between capital and labour in production.

Use the derived functions from the previous exercises in the revised Fortran code of Program 3.6 and solve the model for $v = v_x = \sigma_i = 0.5$. All other parameters are the same as in the previous model. Now successively increase v , v_x , and σ_i from 0.5 to 1.5 and explain the changes in economic terms.

- 3.7. Given Cobb-Douglas preferences and technologies of the intermediate goods model of Section 3.2.2, we now assume a small open economy (smopec) with homogenous goods. Consequently, the world interest rate as well as the price of the traded Good 2 is derived from the world market and normalized to unity, i.e. $\bar{r} = \bar{q}_2 = 1$. The small open economy has no influence on world prices. Your computer code for a smopec should be organized as follows: Starting point are guesses for the wage rate w and the endogenous tax rate τ_i . Next, as in the closed economy model, compute the per-unit factor demands k_i, l_i and derive the producer price q_1 from the respective zero-profit condition. Given the consumer prices p_i and r^n, w^n it is possible to compute household demand for goods and leisure. In order to determine the output levels Y_i , use the goods and labour market equilibrium conditions

$$(1 - a_{11})Y_1 - a_{12}Y_2 = X_1 + G \quad \text{and} \quad l_1 Y_1 + l_2 Y_2 = \bar{T} - \ell.$$

Given Y_i derive K_i and L_i and check the government budget and the zero-profit condition of firm 2. Adjust the endogenous tax rate τ_i and the wage rate w using subroutine fzero until the equilibrium is reached.

- a) Set $G = 15$ and keep all remaining parameters from the closed economy model. Simulate the model with $\tau_1 = \tau_2$. Explain the resulting trade balance and print out the IO-matrix.
- b) Now assume that $\tau_2 = 5 \times \tau_1$ and solve the respective equilibrium. What happens to the trade balance? Explain in economic terms!
- 3.8. Again consider the closed economy case, but assume sector-specific factor input taxes τ_i^k and τ_i^l for companies. As a result, the firm's optimization problem changes to

$$\min_{L_i, K_i} w(1 + \tau_i^l)L_i + r(1 + \tau_i^k)K_i \quad \text{s.t.} \quad Y_i = \frac{L_i^{\beta_i} K_i^{1-\beta_i}}{a_{0i}}.$$

Starting from an initial equilibrium allocation as in the left part of Table 4.4, analyse an isolated introduction of a

- a) corporate tax in sector 1, i.e. $\tau_1^k = 0.2, \tau_2^k = \tau_1^l = 0$;
 b) a labour input tax in sector 2, i.e. $\tau_2^l = 0.2, \tau_1^l = \tau_1^k = 0$.

In both cases the consumption tax rate is endogenous. Discuss the resulting price and quantity changes at the sectoral level. Explain the welfare changes for the household.

- 3.9. In this chapter, we assumed for simplicity that the consumption tax is levied as a retail sales tax, i.e. it does not affect production decisions. However, in practice the value-added tax (VAT) in Europe is computed at the firm level using the so-called credit or invoice method, where firm i calculates the actual tax liability by first computing the gross tax liability and then deducting all input VAT on purchases of intermediate goods, i.e.

$$T_i = \tau_i q_i Y_i - \sum_{j=1}^2 \tau_j q_j X_{ji}.$$

Consequently, the zero-profit condition of firms becomes

$$p_i Y_i = p_i X_{ii} + p_j X_{ji} + wL_i + rK_i + T_i,$$

so that the linear equation system (3.10) does not change if we explicitly consider a VAT system. In practice some sectors are exempted from VAT, i.e. $\tau_i = 0$. In some countries these sectors are allowed to reclaim their VAT payments on purchases on intermediate goods (so-called zero rating). In other countries this is not allowed (so-called exemption). Gottfried and Wiegard (1991) analyse the two systems in detail using a static CGE model. In this exercise we simply move from zero rating with $\tau_1 = 0.2, \tau_2 = 0$ (i.e. $T_1 > 0, T_2 < 0$) towards exemption (i.e. $T_1 > 0, T_2 = 0$) when τ_1 is endogenously adjusted to balance the budget.

- a) How does exemption change the above zero-profit condition of firm 2? How does the linear equation system (3.10) change?
 - b) Adjust Program 3.6 to compute the resulting new equilibrium and compare the macroeconomic and welfare changes. Can you explain the differences in economic terms?
- 3.10. Now consider the two-country model where both countries apply a uniform consumption tax. In the above example we implicitly assumed that both countries apply the destination principle for the taxation of traded goods. Consequently, goods are traded at producer prices and consumer prices of traded goods could be different in both countries. Now assume that both countries switch towards the origin principle, where taxes are levied at primary factor inputs so that goods are traded at producer prices.
- a) How does this reform change the price and tax revenue formulas above?
 - b) Adjust the Fortran code of Program 3.7 and compute the resulting new equilibrium when the consumption tax is endogenously adjusted. Explain the resulting changes in economic variables and welfare.

4 Topics in finance and risk management

This chapter introduces basic concepts of modern finance theory and demonstrates how to apply them in complex real-world problems. Financial deals and investment decisions are typically determined under uncertainty. Therefore, although this chapter is self-contained, we have to expect some theoretical background in individual decision-making and optimal investment under uncertainty. We organize our discussion into four central sections. The starting point is a portfolio choice problem, where an investor has to choose between different assets with specific risk and return characteristics. We then move on to some option pricing applications. We first derive analytical formulas and then evaluate numerical procedures for pricing European and American options as well as more exotic option products. The third section elaborates on credit risk measurement and management using a corporate bond portfolio as example. In the last section we discuss mortality risk and the optimal portfolio structure of a life insurance company.

4.1 Mean-variance portfolio theory

This section provides different numerical approaches to find an optimal portfolio structure with many risky assets. It begins with simple measures of risk and return of a single asset and then develops decision rules to choose optimal portfolios that maximize expected utility of wealth in worlds without and with riskless borrowing and lending opportunities.

4.1.1 PORTFOLIO CHOICE WITH RISKY ASSETS

The purpose of this section is to optimize a portfolio of equity shares and a risk-free investment opportunity. The investor faces the most basic two-period investment choice problem: He buys assets in the first period and these assets pay off in the next period. The problem of the investor is to choose from $i = 1, \dots, N$ risky assets which may be shares, bonds, real estate, etc. The gross return of each asset i is denoted by $r_{it} = q_{it}/q_{it-1} - 1$,

where q_{it-1} is the first-period market price and $q_{it} - q_{it-1}$ the second-period payoff. Since the payoff is uncertain, the return of asset i is characterized by an expected value $\mu_i = E(r_i)$, a variance $\sigma_{ii} = \text{Var}(r_i)$ (instead of σ_i^2 as usual) and covariances $\sigma_{ij} = \text{Cov}(r_i, r_j)$ with the returns of asset $j \neq i$. A specific portfolio consists of proportions ω_i of wealth that are invested in the risky asset i in the first period, so that the expected return and the variance of the portfolio are

$$E(r_p) = \mu_p = \sum_{i=1}^N \omega_i \mu_i = \omega^T \mu \quad (4.1)$$

$$\text{Var}(r_p) = \sigma_p^2 = \sum_{i=1}^N \sum_{j=1}^N \omega_i \omega_j \sigma_{ij} = \omega^T \Sigma \omega, \quad (4.2)$$

where variables without indices denote vectors and T the transposed of a vector or matrix. Hence we have

$$\omega = \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_N \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_N \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} \sigma_{11} & \dots & \sigma_{1N} \\ \vdots & \ddots & \vdots \\ \sigma_{N1} & \dots & \sigma_{NN} \end{pmatrix},$$

where Σ denotes the variance-covariance matrix of returns. For our numerical example we look at annual price data for three stocks A, B, and C. Table 4.1 reports price data for six years on which the following calculations are based. Program 4.1 shows how we compute means and variances in the subroutine `returns`. We first store the price data of Table 4.1 in an array `odat(0:TT, NN)`, where `TT` is the number of years and `NN` the number of stocks. We then derive returns r_{it} for each stock $i = 1, \dots, N$ in periods $t = 1, \dots, T$. After that, we compute the expected return and the covariance matrix from

$$\mu_i = \frac{1}{T} \sum_{t=1}^T r_{it} \quad \text{and} \quad \sigma_{ij} = \frac{1}{T} \sum_{t=1}^T (r_{it} - \mu_i)(r_{jt} - \mu_j).$$

Table 4.1 Yearly stock price data

| Year | Stock A | Stock B | Stock C |
|------|---------|---------|---------|
| 0 | 1.00 | 2.00 | 3.00 |
| 1 | 1.02 | 2.65 | 2.80 |
| 2 | 1.17 | 2.40 | 4.50 |
| 3 | 1.08 | 2.70 | 4.20 |
| 4 | 1.16 | 2.85 | 3.20 |
| 5 | 1.26 | 2.75 | 4.20 |

Program 4.1 Computing means and variances

```

subroutine returns
  [.....]
  ! compute return per period
  do ii = 1, NN
    do ij = 1, TT
      r(ij, ii) = odat(ij, ii)/odat(ij-1, ii)-1d0
    enddo
  enddo

  ! compute expected value
  do ii = 1, NN
    mu(ii) = sum(r(:, ii))/dble(TT)
  enddo

  ! compute covariance matrix
  do ii = 1, NN
    do ij = 1, NN
      sig(ii,ij) = dot_product(r(:,ii)-mu(ii), r(:,ij)-mu(ij))
    enddo
  enddo
  sig = sig/dble(TT)

end subroutine

```

In the current example we get

$$\mu = \begin{pmatrix} 0.050 \\ 0.075 \\ 0.110 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.006 & -0.007 & 0.015 \\ -0.007 & 0.021 & -0.031 \\ 0.015 & -0.031 & 0.094 \end{pmatrix}.$$

Efficient portfolios An *efficient portfolio* is the portfolio of risky assets that has the lowest return variance within all portfolios that yield the same expected return μ_p . Mathematically, the efficient portfolio is defined by the portfolio weights $\omega = (\omega_1, \dots, \omega_N)^T$ that minimize σ_p^2 for a given μ_p , meaning

$$\min_{\omega} \frac{1}{2} \omega^T \Sigma \omega \quad \text{s.t.} \quad I^T \omega = 1 \quad \text{and} \quad \mu^T \omega = \mu_p$$

where $I = (1, \dots, 1)^T$ denotes the unit vector. The first constraint thereby guarantees that the weights ω sum up to a value of one. The second constraint fixes the expected return of the portfolio at μ_p . The first-order condition of this problem yields

$$\omega = \Sigma^{-1}(\lambda_1 I + \lambda_2 \mu), \tag{4.3}$$

where λ_1 and λ_2 denote the Lagrangian multipliers of the two constraints. In order to determine λ_1 and λ_2 we substitute the optimality condition (4.3) into the two constraints and obtain

$$1 = I^T \omega = I^T \Sigma^{-1}(\lambda_1 I + \lambda_2 \mu) = \lambda_1 I^T \Sigma^{-1} I + \lambda_2 I^T \Sigma^{-1} \mu =: \lambda_1 a + \lambda_2 b$$

$$\mu_p = \mu^T \omega = \mu^T \Sigma^{-1}(\lambda_1 I + \lambda_2 \mu) = \lambda_1 \mu^T \Sigma^{-1} I + \lambda_2 \mu^T \Sigma^{-1} \mu =: \lambda_1 b + \lambda_2 c$$

The two equations can be solved for the two unknowns yielding

$$\lambda_1 = \frac{c - b\mu_p}{ac - b^2} \quad \text{and} \quad \lambda_2 = \frac{a\mu_p - b}{ac - b^2}.$$

Note that λ_1 and λ_2 depend on μ_p , which is the targeted expected return of the minimization problem. Consequently, for any given value of μ_p the minimum portfolio variance equals

$$\sigma_p^2 = \omega^T \Sigma \omega = \omega^T \Sigma \Sigma^{-1}(\lambda_1 I + \lambda_2 \mu) = \lambda_1 + \lambda_2 \mu_p = \frac{a\mu_p^2 - 2b\mu_p + c}{ac - b^2} \quad (4.4)$$

with parameters

$$a = I^T \Sigma^{-1} I, \quad b = I^T \Sigma^{-1} \mu \quad \text{and} \quad c = \mu^T \Sigma^{-1} \mu.$$

The set of minimum variance portfolios can be represented by a parabolic curve in the σ_p^2 - μ_p -plane. Finally, it is straightforward to find the global minimum variance portfolio by simply setting

$$\frac{\partial \sigma_p^2}{\partial \mu_p} = \frac{2a\mu_p - 2b}{ac - b^2} = 0,$$

so that $(\mu_p, \sigma_p^2)^{mv} = (b/a, 1/a)$. For this minimum variance portfolio we have $\lambda_1 = 1/a$ and $\lambda_2 = 0$. Substituting these values of the Lagrangian multipliers back into the optimality condition (4.3), we can finally get the weight vector for the global minimum variance portfolio as

$$\omega^{mv} = \frac{\Sigma^{-1} I}{a} = \frac{\Sigma^{-1} I}{I^T \Sigma^{-1} I}. \quad (4.5)$$

Program 4.1.a shows how to implement equation (4.4) and how to compute the weight factors for the global minimum variance portfolio. We therefore first have to invert the variance-covariance matrix Σ , which can be done using the function `lu_invert`, see Chapter 2. We then can calculate the parameters and the minimum variance portfolio shares `omega_mv`. Finally, we can create plotting data to picture the set of efficient portfolios from (4.4).

Figure 4.1 shows the resulting plot of the mean returns and variances of the efficient portfolios in our numerical example. The minimum variance portfolio

Program 4.1.a Computation of minimum variance portfolios

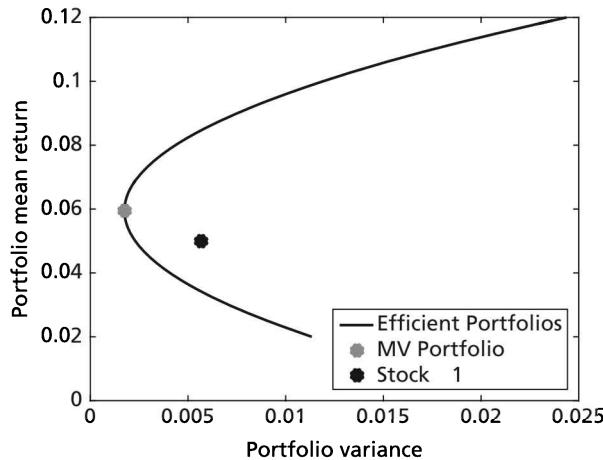
```

! invert variance-covariance matrix
siginv = lu_invert(sig)

! derive parameters
a = dot_product(matmul(ID, siginv), ID)
b = dot_product(matmul(ID, siginv), mu)
c = dot_product(matmul(mu, siginv), mu)

! calculate minimum variance portfolio
omega_mv = matmul(siginv, ID)/a
[.....]
! create plot data for efficient portfolios
do ii = 0,100
    x1(ii) = 1d0/1000d0*dble(ii+20)
enddo
y1 = (a*x1**2 - 2*b*x1+c)/(a*c - b**2)

```

**Figure 4.1** Portfolio mean and variances

$(\mu_p, \sigma_p^2)^{mv} = (0.059, 0.002)$ can be directly observed in the figure. Note that it is not optimal to put all wealth into stock A, which is the stock with the lowest risk–return combination. In this case the expected return of the portfolio would be 0.050 while the variance would be 0.006. Hence, the portfolio lies inside the parabola, meaning that there would be a portfolio that offers the same variance, but would yield a higher return on average. This is possible because the returns of stocks A and B as well as stocks B and C are negatively correlated which allows for a more sophisticated diversification.

4.1.2 INTRODUCING RISK-FREE ASSETS

Next let us assume that there is also a risk-free asset that pays a fixed return r_f . This return is already known to the investor when he has to make the investment decision.

Consequently, the investor may choose to invest a fraction ω_f of his wealth into the riskless asset. The optimal asset allocation consisting of risky assets and a riskless security depends on the investor's preferences, which can be defined by the utility function

$$U(\mu_p, \sigma_p^2) = \mu_p - \frac{\gamma}{2} \cdot \sigma_p^2,$$

where γ is the risk-aversion of the investor. Different investors have different risk–return preferences. Investors with a higher level of risk-aversion choose portfolios with a low level of expected return and variance and vice versa. Given that there exists a risk-free security, the classical *separation theorem* applies, which states that investors should only diversify between the risk-free asset and a single optimal portfolio of risky assets called the *tangent portfolio*.

Figure 4.2 illustrates this theorem in the mean-variance diagram. As before, investors may choose from different combinations of the risky stocks A, B, and C, but in addition there also exists a risk-free asset (say cash). The latter has an even lower return r_f than the minimum variance portfolio, but is also riskless so that it lies on the vertical zero-risk axis. According to the separation theorem the portfolio choice problem is split into two parts. In the first, the *tangent portfolio* of stocks is determined independent of the investor's risk preferences. In the second step, the optimal mix of this tangent portfolio and the risk-free asset is calculated by optimizing the investor's risk preferences along the *efficiency frontier*. An extremely risk-averse investor allocates all his wealth in the riskless asset and receives the return r_f . If the investor is less risk-averse, he combines his cash holdings with investments in the tangent portfolio like the conservative investor on the efficiency frontier. At the tangent portfolio the investor only holds stocks, so that the outer efficiency frontier touches the inner parabola, which shows efficient portfolios without the riskless asset. Aggressive investors may even borrow to leverage their holdings of the

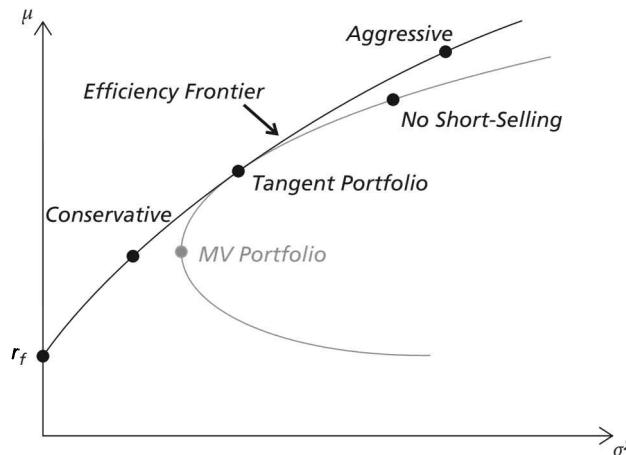


Figure 4.2 Separation theorem

tangent portfolio, reaching a point on the outer efficiency frontier that is even riskier than the tangent portfolio. Only if the investor is not allowed to borrow against a risk portfolio position, he might no longer hold the tangent portfolio but invest in a more risky portfolio and consequently move along the inner parabola.

In order to compute the structure of the tangent portfolio we have to solve the maximization problem

$$\max_{\omega_f, \omega} U(\mu_p, \sigma_p^2) = \omega_f r_f + \omega^T \mu - \frac{\gamma}{2} \omega^T \Sigma \omega \quad \text{s.t. } \omega_f + \omega^T I = 1.$$

After substituting the budget constraint $\omega_f = 1 - \omega^T I$, we can reformulate the problem as

$$\max_{\omega} \omega^T (\mu - r_f I) - \frac{\gamma}{2} \omega^T \Sigma \omega \quad (4.6)$$

which yields the optimality conditions

$$\mu_i - r_f - \gamma \sum_{j=1}^N \omega_j \sigma_{ji} = 0 \quad i = 1, \dots, N$$

so that the solution is

$$\omega = \frac{1}{\gamma} \cdot \Sigma^{-1} (\mu - r_f I). \quad (4.7)$$

The optimal portfolio share of the risky asset equals the expected excess return (or risk premium) multiplied by the inverse of the variance-covariance matrix of returns. Note that the investor's preferences enters the solution (4.6) only through the scalar term $1/\gamma$. Consequently different types of investors only differ in the overall share of their risky portfolio, not in its composition. Conservative investors with a high γ hold more of the riskless asset and less of all risky assets. However, they do not change the relative proportions of their risky assets, which are determined by the vector $\Sigma^{-1}(\mu - r_f I)$. Program 4.1.b shows how to derive the tangent portfolio and the optimal choice of the investor from (4.7). Note that after having calculated the optimal choice of the investor, we normalize the portfolio shares of the tangent portfolio as the values of `omega_tan` will typically not sum up to 1. Table 4.2 summarizes the minimum variance and tangent portfolios as well, including their expected return and riskiness as well as the investor's optimal choice for different degrees of risk-aversion. While the minimum variance portfolio contains a lot of stocks of type A, which have a low variance but also a low mean return, the tangent portfolio turns out to be much more risky. Different degrees of risk-aversion define how much the investor wants to hold of the riskless asset and how much he desires to hold of the tangent portfolio. The lower the degree of risk-aversion,

Program 4.1.b Computing the tangent portfolio without constraints

```

! compute tangent portfolio
omega_tan = matmul(siginv, mu - r_f)

! get investors choice
omega = omega_tan/gamma
omega_f = 1d0 - sum(omega)

! normalize tangent portfolio shares to 1
omega_tan = omega_tan/sum(omega_tan)

```

Table 4.2 MV, Tangent, and Optimal Portfolios ($r_f = 0.05$)

| Portfolio | Riskless | Stock A | Stock B | Stock C | μ_p | σ_p^2 |
|--------------------|----------|---------|---------|---------|---------|--------------|
| MV | | 65.5 | 32.5 | 2.0 | 0.0594 | 0.0017 |
| Tangent | | -7.5 | 71.4 | 36.0 | 0.0895 | 0.0073 |
| Opt. $\gamma = 10$ | 46.1 | -4.0 | 38.5 | 19.4 | 0.0713 | 0.0021 |
| Opt. $\gamma = 8$ | 32.6 | -5.0 | 48.1 | 24.3 | 0.0766 | 0.0033 |
| Opt. $\gamma = 5$ | -7.8 | -8.0 | 77.0 | 38.8 | 0.0926 | 0.0085 |

the more the investor puts into risky stocks. In fact, for a value of $\gamma = 5$, he would like to borrow in the riskless security to seek more risk and a higher return than the tangent portfolio.

4.1.3 SHORT-SELLING CONSTRAINTS

The optimal portfolios in Table 4.2 involve a lot of short-selling. In practice, however, the investor's choices might be limited by short-selling constraints. We can include such restrictions into our problem by addition non-negativity constraints $\omega_f, \omega \geq 0$. Consequently, investors can neither build up debt in cash nor in risky assets. In order to implement such constraints numerically, we reformulate the optimization problem (4.6) to

$$\max_{\omega_f, \omega \geq 0} \omega^T(\mu - r_f I) - \frac{\gamma}{2}\omega^T \Sigma \omega \quad \text{s.t.} \quad \omega \geq 0 \quad \text{and} \quad \omega_f = 1 - \omega^T I. \quad (4.8)$$

Using multipliers η_i and η_f associated to the constraints $\omega_i \geq 0$ and $\omega_f \geq 0$, the Lagrangian of this problem reads

$$\mathcal{L} = \omega^T(\mu - r_f I) - \frac{\gamma}{2}\omega^T \Sigma \omega + \eta^T \omega + \eta_f(1 - \omega^T I),$$

where we already substituted the constraint $\omega_f = 1 - \omega^T I$. The optimality conditions of problem (4.8) consequently read

$$\begin{aligned}\mu_i - r_f - \gamma \sum_{j=1}^N \omega_j \sigma_{ji} + \eta_i - \eta_f &= 0 \quad i = 1, \dots, N \\ \eta_i \omega_i &= 0 \quad i = 1, \dots, N \\ \eta_f (1 - I^T \omega) &= 0,\end{aligned}$$

where the last $N + 1$ conditions are known as *Kuhn-Tucker conditions*. These conditions make sure that out of the inequalities

$$\eta_i \geq 0 \quad \text{and} \quad \omega_i \geq 0$$

at least one will hold with equality. The same of course will then be true for η_f and $1 - I^T \omega$. We can solve this *nonlinear complementarity problem (NCP)* numerically by reformulating it as a system of equations using so-called NCP-functions. One popular choice of an NCP-function is the *Fischer-Burmeister function* defined as

$$f_{FB}(\eta_i, \omega_i) = \eta_i + \omega_i - \sqrt{\eta_i^2 + \omega_i^2} = 0.$$

This function takes a value of 0 only when either $\eta_i = 0$, or $\omega_i = 0$ or both are zero at the same time. Hence, by finding the root of this function, we can be sure that our Kuhn-Tucker conditions hold exactly.

Program 4.2 shows how the optimal investment fractions with short-selling constraints can be computed in practice. We cannot solve this problem analytically anymore. Hence, we have to make use of the subroutine `fzero` from the toolbox that allows to solve a system of nonlinear equations numerically. We implement the equation system we have

Program 4.2 Computing the tangent portfolio with constraints

```
program portfoliochoice_short
[.....]
! initial guess for portfolio shares
x_in(1:NN) = 0.2d0

! initial guess for multipliers
x_in(NN+1:2*NN+1) = 0d0

! solve first order conditions
call fzero(x_in, focs, check)

! check for convergence
if(check) stop 'Error: fzero did not converge'

! copy decision
omega = x_in(1:NN)
omega_f = 1d0 - sum(omega)
[.....]
end program
```

to solve in order to determine optimal portfolio shares in a function `focs`, which is stored in a module `globals`. This function contains the $2 \times N + 1$ equations to solve for the $2 \times N + 1$ variables ω , η , and η_f . ω_f can then simply be computed from $1 - I^T \omega$. To use the subroutine `fzero` we have to provide an initial guess for the input variable `x_in`. It is generally a good idea to set non-zero guesses for the choice variables (in our case the value of ω) and initialize the multipliers at a value of zero. We then call the subroutine `fzero`, check whether there has been an error in the solution process and copy the solution to the variable `omega` and `omega_f`.

Module 4.2m shows how we implemented the equation system in the function `focs`. We first copy the values of the portfolio shares and calculate the fraction of wealth invested in the risk-free asset. We then also copy the Lagrangian multipliers. In the first N equations, we store the actual first-order conditions with respect to the portfolio shares. Equations $N + 1$ to $2N + 1$ contain the values of the Fisher-Burmeister functions.

Table 4.3 reports how investors' risk preferences and the risk-free return influence the optimal portfolio structure. Initially we assume a risk-free return of 4 per cent. We consider two investors, where the left part shows the more aggressive one (with $\gamma = 8$). If there were no short-selling constraints, he would run up a credit of almost 39 per cent of his wealth in order to buy shares from the three companies. The expected portfolio return then amounts to 8.7 per cent and the variance of the portfolio would be 0.6. This choice reflects the aggressive investor in Figure 4.2 on the outer efficiency frontier. The tangent portfolio consists of roughly $[41.8/(41.8 + 71.4 + 25.7)] = 30.1$ per cent of risky assets invested in stock A, 51.4 per cent invested in stock B and the remaining 18.5 per cent invested in stock C. If the investor cannot run up debt, he would not hold any cash and invest all his wealth in risky assets. Since his opportunity set now includes all portfolios to the right of the tangent portfolio, it is optimal for him to deviate from the tangent portfolio in this case and invest a higher fraction in the risky asset C.

Module 4.2m Function containing nonlinear equation system

```

function focs(x_in)
    [.....]
    ! copy portfolio weights
    omega = x_in(1:NN)
    omega_f = 1d0 - sum(omega)

    ! copy Lagrangian multipliers
    eta = x_in(NN+1:2*NN)
    eta_f = x_in(2*NN+1)

    ! set up first-order conditions
    focs(1:NN) = mu - r_f - gamma*matmul(sig, omega) &
                    + eta - eta_f
    focs(NN+1:2*NN) = omega + eta - sqrt(omega**2 + eta**2)
    focs(2*NN+1) = omega_f + eta_f - sqrt(omega_f**2 + eta_f**2)

end function

```

Table 4.3 The optimal portfolio structure

| r_f (%) | $\gamma = 8$ | | | | $\gamma = 10$ | | | |
|--------------|--------------|------|------|------|---------------|------|------|------|
| | 4.0 | 4.0 | 5.0 | 5.0 | 4.0 | 4.0 | 5.0 | 5.0 |
| ω_f | -38.9 | 0.0 | 32.6 | 27.1 | -11.1 | 0.0 | 46.1 | 41.7 |
| ω_1 | 41.8 | 16.3 | -5.0 | 0.0 | 33.5 | 26.2 | -4.0 | 0.0 |
| ω_2 | 71.4 | 58.7 | 48.1 | 49.0 | 57.1 | 53.5 | 38.5 | 39.2 |
| ω_3 | 25.7 | 25.0 | 24.3 | 23.9 | 20.5 | 20.3 | 19.4 | 19.1 |
| μ_p | 8.7 | 8.0 | 7.6 | 7.6 | 7.8 | 7.6 | 7.1 | 7.1 |
| σ_p^2 | 0.6 | 0.4 | 0.3 | 0.3 | 0.4 | 0.3 | 0.2 | 0.2 |

The more risk-averse investor in the right part of Table 4.3 already builds up less debt in cash in the situation without short-selling constraints. Consequently the expected return on his optimal portfolio is lower but he also faces less risk. In Figure 4.2 the more risk-averse investor is on the efficiency frontier, still above the tangent portfolio but below the aggressive investor. Note that the relative shares of risky assets are the same as for the aggressive investor, i.e. $33.5/111.1 = 30.1$ per cent, etc. When the short-selling constraint is binding, the more risk-averse investor also moves away from the tangent portfolio, but less than the aggressive investor. Of course, if we increase risk-aversion further, we could find investors who invest in the tangent portfolio without cash holdings and others who invest in the tangent portfolio and hold cash in addition.

If the risk-free return increases up to 5.0 per cent, both investors hold cash and build up debt in stock A which has a low return. Since both investors hold cash, the situation reflects the conservative investor in Figure 4.2. Note that the (new) tangent portfolio lies to the right of the previous one, since the outer efficiency frontier has turned around the inner parabola. Under the presence of short-selling constraints, both investors are not able to build up debt in the stock that offers the lowest return-risk combination. Consequently they will lower their exposure to the other asset with a low return and risk, namely the risk-free security.

4.1.4 MONTE CARLO MINIMIZATION

Solving the investor's optimization problem with short-selling constraints using a root-finding method is a good option if there are only few stocks the investor can choose to invest in. However, when the number of investment opportunities increases, this method can slow down quickly, since we have to solve $2N + 1$ equations simultaneously. *Monte Carlo Minimization (MCM)* offers an alternative approach to deriving the investor's optimal portfolio that is based on iteratively simulating different portfolios and picking the one that maximizes the investor's utility. To implement this method, we include the risk-free investment opportunity into the vector ω . With an interest rate of 4 per cent we consequently have

$$\mu = \begin{pmatrix} 0.040 \\ 0.050 \\ 0.075 \\ 0.110 \end{pmatrix} \quad \text{and} \quad \Sigma = \begin{pmatrix} 0.000 & 0.000 & 0.000 & 0.000 \\ 0.000 & 0.006 & -0.007 & 0.015 \\ 0.000 & -0.007 & 0.021 & -0.031 \\ 0.000 & 0.015 & -0.031 & 0.094 \end{pmatrix}.$$

and the investor's utility function simply reads

$$\omega^T \mu - \frac{\gamma}{2} \omega^T \Sigma \omega.$$

Program 4.3 shows how we can implement the MCM method. Our program starts with simulating NP different initial portfolios. To do so, we use the subroutine `simulate_uniform` from the toolbox. This subroutine fills up the array

Program 4.3 Portfolio problem with MCM

```
program portfoliochoice_MCM
[.....]
! simulate initial portfolios
do ii = 1, NP
    call simulate_uniform(omega_mc(:, ii), 0d0, 1d0)
    omega_mc(:, ii) = omega_mc(:, ii)/sum(omega_mc(:, ii))
enddo

! perform updating steps
do ik = 1, 1000

    ! calculate means and returns of the portfolios
    do ii = 1, NP
        mu_mc(ii) = dot_product(mu, omega_mc(:, ii))
        sig_mc(ii) = dot_product(omega_mc(:, ii), &
                                matmul(sig, omega_mc(:, ii)))
    enddo

    ! get portfolio that maximizes investors utility
    maxi = maxloc(mu_mc - gamma/2d0*sig_mc, 1)

    ! store maximum
    omega = omega_mc(:, maxi)

    ! simulate an updated set of portfolios around maximum
    do ii = 1, NP-1

        ! get new normalized portfolio
        call simulate_uniform(omega_mc(:, ii), 0d0, 1d0)
        omega_mc(:, ii) = omega_mc(:, ii)/sum(omega_mc(:, ii))

        ! create linear combination between old and new
        omega_mc(:, ii) = 0.9d0*omega + 0.1d0*omega_mc(:, ii)
    enddo

    ! store maximum in last entry
    omega_mc(:, NP) = omega
enddo
[.....]
end program
```

`omega_mc(:, ii)` with independent realizations of a uniform random variable. To generate portfolio shares out of these random variables, we simply have to normalize them by their sum. Note that the share of risk-free assets is stored in the entries `omega_mc(0, ii)`. Having provided such an initial guess, we start the iterative part of the MCM method. We first calculate the average returns and return variances associated with the different portfolios. Next, we evaluate the utility the investor would get from choosing any of these investment options. Using the function `maxloc`, we can identify the portfolio that guarantees the highest utility to the investor and store the respective portfolio shares in the array `omega`. We now have to provide a new set of simulated portfolios that is located somewhere near the maximum utility portfolio in `omega`. To this end, we just simulate a completely new portfolio using `simulate_uniform` and create a linear combination between this new portfolio and the one stored in `omega`. The weight attached to the new portfolio governs how far away from the portfolio in `omega` we actually want to search. Lastly, we store our maximum utility portfolio in the last entry of `omega_mc` and start the iteration process again. We repeat this iteration 1,000 times. After the whole iteration procedure has ended, the optimal portfolio of the investor is stored in the array `omega`, where `omega(0)` informs us about the share of risk-free assets the investor wants to hold. Note that the MCM procedure relies on drawing realizations of a random variable. Hence, it is quite likely that the results of this procedure will differ slightly every time we restart the program. In our example, we simulate `NP = 1000` portfolios 1,000 times. This already brings us close to the optimal choice of the investor arrived at in Section 4.1.2. Increasing either `NP` or the number of iterations will certainly bring us even closer to the optimum.

4.2 Option pricing theory

Derivatives are financial contracts for which the payoff depends on the value of some underlying financial variable in the future. Option contracts are a specific form of derivatives, where the *holder* has the right but not the obligation to buy or sell a certain asset for a specified *strike price* at a defined future date. The *writer* is the party who sells the option and has the responsibility to fulfil the trade if the holder opts to action it. A *call option* gives the holder the right to buy the underlying asset for the strike price from the writer, while a *put option* gives the holder the right to sell the underlying asset to the writer for the strike price. We can make a distinction between *European options* which can be exercised only on the expiry date, and *American options* which can also be exercised on any date prior to expiry. One of the fundamental problems in finance is how to determine the price of such options. In the following we first present an intuitive binomial approach to provide a solution for the European and the more difficult American-option pricing problem. We then derive a closed-form solution for

European options with more sophisticated mathematical tools and compare the results of both approaches numerically. Note that in both approaches we do not consider any cash payments (such as dividends) made by the underlying asset.

4.2.1 THE BINOMIAL APPROACH BY COX-ROSS-RUBINSTEIN

Option pricing is mostly concerned with the question of how to calculate premiums, meaning the price the holder has to pay to the writer at the outset to obtain the option. The major determinant of such premiums is the assumption made about the way asset prices behave over time. Clearly the price of any risky financial asset is uncertain, but what can be said about its distribution? In order to answer this question, we assume the following reasonable properties: (i) the price of the underlying asset is never zero, and (ii) the return as well as the associated uncertainty tend to increase over time. Consider a specific stock and let S_{tn} denote one possible price of that stock at time t . In this subsection we assume that the stock price follows a multiplicative binomial generating process, where at the end of one time period the stock price may either increase to $u \cdot S_{tn}$ with probability p or decrease to $d \cdot S_{tn}$ with probability $1 - p$. The stock price thus evolves along a binomial tree as shown in Figure 4.3. We denote the starting price of the stock by S_0 . Owing to the multiplicative nature of the binomial process described above, we can compute all potential future prices of the stock S_{tn} for $t > 0$ by

$$S_{tn} = S_0 u^n d^{t-n},$$

where n denotes the number of price increases. In order to ensure that the stock price will not fall below a value of zero, the downward multiplier d must be less than one and greater than zero. On the other hand, there is no upper bound on the value the stock price may take. Hence, u can take on any value greater than one. We now want to look at the development of the stock price over a time period $[0, \Delta T]$, where $t = 0$ denotes

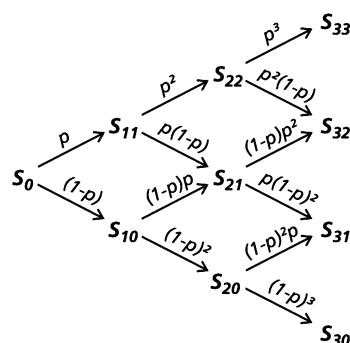


Figure 4.3 Stock prices with a three-period binomial process

the period in which the option is issued and $t = \Delta T$ is the expiry date of the option. We divide this time period into T sub-periods of equal length $\Delta t = \frac{\Delta T}{T}$ and assume that in each of these sub-periods the stock price either increases or decreases with probabilities p and $1 - p$, respectively.

Now let us first look at a European option. 1 The payoff of such an option at expiry date ΔT is the difference between the strike price K —meaning the price for which the holder can buy or sell the underlying asset—and the actual market price of the asset S_{Tn} .¹ The holder of a call option will only buy the asset when the market price is higher than the strike price. Consequently, for the holder of a European call option the payoff is $\max[S_{Tn} - K, 0]$. Vice versa, the payoff for the holder of a European put option is $\max[K - S_{Tn}, 0]$. When signing the option contract, the holder has to pay a premium c^E to the writer. Hence, the holder's overall *profit* is defined as the difference between payoff and premium, meaning that her maximum loss is limited to the premium paid while the profit has no upper limit.² On the other hand, the profit of the writer is the negative of the profit for the holder. Consequently, the writer's profit of a call option is capped at the value of the premium, while the loss is potentially unlimited, see Figure 4.4. The opposite holds for a European put option.

In order to determine the value of a European option, we have to compute the expected payoff at expiry ΔT discounted to the initial period 0. Therefore we need to know the probability distribution of possible stock prices S_{Tn} at the expiry date. For example, from Figure 4.3 we can see that the probability for a price of S_{30} is $(1 - p)^3$, for a price of S_{31} it is $2p(1 - p)^2$, for a price of S_{32} it is $2p^2(1 - p)$, and the price S_{33} occurs with probability p^3 . More generally, the probability of seeing a final price S_{Tn} is a binomial probability of seeing n price increases during T sub-periods, given that the probability of a price increase is p , that is

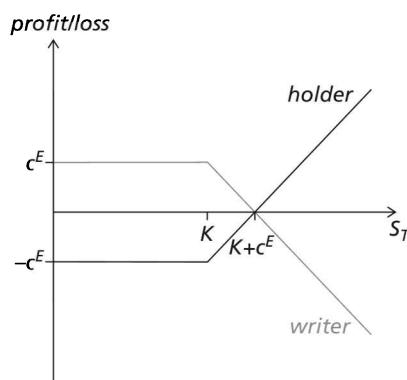


Figure 4.4 Profit/loss diagram for European call option

¹ Recall that at time ΔT there are T different potential stock prices as we separated the time period we consider into T sub-periods.

² Note that we abstracted from issues of time discounting in this graph.

$$P(S_{Tn}) = B(n|T, p) = \binom{T}{n} p^n (1-p)^{T-n} = \frac{T!}{(T-n)!n!} p^n (1-p)^{T-n}. \quad (4.9)$$

Given this probability as well as the no-arbitrage condition that an investor cannot make an expected positive profit of investing in the option, we can compute the premium of a European call option as a function of the initial stock price S_0 , the strike price K and the time to expiry ΔT as

$$c^E(S_0, K, \Delta T) = e^{-r\Delta T} \sum_{n=0}^T B(n|T, p) \times \max[S_{Tn} - K, 0]. \quad (4.10)$$

r thereby denotes the discount factor with which we discount profit flows in period ΔT to the initial period 0.³

To be able to compare the binomial option price in (4.10) derived in discrete time with respective prices from the continuous-time Black-Scholes model that we analyse below, we need to rearrange the call price in the following way: First, we drop all final states in which the payoff of the call option is zero. Let a denote the smallest index for which the final stock price is greater than the strike price. Then we can write

$$\begin{aligned} c^E(S_0, K, \Delta T) &= e^{-r\Delta T} \sum_{n=a}^T \frac{T!}{(T-n)!n!} p^n (1-p)^{T-n} [S_0 u^n d^{T-n} - K] \\ &= S_0 \sum_{n=a}^T \frac{T!}{(T-n)!n!} p^n (1-p)^{T-n} e^{-r\Delta T} u^n d^{T-n} \\ &\quad - e^{-r\Delta T} K B(n \geq a|T, p). \end{aligned}$$

Second, we can write the sum in the previous equation as

$$\begin{aligned} &\sum_{n=a}^T \frac{T!}{(T-n)!n!} p^n (1-p)^{T-n} e^{-r\Delta T} u^n d^{T-n} \\ &= \sum_{n=a}^T \frac{T!}{(T-n)!n!} [e^{-r\Delta t} up]^n [e^{-r\Delta t} d(1-p)]^{T-n} \\ &=: B(n \geq a|T, p', q') \end{aligned}$$

with $p' = e^{-r\Delta t} up$ and $q' = e^{-r\Delta t} d(1-p)$. $B(n \geq a|T, p', q')$ denotes the complementary binomial probability that $n \geq a$ measured as of date 0. Note that the p' and q' do not have to sum up to a total value of 1, which is why we use a more general definition of the binomial probability in this case. Finally we can write

³ This discount factor equals the return of a risk-free investment opportunity.

$$c^E(S_0, K, \Delta T) = S_0 B(n \geq a | T, p', q') - e^{-r\Delta T} K B(n \geq a | T, p). \quad (4.11)$$

Of course, the price of a put option can be derived in an analogous way. The premium of the call option increases in value when the initial stock price S_0 rises, and declines with a higher exercise price K . An increase in the risk-free rate r decreases the discounted values of the final stock price as well as the exercise price, so that the total effect on the call option premium is unclear. The same is true for a put option, but with opposite signs.

In contrast to European options which can only be exercised at the date of maturity ΔT , American options can also be exercised before expiry, so that they must be evaluated in each period t using a recursive formulation. We denote by c_{tn} the value of the American option in sub-period t if the price of the underlying stock is equal to S_{tn} . If an American call option is not exercised until the date of maturity, the possible payoff is

$$c_{Tn} = \max[S_{Tn} - K, 0].$$

To be able to decide whether it is worth exercising earlier, the investor compares the discounted expected value of future payoffs \tilde{c}_{tn} and the payoff $S_{tn} - K$ from exercising right now, so that the value of the option in t is

$$c_{tn} = \max[S_{tn} - K, \tilde{c}_{tn}] \quad \text{with} \quad \tilde{c}_{tn} = e^{-r\Delta t} [pc_{t+1n+1} + (1-p)c_{t+1n}]. \quad (4.12)$$

This recursive formulation can be applied in every sub-period $t = T-1, \dots, 1$ until in $t=0$ the price of the American call option reads

$$c^A = e^{-r\Delta t} [pc_{11} + (1-p)c_{10}].$$

We can compute the premium of an American put option in an analogous way. Note that owing to the opportunity to exercise early, the value of the American option must be always equal to or greater than its European option counterpart.

4.2.2 THE BLACK-SCHOLES FORMULA

The previous considerations used a discrete time setup to evaluate the premiums of different options. There is a continuous-time equivalent option pricing formula, which can be derived directly from the binomial pricing model by holding the time to maturity constant and dividing it into more and more sub-periods. In the following, however, we derive the formula explicitly from the underlying continuous-time price dynamics. Let us now assume that the price of the stock is distributed log-normally. We assume that the stock has an annual mean return of r per cent and the log of the stock price has a standard

deviation of σ per cent. The daily price movement of the stock between two dates t and $t + \Delta t$ can then be simulated by

$$S_{t+\Delta t} = S_t e^{\left(r - \frac{\sigma^2}{2}\right) \Delta t + \sigma \sqrt{\Delta t} z} \quad \text{with } z \sim N(0, 1). \quad (4.13)$$

Since z is a standard normal variable (i.e. it has mean 0 and a standard deviation of 1), the logarithm of $S_{t+\Delta t}$ is also normally distributed, i.e.

$$\log S_{t+\Delta t} \sim N\left(\log S_t + \left(r - \frac{\sigma^2}{2}\right) \Delta t, \sigma^2 \Delta t\right).$$

Consequently, given a stock price S_0 at $t = 0$, we can compute the expected value and the variance in a future period ΔT from

$$\begin{aligned} E[S_{\Delta T}] &= e^{\log S_0 + \left(r - \frac{\sigma^2}{2}\right) \Delta T + \frac{\sigma^2}{2} \Delta T} = S_0 e^{r \Delta T} \quad \text{and} \\ \text{Var}[S_{\Delta T}] &= e^{2[\log S_0 + \left(r - \frac{\sigma^2}{2}\right) \Delta T] + \sigma^2 \Delta T} \left(e^{\sigma^2 \Delta T} - 1\right) = S_0^2 e^{2r \Delta T} (e^{\sigma^2 \Delta T} - 1). \end{aligned}$$

We thereby used the fact that if $X \sim N(\mu, \sigma^2)$ is normally distributed with mean μ and variance σ^2 , then e^X is log-normally distributed with mean and variance

$$E[e^X] = e^{\mu + \sigma^2/2} \quad \text{and} \quad \text{Var}[e^X] = e^{2\mu + \sigma^2} (e^{\sigma^2} - 1),$$

respectively. Using the normalized value $r - \frac{\sigma^2}{2}$ for the mean of the annual log-normal stock price distribution, the expected stock price grows exactly with the interest rate r .

Program 4.4 shows how we can simulate a potential realization of log-normal stock prices. We first have to set some parameters. We want to simulate the movement of a stock price over 62 business days. Assuming the year has (approximately) 250 business days, we have to set $\Delta T = \frac{62}{250}$. The stock has an initial price of $S_0 = 25$, its annual return is $r = 4$ per cent and the standard deviation is $\sigma = 10$ per cent. To calculate option prices later on, we set the strike price to a value of $K = 25$. For the binomial option pricing model we furthermore assume $T = 200$ sub-periods. In order to simulate a log-normal price process, we first need to generate a sequence of standard normally distributed random numbers z . We therefore use the subroutine `simulate_normal()` that receives as input a vector z of arbitrary length as well as the mean and variance of the normal distribution. Given the vector of realizations z , we can compute the corresponding path for the stock price using equation (4.13). Figure 4.5 shows the result. On the horizontal axis we can see the time frame in business days. The vertical axis shows the simulated stock price. In our example, the stock appreciates quite a bit after a couple of business days, which would make a call option quite profitable. Yet, this simulated path is only one out of an infinite number of potential price paths. By rerunning the program a couple of times we might equally likely simulate a stock market crash.

Program 4.4 Simulation of log-normal stock prices

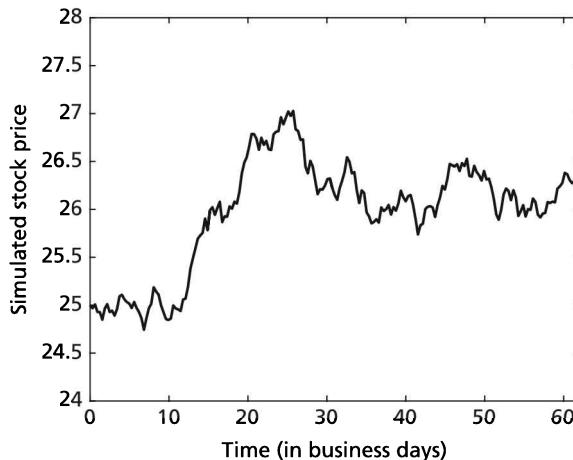
```

program Optionprice
[.....]
real*8, parameter :: Del_TT = 62d0/250d0
real*8, parameter :: S_0 = 25d0
real*8, parameter :: r = 0.04d0
real*8, parameter :: sigma = 0.10d0
real*8, parameter :: K = 25d0
integer, parameter :: TT = 200
[.....]
! get length of sub-periods
del_t = Del_TT/dble(TT)

! generate standard normal random numbers
call simulate_normal(z, 0d0, 1d0)

! simulate stock price movement
S(0) = S_0
do it = 1, TT
    t(it) = dble(it)*del_t*250d0
    S(it) = S(it-1)*exp((r-sigma**2/2d0)*del_t +
                         + sigma*sqrt(del_t)*z(it))
enddo

```

**Figure 4.5** Simulated stock price for one year

To derive the Black-Scholes formula, we assume that the distribution of stock values at time ΔT is described by the probability density function $f(S_{\Delta T})$ and the cumulative density function $F(S_{\Delta T})$. As before, the price of a European call option is the discounted expected profit

$$\begin{aligned}
 c^E(S_0, K, \Delta T) &= e^{-r\Delta T} E[\max [S_{\Delta T} - K, 0]] \\
 &= e^{-r\Delta T} \int_{-\infty}^{\infty} \max [S_{\Delta T} - K, 0] f(S_{\Delta T}) dS_{\Delta T}.
 \end{aligned}$$

Since negative profit values are censored at zero, we can also write

$$\begin{aligned}
 c^E(S_0, K, \Delta T) &= e^{-r\Delta T} \int_K^\infty (S_{\Delta T} - K) f(S_{\Delta T}) dS_{\Delta T} \\
 &= e^{-r\Delta T} \int_K^\infty S_{\Delta T} f(S_{\Delta T}) dS_{\Delta T} - e^{-r\Delta T} K \int_K^\infty f(S_{\Delta T}) dS_{\Delta T} \\
 &= e^{-r\Delta T} S_0 e^{r\Delta T} \Phi \left(\frac{-\log K + \log S_0 + (r + \sigma^2/2)\Delta T}{\sigma \sqrt{\Delta T}} \right) \\
 &\quad - e^{-r\Delta T} K \left(1 - \Phi \left(\frac{\log K - \log S_0 - (r - \sigma^2/2)\Delta T}{\sigma \sqrt{\Delta T}} \right) \right) \\
 &= S_0 \Phi(d_1) - e^{-r\Delta T} K \Phi(d_2)
 \end{aligned}$$

with

$$d_1 = \frac{\log \left(\frac{S_0}{K} \right) + \left(r + \frac{\sigma^2}{2} \right) \Delta T}{\sigma \sqrt{\Delta T}} \quad \text{and} \quad d_2 = d_1 - \sigma \sqrt{\Delta T}.$$

In the above formulas $\Phi(\cdot)$ defines the cumulative density function of the standard normal distribution. Note the similarity with formula (4.11) from the binomial model. For more details on the computation see the appendix to this chapter. Similarly the Black-Scholes formula for pricing European put options is

$$\begin{aligned}
 p^E(S_0, K, \Delta T) &= e^{-r\Delta T} \int_{-\infty}^K (K - S_{\Delta T}) f(S_{\Delta T}) dS_{\Delta T} \\
 &= -S_0 \Phi(-d_1) + e^{-r\Delta T} K \Phi(-d_2).
 \end{aligned}$$

4.2.3 NUMERICAL IMPLEMENTATION OF BOTH APPROACHES

Program 4.4.a how to calculate the prices of European call and put options using the Black-Scholes formula. We first have to compute the values of the parameters d_1 and d_2 according to the above formula. We then apply the function `normalCDF` from the toolbox. This function receives three arguments. A point at which to evaluate the cumulative normal distribution as well as the distributions mean and variance.

Next we turn to the binomial model. In order to compare the results of both approaches we have to choose the probabilities p and the upward and downward movement u and d in a specific way, so that expected values and variances are identical in both approaches. In the appendix we show that this yields

$$u = b + \sqrt{b^2 - 1}, \quad d = b - \sqrt{b^2 - 1} \quad \text{and} \quad p = \frac{e^{r\Delta t} - d}{u - d}, \quad (4.14)$$

Program 4.4.a Computation of Black-Scholes option prices

```

! use Black-Scholes formula for call and put price
d_1 = (log(S_0/K) + (r+sigma**2/2)*Del_TT)/(sigma*sqrt(Del_TT))
d_2 = d_1 - sigma*sqrt(Del_TT)
c_E = S_0*normalCDF( d_1, 0d0, 1d0) &
      - K*exp(-r*Del_TT)*normalCDF( d_2, 0d0, 1d0)
p_E = -S_0*normalCDF(-d_1, 0d0, 1d0) &
      + K*exp(-r*Del_TT)*normalCDF(-d_2, 0d0, 1d0)

```

Program 4.4.b European option prices using the binomial tree

```

! initialize parameters for a binomial tree
b = 0.5d0*(exp(-r*del_t) + exp((r + sigma**2)*del_t))
u = b + sqrt(b**2 - 1d0)
d = b - sqrt(b**2 - 1d0)
p = (exp(r*del_t) - d)/(u - d)

! determine stock price realizations
S_tn(0, 0) = S_0
do it = 1, TT
    do n = 0, it
        S_tn(it, n) = S_0*u**n*d** (it-n)
    enddo
enddo

! compute call and put price of a European option
c_E = 0d0
p_E = 0d0
do n = 0, TT
    c_E = c_E + binomialPDF(n, TT, p)*max(S_tn(TT, n) - K, 0d0)
    p_E = p_E + binomialPDF(n, TT, p)*max(K - S_tn(TT, n), 0d0)
enddo
c_E = c_E*exp(-r*Del_TT)
p_E = p_E*exp(-r*Del_TT)

```

where $b = 0.5(e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t})$. In Program 4.4.b we first initialize these parameters. We then simulate the values of all possible future stock prices S_{tn} at each time t . Having done that, we can immediately calculate the value of a European call or put option. To this end, we only have to multiply the profit the investor would make in each potential state at the time of expiry $\max [S_{Tn} - K, 0]$ with the respective binomial probability $B(n|T, p)$ for this stock price to occur. For calculating binomial probabilities we use the function `binomialPDF` from the toolbox, which should be self-explanatory.

Program 4.4.c shows how to compute the premium of an American call or put option using the binomial approach. Here we proceed recursively. We first calculate the (potential) profit for the investor for each possible realization of the stock price at the date of expiry c_{Tn} . Then we iterate backwards and calculate the value of the American option at prior dates applying formula (4.12). Finally, the value of the American option at time $t = 0$ is the discounted value of the two potential option values at time $t = 1$, weighted by the respective probabilities.

Program 4.4.c American options using the binomial tree

```

do n = 0, TT
    c_A(TT, n) = max(S_tn(TT, n) - K, 0d0)
    p_A(TT, n) = max(K - S_tn(TT, n), 0d0)
enddo

! recursive computation of premiums
do it = TT-1, 1, -1
    do n = 0, it
        c_A(it, n) = max(S_tn(it, n) - K, exp(-r*del_t)* &
                           (p*c_A(it+1, n+1) + (1d0-p)*c_A(it+1, n)))
        p_A(it, n) = max(K - S_tn(it, n), exp(-r*del_t)* &
                           (p*p_A(it+1, n+1) + (1d0-p)*p_A(it+1, n)))
    enddo
enddo
c_A(0, 0) = exp(-r*del_t)*(p*c_A(1, 1) + (1d0-p)*c_A(1, 0))
p_A(0, 0) = exp(-r*del_t)*(p*p_A(1, 1) + (1d0-p)*p_A(1, 0))

```

Table 4.4 Option prices for European and American calls and puts

| K (in €) | Parameter values | | | Call options | | Put options | |
|----------|------------------|----------|-------------|--------------|--------------|--------------|--------------|
| | σ (in %) | r (in %) | T (in days) | $c^E(\cdot)$ | $c^A(\cdot)$ | $p^E(\cdot)$ | $p^A(\cdot)$ |
| 25 | 10.0 | 4.0 | 62 | 0.63 | 0.63 | 0.38 | 0.40 |
| 30 | | | | 0.00 | 0.00 | 4.70 | 5.00 |
| | 20.0 | | | 1.12 | 1.12 | 0.87 | 0.89 |
| | | 8.0 | | 0.78 | 0.78 | 0.28 | 0.33 |
| | | | 125 | 0.97 | 0.97 | 0.48 | 0.53 |

Initial stock price $S_0 = 25$.

What is the difference between the binomial approach and the Black-Scholes formula and the prices of European and American options? The answer to the first question is that the Black-Scholes formula represents a limiting case for the binomial option pricing models. Hence, the more sub-periods we use for calculating the option prices in the binomial model, the closer we approach the option value derived from the Black-Scholes formula. With 200 sub-periods as chosen in our program, the option prices derived from the two models are almost indistinguishable. To answer the second question, we should look at Table 4.4, which reports option prices for specific parameter combinations that are all derived from the binomial model. The benchmark parametrization reflects the assumptions above, so that the initial price of the share S_0 is set at 25, and the annual return and standard deviation are set at 4 per cent and 10 per cent respectively. In addition we assume a strike price of 25 and set the time to maturity at 62 business days which is roughly three months. Since the expected price of the stock in three months is higher than the current price, it should not be surprising that the call option is more expensive than the put option given our benchmark assumptions. Surprising on first sight is that the American call option has the same price as the European call option. This means that the right to exercise early is (almost) worthless for the holder of an American call option.

and it is always best to hold the American option until the expiry date. On the other hand, the price of the American put option is much higher than the respective European put option, as the holder may exercise the option prior to the expiry date. When the strike price increases it becomes less likely that the call option will be exercised. Thus, the price of the call option falls while the price of the put option increases. In the present example a strike price of 30 already reduces the price of the call option to zero. At the same time the American put option will be exercised immediately so that the price is equal to 5. Next we again assume a strike price of 25 but double the volatility of the share. In this case, the expected future price of the share remains unaffected, but the probability of exercising the option profitably increases for calls and puts. Consequently, all option prices increase compared to the benchmark.⁴ Doubling the return increases the expected future price $E[S_{\Delta T}]$. Therefore, the impact on option prices is asymmetric. While the price of the call option increases, the put option price falls. Finally, doubling the length ΔT of the options increases the expected price but also the uncertainty. Consequently, both option prices increase compared to our benchmark parameterization.

4.2.4 OPTION PRICING WITH MONTE CARLO SIMULATION

The prices of European options can most conveniently be computed with the Black-Scholes formulas. For American options, we can use the binomial tree model. Pricing more complicated and exotic option products can however be quite challenging. An example for such an exotic option is the so-called *Asian option*, where the payoff is a function of the average of all stock price realizations between the date the option was written $t = 0$ and the expiry date $t = \Delta T$. For a given realization of stock prices, we can hence write the (ex post) payoff of a call option as

$$\pi(\bar{S}) = \max[\bar{S} - K; 0] \quad \text{with} \quad \bar{S} = \frac{1}{\Delta T} \int_0^{\Delta T} S_t dt.$$

Determining the premium for such an option cannot be done analytically. Even in the binomial model this will be a very complicated task since we would need to trace the development of the stock price along the whole binomial tree and cannot just use the terminal stock price realizations. Yet we can apply a numerical simulation approach to approximate the discounted expected payoff of the option. A popular method is the Monte Carlo (MC) simulation approach. For a risk-neutral investor, we can implement an MC simulation-based method to determine the price of an Asian option by again splitting the time frame $[0, \Delta T]$ into T distinct sub-periods of length $\Delta t = \frac{\Delta T}{T}$. We then proceed along the following three steps:

⁴ Intuitively one might think that a higher volatility implies more risk so that prices decline. However, a call option protects the holder against downward movements of the stock price. Consequently the holder is interested in upward movements only, and vice versa for the holder of a put.

- Simulate $j = 1, \dots, J$ sample price paths of the underlying stock S_t , given the initial value S_0 and the relevant time frame ΔT as

$$\hat{S}_t^j = \hat{S}_{t-1}^j e^{(r-\sigma^2/2)\Delta t + \sigma\sqrt{\Delta t}z_t} \quad t = 1, \dots, T,$$

where z_t is a sequence of independent realizations of a standard normally distributed random variable and $\hat{S}_0^j = S_0$.

- Given such a simulated price path $P_j = (\hat{S}_0^j, \dots, \hat{S}_T^j)$ compute the corresponding terminal payoff

$$\pi_j = \begin{cases} \max[\bar{S}_j - K; 0] & \text{for a call option} \\ \max[K - \bar{S}_j; 0] & \text{for a put option} \end{cases}.$$

where $\bar{S}_j = \frac{1}{T} \sum_{t=1}^T \hat{S}_t^j$.

- Compute the average of discounted payoffs π_j from the J sample paths

$$c^{AS} / p^{AS} = \frac{1}{J} \sum_{j=1}^J e^{-r\Delta T} \pi_j.$$

In Program 4.5 we compute the premiums of an Asian call and put option applying the MC simulation method described above. The variable `JJJ` defines the number of sample paths we simulate. All other parameters are identical to the ones in Section 4.2.3. For each simulated path `ij`, we draw a fresh series of standard normally distributed random variable z and generate a price path for the stock. We discussed how to do this in Program 4.4. We then calculate the average of all stock-price realizations over the time period $[0, \Delta T]$ and store the associated profit of a call and a put option in the arrays `pi_c` and `pi_p`, respectively. Having simulated J price paths and computed the associated profits π_j , we can determine the premium of the Asian option by discounting the average payoff from time $t = \Delta T$ to time $t = 0$. To show how the predicted price of the Asian option depends on the number of paths J we actually simulate, we successively increase the number of paths used for the computation of the option price. This means that we start out calculating the option price from only the first simulated path, then we add the second to this, the third, etc. Figure 4.6 shows the evolution of the option price for the first 2,500 simulated paths. Initially there is obviously a lot of fluctuation in the price, since any additional simulated path adds a lot of new information to the pricing equation. As the number of simulated paths increases, however, the option value stabilizes, which is just an application of the law of large numbers. After around 1,000 simulated paths, the price of the Asian call option is around 0.35 and the price of the put option is 0.23. Note that these prices range substantially below the respective prices of the European option. The reason is that the Asian option is based on an average stock price over the time period

Program 4.5 Monte Carlo approximation of Asian option

```

program MC_Option_Pricing
[.....]
! simulate price paths and corresponding profits
do ij = 1, JJ

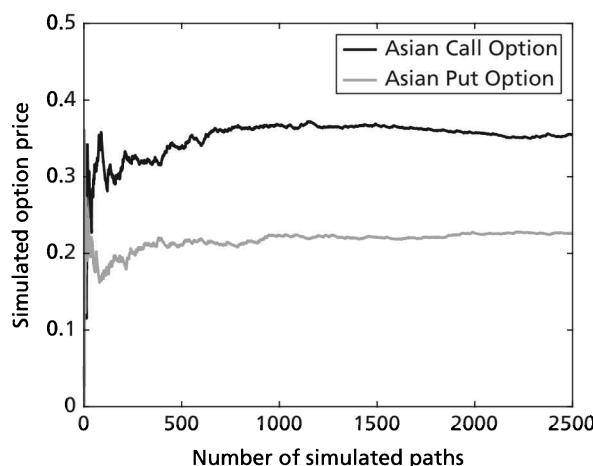
    ! generate a price path
    call simulate_normal(z, 0d0, 1d0)
    S(0) = S_0
    do it = 1, TT
        S(it) = S(it-1)*exp((r-sigma**2/2d0)*del_t &
                           + sigma*sqr(del_t)*z(it))
    enddo

    ! get average stock price
    S_bar = 0
    do it = 1, TT
        S_bar = S_bar + S(it)
    enddo
    S_bar = S_bar/dble(TT)

    ! calculate associated payoff
    pi_c(ij) = max(S_bar - KK, 0d0)
    pi_p(ij) = max(KK - S_bar, 0d0)
enddo

! use different number of paths to compute premium
do ij = 1, JJ
    c_AS(ij) = exp(-r*Del_TT)*sum(pi_c(1:ij))/dble(ij)
    p_AS(ij) = exp(-r*Del_TT)*sum(pi_p(1:ij))/dble(ij)
    npaths(ij) = dble(ij)
enddo
[.....]
end program

```

**Figure 4.6** Convergence of simulated prices of Asian call and put option

$[0, \Delta T]$, while the European option is only based on the final price. Since the final price has a much larger variance than a price average, the European call option gives much more opportunity to seek extreme profit realizations at the time of expiry and therefore creates a higher expected payoff.

4.3 Managing credit risk with corporate bonds

This section provides an introduction to quantitative credit risk management with corporate bonds. The problem can be broadly defined as the estimation of losses due to events such as a default of a debtor or a change in the quality of a credit. Owing to a number of reasons (rising importance of banking regulation, increasing diversification of asset classes/borrowers/investments, etc.) credit risk management has become more and more important in banking. In the following, we explain the basic ideas of credit risk management, but neither elaborate on technical details nor discuss problems associated with the assumed exogenous parameters. Starting with a single corporate loan example, we discuss the required input parameters and then evaluate the associated credit risk with different loss calculations and risk measures. We then move on to a complete credit loan portfolio. Finally, we present some Monte Carlo simulations to estimate the portfolio loss distribution.

4.3.1 MODELLING CREDIT RISK WITH A SINGLE CORPORATE BOND

A corporate bond is a debt instrument a corporation can issue in order to finance its business. In contrast to equity, which pays annual variable dividends that depend on business success, profits, and strategy, a corporate bond has a fixed *maturity* date T at which the holder is entitled to a payment F from the corporation. F is called the face value of the bond. In addition to the face value, the corporation might promise to pay the holder an annual *coupon payment* C_t for $t = 1, \dots, T$. If the size of the coupon payment is constant over time, we call $cr = \frac{C_t}{F}$ the *coupon rate*. We assume that the corporation issues the bond at time $t = 0$. At this point, an investor has to decide at what price he wants to buy the bond. To this end, the investor has to calculate the present value of coupon payments and face value as

$$P_0 = \sum_{t=1}^T \frac{C_t}{(1+r)^t} + \frac{F}{(1+r)^T}.$$

Dividing both sides by the face value, the price of the corporate bond at the issuance date *per unit of nominal exposure* has to be equal to

$$p_0 = \frac{P_0}{F} = \sum_{t=1}^{T-1} \frac{cr}{(1+r)^t} + \frac{1+cr}{(1+r)^T}. \quad (4.15)$$

The question that remains is how to choose a suitable discount rate r for calculating the bond price. Whenever there is a real risk that the corporation that issues the bond might default on its obligation to pay coupons and face value, this discount rate should be higher than the risk-free rate r_f , reflecting the fact that the investor will require a certain risk premium. The *credit spread* cs defines the size of this premium, so that we can write $r = r_f + cs$. The size of the required credit spread in turn depends on the actual probability with which the corporation is expected to default and therefore on the *credit quality*. Note that for a given coupon rate and face value a higher credit spread and therefore a lower credit quality leads to a lower bond price p_0 . Of course, corporate bonds can not only be bought at the issuance date. Instead, there typically exist so-called *secondary markets* in which different investors can buy corporate bonds from each other. The price of a corporate bond in these markets follows the same valuation procedure as above, with the only difference being that the maturity date of the bond comes closer over time. Hence, the price of the same bond at any date $s > 0$ can be calculated from

$$p_s = \sum_{t=1}^{T-s-1} \frac{cr}{(1+r)^t} + \frac{1+cr}{(1+r)^{T-s}}.$$

Bond valuation in practice Next to coupon rate and face value, the most important determinant of a bond's price is the credit quality, which is associated with a certain *probability of default* π_D and a credit spread cs . In practice, estimates for one-year default probabilities are typically provided by rating agencies such as Moody's Investors Service or Fitch Ratings. These rating agencies do not provide individual probabilities for each existing corporate bond. Instead, a bond is assigned a certain credit rating. Every rating agency has its own rating symbols and classification system. For example, Moody's assigns nine generic rating categories from Aaa to C and applies additional numerical modifiers to specific categories. In our numerical example we only use $K = 8$ categories, including ratings from AAA to CCC, see Table 4.5. The rating D is given to bonds where the issuer is actually *in default*. The column D in the table then shows the associated probabilities π_D for a bond in each of the different rating categories to move to the default ranking. Consequently, a corporation that issued a bond rated BBB is expected to default on its obligations within the next year with a probability of 0.26 per cent. The column cs shows the credit spread associated with each of the different rating categories. Again, a bond rated BBB should be evaluated with a credit spread of two percentage points.

During a time horizon of one year, default of the bond issuer is only one possible credit event. The rating agency may also downgrade or upgrade the bond's rating by one or

Table 4.5 Rating transition matrix and credit spreads in per cent

| Rating | AAA | AA | A | BBB | BB | B | CCC | D | cs |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| AAA | 90.24 | 8.99 | 0.56 | 0.05 | 0.08 | 0.03 | 0.05 | 0.00 | 0.00 |
| AA | 0.58 | 90.03 | 8.65 | 0.56 | 0.06 | 0.08 | 0.02 | 0.02 | 0.05 |
| A | 0.04 | 2.00 | 91.58 | 5.71 | 0.40 | 0.17 | 0.02 | 0.08 | 1.00 |
| BBB | 0.01 | 0.13 | 3.89 | 90.69 | 4.18 | 0.68 | 0.16 | 0.26 | 2.00 |
| BB | 0.02 | 0.04 | 0.18 | 5.81 | 84.15 | 7.97 | 0.83 | 1.00 | 5.00 |
| B | 0.00 | 0.05 | 0.15 | 0.25 | 6.31 | 83.16 | 5.01 | 5.07 | 10.00 |
| CCC | 0.00 | 0.00 | 0.20 | 0.30 | 0.91 | 15.96 | 51.31 | 31.32 | 20.00 |

cs: Credit spread compared to a riskless loan in percentage points.

Table 4.6 Industry-specific recovery rates (in %)

| Industry | M | F | U | T | S |
|------------------------------|------|------|------|------|------|
| Expectation (μ_{RR}) | 43.7 | 24.6 | 57.5 | 32.7 | 49.3 |
| Volatility (σ_{RR}) | 25.0 | 25.0 | 25.0 | 25.0 | 25.0 |

several categories if the creditworthiness of the issuer has changed. This so-called *rating migration* directly impacts on the perceived probability of default and therefore on the credit spread vis-à-vis the risk-free rate. For example, if an issuer is currently rated BBB the probability that his rating deteriorates to category BB is 4.18 per cent and it appreciates to A with a probability of 3.89 per cent.

Once an issuer has defaulted on contractual payments, a lawsuit will determine how much of the promised payments the investor will be able to recover. The *recovery rate* denotes the size of these recovery payments as a fraction of the face value F . Recovery rates typically depend on the seniority of the bond and the prevailing economic environment in the specific sector of the bond issuer. In our numerical analysis, we therefore assume that the recovery rate is a random variable, where the expected value depends on the sector a corporation is operating in. Specifically, we distinguish between firms in manufacturing (M), finance (F), utilities (U), transport (T), or services (S). Table 4.6 shows expectations and volatility of the recovery rates for each sector. For simplicity, we assume that the standard deviation is identical for all sectors and equal to 25 per cent.

Expected and unexpected losses in default mode In the following we consider a corporate bond from the finance sector with issuer rating BBB and a nominal exposure (NE) of €1 million.⁵ The bond has a coupon rate of $cr = 5$ per cent and time to maturity is $T =$ four years. The bond price at the issuance date hence can be calculated from (4.15)

⁵ We assume that the face value of each bond is equal to 1, such that for a nominal exposure of €1 million the investor has to buy 1 million bonds. This is, however, not a very restrictive assumption.

with a credit spread of $cs = 5$ per cent and equals $p_0 = 1$. In order to quantify the credit risk associated with our bond, we compute the so-called *expected loss* (EL) as well as the *unexpected loss* (UL). The former measures the average loss an investor can expect to incur during the next year, while the latter quantifies the standard deviation of losses. The literature typically distinguishes loss calculations in *default mode* and *migration mode*. When calculating expected and unexpected losses, default mode calculations only take into account the chance of default as well as the associated loss for the investor. Under migration mode, on the other hand, we also account for the fact that the rating of the bond issuer might be up- or downgraded with a certain probability, which will have an impact on the bond's price.

We first want to focus on calculating the losses of our corporate bond in default mode. We can therefore model the default process as a Bernoulli random variable θ which takes values 0 or 1, where $\theta = 1$ signals default of the bond issuer. This random variable is characterized by a mean and variance of

$$E[\theta] = \pi_D \cdot 1 + (1 - \pi_D) \cdot 0 = \pi_D \quad \text{and}$$

$$\text{Var}[\theta] = \pi_D \cdot (1 - \pi_D)^2 + (1 - \pi_D) \cdot (0 - \pi_D)^2 = \pi_D(1 - \pi_D).$$

Upon default the investor will only recover a certain fraction of the face value. As already discussed, we model the recovery rate as random variable ψ , which has a mean $E[\psi] = \mu_{RR}$ and variance $\text{Var}(\psi) = \sigma_{RR}^2$. Consequently, when the bond's rating changes to D , the price of the bond in the secondary market immediately falls to a value of $p' = \psi$. In default mode, the working assumption is that without default the bond price remains unaffected. The left panel of Figure 4.7 illustrates these considerations.

Given the initial price of p_0 , we can define losses as a random variable \tilde{L} for which we have

$$\tilde{L} = NE \cdot (p_0 - p') = \theta \cdot NE \cdot (p_0 - \psi). \quad (4.16)$$

The expected loss per bond then is given by

$$E[\tilde{L}] = \pi_D \cdot E[NE \cdot (p_0 - \psi) | \theta = 1] + (1 - \pi_D) \cdot 0$$

$$= NE \cdot \pi_D \cdot (p_0 - \mu_{RR}) = NE \cdot \pi_D \cdot LGD,$$

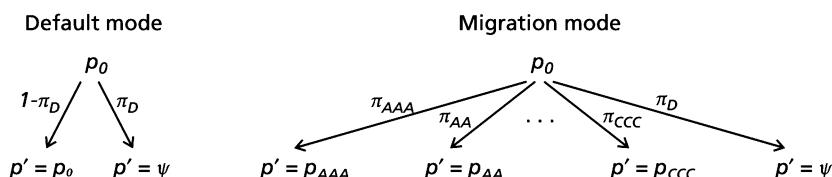


Figure 4.7 Bond price distribution under default and migration mode

where LGD is the expected *loss given default*. With recovery rates and default probabilities from Tables 4.5 and 4.6, the total expected loss of the considered corporate bond is

$$EL_d = NE \cdot \pi_D \cdot LGD = 1,000,000\text{€} \cdot 0.0026 \cdot 0.754 = 1,960\text{€}. \quad (4.17)$$

Similarly, the variance of losses is

$$\begin{aligned} Var(\tilde{L}) &= E[\theta^2 \cdot NE^2 \cdot (p_0 - \psi)^2] - E[\tilde{L}]^2 \\ &= NE^2 \cdot [\pi_D \cdot E[(p_0 - \psi)^2 | \theta = 1] - \pi_D^2 \cdot LGD^2] \\ &= NE^2 \cdot [\pi_D \cdot [\sigma_{RR}^2 + (p_0 - \mu_{RR})^2] - \pi_D^2 \cdot LGD^2] \\ &= NE^2 \cdot [\pi_D \cdot \sigma_{RR}^2 + LGD^2 \cdot \pi_D(1 - \pi_D)], \end{aligned}$$

where we have used the definition $\sigma_z^2 = E[z^2] - E[z]^2$ for the variance of a random variable. The unexpected loss of the corporate bond, which is equal to one standard deviation of the loss variable, can consequently be computed from

$$\begin{aligned} UL_d &= NE \cdot \sqrt{\pi_D \cdot \sigma_{RR}^2 + LGD^2 \cdot \pi_D(1 - \pi_D)} \\ &= 1,000,000\text{€} \cdot \sqrt{0.0026 \cdot 0.25^2 + 0.754^2 \cdot 0.002593} = 40,457\text{€}. \end{aligned}$$

Expected and unexpected losses in migration mode The default mode calculations only account for the possibility of default of the issuer. However, a much more common credit event is an up- or downgrade in the issuer's rating, leading to a change in the quality of the credit and therefore in the price of the bond on the secondary market. For a given maturity T and coupon rate cr , the price of the bond falls or increases upon a change in credit rating to

$$p' = p_k = \sum_{t=1}^{T-1} \frac{cr}{(1 + r_f + cs_k)^t} + \frac{1 + cr}{(1 + r_f + cs_k)^T},$$

where $k \in \mathcal{R} = \{AAA, AA, \dots, CCC\}$ is the new rating and cs_k the corresponding credit spread, see (4.15). Again, when the issuer defaults, the price of the bond drops to $p' = \psi$. The probability distribution of the different credit scenarios occurring is given by the probabilities π_k in the respective row of Table 4.5. The right panel of Figure 4.7 illustrates the possible price changes of the bond in migration mode.

With the random variable of losses again being $\tilde{L} = NE \cdot (p_0 - p')$, we can immediately calculate the expected loss under migration mode as

$$EL_m = E[\tilde{L}] = NE \cdot \left[\sum_{k \in \mathcal{R}} \pi_k \cdot (p_0 - p_k) + \pi_D \cdot LGD \right]. \quad (4.18)$$

In the same way, we can define the variance of \tilde{L} as

$$\begin{aligned} V[\tilde{L}] &= \sum_{k \in \mathcal{R}} \pi_k \cdot NE^2 \cdot (p_0 - p_k)^2 + \pi_D \cdot E[NE^2 \cdot (p_0 - \psi)^2 | k = D] - E[\tilde{L}]^2 \\ &= NE^2 \cdot \left[\sum_{k \in \mathcal{R}} \pi_k (p_0 - p_k)^2 + \pi_D \cdot [\sigma_{RR}^2 + LGD^2] - \left[\frac{EL_m}{NE} \right]^2 \right] \end{aligned}$$

and hence the unexpected loss reads

$$UL_m = NE \cdot \sqrt{\sum_{k \in \mathcal{R}} \pi_k (p_0 - p_k)^2 + \pi_D \cdot [\sigma_{RR}^2 + LGD^2] - \left[\frac{EL_m}{NE} \right]^2}.$$

Numerical implementation Before we think about how to calculate expected and unexpected losses, we need to find a way to represent the properties of a certain bond in a nice and efficient way. One possibility would be to define several different variables that capture the coupon rate, maturity, credit rating, etc. There is however a more elegant way to do this. Fortran enables the user to define so-called *derived data types*. A derived data type allows the user to group together different variables under one name. In this sense, the derived data type is similar to an array. However, unlike in an array, the variables grouped together can be of different formats. The data type declaration must be done within the variable declaration statement. Program 4.6 illustrates this. The definition of a new data type starts with the statement `type` followed by two colons and the name the new data type should have. In our case, we name our data type `bond_type`. The declaration statement ends with the command `end type`. In between those two statements, we can associate different variables (or arrays or previously defined new data types) with our new variable `bond_type`. We want our `bond_type` to be defined

Program 4.6 User-defined data types

```
program creditriskmanagement
[.....]
! define a bond type
type :: bond_type
    character(len=1) :: industry
    character(len=3) :: rating
    real*8 :: couponrate
    integer :: maturity
    real*8 :: exposure
end type

! declare the bond variable
type(bond_type) :: bond
[.....]
! set up a bond
bond = bond_type('F', 'BBB', 5.00d0, 4, 1000000d0)
[.....]
end program
```

by an industry, a certain rating (both in characters), a coupon rate, a maturity date, and the amount of nominal exposure. Once we have defined our new data type, we have to actually declare a variable that is of this new type. This is done in a way similar to when we declare any other variable. We just write `type(bond_type)` followed by two colons and then can define the variable name, in our case `bond`. Last but not least, we have to assign values to the single elements of our new `bond` variable. There are in fact two ways to do this. Whenever we define a new data type, Fortran automatically creates a new function that allows us to assign values to all elements of this new type in one statement. This is shown in Program 4.6 where we just write `bond = bond_type(...)`. The inputs to the function `bond_type` are five values that define industry, rating, coupon rate, maturity, and nominal exposure as defined in our new data type. We use the same specification of the `bond` as for all our previous calculations. An alternative way of defining the values of the elements of `bond` would be to separately assign a value to each of the single elements. We can access such an element by writing `bond%industry = 'F'`, for example. The `%` sign allows us to access a specific element of `bond` both to write a value to it or to get and use the respective value.

After having defined the bond's properties, we also need to define the corresponding rating data. This is done in the subroutine `init_data`, in which we define a two-dimensional array `RM_data` that hosts the rating migration probabilities of Table 4.5, an array `cs_data` in which we store the credit spreads associated with each rating as well as an array `RR_data` that contains the expectations of the recovery rate μ_{RR} from Table 4.6. While the single elements of these arrays are specified by integer numbers, our industry and rating data stored in the type `bond` are given in characters. To translate this character data into the corresponding integer entries of the arrays `RM_data`, `cs_data`, and `RR_data`, we use coding functions `ik_code(rating)` and `ij_code(industry)`. These functions receive as input a rating or an industry classification in character form, for example '`BBB`' or '`F`', and return an integer value that allows us to access the respective data, in our case `ik = 4` and `ij = 2`. Program 4.6.a shows the coding function for the industry of the corporation that issued the bond. The function simply takes the industry classification, which is a single character, and uses a simple `select` statement to find out what is the corresponding industry code. This industry code is then returned to the user. Using the coding functions, we can retrieve our bond's properties as shown in Program 4.6.b. Note that the coupon rate, maturity, and nominal exposure can be defined directly out of the respective entries in the variable `bond`.

Program 4.6.c shows how to use this data to calculate losses under default mode. We first have to calculate the respective bond price p_0 using the formula in (4.15). Next, we can derive the expected loss given default and apply the formulas for the expected and the unexpected loss. The migration mode calculations in Program 4.6.d are only slightly more complicated. We therefore iterate over all possible rankings `ik_p` the bond could move to. The respective price p' can be calculated from using the bonds coupon rate and the credit spread that corresponds to the respective ranking. We then weight the loss (or gain) in value by the respective probability of the credit event. Last but not least, we have to add the default event weighted by the default probability.

Program 4.6.a Recoding characters to integer values

```

function ij_code(industry)
    character(len=1), intent(in) :: industry
    integer :: ij_code

    select case (industry)
        case('M')
            ij_code = 1
        case('F')
            ij_code = 2
        case('U')
            ij_code = 3
        case('T')
            ij_code = 4
        case('S')
            ij_code = 5
        case default
            stop 'Industry does not exist'
    end select

end function

```

Program 4.6.b Retrieving a bond's properties

```

! retrieve the bond's properties
cr      = bond%couponrate/100d0
TT      = bond%maturity
NE      = bond%exposure
ij      = ij_code(bond%industry)
ik      = ik_code(bond%rating)
pi_D   = RM_data(ik, KK)
pi_k   = RM_data(ik, :)
cs     = cs_data(ik)
mu_RR  = RR_data(ij)

```

Program 4.6.c Losses under default mode

```

! calculate price under current rating
p_0 = 0d0
do it = 1, TT-1
    p_0 = p_0 + cr/(1d0 + r_f + cs)**it
enddo
p_0 = p_0 + (1d0 + cr)/(1d0 + r_f + cs)**TT

! define loss given default
LGD = p_0 - mu_RR

! default mode
EL_d = NE*pi_D*LGD
UL_d = NE*sqr(pi_D*sig_RR**2 + LGD**2*pi_D*(1d0-pi_D))

```

Table 4.7 shows the different credit event as well as the corresponding price changes $p_0 - p'$ and probabilities π_k . When we weight the price changes by the respective probabilities and sum the resulting values, we obtain the expected loss per unit of nominal exposure in migration mode. Table 4.8 summarizes the expected and the unexpected losses under both modes in € values as well as in basis points of nominal exposure. Note that under the migration mode the expected loss is significantly higher, as this mode takes into

Program 4.6.d Losses under migration mode

```

! migration mode
EL_m = 0d0
UL_m = 0d0

! iterate over all potential credit events (except default)
do ik_p = 1, KK-1

    cs = cs_data(ik_p)

    ! calculate price change if credit rating changes
    p_prime = 0d0
    do it = 1, TT-1
        p_prime = p_prime + cr/(1d0 + r_f + cs)**it
    enddo
    p_prime = p_prime + (1d0 + cr)/(1d0 + r_f + cs)**TT

    EL_m = EL_m + pi_k(ik_p)*(p_0 - p_prime)
    UL_m = UL_m + pi_k(ik_p)*(p_0 - p_prime)**2
enddo

! add default state
EL_m = EL_m + pi_D*LGD
UL_m = UL_m + pi_D*(sig_RR**2 + LGD**2)

! determine EL and UL in migration model
EL_m = NE*EL_m
UL_m = NE*sqrt(UL_m - (EL_m/NE)**2)

```

Table 4.7 Credit events, price changes, and EL under migration mode

| Event | π_k | $p_0 - p'$ | $\pi_k(p_0 - p')$ |
|----------|---------|------------|-------------------|
| AAA | 0.0001 | -0.0743 | -0.000007 |
| AA | 0.0013 | -0.0724 | -0.000094 |
| A | 0.0389 | -0.0363 | -0.001412 |
| BBB | 0.9069 | 0.0000 | 0.000000 |
| BB | 0.0418 | 0.0994 | 0.004153 |
| B | 0.0068 | 0.2380 | 0.001618 |
| CCC | 0.0016 | 0.4407 | 0.000705 |
| D | 0.0026 | 0.7540 | 0.001960 |
| Σ | 1.0000 | | 0.006923 |

Table 4.8 Credit risk of corporate bond

| | Expected loss | | Unexpected loss | |
|----------------|---------------|-------|-----------------|--------|
| | in € | in bp | in € | in bp |
| Default mode | 1,960 | 19.60 | 40,457 | 404.57 |
| Migration mode | 6,923 | 69.23 | 52,534 | 525.34 |

account credit events that lead to a substantial downward adjustment in prices but don't lead to immediate default. On the other hand, the increase in the unexpected loss from default to migration mode is fairly modest since default already contains the largest price changes. Overall the investor can expect losses between 20 and 70 basis points, while within one standard deviation, the loss can even be 400 to 525 basis points larger than the expected loss.

4.3.2 CREDIT RISK IN A BOND PORTFOLIO

In subsection 4.3.1 we quantified credit risk in terms of expected and unexpected losses at the level of one security. The natural extension of this analysis is to quantify credit risk when the investor holds a portfolio of bonds. In this case we have to model the co-movement of credit migration and defaults between several bonds. Since measuring the correlation of credit events is very complicated in practice, we only consider two scenarios. One in which credit events are fully uncorrelated and one in which credit events are due to some common underlying shock and therefore have a high positive correlation. As in Section 4.3.1 we first quantify the expected and unexpected portfolio loss in the event of uncorrelated shocks using an analytical approach. We then move to simulating the performance of a bond portfolio using Monte Carlo techniques. This not only allows us to quantify portfolio risk under highly correlated credit events, it also enables us to trace the complete portfolio loss distribution. Using this simulated distribution, we can compute different and more informative tail risk measures.

A corporate bond portfolio Let us assume that our bond portfolio comprises $B = 16$ different bonds with ratings from AA to B as well as different coupon rates, maturities, and nominal exposures. Let us denote \tilde{L}_b the random variable of losses of bond $b = 1, \dots, B$. Then the total portfolio loss is

$$\tilde{L}_p = \sum_{b=1}^B \tilde{L}_b.$$

The expected portfolio loss is then simply given by

$$EL_p = E[\tilde{L}_p] = \sum_{b=1}^B E[\tilde{L}_b] = \sum_{b=1}^B EL_b,$$

where EL_b denotes the expected loss of bond b . When the losses of different bonds are uncorrelated, we can write

$$\text{Var} \left[\tilde{L}_p \right] = \sum_{b=1}^B \text{Var} \left[\tilde{L}_b \right] = \sum_{b=1}^B UL_b^2.$$

Thus, we have

$$UL_p = \sqrt{\sum_{b=1}^B UL_b^2}.$$

Program 4.7 shows how to analytically quantify expected and unexpected losses for a portfolio with uncorrelated credit events between bonds. The first extension to the previous program is that we now need to provide specifications for B different bonds. To this end, we can make our variable `portofolio` of type `bond_type` an array of length `BB`. The array declaration works like any other array declaration in Fortran and the variable `portfolio` follows the same logic as any other array. We can now successively fill up the different entries of `portfolio` using the function `bond_type`. Next we retrieve characteristics for each different bond by using the bond properties stored

Program 4.7 Portfolio with uncorrelated credit events

```
program creditriskmanagement_portfolio
[.....]
! declare the bond variable
type(bond_type) :: portfolio(BB)
[.....]
! set up the portfolio
portfolio(1) = bond_type('F', 'BBB', 5.00d0, 4, 1000000d0)
portfolio(2) = bond_type('M', 'A', 4.63d0, 3, 3000000d0)
portfolio(3) = bond_type('M', 'BBB', 3.13d0, 6, 1000000d0)
[.....]
! iterate over all bonds in portfolio
do ib = 1, BB
    ! retrieve bond's properties
    cr = portfolio(ib)%couponrate/100d0
    TT = portfolio(ib)%maturity
    NE = portfolio(ib)%exposure
    [.....]
    ! define loss given default
    LGD = p_0 - mu_RR
    sum_NE = sum_NE + NE

    ! default mode
    EL_d(ib) = NE*pi_D*LGD
    UL_d(ib) = NE*sqrt(pi_D*sig_RR**2 + LGD**2*pi_D*(1d0-pi_D))
    [.....]
enddo

! calculate expected and unexpected loss for total portfolio
EL_d(BB+1) = sum(EL_d(1:BB))
UL_d(BB+1) = sqrt(sum(UL_d(1:BB)**2))
[.....]
end program
```

in `portfolio(ib)`. From these we can calculate the expected and unexpected loss of each bond. Last but not least, we can use the above formulas to define the expected and unexpected loss of the total portfolio, which we store in the index `BB+1` of the arrays `EL_d` and `UL_d`, respectively.

Table 4.9 summarizes the properties as well as the expected losses of the different portfolio positions. The initial prices of the different bonds vary between 0.79 and 1.31, depending on the coupon rate, maturity, and the issuer rating. In addition to the credit quality, the expected loss in € values is also a function of the nominal exposure in that specific bond. All in all, our portfolio, which features a total nominal exposure of €45 million, has an expected loss in default mode of €53,241 and an expected loss in migration mode of €274,801. In terms of total nominal exposure, the expected losses amount to around 12 and 61 basis points, respectively. The unexpected losses, which are not shown in this table, are a bit higher. In default mode, the unexpected loss amounts to roughly 65 basis points and in migration mode to as high as 116 basis points. Note that our working assumption was that the credit events of different bonds are completely uncorrelated. For the expected loss this is not a relevant assumption as the expected value is a linear operator. For the unexpected loss, however, the assumption about the correlation of credit events between different bonds is essential. More generally, when we assume that the losses between bonds i and j have a correlation of ρ_{ij} , we could calculate the total unexpected loss of the bond portfolio as

Table 4.9 Composition of bond portfolios

| Issuer | Industry | Issuer rating | Coupon rate (cr) (in %) | Maturity (T) (in years) | Price (p_0) | Expected loss (in €) | |
|-----------------|----------|---------------|--------------------------------|--------------------------------|--------------------|-------------------------|---------|
| | | | | | | EL_d | EL_m |
| 1 | F | BBB | 5.00 | 4 | 1.0000 | 1,960 | 6,923 |
| 2 | M | A | 4.63 | 3 | 1.0175 | 1,393 | 7,030 |
| 3 | M | BBB | 3.13 | 6 | 0.9051 | 1,217 | 7,471 |
| 4 | S | BBB | 2.88 | 10 | 0.8363 | 892 | 8,949 |
| 5 | T | BB | 6.50 | 2 | 0.9733 | 12,925 | 22,810 |
| 6 | M | A | 5.00 | 6 | 1.0524 | 2,461 | 19,607 |
| 7 | M | A | 2.63 | 5 | 0.9390 | 1,606 | 12,627 |
| 8 | M | A | 1.88 | 7 | 0.8728 | 174 | 1,889 |
| 9 | F | AA | 6.50 | 11 | 1.3183 | 1,715 | 86,854 |
| 10 | U | A | 6.38 | 3 | 1.0660 | 1,767 | 10,505 |
| 11 | F | A | 4.00 | 5 | 1.0000 | 603 | 3,471 |
| 12 | M | A | 2.00 | 9 | 0.8513 | 198 | 2,628 |
| 13 | M | A | 4.25 | 5 | 1.0111 | 183 | 1,339 |
| 14 | U | BB | 5.13 | 4 | 0.9049 | 6,598 | 22,166 |
| 15 | M | B | 4.38 | 3 | 0.7965 | 18,225 | 18,927 |
| 16 | M | AA | 5.63 | 4 | 1.0958 | 1,317 | 41,599 |
| \sum in bp | | | | | | 53,241 | 274,802 |
| | | | | | | 11.83 | 61.07 |

Industry: F=Finance, M=Manufacturing, S=Services, T=Transport, U=Utilities.

$$UL_p = \sqrt{\sum_{i=1}^B \sum_{j=1}^B \rho_{ij} \cdot UL_i \cdot UL_j}$$

This would, however, require estimating a complete variance-covariance matrix between the losses of different bonds, an avenue we don't want to pursue here. Instead we want to continue with simulating the complete portfolio loss distribution using Monte Carlo techniques.

Monte Carlo simulations In the following we want to simulate the random variables of losses \tilde{L}_b for each of the bonds $b = 1, \dots, B$ in our portfolio. To do this, we proceed in the following steps:

1. Generate a sequence of N independent draws $u_{b,i}$, $i = 1, \dots, N$ from a uniform random variable. The realization $u_{b,i}$ defines the respective credit event of bond $b = 1, \dots, B$.
2. Determine the credit event that is associated with the draw $u_{b,i}$, see below.
3. Compute the price change $p_0 - p'$ that results from the credit event and set the realization of the loss variable to

$$L_{b,i} = NE \cdot (p_0 - p').$$

4. Having derived the loss variable for all bonds, the total loss of the portfolio is

$$L_{p,i} = \sum_{b=1}^B L_{b,i}.$$

When simulating the credit events and the resulting price changes, we consider two different scenarios. In one scenario, we draw independent realizations of the random variable $u_{b,i}$ separately for each of the different bonds in our portfolio. As a result, the portfolio losses are fully uncorrelated like under the previous considerations. In the other scenario, we assume that the credit events of all bonds in simulation i are determined by a common realization of the uniform random variable, meaning $u_{1,i} = u_{2,i} = \dots = u_{B,i}$. As a result, the portfolio credit events are highly correlated.

Program 4.8 shows how we initialize the Monte Carlo simulation procedure. When we want shocks to the credit events of bonds to be correlated we have to set the logical variable `correlated` to a value of `.true.`. In this case, we draw a series of random variables that is common to all bonds $b = 1, \dots, B$. Otherwise, we draw a new series of the uniform random variable for each specific bond. Note that for generating the draws of the uniform random variable $u_{b,i}$, we use the method of *antithetic variates*. In order to simulate N draws from a uniform random variable, we only draw $\frac{N}{2}$ realizations using

Program 4.8 Preparation of the MC simulation of portfolio losses

```

program creditriskmanagement_MC
[.....]
! draw one common series of shocks when correlated
if(correlated)then
    call simulate_uniform(u(1:NN/2), 0d0, 1d0)
    u(NN/2+1:NN) = 1d0 - u(1:NN/2)
endif

! iterate over all bonds in portfolio
do ib = 1, BB

    ! draw new series of random shocks when uncorrelated shocks
    if(.not. correlated)then
        call simulate_uniform(u(1:NN/2), 0d0, 1d0)
        u(NN/2+1:NN) = 1d0 - u(1:NN/2)
    endif

    ! retrieve bond's properties
    cr      = portfolio(ib)%couponrate/100d0
    TT      = portfolio(ib)%maturity
    NE      = portfolio(ib)%exposure
    ij      = ij_code(portfolio(ib)%industry)
    ik      = ik_code(portfolio(ib)%rating)
    pi_D   = RM_data(ik, KK)
    pi_k   = RM_data(ik, :)
    cs     = cs_data(ik)
    mu_RR  = RR_data(ij)

    ! compute price under current rating
    p_0 = 0d0
    do it = 1, TT-1
        p_0 = p_0 + cr/(1d0 + r_f + cs)**it
    enddo
    p_0 = p_0 + (1d0 + cr)/(1d0 + r_f + cs)**TT

    ! define beta distribution parameters and sum up NE
    alpha = mu_RR**2*(1.0d0 - mu_RR)/sig_RR**2 - mu_RR
    beta  = alpha/mu_RR - alpha
    sum_NE = sum_NE + NE

    ! start MC simulation
    [.....]
    enddo
enddo
[.....]
end program

```

the subroutine `simulate_uniform` from the toolbox and for each of these realizations also include the symmetric variant $1 - u_{b,i}$. This ensures that the empirical mean of all $u_{b,i}$ realizations is equal to the true mean of 0.5 of the uniform distribution and reduces the variance of the Monte Carlo estimated expected and unexpected losses substantially. Being equipped with a series of uniform random variable realizations, we iterate over all bonds b , retrieve the bond's properties from the array `portfolio`, and calculate the bond's price p_0 . What we haven't specified so far is our treatment of the recovery rate ψ . We assume that this variable follows a *beta distribution*, meaning that

$\psi \sim \text{Beta}(\alpha, \beta)$. The beta distribution is a family of continuous probability distributions defined on the interval $[0, 1]$ and parametrized by two positive parameters, α and β , that control the shape of the distribution. The advantage of a beta distribution over, for example, the normal distribution is that its realizations are bound to the interval $[0, 1]$. Compared to the uniform distribution, on the other hand, the beta distribution shows more flexibility regarding its mean and variance. In fact, we can write

$$E[\psi] = \mu_{RR} = \frac{\alpha}{\alpha + \beta} \quad \text{and}$$

$$\text{Var}(\psi) = \sigma_{RR}^2 = \frac{\alpha\beta}{(\alpha + \beta)^2(1 + \alpha + \beta)}.$$

Solving these two equations for α and β yields

$$\alpha = \frac{\mu_{RR}^2(1 - \mu_{RR})}{\sigma_{RR}^2} - \mu_{RR} \quad \text{and} \quad \beta = \frac{\alpha}{\mu_{RR}} - \alpha.$$

Last but not least, we sum up the total amount of nominal exposure of the portfolio in a variable `sum_NE`.

After these preparatory steps, we can move on to the actual MC simulation procedure shown in Program 4.8.a. The first step in the MC procedure is to derive the credit event of bond b associated with the draw in the random variable $u_{b,i}$. To this end, we divide the interval $[0, 1]$ into K distinct sub-intervals according to the probability distribution of credit events π_k that depends on the bond's rating. Figure 4.8 shows how to do this. The credit event associated with $u_{b,i}$ then is given by the interval into which the realization actually falls. In the example of Figure 4.8 the credit event would be a rating migration to A . In Program 4.8.a we specifically proceed as follows. We first check whether the credit event is default D , by asking whether $u_{b,i} > 1 - \pi_D$. If the bond issuer does not default, we determine in which subinterval the realization $u_{b,i}$ is located and retrieve the corresponding credit spread. Using this credit spread and the characteristics of the bond we already got in Program 4.8, we can calculate the price p' and set the realization of the loss variable $L_{b,i}$ as nominal exposure multiplied by the price change. Note that under default mode, the loss variable equals zero whenever the bond issuer doesn't default. Only under migration mode are price changes due to rating migration taken into account. Whenever the bond issuer does default, we draw a realization ψ from a beta distribution with parameters α and β as defined before. This can be done using the subroutine `simulate_beta` from the toolbox, which should be self explanatory. We can again define the realization of the loss variable as described above. Having determined all realizations of the bond specific loss variables, we can calculate the portfolio loss for each simulation step i by just adding up the corresponding losses of the different portfolio positions. We store this portfolio loss in the variables `L_d` and `L_m` in index `BB+1`.

Program 4.8.a Actual MC simulation of portfolio losses

```

do ib = 1, BB
    [.....]
    do ii = 1, NN

        ! check whether bond is in default
        if(u(ii) <= 1d0 - pi_D)then

            ! where does the bond migrate
            do ik_p = 1, KK-1
                if(u(ii) <= sum(pi_k(1:ik_p)))then
                    cs = cs_data(ik_p)
                    exit
                endif
            enddo

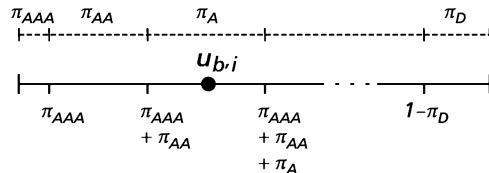
            ! calculate price change if credit rating changes
            p_prime = 0d0
            do it = 1, TT-1
                p_prime = p_prime + cr/(1d0 + r_f + cs)**it
            enddo
            p_prime = p_prime + (1d0 + cr)/(1d0 + r_f + cs)**TT

            L_d(ib, ii) = 0d0
            L_m(ib, ii) = NE*(p_0 - p_prime)
        else

            ! simulate recovery rate under default
            call simulate_beta(psi, alpha, beta)
            L_d(ib, ii) = NE*(p_0 - psi)
            L_m(ib, ii) = NE*(p_0 - psi)
        endif
    enddo
enddo

! get loss of total portfolio
L_d(BB+1, :) = sum(L_d(1:BB, :), 1)
L_m(BB+1, :) = sum(L_m(1:BB, :), 1)

```

**Figure 4.8** Determining the credit event

Having calculated N realizations of the bond specific and portfolio loss distribution, we can visualize the distribution of portfolio losses by drawing a histogram of the realizations $L_{p,i}$. We generate this histogram using the subroutine `plot_hist` from the toolbox for the loss distribution under migration mode. Program 4.8.b shows how this is done. The first input to the subroutine `plot_hist` is the data it should use to generate the histogram, in our case the portfolio loss realizations stored in index $BB+1$ of the variable `L_m`. We express

Program 4.8.b Plot histogram of loss distribution

```

! plot migration mode loss histogram
call plot_hist(L_m(BB+1, :)/sum_NE*10000d0, 99, -200d0, 600d0)
call execplot(xlim=(-/200d0, 600d0/), &
              xlabel='Portfolio loss in bp', ylabel='Relative frequency')

! plot tail distribution
call plot_hist(L_m(BB+1, :)/sum_NE*10000d0, 99, 200d0, 600d0)
call execplot(xlim=(/200d0, 600d0/), &
              xlabel='Portfolio loss in bp', ylabel='Relative frequency')

! calculate EL and UL
do ib = 1, BB+1
    EL_d(ib) = sum(L_d(ib, :))/dble(NN)
    UL_d(ib) = sqrt(sum((L_d(ib, :)-EL_d(ib))**2)/dble(NN-1) )
    EL_m(ib) = sum(L_m(ib, :))/dble(NN)
    UL_m(ib) = sqrt(sum((L_m(ib, :)-EL_m(ib))**2)/dble(NN-1) )
enddo

```

the portfolio losses in basis points of the portfolio's total nominal exposure `sum_NE`. The next three inputs define the number of histogram bins to use as well as the left and right histogram endpoints. In our case, we use 99 bins on the interval -200 to 600 basis points to plot the loss distribution. In a second step, we plot a zoomed-in version of the right tail of the portfolio loss distribution. To this end, we increase the left interval endpoint from -200 to 200 basis points. Figure 4.9 shows the resulting histograms. It should be clear that a negative credit loss refers to an increase in the market value of the portfolio resulting from rating upgrades for some bonds. The scenario that is most likely to happen is that the portfolio value doesn't actually change at all. The likelihood for this event is about 20 per cent. Losses in the portfolio value are much more likely than rating upgrades, which explains a positive expected loss and leads to a fat right tail of the loss distribution. The right panel of Figure 4.9 shows the zoomed in version of the histogram that focuses on this right tail. Note that by default the subroutine `plot_hist` ignores all entries in `L_m` that lead to portfolio losses or gains outside the interval $[200, 600]$ basis points. This

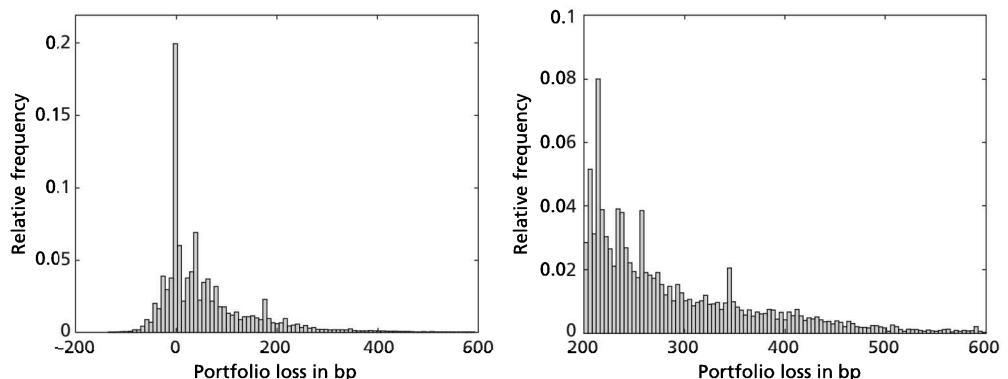


Figure 4.9 Simulated loss distribution under migration mode

explains the higher relative frequency. Overall, losses larger than 500 bp or 5 per cent of the total portfolio nominal expenditure are very unlikely to occur.

Simulated portfolio risk measures Given the simulated loss distribution we can compute various risk measures. The expected and unexpected credit loss for the portfolio can simply be calculated as the empirical mean and standard deviation of the loss realizations

$$EL_p = \frac{1}{N} \sum_{i=1}^N L_{p,i} \quad \text{and} \quad UL_p = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (L_{p,i} - EL_p)^2}.$$

The simulated loss measures can be computed in both default and migration modes. With an increasing number of simulations N , the simulated numbers should converge to the analytic figures reported in the last line of Table 4.9. The expected and unexpected loss of our portfolio only give us useful information about loss variation around the mean. Yet, as we saw in Figure 4.9, the loss distribution in the present problem has quite a long right tail. The unexpected credit loss, however, contains hardly any information about the probabilities of large credit losses along this fat tail. It is therefore useful to complement our analysis of expected and unexpected loss with risk measures that focus particularly on the downside risk of the credit portfolio. As we have simulated the full credit loss distribution, we are able to quantify the probability mass associated with specific tail quantiles of the loss distribution. Figure 4.10 illustrates the information contained in expected and unexpected loss and introduces the *value at risk* (VaR) and the expected shortfall risk or *conditional value at risk* (CVaR), which we want to derive in the following.

The VaR quantifies the maximum loss that we can expect from credit events that occur at a predefined level of confidence. For example, the VaR_{90} provides an estimate of the maximum credit loss we incur with a confidence level of 90 per cent. Hence, the VaR_α at confidence level α simply equals the α -quantile of the loss distribution. Program 4.8.c shows how to calculate the VaR in our example. We therefore copy the portfolio loss

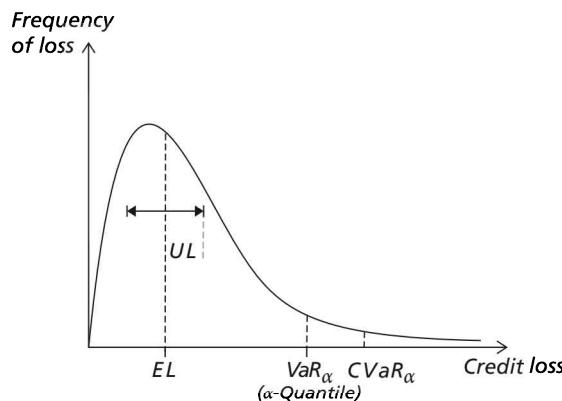


Figure 4.10 Risk measures for portfolio credit risk

Program 4.8.c Calculation of tail-risk measures

```

! calculate value-at-risk
call tic

ii_VaR = ceiling(0.9d0*NN)

! sort portfolio losses (default mode)
L_temp = L_d(BB+1, :)
call sort(L_temp)
VaR_d = L_temp(ii_VaR)
CVaR_d = sum(L_temp(ii_VaR:NN))/db1e(NN-ii_VaR+1)

! sort portfolio losses (migration mode)
L_temp = L_m(BB+1, :)
call sort(L_temp)
VaR_m = L_temp(ii_VaR)
CVaR_m = sum(L_temp(ii_VaR:NN))/db1e(NN-ii_VaR+1)

call toc

```

distribution stored in the index `BB+1` of the array `L_d` or `L_m` into an array `L_temp`. This is simply done to speed up the sorting process that follows next. Since the VaR is the α -quantile of the loss distribution, we have to sort the realizations $L_{p,i}$ in ascending order. This is done using the subroutine `sort` from the toolbox. As sorting typically involves many copying operations, it is much more efficient to operate on the one-dimensional array `L_temp` than on the two-dimensional arrays `L_d` or `L_m`. The sorting algorithm provided in `sort` is quite efficient. However, once the number of simulated realizations exceeds 500,000, sorting can take a couple of minutes (or even hours). To measure the effectiveness of the sorting algorithm, we measure the time we need to calculate the VaR using the subroutines `tic` and `toc`. To derive the VaR_{90} from our sorted portfolio loss data, we have to realize that each simulated $L_{p,i}$ makes up a fraction $\frac{1}{N}$ of our sample. Hence, 90 per cent of the sample is defined by the observation i that satisfies $\frac{i}{N} = 90$ per cent. We can therefore simply read off the VaR_{90} from the entry $i = 0.9N$ of the sorted portfolio loss distribution. The VaR-measure shows us in which interval we can expect losses to be with a certain confidence level. However, it does not tell us a lot about the severity of losses we face whenever losses exceed the VaR level. It could be that two portfolios have the same VaR but the loss severity differs substantially due to different shapes of the distribution's tails. The $CVaR_{90}$ is a measure of the loss severity in the tail part of the loss distribution. The $CVaR$ is the conditional expectation of that part of the credit loss that exceeds the VaR limit. Given the sorted credit losses, the $CVaR_{90}$ is simply the average of all credit loss observations with indices $i = 0.9N, 0.9N + 1, \dots, N$.

Table 4.10 summarizes our simulated portfolio risk measures both in € values and in basis points of the portfolios total nominal exposure for a total number of $N = 100,000$ simulations. We thereby assume that shocks to the credit events of bonds are completely uncorrelated. With this specification the program takes around seven seconds to sort the portfolio loss distribution in both default and migration mode. Note first that the simulated expected losses reported in the first line of the table are quite close to the

Table 4.10 Portfolio risk measures with uncorrelated credit events

| | Default mode | | Migration mode | |
|--------------------|--------------|---------------------|----------------|---------------------|
| | in € | Relative to NE (bp) | in € | Relative to NE (bp) |
| Expected loss | 54,265 | 12.06 | 276,316 | 61.40 |
| Unexpected loss | 299,958 | 66.66 | 529,357 | 117.63 |
| VaR ₉₀ | 0 | 0.00 | 821,217 | 182.49 |
| CVaR ₉₀ | 547,732 | 121.72 | 1,397,084 | 310.46 |

bp: basis points.

Table 4.11 Portfolio risk measures with correlated credit events

| | Default mode | | Migration mode | |
|--------------------|--------------|---------------------|----------------|---------------------|
| | in € | Relative to NE (bp) | in € | Relative to NE (bp) |
| Expected loss | 52,989 | 11.78 | 272,559 | 60.57 |
| Unexpected loss | 622,720 | 138.38 | 1,392,130 | 309.36 |
| VaR ₉₀ | 0 | 0.00 | 170,987 | 38.00 |
| CVaR ₉₀ | 532,917 | 118.43 | 3,219,480 | 715.44 |

bp: basis points.

respective numbers in the last line of Table 4.9, which were derived analytically. The same is true for the simulated unexpected loss numbers, which are also in line with those derived in Program 4.7. Since defaults only occur with a probability much lower than 10 per cent, the VaR₉₀ under default mode equals zero. However, the CVaR₉₀ under default mode clearly indicates a significant risk in the event of default. Under migration mode the VaR₉₀ is positive and the CVaR₉₀ is even greater than 300 basis points or 3 per cent of the portfolio's nominal exposure.

Table 4.11 shows the same measures in the case where we assume highly correlated credit events, meaning that the events of all bonds are owing to one common underlying shock. The assumption of a positive correlation of credit events has no impact on the expected value of the loss distribution. Hence, the expected portfolio loss is hardly affected. However, in the event of a very negative shock, all bonds are affected in the same way, which leads to a much thicker right tail of the portfolio loss distribution. As a consequence the unexpected loss increases substantially compared to Table 4.10. The same is true for the CVaR₉₀ in migration mode. Yet, while the tail of the loss distribution becomes thicker, joint negative credit events become less likely overall. Hence, the VaR declines from 182 to just 38 basis points. This means that with a probability of 90 per cent, the losses of the portfolio will not exceed 38 basis points of the nominal exposure in migration mode. However, if we incur a loss larger than the VaR, we can expect this loss to amount to about 7 per cent of the portfolio.

4.4 Mortality risk management

Risk management is not only done in the banking sector but in many other industries as well. This section presents an example from the life insurance industry, which offers a wide variety of life insurance and pension products. The ongoing demographic transition steadily increases the demand for such products, but at the same time also increases the investment risk of insurance companies. Their profits depend on future mortality rates and mortality projections have become more and more difficult. Insurance companies therefore are trying to keep an efficient risk–return relationship while at the same time meeting some solvency requirements for possible catastrophes. The starting point of our discussion is the generation of mortality tables, which reflect existing longevity risk. Based on these tables it is possible to price specific life insurance and annuity products and analyse the risk structure of insurance portfolios. In a final step, we develop a natural hedging strategy for an insurance company.

4.4.1 MODELLING LONGEVITY RISK

Longevity risk from the perspective of an insurance company denotes the exposure to unexpected changes in life expectancy of the insured clients. Hence, there is a direct connection to mortality risk, which in the literature is typically classified according to three categories:

1. *Individual or unsystematic mortality risk* refers to the risk that, while population-wide probabilities of death and therefore life expectancy are known, an individual's remaining lifetime is still a random variable, meaning that different individuals die at different ages.
2. *Aggregate or systematic mortality risk* is the risk of longer-term unexpected changes to the mortality probabilities of the total population and therefore to average life expectancy.
3. *Adverse selection or basis risk* arises from the fact that owing to heterogeneity in mortality rates and information asymmetries between the insurance company and the insured, mortality of the insured clients does not reflect the mortality profile of the total population.

In our simplified analysis, we completely neglect the issue of adverse selection risk and therefore assume that mortality of the total population and mortality of the insured perfectly coincide. Unsystematic mortality risk, on the other hand, can be (close to) eliminated by enlarging the number of clients and therefore the size of the insurer's portfolio. This is just an application of the law of large numbers which tells us that the life expectancy of the insured approximately equals the life expectancy of the total

population only when the number of clients is large enough. What the insurer is then left with is the systematic component of mortality risk. In the following discussion, longevity risk management will thus refer to the reduction or elimination of such systematic mortality risk.

Death probabilities and life expectancy To simplify the discussion further, we neglect mortality differences that are due to sex, income, region, and so on. Individuals hence are assumed to only differ with respect to their age x . We measure age in five-year intervals. Specifically we consider the age groups 20–24, 25–29, 30–34, ..., 90–94, and 95–99, thereby assuming that individuals don't buy any life insurance products before age 20 and only live up to a maximum age of 99 after which they die with certainty. All in all this makes $X = 16$ age groups, which we index with $x = 1, \dots, X$. The most important variable related to mortality is the *one-period death probability* $q_{x,t}$. This conditional probability quantifies the risk that a person aged x dies within a given period $t = 0, \dots, T$.⁶ The fundamental problem of a life insurer is that those death probabilities are not constant over time. What the insurer can typically observe are death probabilities $q(0) = [q_{1,0}, \dots, q_{T,0}]$ in a given base year $t = 0$ in which he wants to evaluate his portfolio. Yet, these probabilities apply to different cohorts in this base year. When a client at a certain age x buys a life insurance product from the insurer today, however, the relevant mortality rates are the future probabilities of the cohort the client belongs to, meaning $q_{x+t,t}$. To actually price the life insurance product for the client and quantify the product's riskiness, the insurer thus has to forecast these future probabilities and obtain a prospective mortality table and the remaining life expectancy for a certain cohort.

Table 4.12 shows the mortality rates taken from the data in the base year as well as the projected future rates. In the following numerical exercise, we computed the base-year data from the average mortality rates of both sexes in the year 2010 in Germany. Owing to the fact that all surviving individuals die in the terminal age $X = 16$, we know

Table 4.12 Mortality table including future probabilities

| Ages (in years) | Data | Future mortality probabilities | | | | | | | |
|--------------------|------------|--------------------------------|--------------------|--------------------|------|---------------------|---------------------|------|------|
| | | 2010 | 2015 | 2020 | 2025 | ... | 2075 | 2080 | 2085 |
| 95–99 | 1.00 | 1.00 | 1.00 | 1.00 | ... | 1.00 | 1.00 | 1.00 | |
| 90–94 | $q_{15,0}$ | $\tilde{q}_{15,1}$ | $\tilde{q}_{15,2}$ | $\tilde{q}_{15,3}$ | ... | $\tilde{q}_{15,13}$ | $\tilde{q}_{15,14}$ | | |
| 85–89 | $q_{14,0}$ | $\tilde{q}_{14,1}$ | $\tilde{q}_{14,2}$ | $\tilde{q}_{14,3}$ | ... | $\tilde{q}_{14,13}$ | | | |
| : | : | : | : | : | : | | | | |
| 35–39 | $q_{4,0}$ | $\tilde{q}_{4,1}$ | $\tilde{q}_{4,2}$ | $\tilde{q}_{4,3}$ | | | | | |
| 30–34 | $q_{3,0}$ | $\tilde{q}_{3,1}$ | $\tilde{q}_{3,2}$ | | | | | | |
| 25–29 | $q_{2,0}$ | $\tilde{q}_{2,1}$ | | | | | | | |
| 20–24 | $q_{1,0}$ | | | | | | | | |

⁶ Of course we measure periods in the same length as age, meaning over a five-year horizon.

that $q_{X,t} = 1.00$ both in the data and in all future periods. All other mortality rates $\tilde{q}_{x,t}$ are uncertain and we can therefore consider them as random variables. Albeit that we only need to estimate future probabilities of cohorts that are already living in the base year to evaluate an insurer's portfolio, we still need to provide forecasts up to the year 2080.

The conditional survival probabilities $\tilde{q}_{x,t}$ define the probability that an individual of age x in the base year will survive at least t more periods as

$$\tilde{P}_{x,t} = \begin{cases} 1.0 & \text{if } t = 0 \\ \prod_{s=0}^{t-1} (1 - \tilde{q}_{x+s,s}) & \text{if } t \geq 1. \end{cases} \quad (4.19)$$

If the individual has survived for t periods, the probability of dying between period t and period $t+1$ equals $\tilde{q}_{x+t,t}$. Consequently, the probability of living for exactly t more periods is $\tilde{P}_{x,t} \cdot \tilde{q}_{x+t,t}$, so that the remaining life expectancy of an individual aged x in the base year is

$$\tilde{LE}_x = \sum_{t=0}^{X-x} \tilde{P}_{x,t} \cdot \tilde{q}_{x+t,t} \cdot t. \quad (4.20)$$

As the conditional survival probabilities at years $t = 1, \dots, T$ are random variables, remaining life expectancy is a random variable as well.

Modelling systematic mortality risk To quantify the uncertainty in the probability distribution of future mortality, we need to specify a stochastic model for survival probabilities. There are many different approaches to mortality modelling. In this example, we adopt the popular one-factor model pioneered by Lee and Carter, which proposes the process

$$\log \tilde{q}_{x,t} = \alpha_x + \beta_x \cdot \kappa_t + \zeta_{x,t}$$

for the death rate $\tilde{q}_{x,t}$ of a person aged x at period t , where α_x describes the average age-specific mortality. κ_t represents a common time trend in the general mortality level and β_x defines how strong the time trend impacts on mortality rates at different ages x . The error term $\zeta_{x,t}$ is normally distributed with mean zero and represents independent impact factors on the log mortality rate that cannot be captured otherwise. To obtain a uniquely identified model, we impose the two constraints

$$\sum_{x=1}^X \beta_x = 1 \quad \text{and} \quad \kappa_0 = 0,$$

so that the intercept is equal to the log of the death probabilities $\alpha_x = \log q_{x,0}$ in the base year. The parameters of this model are typically estimated from life tables of years prior

to the base year. For the following calculations, we simply use values from the literature and set

$$\beta_x = [0.02, 0.12, 0.12, 0.11, 0.10, 0.08, 0.07, 0.06, 0.05, \\ 0.05, 0.05, 0.05, 0.05, 0.04, 0.03, 0.00].$$

We thereby accounted for the fact that in the last period of life X we always have $q_{X,t} = 1$ which requires $\beta_X = 0$. We furthermore assume that the common time trend κ_t follows a *random walk* with *drift* c , that is

$$\kappa_t = \kappa_{t-1} + c + \epsilon_t \quad \text{with} \quad \epsilon_t \sim N(0, \sigma_\epsilon^2).$$

ϵ_t is independent over time and called the *innovation* to the random walk. The drift is like a linear time trend. Assuming $\kappa_0 = 0$, we consequently have

$$E[\kappa_t] = c + E[\kappa_{t-1}] = c \cdot t + E[\kappa_0] = c \cdot t.$$

Monte Carlo simulations of mortality tables We can use standard MC techniques to simulate different scenarios for future mortality. Overall, we want to provide $k = 1, \dots, K$ different scenarios for the evolution of conditional survival probabilities $q_{x,t}^k$ and therefore of the mortality table shown in Table 4.12. Our working assumption thereby is that we are not interested in noise factors that lie outside the basic mechanisms of the Lee-Carter model, meaning that we assume $\zeta_{x,t} = 0$. Program 4.9 shows how we proceed. The first thing we need to do is to initialize the data input into our model. The data consist of conditional survival probabilities $q_{x,0}$ in the base year as well as the coefficients α_x and β_x of the stochastic process. This initialization is carried out in the subroutine `init_data`, which we don't show here. Next, we simulate the different scenarios for the mortality tables. To this end, we first draw a sequence $\{\epsilon_1^k, \epsilon_2^k, \dots, \epsilon_X^k\}$ of realizations of a normally distributed random variable with mean zero and variance `sigma_eps` using the subroutine `simulate_normal` from the toolbox. Using these realizations we can generate the random walk process according to

$$\kappa_t^k = \kappa_{t-1}^k + c + \epsilon_t^k.$$

Last but not least, we can calculate the corresponding conditional death probabilities from

$$q_{x,t}^k = \exp \left[\alpha_x + \beta_x \cdot \kappa_t^k \right].$$

In addition to the simulated mortality tables $q_{x,t}^k$ for $k = 1, \dots, K$, we add an additional scenario to the index `ik = 0` of the array `q(ix, it, ik)`. In this scenario, we assume

Program 4.9 MC simulation of future mortality scenarios

```

program mortality_riskmanagement
[.....]
! initialize mortality data
call init_data

! MC simulations of mortality tables
do ik = 1, KK

    ! simulate innovations to the random walk
    call simulate_normal(eps, 0d0, sigma_eps)

    ! calculate common risk-factor
    kappa(0) = 0d0
    do it = 1, TT
        kappa(it) = kappa(it-1) + c + eps(it)
    enddo

    ! determine conditional mortality rates (Lee-Carter model)
    do it = 0, TT
        q(:, it, ik) = exp(alpha + beta*kappa(it))
    enddo
enddo

! store current probabilities in entry zero
do it = 1, TT
    q(:, it, 0) = q(:, 0, 0)
enddo

! calculate unconditional survival probabilities and LE
do ik = 0, KK
    do ix = 1, XX
        LE(ix, ik) = 0d0
        PP(ix, 0, ik) = 1d0
        do it = 1, XX - ix
            PP(ix, it, ik) = PP(ix, it-1, ik) &
                *(1d0 - q(ix+it-1, it-1, ik))
            LE(ix, ik) = LE(ix, ik) + PP(ix, it, ik) &
                *q(ix+it, it, ik)*dble(it)
        enddo
    enddo
enddo
[.....]
end program

```

that all future mortality rates are equal to our cross-sectional base-year probabilities $q_{x,0}$.⁷ This will be important for pricing the insurer's payments later on. Using the mortality probabilities in the different simulated scenarios, we can finally derive realizations of the unconditional survival probabilities of an agent aged x in the base year over t periods, $P_{x,t}^k$, as well as the corresponding life expectancies LE_x^k from the formulas (4.19) and (4.20).

Figure 4.11 shows the average of unconditional survival probabilities over all K different simulations as well as the set of probabilities that generates the smallest and

⁷ In terms of the Lee-Carter model this would mean $c = 0$ and $\epsilon_t^k = 0$.

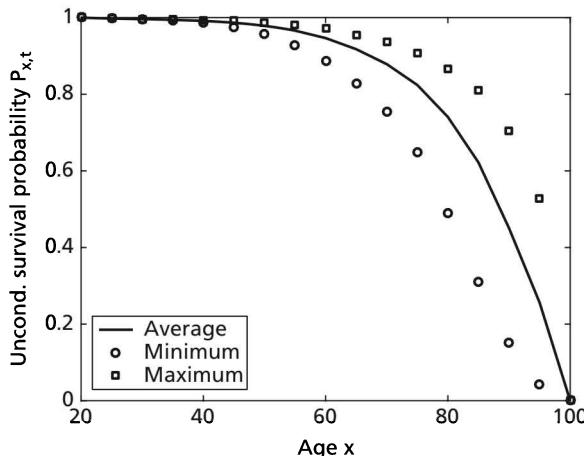


Figure 4.11 Simulated unconditional survival probabilities at age 20–24

the largest life expectancy L_1^k at age 20–24 out of the K simulated mortality tables. The probabilities are generated using parameter values $c = -1$, $\sigma_\epsilon^2 = 2$, and $K = 100,000$. A lower c generally increases remaining life expectancy of all individuals, while with a higher σ_ϵ^2 the dispersion of life expectancies between different simulated paths k rises. Overall, the K realization of mortality tables from the MC simulation of the Lee-Carter model generate life expectancies at age 20–24 between approximately 74 and 88 years.

4.4.2 PRICING AND RISK ANALYSIS OF INSURANCE PRODUCTS

The products life insurers sells to their clients can broadly be classified into two categories, *annuity (or pension) insurances* (AI) and *life insurances* (LI). These two products are essentially counterparts.

Annuity insurances An annuity insurance is a contract in which the insurer is obliged to pay the client some benefit if the client *survives until a certain age*. We call this age the age of maturity M . If the insured dies before maturity, the insurer will face no payments at all. Once the client has reached the defined maturity age, the insurer has to pay benefits either for a fixed number of contracted years (a so-called *term annuity*) or until the insured has eventually passed away (a so-called *whole life annuity*). In the case of a term annuity, the obligation of the insurer to pay benefits also ends when the client dies within the contracted term period. An annuity insurance is therefore specified by three characteristics: the client's age x at which the contract was made, a deferral period m after which the insurer will have to start paying benefits, and a contract duration d that determines the number of periods for which benefits are payed. The age of maturity

consequently is equal to $M = x + m$. For simplicity, we assume that all contracts are signed in the base year $t = 0$ and bought with a single premium. Given a specific projection of mortality rates $q_{x,t}^k$ as well as an interest rate r to discount future payments, we can compute the expected present value of benefit payments for an annuity insurance issued at $t = 0$ that pays €1 in benefits as

$$A_{(x):m:d}^k = \sum_{t=m}^{m+d-1} \frac{P_{x,t}^k}{(1+r)^t}. \quad (4.21)$$

Note that in our setup an agent aged x in the base year can live for at most $X - x$ more periods before he dies with certainty. Hence, whenever we set $m + d - 1 = X - x$ or $d = X - x - m + 1$, the annuity contract is equivalent to a whole life annuity. There are two more special cases of annuity contracts: (i) in an *immediate annuity* we have $m = 0$, so that the client will immediately receive payments from the insurer once the contract is signed; (ii) a *pure endowment insurance* has a contracted number of years of $d = 1$, meaning that the insured receives a one-time payment if he survives for m periods.

Life insurances In contrast, a *life insurance* offers a payment *in case the client dies* and therefore constitutes the counterpart to an annuity insurance. Again such a contract is specified by the age of the insured x on the date of issuance, a deferral period m , and a contract duration d . However, now the insurer will only have to pay benefits if the client dies between the periods m and $m + d - 1$. Once the terminal period has passed and the insured is still living, the insurer will have incurred no costs, which makes $M = x + m + d - 1$ the age of maturity of a life insurance contract. In the same way as before, we can write the expected present value of benefit payments for a life insurance issued at $t = 0$ that pays €1 in benefits as

$$L_{(x):m:d}^k = \sum_{t=m}^{m+d-1} \frac{P_{x,t} \cdot q_{x+t,t}}{(1+r)^{t+1}}. \quad (4.22)$$

Note that in contrast to annuity insurance benefits, we assume life insurance benefits will be paid at the end of a given period, which is why we have to discount them with a factor of $(1+r)^{t+1}$. Again one distinguishes a *term life* and a *whole life insurance*. With the former the age of maturity is explicitly specified while with the latter the age of maturity is the age at death of the insured.⁸

⁸ In practice we also observe combinations of annuity and life insurance products. A so-called *endowment insurance* for example combines a term life insurance and a pure endowment insurance, so that benefits are due either in case the insured dies before the age of maturity or the insured receives a payment when reaching the maturity age.

Program 4.9.a Expected value of benefits of AI and LI

```

! expected present value of annuity insurance benefits
function AI(ix, im, id, ik)
[.....]
AI = 0d0
do it = im, im+id-1
  AI = AI + PP(ix, it, ik)/(1+r)**it
enddo

end function AI

! expected present value of life insurance benefits
function LI(ix, im, id, ik)
[.....]
LI = 0d0
do it = im, im+id-1
  LI = LI + PP(ix, it, ik)*q(ix+it, it, ik)/(1+r)**(it+1)
enddo

end function LI

```

Table 4.13 Policy characteristics of mortality portfolio ($r = 0.16$, $\xi_1 = \xi_2 = 0.05$)

| no. Policy | Insurance type | Age at issue x | Deferral periods m | Contract duration d | Periodical benefit B_i (in €) | Market price p_i^m (in €) | Weight ω_i (in %) |
|-----------------------|-------------------|------------------------|----------------------------|-----------------------------|------------------------------------------|--------------------------------------|--------------------------------|
| 1 | AI | 9 | 1 | 7 | 50,000 | 2.87 | 18.5 |
| 2 | AI | 10 | 0 | 7 | 50,000 | 3.48 | 22.4 |
| 3 | LI | 6 | 0 | 7 | 500,000 | 0.17 | 10.7 |
| 4 | LI | 8 | 0 | 7 | 500,000 | 0.34 | 22.0 |
| 5 | LI | 8 | 0 | 9 | 500,000 | 0.41 | 26.4 |
| Total balance (in €m) | | | | | 776.2 | | |

Program 4.9.a shows how we can implement the pricing equations (4.21) and (4.22) in two separate functions. These pricing functions take as input the contract parameters, meaning the age of the client on the date of issue ix , the deferral period im , and the contract length id . Furthermore, we submit the index ik to the functions, which indicates the mortality table with which the price of the insurance should be calculated. Recall that for a value of ik between 1 and K , the function uses the MC simulated mortality tables. If ik is equal to zero, then the mortality probabilities are equal to the data $q_{x,0}$ in our baseline year.

Calculating premiums of insurance contracts We now want to study a specific portfolio of life insurance contracts. Table 4.13 shows the main characteristics of a mortality portfolio that consists of $N =$ five policies. There are two annuity insurances and three life insurances in the portfolio of the insurer. We assume that each of the different insurances

Module 4.10m Declaring a data type for insurance contracts

```

! define an insurance type
type :: insurance_type
    character(len=2) :: itype
    integer :: ix, im, id
    real*8 :: BB, n_ins
end type

! declare the insurance variable
type(insurance_type) :: ins(NN)

```

has been issued to $n_i = 1,000$ clients. The ages of the insured in the base year vary between 45 ($x = 8$) and 69 years ($x = 10$). The first contract is a whole life annuity with a payout benefit of €50,000 per period paid one period from the issuing date. The second contract is an immediate whole life annuity with the same payout benefit and contract duration. The age of maturity of both contracts roughly reflects the retirement age 65–69. Contracts three and four are term life insurances with different ages of maturity owing to differences in the age x at which the contract was issued. The last policy is a whole life insurance. The payout benefit of all life insurances is €500,000.

We again use a derived data type to declare an insurance contract in our program, see Module 4.10m. The data type is called `insurance_type` and is defined by a character `itype` that either equals '`AI`' for annuity insurance or '`LI`' for life insurance. The variables `ix`, `im`, and `id` define the relevant ages and period lengths. Finally the variables `BB` and `n_ins` determine the size of benefits that are payed by the insurer to the client in the insurance case as well as the number of insured clients under the respective contract. Having declared the data type, we define an array of type `insurance_type` of length `NN = 5` that hosts all our different insurance policies. The different insurance policies are initialized in the subroutine `init_data`, which is not shown here.

Given the characteristics of a certain insurance policy, we need to determine the premium a client has to pay in the base year $t = 0$ in order to purchase such an insurance contract. We denote the market price of an insurance policy by p_i^m with $i = 1, \dots, N$ and assume that the insurer calculates insurance premiums bases on the cross-sectional mortality rates $q_{x,0}$ observed in the base year. In order to account for changes in future life expectancy, the insurer charges a loading factor of ξ_1 for each annuity insurance and a discount factor ξ_2 for each life insurance policy. Hence the market prices of the two types of insurance contracts are

$$p_i^m = \begin{cases} A_{(x):m:d}^0(1 + \xi_1) & \text{if } i \text{ is an AI and} \\ L_{(x):m:d}^0(1 - \xi_2) & \text{if } i \text{ is a LI.} \end{cases}$$

Recall that the index $k = 0$ refers to the mortality table generate directly out of the mortality rates in the base year, see above.

Program 4.9.b Calculating premiums of insurance products

```

! premium calculation for different insurance polices
do ii = 1, NN
    ix = ins(ii)%ix
    im = ins(ii)%im
    id = ins(ii)%id

    if(ins(ii)%itype == 'AI')then
        p_m(ii) = AI(ix, im, id, 0)*(1d0 + xi_1)
    else
        p_m(ii) = LI(ix, im, id, 0)*(1d0 - xi_2)
    endif
enddo

! get the total sum of premia payed for insuance contracts
sum_premia = sum(ins(:)%n_ins*p_m*ins(:)%BB)

! determine shares in the insurer's portfolio
omega = ins(:)%n_ins*p_m*ins(:)%BB/sum_premia

```

Program 4.9.b shows how we calculate the premium of an insurance contract. We therefore iterate over all different contracts *ii* and retrieve the conditions of the respective contract from the array of deferred data types *ins*. If the insurance is an annuity insurance, meaning that *ins(ii)%itype* is equal to 'AI', we use the pricing function *AI* with an index of *ik* = 0 and add the loading factor *xi_1*. Otherwise we apply the life insurance pricing function and multiply it with the discount factor. The respective market prices are reported in the seventh column of Table 4.12. We assume an annual interest rate of 3 per cent⁹ and set the loading and discount factors both at 5 per cent. Buying a whole life annuity with identical duration in period 9 instead of period 10 reduces the cost by roughly 20 per cent. The price of a term life insurance almost triples when the age of maturity rises from 80 to 100. Given the respective market prices of an insurance contract, the actual premium a client has to pay for the contract in the base year is the market price p_m^i multiplied by the face value of the contract B_i , meaning the amount of benefits the insured expects to receive, see the sixth column of Table 4.13. Consequently, as the insurer issued $n_i = 1,000$ contracts of each insurance, the total amount of premiums can be calculated from $\sum_{i=1}^N n_i p_m^i B_i$. This total portfolio balance roughly amounts to €776m. Last but not least, the weights ω_i of each policy in the total portfolio of the insurer are the fraction of total premia received from one contract $n_i p_m^i B_i$ and the total portfolio balance. The last column of Table 4.13 shows these weights.

The risk structure of the portfolio While the insurer receives premiums from clients in the base year, the amount of benefits he actually has to pay out over the full duration of the different contracts is highly uncertain and evolves with future mortality rates. In the

⁹ Which equals a five-year interest rate of $1.03^5 - 1 = 0.16$.

following, we want to take a deeper look into the risk structure of the portfolio. A good measure to evaluate the profitability of an insurance contract is the benefit–price ratio. Under a certain simulated mortality table $q_{x,t}^k$, we can calculate the benefit–price ratio as

$$\ell_i^k = \begin{cases} \frac{A_{(x):m:d}^k}{p_i^m} & \text{if } i \text{ is an AI and} \\ \frac{L_{(x):m:d}^k}{p_i^m} & \text{if } i \text{ is a LI.} \end{cases}$$

If the benefit–price ratio is smaller than one, the insurer will make some profit on the contract as the premium he received is larger than the present value of payed out benefits. If the ratio is greater than 1, the insurer makes losses. Program 4.9.c shows how we can simulate the benefits price ratios ℓ_i^k for each insurance contract i under the different scenarios k of simulated mortality tables. The procedure is almost identical to calculating the insurance premia in Program 4.9.b. Having determined the benefit–price ratios of each insurance contract, we can also determine the benefit–price ratio of the total portfolio, by weighting the values for each insurance product by their respective portfolio weight ω_i .

In Program 4.9.d we can eventually determine some risk measures of the insurer's portfolio. The expected values and the variance-covariance matrix of benefit–price ratios of each portfolio position can be computed from

$$\mu_i = \frac{1}{K} \sum_{k=1}^K \ell_i^k \quad \text{and} \quad \sigma_{ij} = \frac{1}{K-1} \sum_{k=1}^K (\ell_i^k - \mu_i)(\ell_j^k - \mu_j) \quad \forall i, j = 1, \dots, N.$$

Program 4.9.c Simulating benefit–price ratios

```

! calculate benefit-premium ratios for all simulations
do ii = 1, NN
    ix = ins(ii)%ix
    im = ins(ii)%im
    id = ins(ii)%id

    if(ins(ii)%itype == 'AI') then
        do ik = 1, KK
            l(ii, ik) = AI(ix, im, id, ik)/p_m(ii)
        enddo
    else
        do ik = 1, KK
            l(ii, ik) = LI(ix, im, id, ik)/p_m(ii)
        enddo
    endif
enddo

! sum up portfolio weighted benefit-premium ratios
do ik = 1, KK
    l_p(ik) = sum(omega*l(1:NN, ik))
enddo

```

Program 4.9.d Risk measures of the insurer's portfolio

```

! determine mean of benefit-premium ratios
do ii = 1, NN
    mu(ii) = sum(l(ii,:))/dbl(e(KK)
enddo
mu_p = sum(omega*mu)

! compute variance-covariance matrix of benefit-premium ratios
sig_p = 0d0
do ii = 1, NN
    do ij = 1, NN
        sig(ii, ij) = sum((l(ii,:)-mu(ii))*(l(ij,:)-mu(ij))) &
                        /dbl(e(KK-1)
        sig_p = sig_p + sig(ii,ij)*omega(ii)*omega(ij)
    enddo
enddo

! compute VaR and CVaR for portfolio
call sort(l_p)
ii_VaR = ceiling(0.95d0*KK)
VaR    = l_p(ii_VaR)
CVaR   = sum(l_p(ii_VaR:KK)) / dbl(e(KK-ii_VaR+1))

```

The corresponding mean and variance of the total portfolio can then be immediately derived using the formulas in (4.1) and (4.2). In our current example we get

$$\mu = \begin{pmatrix} 0.988 \\ 0.977 \\ 0.888 \\ 0.933 \\ 1.013 \end{pmatrix} \text{ and } \Sigma = \begin{pmatrix} 0.001 & 0.000 & -0.002 & -0.002 & -0.001 \\ 0.000 & 0.000 & -0.002 & -0.001 & 0.000 \\ -0.002 & -0.002 & 0.009 & 0.007 & 0.002 \\ -0.002 & -0.001 & 0.007 & 0.005 & 0.002 \\ -0.001 & 0.000 & 0.002 & 0.002 & 0.001 \end{pmatrix},$$

which leads to a total portfolio mean and variance of

$$\mu_p = 0.969 \quad \text{and} \quad \sigma_p^2 = 0.001.$$

It turns out that in expectation the insurance company will make profits on the first four contracts, but has to expect a loss of 1.3 per cent of the face value, i.e. 6500 € from each contract of type $i = 5$. Note that the third policy has the lowest mean benefit-price ratio, but at the same time the diagonal value of the covariance matrix shows that this policy also has the highest risk involved. The covariance matrix clearly indicates a negative correlation between annuities and life insurances, again showing that the two types of policies are counterparts. Finally, we can calculate the VaR and the CVaR of the insurer's portfolio at a 95 per cent confidence level by sorting the simulated benefit-price ratios in ascending order like we did in Section 4.3.2. With a probability of 95 per cent, the loss of the insurer will not exceed a value of 1.1 per cent of the portfolio balance, or

approximately €8.5m. On the other hand, if the total loss exceeds the VaR₉₅, the insurer has to expect an average loss of 2.3 per cent of the portfolio value, or almost €18m.

4.4.3 OPTIMIZATION OF A MORTALITY PORTFOLIO

As we have just seen, the profitabilities of annuity and life insurance contracts as measured by the benefit–price ratio are negatively correlated. This negative correlation can serve as a hedging strategy for the insurer. In order to derive an *optimal portfolio structure* for the insurance company, we again have to specify risk preferences. We use the same preferences as in the first section of this chapter

$$U(\mu_p, \sigma_p^2) = \mu_p + \frac{\gamma}{2} \sigma_p^2,$$

where γ again defines the risk aversion of the insurer. Recall that given a certain portfolio composition ω , we can determine the mean and variance of the portfolio from equations (4.1) and (4.2). Hence, we can write the portfolio optimization problem of the insurer as

$$\min_{\omega} \quad \omega^T \mu + \frac{\gamma}{2} \omega^T \Sigma \omega + \lambda(1 - \omega^T I) + \eta^T \omega, \quad (4.23)$$

where λ and η_i are $N + 1$ Lagrangian multipliers associated to the constraints

$$\sum_{i=1}^N \omega_i = 1 \quad \text{and} \quad \omega_i \geq 0.$$

The first-order conditions of problem (4.23) then immediately read

$$\begin{aligned} -\mu_i - \gamma \sum_{j=1}^N \omega_j \sigma_{ji} - \lambda + \eta_i &= 0, \quad i = 1, \dots, N \\ \sum_i \omega_i - 1 &= 0 \\ \eta_i \omega_i &= 0, \quad i = 1, \dots, N. \end{aligned}$$

The last N conditions are again the Kuhn-Tucker conditions. We use the toolbox function `fzero` to solve this equation system, which works in exactly the same way as in Program 4.2. The equation system is implemented in the function `focs`, which is shown in Module 4.9m.a. The function takes as input an array `x_in` of size $2*NN+1$. The first NN entries represent the portfolio shares ω , the entry $NN+1$ the Lagrangian multiplier λ and the entries $NN+2$ to $2*NN+1$ the Lagrangian multipliers on the Kuhn-Tucker conditions.

Module 4.9m.a Function containing nonlinear equation system

```

function focs(x_in)
    [.....]
    ! copy portfolio weights
    omega = x_in(1:NN)

    ! copy Lagrangian multipliers
    lamda = x_in(NN+1)
    eta    = x_in(NN+2:2*NN+1)

    ! set up first-order conditions
    focs(1:NN) = - mu - gamma*matmul(sig, omega) - lamda + eta
    focs(NN+1) = sum(omega) - 1d0
    focs(NN+2:2*NN+1) = omega + eta - sqrt(omega**2 + eta**2)

end function focs

```

Table 4.14 Sample statistics for benefit–price ratios of different mortality portfolios

| Portfolio | Weights | Mean | Variance | VaR ₉₅ | CVaR ₉₅ |
|---------------|---------------------|-------|----------|-------------------|--------------------|
| Initial | [0.185, ..., 0.264] | 0.969 | 0.001 | 1.011 | 1.023 |
| Optimized | | | | | |
| $\gamma = 10$ | [0,0.168,0.832,0,0] | 0.904 | 0.006 | 1.038 | 1.079 |
| $\gamma = 15$ | [0,0.407,0.593,0,0] | 0.926 | 0.003 | 1.013 | 1.039 |
| $\gamma = 20$ | [0,0.511,0.489,0,0] | 0.934 | 0.001 | 1.001 | 1.023 |

The function then returns the first-order conditions associated with the optimal portfolio composition, the equality constraint that guarantees the portfolio weights to sum up to a total of 1 as well as the N complementarity conditions associated with the Kuhn-Tucker constraints that are implemented using the Fischer-Burmeister complementarity function.

Table 4.14 summarized our results. The first line of the table repeats the risk measures of the insurer's current portfolio, which yields an expected profit of 3.1 per cent of the total portfolio balance. However, with a probability of 5 per cent there will be losses greater than 1.1 per cent of the total balance, and the expected value of those losses is 2.3 per cent. The lower part of Table 4.14 shows the optimized portfolio weights for alternative risk preferences of the insurer. Regardless of the insurer's risk aversion, the optimal portfolio comprises both an annuity and a life insurance policy. These two counterpart policies guarantee that the insurer is optimally hedged against fluctuations in life expectancy. In fact, the insurer should only sell annuity contracts of type 2 and life insurance contracts of type 3. All other policies should not be sold by the company. There is also a clear trade-off between risk and return. For a low risk aversion of $\gamma = 10$, the optimal portfolio mainly consists of the term life insurance which matures at age 80-84, since this policy offers the highest expected benefit–price ratio. However, this policy is also associated with the highest risk. Hence, as risk aversion increases, the optimal portfolio structure

shifts towards the immediate whole life annuity contract. Expected profits of the resulting portfolio therefore decline but all risk measures improve. The last line in Table 4.14 shows that, compared to the insurer's original portfolio, a (close to) balanced portfolio of the immediate whole life annuity and the term life insurance yields a much higher expected profit at the same level of variance and CVaR₉₅.

This short treatment should be enough to give a basic idea about the principles and mechanics of portfolio optimization and risk management. We can start extending the baseline models presented in this chapter and use alternative assumptions. Some of this is done in the exercises section. Other extensions can be found in the suggested literature.

4.5 Appendix

Log-normal distribution The random variable $\log S$ is normally distributed with mean μ and variance σ^2 , i.e. $\log S \sim N(\mu, \sigma^2)$. Then $S = e^{\log S}$ follows the log-normal distribution with mean and variance

$$E[S] = e^{\mu + \sigma^2/2} \quad \text{and} \quad \text{Var}[S] = (e^{\sigma^2} - 1) e^{2\mu + \sigma^2}.$$

The probability density function (pdf) for S is

$$f(x) = \frac{1}{\sigma x \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log x - \mu}{\sigma}\right)^2}$$

while the cumulative density function (cdf) is

$$F(x) = \Phi\left(\frac{\log x - \mu}{\sigma}\right)$$

where $\Phi(y)$ is the standard normal cdf.

Define $L_S(K)$ the expected value of the censored stock price S , where all values of S below K are censored at zero, so that

$$L_S(K) = E[S|S > K] \times (1 - F(K)) + 0 \times F(K) = \int_K^\infty x f(x) dx.$$

Using the definition of the pdf from above we have

$$L_S(K) = \int_K^\infty \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\log x - \mu}{\sigma}\right)^2} dx = \int_{\log K}^\infty \frac{e^y}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y - \mu}{\sigma}\right)^2} dy.$$

where we have defined $y = \log x$ so that $x = e^y$ and $dx = e^y dy$. Combining terms the exponent can be rearranged

$$\begin{aligned} y - \frac{1}{2} \cdot \left(\frac{y - \mu}{\sigma} \right)^2 &= -\frac{1}{2} \frac{y^2 - 2\mu y + \mu^2 - 2\sigma^2 y}{\sigma^2} \\ &= -\frac{1}{2} \cdot \frac{y^2 - 2y(\mu + \sigma^2) + (\mu + \sigma^2)^2 - (\mu + \sigma^2)^2 + \mu^2}{\sigma^2} \\ &= -\frac{1}{2} \cdot \left(\frac{y - (\mu + \sigma^2)}{\sigma} \right)^2 + \mu + \frac{\sigma^2}{2} \end{aligned}$$

Consequently, we get

$$\begin{aligned} L_S(K) &= e^{\mu + \frac{\sigma^2}{2}} \int_{\log K}^{\infty} \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left(\frac{y - (\mu + \sigma^2)}{\sigma} \right)^2} dy \\ &= e^{\mu + \frac{\sigma^2}{2}} \left(1 - \Phi \left(\frac{\log K - \mu - \sigma^2}{\sigma} \right) \right) = e^{\mu + \frac{\sigma^2}{2}} \Phi \left(\frac{-\log K + \mu + \sigma^2}{\sigma} \right) \end{aligned}$$

where we have used the symmetry of the standard normal distribution $1 - \Phi(x) = \Phi(-x)$.

Parametrization of the binomial model Next we explain how to select parameters of the binomial model such that the model yields the same mean and variance as the continuous time model. Therefore, we have

$$E[S_{t+1n}] = S_{tn} e^{r\Delta t} = (pu + (1-p)d)S_{tn}$$

so that

$$p = \frac{e^{r\Delta t} - d}{u - d}.$$

In addition require

$$\begin{aligned} \text{Var}[S_{t+1n}] &= S_{tn}^2 e^{2r\Delta t} (e^{\sigma^2 \Delta t} - 1) \\ &= p(uS_{tn})^2 + (1-p)(dS_{tn})^2 - (puS_{tn} + (1-p)dS_{tn})^2 \end{aligned}$$

so that

$$e^{(2r+\sigma^2)\Delta t} - e^{2r\Delta t} = pu^2 + (1-p)d^2 - (pu + (1-p)d)^2.$$

Using the above condition for p , we immediately get

$$\begin{aligned} e^{(2r+\sigma^2)\Delta t} &= pu^2 + (1-p)d^2 \\ &= \frac{(e^{r\Delta t} - d)u^2 + (u - e^{r\Delta t})d^2}{u - d} \\ &= \frac{(u^2 - d^2)e^{r\Delta t} - ud(u - d)}{u - d} \\ &= (u + d)e^{r\Delta t} - ud \end{aligned}$$

We now set $ud = 1$ in order to get a symmetric binomial tree. This yields

$$u + d = e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t}$$

so that after using again $ud = 1$ and rearranging

$$u^2 - 0.5(e^{-r\Delta t} + e^{(r+\sigma^2)\Delta t})u + 0.5 = 0$$

which finally yields the parameters defined in (4.14).

4.6 Further reading

There are numerous books that use numerical techniques to solve problems in quantitative finance. Benninga (2014) is a standard text book in the area of portfolio optimization and option pricing that uses Excel to implement practical problems. Alternatively, Arratia (2014), Capinski and Zastawniak (2012) and Brandimarte (2002) present similar material, but provide programming examples using the software R, C++ and Matlab, respectively. Other books include specific easy-to-use software designed for special purposes. For example, Hull (2009) provides DerivaGem for valuing a wide range of options. For a comprehensive introduction into credit risk modelling, the interested reader is referred to one of the standard textbooks, including Ramaswamy (2004) which uses the commercial CreditManager software. De Waegenaere, Melenberg and Stevens (2010) survey the approach and the recent literature on mortality risk management. The quantitative discussion of Section 4.4 is mostly based on Cox et al. (2013). Gatzert and Wesker (2014) provide a more sophisticated dynamic analysis over the whole contract duration that also accounts for adverse selection as well as stochastic interest rates and compares different risk management instruments and risk measures.

4.7 Exercises

- 4.1. Write a program that uses the minimization subroutine `fminsearch` to compute the optimal portfolio weights without and with a risk-free asset. Use the stock price data from Table 4.1, assume a return of 4 per cent per year for the risk-free asset and suppose a risk-aversion parameter $\gamma = 10$ for the investor.
- 4.2. Write a program that allows to calculate the implied standard deviation σ of the stock's return given the price of a European call option c^E , the current stock price S_0 , the time to expiration ΔT and the interest rate r . Use the values from Table 4.4 for an explicit solution.
- 4.3. Write a program that uses Monte Carlo techniques to compute the price of a European call option and a European put option. Discuss how many random draws are necessary until the results converge to the analytical solutions for the values reported in Table 4.4?
- 4.4. Similarly to Asian options (see Section 4.2.4), Barrier options are path-dependent so that there does not exist a closed-form solution for their price. Monte Carlo methods are hence an easy and robust way to price them. The payoff of Barrier options at the expiry date ΔT is either equal to that of a standard European option or zero, conditional on whether a certain barrier level of the underlying's price is reached during the lifetime of the option. Barrier options are of knock-in type, if reaching the barrier means that the payoff becomes equal to a European option and the option expires worthless otherwise. A knock-out barrier option is equal to a usual European option as long as the barrier is not hit, but becomes worthless when the barrier level is reached.
 - a) Write a program with a Monte Carlo simulation to value a Down-and-in Barrier option. In this context "Down" refers to the position of the barrier relative to the initial price of the underlying stock at the date of issue and "in" to the knock-in feature of this barrier. Use the parameter-specifications from Table 4.4 and consider a barrier of $H = 24$.
 - b) Compute the price of the Barrier option from (a) for both cases, if the Barrier is only active the first 10 days or the last 10 days of the option's lifetime. Which of these two alternatives do you expect to be more expensive?
- 4.5. A special type of Barrier options are Binary options. Continue with the specifications from Table 4.4 and assume that the holder of a Binary option receives either a fixed amount $q = 30$ if the price of the underlying finishes above the strike price, but has a payoff of $q = 0$ if the price of the underlying is below the strike price at maturity. Derive a closed form solution for these options similar to the Black-Scholes formula.

- 4.6. Lookback options are another type of path-dependent options. Their payoff depends on the maximum or minimum of the underlying's price realized during an option's lifetime. For so-called Floating Strike Lookback options, the strike price K at maturity is equal to the minimum price of the underlying realized during the lifetime of the option for a call and vice versa for a put. In the case of a Fixed Strike Lookback option, however, the spot price at maturity is replaced by the maximum price realized during the life of the option for a call and by the minimum price for a put.
- a) Write a program that uses Monte Carlo technique to price Floating Strike Lookback options with the parameter-specifications from Table 4.4.
 - b) Write a program that uses Monte Carlo technique to price Fixed Strike Lookback options with the parameter-specifications from Table 4.4.
- 4.7. Chooser options allow their holder to decide at a certain date ΔT_1 prior to maturity ΔT_2 , whether the option should be a call or a put option, meaning that at date ΔT_1 the option can be converted into either a call or a put option. Consequently, the Chooser option is worth the maximum of the call and the put option at date ΔT_1 . Given the values of the call and put options $c^E(S_{\Delta T_1}, K, \Delta T_2 - \Delta T_1)$ and $p^E(S_{\Delta T_1}, K, \Delta T_2 - \Delta T_1)$ at date ΔT_1 , respectively, the price of the Chooser option at date $t = 0$ is then given by
- $$p^{CH} = e^{-r\Delta T_1} \max[c^E(S_{\Delta T_1}, K, \Delta T_2 - \Delta T_1), p^E(S_{\Delta T_1}, K, \Delta T_2 - \Delta T_1)].$$
- a) Note from Section 4.2. that the relationship $p^E = c^E + Ke^{-r\Delta T} - S_0$ ("put-call-parity") has to hold true for European options. Apply it to derive a closed form solution for Chooser options from the Black-Scholes formula. Use the values from Table 4.4 for an explicit solution and assume a decision date of 31 business days into the option's runtime ($\Delta T_1 = \frac{31}{250}$).
 - b) Write a program that uses Monte Carlo techniques to simulate the price $S_{\Delta T_1}$ and derive the prices for call and put options at ΔT_1 . Then compute the price of the Chooser option in $t = 0$ numerically.
- 4.8. A Bermuda option is a special form of an American option. It can no longer be exercised at any arbitrary point of time, but only at a certain number of pre-specified dates prior to maturity. Use the values from Table 4.4, but assume a spot price of $S_0 = 24$ for the underlying and a maturity of 62 business days. Compute the fair value of Bermuda call and put options when it is only possible to exercise the option after 20, 40 or 62 business days. Compare the results to the respective prices of European and American options.
- 4.9. The sensitivity of an option price with respect to the spot price S_0 at the date of issue, its volatility σ , the risk-free interest rate r and the time to maturity ΔT can be characterized by its partial derivatives (so called "Greeks") in the respective dimension, i.e.

$$\begin{aligned}\Delta_c &= \frac{\partial c}{\partial S}, & \Delta_p &= \frac{\partial p}{\partial S}, & \Gamma &= \frac{\partial^2 c}{\partial S^2} = \frac{\partial^2 p}{\partial S^2}, & \mathcal{V} &= \frac{\partial c}{\partial \sigma} = \frac{\partial p}{\partial \sigma} \\ \rho_c &= \frac{\partial c}{\partial r}, & \rho_p &= \frac{\partial p}{\partial r}, & \theta_c &= \frac{\partial c}{\partial \Delta T}, & \theta_p &= \frac{\partial p}{\partial \Delta T}.\end{aligned}$$

Write a program to approximate these partial derivatives for European options numerically using a difference quotient. Use the values from Table 4.4 and plot the results for different spot prices S_0 of the underlying at the date of issue. Explain and discuss your results.

- 4.10. Options are often part of more complex trading strategies to profit from a certain expected developments on the stock market. Assume a risk-free interest rate $r = 0.1$, a stock price $S_0 = 100$ at the date of issuance, a volatility of $\sigma = 0.4$, and a maturity of 21 business days for the underlying (250 working days per year). Compute the price of the following trading strategies, plot the respective payoff diagram in the interval $S_{\Delta T} \in [90; 110]$ and interpret the results.
- (Straddle) Buy a European call option and a European put option with strike price $K = 100$ on the same underlying.
 - (Strangle) Buy a European call option with strike price $K_1 = 105$ and a European put option with strike price $K_2 = 95$.
 - (Butterfly Spread) Buy a European call option with strike price $K_1 = 95$, a European put option with strike price $K_2 = 105$, and sell a European call option as well as a European put option with strike price $K = \frac{1}{2}(K_1 + K_2)$.
- 4.11. In practical insurance risk management, skewness and excess of the loss distribution are used to better describe the shape of the distribution function. With respect to the benefit–price ratios of different insurance policies the two parameters are computed from

$$skew_i = \frac{1}{K} \sum_{k=1}^K (l_i^k - \mu_i)^3 / \sigma_{ii}^3 \quad \text{and} \quad excess_i = \frac{1}{K} \sum_{k=1}^K (l_i^k - \mu_i)^4 / \sigma_{ii}^4 - 3$$

If the distribution is symmetric around the mean then the skewness is zero. Negative values for the skewness indicate data where the left tail is long relative to the right tail and positive values for the skewness indicate data for which it is the other way round. The excess measures the ‘peakedness’ or ‘flatness’ of a distribution. The excess for a standard normal distribution is 0. Higher numbers indicate that the data is more concentrated towards the mean.

- Compute both parameters for the distribution of the benefit–price ratios of each policy and the different portfolios in Table 4.14. What do the results indicate in terms of risk exposure?

- b) Plot the set of minimum variance portfolios for a given level of return as we did in the portfolio choice example in Program 4.1.a. Identify the location of the initial mortality portfolio.
 - c) Compute the weights for the minimum variance portfolio when we keep the mean of the initial portfolio return constant. Compute the resulting variance, skewness, excess, VaR_{95} , and $CVaR_{95}$.
 - d) Compute the weights for the global minimum variance portfolio. Compute the resulting mean, variance, skewness, excess, VaR_{95} , and $CVaR_{95}$. What does the result imply for the insurance company? Is it reasonable to use this method for the computation of optimal mortality portfolio weights? Justify!
- 4.12. In order to eliminate individual mortality risk, we assumed that the insurance sells each contract to $n_i = 1,000$ clients. Check whether this number of contracts is sufficient for life insurance policy number 5. Proceed as follows:
- a) Initialize $q_{x,0}$, the maximum number of insured people $N = 1,000$, $x = 8$, $d = 9$, $\xi_2 = 0.05$, as well as $r_f = 0.16$.
 - b) Compute interval borders k_j on the unit interval which reflect the death probability distribution over periods of life remaining using base year probabilities from

$$k_j = \begin{cases} 0 & \text{if } j = 0 \\ k_{j-1} + q_{x+j-1,0} P_{x,j-1} & \text{otherwise} \\ 1 & \text{if } j = T - x + 1 \end{cases}$$

- c) Given a specific number n_i of insured clients, simulate the period of death for each insured by drawing a random number in the unit interval, which you partitioned according to the different k_j values. Compute the resulting present value from a €1 benefit for each simulated client. Take the average of all clients and apply the discount factor ξ_2 . Print the average payments for the $n_i = 1, \dots, N$ insured people. If your computations are correct, this average should converge to the respective market price reported in Table 4.13. How many insured clients N are actually necessary to make this convergence happen?

5 The life-cycle model and intertemporal choice

The discussion in the Chapters 3 and 4 centred around static optimization problems. The static general equilibrium model of Chapter 3 features an exogenous capital stock and Chapter 4 discusses investment decisions with risky assets, but in a static context. In this chapter we take a first step towards the analysis of dynamic problems. We introduce the life-cycle model and analyse the intertemporal choice of consumption and individual savings. We start with discussing the most basic version of this model and then introduce labour-income uncertainty to explain different motives for saving. In later sections, we extended the model by considering alternative savings vehicles and explain portfolio choice and annuity demand. Throughout this chapter we follow a partial equilibrium approach, so that factor prices for capital and labour are specified exogenously and not determined endogenously as in Chapter 3.

5.1 Why do people save?

This section assumes that households can only save in one asset. Since we abstract from bequest motives in this chapter, households do save because they need resources to consume in old age or because they want to provide a buffer stock in case of uncertain future outcomes. The first motive is the so-called *old-age* savings motive while the second is the *precautionary* savings motive.

5.1.1 OPTIMAL SAVINGS IN A CERTAIN WORLD

In order to derive savings decisions it is assumed in the following that a household lives for three periods. In the first two periods the agent works and receives labour income w while in the last period the agent lives from his accumulated previous savings. In order to derive the optimal asset structure a_2 and a_3 (i.e. the optimal savings), the agent maximizes the utility function

$$U(c_1, c_2, c_3) = u(c_1) + \beta u(c_2) + \beta^2 u(c_3)$$

where β denotes a time discount factor and $u(c) = \frac{c^{1-\gamma}}{1-\gamma}$ describes the preference function with $\gamma \geq 0$ measuring the intertemporal elasticity of substitution.¹ Households have no assets initially (i.e. $a_1 = 0$), so that their periodic budget constraints in the three periods are

$$c_1 = w - a_2, \quad c_2 = w + Ra_2 - a_3 \quad \text{and} \quad c_3 = Ra_3, \quad (5.1)$$

where $R = 1 + r$ denotes the return on savings. After substituting the budget constraints into the utility function, the maximization problem becomes

$$\max_{a_2, a_3} U(a_2, a_3) = u(w - a_2) + \beta u(Ra_2 + w - a_3) + \beta^2 u(Ra_3). \quad (5.2)$$

Program 5.1 shows how we can solve this problem using the subroutine `fminsearch` coming from the toolbox. We thereby store our model parameters in the module `globals`. The solution to the problem is calculated using `fminsearch`. Recall that this subroutine takes a starting value `x`, a `real*8` value `fret` in which the function value in the minimum will be stored, a lower and upper bound for the optimization interval and the function that should be minimized as input variables. We define the lower bound as zero, i.e. households will not be allowed to run into debt. The upper bound is set such that assets can not exceed the maximum available resources in periods 2 and 3, respectively.

The function `utility` which is shown in Module 5.1m is the function to be minimized. It is also stored in the module `globals`. We start by copying the input vector of the function to the savings decision `a`. Note that `a` and `c` are `real*8` arrays of length 3, that also are defined in the module `globals`. Thereby `a` denotes the asset structure in the second and third period (assets in the first period have to be equal to zero)

Program 5.1 Optimal savings in a certain world

```

program household1
[.....]
! lower and upper border and initial guess
low = (/0d0, 0d0/)
up = (/w, R*w+w/)
x = up/2d0

! minimization routine
call fminsearch(x, fret, low, up, utility)

! output
write(*,'(/a/)') AGE    CONS    WAGE      INC      SAV'
write(*,'(i4,4f7.2/)') 1,c(1),w,w,a(2)
write(*,'(i4,4f7.2/)') 2,c(2),w,w+R*a(2),a(3)
write(*,'(i4,4f7.2/)') 3,c(3),0d0,R*a(3),0d0

end program

```

¹ $1/\gamma$ then is a measure for relative risk-aversion.

Module 5.1m Utility function in a certain world

```

function utility(x)

    implicit none
    real*8, intent(in) :: x(:)
    real*8 :: utility

    ! savings
    a(1) = 0d0
    a(2:3) = x

    ! consumption (insure consumption > 0)
    c(1) = w - a(2)
    c(2) = R*a(2) + w - a(3)
    c(3) = R*a(3)
    c = max(c, 1d-10)

    ! utility function
    utility = -(c(1)**egam + beta*c(2)**egam &
                 + beta**2*c(3)**egam)/egam

end function

```

and c consumption during the three periods of the life cycle. Having copied the savings decisions, we can calculate consumption using the three budget constraints in (5.1). We limit consumption to positive values. If we didn't do that, it might happen that consumption turns negative during the optimization process and we therefore run into an error. Finally, we calculate households utility multiplied by -1 and return the respective value. The parameter `egam` defines the exponent gamma, i.e. $1 - \frac{1}{\gamma}$. The main program then prints our results on the console.

The first-order conditions of the maximization problem (5.2) are

$$u'(c_1) = \beta Ru'(c_2) \quad \text{and} \quad u'(c_2) = \beta Ru'(c_3).$$

When we assume that $\beta = 1/R$ then marginal utilities and consumption have to be the same in all periods, i.e.

$$u'(c_1) = u'(c_2) = u'(c_3) \quad \Rightarrow \quad c_1 = c_2 = c_3.$$

For simplicity we assume next $w = R = 1$ so that the optimal choices are

$$c_i = \frac{2}{3}, \quad i = 1, 2, 3, \quad a_2 = \frac{1}{3} \quad \text{and} \quad a_3 = \frac{2}{3}.$$

5.1.2 UNCERTAIN LABOUR INCOME AND PRECAUTIONARY SAVINGS

When income is certain, agents only build up savings in order to smooth consumption during the years of retirement when they have no labour income. This is the so-called

old-age savings motive. If the interest rate equals the time preference rate, agents would not accumulate any savings in case they receive a certain income also in the last period of life. In reality, the income process during the life cycle is much more disrupted. People are not able to work at the end of their lives (maybe due to health problems) and they receive a highly uncertain income during their middle years. This uncertainty of income in the second period gives rise to a second savings motive, so-called *precautionary savings*. Risk averse agents will then save more in the first period in order to damp the volatility of their consumption in the second period.

We therefore assume that wages in the second period \tilde{w} are log-normally distributed with mean μ_w and variance σ_w^2 , i.e. $\tilde{w} \sim \log N(\mu_w, \sigma_w^2)$. Figure 5.1 shows densities of log-normally distributed variables for three combinations of expectation and variance. The advantage of taking a log-normal rather than a standard normal distribution is that the log-normal distribution is bounded from below by zero. Hence, wages cannot be negative. For our simulations, we choose $\mu_w = w$ which equals the certain wage in Section 5.1.1. However, we will let the variance of the distribution differ.

The maximization problem of our household now changes to

$$\max_{a_2, \tilde{a}_3} E_1 U(a_2, \tilde{a}_3) = u(w - a_2) + \beta E_1 \left[u(Ra_2 + \tilde{w} - \tilde{a}_3) + \beta u(R\tilde{a}_3) \right],$$

where E_1 denotes the expectation at the beginning of period 1 regarding earnings and consumption in the following periods. Obviously, there is only one amount of assets a_2 in the second period. Hence, with income being certain in this period, consumption and utility are deterministic and we have $c_1 = w - a_2$. In the second period, however, income is uncertain and therefore household may choose a different level of assets \tilde{a}_3 for any realization of income \tilde{w} . Hence, we are looking for a optimal function of savings $\tilde{a}_3(w)$

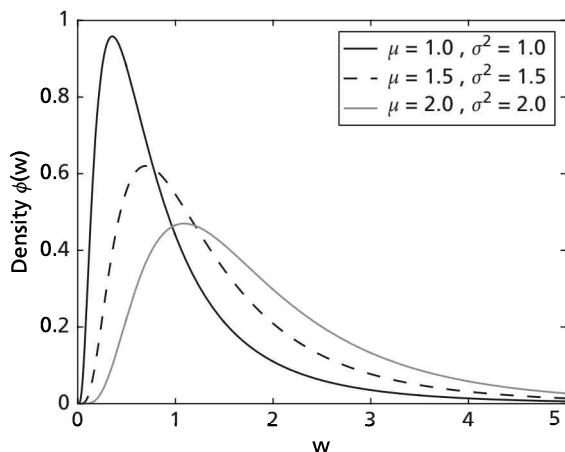


Figure 5.1 Log-normal density function

in the second period. Given the density ϕ of the distribution of w , we can write expected utility of the second and third period as

$$\begin{aligned} E_1 & \left[u(Rs_1 + \tilde{w} - \tilde{a}_3) + \beta u(R\tilde{a}_3) \right] \\ & = \int_0^\infty \phi(w) \left[u(Ra_2 + w - \tilde{a}_3(w)) + \beta u(R\tilde{a}_3(w)) \right] dw. \end{aligned}$$

This gives rise to using a Gaussian quadrature method like implemented in the subroutine `normal_discrete` in the toolbox in order to calculate expectations. However, we will come to that later.

The optimality condition for maximizing expected utility is that *expected marginal utility* be equated across periods, i.e.

$$u'(c_1) = E_1[u'(\tilde{c}_2)] = E_1[u'(c_3)] \quad (5.3)$$

where again $\beta = 1/R$ is assumed for simplicity. If marginal utility is convex (i.e. $u'' < 0, u''' > 0$) then expected marginal utility in period 2 $E_1[u'(c_2)]$ exceeds marginal utility of expected consumption $u'(E_1[c_2])$. Figure 5.2 shows an example with convex marginal utility and two possible consumption realizations in period 2 which have the same probability. Consequently, the expected consumption value $E(c_2)$ is in the middle of the two consumption realizations. Since marginal utility increases in the bad state more than it reduces in the good state, expected marginal utility of period 2 consumption rises with increasing uncertainty. Consequently, compared to the case of certainty where $c_2^1 = c_2^2 = c_2$, marginal utility has to increase in period 1 when income in period 2 is uncertain in order to satisfy the first-order condition (5.3). This means that consumption in period 1 has to decline, which implies an increase in first-period savings Δs compared to the certainty case in Section 5.1.1. The increase Δs due to uncertainty is the amount of precautionary savings the household makes. Figure 5.2 also clearly shows that precautionary savings increase when marginal utility becomes more convex

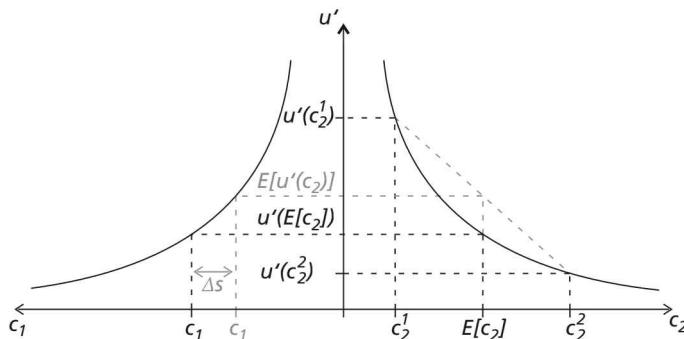


Figure 5.2 Expected marginal utility

(i.e. relative risk aversion $\frac{1}{\gamma}$ increases) or when uncertainty increases due to a higher variance σ_w^2 which increases the distance between c_2^1 and c_2^2 . With respect to savings a_3 in period 2 note that they are only stochastic at the beginning of period 1. When the actual savings decision in period 2 is made, uncertainty is gone since agents then already know the realization of their income. Consequently $E_2[u'(c_i)] = u'(c_i), i = 2, 3$.

After these theoretical considerations, we have to talk about how to solve this maximization problem. At first, we want to discretize the distribution of \tilde{w} in order to avoid calculating a whole integral. This can be done with the subroutine `log_normal_discrete` coming along with the toolbox. Like the subroutine `normal_discrete` this subroutine generates n_w quadrature nodes w and weights `weight_w`.

Having discretized the distribution of \tilde{w} , we can calculate our expected utility function in periods 2 and 3 as

$$\begin{aligned} E_1 & \left[u(Ra_2 + \tilde{w} - \tilde{a}_3) + \beta u(R\tilde{a}_3) \right] \\ & \approx \sum_{i=1}^{n_w} \omega_{w,i} \left[u(Ra_2 + w_i - a_{3,i}) + \beta u(Ra_{3,i}) \right]. \end{aligned}$$

The utility function is implemented in Module 5.2m. Note that now we also avoid the problem of calculating a whole optimal savings function $\tilde{a}_3(\tilde{w})$. Instead we only have to calculate one savings level $a_{3,i}$ for any of the possible realizations w_i of the discretized random variable \tilde{w} .

We can again use `fminsearch` to minimize the utility of the household which is calculated in the function `utility`. This function again receives an input vector `x`.

Module 5.2m Utility function with wage risk

```
function utility(x)
    [.....]
    ! savings
    a(1, :) = 0d0
    a(2, :) = x(1)
    a(3, :) = x(2:1+n_w)

    ! consumption (ensure consumption > 0)
    c(1,:) = mu_w - a(2,1)
    c(2,:) = R*a(2,1) + w(:) - a(3,:)
    c(3,:) = R*a(3,:)
    c = max(c, 1d-10)

    ! expected utility of periods 2 and 3
    utility = 0d0
    do iw = 1, n_w
        utility = utility + &
                    weight_w(iw)*(c(2,iw)**egam+beta*c(3,iw)**egam)
    enddo
    utility = -(c(1,1)**egam + beta*utility)/egam

end function
```

However, the variables in x now have changed considerably compared to Section 5.1.1. The first entry still is savings from period 1 to period 2. The next n_w entries of x however denote savings in any possible realization w_i of the wage in period 2, i.e.

$$x = (a_2, a_{3,1}, a_{3,2}, \dots, a_{3,n_w})^T.$$

Consequently, x must be of length $1+n_w$. After having copied savings into a vector of dimensions $2 \cdot n_w$, we can calculate consumption in every period in every state of the world, i.e. any realization of wages. Note that savings in the first period are certain as household earns the certain average wage μ_w , i.e. we only need to use one asset value $a(2, 1)$. For convenience, however, we store the same asset and consumption level for every entry in dimension 2 of the arrays a and c . Consumption in the second and third period is stochastic and depends on the realization of the wage and the savings levels in this state. Finally we can compute expected utility for periods 2 and 3 as shown above and add the first periods utility afterwards. Having minimized the function utility, we can compute mean and standard deviations of consumption, income, and savings in different periods. We will not show how to do this due to space restrictions. However, the respective functions and calculations can be seen from the program accompanying this book.

Table 5.1 shows the outcome of our simulations. In order to compare it with the certain case we have again set $w = \mu_w = \beta = R = 1$. In addition we assume a risk aversion of 2 and use $n_w = 5$ quadrature nodes and weights to approximate the log-normal distribution. We then vary the variance of the wage distribution holding its expectation constant. We find what we already concluded from our theoretical analysis. With increasing variance of the wage distribution consumption in period 1 decreases and savings increase. This is due to the precautionary savings motive. In addition we find that savings and consumption in the second period always have same expectation and standard deviation. This is clear from the fact that, after the second-period wage level is realized there is no more uncertainty in the decision problem. Consequently, households will react exactly in the same way as in the case of certain income, i.e. equally split resources so that $c_2 = c_3$. As with $R = 1$ $c_3 = a_3$, expectation and variance of c_2 and a_3 have to be identical. In addition, the variance of consumption in period 2 always is one fourth of the variance

Table 5.1 Precautionary savings motive when wages are risky

| σ_w^2 | c_1 | a_2 | $E[c_2]$ | $Std[c_2]$ | $E[c_3]$ | $Std[a_3]$ |
|--------------|-------|-------|----------|------------|----------|------------|
| 0.00 | 0.67 | 0.33 | 0.67 | 0.00 | 0.67 | 0.00 |
| 0.20 | 0.61 | 0.39 | 0.69 | 0.22 | 0.69 | 0.22 |
| 0.40 | 0.58 | 0.42 | 0.71 | 0.32 | 0.71 | 0.32 |
| 0.60 | 0.56 | 0.44 | 0.72 | 0.39 | 0.72 | 0.39 |
| 0.80 | 0.54 | 0.46 | 0.73 | 0.45 | 0.73 | 0.45 |
| 1.00 | 0.53 | 0.47 | 0.74 | 0.50 | 0.74 | 0.50 |

$w = \mu_w = R = \beta = 1, \gamma = 0.5, n_w = 5$.

of wages. Again this is clear if one recalls that $Ra_2 + \tilde{w} = \tilde{c}_2 + \tilde{a}_3$ and $\tilde{a}_3 = \tilde{c}_2$. We therefore have

$$\tilde{c}_2 = \frac{Rs_1 + \tilde{w}}{2}$$

from the second period's budget constraint and, as a_2 is deterministic, we obtain

$$Var(\tilde{c}_2) = \frac{Var(\tilde{w})}{4}.$$

5.1.3 UNCERTAIN CAPITAL AND LABOUR INCOME

In Section 5.1.2 we assumed capital income to be certain at a rate $R = 1$. In reality, however, we observe a lot of fluctuation in interest rates and asset returns. It therefore seems natural to also let interest rates be stochastic in our model. Saving then turns out to be a risky investment.

Beneath wages which still are log-normally distributed in the second period we therefore also assume the return on capital \tilde{R} to be log-normally distributed with mean μ_R and variance σ_R^2 and to be independent over time. Therefore R_2 and R_3 , i.e. the return on capital in periods 2 and 3, respectively, are drawn from the same distribution but are independent.

Our optimization problem now turns into

$$\max_{a_2, \tilde{a}_3} E_1 U(a_2, \tilde{a}_3) = u(w - a_2) + \beta E_1 \left[u(\tilde{R}_2 a_2 + \tilde{w} - \tilde{a}_3) + \beta u(\tilde{R}_3 \tilde{a}_3) \right].$$

Obviously, \tilde{a}_3 now not only depends on the realization of the wage \tilde{w} but also on the return on capital of the second period \tilde{R}_2 , so that we are looking for an optimal savings function $\tilde{a}_3(\tilde{w}, \tilde{R}_2)$. In order to compute our model solution we again have to discretize our shock values regarding interest rates. Beneath weights and quadrature nodes $\omega_{w,i}$ and w_i , we therefore also compute weights and nodes $\omega_{R,i}$ and R_i for the interest-rate distribution and rewrite the expectation of utility in periods 2 and 3 as

$$\begin{aligned} & E_1 \left[u(\tilde{R}_2 a_2 + \tilde{w} - \tilde{a}_3) + \beta u(\tilde{R}_3 \tilde{a}_3) \right] \\ & \approx \sum_{i=1}^{n_w} \sum_{j=1}^{n_R} \sum_{k=1}^{n_R} \omega_{w,i} \omega_{R,j} \omega_{R,k} \left[u(R_j a_2 + w_i - a_{3,i,j}) + \beta u(R_k a_{3,i,j}) \right]. \end{aligned} \quad (5.4)$$

Module 5.3m shows the utility function that is needed to solve the model with uncertain labour and capital income. The structure of the solution method is quite similar to the one

Module 5.3m Optimal savings with wage risk and risky capital returns

```

function utility(x)
    [.....]
    ! savings
    a(1,:,:)=0d0
    a(2,:,:)=x(1)
    ic=2
    do iw=1,n_w
        do ir2=1, n_R
            a(3, iw, ir2) = x(ic)
            ic=ic+1
        enddo
    enddo

    ! consumption
    c(1,:,:,:)=mu_w - a(2,1,1)
    do iw=1, n_w
        do ir2=1, n_R
            c(2,iw,ir2,:)=R(ir2)*a(2,1,1) + w(iw) - a(3,iw,ir2)
            do ir3=1, n_R
                c(3,iw,ir2,ir3) = R(ir3)*a(3,iw,ir2)
            enddo
        enddo
    enddo
    c = max(c, 1d-10)

    ! expected utility of periods 2 and 3
    utility=0d0
    do iw=1, n_w
        do ir2=1, n_R
            do ir3=1, n_R
                prob = weight_<math>w</math>(iw)*weight_<math>R</math>(ir2)*weight_<math>R</math>(ir3)
                utility = utility + prob*(c(2,iw,ir2,1)**egam + &
                    beta*c(3,iw,ir2,ir3)**egam)
            enddo
        enddo
    enddo
    utility = -(c(1,1,1,1)**egam + beta*utility)/egam
end function

```

in Section 5.1.2. In addition to weights and quadrature nodes for wages we also calculate weights weight__R and nodes R for capital returns. Note that we now have one asset level a_2 in the second period and $n_w \cdot n_R$ asset levels in the third period, as savings depend on the realization of income and capital returns. We again stack the different savings levels in the input vector x to the function `utility` in the following way:

$$x = (a_2, a_{3,1,1}, a_{3,1,2}, \dots, a_{3,1,n_R}, \dots, a_{3,n_w,n_R})^T.$$

After copying the values from x into our savings vector we can again calculate consumption for all possible realizations of wages and interest rates. Note that consumption in the first period again is the same for any shock realization as it is chosen before households know about their income and capital returns. Having calculated consumption we again can compute expected utility in periods 2 and 3 according to (5.4). We then add first period's utility and multiply the result by -1 .

Table 5.2 Optimal savings when wages and capital income are risky

| σ_w^2 | σ_R^2 | c_1 | a_2 | $E[c_2]$ | $Std[c_2]$ | $E[a_3]$ | $Std[a_3]$ | $Std[c_3]$ |
|--------------|--------------|-------|-------|----------|------------|----------|------------|------------|
| 0.40 | 0.00 | 0.58 | 0.42 | 0.71 | 0.32 | 0.71 | 0.32 | 0.32 |
| 0.40 | 0.40 | 0.56 | 0.44 | 0.66 | 0.32 | 0.78 | 0.37 | 0.66 |
| 0.40 | 0.80 | 0.55 | 0.45 | 0.62 | 0.32 | 0.83 | 0.43 | 0.94 |
| 0.40 | 1.00 | 0.54 | 0.46 | 0.61 | 0.32 | 0.86 | 0.46 | 1.07 |
| 1.00 | 1.00 | 0.48 | 0.52 | 0.63 | 0.47 | 0.89 | 0.66 | 1.29 |

$$w = \mu_w = \mu_R = \beta = 1, \gamma = 0.5, n_w = n_R = 5.$$

Table 5.2 shows some simulations for the model with wage and interest-rate risk. We start out with a situation we already reported in Table 5.1, namely $\sigma_w^2 = 0.40$. We then successively increase the variance of the capital return risk. A positive variance in capital returns adds another precautionary savings motive. Now the household not only saves for times in which wages might be low, but also for times of low returns on capital. Consequently, savings in both periods increase as σ_R^2 rises. Remember that the expectation and standard deviation of a_3 and c_3 were exactly the same in the model without capital-income risk. This is not true anymore in the current model. Since $\tilde{c}_3 = \tilde{R}\tilde{a}_3$ and $\mu_R = 1$ the expectations for savings in period 2 and consumption in period 3 are still the same. However, the variance for c_3 is much higher than the variance of a_3 , since capital income in period 3 is again uncertain.

5.2 Where do people save and invest?

Now we allow for different assets with specific return characteristics. Compared to the previous chapter the optimal mix between risk and return is now analysed in a dynamic setup and the demand for annuities to hedge against longevity risk is explicitly derived.

5.2.1 UNCERTAIN CAPITAL INCOME AND PORTFOLIO CHOICE

We now assume that the individual can split his investment between two different assets in both investment periods $t = 1, 2$. One is riskless (e.g. bonds) and yields a gross return of R_f , the other one is risky (e.g. stocks) and has a gross return of \tilde{R}_{t+1} with mean $\mu_R > R_f$ being the average return on equity. The joint distribution of labour income and capital return in period 2 is a two-dimensional log-normal distribution

$$\begin{bmatrix} \tilde{w} \\ \tilde{R}_2 \end{bmatrix} \sim \log N \left(\begin{bmatrix} \mu_w \\ \mu_R \end{bmatrix}, \begin{bmatrix} \sigma_w^2 & \rho\sigma_w\sigma_R \\ \rho\sigma_w\sigma_R & \sigma_R^2 \end{bmatrix} \right),$$

where ρ represents the correlation between the two. For simplicity, we assume the distribution of \tilde{R}_3 to have the same mean and variance as that of \tilde{R}_2 , but to be independent of both \tilde{w} and \tilde{R}_2 . Let ω_t be the share of agent's portfolio being invested in risky assets. Consequently, the return on the portfolio is given by

$$\tilde{R}_{t+1}^p = \omega_t \tilde{R}_{t+1} + (1 - \omega_t) R_f = R_f + \omega_t (\tilde{R}_{t+1} - R_f).$$

The mean portfolio return and variance are defined by

$$E_t [\tilde{R}_{t+1}^p] = R_f + \omega_t (\mu_R - R_f) \quad \text{and} \quad \text{var} [\tilde{R}_{t+1}^p] = \omega_t^2 \sigma_R^2,$$

where $\mu_R - R_f$ defines the *risk premium*.

Besides deciding about the optimal savings amount, the individual now also has to optimally allocate a proportion ω_t of his savings to risky assets in periods 1 and 2. Consequently, the periodic budget constraints change to

$$\begin{aligned} c_1 &= w - a_2 \\ \tilde{c}_2 &= \tilde{w} + [R_f + \omega_1 (\tilde{R}_2 - R_f)] a_2 - \tilde{a}_3 \\ \tilde{c}_3 &= [R_f + \tilde{\omega}_2 (\tilde{R}_3 - R_f)] \tilde{a}_3. \end{aligned} \tag{5.5}$$

Module 5.4m shows the utility function that is needed to compute the solution of the portfolio choice problem. Before minimizing this function we have to discretize the two-dimensional log-normal distribution using the subroutine `log_normal_discrete` from the toolbox. In the case of a two-dimensional distribution, however, this subroutine receives an array of integer values defining the number of points to use in each direction as first argument. We then pass an array of dimension $n_w \cdot n_R \times 2$ and one of dimension $n_w \cdot n_R$. In the former the subroutine will store the \tilde{w} and \tilde{R}_2 quadrature nodes, the latter represents the respective weights. The moments of the approximated distribution can then be calculated as shown in the full program code. Compared to the approximation of independent variables in subsection 5.1.3, we now do not get n_w quadrature nodes for \tilde{w} and n_R nodes for \tilde{R}_2 . This is due to the possible correlation of the two variables. If the variables were e.g. positively correlated, then at least in tendency we can see that the larger \tilde{w} , the larger \tilde{R}_2 . This fact can only be matched by assigning any approximation value w_i of \tilde{w} a separate set $R_{i,j}, j = 1, \dots, n_R$ of approximation values for \tilde{R}_2 . In the array `wR` we therefore will have the entries

$$wR = \begin{bmatrix} w_1 & w_1 & \dots & w_1 & w_2 & \dots & w_{n_w} & \dots & w_{n_w} \\ R_{11} & R_{12} & \dots & R_{1n_R} & R_{21} & \dots & R_{n_w 1} & \dots & R_{n_w n_R} \end{bmatrix}^T.$$

As the number of choice variables now has increased dramatically, we adjust the tolerance level of our minimization routine to 10^{-14} . This can be done using the subroutine

Module 5.4m Optimal portfolio choice with risky assets

```

function utility(x)
    [.....]
    ! savings
    a(1,:) = 0d0
    a(2,:) = x(1)
    omega(1,:) = x(2)
    ic = 3
    iwR = 1
    do iw = 1,n_w
        do ir2 = 1, n_R
            a(3, iwR) = x(ic)
            omega(2, iwR) = x(ic+1)
            ic = ic+2
            iwR = iwR+1
        enddo
    enddo

    ! consumption
    c(1,:,:,:) = mu_w - a(2,1)
    iwR = 1
    do iw = 1, n_w
        do ir2 = 1, n_R
            c(2,iwR,:,:) = (Rf+omega(1,1)*(wR(iwR,2)-Rf))*a(2,1) &
                            + wR(iwR,1) - a(3,iwR)
            do ir3 = 1, n_R
                c(3,iwR,ir3) = (Rf + omega(2,iwR)*(R(ir3)-Rf)) &
                                *a(3,iwR)
            enddo
            iwR = iwR+1
        enddo
    enddo
    c = max(c, 1d-10)
    [.....]
end function utility

```

`settol_min` from the toolbox. Before starting `fminsearch`, we still have to set lower and upper bound for minimization. Note that we now have an overall number of choice variables of $2 \cdot (1 + n_w \cdot n_R)$, i.e. a savings amount s and a portfolio composition ω for period 1 and any wage and interest-rate combination in wR of period 2. As we don't want to allow for short selling of bonds equity, we have $\omega_t \in [0, 1]$.

We can now start minimizing the function `utility` part of which is shown in Module 5.4m. After having declared variables and modules, we start with copying savings levels and portfolio compositions from the input array x . Note that we again have to stack all the decision variables into one array for the optimizer to work properly. With the savings levels and portfolio compositions we can calculate consumption in the different periods and approximation values for \tilde{w} , \tilde{R}_2 and \tilde{R}_3 using the constraints in (5.5). Obviously, there is only one consumption level in period 1 and $n_w \cdot n_R$ levels in period 2, as in the second period, the realization of \tilde{R}_3 is not yet revealed. In the third period, we finally have $n_w \cdot n_R \cdot n_R$ different levels depending on the realizations of the three random variables. Last but not least we have to compute individual utility. The probability for any of the $n_w \cdot n_R \cdot n_R$ states to occur is the product of the probability for a certain \tilde{w} and

Table 5.3 Uncertain income and individual portfolio choice

| σ_R^2 | σ_w^2 | ρ | c_1 | \bar{a}_2 | ω_1 | $E[c_2]$ | $E[a_3]$ | $E[\omega_2]$ | $E[c_3]$ |
|--------------|--------------|--------|-------|-------------|------------|----------|----------|---------------|----------|
| 0.00 | 0.00 | 0.00 | 0.67 | 0.33 | 1.00 | 0.74 | 0.67 | 1.00 | 0.81 |
| 0.50 | 0.00 | 0.00 | 0.64 | 0.36 | 1.00 | 0.72 | 0.71 | 0.33 | 0.77 |
| 0.50 | 0.50 | 0.00 | 0.57 | 0.43 | 0.75 | 0.75 | 0.74 | 0.33 | 0.80 |
| 0.50 | 0.50 | 0.50 | 0.52 | 0.48 | 0.00 | 0.75 | 0.74 | 0.33 | 0.79 |
| 0.50 | 0.50 | -0.50 | 0.59 | 0.41 | 1.00 | 0.76 | 0.75 | 0.33 | 0.80 |

$$w = \mu_w = R_f = \beta = 1, \mu_R = 1.22, \gamma = 0.5, n_w = n_R = 5.$$

\tilde{R}_2 combination and that for a realization of \tilde{R}_3 due to the independency of \tilde{R}_3 from the other random variables.

In order to analyse individual portfolio choice, we assume in the first simulation in Table 5.3 that wage income and interest rates are certain, i.e. $\sigma_R^2 = \sigma_w^2 = 0$. We set the expected return on equity to 1.22, which results in a risk premium of 0.22. If we assume one period to cover about twenty years, this figure amounts to an annual premium of about 1 per cent. This seems fairly low but reasonable with the low risk-aversion parameter of 2. While the return on bonds remains at $R = 1$, stocks now yield a certain return of $R_2 = R_3 = 1.22$, so that people will only save in stocks (i.e. $\omega_1 = \omega_2 = 1$). Since the discount rate is kept at zero and the increase in the interest rate is fairly modest, savings in both periods remain the same as in the certainty case of Section 5.1.1, but the consumption level increases with rising age in all simulations. The second simulation introduces risky stock investment but keeps wages certain. As a consequence, the portfolio composition changes noticeably. While in the first period, the agent still invests everything in equity, however in the second period, the equity share decreases to about 0.33. This is due to the fact that our household is of constant relative risk aversion. With this preference structure, the agent will always put the same fraction of the invested wealth into equity. The fraction then only depends on the risk structure and equity premium of the portfolio, but not on the invested amount. In the first period, invested wealth however not only consists of savings but also of human capital, i.e. the wage that is payed in period 2. As this wage is certain, the household already has a large riskless ‘investment’. Consequently, the agent puts the remaining invested amount, namely all the savings, into equity. The agent would even prefer to sell short bonds in order to finance further equity investment. This, however, is not possible due to the imposed short-selling constraints.² In the second period, invested wealth only consists of assets and therefore the share invested in equity decreases to 0.33. With the constant relative risk-aversion specification, the share is the same for all possible realizations of \tilde{R}_2 . Consequently, the standard deviation of ω_2 is 0. Due to the decreased equity share in the portfolio in period 2, the expected rate of return decreases from 1.22

² Try to show this behaviour by loosening the upper constraint for ω . You will see that $\omega_1 > 1$ will then be optimal.

to $0.67 \cdot 1 + 0.33 \cdot 1.22 = 1.07$. In order to compensate for the resulting decrease in expected consumption in the third period, assets in periods 2 and 3 increase. Next, we introduce uncorrelated wage uncertainty, so that the uncertainty of second-period income increases. Now, human capital, i.e. the wage in the second period, is already a risky investment. Hence, the share of equity in the first period's portfolio has to decrease to $\omega_1 = 0.75$. Due to the precautionary savings motive that was introduced in Section 5.2, overall assets in the second period will rise considerably. Since the risk structure of the second period's investment is not affected by wage uncertainty and preferences are of constant relative risk-aversion type, ω_2 remains unaltered. Note, that the equity share in the portfolio still decreases over the life cycle. This pattern, however, is turned around in the next simulation, where we assume that wage and interest income risk are positively correlated. Consequently, a bad realization of wage income is very much likely accompanied by a low interest realization. In order to avoid situations with low wage and capital income, all savings in the first period are now invested in bonds. This again reduces the expected portfolio return and therefore overall savings increase in the first period. Since the risk structure of capital returns in the third period again is not influenced by the correlation, ω_2 remains the same. Note that now the equity share in the portfolio increases over the life cycle. Of course, the opposite happens when wage and interest risk are negatively correlated. In this case, it is possible to hedge against wage income risk by investing in stocks, as low wage income is most likely to come along with high interest rates. Consequently, precautionary savings decrease and all second-period assets are invested in stocks.

5.2.2 UNCERTAIN LIFESPAN AND ANNUITY CHOICE

As already discussed in Chapter 4 individuals also face mortality risk over the life cycle. In order to insure against the risk of outliving one's resources, they may buy *annuity* contracts that pay out a lot more compared to regular assets in the event that the household dies very late. Of course, if the individual dies early, the insurer receives the remaining amount of savings. At first sight, one would expect annuity contracts to be the preferred savings vehicle for households facing lifespan uncertainty. Nevertheless, taking a look at the data, we find that people save much more in regular assets than in annuities. In the literature, this fact is called the *annuity puzzle*. Various attempts were made to explain this behaviour. Bequest motives or market failure due to adverse selection problems might be one explanation. However, as we will show in this section, uncertainty about labour income can also explain part of the non-annuitization of assets, especially in earlier stages of life.

In the following, we will introduce uncertain lifespan into the life-cycle model. As in Chapter 4 the probability of an individual aged i to die within the current period is denoted by q_i . The household will survive with probability $\psi_2 = 1 - q_1$ from period 1 to 2 and, conditional on having survived to period 2, the individual only lives up to

period 3 with probability $\psi_3 = 1 - q_2$. In order to insure against the risk of longevity, he might purchase annuities. As in Chapter 4 the price of an annuity contract bought in the first or second period of life that pays €1 in all future periods is

$$p_{a,1} = \frac{\psi_2}{R} + \frac{\psi_2\psi_3}{R^2} \quad \text{and} \quad p_{a,2} = \frac{\psi_3}{R}.$$

If we assume a perfectly competitive annuity market (and abstract from systematic mortality risk), insurers will not run any surplus. Therefore, for one unit of income invested in annuities at period i , the household will receive a constant income stream of $\frac{1}{p_{a,i}}$ in all remaining future periods.

Beneath choosing the optimal asset level, the agent now also has to decide about which fraction ω_i of the savings to invest in annuities. The remaining part of assets $1 - \omega_i$ will be invested in regular assets at a fixed rate R . Hence, the optimization problem turns into

$$\max_{a_2, \omega_1, \tilde{a}_3, \tilde{\omega}_2} u(c_1) + \psi_2 \beta E[u(\tilde{c}_2) + \psi_3 \beta u(\tilde{c}_3)],$$

where expectations now not only are formed with respect to labour-income uncertainty but also with respect to survival. The household therefore only receives utility in a certain period if he survives. We assume that assets have to be greater than a lower threshold \underline{a} . With this assumption, we can loosen households borrowing limit of $a \geq 0$ imposed in Sections 5.1 and 5.2.1. The budget constraints finally turn into

$$\begin{aligned} c_1 &= w - a_2 \\ \tilde{c}_2 &= \tilde{w} + R(1 - \omega_1)a_2 + \frac{\omega_1 a_2}{p_{a,1}} - \tilde{a}_3 \\ \tilde{c}_3 &= pen + R(1 - \tilde{\omega}_2)\tilde{a}_3 + \frac{\tilde{\omega}_2 \tilde{a}_3}{p_{a,2}} + \frac{\omega_1 a_2}{p_{a,1}}, \end{aligned}$$

where pen is a pension that is payed out to the individual in the last period of life.

Module 5.5m shows part of the function that is needed to compute the solution of the annuitization problem. Before minimizing this function, we again discretize the wage distribution into integration nodes and weights and set the tolerance level to 10^{-14} .³ Next we compute the annuity factors $p_{a,i}$ and set the intervals for optimization. Note that in this program we set the lower limit of optimization for a_i to \underline{a} . However, we still assume that $0 \leq \omega_i \leq 1$ has to hold. This means that agents will be allowed to run into debt if $\underline{a} < 0$, however, they will not be able to short sell regular assets in order to finance annuity purchases. This is a necessary condition for the existence of a solution, as in the second period of life annuities are strictly preferred to regular assets. Consequently, households

³ We could keep the tolerance level at the value that is predefined by the compiler. With that many variables this would, however, lead to relatively inaccurate results.

Module 5.5m Lifespan uncertainty and annuity choice

```

function utility(x)
    [.....]
    ! savings
    a(1, :) = 0d0
    a(2, :) = x(1)
    omega(1, :) = x(2)
    ic = 3
    do iw = 1, n_w
        a(3, iw) = x(ic)
        omega(2, iw) = x(ic+1)
        ic = ic+2
    enddo

    ! consumption (insure consumption > 0)
    c(1,:) = mu_w - a(2,1)
    c(2,:) = R*(1d0-omega(1,1))*a(2,1) + &
               omega(1,1)*a(2,1)/p_a(1) + w(:) - a(3,:)
    c(3,:) = R*(1d0-omega(2,:))*a(3,:) + &
               omega(2,:)*a(3,:)/p_a(2) + omega(1,1)*a(2,1)/p_a(1)+pen
    c = max(c, 1d-10)

    ! expected utility of periods 2 and 3
    utility = 0d0
    do iw = 1, n_w
        utility = utility+weight_w(iw)*(c(2,iw)**egam + &
                                         psi(3)*beta*c(3,iw)**egam)
    enddo

    ! add first period utility
    utility = -(c(1,1)**egam + psi(2)*beta*utility)/egam

end function

```

Table 5.4 Uncertain lifespan and annuity choice

| σ_w^2 | a | c_1 | a_2 | a_2^d | $E[c_2]$ | $E[a_3]$ | $E[a_3^d]$ | $E[c_3]$ |
|--------------|-----------|-------|-------|---------|----------|----------|------------|----------|
| 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 | 0.00 | 0.00 | 1.00 |
| 0.30 | 0.00 | 0.88 | 0.00 | 0.12 | 1.03 | 0.00 | 0.07 | 1.23 |
| 0.60 | 0.00 | 0.83 | 0.08 | 0.09 | 1.06 | 0.00 | 0.10 | 1.27 |
| 0.90 | 0.00 | 0.80 | 0.17 | 0.03 | 1.08 | 0.00 | 0.12 | 1.27 |
| 1.20 | 0.00 | 0.78 | 0.22 | 0.00 | 1.09 | 0.00 | 0.14 | 1.28 |
| 1.20 | $-\infty$ | 0.91 | 0.00 | 0.09 | 1.24 | -0.29 | 0.12 | 1.02 |

$$R = \beta = w = \mu_w = pen = 1, \gamma = 0.5, \psi_2 = 0.8, \psi_3 = 0.5, n_w = 5.$$

would always like to borrow from regular assets and invest in annuities as this would be an arbitrage strategy. Having specified the optimization intervals, we can start minimizing the function `utility`. In this function, we again first copy savings levels and annuity shares in the portfolio into the respective variables. After that we calculate consumption in the different periods and expected utility. Note that beneath the weights for the wage distribution, we also have to compute expectations using survival probabilities.

Table 5.4 shows some simulation results of the model. We set survival probabilities at $\psi_2 = 0.8$ and $\psi_3 = 0.5$ and assume a pension that fully replaces labour income.

Consequently, in the first simulation where wages are certain in both periods, households will not save in either regular assets nor in annuities, as their pension is enough to finance old-age consumption. If we now increase wage uncertainty, we see that overall savings steadily increase. This is due to the precautionary savings motive. If uncertainty is low, agents start to invest in annuity contracts, as those are the preferred savings vehicle due to higher returns. However, with increasing wage risk, it turns out that households start to put their savings in regular assets rather than annuities in the first period. This behaviour is quite reasonable: annuities bought in the first period only pay out part of the asset amount in period 2. As the need for precautionary savings to insure a minimum consumption level for bad realizations of wage income increases, households would like to transfer more income into the second period without increasing third-period consumption too much. This can only be done by saving in regular assets. In the extreme case of very high income uncertainty in period 2, agents will only save in regular assets in period 1. Note that for period 2 the dominant savings strategy is $\omega_2 = 1$, as the return on annuities is higher than that of regular assets and the full savings amount will be payed out in the next period. In the last simulation of Table 5.3, we loosen the household's borrowing constraint by setting a to $-\infty$.⁴ Without borrowing constraints, the full savings amount in period 1 will again be invested in annuities. As agents can now borrow against future annuity income, they are able to transfer income from period 3 into period 2. Therefore they don't mind that part of their annuity savings will be payed out in period 3 and annuities again become the dominant savings vehicle. Note, however, that the assumption of no borrowing constraints at later stages in life is quite unrealistic, since an annuity can't be seen as a real collateral. If agents die early, the insurer that issued the annuity will receive all the capital left in the contract. Hence, the bank will not be repaid. Therefore we think that borrowing constraints might be a reasonable assumption especially in later stages of life.

5.3 Further reading

The life-cycle model was introduced in the early 1950s by Franco Modigliani and his student Richard Brumberg. The implications of the life-cycle hypothesis for economic theory and policymaking are discussed extensively by Deaton (2005), Baranzini (2005), and Jappelli (2005) in an international conference to honour Modigliani. Browning and Crossley (2001), as well as Attanasio and Weber (2010), recently discussed the standard life-cycle model. A more extensive literature review on recent approaches to life-cycle modelling looking at labour-supply behaviour and portfolio choice can be found in Chapter 10.

⁴ In the program we set `a_lower` to -1 in order to avoid computational problems. However, we have checked that no agent is borrowing-constrained in this simulation.

5.4 Exercises

- 5.1. Up to this point we have assumed that goods are perishable, meaning that they have to be consumed in one period. This exercise introduces so-called *durable goods*, which can be consumed in different periods during the life cycle once they have been purchased. Let's assume the household has to decide between a housing asset a_h as a durable consumption good and ordinary consumption c . The periodical utility function is

$$u(c, a_h) = \frac{[\theta c^\nu + (1 - \theta)a_h^\nu]^{1-1/\gamma}}{1 - 1/\gamma},$$

where θ defines the share of non-housing consumption and ν defines the elasticity of substitution between housing and non-housing consumption.

In the first period of life the household has to split earnings up into ordinary consumption c_1 , durable housing consumption a_h , and savings (which could be negative). In each future period the household has to incur maintenance cost $\delta_h a_h$ for the house and in the last period the house is sold and rented at price $p_h = r + \delta_h$. The three budget constraints therefore are

$$\begin{aligned} c_1 &= w - a_h - a_2 \\ c_2 &= w + Ra_2 - a_3 - \delta_h a_h \\ c_3 &= Ra_3 + (1 - p_h - \delta_h) a_h. \end{aligned}$$

Implement this durable good model in the deterministic life-cycle framework of Program 5.1. Do not forget to specify a maximum debt level $a_{\text{low}}=0.5d0$, which should be included as a lower bound for assets and (together with wages w) as an upper bound for housing.

- a) Set $\delta_h = 1$, $\nu \leq 0.7$ and $\theta = 0.5$ and simulate the model. Explain your results in economic terms.
 - b) Now increase $\theta = 0.7$, reduce δ_h to 0.3 and increase R to 1.2. Explain your results in economic terms.
 - c) What would happen with housing demand if we introduce labour-income uncertainty in period 2? (No programming required!)
- 5.2. Let the wage rate w and interest factor R be fixed as in Program 5.1. However, assume that the household faces uncertain health expenditures hc in the last period of life. These expenditures are uniformly distributed on the interval $hc \in [\mu - \sigma; \mu + \sigma]$. Write a program that solves this life-cycle problem using the parameters of Program 5.1 and test how savings and consumption in different periods vary with σ given $\mu = 0.5$.

- 5.3. Now assume that in period 2 of the life cycle the household faces disability risk. We model disability risk by letting the household's wage rate drop to zero with a certain probability ω . Otherwise the wage rate is equal to the wage rate in period 1. Consequently we have

$$\tilde{w}_2 = \begin{cases} 0 & \text{with probability } \omega \\ w & \text{with probability } 1 - \omega. \end{cases}$$

We abstract from other health risks in all periods.

Examine how the household's savings in period 1 and consumption in periods 2 and 3 react to rising disability risk by setting $\omega \in \{0, 0.01, 0.1, 0.5\}$.

- 5.4. The literature has emphasized the role of entry cost in explaining a limited stock market participation. In the present model such cost can be modelled as a one-time fixed cost $F < w$ an investor incurs when entering the stock market in the first period. The problem is that this creates a discontinuity in the utility function. Consequently, one has to compute the welfare with (i.e. including the fixed cost) and completely without stock market participation separately and then decide whether to participate or not.

Adjust Module 5.4m by introducing two utility functions `utility_st` and `utility_b`, which calculate utility with and without stock market participation. Use these functions to derive the optimal utility levels corresponding to the two cases in the main program and derive the participation decision. Compare the entry costs that make a household indifferent between participating and not participating in the situation without uncertainty and with uncertainty (i.e. use the parameters of the first and third rows of Table 5.3).

- 5.5. The concepts of relative and absolute risk aversion are important in understanding the economic behaviour of households in uncertain situations. Absolute risk aversion is defined by $-\frac{u''(c)}{u'(c)}$ while relative risk aversion is defined by $-c \frac{u''(c)}{u'(c)}$. As in most of the literature, we most often apply a utility function that exhibits a constant relative risk aversion (CRRA) of $\frac{1}{\gamma}$. Consequently, the relative shares of risky investment are independent of the available resources, see the discussion of the results from Table 5.3.

In this exercise you should use the utility function

$$u(c) = 1 - e^{-c/\gamma}.$$

It is easy to check that this function exhibits a constant absolute risk aversion of $\frac{1}{\gamma}$ and therefore features increasing relative risk aversion. Examine the consequences of this preference structure on risky investments in period 2 (i.e. Table 5.3). Prepare a graph that shows the correlation of available resources in period 2 and the share of risky investment.

- 5.6. Consider the model with uncertain lifespan, but now assume that households derive utility from leaving bequest to descendants. The literature often draws on the so-called ‘warm glow’ bequest motive, where households derive utility directly from the amount of bequest b they leave. For simplicity, we assume that the utility function that values bequests is the same as for ordinary consumption, so that the household’s utility function turns into

$$\begin{aligned} \max_{a_2, \omega_1, \tilde{a}_3, \tilde{\omega}_2} & u(c_1) + \beta(1 - \psi_2)\nu u(b_2) + \beta\psi_2 \left[Eu(\tilde{c}_2) + \beta(1 - \psi_3)\nu Eu(\tilde{b}_3) \right] \\ & + \beta^2\psi_2\psi_3 \left[Eu(\tilde{c}_3) + \beta\nu Eu(\tilde{b}_4) \right]. \end{aligned}$$

$\nu \geq 0$ measures the intensity of the household’s bequest motive. Only savings in non-annuitized assets can be bequeathed, so that we can calculate the bequest levels as

$$b_2 = R(1 - \omega_1)a_2, \quad \tilde{b}_3 = R(1 - \tilde{\omega}_2)\tilde{a}_3 \quad \text{and} \quad \tilde{b}_4 = R\tilde{a}_4.$$

Write a program that solves this problem:

- a) In the model without uncertainty, eliminate pension payments and the bequest motive (i.e. $pen = \nu = 0$). How much do households annuitize?
 - b) Successively increase the bequest intensity and pensions to $\nu = 0.05$ and $pen = \mu_w$. What happens to annuitized assets? Explain in economic terms.
- 5.7. Combine the models with annuity demand and risky investment choices. Households have to decide how much they want to invest in bonds, equity, and annuities. With respect to the annuity fund, for simplicity we assume a deterministic interest factor \hat{R} . In addition, we allow for a so-called ‘load factor’ ξ that accounts for the fact that the annuity is not priced at its actuarially fair value. Consequently, annuity prices are determined by

$$p_{a,1} = \left[\frac{\psi_2}{\hat{R}} + \frac{\psi_2\psi_3}{(\hat{R})^2} \right] \cdot (1 + \xi) \quad \text{and} \quad p_{a,2} = \frac{\psi_3}{\hat{R}} \cdot (1 + \xi).$$

With respect to the shares of equity and (retirement) annuities we distinguish ω_i and $\omega_{r,i}$ in the periods $i = 1, 2$, respectively.

- a) Derive the budget constraints of this model for the three periods.
- b) Implement and solve the model using $\sigma_w^2 = \sigma_R^2 = \rho = \xi = 0$ and $\hat{R} = R_f$. Then consecutively introduce $\sigma_w^2 = \sigma_R^2 = \rho = 0.5$, $\xi = 0.2$ and $\hat{R} = 0.9$. Explain the results in economic terms.

6 The overlapping generations model

In discussing the life-cycle model, we focused on the individual-choice problem without taking into account the interaction between households, the production sector of the economy, and the government. In this chapter we take a broader perspective and embed the life-cycle model into a general equilibrium framework. In this framework, prices adjust in order to balance supply and demand in goods and factor markets and the government has to operate under some balanced-budget rules. As in the previous chapter, individuals save in order to smooth consumption over the life cycle. However now, individual savings behaviour endogenously determines the capital stock. This is the central difference from the static general equilibrium model discussed in Chapter 3. Since in our equilibrium framework we have to distinguish households within a given period according to their age or birth year, the models we study are called overlapping generations (OLG) models. In this chapter we introduce the most basic version of the OLG model and discuss the computation of a transition path and the intergenerational welfare effects of policy reforms. In Chapter 7 we extend this baseline model version in various directions.

6.1 General structure and long-run equilibrium

6.1.1 DEMOGRAPHICS, BEHAVIOUR AND MARKETS

This subsection sketches the economic environment used in this chapter and Chapter 7. We describe the lifetime of people who inhabit the economy as well as their consumption decisions. Then we move on to the production side and the government structure. Finally, the equilibrium conditions for goods and factor markets which close the model are derived.

Demographics As in Chapter 5 we assume that households in the model live for three periods. For simplicity we do not account for income and lifespan uncertainty. However, now in each successive period t a new cohort is born, where the number of households N_t in this cohort grows at a rate $n_{p,t}$, i.e.

$$N_t = (1 + n_{p,t})N_{t-1}.$$

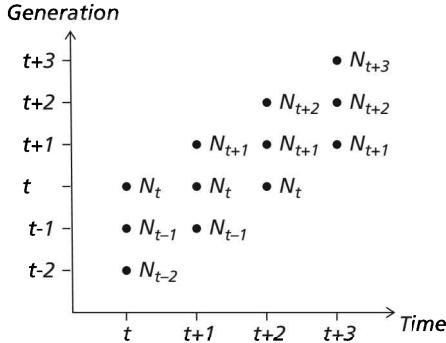


Figure 6.1 Overlapping generations

From Figure 6.1 one can understand why this demographic structure is called ‘overlapping generations’. In each period t a cohort N_t is born, but this ‘new’ cohort overlaps with the two cohorts N_{t-1} and N_{t-2} born in the previous two periods. Consequently, in period t the total population size is $N_t + N_{t-1} + N_{t-2}$. Since we again assume that households work in the first two periods of their life and retire in the third, $N_t + N_{t-1}$ defines the size of the workforce in period t .

Households’ decisions Households decide about how much to consume and save in the different periods of their life. They have to pay taxes on income and consumption as well as a payroll tax to the pension system, in reward for which they receive pension benefits pen_{t+2} in the third period of life. The *dynamic budget constraints* in the three different periods are then given by

$$p_t c_{1,t} = w_t^n - a_{2,t+1} \quad (6.1)$$

$$p_{t+1} c_{2,t+1} = R_{t+1}^n a_{2,t+1} + w_{t+1}^n - a_{3,t+2} \quad (6.2)$$

$$p_{t+2} c_{3,t+2} = R_{t+2}^n a_{3,t+2} + pen_{t+2} \quad (6.3)$$

where $w_t^n = w_t(1-\tau_t^w - \tau_t^p)$, $R_t^n = 1+r_t(1-\tau_t^r)$, and $p_t = 1+\tau_t^c$ define net wage rates, net interest rates, and consumer prices, respectively. τ_t^c , τ_t^w , τ_t^p , and τ_t^r denote consumption, labour-income, payroll, and capital-income tax rates. As in Chapter 5, households born in period t maximize the utility function

$$U_t = U(c_{1,t}, c_{2,t+1}, c_{3,t+2}) = u(c_{1,t}) + \beta u(c_{2,t+1}) + \beta^2 u(c_{3,t+2})$$

subject to the *intertemporal budget constraint*

$$p_t c_{1,t} + \frac{p_{t+1} c_{2,t+1}}{R_{t+1}^n} + \frac{p_{t+2} c_{3,t+2}}{R_{t+2}^n} = w_t^n + \frac{w_{t+1}^n}{R_{t+1}^n} + \frac{pen_{t+2}}{R_{t+2}^n} =: W_{1,t}, \quad (6.4)$$

which can be derived from the dynamic budget constraints (6.1) to (6.3). The resulting first-order conditions

$$p_{t+1}u'(c_{1,t}) = \beta R_{t+1}^n p_t u'(c_{2,t+1}) \quad (6.5)$$

$$p_{t+2}u'(c_{2,t+1}) = \beta R_{t+2}^n p_{t+1} u'(c_{3,t+2}) \quad (6.6)$$

together with the constraint (6.4) define the three consumption functions

$$\begin{aligned} c_{1,t} &= C_1(w_t^n, w_{t+1}^n, pen_{t+2}, R_{t+1}^n, R_{t+2}^n, p_t, p_{t+1}, p_{t+2}), \\ c_{2,t+1} &= C_2(w_t^n, w_{t+1}^n, pen_{t+2}, R_{t+1}^n, R_{t+2}^n, p_t, p_{t+1}, p_{t+2}) \quad \text{and} \\ c_{3,t+2} &= C_3(w_t^n, w_{t+1}^n, pen_{t+2}, R_{t+1}^n, R_{t+2}^n, p_t, p_{t+1}, p_{t+2}). \end{aligned}$$

for the cohort born in period t . As initial assets are zero, the savings functions

$$\begin{aligned} a_{2,t+1} &= S_1(w_t^n, w_{t+1}^n, pen_{t+2}, R_{t+1}^n, R_{t+2}^n, p_t, p_{t+1}, p_{t+2}) \quad \text{and} \\ a_{3,t+2} &= S_2(w_t^n, w_{t+1}^n, pen_{t+2}, R_{t+1}^n, R_{t+2}^n, p_t, p_{t+1}, p_{t+2}) \end{aligned}$$

can be derived by plugging the consumption functions into the budget constraints (6.1) to (6.3).

Aggregation Since different households are living in the same period t , the demand for goods and the supply of labour and capital need to be determined by aggregating individual household decisions. We normalize all aggregate variables to the size of the newborn cohort. Since each working household is assumed to supply one unit of labour inelastically, aggregate labour supply in period t can be computed from

$$L_t = \frac{1}{N_t} (N_t + N_{t-1}) = 1 + \frac{1}{1 + n_{p,t}} = \frac{2 + n_{p,t}}{1 + n_{p,t}}.$$

Similarly, we derive aggregate per capita assets

$$A_t = \frac{1}{N_t} (N_{t-1}a_{2,t} + N_{t-2}a_{3,t}) = \frac{a_{2,t}}{1 + n_{p,t}} + \frac{a_{3,t}}{(1 + n_{p,t})(1 + n_{p,t-1})},$$

where $a_{2,t}$ and $a_{3,t}$ denote savings of the two older cohorts built in the previous year, and per capita consumption from

$$\begin{aligned} C_t &= \frac{1}{N_t} (N_t c_{1,t} + N_{t-1} c_{2,t} + N_{t-2} c_{3,t}) \\ &= c_{1,t} + \frac{c_{2,t}}{1 + n_{p,t}} + \frac{c_{3,t}}{(1 + n_{p,t})(1 + n_{p,t-1})}. \end{aligned}$$

Firms' decisions The production technology is represented by the neoclassical production function $F(K_t, L_t)$, which is homogeneous of degree one. The capital stock depreciates physically during the production process at rate δ . Consequently, a firm's output Y_t can be derived from

$$Y_t = F(K_t, L_t) \quad \text{or} \quad y_t = \frac{Y_t}{L_t} = f(k_t) \quad (6.7)$$

where y_t denotes per capita output and $k_t = \frac{K_t}{L_t}$ capital intensity. Firms are assumed to maximize (per capita) profits $\pi_t = y_t - (r_t + \delta)k_t - w_t$ by choosing the optimal capital intensity. They produce under perfect competition conditions. Therefore equilibrium factor prices r_t and w_t are derived from first-order and zero-profit conditions as

$$r_t = f'(k_t) - \delta \quad (6.8)$$

$$w_t = f(k_t) - f'(k_t)k_t. \quad (6.9)$$

Note that equations (6.7) to (6.9) imply $y_t = w_t + (r_t + \delta)k_t$ or $Y_t = w_t L_t + (r_t + \delta)K_t$.

The government The government sector comprises two separate budgets. It finances the provision of a public good G_t and interest payments on public debt $r_t B_t$ by means of tax revenues and deficit spending, i.e.

$$T_t + (1 + n_{p,t+1})B_{t+1} - B_t = G_t + r_t B_t, \quad (6.10)$$

where $T_t = \tau_t^c C_t + \tau_t^w w_t L_t + \tau_t^r r_t A_t$ defines tax revenues from consumption and income taxation. The amount of public good is computed depending on the population structure, i.e.

$$\begin{aligned} G_t &= \frac{1}{N_t} (N_t g_1 + N_{t-1} g_2 + N_{t-2} g_3) \\ &= g_1 + \frac{g_2}{1 + n_{p,t}} + \frac{g_3}{(1 + n_{p,t})(1 + n_{p,t-1})} \end{aligned}$$

where g_i represent per capita coefficients of cohort i for public good provision. Furthermore, we determine the deficit path exogenously such that

$$B_{t+1} = b_{y,t} Y_{t+1}$$

holds in every period. Given expenditures for the public good G_t and the deficit path $b_{y,t}$, we let one of the three tax rates balance the periodical budget of the government.

With respect to the second government budget we assume that in each period t the level of pension benefits is computed as a fraction κ_t of the previous period's gross wage, i.e.

$$pen_t = \kappa_t \cdot w_{t-1}.$$

The pension system is fully pay-as-you-go financed. Hence, contributions from workers have to finance benefit payments to retirees

$$\tau_t^P w_t (N_t + N_{t-1}) = pen_t N_{t-2} \quad \text{or} \quad \tau_t^P w_t = \frac{pen_t}{(2 + n_{p,t})(1 + n_{p,t-1})}. \quad (6.11)$$

Market equilibrium For a market equilibrium the goods market and factor markets need to be balanced. On the capital market of a closed economy, interest rates are determined so as to equalize households' aggregate savings A_t with the demand for capital of the firms K_t and the government B_t , i.e.

$$A_t = K_t + B_t. \quad (6.12)$$

In the labour market, wages are set in a way that guarantees that aggregate labour supply of households matches labour demand of the firms.¹ Finally, in the goods market total output of the firms must be used either for private and public consumption or for building the new capital stock, i.e.

$$Y_t = C_t + G_t + \underbrace{(1 + n_{p,t+1})K_{t+1} - (1 - \delta)K_t}_{=I_t}. \quad (6.13)$$

Note that we can derive this last equation from aggregating the budget constraints of households (6.1) to (6.3), the government's budgets (6.10) and (6.11) as well as the capital market equilibrium condition (6.12), which means that the goods market automatically clears if all the factor markets are in equilibrium. This feature is called *Walras' law*.

6.1.2 COMPUTATION OF THE LONG-RUN EQUILIBRIUM

This subsection focuses on the computation of long-run equilibria or so-called *steady states*. Steady states are characterized by the fact that per capita variables are constant over time. Consequently, the time index t can be omitted in all equations of this subsection.

Functional forms In order to be able to compute a solution of our model, we first have to define concrete functional forms of preference and production functions. As in Chapter 5, we apply the CRRA utility function given by

¹ Since we substitute labour supply of households into the production function, the labour market is always balanced.

$$u(c) = \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}} \quad \text{with} \quad u'(c) = c^{-\frac{1}{\gamma}},$$

where γ denotes the intertemporal elasticity of substitution between consumption in different years. Substituting the first-order conditions of the household (6.4) and (6.5) which are now

$$c_2 = (\beta R^n)^\gamma c_1 \quad \text{and} \quad c_3 = (\beta R^n)^\gamma c_2 = (\beta R^n)^{2\gamma} c_1$$

into the intertemporal budget constraint (6.4) which is now

$$pc_1 + \frac{pc_2}{R^n} + \frac{pc_3}{(R^n)^2} = w^n + \frac{w^n}{R^n} + \frac{pen}{(R^n)^2} =: W_1,$$

consumption in the first period of the life of a household can be derived from

$$c_1 = \Psi_1 W_1 \quad \text{with} \quad \Psi_1 = \frac{1}{p} \left\{ 1 + \beta^\gamma (R^n)^{\gamma-1} + \beta^{2\gamma} (R^n)^{2(\gamma-1)} \right\}^{-1}.$$

Given c_1 , the remaining consumption path c_2, c_3 is derived using the first-order conditions above.

The savings decisions are given by

$$a_2 = w^n - pc_1 \quad \text{and} \quad a_3 = w^n + R^n a_2 - pc_2.$$

The production technology is assumed to be of Cobb-Douglas type, so that

$$f(k) = k^\alpha \quad \text{and} \quad f'(k) = \alpha k^{\alpha-1}.$$

Interest and wage rates are then determined by

$$r = \alpha k^{\alpha-1} - \delta \quad \text{and} \quad w = (1 - \alpha)k^\alpha.$$

Exogenous parameters We can split all variables of our model into *exogenous* and *endogenous* ones. On the endogenous side there are household decisions, aggregate quantities, factor prices, the payroll tax rate, as well as one additional tax rate (consumption, labour income, or capital income) which balances the government's budget. We initially let the consumption tax rate be endogenous. Exogenous parameters of our model then are preference parameters (γ, β) , technology parameters (α, δ) , the population growth rate n_p , as well as the government parameters $(g_j, \tau^w, \tau^r, b_y)$ and the replacement rate κ of the pension system. We chose similar preference parameters as we did in Chapter 5. We exclude depreciation and assume a capital-income share in production of 30 per cent,

i.e. $\alpha = rK/Y = 0.3$. Note that in our three-period model one period roughly reflects 20 years. Consequently, the assumed population growth rate of 0.2 reflects an annual growth rate of less than 1 per cent. Finally, in order to generate a realistic government share in GDP, we set per capita public goods consumption $g_1 = g_2 = 0.12$ and $g_3 = 0$. In the initial equilibrium we leave out debt, the pension system, and balance the budget only by consumption taxes.

The long-run equilibrium Due to constant population growth, the labour force is given by $L = \frac{2+n_p}{1+n_p}$ while the payroll tax rate can be derived from the budget of the pension system (6.10) as

$$\tau^P = \frac{\kappa}{(2+n_p)(1+n_p)}.$$

Given that for a certain capital intensity and combination of tax rates we can compute factor prices and individual decision variables from the equations presented above, there are two remaining variables we have to solve for: the capital stock K_t that balances the capital market equilibrium condition and the consumption tax rate that balances the government's budget. This is done in Program 6.1. We thereby first construct a module `globals` in which we store all global variables of our model. The module is shown in Module 6.1m. Beneath the model parameters, we also store the endogenous variables in this module. Note that, as Fortran does not distinguish variables in small or capital letters, we use variables with twice the same letter as capital letter variables, i.e. K in the model description is `kk` in the program etc. The parameter `tax` indicates which of the different tax rates should be used to balance the government's budget: 1 for consumption taxes, 2 for income taxes, 3 for labour-income taxes, and all other values for capital-income taxes.

Program 6.1 Computation of long-run equilibrium

```

program LR_OLG
[.....]
! initialize labour supply, pension payments, and tax rates
LL = (2d0+n_p)/(1d0+n_p)
taup = kappa/((2d0+n_p)*(1d0+n_p))
tauc = 0d0
tauw = 0d0
taur = 0d0

! get initial guess
x(:) = 0.7d0

! solve the steady-state equation system
call fzero(x, eqns, check)

! check whether the solution is valid
if(check) stop 'No equilibrium found !!!'
[.....]
end program

```

Module 6.1m Parameters and variables for long-run equilibrium

```

module globals

    implicit none

    ! model parameters
    real*8, parameter :: gamma = 0.5d0
    real*8, parameter :: egam = 1d0 - 1d0/gamma
    real*8, parameter :: beta = 0.9d0
    real*8, parameter :: alpha = 0.3d0
    real*8, parameter :: delta = 0.0d0
    real*8, parameter :: by = 0.0d0
    real*8, parameter :: kappa = 0.0d0
    real*8, parameter :: n_p = 0.2d0
    integer, parameter :: tax = 1
    real*8 :: g(3) = (/ 0.12d0, 0.12d0, 0.0d0 /)

    ! model variables
    real*8 :: w, r, wn, Rn, p, tauw, taur, tauc, taup, pen
    real*8 :: KK, LL, YY, AA, CC, BB, GG, II
    real*8 :: a(3), c(3), util
    [.....]
end module

```

The computation of the long-run equilibrium is shown in Program 6.1. We first initialize the size of the work force `LL`, the pension contribution rate `taup` as well as the tax rates on consumption, labour income, and capital income. The consumption tax rate will be endogenously determined later on. We then minimize the function `eqns`, which calculates the two equations needed to pin down aggregate capital and the endogenous tax rate. We use `fzero` to find the root of this equation system. Afterwards, we check for the correctness of the solution.

The equation system that determines the capital stock of the economy and the endogenous tax rate is calculated in the function `eqns` in Module 6.1m.a. This function receives guesses for the capital stock and the endogenous tax rate in the two entries of the variable `x`. With `KK` and `LL` as defined above we can calculate the capital intensity of the economy and therefore derive factor prices. Net factor prices result from gross factor prices subtracting taxes. Having calculated prices and knowing all tax rates, we can derive individual decisions and aggregate quantities. The function `eqns` then returns the difference on the capital market as well as the difference in the government's budget, which should be zero in equilibrium.

6.1.3 LONG-RUN ANALYSIS OF POLICY REFORMS

Table 6.1 summarizes long-run equilibrium outcomes of different modelling scenarios. We start in simulation (0) with our benchmark scenario in which consumption taxes balance the government's budget. In this benchmark the consumption tax rate is 29 per cent, the annual capital-output ratio is roughly 5.2 ($= 20 \times K/Y$), the wage rate

Module 6.1m.a Equation system for long-run equilibrium

```

function eqns(x)
    [.....]
    ! copy values to respective variables
    KK = x(1)
    if(tax == 1) then
        tauc = x(2)
    elseif(tax == 2) then
        tauw = x(2)
        taur = x(2)
    elseif(tax == 3) then
        tauw = x(2)
    else
        taur = x(2)
    endif

    ! factor prices and pension payments
    r = alpha*(KK/LL)**(alpha-1d0)-delta
    w = (1d0-alpha)*(KK/LL)**alpha
    wn = w*(1d0-tauw-taup)
    Rn = 1d0+r*(1d0-taur)
    p = 1d0+tauc
    pen = kappa*w

    ! individual decisions
    PVI = wn + wn/Rn + pen/Rn**2
    PSI = p*(1d0 + (beta*Rn)**gamma/Rn &
        + (beta*Rn)**(2*gamma)/Rn**2)
    c(1) = PVI/PSI
    c(2) = (beta*Rn)**gamma*c(1)
    c(3) = (beta*Rn)**gamma*c(2)
    a(2) = wn - p*c(1)
    a(3) = wn + Rn*a(2) - p*c(2)

    ! quantities
    YY = KK**alpha * LL**(1d0-alpha)
    CC = c(1) + c(2)/(1d0+n_p) + c(3)/(1d0+n_p)**2
    GG = g(1) + g(2)/(1d0+n_p) + g(3)/(1d0+n_p)**2
    AA = a(2)/(1d0+n_p) + a(3)/(1d0+n_p)**2
    II = (n_p+delta)*KK
    BB = by*YY

    ! get equations defining general equilibrium
    eqns(1) = KK + BB - AA
    eqns(2) = tauc*CC + tauw*w*LL + taur*r*AA - (r-n_p)*BB - GG

```

end function

is 0.39, and the interest rate is 1.15 (which amounts to an annual interest rate of roughly 3.9 per cent). Simulation (1) reveals that tax rates could be reduced to 24 per cent, if public goods were financed by taxes on labour and capital income instead of consumption. However, as taxation in such a regime takes place earlier in life and the net interest rate on capital decreases, the long-run capital stock is lower than in simulation (0). Consequently, interest rates rise and wages fall, which results in a long-run welfare loss in a dynamically efficient economy. The opposite happens in simulation (2), where all expenditure on public goods is financed by capital-income taxes. In this case, tax burdens are shifted from younger towards older cohorts. Consequently consumption increases at young ages and

Table 6.1 Long-run policy analysis

| | b_y | κ | n_p | τ^w | τ^r | τ^c | c_1 | c_2 | c_3 | K | U |
|-----|--------|----------|-------|----------|----------|----------|-------|-------|-------|------|--------|
| (0) | 0.00 | 0.00 | 0.20 | | | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | -9.54 |
| (1) | | | | 0.24 | 0.24 | | 0.19 | 0.26 | 0.37 | 0.18 | -10.94 |
| (2) | | | | | 0.71 | | 0.28 | 0.30 | 0.33 | 0.27 | -9.03 |
| (3) | | | | 0.37 | | | 0.15 | 0.23 | 0.37 | 0.14 | -12.93 |
| (4) | -0.059 | | | 0.22 | | | 0.22 | 0.30 | 0.42 | 0.27 | -9.54 |
| (5) | | 0.50 | | 0.19* | | 0.37 | 0.14 | 0.23 | 0.37 | 0.14 | -13.01 |
| (6) | 0.099 | | | | | 0.60 | 0.14 | 0.23 | 0.37 | 0.14 | -13.01 |
| (7) | | | 0.00 | | | 0.24 | 0.25 | 0.32 | 0.43 | 0.40 | -8.74 |

$$\gamma = 0.5, \beta = 0.9, \alpha = 0.3, \delta = 0, g_1 = g_2 = 0.12, g_3 = 0, * \tau^P.$$

decreases in old age. While income effects have a positive impact on individual savings, higher capital-income taxes will damp them. Overall the capital stock remains roughly constant compared to the benchmark simulation (0) and future cohorts benefit slightly from the reform. On the other hand, future cohorts suffer the most when public goods are solely financed by labour-income taxes in simulation (3), as the negative effects described in simulation (2) are strengthened. Yet, a switch towards labour-income taxation could be neutralized in the long run, if the government decreased its debt by -6 per cent of GDP, i.e. start saving, see simulation (4). Simulation (5) demonstrates that the introduction of a pay-as-you-go-financed pension system reduces long-run welfare, since it crowds out private savings. Consequently the capital stock as well as wages decrease. Interestingly, exactly the same happens in simulation (6), where the government increases public debt compared to the benchmark simulation. Finally, the last simulation (7) quantifies the long-run effects of a decline in fertility. With a falling population growth rate, the ratio between workers and pensioners decreases. In the present model, this implies a reduction of expenditure on public goods (which only depend on the size of the labour force), so that consumption taxes can be reduced. This induces an increase in savings and wages. Note, that a different specification of the demand for public goods and the tax structure could lead to quite different results. If, for example, expenditure on public goods only depended on the number of pensioners (i.e. $g_1 = g_2 = 0, g_3 > 0$) and was financed solely by labour-income taxes, quite the opposite would happen and future cohorts would be hurt by lower fertility rates. This also happens when fertility declines under a pay-as-you-go-financed pension system, which is discussed further in Section 6.2.3. In this case, the change in demographic structure increases the payroll tax rate, which again crowds out savings and reduces wages in the long run.

6.2 Transitional dynamics and welfare analysis

The steady-state analysis of the Section 6.1 allows us to quantify the long-run effects of fiscal policy and population dynamics. However, such a long-run analysis may be incomplete or may yield a distorted view of a policy reform, since the periods of the

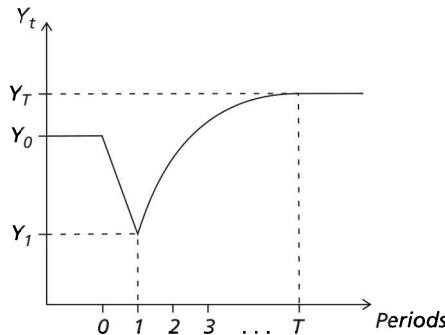


Figure 6.2 Dynamic adjustment between steady states

so-called *transitional path* in which the economy successively adjusts from the initial steady state towards the new long-run equilibrium is completely ignored. For example, in subsection 6.1.3 the introduction of a pay-as-you-go-financed pension system yielded a long-run welfare loss. Consequently, the OLG model seems to favor the reduction or privatization of the unfunded pension system. However, as will be shown in the following, welfare losses of future cohorts are mainly due to welfare gains of transitional cohorts. Therefore, the introduction of a pay-as-you-go pension system mainly redistributes resources across generations, so that only a government with specific distributional goals would implement such a policy.

This section discusses how to quantify the intergenerational effects of government policy. In order to do so, we have to distinguish between different simulation periods $t = 0, 1, \dots, T$. Period $t = 0$ denotes the initial long-run equilibrium. If the economic environment stayed unchanged, all per capita variables would remain constant for any future period. However, in period $t = 1$ a specific policy reform is introduced or birth rates change, so that individuals adapt their behaviour and factor prices adjust. If the economic model fulfils stability conditions (which is assumed in the following), after some time the system will again approach a new long-run equilibrium. In the numerical model we let this new steady state be reached after T periods. Figure 6.2 describes a hypothetical dynamic adjustment path between two long-run equilibria for per capita output Y_t . Before the policy reform has taken place, per capita output is constant over time at Y_0 . When the reform is implemented in period 1, per capita output declines to Y_1 initially. In the years afterwards it again increases until it finally reaches its new steady-state level Y_T . This long-run level remains constant in all periods after T . Of course, all remaining variables of the model experience similar transitional dynamics. We now go on to discuss how to compute transition paths in the model described above.

6.2.1 COMPUTATION OF TRANSITIONAL DYNAMICS

Module extensions In order to account for transitional dynamics in our simulation model, we first need to make variables time dependent. Module 6.2m shows the

Module 6.2m The global variables module under transitional dynamics

```

module globals

    implicit none

    ! model parameters
    integer, parameter :: TT = 25
    [.....]
    real*8, parameter :: tol = 1d-5
    real*8, parameter :: damp = 0.25d0
    integer, parameter :: itermmax = 1000
    real*8 :: g(3) = (/ 0.12d0, 0.12d0, 0.0d0 /)

    real*8 :: by(0:TT) = 0d0
    real*8 :: kappa(0:TT) = 0.0d0
    real*8 :: n_p(0:TT) = 0.2d0
    integer :: tax(0:TT) = 1
    logical :: lsra_on = .false.

    ! model variables
    real*8 :: w(0:TT), r(0:TT), wn(0:TT), Rn(0:TT), p(0:TT)
    real*8 :: tauw(0:TT),taur(0:TT),tauc(0:TT),taup(0:TT),pen(0:TT)
    real*8 :: KK(0:TT),LL(0:TT),YY(0:TT),AA(0:TT),CC(0:TT),II(0:TT)
    real*8 :: BB(0:TT), GG(0:TT), BA(0:TT)
    real*8 :: a(3,0:TT), c(3,0:TT), util(3,0:TT), v(-1:TT)
    [.....]
end module

```

respective changes that need to be made in the module `globals`. We first specify the number of transition periods needed to converge to a new long-run equilibrium `TT`. Preference, technology, and public good parameters will not depend on time, but the other variables do. We store in the array entry 0 the values of the initial equilibrium, in the entries 1:`TT`-1 transitional values, and in `TT` the new long-run equilibrium. In the initial equilibrium we again let consumption taxes balance the government's budget. Because of this `tax` is set to 1.

The main program The beginning of the main program, parts of which are presented in Program 6.2, is nearly identical to the long-run equilibrium version in Program 6.1. We first initialize aggregate labour supply, the payroll tax rate, and all other tax rates. In addition, we specify the change in policy or demographic parameters that should induce the transition path. In the case presented we change from a consumption to a purely labour-income tax system. Once everything is initialized, we compute the initial long-run equilibrium by finding the root of the equation system `eqns_Initial`. Having written the initial equilibrium values to a file called `output.out` by means of the subroutine `output`, we compute the transition path and long-run equilibrium using the subroutine `get_Transition`. We finally write the output of all transition years to the same file.

Program 6.2 The main program with transitional dynamics

```

program TR_OLG
[.....]
! set reform values
tax(1:TT) = 2
[.....]
! get initial guess
x(:) = 0.7d0

! solve the steady state equation system
call fzero(x, eqns_Initial, check)

! check whether the solution is valid
if(check) stop 'No equilibrium found !'

! write output
open(20, file='output.out')
call output(0, 20)

! initialize transitional values
call get_Transition

! write output
do it = 1, TT
    call output(it, 20)
enddo
close(20)
[.....]
end program

```

Prices and quantities along the transition path The calculation of transitional dynamics and the new long-run equilibrium is done in the subroutine `get_Transition`, which is shown in Module 6.2m.a. A straightforward extension of Program 6.1 would be to set up a similar equation system for the transition path as for the long-run equilibrium and solve it using `fzero`. A program with such a solution mechanism is available upon request. Unfortunately, the computation time for the solution of the transitional dynamics equation system grows quadratically in the number of transition periods `TT`. Therefore we present an iterative solution method, often called the *Gauss-Seidel method*. This method is used in Module 6.2m.a and works as follows:

1. We first initialize the capital stock along the transition at initial equilibrium values, $K_t = K_0$ for all $t \geq 1$, and assume that labour supply and tax rates have values as initialized in the main program.
2. With the capital stock K_t and labour supply L_t we can calculate the capital intensity $\frac{K_t}{L_t}$ and therefore derive factor prices r_t and w_t in the subroutine `factor_prices` for every year $t \geq 1$. We then use tax rates to determine net prices.
3. Next we use net prices in subroutine `decisions` to compute individual decisions of young cohorts in exactly the same way as presented in Section 6.1.2. Note, however,

Module 6.2m.a Computation of transitional dynamics

```

subroutine get_Transition()
[.....]
! initialize values from initial equilibrium
KK(:) = KK(0)
nmarket = 0

do iter = 1, itermax

    ! get prices, decisions and quantities
    do it = 1, TT
        call factor_prices(it)
    enddo
    do it = 1, TT
        call decisions(it)
    enddo

    if(lsra_on)call lsra()

    do it = 1, TT
        call quantities(it)
    enddo
    do it = 1, TT
        call government(it)
    enddo

    ! check for the number of markets in equilibrium
    nmarket = 0
    do it = 1, TT
        if(abs(YY(it)-CC(it)-II(it)- GG(it))/YY(it) < tol) &
            nmarket=nmarket + 1
    enddo
    [.....]
    if(nmarket == TT)exit
    enddo
    [.....]
end subroutine

```

that now consumer prices may change along the transition path. Consequently we now have

$$c_{1,t} = \Psi_{1,t} W_{1,t} \quad \text{with}$$

$$\Psi_{1,t} = \frac{1}{p_t} \left\{ 1 + \beta^\gamma \left(\frac{1}{R_{t+1}^n} \frac{p_{t+1}}{p_t} \right)^{1-\gamma} + \beta^{2\gamma} \left(\frac{1}{R_{t+1}^n R_{t+2}^n} \frac{p_{t+2}}{p_t} \right)^{1-\gamma} \right\}^{-1}$$

$$\text{and } c_{2,t+1} = \left(\beta R_{t+1}^n \frac{p_t}{p_{t+1}} \right)^\gamma c_{1,t}, \quad c_{3,t+2} = \left(\beta R_{t+2}^n \frac{p_{t+1}}{p_{t+2}} \right)^\gamma c_{2,t+1}$$

where $W_{1,t}$ is defined in (6.4). The optimal consumption path of middle age and older cohorts in the initial period, i.e. c_{21} , c_{32} and c_{31} has to be computed separately. We first compute the remaining resources of both older households in period 1, i.e.

$$W_{31} = R_1^n a_{30} + pen_1 \quad \text{and} \quad W_{21} = R_1^n a_{20} + w_1^n + \frac{pen_2}{R_2^n},$$

and then derive their consumption from $c_{31} = W_{31}$,

$$c_{21} = \frac{1}{p_1} \left[1 + \beta^\gamma \left(\frac{1}{R_2^n p_1} \right)^{1-\gamma} \right]^{-1} W_{21} \quad \text{and} \quad c_{32} = (\beta R_2^n p_1 / p_2)^\gamma c_{21}.$$

4. We aggregate individual decisions to quantities $C_t, A_t, Y_t, I_t, G_t, B_t$, and K_t in the subroutine `quantities`. What is different here is that we derive the capital stock K_t from the capital market equation (6.12). Yet, we let the capital stock not fully adjust, but assume that the new capital stock is a linear combination between the capital stock used to determine prices (see point 2) and the capital stock that fully clears the capital market. The speed of adjustment is determined by the parameter `damp`. If this parameter is equal to 1, the capital stock immediately adjusts to completely clear the capital market. If `damp` is small, adjustment takes place only gradually.
5. Knowing aggregated quantities, we determine budget-balancing tax and payroll rates for the government in the subroutine `government` for every period $t \geq 1$.
6. Finally we have to check whether the calculated quantities and prices define a general equilibrium. We do so by looking at the goods market equilibrium condition (6.13). We say that a goods market is in equilibrium if the absolute value of the relative difference between demand and supply is lower than some threshold level `tol`, i.e.

$$\left| \frac{Y_t - C_t - G_t - I_t}{Y_t} \right| < \text{tol}.$$

We can verify for how many markets $t = 1, \dots, T$ this criterion holds.

If all T markets are in equilibrium, we have found an equilibrium transition path and stop the computational process. Yet, if there are some markets that have not yet converged, we use the current values of K_t and tax rates as initial values and start another iteration at point 2. We iterate this process until all markets have sufficiently converged. The speed of convergence depends on the choice of `damp`. In general we prefer values close to 1 for this parameter since this speeds up the convergence process. Yet for large values of `damp` the convergence process tends to be less stable. Therefore, choosing a suitable value for `damp` involves a trade-off between convergence speed and stability. We cancel the iteration process after a maximum of `itermax` iterations. If there are still some markets not in equilibrium, the subroutine will print an error message.

6.2.2 GENERATIONAL WELFARE AND AGGREGATE EFFICIENCY

Next, it is useful to introduce a welfare measure which allows us to compare and interpret the welfare consequences of a certain policy reform for different cohorts living along the transition path and in the new long-run equilibrium. The reference points are the utility levels of the three cohorts that were living in the initial steady state before the reform (i.e. in period 0) $U_{j0}, j = 1, 2, 3$. Conducting a policy reform has an impact on different types of cohorts. We refer to current generations as those who were already making economic decisions in the initial equilibrium and were ‘surprised’ by the policy change during their life cycle. Their respective utility levels are U_{21} and U_{31} after the reform. Future generations, on the other hand, are those who enter their economically relevant phase of life along the transition path. Their utility levels are $U_t, t = 1, \dots, T$. As utility is not cardinally scaled, direct comparison of utility levels would only indicate whether a specific cohort benefits from or is hurt by a specific reform. In order to derive a meaningful quantitative measure, we utilize the homogeneity property of the utility function and derive the relative increase (or decrease) in income necessary to reach the post-reform utility level given pre-reform prices. Figure 6.3 points out the general idea. The pre-reform optimal c_1, c_2 combination given initial resources $W_0 = w^n + \frac{w^n}{R^n}$ and the price p_0 is shown in A, where the utility level is U_0 . The policy reform affects resources and/or prices in period t , so that we can compute W_t and p_t . The equilibrium changes to point B with the new utility level of $U_t > U_0$. Owing to homogeneity of utility, an increase in initial resources W_0 only holding constant prices would not change the relative composition of c_1, c_2 . Consequently the original budget constraint can be shifted upwards until it is tangent to the new indifference curve U_t at point C. At point C the relative increase in goods consumption compared to the initial equilibrium A equals the relative income increase of Δ_t . Consequently, the reform has increased the welfare of the considered household in the same way as an increase of initial resources by Δ_t per cent. Therefore Δ_t is called the income equivalent variation or—since it was introduced by

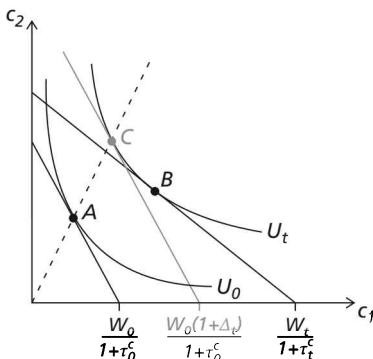


Figure 6.3 Computation of welfare changes

Sir John R. Hicks—the *Hicksian equivalent variation* (HEV). In order to derive Δ_t in practice, we specify

$$\begin{aligned} U_t &= \frac{[c_{10}(1 + \Delta_t)]^{1-1/\gamma}}{1 - 1/\gamma} + \beta \frac{[c_{20}(1 + \Delta_t)]^{1-1/\gamma}}{1 - 1/\gamma} + \beta^2 \frac{[c_{30}(1 + \Delta_t)]^{1-1/\gamma}}{1 - 1/\gamma} \\ &= U_0(1 + \Delta_t)^{1-1/\gamma}, \end{aligned}$$

so that the HEV is given by

$$\Delta_t = \left(\frac{U_t}{U_0} \right)^{\frac{1}{1-1/\gamma}} - 1.$$

The change in welfare expressed as a percentage of initial resources of a cohort is the main indicator of intergenerational welfare consequences of a reform. Typically some cohorts will gain from the reform while others will lose. In Figure 6.4 the \times -path shows such a possible outcome of a policy experiment, where the two older cohorts born in period -1 and 0 are worse off and all cohorts born in period 1 (the year the reform is implemented) and afterwards are better off.

Efficiency effects Yet we would like to go one step beyond the comparison of welfare effects. Specifically, we would like to know whether welfare consequences are only due to intergenerational redistribution or whether the reform has actually improved (or worsened) the resource allocation. In order to evaluate these efficiency effects, we proceed as follows: First, we calculate lump-sum transfers v_{-1} and v_0 needed to be given to current generations to make them as well off as in the initial equilibrium. Second, we calculate transfer payments $v_t, t = 1, \dots, T$ to future generations so as to make them all equally well off, i.e. they all experience the identical utility level U^* . The utility level U^* is determined by the requirement that the present values of all transfer payments add up to zero. The effects of this ‘compensation scheme’ can be seen from the \bullet -path in Figure 6.4. In this situation all current generations would receive lump-sum transfers while future

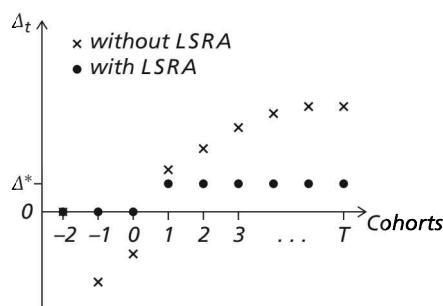


Figure 6.4 Welfare vs. efficiency effects of policy reforms

cohorts would have to pay lump-sum taxes. Future cohorts therefore realize a positive welfare change after compensation, so that

$$\Delta^* = \left(\frac{U^*}{U_0} \right)^{\frac{1}{1-\gamma}} - 1 > 0.$$

Δ^* can be interpreted as a measure of the aggregate efficiency gain of the policy reform as percentage of initial resources. We say that a reform is ‘efficiency improving’ or ‘Pareto superior after compensation’ if $\Delta^* > 0$.

Deriving U^* and therefore Δ^* is not an easy task. We do this by implementing a so-called Lump-Sum Redistribution Authority (LSRA). We allow this authority to pay lump-sum transfers to all generations, current and future. The LSRA thereby proceeds in 3 steps:

1. Compute required transfers that bring older cohorts in the reform year from their current utility level U_{i1} back to their initial equilibrium utilities $U_{i0}, i = 2, 3$. According to the above calculations this can be done by setting

$$\nu_{-1} = \left[\left(\frac{U_{30}}{U_{31}} \right)^{\frac{1}{1-\gamma}} - 1 \right] W_{31} \quad \text{and} \quad \nu_0 = \left[\left(\frac{U_{20}}{U_{21}} \right)^{\frac{1}{1-\gamma}} - 1 \right] W_{21}$$

where W_{21} and W_{31} define the resources of both older households *after* the reform in period 1.

2. Transfers to future cohorts are calculated using the intertemporal budget of the redistribution agency. This budget guarantees that the discounted sum of LSRA-transfers is exactly equal to zero, i.e. the LSRA does not make any profits. Consequently, as of period 1

$$\nu_{-1}N_{-1} + \nu_0N_0 + \nu_1N_1 + \dots + \frac{\nu_T N_T}{\prod_{k=2}^T R_k} + \frac{\nu_T N_T (1 + n_{p,T})}{\prod_{k=2}^T R_k R_T} + \dots = 0$$

must hold. Note that starting in period T the equation sums an infinite geometric series

$$\frac{\nu_T N_T}{\prod_{k=2}^T R_k} \left[1 + \frac{1 + n_{p,T}}{R_T} + \left(\frac{1 + n_{p,T}}{R_T} \right)^2 + \dots \right] = \frac{\nu_T N_T R_T}{\prod_{k=2}^T R_k (r_T - n_{p,T})}$$

so that we have

$$\nu_{-1}N_{-1} + \nu_0N_0 + \nu_1N_1 + \frac{\nu_2 N_2}{R_2} + \dots + \frac{\nu_T N_T R_T}{\prod_{k=2}^T R_k (r_T - n_{p,T})} = 0.$$

Dividing by N_1 and substituting the definition of ν_t yields

$$\frac{\nu_{-1}}{(1+n_{p,0})(1+n_{p,1})} + \frac{\nu_0}{1+n_{p,1}} + \sum_{t=1}^{T-1} \left[\left(\frac{U^*}{U_t} \right)^{\frac{1}{1-\gamma}} - 1 \right] W_t \times \\ \prod_{s=2}^t \frac{1+n_{p,s}}{1+r_s} + \left[\left(\frac{U^*}{U_T} \right)^{\frac{1}{1-\gamma}} - 1 \right] W_T \frac{1+r_T}{r_T - n_{p,T}} \prod_{s=2}^T \frac{1+n_{p,s}}{1+r_s} = 0.$$

which can be solved for U^* if we make sure that utilities are positive by normalizing $\frac{U^*}{U_s} = \frac{(1-1/\gamma)U^*}{(1-1/\gamma)U_s}$ with $s = t, T$.

3. Given U^* the compensating lump-sum transfers for future cohorts can be computed from

$$\nu_t = \left[\left(\frac{U^*}{U_t} \right)^{\frac{1}{1-\gamma}} - 1 \right] W_t \quad \text{for } t = 1, \dots, T.$$

Of course, the compensation payments ν_t have to be included in the respective individual budget constraints when savings and consumption values are computed. In addition, we assume that the LSRA is a regular actor of the model, i.e. we have to take into account that the sum of initial transfers $\nu_{-1}N_{-1} + \nu_0N_0 + \nu_1N_1$ needs to be financed on the capital market. Consequently, debt (or assets) of the agency per capita of workers in period 2 are given by

$$B_2^a = \frac{1}{1+n_{p,2}} \left[\frac{\nu_{-1}}{(1+n_{p,0})(1+n_{p,1})} + \frac{\nu_0}{1+n_{p,1}} + \nu_1 \right].$$

In all future periods $t = 2, \dots, T$ the agency has to pay (or receives) interest and finances lump-sum transfers, so that the periodic budget constraint reads

$$(1+n_{p,t+1})B_{t+1}^a = (1+r_t)B_t^a + \nu_t.$$

In order to account for assets and liabilities of the LSRA the capital market equilibrium condition (6.11) changes to

$$A_t = K_t + B_t + B_t^a \quad \text{for } t = 2, \dots, T.$$

The calculation of LSRA payments is carried out in the subroutine `lsra` shown in Module 6.2m.b. The function `year` is the time manager in our program. It takes a current year `it` as input and derives the year in which an agent currently aged `ij` turns `ijp` years old. It thereby accounts for the fact that we only simulate the economy for `T` periods along the transition. It also governs the steady-state calculations, meaning that whenever the

Module 6.2m.b Computation of LSRA payments

```

subroutine lsra()
[.....]
! calculate utility for each generation
do it = 1, TT
    call utility(it)
enddo

! transfers to old generations
PVI(-1) = Rn(1)*a(3,1) + pen(1) + v(-1)
PVI(0) = Rn(1)*a(2,1) + wn(1) + pen(2)/Rn(2) + v(0)
v(-1) = v(-1) + PVI(-1)*((util(3,0)/util(3,1))** (1d0/egam)-1d0)
v(0) = v(0) + PVI(0) * ((util(2,0)/util(2,1))** (1d0/egam)-1d0)
BA(2) = v(-1)/((1+n_p(0))*(1d0+n_p(1))) + v(0)/(1d0+n_p(1))

! long-run equilibrium
PVI(TT) = wn(TT) + wn(TT)/Rn(TT) + pen(TT)/Rn(TT)**2 + v(TT)
PVV = v(TT)*(1d0+r(TT))/(r(TT)-n_p(TT))
sum1 = PVI(TT)*(1d0+r(TT))/(r(TT)-n_p(TT))
sum2 = PVI(TT)*(util(1,TT)*egam)**(-1d0/egam)*(1d0+r(TT))/ &
(r(TT)-n_p(TT))

! transition path
do it = TT-1, 1, -1
    it1 = year(it, 1, 2)
    it2 = year(it, 1, 3)
    PVI(it) = wn(it)+wn(it1)/Rn(it1)+pen(it2)/ &
(Rn(it1)*Rn(it2))+ v(it)
    PVV = PVV*(1d0+n_p(it1))/(1d0+r(it1)) + v(it)
    sum1 = sum1*(1d0+n_p(it1))/(1d0+r(it1)) + PVI(it)
    sum2 = sum2*(1d0+n_p(it1))/(1d0+r(it1)) + PVI(it) * &
(util(1,it)*egam)**(-1d0/egam)
enddo

! calculate ustarr for future generations
ustar = ((sum1-BA(2)-PVV)/sum2)**(1d0-1/gamma)/(1d0-1/gamma)

! calculate transfers to future generations and debt of LSRA
do it = 1, TT
    v(it) = v(it)+ PVI(it)*((ustar/util(1, it))** (1d0/egam)-1d0)
    if(it == 2)BA(2) = (BA(2) + v(1))/(1d0+n_p(2))
    if(it > 2)BA(it)=((1d0+r(it-1))*BA(it-1)+v(it-1)) &
/(1d0+n_p(it))
enddo

end subroutine

```

input `year it=0`, the return value of the function `year` is also equal to zero. We use the subroutine `utilities` to calculate the utility levels U_{1t} , U_{2t} , and U_{3t} for every year of the transition and store them in the array `u(3, T)`. With this utility levels we compute LSRA payments as described above. Yet, we take payments of the previous iteration as given and only calculate additional transfers needed. Therefore, as the iteration process proceeds additional transfers tend to decrease quite rapidly to zero. Note that the computation of the compensated welfare changes Δ^* involves a complete new simulation of the transition path. In general, however, we do not report macroeconomic data from these reforms but only the measure of efficiency.

6.2.3 COMPREHENSIVE ANALYSIS OF POLICY REFORMS

In this subsection we discuss the same policy reforms as in Section 6.1. However, now we include numerical results for the transition path, the intergenerational welfare effects and the aggregate efficiency consequences.

Alternative tax systems Table 6.2 again considers a switch from consumption towards income taxation. In the right column we report welfare changes for cohorts along the transition path. Owing to space restrictions we only concentrate on the initial cohorts born before period 4 and the cohort living in the new steady-state equilibrium (i.e. period T cohort). Tax rates and economic variables reported in periods 0 and ∞ (long-run effects) are already known from Table 6.1. In addition, we now also quantify the short-run effects of the policy reform in implementation period 1 and along the transition towards the new long-run equilibrium. As reported in Table 6.2, when switching from consumption to income taxation, the income tax rate initially jumps to 21 per cent and then slowly increases towards the long-run value of 24 per cent. Due to the reduction in tax payments, the consumption of cohort-age 3 in the initial year jumps up from 0.42 to 0.48. As a consequence, the oldest cohort experiences a welfare increase of 14.33 per cent of initial resources. Consumption of the middle cohort in period 1 increases slightly from 0.30 to 0.32, but stays constant in the last period of life. As a consequence overall welfare for the cohort born in year 0 increases only by 3.63 per cent. All other cohorts experience welfare losses, since they have to pay higher taxes after the reform. They consequently save less, so that the capital stock decreases, which in turn induces a rise in interest rates and a decline in future wages. In addition, this factor price repercussion hurts future generations to the benefit of those currently living. In the present model, income taxes distort the savings decision whereas time-invariant consumption taxes generate no efficiency effects. Consequently switching from consumption towards income taxes reduces the efficiency of the resource allocation. This is demonstrated in the last line of Table 6.2, which shows that lump-sum compensation to initial cohorts could not generate enough resources to

Table 6.2 From consumption to income taxation

| t | τ^w, τ^r | τ^c | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|------------------|----------|-------|-------|-------|------|------|------|--------|
| -1 | | | | | | | | | 14.33 |
| 0 | 0.00 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | 3.63 |
| 1 | 0.21 | 0.00 | 0.22 | 0.32 | 0.48 | 0.27 | 0.39 | 1.15 | -2.75 |
| 2 | 0.22 | 0.00 | 0.20 | 0.29 | 0.42 | 0.23 | 0.37 | 1.30 | -7.41 |
| 3 | 0.23 | 0.00 | 0.20 | 0.28 | 0.39 | 0.20 | 0.36 | 1.39 | -9.93 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.24 | 0.00 | 0.19 | 0.26 | 0.37 | 0.18 | 0.35 | 1.51 | -12.78 |
| Δ^* | | | | | | | | | -0.25 |

$$g_1 = g_2 = 0.12, \quad g_3 = 0, \quad b_y = 0, \quad \kappa = 0, \quad n_p = 0.2.$$

bring all future cohorts back to the pre-reform utility level. After compensation all future cohorts would experience a welfare reduction of 0.25 per cent.

In the next experiment shown in Table 6.3, we substitute consumption taxes by capital-income taxation. In this simulation the older generations living in the reform year have to pay higher taxes after rather than before the reform. Consequently the oldest and the middle cohort of period 1 experience a significant welfare reduction of 19.93 and 9.89 per cent respectively. Hardly anything happens with respect to macroeconomic variables. The reason is that in this case income and substitution effects balance each other. Younger cohorts benefit from the reduction in tax burdens which would increase their savings. At the same time the increase in the capital-income tax rate induces a substitution effect, reducing savings. Both effects neutralize each other given the present preference structure. As a result, there is only a slight adjustment in consumption but hardly any change in savings, capital stock, and factor prices. Note that this reform reduces efficiency remarkably. The reason for this is that we switch from a lump-sum tax towards a tax system that heavily distorts savings decisions. Therefore the efficiency of the resource allocation decreases by 3.52 per cent of initial resources.

The intergenerational redistribution of welfare changes direction completely when we switch from consumption towards labour-income taxation in Table 6.4. Of course

Table 6.3 From consumption to capital-income taxation

| t | τ^r | τ^c | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|----------|----------|-------|-------|-------|------|------|------|--------|
| -1 | | | | | | | | | -19.93 |
| 0 | 0.00 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | -9.89 |
| 1 | 0.71 | 0.00 | 0.28 | 0.30 | 0.34 | 0.27 | 0.39 | 1.15 | 5.49 |
| 2 | 0.71 | 0.00 | 0.28 | 0.30 | 0.33 | 0.27 | 0.39 | 1.15 | 5.50 |
| 3 | 0.71 | 0.00 | 0.28 | 0.30 | 0.33 | 0.27 | 0.39 | 1.14 | 5.58 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.47 | 0.00 | 0.28 | 0.30 | 0.33 | 0.27 | 0.39 | 1.14 | 5.63 |
| Δ^* | | | | | | | | | -3.52 |

$$g_1 = g_2 = 0.12, \quad g_3 = 0, \quad b_y = 0, \quad \kappa = 0, \quad n_p = 0.2.$$

Table 6.4 From consumption to labour-income taxation

| t | τ^w | τ^c | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|----------|----------|-------|-------|-------|------|------|------|--------|
| -1 | | | | | | | | | 29.02 |
| 0 | 0.00 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | 9.09 |
| 1 | 0.30 | 0.00 | 0.19 | 0.32 | 0.57 | 0.27 | 0.39 | 1.15 | -8.36 |
| 2 | 0.33 | 0.00 | 0.17 | 0.28 | 0.47 | 0.21 | 0.36 | 1.38 | -15.63 |
| 3 | 0.35 | 0.00 | 0.16 | 0.26 | 0.42 | 0.18 | 0.35 | 1.55 | -20.10 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.37 | 0.00 | 0.15 | 0.23 | 0.37 | 0.14 | 0.32 | 1.83 | -26.20 |
| Δ^* | | | | | | | | | 0.00 |

$$g_1 = g_2 = 0.12, \quad g_3 = 0, \quad b_y = 0, \quad \kappa = 0, \quad n_p = 0.2.$$

now welfare of the initial cohorts increases much more than in Table 6.2, since they don't have to pay taxes on accumulated capital income. At the same time, newborn and future cohorts lose more, as higher tax rates increases their tax burden compared to the previous simulation. Again, changes in tax burdens reduce savings and capital accumulation during the transition, so that wages decrease and interest rates rise. Note however that compensation reveals that the reform induces no efficiency effects. In the present model consumption and labour-income taxes are both lump-sum, i.e. they generate no efficiency losses. This results from labour supply being completely inelastic (fixed). Consequently transitional and long-run welfare losses shown in Table 6.4 are only due to intergenerational redistribution. It is possible to compensate this redistribution via lump-sum transfers in a way that guarantees all cohorts to end up at their initial utility level.

Next we want to see what happens when we combine the switch towards labour-income taxation with an increase in public savings. As shown in Table 6.5 the increase in public savings leads to the new long-run equilibrium being the same as the initial steady state. In the initial reform year, labour income tax rates have to increase much higher than in Table 6.4 in order to finance the build-up of public assets. These assets generate interest income for the government budget in future periods. The adjusted intertemporal tax structure induces additional intergenerational redistribution effects, which completely neutralize the effects from the original change in tax structure in the long run. However, as shown in Table 6.5, the introduction of public savings cannot neutralize any intergenerational welfare change. While the oldest cohort is as well off from the reform as in Table 6.4, middle-aged and newborns lose much more now. Furthermore, future generations are significantly better off. Of course, since public debt only induces intergenerational income effects and no substitution effects, aggregate efficiency effects are again zero.

Public pensions and debt Next, we introduce in Table 6.6 an unfunded pension system giving the oldest cohort a benefit of 50 per cent of previous wages. It should not be surprising that this policy increase the welfare of the two oldest cohorts. They have not

Table 6.5 From consumption to labour-income taxation with public savings

| <i>t</i> | <i>b_y</i> | τ^w | τ^c | <i>c₁</i> | <i>c₂</i> | <i>c₃</i> | <i>K</i> | <i>w</i> | <i>r</i> | <i>HEV</i> |
|------------|----------------------|----------|----------|----------------------|----------------------|----------------------|----------|----------|----------|------------|
| -1 | | | | | | | | | | 29.02 |
| 0 | 0.00 | 0.00 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | -0.79 |
| 1 | -0.06 | 0.40 | 0.00 | 0.18 | 0.30 | 0.54 | 0.27 | 0.39 | 1.15 | -15.60 |
| 2 | -0.06 | 0.23 | 0.00 | 0.21 | 0.26 | 0.42 | 0.23 | 0.38 | 1.27 | -3.35 |
| 3 | -0.06 | 0.23 | 0.00 | 0.21 | 0.29 | 0.36 | 0.24 | 0.38 | 1.24 | -2.62 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ∞ | -0.06 | 0.22 | 0.00 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | 0.00 |
| Δ^* | | | | | | | | | | 0.00 |

$$g_1 = g_2 = 0.12, \quad g_3 = 0, \quad \kappa = 0, \quad n_p = 0.2.$$

Table 6.6 Introduction of pay-as-you-go pensions

| t | κ | τ^P | τ^C | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|----------|----------|----------|-------|-------|-------|------|------|------|--------|
| -1 | | | | | | | | | | 40.12 |
| 0 | 0.00 | 0.00 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | 7.54 |
| 1 | 0.50 | 0.19 | 0.26 | 0.18 | 0.32 | 0.59 | 0.27 | 0.39 | 1.15 | -10.91 |
| 2 | 0.50 | 0.21 | 0.31 | 0.16 | 0.27 | 0.46 | 0.19 | 0.35 | 1.48 | -20.98 |
| 3 | 0.50 | 0.20 | 0.34 | 0.15 | 0.24 | 0.41 | 0.16 | 0.33 | 1.68 | -24.36 |
| : | : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.50 | 0.19 | 0.37 | 0.14 | 0.23 | 0.37 | 0.14 | 0.32 | 1.85 | -26.68 |
| Δ^* | | | | | | | | | | 0.00 |

$g_1 = g_2 = 0.12$, $g_3 = 0$, $b_y = 0$, $n_p = 0.2$.

Table 6.7 Introduction of public debt

| t | b_y | τ^C | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|-------|----------|-------|-------|-------|------|------|------|--------|
| -1 | | | | | | | | | 14.31 |
| 0 | 0.000 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | 7.49 |
| 1 | 0.10 | 0.13 | 0.23 | 0.33 | 0.48 | 0.27 | 0.39 | 1.15 | 2.42 |
| 2 | 0.10 | 0.47 | 0.17 | 0.29 | 0.43 | 0.20 | 0.36 | 1.40 | -15.43 |
| 3 | 0.10 | 0.50 | 0.16 | 0.26 | 0.44 | 0.18 | 0.35 | 1.53 | -19.11 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.10 | 0.60 | 0.14 | 0.23 | 0.37 | 0.14 | 0.32 | 1.85 | -26.68 |
| Δ^* | | | | | | | | | -0.27 |

$g_1 = g_2 = 0.12$, $g_3 = 0$, $\kappa = 0$, $n_p = 0.2$.

paid anything (or only partially) into the system, but they receive full benefits. Younger and future cohorts pay taxes and receive benefits, but net tax payments are positive (i.e. tax payments are higher than benefits received). Consequently, younger cohorts save less inducing the capital stock to decrease and factor prices to adjust accordingly. Again, since we do not consider the labour-supply decisions of households, intergenerational welfare effects of the unfunded pension scheme are only due to redistribution. As shown in the last line of Table 6.6, overall efficiency effects are zero after compensation.

The following simulation reported in Table 6.7 demonstrates that the introduction of public debt has very similar economic consequences as the introduction of an unfunded pension system. Since consumption tax rates can temporarily be reduced from 29 to 13 per cent, initial older and middle-aged cohorts experience a welfare increase of 14.31 and 7.49 per cent. In the second period, however, consumption tax rates need to rise even beyond initial levels since tax revenues now have to cover interest payments on public debt. The increase in future consumption tax rates reduces individual savings, capital accumulation and future wages, while interest rates increase. Consequently future cohorts experience a welfare reduction of 26.68 per cent. Overall efficiency now declines

by 0.27 per cent. This is due to the fact that after LSRA compensation consumption taxes vary over time, which is in contrast to what happened in Table 6.6.

Analysis of fertility decline In Table 6.8 we consider the economic effects of a permanent fertility decline that reduces population growth from 0.2 to zero per cent. The adjustment in population growth rates induces a change in the population structure. This increases the share of pensioners and reduces the share of working-age cohorts in the total population. With respect to the economic consequences we have to identify the effects on public budgets and on factor prices. With respect to the latter we now observe that the capital stock already adjusts in period 1. Since fewer new workers enter the labour market than before, the capital stock per capita increases immediately, which in turn increases wages and reduces interest rates. The immediate reduction of the interest rate hurts pensioners while the wage increase is beneficial for all working-age cohorts. Concerning the public budget we have to keep in mind that public goods demand is constant per capita of workers in the present setup. Since the number of workers decreases, public goods demand decreases as well, so that consumption taxes fall throughout the transition. The reduction of consumption taxes is beneficial for all cohorts. However, as shown in the last column of Table 6.8 the (negative) factor price effect initially dominates the (positive) tax rate reduction. Hence, pensioners experience a decline in welfare while all other cohorts benefit from the fertility decline. As shown in the last line of Table 6.8, overall efficiency rises by 3.90 per cent due to the permanent tax reduction.

Finally, in Table 6.9 we consider the reduction in fertility when the government sector runs an unfunded pension system. Whereas in the previous simulation the population aging reduces public goods demand and consumption taxes, it now increases the payroll taxes required to finance pension benefits. The new population structure implies an increase in the payroll tax rate from 19 per cent to 25 per cent in the long run. The rising payroll tax rate distorts savings so that the capital stock increases as well as wages. Older cohorts lose due to lower interest rates, young and future cohorts lose due to higher payroll taxes. When existing cohorts are compensated, the distortion of savings reduces aggregate efficiency by 6.23 per cent of resources.

Table 6.8 Permanent fertility decline

| t | n_p | τ^c | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|-------|----------|-------|-------|-------|------|------|------|-------|
| -1 | | | | | | | | | -2.32 |
| 0 | 0.20 | 0.29 | 0.22 | 0.30 | 0.42 | 0.27 | 0.39 | 1.15 | -0.93 |
| 1 | 0.00 | 0.28 | 0.23 | 0.30 | 0.41 | 0.32 | 0.41 | 1.07 | 2.89 |
| 2 | 0.00 | 0.25 | 0.24 | 0.31 | 0.41 | 0.37 | 0.42 | 0.97 | 6.74 |
| 3 | 0.00 | 0.25 | 0.24 | 0.32 | 0.41 | 0.38 | 0.43 | 0.95 | 7.77 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ∞ | 0.00 | 0.24 | 0.25 | 0.32 | 0.43 | 0.40 | 0.43 | 0.92 | 9.12 |
| Δ^* | | | | | | | | | 3.90 |

$$g_1 = g_2 = 0.12, \quad g_3 = 0, \quad b_y = \kappa = 0.$$

Table 6.9 Permanent fertility decline with pay-as-you-go pensions

| t | κ | n_p | τ^P | c_1 | c_2 | c_3 | K | w | r | HEV |
|------------|----------|-------|----------|-------|-------|-------|------|------|------|-------|
| -1 | | | | | | | | | | -2.86 |
| 0 | 0.50 | 0.20 | 0.19 | 0.20 | 0.32 | 0.51 | 0.14 | 0.32 | 1.85 | -2.72 |
| 1 | 0.50 | 0.00 | 0.20 | 0.20 | 0.31 | 0.49 | 0.16 | 0.33 | 1.73 | -1.27 |
| 2 | 0.50 | 0.00 | 0.24 | 0.20 | 0.31 | 0.48 | 0.18 | 0.34 | 1.60 | -2.72 |
| 3 | 0.50 | 0.00 | 0.25 | 0.19 | 0.30 | 0.47 | 0.18 | 0.34 | 1.64 | -4.37 |
| : | : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.50 | 0.00 | 0.25 | 0.19 | 0.30 | 0.46 | 0.17 | 0.33 | 1.69 | -4.97 |
| Δ^* | | | | | | | | | | -6.23 |

$$g_1 = g_2 = g_3 = 0, \quad b_y = 0.$$

Summing up the simulations of this section, we find that our model is able to analyse and quantify the intertemporal effects of a broad range of governmental policies. Note, however, that our main concern in this chapter is to focus on the difference between static and dynamic general equilibrium models in one of the most simple economic environments. Since simplicity does not come without cost, the numerical results of this section should not be taken too seriously. In reality, for example, distortions of labour supply are a central policy concern and considering a model with endogenous labour supply would alter our results considerably. Such (realistic) extensions of the model will be discussed in Chapter 7.

6.3 Further reading

Auerbach and Kotlikoff (1987) were probably the first to simulate a dynamic general equilibrium model with 55 overlapping generations. They discuss many of the above policy reforms and also introduce a LSRA mechanism in order to separate welfare from aggregate efficiency effects. Many textbooks discuss dynamic macroeconomic issues using the overlapping generations approach. For example, McCandless and Wallace (1991) provide a good introduction.

6.4 Exercises

- 6.1. Program 6.2 is not well designed, as it uses different numerical approaches for the initial equilibrium (`fzero`) and for the transition path (Gauss-Seidel). A clearer arrangement would also apply the Gauss-Seidel to solve the initial equilibrium. Develop two subroutines `initialize` and `get_steadystate`. The first should

initialize all model-relevant variables. The second computes the initial steady-state equilibrium using a Gauss-Seidel iteration. Hence, your main program changes to

```
program TR_OLG
[.....]
! initialize variables and government parameters
call initialize

! compute initial long-run equilibrium
call get_SteadyState

! write output
call output(0, 1)

! compute transitional values
call get_Transition
[.....]
end program
```

Make sure that your new code gives the same results as the previous one.

- 6.2. Use the new program with transitional dynamics from the previous exercise. The initial equilibrium remains unaltered. However, assume that the policy reform in year 1 is introduced in a small open economy. Consequently factor prices do not change any more (i.e. $r_t = r_0, w_t = w_0$) as immediate capital flows from abroad keep the interest rate constant. Proceed in the following steps:
- Adjust the derivation of the capital stock so that $k_t = f'^{-1}(r_0)$. Furthermore adapt the goods market and the asset market equilibrium condition, accounting for the fact that the trade balance TB_t and the service balance $r_0 B_t^f$ have to offset changes in the net foreign asset position $(1 + n_{p,t})B_{t+1}^f - B_t^f$. Introduce the logical variable `slope`, which allows us to easily simulate and compare the same policy reforms in closed and small open economies. Note that the output file should report the trade balance and the net foreign asset position.
 - Simulate the move from consumption to income taxation (Table 6.2) in a small open economy and explain the differences compared to a closed economy.
- 6.3. Introduce a corporate tax that is computed as $T_{k,t} = \tau^k[y_t - w_t - \delta k_t - \epsilon r_t k_t]$. Accounting for corporate taxation, the first-order condition (6.8) from profit maximization of companies changes to

$$r_t = \frac{1 - \tau_k}{1 - \epsilon \tau_k} (f'(k_t) - \delta).$$

Implement the corporate tax system in your model.

- Assume a benchmark economy that only features consumption taxation. Compare the introduction of a corporate tax with $\tau_k = 0.2$ and $\epsilon = 0$ in a closed and in a small open economy. Explain the macroeconomic effects, the welfare changes, and the efficiency consequences in detail.

- b) Now start from an initial equilibrium that already includes a corporate tax with $\tau^k = 0.2$ and $\epsilon = 0$. Introduce an allowance of imputed interest cost (i.e. $\epsilon = 1$) in the closed economy and the small open economy. Explain your results in economic terms.
- 6.4. Simulate a move from consumption to wage-income taxation with an implementation lag of one period. Hence, households already know at the beginning of period 1 that the tax system will change from consumption towards wage taxation in period 2. How do households respond with their intertemporal consumption pattern? Explain the differences to Table 6.4 in economic terms.
- 6.5. Assume that $g_1 = g_2 = 0$ and $g_3 = 0.24$. Replicate the fertility decline from $n_p = 0.2$ to $n_p = 0$ and compare the results when public goods are financed by consumption taxes. Can you explain the difference compared to Table 6.8?
- 6.6. Assume that households who are born in period t live up to J years and retire at age j_r . The government runs a tax system with income and consumption taxes and a pay-as-you-go flat pension system. Public goods are provided as a fixed fraction g_y of GDP in the initial equilibrium and then held constant in per capita terms.
- Derive the optimization problem, the optimality conditions as well as the aggregation equations on the household side.
 - Extend the baseline model by writing two functions `get_W(ij, it)` and `get_Psi(ij, it)` as well as a subroutine `get_path(ij, it)`. The functions should compute the available resources and the marginal consumption of a household at age ij in year it . The subroutine derives the remaining life-cycle consumption and asset path for such a household. Adapt all other subroutines such that they account for the more general definition of the life-cycle dates J and j_r . Set $J = 3$ and $j_r = 3$ and check whether your program returns the same results as in Section 6.2.3. Note: Set $g_y = 0.2132$ in order to get the same public good level.
 - Increase the number of periods to $J = 12$ and $j_r = 9$ and/or $J = 55$ and $j_r = 41$. Assume that the human-capital profile is hump-shaped

$$h_j = 1 + 0.05j - 0.001j^2.$$

Is there anything else you need to adjust?

Simulate a move from consumption to income taxation and compare your results with those of Table 6.3.

7 Extending the OLG model

In Chapter 6 we used our basic OLG model to discuss the welfare and efficiency effects of various policy reforms. Of course, we have to be cautious in drawing robust conclusions from such a policy analysis. In the basic model households only decide on their intertemporal consumption allocation. Hence, public policy solely distorts the savings decision and, consequently, most of the policy reforms hardly impact on economic efficiency but only redistribute across cohorts. Our analysis could be much more instructive when decisions of economic agents are multidimensional, so that various distortions induced by public policy interact. In this chapter we therefore introduce an extended individual decision process. Households not only decide on their savings, but also on their time use. Given a specific time endowment (say a day or a year), agents can either work in the market (and earn income), go to school (and acquire human capital for future income generation), or consume leisure. Public policy may distort all of these decisions. A good policy thus has to create a balance between intertemporal and intratemporal distortions. Finally, we study the implications of lifespan uncertainty and missing annuity markets, asking how public policy can improve the allocation of resources by providing insurance against longevity risk.

7.1 Accounting for variable labour supply

In this section we allow households to decide how many hours to work in each period. The remaining time is used for leisure consumption which now features in household utility. Leisure demand in each period of the life cycle strongly depends on the respective value of human capital h_j , which measures the value of the time endowment in terms of labour market productivity. Hence agents may work the same number of hours, but they may be differently productive, so that they earn a different wage per time unit. Whenever the wage a household earns in the labour market is very small, the household might want to consume more leisure than the actual time endowment. In order to guarantee that the time endowment is met, we calculate a so-called *shadow wage* $\mu_{j,s}$. The shadow wage is added to the regular wage of the household and calculated such that the household's optimal decision consists in consuming the household's total endowment of time as leisure. As this also means that the agent provides zero working hours to the market, the shadow wage is a fictitious wages that will never be paid to the household.

7.1.1 THE HOUSEHOLD DECISION PROBLEM

Households now have to decide how much to consume and save in each period and how to split up their total time endowment of one unit between working time and leisure $\ell_{j,s}$. The dynamic budget constraints of the individual entering the labour market in period t now change to

$$p_t c_{1,t} = (1 - \ell_{1,t}) w_{1,t}^n - a_{2,t+1} \quad (7.1)$$

$$p_{t+1} c_{2,t+1} = R_{t+1}^n a_{2,t+1} + (1 - \ell_{2,t+1}) w_{2,t+1}^n - a_{3,t+2} \quad (7.2)$$

$$p_{t+2} c_{2,t+2} = R_{t+2}^n a_{3,t+2} + (1 - \ell_{3,t+2}) w_{3,t+2}^n + pen_{t+2} \quad (7.3)$$

where

$$w_{j,s}^n = [w_s h_j + \mu_{j,s}] (1 - \tau_s^w - \tau_s^p)$$

defines the net wage of a j -year-old worker in period $s = t + j - 1$. w_s denotes the gross wage per unit of human capital. Households of different ages may earn more or less *per hour* because of differences in human-capital endowments. Thus $w_s h_j$ may be interpreted as the individual's gross wage rate. In the following the human-capital profile h_1, h_2, h_3 is set exogenously and is separate from the general level of wages w_s , the time path of which is determined endogenously in the model. If, for a given wage level, leisure demand $\ell_{j,s}$ exceeds the total time endowment of unity, we compute a shadow wage rate $\mu_{j,s}$ that reduces leisure demand exactly to the time endowment. If $\ell_{j,s}$ is less than the time endowment, then $\mu_{j,s}$ is zero, i.e.

$$\ell_{j,s} \leq 1, \quad \mu_{j,s} \geq 0 \quad \text{and} \quad \mu_{j,s}(1 - \ell_{j,s}) = 0.$$

The last condition again is a complementarity constraint, as we discussed before in Chapter 4.

Households born in period t maximize the utility function

$$U_t = U(c_{1,t}, \ell_{1,t}, \dots) = \sum_{j=1}^3 \beta^{j-1} u(c_{j,s}, \ell_{j,s}) \quad \text{with} \quad s = t + j - 1$$

subject to the intertemporal budget constraint

$$\begin{aligned} p_t c_{1,t} + w_{1,t}^n \ell_{1,t} + \frac{p_{t+1} c_{2,t+1} + w_{2,t+1}^n \ell_{2,t+1}}{R_{t+1}^n} + \frac{p_{t+2} c_{3,t+2} + w_{3,t+2}^n \ell_{3,t+2}}{R_{t+1}^n R_{t+2}^n} \\ = w_{1,t}^n + \frac{w_{2,t+1}^n}{R_{t+1}^n} + \frac{w_{3,t+2}^n + pen_{t+2}}{R_{t+1}^n R_{t+2}^n} =: W_{1,t}, \end{aligned} \quad (7.4)$$

which is again derived from the dynamic budget constraints (7.1) to (7.3). The resulting first-order conditions

$$w_{1,t}^n u_{c_1}(c_{1,t}, \ell_{1,t}) = p_t u_{\ell_1}(c_{1,t}, \ell_{1,t}) \quad (7.5)$$

$$w_{2,t+1}^n u_{c_2}(c_{2,t+1}, \ell_{2,t+1}) = p_{t+1} u_{\ell_2}(c_{2,t+1}, \ell_{2,t+1}) \quad (7.6)$$

$$w_{3,t+2}^n u_{c_3}(c_{3,t+2}, \ell_{3,t+2}) = p_{t+2} u_{\ell_3}(c_{3,t+2}, \ell_{3,t+2}) \quad (7.7)$$

$$p_{t+1} u_{c_1}(c_{1,t}, \ell_{1,t}) = \beta R_{t+1}^n p_t u_{c_2}(c_{2,t+1}, \ell_{2,t+1}) \quad (7.8)$$

$$p_{t+2} u_{c_1}(c_{1,t}, \ell_{1,t}) = \beta^2 R_{t+1}^n R_{t+2}^n p_t u_{c_3}(c_{3,t+2}, \ell_{3,t+2}), \quad (7.9)$$

where $u_{c_j} = \frac{\partial u(\cdot)}{\partial c_j}$ and $u_{\ell_j} = \frac{\partial u(\cdot)}{\partial \ell_j}$ together with the constraint (7.1.1) define the six consumption and leisure demand functions

$$c_{j,s} = C_j(w_{1,t}^n, w_{2,t+1}^n, w_{3,t+2}^n, p_{t+2}, R_{t+1}^n, R_{t+2}^n, p_t, p_{t+1}, p_{t+2}) \text{ and } \ell_{j,s} = L_j(\cdot)$$

for the cohort ‘born’ in period t . As before, the savings functions $a_{2,t+1}, a_{3,t+2}$ are then derived by plugging the consumption and leisure demand functions into the budget constraints (7.1) and (7.2). Finally, aggregate labour supply in period t —now measured in units of human capital—is computed from

$$\begin{aligned} L_t &= \frac{1}{N_t} [h_1(1 - \ell_{1,t})N_t + h_2(1 - \ell_{2,t})N_{t-1} + h_3(1 - \ell_{3,t})N_{t-2}] \\ &= \sum_{j=1}^3 h_j(1 - \ell_{j,t})m_{j,t} \end{aligned} \quad (7.10)$$

with $m_{1,t} = 1$ and $m_{j,t} = \frac{m_{j-1,t-1}}{1+n_{p,t}}$.

7.1.2 FUNCTIONAL FORMS AND NUMERICAL IMPLEMENTATION

The utility function in a specific period s is given by

$$u(c_{j,s}, \ell_{j,s}) = \frac{1}{1 - \frac{1}{\gamma}} \left[c_{j,s}^{1-1/\rho} + \nu \ell_{j,s}^{1-1/\rho} \right]^{\frac{1-1/\gamma}{1-1/\rho}}, \quad (7.11)$$

where γ again denotes the intertemporal elasticity of substitution between consumption in different years and ρ defines the intratemporal elasticity of substitution between consumption and leisure. The leisure preference parameter ν is used to calibrate a realistic fraction of working time. From the first-order conditions (7.5) to (7.7) we derive

$$\ell_{j,s} = \left(\frac{w_{j,s}^n}{\nu p_s} \right)^{-\rho} c_{j,s} \quad (7.12)$$

for $j = 1, 2, 3$. Substituting (7.12) into the first order conditions (7.8) and (7.9) yields

$$c_{2,t+1} = \frac{\nu_{2,t+1}}{\nu_{1,t}} \left[\beta R_{t+1}^n \frac{p_t}{p_{t+1}} \right]^\gamma c_{1,t} \quad \text{and} \quad (7.13)$$

$$c_{3,t+2} = \frac{\nu_{3,t+2}}{\nu_{1,t}} \left[\beta^2 R_{t+1}^n R_{t+2}^n \frac{p_t}{p_{t+2}} \right]^\gamma c_{1,t} \quad (7.14)$$

with $\nu_{j,s} = \left[1 + \nu^\rho \left(\frac{w_{j,s}^n}{p_s} \right)^{1-\rho} \right]^{\frac{\rho-\gamma}{1-\rho}}$. Substituting (7.12) into the budget constraint (7.1.1) we obtain

$$p_t v_{1,t}^{\frac{1-\rho}{\rho-\gamma}} \cdot c_{1,t} + \frac{p_{t+1} \nu_{2,t+1}^{\frac{1-\rho}{\rho-\gamma}}}{R_{t+1}^n} \cdot c_{2,t+1} + \frac{p_{t+2} \nu_{3,t+2}^{\frac{1-\rho}{\rho-\gamma}}}{R_{t+1}^n R_{t+2}^n} \cdot c_{3,t+2} = W_{1,t}. \quad (7.15)$$

Finally, substituting the first-order conditions (7.13) into (7.15) we get

$$c_{1,t} = \Psi_{1,t} W_{1,t} \quad \text{with}$$

$$\Psi_{1,t} = \frac{\nu_{1,t}}{p_t} \left\{ \sum_{j=1}^3 \beta^{\gamma(j-1)} \left[\frac{p_s}{p_t} \Pi_{k=t+1}^s (R_k^n)^{-1} \right]^{1-\gamma} \nu_{j,s}^{\frac{1-\rho}{\rho-\gamma}} \right\}^{-1}, \quad (7.16)$$

where we again have set $s = t + j - 1$. Given $c_{1,t}$, we can derive the optimal path for consumption $c_{2,t+1}, c_{3,t+2}$ and leisure $\ell_{1,t}, \ell_{2,t+1}, \ell_{3,t+2}$ using the first order conditions (7.12) and (7.13) above. As in the programs of Chapter 6, we can calculate the levels of consumption and leisure demand of the two cohorts already living in the first period of the transition with adjusted budget constraints and first-order conditions.

Numerical implementation Compared to the programs presented in Chapter 6, we make a couple of adjustments. First of all, we do not solve for the initial steady state using the subroutine `fzero` from the toolbox. Instead, we apply a Gauss-Seidel algorithm to solve both the initial equilibrium and the transition path. As a result, we don't have to store functions and subroutines in a module, but can move everything into the main program code. We also add a subroutine `initialize`, in which we specify policy parameters, derive the population aggregator variable $m_{j,s}$, and initialize other variables of the model. From here on, we assume that government consumption equals a fixed fraction `gy` of output in the initial steady state and stays constant throughout the transition.

In order to include variable labour supply in our model, we need to specify additional parameters and variables. Since we got rid of our module, parameters and variables are

defined at the beginning of the main program as shown in Program 7.1. Compared to Program 6.2, we now specify the maximum age JJ and a retirement age JR in variables with the parameter attribute, which allows us to vary the dimension of the model with only minor adjustments. There are two additional parameters ν and ρ that govern the preference for leisure as well as the elasticity of substitution between market and leisure consumption, respectively. Throughout the declaration of model variables, we furthermore define variables for human capital h_j , leisure $\ell_{j,s}$, and shadow wages $\mu_{j,s}$. We also distinguish a net wage $w_{j,s}^n$ for each age j from the gross wage rate w_s .

The necessary adjustments we need to make to the household decision problem all can be found in the subroutine `decisions(it)`. We thereby use the structure introduced in the exercise part of Section 7.1.1. Function `get_Psi(ij, it)` computes the marginal consumption $\Psi_{j,t}$ as defined in (7.16), while function `get_W(ij, it)` derives the available resources $W_{j,t}$ of the respective household as defined in (7.1.1). Given initial consumption in period t , subroutine `get_path(ij, it)` computes the optimal time path for consumption and leisure in future periods using equations (7.12) and (7.13).

Program 7.1 The global variables for variable labour supply

```
program TRVL_OLG
    implicit none

    ! model parameters
    integer, parameter :: TT      = 24
    integer, parameter :: JJ      = 3
    integer, parameter :: JR      = 3
    [.....]
    real*8, parameter :: nu      = 1.5d0
    real*8, parameter :: rho     = 0.6d0
    [.....]
end program
```

Program 7.1.a Computation of household decisions with variable labour supply

```
subroutine decisions(it)
    [.....]
    ! solve cohort that just entered the economy
    c(1, it) = get_Psi(1, it)*get_W(1, it)
    call get_path(1, it)
    if(nu > 0d0) call shadw(1, it)

    ! derive behaviour for all other cohorts in year 1 of transition
    if(it == 1)then
        do ij = 2, JJ
            c(ij, it) = get_Psi(ij, it)*get_W(ij, it)
            call get_path(ij, it)
            if(nu > 0d0) call shadw(ij, it)
        enddo
    endif

end subroutine
```

Whenever there is a positive weight of leisure in utility ($\nu > 0$), we call the subroutine `shadw(ij, it)`. This subroutine checks whether leisure demand is below the time endowment. If the time constraint is satisfied, `shadw(ij, it)` sets the shadow wage $\mu_{j,s}$ to zero. Whenever leisure demand exceeds the time endowment of unity, the shadow wage reduces leisure demand exactly to the time endowment. Consequently, whenever $\ell_{j,s} > 1$ we compute the shadow wage as

$$\mu_{j,s} = \frac{\nu p_s c_{j,s}^{1/\rho}}{1 - \tau_s^w - \tau_s^p} - w_s h_j.$$

A final adjustment has to be made in subroutine `quantities`, where total labour supply is computed according to (7.10).

7.1.3 SIMULATION RESULTS AND ECONOMIC INTERPRETATIONS

Before we simulate policy reforms with Program 7.1 it is important to understand the impact of the human-capital profile h_i on individual decisions and macroeconomic aggregates. Table 7.1 compares the impact of four specific human-capital profiles on the long-run equilibrium. In practice, these profiles are either estimated from micro data or they are endogenously determined as discussed in Section 7.2. For simplicity we exclude governmental activity in this table. The benchmark of simulation (0) in the first row considers a completely flat human-capital profile. Since in equilibrium $\beta R > 1$, consumption as well as leisure demand increases over the life cycle. Although individuals work in their last period of life, they also have to save in the first two periods in order to finance their rising consumption demand. As a consequence the capital stock is positive and the interest rate is fairly low.

Simulation (1) assumes the same lifetime human-capital endowment, but now it is increasing with age. Consumption of goods and leisure time are still rising over the life cycle, but now households save significantly less, since they receive higher income from working in the last period of life. The resulting lower capital stock reduces the wage rate and increases the interest rate compared to the benchmark. As a result, individual welfare decreases in the last column. Simulation (2) assumes that human capital decreases

Table 7.1 Long-run analysis of human-capital profile

| | h_1 | h_2 | h_3 | ℓ_1 | ℓ_2 | ℓ_3 | K | μ_3 | w | r | U |
|-----|-------|-------|-------|----------|----------|----------|------|---------|------|------|--------|
| (0) | 1.0 | 1.0 | 1.0 | 0.37 | 0.54 | 0.78 | 0.14 | 0.00 | 0.37 | 1.35 | -31.79 |
| (1) | 0.5 | 1.0 | 1.5 | 0.33 | 0.43 | 0.68 | 0.04 | 0.00 | 0.25 | 3.27 | -47.15 |
| (2) | 1.5 | 1.0 | 0.5 | 0.38 | 0.61 | 1.00 | 0.29 | 0.07 | 0.45 | 0.82 | -25.87 |
| (3) | 0.5 | 2.0 | 0.5 | 0.40 | 0.36 | 1.00 | 0.05 | 0.13 | 0.25 | 3.21 | -40.33 |

$$\gamma = 0.5, \beta = 0.9, \nu = 1.5, \rho = 0.6, \alpha = 0.3, \delta = 0, g_y = 0, n_p = 0.2.$$

over the life cycle. Consequently people save more initially and consume more in later life. Since now leisure consumption is restricted by the time endowment, the shadow wage turns positive. Higher savings increase the capital stock so that the wage rises and the interest rate declines. As a consequence individual welfare increases. Simulation (3) finally assumes a hump-shape profile for individual productivity. Households work (i.e. consume less leisure) now especially in the second period when productivity is high. Savings are significantly lower again so that the wage decreases and the interest rate rises again compared to the previous simulation.

Tax and pension reform analysis With fixed labour supply, a move from consumption to wage taxation and the introduction of an unfunded social security system had no efficiency consequences, see Tables 6.4 and 6.6 in the previous chapter. With variable labour supply, however, consumption as well as labour taxes now are distortionary, see the first-order condition (7.12). We first replicate the tax reform of Table 6.4 with Program 7.1 by setting $v = 0, g_y = 0.195$ and $h_1 = h_2 = 1, h_3 = 0$. If we now only set $v = 1.5$, goods consumption would be reduced substantially, so that an extremely high consumption tax is required to finance public expenditure. In order to make our model more comparable with the previous simulations, we therefore increase human capital in the first two periods to $h_1 = h_2 = 2$. Table 7.2 reports the resulting consequences when we move from a consumption tax equilibrium to a labour-income tax.

As in Table 6.4 the switch from consumption to labour-income taxation redistributes from currently young and future cohorts towards older cohorts of the first period. Welfare losses increase throughout the transition due to the crowding out of capital and the induced fall in gross wages. Note that aggregate labour supply only temporarily decreases but then increases again so that labour input is higher in the new long-run equilibrium with wage taxation. Of course, the latter is due to negative income effects for future cohorts which reduces their leisure demand. Due to higher labour-supply distortions, the policy reform now generates an efficiency loss of 0.5 per cent of aggregate resources as reported in the last line of Table 7.2. The efficiency loss is not

Table 7.2 From consumption to labour-income taxation with variable labour supply

| t | τ^w | τ^c | C | L | K | w | r | HEV |
|------------|----------|----------|------|------|------|------|------|--------|
| -1 | | | | | | | | 12.15 |
| 0 | 0.00 | 0.26 | 0.85 | 1.91 | 0.33 | 0.41 | 1.02 | 5.58 |
| 1 | 0.28 | 0.00 | 0.91 | 1.86 | 0.33 | 0.42 | 1.00 | -3.32 |
| 2 | 0.30 | 0.00 | 0.82 | 1.91 | 0.26 | 0.39 | 1.19 | -6.64 |
| 3 | 0.31 | 0.00 | 0.77 | 1.93 | 0.24 | 0.37 | 1.31 | -8.32 |
| : | : | : | : | : | : | : | : | : |
| ∞ | 0.32 | 0.00 | 0.73 | 1.95 | 0.21 | 0.36 | 1.43 | -10.06 |
| Δ^* | | | | | | | | -0.50 |

$$g_y = 0.195, h_1 = h_2 = 2, h_3 = 0, v = 1.5, \rho = 0.6, n_p = 0.2.$$

Table 7.3 Introduction of pay-as-you-go pensions with variable labour supply

| t | κ | τ^P | τ^C | C | L | K | w | r | HEV |
|------------|----------|----------|----------|------|------|------|------|------|--------|
| -1 | | | | | | | | | 17.95 |
| 0 | 0.00 | 0.00 | 0.26 | 0.85 | 1.91 | 0.33 | 0.41 | 1.02 | 3.82 |
| 1 | 0.50 | 0.19 | 0.24 | 0.93 | 1.84 | 0.33 | 0.42 | 0.99 | -4.69 |
| 2 | 0.50 | 0.19 | 0.28 | 0.78 | 1.87 | 0.24 | 0.38 | 1.26 | -9.00 |
| 3 | 0.50 | 0.18 | 0.30 | 0.74 | 1.91 | 0.21 | 0.36 | 1.39 | -10.55 |
| : | : | : | : | : | : | : | : | : | : |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.50 | 0.18 | 0.31 | 0.70 | 1.92 | 0.19 | 0.35 | 1.50 | -11.84 |
| Δ^* | | | | | | | | | -1.33 |

$$g_y = 0.195, h_1 = h_2 = 2, h_3 = 0, v = 1.5, \rho = 0.6, n_p = 0.2.$$

intuitive on first sight, since consumption as well as labour-income taxes distort labour supply. However, consumption taxation implicitly combines a distortionary wage tax and a non-distortionary lump-sum tax on existing wealth. For that reason, a move from consumption towards wage taxation increases aggregate distortions in the economy.

The next simulation analyses again the introduction of an unfunded social security system. The initial equilibrium is the same as in Table 7.2. But now a pay-as-you-go-financed pension system is introduced in period 1 with $\kappa = 0.5$. Table 7.3 reports the resulting individual and macroeconomic adjustment in and after the reform year. Like in the basic model (see Table 6.6), the introduction of an unfunded pension system redistributes from current young and future cohorts towards existing older cohorts as the latter receive benefits although they have not (or only partially) contributed to the system. The contribution rate of the pension system now distorts individual labour supply and consequently aggregate labour input declines temporarily after the reform. As in the previous simulation, due to negative income effects future cohorts consume less leisure so that aggregate labour input increases again during the transition even above its initial level. Households also save less since they expect pension benefits in the future. As before, this reduces the capital stock and the wage rate. However, the saving decision is not distorted by the social security system. The efficiency loss of 1.3 per cent of aggregate resources reported in the last line of Table 7.3 is solely due to labour supply distortions. The latter may seem counterintuitive on first sight since labour input actually increases after the reform. Again, we have to distinguish between income and substitution effects. Efficiency changes are only due to substitution effects while the observed labour-supply reaction is mainly due to income effects.

Consequently the last two simulations clearly demonstrate that the macroeconomic repercussions are a bad indicator of the aggregate efficiency consequences of policy reforms. In both simulations aggregate labour supply increases while the capital stock clearly falls in the long run. Nevertheless, since both policy reforms do not effect savings distortions, the reported efficiency losses are only due to labour-supply distortions.

7.1.4 A NOTE ON LABOUR-AUGMENTING TECHNOLOGICAL PROGRESS

Up to this point, the only source of growth in our model was population growth. However, the main driving force for growth in reality is labour-augmenting technological change. Typically this would be included by multiplying the labour input in the production function L_t by a productivity factor which grows at a constant rate λ through time. In the present setup it seems quite simple to augment human capital with such a productivity factor. However, if human-capital endowment grew over time, then wages $w_{j,s}$, the value of lifetime resources $W_{1,t}$, and consumption $c_{j,s}$ would also grow over time. The first-order condition (7.12) already indicates that if wages and consumption grow at the same rate, then leisure demand will also have a positive or negative trend. This, however, is not compatible with a long-run equilibrium path. Consequently, in order to allow for labour-augmenting technological change in the model with variable labour supply, we need to make some additional assumptions. The literature has mainly used two approaches, which we discuss in this subsection.¹

Time-augmenting technological change One option for reaching a steady-state growth path is to assume that technological change is time-augmenting. This means that due to productivity growth households can use working time but also the time spent for leisure more efficiently. Formally, this means that the time endowment $\bar{T}_{j,t}$ of a cohort is not unity anymore, but increases with the growth factor n_e over time, i.e.

$$\bar{T}_{1,t} = (1 + n_e)\bar{T}_{1,t-1} \quad \text{and} \quad \bar{T}_{j,t} = \bar{T}_{j-1,t-1} \quad \forall j > 1,$$

so that the total resources (in efficiency units) of the cohort entering the labour market in period t increase by

$$\tilde{W}_{1,t} = (1 + n_e)\tilde{W}_{1,t-1}.$$

Since the utility function (7.11) is homothetic, the increase in resources shifts the budget line outward, but does not change the consumption-leisure ratio. Consequently, in the steady state each successive cohort earns and consumes n_e per cent more in each year than the previous cohort. The aggregation of labour supply in efficiency units (per capita of the young cohort) now changes to

$$\tilde{L}_t = \sum_{j=1}^3 h_j(\bar{T}_{j,t} - \tilde{\ell}_{j,t})m_{j,t} = \sum_{j=1}^3 h_j(1 - \ell_{j,t})\bar{T}_{j,t}m_{j,t}$$

¹ A third approach would be to assume a continuous change in tastes for leisure in the utility function, i.e. where v in (7.11) has a time trend.

where we use $\ell_{j,t} = \tilde{\ell}_{j,t}/\bar{T}_{j,t}$. Since \tilde{L}_t is rising in the steady state we have to normalize it per unit of the time endowment of the first generation $\bar{T}_{1,t}$ in order to get

$$L_t = \frac{\tilde{L}_t}{\bar{T}_{1,t}} = \sum_{j=1}^3 h_j(1 - \ell_{j,t})\hat{m}_{j,t} \quad \text{with} \quad \hat{m}_{j,t} = \frac{\hat{m}_{j-1,t-1}}{(1 + n_e)(1 + n_{p,t})}.$$

Note that $\hat{m}_{1,t} = 1$ then is normalized labour input, which is constant in a steady state. The same normalization is applied when calculating the aggregate values A_t, C_t, K_t . The growth factor n_e then also appears when changes in aggregate variables are computed, i.e.

$$I_t = (1 + n_e)(1 + n_{p,t+1})K_{t+1} - (1 - \delta)K_t.$$

Cobb-Douglas utility Instead of assuming growth in time endowment, we could also ensure the feasibility of a steady state with productivity growth by employing the utility function

$$u(c_{j,s}, \ell_{j,s}) = \frac{1}{1 - \frac{1}{\gamma}} \left[c_{j,s}^\nu \ell_{j,s}^{1-\nu} \right]^{1-1/\gamma}.$$

where ν denotes the weight of consumption in periodic utility. Cobb-Douglas utility has the nice feature that income and substitution effects of changes in wages exactly cancel each other out. In order to see this, we look at the first-order conditions (7.5) to (7.7) which now imply

$$\ell_{j,s} = \frac{p_s}{w_{j,s}^n} \frac{1-\nu}{\nu} c_{j,s}$$

for $j = 1, 2, 3$. Consequently, when wages and consumption demand grow at the same rate on the long-run equilibrium path, leisure demand remains constant. Substituting the relationship between leisure demand and consumption into the first-order conditions (7.8) and (7.9) and using the budget constraint (7.6) now yields

$$c_{j,s} = \left[\frac{w_{1,t}^n}{w_{j,s}^n} \right]^{(1-\nu)(\gamma-1)} \left[\frac{p_s}{p_t} \right]^{\nu(1-\gamma)-1} [\beta^{j-1} \prod_{k=t+1}^s R_k^n]^\gamma c_{1,t}$$

and

$$p_t c_{1,t}/\nu + \frac{p_{t+1} c_{2,t+1}/\nu}{R_{t+1}^n} + \frac{p_{t+2} c_{3,t+2}/\nu}{R_{t+1}^n R_{t+2}^n} = W_{1,t}.$$

As before we again derive $c_{1,t} = \Psi_{1,t} W_{1,t}$. However, the fraction of initial consumption is now defined by

$$\Psi_{1,t} = \frac{\nu}{p_t} \left\{ \sum_{j=1}^3 \beta^{\gamma(j-1)} \left[\left(\frac{p_s}{p_t} \right)^\nu \prod_{k=t+1}^s (R_k^n)^{-1} \right]^{1-\gamma} \left(\frac{w_{j,s}^n}{w_{1,t}^n} \right)^{(1-\nu)(1-\gamma)} \right\}^{-1}.$$

In the long run equilibrium wages grow with a constant rate n_e , so that the fraction Ψ_1 is constant. Consequently initial consumption $c_{1,t}$ grows at the same rate n_e as initial resources $W_{1,t}$, while initial leisure demand is constant as well. The latter can be verified by the leisure demand equation at age 1, which reads

$$\ell_1 = p \cdot \frac{1-\nu}{\nu} \cdot \Psi_1 \cdot \frac{W_{1,t}}{w_{1,t}^n}$$

where net wages $w_{1,t}^n$ grow at the same rate as total resources $W_{1,t}$. Future leisure demand $\ell_{j,s}$ is also constant in the long run, since growth in net wages and consumption cancel each other out. Consequently, despite a positive trend in wages, age-specific labour supply is constant in the long run equilibrium. If we now assume

$$h_{j,s} = (1 + n_e)h_{j,s-1} \quad \text{and} \quad h_{j,s} = (1 + n_e)h_{j-1,s-1}$$

aggregate labour supply measured in units of initial human capital

$$L_t = \frac{1}{h_{1,t}} \sum_{j=1}^3 h_{j,t} (1 - \ell_{j,t}) m_{j,t} = \sum_{j=1}^3 h_j (1 - \ell_{j,t}) m_{j,t}$$

grows in the long-run equilibrium at rate $(1 + n_e)(1 + n_p)$. Consequently the capital stock and all other aggregate variables have to grow at the same rate.

Therefore accounting for exogenous technological change in combination with variable labour supply is not a problem. However, exogenous growth does not explain the origin of the growth process. One possible explanation for growth is discussed in Section 7.2.

7.2 Human capital and the growth process

To keep things tractable, we return to the model with exogenous labour supply. We assume that in the first working period households can use a share of their time endowment for education, which increases their human capital in the second period of life. Education can be seen as a private investment, where households invest time

they could otherwise have used to generate labour income in order to yield higher labour productivity and therefore income in the future. On top of this private return to education, we assume that there may also be social externality associated with the average stock of human capital per worker which increases output and factor returns. One problem is to specify the human-capital endowment in the first working period h_1 . In subsection 7.2.1, we let the initial endowment of human capital be constant and equal to unity. Consequently the long-run growth rate of the economy is constant and cannot be influenced by public policy. In a next step, we model human-capital spillovers from parent cohorts to the generation of children. Successive cohorts then start with different initial human-capital endowments. As a result, the growth rate of the economy is endogenous.

7.2.1 EDUCATION INVESTMENT AND EXTERNALITIES

Households have to decide how to split their time endowment in the first period between working time and education investment e_t and how much to consume and save in each period. Their productivity in the second period $h_{2,t+1}$ is related to their training decision by

$$h_{2,t+1} = h_1[1 + \Phi(e_t)], \quad (7.17)$$

where $\Phi(e_t)$ is a decreasing returns to scale function measuring the agent's productivity growth for a given education investment. As already mentioned, we assume h_1 to be constant for each new cohort. The budget constraint and household decisions are very similar as in the basic model covered in Chapter 6. The first-period budget constraint now changes to

$$p_t c_{1,t} = [1 - (1 - \tau_t^s)e_t]w_t^n h_1 - a_{2,t+1},$$

where τ_t^s denotes the subsidy rate of the government for education costs which are forgone net wages.

Given the time-separable utility function from Chapter 6 as well as the fact that there is no leisure consumption, we can solve the household optimization problem in two steps. First, we maximize the household's lifetime income by choosing the optimal education investment

$$\max_{e_t} W_{1,t} = [1 - (1 - \tau_t^s)e_t]w_t^n h_1 + \frac{w_{t+1}^n h_1 [1 + \Phi(e_t)]}{R_{t+1}^n} + \frac{pen_{t+2}}{R_{t+1}^n R_{t+2}^n}.$$

The resulting first-order condition

$$\frac{w_{t+1}^n \Phi'(e_t)}{R_{t+1}^n} = (1 - \tau_t^s)w_t^n. \quad (7.18)$$

shows that optimal education investment balances the benefits from higher wages in the future against the current cost from forgone wages (net of subsidies). Note that in steady state, the optimal education decision is independent of net wages. If we specify the education function as

$$\Phi(e_t) = \xi e_t^\nu \quad \text{with} \quad \xi > 0 \quad \text{and} \quad 0 < \nu < 1,$$

condition (7.18) yields an optimal education time of

$$e_t = \left(\frac{\xi \nu w_{t+1}^n}{w_t^n (1 - \tau_t^s) R_{t+1}^n} \right)^{\frac{1}{1-\nu}}. \quad (7.19)$$

The education investment increases with the discounted level of future net wages and decreases with the current net wage (including the subsidy rate) which represents an opportunity cost. With the optimal education time e_t and the implied lifetime resources $W_{1,t}$, the second step of solving the household problem consists in deriving the optimal path of consumption and savings over the life cycle. This can simply be done using equations (6.4) and (6.5).

Human capital may affect aggregate output via two distinct channels. On the one hand, it increases aggregate labour supply L_t which is measured in units of human capital. On the other hand, we assume that there may be an additional externality associated with human capital. More specifically, we let the average stock of human capital per worker H_t feature directly in the production function, so that

$$Y_t = F(K_t, L_t, H_t) = K_t^\alpha L_t^{1-\alpha} H_t^\epsilon$$

where ϵ denotes the strength of the externality. Labour input and the average stock of human capital are computed from

$$L_t = h_1(1 - e_t) + \frac{h_{2,t}}{1 + n_{p,t}} \quad \text{and} \quad H_t = \frac{L_t}{1 - e_t + \frac{1}{1+n_{p,t}}}. \quad (7.20)$$

The wage rate and the interest rate then read

$$w_t = (1 - \alpha) k_t^\alpha H_t^\epsilon \quad \text{and} \quad r_t = \alpha k_t^{\alpha-1} H_t^\epsilon - \delta.$$

Finally, the education subsidy has to be included in the budget constraint of the government and equation (6.9) changes to

$$T_t + (1 + n_{p,t+1}) B_{t+1} - B_t = G_t + r_t B_t + \tau_t^s e_t w_t^n h_{1,t}.$$

7.2.2 NUMERICAL IMPLEMENTATION AND SIMULATION

Program 7.2 shows the parameters and variables required to implement endogenous human capital in our model. Compared to the model with fixed labour supply, there are three additional parameters ξ , υ , and ϵ as well as the variables for the subsidy rate τ_t^s , education investment e_t , human capital $h_{j,t}$, and average human capital (per capita) H_t . Compared to previous programs, human capital now needs a time index since it can change over time. Note that we are back in our basic model with fixed labour supply whenever we set $\xi = \epsilon = 0$. In this case, the household will choose an education level of zero and human capital is constant over the life cycle at a level of 1.

We set the initial human-capital endowment to $h_{1,t} = 1$ in the subroutine `initialize`. As in the previous Program 7.1, we solve the initial equilibrium using the subroutine `get_SteadyState`. The only difference is that we do not need to compute shadow wages any more. Instead, we use the subroutine `decisions` to calculate the optimal education time and human-capital path, which is shown in Program 7.2.a. Note that this is the only required adjustment in subroutine `decisions`. We do, however, have to account for the impact of the education choice on available resources $W_{j,t}$ and savings $a_{j,t}$ in the function `get_W` as well as the subroutine `get_path`, respectively. We also need to adjust the aggregation subroutine `quantities`, in order to correctly calculate aggregate labour supply L_t and average human capital H_t from equation (7.20).

Program 7.2 The global variables for endogenous human capital

```
program TRHC_OLG
    [.....]
    real*8, parameter :: xi      = 2.0d0
    real*8, parameter :: upsi    = 0.5d0
    real*8, parameter :: epsi    = 0.0d0
    [.....]
    real*8 :: taus(0:TT), e(0:TT), h(JJ, 0:TT), HH(0:TT)
    [.....]
end program
```

Program 7.2.a Computation of individual decisions with endogenous human capital

```
subroutine decisions(it)
    [.....]
    ! derive optimal education time and future human capital
    itp = year(it, 1, 2)
    e(it) = (xi*upsi*wn(it)/&
              (wn(it)*(1d0-taus(it))*Rn(itp)))**((1d0/(1d0-upsi)))
    h(2,itp) = 1d0 + xi*e(it)**upsi

    ! consumption path for cohort that just entered the economy
    c(1, it) = get_Psi(1, it)*get_W(1, it)
    call get_path(1, it)
    [.....]
end subroutine
```

Tax reform analysis We first want to simulate a move from consumption to wage taxation as in Table 6.4. Starting with the same parameters as in the previous chapter but setting $\xi = 2$ and $v = 0.5$ yields a similar initial consumption tax as in the basic model. Table 7.4 reports the resulting consequences of the tax reform when education time is endogenous.

As before in Table 6.4 the policy reform reduces net incomes of younger cohorts who will save less so that capital stock decreases after the reform period. The induced increases in interest rates reduces the optimal education investment (see condition (7.19) above) already in the reform period. Consequently, second-period human capital falls after the reform. Note that aggregate labour supply (in human-capital units) now temporarily increases in the reform period (when human capital is constant) but then decreases again (when human capital is endogenous) so that labour input is lower in the new long-run equilibrium with wage taxation. As in Table 6.4 the policy reform redistributes from those currently young and future cohorts towards older cohorts of the first period. Welfare losses increase throughout the transition due to the crowding out of capital and the induced fall in gross wages. At first sight one would probably expect efficiency losses since condition (7.19) seems to indicate distortions of the education decision from labour-income taxation. However, wage taxes are cancelled out completely as long as the policy reform arrives unexpected. In this case cohorts adjust their education decision but this is due to income effects from changing wages and interest rates.

Distortions arise when the same policy reform is pre-announced one period ahead of time as shown in Table 7.5. We again start from the same initial equilibrium as in the previous Table 7.4. However, now the government announces in period 1 the move towards the labour-income tax in period 2. As a consequence, young individuals in the reform year reduce their education time in period 1 in a much stronger way than before because they want to work more when wages are not taxed. When the reform is implemented in the second period, young cohorts cannot benefit from intertemporal labour supply adjustments any more. Consequently their education time increases again. Note that the oldest cohort now only slightly benefits from the policy reform due to the

Table 7.4 From consumption to labour-income taxes with endogenous human capital

| <i>t</i> | τ^w | τ^c | <i>C</i> | <i>e</i> | h_2 | <i>L</i> | <i>K</i> | <i>w</i> | <i>r</i> | HEV |
|------------|----------|----------|----------|----------|-------|----------|----------|----------|----------|--------|
| -1 | | | | | | | | | | 27.96 |
| 0 | 0.00 | 0.27 | 0.82 | 0.13 | 1.72 | 2.31 | 0.18 | 0.33 | 1.76 | -2.66 |
| 1 | 0.29 | 0.00 | 0.88 | 0.09 | 1.72 | 2.35 | 0.18 | 0.33 | 1.79 | -10.85 |
| 2 | 0.32 | 0.00 | 0.75 | 0.09 | 1.60 | 2.24 | 0.15 | 0.31 | 2.03 | -16.93 |
| 3 | 0.33 | 0.00 | 0.72 | 0.09 | 1.60 | 2.25 | 0.13 | 0.30 | 2.19 | -19.58 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ∞ | 0.35 | 0.00 | 0.67 | 0.08 | 1.58 | 2.23 | 0.11 | 0.29 | 2.44 | -24.02 |
| Δ^* | | | | | | | | | | 0.00 |

$$g_y = 0.205, h_1 = 1, \xi = 2, v = 0.5, \epsilon = 0, n_p = 0.2.$$

Table 7.5 From consumption to labour-income taxation with policy announcement

| <i>t</i> | τ^w | τ^c | <i>C</i> | <i>e</i> | <i>h</i> ₂ | <i>L</i> | <i>K</i> | <i>w</i> | <i>r</i> | <i>HEV</i> |
|------------|----------|----------|----------|----------|-----------------------|----------|----------|----------|----------|------------|
| -1 | | | | | | | | | | 1.04 |
| 0 | 0.00 | 0.27 | 0.82 | 0.13 | 1.72 | 2.31 | 0.18 | 0.33 | 1.76 | 6.32 |
| 1 | 0.00 | 0.27 | 0.82 | 0.08 | 1.72 | 2.36 | 0.18 | 0.33 | 1.79 | -0.52 |
| 2 | 0.30 | 0.00 | 0.86 | 0.10 | 1.56 | 2.20 | 0.20 | 0.34 | 1.65 | -8.51 |
| 3 | 0.31 | 0.00 | 0.79 | 0.09 | 1.62 | 2.26 | 0.16 | 0.31 | 1.99 | -14.95 |
| : | : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.35 | 0.00 | 0.67 | 0.08 | 1.58 | 2.23 | 0.11 | 0.29 | 2.44 | -24.04 |
| Δ^* | | | | | | | | | | -0.66 |

$g_y = 0.205$, $h_1 = 1$, $\xi = 2$, $v = 0.5$, $\epsilon = 0$, $n_p = 0.2$.

Table 7.6 Education subsidies without human-capital externalities

| <i>t</i> | τ^s | τ^c | <i>C</i> | <i>e</i> | <i>h</i> ₂ | <i>L</i> | <i>K</i> | <i>w</i> | <i>r</i> | <i>HEV</i> |
|------------|----------|----------|----------|----------|-----------------------|----------|----------|----------|----------|------------|
| -1 | | | | | | | | | | -3.30 |
| 0 | 0.00 | 0.27 | 0.82 | 0.13 | 1.72 | 2.31 | 0.18 | 0.33 | 1.76 | 0.23 |
| 1 | 0.25 | 0.29 | 0.81 | 0.20 | 1.72 | 2.23 | 0.18 | 0.33 | 1.73 | 1.50 |
| 2 | 0.25 | 0.29 | 0.83 | 0.21 | 1.90 | 2.37 | 0.17 | 0.32 | 1.86 | -0.16 |
| 3 | 0.25 | 0.29 | 0.83 | 0.21 | 1.92 | 2.39 | 0.17 | 0.32 | 1.88 | -0.49 |
| : | : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.25 | 0.29 | 0.82 | 0.21 | 1.92 | 2.39 | 0.17 | 0.32 | 1.91 | -1.11 |
| Δ^* | | | | | | | | | | -0.86 |

$g_y = 0.205$, $h_1 = 1$, $\xi = 2$, $v = 0.5$, $\epsilon = 0$, $n_p = 0.2$.

slight reduction in the consumption tax rate (not reported) and the increase in the interest rate. They die before the reform is fully implemented. Of course, the long-run equilibrium has to be the same as in Table 7.4 since the policy announcement changes behaviour only temporarily. Finally, the last line of Table 7.5 shows that the pre-announcement generates an efficiency loss of roughly 0.6 per cent of aggregate resources which reflects the induced reaction of the initial cohort.

Education subsidies Next we want to show that not only taxes but also subsidies reduce economic efficiency in the present model. In Table 7.6 we start again from the same initial equilibrium as before. However, the government now introduces in period 1 a subsidy of 25 per cent ($\tau_t^s = 0.25$) in order to increase education and human-capital growth. The subsidy has an enormous impact on education investment which increases immediately from 13 to 20 per cent of the time endowment. As a result, human capital and labour input also increase significantly, with a lag of one period. However, in order to finance the subsidy, consumption taxes have to be increased so that savings and the capital stock fall inducing a reduction in wages and an increase in the interest rate. The immediate rise in consumption taxation burdens the older generations living in year 1

of the transition (who do not benefit from the subsidy at all), the youngest cohort in the reform period benefits from the reduced cost of education while the future cohorts lose again due to higher taxes and lower wages. Overall, distortions rise in the economy so that after compensation aggregate efficiency losses amount to almost 0.9 per cent of aggregate resources.

Finally, we analyse education subsidies in a model where the average human-capital level increases production and factor prices. Since individuals do not take this positive externality into account when they optimize their education investment, education time is too low in the initial equilibrium of Table 7.7. Note that this initial equilibrium is different compared to the initial equilibrium from Table 7.6 since individuals realize higher wages despite the same education time and human capital-endowment. Consequently consumption is higher for all cohorts as well as savings and the resulting initial capital stock. As in the previous simulation of Table 7.6, the introduction of the education subsidy increases education investment immediately from 13 to 20 per cent of the time endowment. Consequently human capital and labour input rise as before. However, due to the positive externality, wages do not fall and consumption taxes do not have to rise as much as before. Therefore, due to the higher human-capital stock, consumption expenditures can even increase during the transition. While the older generations living in year 1 of the transition are hurt again (but less than before), all other cohorts benefit from the increase in education investment. Consequently, after compensation the economy realizes an overall efficiency gain of 1.3 per cent of aggregate resources.

It can be shown that for the present externality specification (i.e. $\epsilon = 0.2$) there exists an optimal subsidy rate of 32 per cent which maximizes the aggregate efficiency gain at 1.4 per cent of aggregate resources. This optimal subsidy rate balances the marginal benefits from the positive education externality and the marginal cost from the increased tax distortions. If the subsidy rate is further increased beyond that optimal level, marginal distortions dominate the marginal benefits so that aggregate efficiency declines again.

Table 7.7 Education subsidies with human-capital externalities

| <i>t</i> | τ^s | τ^c | <i>C</i> | <i>e</i> | <i>h</i> ₂ | <i>L</i> | <i>K</i> | <i>w</i> | <i>r</i> | HEV |
|------------|----------|----------|----------|----------|-----------------------|----------|----------|----------|----------|-------|
| -1 | | | | | | | | | | -3.04 |
| 0 | 0.00 | 0.27 | 0.90 | 0.13 | 1.72 | 2.31 | 0.20 | 0.36 | 1.76 | 1.08 |
| 1 | 0.25 | 0.29 | 0.89 | 0.20 | 1.72 | 2.23 | 0.20 | 0.36 | 1.73 | 2.62 |
| 2 | 0.25 | 0.28 | 0.92 | 0.21 | 1.90 | 2.37 | 0.19 | 0.35 | 1.90 | 1.90 |
| 3 | 0.25 | 0.28 | 0.93 | 0.21 | 1.92 | 2.39 | 0.19 | 0.35 | 1.91 | 2.03 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| ∞ | 0.25 | 0.28 | 0.93 | 0.21 | 1.92 | 2.39 | 0.19 | 0.35 | 1.91 | 1.86 |
| Δ^* | | | | | | | | | | 1.33 |

$$g_y = 0.205, h_1 = 1, \xi = 2, v = 0.5, \epsilon = 0.2, n_p = 0.2.$$

7.2.3 HUMAN-CAPITAL SPILLOVERS AND ENDOGENOUS GROWTH

In subsection 7.2.2 the initial human-capital stock h_1 was constant and human capital was only endogenous in the second working period. Implicitly we therefore assumed that there are no technological spillovers from the human capital of the parent cohort to the children generation. This seems to be highly unrealistic in practice. The education level of children typically depends on the education of their parents and this intergenerational externality can be an important determinant of the growth rate of the economy. In this subsection we develop a model variant in which the growth rate of the economy is driven by human-capital accumulation and the respective spillovers from the parent cohort to their children. Consequently, in contrast to equation (7.17) we assume that second-period human capital $h_{2,t+1}$ increases by

$$h_{2,t+1} = h_{1,t}(1 + \xi e_t^v), \quad (7.21)$$

so that the initial human-capital endowment $h_{1,t}$ is cohort-specific. The most simple way to model the spillover process of human capital from parents to children is to assume a linear relationship

$$h_{1,t} = \mu h_{2,t}. \quad (7.22)$$

$\mu > 0$ defines the fraction of parental human capital that is acquired by the children generation. Substituting (7.22) into (7.21) we can derive the (endogenous) human capital growth rate $n_{e,t+1}$ as

$$h_{1,t+1} = h_{1,t}\mu(1 + \xi e_t^v) = h_{1,t}(1 + n_{e,t+1}). \quad (7.23)$$

The way households make their optimal life-cycle choices is very similar to Section 7.2.1. They first choose their optimal education investment by maximizing lifetime resources

$$\max_{e_t} \widetilde{W}_{1,t} = [1 - (1 - \tau_t^s)e_t]w_t^n h_{1,t} + \frac{w_{t+1}^n h_{2,t+1}}{R_{t+1}^n} + \frac{\widetilde{pen}_{t+2}}{R_{t+1}^n R_{t+2}^n}.$$

After substituting (7.21), the optimal education level can again be calculated from (7.19). For every cohort total resources $\widetilde{W}_{1,t}$ grow owing to the rising human-capital stock. As before, available resources are spent for private consumption, so that we have

$$\widetilde{W}_{1,t} = p_t \tilde{c}_{1,t} + \frac{p_{t+1} \tilde{c}_{2,t+1}}{R_{t+1}^n} + \frac{p_{t+2} \tilde{c}_{3,t+2}}{R_{t+1}^n R_{t+2}^n}.$$

In order to compute a long-run equilibrium of the economy, we normalize all individual variables in terms of the human-capital endowment of a cohort, meaning we calculate

$$\begin{aligned} W_{1,t} = \frac{\tilde{W}_{1,t}}{h_{1,t}} &= [1 - (1 - \tau_t^s)e_t]w_t^n + \frac{w_{t+1}^n(1 + n_{e,t+1})/\mu}{R_{t+1}^n} + \frac{pen_{t+2}}{R_{t+1}^n R_{t+2}^n} \\ &= p_t c_{1,t} + \frac{p_{t+1} c_{2,t+1}}{R_{t+1}^n} + \frac{p_{t+2} c_{3,t+2}}{R_{t+1}^n R_{t+2}^n}. \end{aligned}$$

$W_{1,t}$, $c_{j,s} = \frac{\tilde{c}_{j,s}}{h_{1,t}}$ and $pen_{t+2} = \frac{\tilde{pen}_{t+2}}{h_{1,t}}$ then define total resources, consumption as well as pensions measured in units of initial human capital, respectively. Note that the resources of a household aged $j = 2$ need to be computed from

$$W_{2,t} = w_t^n(1 + n_{e,t})/\mu + \frac{pen_{t+1}}{R_{t+1}^n},$$

so that one has to account for the increase in resources compared to the previous cohort. Of course, we then also have to normalize aggregate variables in a similar fashion, therefore distinguishing between total labour input

$$\tilde{L}_t = (1 - e_t)h_{1,t}N_t + h_{2,t}N_{t-1}$$

and aggregate labour input in units of initial human capital and normalized by the size of the young cohort

$$L_t = \frac{\tilde{L}_t}{h_{1,t}N_t} = (1 - e_t) + \frac{1}{\mu(1 + n_{p,t})}.$$

In a similar way we can define aggregate consumption

$$C_t = \frac{1}{h_{1,t}N_t} \cdot \sum_{j=1}^3 \tilde{c}_{j,t}N_{t+1-j} = \frac{1}{h_{1,t}N_t} \sum_{j=1}^3 c_{j,t}h_{j,t+1-j}N_{t+1-j} = \sum_{j=1}^3 c_{j,t}\hat{m}_{j,t}$$

and aggregate assets

$$A_t = \sum_{j=1}^3 a_{j,t}\hat{m}_{j,t},$$

where we have $\hat{m}_{1,t} = 1$ and $\hat{m}_{j,t} = \frac{\hat{m}_{j-1,t-1}}{(1+n_{e,t})(1+n_{p,t})}$.

7.2.4 NUMERICAL IMPLEMENTATION AND SIMULATION

In order to implement endogenous growth in our model we need to specify the parameter μ , the endogenous growth rate $n_{e,t}$ and the new aggregation variable $\hat{m}_{j,s}$.

Program 7.3 Aggregation of individual variables with endogenous growth

```

subroutine quantities(it)
    [.....]
    ! calculate adjusted aggregator
    ma(1, it) = 1d0
    do ij = 2, JJ
        ma(ij, it) = ma(ij-1, it)/(1d0+n_e(it))*(1d0+n_p(it)))
    enddo

    ! calculate economy wide aggregates
    CC(it) = 0d0
    AA(it) = 0d0
    LL(it) = 0d0
    HH(it) = 0d0
    do ij = 1, JJ
        CC(it) = CC(it) + c(ij, it)*ma(ij, it)
        AA(it) = AA(it) + a(ij, it)*ma(ij, it)
        if(ij < JR) LL(it) = LL(it) + mu**db1e(1-ij)*m(ij, it)
        if(ij < JR) HH(it) = HH(it) + m(ij, it)
    enddo
    LL(it) = LL(it) - e(it)
    HH(it) = LL(it)/(HH(it) - e(it))
    [.....]
end subroutine

```

In Program 7.3—which extends Program 7.2—the parameter is μ , the growth rate is $n_e(0 : TT)$, and the adjusted aggregation variable is $ma(JJ, 0 : TT)$. Note that we no longer need a variable for human capital as the level of human capital of agents at age $j = 2$, relative to their human capital at age $j = 1$, is directly defined through the human-capital growth rate as $\frac{h_{2,t+1}}{h_{1,t}} = \frac{1+n_{e,t+1}}{\mu}$. As before, optimal education is derived in subroutine decisions. But instead of the human-capital stock, we only compute the endogenous growth rate using (7.23). In addition, we have to slightly adjust the computation of available resources in subroutine get_w and of assets in subroutine get_path. The aggregation of individual decisions to aggregate variables in the subroutine quantities has to be adjusted as shown in Program 7.3. Since the growth rate of the economy now is $(1 + n_{e,t})(1 + n_{p,t})$, we need to adapt the computation of aggregate investment and public deficit. Last but not least, it is not straightforward to measure welfare and especially aggregate efficiency in economies with varying long-run growth rates. Hence, we refrain from welfare calculations in our analysis, but only focus on the growth effects of different policies.

Steady-state comparisons Before we simulate policy reforms with Program 7.3, it is useful to examine the impact of the human-capital transmission parameter μ on individual decisions and macroeconomic aggregates. Table 7.8 compares the impact on the long-run equilibrium for three specific values. It is not surprising that a strong spillover effect from parents to children's human capital increases the growth rate of the economy. However, at the same time the incentive to invest in education decreases so that

Table 7.8 Long-run analysis of human-capital acquisition

| μ | C | e | n_e | L | K | w | r |
|-------|------|------|-------|------|------|------|------|
| 1.2 | 0.62 | 0.07 | 0.86 | 1.76 | 0.08 | 0.27 | 2.66 |
| 1.0 | 0.73 | 0.09 | 0.60 | 1.91 | 0.10 | 0.29 | 2.31 |
| 0.8 | 0.91 | 0.11 | 0.34 | 2.14 | 0.15 | 0.31 | 1.96 |

$g_y = 0$, $\xi = 2$, $v = 0.5$, $\epsilon = 0$, $n_p = 0$.

Table 7.9 From consumption to labour-income taxation with endogenous growth

| t | τ^w | τ^c | C | e | n_e | L | K | w | r |
|----------|----------|----------|------|------|-------|------|------|------|------|
| -1 | | | | | | | | | |
| 0 | 0.00 | 0.30 | 0.70 | 0.11 | 0.34 | 2.14 | 0.15 | 0.31 | 1.96 |
| 1 | 0.31 | 0.00 | 0.75 | 0.08 | 0.34 | 2.17 | 0.15 | 0.31 | 1.98 |
| 2 | 0.33 | 0.00 | 0.68 | 0.08 | 0.25 | 2.17 | 0.12 | 0.29 | 2.25 |
| 3 | 0.34 | 0.00 | 0.66 | 0.08 | 0.26 | 2.17 | 0.11 | 0.29 | 2.36 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.35 | 0.00 | 0.63 | 0.08 | 0.25 | 2.17 | 0.10 | 0.28 | 2.54 |

$g_y = 0.22$, $\mu = 0.8$, $\xi = 2$, $v = 0.5$, $\epsilon = 0$, $n_p = 0$.

aggregate labour input (measured in human-capital units of the young cohort) is now fairly low as well as the aggregate capital stock. As a consequence, wages are low and the interest rate is fairly high in the long-run equilibrium. When the spillover effect declines in the following simulations, the value of the human capital in the second period (relative to human capital of the respective young cohort) increases implicitly. Consequently, despite the lower growth rate, households invest more in education, so that labour input and savings increase, inducing a rise in wages and a fall in the interest rate.

Tax reform analysis As in Section 7.2.2, we again simulate a move from consumption to wage taxation. Table 7.9 indicates that this policy leads to a decline in asset accumulation and therefore to an increase in the interest rate. A higher interest rate makes capital market investments more attractive relative to human-capital investments. Hence, the education effort of future cohorts declines. Through the intergenerational spillover effect, this leads to a persistent reduction in economic growth. We also simulate the introduction of an education subsidy in the endogenous growth model. In Table 7.10 we start from the same initial equilibrium as in Table 7.9. As before, the government introduces a subsidy of 25 per cent in period 1 in order to increase education investment and the growth rate of the economy. The subsidy increases education investment immediately from 11 to 17 per cent of the time endowment. As a result, the growth rate rises from 0.34 to 0.46 in the second period.

Table 7.10 Education subsidies with endogenous growth

| t | τ^S | τ^C | C | e | n_e | L | K | w | r |
|----------|----------|----------|------|------|-------|------|------|------|------|
| -1 | | | | | | | | | |
| 0 | 0.00 | 0.30 | 0.70 | 0.11 | 0.34 | 2.14 | 0.15 | 0.31 | 1.96 |
| 1 | 0.25 | 0.33 | 0.68 | 0.17 | 0.34 | 2.08 | 0.15 | 0.32 | 1.92 |
| 2 | 0.25 | 0.34 | 0.65 | 0.17 | 0.46 | 2.08 | 0.13 | 0.30 | 2.09 |
| 3 | 0.25 | 0.36 | 0.63 | 0.17 | 0.46 | 2.08 | 0.12 | 0.30 | 2.19 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.25 | 0.36 | 0.61 | 0.17 | 0.45 | 2.08 | 0.12 | 0.29 | 2.27 |

$$g_y = 0.22, \mu = 0.8, \xi = 2, v = 0.5, \epsilon = 0, n_p = 0.$$

7.3 Longevity risk and annuitization

In Chapter 5, we discussed the problem of longevity risk and the resulting individual demand for annuity insurance. The assumption of a certain lifespan in the previous OLG models can be justified by arguing that it reflects a situation in which households can buy a perfect annuity insurance that completely eliminates their longevity risk. However, in practice such a perfect annuity insurance is hardly available, an observation the literature describes as the *annuity puzzle*. If at all, households can buy expensive annuity products, so that the annuity market remains extremely thin. The question therefore is what missing (or imperfect) annuity markets imply for public policy and what the consequences of the introduction of a perfect annuity market would be.

7.3.1 THE HOUSEHOLDS' PROBLEM WITHOUT ANNUITY MARKETS

As in Chapter 6 we assume that a household that enters the labour market in period t reaches adulthood with certainty (i.e. the survival probability to the first period is $\psi_{1,t} = 1$). After working in the first period, the household survives with probability $\psi_{2,t+1} = 1 - q_{1,t}$ from period 1 to period 2. Conditional on having survived to the second working period, the household will live up to the retirement period with probability $\psi_{3,t+2} = 1 - q_{2,t+1}$. Again the maximum lifespan is restricted to three periods, meaning that $\psi_{4,t+3} = 0$. Since the number of members of a cohort declines with growing age, it is useful to define the size of a cohort aged j in period t as

$$N_{j,t} = \psi_{j,t} N_{j-1,t-1} \quad \text{with} \quad N_{1,t} = (1 + n_{p,t}) N_{1,t-1}. \quad (7.24)$$

Consequently, cohort weights are defined as $m_{1,t} = 1$ and $m_{j,t} = \frac{\psi_{j,t}}{1+n_{p,t}} \cdot m_{j-1,t-1}$.

As before, households born in period t maximize the time-separable utility function, where now future consumption is further discounted by the survival probabilities

$$U_t = U(c_{1,t}, c_{2,t+1}, c_{3,t+2}) = \sum_{j=1}^3 \beta^{j-1} \left(\prod_{i=1}^j \psi_{i,k} \right) u(c_{j,s})$$

with $s = t + j - 1$ and $k = t + i - 1$. Since annuity markets are absent, the return on individual assets is still the net interest rate. In the previous programs of this chapter, agents knew with certainty when their life was over. Hence, they were able to perfectly plan at which point in time they wanted to exhaust all their savings. In a setup with uncertain survival, households may die prior to their maximum lifespan J and therefore leave unintended bequests. These bequests are transferred to the surviving agents of the economy according to some scheme we will discuss later. We denote by $b_{j,t}$ the unintended bequest an agent at age j receives at date t . Consequently the intertemporal budget constraint is now defined as

$$\sum_{j=1}^3 \frac{p_s c_{j,s}}{\prod_{k=t+1}^s R_k^n} = \sum_{j=1}^3 \frac{w_s^n + b_{j,s} + pen_{j,s}}{\prod_{k=t+1}^s R_k^n} =: W_{1,t} \quad (7.25)$$

where $pen_{j,s} = 0$ for $j < 3$.

One question that arises is how the total assets (including interest payments) of the deceased are distributed over the surviving cohort members. We therefore employ a flexible distribution scheme $\Gamma_{j,s} = \frac{\omega_{b,j}}{\sum_{i=1}^3 \omega_{b,i} m_{i,s}}$, where the $\omega_{b,i}$ s are exogenously specified. The cohort amount of unintended bequests can then be calculated from

$$b_{j,t} = \Gamma_{j,t} BQ_t,$$

where BQ_t defines aggregate bequest in period t specified below.

Maximizing utility subject to the budget constraint (7.25) yields the first-order conditions

$$\begin{aligned} p_{t+1} u'(c_{1,t}) &= \beta \psi_{2,t+1} R_{t+1}^n p_t u'(c_{2,t+1}) \\ p_{t+2} u'(c_{2,t+1}) &= \beta \psi_{3,t+2} R_{t+2}^n p_{t+1} u'(c_{3,t+2}), \end{aligned}$$

so that for the CRRA utility function $u(c) = \frac{c^{1-1/\gamma}}{1-1/\gamma}$ the resulting consumption path is given by

$$c_{2,t+1} = \left[\beta \psi_{2,t+1} R_{t+1}^n \frac{p_t}{p_{t+1}} \right]^\gamma c_{1,t} \quad (7.26)$$

$$c_{3,t+2} = \left[\beta^2 \psi_{2,t+1} \psi_{3,t+2} R_{t+1}^n R_{t+2}^n \frac{p_t}{p_{t+2}} \right]^\gamma c_{1,t}. \quad (7.27)$$

Substituting conditions (7.26) and (7.27) into the budget constraint (7.25) yields

$$c_{1,t} = \Psi_{1,t} W_{1,t} \quad \text{with}$$

$$\Psi_{1,t} = \frac{1}{p_t} \left\{ \sum_{j=1}^3 \left[\beta^{(j-1)} \Pi_{i=1}^j \psi_{i,t+i-1} \right]^\gamma \left[\frac{p_s}{p_t} \Pi_{k=t+1}^s (R_k^n)^{-1} \right]^{1-\gamma} \right\}^{-1}. \quad (7.28)$$

Given $c_{1,t}$, we can derive the optimal path for consumption $c_{2,t+1}, c_{3,t+2}$ by substituting back into equations (7.26) and (7.27) above. Finally, we can compute savings from

$$\begin{aligned} a_{2,t+1} &= w_t^n + b_{1,t} - p_t c_{1,t} \quad \text{and} \\ a_{3,t+2} &= R^n a_{2,t+1} + w_{t+1}^n + b_{2,t+1} - p_{t+1} c_{2,t+1}. \end{aligned}$$

Aggregation It is straightforward to compute aggregate variables for labour supply (in efficiency units), consumption, assets, and bequests in period t by just using the cohort aggregators as defined above:

$$\begin{aligned} L_t &= \sum_{j=1}^2 h_j m_{j,t}, \quad C_t = \sum_{j=1}^3 c_{j,t} m_{j,t} \\ A_t &= \sum_{j=2}^3 a_{j,t} \frac{m_{j,t}}{\psi_{j,t}} \quad \text{and} \quad BQ_t = R_t^n \cdot \sum_{j=2}^3 a_{j,t} \frac{m_{j,t}}{\psi_{j,t}} \cdot (1 - \psi_{j,t}) \end{aligned}$$

Note that aggregate assets include the savings of the deceased that are bequeathed to surviving cohorts. Aggregate bequests are then simply the fraction of total assets which can be attributed to those who died at the end of the previous period (including interest).

The budget of the pay-as-you-go-financed pension system changes to

$$\tau_t^P w_t L_t = pen_t \cdot m_{3,t}.$$

Finally, the budget constraint of the LSRA agency needs to take into account different cohort sizes as well, so that

$$\nu_{-1} N_{3,1} + \nu_0 N_{2,1} + \nu_1 N_{1,1} + \frac{\nu_2 N_{1,2}}{R_2} + \dots = 0$$

must hold. The initial debt (or asset) level of the LSRA agency—expressed in the size of the first cohort in period $t = 2$ —has to be adjusted by

$$B_2^a = \frac{1}{1 + n_{p,2}} \sum_{j=1}^3 \nu_{2-j} m_{j,1}.$$

7.3.2 NUMERICAL IMPLEMENTATION AND SIMULATION

Program extensions In order to implement the model with uncertain lifespan, we need to specify survival probabilities $\psi_{j,s}$, aggregate savings of the deceased BQ_t , the weights for the bequest distribution $\omega_{b,j}$, and the resulting unintended bequest levels $b_{j,s}$. In Program 7.4, which again is based on Program 6.2, we therefore define arrays $\text{psi}(3, 0:\text{TT})$, $\text{BQ}(0:\text{TT})$ and $\text{omega_b}(\text{JJ})$, as well as $\text{GAM}(\text{JJ}, 0:\text{TT})$ for the bequest distribution scheme and $\text{beq}(\text{JJ}, 0:\text{TT})$ for individual bequest levels. The changes in the program are straightforward. Besides specifying survival rates $\psi_{j,s}$, the cohort weights $m_{j,s}$, and the distribution $\Gamma_{j,s}$ in subroutine `initialize`, bequests have to be added to the lifetime resources in function `get_w`. Program 7.4 shows the changes made to the function `get_Psi`, which derives $\Psi_{j,t}$ from equation (7.28). Given an initial consumption, we can compute individual bequest, the optimal consumption path, and savings in subroutine `get_path` shown in Program 7.4.a. The final adjustment is made in the aggregation subroutine `quantities`, where we compute aggregate savings of the deceased BQ_t . Note that since we used cohort weights $m_{j,t}$ for aggregation in all budgets before, no adjustments in the pension budget or the aggregation of LSRA transfers is required in our coding.

Pension reform Now we are able to simulate policy reforms and demographic transitions with rising lifespan. The analysis of a pay-as-you-go-financed pension system seems most interesting in this regard. Intuitively one would expect that at such a system atleast partially makes up for missing annuities. In the following we assume $\psi_{2,t} = 0.85$ and $\psi_{3,t} = 0.8$ with respect to survival rates, so that life expectancy amounts to roughly 2.5 periods.² Then we simulate the same policy reform as in Table 6.6 of Chapter 6.

Program 7.4 Computation of marginal consumption with lifespan uncertainty

```

function get_Psi(ij, it)
    [.....]
    get_Psi = 1d0
    PRn = 1d0
    PPs = 1d0

    do ijp = ij+1, JJ
        itp = year(it, ij, ijp)
        PRn = PRn*Rn(itp)
        PPs = PPs*psi(ijp, itp)
        get_Psi = get_Psi + (beta**((ijp-ij)*PPs)**gamma &
                             * (p(itp)/PRn/p(it))**((1d0-gamma)
    enddo
    get_Psi = 1d0/p(it)/get_Psi

end function

```

² In a closed economy setup it is difficult to simulate the 3-period OLG model with a much lower life expectancy because agents would not save for old age at all.

Program 7.4.a Computation of the optimal path without annuities

```

subroutine get_path(ij, it)
[.....]
! determine bequests at age ij
PRn = 1d0
beq(ij, it) = GAM(ij, it)*BQ(it)

! determine consumption path for remainder of lifetime
do ijp = ij+1, JJ
[.....]
! get consumption and bequests
c(ijp, itp) = &
(beta** (ijp-ij)*PRn*p(itp)/p(itp))**gamma*c(ij, it)
beq(ijp, itp) = GAM(ijp, itp)*BQ(itp)

! calculate assets
a(ijp, itp) = wn(itm)*h(ijp-1) + beq(ijp-1, itm) &
+ pen(ijp-1, itm) + Rn(itm)*a(ijp-1, itm) &
- p(itm)*c(ijp-1, itm)
if(itp == 2)a(ijp, itp) = a(ijp, itp) + v(-ijp+3)
if(itp > 2 .and. ijp == 2)a(ijp, itp) = a(ijp, itp) + v(itm)
enddo

end subroutine

```

Table 7.11 Introducing pay-as-you-go pensions with uncertain life span

| t | κ | τ^P | τ^C | C | BQ | K | w | r | HEV |
|------------|----------|----------|----------|------|------|------|------|------|--------|
| -1 | | | | | | | | | 38.17 |
| 0 | 0.00 | 0.00 | 0.27 | 0.71 | 0.09 | 0.24 | 0.39 | 1.20 | 7.87 |
| 1 | 0.50 | 0.14 | 0.24 | 0.78 | 0.09 | 0.24 | 0.39 | 1.20 | -5.58 |
| 2 | 0.50 | 0.14 | 0.28 | 0.67 | 0.08 | 0.18 | 0.36 | 1.46 | -13.80 |
| 3 | 0.50 | 0.14 | 0.30 | 0.63 | 0.07 | 0.16 | 0.34 | 1.61 | -17.53 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.50 | 0.14 | 0.33 | 0.58 | 0.07 | 0.13 | 0.32 | 1.81 | -21.94 |
| Δ^* | | | | | | | | | 0.00 |

$$g_y = 0.20, \omega_{b,1} = \omega_{b,2} = 0.5, \psi_2 = 0.85, \psi_3 = 0.8, n_p = 0.2.$$

Table 7.11 demonstrates that such a policy basically produces the same adjustment process as in the basic model with certainty of lifespan. Again the policy redistributes across cohorts but does not generate any efficiency gains or losses. At first sight especially, this last result seems to be surprising, since one would expect an efficiency gain due to the implicit annuity provision. However, in the present model with certainty of income the introduction of the unfunded pension system does not alter individual decisions as the benefit only provides an additional income in old age. Younger agents reduce their savings as before, but the marginal consumption decisions (7.26) to (7.28) are not altered by the policy reform. Consequently no efficiency effects are induced.

Table 7.12 Macroeconomics of rising life expectancy

| t | $\psi_{2,t}$ | $\psi_{3,t}$ | $E[T]$ | C | L | BQ | K | w | r |
|----------|--------------|--------------|--------|------|------|------|------|------|------|
| 0 | 0.85 | 0.80 | 2.53 | 0.90 | 1.71 | 0.09 | 0.24 | 0.39 | 1.20 |
| 1 | 0.85 | 0.80 | 2.67 | 0.89 | 1.71 | 0.09 | 0.24 | 0.39 | 1.20 |
| 2 | 0.90 | 0.80 | 2.80 | 0.90 | 1.75 | 0.09 | 0.24 | 0.39 | 1.19 |
| 3 | 0.95 | 0.85 | 2.80 | 0.94 | 1.79 | 0.06 | 0.26 | 0.39 | 1.15 |
| : | : | : | : | : | : | : | : | : | : |
| ∞ | 0.95 | 0.90 | 2.80 | 0.95 | 1.79 | 0.05 | 0.26 | 0.39 | 1.15 |

$$g_y = 0, \omega_{b,1} = \omega_{b,2} = 0.5, n_p = 0.2.$$

Rising life expectancy The ongoing demographic transition in most developed countries comprises a decline in fertility rates and an increase in life expectancy. In the present model life expectancy can be computed from

$$E[T] = q_{1,t} + 2q_{2,t+1}\psi_{2,t+1} + 3\psi_{2,t+1}\psi_{3,t+2}$$

and a change in life expectancy is easy to simulate as long as survival probabilities are increasing. In Table 7.12 it is assumed that the newborn cohorts in periods 1 and 2 experience an increase in their life expectancy from 2.53 to 2.80 periods. As a consequence they will save more and leave less in terms of unintended bequests to succeeding younger cohorts. Consequently, initial consumption decreases while old-age consumption increases slightly. The lower death rates of the second cohort increase aggregate labour input. However, the impact on factor prices is very modest since the capital stock also increases. Overall the macroeconomic impact depends very strongly on the assumed reaction of public goods. For this reason we do not include a public sector in the present simulation. The other problem is that a consistent welfare analysis of lifespan extension is not possible in the present model as changing survival probabilities also change individual preferences. However, it is still possible to analyse the welfare effects of policy reforms given a specific path of survival probabilities. Consequently for a exogenously specified demographic transition one has to simulate the path of the economy twice with different policy parameters. Then it is possible to compute the cohort-specific welfare changes induced by the policy reform. Of course, one could also derive the aggregate efficiency effects of this policy reform. We will not follow this line of policy analysis further here.

7.3.3 INTRODUCING PRIVATE ANNUITY MARKETS

It is more interesting to analyse the introduction of a perfect annuity insurance. In this case households do not receive unintended bequests anymore. Instead, a perfect annuity contract guarantees that savings of the deceased are directly distributed to the

survivors of a cohort. Consequently, the total return $\tilde{R}_{j,t}^n$ from annuitized savings $a_{j,t}$ has to satisfy

$$\tilde{R}_{j,t}^n a_{j,t} := R_t^n a_{j,t} + (1 - \psi_{j,t})/\psi_{j,t} R_t^n a_{j,t} = \frac{R_t^n}{\psi_{j,t}} a_{j,t}.$$

Hence, the return on savings increases with the death probability of a cohort and the budget constraint (7.25) changes to

$$p_t c_{1,t} + \frac{p_{t+1} c_{2,t+1}}{\tilde{R}_{2,t+1}^n} + \frac{p_{t+2} c_{3,t+2}}{\tilde{R}_{2,t+1}^n \tilde{R}_{3,t+2}^n} = W_{1,t} \quad (7.29)$$

where lifetime resources $W_{1,t}$ are redefined with the adjusted discount rate $\tilde{R}_{j,s}^n$ and without bequests. Maximizing utility subject to the new budget constraint (7.29) changes the first-order conditions to

$$\begin{aligned} c_{2,t+1} &= \left[\beta \psi_{2,t+1} \tilde{R}_{2,t+1}^n \frac{p_t}{p_{t+1}} \right]^\gamma c_{1,t} \\ c_{3,t+2} &= \left[\beta^2 \psi_{2,t+1} \psi_{3,t+2} \tilde{R}_{2,t+1}^n \tilde{R}_{3,t+2}^n \frac{p_t}{p_{t+2}} \right]^\gamma c_{1,t}, \end{aligned}$$

so that after substitution in the budget constraint (7.29) we get $c_{1,t} = \Psi_{1,t} W_{1,t}$ where $\Psi_{1,t}$ is now defined as

$$\Psi_{1,t} = \frac{1}{p_t} \left\{ \sum_{j=1}^3 \left[\beta^{(j-1)} \prod_{i=1}^j \psi_{i,t+i-1} \right]^\gamma \left[\frac{p_s}{p_t} \prod_{k=t+1}^s (\tilde{R}_{v,k}^n)^{-1} \right]^{1-\gamma} \right\}^{-1} \quad (7.30)$$

with $v = k - t + 1$.

The introduction of private annuity markets can be implemented quite easily. It is enough to specify an age-specific discount factor $\tilde{R}_{j,t}^n$, which we can do in subroutine `factor_price`. The procedures for calculating available resources, optimal initial consumption and the optimal life-cycle path can be left almost unaltered (except for the age-specific net return of course). As shown in Program 7.5, we govern the existence of perfect annuity markets by means of a logical variable `pam`. When `pam` is set to `false`., the program does exactly the same as Program 7.4. If `pam` is set to `true`., the initial equilibrium still abstracts from annuity markets, but in period 1 private annuity markets are introduced. Note that annuitization is only applied to new savings, meaning that savings of period 0 still earn the regular return R_t^n and bequests from steady-state savings are still distributed. The only additional adjustment had to be made in subroutine `quantities`, where aggregate bequests BQ_t are only positive in case the returns $R_{j,t}^n$ don't stem from annuities, see Program 7.5.a. It should be clear that this

Program 7.5 Computation of net return with private annuity markets

```

subroutine factor_prices(it)
[.....]
  do ij = 1, JJ
    if(pam .and. it >= 2) then
      Rn(ij, it) = (1d0+r(it)*(1d0-taur(it)))/psi(j,it)
    else
      Rn(ij, it) = 1d0+r(it)*(1d0-taur(it))
    endif
  enddo
end subroutine

```

Program 7.5.a Computation of aggregate bequest

```

subroutine quantities(it)
[.....]
  BQ(it) = BQ(it) + ((1d0+r(it)*(1d0-taur(it)))/psi(ij, it) &
                      - Rn(ij, it))*a(ij, it)*m(ij, it)
[.....]
end subroutine

```

Table 7.13 Introducing private annuity insurance

| <i>t</i> | τ^c | <i>C</i> | <i>BQ</i> | <i>K</i> | <i>Y</i> | <i>w</i> | <i>r</i> | <i>HEV</i> |
|------------|----------|----------|-----------|----------|----------|----------|----------|------------|
| -1 | | | | | | | | 0.13 |
| 0 | 0.27 | 0.71 | 0.09 | 0.24 | 0.94 | 0.39 | 1.20 | 8.66 |
| 1 | 0.27 | 0.71 | 0.09 | 0.24 | 0.94 | 0.39 | 1.20 | 3.75 |
| 2 | 0.26 | 0.73 | 0.00 | 0.23 | 0.94 | 0.38 | 1.21 | -6.06 |
| 3 | 0.27 | 0.70 | 0.00 | 0.21 | 0.92 | 0.38 | 1.28 | -8.30 |
| : | : | : | : | : | : | : | : | : |
| ∞ | 0.29 | 0.66 | 0.00 | 0.19 | 0.88 | 0.36 | 1.40 | -11.69 |
| Δ^* | | | | | | | | 0.49 |

$$g_y = 0.20, \omega_{b,1} = \omega_{b,2} = 0.5, \psi_2 = 0.85, \psi_3 = 0.8, \eta_p = 0.2.$$

formulation yields aggregate bequests exactly as before, whenever there are no annuity markets.

Note that in contrast to Chapter 5, we do not give households a choice whether to invest in annuities or not. In addition, the annuity is fully paid out every period and therefore has the same financial liquidity as other assets. Consequently, in the present setup (no uncertainty, full rationality, no overhead cost, no bequest motive) household would always invest all their savings into annuities. Of course, this is not a realistic situation, but it serves as a good benchmark case. Now we will go on to change some of the major assumptions.

Table 7.13 reports the resulting macroeconomic and welfare effects when we start from the same initial equilibrium without pension provision and consumption tax financing as

in Table 7.11. In the reform period both younger cohorts still receive unintended bequests from the previous cohorts, although they can take full advantage of the introduced annuity insurance. While the cohort entering the labour market in period 0 only benefits from the reform, all ‘newborn’ cohorts in and after period 1 are hurt by the loss in bequests. Given our bequest distribution, the cohort born in period 1 still receives some bequest (in their first period), but all cohorts born afterwards experience a welfare loss since the annuity provision does not compensate for the loss in unintended bequests (now in both periods). Welfare losses increase during the transition due to rising taxes and lower wages. However, the annuity provision induces not only income, but also (positive) insurance effects. Therefore, after compensating income effects during the transition, the overall efficiency gain is roughly 0.5 per cent of aggregate resources.

7.4 Further reading

Since the seminal work of Auerbach and Kotlikoff (1987), various extensions of the deterministic OLG model have been developed which are surveyed by Kotlikoff (2000) as well as Zodrow and Diamond (2013). Among others, Fehr (1999) distinguishes skill classes within a cohort while Jokisch (2006) analyses policy reforms in a multi-country OLG setup. Broer and Lassila (1997) present applications that deal mostly with pension reform issues in different countries, while Harrison et al. (2000) discuss a broader range of issues including human-capital formation. The latter is also studied in Bouzahzah, la Croix, and Docquier (2002) as well as in Buyse, Heylen, and van de Kerckhove (2017). Heijdra, Mierau, and Reijnders (2014) examine the annuitization of resources in a similar model framework comparing exogenous and endogenous growth. Note that typically the OLG model is disaggregated on the household side to make it suitable for the analysis of both intra- and intergenerational as well as international redistribution issues. The production side of the economy is typically modelled as a representative sector. In principle it would be no problem to disaggregate the production side further (see Zodrow and Diamond, 2013), but this is computationally costly and typically more efficiently achieved with other models.

7.5 Exercises

- 7.1. Introduce two logical variables `CES` and `CD` into the model with variable labour supply which allows us to switch between different preferences. If `CES==.true.`, then the model should derive labour supply under CES preferences as in Program 7.1. If `CES==.false.` and `CD==.true.`, then use Cobb-Douglas preferences

$$u(c_{j,s}, \ell_{j,s}) = \frac{1}{1 - \frac{1}{\gamma}} \left[c_{j,s}^\nu \ell_{j,s}^{1-\nu} \right]^{1-1/\gamma}.$$

If `CES==.false.` and `CD==.false.`, the model is simulated with fixed labour supply.

Compare a move from consumption to wage taxation and the introduction of the pay-as-you-go pension system under CES preferences (compare Tables 7.2 and 7.3) and under CD preferences (where the consumption share is set to $\nu = 0.4$). Do you see any qualitative difference?

- 7.2. Introduce a parameter `ss=2` into the original model with variable labour supply which denotes the number of skill classes in each cohort. Assume that all skill types have the same preferences, but the highly skilled have an endowment of $h_{j=2}$ units of human capital while the low skilled are endowed with only $h_{j=1}$.
- a) Think carefully about which variables of the model need to be adjusted to include an additional dimension `ss`. Assume that the population growth rate is the same for both skill classes, but the weights are different. Hence, you have to set $0 < m_{1,1,t} < 1$ and $m_{2,1,t} = 1 - m_{1,1,t}$ and we have $m_{k,j,t} = m_{k,j-1,t-1}/(1 + n_{p,t})$.
 - b) How do you have to adjust the functions and subroutines on the household side (meaning in subroutine `decisions`)? How do the subroutines for aggregation and the LSRA change?
 - c) Simulate the introduction of a flat-rate pay-as-you-go pension system with different weights $m_{1,1,t}$ for the highly skilled and explain your results in economic terms.
- 7.3. Introduce a parameter `ss=2` into the original model with variable labour supply, which now denotes the number of countries considered. Assume that both countries are identical with respect to preferences, technologies, population growth rates, and human-capital endowment, but that they differ in their size and their economic policies.
- a) Think carefully about which variables of the model need to be adjusted to include an additional dimension `ss`. Allow for the different sizes $m_{k,1,t}$ of both countries, so that we again have $m_{k,j,t} = m_{k,j-1,t-1}/(1 + n_{p,t})$.
 - b) How do you have to adjust the functions and subroutines on the household side (meaning in subroutine `decisions`)? How do the subroutines for aggregation and the LSRA change?
 - c) Assume an initial equilibrium in which both countries use a consumption tax and are of identical size. Why does a change in the size of both countries not induce any trade?
 - d) Simulate the move from consumption taxation towards income taxation in the second country using different country weights. Explain the macroeconomic effects as well as the welfare changes in economic terms.

- 7.4. Extend the model with endogenous human capital from three to six periods. In the first period households have to choose between education and working time, taking into account that education increases their human capital in all future working periods as

$$h_{2,t+1} = h_1(1 + \xi e_t^\nu) \quad h_{3,t+2} = h_{2,t+1}(1 - \delta_h) \quad h_{4,t+3} = h_{3,t+2}(1 - \delta_h),$$

where δ_h denotes the depreciation rate of human capital. In periods 5 and 6 the household retires and may receive flat pensions $pen_{j,s}$.

- a) Introduce a subroutine `get_edu(it)` that computes the optimal education time of a cohort 'born' in period it and the corresponding future human-capital levels.
 - b) Simulate the model with consumption taxes ($g_y = 0.205$, $\xi = 1.2$, $\nu = 0.5$, $\epsilon = 0$, $n_p = 0.2$) with and without human-capital depreciation ($\delta_h = 0.5$). In economic terms explain how depreciation affects the education decision.
 - c) Simulate the introduction of a flat pay-as-you-go pension system ($\kappa = 0.5$) and explain the resulting macroeconomic effects as well as the welfare changes in economic terms.
- 7.5. Use the model from the previous exercise (six periods, endogenous human capital) and introduce variable labour supply in all working periods. Assume a CES utility function of the form

$$u(c_{j,s}, \ell_{j,s}) = \frac{1}{1 - \frac{1}{\gamma}} \left[c_{j,s}^{1-1/\rho} + \nu \ell_{j,s}^{1-1/\rho} \right]^{\frac{1-1/\gamma}{1-1/\rho}}.$$

Note that in the first period the time restriction changes to $\ell_{1,t} + e_t \leq 1$. Also assume that the government may pay a subsidy τ_t^s on gross wages, in order to facilitate the computation of shadow wages in the first period.

- a) Adjust the functions `get_Psi` and subroutines `get_path` as well as `shadw` from the previous program.
 - b) Simulate the introduction of an education subsidy of $\tau_t^s = 0.2$ in the model with consumption taxes ($g_y = 0.205$, $\xi = 1.2$, $\nu = 0.5$, $\epsilon = 0$, $n_p = 0.2$, $\delta_h = 0.2$, $\rho = 0.6$, $\nu = 1.5$). (Technical note: Damping must be reduced significantly!) Explain in economic terms how the subsidy affects the education decision, labour supply, and welfare.
 - c) Simulate the introduction of a flat pay-as-you-go pension system ($\kappa = 0.5$) (in the model without the subsidy) and explain the resulting macroeconomic effects as well as the welfare changes in economic terms.
- 7.6. Again use the model from exercise 7.4 and introduce variable retirement in period $j_r = 5$. This means that labour supply is fixed in all periods except the retirement period. Preferences in this period are identical to the previous exercise and the (periodical) budget constraint is

$$p_t c_{j_r,t} + a_{j_r+1,t+1} = R_t^n a_{j_r,t} + w_t^n(1 - \ell_t) + \ell_t p_{\text{en}} a_{j_r,t}.$$

Consequently households decide when to enter and when to leave the labour market, but they do not vary their labour supply during working periods.

- a) Set up the theoretical problem and derive the optimal path for consumption and labour supply. Pay special attention to those who in the reform year are exactly at the retirement age j_r . What does their consumption function look like and how is it related to the general consumption function?
- b) What functions and subroutines of the model need the most adjustment?
- c) Simulate the introduction of an education subsidy of $\tau_t^s = 0.1$ in the previous model with consumption taxes ($g_y = 0.205, \xi = 1, v = 0.5, \epsilon = 0, n_p = 0.2, \delta_h = 0.2, \rho = 0.6, \nu = 0.5$). (Technical note: Introduce an exogenous age-related productivity profile e_j with $e_1 = e_2 = 1, e_3 = 1.1, e_4 = 1.2, e_5 = 1.3, e_6 = 0.0$ which increases the human-capital stock multiplicatively!) In economic terms explain how the subsidy affects the decision to invest in education, retirement, and welfare.
- d) Simulate the introduction of a flat pay-as-you-go pension system ($\kappa = 0.5$) (in the model without the subsidy) and explain the resulting macroeconomic effects as well as the welfare changes in economic terms.

- 7.7. Introduce lifespan uncertainty into the model with variable retirement from the previous Exercise 7.6. For simplicity assume that $\psi_2 = \dots = \psi_6 = 0.85$ in the initial equilibrium. Assume that there is no government sector. In order to induce households to retire later we set $v = 0.4$ and $\omega_3 = \omega_4 = 0.5$. (Technical note: It is useful to damp individual bequest levels in every iteration!)

The question we are addressing is how households react in terms of their decisions to invest in education and their decisions about retirement when life expectancy rises. To this end, increase the survival probabilities to $\psi_{2,2} = \dots = \psi_{2,T} = 0.9, \psi_{3,4} = \dots = \psi_{3,T} = 0.9$ and so on, so that life expectancy rises from 4.15 in the initial equilibrium to 4.69 for those who are born in period 5 and later. How does this affect the education and the retirement decisions?

- 7.8. Extend the model with endogenous growth to six periods ($j_r = 5$) and include human-capital depreciation so that $h_{3,t+1} = (1 - \delta_h)h_{2,t}$ and $h_{4,t+2} = (1 - \delta_h)^2 h_{2,t}$.
- a) How does the optimal education time of the model change compared to the three-period model discussed above? How does the definition of labour input change?
 - b) Simulate the model with and without human-capital depreciation. What are the economic effects of higher depreciation?
 - c) What is the appropriate basis for the replacement rate of a pay-as-you-go pension system?
- 7.9. Introduce a more general human-capital production function $\Phi(e_t, X_t) = \xi e_t^{v_1} X_t^{v_2}$ into the model with endogenous growth from Exercise 7.8. In this specification,

human-capital production requires inputs of public infrastructure X_t . Public infrastructure is provided by the government. Specify a government parameter $\text{pin}(0 : \text{TT})$, which defines the fraction of public infrastructure relative to GDP.

- a) How does this affect the optimal education choice? What functions and subroutines of the model need to be adjusted?
 - b) Assume that public infrastructure is financed by a consumption tax. Quantify the fraction of public infrastructure that maximizes the growth rate in the steady state for $\xi = 5$, $v_1 = 0.2$, $v_2 = 0.7$. For simplicity, assume no public consumption (i.e. $g_y = 0$).
 - c) Simulate an increase in public infrastructure from 0.5 per cent of GDP to the growth-maximizing level and explain the resulting macroeconomic effects in economic terms.
 - d) Simulate the same increase in public infrastructure in a small open economy and explain the difference compared to the simulation in a closed economy.
- 7.10. Extend the model with lifespan uncertainty to six periods ($j_r = 5$). Use the following values for survival rates, human capital, population growth, and damping: $\psi_2 = \psi_3 = 0.95$, $\psi_4 = 0.9$, $\psi_5 = 0.85$, $\psi_6 = 0.8$, $h_j = 3.0 \forall j < j_r$, $n_p = 0.2$, and set damp at a value of 0.05.
- a) Simulate the introduction of a pay-as-you-go pension system with $\kappa = 0.5$ and $g_y = 0$. Make sure that you get no efficiency effects after LSRA compensation.
 - b) Now change parameters, i.e. increase damping, reduce population growth, human capital, or survival. Why is it hardly possible to generate either an equilibrium at all or an equilibrium without efficiency effects?
- 7.11. Introduce a ‘warm glow’ bequest motive into the model of the previous exercise. Like in Exercise 5.6 of Chapter 5, the household utility function then reads

$$U_t = \sum_{j=1}^3 \beta^{j-1} \left(\prod_{i=1}^j \psi_{i,m} \right) [u(c_{j,s}) + \beta(1 - \psi_{j+1,s+1}) \nu u(R_{s+1}^n a_{j+1,s+1})],$$

where the budget constraint in the last period becomes

$$p_{t+2} c_{3,t+2} = R_{t+2}^n a_{3,t+2} + \text{pen}_{t+2} - a_{4,t+3}.$$

- a) Derive the equation system that solves the household problem.
- b) Simulate the introduction of a pay-as-you-go pension system with $\kappa = 0.5$ and $g_y = 0$ with alternative parameter combinations. Why is the model much more flexible than the previous one?

Technical note: Set $\nu = 1$ and the damping parameter for the Gauss-Seidel algorithm to 0.2! LSRA compensation payments are not needed!

- c) Do higher pensions increase or decrease bequest? Explain in economic terms!

Part III

Advanced Computational Economics

8 Introduction to dynamic programming

Dynamic optimization is widely used in many fields of economics, finance, and business management. Typically one searches for the optimal time path of one or several variables that maximizes the value of a specific objective function given certain constraints. While there exist some analytical solutions to deterministic dynamic optimization problems, things become much more complicated as soon as the environment in which we are searching for optimal decisions becomes uncertain. In such cases researchers typically rely on the technique of *dynamic programming*. This chapter introduces the principles of dynamic programming and provides a couple of solution algorithms that differ in accuracy, speed, and applicability. Chapters 8 to 11 show how to apply these dynamic programming techniques to various problems in macroeconomics and finance.

8.1 Motivation: The cake-eating problem

To get things started we want to lay out the basic idea of dynamic programming and introduce the language that is typically used to describe it. The easiest way to do this is with a very simple example that we can solve both 'by hand' and with the dynamic programming technique.

Let's assume an agent owns a certain resource (say a cake or a mine) which has the size a_0 . In every period $t = 0, 1, 2, \dots, \infty$ the agent can decide how much to extract from this resource and consume, i.e. how much of the cake to eat or how many resources to extract from the mine. We denote his consumption in period t as c_t . At each point in time the agent derives some utility from consumption which we express by the so-called *instantaneous utility function* $u(c_t)$. We furthermore assume that the agent's utility is additively separable over time and that the agent is impatient, meaning that he derives more utility from consuming in period t than in any later period. We describe the extent of his impatience with the *time discount factor* $0 < \beta < 1$. The total utility of the agent as of time $t = 0$ should be given by

$$U_0 = \sum_{t=0}^{\infty} \beta^t u(c_t).$$

Obviously when choosing his optimal path of consumption, the agent has to respect a constraint, namely that he can not extract more than the total size of his resource. This means that his total amount of consumption has to be less or equal to the size of the mine

$$a_0 \geq \sum_{t=0}^{\infty} c_t.$$

Under the typical assumption that there is no saturation in utility, i.e. $u'(c_t) > 0$,¹ it is immediately clear that the agent in the end consumes all his resources meaning that nothing is wasted. We can consequently write the above inequality constraint as an equality constraint

$$a_0 = \sum_{t=0}^{\infty} c_t.$$

Now assume that instantaneous utility can (as in Chapters 5 and 6) be represented by the following function

$$u(c_t) = \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

In this functions γ measures the intertemporal elasticity of substitution. All in all we can write the problem the agent has to solve as

$$\max_{\{c_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \quad \text{subject to} \quad a_0 = \sum_{t=0}^{\infty} c_t. \quad (8.1)$$

(8.1) is a dynamic optimization problem.

8.1.1 THE ALL-IN-ONE SOLUTION

There are several ways to solve this dynamic optimization problem. One way would be to apply what we could call an all-in-one solution. This means trying to find the complete path of consumption $\{c_t\}_{t=0}^{\infty}$ in one step. In order to achieve this we have to set up a Lagrangian of the following form

¹ Or put differently: more consumption means more utility.

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \lambda \left[a_0 - \sum_{t=0}^{\infty} c_t \right].$$

λ is called the Lagrange multiplier. We can now take first-order conditions

$$\frac{\partial \mathcal{L}}{\partial c_t} = \beta^t \cdot c_t^{-\frac{1}{\gamma}} - \lambda = 0 \quad \text{for all } t = 0, 1, 2, \dots, \infty. \quad (8.2)$$

Using these first-order conditions, we find that

$$\begin{aligned} \beta^t c_t^{-\frac{1}{\gamma}} &= \beta^{t-1} c_{t-1}^{-\frac{1}{\gamma}} = \dots = c_0^{-\frac{1}{\gamma}} \\ \Leftrightarrow c_t &= \beta^\gamma c_{t-1} = \dots = \beta^{t\gamma} c_0. \end{aligned} \quad (8.3)$$

We can substitute this into the constraint and immediately obtain

$$a_0 = \sum_{t=0}^{\infty} \beta^{t\gamma} c_0 = c_0 \cdot \sum_{t=0}^{\infty} (\beta^\gamma)^t = \frac{c_0}{1 - \beta^\gamma}.$$

Consequently, the optimal solution to the dynamic optimization problem in (8.1) is given by

$$c_t = \beta^{t\gamma} \cdot (1 - \beta^\gamma) \cdot a_0.$$

Program 8.1 calculates the time path of consumption for a certain set of parameter values a_0 , β , and γ and plots the first 200 consumption values by means of the respective toolbox routines. Figure 8.1 shows the optimal consumption path for the specific combination $a_0 = 100$, $\beta = 0.95$, and $\gamma = 0.5$.

8.1.2 THE DYNAMIC PROGRAMMING APPROACH

Applying the all-in-one solution is very convenient for the simple cake-eating problem, as an analytical solution does exist. Yet if there were no analytical solution, we would quickly run into a problem, namely that the first-order conditions in (8.2) together with the agent's constraint constitute an infinite number of equations that would have to be solved simultaneously. This obviously isn't possible on a computer. One way around this problem is to use dynamic programming, i.e. to decompose the all-in-one problem into many small sub-problems. When the sub-problems are structured in the right way, we can again tackle them with our computer.

Program 8.1 All-in-one solution to the cake-eating problem

```

program allinone

    use toolbox

    implicit none

    ! model parameters
    real*8, parameter :: gamma = 0.5d0
    real*8, parameter :: beta = 0.95d0
    real*8, parameter :: a0 = 100d0

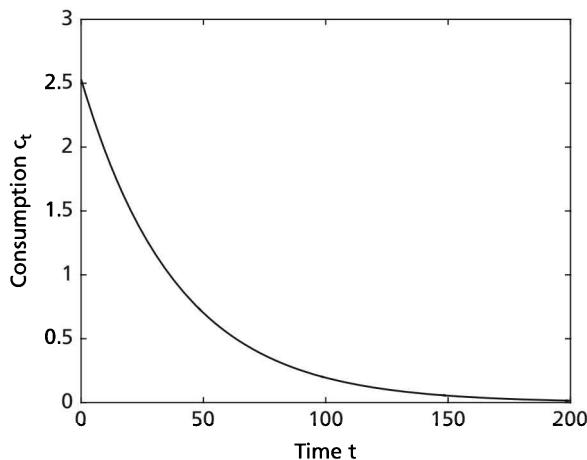
    ! other variables
    integer, parameter :: TT = 200
    real*8 :: c_t(0:TT)
    integer :: it

    ! calculate the time path of consumption
    do it = 0, TT
        c_t(it) = beta**(dble(it)*gamma)*(1d0 - beta**gamma)*a0
    enddo

    call plot(c_t)
    call execplot

end program

```

**Figure 8.1** Optimal consumption path in the cake-eating problem

We begin with restructuring the constraint of the agent. Obviously we can rewrite this constraint as

$$a_0 = \sum_{v=0}^{t-1} c_v + c_t + \sum_{v=t+1}^{\infty} c_v \quad \Leftrightarrow \quad a_0 - \sum_{v=0}^{t-1} c_v = c_t + \sum_{v=t+1}^{\infty} c_v.$$

The left-hand side of this equation tells us how much out of the total value of the resource the agent has already consumed in the periods prior to time t . We can interpret this as the amount of remaining resources a_t at time t and therefore write

$$a_0 - \sum_{v=0}^{t-1} c_v = a_t = c_t + \sum_{v=t+1}^{\infty} c_v. \quad (8.4)$$

The last equality just tells us that the total consumption in period t as well as in any future period needs to be equal to the remaining amount of resources. a_t , therefore, is a statistic sufficient for summarizing every decision that has been made prior to time t . It is easy to see that $\sum_{v=t+1}^{\infty} c_v = a_{t+1}$ must hold and therefore we can write the *dynamic budget constraint* or *transition equation* of the agent's resource as

$$a_t = c_t + a_{t+1} \Leftrightarrow a_{t+1} = a_t - c_t \quad \text{for all } t = 0, 1, \dots, \infty. \quad (8.5)$$

This dynamic budget constraint has a straightforward interpretation. It tells us that what remains at time $t + 1$ is the amount of remaining resources a_t at time t minus what the agent extracts c_t .

Having restructured the budget constraint of the agent, we now have to consider the utility function. We first define the sub-utility functions

$$U_t := \sum_{v=t}^{\infty} \beta^{v-t} u(c_v) = u(c_t) + \beta \underbrace{\sum_{v=t+1}^{\infty} \beta^{v-(t+1)} u(c_v)}_{=U_{t+1}} = u(c_t) + \beta U_{t+1}.$$

Let us now take a look at the sub-optimization problem

$$\max_{\{c_v\}_{v=t}^{\infty}} U_t = \sum_{v=t}^{\infty} \beta^{v-t} u(c_v) \quad \text{s.t.} \quad a_t = \sum_{v=t}^{\infty} c_v, \quad (8.6)$$

where the constraint is the same as in (8.4). This sub-optimization tells us what the agent's optimal consumption plan from period t onwards would look like under the assumption that only the amount a_t of his resource were left.² We can again set up the Lagrangian for this problem and derive the first-order conditions as

$$\begin{aligned} u'(c_t) &= \beta^{v-t} u'(c_v) \\ \Leftrightarrow c_v &= (u')^{-1} [\beta^{t-v} u'(c_t)] \quad \text{for all } v = t, t+1, \dots, \infty, \end{aligned}$$

² Note that maximizing $\sum_{v=t}^{\infty} \beta^{v-t} u(c_v)$ is equivalent to maximizing $\sum_{v=t}^{\infty} \beta^v u(c_v)$.

where we assumed that the marginal utility function is invertible. Using the constraint we obtain

$$a_t = \sum_{v=t}^{\infty} (u')^{-1} [\beta^{t-v} \cdot u'(c_t)] = \sum_{v=0}^{\infty} (u')^{-1} [\beta^{-v} \cdot u'(c_t)].$$

Note that we have used an index transformation to obtain the second equality. Since $u'(c_t)$ is constant over all the elements of the infinite sum, the solution for c_t in the above equation is independent of the time index t but only depends on the amount of remaining resources a_t . Hence we can write

$$c_t = f(a_t) \quad \text{and} \quad c_v = (u')^{-1} [\beta^{t-v} \cdot u'[f(a_t)]] \quad (8.7)$$

with a suitable function f . Finally plugging all this back into the utility function, we find that

$$\begin{aligned} \max_{\{c_v\}_{v=t}^{\infty}} U_t & \quad \text{s.t.} \quad a_t = \sum_{v=t}^{\infty} c_v \\ & = \sum_{v=t}^{\infty} \beta^{v-t} \cdot u \{ (u')^{-1} [\beta^{t-v} \cdot u'[f(a_t)]] \} \\ & = \sum_{v=0}^{\infty} \beta^v \cdot u \{ (u')^{-1} [\beta^{-v} \cdot u'[f(a_t)]] \} =: V(a_t). \end{aligned} \quad (8.8)$$

This result already tells us a lot about the nature of our dynamic optimization problem. Basically it means that the solution of the sub-problem (8.6) is independent of the time t , but only depends on the amount of resources a_t . This is quite intuitive when one recalls that the remaining time horizon under which the agent is making an optimal consumption plan is always infinity. In consequence, the maximum amount of utility the agent can derive from time t onwards under the assumption that only an amount of a_t of his resources are left is independent of time t and can be subsumed in the function $V(a_t)$ which only depends on the amount of remaining resources. V is usually called the *value function*.

Only one last step remains before we have written up our dynamic optimization problem as a dynamic program. It is possible to show that the optimization problem

$$V(a_t) = \max_{\{c_v\}_{v=t}^{\infty}} U_t = u(c_t) + \beta U_{t+1} \quad \text{s.t.} \quad \sum_{v=t}^{\infty} c_v = a_t$$

can be split into the two sub-problems

$$V(a_t) = \max_{c_t} u(c_t) + \beta \max_{\{c_v\}_{v=t+1}^{\infty}} U_{t+1}$$

and the two constraints

$$a_{t+1} = a_t - c_t \quad \text{and} \quad \sum_{v=t+1}^{\infty} c_v = a_{t+1}.$$

The literature refers to this as *Bellman's principle of optimality*.³ We therefore can solve the optimal choice problem at time t in two steps: First we determine what is the maximum possible amount of utility that can be attained from time $t+1$ onwards, given a remaining amount of resources a_{t+1} . Having solved for this, we solve for the optimal consumption level at time t given the amount of resources a_t as well as the fact that $a_{t+1} = a_t - c_t$. Consequently we can write

$$V(a_t) = \max_{c_t} u(c_t) + \beta V(a_{t+1}) \quad \text{s.t.} \quad a_{t+1} = a_t - c_t$$

or better

$$V(a) = \max_c u(c) + \beta V(a^+) \quad \text{s.t.} \quad a^+ = a - c. \quad (8.9)$$

We actually prefer the second notation, as it avoids confusion between the actual path of consumption c_t and resources a_t over time and the more general solution to the dynamic program. We use a $+$ sign to indicate variables in the successive period. Equation (8.9) is the so-called *Bellman-equation*. It is also called a *functional equation*, since the value function V in this equation is the true unknown. The solution V is a fixed point of the functional equation (8.9). The variable c is called *control variable*, as it controls the reduction of the resource. Finally, a is called *state variable*, as it describes the state of the dynamic problem at time t , in our case the remaining amount of resources. The state variable connects the different periods.

8.1.3 AN ANALYTICAL SOLUTION

Fortunately, since our problem has such a simple structure, an analytical solution exists. We can again tackle (8.9) using Lagrange optimization. The respective Lagrangian reads

$$\mathcal{L} = u(c) + \beta V(a^+) + \lambda [a - c - a^+].$$

The first-order conditions with respect to c and a^+ are

$$\frac{\partial \mathcal{L}}{\partial c} = u'(c) - \lambda = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial a^+} = \beta V'(a^+) - \lambda = 0$$

³ Named after Richard Bellman (1920–84), the ‘inventor’ of dynamic programming.

which results in

$$u'(c) = \beta V'(a^+). \quad (8.10)$$

Since

$$\begin{aligned} V(a^+) &= \max_{c^+} u(c^+) + \beta V(a^{++}) \quad \text{s.t. } a^{++} = a^+ - c^+ \\ &= \max_{a^{++}} u(a^+ - a^{++}) + \beta V(a^{++}) \end{aligned}$$

we obtain⁴

$$V'(a^+) = u'(c^+)$$

so that we get

$$u'(c) = \beta u'(c^+) \quad \text{or} \quad c^+ = \beta^\gamma c \quad (8.11)$$

with the above specification of the utility function. Not surprisingly the consumption path evolves exactly as in (8.3).

In order to get a complete solution of the dynamic program, we have to apply a simple trick. We therefore make a (sophisticated) guess of the functional form of the value function. Given what we already know from equation (8.8), we assume that

$$V(a) = B \frac{a^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

From the first-order condition (8.10) we now get

$$c^{-\frac{1}{\gamma}} = \beta B(a^+)^{-\frac{1}{\gamma}} \quad \text{so that} \quad a^+ = (\beta B)^\gamma c.$$

Using $c = a - a^+$ we can derive

$$a^+ = \frac{(\beta B)^\gamma a}{1 + (\beta B)^\gamma} \quad \text{and} \quad c = \frac{a}{1 + (\beta B)^\gamma}. \quad (8.12)$$

Obviously the value function must satisfy

$$B \frac{a^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} = \max_c \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta B \frac{(a^+)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}},$$

⁴ A more formal treatment of the derivative of the value function with respect to the state variable can be found in Chapter 9.

so that we obtain

$$Ba^{1-\frac{1}{\gamma}} = \left[\frac{a}{1 + (\beta B)^\gamma} \right]^{1-\frac{1}{\gamma}} + \beta B \left[\frac{(\beta B)^\gamma a}{1 + (\beta B)^\gamma} \right]^{1-\frac{1}{\gamma}}$$

after substituting the optimal decisions from (8.12). Eliminating $a^{1-\frac{1}{\gamma}}$ on both sides and rearranging finally leads to $B = [1 - \beta^\gamma]^{-\frac{1}{\gamma}}$. We can consequently write

$$V(a) = [1 - \beta^\gamma]^{-\frac{1}{\gamma}} \frac{a^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Finally, substituting B in the optimal values for savings and consumption (8.12) we get

$$a^+ = a^+(a) = a\beta^\gamma \quad \text{and} \quad c = c(a) = a(1 - \beta^\gamma).$$

The function $c(a)$ that tells us the value of the control variable for any possible state a is called *policy function*. The function $a^+(a)$, on the other hand, describes the development of the state variable. In the context of our simple problem the optimal policy is to always eat a constant fraction of the remaining cake.

The dynamic development of the cake-eating problem can be described with Figure 8.2. Starting with an initial resource value of a_0 in period 1, function $a^+(a)$ gives the resource value of the next period. The remainder of the resource c_0 is consumed. In the second period the resource value is a_1 . Again, the same fraction β^γ is saved to the next period and the remainder is consumed as c_1 . This process iterates so that the remaining value of the resource converges to zero.

Program 8.2 shows how to calculate the dynamics of the cake-eating problem from the policy function $c(a)$. It furthermore plots the policy and the value function as functions

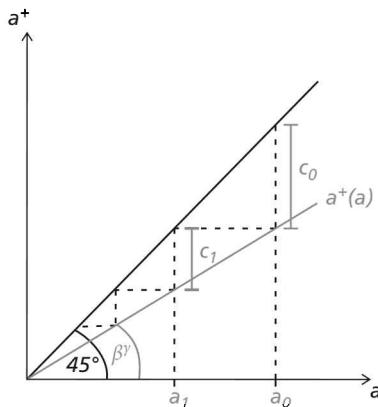


Figure 8.2 Dynamics of the cake-eating problem

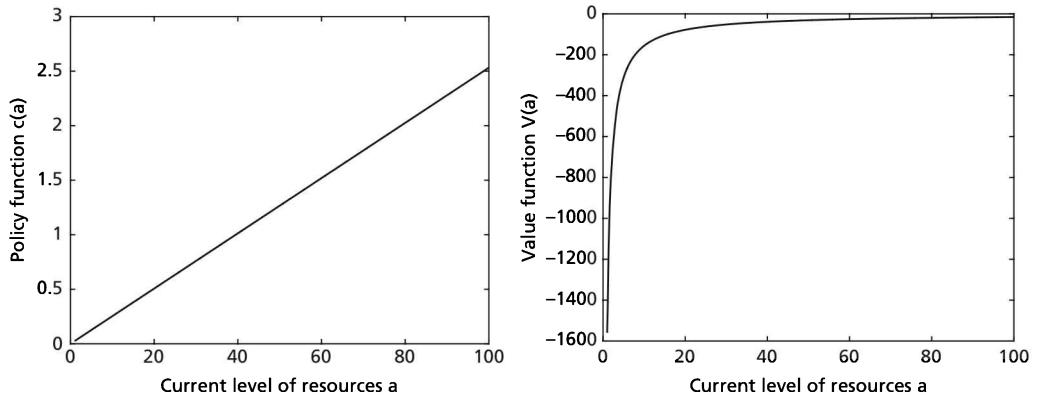
Program 8.2 Dynamic solution to the cake-eating problem

```

program dynamic
    [.....]
    ! calculate the time path of consumption
    a_t(0) = a0
    c_t(0) = a_t(0)*(1d0-beta**gamma)
    do it = 1, TT
        a_t(it) = a_t(it-1) - c_t(it-1)
        c_t(it) = a_t(it)*(1d0-beta**gamma)
    enddo

    call plot(c_t)
    call execplot
    [.....]
end program

```

**Figure 8.3** Policy and value function of the cake-eating problem

of the remaining value of resources a . Figure 8.3 shows the policy and value function for the parameters $a_0 = 100$, $\beta = 0.95$ and $\gamma = 0.5$.

8.2 Numerical solution by value function iteration

Guessing a solution of the value function is usually not as easy as in our simple cake-eating example. We therefore have to think about how to implement a dynamic programming problem on a computer. There actually are ample ways of doing so. In the following we describe a couple of solution methods. Some are more sophisticated and accurate than others, some are faster, and some are only applicable to very specific types of problems. In the end we have to find out which is the most appropriate method for the specific problem we want to solve.

All possible solution methods have one thing in common: the problem that a computer cannot handle arbitrary functions, as a function is an infinite dimensional object. Put differently, a computer cannot solve a dynamic programming problem on a continuous state space, e.g. $a \in [0, a_0]$. This would involve solving infinitely many optimization problems. Therefore for every method we apply, we have to first *discretize* the state space. In the case of our example this means that we do not solve the dynamic program on the whole interval $[0, a_0]$ but on a finite subset $\mathcal{A} = \{\hat{a}_0, \hat{a}_1, \dots, \hat{a}_n\}$ with $\hat{a}_v \in [0, a_0]$ and $\hat{a}_v < \hat{a}_{v+1}$. Figure 8.4 shows a potential discretization of the state space in the cake-eating problem and the respective values of the policy and value function at the discrete points. The set of discrete points is often also called a *grid* and the points themselves are called the *gridpoints*. In our example we have chosen an evenly spaced grid of points to discretize the state space which would also be called an equidistant grid, i.e.

$$\hat{a}_v = a_0 \cdot \frac{v}{n} \quad \text{for } v = 0, 1, \dots, n. \quad (8.13)$$

Having discretized the state space, our dynamic programming problem collapses to the following set of optimization problems

$$V(\hat{a}_v) = \max_c u(c) + \beta V(a^+) \quad \text{s.t.} \quad a^+ = \hat{a}_v - c \quad \text{for all } v = 0, 1, \dots, n.$$

Plugging the constraint directly into the function we can also write this problem as

$$V(\hat{a}_v) = \max_{a^+} u(\hat{a}_v - a^+) + \beta V(a^+) \quad \text{for } v = 0, 1, \dots, n. \quad (8.14)$$

The result of these optimization problems is a set of decisions and utility levels

$$\{c(\hat{a}_v)\}_{v=0}^n \quad \text{and} \quad \{a^+(\hat{a}_v)\}_{v=0}^n \quad \text{as well as} \quad \{V(\hat{a}_v)\}_{v=0}^n.$$

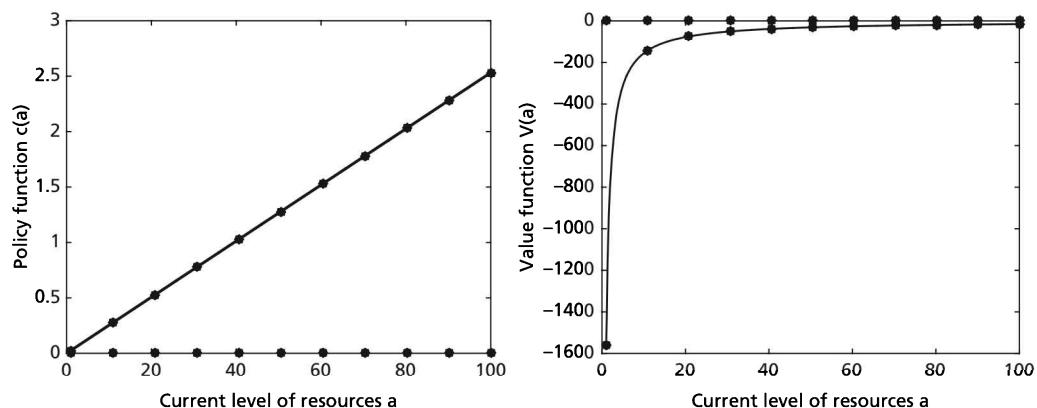


Figure 8.4 Discretization of the state space

There are two issues that arise when looking at the optimization problems in (8.14):

1. How can we determine the value function $V(a^+)$?
2. Knowing the value function $V(a^+)$ (or at least having a guess of it), how can we come up with a solution to the optimization problems in (8.14)?

We deal with issue number 2 next. For issue number 1 there exists a very simple algorithm called *value function iteration*. This algorithm follows the principles of a *fixed-point iteration* and proceeds via the following steps:

1. We start with a (more or less arbitrary) initial guess of the value function, e.g. $V(a) = 0$.
2. We then determine the optimal policies $\{c(\hat{a}_v)\}_{v=0}^n$ and $\{a^+(\hat{a}_v)\}_{v=0}^n$ for all $\hat{a}_v \in \mathcal{A}$ by solving the set of optimization problems in (8.14). We do not care about how we determine this solution for now, but just assume we could do so. We can then compute the resulting value function as

$$V_{\text{new}}(\hat{a}_v) = u[c(\hat{a}_v)] + \beta V[a^+(\hat{a}_v)] \quad \text{for } v = 0, 1, \dots, n.$$

3. We now check whether the maximum relative difference between the old and the new value function is below a certain exogenous tolerance level ϵ meaning whether

$$\max_{\hat{a}_v} \left| \frac{V_{\text{new}}(\hat{a}_v) - V(\hat{a}_v)}{V(\hat{a}_v)} \right| < \epsilon.$$

If this is the case we have found a fixed point of our dynamic program and can end the program. If, however, this is however not the case we update our guess of $V(a)$ using $V_{\text{new}}(a)$ and go back to step 2.

It is possible to show theoretically that the result of this approach converges to the true solution of our dynamic program using the *contraction mapping theorem*.

Program 8.3 shows a value function iteration for the cake-eating problem. We will now discuss how to fill the intermediate part on computing policies and value functions at the gridpoints. The first part of the program defines the discrete state space. The gridpoints are chosen to be equidistant as in formula (8.13). We can construct equidistant nodes using the subroutine `grid_Cons_Equi` and store them in the array `a`, see Section 2.5. Then the value function is initialized at 0 and the iteration process can begin. Note that a maximum number of iterations is defined in case the algorithm does not converge because of a programming error. In such a case the program prints an error message to the console. In each iteration step the optimal policies and value function are computed. The new values of the value function at the gridpoints are stored in the array `v_new`. Then the variable `con_lev` computes the maximum relative difference between the old

Program 8.3 Value function iteration

```

! initialize a and value function
call grid_Cons_Equi(a, 0d0, a0)
V(:) = 0d0

! iterate until value function converges
do iter = 1, itermax

    ! determine the policies and value function at the gridpoints
    c(:) = [.....]
    V_new(:) = [.....]

    ! get convergence level
    con_lev = maxval(abs((V_new(:)-V(:))/max(abs(V(:)), 1d-10)), 1)
    write(*,'(i5,2x,f20.7)')iter, con_lev

    ! check for convergence
    if(con_lev < sig)then
        call output()
    endif

    V = V_new
enddo

write(*,*) 'No Convergence'

```

and the new value function as described in step 3 above. If this difference is below the tolerance level specified in the variable `sig`, a subroutine that creates some output graphs is called up and the program ends. If the tolerance level is not reached, we update the value function by setting `V = V_new` and start with a new iteration step.

Having learned how to calculate a fixed point of our dynamic program with the help of value function iteration, we will now be concerned with how to calculate the new policies given a guess for the future value function. In the following we introduce different methods of doing this which vary both in degree of accuracy and speed but also in applicability.

8.2.1 GRID SEARCH

Grid search is probably the most intuitive way of solving a dynamic program. In order to understand how it works, we again write the optimization problem as in (8.14), i.e.

$$\max_{a^+} u(a - a^+) + \beta V(a^+).$$

Figure 8.5 shows (in a stylized way) what the two parts of the above function look like and how the maximum is determined. Given a current remaining value of resources a (in this case $a = 100$) and for any possible future resource value $0 \leq a^+ \leq a$, we can compute current utility $u(c) = u(a - a^+)$ as well as the value of the discounted future

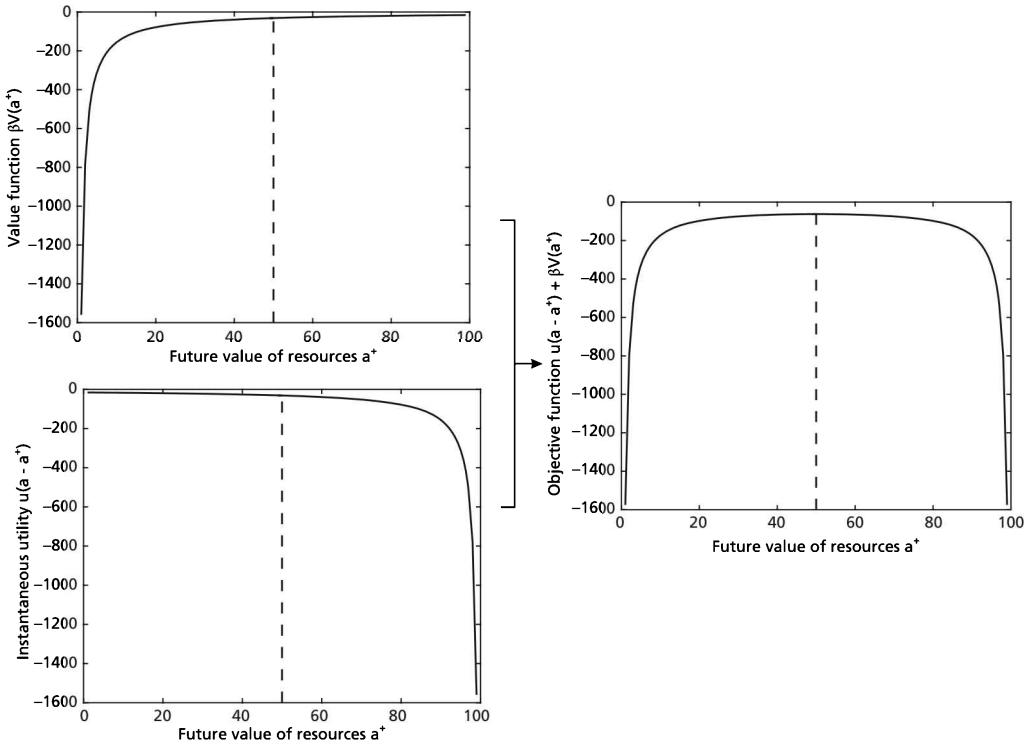


Figure 8.5 Optimization process of the cake-eating problem (for $a = 100$)

value function $\beta V(a^+)$. The sum of both is the objective function of the maximization problem $u(c) + \beta V(a^+)$. The maximum value of this objective function determines the optimum value $a^{+,*}$ as well as the related optimal consumption value $c^* = a - a^{+,*}$. Doing this for all a leads to the functions $c(a)$ and $a^+(a)$.

As already mentioned above, the problem is that in general we do not know the functional form of the value function $V(a)$. We therefore discretize the state space $[0, a_0]$ by means of a finite grid $\mathcal{A} = \{\hat{a}_0, \hat{a}_1, \dots, \hat{a}_n\}$. In the example of Figure 8.6 these points are again uniformly distributed. The grid-search algorithm now computes the solution to our dynamic program as follows: For any given resource level $a = \hat{a}_v \in \mathcal{A}$ and any potential level of future resources $a^+ = \hat{a}_k$ with $k < v$ we compute the utility

$$u(\hat{a}_v - \hat{a}_k) + \beta V(\hat{a}_k).$$

The optimal decision $a^{+,*}$ at the current resource level $a = \hat{a}_v$ then is the \hat{a}_k that maximizes the utility function, see Figure 8.6. In a formula we can write this as

$$V_{new}(\hat{a}_v) = \max_{\{\hat{a}_k\}_{k=0}^{v-1}} u(\hat{a}_v - \hat{a}_k) + \beta V(\hat{a}_k).$$

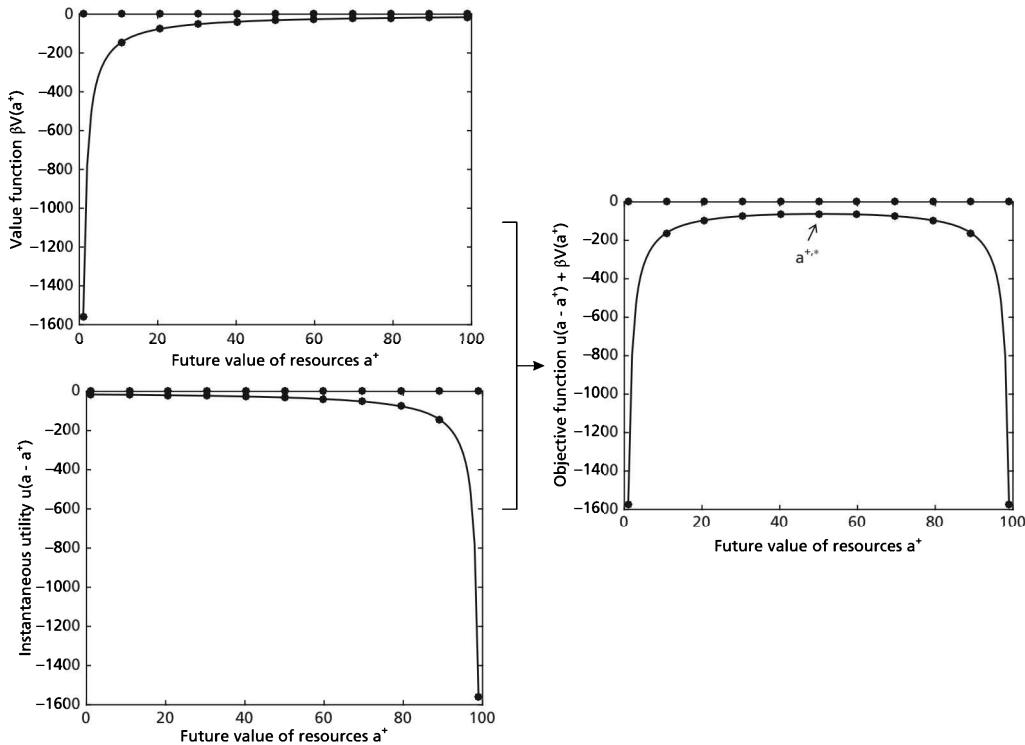


Figure 8.6 Discretization and optimization

From this optimizing procedure we can derive the discrete decision function $a^+(\hat{a}_v)$ as well as the policy function $c(\hat{a}_v)$.

Program 8.3.a shows how to implement this procedure in a very straightforward way. We start our grid-search mechanism by manually setting the decision in the situation in which nothing of the resource is left, i.e. $a(0)$. In this case there is nothing left to consume and we set the value function to something large and negative.⁵ We then iterate over all the other resource values $a(ia)$ and compute for any possible future resource value $a(ia_p)$ the value of the utility function which is stored in the array $u_temp(0:ia-1)$. Having computed these potential utility values, we determine the optimal resource value of the next period ia_opt as the one that maximized u_temp . Note that we only store the index of the gridpoint the agent would move to next period and not the value of the future resource. Note further that the intrinsic Fortran function `maxloc` always returns the maximum entry of an array starting counting from 1. Since the first index of our array u_temp is 0, we have to subtract 1 from the result returned by `maxloc`. Finally the policy function c as well as the utility value resulting from the optimization process v_new are computed.

⁵ By setting $V_new(0) = V(1) - 100d0$ we ensure that the value function has the most negative point at a_0 .

Program 8.3.a Grid search

```

program GridSearch
[.....]
! set a = 0 manually
c(0) = 0d0
ia_opt(0) = 0
V_new(0) = V(1)-100d0

! calculate optimal decision for every gridpoint
do ia = 1, NA

    ! get utility for every possible a_p
    do ia_p = 0, max(ia-1, 0)

        ! calculate consumption
        cons = max(a(ia) - a(ia_p), 1d-10)

        ! calculate utility
        u_temp(ia_p) = cons**egam/egam + beta*V(ia_p)
    enddo

    ! get optimal a_p, consumption and value function
    ia_opt(ia) = maxloc(u_temp(0:max(ia-1, 0)), 1)-1
    c(ia) = a(ia)-a(ia_opt(ia))
    V_new(ia) = u_temp(ia_opt(ia))
enddo
[.....]
end program

```

Program 8.3.b Simulation of a consumption path for grid search

```

subroutine output()
[.....]
! calculate the time path of consumption numerically
ia_t(0) = NA
c_t(0) = c(NA)
do it = 1, TT
    ia_t(it) = ia_opt(ia_t(it-1))
    c_t(it) = c(ia_t(it))
enddo
call plot(c_t)
[.....]
end subroutine

```

Together with our value function iteration process, Program 8.3 determines the value and policy function of the cake-eating optimization problem. What remains to be discussed is how we simulate a consumption path $\{c_t\}_{t=0}^{\infty}$ over time using these numerical results. This simulation process takes place in the subroutine `output` which is shown in Program 8.3.b. We start out by setting the index of the initial resource level to `NA` which means we set $a_0 = \hat{a}_n$ and compute the consumption value at this gridpoint. We then successively go through time and for each period compute the gridpoint we are currently at by checking the optimal solution in the previous year. The process continues until we reach the maximum number of simulation periods `TT`.

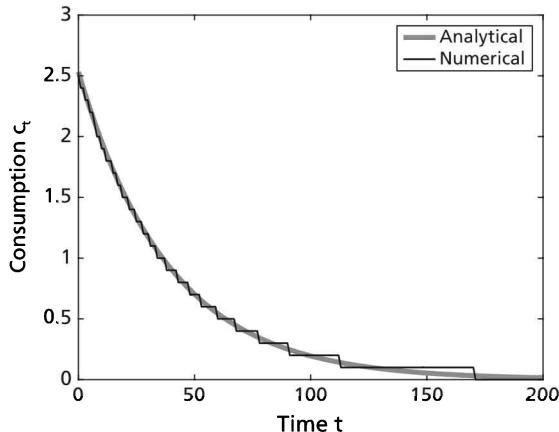


Figure 8.7 Consumption path resulting from grid search

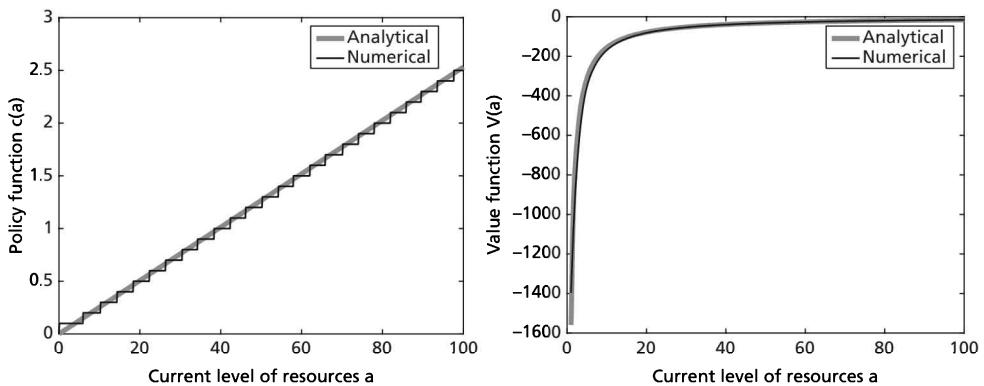


Figure 8.8 Policy and value function resulting from grid search

Figure 8.7 shows the simulated consumption path and Figure 8.8 the policy and value function that result from our grid-search algorithm for the parameters $\gamma = 0.5$, $\beta = 0.95$, and $a_0 = 100$ as well as $n = 1,000$. Both figures compare the numerical solution (black line) with the analytical solution (grey line) we obtained above. Note that the simulated consumption path as well as the policy and value function are not differentiable in many areas. Rather they are step functions. This is of course an artefact of the decision possibilities, i.e. the choice set for a^+ , being discrete and not continuous. All in all we cannot consider the grid-search algorithm to deliver very accurate solutions. One way to test the accuracy of our solution is to calculate consumption function errors, i.e. the maximum relative difference between the numerical consumption function and the analytical solution to the consumption function. Formally this means calculating

$$\max_{v \in \{0,1,\dots,n\}} \left| \frac{c(\hat{a}_v) - \hat{a}_v(1 - \beta^\gamma)}{\hat{a}_v(1 - \beta^\gamma)} \right|.$$

This is done in the last part of subroutine `output` for all the different gridpoints. The size of the consumption function error ranges around 10, meaning around 1000 per cent. This is very inaccurate as we will find out later.

The computational time needed for the value function iteration to find a fixed point is around 30 seconds. This results from our straightforward computation of the grid-search algorithm. Using the intrinsic routine `maxloc` is certainly extremely easy, yet it has one major disadvantage. By tackling the grid search problem this way, we have to calculate the utility levels of all possible values of $a^+ \in \{\hat{a}_0, \dots, \hat{a}_{v-1}\}$ for each \hat{a}_v . Given the structure of our problem, however, there is a much more efficient way to perform a grid search. When we look at Figure 8.6 again, we can see that there is a unique maximum of our value function somewhere in between 0 and the amount of remaining resources a_v . A more efficient way of solving for the $a^{+,*} = \hat{a}_k$ that maximizes the agent's utility might therefore be the following:

1. Start by setting $a^+ = a_{v-1}$. Compute the utility

$$u_{temp} = u(\hat{a}_v - \hat{a}_{v-1}) + \beta V(\hat{a}_{v-1}).$$

2. Now iterate backward for $k = v - 2, v - 3, \dots, 0$ and calculate

$$u_{new} = u(\hat{a}_v - \hat{a}_k) + \beta V(\hat{a}_k).$$

If $u_{new} > u_{temp}$ set $u_{temp} = u_{new}$. If however $u_{new} \leq u_{temp}$ then the value function will only decrease further as we iterate. Consequently u_{temp} is the maximum amount of utility the agent can get. The optimal decision at the gridpoint \hat{a}_v , therefore is \hat{a}_{k+1} with a utility level of u_{temp} . We can then terminate the iteration process.

The great advantage of this algorithm is that we do not have to calculate the agent's utility for any possible \hat{a}_k , but only for selected levels. Program 8.4 shows how to implement this algorithm. The computational time using this algorithm decreases from about 30 seconds to roughly 1 to 2 seconds. However, one deficiency of this algorithm remains, namely the fact that neither policy nor value function are continuous along the interval $[0, a_0]$. What follows shows how to overcome this issue.

8.2.2 OPTIMIZATION AND INTERPOLATION

In order to make the policy and value function continuous, we have to find a way to again create a continuous function out of the discrete set of value function values $\{V(\hat{a}_v)\}_{v=0}^n$. To achieve this, we have to use an interpolation routine that interpolates the value function in between the gridpoints. Chapter 2 showed us a couple of interpolation routines we can apply. Since we know that our value function is actually a smooth

Program 8.4 How to speed up grid search

```

ia_opt(ia) = max(ia-1, 0)
cons = max(a(ia) - a(ia_opt(ia)), 1d-10)
u_temp = cons**egam/egam + beta*V(ia_opt(ia))

! check whether there is an a_p that gives higher utility
do ia_p = max(ia-2, 0), 0, -1

    ! calculate consumption and new utility
    cons = max(a(ia) - a(ia_p), 1d-10)
    u_new = cons**egam/egam + beta*V(ia_p)

    ! test which utility level is higher
    if(u_new > u_temp)then
        ia_opt(ia) = ia_p
        u_temp = u_new
    else
        exit
    endif
enddo

! get optimal consumption and value function
c(ia) = a(ia)-a(ia_opt(ia))
V_new(ia) = u_temp

```

function, spline interpolation is a useful tool to apply here.⁶ We therefore calculate a spline function S that satisfies the interpolation conditions

$$S(\hat{a}_v) = V(\hat{a}_v) \quad \text{for all } v = 0, \dots, n.$$

Having calculated this function, we can now evaluate the value function at any arbitrary point $a^+ \in [0, a_0]$ by just calculating $S(a^+)$.

Doing this, however, we might run into a small problem. The cause of this problem is the fact that the value function diverges to minus infinity as soon as a approaches 0. Any interpolating function that does not have this same property will always have difficulty approximating the true function values of the value function near values of $a = 0$. So does the spline function as is illustrated in the left panel of Figure 8.9. The figure is constructed by interpolating the analytical value function on the interval $[0, a_0]$ using 16 interpolation nodes, i.e. discretizing the state space with $n = 15$. The spline function obviously perfectly matches the value function in the interpolation nodes \hat{a}_v , as it is explicitly constructed to do so. In between two points, however, it shows some weird behaviour. As the spline function needs to increase from a value of -1600 to -200 very quickly between the first and the second interpolation node, it starts oscillating afterwards. This is something that we absolutely do not want to see in a value function as it would lead to weird results in the optimization process.

⁶ In case of a non-smooth function, linear interpolation would be the more appropriate routine.

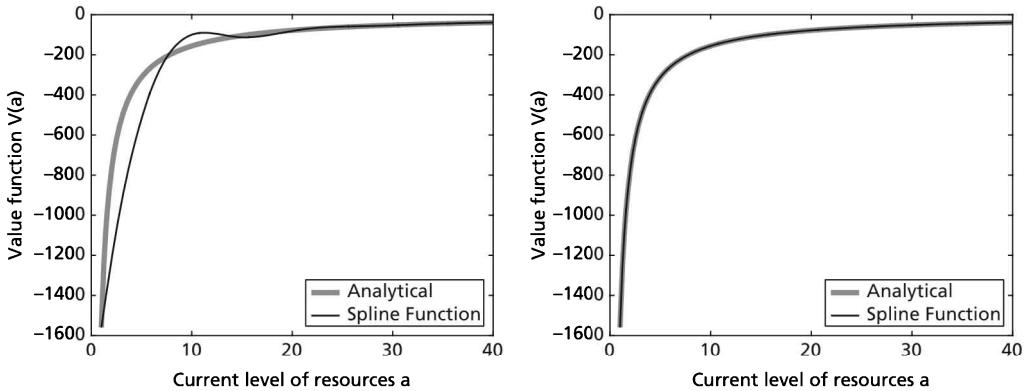


Figure 8.9 Interpolation problem when function diverges to minus infinity

There are a couple of fixes for this problem. We could increase the number of interpolation nodes, i.e. the number of gridpoints n . We could try to allocate more interpolation nodes around the steep area of the value function to increase interpolation accuracy there. Or we could use so-called shape-preserving spline interpolation, which is much more complicated to calculate. All these fixes would, however, lead to a significant increase in computational time. Yet there is quite an easy way to overcome this interpolation problem that works for many classes of functions. In the case of our utility function $\frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}}$ we already know that the value function has the form $V(a) = B \cdot \frac{a^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}}$ with some factor B . Consequently we could try to interpolate the function $\left[\left(1 - \frac{1}{\gamma}\right) \cdot V(a)\right]^{\frac{1}{1-\frac{1}{\gamma}}}$ instead of $V(a)$, i.e. we could determine a spline function that satisfies

$$S(\hat{a}_v) = \left[\left(1 - \frac{1}{\gamma}\right) \cdot V(\hat{a}_v) \right]^{\frac{1}{1-\frac{1}{\gamma}}} = B^{\frac{1}{1-\frac{1}{\gamma}}} \cdot \hat{a}_v \quad \text{for all } v = 0, \dots, n.$$

The great advantage of this is that now the spline function only has to approximate a function that is linear in a and does not diverge to minus infinity for a approaching zero.⁷ The original value function can then always be restored by calculating

$$V(a) = \frac{S(a)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

⁷ Obviously spline interpolation is overshooting a little here, since we could interpolate a linear function just by means of a linear function. Yet we want to illustrate the general approach to dynamic programming that would also be applicable to more complicated problems.

The right panel of Figure 8.9 shows the result of this interpolation process. Obviously in this case the spline perfectly fits the value function, even when only $n = 15$ interpolation nodes are used.

Applying an interpolation method to our discrete value function $\{V(\hat{a}_v)\}_{v=0}^n$, we can again write our optimization problems in continuous form, i.e. our set of optimization problems now reads

$$V_{new}(\hat{a}_v) = \max_{a^+} u(\hat{a}_v - a^+) + \beta \frac{S(a^+)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

We can tackle this with a numerical minimization routine as described in Chapter 2.

Program 8.5 shows how to implement the whole process. The value function iteration process is in this way exactly the same as in the previous programs. Our subroutines for interpolation and minimization are included in the toolbox. Within one iteration of the value function iteration process, we now determine optimal policy and the value function by means of our numerical routines. We therefore again first set the solution for the case in which no resources are left $a(0)$. At this point consumption is zero and the value function takes a very negative value. We then iterate over all other possible gridpoints. At each of these gridpoints, we use the function `fminsearch` to determine our optimal policy. This function takes five arguments: a scalar `x_in` which contains

Program 8.5 Solving the cake-eating problem with optimization and interpolation

```

program Minimize
[.....]
do iter = 1, itermax

  ! set a = 0 manually
  c(0) = 0d0
  V_new(0) = -1d10

  ! calculate optimal decision for every gridpoint
  do ia = 1, NA

    ! initialize starting value and communicate resources
    x_in = a(ia) - c(ia)
    ia_com = ia

    call fminsearch(x_in, fret, 0d0, a(ia), utility)

    ! get optimal consumption and value function
    c(ia) = a(ia) - x_in
    V_new(ia) = -fret

  enddo

  ! interpolate coefficients
  call spline_interp((egam*V_new)**(1d0/egam), coeff_V)
  [.....]
enddo
end program

```

the initial guess for the optimum, a scalar `fret` in which the subroutine stores the minimum function value, the left and right corner of the interval on which the subroutine should search for a minimum and the function that should be minimized, in our case a function called `utility`, see below. Note that we constructed the function `utility` such that `fminsearch` again optimizes over future resource values and not directly over the control variable c . As an initial guess, we take the optimal decision from the previous iteration of the value function iteration process. The left and right interval endpoints are 0 as well as the remaining amount of resources today `a(ia)`. After `fminsearch` has determined the minimum of the function `utility`, we store the respective optimal policy and value function in the arrays `c` and `v_new`. `fminsearch` returns the result of the optimization process in the variables `x_in` and `fret`. We have to use the negative of `fret` here as `fminsearch` minimizes the negative of the utility function.⁸ Finally, after we have determined the optimal policy and value function for every possible gridpoint `a(ia)`, we can interpolate the new value function `v_new`. We therefore use the transformation discussed above. The interpolation process can be achieved by means of the function `spline_interp`. This function takes the function values of the value function we have stored in the array `v_new(0:NA)`. It stores the result of the interpolation process in an array `coeff_v(NA+3)` that needs two more entries than the value function vector.

The function `utility` is located in the module `globals` together with all variables. Module 8.5m shows an excerpt of this module. The function has as input the potential future asset level `x_in`. Furthermore it uses the variable `ia_com`. This variable is what we call a communication variable. It tells the function `utility` for which actual asset level a to calculate the utility function. Owing to the restrictions of the Fortran language, the function we want to minimize can only have one argument. All other arguments have to be given to the function through communication variables. As we store the communication variables as well as any other variable in the module `globals`, the function can directly access them. Furthermore the function has to use the `toolbox`, which includes the spline interpolation routines. It contains the function `spline_eval` with which we can calculate the value of the spline function with coefficients `coeff_v` at any point. Within the function we proceed as follows: We first compute the value of current consumption given a current resource level `a(ia_com)` as well as the potential future resources `x_in`. We limit consumption to a minimum of 10^{-10} in order to avoid calculation errors. Having calculated current consumption, we evaluate the spline function at the future resource level $a^+ = x_{in}$. The function `spline_eval` therefore takes the point at which to evaluate the spline function `x_in`, the coefficients that define the spline `coeff_v`, as well as the interval limits on which we interpolated the value function as an input argument. Since the value function was calculated on the interval $[0, a_0]$ we use the respective interval bound as input variables. We also limit the value of the spline function to 10^{-10} to avoid computational errors. Note that we again have to transform

⁸ Which is equivalent to maximizing the utility function itself.

Module 8.5m Module for solving optimization problems with spline interpolation

```

module globals
    [.....]
    ! variables to communicate with function
    integer :: ia_com

contains

    ! the function that should be minimized
    function utility(x_in)

        use toolbox

        implicit none
        real*8, intent(in) :: x_in
        real*8 :: utility, cons, vplus

        ! calculate consumption
        cons = max(a(ia_com) - x_in, 1d-10)

        ! calculate future utility
        vplus = max(spline_eval(x_in, coeff_v, 0d0, a0), &
                    1d-10)**egam/egam

        ! get utility function
        utility = - (cons**egam/egam + beta*vplus)

    end function

end module

```

Program 8.5.a How to simulate the consumption path

```

! interpolate policy function
call spline_interp(c, coeff_c)

! calculate the time path of consumption numerically
a_t(0) = a0
c_t(0) = spline_eval(a_t(it), coeff_c, 0d0, a0)
do it = 1, TT
    a_t(it) = a_t(it-1) - c_t(it-1)
    c_t(it) = spline_eval(a_t(it), coeff_c, 0d0, a0)
enddo
call plot(c_t)

```

the spline function so that we obtain the original function value of the value function, see discussion above. Finally, we calculate the negative of the agent's utility function for the combination of current resources and potential future resources.

Finally we have to think about how to simulate the consumption path over time in this program. Program 8.5.a shows how this is done in the subroutine `output`. The algorithm we apply here is basically the same as for the analytical case, in which we had a closed-form solution for the policy function, see Program 8.2. Thus we again use our interpolation technique to interpolate the policy function in between the chosen gridpoints `a(ia)`. Since the policy function is linear, we can use spline interpolation or

could even use linear interpolation. Having found the respective coefficients of the spline function, we can simulate the consumption path forward as in the analytical case.

Figure 8.10 shows the simulated consumption path and Figure 8.11 the respective policy and value function for the parameters $\gamma = 0.5$, $\beta = 0.95$, and $a_0 = 100$ as well as $n = 1,000$. First of all, we find that the algorithm provided here is much more accurate in determining the consumption path as well as policy and value function. We can again formally check this by calculating consumption function errors. This is done in the last part of subroutine `output`. Since our consumption function is now continuous and not discrete as when we applied grid search, we can calculate the consumption function errors for many more points (namely 10,000 in this example). The consumption function error lies within the range of 10^{-5} , which is way more accurate than the grid search method. When we look at how fast this algorithm performs, we get a run time of roughly 7 seconds. However, we have to keep in mind that we used $n = 1000$ gridpoints for the discretization of the state space. For the current algorithm this is not necessary, since it is so accurate.

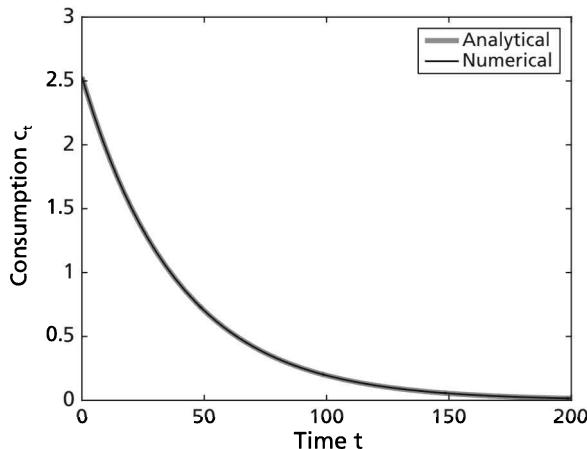


Figure 8.10 Consumption path resulting from optimization and interpolation

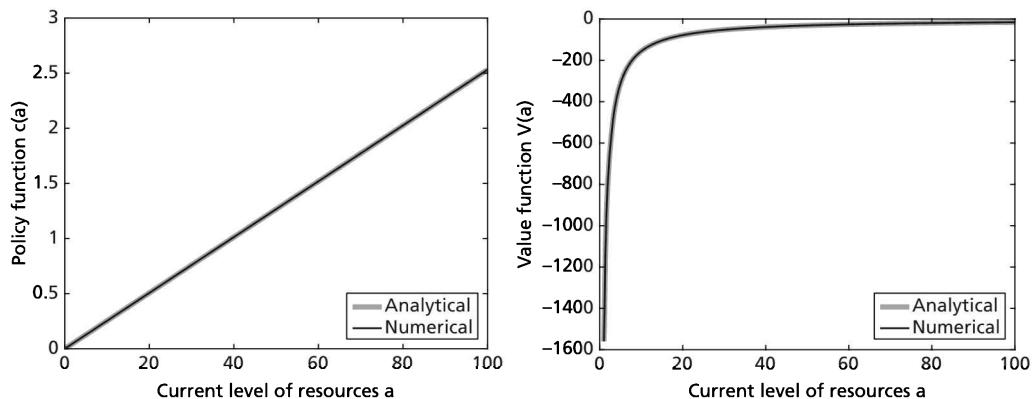


Figure 8.11 Policy and value function resulting from optimization and interpolation

Actually we can get results with a very similar consumption function error using only $n = 20$. You can check this on your own. In this case the program runtime decreases to a fraction of a second.

8.3 Numerical solution by policy function iteration

Up to this point, we used the utility function of the agent directly to determine his optimal decisions at the gridpoints and iterated over the value function until we found a fixed point. This procedure, however, turns out to be fairly slow, especially in the case of infinite horizon problems. In fact, it can be shown that the convergence speed of this algorithm is tied to the discount factor β , where a β close to 1 results in slow convergence. In addition, solving minimization problems numerically is not the fastest possible method either. An alternative approach that replaces the value function iteration procedure and can be applied to many problems is called *iteration in policy space* or *policy function iteration*. This algorithm is quite similar to the value function iteration procedure, but the iteration process is carried out on the policy function.⁹ The approach reduces computational time, since in many cases it results in faster convergence and it allows us to derive the policy function without evaluating the value function. In order to implement the policy function iteration in the most efficient way, we simply use the first-order condition $u'(c) = \beta V'(a^+)$ in order to compute a new policy function in each iteration step. The algorithm proceeds in the following steps:

1. Start with a (more or less arbitrary but feasible) initial guess of the policy function, e.g. $c(a) = \frac{a}{2}$.
2. Determine the optimal policies $\{c_{\text{new}}(\hat{a}_v)\}_{v=0}^n$ for all $\hat{a}_v \in \mathcal{A}$ by solving the set of first-order conditions

$$\hat{a}_v - a^+ - \beta^{-\gamma} c(a^+) = 0.$$

3. Check for convergence i.e. calculate

$$\max_{\hat{a}_v} \left| \frac{c_{\text{new}}(\hat{a}_v) - c(\hat{a}_v)}{c(\hat{a}_v)} \right| < \epsilon.$$

4. If the iteration hasn't converged, update $c(a) = c_{\text{new}}(a)$ and repeat from step 2.

⁹ The term *policy function iteration* is often also used to describe *Howard's improvement algorithm*, which is actually an improved version of the standard value function iteration. Since Section 8.3 we only work on policy functions and not on value functions, we think policy function iteration is a good description. Others also like to call the methods we discuss here *Euler equation-based methods*.

Like with value function iteration, we need to determine optimal policies during each iteration step of the policy function iteration. There are actually two ways to do this: In subsection 8.3.1 we use the root-finding subroutine `fzero` together with spline interpolation of the policy functions in order to solve the first-order condition. The root-finding procedure was introduced in Chapter 2. Then we introduce the method of endogenous grid points which does not require any numerical optimization routine and further reduces computational demands.

8.3.1 ROOT-FINDING AND INTERPOLATION

In our model, consumption is determined by the first-order condition

$$c = \beta^{-\gamma} c^+ \Leftrightarrow c - \beta^{-\gamma} c^+ = 0 \Leftrightarrow a - a^+ - \beta^{-\gamma} c(a^+) = 0,$$

see equation (8.11). The last equivalence is obtained by plugging in the dynamic budget constraint of the problem and acknowledging the fact that consumption in the next period is again only a function of the remaining resources at this date. We can therefore determine our consumption function $c(\hat{a}_v)$ by solving the set of root-finding problems

$$\hat{a}_v - a^+ - \beta^{-\gamma} c(a^+) = 0 \quad \text{for all } v = 0, 1, \dots, n.$$

As before, $c(\hat{a}_v)$ is only a set of discrete points. To make it a continuous function again, we construct a spline function S that satisfies

$$S(\hat{a}_v) = c(\hat{a}_v) \quad \text{for all } v = 0, 1, \dots, n.$$

We therefore use the values of the consumption function from the previous iteration step. Our root-finding problems then read

$$\hat{a}_v - a^+ - \beta^{-\gamma} S(a^+) = 0 \quad \text{for all } v = 0, 1, \dots, n.$$

These fairly simple one-dimensional root-finding problems can be tackled with the root-finding routine `fzero` out of the toolbox.

Program 8.6 shows how to implement this algorithm. The first step is to again calculate the gridpoints and make an initial guess for the consumption levels at these point. We use a fairly uneducated guess of $c(\hat{a}_v) = \frac{\hat{a}_v}{2}$. We have to interpolate the consumption function using spline interpolation to give our root-finding problem an initial guess. The iteration process then starts. Yet, this time we do not determine new value functions but only new consumption functions. The new guess for the consumption function is stored in the array `c_new`. For `a(0)` we again set the value of the consumption function manually to zero. For all other gridpoints we determine consumption through finding

Program 8.6 Determining the policy function with first-order conditions

```

program FirstOrder
    [.....]
    ! initialize a and c
    call grid_Cons_Equi(a, 0d0, a0)
    c(:) = a(:)/2d0
    call spline_interp(c, coeff_c)

    ! iterate until policy function converges
    do iter = 1, itermax

        ! set a = 0 manually
        c_new(0) = 0d0

        ! calculate optimal decision for every gridpoint
        do ia = 1, NA

            ! initialize starting value and communicate resources
            x_in = a(ia) - c(ia)
            ia_com = ia
            check = .false.

            call fzero(x_in, foc, check)

            if(check)write(*,'*')'ERROR IN ROOTFINDING PROCESS'

            ! get optimal consumption and value function
            c_new(ia) = a(ia) - x_in

        enddo

        ! interpolate coefficients
        call spline_interp(c_new, coeff_c)

        ! get convergence level
        con_lev = maxval(abs(c_new(:) - c(:)) / max(abs(c(:)), 1d-10))
        write(*,'(i5,2x,f20.7)')iter, con_lev

        ! check for convergence
        if(con_lev < sig)then
            call output()
        endif

        c = c_new
    enddo
    [.....]
end program

```

the root of the function `foc`. We therefore use the subroutine `fzero` out of the toolbox, see Chapter 2 for further information. This subroutine takes an initial guess `x_in` as well as the function that it should find a root of. It furthermore gives the logical variable `check` the value `.true.` in case something goes wrong during the root-finding process. Having computed the new consumption values `c_new` at the gridpoints we first update our spline coefficients and then check for convergence of our policy function iteration. We apply the same criterion as in the value function iteration procedure. Finally, if the process hasn't converged yet, we store the new consumption function in array `c`.

Module 8.6m First-order condition

```

function foc(x_in)
    use toolbox

    implicit none
    real*8, intent(in) :: x_in
    real*8 :: foc, cplus

    ! calculate right hand side of foc
    cplus = spline_eval(x_in, coeff_c, 0d0, a0)

    ! get foc
    foc = a(ia_com) - x_in - beta**(-gamma)*cplus

end function

```

The module `globals` is sketched in Module 8.6m. This module now contains a function `foc`, which calculates the value of the first-order condition. It takes as input a scalar `x_in` which again is the amount of resources that should be left for the next period. Furthermore it uses the communication variable `ia_com`. For the potential resource level `x_in` it then determines tomorrow's consumption level by evaluating the spline function at `x_in`. Finally, it calculates the value of the first-order condition at `x_in`.

When we run the program we first of all see that its runtime is only about 0.4 seconds even for $n = 1,000$. This is certainly a result of the fact that a root-finding routine generally converges much faster than a minimization routine. When it comes to accuracy, the consumption function errors are about the same as in Program 8.5. Obviously, we can again create an even faster runtime by reducing n to say 20. Accuracy of the solution can then even be increased by rising the tolerance level `sig` e.g. to 10^{-8} . All in all we find that working on first-order conditions is much more efficient in terms of computation than using minimization routines. Therefore, whenever we can solve a problem by means of solving a first-order condition, we should apply this method.

8.3.2 THE METHOD OF ENDOGENOUS GRIDPOINTS

The method of endogenous gridpoints is probably the most efficient method to solve a certain set of dynamic programs. Its great advantage is that it does not need a root-finding or minimization routine at all, but only relies on an analytical solution of the first-order condition. It is called the method of *endogenous* gridpoints, since the discrete set of points on the state space is endogenously determined.

Like our root-finding method, the method of endogenous gridpoints uses the first-order conditions

$$u'(c) = \beta V'(a^+) \quad \text{s.t.} \quad a^+ = a - c$$

to solve the dynamic program. In order to make it work, we first have to again define an *exogenous grid* on the state space $\{\hat{a}_v\}_{v=0}^n$. Yet, in contrast to our previous methods, an exogenous gridpoint \hat{a}_v does not denote the current level of resources a , but the remainder of resources as of tomorrow, i.e. $a^+ = \hat{a}_v$.

To solve the dynamic programming problem the method of endogenous gridpoints proceeds as follows: For every potential remaining resource level \hat{a}_v it looks at the first-order condition and asks what would consumption be today if the agent behaved optimally and left resources of an amount \hat{a}_v for tomorrow. As long as u' is an invertible function we can easily answer this question from the first-order condition

$$c = (u')^{-1} [\beta V'(\hat{a}_v)].$$

In the case of the cake-eating problem we can write this condition as

$$c = \beta^{-\gamma} c(\hat{a}_v).$$

Thereby $c(\hat{a}_v)$ can again be calculated from the policy function derived in the previous iteration step of the iteration in policy space. When we now plug this into the dynamic resource constraint, we obtain a level of current resources called the *endogenous gridpoints*

$$a^+ = a - c \Leftrightarrow a = a^+ + c \Rightarrow \tilde{a}_v := \hat{a}_v + \beta^{-\gamma} c(\hat{a}_v). \quad (8.15)$$

The method of endogenous gridpoints therefore tells us the following: If the remaining resource level of tomorrow would be \hat{a}_v , what would consumption and resources have to be today if the agent behaved optimally. The answer to this question is an endogenous grid of resource levels $\{\tilde{a}_v\}_{v=0}^n$, together with respective consumption levels $\{c(\tilde{a}_v)\}_{v=0}^n$ at the endogenous gridpoints. Figure 8.12 shows us what such a set of endogenous gridpoints could look like in the cake-eating problem.

The grey dots on the x-axis are the exogenously chosen gridpoints \hat{a}_v . The black dots mark the respective endogenous gridpoints. From equation (8.15) it is clear that the endogenous gridpoint values have to be greater than those of the exogenous gridpoints. The endogenous gridpoint method therefore delivers a new policy function that is characterized by the endogenous gridpoint levels $\{\tilde{a}_v\}_{v=0}^n$ and the consumption values $\{c(\tilde{a}_v)\}_{v=0}^n$.

One remaining problem is that in order to be able to compare the new policy function with the policy function of the previous iteration step, we need to again know the new consumption values $\{c(\hat{a}_v)\}_{v=0}^n$ at the exogenous gridpoints. These values can be relatively easily calculated by interpolating the consumption function $\{c(\tilde{a}_v)\}_{v=0}^n$ in between the endogenous gridpoints. Note that $\{\tilde{a}_v\}_{v=0}^n$ is an arbitrary set of points and not an evenly spaced grid anymore. Therefore our spline interpolation routine does not work with the endogenous gridpoint method. However the toolbox provides a simple piecewise

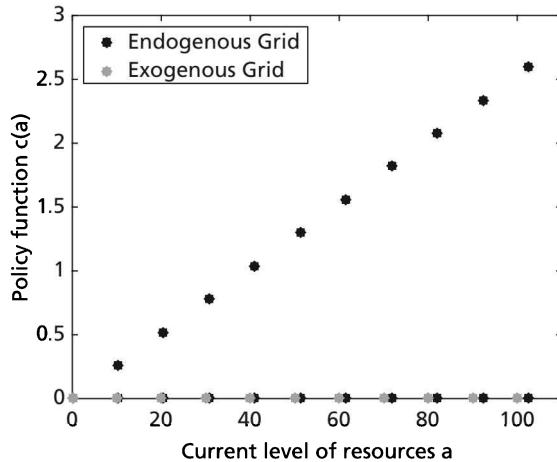


Figure 8.12 Endogenous gridpoints and consumption values

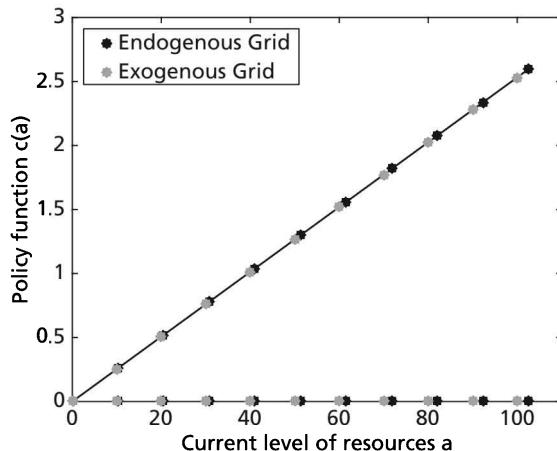


Figure 8.13 Interpolation of consumption values

linear interpolation algorithm that we can use instead, see below for further discussion. Linear interpolation just draws straight lines between the consumption values at the endogenous gridpoints. The consumption values at the exogenous gridpoints can then be easily calculated as we can see from Figure 8.13.

Program 8.7 shows how to implement the solution to the cake-eating problem with the method of endogenous gridpoints. Note that the only thing that changes compared to Program 8.6 is how we calculate the new policy function. The policy function iteration algorithm itself remains exactly the same. In each iteration step the method of endogenous gridpoints calculates the new consumption levels which are stored in the array `c_endog` as well as the endogenous gridpoints `a_endog` from equation (8.15). The new consumption function `c_new` at the exogenous gridpoints is then computed by the piecewise linear interpolation scheme in the function `linint_Gen` from the toolbox.

Program 8.7 Cake-eating and endogenous gridpoints

```

program EndogenousGrid
    [.....]
    do iter = 1, itermax

        ! set a = 0 manually
        a_endog(0) = 0d0
        c_endog(0) = 0d0

        ! calculate optimal decision for every gridpoint
        do ia = 1, NA

            ! calculate endogenous gridpoint and consumption
            c_endog(ia) = c(ia)*beta**(-gamma)
            a_endog(ia) = a(ia) + c_endog(ia)

        enddo

        ! stretch out to exogenous grid again
        do ia = 0, NA
            c_new(ia) = linint_Gen(a(ia), a_endog, c_endog, ia)
        enddo
        [.....]
    enddo
    [.....]
end program

```

`linint_Gen` stands for *general linear interpolation* on arbitrarily spaced grids. In Section 2.6.2 we already learned how to interpolate with piecewise linear functions on equidistant grids. Equidistant grids have the advantage that, once we know the left and the right endpoint \underline{a} and \bar{a} as well as the number of gridpoints n , we can calculate the complete grid using the simple formula

$$\hat{a}_v = \underline{a} + \frac{\bar{a} - \underline{a}}{n} \cdot v, \quad \text{for } v = 0, \dots, n.$$

Our exogenous grid follows such a regular pattern. Since this formula can be inverted and solved for v , interpolation on such a grid is very simple. Unfortunately we do not know what the pattern of the endogenous grid is as it is some ‘arbitrary’ function of the exogenous grid. The subroutine `lininit_Gen` can deal with such irregular grids, given that the grid is provided in ascending order.¹⁰ For endogenous grids resulting from well-behaved, convex problems, this is usually the case. The subroutine takes three mandatory and one optional input variable. The first argument is the point at which the piecewise linear interpolant should be evaluated. As we want to know the values of the consumption policy function at the exogenous grid, this first input is the exogenous gridpoint `a(ia)`. The second and third input define the x- and y-data that should be interpolated. In our example, the x-data are the endogenous gridpoints `a_endog` and the y-data the policy

¹⁰ The subroutine can obviously also deal with equidistant grids. But for those the subroutine `linint_Equi` is much more efficient.

function at the endogenous grid `c_endog`. The last argument is an optional input. It can be used to speed up the process of interpolation by providing a rough guess for a datapoint in the `x-data`, which is close to the point at which we want to interpolate. Taking a look again at Figure 8.13, we can conjecture that it is quite likely that the exogenous gridpoint `a(ia)` and the endogenous point `a_endog(ia)` lie somewhat close together. Hence `ia` should be a good guess.¹¹

When we run Program 8.7 for the same parameter combination as previously, we find that it performs better than any implementation we have seen so far. This is not surprising, since the method of endogenous gridpoints does not rely on any form of minimization or root-finding and therefore avoids the frequent calculation of a value function or first-order condition. The method of endogenous gridpoints, however, has its limits. First, it relies on an analytical solution of the first-order condition. As soon as we can't solve the first-order condition analytically anymore, we again have to use some form of root-finding algorithm. Second, the method of endogenous gridpoints is only easily applicable to problems with a one-dimensional state space. In this case we can use the piecewise linear interpolation scheme to derive the values of the policy function at the exogenous grid from the endogenous gridpoints. In more than one dimension however, we would have to interpolate on an arbitrary multidimensional grid that does not necessarily follow any order. Such interpolations are possible but usually costly in terms of computational time. Summing up, the method of endogenous gridpoints is a useful and efficient tool to solve dynamic programming problems. Yet one has to be aware of its limitations.

8.4 Further reading

In this chapter we focused on numerical techniques to solve dynamic programming problems. Readers interested in a more formal mathematical treatment should consult Stokey and Lucas (1989), Miao (2014), or Wälde (2012). While the former two textbooks might be suitable for PhD students and experienced researchers, the latter book offers a quick access to all methods of dynamic optimization for students at different levels. A further discussion of numerical techniques can be found in Miranda and Fackler (2002) or Adda and Cooper (2003); both books provide computer codes in MATLAB and Gauss on their companion websites. Rendahl (2014) discusses the convergence properties of our policy function iteration methods. The method of endogenous gridpoints was introduced by Carroll (2006). Barillas and Fernandez-Villaverde (2007) show how to accommodate multiple controls while White (2015) introduces a computationally efficient interpolation

¹¹ Verify this by taking away the last argument from the call of the subroutine and comparing the execution times with and without the guess.

technique for problems with higher dimensional states. Finally, Aruoba, Fernandez-Villaverde, and Rubio-Ramirez (2006) compare the computational efficiency of other solution methods which are not discussed in this chapter.

8.5 Exercises

- 8.1. Show that the solution to the maximization problem

$$\max_{\{c_v\}_{v=t}^{\infty}} U_t = \sum_{v=t}^{\infty} \beta^{v-t} u(c_t) \quad \text{s.t.} \quad \sum_{v=t}^{\infty} c_v = a_t$$

is equal to the solution of the sequential problem

$$\max_{c_t} u(c_t) + \beta \max_{\{c_v\}_{v=t+1}^{\infty}} U_{t+1}$$

subject to the two constraints

$$a_{t+1} = a_t - c_t \quad \text{and} \quad \sum_{v=t+1}^{\infty} c_v = a_{t+1}.$$

- 8.2. Consider an agent who lives for an infinite number of periods indexed by $t = 0, 1, \dots, \infty$. In each of the periods he receives a constant income stream w . At each point in time the agent has to decide how much to consume c_t and how much to save for the next period a_t . Negative savings, i.e. debt, are also allowed. Having saved an amount of a_t in period t , the agent receives an amount of $(1+r)a_t$ in the next period. The interest rate therefore is r . His utility should be representable by a discounted additively separable utility specification of the form

$$U_0 = \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

The agent's optimization problem consequently reads

$$\max_{\{c_t\}_{t=0}^{\infty}} U_0 = \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \quad \text{s.t.} \quad \sum_{t=0}^{\infty} \frac{c_t}{(1+r)^t} = \sum_{t=0}^{\infty} \frac{w}{(1+r)^t}.$$

Calculate the all-in-one solution of this problem.

- 8.3. Now write the problem in 2 as a dynamic programming problem. Can you derive a closed-form solution for the policy and value function?

Technical note: Assume that the value function takes the form:

$$V(a) = B \cdot \frac{\left[a + \frac{w}{r}\right]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

- 8.4. Implement the solution to the problem in 2 using minimization and interpolation.
 Plot the policy and value function for the parameter combination $\gamma=0.5$, $\beta=0.975$, $r=0.02$, and $w=1$. In addition, simulate a consumption path for 200 periods. Now change r to 0.03. How does the solution change?
- 8.5. Implement the same solution using root-finding and interpolation.
- 8.6. Implement the same solution using the method of endogenous gridpoints.
- 8.7. Assume that the agent can save as much as he wants, but is only allowed to borrow against future income to a certain limit \bar{a} . Consequently as an additional restriction $a_t \geq -\bar{a}$ has to hold for all t . Try to implement this in your solution with minimization and interpolation. Test your program for different values of \bar{a} and r . How does it behave?
- 8.8. Can you think of a way of implementing the constraint \bar{a} in the root-finding and interpolation solution?
- 8.9. Can you think of a way of implementing the constraint \bar{a} in the solution that uses the method of endogenous gridpoints? The original article by Carroll (2006) might help you here.

9 Dynamic macro I: Infinite horizon models

In this chapter we apply the principles of dynamic programming to some standard macroeconomic models. For now we stay in the world of infinite horizon models, which are characterized by the fact that they are populated by one or several households with an infinite planning horizon, similar to the previous chapter. There are several justifications for such an assumption. Beneath simplicity, altruism is probably the most famous argument in favour of infinite horizon models. Assume that in a period t there is one generation that dies with certainty after this period. The utility of this generation from its own consumption is $u(\cdot)$. Yet, each generation is altruistic towards its descendants. Consequently, total utility of the generation is

$$U_t = u(\cdot) + \beta U_{t+1}$$

where $\beta \leq 1$ can be interpreted as the degree of altruism. All generations together then form a dynasty. The utility of the very first generation of this dynasty includes the utility of any descendant generation and can be expressed as

$$U_0 = \sum_{t=0}^{\infty} \beta^t u(\cdot).$$

The *representative agent model* pioneered by Frank Ramsey to study consumption-savings decisions is founded on these considerations. We introduce the most basic version of this model in Section 9.1. In the following sections we extend the standard growth model to account for aggregate uncertainty and variable labour supply, which leads us to the real business-cycle (RBC) framework. Finally, we discuss the so-called *heterogeneous agent model*, in which households face idiosyncratic uncertainty regarding their labour income.

9.1 The basic neoclassical growth model

In the following we explain the structure of the most basic Ramsey economy and show how to represent it as a dynamic programming problem. We then implement this model

on the computer and simulate the transition path of such an economy towards the long-run equilibrium.

9.1.1 THE MODEL ECONOMY

Assume that the economy consists of only two sectors: firms and households. For now we want to ignore both government activity as well as trade with foreign countries. Firms in this economy produce a single output good Y_t which can be used both for consumption of the households C_t as well as investment in the new capital stock I_t . In order to be able to produce the output good, firms have to employ both capital K_t and labour L_t from the household sector. Firms and households interact on three markets: the goods, the capital, and the labour market. In each of the three markets prices p_t , r_t , and w_t will, in equilibrium, be such that demand equals supply. We denote by N_t the number of households that live in the economy at time t . For the moment we assume a constant population, i.e. $N_t = N$.

The firms' problem Firms transform capital K_t and labour L_t into a single output good Y_t according to the production technology

$$Y_t = \Omega_t \cdot F(K_t, L_t).$$

Ω_t denotes the level of technology which we for now normalize to $\Omega_t = 1$. For simplicity we assume that $F(\cdot, \cdot)$ is a linear homogeneous function. Consequently, we can normalize production by the number of households N_t in the economy and write

$$y_t = \frac{Y_t}{N_t} = F\left(\frac{K_t}{N_t}, \frac{L_t}{N_t}\right) = f(k_t, l_t),$$

where y_t , k_t , and l_t denote per capita output, capital stock, and labour supply. In the following we will express every variable in per capita values. When employed in the production process, capital depreciates at constant rate δ . Firms can sell the final output good in the goods market at price p_t , but have to pay the prices r_t and w_t for capital and labour to the household. The price of the consumption good serves as numeraire, i.e. we set $p_t = 1$. By assumption, a large number of identical firms produce under perfect competition in this economy so that firms' profits are equal to zero in equilibrium and we can derive firms' behaviour from looking at one representative firm.

The representative firm maximizes profits Π_t by choosing its optimal inputs of capital and labour given the respective costs, i.e. it solves

$$\max_{k_t, l_t} \Pi_t = f(k_t, l_t) - (r_t + \delta)k_t - w_t l_t.$$

The first-order conditions of this problem lead us to

$$r_t = \frac{\partial f(k_t, l_t)}{\partial k_t} - \delta \quad \text{and} \quad w_t = \frac{\partial f(k_t, l_t)}{\partial l_t}.$$

Note that we could also calculate labour income from the zero-profit condition, i.e.

$$w_t l_t = f(k_t, l_t) - \frac{\partial f(k_t, l_t)}{\partial k_t} k_t.$$

Firms can invest the amount i_t in order to increase tomorrow's capital stock. The law of motion for capital then is given by

$$k_{t+1} = (1 - \delta)k_t + i_t.$$

Investment goods need to be purchased on the goods market.

The households The economy is populated by a large number of atomistic and identical individuals with a planning horizon of infinitely many periods of total mass N_t . For now we assume that each household supplies one unit of labour inelastically to the market $l_t = 1$. Consequently the utility of each individual (or dynasty) is derived only from consumption c_t . Preferences are such that they can be represented by the additively separable utility function

$$\sum_{t=0}^{\infty} \beta^t u(c_t).$$

Thereby $\beta = (1 + \theta)^{-1}$ is a time discount factor which can easily be calculated from the individual subjective time preference rate θ . At each point in time the households receive a labour income of w_t because labour supply equals unity. Each unit of savings yields an interest payment of r_t , so that the household's total disposable income at time t is equal to $(1 + r_t)a_t + w_t$. The household has to decide how much to consume and how much to save from these resources at each point in time. Given an endowment of a_0 of savings in the initial period $t = 0$ the dynamic budget constraint reads

$$(1 + r_t)a_t + w_t = c_t + a_{t+1}.$$

We can then write the household's optimization problem as a dynamic program

$$V(a) = \max_{c, a^+} u(c) + \beta V(a^+) \quad \text{s.t.} \quad (1 + r)a + w = c + a^+.$$

The Lagrangian of this problem reads

$$\mathcal{L} = u(c) + \beta V(a^+) + \lambda[(1 + r)a + w - c - a^+].$$

Taking derivatives of the Lagrangian yields the system

$$\frac{\partial \mathcal{L}}{\partial c} = u'(c) - \lambda = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial a^+} = \beta V'(a^+) - \lambda = 0$$

which combines with the first-order condition

$$u'(c) = \beta V'(a^+).$$

The question that remains is how to determine $V'(a^+)$. For this we have to recall that the first-order condition above, together with the budget constraint, defines the policy function $c(a)$. Using this policy function we can write

$$V(a) = u[c(a)] + \beta V[(1+r)a + w - c(a)],$$

where we have plugged the budget constraint directly into the second term. Taking the derivative with respect to a , we obtain

$$\begin{aligned} V'(a) &= u'(c) \cdot c'(a) + \beta V'(a^+) \cdot [(1+r) - c'(a)] \\ &= \underbrace{[u'(c) - \beta V'(a^+)]}_{=0} \cdot c'(a) + (1+r) \cdot \underbrace{\beta V'(a^+)}_{=\lambda} = (1+r) \cdot \lambda. \end{aligned}$$

From the first-order conditions of the optimization problem we know that $u'(c) = \beta V'(a^+)$ and that $\beta V'(a^+) = \lambda$ which directly yields the last equality. Note that there is a much faster way that leads us to exactly the same result. The so-called *envelope theorem*¹ tells us that the derivative of $V(a)$ with respect to a is the same as the derivative of the Lagrangian \mathcal{L} with respect to a . Using this theorem with the Lagrangian being defined as above we see that

$$V'(a) = \frac{\partial \mathcal{L}}{\partial a} = (1+r)\lambda.$$

Finally, we know from the first-order conditions that λ has to be equal to $u'(c)$ in the optimum, which finally leads us to

$$V'(a) = (1+r)u'(c) \quad \text{or} \quad V'(a^+) = (1+r^+)u'(c^+).$$

The first-order condition then becomes

$$u'(c) = \beta V'(a^+) = \beta(1+r^+)u'(c^+).$$

¹ Which can be applied whenever the value function as well as the policy functions and the Lagrangian multipliers are continuously differentiable.

Using time indexes we can also write

$$u'(c_t) = \beta(1 + r_{t+1})u'(c_{t+1}),$$

The solution to the household problem is characterized by the above first-order condition together with the dynamic budget constraint.

Market equilibrium Factor markets clear when labour supply equals labour demand and the total amount of capital employed in the production process is held by the households. The latter implies that the total amount of aggregate savings needs to equal the total amount of newly employed capital at each date, i.e. $a_{t+1} = k_{t+1}$. The goods market clears as soon as demand equals supply, i.e.

$$y_t = c_t + i_t.$$

By Walras' law whenever two of the markets are in equilibrium, the last market will also be in equilibrium.

A centralized representation of the model economy The above representation of our economy is called decentralized. In a decentralized economy households and firms are price takers, i.e. they make their optimal choices under the assumption that they do not have an impact on prices r_t and w_t . The overall economy is then characterized by the following set of equations

$$\begin{aligned} u'(c_t) &= \beta(1 + r_{t+1})u'(c_{t+1}), \quad (1 + r_t)a_t + w_t = c_t + a_{t+1} \\ y_t &= f(k_t, l_t), \quad r_t = \frac{\partial f(k_t, l_t)}{\partial k_t} - \delta, \quad w_t = y_t - \frac{\partial f(k_t, l_t)}{\partial k_t}k_t \\ y_t &= c_t + i_t, \quad a_{t+1} = k_{t+1}. \end{aligned}$$

It is however easy to show that the decentralized economy is equivalent to a centralized economy, in which a social planner decides how much capital and labour is used in the production process and how output is split between consumption and investment. This is easy to see when we combine a couple of the equations above. First of all we can set $a_t = k_t$ and $a_{t+1} = k_{t+1}$. The budget constraint of the household then reads

$$(1 + r_t)k_t + w_t = c_t + k_{t+1}.$$

Using the factor price equations we can then write

$$\left(1 + \frac{\partial f(k_t, l_t)}{\partial k_t} - \delta\right)k_t + y_t - \frac{\partial f(k_t, l_t)}{\partial k_t}k_t = c_t + k_{t+1}$$

which immediately reduces to

$$(1 - \delta)k_t + y_t = c_t + k_{t+1}.$$

Finally we can use $y_t = f(k_t, l_t) = f(k_t, 1) =: f(k_t)$ to write

$$(1 - \delta)k_t + f(k_t) = c_t + k_{t+1}.$$

The first-order condition of the household on the other hand becomes

$$u'(c_t) = \beta \left(1 + \frac{\partial f(k_{t+1}, l_{t+1})}{\partial k_{t+1}} - \delta \right) u'(c_{t+1}) = \beta [1 + f'(k_{t+1}) - \delta] u'(c_{t+1}).$$

Note that the equation system that describes the dynamics of the economy then reduces to

$$u'(c_t) = \beta [1 + f'(k_{t+1}) - \delta] u'(c_{t+1}) \quad \text{and} \quad (1 - \delta)k_t + f(k_t) = c_t + k_{t+1}. \quad (9.1)$$

This equation system, however, is also the solution to the problem of the social planner in the centralized economy, i.e.

$$V(k) = \max_c u(c) + \beta V(k^+) \quad \text{s.t.} \quad (1 - \delta)k + f(k) = c + k^+. \quad (9.2)$$

Summing up, in order to compute a solution of the decentralized Ramsey economy described above, it is enough to solve the (much simpler) dynamic programming problem (9.2). Knowing the time paths for consumption $\{c_t\}_{t=0}^\infty$ and capital $\{k_t\}_{t=0}^\infty$ we could then still calculate the evolution of factor prices, production, and investment over time.

The stationary equilibrium Unfortunately no closed-form solution to the policy or value function in the Ramsey model exists. However, we can still derive a closed-form solution for a stationary equilibrium or a so-called *steady state*. A stationary equilibrium is characterized by the fact that all per capita variables are constant over time, i.e. $c_t = c_{t+1} = c_{t+2} = \dots = \bar{c}$ and $k_t = k_{t+1} = k_{t+2} = \dots = \bar{k}$. From the equations in (9.1) we find that the stationary equilibrium has to satisfy

$$f'(\bar{k}) = \theta + \delta \quad \text{which implies} \quad r = \theta \quad \text{and} \quad \bar{c} = f(\bar{k}) - \delta \bar{k},$$

where we used $\beta = (1 + \theta)^{-1}$. Assuming the usual Cobb-Douglas production function $f(k) = k^\alpha$, the stationary equilibrium values of capital and consumption are

$$\bar{k} = \left[\frac{\alpha}{\theta + \delta} \right]^{\frac{1}{1-\alpha}} \quad \text{and} \quad \bar{c} = (\bar{k})^\alpha - \delta \bar{k} = \left[\frac{\alpha}{\theta + \delta} \right]^{\frac{\alpha}{1-\alpha}} - \delta \left[\frac{\alpha}{\theta + \delta} \right]^{\frac{1}{1-\alpha}}. \quad (9.3)$$

Interestingly, the functional form of the utility function does not affect the stationary equilibrium quantities at all.

9.1.2 NUMERICAL IMPLEMENTATION

Calculating only stationary equilibrium values is in general not very satisfactory. Economists are usually much more interested in the dynamics of an economy, i.e. how capital, consumption, etc. evolve over time when they are not in steady state. These dynamics are also called *transitional dynamics*.

In order to calculate the transitional dynamics of the Ramsey economy, we have to solve the dynamic program in (9.2) numerically. Before we do so, we need to choose some functional forms for the production and the utility function. As already mentioned above, the production function should be represented by a Cobb-Douglas technology $f(k, l) = k^\alpha l^{1-\alpha}$, so that we obtain $f(k) = k^\alpha$ for $l = 1$ with α denoting the capital share in production. As in the previous chapter the instantaneous utility function takes the form $u(c) = \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}}$, where γ denotes the intertemporal elasticity of substitution.

The dynamic programming problem we have to solve then reads

$$V(k) = \max_c \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}} + \beta V(k^+) \quad \text{s.t.} \quad (1-\delta)k + k^\alpha = c + k^+.$$

and the respective first-order condition is

$$c = [\beta(1 + \alpha(k^+)^{\alpha-1} - \delta)]^{-\gamma} c^+ \quad \text{with} \quad (1-\delta)k + k^\alpha = c + k^+. \quad (9.4)$$

There are again ample ways to solve the dynamic programming problem, i.e. to derive a policy function $c(k)$. Since the only constraint we have is the budget constraint and the problem is differentiable for every $c > 0$, our method of choice is policy function iteration. However, it is difficult to apply the method of endogenous gridpoints here. It would be straightforward to calculate c for any future level of resources $k^+ = \hat{k}_v$. Yet in order to determine the endogenous gridpoint $k(\hat{k}_v)$ we would have to solve the equation

$$(1-\delta) \cdot k(\hat{k}_v) + [k(\hat{k}_v)]^\alpha = c(\hat{k}_v) + \hat{k}_v$$

which cannot be solved analytically for $\alpha \neq 1$. Therefore we choose the root-finding and interpolation algorithm for solving our dynamic programming problem.

Program 9.1 shows how to do this. As in the cake-eating problem, we first discretize the state space. In general, the Ramsey economy exists on the state space $k \in (0, \infty)$. Yet in order to determine the dynamics of the Ramsey problem given an initial capital stock k_0 ,

Program 9.1 The Ramsey economy with root-finding and interpolation

```

program Ramsey
    [.....]
    ! initialize grid and policy function
    call grid_Cons_Equi(k, k_l, k_u)
    c(:) = k(:)**alpha - delta*k(:)

    ! iterate until policy function converges
    do iter = 1, itermax

        ! interpolate coefficients
        call spline_interp(c, coeff_c)

        ! calculate decisions for every gridpoint
        do ik = 0, NK

            ! set starting value and communicate resource level
            x_in = c(ik)
            k_com = k(ik)

            ! find the optimal consumption level
            call fzero(x_in, foc, check)
            if(check)write(*,*)'ERROR IN ROOTFINDING PROCESS'

            ! get optimal consumption function
            c_new(ik) = x_in

        enddo
        [.....]
    enddo
    [.....]
end program

```

it is enough to look at a subinterval $[k_l, k_u]$. Note that this sub interval needs to contain both the initial capital stock k_0 as well as the long-run stationary capital level \bar{k} from (9.3). If this is the case, we will be able to simulate the dynamics of the Ramsey economy. To perform the iteration procedure, we discretize the state space $[k_l, k_u]$ using equidistant gridpoints as implemented in the subroutine `grid_Cons_Equi`, see again Section 2.5. We then make an initial guess for the consumption values $\{c(\hat{k}_v)\}_{v=0}^n$ at the gridpoints. Setting

$$c(\hat{k}_v) = \left(\hat{k}_v\right)^\alpha - \delta\hat{k}_v,$$

would mean that the capital stock k^+ is identical to $k = \hat{k}_v$ at each gridpoint. The iteration procedure first approximates the policy function with a spline S , i.e. we construct a spline function $S(k)$ such that

$$S(\hat{k}_v) = c(\hat{k}_v)$$

holds for all $v = 0, \dots, n$. Then it solves the first-order condition (9.4) for every gridpoint \hat{k}_v . The root-finding algorithm takes the consumption level as input. The best guess for consumption at the gridpoint \hat{k}_v obviously is the consumption level that we

found in the last step of the policy function iteration. The variable `k_com` is used as communication variable and contains the current level of capital.² Finally, `fzero` searches for the root of the first-order condition and stores the result in the scalar `x_in`. The remainder of the policy function iteration process is the same as in Chapter 8.

Beneath all the parameters and variables, the module `globals` contains the function `foc` that is shown in Module 9.1m. The function takes as input variable a consumption level and uses the current level of capital `k_com`. From this it computes tomorrow's level of capital from the budget constraint of the economy. Knowing tomorrow's capital k^+ , we can compute the predicted consumption level `c_plus` of tomorrow using our interpolated policy function. Finally, the first-order condition which has to be solved by the root-finding routine is

$$c(\hat{k}_v) = [\beta(1 + \alpha(k^+)^{\alpha-1} - \delta)]^{-\gamma} S(k^+).$$

Having found the policy function of the Ramsey economy on the gridpoints \hat{k}_v , the policy function iteration process will end and the program calls the subroutine `output`. This subroutine simulates a consumption path $\{c_t\}_{t=0}^T$ for the economy, given an initial level of capital k_0 . It then plots the dynamics of the capital stock as well as the goods market. In each plot we also show the respective steady-state levels that we calculated analytically. Finally, the policy function is plotted on the interval $[k_l, k_u]$. Figure 9.1 shows the dynamics of capital and the goods market of the Ramsey economy. We chose the parameter values $\gamma = 0.5$, $\alpha = 0.40$, $\beta = 0.99$, $\delta = 0.019$, and $k_0 = 10$. With these

Module 9.1m The first-order condition of the Ramsey economy

```
function foc(x_in)
use toolbox

implicit none
real*8, intent(in) :: x_in
real*8 :: foc, kplus, cplus

! future capital
kplus = (1d0-delta)*k_com + k_com**alpha - x_in

! calculate future consumption
cplus = spline_eval(kplus, coeff_c, k_l, k_u)

! get first-order condition
foc = x_in - (beta*(1d0+alpha*kplus**(alpha-1d0) &
-delta))**(-gamma)*cplus

end function
```

² We use the level of capital and not the index of the gridpoint here on purpose. We will see later exactly why we do this.

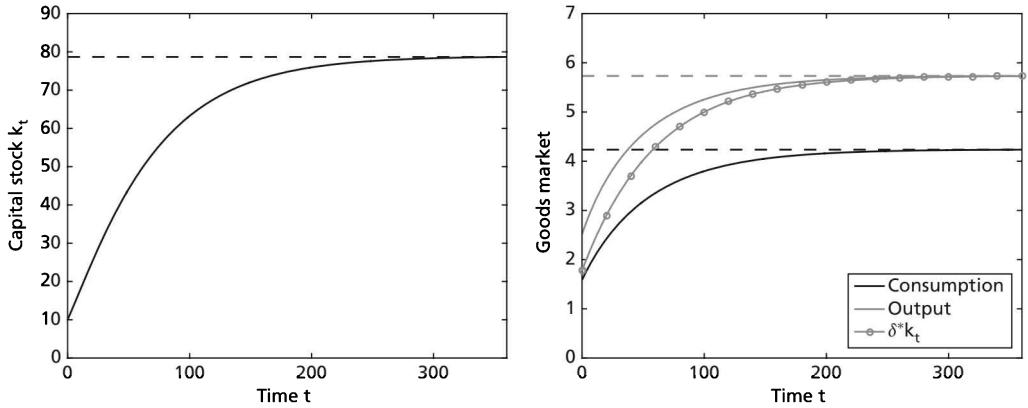


Figure 9.1 Dynamics of the Ramsey economy

parameters we assume that each period t covers one quarter of a year so that the annual depreciation rate amounts to 10 per cent and the annual time preference rate is roughly 5 per cent. Knowing that the stationary capital level lies somewhere around a value of 80, we let the interval endpoints be $k_l = 5$ and $k_u = 100$. In consequence the interval contains the initial and the stationary capital level. We furthermore chose $n = 100$ points to discretize the state space and simulate the economy for $T = 360$ quarters, i.e. 90 years. The dynamics of the capital stock obviously starts at $k_0 = 10$. Since the steady-state capital stock is $\bar{k} = 78.68$, the capital stock will slowly increase over time until it converges at to this steady-state level. We call this convergence process a *transition path* or *transitional dynamics*. Note that it takes about 300 quarters for the capital stock to converge. The goods market shows us how the capital stock increases. Remember that the goods market equilibrium was given by the fact that aggregate output equals consumption plus gross investment, i.e. $y_t = c_t + i_t$. In Figure 9.1 gross investment is consequently the difference between the grey and the black line. When we look at the accumulation equation for capital

$$k_{t+1} = (1 - \delta)k_t + i_t \quad \Rightarrow \quad i_t = \delta k_t + k_{t+1} - k_t$$

we can see that we can split up gross investment i_t into two parts: that part of the investment that replaces the capital stock that depreciated throughout the production process δk_t and that part that actually increases the capital stock k_{t+1} beyond the capital level k_t , $i_t^n = k_{t+1} - k_t$. The latter is also called *net investment*. In the right part of Figure 9.1 we have decomposed the investment part of the goods market into these two components. The dotted grey line thereby denotes *replacement investment*, i.e. δk_t . The difference between the grey and the dotted grey line then is net investment. Net investment is obviously positive in the initial periods t , since the capital stock has to increase. As soon as the capital stock converges to its steady-state level, however, net investment has to decrease to zero. All that is left then is replacement investment δk_t .

Finally, since the capital stock rises with time, the economy's output and therefore consumption possibilities increase.

To conclude, we want to test the accuracy of our numerical solution method. In Chapter 8 this was quite easy, since we knew the analytical solution of the policy function. Hence, we were able to calculate the consumption function errors for a large amount of potential asset values. In the absence of an analytical solution, we cannot do this in the present model anymore. Yet there is an analogue method in this case, the so-called Euler equation error. The Euler equation of our Ramsey growth model is shown in equation (9.4). This Euler equation, however, only holds with equality in the exact (that would be analytical) solution of the problem. In a numerical solution the Euler equation will only be approximately right. The Euler equation error then tests with how much accuracy the Euler equation holds on the whole interval $[k_l, k_u]$. It therefore uses the approximated policy function $S(k)$. We already know that the Euler equation will hold with quite high accuracy in the gridpoints \hat{k}_v as we used it to determine the policy function. The more interesting question then is how accurately the Euler equation is satisfied in the areas between the gridpoints. At an arbitrary point k we therefore calculate the difference

$$EER(k) = S(k) - [\beta(1 + \alpha(k^+)^{\alpha-1} - \delta)]^{-\gamma} S(k^+) \\ \text{with } k^+ = (1 - \delta)k + k^\alpha - S(k)$$

the so-called *Euler equation residual*. Similar to the consumption function error we calculated in Chapter 8 we finally determine the Euler equation error as

$$\max_k \left| \frac{EER(k)}{S(k)} \right|.$$

The Euler equation error has a direct economic interpretation. It tells us that if the agent made a consumption expenditure decision of, for example, US\$1,000, he would at most make an error of $1000 \cdot \max_k \left| \frac{EER(k)}{S(k)} \right|$ US\$. Program 9.1.a shows us how the Euler equation error is determined in the Ramsey model. We therefore check the first-order condition at a large number of points on the state space—in our case `nerr = 10000`—and

Program 9.1.a Calculating Euler equation errors

```
err = 0d0
do ik = 0, n_err
    k_com = k_l + (k_u-k_l)*dble(ik)/dble(n_err)
    c_err = spline_eval(k_com, coeff_c, k_l, k_u)
    err_temp = abs(foc(c_err)/c_err)
    if(err_temp > err)err = err_temp
enddo
write(*,'(a, es15.7)')'Euler equation error:',err
```

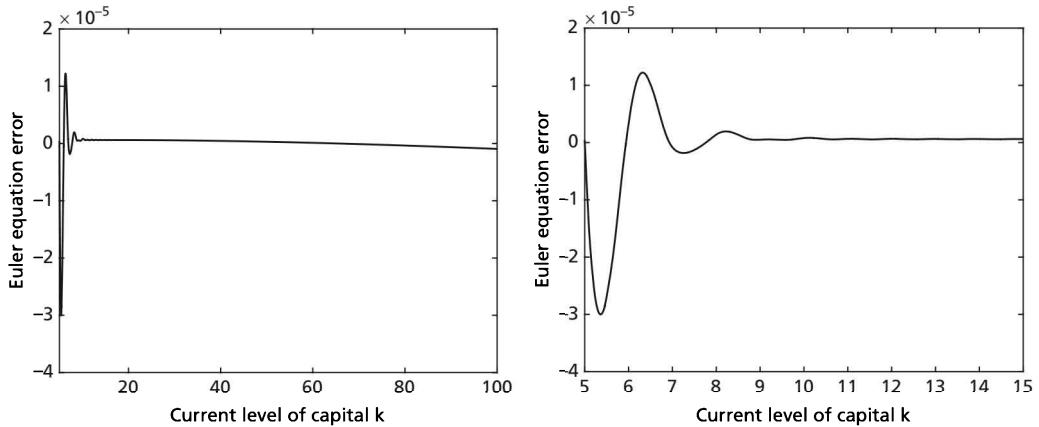


Figure 9.2 Euler equation residual of the Ramsey economy

calculate the maximum absolute difference.³ The Euler equation error is then printed on the console. Figure 9.2 shows the evolution of the relative Euler equation residuals $\frac{EER(k)}{S(k)}$ over the state space in the Ramsey model. Since the policy function is curved towards the lower end of the state space k_l , the Euler equation errors are greatest towards this point.

9.1.3 A MODEL WITH A PUBLIC SECTOR

Up to this point we have learned how to calculate the dynamics of a Ramsey economy given an initial level of the capital stock in the simplest one-sector setup. In the following we want to add some additional flavour to the model by introducing government activity and studying how the macroeconomy and households' level of welfare react to changes in the government policy.

For simplicity we assume that the government finances a *fixed* amount of public expenditure per capita, which is defined as a fraction $0 \leq g_y < 1$ of steady-state GDP in an economy without government activity, i.e.

$$\bar{g} = g_y \cdot \left[\frac{\alpha}{\theta + \delta} \right]^{\frac{\alpha}{1-\alpha}}. \quad (9.5)$$

The instruments that are available are a tax on wage income τ^w , a tax on income from savings τ^r and public debt b . The government's budget constraint reads

³ Note that here it becomes crucial that the function `foc` receives the actual capital stock and not the index of a gridpoint as communication variable.

$$\bar{g} + (1 + r_t)b_t = \tau_t^w \cdot w_t + \tau^r \cdot r_t a_t + b_{t+1}.$$

At each date in time, the government needs to finance its consumption expenditure as well as repay its debt, including interest. The total expenditure needs to equal revenue from the taxation of income as well as resources generated from issuing new debt. Note that we made two assumptions when writing down this equation. First, we assume that the government can only issue one period bonds. Consequently we do not have to consider issues related to the term structure of government debt. Second, we let the tax on capital income be time-invariant, which means that the labour tax rate has to vary over time to balance the government's budget. Let us now further assume that debt amounts to a fixed fraction b_y of *current actual* GDP, i.e. $b_t = b_y \cdot y_t$. Consequently in the long-run equilibrium where GDP is constant over time,⁴ the government budget reads

$$\bar{g} + rb = \tau^w \cdot w + \tau^r \cdot ra.$$

Note that when setting $\bar{g} = 0$, we can also analyse pure redistribution policies between wage and capital income without reducing the average consumption expenditure of households.

Including a public sector in the economy will not alter the optimization problem of the firm as taxes are levied solely on households. Therefore, in a decentralized version of the economy, we still have

$$r_t = \alpha k_t^{\alpha-1} - \delta \quad \text{and} \quad w_t = (1 - \alpha)k_t^\alpha$$

with the same specification of the production technology as above.

Yet the household optimization problem will change. In fact we can write the household's dynamic budget constraint as

$$a_{t+1} + c_t = [1 + (1 - \tau^r)r_t]a_t + (1 - \tau_t^w)w_t.$$

The household's optimization problem in dynamic programming form then reads

$$V(a) = \max_{c, a^+} u(c) + \beta V(a^+) \quad \text{s.t.} \quad a^+ + c = (1 - \tau^w)w + [1 + (1 - \tau^r)r]a.$$

From the Lagrangian we can derive the first-order condition

$$c = [\beta (1 + (1 - \tau^r)r^+)]^{-\gamma} c^+.$$

⁴ Remember that we assumed the population to be constant.

To complete the model, we have to think about the market-clearing conditions. The labour market is again in equilibrium when the demand of the firms and supply of the households is identical. However, in the capital market there are now two competitors for the savings of the households: firms that want to employ savings to form capital and the government which issues debt to finance expenditure. Therefore the capital market-clearing condition reads

$$a_t = k_t + b_t.$$

In the goods market the government uses \bar{g} for its own consumption, so that the goods market is in equilibrium whenever

$$y_t = c_t + i_t + \bar{g}$$

holds.

Using all the above equations, we can show that the solution to the model with a government sector is comprehensively defined by the first-order condition

$$c = \{\beta [1 + (1 - \tau^r) (\alpha(k^+)^{\alpha-1} - \delta)]\}^{-\gamma} c^+$$

together with the law of motion for capital

$$k^+ = k^\alpha + (1 - \delta)k - c - \bar{g}.$$

Note that at this point it pays to assume that the capital-income tax rate is time-invariant otherwise we would have to calculate a budget-balancing $\tau^{r,+}$ for each potential future level of capital k^+ . All other variables of the model, i.e. prices w and r , the debt level b , as well as the labour-income tax rates τ^w are, in the end, only functions of the current and future levels of capital k and k^+ . In fact we can calculate the budget-balancing labour-income tax rate from

$$\tau^w = \frac{\bar{g} + (1 + r)b - b^+ - \tau^r \cdot r(k + b)}{w}.$$

In order to solve the model, we again use policy function iteration and therefore determine a spline function that satisfies

$$S(\hat{k}_v) = c(\hat{k}_v) \quad \forall v = 0, \dots, n.$$

The set of first-order condition then reads

$$c(\hat{k}_v) = \{\beta [1 + (1 - \tau^r) (\alpha(k^+)^{\alpha-1} - \delta)]\}^{-\gamma} S(k^+).$$

Module 9.2m Policy functions of the Ramsey economy with government

```

function foc(x_in)
    [.....]
    ! future capital
    kplus = (1d0-delta)*k_com + k_com**alpha - x_in - gbar

    ! calculate future consumption
    cplus = spline_eval(kplus, coeff_c, k_l, k_u)

    ! get first-order condition
    foc = x_in - (beta*(1d0+(1d0-tau_r) &
        *(alpha*kplus**((alpha-1d0)-delta))))**(-gamma)*cplus

end function

```

with

$$k^+ = (1 - \delta)\hat{k}_v + (\hat{k}_v)^\alpha - c - \bar{g}.$$

Including a public sector hardly changes the main program. In fact we only have to set the level of government consumption according to the formula given in (9.5), which we do not show here. In Module 9.2m we can see how the first-order condition of the previous program has to be adjusted. The resource constraint which is used to derive future capital now also includes public consumption \bar{g} . In addition we have to account for the effects of the capital-income tax on the intertemporal price.

Finally we want to briefly discuss how we calculate a time path of the economy. This is done in the subroutine `output` of the main program, parts of which are shown in Program 9.2. We always assume that the time path of the economy starts with the steady-state capital level without government activity, i.e.

$$k_0 = \left[\frac{\alpha}{\theta + \delta} \right]^{\frac{1}{1-\alpha}},$$

and a public debt level of $b_0 = 0$. Knowing the initial capital we can calculate consumption using our interpolated policy function and determine all other variables of the economy, like prices or GDP. We then simulate the economy forward as usual. Note that we always calculate the budget-clearing labour-income tax rate for the previous year, as we need to know the debt level of two successive periods ($t-1$ and t). Finally, we compute households' utility level. There are actually two different concepts of measuring welfare that are widely used in this kind of model. The first measure we compute only takes into account consumption in the long-run equilibrium.⁵ This welfare measure can be easily computed by recognizing that for a constant level of consumption c we have

⁵ In our case long-run equilibrium consumption is the same as consumption in the last simulated period as we choose a long time horizon for simulation, i.e. 3,600 quarters or 900 years.

Program 9.2 Simulating a time path of the economy

```

k_t(0) = (alpha/(theta+delta))** (1d0/(1d0-alpha))
c_t(0) = spline_eval(k_t(0), coeff_c, k_l, k_u)
y_t(0) = k_t(0)**alpha
i_t(0) = y_t(0)-c_t(0)-gbar
w_t(0) = (1d0-alpha)*k_t(0)**alpha
r_t(0) = alpha*k_t(0)**(alpha-1d0)-delta
b_t(0) = 0d0
do it = 1, TT
    k_t(it) = (1d0-delta)*k_t(it-1) + i_t(it-1)
    c_t(it) = spline_eval(k_t(it), coeff_c, k_l, k_u)
    y_t(it) = k_t(it)**alpha
    i_t(it) = y_t(it)-c_t(it)-gbar
    w_t(it) = (1d0-alpha)*k_t(it)**alpha
    r_t(it) = alpha*k_t(it)**(alpha-1d0)-delta
    b_t(it) = by*k_t(it)**alpha
    tauw_t(it-1) = (gbar+(1d0+r_t(it-1))*b_t(it-1)-b_t(it) &
                     -tau_r*r_t(it-1)*(k_t(it-1)+b_t(it-1)))/w_t(it-1)
enddo
tauw_t(TT) = (gbar+r_t(TT)*b_t(TT)-tau_r*r_t(TT) &
               *(k_t(TT)+b_t(TT)))/w_t(it-1)

! calculate the steady state utility level
UU = c_t(TT)**egam/egam/(1d0-beta)

! calculate the utility level
UU_0 = 0d0
do it = 0, TT-1
    UU_0 = UU_0 + beta** (it)*c_t(it)**egam/egam
enddo
UU_0 = UU_0 + beta**TT*UU

```

$$\sum_{t=0}^{\infty} \beta^t \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} = \frac{1}{1 - \beta} \cdot \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

The second welfare measure takes into account the fact that the economy starts at a certain point without any government activity and as soon as the government starts changing its policy, the economy needs a while to converge to the long-run equilibrium. In order to account for the consumption effects along this transition, we calculate the second welfare measure using the full simulated consumption path, i.e.

$$U_0 = \sum_{t=0}^{\infty} \beta^t \cdot \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Table 9.1 displays output and welfare effects resulting from different combinations of government policy parameters. We report long-run equilibrium values of capital, consumption, and government consumption as a percentage of GDP.⁶ In addition,

⁶ We adjusted the capital stock so that it reflects annualized values. Note that due to labour being a fixed factor here, the annual capital to GDP ratio is given by $\frac{k}{4y} = \frac{k}{4k^\alpha} = \frac{1}{4} \cdot k^{1-\alpha}$. Therefore a higher capital-output ratio in our economy necessarily accompanies a larger capital stock.

the table includes the actual level of long-run consumption as well as the two welfare measures discussed above.

The upper panel focuses on pure redistribution policies. In the first line, we set all government parameters equal to zero. Therefore we replicate the economy without government activity. Given our choice for the initial capital stock there will be no transitional dynamics. Instead, capital stock and consumption will remain constant at their long-run equilibrium values for all simulated periods. As a result the long-run welfare measure U as well as the transitional welfare measure U_0 have to be identical. Next we subsidize capital income by a subsidy rate of 20 per cent, financed by means of a labour-income tax. The labour-income tax necessary to finance this subsidy amounts to 4.09 per cent. The subsidy increases the return for private savings so that households start saving more. This leads to additional capital accumulation in the long run. The capital-to-GDP ratio increases from 343 per cent to 364 per cent. While the consumption share of GDP declines, the absolute level of consumption increases as the economy becomes more productive. A higher level of consumption obviously leads to a higher level of long-run welfare. Yet the transitional welfare measure U_0 indicates an overall welfare loss. The reason for this can be found in the transitional dynamics, see the right part of Figure 9.3. While the rise of the capital stock leads to higher consumption in the long run, it comes at the cost of low consumption levels in the initial periods of the transition. In fact, the consumption level drops significantly below the level in the equilibrium without government activity (black line). The foregone short-run consumption will be used for additional investment. In terms of welfare, the short-run loss in consumption outweighs the future benefits of a higher capital stock so that overall welfare declines. The long-run welfare benefits of a interest-rate subsidy are therefore solely due to intertemporal redistribution. When the government levies a 20 per cent tax on capital income, the long-run effects are opposite to those of a subsidy, see the next line of Table 9.1. Yet the transitional welfare effects are again negative. This time consumption increases in the short run and declines in the long run. However, the long-run consumption level turns out to be so low that the short-term consumption increase can not make up for these losses.

The lower panel of Table 9.1 considers a government with a public consumption level of $g_y = 0.15$. The benchmark case shows the situation when the public good

Table 9.1 Steady-state analysis of tax structure

| g_y | τ^r | b_y | τ^w | k/y | c/y | \bar{g}/y | c | U | U_0 |
|-------|----------|-------|----------|--------|-------|-------------|------|---------|---------|
| 0.0 | 0.0 | 0.0 | 0.00 | 343.63 | 73.88 | 0.00 | 4.24 | -23.587 | -23.587 |
| | -20.0 | | 4.09 | 364.73 | 72.28 | 0.00 | 4.32 | -23.172 | -23.613 |
| | 20.0 | | -5.32 | 316.19 | 75.97 | 0.00 | 4.12 | -24.249 | -23.629 |
| 15.0 | 0.0 | | 25.00 | 343.63 | 58.89 | 15.00 | 3.38 | -29.597 | -29.596 |
| | 20.0 | | 21.10 | 316.17 | 60.11 | 15.86 | 3.26 | -30.645 | -29.657 |
| | 40.0 | | 16.19 | 279.04 | 61.56 | 17.23 | 3.07 | -32.525 | -29.922 |
| | 50.0 | | 17.04 | 279.04 | 61.56 | 17.23 | 3.07 | -32.525 | -29.922 |

$\gamma = 0.5$, $\beta = 0.99$, $\alpha = 0.40$, $\delta = 0.019$. Except for utility, all variables in per cent.

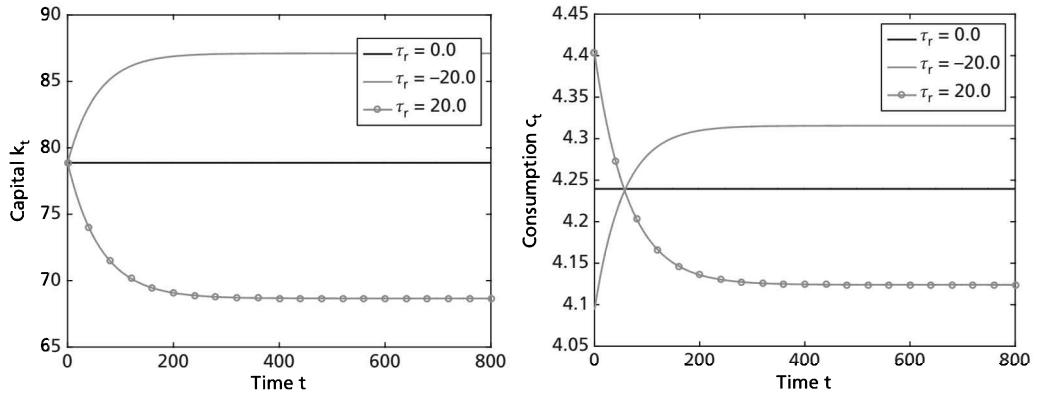


Figure 9.3 Capital-income taxation in the Ramsey economy

is purely financed by labour-income taxes (which in our special case are lump-sum taxes). In this case the labour tax rate is 25 per cent. Since capital accumulation is not distorted, the capital—output ratio and therefore the level of capital is the same as in the first line of Table 9.1. Lower labour income only reduces consumption by exactly the amount of government consumption, i.e. 15 percentage points of GDP. Therefore the absolute consumption level falls to 3.38, reducing aggregate utility to -29.596. When we successively increase the tax rate on capital income to 20 per cent and 40 per cent, the labour-income tax rate falls to about 21 per cent and 16 per cent, respectively. The distortion of capital accumulation reduces the capital stock, output, and consumption. Consequently the welfare of households is lower both according to the long-run welfare measures and according to the transitional one. Finally, as shown in the last line of Table 9.1, debt policy has no impact on capital accumulation, output, and consumption in this economy as long as the endogenous tax creates no distortion. Financing public consumption by public debt allows us to reduce the wage tax in the first year of the transition,⁷ but will lead to higher taxes from period 1 onwards as the interest payments on debt need to be financed. Taking into account these differences in tax rates over time, households will save more from period 0 to period 1, but they will not alter their level of consumption. The additional savings of the households will be completely absorbed by the government's demand resulting from debt issuance. The households then use the interest payments on the additional savings to finance the higher tax burden from period 1 onwards. This is the famous *Ricardian equivalence theorem* which states that government debt is neutral in terms of economic performance as long as the tax system creates no distortions.

Summing up, we have learned so far how to calculate the transitional dynamics of a classical Ramsey model. We have seen that when we start at an arbitrary level of

⁷ Note that we started with a debt level of $b_0 = 0$ and the debt level will jump to $b_y \cdot y_t$ for $t > 0$.

initial capital, the economy will converge successively to a steady state in which gross investment equals replacement investment and net investment is zero. However, as there is no source of uncertainty in our basic Ramsey model, convergence takes place in a very ordered and monotonic way. In Section 9.2, we want to take a look at how the dynamics of the Ramsey economy change when some source of uncertainty is introduced in the production process. We also found that the standard Ramsey model indicates that capital-income taxation will have a strong negative effect on households' welfare, while issuing public debt has no welfare impact. We will investigate in the last section of this chapter, Section 9.4, whether these results still hold when we extend the representative agent model by heterogeneous agents.

9.2 The stochastic growth model

Building on the discussion of the deterministic growth model we now introduce randomness into the economic environment. The underlying assumption is that economic fluctuations originate from shocks to the production process. A 'positive shock' leads to a higher total factor productivity in the present and at the same time might change expectations about future productivity. The planner can respond to such fluctuations by adjusting the accumulation of capital. To introduce all these features into our economic environment we proceed in several steps: We first explain how to model aggregate productivity shocks and how to formulate a corresponding dynamic programming problem. We then show how to solve this dynamic program using a discrete approximation of the shock process. Finally, we discuss how to simulate a time path of the economy and how to optimize the computational process in terms of its speed.

9.2.1 MODELLING AGGREGATE UNCERTAINTY

We start this section by making only one tiny change to the deterministic Ramsey economy we analysed above. We assume that aggregate (or total factor) productivity Ω_t is no longer normalized to 1, but follows a stochastic process over time. Specifically, we let $\Omega_t = \exp(\eta_t)$, where η_t follow a first-order autoregressive process. This means that we can write the law of motion for η_t over time as

$$\eta_t = \rho\eta_{t-1} + \epsilon_t \quad \text{with} \quad \epsilon_t \sim N(0, \sigma_\epsilon^2) \quad \text{and} \quad 0 \leq |\rho| < 1.$$

ϵ_t is normally distributed with mean 0 and variance σ_ϵ^2 . It is called the *innovation* of the process. In each period t a new random draw from this normal distribution is added to the process. ρ on the other hand is called the *auto-correlation* or *persistence* of the process.

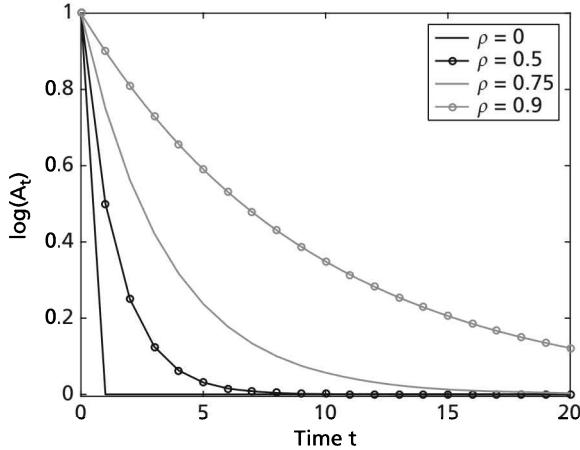


Figure 9.4 Impulse response function of the AR(1) process for different values of ρ

The higher ρ is the longer an innovation will persist in the process, i.e. the innovation ϵ_t has a higher impact on all future values $\eta_{t+1}, \eta_{t+2}, \dots$. Figure 9.4 shows the so-called *impulse response function* of an autoregressive process for different values of ρ . The impulse response function is calculated by setting $\epsilon_0 = 1$ and $\epsilon_t = 0$ for all $t > 0$. As we can see in the figure, the higher the auto-correlation coefficient of the process is, the longer the time required until the process converges back to zero. Since $|\rho| < 1$, however, it will eventually do so. The process is therefore also called *mean reverting*. Note that if $\rho < 0$, then the process does not converge monotonically back to zero but oscillates around the x-axis. Note further that as long as $|\rho| < 1$, the stochastic process η_t has an *unconditional stationary distribution*, which is a normal distribution with parameters

$$\mu_u = 0 \quad \text{and} \quad \sigma_u^2 = \frac{\sigma_\epsilon^2}{1 - \rho^2}.$$

The distribution is called unconditional, since it describes the overall distribution of η_t . On the other hand a *conditional distribution* $\pi[\eta_{t+1} | \eta_t]$ exists which describes the distribution of the next period's realization η_{t+1} , given that the current realization of the process is η_t . π is also a normal distribution with parameters

$$\mu_c = \rho\eta_t \quad \text{and} \quad \sigma_c^2 = \sigma_\epsilon^2.$$

If ρ were equal to 1 we would call η_t a *random walk*. For such a process, an innovation persists forever, i.e. the process does not revert to its mean. In consequence, the process is no longer stationary and the variance would be ∞ . Finally, by setting $\Omega_t = \exp(\eta_t)$ we ensure that technology does not turn negative. Furthermore, the unconditional distribution of technology will then be log-normal with

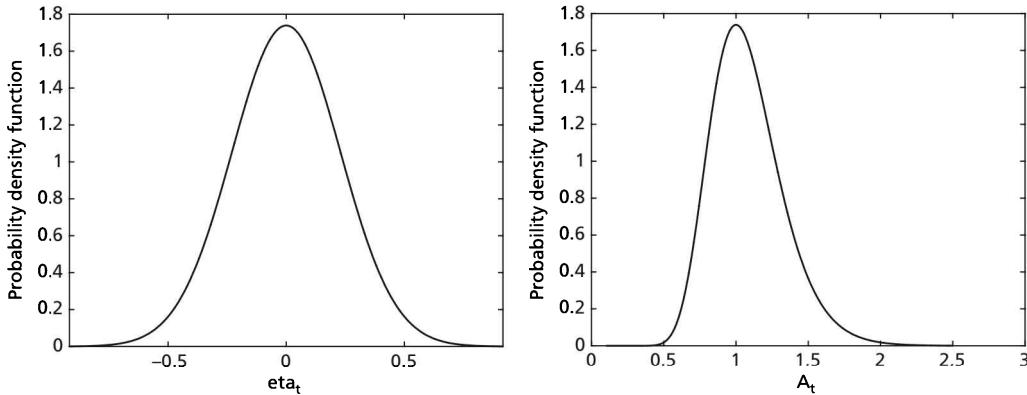


Figure 9.5 Unconditional probability density function of η_t and Ω_t

$$E[\Omega_t] = \exp\left(\frac{\sigma_u^2}{2}\right) \quad \text{and} \quad \text{Var}[\Omega_t] = \exp(\sigma_u^2) \cdot \exp(\sigma_u^2 - 1).$$

Figure 9.5 shows the unconditional probability density functions of η_t and Ω_t for $\sigma_\epsilon^2 = 0.01$.

As already mentioned above, the only difference between the stochastic and the deterministic growth model we analysed above is that Ω_t now follows a stochastic process. We can write the stochastic growth model in its centralized form as

$$\max_{\{c_t\}_{t=0}^\infty} E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t) \middle| \eta_0 \right] \quad \text{s.t.} \quad (1 - \delta)k_t + \exp(\eta_t)f(k_t) = c_t + k_{t+1}$$

and $\eta_{t+1} = \rho\eta_t + \epsilon_{t+1}$ with $\epsilon_{t+1} \sim N(0, \sigma_\epsilon^2)$.

There are now two initial conditions for the model: an initial level of capital k_0 as well as an initial level of the technology shock η_0 . The discounted sum of utilities is now replaced by an expected discounted sum of utilities. Expectations are formed with respect to the distribution of all technology levels η_t , conditional on the fact that the initial technology shock is η_0 .

The dynamic programming problem The question at hand is whether we can write the above maximization problem in a manageable dynamic program. In order to figure this out, we assume that the first t realizations of the technology shock $\{\eta_s\}_{s=0}^t$ were already resolved and the consumption and investment decisions up to time t were already made. This would mean that we already know the capital stock k_t at time t and would only have to determine the optimal path of consumption from time t onwards. Since part of the uncertainty regarding the stochastic process has already been resolved, we can write the respective sub-optimization problem as

$$\begin{aligned}
& \sum_{s=0}^{t-1} \beta^s u(c_s) + \max_{\{c_s\}_{s=t}^{\infty}} \beta^t u(c_s) + E_t \left[\sum_{s=t+1}^{\infty} \beta^s u(c_s) \middle| \eta_0, \eta_1, \dots, \eta_t \right] \\
& \text{s.t. } (1 - \delta)k_s + \exp(\eta_s)f(k_s) = c_s + k_{s+1} \\
& \text{and } \eta_{s+1} = \rho\eta_s + \epsilon_{s+1} \quad \text{with } \epsilon_{s+1} \sim N(0, \sigma_{\epsilon}^2) \quad \text{for all } s \geq t + 1.
\end{aligned}$$

with a given k_t . This is obviously equivalent to maximizing

$$\max_{\{c_s\}_{s=t}^{\infty}} u(c_t) + \beta E_t \left[\sum_{s=t+1}^{\infty} \beta^{s-(t+1)} u(c_s) \middle| \eta_0, \eta_1, \dots, \eta_t \right].$$

The interpretation of this sub-maximization problem is straightforward. The uncertainty about the technology level has already been revealed up to time t . Therefore we can choose the current utility level under certainty. The only risk that remains is with respect to the period $t + 1$ and beyond. We have to form correct expectations about this risk knowing all the previous realizations of $\eta_s, s \leq t$. Due to the way our stochastic process is constructed we can immediately see that η_t is a sufficient statistic for the state of the stochastic process at time t . This means that η_t summarizes everything that has been going on with the stochastic process until date t and it is enough to know η_t when one wants to make predictions about any future $\eta_{t+1}, \eta_{t+2}, \dots$. We can then transform the maximization problem as we did in Chapter 8 to

$$\begin{aligned}
& \max_{\{c_s\}_{s=t}^{\infty}} u(c_t) + \beta E_t \left[\sum_{s=t+1}^{\infty} \beta^{s-(t+1)} u(c_s) \middle| \eta_t \right] \\
\Leftrightarrow & \max_{c_t} u(c_t) + \beta E_t \left[\max_{\{c_s\}_{s=t+1}^{\infty}} u(c_{t+1}) + \beta E_{t+1} \left[\sum_{s=t+2}^{\infty} \beta^{s-(t+2)} u(c_s) \middle| \eta_{t+1} \right] \middle| \eta_t \right]
\end{aligned}$$

and so on. Hence, we formulate the dynamic programming problem of the stochastic growth model as

$$\begin{aligned}
V(k, \eta) &= \max_c u(c) + \beta E \left[V(k^+, \eta^+) \middle| \eta \right] \tag{9.6} \\
&\text{s.t. } (1 - \delta)k + \exp(\eta)f(k) = c + k^+ \\
&\text{and } \eta^+ = \rho\eta + \epsilon^+ \quad \text{with } \epsilon^+ \sim N(0, \sigma_{\epsilon}^2).
\end{aligned}$$

9.2.2 A NUMERICAL IMPLEMENTATION USING DISCRETIZED SHOCKS

Our dynamic optimization problem now has two state variables which are continuous. The first is the capital stock k of the economy. We learned how to treat this variable

through discretization in Chapter 8. The question that remains is what to do with our auto-regressive stochastic process η . As we saw above, a conditional distribution function $\pi(\eta^+|\eta)$ exists that tells us the distribution of η^+ , conditional on the fact that η is already known. With this knowledge we can write the dynamic programming problem in (9.6) as

$$\begin{aligned} V(k, \eta) = \max_c & u(c) + \beta \int_{\mathcal{E}} V(k^+, \eta^+) \pi(d\eta^+|\eta) \\ \text{s.t. } & (1 - \delta)k + \exp(\eta)f(k) = c + k^+, \end{aligned}$$

where $\mathcal{E} = (-\infty, \infty)$ is the support of the stochastic process η^+ . This formulation tells us that in order to form correct expectations about tomorrow's value function, we have to solve our problem for every potential $\eta^+ \in \mathcal{E}$ and then attach a weight according to the distribution function $\pi(\eta^+|\eta)$ to each of the possible states. This would involve solving infinitely many optimization problems, since η^+ is a continuous variable.

Fortunately, there also exist discretization methods for stochastic processes that approximate an infinite dimensional autoregressive process by means of a finite Markov chain. In the following we apply the so-called Rouwenhorst method in order to discretize the stochastic process. The latter (as well as any other discretization method) constructs a finite grid of realizations $\hat{\mathcal{E}} = \{\hat{\eta}_g\}_{g=1}^m$. In addition, it determines a so-called *transition matrix*

$$\hat{\pi}(\hat{\eta}_{g+} | \hat{\eta}_g) = \begin{bmatrix} \pi_{11} & \pi_{12} & \dots & \pi_{1m} \\ \pi_{21} & \pi_{22} & \dots & \pi_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{m1} & \pi_{m2} & \dots & \pi_{mm} \end{bmatrix}.$$

The transition matrix tells us with which probability tomorrow's shock takes the value $\hat{\eta}_{g+}$ given that today's shock value was $\hat{\eta}_g$. The discrete shock levels $\{\hat{\eta}_g\}_{g=1}^m$ and the entries of the transition matrix π_{gg+} are then chosen in a way so that they approximate the autoregressive process η_t . As an example, for $m = 2$ the Rouwenhorst method will give us

$$\hat{\mathcal{E}} = \left\{ -\frac{\sigma_\epsilon}{\sqrt{1 - \rho^2}}, \frac{\sigma_\epsilon}{\sqrt{1 - \rho^2}} \right\} \quad \text{and} \quad \hat{\pi}(\hat{\eta}_{g+} | \hat{\eta}_g) = \begin{bmatrix} \frac{1+\rho}{2} & \frac{1-\rho}{2} \\ \frac{1-\rho}{2} & \frac{1+\rho}{2} \end{bmatrix}.$$

Having discretized the state space for capital to $\{\hat{k}_v\}_{v=0}^n$ and knowing a discrete grid of shock realizations $\hat{\mathcal{E}}$ as well as a corresponding transition matrix that approximate our original process η_t , we can write the above dynamic program as

$$\begin{aligned}
V(\hat{k}_v, \hat{\eta}_g) = & \max_c u(c) + \beta \sum_{g^+=1}^m \pi_{gg^+} \cdot V(k^+, \hat{\eta}_{g^+}) \\
\text{s.t. } & (1 - \delta)\hat{k}_v + \exp(\hat{\eta}_g)f(\hat{k}_v) = c + k^+ \\
& \text{for all } v = 0, \dots, n \text{ and } g = 1, \dots, m.
\end{aligned} \tag{9.7}$$

This set of $(n + 1) \cdot m$ optimization problems can then be tackled using our computer.

First-order conditions We want to solve the problem in (9.7) with our standard utility function using first-order conditions. The Lagrangian of this problem reads

$$\begin{aligned}
\mathcal{L} = & \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \sum_{g^+=1}^m \pi_{gg^+} \cdot V(k^+, \hat{\eta}_{g^+}) \\
& + \lambda \left[(1 - \delta)\hat{k}_v + \exp(\hat{\eta}_g)f(\hat{k}_v) - c - k^+ \right]
\end{aligned}$$

and the first-order condition is

$$c^{-\frac{1}{\gamma}} = \beta \sum_{g^+=1}^m \pi_{gg^+} \cdot V_k(k^+, \hat{\eta}_{g^+})$$

where we denote $V_k(k^+, \hat{\eta}_{g^+}) = \frac{\partial V(k^+, \hat{\eta}_{g^+})}{\partial k^+}$. Like in the deterministic growth model we obtain

$$V_k(k^+, \hat{\eta}_{g^+}) = [1 + \exp(\hat{\eta}_{g^+})\alpha(k^+)^{\alpha-1} - \delta] c(k^+, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}}$$

by applying the envelope theorem. All in all, in order to get a solution to the stochastic growth model we have to solve the set of first-order conditions

$$c(\hat{k}_v, \hat{\eta}_g) = \left[\beta \sum_{g^+=1}^m \pi_{gg^+} \cdot [1 + \exp(\hat{\eta}_{g^+})\alpha(k^+)^{\alpha-1} - \delta] c(k^+, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}} \right]^{-\gamma}.$$

We can determine the solution to the stochastic growth model through policy function iteration.

Computational approach Program 9.3 shows how to apply the policy function iteration procedure to the stochastic growth model. To do this, we have to prepare a couple of things. First of all, we have to define all relevant variables. This is done in the module `globals` that we do not want to show here. The main difference to the deterministic

Program 9.3 The stochastic growth model: policy function iteration

```

program StochasticGrowth
    [.....]
    ! calculate the shock process
    call discretize_AR(rho, 0d0, sigma_eps, eta, pi)

    ! initialize grid and policy function
    call grid_Cons_Equi(k, k_l, k_u)
    do is = 1, NS
        c(:, is) = exp(eta(is))*k(:)**alpha
    enddo

    ! iterate until policy function converges
    do iter = 1, itermax

        ! interpolate coefficients
        do is = 1, NS
            call spline_interp(c(:, is), coeff_c(:, is))
        enddo

        ! calculate decisions for every gridpoint and shock level
        [.....]

        ! get convergence level
        con_lev = maxval(abs(c_new(:, :) - c(:, :)) &
                           /max(abs(c(:, :)), 1d-10))
        write(*,'(i5,2x,f20.7)')iter, con_lev

        ! check for convergence
        [.....]
    enddo
    [.....]
end program

```

Ramsey model is that our state space is two-dimensional now. Consequently, the policy function needs to be an array of two dimensions $c(0:NK, NS)$. Thereby NK indicates the number of gridpoints on the state space for capital and NS the number of points with which we discretize our stochastic process. We can discretize the stochastic process using the subroutine `discretize_AR` provided in our toolbox. This subroutine takes as inputs three parameters. The autocorrelation coefficient ρ of the shock process, the unconditional mean (which is 0 in our case), and the variance of the innovation σ_{ϵ} . It then calculates the nodes $\hat{\eta}_g$ and the respective transition matrix entries $\pi_{gg'}$ and stores them in the arrays `eta(NS)` and `pi(NS, NS)`. The state space for capital k is discretized as usual.

Before starting with the policy function iteration process, we have to initialize the policy function at every gridpoint for capital and every potential shock level. Setting c to $\exp(\hat{\eta}_g)\hat{k}_v^\alpha$ we assume as an initial guess that the total value of production is immediately consumed. Having initialized the policy function, we have to interpolate it. Note that we do not interpolate the policy function in-between the different values of `eta` but only in-between the values of `k`. The reason is that `eta` only takes values from the discrete set of nodes determined in the subroutine `discretize_AR`. We consequently determine m different spline functions $S(k, \hat{\eta}_g)$ for $g = 1, \dots, m$ that satisfy

$$S(\hat{k}_v, \hat{\eta}_g) = c(\hat{k}_v, \hat{\eta}_g) \quad \text{for all } v = 0, \dots, n.$$

The first-order conditions we have to solve then read

$$c(\hat{k}_v, \hat{\eta}_g) = \left[\beta \sum_{g^+=1}^m \pi_{gg^+} \cdot [1 + \exp(\hat{\eta}_{g^+})\alpha(k^+)^{\alpha-1} - \delta] S(k^+, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}} \right]^{-\gamma}.$$

We proceed with the policy function iteration process as usual, meaning we determine new policy functions by solving the first-order conditions, check for convergence, and update the policy functions if the policy function iteration process has not yet converged.

The more important part, however, is how to determine a new policy function in the stochastic growth model. Program 9.3.a shows the respective algorithm, which is a straightforward extension of the algorithm in Program 9.1. We iterate over our complete policy space and determine the root of the first-order condition for every potential level of current capital \hat{k}_v and technology $\hat{\eta}_g$. We use the consumption level of the previous iteration period as our starting guess and apply the root-finding method in the subroutine `fzero` to the function `foc`. This time, the procedure requires two communication variables. The variable `k_com` tells the function `foc` for which current capital level we want to calculate the first-order condition. The variable `is_com`, on the other hand, passes the index of the current technology shock to the function `foc`. Note that it is essential to pass the index, not the actual shock level, as our spline functions do not interpolate policy functions along the shock dimension.

The function that calculates the first-order condition is stored in the module `globals`. Module 9.3m shows what this function looks like. The function takes as input the current level of consumption `x_in`. From this it calculates the value of tomorrow's capital stock

Program 9.3.a The stochastic growth model: root-finding

```

! calculate decisions for every gridpoint and shock level
do ik = 0, NK
    do is = 1, NS

        ! set starting value and communicate resource level
        x_in = c(ik, is)
        k_com = k(ik)
        is_com = is

        ! find the optimal consumption level
        call fzero(x_in, foc, check)
        if(check) write(*,*) 'ERROR IN ROOTFINDING PROCESS'

        ! get optimal consumption and value function
        c_new(ik, is) = x_in

    enddo
enddo

```

Module 9.3m The stochastic growth model: the first-order condition

```

function foc(x_in)
    [.....]
    ! future capital stock
    kplus = (1d0-delta)*k_com + exp(eta(is_com))*k_com**alpha - x_in

    ! calculate future expected marginal utility
    foc = 0d0
    do is_p = 1, NS
        cplus = spline_eval(min(kplus, k_u), &
                            coeff_c(:, is_p), k_l, k_u)
        if (kplus > k_u) then
            cplus = cplus + (c(NK,is_p)-c(NK-1,is_p)) &
                    / (k(NK)-k(NK-1))*(kplus-k_u)
        endif
        foc = foc + pi(is_com, is_p) &
              *(1d0+exp(eta(is_p))*alpha &
              *kplus**alpha-1d0)-delta)*cplus**(-1d0/gamma)
    enddo

    ! get first-order condition
    foc = x_in - (beta*foc)**(-gamma)

```

function

`kplus` and evaluates the right-hand side of the first-order condition at this point. In order to do this we have to iterate over any potential level of tomorrow's technology, which we index by `is_p`. At each level of technology $\hat{\eta}_g^+$ we evaluate the spline function $S(k^+, \hat{\eta}_g^+)$. However, the spline function only gives us reasonable values for the policy function on the interval $[k_l, k_u]$. Outside of this interval, the spline function quickly drops down to zero. To avoid this, we extrapolate the policy function to the right of our interval $[k_l, k_u]$ as soon as `kplus` exceeds k_u . This is a fairly easy procedure. We just calculate the policy function at the point `k_u` and then simply assume that it would evolve linearly to the right, where the slope of this linear curve is approximated using the last two points $c(\hat{k}_{n-1}, \hat{\eta}_g^+)$ and $c(\hat{k}_n, \hat{\eta}_g^+)$ of the policy function. Having calculated consumption at `kplus` and the technology level $\hat{\eta}_g^+$, we can attach the respective weight from the transition matrix to the technology state `is_p`, conditional on the current technology level being `is_com`. The value of the first-order condition is finally returned to our root-finding routine.

Policy functions We run the model with the same parameter values as in the deterministic model. In addition, the persistence of the process for technology is set to $\rho = 0.95$ and the variance of the innovation term to $\sigma_\epsilon^2 = 0.000049$. As soon as the policy function iteration process has converged, the subroutine `output` produces some graphs that inform us about the properties of our model. Figure 9.6 shows the policy functions for different technology levels $\hat{\eta}_g$. Since the production of consumption goods increases in the technology level, a higher technology shock comes along with a larger consumption level.

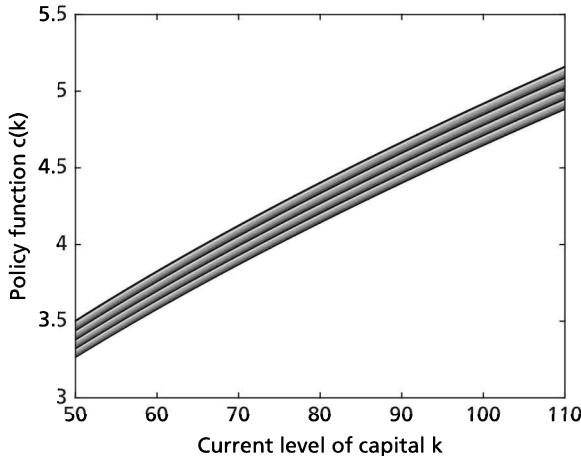


Figure 9.6 Policy functions of the stochastic growth model

9.2.3 SIMULATING TIME PATHS

Looking only at policy functions conditional on the current technology shock is fairly easy. The more tricky part is to simulate a time path of consumption and capital. Since this model has a stochastic nature, there will no longer be any well-behaved convergence to a steady state given an initial capital level. Instead the economy will fluctuate with the realizations of the stochastic process. The key element of generating these fluctuations is to simulate a time path of the stochastic process η_t . This can be done with the subroutine `simulate_AR` from the toolbox. This subroutine receives as input a transition matrix `pi` and a vector of integer values, in our case `is_t(0:TT)`. The subroutine then stores a time path of shock realizations that arise from the transition matrix `pi` in this vector. It thereby proceeds as follows: The first realization `is_t(0)` will be just the middle point of the discretized process $m/2 + 1$. In our case, in which we used `NS = 21`, we obtain `is_t(0) = 11`. For every follow-up realization the subroutine makes a random draw `x` from the uniform distribution on the interval $[0, 1]$. It then divides the interval $[0, 1]$ into `NS` intervals according to the transition matrix entries. Specifically, conditional on having a draw `is_t(it)` for period t , the index of which we denote j , the subroutine divides the interval $[0, 1]$ into the intervals

$$A_k = \left[\sum_{g=1}^{k-1} \pi_{jg}, \sum_{g=1}^k \pi_{jg} \right].$$

The realization `is_t(it+1)` then is equal to k if the random draw `x` lies in the interval A_k . Figure 9.7 illustrates this algorithm for our stochastic growth model with $m = 21$ discretization nodes for the stochastic process. As the previous realization of the process

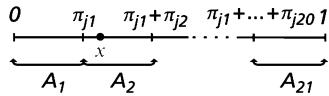


Figure 9.7 Division of $[0, 1]$ according to transition matrix entries

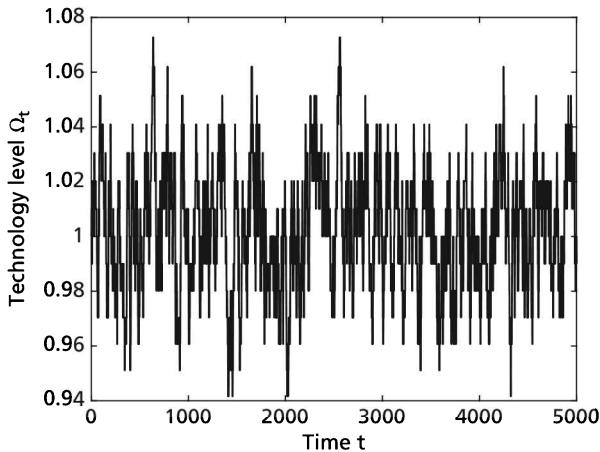


Figure 9.8 Simulated technology path

had the index j , the interval is divided according to the j th row of the transition matrix. Since the realization of the uniformly distributed variable x lies in the second interval, the next index will be $\text{is_t}(it+1) = 2$. Figure 9.8 shows a simulated series of technology shocks $\Omega_t = \exp(\eta_t)$ resulting from this stochastic process. The realizations could be characterized as moving in waves. This is the typical result of a high auto-correlation. If the auto-correlation was low, the wave movement would disappear. Note that the sequence of realizations will look different every time we simulate the model, since they are drawn from a random number generator.

Finally given the time path of technology shocks, we can simulate a path for consumption and capital stock of our economy. We therefore set $k_0 = 79$ which is close to the mean capital stock of this economy. Program 9.3.b shows how this simulation is accomplished. We first generate the sequence of shock realizations that is stored in the integer array is_t . We then initialize the capital stock $k_t(0) = k_0$ and calculate the consumption level in period 0 using our interpolated policy function. For each period following we can now calculate the capital stock from the budget constraint of the economy and the consumption realization from the policy function. Note that we use the shock realization both to determine the value of production as well as when choosing the coefficients of the interpolated policy function. In Figure 9.9 we see how consumption and capital evolve over time for the technology realizations in Figure 9.7. We find that consumption and capital both follow the ups and downs of technology. Yet consumption

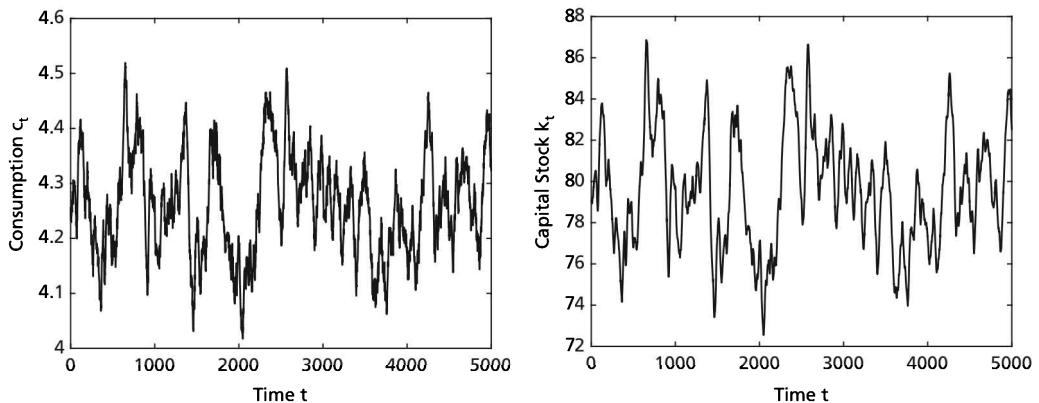
Program 9.3.b The stochastic growth model: simulating time paths

```

call simulate_AR(pi, is_t)

! calculate the time path of consumption and capital numerically
eta_t(0) = eta(is_t(0))
k_t(0) = k0
c_t(0) = spline_eval(k_t(0), coeff_c(:, is_t(0)), k_l, k_u)
do it = 1, TT
    eta_t(it) = eta(is_t(it))
    k_t(it) = (1d0-delta)*k_t(it-1) +
               + exp(eta_t(it-1))*k_t(it-1)**alpha - c_t(it-1)
    c_t(it) = spline_eval(k_t(it), coeff_c(:, is_t(it)), k_l, k_u)
enddo

```

**Figure 9.9** Simulated paths for consumption and capital

exhibits a few more short-run fluctuations while capital is more rigid. This is due to the fact that capital needs time to build through investment. Hence, it will only react to a technology shock in the periods following not in the period of the shock itself.

9.2.4 SPEEDING UP THE COMPUTATIONAL PROCESS

Calculating the solution to the stochastic growth model takes roughly 12 seconds. There is, however, quite an easy way to speed up this computational process. In Program 9.3 we interpolated the policy function for each realization of the stochastic process $\{\hat{\eta}_{g^+}\}_{g^+=1}^m$. The first-order condition was then calculated by computing

$$\left[\beta \sum_{g^+=1}^m \pi_{gg^+} \cdot [1 + \exp(\hat{\eta}_{g^+})\alpha(k^+)^{\alpha-1} - \delta] S(k^+, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}} \right]^{-\gamma}.$$

In consequence we evaluate the spline function m times every time we call the function `foc`. With our parameter choice of $m = 21$ this takes quite some time. An easy and much faster alternative, however, is to not interpolate the policy function for each realization $\{\hat{\eta}_{g^+}\}_{g^+=1}^m$ of tomorrow's technology level, but to interpolate the complete right-hand side of the first-order condition. This means that for each *today's* technology level $\{\hat{\eta}_g\}_{g=1}^m$ we have to find a spline function $S(k, \hat{\eta}_g)$ that satisfies

$$S(\hat{k}_v, \hat{\eta}_g) = \left[\beta \sum_{g^+=1}^m \pi_{gg^+} \cdot \left[1 + \exp(\hat{\eta}_{g^+}) \alpha (\hat{k}_v)^{\alpha-1} - \delta \right] c(\hat{k}_v, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}} \right]^{-\gamma}.$$

The first-order condition then simply reduces to

$$c(\hat{k}_v, \hat{\eta}_g) = S(k^+, \hat{\eta}_g)$$

which means that we only have to evaluate a spline function once each time the function `foc` is called. Program 9.4 uses this interpolation technique. The interpolation is now organized in the subroutine `interpolate`. The subroutine loops over every potential realization of today's technology and calculates the value of the right-hand side of the first-order condition for any gridpoint. Thus, it has to loop over all potential future technology levels and attach the respective weight from the transition matrix. Finally the subroutine `spline_interp` calculates the spline coefficients and stores them in the array `coeff_r`.

Module 9.4m shows how the first-order condition simplifies compared to Module 9.3m. The spline function now has to be evaluated only once. Note that we keep up the same

Program 9.4 Speeding up the stochastic growth model

```

! For interpolating the RHS of the first-order condition
subroutine interpolate()
    [.....]
    do is = 1, NS
        do ik = 0, NK

            ! calculate the RHS of the first-order condition
            RHS(ik, is) = 0d0
            do is_p = 1, NS
                RHS(ik, is) = RHS(ik, is) + pi(is, is_p) &
                    *(1d0+exp(eta(is_p))*alpha*k(ik)**(alpha-1d0) &
                     -delta)*c_new(ik, is_p)**(-1d0/gamma)
            enddo
            RHS(ik, is) = (beta*RHS(ik, is))**(-gamma)
        enddo

        ! interpolate
        call spline_interp(RHS(:, is), coeff_r(:, is))
    enddo

end subroutine

```

Module 9.4m Speeding up the stochastic growth model: first-order condition

```

function foc(x_in)
    [.....]
    ! future capital stock
    kplus = (1d0-delta)*k_com + exp(eta(is_com))*k_com**alpha - x_in

    ! calculate future expected marginal utility
    foc = spline_eval(min(kplus, k_u), &
                      coeff_r(:, is_com), k_l, k_u)
    if(kplus > k_u)then
        foc = foc + (RHS(NK, is_com)-RHS(NK-1, is_com)) &
                           / (k(NK)-k(NK-1)) * (kplus-k_u)
    endif

    ! get first-order condition
    foc = x_in - foc
end function

```

linear extrapolation procedure in case k^+ exceeds the upper bound of the grid. When we run the program we see that the run-time reduces to about three seconds, while the Euler equation error becomes even smaller.

9.3 The real business-cycle model

The stochastic growth model of Section 9.2 only constitutes an intermediate step on the way to the real business-cycle (RBC) model. The amount of capital in a given period is fully determined through investment in the period before. Consequently, when a productivity shock hits the economy in the stochastic growth model, it is fully passed through to aggregate output since the production factors can no longer be adjusted. To deal with this issue, the RBC model combines aggregate uncertainty with endogenous labour supply so as to allow for immediate reactions to fluctuations in total factor productivity. In the following we introduce the RBC model by making labour supply a choice variable. We first derive the planner's dynamic programming problem when there is a labour-leisure trade-off and show how to solve this problem numerically. We then compare business-cycle statistics derived from the model with actual data. Finally, we want to quantify the welfare costs of the business cycle and study the effects of pro-cyclical government expenditure in relation to expenditure that is constant over the cycle.

9.3.1 A DYNAMIC PROGRAM WITH ENDOGENOUS LABOUR SUPPLY

When there is a labour-leisure trade-off, household's instantaneous utility function has as argument both (regular) consumption and leisure consumption, i.e. it reads $u(c_t, 1-l_t)$ where l_t denotes the amount of labour that is supplied to the market. We thereby

assume that the household has a total time endowment of 1 which can be split between leisure consumption and labour supply.⁸ The remainder of the model is almost left untouched with the exception that we have to write total production as a function of capital and labour input $f(k_t, l_t)$. The dynamic program that describes the RBC economy consequently reads

$$\begin{aligned} V(k, \eta) = \max_{c, l} & u(c, 1 - l) + \beta E \left[V(k^+, \eta^+) \middle| \eta \right] \\ \text{s.t. } & (1 - \delta)k + \exp(\eta)f(k, l) = c + k^+ \\ \text{and } & \eta^+ = \rho\eta + \epsilon^+ \quad \text{with } \epsilon^+ \sim N(0, \sigma_\epsilon^2). \end{aligned}$$

A possible specification of household preferences is

$$u(c, 1 - l) = \frac{[c^\nu(1 - l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Utility of consumption and leisure $1 - l$ is aggregated according to a Cobb-Douglas function, where ν is a taste parameter for consumption. Aggregate utility from consumption and leisure again takes the form of a constant elasticity of substitution utility with an intertemporal elasticity of substitution of γ .

Using our discretized shocks and transition matrix, we can write the set of dynamic programs we have to solve as

$$\begin{aligned} V(\hat{k}_v, \hat{\eta}_g) = \max_{c, l} & \frac{[c^\nu(1 - l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \sum_{g^+=1}^m \pi_{gg^+} \cdot V(k^+, \hat{\eta}_{g^+}) \\ \text{s.t. } & (1 - \delta)\hat{k}_v + \exp(\hat{\eta}_g)(\hat{k}_v)^\alpha l^{1-\alpha} = c + k^+ \\ & \text{for all } v = 0, \dots, n \quad \text{and } g = 1, \dots, m, \end{aligned}$$

where we again assumed a Cobb-Douglas technology $f(k, l) = \exp(\eta)k^\alpha l^{1-\alpha}$ in production. The first-order conditions of this problem read

$$\frac{\nu}{c} \cdot [c^\nu(1 - l)^{1-\nu}]^{1-\frac{1}{\gamma}} = \lambda \tag{9.8}$$

$$\frac{1 - \nu}{1 - l} \cdot [c^\nu(1 - l)^{1-\nu}]^{1-\frac{1}{\gamma}} = \lambda \left[(1 - \alpha) \exp(\hat{\eta}_g) \left(\frac{\hat{k}_v}{l} \right)^\alpha \right] \tag{9.9}$$

$$\beta \sum_{g^+=1}^m \pi_{gg^+} \cdot V_k(k^+, \hat{\eta}_{g^+}) = \lambda \tag{9.10}$$

⁸ Of course we could also write the instantaneous utility function as a function of consumption and labour supply $u(c_t, l_t)$, with $u_l < 0$ and $u_{ll} \leq 0$. However, given our choice of functional form, a formulation with leisure seems more natural.

with

$$V_k(k^+, \hat{\eta}_{g^+}) = \left[1 + \exp(\hat{\eta}_{g^+})\alpha \left(\frac{k^+}{l^+} \right)^{\alpha-1} - \delta \right] \cdot \frac{\nu \cdot [(c^+)^{\nu}(1-l^+)^{1-\nu}]^{1-\frac{1}{\gamma}}}{c^+}. \quad (9.11)$$

Combining (9.8) and (9.9) we immediately see that

$$c = \frac{\nu}{1-\nu} \cdot (1-l) \cdot \exp(\hat{\eta}_g)(1-\alpha) \left(\frac{\hat{k}_v}{l} \right)^\alpha \quad (9.12)$$

needs to hold. Finally combining first-order conditions (9.8) and (9.10) we get

$$\frac{\nu \cdot [c^{\nu}(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{c} = \beta \sum_{g^+=1}^m \pi_{gg^+} \cdot V_k(k^+, \hat{\eta}_{g^+}). \quad (9.13)$$

Note that (9.13) together with (9.11) and (9.12) as well as the resource constraint define an equation that solely depends on the current gridpoint \hat{k}_v , the current shock level $\hat{\eta}_g$, as well as the control variable l . Or put differently, given a gridpoint \hat{k}_v , a shock $\hat{\eta}_g$, and a level of labour supply l , we can use (9.12) to directly calculate the corresponding consumption level c and the resource constraint to determine tomorrow's level of capital k^+ . The only equation to be solved is then the first-order condition (9.13) with V_k being defined in (9.11). This equation defines the optimal level of labour supply.

9.3.2 NUMERICAL IMPLEMENTATION WITH POLICY FUNCTION ITERATION

We solve the RBC model using policy function iteration in the same way as we solved the stochastic growth model. Program 9.5 shows how the main program needs to be adapted to accomplish this. Before initializing the consumption policy function, we have to give the program starting values for labour supply $l(:, is)$. We assume that a household works about 30 per cent of its total time endowment. Given this guess of labour supply, we set the initial value of consumption to the level of aggregate production.

The policy function iteration then proceeds as usual. In order to solve the first-order condition (9.13) numerically, we have to interpolate its right-hand side, which is a function solely of the future level of capital k^+ and the current shock level $\hat{\eta}_g$. Therefore, we determine spline functions $S(k, \hat{\eta}_g)$ in the subroutine `interpolate` that satisfy

Program 9.5 The RBC model: policy function iteration

```

program RBC
    [.....]
    ! initialize grid and policy function
    call grid_Cons_Equi(k, k_l, k_u)
    do is = 1, ns
        l(:, is) = 0.3d0
        c(:, is) = exp(eta(is))*k(:)**alpha*l(:, is)**(1d0-alpha)
        c_new(:, is) = c(:, is)
    enddo

    ! iterate until policy function converges
    do iter = 1, itermax

        ! interpolate coefficients
        call interpolate()

        ! calculate decisions for every gridpoint and shock level
        do ik = 0, NK
            do is = 1, NS

                ! set starting value and communicate resource level
                x_in = l(ik, is)
                k_com = k(ik)
                is_com = is

                ! find the optimal labour-supply level
                call fzero(x_in, foc, check)
                if(check)write(*,*)'ERROR IN ROOTFINDING PROCESS'

                ! get optimal consumption function
                l(ik, is) = x_in
                c_new(ik, is) = nu/(1d0-nu)*(1d0-x_in) &
                    *exp(eta(is))*(1d0-alpha)*(k(ik)/x_in)**alpha

            enddo
        enddo
    [.....]
end program

```

$$S(\hat{k}_v, \hat{\eta}_g) = \left\{ \beta \sum_{g^+=1}^m \pi_{gg^+} \cdot \left[1 + \exp(\hat{\eta}_{g^+}) \alpha \left(\frac{\hat{k}_v}{l^+} \right)^{\alpha-1} - \delta \right] \cdot \frac{\nu \cdot [(c^+)^{\nu} (1-l^+)^{1-\nu}]^{1-\frac{1}{\gamma}}}{c^+} \right\}^{-\gamma}$$

with

$$c^+ = c(\hat{k}_v, \hat{\eta}_{g^+}) \quad \text{and} \quad l^+ = l(\hat{k}_v, \hat{\eta}_{g^+}).$$

Note that we attach an exponent $-\gamma$ to the right-hand side of the first-order condition before we interpolate it. The idea behind this is the same as in Section 8.2.2. Attaching

an exponent of $-\gamma$, the value of the right-hand side of the first-order condition will not diverge to infinity, but converge to zero for small levels of capital. Therefore the interpolating spline function will do a much better job on this transformed first-order condition than on the original one. Having interpolated the right-hand side of the first-order condition, we can solve it using the root-finding routine provided in `fzero`. The unknown variable in this procedure is labour supply l . In addition to setting an initial guess for l , we have to communicate the current amount of capital stock `k_com` and the current productivity shock `is_com` to the first-order condition `foc`. The subroutine `fzero` returns an optimal level of labour supply by means of which we can update the consumption policy function `c_new`.

The nonlinear equation we have to solve in order to determine the optimal level of labour supply l reads

$$\frac{\nu \cdot [c^\nu (1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{c} = S(k^+, \hat{\eta}_g)^{-\frac{1}{\gamma}} \quad (9.14)$$

with

$$c = \frac{\nu}{1-\nu} \cdot (1-l) \cdot \exp(\hat{\eta}_g)(1-\alpha) \left(\frac{\hat{k}_v}{l} \right)^\alpha \quad \text{and}$$

$$k^+ = (1-\delta)\hat{k}_v + \exp(\hat{\eta}_g)(\hat{k}_v)^\alpha l^{1-\alpha} - c.$$

This optimality condition is implemented in the function `foc`, which is shown in Module 9.5m. The function `foc` takes as its only input argument labour supply l . From this and the communication variables it calculates consumption as well as tomorrow's capital stock. It then derives the value of the first-order condition that should be set to zero.

9.3.3 COMPARING MODEL RESULTS TO THE DATA

Once we have solved for the policy functions $c(\hat{k}_v, \hat{\eta}_g)$ and $l(\hat{k}_v, \hat{\eta}_g)$ with our iteration routine, we want to compare some model results to real-life data. In order to do so, we need to simulate time series of all the economic variables of our economy. This is done in the subroutine `output`, parts of which are shown in Program 9.5.a. We thereby assume the same parameterization as in Section 9.3.2 and set $\nu = 0.36$.

Before we can simulate our economy we interpolate the policy functions $c(\hat{k}_v, \hat{\eta}_g)$ and $l(\hat{k}_v, \hat{\eta}_g)$ along the capital dimension for each level of aggregate technology $\hat{\eta}_g$. In addition, we need a suitable starting level for the capital stock k_0 . We obtain this initial level of capital by setting $\eta_t = 0$ for all $t \geq 0$ and simulating the economy with

Module 9.5m First-order condition of the RBC model

```

function foc(x_in)
    [.....]
    ! calculate optimal actual consumption level
    c_act = nu/(1d0-nu)*(1d0-x_in)*exp(eta(is_com)) &
            *(1d0-alpha)*(k_com/x_in)**alpha

    ! calculate future capital
    kplus = (1d0-delta)*k_com + exp(eta(is_com))*k_com**alpha &
            *x_in***(1d0-alpha) - c_act

    ! evaluate spline function
    foc = spline_eval(min(kplus, k_u), coeff_r(:, is_com), k_l, k_u)
    if(kplus > k_u)then
        foc = foc + (RHS(NK,is_com)-RHS(NK-1,is_com)) &
                /(k(NK)-k(NK-1))*(kplus-k_u)
    endif
    foc = foc**(-1d0/gamma)

    ! evaluate first-order condition
    foc = nu*(c_act**nu*(1d0-x_in)**(1d0-nu))**egam/c_act - foc

end function

```

Program 9.5.a Simulation of time series of economic variables

```

subroutine output()
    [.....]
    ! interpolate policy functions
    do is = 1, NS
        call spline_interp(c_new(:, is), coeff_c(:, is))
        call spline_interp(l(:, is), coeff_l(:, is))
    enddo

    ! determine the stochastic steady state
    is_t = floor(dble(NS)/2d0)+1
    call simulate_economy()
    k0 = k_t(TT)

    ! simulate a series of shocks
    call simulate_AR(pi, is_t)

    ! simulate the economy forward
    call simulate_economy()

    ! calculate business cycle statistics
    call bc_statistics()

    ! write output
    [.....]
end subroutine

```

the decision rules $c(\hat{k}_v, \hat{\eta}_g)$ and $l(\hat{k}_v, \hat{\eta}_g)$. The simulation of the economy is conducted in the subroutine `simulate_economy` which will be discussed in detail later. Note that when the number of shocks is uneven, then the shock level `eta(NS/2+1)`—meaning the median shock—is exactly zero. This simulation procedure leads us to the so-called

*stochastic steady state.*⁹ We use the capital level in the stochastic steady state as a starting point for our stochastic economy. Finally, we draw a series of $TT = 1000000$ realizations for the technology shock process using the subroutine `simulate_AR` and use them to simulate the economy again. However this time the economy will not converge to a steady state but we will obtain real stochastic time series for consumption, capital, output, etc. Finally, the subroutine `bc_statistics` calculates relevant business-cycle statistics from our simulated time series and the corresponding output is printed to the console.

Program 9.5.b shows how the time series of economic variables are generated in the subroutine `simulate_economy`. For a given path of technology shocks `is_t` and the specified initial level of capital `k0`, the simulation routine calculates the values of consumption and labour supply from the interpolated policy functions $c(\hat{k}_v, \hat{\eta}_g)$ and $l(\hat{k}_v, \hat{\eta}_g)$. Knowing current capital and labour supply it is easy to calculate total output. Gross investment can be determined from the goods market equilibrium and factor prices from marginal products of capital and labour. Using the law of motion for capital, we can derive next period's level of capital and repeat all the simulation steps to successively simulate the economy going forward.

Program 9.5.b Simulating the economy

```

subroutine simulate_economy()
    [.....]
    eta_t(0) = eta(is_t(0))
    k_t(0) = k0
    c_t(0) = spline_eval(k_t(0), coeff_c(:, is_t(0)), k_l, k_u)
    l_t(0) = spline_eval(k_t(0), coeff_l(:, is_t(0)), k_l, k_u)
    y_t(0) = exp(eta_t(0))*k_t(0)**alpha*l_t(0)**(1d0-alpha)
    i_t(0) = y_t(0)-c_t(0)
    r_t(0) = exp(eta_t(0))*alpha*(l_t(0)/k_t(0))**(1d0-alpha)-delta
    w_t(0) = exp(eta_t(0))*(1d0-alpha)*(k_t(0)/l_t(0))**alpha
    do it = 1, TT
        eta_t(it) = eta(is_t(it))
        k_t(it) = (1d0-delta)*k_t(it-1) + i_t(it-1)
        c_t(it) = spline_eval(k_t(it), &
                               coeff_c(:, is_t(it)), k_l, k_u)
        l_t(it) = spline_eval(k_t(it), &
                               coeff_l(:, is_t(it)), k_l, k_u)
        y_t(it) = exp(eta_t(it))*k_t(it)**alpha*l_t(it)**(1d0-alpha)
        i_t(it) = y_t(it)-c_t(it)
        r_t(it) = exp(eta_t(it))*alpha &
                  *(l_t(it)/k_t(it))**(1d0-alpha) - delta
        w_t(it) = exp(eta_t(it))*(1d0-alpha) &
                  *(k_t(it)/l_t(it))**alpha
    enddo
end subroutine

```

⁹ There is a fundamental difference between the stochastic and the deterministic steady state calculated in Section 9.1. In the deterministic steady state the planner makes decisions in a certain world without shocks. In the stochastic steady state, however, he acts as if there were shocks to the economy, but the shocks are manually shut down.

Program 9.5.c Simulating the economy

```

subroutine bc_statistics()

implicit none

! calculate expectations
E_c = sum(c_t)/dble(TT+1)
E_i = sum(i_t)/dble(TT+1)
E_k = sum(k_t)/dble(TT+1)
[.....]
! normalization on annual level
E_k = E_k/4d0
E_r = (1d0+E_r)**4-1d0

! calculate coefficient of variation
CV_c = CofVar(c_t)
CV_i = CofVar(i_t)
CV_k = CofVar(k_t)
[.....]
! calculate correlation coefficients of logs
Cor_c = correlation(c_t, y_t)
Cor_i = correlation(i_t, y_t)
Cor_k = correlation(k_t, y_t)
[.....]
end subroutine

```

The subroutine `bc_statistics`, excerpts of which are shown in Program 9.5.c, calculates important business-cycle statistics. We first derive averages (or expectations) of all macroeconomic variables by just summing up the values in the time series and dividing them by the number of simulation periods.¹⁰ Note that for the capital stock we make an adjustment as all model variables are measured on a quarterly basis. To express the (mean) capital stock as a fraction of annual output, we divide it by a factor of four. In a similar way we adjust the expected interest rate to an annual level. Next we want to compare the volatility of different macroeconomic variables, especially in relation to the volatility of aggregate output. As our economic variables, however, have very different magnitudes, we do not use their standard deviation but the *coefficient of variation (CV)*. The coefficient of variation is the standard deviation divided by the mean and can therefore be interpreted as the standard deviation of a variable measured in per cent. For a time series x_t , we can consequently calculate the CV from the formula

$$\text{CV}(x) = \frac{\sqrt{\frac{1}{T} \cdot \sum_{t=0}^T (x_t - \mu_x)^2}}{\mu_x} \quad \text{with} \quad \mu_x = \frac{1}{T+1} \cdot \sum_{t=0}^T x_t.$$

¹⁰ One could also drop the first maybe 5,000 values from the simulated time series, as these first values are influenced by our specific choice of the initial shock level and the initial capital stock k_0 . However, since we simulate 1,000,000 values for the time series, it turns out that the impact of initial choices for the business-cycle moments is minor. Therefore we use the full time series to keep things as simple as possible.

Finally we calculate the empirical correlation coefficient between certain variables and aggregate output from the formula

$$\rho(x, y) = \frac{1}{T} \cdot \frac{\sum_{t=0}^T (x_t - \mu_x)(y_t - \mu_y)}{\sigma_x \sigma_y}.$$

The formulas for the CV and the correlation coefficient are implemented in the functions `CofVar` and `correlation`, respectively.

Business-cycle statistics Table 9.2 shows the calculated business-cycle statistics from our model. In the first row we can see the means of variables, where consumption, investment, and capital stock are expressed as a percentage of GDP. Labour hours are reported as a percentage of the total time endowment. Investment amounts to about 26 per cent of GDP. The capital to (annual) output ratio is roughly 3.4 which is a bit too large for the United States. However, it generates a realistic average interest rate of roughly 4 per cent per year. Our choice of ν guarantees that households work about 30 per cent of their time endowment. Finally, the wage level equals 3.44. The next rows display the coefficients of variation in relation to the CV of GDP generated from our model, as well as the respective data for the US. The values can be read as follows: A CV of 0.71 for consumption relative to GDP means that consumption is only 71 per cent as volatile as aggregate output. This indicates that there is a significant amount of consumption-smoothing going on in the economy. The social planner adjusts investment and labour supply such that relatively more can be consumed in times when total factor productivity is low. To achieve this there must be a lot of investment in times when output is high in order to build up contingency capital stock that can be used as insurance against future falls in output. Hence investment is much more volatile than output, which is in line with the US data. Overall we find that the model fits the business-cycle statistics for consumption quite well. However, when it comes to labour supply it predicts a correlation of 0.8 with output. In the US economy the empirical

Table 9.2 Stochastic properties of the RBC model

| | c | i | k | l | r | w |
|------------------------|-------|-------|--------|-------|------|------|
| Average (in %) | 73.87 | 26.13 | 343.77 | 31.35 | 4.10 | 3.44 |
| CV (in % of CV of GDP) | 0.71 | 2.12 | 1.00 | 0.30 | 2.02 | 0.79 |
| US Data* | 0.69 | 1.35 | | 0.52 | | 1.14 |
| Correlation with GDP | 0.92 | 0.93 | 0.75 | 0.77 | 0.35 | 0.97 |
| US Data* | 0.85 | 0.60 | | 0.07 | | 0.76 |

* Adda and Cooper (2003), p. 127.

correlation is only 0.07. At the same time the model-generated volatility of labour hours is too low.

Euler equation errors Last but not least, we want to test the accuracy of our approximation algorithm. Note that we cannot use the first-order condition in (9.14), since this equation does not measure the error in the first-order condition in consumption values. We instead use the transformed euler equation

$$c(\hat{k}_v, \hat{\eta}_g, l) - \frac{v \cdot \left[c(\hat{k}_v, \hat{\eta}_g, l)^v (1-l)^{1-v} \right]^{1-\frac{1}{\gamma}}}{S(k^+, \hat{\eta}_g)^{-\frac{1}{\gamma}}} = 0.$$

This transformation is implemented in the function `foc2` and its units are again consumption units so that we can calculate a proper Euler equation error from it. In our numerical setup with $k_l = 10$, $k_u = 40$ and $n = 100$ the Euler equation error is of order 10^{-8} .

9.3.4 THE WELFARE COSTS OF BUSINESS-CYCLE FLUCTUATIONS

As we've just seen, fluctuations in total factor productivity cause fluctuations in all economics variables, especially in consumption and labour supply. One question typically posted in the literature is whether such fluctuations induce welfare costs for households, and if yes, how large these welfare costs are. In the following, we want to quantify the welfare costs of business-cycle fluctuations and break the costs down into different components.

We start this analysis by using a second-order Taylor approximation to write the instantaneous utility function as

$$\begin{aligned} u(c, 1-l) &\approx u(\mu_c, 1-\mu_l) + u_c(\mu_c, 1-\mu_l) \cdot (c - \mu_c) \\ &+ u_{1-l}(\mu_c, 1-\mu_l) \cdot (1-l - (1-\mu_l)) \\ &+ \frac{1}{2} \cdot u_{c,c}(\mu_c, 1-\mu_l) \cdot (c - \mu_c)^2 \\ &+ \frac{1}{2} \cdot u_{1-l,1-l}(\mu_c, 1-\mu_l) \cdot (1-l - (1-\mu_l))^2 \\ &+ u_{c,1-l}(\mu_c, 1-\mu_l) \cdot (c - \mu_c) \cdot (1-l - (1-\mu_l)). \end{aligned}$$

Note that we expand the utility function around the average values of consumption μ_c and leisure consumption $1 - \mu_l$ in the RBC model, as shown in the first row of Table 9.2.

With this approximation of the instantaneous utility function, we can write the expected utility function of households as of time $t = 0$ as

$$\begin{aligned} E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right] &\approx \\ &\approx \frac{1}{1 - \beta} \cdot \left\{ u(\mu_c, 1 - \mu_l) + \frac{1}{2} \cdot u_{c,c}(\mu_c, 1 - \mu_l) \cdot E[(c - \mu_c)^2] \right. \\ &\quad + \frac{1}{2} \cdot u_{1-l,1-l}(\mu_c, 1 - \mu_l) \cdot E[(1 - l - (1 - \mu_l))^2] \\ &\quad \left. + u_{c,1-l}(\mu_c, 1 - \mu_l) \cdot E[(c - \mu_c) \cdot (1 - l - (1 - \mu_l))] \right\}, \end{aligned}$$

where the terms $E[c - \mu_c]$ and $E[1 - l - (1 - \mu_l)]$ cancel out to zero. By suitably multiplying and dividing with μ_c and $1 - \mu_l$ the term on the right-hand side becomes

$$\begin{aligned} &\frac{1}{1 - \beta} \cdot \left\{ u(\mu_c, 1 - \mu_l) + \frac{1}{2} \cdot u_{c,c}(\mu_c, 1 - \mu_l) \cdot (\mu_c)^2 \cdot E \left[\left(\frac{c - \mu_c}{\mu_c} \right)^2 \right] \right. \\ &\quad + \frac{1}{2} \cdot u_{1-l,1-l}(\mu_c, 1 - \mu_l) \cdot (1 - \mu_l)^2 \cdot E \left[\left(\frac{1 - l - (1 - \mu_l)}{1 - \mu_l} \right)^2 \right] \\ &\quad \left. + u_{c,1-l}(\mu_c, 1 - \mu_l) \cdot \mu_c \cdot (1 - \mu_l) \cdot E \left[\frac{c - \mu_c}{\mu_c} \cdot \frac{1 - l - (1 - \mu_l)}{1 - \mu_l} \right] \right\}. \end{aligned}$$

Note that $E \left[\left(\frac{c - \mu_c}{\mu_c} \right)^2 \right]$ measures the variance of consumption in relative terms (or in per cent), which is just equal to the squared coefficient of variation $CV[c]^2$.¹¹ By the same reasoning we obtain

$$\begin{aligned} E \left[\left(\frac{1 - l - (1 - \mu_l)}{1 - \mu_l} \right)^2 \right] &\approx CV[1 - l]^2 \quad \text{and} \\ E \left[\frac{c - \mu_c}{\mu_c} \cdot \frac{1 - l - (1 - \mu_l)}{1 - \mu_l} \right] &\approx \rho[c, 1 - l] \cdot CV[c] \cdot CV[1 - l]. \end{aligned}$$

¹¹ Alternatively, one could argue that the variance in relative terms is approximately equal to the variance of the log of consumption, as is done in other studies. This approximation works equally well. However, since we have already implemented the CV in our program, we will stick to it in our approximation of the business-cycle costs.

Therefore we can write

$$\begin{aligned} E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right] &\approx \\ &\approx \frac{1}{1-\beta} \cdot \left\{ u(\mu_c, 1 - \mu_l) + \frac{1}{2} \cdot u_{c,c}(\mu_c, 1 - \mu_l) \cdot (\mu_c)^2 \cdot \text{CV}[c]^2 \right. \\ &+ \frac{1}{2} \cdot u_{l-l,1-l}(\mu_c, 1 - \mu_l) \cdot (1 - \mu_l)^2 \cdot \text{CV}[1 - l]^2 \\ &\left. + u_{c,1-l}(\mu_c, 1 - \mu_l) \cdot \mu_c \cdot (1 - \mu_l) \cdot \rho[c, 1 - l] \cdot \text{CV}[c] \cdot [1 - l] \right\}. \end{aligned}$$

Having approximated the expected utility function in the RBC model, we have to find a suitable point of comparison. The most natural point to compare this utility with is the utility households would attain in a world in which there were no productivity shocks but aggregate production would just be equal to the average aggregate productivity in the RBC model $\bar{\Omega}$. Obviously when there are no aggregate fluctuations, the economy would be in a steady state. We can calculate the corresponding steady-state levels of consumption and labour using the first-order conditions in (9.12) and (9.13) as well as the resource constraint which yields

$$l^* = \frac{\nu \cdot \bar{\Omega} \cdot (1 - \alpha) \cdot [k^*]^\alpha}{(1 - \nu\alpha) \cdot \bar{\Omega} \cdot [k^*]^\alpha - (1 - \nu) \cdot \delta \cdot k^*} \quad \text{and} \quad c^* = [\bar{\Omega} \cdot [k^*]^\alpha - \delta \cdot k^*] \cdot l^* \quad (9.15)$$

with $k^* = \left[\frac{\bar{\Omega} \cdot \alpha}{\theta + \delta} \right]^{\frac{1}{1-\alpha}}$ measuring the capital to labour ratio in the steady state. The aggregate utility level in the steady state is consequently

$$\sum_{t=0}^{\infty} \beta^t u(c^*, 1 - l^*) = \frac{1}{1-\beta} \cdot u(c^*, 1 - l^*).$$

We now want to measure the cost of business-cycle fluctuations in consumption equivalent variation, meaning we ask by how many percentage points we would have to increase or decrease the consumption of a household living in the steady state, with a fixed level of total factor productivity, so as to make that household as well off as a household living in a world with business-cycle fluctuations. The equation we need to solve to answer this question reads

$$\frac{1}{1-\beta} \cdot u((1 + \Delta)c^*, 1 - l^*) = E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right],$$

with Δ being the unknown. Using a first-order Taylor expansion of the utility in the steady state we can write this equation as

$$\frac{1}{1-\beta} [u(c^*, 1-l^*) + u_c(c^*, 1-l^*) \cdot c^* \cdot \Delta] = E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1-l_t) \right].$$

With the above approximation of the expected utility level in the RBC model, we quickly find that

$$\begin{aligned} \Delta \approx & \frac{u(\mu_c, 1-\mu_l) - u(c^*, 1-l^*)}{u_c(c^*, 1-l^*) \cdot c^*} + \frac{1}{2} \cdot \left\{ \frac{u_{cc}(\mu_c, 1-\mu_l) \cdot (\mu_c)^2}{u_c(c^*, 1-l^*) \cdot c^*} \cdot \text{CV}[c]^2 \right. \\ & + \frac{u_{1-l,1-l}(\mu_c, 1-\mu_l) \cdot (1-\mu_l)^2}{u_c(c^*, 1-l^*) \cdot c^*} \cdot \text{CV}[1-l]^2 \\ & \left. + 2 \cdot \frac{u_{c,1-l}(\mu_c, 1-\mu_l) \cdot \mu_c \cdot (1-\mu_l)}{u_c(c^*, 1-l^*) \cdot c^*} \cdot \rho[c, 1-l] \cdot \text{CV}[c] \cdot \text{CV}[1-l] \right\}. \end{aligned}$$

Finally, we can use the derivatives of the utility function to write

$$\begin{aligned} \Delta \approx & \frac{\hat{u} - 1}{v \left(1 - \frac{1}{\gamma} \right)} - \frac{\hat{u}}{2v\gamma} \cdot \left\{ [\nu^2 + \bar{\nu}] \cdot \text{CV}[c]^2 + [(1-\nu)^2 + \bar{\nu}] \cdot \text{CV}[1-l]^2 \right. \\ & \left. + 2 \cdot [\nu(1-\nu) - \bar{\nu}] \cdot \rho[c, 1-l] \cdot \text{CV}[c] \cdot \text{CV}[1-l] \right\} \end{aligned} \quad (9.16)$$

with $\bar{\nu} = (1-\nu)\nu\gamma$ and $\hat{u} = \frac{u(\mu_c, 1-\mu_l)}{u(c^*, 1-l^*)}$.

The cost of business-cycle fluctuations in our RBC model has four different factors. Three factors are linked to the fluctuations of the two utility components. Specifically, when individuals are risk-averse, fluctuations in consumption and leisure consumption will deteriorate a household's utility. Such variations in the utility components are especially bad when consumption and leisure are positively correlated, i.e. a drop in consumption is accompanied by a drop in leisure. Note that, as households become more risk-averse, meaning $\frac{1}{\gamma}$ increases, the utility loss from consumption and leisure fluctuations will increase. However, there is a fourth factor driving the welfare effects of business-cycle fluctuations. This factor is called a level effect. It is related to the fact that average consumption and/or leisure in a world with productivity fluctuations might differ from average consumption and/or leisure in a world in which productivity is just constant. We will discuss the magnitude of this effect as well as whether it is positive or negative below.

Calculating the cost of fluctuations in the model Program 9.6 shows how to calculate the costs of business-cycle fluctuations in the RBC model. The entire calculation can be

done in the subroutine `output`, so that no other part of the program needs to be changed. First we calculate the CV of leisure consumption as well as the correlation coefficient between consumption and leisure. We then determine the steady-state consumption level and labour supply from (9.15). We therefore use the average level of total factor productivity μ_Ω . Recalling the discussion about autoregressive processes in Section 9.2.1, we know that expected productivity equals $\mu_\Omega = \exp\left(0.5 \cdot \frac{\sigma_\epsilon^2}{(1-\rho^2)}\right)$. Next we calculate the utility level generated from the steady-state values as well as the utility level from the average values of consumption and leisure in the RBC model with fluctuations. From these we can directly derive \hat{u} . In addition, we calculate the expected level of utility of a household that lives in the RBC world. We can do this simply by calculating the average of all utility levels in each simulated period.¹² Note that we drop the factor $\frac{1}{1-\beta}$, since we will compare the utility level `E_V` directly to the steady-state level of utility which equals $\frac{u^*}{1-\beta}$. Therefore the factor will cancel out anyway. Last but not least, we can derive all four components of the cost of business-cycle fluctuations.

Table 9.3 summarizes the results of our calculations. In the first row, we can see the squared coefficients of variation of consumption and leisure as well as their correlation coefficient. We already observed that the fluctuations are quite small. When we look at the correlation coefficient we find that consumption and leisure are negatively correlated. This is not surprising when we recall that both labour and consumption are procyclical, see Table 9.2. In the next row we can see the four factors that influence the welfare

Program 9.6 Calculating the cost of business-cycle fluctuations

```

! get CV and correlation
CV_leis = CofVar(1d0-1_t)
Cor_leis = correlation(c_t, 1d0-1_t)

! determine steady state values under certainty
Omega = exp(0.5d0*sigma_eps/(1d0-rho**2))
k_star = ((Omega*alpha)/(1d0/beta-1d0+delta))**((1d0/(1d0-alpha)))
l_star = nu*Omega*(1d0-alpha)*k_star**alpha &
        /((1d0-nu*alpha)*Omega*k_star**alpha - (1d0-nu)*delta*k_star)
c_star = (Omega*k_star**alpha-delta*k_star)*l_star

! get different utility levels
u_star = (c_star**nu*(1d0-1_star)**(1d0-nu))**egam/egam
u_mu = (E_c**nu*(1d0-E_l)**(1d0-nu))**egam/egam
u_hat = u_mu/u_star
E_V = sum((c_t**nu*(1d0-1_t)**(1d0-nu))**egam/egam)/db1e(TT+1)

! get different components of business cycle costs
bar_nu = (1d0-nu)*nu*gamma
bc_lev = (u_hat-1d0)/(nu*egam)
bc_c = -u_hat/(2d0*nu*gamma)*(nu**2 + bar_nu)*CV_c**2
bc_leis = -u_hat/(2d0*nu*gamma)*((1d0-nu)**2 + bar_nu)*CV_leis**2
bc_cor = -u_hat/(nu*gamma)*(nu*(1d0-nu)-bar_nu)*Cor_leis*CV_c*CV_leis
bc_tot = bc_lev + bc_c + bc_leis + bc_cor

```

¹² It can be shown that this is identical to actually calculating the utility function $E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right]$.

Table 9.3 The welfare costs of business-cycle fluctuations

| | Level | CV(c) | CV($1 - l$) | $\rho(c, 1 - l)$ | Total |
|------------------------|--------|-----------|---------------|------------------|---------|
| Values | | 0.0276 | 0.0054 | -0.4604 | |
| Approx. cost (in %) | 0.0304 | -0.0519 | -0.0043 | 0.0044 | -0.0214 |
| Model simulated (in %) | | | | | -0.0214 |

costs of business-cycle fluctuations. Interestingly, the level effect is positive, meaning that consumption and/or leisure are, on average, higher in a world with fluctuations than in a world with constant aggregate productivity. The reason for this is that in a world with stochastic total factor productivity, the planner can exploit times of high productivity by allocating more labour supply to periods with positive shocks. In turn, when productivity is low, households will be allowed to consume more leisure. This reallocation of labour supply to more productive periods leads to higher average consumption and therefore to a positive-level effect. Fluctuations in consumption and leisure, on the other hand, reduce the welfare of households. Note that, since the CV of consumption is much larger than the CV in leisure, the costs of consumption fluctuations are more than 10 times as large than the costs of leisure fluctuations. Finally, the fact that consumption and leisure are negatively correlated slightly reduces the welfare costs of business-cycle fluctuations. Overall, however, the total costs of shocks to aggregate productivity are fairly small. They only amount to about 0.021 per cent of aggregate consumption.¹³ This is in line with what the previous literature has found. Last but not least, we want to compare the approximated total welfare cost which we derived from a second-order Taylor expansion of the utility function with the actual welfare costs. This will tell us something about the accuracy of our approximated welfare measure. Recall that we calculate the actual expected utility function $E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right]$ from averaging all instantaneous utility levels. We can use this expected utility level to calculate the welfare costs of business-cycle fluctuations directly from

$$\frac{u((1 + \Delta)c^*, 1 - l^*)}{1 - \beta} = E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right]$$

$$\Rightarrow \Delta = \left[\frac{E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_t, 1 - l_t) \right]}{\frac{u(c^*, 1 - l^*)}{1 - \beta}} \right]^{\frac{1}{\nu(1-\frac{1}{\gamma})}} - 1.$$

Note that we used the properties of our choice of functional form for u to solve for Δ . The resulting welfare change (in %) is shown in the last row of Table 9.3. We find that the two numbers are almost identical, which is not too surprising given the fact that we look at small shocks to aggregate productivity and that Taylor approximations work pretty well if

¹³ Note that these numbers cannot be replicated exactly by simulation.

deviations from the point of expansion are not too large. Therefore it seems safe to use the approximated welfare measure, the great advantage of which is that we can break down the welfare effect directly into level and fluctuation components.

Overall we have seen that the existence of technology shocks and the resulting fluctuations in economic variables can lead to welfare costs for households. If we want to calculate these costs, we have to keep in mind that we should not exclusively look at the variance of consumption and leisure, but should also take into account that a suitable reallocation of labour supply to periods with high productivity can lead to a positive-level effect, which mitigates the negative welfare impact of fluctuations. In Section 9.3.5, we want to study whether the government can influence the volatility of consumption, leisure, and aggregate output by letting its consumption be procyclical and whether such procyclical policy impacts households' welfare.

9.3.5 PROCYCLICAL VS. CONSTANT GOVERNMENT EXPENDITURE

We extend the standard RBC model by government activity in the simplest possible way. We assume that the government consumes the amount g_t in period t . To finance its consumption it levies lump-sum taxes T_t from households. Like in the deterministic Ramsey economy, the presence of government consumption changes the aggregate resource constraint to

$$(1 - \delta)k_t + \exp(\eta_t)f(k_t, l_t) = c_t + g_t + k_{t+1}.$$

Since we allow the government to collect taxes in a non-distortionary way,¹⁴ this is actually the only equation that changes in the model. The first-order conditions defining consumption and labour supply will stay unchanged. In order to study the effects of procyclical government expenditure policy, we consider two scenarios: In Scenario 1 the government will always consume a fixed fraction g_y of total output, meaning that $g_t = g_y \cdot y_t$. In Scenario 2, government consumption is constant and equals the average level of government consumption in Scenario 1, i.e. $g_t = \bar{g}$.

We want to compare the two scenarios above both in terms of the business-cycle properties, but also in terms of household welfare. Specifically, we would again like to calculate by how many per cent Δ consumption would have to change in Scenario 1 so as to make households as equally well off as they are in Scenario 2. With the above discussion about the welfare costs of business cycle fluctuations in mind, it is possible to show that this consumption change can be calculated from

$$\Delta \approx \hat{u} \cdot W_2 - W_1, \quad (9.17)$$

¹⁴ We do not consider taxes on capital and labour income for simplicity, both of which would distort households' decisions.

where

$$W_i = \frac{1}{\nu \left(1 - \frac{1}{\gamma}\right)} - \frac{1}{2\nu\gamma} \cdot \left\{ [\nu^2 + \bar{v}] \cdot \text{CV}_i[c]^2 + [(1-\nu)^2 + \bar{v}] \cdot \text{CV}_i[1-l]^2 + 2 \cdot [\nu(1-\nu) - \bar{v}] \cdot \rho_i[c, 1-l] \cdot \text{CV}_i[c] \cdot \text{CV}_i[1-l]\right\}.$$

$\text{CV}_i[c]$ thereby denotes the CV of consumption in Scenario $i = 1, 2$ and $\hat{u} = \frac{u(\mu_{2,c}, 1 - \mu_{2,l})}{u(\mu_{1,c}, 1 - \mu_{1,l})}$ is again the ratio of utilities derived from average consumption and leisure in the two scenarios.

Simulating the two scenarios Program 9.7 shows how we have to adapt the main program code in order to properly simulate the two scenarios.

While in the previous section the comparison economy did not feature any aggregate productivity shock, this time both scenarios involve business-cycle fluctuations. To make our calculation accurate, we use the same sequence of shocks is_t to simulate both scenarios.¹⁵ Consequently after having discretized the shock process, we simulate one series of productivity shocks using the subroutine `simulate_AR` and store the result in an array `is_sim(0:TT)`. The subroutine `output` falls back on this shock series whenever an economy is simulated. The logical variable `variable_g` indicates whether government policy should be procyclical or constant over the business cycle. As we have to calculate policy functions several times, we moved the policy function iteration code from the main program into a subroutine `getPolicy`. Having calculated policy functions under a procyclical government consumption regime, we call up the subroutine `output`, which simulates the economy forward, calculates relevant business cycle moments, and prints them on the console. Finally, we have to store the means, coefficients of variation and correlation needed to calculate our welfare measure Δ . We then repeat all these steps, but with government consumption that is constant over time and equal to the average level of government consumption in the first scenario. Having done that we can eventually turn to the welfare analysis, which is implemented in the subroutine `welfare_analysis`. The remainder of the program is almost left untouched. We only have to slightly adapt the resource constraint to include government consumption. In addition, when we simulate the economy we also have to calculate a time series for g_t . However, these changes are very similar to those in the deterministic Ramsey economy, so we do not have to repeat them here.

Table 9.4 shows the business-cycle statistics in the two scenarios. When we compare the averages (in the rows Scenario 1 and 2) to the corresponding values in Table 9.2,

¹⁵ If we used different shock sequences for the two scenarios, differences in welfare effects might arise, e.g. from the two sequences having slightly different means.

Program 9.7 Simulating two policy scenarios

```

! calculate the shock process
call discretize_AR(rho, 0d0, sigma_eps, eta, pi)

! simulate a series of shocks (for both models)
call simulate_AR(pi, is_sim)

! simulate economy with variable g
variable_g = .true.
call getPolicy()
call output()

! store variables for welfare analysis
WE_c(1)      = E_c
WE_leis(1)    = 1d0-E_1
WCV_c(1)      = CV_c
WCV_leis(1)   = CofVar(1d0-l_t)
WCor_leis(1)  = correlation(c_t, 1d0-l_t)

! simulate economy with fixed g
g_level      = E_g
variable_g = .false.
call getPolicy()
call output()

! store variables for welfare analysis
WE_c(2)      = E_c
WE_leis(2)    = 1d0-E_1
WCV_c(2)      = CV_c
WCV_leis(2)   = CofVar(1d0-l_t)
WCor_leis(2)  = correlation(c_t, 1d0-l_t)

! perform welfare analysis
call welfare_analysis()

```

Table 9.4 Stochastic properties of the RBC model

| | c | g | i | k | l | y |
|------------------------------------------------|-------|-------|-------|--------|-------|------|
| Scenario 1: Procyclical government expenditure | | | | | | |
| Average (in %) | 58.87 | 15.00 | 26.13 | 343.77 | 36.43 | |
| CV (in %) | 2.58 | 3.74 | 7.37 | 3.50 | 1.13 | 3.74 |
| Correlation with GDP | 0.91 | 1.00 | 0.94 | 0.73 | 0.79 | |
| Scenario 2: Constant government expenditure | | | | | | |
| Average (in %) | 58.87 | 15.00 | 26.13 | 343.77 | 36.43 | |
| CV (in %) | 3.07 | 0.00 | 7.95 | 3.85 | 0.91 | 3.59 |
| Correlation with GDP | 0.92 | 0.00 | 0.94 | 0.74 | 0.55 | |

we find that private consumption has declined by 15 percentage points of GDP. This difference is now completely consumed by the government so that aggregate investment relative to GDP stays unchanged. The same is true for the capital-to-output ratio. This results from the fact that we let the government levy lump-sum taxes. Therefore the

savings decision of the household is not distorted and the (average) capital intensity must be identical to the one in a world without government consumption. Nevertheless, the lump-sum tax on households triggers an income effect, so that leisure consumption falls and labour supply increases compared to the situation in Table 9.2. Finally, note that whether government consumption is procyclical or constant over the cycle has no impact on the average values of consumption, investment, and labour supply, since the average level of government consumption is identical in the two scenarios.

As the design of government policy has no impact on average economic activity, it is most interesting to look at the second moments of relevant economic variables. Note that this time we show the actual coefficient of variation (in %) and not the CV relative to the CV of GDP. This facilitates the interpretation of results more. We find that when government consumption is constant, the volatility of aggregate consumption relative to GDP is much greater than under a procyclical government expenditure regime. This is not surprising in light of the fact that government consumption crowds out private consumption. When government expenditure is constant over the business cycle, then relatively more of GDP can be consumed in times of high productivity and vice versa. On the other hand, procyclical government consumption leads to a higher volatility of labour supply as well as a more positive correlation of labour hours with aggregate output. This can be explained in the following way: When government expenditure rises in an upswing of economic activity, households experience a negative income effect. This negative income effect leads to a decline in the absolute level of consumption and also in leisure consumption. Consequently, labour supply has to increase. The same reasoning holds for an economic downswing, but in the opposite direction. This income effect is absent when government expenditure is constant over the cycle. The last column of Table 9.2 reports the CV of GDP y . Beneath fluctuations in total factor productivity, aggregate output is affected by the two factor inputs capital and labour. While the volatility of the capital stock is larger in an economy with constant government expenditure, the CV of labour hours falls. At the same time, labour becomes less cyclical. Overall, the effects of labour input dominate the impact of a higher volatility of capital, so that aggregate output has a smaller variance when government expenditure is constant over the business cycle. Consequently in the current model setup with our choice of calibration, procyclical government policy destabilizes the economy.

Welfare analysis Although pro-cyclical government policy is destabilizing and increases the variance of labour hours, it leads to a lower volatility of consumption. The question that remains to be answered is in which of the two scenarios households would prefer to live. To answer this question, we use the welfare breakdown discussed above. The respective welfare numbers are calculated in the subroutine `welfare_analysis` which is shown in Program 9.7.a. In this subroutine we first calculate the utility levels that correspond to the average levels of consumption and leisure in each scenario and then derive all the single factors impacting W_i , i.e. a level component as well as the effects of consumption and leisure volatility. Finally, we can again calculate the welfare effect

Program 9.7.a A welfare analysis of government expenditure policy

```

subroutine welfare_analysis
    [.....]
    ! calculate the different welfare components
    do ii = 1, 2
        util(ii) = (WE_c(ii)**nu*WE_leis(ii)**(1d0-nu))**egam/egam
        W_lev(ii) = 1d0/(nu*(1d0-gamma))
        W_c(ii) = -1d0/(2d0*nu*gamma) &
                   *(nu**2 + bar_nu)*WCV_c(ii)**2
        W_leis(ii) = -1d0/(2d0*nu*gamma) &
                   *((1d0-nu)**2 + bar_nu)*WCV_leis(ii)**2
        W_cor(ii) = -1d0/(nu*gamma)*(nu*(1d0-nu) - bar_nu) &
                   *WCor_leis(ii)*WCV_c(ii)*WCV_leis(ii)
        W_tot(ii) = W_lev(ii) + W_c(ii) + W_leis(ii) + W_cor(ii)
    enddo
    ! calculate welfare changes due to different components
    uhat = util(2)/util(1)
    Del_lev = uhat*W_lev(2)-W_lev(1)
    Del_c = uhat*W_c(2)-W_c(1)
    Del_leis = uhat*W_leis(2)-W_leis(1)
    Del_cor = uhat*W_cor(2)-W_cor(1)
    Del_tot = uhat*W_tot(2)-W_tot(1)
    [.....]
end subroutine

```

Table 9.5 The welfare effects of procyclical vs. constant government consumption

| | Level | CV(c) | CV(1 - l) | $\rho(c, 1 - l)$ | Total |
|---------------------|---------|---------|-----------|------------------|---------|
| Values scenario 1 | -1.2398 | 0.0258 | 0.0065 | -0.4536 | |
| Values scenario 2 | -1.2398 | 0.0306 | 0.0053 | -0.1811 | |
| Approx. cost (in %) | 0.0055 | -0.0184 | 0.0021 | -0.0030 | -0.0138 |

of moving from Scenario 1 to Scenario 2. The first two rows of Table 9.5 list the utility level as well as the square CVs of consumption and leisure and their correlation in the two scenarios. As expected, since average consumption and labour supply are (almost exactly) the same in the two scenarios, the utility levels are identical and there is no level effect of moving from one scenario to the other. As the volatility of consumption increases from moving to a constant government consumption, policy welfare declines by 0.018 per cent, while the fall in the variance of leisure leads to a (tiny) increase in welfare. Finally, a reduction in the correlation between consumption and leisure depresses welfare. Overall, the impact of a reduction in consumption volatility weighs heavier than any other effect, so that in total welfare decreases by 0.0138 per cent (measured in consumption terms) when we move from a procyclical government consumption scheme to one that is constant over the cycle.¹⁶

The previous analysis showed how we can use the RBC model to estimate the welfare costs of business-cycle fluctuations and conduct some analysis of government policy. We

¹⁶ Again, these numbers will change slightly in each simulation.

saw that both level effects as well as changes in the volatility of economic variables can impact households, welfare. That said, we also learned that the RBC model performs relatively poorly in terms of the business-cycle statistics of labour supply when compared to the data. Therefore there is certainly room for improving the model setup and potentially its calibration. Nevertheless, what we have shown should suffice to explain the principles of macroeconomic models with aggregate risk and a representative consumer.

9.4 The heterogeneous agent model

One simplifying assumption of the models we have studied so far was that the household side of the economy can be described by a representative consumer. Such an assumption can be useful if one is only interested in the effects of *aggregate shocks*, i.e. shocks that hit the economy as a whole and therefore are common to each individual living in this economy. However, the economics literature has also recognized that there are many shocks that only hit one individual, so-called *idiosyncratic shocks*. The most important representatives of this kind are idiosyncratic shocks to individual income. By making the assumption of a representative consumer one implicitly postulates that individuals can comprehensively insure themselves against any form of idiosyncratic shocks. If a fully comprehensive insurance mechanism does not exist, shocks will hit different agents differently, so that they become heterogeneous within a cohort. In response, households will adjust their choices to this uncertain environment in order to (at least partly) self-insure against idiosyncratic risk. Guided by this idea, a strand of literature has emerged that is concerned with individual consumption and savings behaviour under idiosyncratic income uncertainty as well as its implications for the macroeconomy.

In Section 9.4.1 we first explain how to model idiosyncratic risk at the household level and how to solve for individual decision rules using policy function iteration. We then show how to aggregate individual decisions to macroeconomic quantities and derive equilibrium prices. Finally we discuss the parameterization of the model and present some simulation results for different degrees of idiosyncratic income risk as well as for government policy reforms.

9.4.1 THE BASIC SETUP

Like our simple neoclassical growth model, the most basic heterogeneous agent model features two sectors: firms and households. In the following, we describe the behaviour of the members of these sectors and define a market equilibrium of the economy.

The firms' problem Firms in the heterogeneous agent model act in exactly the same way as in the neoclassical growth model described above, i.e. in each period t there is an infinite number of identical firms that are perfectly competitive and hire capital K_t and labour L_t in order to produce a single output good using the Cobb-Douglas production technology

$$Y_t = K_t^\alpha \cdot L_t^{1-\alpha},$$

where we normalized the technology level Ω again to 1. For simplicity we assume that the population size is constant over time and normalized to $N_t = 1$, so that per capita variables equal aggregate variables. Conventionally, we denote such aggregate variables with capital letters. In contrast we describe individual variables on the household level with lower-case letters. Note that in the Ramsey growth model, a clear distinction between macroeconomic aggregates and individual variables was not necessary because of the representative agent assumption. In the heterogeneous agent world, however, there will be many different types of households who receive different shocks and make different choices. All of these choices taken together then aggregate to macroeconomic variables.

In the following we restrict our attention to steady states of the economy. While individuals are exposed to idiosyncratic shocks, there is no risk at the aggregate level, like, for example, a technology shock. Consequently prices and aggregate quantities are constant in a steady state and we denote them by K , L , Y , r , and w . Assuming firms maximize profits under the perfect-competition condition, the demand functions of the firms for capital and labour read

$$r = \alpha \cdot \left[\frac{K}{L} \right]^{\alpha-1} - \delta \quad \text{and} \quad w = (1 - \alpha) \cdot \left[\frac{K}{L} \right]^\alpha. \quad (9.18)$$

From the firms' investment equation we can then directly infer that

$$I = K - (1 - \delta)K = \delta \cdot K.$$

A household model with idiosyncratic income shocks Let us assume that the economy is populated by a large number of households indexed by i who live forever. They have identical preferences over consumption $c_{i,t}$ that can be represented by a time-separable discounted utility function

$$E_0 \left[\sum_{t=0}^{\infty} \beta^t u(c_{i,t}) \right]$$

and supply one unit of labour inelastically in every period t . Individuals differ in their labour productivity (or labour efficiency) which we denote $\exp(\eta_{i,t})$. We thereby assume

that $\eta_{i,t}$ is an idiosyncratic shock that follows an AR(1) process. Consequently, labour earnings can be written as

$$y_{i,t} = w \cdot \exp(\eta_{i,t}) \quad \text{with} \quad \eta_{i,t+1} = \rho\eta_{i,t} + \epsilon_{i,t+1} \quad \text{and} \quad \epsilon_{i,t+1} \sim N(0, \sigma_\epsilon^2).$$

Labour earnings therefore are the product of the wage rate w per efficiency unit of labour, the individual's labour efficiency, and the total amount of hours worked (which is equal to 1). At each point in time the individual decides how much to consume $c_{i,t}$ and how much to save for the next period $a_{i,t+1}$. Note that due to the uncertainty of labour income, individuals use savings for precautionary reasons, i.e. to smooth consumption across good and bad states of labour income.¹⁷ Consequently individuals build up a buffer of wealth in good times which they will use in bad times to realize a consumption level that is larger than their income. By assumption, households are not allowed to run into debt, i.e. their savings must always be non-negative.¹⁸ Using the same techniques that we applied before, we can write the choice problem of the individual as a dynamic program. The individual state variables are savings a as well as the shock to labour income η , the control variable is individual consumption c . The respective dynamic programming problem then reads

$$\begin{aligned} V(a, \eta) &= \max_{c, a^+} u(c) + \beta E[V(a^+, \eta^+) | \eta] \\ \text{s.t. } & (1+r)a + w \cdot \exp(\eta) = c + a^+, \quad a^+ \geq 0 \\ \text{and } & \eta^+ = \rho\eta + \epsilon^+ \quad \text{with} \quad \epsilon^+ \sim N(0, \sigma_\epsilon^2). \end{aligned} \tag{9.19}$$

Note that since individuals have identical preferences and they only differ in the realization of the idiosyncratic stochastic process η , we do not have to make i a state variable in the dynamic programming problem.

First-order conditions The Lagrangian of the above optimization problem reads

$$\mathcal{L} = u(c) + \beta E[V(a^+, \eta^+) | \eta] + \lambda [(1+r)a + w \cdot \exp(\eta) - c - a^+]$$

which leads to the first-order condition

$$u'(c) = \beta E[V_a(a^+, \eta^+) | \eta].$$

¹⁷ We already discussed the role of precautionary savings in Section 5.2.

¹⁸ Such a non-indebtedness restriction seems fairly ad hoc. There is, however, also an endogenous borrowing limit in this model. Since households are not allowed to default, they can't borrow more than they will be able to pay back if they were facing the smallest possible labour productivity for the remainder of their existence. If this smallest income level is equal to zero, then the non-indebtedness restriction would be an endogenous outcome of the model.

Applying the envelope theorem in the steady state yields

$$V_a(a^+, \eta^+) = (1 + r)u'(c^+(a^+, \eta^+))$$

and therefore the first-order condition

$$u'(c) = \beta(1 + r)E[u'(c^+(a^+, \eta^+))|\eta].$$

Using the same parameterization of the utility function as above, we then quickly obtain the solution to the household optimization problem as

$$\begin{aligned} c(a, \eta) &= \left[\beta(1 + r)E\left[c^+(a^+, \eta^+)^{-\frac{1}{\gamma}} \middle| \eta\right] \right]^{-\gamma} \\ \text{with } a^+ &= (1 + r)a + w \cdot \exp(\eta) - c, \quad a^+ \geq 0 \\ \text{and } \eta^+ &= \rho\eta + \epsilon^+ \quad \text{with } \epsilon^+ \sim N(0, \sigma_\epsilon^2). \end{aligned}$$

Market equilibrium Firms and households again interact on three markets: the capital market, the labour market, and the goods market. In equilibrium, prices have to be such that all of these markets clear. This means that the interest rate r will ensure that capital demand K of the firms meets the aggregate supply of savings by the households A . In addition the wage rate w will be such that labour demand of the firms L is equal to labour supply L^s of households. Finally, by Walras' law, the goods market will then also be in equilibrium. Note that we use the price of the consumption good as numeraire and normalize it to $p = 1$. One of the challenges when solving the heterogeneous agent model is to aggregate the individual decisions about asset holdings a , consumption c , and the individual labour-supply levels $\exp(\eta)$ to aggregate quantities A , C , and L^s . We will devote a separate section to this task. For now, however, let us just assume that we are able to do this. Of course, since labour is supplied inelastically in this model, aggregate labour supply L^s is constant and does not react to changes in prices. Nevertheless, both aggregate savings and aggregate consumption will respond if prices r and w vary, so that we can write them as functions $A(r, w)$ and $C(r, w)$. The market-clearing conditions of the economy then read

$$L = L^s, \quad K = A(r, w) \quad \text{and} \quad Y = C(r, w) + \delta K.$$

9.4.2 SOLVING FOR MARKET-CLEARING PRICES

We first want to discuss how to solve for market-clearing prices in the heterogeneous agent model. When looking again at the firms' price-setting equations (9.18), we can see that when aggregate labour supply is fixed and given as $L = L^s$, there is a direct

connection between the interest rate r and the capital stock K . In fact, for some arbitrary level of the interest rate, we can back out the corresponding capital stock as

$$K = \left[\frac{\alpha}{r + \delta} \right]^{\frac{1}{1-\alpha}} \cdot L.$$

Knowing the capital stock, we can then directly derive the wage rate w . Hence, there is only one price we actually have to solve for when we want to determine the market equilibrium of the economy, namely the interest rate r that causes the capital market to clear.

We can solve for this price using a standard root-finding method. Program 9.8 shows how to set up the root-finding process. Our program starts with calling the subroutine `initialize` in which we specify a couple of variables that we need for solving the household problem. In addition, this subroutine determines the level of aggregate labour supply stored in the variable `LL`.¹⁹ We discuss this subroutine in detail at later. We then have to set an initial guess for the equilibrium interest rate. Guessing the equilibrium interest rate is much easier than guessing the equilibrium capital level. In fact it can be shown that the former has a natural upper bound related to the time discount factor. Specifically, if the interest rate r was greater or equal to the time preference rate $\theta = \frac{1}{\beta} - 1$, individuals would accumulate more and more assets over time. In the end this would result in an infinitely large asset supply A . Given our guess for the interest rate at 4 per cent, we consequently have to make sure that the time preference rate is larger than this. Finally, the main program calls the subroutine `solve_model`, which organizes the root-finding process needed to determine the equilibrium interest rate.

The subroutine `solve_model` first starts the timer and writes an output headline to the screen. We then set the tolerance level of the root-finding routine to the value `sig_out`. This can be done easily using the subroutine `settol_root` provided by the toolbox. The level of tolerance for this *outer root-finding process* to determine the interest rate is specified in the accompanying module `globals`.²⁰ For our purposes a tolerance level of 10^{-6} should suffice. We then copy our initial guess of the interest rate into a variable `x`, set `check` to the value `.false.` and call the subroutine `fzero` from the toolbox, which will determine the root of the function `asset_market`. Having determined the market-equilibrium interest rate correctly — which we can check for by looking at the return value of `check` — we stop the timer and write some model output to the console.

The function `asset_market` that is shown in Module 9.8m takes as input a level of the interest rate `r_input` and copies it to the variable `r`. Already knowing aggregate labour

¹⁹ Note that as in previous chapters, we again use variables with twice the same letter for variables that in our formulas are written with capital letters.

²⁰ Note that we will also have an *inner root-finding process* that we need to solve the household optimization problem.

Program 9.8 The heterogeneous agent model

```

program HetAg_Sim
[.....]
! initialize variables
call initialize()

! set initial guess of the interest rate
r = 0.040d0

! solve initial equilibrium
call solve_model()

contains

! For solving the model
subroutine solve_model

implicit none
real*8 :: x_in
logical :: check

! start timer
call tic()

write(*, '(a)')      K/Y          r          diff_K'

! set tolerance level for outer optimization
call settol_root(sig_out)

! set initial guess
x_in = r

! solve asset market equilibrium
call fzero(x_in, asset_market, check)

if(check)write(*,'(a)')'ERROR IN ROOTFINDING'

call toc

! write output to the screen
call output

end subroutine
[.....]
end program

```

supply LL, we can immediately calculate the corresponding capital stock KK and derive the wage rate w as well as aggregate production yy. Having determined all factor prices, we solve the household problem and aggregate individual decisions about savings to total asset supply AA, which will be discussed in Sections 9.4.3 and 9.4.4. Finally, the function asset_market returns the difference between asset supply of the households and capital demand of the firms. We calculate this difference in relative terms. In equilibrium, all assets supplied by the households must be employed as capital by the firms. Hence, the difference AA - KK should be equal to zero. As already argued above, when the capital and labour market clear, the goods market will also be in equilibrium.

Module 9.8m Capital market equilibrium equation

```

function asset_market(r_input)
    [.....]
    ! set the interest rate
    r = r_input

    ! calculate the corresponding capital stock
    KK = (alpha/(r+delta))**((1d0/(1d0-alpha))*LL

    ! get wages and output
    w = (1d0-alpha)*(KK/LL)**alpha
    YY = KK**alpha*LL**((1d0-alpha)

    ! solve the household value function iteration problem
    call solve_household()

    ! simulate an asset and consumption path
    call simulate_path()

    ! get aggregate assets
    AA = sum(a_t(0:TT))/dble(TT+1)

    ! get the difference between asset demand and supply
    asset_market = (AA - KK)/AA

    write(*,'(2f10.4,f12.8)')KK/YY*100d0, r*100d0, AA-KK

end function

```

9.4.3 DETERMINING HOUSEHOLD POLICY FUNCTIONS

Discretization of the state space We can solve the household optimization problem described in (9.19) using policy function iteration. In order to do so, we have to discretize the state space. We have already learned how to determine shock realizations $\hat{\eta}_g$ and a transition matrix $\hat{\pi}(\hat{\eta}_{g^+} | \hat{\eta}_g)$ using the Rouwenhorst method. Having done that the solution to the household problem simplifies to

$$c(a, \hat{\eta}_g) = \left[\beta(1+r) \sum_{g^+=1}^m \pi_{gg^+} \cdot c^+(a^+, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}} \right]^{-\gamma} \quad (9.20)$$

$$\text{with } a^+ = (1+r)a + w \cdot \exp(\hat{\eta}) - c, \quad a^+ \geq 0. \quad (9.21)$$

To discretize the asset state space, we choose gridpoints $\hat{a}_v, v = 0, 1, \dots, n$ on the interval $[a_l, a_u]$, where we set $a_l = 0$ because of the non-negativity constraint on assets. One feature that comes along with the borrowing constraint $a^+ \geq 0$ is that the policy function may become curved or kinky when the individual asset level a is close to 0. To deal with this issue and get the most accurate approximation of the policy function over the whole grid space, we do not use an equidistant grid as before. Instead we let the distance between grid points grow as we get further away from 0. Specifically we set

$$\hat{a}_v = a_l + (a_u - a_l) \cdot \frac{(1 + u_a)^v - 1}{(1 + u_a)^n - 1} \quad \text{for } v = 0, 1, \dots, n, \quad (9.22)$$

where u_a denotes the growth rate of the distances between two successive pairs of points. We can see that from calculating

$$\hat{a}_{v+1} - \hat{a}_v = \frac{(a_u - a_l) \cdot u_a}{(1 + u_a)^n - 1} \cdot (1 + u_a)^v$$

which directly leads to

$$\frac{\hat{a}_{v+2} - \hat{a}_{v+1}}{\hat{a}_{v+1} - \hat{a}_v} - 1 = u_a.$$

By using a higher u_a , more points will be allocated to the lower end of the grid. Figure 9.10 shows gridpoints between $a_l = 0$ and $a_u = 1$ with a growth rate of $u_a = 0.10$ and $n = 10$.

We organize the discretization of the state space in the subroutine `initialize` that is shown in Program 9.8.a. We first use the subroutine `discretize_AR` from the toolbox to determine shock levels `eta` as well as a transition matrix `pi`. In contrast to earlier calls of this subroutine, we add an additional argument `weights`, which is an array of the same length as `eta`. In this array the subroutine will store the unconditional stationary distribution of the shock process, see again Section 9.2.1. An element `weight(is)` of this stationary distribution tells us how many individuals at each point in time t will receive an idiosyncratic income shock of level `eta(is)`, when the income distribution is in its steady state. From this unconditional distribution, we can directly derive the steady-state level of aggregate labour supply `LL` by summing up the different shock levels `eta` weighted with the unconditional stationary distribution `weights`. Before we do this, however, we have to transform the levels of `eta` by applying the exponential function. To complete the discretization of the shock process, we draw a series of idiosyncratic productivity shocks using the subroutine `simulate_AR`. An array of grid points with growing distances can be created with the subroutine `grid_Cons_Grow` from the toolbox. This function receives as input arguments the left and right interval points of the grid a_l and a_u as well as the growth rate u_a . It fills up the array `a` with gridpoints similar to the ones pictured in Figure 9.10.

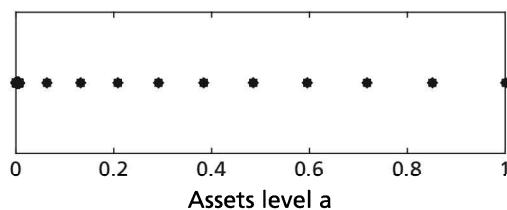


Figure 9.10 Growing asset grid ($a_l = 0$, $a_u = 1$, $n = 10$, $u_a = 0.10$)

Program 9.8.a Discretizing the state space

```

subroutine initialize()
[.....]
! calculate the shock process
call discretize_AR(rho, 0d0, sigma_eps, eta, pi, weights)
eta = exp(eta)

! calculate aggregate labour supply
LL = sum(eta*weights)

! simulate draws from the AR(1) process
call simulate_AR(pi, is_t)

! initialize grid
call grid_Cons_Grow(a, a_l, a_u, a_grow)

! initialize policy function
do is = 1, ns
    c(:, is) = 0.04d0*a(:) + eta(is)
    c_new(:, is) = c(:, is)
enddo

end subroutine

```

Last but not least, after having discretized the state space completely, we provide an initial guess for the consumption policy function of the household. By setting consumption equal to $0.04d0*a(:) + \eta(is)$, we assume that the interest rate is equal to 4 per cent and households always consume their entire labour income plus the interest income from savings. Such consumption behaviour would imply that today's and tomorrow's level of assets would be identical. Hence, aggregate private savings would be constant over time.

Computation of individual decision rules Given the discretized state space, we solve the household's policy function using policy function iteration as shown in Module 9.8m.a. This policy function iteration is organized in a subroutine `solve_household` which is called from the function `asset_market` in Module 9.8m. It follows the same logic as in previous programs. Before turning to the actual policy function iteration procedure, we adjust the tolerance level for the *inner root-finding procedure* to a level of `sig_in`. We choose a tolerance level of 10^{-10} here, leading to quite accurate solutions for the policy function. In general, the tolerance level for the inner root-finder should result in more precise solutions than that of the outer root-finding mechanism to guarantee convergence of the outer root-finding algorithm that is used to determine the market-clearing interest rate. Having adjusted the tolerance level, we can start our usual policy function iteration procedure.

In order to solve for an updated policy function, we first have to interpolate the right-hand side of the first-order condition (9.20). In the growth models discussed so far, we knew that the policy functions are curved and, more importantly, that they are differentiable at each point of the state space. Since our heterogeneous agent model features a potentially binding borrowing constraint at $a^+ = 0$, there is, however, no

Module 9.8m.a Policy function iteration in the heterogeneous agent model

```

subroutine solve_household()
    [.....]
    ! set tolerance level for interior optimization
    call settol_root(sig_in)

    ! do a policy function iteration with rootfinding
    do iter = 1, itermax

        ! interpolate coefficients
        call interpolate()

        ! solve the household problem for all gridpoints
        do ia = 0, NA
            do is = 1, NS

                ! initialize starting value/communication variables
                x_in = c(ia, is)
                a_com = a(ia)
                is_com = is

                ! find the optimal consumption level
                call fzero(x_in, foc, check)
                if(check)write(*,'ERROR IN ROOTFINDING PROCESS'

                ! check for borrowing constraint
                if (x_in > (1d0+rn)*a_com+wn*eta(is_com)-a_l) &
                    x_in = (1d0+rn)*a_com+wn*eta(is_com)-a_l

                ! get optimal consumption and value function
                c_new(ia, is) = x_in

            enddo
        enddo

        ! get convergence level
        con_lev = maxval(abs(c_new(:, :)) - c(:, :)) &
                   /max(abs(c(:, :)), 1d-10))

        ! check for convergence
        if(con_lev < sig_in)then
            return
        endif

        c = c_new
    enddo

    write(*,'Policy Function Iteration did not converge'

end subroutine

```

guarantee that policy functions are still differentiable in the present setup. In fact, they might exhibit a kink at the asset level a at which the borrowing constraint starts binding. Hence, interpolating the right-hand side of the first-order condition with a two-times differentiable spline function is probably not the wisest choice. Instead, we will use piecewise liner interpolation to perform this interpolation task. As in previous programs, we do not interpolate each future policy function individually and then calculate expectations about

future marginal utilities of consumption, but directly interpolate the complete right-hand side of the first-order condition (9.20). Consequently, the subroutine `interpolate` stores in an array `RHS` the value of this side for each potential level of future savings a^+ and today's level of shocks $\hat{\eta}_g$.

Afterwards, we can solve for the solution to the *unrestricted* first-order condition for each current level of savings `a(is)` and each idiosyncratic income shock level `is`, applying a standard root-finding method to the function `foc` which is shown in Module 9.8m.b. We call this the unrestricted first-order condition, since at this stage we do not take into account the borrowing constraint of $a^+ \geq 0$. Instead, we also allow for negative savings levels in the first-order condition and manually restrict savings to non-negative levels later on. The function `foc` receives as input argument a level of consumption `x_in`. From this consumption level as well as the level of current assets `a_com` and the current idiosyncratic shock to labour income `eta(is_com)` that we communicated from the subroutine `solve_household`, we can calculate the future level of savings `aplus`. With this, we can linearly interpolate the right-hand side of the first-order condition by means of the subroutine `linint_Grow`. This subroutine works in exactly the same way as the routine `linint_Equi` we used in Program 2.16, but only on a grid with growing distances. Having determined the left and right gridpoint for interpolation as well as the interpolation share `phi`, it is easy to calculate the right-hand side of the first-order condition using the array `RHS`. Finally, the function `foc` returns the difference between the right-hand side and current consumption `x_in`, which the root-finder should set to zero.

Once we have determined the solution to the unrestricted first-order condition, we have to check whether the solution satisfies the constraint that assets cannot be smaller than `a_l`, which in our case means they cannot be negative. This is done in the next step in the subroutine `solve_household`. The borrowing limit would be violated once future savings decrease below a value of `a_l`, meaning that we have

$$a^+ = (1 + r)\hat{a}_v + w \cdot \exp(\hat{\eta}_g) - c < a_l \quad \Leftrightarrow \quad c > (1 + r)\hat{a}_v + w \cdot \exp(\hat{\eta}_g) - a_l.$$

Module 9.8m.b Unrestricted first-order condition of the household problem

```

function foc(x_in)
    [.....]
    ! future assets
    aplus = (1d0+r)*a_com + w*eta(is_com) - x_in

    ! calculate future expected marginal utility
    call linint_Grow(aplus, a_l, a_u, a_grow, NA, ial, iar, varphi)
    foc = varphi*RHS(ial,is_com) + (1d0-varphi)*RHS(iar,is_com)

    ! get first-order condition
    foc = x_in - foc

end function

```

If this condition holds, we adjust current consumption—in this case stored in the variable `x_in`—so that the constraint on savings holds exactly with equality. Finally, we store the updated value of the consumption policy function at gridpoints `ia` and `is` in the array `c_new` and continue with the usual steps of a policy function iteration. We repeat the above steps until the policy function has converged sufficiently, that is, until the absolute of the relative difference of two successive iterations is smaller than `sig_in`.

9.4.4 AGGREGATION OF INDIVIDUAL DECISIONS

The last remaining step consists of transforming the individual decision rules, namely the policy function for consumption and the budget constraint, into an aggregate supply A of savings and an aggregate level of consumption C . In the representative agent economy it was easy to determine these aggregate quantities since there is only one (representative) agent at each point in time. Aggregate consumption therefore equals this agent's consumption and the same for all the other quantities. In the heterogeneous agent economy, however, we have to find a way to derive aggregate quantities from individual choices. There are essentially two approaches to do this. On the one hand, we can draw a series of idiosyncratic productivity shocks and simulate the individual model forward in the same way as we did it in the stochastic growth or the RBC model. We can then determine aggregate values and distributions from the simulated path. On the other hand, we can also compute the distribution of households over the state space directly. In the following we will explain both approaches in more detail.

Forward simulation The first approach is implemented in Program 9.8 and shown in Module 9.8m.c. To simulate the individual model, we make use of the sequence of shocks we had already drawn in the subroutine `initialize` and stored in the array `is_t`. Note that we use the same sequence of shocks over and over again when we iterate to find the equilibrium quantities and prices. The forward simulation process starts at some period, e.g. `t=-5000`, with an initial value for private wealth `a_t(-5000)` as well as the initial shock level `eta(is_t(-5000))`. We use 5,000 leading periods in this simulation that we will not use to calculate aggregate quantities. These periods are employed to get rid of the *initialization effect*, namely the fact that we choose an arbitrary level of initial wealth here.²¹ We then simulate the model forward until some final period `TT` using the policy function for consumption as well as the budget constraint in the same way as we did in the stochastic growth model. Note that we use the same piecewise linear interpolation technique for the consumption policy as we have done for the right-hand side of the first-order condition. Using the simulated path for individual wealth `a_t` and

²¹ If we choose a reasonable value for initial wealth, we can also use fewer leading periods as the initialization is not very strong for `a_t(-5000)`.

Module 9.8m.c Simulating paths in the heterogeneous agent model

```

subroutine simulate_path()
    [...]
    ! initialize variables
    eta_t(-5000) = eta(is_t(-5000))
    a_t(-5000) = 1d0
    call linint_Grow(a_t(-5000), a_l, a_u, a_grow, &
                       NA, ial, iar, varphi)
    c_t(-5000) = varphi*c(ial, is_t(-5000)) &
                  + (1d0-varphi)*c(iar, is_t(-5000))

    ! calculate the time path of all economic variables
    do it = -4999, TT
        eta_t(it) = eta(is_t(it))
        a_t(it) = min(w*eta_t(it-1) + (1d0+r)*a_t(it-1) &
                      - c_t(it-1), a_u)
        call linint_Grow(a_t(it), a_l, a_u, a_grow, &
                           NA, ial, iar, varphi)
        c_t(it) = varphi*c(ial, is_t(it)) &
                  + (1d0-varphi)*c(iar, is_t(it))
    enddo

end subroutine

```

consumption c_t , we can now calculate aggregate quantities. We thereby make use of the fact that the economy is in a steady state. Because of this we can interpret our realizations of $a_t(0:TT)$ and $c_t(0:TT)$ not only as realizations of a time series $\{a_{i,t}\}_{t=0,T}$ and $\{c_{i,t}\}_{t=0,T}$ for one individual i , but also as realizations $\{a_{i,t}\}_{i=0,T}$ and $\{c_{i,t}\}_{i=0,T}$ of $TT+1$ different individuals i that live at the same time t .²² Consequently we can calculate aggregate wealth and consumption in the steady state as

$$A = \frac{1}{T+1} \cdot \sum_{i=0}^T a_{i,t} \quad \text{and} \quad C = \frac{1}{T+1} \cdot \sum_{i=0}^T c_{i,t},$$

respectively. For aggregate wealth, this is done in the function `asset_market` in Module 9.8m. Aggregate consumption, on the other hand, is derived only in the subroutine `output`.

Direct computation of the household distribution One downside of the computational algorithm described above is that the presented results rely mainly on a series of shocks that are drawn before the simulation. It then uses the law of large numbers which tells us that only if the number of simulated shocks is large enough will the quantities computed by the simulation converge to give us the true quantities. However, while the law of large numbers ensures convergence, the convergence speed is usually quite slow, meaning that it takes a huge amount of simulated data points to obtain accurate

²² Note that this property no longer holds once the economy is on a transition path, i.e. once prices r_t and w_t vary over time.

solutions. As a result, when we start the same program over and over again, always drawing a new series of shocks, the quantities and prices we compute will exhibit quite some variation.²³ There are two ways around this issue. One way would be to heavily increase the number of simulation periods until we can no longer see any difference in the results of different simulations. The other way would be to calculate the stationary distribution of households over the state space directly. The idea behind this is fairly simple. What we are looking for is a distribution²⁴

$$\phi(\hat{a}_v, \hat{\eta}_g) \quad \text{for } v = 0, 1, \dots, n \quad \text{and} \quad g = 1, 2, \dots, m$$

with the property that

$$\sum_{v=0}^n \sum_{g=1}^m \phi(\hat{a}_v, \hat{\eta}_g) = 1.$$

From this distribution we want to be able to calculate the quantities of the economy as

$$A = \sum_{v=0}^n \sum_{g=1}^m \phi(\hat{a}_v, \hat{\eta}_g) \cdot \hat{a}_v \quad \text{and} \quad C = \sum_{v=0}^n \sum_{g=1}^m \phi(\hat{a}_v, \hat{\eta}_g) \cdot c(\hat{a}_v, \hat{\eta}_g). \quad (9.23)$$

With the economy in a steady state this distribution will be a stationary object.

We use an iterative algorithm to calculate the distribution ϕ . This algorithm is based on the piecewise linear interpolation scheme we used to interpolate the right-hand side of the first-order condition. Specifically, we proceed as follows:

1. First, we provide some (arbitrary) initial guess for the distribution function ϕ . In the subroutine `initialize` we set

$$\phi(\hat{a}_v, \hat{\eta}_g) = \frac{1}{(n+1) \cdot m} \quad \text{for all } v = 0, \dots, n \text{ and } g = 1, \dots, m.$$

Hence, we assume that at each asset grid point and shock combination there is exactly the same mass of households.

2. Given some current estimate ϕ of the distribution function, we calculate an update ϕ_{new} of this distribution as follows: For each value of the asset grid and each shock level $(\hat{a}_v, \hat{\eta}_g)$ we determine where an agent with these characteristics will end up in the next period. For the shocks values $\hat{\eta}_{g+}$ this is fairly easy as the transition matrix $\hat{\pi}$

²³ Try this by running the program a couple of times and comparing the outcomes.

²⁴ We are quite sloppy in defining the distribution of households in this book as we want to keep things as accessible and intuitive as possible. For a more analytical discussion of this topic we refer the reader to one of the standard advanced macroeconomics textbooks.

tells us exactly with which probability the agent with shock $\hat{\eta}_g$ ends up in any future shock level. We then just assume that this agent splits up into m different agents and attach the transition probabilities π_{gg^+} to each of these agents. For the next period's savings we use the policy function together with the budget constraint to get

$$a^+ = (1 + r)\hat{a}_v + w \cdot \exp(\hat{\eta}_g) - c(\hat{a}_v, \hat{\eta}_g).$$

This level of savings typically does not lie directly on the asset grid but is located between two grid points. The problem is that our distribution function ϕ —and therefore also ϕ_{new} —is an object that exists only exactly on the asset grid. Hence, we have to somehow map the decision a^+ back onto the discrete grid $\{\hat{a}_v\}_{v=1}^n$. This is where our linear interpolation scheme is needed. Let us assume, without loss of generality, that the gridpoints in-between which a^+ is located are \hat{a}_{v+3} and \hat{a}_{v+4} . We then calculate the share φ such that

$$a^+ = \varphi \cdot \hat{a}_{v+3} + (1 - \varphi) \hat{a}_{v+4}.$$

Once we have calculated this share, we assume that the agent splits up into two and map a fraction φ on the left gridpoint and the remaining fraction on the right gridpoint. This means that we calculate

$$\begin{aligned}\phi_{new}(\hat{a}_{v+3}, \hat{\eta}_g^+) &= \phi_{new}(\hat{a}_{v+3}, \hat{\eta}_g^+) + \pi_{gg^+} \cdot \varphi \cdot \phi(\hat{a}_v, \hat{\eta}_g) \\ \phi_{new}(\hat{a}_{v+4}, \hat{\eta}_g^+) &= \phi_{new}(\hat{a}_{v+4}, \hat{\eta}_g^+) + \pi_{gg^+} \cdot (1 - \varphi) \cdot \phi(\hat{a}_v, \hat{\eta}_g)\end{aligned}$$

for all realizations of the next period's productivity $\hat{\eta}_g^+, g^+ = 1, \dots, m$. Figure 9.11 visualizes this approach.

3. We now check whether the maximum relative difference between the old and the new distribution functions is below a certain tolerance level. If this is the case then the distribution is obviously invariant, meaning that we have found our solution and can stop the iteration procedure. If this is not the case, we update our guess of ϕ using ϕ_{new} and go back to step 2.

The algorithm to calculate the invariant distribution of households is shown in Module 9.9m. The subroutine `get_distribution` replaces the routine `simulate_path`

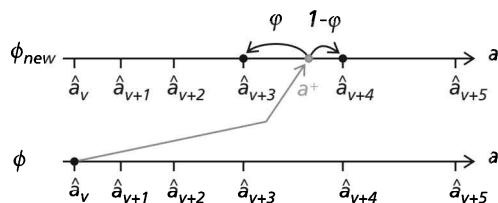


Figure 9.11 Linear interpolation between two grid points

Module 9.9m Direct computation of the distribution of households

```

subroutine get_distribution()
    [.....]
    ! get the interpolation shares and points
    do ia = 0, NA
        do is = 1, NS

            ! calculate where this guy would go
            aplus = (1d0+r)*a(ia) + w*eta(is) - c(ia,is)
            aplus = min(max(aplus, a_l), a_u)

            ! determine the gridpoints in between this decision lies
            call linint_Grow(aplus, a_l, a_u, a_grow, NA, &
                ial(ia, is), iar(ia, is), varphi(ia, is))
        enddo
    enddo

    ! iterate until the distribution function converges
    do iter = 1, itermax

        phi_new = 0d0

        do ia = 0, NA
            do is = 1, NS

                do is_p = 1, NS
                    phi_new(ial(ia, is), is_p) = &
                        phi_new(ial(ia, is), is_p) &
                        + pi(is, is_p)*varphi(ia, is)*phi(ia, is)
                    phi_new(iar(ia, is), is_p) = &
                        phi_new(iar(ia, is), is_p) &
                        + pi(is, is_p)*(1d0-varphi(ia, is))*phi(ia, is)
                enddo
            enddo
        enddo

        con_lev = maxval(abs(phi_new(:, :) - phi(:, :)) &
            /max(abs(phi(:, :)), 1d-10))

        ! update distribution
        phi = phi_new

        ! check for convergence
        if(con_lev < sig_in)then
            return
        endif
    enddo

    write(*,'Distribution Function Iteration did not converge')

end subroutine

```

in the function `asset_market`. Aggregate savings supply can then be calculated from (9.23). Note that we can write

$$A = \sum_{v=0}^n \hat{a}_v \cdot \sum_{g=1}^m \phi(\hat{a}_v, \hat{\eta}_g),$$

since the asset grid space and the productivity shocks are orthogonal by construction. Using the `sum` function we consequently get

$$\text{AA} = \text{sum}(a * \text{sum}(\phi, 2)).$$

There are two things to note here: First, the subroutine `initialize` is only called up once before we start the outer root-finding procedure to determine the equilibrium interest rate. Consequently our initial guess will only be used the first time we enter the subroutine `get_distribution`, meaning also the first time the function `asset_market` is evaluated. In all subsequent calls of `get_distribution`, we just use the distribution function `phi` we obtained from the previous call as a starting point. Second, throughout the distribution function iteration procedure the policy function $c(\hat{a}_v, \hat{\eta}_g)$ and therefore the future asset level to which an agent at state $(\hat{a}_v, \hat{\eta}_g)$ moves remain unchanged. For the efficiency of the algorithm, it is therefore useful to calculate the left and right gridpoints as well as the interpolation shares φ only once before the iteration starts. This is done by means of the subroutine `linint_Grow`. We thereby ensure that tomorrow's level of savings will stay within the bounds of our specified asset grid. We then store the results of the interpolation procedure in arrays `ial(ia, is)`, `iar(ia, is)`, and `varphi(ia, is)`, to which we revert in each iteration step of the distribution function iteration.

9.4.5 MODEL PARAMETRIZATION AND SIMULATION

To simulate the model we have to specify both economic and numerical parameters. The economic parameters we choose are fairly standard. We let $\gamma = 0.5$, $\alpha = 0.36$, $\beta = 0.96$, $\delta = 0.08$. An intertemporal elasticity of substitution of 0.5 implies an individual relative risk-aversion of 2. Individuals discount future consumption streams by $\frac{1}{\beta} - 1 \approx 4.166$ per cent. Note that this choice of β ensures that the time preference rate is greater than our initial guess of the interest rate of 0.04. An autocorrelation of $\rho = 0.6$ leads to a moderate persistence of idiosyncratic shocks. Finally, we set the variance of the innovation term at $\sigma_\epsilon^2 = 0.04 \cdot (1 - \rho^2)$, which causes the overall variance of the process η to equal $\frac{\sigma_\epsilon^2}{1 - \rho^2} = 0.04$. For the choice of numerical parameters we use $n = 1,000$ points to discretize the asset state space. This should suffice to get an accurate approximation of the household's policy function. The asset minimum of $a_l = 0$ reflects the borrowing constraint while we chose an asset maximum of $a_u = 100$ to ensure that no individual (under normal circumstances) will ever reach it. The growth rate of the asset grid is set at $u_a = 0.01$, which means that the distance between the first and the second gridpoint on the wealth grid is in the order of 10^{-5} . We use a number of $m = 7$ shock levels, which delivers accurate enough approximations for our purposes. Finally, the level of convergence for the policy and distribution function iteration process is chosen as 10^{-10} , which is the same accuracy we specified for the inner root-finding procedure.

Table 9.6 The baseline heterogeneous agent economy

| Aggregate quantities and prices: | | |
|----------------------------------|----------------------------|--------|
| Capital-to-output ratio (in %) | K/Y | 299.72 |
| Interest rate (in %) | r | 4.01 |
| Consumption (in % of output) | C/Y | 76.02 |
| Investment (in % of output) | I/Y | 23.98 |
| Distributional measures: | | |
| CV of labour earnings (in %) | $\text{Std}(\exp(\eta))/L$ | 20.12 |
| CV of consumption (in %) | $\text{Std}(c)/C$ | 13.89 |
| CV of wealth (in %) | $\text{Std}(a)/A$ | 75.62 |

Table 9.6 shows the resulting steady state of the heterogeneous agent model. First of all, capital amounts to roughly three times GDP, which results in an equilibrium interest rate of $r = 4.01$ per cent which is only slightly lower than the time preference rate of 4.166 per cent. Consumption and investment add up to total output, while consumption makes up the much greater proportion of GDP. In addition to these aggregate quantities and prices, we can also compute some distributional measures on the individual level. An easily interpretable number is again the coefficient of variation (CV), that is the standard deviation of a variable normalized by its mean. Labour earnings exhibit a CV of 20.12 per cent, meaning that a one standard deviation fluctuation increases or decreases labour earnings by 20 per cent. The CV of consumption is much smaller than that of labour income. This is because households use private wealth as a means of insuring themselves against income fluctuations. Building up precautionary savings, however, only provides partial insurance against consumption fluctuations, so that a one standard deviation decline in consumption still amounts to around 13 per cent. Finally, the CV of wealth is by far the largest. This is a result of households having built up a larger stock of wealth when they had a series of positive shocks and then consuming out of their wealth stock when they experience a series of negative shocks. Note that we calculate the CV of a variable using the household distribution as shown in Program 9.9.

In addition to calculating such statistics, we can plot the distribution of households as a function of private wealth. The left panel of Figure 9.12 depicts the distribution of wealth in the heterogeneous agent economy with the above parametrization. In order to get a good picture of the wealth distribution, we only plot it on the interval from 0 to 30. Note that the distribution has a spike at a wealth level of 0 which indicates the number households for which the borrowing constraint is actually binding. Afterwards it evolves fairly smoothly (except for some minor fluctuations that are due to computational accuracy). The distribution then exhibits another peak at a wealth level of roughly 6 and afterwards slowly fades out. The right panel of Figure 9.12 shows the amount of savings a^+ for tomorrow as a function of today's stock of wealth. We plot these functions for different levels of the idiosyncratic productivity shock $\hat{\eta}_g$. The black line consequently indicates the savings policy for the household with the lowest

Program 9.9 Using the distribution function to create output

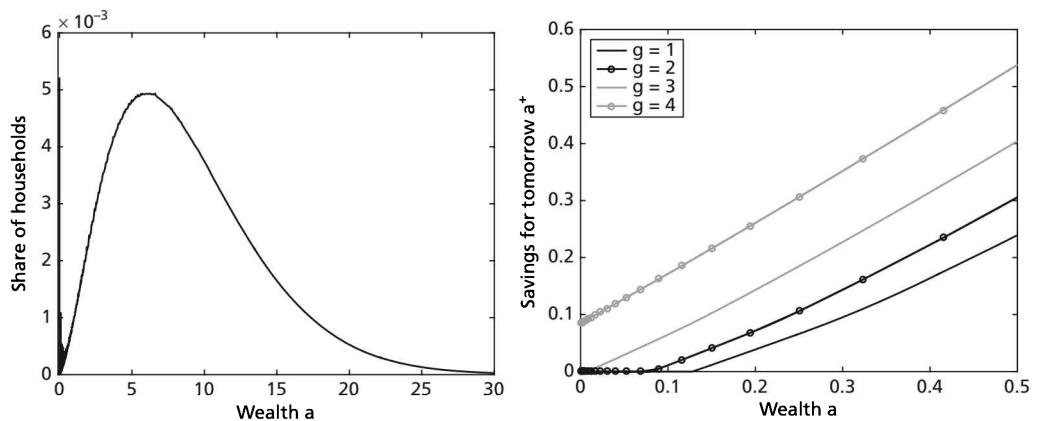
```

subroutine output()
    [.....]
    ! calculate coefficients of variation
    Std_eta = 0d0
    Std_c   = 0d0
    Std_a   = 0d0
    do ia = 0, NA
        do is = 1, NS
            Std_eta = Std_eta + phi(ia, is)*(eta(is) - LL)**2
            Std_c   = Std_c   + phi(ia, is)*(c(ia, is) - CC)**2
            Std_a   = Std_a   + phi(ia, is)*(a(ia) - KK)**2
        enddo
    enddo
    Std_eta = sqrt(Std_eta)/LL
    Std_c   = sqrt(Std_c) /CC
    Std_a   = sqrt(Std_a) /AA
    [.....]

    ! plot the distribution of wealth
    call plot(a, sum(phi, 2))
    call execplot(xlim=(/a_1, 30d0/), title='Distribution of Wealth')

    ! plot savings decisions
    do is = 1, 4
        call plot(a, (1+r)*a+w*eta(is)-c(:, is))
    enddo
    call execplot(xlim=(/a_1, 0.5d0/), title='Policy Functions')
end subroutine

```

**Figure 9.12** The heterogeneous agent model

productivity level. In this plot we can see the effect of the borrowing constraint on private savings. An individual with low labour productivity and a low stock of wealth would actually want to run into debt. Since in expectation of this she will move to higher idiosyncratic income levels in the future, this would be the ideal strategy to smooth consumption over time. However, the existing borrowing constraint prevents

her from accumulating such debt. Hence, the asset policy function (and therefore also the consumption policy) exhibits a kink at the point where it hits the level of $a^+ = 0$ (coming from the right). Individuals that face a binding borrowing constraint are often called *hand-to-mouth consumers*, since they just consume all the income from labour and savings they have available in a given period t . The higher the individual productivity level $\hat{\eta}_g$ is, the lower the chance that an agent runs into the borrowing constraint. The reason is that in times of higher income, households save a lot to build up a buffer stock that they can use to smooth consumption over any future periods with low income.

Self-insurance, overaccumulation, and the interest rate One of the many questions one can analyse using the heterogeneous agent model is how effectively households can self-insure against idiosyncratic income risk using private savings under different assumptions about the stochastic income process. We therefore exploit the property of the AR(1) process that $\sigma_\eta^2 = \frac{\sigma_\epsilon^2}{1-\rho^2}$. Consequently, if we let $\sigma_\epsilon^2 = \sigma_\eta^2 \cdot (1 - \rho^2)$ and vary ρ , the overall variance of labour earnings will stay constant. We simulate the model for different levels of both ρ and σ_η . The resulting coefficient of variation in consumption and the interest rate are shown in Table 9.7. First of all we find that with the overall standard deviation of the stochastic process being constant, the ability of households to insure themselves against idiosyncratic shocks decreases significantly with a higher autocorrelation. This is quite intuitive, since with a high autocorrelation individuals will experience the same good or bad income shock for a longer time period. Therefore it is quite likely that for unlucky individuals the borrowing constraint will be binding more frequently. If the borrowing constraint is binding, then the individuals become hand-to-mouth consumers and are not able to smooth any risk at all. On the other hand, in good times when their labour productivity is high, individuals will save more for precautionary reasons so that the interest rate decreases with a higher autocorrelation. A larger variance of the stochastic process increases the need for self-insurance, as with the variance of earnings the variance of consumption also increases. Therefore, given a constant ρ , the need for precautionary savings rises with σ_η^2 , which in turn decreases the interest rate further. This effect is especially pronounced for large values of the autocorrelation parameter.

Table 9.7 Self insurance in the heterogeneous agent model

| | CV of consumption | | Interest rate | |
|--------------|------------------------|------------------------|------------------------|------------------------|
| | $\sigma_\eta^2 = 0.04$ | $\sigma_\eta^2 = 0.16$ | $\sigma_\eta^2 = 0.04$ | $\sigma_\eta^2 = 0.16$ |
| $\rho = 0.0$ | 12.58 | 12.91 | 4.12 | 3.96 |
| $\rho = 0.3$ | 12.79 | 14.55 | 4.09 | 3.79 |
| $\rho = 0.6$ | 13.89 | 18.15 | 4.01 | 3.47 |
| $\rho = 0.9$ | 19.72 | 29.77 | 3.81 | 2.88 |

9.4.6 THE OPTIMUM QUANTITY OF DEBT

The heterogeneous agent model is a workhorse model for the analysis of government policy. While in the RBC model the major point of interest was the cyclical properties of government policy and whether policy stabilizes business-cycle fluctuations, policy analysis in the heterogeneous agent model is concerned more with questions of redistribution between different households and the provision of insurance against idiosyncratic income shocks. In the following our analysis will focus on determining the optimum quantity of public debt. We already saw in Table 9.1 that in the standard neoclassical growth model public debt follows the notion of Ricardian equivalence. This means that in the long run, any level of debt leads to the same capital-to-output ratio and the same household utility since households clearly foresee the additional tax burden that accompanies a higher stock of government debt and increase their savings accordingly. This changes in the heterogeneous agent model. To investigate the effects of public debt in our model, we have to make two adjustments: we introduce (exogenous) productivity growth and model a government sector that raises taxes on labour and capital income to finance public expenditure and to repay interest payments on public debt.

A model with exogenous productivity growth Let us assume that the production technology of firms was given by

$$Y_t = K_t^\alpha \cdot (\Omega_t \cdot L_t)^{1-\alpha},$$

where Ω_t now measures the productivity of labour input and not the aggregate technology level (or total factor productivity). We normalize $\Omega_0 = 1$ and assume that aggregate labour productivity grows at the constant rate ϖ , so that $\Omega_t = (1 + \varpi)^t$. We can then write the firms' demand equations as

$$r_t = \alpha \cdot \left[\frac{K_t}{\Omega_t L_t} \right]^{\alpha-1} - \delta \quad \text{and} \quad \frac{w_t}{\Omega_t} = (1 - \alpha) \cdot \left[\frac{K_t}{\Omega_t L_t} \right]^\alpha.$$

We assume that the specification of the household sector of the economy remains unchanged, meaning that labour supply is inelastic and individual labour productivity $\exp(\eta)$ has a stable mean. Hence, L_t is constant over time and $\Omega_t L_t = (1 + \varpi)^t L$ grows at rate ϖ . In the following we restrict ourselves to looking at so-called *balanced growth paths* of the economy. A balanced growth path is characterized by the fact that all aggregate and individual variables grow at the same rate. Since total labour input $\Omega_t L_t$ grows at rate ϖ , all other variables consequently need to grow at the very same rate. Now let us define normalized variables

$$\tilde{x}_t = \frac{x_t}{\Omega_t} = \frac{x_t}{(1 + \varpi)^t}.$$

As in a balanced growth path all aggregate and individual variables grow at the same rate, their normalized counterparts \tilde{x}_t have to be constant over time. Looking again at the above demand equations, we find that in the balanced growth path they can be written as

$$r = \alpha \cdot \left[\frac{\tilde{K}}{L} \right]^{\alpha-1} - \delta \quad \text{and} \quad \tilde{w} = (1 - \alpha) \cdot \left[\frac{\tilde{K}}{L} \right]^\alpha. \quad (9.24)$$

Consequently wages grow at the rate ϖ , while the interest rate r is constant over time.²⁵

Households again maximize expected utility

$$E_0 \left[\sum_{t=0}^{\infty} \beta^t \cdot \frac{(c_{i,t})^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right]$$

Recognizing that we have $x_t = (1 + \varpi)^t \tilde{x}_t$, we can rewrite the utility function as

$$E_0 \left[\sum_{t=0}^{\infty} \beta^t \cdot \frac{((1 + \varpi)^t \tilde{c}_{i,t})^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right] = E_0 \left[\sum_{t=0}^{\infty} \hat{\beta}^t \cdot \frac{(\tilde{c}_{i,t})^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right]$$

with $\hat{\beta} = \beta(1 + \varpi)^{1-\frac{1}{\gamma}}$. The dynamic budget constraint of the household reads

$$(1 + r_t^n) a_{i,t} + w_t^n \cdot \exp(\eta_{i,t}) = a_{i,t+1} + c_{i,t}$$

with an AR(1) process for η_t . $r_t^n = (1 - \tau_r)r_t$ and $w_t^n = (1 - \tau_w)w_t$ denote net prices after taxation by the government. We can transform this budget constraint to

$$(1 + r_t^n)(1 + \varpi)^t \tilde{a}_{i,t} + (1 + \varpi)^t \tilde{w}_t^n \cdot \exp(\eta_{i,t}) = (1 + \varpi)^{t+1} \tilde{a}_{i,t+1} + (1 + \varpi)^t \tilde{c}_{i,t}$$

and by dividing through $(1 + \varpi)^t$ we get

$$(1 + r_t^n) \tilde{a}_{i,t} + \tilde{w}_t^n \cdot \exp(\eta_{i,t}) = (1 + \varpi) \tilde{a}_{i,t+1} + \tilde{c}_{i,t}.$$

Summing up, we can solve for the balanced growth path of the economy by solving the dynamic programming problem

²⁵ This is quite an intuitive result. Since labour input L as measured by the sum of all labour hours weighted by individual labour productivity is constant over time, labour input gets rewarded at growing rates. Capital on the other hand grows at rate ϖ . Hence, the return on it is constant.

$$\begin{aligned}\tilde{V}(\tilde{a}, \eta) = \max_{\tilde{c}, \tilde{a}^+} & u(\tilde{c}) + \hat{\beta} E[\tilde{V}(\tilde{a}^+, \eta^+) | \eta] \\ \text{s.t. } & (1 + r^n)\tilde{a} + \tilde{w}^n \cdot \exp(\eta) = \tilde{c} + (1 + \varpi)\tilde{a}^+, \quad \tilde{a}^+ \geq 0 \\ \text{and } & \eta^+ = \rho\eta + \epsilon^+ \quad \text{with } \epsilon^+ \sim N(0, \sigma_\epsilon^2).\end{aligned}$$

with $\hat{\beta} = \beta(1 + \varpi)^{1 - \frac{1}{r}}$ and prices as in (9.24). The policy function $\tilde{c}(\tilde{a}, \eta)$ as well as the value function $\tilde{V}(\tilde{a}, \eta)$ are then constant over time.

The government sector Last but not least, we have to define the policy measures of the government sector. The government can tax labour and capital income at rates τ_w and τ_r . It does so to finance some government expenditure $G_t = g_y \cdot Y_t$, which we assume to be a fixed fraction g_y of aggregate output. In addition, it can issue public debt $B_t = b_y \cdot Y_t$, which is a constant fraction of GDP as well. The budget constraint of the government reads

$$\tau_w w_t L_t + \tau_r r_t A_t + B_{t+1} = G_t + (1 + r_t)B_t$$

or in normalized terms

$$\tau_w \tilde{w}_t L_t + \tau_r r_t \tilde{A}_t + (1 + \varpi) \tilde{B}_{t+1} = \tilde{G}_t + (1 + r_t) \tilde{B}_t.$$

We let the capital-income tax rate balance the government's budget. In the balanced growth path, the budget-clearing tax rate can be calculated from

$$\tau_r = \frac{\tilde{G} + (r - \varpi)\tilde{B} - \tau_w \tilde{w} L}{r \tilde{A}}.$$

Solving the model Module 9.10m shows how we have to adapt the function `asset_market` in order to solve the model with growth and government activity. Again, we take a guess `r_input` of the interest rate and calculate the aggregate capital stock as well as the wage rate and aggregate output from it. Knowing GDP, we can calculate government expenditure as well as the public debt level. Note that like in Section 9.1.3 the government competes with firms for the savings of the private households so that the asset market equilibrium condition reads

$$A_t = K_t + B_t.$$

Knowing both the aggregate capital stock as well as public debt holdings, we can determine aggregate savings from this equation. Last but not least, we calculate the budget-balancing tax rate on interest income and set net prices accordingly. The remainder of the subroutine remains untouched.

Module 9.10m A model with growth and government activity

```

function asset_market(r_input)
    [.....]
    ! set the interest rate
    r = r_input

    ! calculate the corresponding capital stock
    KK = (alpha/(r+delta))** (1d0/(1d0-alpha))*LL

    ! get wages and output
    w = (1d0-alpha)*(KK/LL)**alpha
    YY = KK**alpha*LL** (1d0-alpha)

    ! calculate government quantities
    GG = gy*YY
    BB = by*YY

    ! back out aggregate asset holdings
    AA = KK + BB

    ! get budget balancing tax rates
    tau_r = (GG + (r-growth)*BB - tau_w*w*LL) / (r*AA)

    ! define net prices
    rn = r*(1d0-tau_r)
    wn = w*(1d0-tau_w)
    [.....]
end function

```

Welfare effects of debt adjustments In order to derive the optimum quantity of debt, we need a criterion to maximize. We use a utilitarian social welfare criterion

$$W^*(b_y) = \sum_{v=0}^n \sum_{g=1}^m \phi(\hat{a}_v, \hat{\eta}_g) \cdot V(\hat{a}_v, \hat{\eta}_g),$$

which weighs the utility of all agents living with their respective distribution, given a certain debt to GDP ratio b_y .²⁶ We compare the utility level $W^*(b_y)$ to the utility level $W_0 = W^*(0.6)$ on a balanced growth path with a debt share in GDP of 60 per cent. To interpret the resulting numbers in a meaningful way, we again use the concepts of Hicksian equivalent variation in the same way as in Chapter 6. We consequently calculate

$$\Delta(b_y) = \left[\frac{W^*(b_y)}{W_0} \right]^{\frac{1}{1-\gamma}} - 1,$$

which indicates by how much individual consumption c has to increase for all households—regardless of their productivity and private wealth—to increase (or decrease) utilitarian welfare from W_0 to $W^*(b_y)$.

²⁶ This welfare measure is often also called *ex ante* utility as it measures the expected utility of a household just before the household enters the economy and before any information about its future productivity levels is revealed.

Calculating ex ante welfare involves deriving the households' value function $V(a, \eta)$, which up to this point was not necessary, as we use policy function iteration to solve the household optimization problem. To keep things as efficient as possible, we derive the value function after we have solved the model completely and obtained the equilibrium policy function $\tilde{c}(\tilde{a}, \eta)$. The calculation of the value function is managed in the subroutine `value_function` in the main program, and follows the same iterative principle as the computation of the distribution of households. Program 9.10 shows an excerpt of this routine. We first derive the linear interpolation points and weights and store them in arrays `ial`, `iar`, and `varphi`. This is not shown here again. We then start with a guess of the value function. Note that we do not calculate V directly but its transformation $\left[(1 - \frac{1}{\gamma})V\right]^{\frac{1}{1-\gamma}}$, for the same reasons we have already discussed in this chapter and in

Program 9.10 Iterative calculation of the value function

```

subroutine value_function()
[.....]
! initialize value function
V = c

! iterate until the value function converges
do iter = 1, itermax

  ! calculate new value function
  do ia = 0, NA
    do is = 1, NS

      ! interpolate over all future states
      V_new(ia, is) = 0d0
      do is_p = 1, NS
        V_new(ia, is) = V_new(ia, is) + pi(is, is_p) &
          *(varphi(ia, is)*V(ial(ia, is), is_p) + &
          (1d0-varphi(ia, is))*V(iar(ia, is), is_p))**egam
      enddo

      ! add current utility
      V_new(ia, is) = (c(ia, is)**egam &
        + beta*V_new(ia, is))** (1d0/egam)
    enddo
  enddo

  con_lev = maxval(abs(V_new(:, :) - V(:, :))) &
    /max(abs(V(:, :)), 1d-10))

  ! update value function
  V = V_new

  ! check for convergence
  if(con_lev < sig_in)then
    V = V**egam/egam
    return
  endif
  endif
enddo

end subroutine
  
```

Chapter 8. A solid initial guess for this transformation is $v = c$. We then update the value function at each (discretized) point in the state space by using the equilibrium policy function to determine today's instantaneous utility level and an interpolation of the current guess of v for future utility. We perform these updating steps up to the point where the value function has converged, meaning that the difference in v between two successive iteration steps is sufficiently small. The subroutine then sets v to the actual value function, i.e. without the transformation.

Program 9.10.a shows how the main program calculates the welfare effects of changing the long-run government debt level b_y . We first define an array of length $0:NB$ in which we store the respective debt levels b_y for which we want to calculate utilitarian welfare. We then initialize the model and calculate a benchmark balanced growth path with debt-to-output ratio of 60 per cent. The corresponding utilitarian welfare level is stored in the variable w_0 . We then choose $NB+1$ different points on the interval $[0.5, 1.5]$ to search for the welfare maximizing debt level b_y . Note that we, of course, have to calculate a

Program 9.10.a Calculating welfare for different levels of public debt

```

program HetAg_Gov
[.....]
integer, parameter :: NB = 4
integer :: ib
real*8 :: blevels(0:NB), Wlevels(0:NB), w0

! initialize variables
call initialize()

! benchmark equilibrium
by = 0.6d0

! set initial guess of the interest rate
r = 0.042d0

! solve initial equilibrium
call solve_model()

! store initial uility level
W0 = sum(phi(:,:,)*V(:,:))

! set up number of debt levels
call grid_Cons_Equi(blevels, 0.5d0, 1.5d0)

! calculate equilibrium with different debt levels
do ib = 0, NB
    by = blevels(ib)
    call solve_model()
    Wlevels(ib) = sum(phi(:,:,)*V(:,:))
enddo

! plot compensating variation
call plot(blevels, ((Wlevels/W0)**(1d0/egam)-1d0)*100d0)
call execplot
[.....]
end program

```

completely new balanced growth path for any of these levels. The corresponding value for the welfare criterion is stored in the array `wlevels`. Finally, we plot the Hicksian equivalent variation $\Delta(b_y)$ for the different debt levels under investigation using the plotting routines from the toolbox.

We parameterize the model in accordance with the literature. Specifically, we let $\gamma = \frac{1}{1.5}$, $\beta = 0.98$, $\alpha = 0.3$, $\delta = 0.075$, $\rho = 0.6$, and $\sigma_\epsilon^2 = 0.09$. The growth rate is set at a moderate 1.85 per cent. We let the labour-income tax rate be $\tau_w = 0.27$ and assume that government spending to GDP amounts to $g_y = 0.217$. Using this parametrization generates the welfare effects shown in Figure 9.13. The optimum quantity of debt in this setup amounts to roughly 100 per cent of GDP.²⁷ In the heterogeneous agent model, the role of public debt is twofold: On the one hand, it provides liquidity and its crowding-out effect raises the interest rate. This improves households' opportunities to insure themselves against income fluctuations using private savings and therefore increases welfare. On the other hand, when crowding out private capital, the economy moves further away from the golden rule and aggregate production activity declines. In addition, a higher stock of debt requires higher tax rates for the government to service interest repayments. Both a lower aggregate activity and higher taxes harm long-run consumption levels and therefore lower welfare. The optimum quantity of debt balances the positive and negative effects. Yet, we also find that the welfare costs of not being at the optimal debt level are rather small. Moving from the benchmark debt level of 0.6 to the optimum level of 1.0 is equivalent to an increase in consumption of 0.014 per cent for each household. A debt level of 150 per cent of GDP is associated with a welfare loss of 0.01 percent in consumption-equivalent terms.

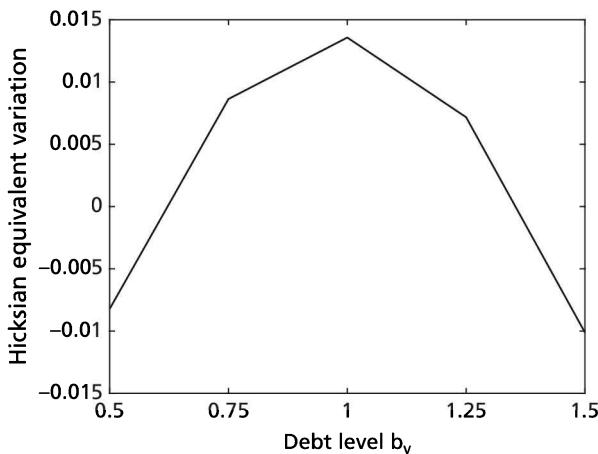


Figure 9.13 The optimum quantity of debt

²⁷ Note that the curve loses its kinks when we significantly increase the number of debt levels for which we calculate welfare effects.

9.5 Further reading

In this chapter we introduced the principles of dynamic macroeconomics in infinite-horizon models pioneered by Ramsey (1928). RBC analysis was initiated by Kydland and Prescott (1982). King, Plosser, and Rebelo (1988), and Cooley and Prescott (1995)—from whom we took parameter values—provide good introductions to the basic concepts as well as to analytical and numerical methods. Galí (1994) studies the effects of government expenditure on macroeconomic stability both theoretically and empirically. Popular early representatives of heterogeneous agent models are Bewley (1986), Imrohoroglu (1989), Huggett (1993), and Huggett (1997), but Aiyagari (1994) has become the standard reference for this research field. Marcat, Obiols-Homs, and Weil (2007) discuss the heterogeneous agent model with variable labour supply.

There are many potential extensions to the basic growth model. One could, for example, introduce a government sector that levies distortionary taxes like in McGrattan (1994) or disaggregate the production sector, see e.g. Huffman and Wynne (1999). We could also think about different or additional shocks to the economy. Bencivenga (1992) studies taste shocks that alter the utility function from period to period, Greenwood, Hercowitz, and Huffman (1988) consider technology shocks that affect the value of investment in the successive period. In more recent work, Beaudry and Portier (2007) analyse under which conditions changes in expectations about future productivity can drive business-cycle movements.

Readers more interested in alternative solution methods should look at Aruoba, Fernandez-Villaverde, and Rubio-Ramirez (2006) or consult Heer and Maussner (2009), who also discuss how to take models to the data by proper calibration. Adda and Cooper (2003), on the other hand, show how to estimate models using various data from different sources. Kopecky and Suen (2010) give a good overview on different discretization methods for stochastic processes and test their accuracy. Since it gives the best approximation to processes with a high persistence parameter ρ , the method of Rouwenhorst (1995) is their method of choice. As a high persistence is quite common in economic models, this method is better suited than the widely used method of Tauchen (1986). However, the method of Tauchen (1986) has other advantages.

Finally, the most natural next step is to unify aggregate and idiosyncratic shocks in a single infinite-horizon model. This has been pioneered by Krusell and Smith (1998) and their computational approach is explained in detail in Heer and Maussner (2009).

9.6 Exercises

- 9.1. Introduce population growth $N_t = (1 + n_p)N_{t-1}$ into the deterministic Ramsey model with $u(c) = \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}}$.

- a) Derive the steady-state solutions and explain in economic terms how the population growth rate affects the long-run equilibrium.
- b) Solve the model numerically using policy function iteration. What happens with the consumption function when the population growth rate increases from $n_p = 0.005$ to $n_p = 0.025$?
- c) Solve the model numerically using value function iteration. What happens in case $\beta(1 + n_p) > 1$? Explain!

Compute the error of the approximate solution for both numerical approaches using the Euler equation error.

- 9.2. Consider a simplified version of the deterministic growth model where we set $n_p = 0$, $\delta = 1$ and assume $u(c) = \ln c$.

- a) In this special case it is possible to derive an analytical solution for a value function of the form $V(k) = A + B \ln k$. Compute the policy function $c(k)$ and the value function $V(k)$. Hint: $c(k) = (1 - \beta\alpha)k^\alpha$.
- b) Solve the model numerically using policy function iteration.
- c) Solve the model numerically using value function iteration.

Use the numerical parameters $[k_l, k_u] = [0.1, 0.4]$ and $k_0 = 0.1$. Derive the error of the approximate solution for both numerical approaches using the analytical solution for $c(k)$.

- 9.3. Consider a version of the deterministic growth model with two sectors that produce a consumption and a capital good under perfect competition, meaning

$$c_t = (k_{c,t})^{\alpha_c} \quad \text{and} \quad i_t = (k_{i,t})^{\alpha_i}$$

with $0 < \alpha_c, \alpha_i < 1$ and $k_t = k_{c,t} + k_{i,t}$. The capital stock of the future depends on the amount of capital that is produced in the investment sector, i.e. $k_{t+1} = (1 - \delta)k_t + k_{i,t}^{\alpha_i}$. The social planner wants to maximize the utility of a representative agent with utility function

$$\max_{\{k_{c,t}, k_{i,t}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

subject to the above constraints.

- a) Derive the first-order condition for this dynamic programming problem.
- b) Solve the two-sector Ramsey model using policy function iteration with $\delta > 0.075$, $\alpha_c = 0.4$, and $\alpha_i = 0.5$.

Hint: In order to improve the stability of the algorithm, make sure that $0.1 < k_c < k$ and $0.1 < k_c^+ < k^+$.

- 9.4. Consider a version of the deterministic growth model with variable labour supply. Assume the same preferences and parameter choices as in the RBC model

$$u(c, 1 - l) = \frac{[c^\nu(1 - l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

with $\nu = 0.36$.

- a) Set up the dynamic programming problem and derive the first-order conditions.
- b) Compute the steady state of the model either analytically or numerically using subroutine `fzero`.
- c) Solve the policy functions numerically using value function iteration and identify the steady state.

- 9.5. Consider the deterministic growth model with variable labour supply from the previous exercise and introduce a government sector that provides a public good g , which amounts to 10 per cent of *steady-state* output *without* taxes. Taxes can be either levied on capital income or on wage income. Hence, the government budget constraint in period t is

$$g = \tau_t^w w_t l_t + \tau_t^r r_t k_t.$$

- a) Set up the dynamic programming problem and derive the first-order conditions.
- b) Compute the steady state of the model numerically using subroutine `fzero` with different combinations of (τ^w, τ^r) . What is the optimal tax structure if the social planner wants to maximize the utility of a representative agent?
- c) Solve the model numerically using policy function iteration for $\tau^r \leq 0.15$.

- 9.6. Consider the autocorrelated process

$$\eta_t = \rho \eta_{t-1} + \epsilon_t \quad \text{with} \quad \epsilon_t \sim N(0, \sigma_\epsilon^2) \quad \text{and} \quad 0 \leq |\rho| < 1.$$

Derive the grid of realizations $\{\hat{\eta}_g\}_{g=1}^m$ and the transition matrix $\hat{\pi}(\hat{\eta}_{g+1} | \hat{\eta}_g)$ using the subroutine `discretize_AR` for $m = 15$, $\sigma_\epsilon^2 = 1$, and $\rho = 0.5$.

- a) Compute the unconditional distribution $\hat{\mu}_u$ and $\hat{\sigma}_u^2$ as well as the autocorrelation parameter $\hat{\rho}$ of the approximated AR(1) process by iterating the transition matrix n times. Determine the approximation error $\hat{\sigma}_u^2 - \frac{\sigma_\epsilon^2}{1-\rho^2}$ for alternative iteration values $n = 5, 10, 20, 50$.
- b) Increase the autocorrelation value to $\rho = 0.9$ and report the same results.
- c) Why does the approximation process converge so differently?
- d) Plot the approximated and the true cumulative distribution of η .

- 9.7. Consider the stochastic growth model with fixed labour supply.
- Solve the model numerically using value function iteration (VFI) and subroutine `fminsearch`.
 - Solve the model numerically using endogenous grid points and VFI. A good description of the required algorithm is provided by Barillas and Fernandez-Villaverde (2007).
 - Compare the two approaches in terms of required iterations, speed, and accuracy using the Euler equation error.

- 9.8. Consider the standard RBC model with variable labour supply.

- Solve the model numerically using value function iteration.
- Solve the model numerically with a two-dimensional root-finding approach with unknowns $c(\hat{k}_v, \hat{\eta}_g)$ and $l(\hat{k}_v, \hat{\eta}_g)$.

Which solution algorithm is more efficient?

- 9.9. Hansen (1985) discusses the stochastic growth model with indivisible labour. Here a fraction κ of households works full-time $h_0 < 1$, while the fraction $1 - \kappa$ does not work at all and consumes the total time endowment as leisure ℓ . Aggregate labour supply in the economy therefore is αh_0 . The utility function is specified as $u(c, \ell) = \log c + A \log \ell$, so that expected individual utility is

$$\begin{aligned} E[u(c, \ell)] &= \kappa [\log c + A \log(1 - h_0)] + (1 - \kappa)[\log c + A \log 1] = \\ &\quad \log c + A\kappa \log(1 - h_0) = u(c, \kappa). \end{aligned}$$

- a) Derive the optimality conditions to solve the dynamic program

$$\begin{aligned} V(\hat{k}_v, \hat{\eta}_g) &= \max_{c, \kappa} \quad u(c, \kappa) + \beta \sum_{g^+ = 1}^m \pi_{gg^+} \cdot V(k^+, \hat{\eta}_{g^+}) \\ \text{s.t.} \quad (1 - \delta)\hat{k}_v + \exp(\hat{\eta}_g)(\hat{k}_v)^{\alpha}(\kappa h_0)^{1-\alpha} &= c + k^+ \end{aligned}$$

- b) Compute the solution of this problem using policy function iteration and the parameter set $\alpha = 0.36$, $\delta = 0.025$, $\beta = 0.99$, $A = 2$, $h_0 = 0.53$, $\rho = 0.95$, and $\sigma_\epsilon = 0.00712$.

- 9.10. Derive the formulas for the welfare changes (9.16) and (9.17).

- 9.11. Consider a version of the RBC model with distortionary labour taxation. The budget constraint of the government reads

$$g_t = \tau_t^w w_t l_t.$$

Hence the government only levies proportional taxes on income from labour and there is no lump-sum tax available.

- a) Write down the first-order conditions of this problem.
 - b) Implement the distortionary tax in the RBC model.
 - c) Now study the effects of procyclical vs. constant government consumption policy. Show the variation of the (endogenous) wage tax rate. Are there any differences to the case with lump-sum taxes in terms of welfare cost?
- 9.12. Consider the same version of the RBC model with distortionary taxation as in the previous exercise. Now compare the stochastic properties and the welfare effects of procyclical government policy with either labour taxes or income taxes. In the latter case the government budget constraint reads

$$g_t = \tau_t^\gamma [w_t l_t + r_t k_t].$$

- 9.13. Solve the heterogeneous agent model using the method of endogenous grid points. Calculate Euler equation errors (EEEs) and see how these errors change when you adapt the growth rate of the wealth grid. Find an optimal growth rate for the wealth grid. Note that the computation of the EEE only makes sense when $\alpha^+ > 0$, as the Euler equation doesn't hold for a liquidity-constrained household.
- 9.14. Consider a version of the heterogeneous agent model with variable labour supply. Assume the same preferences and parameter choices as in the RBC model

$$u(c, l) = \frac{[c^\nu (1 - l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

- a) Set up the dynamic programming problem and derive the first-order conditions.
- b) Compute the steady state of the model using subroutine `fzero`.

Hint: Use the capital-labour ratio $k = K/L$ in the function `eq_capital`.

10 Life-cycle choices and risk

The optimal savings and investment decisions of households along the life cycle were a central issue in Chapter 5. There, savings decisions were made under various forms of risks. However, we restricted our analysis to three period models owing to the limitations of the numerical all-in-one solution we used. In this chapter we want to take a different approach. Applying the dynamic programming techniques learned so far allows us to separate decision-making at different stages of the life cycle into small sub-problems and therefore increase the number of periods we want to look at enormously. This enables us to take a much more detailed look at how life-cycle labour supply, savings, and portfolio-choice decisions are made in the presence of earnings, investment, and longevity risk. Unlike in Chapter 9, the models we study here are partial equilibrium models. Hence, all prices as well as government policies are exogenous and do not react to changes in household behaviour.

This chapter is split into two parts. The first part focuses on labour supply and savings decisions in the presence of labour-productivity and longevity risk. Insurance markets against these risks are missing, such that households will try to self-insure using the only savings vehicle available, a risk-free asset. This model is a quite standard workhorse model in macroeconomics and a straightforward general equilibrium extension exists, the overlapping generations model, which we study in Chapter 11. In the second part of the chapter, we slightly change our viewpoint and look upon the problem of life-cycle decision-making from a financial economics perspective. We therefore exclude labour-supply decisions, but focus on the optimal portfolio choice of households along the life cycle, when various forms of investment vehicles like bonds, stocks, annuities, and retirement accounts are available.

10.1 Labour supply, savings, and risky earnings

This section is devoted to analysing consumption and savings behaviour when households face uncertainty about future earnings and the length of their life span. We study how households can use precautionary savings in a risk-free asset as a means to self-insure against the risks they face. While in our baseline model we assume that agents always work full-time, we relax this assumption later on by considering a model with endogenous labour supply as well as a model with a labour-force participation decision of second earners within a family context.

10.1.1 THE BASELINE MODEL

Let us assume that the life of a household starts at age 1 and this household can live at most up to some age J . The variable $j = 1, \dots, J$ shall denote an individual's actual age. Let us furthermore assume that survival from one period to the next is stochastic and that ψ_j is the age-specific probability for an agent to survive from age $j - 1$ to age j , conditional on still living at age $j - 1$. Consequently, the unconditional probability of surviving to age j is given by $\prod_{i=1}^j \psi_i$ with $\psi_1 = 1$. The household has preferences over consumption c_j at different ages, which can be represented by a time separable, expected, and discounted utility function

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) u(c_j) \right]. \quad (10.1)$$

β denotes the time discount factor and $u(c)$ is the instantaneous utility function. As in Chapters 8 and 9, we let $u(c) = \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}}$, with γ denoting the intertemporal elasticity of substitution.

Agents start their working life at age 1 and, conditional on surviving, retire at age j_r . In each period during the working years, an agent receives an endowment of productive efficiency units h_j which she can supply to the market at the wage rate w . Labour supply is inelastic and equal to 1, so that changes in labour productivity translate one-to-one into changes in labour income. To be more precise, we assume that h_j is a function of a (deterministic) age-profile of earnings e_j , a fixed productivity effect θ that is drawn at the beginning of the life cycle, and a transitory component η_j that evolves stochastically over time and that has an autoregressive structure of degree 1, that is

$$\eta_j = \rho \eta_{j-1} + \epsilon_j \quad \text{with} \quad \epsilon_j \sim N(0, \sigma_\epsilon^2) \quad \text{and} \quad \eta_1 = 0.$$

Given this structure, a household's labour productivity is

$$h_j = \begin{cases} e_j \cdot \exp [\theta + \eta_j] & \text{if } j < j_r \quad \text{and} \\ 0 & \text{if } j \geq j_r. \end{cases}$$

At the mandatory retirement age j_r , labour productivity h_j falls to zero and households receive pension benefits pen_j . To keep things as simple as possible, this pension benefit is unrelated to the individual's labour-income history, but rather computed as a fraction κ of an average of the age-specific productivity component e_j , i.e.

$$pen_j = \begin{cases} 0 & \text{if } j < j_r \quad \text{and} \\ \kappa \cdot \frac{w}{j_r-1} \cdot \sum_{j=1}^{j_r-1} e_j & \text{if } j \geq j_r. \end{cases}$$

In order to smooth consumption over time, households can save resources a_j at age j , which pay a return r in the next period. Households consequently maximize expected utility (10.1) subject to the (periodical) budget constraints

$$a_{j+1} + c_j = (1 + r)a_j + wh_j + pen_j, \quad (10.2)$$

where both r and w are set exogenously due to the partial equilibrium character of our model.

The dynamic programming problem Since our problem is additively separable over time, we can again write the household optimization problem as a dynamic program. In order to do so we first have to define the state space for the household. Obviously, like in the heterogeneous agent model of Chapter 9, we need individual wealth a as well as the labour-productivity shocks θ and η to be part of the state space. In addition, since households are not only heterogeneous with respect to savings and labour productivity, but also with respect to their age, j should become an element of the state space as well. The dynamic programming problem of the household then reads

$$\begin{aligned} V(j, a, \theta, \eta) &= \max_{c, a^+} u(c) + \beta \psi_{j+1} E[V(j+1, a^+, \theta, \eta^+) | \eta] \\ \text{s.t. } &a^+ + c = (1 + r)a + wh + pen, \quad a^+ \geq \underline{a}(j+1, \theta), \\ &\eta^+ = \rho\eta + \epsilon^+ \quad \text{with} \quad \epsilon^+ \sim N(0, \sigma_\epsilon^2). \end{aligned}$$

There are a couple of things to note here. First, we defined a very general borrowing constraint $\underline{a}(j+1, \theta)$, which may depend on the household's age j and the permanent productivity shock θ . This formulation includes the most simple case of a non-negative savings restriction (i.e. $\underline{a}(j+1, \theta) = 0$), but also allows for endogenously derived constraints which we explore below. Second, as the shock η^+ evolves stochastically over time, the household has to form correct expectations. Note that because of the simple autoregressive structure of the labour-productivity process, the current value of η is the only information the household needs in order to make proper forecasts about η^+ . Age just evolves linearly, so that the next period's age is $j + 1$. θ is a fixed productivity component drawn at the beginning of the life cycle. Hence it stays constant over time. Last but not least, the dynamic programming problem is different in nature compared to all the problems we have solved so far in the sense that it is not an infinite-horizon problem. While in the infinite-horizon problem discussed previously the value function was an object that is independent of time, it is now actually a function of age. We know, however, that there is a maximum age J after which the household dies with certainty. Since the agent does not derive any utility from leaving bequests, we can therefore specify a terminal condition for the value function as

$$V(J+1, a, \theta, \eta) = 0. \quad (10.3)$$

First-order condition All individual variables and value functions in this model depend on the full set of state variables. To avoid a very lengthy notation, we use a shortcut and define the state vectors

$$z = (j, a, \theta, \eta) \quad \text{and} \quad z^+ = (j + 1, a^+, \theta, \eta^+).$$

We can then reformulate the household's optimization problem to

$$\begin{aligned} V(z) &= \max_{c, a^+} u(c) + \beta \psi_{j+1} E[V(z^+) | \eta] \\ \text{s.t. } &a^+ + c = (1 + r)a + wh + pen, \quad a^+ \geq \underline{a}(j + 1, \theta), \\ &\eta^+ = \rho\eta + \epsilon^+ \quad \text{with} \quad \epsilon^+ \sim N(0, \sigma_\epsilon^2). \end{aligned}$$

The solution to this problem is a value function $V(z)$ as well as policy functions $c(z)$ and $a^+(z)$ that inform us about the consumption and savings behaviour of the individual at each element of the state space. The Lagrangian that corresponds to the above optimization problem reads

$$\mathcal{L} = u(c) + \beta \psi_{j+1} E[V(z^+) | \eta] + \lambda [(1 + r)a + wh + pen - a^+ - c].$$

It directly leads us to the first-order condition

$$u'(c) = \beta \psi_{j+1} E[V_a(z^+) | \eta].$$

The envelope theorem quickly reveals that

$$V_a(z^+) = (1 + r)u'(c(z^+)) = (1 + r) \cdot c(z^+)^{-\frac{1}{\gamma}}.$$

Hence, the solution to the consumption-savings problem of the household at any state vector z is given by the Euler equation

$$c = \left(\beta \psi_{j+1} (1 + r) \cdot E \left[c(z^+)^{-\frac{1}{\gamma}} \middle| \eta \right] \right)^{-\gamma}. \quad (10.4)$$

Note that this equation only holds if the household is not borrowing-constrained, i.e. if $a^+ \geq \underline{a}(j + 1, \theta)$. We discuss the case of a borrowing-constrained household at a later point.

Numerical implementation Like in the heterogeneous agent model of Chapter 9, solving this problem involves two steps: computing the value and policy functions and deriving a distribution of households over the state space that is consistent with the decisions made at each element of the state space z . To be able to compute policy functions

and the household distribution, we have to discretize the state space, which now has four instead of two dimensions. While age $j = 1, \dots, J$ is already discrete, we have to choose proper values along the asset dimension and discretize both the permanent labour-productivity components θ and the transitory component η . We assume that θ only takes on two values, $\hat{\theta}_1$ and $\hat{\theta}_2$, which both arise with the same probability 0.5.¹ In order to generate a mean and variance of 0 and σ_θ^2 , respectively, we specify

$$\hat{\theta}_1 = -\sigma_\theta \quad \text{and} \quad \hat{\theta}_2 = \sigma_\theta.$$

We discretize the transitory shock component into realizations $\hat{\eta}_g$ and a transition matrix $\pi_{gg'}$ with $g = 1, \dots, m$ using the Rouwenhorst method in `discretize_AR` from the toolbox.

Dealing with the borrowing constraint When it comes to defining the asset state space, we have to take into account that there is a state-dependent borrowing constraint $\underline{a}(j + 1, \theta) \leq 0$. One possible way of dealing with this constraint would be to make the discretized values of the state space contingent on age and the realization of the permanent productivity component, i.e. to specify grids $\{\hat{a}_{v,j,i}\}_{v=0}^n$, where $i = 1, 2$ stands in for the realization of θ . Then we could define those grids such that $\hat{a}_{0,j,i} = \underline{a}(j, \theta_i)$. The more convenient way of dealing with the borrowing constraint, however, is to define a variable

$$\tilde{a} = a - \underline{a}(j + 1, \theta) \quad \Leftrightarrow \quad a = \tilde{a} + \underline{a}(j + 1, \theta).$$

\tilde{a} denotes the savings of a household in excess of the borrowing limit $\underline{a}(j + 1, \theta)$. This means that when $\tilde{a} = 0$, the household is exactly at the borrowing constraint, while a level of assets of $a = 0$ corresponds to $\tilde{a} = -\underline{a}(j + 1, \theta)$. The great advantage of this formulation is that we can transform the budget constraint of the household into

$$[\tilde{a}^+ + \underline{a}(j + 1, \theta)] + c = (1 + r)[\tilde{a} + \underline{a}(j, \theta)] + wh + pen. \quad (10.5)$$

Then the complicated borrowing constraint $a^+ \geq \underline{a}(j + 1, \theta)$ becomes the very simple borrowing constraint $\tilde{a}^+ \geq 0$, which we can handle as easily as in the heterogeneous agent model of Chapter 9. We consequently formulate our household problem in terms of \tilde{a} and discretize its state space like in (9.22), meaning as a growing grid with gridpoints \hat{a}_v for $v = 0, 1, \dots, n$.

¹ Note that once we want to have more realizations of the fixed effect, we can use the subroutine `normal_discrete` to provide realizations $\hat{\theta}_i$ and the corresponding probabilities π_i^θ for these realizations to occur.

Setting up variables The initialization of all variables is carried out in the subroutine `initialize` that is shown in Program 10.1. We first specify prices, survival probabilities, and an age profile of productivities e_j . Knowing these productivities and the wage rate, we can immediately calculate the pension benefit. We then discretize the labour-productivity components as well as the state space for \tilde{a} , which is not shown here but follows the same logic as in the heterogeneous agent model of Chapter 9. Last but not least, we have to define proper values of the borrowing limit $a(j + 1, \theta)$. To do this, we have to realize that there is something we could call a natural or endogenous borrowing limit in our model. This borrowing limit can be defined using the following line of reasoning: Let's assume a household drew the permanent productivity realization $\hat{\theta}_i$, $i = 1, 2$. Now the worst thing that can happen to the individual is that she draws the lowest transitory productivity realization $\hat{\eta}_1$ over and over again, meaning at every age j of her working

Program 10.1 Initializing variables for the life-cycle model

```

subroutine initialize
  [.....]
  ! net prices (after taxes and transfers)
  r = 0.04d0
  w = 1.0d0

  ! set survival probabilities
  psi = [.....]

  ! initialize age earnings process
  eff(1:JR-1) = [.....]
  eff(JR:JJ) = 0d0

  ! old-age transfers
  pen = 0d0
  pen(JR:JJ) = 0.5d0*w*sum(eff)/dble(JR-1)
  [.....]
  ! exogenous borrowing constraint
  a_bor = 0d0

  ! calculate endogenous borrowing constraints
  do ij = 1, JJ
    do ip = 1, NP

      ! calculate natural borrowing limit
      abor_temp = 0d0
      do ijj = JR, ij, -1
        abor_temp = abor_temp/(1d0+r) + &
                     eff(ijj)*theta(ip)*eta(1) + pen(ijj)
      enddo
      abor_temp = min(-abor_temp/(1d0+r)+1d-4, 0d0)

      ! set maximum of both borrowing constraints
      a_bor(ij, ip) = max(a_bor(ij, ip), abor_temp)
    enddo
  enddo
  [.....]
end subroutine
  
```

life. Hence, we can define the minimum present value of all future income of an agent at age j as

$$\sum_{k=j+1}^J \frac{w e_k \cdot \exp(\hat{\theta}_k + \hat{\eta}_k) + p e_n k}{(1+r)^{k-j}}.$$

To make sure that the individual will always be able to repay her full amount of debt and would never need to default (or incur a negative consumption level), we should make sure that the value of debt never exceeds this present value of income.² In addition to this endogenous borrowing limit, we allow for the specification of an exogenous borrowing constraint. The actual borrowing constraint then is defined by whatever value is smaller in absolute terms, the exogenous or the endogenous constraint. In the current specification of the model, we assume a very tight exogenous borrowing constraint of zero. However, this constraint can be relaxed easily.

Solving the household problem Being equipped with a discretization of the state space and a proper definition of borrowing constraints, we can turn to solving for policy functions using the first-order conditions defined in (10.4). This is done in the subroutine `solve_household`, which is shown in Program 10.1.a. In contrast to the policy function iterations of Chapter 9, there is now a terminal period J after which the household dies with certainty. The solution process starts at this last period of life and then works its way backwards in time, i.e. solves policy functions at ages $j = J-1, J-2, \dots, 1$. Since the household will die with certainty at the end of the last period and there is no bequest motive, it is optimal to leave nothing for the future $a(z) = \tilde{a}(z) = 0$ and consume all available resources at age J , i.e.

$$c(z) = (1+r)a(z) + p e_n J = (1+r)[\tilde{a}(z) + \underline{a}(J, \theta)] + p e_n J.$$

Note that the household does not generate any more labour income during retirement. Hence, the amount of available resources only depends on the asset position of the agent, but is independent of the productivity realization θ and η . As the future value function is equal to zero, the current value function can simply be calculated from

$$V(z) = \frac{c(z)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}},$$

which is done in the function `valuefunc` that we discuss below.

² Note that for numerical stability, we add a very small value of 10^{-4} to the natural borrowing limit.

Program 10.1.a Solving the life-cycle household problem

```

subroutine solve_household()
[.....]
! get decision in the last period of life
do ia = 0, NA
  aplus(JJ, ia, :, :) = 0d0
  c(JJ, ia, :, :) = (1d0+r)*(a(ia)+a_bor(JJ, 1)) + pen(JJ)
  V(JJ, ia, :, :) = valuefunc(0d0, c(JJ, ia, 1, 1), JJ, 1, 1)
enddo

! interpolate individual RHS
call interpolate(JJ)

do ij = JJ-1, 1, -1

  ! set ip_max and is_max
  [.....]

do ia = 0, NA

  ! determine decision for zero assets at retirement
  if(ij >= JR .and. ia == 0 .and. pen(ij) <= 1d-10) then
    aplus(ij, ia, :, :) = 0d0
    c(ij, ia, :, :) = 0d0
    V(ij, ia, :, :) = valuefunc(0d0, 0d0, ij, 1, 1)
    cycle
  endif

  ! solve household problem using root-finding method
  do ip = 1, ip_max
    do is = 1, is_max
      [.....]
    enddo
  enddo

  ! copy decision in retirement age
  if (ij >= JR) then
    aplus(ij, ia, :, :) = aplus(ij, ia, 1, 1)
    c(ij, ia, :, :) = c(ij, ia, 1, 1)
    V(ij, ia, :, :) = V(ij, ia, 1, 1)
  endif
  enddo

  ! interpolate individual RHS
  call interpolate(ij)

  write(*,'(a,i3,a)')'Age: ',ij,' DONE!'
enddo

end subroutine

```

Interpolation After having solved for the policy functions $c(z)$ and $a^+(z)$ at age J for all the different elements of the state space, we call up the subroutine `interpolate` which is shown in Program 10.1.b. This subroutine is responsible for calculating the right-hand side of the first-order condition (10.4) for a specific age j . This means that for any potential level of assets \hat{a}_v , any permanent shock level $\hat{\theta}_i$, and any potential transitory

Program 10.1.b Computation of expected marginal utility and value function

```

subroutine interpolate(ij)
    [.....]
    do ia = 0, NA
        do ip = 1, NP
            do is = 1, NS

                ! calculate the RHS of the first order condition
                RHS(ij, ia, ip, is) = 0d0
                EV(ij, ia, ip, is) = 0d0
                do is_p = 1, NS
                    chelp = max(c(ij, ia, ip, is_p), 1d-10)
                    RHS(ij, ia, ip, is) = RHS(ij, ia, ip, is) + &
                        pi(is, is_p)*margu(chelp)
                    EV(ij, ia, ip, is) = EV(ij, ia, ip, is) + &
                        pi(is, is_p)*V(ij, ia, ip, is_p)
                enddo
                RHS(ij, ia, ip, is) = (beta*psi(ij)*(1d0+r)* &
                    RHS(ij, ia, ip, is))**(-gamma)
                EV(ij, ia, ip, is) = &
                    (egam*EV(ij, ia, ip, is))**((1d0/egam)
                enddo
            enddo
        enddo

end subroutine

```

shock component (of the previous period) $\eta_{j-1} = \hat{\eta}_g$, this subroutine computes the values

$$RHS(j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g) = \left[\beta \psi_j (1+r) \sum_{g^+=1}^m \pi_{gg^+} \cdot c(j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_{g^+})^{-\frac{1}{\gamma}} \right]^{-\gamma}. \quad (10.6)$$

As we also know the value functions at the different elements of the state space, we can also calculate the expected value function

$$EV(j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g) = \left[\left(1 - \frac{1}{\gamma}\right) \cdot \sum_{g^+=1}^m \pi_{gg^+} \cdot V(j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_{g^+}) \right]^{\frac{1}{1-\frac{1}{\gamma}}}. \quad (10.7)$$

Note that we apply the same transformation as in the previous Chapters 8 and 9 for reasons of interpolation accuracy.

Given the right-hand side of the first-order condition (10.4) and the expected value function for the last period J , the subroutine `solve_household` of Program 10.1.a can proceed further and solve the household problem for the next-to-last period $J-1$. Since, as argued above, labour-productivity realizations play no role throughout retirement because of $e_j = 0$, we do not have to solve the dynamic program for all values of $\hat{\theta}_i$ and $\hat{\eta}_g$, but only for one particular combination. Consequently, we set `ip_max` and `is_max` to a

value of one if $j \geq j_r$ and to respectively 2 and m otherwise. For retired individuals we then just copy the policy and value functions to the remaining realizations of $\hat{\theta}_j$ and $\hat{\eta}_g$ afterwards. When solving for the optimal household decisions at all potential gridpoints \hat{a}_v throughout retirement, we have to take care of one more exception. When a retired agent has no private assets (i.e. $\tilde{a} = \hat{a}_0$) and there is no pension system in place (i.e. $pen_j = 0$), then there are no resources left over to consume. In this case, the first-order condition and the value function cannot be calculated numerically but, rather we have to set them manually. Consequently we assign a value of 0 to both consumption and the transformed value function. Note that households would never optimally choose a value of zero assets in the event that there is no pension system, since this would lead to exactly zero consumption. Hence, the mass of households at the point \hat{a}_0 should be equal to zero.

Finding the root of the first-order condition Finally, Program 10.1.c shows how we set up and call up the root-finding procedure `fzero` to solve the unrestricted first-order condition for each level of today's assets and productivity shocks. Our approach here is identical to the one we used to solve the heterogeneous agent model in Chapter 9. Hence, we first solve for the unrestricted solution of the household problem by determining the root of the first-order condition and ignoring the borrowing constraint of the household. We then check *ex post* for whether the solution to this problem satisfied the liquidity constraint or not. We specify the first-order condition (10.4) in terms of future savings \tilde{a}^+ , which is what the root-finding method will iterate over. The setup procedure consists

Program 10.1.c Calling the root-finding routine

```

! get initial guess for the individual choices
x_in = aplus(ij+1, ia, ip, is)

! set up communication variables
ij_com = ij
ia_com = ia
ip_com = ip
is_com = is

! solve the household problem using root-finding
call fzero(x_in, foc, check)

! write screen output in case of a problem
if(check)write(*,'(a, 4i4)')'ERROR IN ROOTFINDING : ', ij,ia,ip,is

! check for borrowing constraint
if(x_in < 0d0)then
    x_in = 0d0
    wage = w*eff(ij)*theta(ip)*eta(is)
    available = (1d0+r)*(a(ia) + a_bor(ij, ip)) + wage + pen(ij)
    cons_com = available - a_bor(ij+1, ip)
endif

! copy decisions
aplus(ij, ia, ip, is) = x_in
c(ij, ia, ip, is) = cons_com
V(ij, ia, ip, is) = valuefunc(x_in, cons_com, ij, ip, is)

```

of specifying an initial guess for the optimal level of assets and communicating the element of the state space for which we want to solve the first-order condition. As an initial guess we simply use the optimal savings decision for the same combination of assets and productivity levels in the next period $j + 1$, which we had already solved for in the previous iteration step. We use the communication variables that are specified in the module `globals` and end on `_com` for communication between the subroutine `solve_household` and the function defining the first-order condition. We then call up the subroutine `fzero` from the toolbox that is supposed to find the root of the first-order condition specified in function `foc`. The solution `x_in` which is returned by `fzero` needs to be checked for validity. This is done by means of the variable `check`, which would be assigned a value of `.true.` in case something went wrong during the root-finding process. In addition, we need to check whether the solution of the first-order condition actually satisfies the borrowing constraint. Recall that because we defined our problem in \tilde{a} and not a , checking for this is super easy. We just have to validate that `x_in` is larger than or equal to zero. If this is not the case, we set `x_in = 0d0` and compute the corresponding consumption level from the budget constraint (10.5) of the agent. Last but not least, we copy the optimal level of savings a^+ as well as consumption c into the respective storage variables. Note that the function `foc` stores the consumption level that corresponds to a choice of a^+ in the variable `cons_com`, so we do not have to recalculate it again.

Implementation of the first-order condition The function `foc` that returns the value of the first-order condition of the household optimization problem is shown in Module 10.1m.a. This function takes as input a level of future savings `aplus` and determines the corresponding consumption level of today using the budget constraint (10.5) of the household. It stores this value in the variable `cons_com`. Next we use the subroutine `linint_Grow` from the toolbox to determine left and right interpolation nodes and an interpolation weight for `aplus`. Finally, we interpolate the right-hand side of the first-order condition (10.4) using the values of RHS specified in (10.6) and return the value of the first-order condition to the root-finder.

Computing the value function Given optimal savings and consumption decisions at a specific state z , the function `valuefunc` computes the corresponding value function.³ It is shown in Module 10.1m.b. We therefore first limit the value of consumption to 10^{-10} in order to avoid computational errors. Then we proceed through exactly the same steps as in the subroutine `foc`. We first derive interpolation nodes and weights that correspond to the savings decision `aplus` and interpolate the expected future value function, which we stored in the array `EV` specified in (10.5). Note that we use the transformed value function for interpolation, which means that we have to transforms the interpolated value back to

³ The index for the permanent productivity shock $\hat{\theta}_t$ is `ip` while `is` is the index for the transitory shocks $\hat{\eta}_g$.

Module 10.1m.a The first-order condition

```

function foc(x_in)
    [.....]
    ! calculate tomorrows assets
    a_plus = x_in

    ! calculate the wage rate
    wage = w*eff(ij_com)*theta(ip_com)*eta(is_com)

    ! calculate available resources
    available = (1d0+r)*(a(ia_com) + a_bor(ij_com, ip_com)) &
                + wage + pen(ij_com)

    ! calculate consumption
    cons_com = available - (a_plus + a_bor(ij_com+1, ip_com))

    ! calculate linear interpolation for future part of foc
    call linint_Grow(a_plus, a_l, a_u, a_grow, NA, ial, iar, varphi)

    tomorrow = varphi*RHS(ij_com+1, ial, ip_com, is_com) + &
               (1d0-varphi)*RHS(ij_com+1, iar, ip_com, is_com)

    ! calculate first-order condition for consumption
    foc = cons_com - tomorrow

end function

```

Module 10.1m.b Calculating the value function of the life-cycle model

```

function valuefunc(a_plus, cons, ij, ip, is)
    [.....]
    ! check whether consumption or leisure are too small
    c_help = max(cons, 1d-10)

    ! get tomorrows utility
    call linint_Grow(a_plus, a_l, a_u, a_grow, NA, ial, iar, varphi)

    ! calculate tomorrow's part of the value function
    valuefunc = 0d0
    if (ij < JJ) then
        valuefunc = max(varphi*EV(ij+1, ial, ip, is) + &
                      (1d0-varphi)*EV(ij+1, iar, ip, is), 1d-10)**egam/egam
    endif

    ! add todays part and discount
    valuefunc = c_help**egam/egam + beta*psi(ij+1)*valuefunc

end function

```

the original form of the value function, i.e. we apply the transformation $\frac{(\cdot)^{1-\beta}}{1-\beta}$. Finally, we discount future utility with the factor $\beta\psi_{j+1}$ and add the instantaneous utility resulting from current consumption.

The distribution of households Given optimal decision rules $c(z)$ and $a^+(z)$ at all elements of the state space, we are still not able to compute any statistics from our model.

What is still missing is the distribution of households $\phi(z)$ that informs us about where households are actually located on the state space. The subroutine `get_distribution`, which is shown in Program 10.1.d, is responsible for the derivation of this distribution. This subroutine thereby follows the same logic as the corresponding subroutine in the heterogeneous agent model of Chapter 9. However, in the same way as value and policy functions are not time-invariant but depend on a terminal condition, so the distribution of households is not a time-invariant object either. As we know that agents start their life with zero assets, a productivity shock $\hat{\theta}_i$ that occurs with probability π_i^θ as well as a transitory productivity level of $\eta_1 = 0$, we can directly specify the distribution of households over the state space at age $j = 1$. To this end, we have to realize that a level of zero assets corresponds to a value of $\tilde{a} = -\underline{a}(1, \hat{\theta}_i)$. Since this level of savings \tilde{a} will typically not correspond to exactly one asset grid point \hat{a}_v , we use the subroutine

Program 10.1.d Calculating the measure of households

```

subroutine get_distribution()
[.....]
! set distribution to zero
phi(:, :, :, :) = 0d0

! get initial distribution in age 1
do ip = 1, NP

    ! find the zero on the asset grid
    call linint_Grow(-a_bor(1, ip), a_l, a_u, a_grow, &
                      NA, ial, iar, varphi)
    phi(1, ial, ip, is_initial) = varphi*dist_theta(ip)
    phi(1, iar, ip, is_initial) = (1d0-varphi)*dist_theta(ip)
enddo

! successively compute distribution across ages
do ij = 2, JJ

    ! iterate over yesterdays gridpoints
    do ia = 0, NA
        do ip = 1, NP
            do is = 1, NS

                ! interpolate yesterday's savings decision
                call linint_Grow(aplus(ij-1, ia, ip, is), a_l, &
                                  a_u, a_grow, NA, ial, iar, varphi)
                [.....]
                ! redistribute households
                do is_p = 1, NS
                    phi(ij, ial, ip, is_p) = phi(ij, ial, ip, is_p) + &
                        pi(is, is_p)*varphi*phi(ij-1, ia, ip, is)
                    phi(ij, iar, ip, is_p) = phi(ij, iar, ip, is_p) + &
                        pi(is, is_p)*(1d0-varphi)*phi(ij-1, ia, ip, is)
                enddo
            enddo
        enddo
    enddo
end subroutine

```

`linint_Grow` to determine the indices of the left and right asset values \hat{a}_l and \hat{a}_r between which the level \tilde{a} is located. In addition this subroutine returns an interpolation weight φ . Note that the interpolation nodes and the weight satisfy

$$\tilde{a} = -\underline{a}(1, \hat{\theta}_i) = \varphi \cdot \hat{a}_l + (1 - \varphi) \cdot \hat{a}_r.$$

Given that for an odd number of discretization points of the stochastic process $\hat{\eta}_g$ the median productivity level—i.e. the productivity level with index $\frac{m+1}{2}$ —satisfies $\hat{\eta}_{\frac{m+1}{2}} = 0$, we can specify the distribution of households over the state space at age $j = 1$ as

$$\phi(1, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g) = \begin{cases} \pi_i^\theta \cdot \varphi & \text{if } v = l \text{ and } g = \frac{m+1}{2}, \\ \pi_i^\theta \cdot (1 - \varphi) & \text{if } v = r \text{ and } g = \frac{m+1}{2}, \\ 0 & \text{otherwise.} \end{cases}$$

Knowing the distribution of households over the state space at age 1, we can compute the distribution at any successive age using the policy functions $a^+(z)$. Specifically, for each element of the state space z at age j , we compute the left and right interpolation nodes \hat{a}_l and \hat{a}_r as well as the corresponding interpolation weight φ using the subroutine `linint_Grow`. Again, the nodes and the weight satisfy

$$a^+(z) = \varphi \cdot \hat{a}_l + (1 - \varphi) \cdot \hat{a}_r.$$

As in Chapter 9, we now assume that the mass of agents $\phi(z)$ at the current element of the state space splits up, where a fraction φ moves to the gridpoint \hat{a}_l and the mass $1 - \varphi$ to the gridpoint \hat{a}_r . Taking into account the transition probabilities for the transitory productivity shock η_{gg^+} , this means that we distribute the mass of individuals at the state z to the state space at the next age $j + 1$ according to

$$\phi(j + 1, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_{g^+}) = \begin{cases} \phi(j + 1, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_{g^+}) + \varphi \cdot \pi_{gg^+} \cdot \phi(z) & \text{if } v = l, \\ \phi(j + 1, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_{g^+}) + (1 - \varphi) \cdot \pi_{gg^+} \cdot \phi(z) & \text{if } v = r. \end{cases}$$

Given the policy functions and the distribution of households over the state space, it is easy to compute cohort-specific means and variance of relevant variables. The cohort-means of consumption, income, or savings are given by

$$\bar{c}_j = \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi(z) \cdot c(z), \quad \bar{y}_j = w \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi(z) \cdot h(z) \quad \text{and}$$

$$\bar{a}_j = \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi(z) \cdot \hat{a}_v,$$

where $z = (j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g)$. Similar formulas can be used to calculate variances. The calculation of such cohort statistics is carried out in the subroutine `aggregation`, which we do not show here.

Calibration and outcomes We assume that one period of our model is associated with one year of calendar time. Agents enter the labour market at age 20 (model age 1), retire at age 65 ($j_r = 45$) and may live at most to age 100 ($J = 80$). Mortality rates ψ_j are chosen to match average survival probabilities of males and females in the United States in year 2010, while the preference parameters $\gamma = 0.5$ and $\beta = 0.98$ are set at fairly standard values. With respect to the income process we take an age-productivity profile e_j from US data and specify an initial endowment shock with a variance of $\sigma_\theta^2 = 0.242$. The persistence of the transitory shock is very high with an autocorrelation set at $\rho = 0.985$, which is a typical assumption in this literature. In addition, we let the variance of the innovation term σ_ϵ^2 be 0.022. For the choice of numerical parameters we use $n = 200$ points to discretize the asset state space. This should be enough to get an accurate approximation of the household's policy functions. The asset minimum is obviously $a_l = 0$, while we chose an asset maximum of $a_u = 600$ to ensure that no individual will ever reach it. We set a moderate growth rate of the asset grid of 0.05, leading the difference of the first two asset's gridpoints to be $\hat{a}_1 - \hat{a}_0 \approx 2 \cdot 10^{-3}$. In the current specification, we let the borrowing constraint be very tight at $\underline{a}(j, \theta) = 0$. We set the number of transitory shock levels at $m = 7$, which is accurate enough for our purposes. Finally, we normalize prices (net of tax) at $w = 1$ and $r = 0.04$ and set the replacement rate of the pension system to $\kappa = 0.5$.

The upper two panels of Figure 10.1 show the mean life-cycle profiles resulting from our model specification. Labour earnings increase throughout working life and peak shortly before retirement, owing to our specification of the age-productivity profile e_j . Upon entering retirement, income drops substantially as the pension system only replaces a fraction of average earnings.⁴ Consumption over the life cycle is much smoother than income, because individuals use savings to shift resources from working years to retirement years. This results in average consumption levels ranging below income in the first half of life, so that resources are left over to build up the private savings stock. This savings stock is then consumed throughout the retirement phase. At very old ages, average consumption falls steadily as death probabilities (and therefore discounting of the future value function) are high.

The mean profiles of consumption and income don't contain any information about the risk a household faces and how this risk is dealt with. Therefore the lower left panel of Figure 10.1 shows the age-specific coefficient of variation(CV) of income and consumption.⁵ In the present specification with inelastic labour supply, labour income risk cannot be influenced by the household. The CV of labour income increases over the life cycle, indicating that the labour-income distribution widens with age. The source

⁴ Note that we exclude interest income from private assets in this profile.

⁵ Recall that the CV is the standard deviation of a variable divided by its mean.

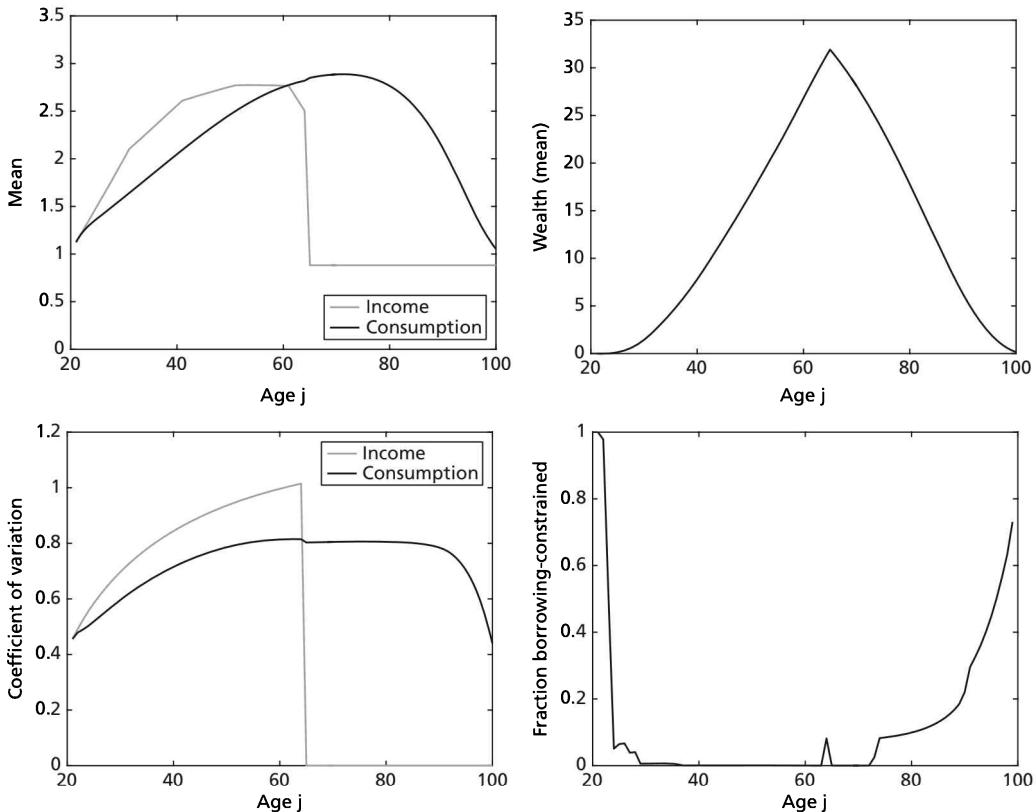


Figure 10.1 Life-cycle profiles in the baseline model

of this observation is the increasing dispersion coming from the transitory labour productivity component η_j . Recall that all agents face the same realization $\eta_1 = 0$ at the beginning of life, but at later periods different realizations of the innovation term ϵ_j lead to an increasing variance in η_j . Note that the variation in income drops to zero when agents enter retirement, since pension payments are identical for all individuals. Consumption inequality, on the other hand, contains an endogenous component that arises from individual decision-making. We find that consumption inequality across households increases less steeply along the life cycle than income inequality. Of course insurance markets against labour productivity are missing in this model. However, the household can use savings to (imperfectly) self-insure against future variations in labour productivity. These precautionary savings are typically built up when labour productivity is relatively high. A stock of precautionary savings can then be consumed upon a very negative productivity realization, ultimately leading consumption inequality to fall below income inequality.

There are two irregularities in this picture. First, when looking at the first periods of life, we find that consumption inequality perfectly tracks income inequality for two

or three years. This happens because in these periods of life borrowing constraints are binding quite strongly. This can be seen from the lower right panel of Figure 10.1 that shows the fraction of households in each cohort that would actually like to borrow against future income. This fraction is very high at initial periods of working life, because agents are expecting a substantial rise in future average labour productivity. In fact, most agents would like to borrow against these future income realizations. As they are not allowed to do so, they run into the borrowing constraint and therefore have no buffer savings available to smooth out consumption fluctuations over time. The second irregularity is to be found towards the end of the life cycle, when consumption inequality starts declining successively. This phenomenon can again be linked to the tight borrowing constraint. Towards the end of the life cycle, those individuals who were unlucky in the labour market and therefore built up only few savings for retirement will be the first to run out of savings in old age. When savings are exhausted, agents will just consume the total amount of the pension payment they receive. As individuals become older and older, more and more agents will exhaust their savings and fully consume the pension payment. Since the pension payment is identical for all households, this behaviour causes consumption inequality to decline.

Summing up, we find that private savings constitute a possibility to at least partially self-insure against future fluctuations in incomes and therefore to reduce levels of consumption inequality below those of income inequality. However, such a mechanism can only work when the individual is not restricted by a tight borrowing limit. In addition, savings serve as a means of redistributing resources between working and retirement ages, leading the average consumption profile of a household to be relatively smooth over time.

10.1.2 THE ROLE OF VARIABLE LABOUR SUPPLY

In this section we want to relax the assumption of inelastic labour supply by allowing individuals to actually choose how many hours they want to work in the market. To avoid complications, we assume from now on that the household always faces a tight borrowing constraint of $\underline{a}(j, \theta) = 0$, leading to $a(z) = \tilde{a}(z)$. When individuals decide how much to work, they need to have preferences over streams of consumption c_j and leisure ℓ_j . As in Chapter 9 we let there be a fixed time endowment of 1 that the agent can split between enjoying leisure and working in the market. Letting l_j denote labour hours, we consequently have $\ell_j + l_j = 1$. We assume that households preferences can again be represented by a time-separable expected utility function of the form

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) u(c_j, 1 - l_j) \right], \quad (10.8)$$

where instantaneous utility is given by

$$u(c_j, 1 - l_j) = \frac{\left[c_j^\nu (1 - l_j)^{1-\nu}\right]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Note that we used the very same specification of utility as in the RBC model discussed in Chapter 9. Specifically, utility of consumption and leisure takes a Cobb-Douglas form with a taste parameter ν for consumption so that the elasticity of substitution between consumption and leisure equals 1. The intertemporal elasticity of substitution is again constant and equal to γ , where $\frac{1}{\gamma}$ is the risk aversion of the household. We maintain all other assumptions with respect to productivity risk and the fiscal system.⁶ Hence, households maximize expected utility (10.8) subject to the (periodical) budget constraints

$$a_{j+1} + c_j = (1 + r)a_j + wh_j l_j + pen_j, \quad (10.9)$$

and the non-negativity constraint for savings $a_{j+1} \geq 0$.

The dynamic programming problem The household optimization problem reads

$$\begin{aligned} V(z) &= \max_{c, l, a^+} u(c, 1 - l) + \beta \psi_{j+1} E[V(z^+) | \eta] \\ \text{s.t. } &a^+ + c = (1 + r)a + whl + pen, \quad a^+ \geq 0, \quad l \geq 0 \\ \text{and } &\eta^+ = \rho\eta + \epsilon^+ \quad \text{with } \epsilon^+ \sim N(0, \sigma_\epsilon^2). \end{aligned}$$

where we again define $z = (j, a, \theta, \eta)$ and $z^+ = (j + 1, a^+, \theta, \eta^+)$ as the vectors of state variables. In addition to the non-negativity constraint on savings, we now need an additional constraint that ensures that households' labour supply will not turn negative. Put differently, this constraint guarantees that leisure consumption doesn't exceed the total time endowment of the household.⁷ The terminal condition for the value function at age $J + 1$ remains unchanged, see (10.3).

⁶ To account for the fact that households' working hours will be substantially fewer than 1, we multiply the pension payment by a factor of 0.33, which roughly corresponds to the average number of working hours of all households.

⁷ Strictly speaking we would also need constraints that ensure that consumption and leisure do not turn negative. With a proper specification of the utility function, however, this will not be a choice for the household anyway.

We can write the Lagrangian of the household optimization problem at age j as

$$\begin{aligned}\mathcal{L} = & \frac{[c^\nu(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta\psi_{j+1}E[V(z^+)|\eta] \\ & + \lambda [(1+r)a + whl + pen - a^+ - c]\end{aligned}$$

which immediately leads to the first-order conditions

$$\frac{\nu}{c} [c^\nu(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}} = \lambda \quad (10.10)$$

$$\frac{1-\nu}{1-l} [c^\nu(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}} = \lambda wh \quad (10.11)$$

$$\beta\psi_{j+1}E[V_a(z^+)|\eta] = \lambda. \quad (10.12)$$

The envelope theorem tells us that

$$V_a(z^+) = (1+r) \cdot \frac{\nu}{c(z^+)} \left[c(z^+)^{\nu} (1-l(z^+))^{1-\nu} \right]^{1-\frac{1}{\gamma}}$$

so that after combining (10.10) and (10.12) the intertemporal first-order condition reads

$$\begin{aligned}\frac{\nu [c^\nu(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{c} = & \\ & \beta\psi_{j+1}(1+r) \cdot E \left[\frac{\nu [c(z^+)^{\nu} (1-l(z^+))^{1-\nu}]^{1-\frac{1}{\gamma}}}{c(z^+)} \middle| \eta \right]. \quad (10.13)\end{aligned}$$

Of course this equation only holds if the household is not borrowing-constrained, i.e. if $a^+ \geq 0$. The intratemporal first-order condition that governs the relation between consumption and leisure can be derived from combining (10.10) and (10.11) to

$$c = \frac{\nu}{1-\nu} \cdot wh \cdot (1-l). \quad (10.14)$$

Again this first-order condition only holds when the non-negativity constraint on labour supply $l \geq 0$ is not binding. Together with the budget constraint (10.9) the inter- and the intratemporal first-order conditions (10.13) and (10.14) define the choice of the household.

Given the structure of the optimality condition (10.14), we can actually simplify the household problem a little bit. Specifically, we can express consumption and labour as

functions of the individual savings decision a^+ only. To do this, we plug condition (10.14) into the budget constraint (10.9) and obtain

$$l = l(a^+) = \min \left\{ \max \left[v + \frac{1-v}{wh} (a^+ - (1+r)a - pen), 0 \right], 1 \right\}. \quad (10.15)$$

The minimum and maximum operators take into account that labour supply remains positive and can't exceed the time endowment of 1. Consumption can then be derived as

$$c = c(a^+) = (1+r)a + whl(a^+) + pen - a^+. \quad (10.16)$$

Note that this calculation strategy also holds for retired individuals where $h = 0$ and therefore $l(a^+) = 0$. Using these definitions we can reduce the household problem to one equation in which the remaining unknown is a^+

$$\frac{v [c(a^+)^v (1 - l(a^+))^{1-v}]^{1-\frac{1}{\gamma}}}{c(a^+)} = \beta \psi_{j+1}(1+r) \cdot E \left[\frac{v [c(z^+)^v (1 - l(z^+))^{1-v}]^{1-\frac{1}{\gamma}}}{c(z^+)} \middle| \eta \right]. \quad (10.17)$$

Numerical implementation To implement the life-cycle model with variable labour supply, we actually do not have to make many changes compared to our baseline model in the previous section. We discretize the state space in the very same way as described above. The policy functions $c(z)$, $l(z)$, and $a^+(z)$ can be calculated using the first-order condition in (10.17) with c and l being defined as in (10.15) and (10.16). These calculations are again carried out in subroutine `solve_household` that starts at the terminal age J at which

$$a^+(z) = 0, \quad l(z) = 0 \quad \text{and therefore} \quad c(z) = (1+r)a(z) + pen_J.$$

The corresponding value function is

$$V(z) = \frac{[c(z)^v (1 - l(z))^{1-v}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Having solved for the policy functions of the terminal age J , the subroutine `interpolate` calculates the right-hand side of the first-order condition (10.17) at an age j for any potential level of assets \hat{a}_v , any permanent shock level $\hat{\theta}_i$ and any potential transitory shock component (of the previous period) $\eta_{j-1} = \hat{\eta}_g$ as

$$RHS(z) = \left[\beta \psi_j (1+r) \sum_{g^+=1}^m \pi_{gg^+} \frac{\nu \left[c(z^+)^\nu (1 - l(z^+))^{1-\nu} \right]^{1-\frac{1}{\gamma}}}{c(z^+)} \right]^{-\gamma}, \quad (10.18)$$

where we used $z = (j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g)$ and $z^+ = (j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_{g^+})$. Note that we do not use the first-order condition in exactly the form stated in (10.17), but apply the transformation $[\cdot]^{-\gamma}$ to both the left- and right-hand side.⁸ Doing so, the right-hand side of the first-order condition will not diverge to $-\infty$ as available future resources approach 0. Like in Chapters 8 and 9 this improves the accuracy of the interpolation method as well as the computational stability of the root-finding algorithm. The computation of the expected value function remains unchanged compared to the model with inelastic labour supply (10.7).

Optimal household decisions With the right-hand side of the first-order condition and the expected value function at hand, we can proceed further and solve the household problem backwards for ages $j=J-1, J-2, \dots, 1$. The structure of the subroutine `solve_household` thereby is identical to the one in Program 10.1.a. The central difference to the baseline model is that in addition to determining optimal consumption $c(z)$ and savings $a^+(z)$, we also have to calculate optimal labour supply $l(z)$ at each element of the state space. We therefore have to slightly adapt the first-order condition, which is shown in Module 10.2m. The first-order condition receives as input a level of future savings a^+ , which is communicated by means of the input variable `x_in`. We then calculate the effective wage rate `wh` of the household, which depends on age as well as the permanent and transitory income component. Lastly, we determine the household's available non-labour income and store it in the variable `available`. The remainder of the function simply uses the equations (10.15) to (10.17) successively, with the right-hand side of the first-order condition being interpolated linearly. The computation of the marginal utility of consumption is thereby outsourced to a function `margu`. The solution to the above first-order condition again constitutes the unrestricted choice of the household. To deal with the (tight) borrowing constraint $a^+ \geq 0$, we still have to check whether this solution satisfies the borrowing constraint. If this is not the case, i.e. when $a^+ < 0$, we manually set $a^+ = 0$ and calculate the corresponding labour-supply and consumption decisions from equations (10.15) and (10.16), which have to hold even when the agent is liquidity-constrained. All of this is done in the subroutine `solve_household`, which is not shown at this point.

Aggregation, calibration, and simulation In order to be able to calculate cohort specific averages and variances, we have to again derive the distribution of households over the

⁸ The shape of the first-order condition is therefore identical to the one specified in (10.4).

Module 10.2m The first-order condition with endogenous labour supply

```

function foc(x_in)
[.....]
! calculate tomorrows assets
a_plus = x_in

! calculate the wage rate
wage = w*eff(ij_com)*theta(ip_com)*eta(is_com)

! calculate available resources
available = (1d0+r)*a(ia_com) + pen(ij_com)

! determine labour
if (ij_com < JR) then
    lab_com = min(max(nu + &
        (1d0-nu)*(a_plus - available)/wage , 0d0), 1d0-1d-10)
else
    lab_com = 0d0
endif

! calculate consumption
cons_com = max(available + wage*lab_com - a_plus, 1d-10)

! calculate linear interpolation for future part of foc
a_plus = max(a_plus, a_1)
call linint_Grow(a_plus, a_1, a_u, a_grow, NA, ial, iar, varphi)

tomorrow = varphi*RHS(ij_com+1, ial, ip_com, is_com) + &
    (1d0-varphi)*RHS(ij_com+1, iar, ip_com, is_com)

! calculate first-order condition for consumption
foc = margu(cons_com, lab_com)**(-gamma) - tomorrow

end function

```

state space. The computational algorithm thereby follows the same logic as in our baseline model. Note, however, that, since we assume a strict borrowing limit of $a^+ \geq 0$, our first element of the discretized asset grid is $\hat{a}_0 = 0$ and therefore the initial distribution of households at age 1 can simply be computed from

$$\phi(1, 0, \hat{\theta}_i, \hat{\eta}_g) = \begin{cases} \pi_i^\theta & \text{if } g = \frac{m+1}{2} \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

π_i^θ denotes the distribution of households over permanent productivity levels $\hat{\theta}_i$. In our simplified case we have $\pi_i^\theta = 0.5$. Given the distribution of households we can compute cohort-specific averages of labour income and labour hours as

$$\bar{y}_j = w \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi(z) \cdot h(z) l(z) \quad \text{and} \quad \bar{l}_j = \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi(z) \cdot l(z),$$

with $z = (j, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g)$.

To be able to simulate the model we only have to specify the new parameter ν , which governs the preference of the household for consumption as compared to leisure. We set $\nu = 0.335$, which leads labour hours to average at approximately one third of the time endowment. Note that as a consequence of endogenous labour-supply decisions, labour income will be much smaller compared to our baseline model in which each household used its total time endowment as labour supply. Consequently, maximum savings will also be smaller than in our baseline model so that we can reduce the upper bound of the asset grid to $a_u = 200$.

Figure 10.2 shows the life-cycle profiles resulting from the endogenous labour-supply model. There are a couple of things to observe here. Next to labour income exhibiting a much smaller magnitude than in the baseline model, we also find that the labour income profile has more of a hump-shape than in the model with exogenous labour supply. The reason for this is the shape of the life-cycle profile of labour hours. Households work quite a lot at young ages, because they want to accumulate precautionary savings to insure against negative productivity shocks. Later on those who were lucky enough in the labour market to accumulate a lot of savings will then reduce their labour hours

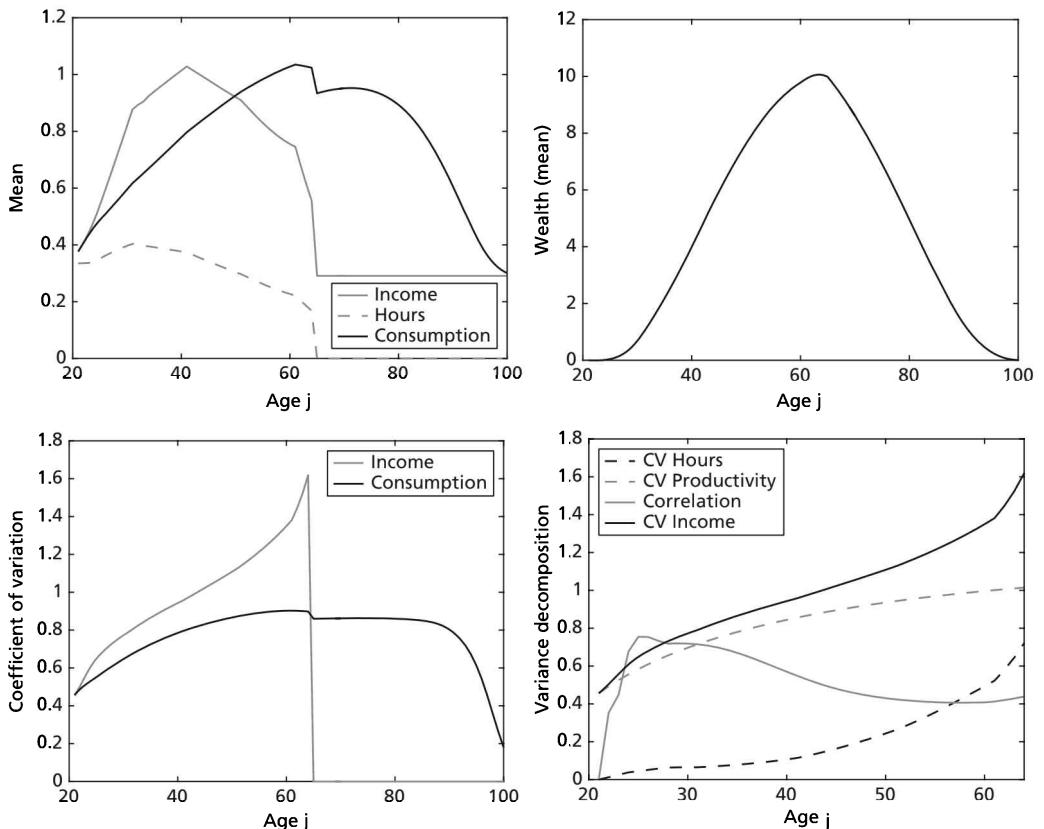


Figure 10.2 Life-cycle profiles with endogenous labour supply

in the years prior to retirement owing to a wealth effect. We also find that average consumption drops when individuals reach mandatory retirement. This is because leisure and regular consumption are (imperfect) substitutes. When leisure suddenly increases with entering retirement, the household reduces regular consumption in order to smooth out utility intertemporally.

The lower left panel of Figure 10.2 sheds light on the variability of labour income and consumption. The picture is similar to the one in Figure 10.1 in the sense that consumption inequality increases much less rapidly than labour-income inequality over the life cycle. Again, individuals partly self-insure against the risk of income fluctuations by means of precautionary savings. What stands out, however, is that the CV of labour income is much larger in this model than in the baseline version with exogenous labour supply. The lower right panel of Figure 10.2 reveals the sources of this observation. The dashed grey line in this graph shows the CV of labour productivity h_j at different stages of working life. By construction, the CV of labour productivity in this model equals the CV of total labour income in the baseline model with inelastic labour supply. The variation of labour income in excess of a variation in productivity is driven by the variability in labour hours. In fact, at the beginning of the life cycle, the coefficient of labour hours is fairly small, indicating that households just work a lot irrespective of their labour productivity in order to be able to build up precautionary savings. At older ages, however, when the lucky households have accumulated enough assets and reduce their labour supply, the variation of labour hours increases and so does the CV of labour income. Note, however, that a positive variation in labour hours does not necessarily lead to a rise in the variance of labour income. If, for example, individuals with a low labour productivity worked substantially more than individuals with a high labour productivity, a positive variation in working hours could also decrease the CV of labour income. However, as the grey line in the lower right panel of Figure 10.2 indicates, labour hours and labour productivity exhibit a positive correlation over all of the household's working life.

10.1.3 FEMALE LABOUR-FORCE PARTICIPATION

In Section 10.1.2 we modelled the labour-supply decision of a household in a rather abstract way by assuming that the household is some entity that provides labour hours to the market and has full control over exactly how many hours to work. In reality, labour-supply decisions are often more complicated and typically involve a decision about whether to participate in the labour market at all. Empirically, the members of a household that exhibit the higher variability in labour-force participation over the life course are typically women, and especially women with children. In the following, we want to focus explicitly on the determinants of the labour-force-participation decision of women in a family context, bringing into focus the conflict between career planning on the one hand and caring for children on the other.

We therefore assume that a household consists of two same-aged adult members, a husband m and a wife f . For simplicity, the husband always works fulltime, meaning that his labour hours are equal to 1. He is equipped with an exogenous labour productivity

$$h_{m,j} = e_j \cdot \exp [\theta + \eta_{m,j}],$$

which is due to the same exogenous age-productivity profile e_j as in the previous models, a household-specific permanent productivity component θ and an individual-specific autoregressive component $\eta_{m,j}$. Supplying $h_{m,j}$ units of productive labour to the market, the husband generates a labour income $y_{m,j} = w_m h_{m,j}$, where w_m denotes the wage rate of men. Labour productivity drops to zero at the exogenous retirement age j_r and the household starts receiving a pension, which like in the previous models is calculated from

$$\text{pen}_j = \begin{cases} 0 & \text{if } j < j_r \quad \text{and} \\ \kappa \cdot \frac{w_m}{j_r - 1} \cdot \sum_{j=1}^{j_r-1} e_j & \text{if } j \geq j_r. \end{cases}$$

Consequently, the wife's labour income has no impact on the size of the pension the household receives.

The female household member is equipped with an initial level of human capital $h_{f,1}$. At each point in time she has to decide whether to participate in the labour market $l_j = 1$ or whether to stay at home $l_j = 0$. Participation in the labour market generates positive spillovers to future periods in the form of experience effects. Specifically, we assume that the human capital of the woman evolves over time according to

$$\log(h_{f,j+1}) = \max [\log(h_{f,j}) + (\xi_1 + \xi_2 \cdot j)l_j - \delta_h(1 - l_j), \log(h_{f,1})].$$

We allow the experience effect of one year of work $\xi_1 + \xi_2 \cdot j$ to be age-specific. In addition, a year of staying at home causes a depreciation of human capital at rate δ_h . Last but not least, human capital cannot fall below its initial level of $h_{f,1}$. In addition to being a function of her own human capital, the labour income a woman earns from participating in the labour market depends on the household-specific permanent productivity component θ , an individual-specific autoregressive shock $\eta_{f,j}$ as well as the wage rate of women w_f . In total, we can write female labour income as

$$y_{f,j} = w_f \cdot h_{f,j} \cdot \exp [\theta + \eta_{f,j}] \cdot l_j. \tag{10.19}$$

A couple has a total of n_c children. To keep things as simple as possible, the household does not decide the number of children nor their spacing. In addition, we assume that all households have the same number of children at exactly the same time. This facilitates the problem substantially in the sense that the number of children does not become an element of the state space of the household. Instead, the presence, age, and number

of children exogenously varies with age. To understand this in more detail, let $\{b_k\}_{k=1}^{n_c}$ indicate the birth years of different children. We assume that a child lives in the household for 18 years. Then we can infer the number of children living in the household at each date j from calculating

$$n_j = \sum_{k=1}^{n_c} \mathbb{1}(b_k \leq j \leq b_k + 17),$$

where $\mathbb{1}$ denotes the indicator function, that is

$$\mathbb{1}(b_k \leq j \leq b_k + 17) = \begin{cases} 1 & \text{if } b_k \leq j \leq b_k + 17 \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The effect of children in this model is twofold. On the one hand, they consume resources. Specifically, we assume that there is a simple mapping from the household's total consumption expenditure c_j to the consumption level per household member $\frac{c_j}{\sqrt{2+n_j}}$. This simple functional form takes into account that a larger number of household members needs more resources in order to attain the same consumption standard. However, these resources need not rise linearly in proportion to the number of household members. Instead we allow for some economies of scale.⁹ The second effect of children is related to the labour-force-participation decision of women. Whenever the female household member decides to work, the household will have to organize someone to care for the children. We assume that the cost of these childcare measures (be it a nanny, a nursery, or something similar) is given by p_j . We will specify these costs in more detail at a later point. The only thing we need to know at the moment is that these costs depend on the number and ages of children present in the household. Since, however, children are an exogenous component of the model, we can safely assume that the (potential) costs of childcare evolve exogenously and can be fully characterized by an age-specific variable.¹⁰

Husband and wife decide jointly how much to consume c_j , how much to save for the future a_{j+1} , and whether the wife should participate in the labour market or not. The household has preferences over consumption and labour supply of the wife l_j , which can be represented by a time-separable expected utility function of the form

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) u(c_j, l_j) \right],$$

⁹ A household with more members, for example, only needs a larger fridge and a larger house, but not two or three.

¹⁰ The reasoning applied here is the same as for the total number of children n_j , which also turns out to only depend on the household's age.

where instantaneous utility is given by

$$u(c_j, l_j) = \frac{1}{1 - \frac{1}{\gamma}} \left[\frac{c_j}{\sqrt{2 + n_j}} \right]^{1 - \frac{1}{\gamma}} - \nu \cdot l_j.$$

There are two things to note here. First, with this formulation we implicitly assume that survival from one period to the next is stochastic but both partners die exactly at the same date. Second, labour-force-participation of the wife only creates a fixed utility cost ν and has no effect on the household's marginal utility of consumption. The joint dynamic budget constraints, under which the partners maximize the above utility function, read

$$a_{j+1} + c_j = (1 + r)a_j + y_{m,j} + y_{f,j} + pen - p_j \cdot l_j,$$

where female labour income is defined as in (10.19).

The dynamic programming problem Before we can suitably define the dynamic programming problem of the household, we should think about what exactly the elements of the state space are. As in previous problems we need, of course, to keep track of the household's age j , its total amount of savings a_j , as well as the stochastic components of labour income θ , $\eta_{m,j}$, and $\eta_{f,j}$. However, in this model there is an additional variable to be included in the state space, which is the human capital $h_{f,j}$ of the female partner that evolves over time with the female labour-force-participation decision. Summing up, we can write the household's state vector as

$$z = (j, a, h_f, \theta, \eta_m, \eta_f)$$

and formulate the dynamic programming problem as

$$\begin{aligned} V(z) &= \max_{c, l, a^+} u(c, l) + \beta \psi_{j+1} E[V(z^+) | \eta_f, \eta_m] \\ \text{s.t. } & a^+ + c = (1 + r)a + y_m + y_f + pen - p_j \cdot l, \quad a^+ \geq 0, \\ & \log(h_f^+) = \max [\log(h_f) + (\xi_1 + \xi_2 \cdot j)l - \delta_h(1 - l), \log(h_{f,1})], \\ & \eta_m^+ = \rho \eta_m + \epsilon_m^+ \quad \text{with} \quad \epsilon_m^+ \sim N(0, \sigma_\epsilon^2), \\ & \eta_f^+ = \rho \eta_f + \epsilon_f^+ \quad \text{with} \quad \epsilon_f^+ \sim N(0, \sigma_\epsilon^2), \end{aligned} \tag{10.20}$$

where $z^+ = (j+1, a^+, h_f^+, \theta, \eta_m^+, \eta_f^+)$. Note that we let the stochastic processes η_m and η_f have identical autocorrelations and variances. In addition, we assume that the processes are independent.

Savings and labour supply The solution to the dynamic programming problem in (10.20) differs from the solution strategy in the previous section, because labour supply

constitutes a discrete choice. Hence, we cannot write down a first-order condition for labour supply, but instead have to solve this problem stepwise. The two steps involved in determining household decisions can be characterized as follows:

1. *Consumption-savings choice*: Let's assume the household had already made a labour-supply decision l . Conditional on this decision, its choice of savings can be determined by solving the conditional optimization problem

$$\begin{aligned}\tilde{V}(z, l) = \max_{c, a^+} \quad & \frac{1}{1 - \frac{1}{\gamma}} \left[\frac{c}{\sqrt{2 + n_j}} \right]^{1 - \frac{1}{\gamma}} - \nu \cdot l + \beta \psi_{j+1} E[V(z^+) | \eta_f, \eta_m] \\ \text{s.t.} \quad & a^+ + c = (1 + r)a + y_m + y_f + pen - p_j \cdot l, \quad a^+ \geq 0\end{aligned}$$

with the human capital and productivity constraints as defined in (10.20). The first-order condition associated with this problem simply reads

$$\frac{c^{-\frac{1}{\gamma}}}{[\sqrt{2 + n_j}]^{1 - \frac{1}{\gamma}}} = \beta \psi_{j+1} E[V_a(z^+) | \eta_f, \eta_m].$$

Using the envelope theorem, we quickly find that

$$V_a(z^+) = (1 + r) \cdot \frac{c(z^+)^{-\frac{1}{\gamma}}}{[\sqrt{2 + n_{j+1}}]^{1 - \frac{1}{\gamma}}}$$

so that we can write the intertemporal first-order condition of the household as

$$\frac{c}{[\sqrt{2 + n_j}]^{1 - \gamma}} = \left\{ \beta \psi_{j+1} (1 + r) \cdot E \left[\frac{c(z^+)^{-\frac{1}{\gamma}}}{[\sqrt{2 + n_{j+1}}]^{1 - \frac{1}{\gamma}}} \middle| \eta_f, \eta_m \right] \right\}^{-\gamma}$$

This first-order condition together with the budget constraint of the household defines levels of savings $\tilde{a}^+(z, l)$ and consumption $\tilde{c}(a, l)$ as well as utility levels $\tilde{V}(z, l)$ that are all conditional on a certain labour-supply decision $l = 0$ or $l = 1$.

2. *Labour-force-participation decision*: Knowing the utility $\tilde{V}(z, l)$ that is associated with each choice of labour supply, the utility maximizing labour-force-participation decision of the household is obviously

$$l(z) = \begin{cases} 1 & \text{if } \tilde{V}(z, 1) \geq \tilde{V}(z, 0) \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

The corresponding levels of savings and consumption then simply have to match the labour-supply decision, that is

$$c(z) = \tilde{c}(z, l(z)) \quad \text{and} \quad a(z) = \tilde{a}(z, l(z)).$$

Numerical implementation In order to solve the above problem numerically, we have to first discretize the state space of the household. As regards the asset dimension, we follow the usual route and define a discrete set of gridpoints $\{\hat{a}_v\}_{v=0}^{n_a}$. The female human-capital dimension can be treated in the same way. Since human capital however evolves linearly in logs, it proves useful to actually define a discrete grid $\{\hat{h}_q\}_{q=0}^{n_h}$ for the log of human capital $\log(h_f)$. In terms of the stochastic components, we apply the same methods for discretizing stochastic variables that we used so far. Note that because we assumed that the transitory productivity risk components of men and women have the same autocorrelation and variance and on top of that are independent, it suffices to discretize the stochastic process η once and use the resulting realizations and transition matrices for both partners. The discretized state vector can be written as

$$z = (j, \hat{a}_v, \hat{h}_q, \hat{\theta}_i, \hat{\eta}_{gm}, \hat{\eta}_{gf}).$$

We initialize the discrete grids for assets and human capital, the stochastic components of the state space, as well as the variables related to children in the subroutine `initialize` which is shown in Program 10.3. Recall that we specify the human-capital grid for $\log(h_f)$, as this evolves linearly over time. We let the initial level of human capital be $h_{f,1} = 1$, which corresponds to $h_l = \log(h_{f,1}) = 0$. As by assumption human capital of the female partner can never drop below its initial level, a value of zero serves as an ideal lower bound of the log human-capital grid. A useful upper end of this grid can be determined by asking what the age-specific levels of log human capital h_j^{\max} of a women would be had she worked full-time during her whole career. Of course, because of the log-linearity of human capital, it is immediately clear that the answer to this question is

$$h_j^{\max} = \sum_{i=1}^{j-1} (\xi_1 + \xi_2 \cdot i).$$

A natural upper bound of the human-capital grid then is the maximum of all these log human-capital levels, i.e. $h_u = \max(h_j^{\max})$.

Next to defining a grid for log human capital, we have to initialize all variables that are related to the number of children and the cost of childcare. Recall that because children are an exogenous factor of the model, children-related variables evolve deterministically over time. To deal with the problem of children, we first specify the birth dates of potential children in an array `children`. This array can have at most 10 entries, where an entry of zero at entry number k means that there is no k th child. In the present specification, the

Program 10.3 Initialization of the labour-force-participation model

```

subroutine initialize
    [.....]
    ! initialize the human capital grid
    h_l = 0d0
    h_max(1) = 0d0
    do ij = 2, JR-1
        h_max(ij) = h_max(ij-1) + xi(1) + xi(2)*(ij-1)
    enddo
    h_max(JR:JJ) = h_max(JR-1)
    h_u = maxval(h_max)

    call grid_Cons_Grow(h, h_l, h_u, h_grow)

    ! define arrival of children
    children = 0
    children(1:2) = (/30, 32/)

    ! set respective child variables
    nchild = 0
    pchild = 0d0
    price = w_f*exp(h_max(10))
    do ic = 1, 10
        if(children(ic) > 20 .and. children(ic) < 50) then
            ! calculate age child is born
            ij = children(ic) - 20

            ! set number of children
            nchild(ij:ij+17) = nchild(ij:ij+17) + 1

            ! set cost of child care
            pchild(ij:ij+2) = pchild(ij:ij+2) + 1.00d0*price
            pchild(ij+3:ij+5) = pchild(ij+3:ij+5) + 0.80d0*price
            pchild(ij+6:ij+11) = pchild(ij+6:ij+11) + 0.60d0*price
            pchild(ij+12:ij+17)= pchild(ij+12:ij+17)+ 0.40d0*price
        endif
    enddo
    [.....]
end subroutine

```

household will therefore give birth to two children, one when the household is aged 30, the other at age 32. Note that for the sake of simplicity, we use a real age notation here. The corresponding model age then equals the real age minus 20. In a next step, we initialize the price of childcare. We assume that the price of one unit of childcare is equal to the wage of a 30-year-old woman who has worked full-time between the ages of 20 and 30, i.e. the price equals $w_f \cdot \exp(h_{10}^{\max})$. Knowing when the children are born and what the price of one unit of childcare is, we can proceed to calculate the total number of children n_j that are present in the household at each date j . Recall that a child lives in the household for exactly 18 years. In addition, we define the (potential) expenditure of the household for childcare in a variable $pchild$. We therefore assume that the cost of childcare is linear in the number of children and depends on the actual age of a child. While for a child at ages 0 to 2 the household has to buy one full unit of childcare, the childcare units needed for older children decline successively with the age of the child.

Program 10.3.a Household problem with labour force participation

```

do ia = 0, NA
[.....]
do ih = 0, ih_max

    ! check whether h(ih) is greater than h_max(ij)
    if(h(ih) > h_max(ij))then
        aplus(ij, ia, ih, :, :, :) = aplus(ij, ia, ih-1, :, :, :)
        c(ij, ia, ih, :, :, :) = c(ij, ia, ih-1, :, :, :)
        l(ij, ia, ih, :, :, :) = l(ij, ia, ih-1, :, :, :)
        V(ij, ia, ih, :, :, :) = V(ij, ia, ih-1, :, :, :)
        cycle
    endif

    do ip = 1, ip_max
        do ism = 1, is_max
            do isf = 1, is_max

                ! determine solution for both working decisions
                do il = 0, il_max
                    [.....]
                    c_temp(il) = cons_com
                    ap_temp(il) = x_in
                    u_temp(il) = valuefunc(x_in, cons_com, il, &
                                         ij, ih, ip, ism, isf)
                enddo

                ! choose the labour force status the gives more utility
                if(u_temp(1) >= u_temp(0))then
                    aplus(ij, ia, ih, ip, ism, isf) = ap_temp(1)
                    c(ij, ia, ih, ip, ism, isf) = c_temp(1)
                    l(ij, ia, ih, ip, ism, isf) = 1d0
                    V(ij, ia, ih, ip, ism, isf) = u_temp(1)
                else
                    aplus(ij, ia, ih, ip, ism, isf) = ap_temp(0)
                    c(ij, ia, ih, ip, ism, isf) = c_temp(0)
                    l(ij, ia, ih, ip, ism, isf) = 0d0
                    V(ij, ia, ih, ip, ism, isf) = u_temp(0)
                endif
            enddo
        enddo
    enddo
[.....]
enddo

```

Solving the household problem The solution to the household problem is again organized in the subroutine `solve_household`, an excerpt of which is shown in Program 10.3.a. We determine the solution to the household optimization problem by backward iteration like in the previous programs of this chapter. Since the terminal condition hasn't changed compared to our baseline problem, the last period's decision-making process is the same as in previous programs. We then iterate backward over ages $j = J - 1, J - 2, \dots, 1$ and compute household decisions for each element z of the state space at a certain age j . We proceed via the following steps: To avoid unnecessary

computational steps, we first check whether the current log human-capital level \hat{h}_q is greater than the maximum possible level h_j^{\max} a female worker could attain at this age by always working full-time. If that is so, we consider the gridpoint invalid and do not need to determine decision rules for \hat{h}_q since nobody will ever end up there. Instead, we simply copy decision rules from the previous grid point \hat{h}_{q-1} . If, however, the human-capital grid point is valid, then we have to determine household decision rules for this grid point as well as any level of assets \hat{a}_v and all potential labour-productivity-shock levels $\hat{\theta}_i$, $\hat{\eta}_{g_m}$, and $\hat{\eta}_g$.¹¹

The variable `i1` controls the potential labour-supply levels for which we need to solve the consumption–savings problem of the household defined in step 1 above. Since the way we set up `fzero` and control for the borrowing constraint as well as the structure of the first-order condition are almost identical to what we did in Program 10.1, we omit showing this part of the subroutine. The outcome of this process is a consumption level stored in `cons_com`, a level of savings `x_in`, as well as a corresponding utility level that we calculate by means of the function `valuefunc`. We store our results in temporary arrays. Having solved the consumption–savings problem for both potential labour-force-participation decisions and having derived the corresponding utility levels, we can calculate the optimal household decision by comparing the entries of the array `u_temp`. If `u_temp(1)` is larger than `u_temp(0)`, then the household derives a higher utility from having the wife work compared to having the wife stay at home and care for the children. We consequently set the labour-supply choice to 1 and copy the respective consumption and savings decision from the temporary arrays.

Interpolation Having determined the optimal decision rules at a certain age, we again need to generate suitable data for the interpolation of the right-hand side of the first-order condition. This is done in the subroutine `interpolate`, a part of which is shown in Program 10.3.b. For the sake of readability, we omitted all the outer do-loop statements. In contrast to how we calculated the right-hand side of the first-order condition in Program 10.1, the variables `RHS` and `EV` now also depend on the potential labour-force-participation decisions of the previous period $j - 1$. Given a level of log human capital \hat{h}_q in this previous period, the labour-supply decision $l \in [0, 1]$ determines log human capital in period j according to

$$h^+ = \max\left(\hat{h}_q + (\xi_1 + \xi_2 \cdot (j - 1)) \cdot l - \delta_h(1 - l), h_l\right).$$

Most likely h^+ will not be an element of the discrete state space $\{\hat{h}_q\}_{q=0}^{n_h}$. Hence, we need to interpolate the consumption function $c(z)$ in order to be able to calculate the

¹¹ Note that the variables ending on `_max` are set like in Program 10.1, so that we do not perform unnecessary calculations throughout the retirement phase.

Program 10.3.b Calculating the right-hand side of the first-order condition

```
RHS(ia, ih, ip, ism, isf, il) = 0d0
EV(ia, ih, ip, ism, isf, il) = 0d0

! interpolate human capital for tomorrow
hplus = h(ih) + (xi(1)+xi(2)*db1e(ij-1))*db1e(il) - del_h*db1e(1-il)
hplus = max(hplus, h_1)
call linint_Grow(hplus, h_1, h_u, h_grow, NH, ihl, ihr, varphi)

! iterate over all potential future states
do ism_p = 1, NS
  do isf_p = 1, NS

    ! right hand side of first-order condition
    chelp = varphi *c(ij, ia, ihl, ip, ism_p, isf_p) + &
             (1d0-varphi)*c(ij, ia, ihr, ip, ism_p, isf_p)
    chelp = max(chelp, 1d-10)
    RHS(ia, ih, ip, ism, isf, il) = &
      RHS(ia, ih, ip, ism, isf, il) + &
      pi(ism, ism_p)*pi(isf, isf_p)*margu(chelp, ij)

    ! expected value function
    [.....]
  enddo
enddo
RHS(ia, ih, ip, ism, isf, il) = &
  ((1d0+r)*beta*psi(ij)*RHS(ia, ih, ip, ism, isf, il))**(-gamma)
EV(ia, ih, ip, ism, isf, il) = &
  (egam*EV(ia, ih, ip, ism, isf, il))**(1d0/egam)
```

marginal utility of consumption. Specifically, we therefore specify interpolation nodes \hat{h}_l and \hat{h}_r , as well as an interpolation weight φ that correspond to the log human-capital level h^+ . We can then calculate the right-hand side of the first-order condition at an element $z = (j, \hat{a}_v, \hat{h}_q, \hat{\theta}_i, \hat{\eta}_{g_m}, \hat{\eta}_{g_f})$ of the discrete state space conditional on the labour-force-participation decision of the households as

$$RHS(z, l) = \left\{ \beta \psi_{j+1}(1+r) \sum_{g_m^+=1}^m \sum_{g_f^+=1}^m \pi_{g_m g_m^+} \pi_{g_f g_f^+} \cdot \frac{(c^+)^{-\frac{1}{\gamma}}}{[\sqrt{2+n_{j+1}}]^{1-\frac{1}{\gamma}}} \right\}^{-\gamma}$$

where

$$c^+ = \varphi \cdot c(z_l^+) + (1 - \varphi) c(z_r^+)$$

and

$$z_l^+ = (j, \hat{a}_v, \hat{h}_l, \hat{\theta}_i, \hat{\eta}_{g_m^+}, \hat{\eta}_{g_f^+}) \quad \text{and} \quad z_r^+ = (j, \hat{a}_v, \hat{h}_r, \hat{\theta}_i, \hat{\eta}_{g_m^+}, \hat{\eta}_{g_f^+}).$$

The same logic can be applied to determine the expected value function at the very same element of the state space.¹²

There is one thing to note here. When we interpolate the consumption function, the labour-force-participation decision can actually be different between the points z_l^+ and z_r^+ . If that is the case, then the actual consumption policy function can exhibit a discontinuous jump between the human-capital gridpoints \hat{h}_l and \hat{h}_r . By using a linear interpolation scheme, we approximate this discrete jump by a continuous line, which introduces some error into the problem, meaning that Euler equation errors will increase. While we are aware of this problem, we decided to live with it at this point for the sake of simplicity. If we wanted to be more accurate, we could simply increase the number of discretization points on both the human-capital and the asset state space. Alternatively, we could design a better interpolation algorithm that explicitly accounts for the next period's labour supply.

The distribution of households over the state space Knowing optimal household decisions $c(z)$, $l(z)$, and $a^+(z)$, the last thing we need to do before we can calculate cohort-specific averages and variances is to determine the distribution of households over the state space, which again is organized in the subroutine `get_distribution`. Program 10.3.c shows how this is done. Note that we again dropped all the do-loop statements for clarity. The aim of this subroutine is to combine the distribution of households at age $j - 1$ as well as the decision rules at the respective discrete elements of the state space z in order to generate a distribution of households at age j . As in previous programs, we make use of a linear interpolation scheme to accomplish this. Specifically, we use the last period's savings decision $a^+(z)$ to derive interpolation nodes \hat{a}_l and \hat{a}_r as well as an interpolation weight φ_a such that

$$a^+(z) = \varphi_a \cdot \hat{a}_l + (1 - \varphi_a) \hat{a}_r.$$

In the same way, we use the human capital level \hat{h}_q and the labour-supply decision $l(z)$ to calculate a current level of human capital h^+ , for which we can also derive interpolation nodes \hat{h}_l and \hat{h}_r and a corresponding interpolation weight φ_h .¹³ We then generate the

¹² Note that there is a small difference in the definition of both `RHS` and `EV` compared to previous programs. In fact, we dropped the age index $i|j$ from these variables. This is not a problem since both arrays are used only at a certain age j to determine decision rules and can then safely be overwritten. We made this choice because the state space of the present model is quite large and therefore the 'full' arrays of this program consume a lot of computer RAM space. Dropping the age indices from `RHS` and `EV` is a way to maintain the space restrictions of the compiler.

¹³ We assume that the human capital of the wife remains constant after the household has entered retirement. Since human capital no longer matters economically once the household is retired, this turns out to be the most practicable choice.

Program 10.3.c Determining the distribution across the asset state space

```

! interpolate yesterday's savings decision
call linint_Grow(aplus(ij-1, ia, ih, ip, ism, isf), a_l, a_u, &
                 a_grow, NA, ial, iar, varphi_a)

! restrict values to grid just in case
ial = min(ial, NA)
iar = min(iar, NA)
varphi_a = min(varphi_a, 1d0)

! get today's human capital
labour = l(ij-1, ia, ih, ip, ism, isf)
htoday = h(ih) + (xi(1)+xi(2)*db1e(ij-1))*labour - del_h*(1d0-labour)
htoday = max(htoday, h_l)
call linint_Grow(htoday, h_l, h_u, h_grow, NH, ihl, ihr, varphi_h)
ihl = min(ihl, NA)
ihr = min(ihr, NA)
varphi_h = min(varphi_h, 1d0)

if(ij >= JR)then
    ihl = ih
    ihr = ih
endif

! redistribute households
do ism_p = 1, NS
    do isf_p = 1, NS
        phi(ij, ial, ihl, ip, ism_p, isf_p) = &
            phi(ij, ial, ihl, ip, ism_p, isf_p) + &
            pi(ism, ism_p)*pi(isf, isf_p)* &
            varphi_a*varphi_h*phi(ij-1, ia, ih, ip, ism, isf)
        phi(ij, iar, ihl, ip, ism_p, isf_p) = &
            phi(ij, iar, ihl, ip, ism_p, isf_p) + &
            pi(ism, ism_p)*pi(isf, isf_p)* &
            (1d0-varphi_a)*varphi_h*phi(ij-1, ia, ih, ip, ism, isf)
        phi(ij, ial, ihr, ip, ism_p, isf_p) = &
            phi(ij, ial, ihr, ip, ism_p, isf_p) + &
            pi(ism, ism_p)*pi(isf, isf_p)* &
            varphi_a*(1d0-varphi_h)*phi(ij-1, ia, ih, ip, ism, isf)
        phi(ij, iar, ihr, ip, ism_p, isf_p) = &
            phi(ij, iar, ihr, ip, ism_p, isf_p) + &
            pi(ism, ism_p)*pi(isf, isf_p)* &
            (1d0-varphi_a)*(1d0-varphi_h)* &
            phi(ij-1, ia, ih, ip, ism, isf)
    enddo
enddo

```

distribution of households at age j —using the distribution across the state space at the previous age and the conditional probabilities of the stochastic processes—by distributing corresponding shares onto the four respective grid-point combinations

$$(\hat{a}_l, \hat{h}_l), \quad (\hat{a}_r, \hat{h}_l), \quad (\hat{a}_l, \hat{h}_r), \quad \text{and} \quad (\hat{a}_r, \hat{h}_r)$$

at age j . Having determined the distribution of households over all elements of the state space at each age j , we can calculate cohort-specific variables in the usual way.

Calibration and simulation We take the standard preference parameters, the auto-correlation and variance of the risk processes, survival probabilities, and the labour-productivity profile for the husband directly from Program 10.1. In order to guarantee that the program runs in a reasonable amount of time, we reduce the size of the asset grid to $n_a = 80$ and set $n_h = 60$. Furthermore, we discretize the transitory risk process with $m = 5$ points.¹⁴ We have already covered our parameter choices regarding the presence and caring cost of children in the discussion of Program 10.3 so we will not repeat this discussion here. To account for the fact that the pension payment now serves two individuals, we increase the replacement rate of the system to $\kappa = 0.8$. With respect to the accumulation of human capital, we let $\xi_1 = 0.05312$ and $\xi_2 = -0.00188$. This choice guarantees that the age-productivity profile of women would be (almost) identical to the productivity profile of men provided that women work full-time throughout their working career.¹⁵ In addition, we let the depreciation rate of human capital for women be $\delta_h = 0.074$ and fix the utility cost of labour-force participation at $v = 0.12$, which guarantees a reasonable labour-force-participation pattern. Last but not least, we let the wage rate of men be $w_m = 1$ and assume a gender wage gap of 25 per cent such that $w_f = 0.75$.

Figure 10.3 shows life-cycle profiles generated from our model. The upper left panel plots total income of the households (grey line) as well as the consumption expenditure profile (black line). Upon the arrival of the first child at age 30, labour income of the household drops as a large proportion of women stop working and stay home to care for the child. At the same time overall consumption expenditure \bar{c}_j of the household rises because parents need to spend additional resources to meet the consumption needs of their children. The expansion in expenditure, however, cannot make up for the increase in the number of hungry mouths to feed, so that per capita consumption $\frac{\bar{c}_j}{\sqrt{(2+n_j)}}$ falls at these ages. The right-hand side of Figure 10.3 shows the labour-income profiles of men and women over the life cycle. Due to the existence of a gender wage gap, women earn less than men even at early stages of the life cycle when almost all women work full-time and therefore have about the same level of human capital as men. Upon the arrival of children, the labour earnings of men and women diverge substantially. On the one hand, this is because a large proportion of women drop out of the labour force to care for the children at home. On the other hand, once out of the labour force women's human capital depreciates and they have no chance of accumulating human capital through experience effects. The lower panels of the figure clarify this mechanism. In fact, while almost 100 per cent of women are working up to age 30, more than 50 per cent decide to withdraw from the workforce upon the arrival of their first child. This has an immediate effect on the

¹⁴ Note that it is straightforward to increase the number of points. This would mainly increase the execution time of the program. At a certain point, one might, however, also violate compiler restrictions with respect to the maximum size of arrays.

¹⁵ One can verify this by setting the utility cost of labour-force participation to zero and assuming that the household has no children.

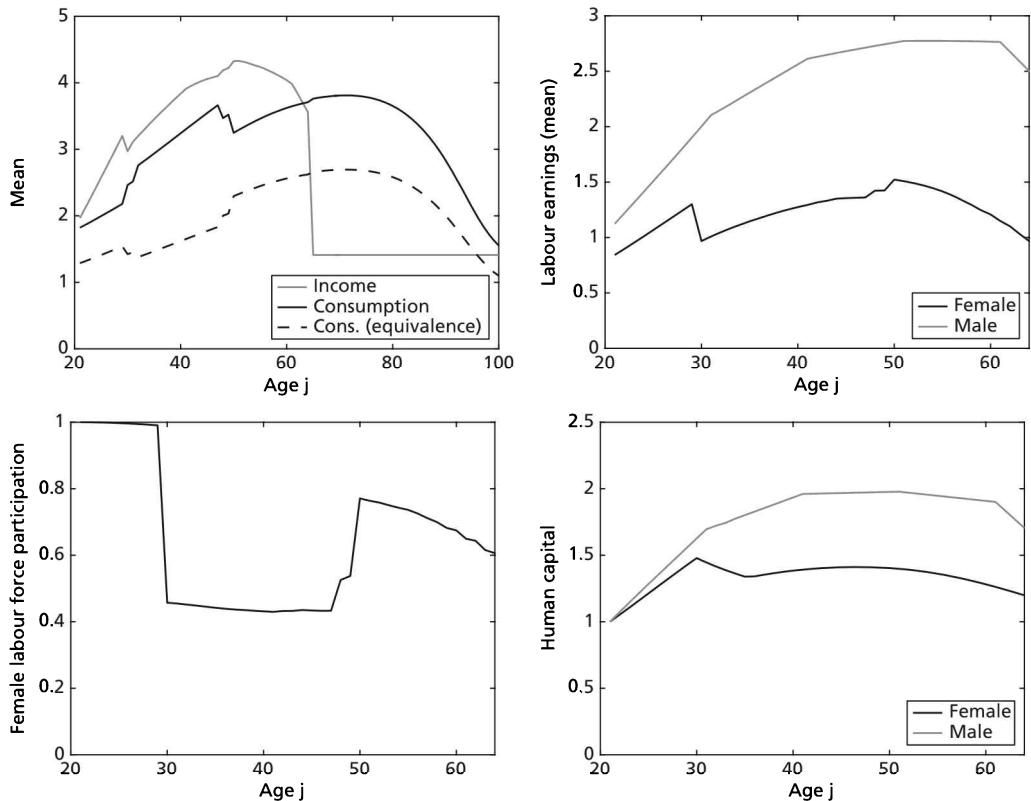


Figure 10.3 Life-cycle profiles with female labour-force participation

human-capital accumulation process of women, which can be seen from the lower right panel. Note that human-capital depreciation seems to stop after the age of 35. This is because most women revert back to their initial human-capital level $h_{1,f}$ below which their human capital cannot fall.

Figure 10.4 shows the corresponding savings behaviour of the household. Since households need a lot of resources for consumption expenditure when the children have arrived, they will start saving up for this time quickly at the beginning of the life cycle. Once the children are there, the accumulating of savings slows down substantially because the couple uses most of the available income for consumption expenditure. Only when the children have left, are the resources of the households around the age of 50 freed up and the household can start accumulating funds quickly for retirement. Because of the rapid buildup of savings at early stages of the life cycle, borrowing constraints are not binding at this phase of the life cycle. This is in sharp contrast to what we found in our baseline model. Instead, liquidity constraints might start binding for some couples when their children have become older but are still living in the household. At this point, at least some partners have exhausted their savings and run into the borrowing constraint.

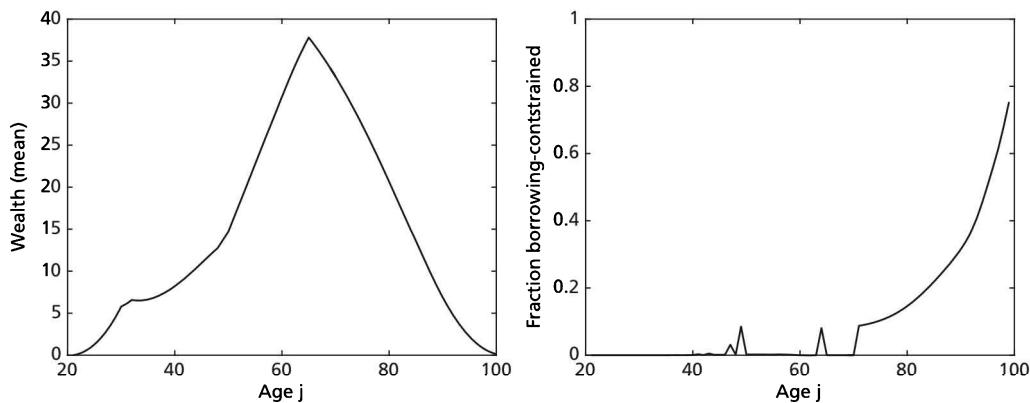


Figure 10.4 Savings in the presence of children

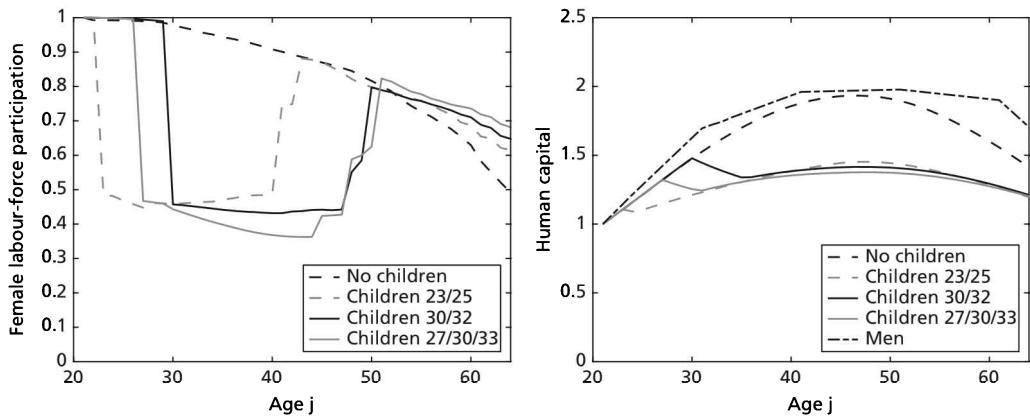


Figure 10.5 Labour supply and human capital with children

Last but not least, Figure 10.5 shows the labour-force-participation patterns and human-capital profiles of women under different assumptions about the number and timing of children. Note that due to the utility costs of participating in the labour force, even women who have no children will slowly start dropping out of the labour force at older ages. This is especially so for women who are unlucky in the labour market, but have a husband who experiences a positive labour-productivity shock. As a result, the average labour-productivity profile of non-mothers still lies below the productivity profile of men. The labour-force-participation patterns of mothers exhibit many similarities even when the timing and number of children is different. Upon the arrival of a child, a large proportion of mothers withdraw from the labour force and therefore the human-capital accumulation process of these mothers stops. When the children leave the family, mothers return to the labour market. What stands out in this picture is that the labour-force participation of mothers at older ages is larger than that of non-mothers. The reason for this behaviour is that couples without children had the option to accumulate

much more in savings during early working years compared to couples with two or more children, see again Figure 10.4. The associated wealth effect therefore pushes non-mothers out of the labour force prior to the mandatory retirement age. Interestingly, this pattern is very similar to what we found in our model with variable labour supply. Mothers, on the other hand, have to continue working until retirement in order to compensate for the resources that have been spent on children.

At this point we want to end the discussion of savings and labour-supply behaviour over the life cycle of a household. We will come back to this theme in Chapter 10, in which we discuss the general equilibrium extension of the life-cycle model with idiosyncratic productivity risk, the stochastic OLG model. The Section 10.2 focuses on optimal investment strategies along the life cycle, when the household has multiple investment opportunities that differ in returns and risk properties.

10.2 Portfolio choice and retirement savings

Up to now we have only been concerned with shocks to individual labour productivity. We thereby assumed that individuals can save in a single, risk-free financial asset. However, in reality individuals have to not only decide how much to save, but also in which kinds of assets to invest. Different asset categories typically feature specific risk-return characteristics. In Chapter 4 we studied optimal portfolio compositions with many investment opportunities in a static model, where an investor makes a one-time choice about her portfolio structure. This section extends the dynamic portfolio choice analysis in a life-cycle model from Chapter 5. While we were previously restricted to a very stylized setup, we can now apply dynamic programming techniques to analyse optimal investment behaviour at many stages of the life cycle. This is an important issue since it has far-reaching consequences for optimal policy design, for example, the management of retirement funds, the effectiveness of saving incentives, and the regulation of private pensions. To this end, we extend the baseline life-cycle model of the Section 10.1 to incorporate both a risk-free and a risky investment opportunity. To keep things tractable, we exclude endogenous labour-supply choices, but assume that labour earnings follow an exogenous risk process.

10.2.1 A MODEL WITH STOCKS AND BONDS

In this section, we take the simplest and most straightforward extension of the baseline life-cycle model by assuming that the household has the opportunity to invest in two types of assets: a bond b_j that offers a risk-free return r_f and stocks s_j that pay a risky

return $r(\vartheta)$, which itself is a function of some underlying shock ϑ .¹⁶ At each age j , households have to decide how much to invest in both alternatives. As we did in Section 10.1, let us define the total amount of invested wealth a_j and the share of stock holdings in the total portfolio ω_j as

$$a_j = b_j + s_j \quad \text{and} \quad \omega_j = \frac{s_j}{a_j},$$

respectively. The total value of savings in the next period can then be written as

$$(1 + r_f) \cdot b_j + (1 + r(\vartheta)) \cdot s_j = [1 + r_f + \omega_j(r(\vartheta) - r_f)] \cdot a_j =: R_p(\omega_j, \vartheta) \cdot a_j,$$

where $r(\vartheta) - r_f$ is the excess return of the risky asset over the risk-free rate. We assume that the relation between r and ϑ is

$$r(\vartheta) - r_f = \mu_r + \vartheta,$$

where ϑ is independent over time. Hence, μ_r defines the risk premium of investing in stocks.

At each age prior to retirement, the household earns some labour income which is the product of the wage rate w and labour productivity

$$h_j = e_j \cdot \exp(\eta_j + \zeta_j).$$

Labour productivity again follows some deterministic profile e_j over the life cycle and is due to two types of productivity shocks: a *white noise*, meaning a transitory component ζ_j that is normally distributed and independent across different ages, and a permanent component η_j . At age j_r the household retires and e_j and therefore labour productivity h_j drop to zero. We assume that throughout retirement we have $\eta_{j+1} = \eta_j$. In this phase the agent receives some pension income pen_j which we assume to be a constant fraction of the last working period's permanent income, i.e.

$$pen_j = \begin{cases} \kappa \cdot w e_{j_r-1} \cdot \exp(\eta_{j_r-1}) & \text{for } j \geq j_r \\ 0 & \text{otherwise.} \end{cases}$$

Throughout this section we assume η to follow a *random walk* and define it as

$$\eta_j = \eta_{j-1} + \epsilon_j.$$

¹⁶ We can, for example, view s_j as already being the efficient portfolio, see Chapter 4. Alternatively, s_j could be an investment fund that combines multiple-risk investment opportunities.

ϵ_j is the idiosyncratic innovation to the random walk and is normally distributed with mean zero. We assume that the innovation to the random walk is correlated with the interest-rate risk factor ϑ .¹⁷ Note that at this point we substantially depart from our previous modelling of labour-productivity risk. Up to now we assumed that labour productivity follows a mean-reverting AR(1) process and that there is no purely transitory component ζ_j . Estimates of the autocorrelation parameter, however, are typically very close to 1. Hence, one can argue that assuming a random walk for productivity risk is equally good as an autoregressive process. Since the life-cycle finance literature seems to have settled on making the random walk assumption their ‘industry standard’, we will also use it in this section. The random walk assumption will allow for a great simplification of the optimization problem as will be explained below. In total, the above investment model features three components of risk that are all normally distributed

$$\zeta_j \sim N(0, \sigma_\zeta^2) \quad \text{and} \quad \begin{bmatrix} \epsilon_j \\ \vartheta \end{bmatrix} \sim N \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_\epsilon^2 & \rho\sigma_\epsilon\sigma_\vartheta \\ \rho\sigma_\epsilon\sigma_\vartheta & \sigma_\vartheta^2 \end{bmatrix} \right). \quad (10.21)$$

The dynamic programming problem At each age j , households can choose consumption c_j as well as the optimal investment strategy a_j and ω_j to maximize standard time-separable preferences

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) u(c_j) \right].$$

This allows us to write the dynamic optimization problem as

$$\begin{aligned} V(j, a, \omega, \eta, \zeta, \vartheta) = \max_{c, a^+, \omega^+} \quad & u(c) + \beta \psi_{j+1} E[V(j+1, a^+, \omega^+, \eta^+, \zeta^+, \vartheta^+) \mid \eta] \\ \text{s.t.} \quad & R_p(\omega, \vartheta)a + wh + pen = c + a^+ \\ & a^+ \geq 0, \quad 0 \leq \omega^+ \leq 1 \\ & \eta^+ = \eta + \epsilon^+ \end{aligned}$$

with the distribution assumptions in (10.21). Note that η is the only shock that is not independent over time because of its random walk structure. Hence, the expectation of tomorrow’s value function has to be formed conditionally on the realization of η .

¹⁷ More formally, we could assume that ϵ_j is due to a purely idiosyncratic component ξ_j and an aggregate component χ , so that $\epsilon_j = \xi_j + \chi$. Then χ would be the common aggregate risk factor that is correlated with interest-rate risk ϑ and that hence causes labour incomes to decline in times when the economy is in bad shape and stock markets don’t perform well. For simplicity we just take a shortcut by assuming that ϵ_j is directly correlated with ϑ .

Formulation in cash-on-hand The above problem obviously features a very large state space, which is not very desirable from a computational perspective. To overcome this problem we will simplify the problem in two steps without losing any of the features. The first step in doing so is to define what the literature calls *cash-on-hand* as

$$X = R_p(\omega, \vartheta)a + wh + pen.$$

This variable summarizes all relevant information about the amount of wealth a , the composition of the household portfolio ω , as well as the interest and labour income shocks at a certain age j that we need in order to determine the choices of the household.¹⁸ Since all shocks but η are independent across time, their exact values need not be included in the state space. In addition, the information about the portfolio composition ω can be dropped as well so that the dynamic programming problem of the household simplifies to

$$\begin{aligned} V(j, X, \eta) &= \max_{c, a^+, \omega^+} u(c) + \beta \psi_{j+1} E[V(j+1, X^+, \eta^+) \mid \eta] \\ \text{s.t. } & X = c + a^+, \quad a^+ \geq 0, \quad 0 \leq \omega^+ \leq 1 \\ & X^+ = R_p(\omega^+, \vartheta^+)a^+ + wh^+ + pen^+ \\ & \eta^+ = \eta + \epsilon^+. \end{aligned}$$

Normalization by permanent productivity While this redefinition of the optimization problem has already substantially reduced the dimension of the state space, some further simplification is still possible. This method relies on the assumption that all *endowment variables* are proportional to the permanent productivity shock $\exp(\eta)$. Since agents in our model start their life with zero assets, the only endowment of the household is income from labour and pensions. By definition the former has to be proportional to the productivity shock. By tying pension benefits to the last income earned, proportionality also holds for the latter.¹⁹ In addition to proportionality we require preferences to be homogeneous, which is satisfied by our standard choice of preferences. As a result of these assumptions, policy functions $c(j, X, \eta)$ and $a^+(j, X, \eta)$ are proportional to the permanent productivity shock $\exp(\eta)$, so that we can simplify the above problem further by defining normalized variables $\tilde{x} = \frac{x}{\exp(\eta)}$. Using this normalization, we can write the budget constraint of the household as

$$\begin{aligned} \tilde{X} &= \tilde{c} + \tilde{a}^+ \quad \text{with} \\ \tilde{X} &= \frac{X}{\exp(\eta)}, \quad \tilde{c} = \frac{c}{\exp(\eta)} \quad \text{and} \quad \tilde{a}^+ = \frac{a^+}{\exp(\eta)}. \end{aligned}$$

¹⁸ This is, of course, only true when labour supply is inelastic.

¹⁹ The proportionality would break down, for example, if the government taxed income from labour and/or savings according to a progressive tax schedule.

Since permanent productivity in the next period equals $\exp(\eta^+)$, we can write

$$\tilde{X}^+ = \frac{X^+}{\exp(\eta^+)} = R_p(\omega^+, \vartheta^+) \frac{a^+}{\exp(\eta^+)} + w \cdot \frac{h^+}{\exp(\eta^+)} + \frac{pen^+}{\exp(\eta^+)},$$

with

$$\tilde{h}^+ = \frac{h^+}{\exp(\eta^+)} = e^+ \cdot \exp(\zeta^+) \quad \text{and} \quad \tilde{pen}^+ = \frac{pen^+}{\exp(\eta^+)} = \kappa w e_{j_r-1},$$

since we have $\eta^+ = \eta$ during retirement years, and

$$\frac{a^+}{\exp(\eta^+)} = \frac{a^+}{\exp(\eta)} \cdot \frac{\exp(\eta)}{\exp(\eta^+)} = \frac{\tilde{a}^+}{\exp(\epsilon^+)}.$$

Summing up we can write the future normalized cash-on-hand as

$$\tilde{X}^+ = R_p(\omega^+, \vartheta^+) \frac{\tilde{a}^+}{\exp(\epsilon^+)} + w \tilde{h}^+ + \tilde{pen}^+.$$

Last but not least, we can write the expected utility function as of age j using our preferences as

$$\begin{aligned} U_j &= \frac{c_j^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\sum_{k=j+1}^J \beta^{k-(j+1)} \left(\prod_{i=j+2}^k \psi_i \right) \frac{c_k^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right] \\ &= \frac{(\exp(\eta_j) \cdot \tilde{c}_j)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\sum_{k=j+1}^J \beta^{k-(j+1)} \left(\prod_{i=j+2}^k \psi_i \right) \frac{(\exp(\eta_k) \cdot \tilde{c}_k)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right] \\ &= \exp(\eta_j)^{1-\frac{1}{\gamma}} \frac{\tilde{c}_j^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\sum_{k=j+1}^J \exp(\eta_k)^{1-\frac{1}{\gamma}} \beta^{k-(j+1)} \left(\prod_{i=j+2}^k \psi_i \right) \frac{\tilde{c}_k^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right], \end{aligned}$$

which is where the homogeneity assumption pays off. Consequently, we normalize

$$\begin{aligned} \tilde{U}_j &= \frac{U_j}{\exp(\eta_j)^{1-\frac{1}{\gamma}}} \\ &= \frac{\tilde{c}_j^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\sum_{k=j+1}^J \frac{\exp(\eta_k)^{1-\frac{1}{\gamma}}}{\exp(\eta_j)^{1-\frac{1}{\gamma}}} \cdot \beta^{k-(j+1)} \left(\prod_{i=j+2}^k \psi_i \right) \frac{\tilde{c}_k^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right]. \end{aligned}$$

The expected future value of utility can be rearranged as

$$\begin{aligned}
& E \left[\sum_{k=j+1}^J \frac{\exp(\eta_k)^{1-\frac{1}{\gamma}}}{\exp(\eta_j)^{1-\frac{1}{\gamma}}} \cdot \beta^{k-(j+1)} \left(\prod_{i=j+2}^k \psi_i \right) \frac{\tilde{c}_k^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right] \\
&= E \left[\frac{\exp(\eta_{j+1})^{1-\frac{1}{\gamma}}}{\exp(\eta_j)^{1-\frac{1}{\gamma}}} \cdot \left(\frac{\tilde{c}_{j+1}^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+2} \sum_{k=j+2}^J \frac{\exp(\eta_k)^{1-\frac{1}{\gamma}}}{\exp(\eta_{j+1})^{1-\frac{1}{\gamma}}} \cdot \beta^{k-(j+2)} \left(\prod_{i=j+3}^k \psi_i \right) \frac{\tilde{c}_k^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right) \right] \\
&= E \left[\exp(\epsilon_{j+1})^{1-\frac{1}{\gamma}} \cdot \tilde{U}_{j+1} \right].
\end{aligned}$$

This means that we can ultimately write a normalized version of the above optimization problem as

$$\begin{aligned}
\tilde{V}(j, \tilde{X}) &= \max_{\tilde{c}, \tilde{a}^+, \omega^+} \frac{\tilde{c}^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} \cdot \tilde{V}(j+1, \tilde{X}^+) \right] \quad (10.22) \\
\text{s.t. } \tilde{X} &= \tilde{c} + \tilde{a}^+, \quad \tilde{a}^+ \geq 0, \quad 0 \leq \omega^+ \leq 1 \\
\tilde{X}^+ &= R_p(\omega^+, \vartheta^+) \frac{\tilde{a}^+}{\exp(\epsilon^+)} + w\tilde{h}^+ + \tilde{p}\tilde{e}\tilde{n}^+
\end{aligned}$$

with $\tilde{V}(j, \tilde{X}) = \frac{V(j, X, \eta)}{\exp(\eta)^{1-\frac{1}{\gamma}}}$ and distributional assumptions as in (10.21). Note that we do not need to normalize ω since ω is a relative variable that measures the relationship between two normalized variables, i.e. normalized stock holdings \tilde{s} over total normalized wealth \tilde{a} .

Separation of investment choice When looking at the optimization problem in (10.22), we recognize that it has a particular structure. Specifically, given any level \tilde{a}^+ of savings for the future, the only way the portfolio composition ω^+ influences the household's utility is through altering the return-risk combination of tomorrow and, hence, impacting both the value of future cash-on-hand \tilde{X}^+ and the expected future value function $E \left[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} \cdot \tilde{V}(j+1, \tilde{X}^+) \right]$. Current consumption, on the other hand, is independent of the investment structure ω^+ . Because of this, we can separate the optimization problem of the household into two sub-problems:

1. *Optimal portfolio composition:* For each level of future savings \tilde{a}^+ , we can determine the optimal investment structure $\omega^+(j, \tilde{a}^+)$ by solving

$$\begin{aligned}\tilde{Q}(j, \tilde{a}^+) &= \max_{0 \leq \omega^+ \leq 1} E \left[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} \cdot \tilde{V}(j+1, \tilde{X}^+) \right] \\ \text{s.t. } \tilde{X}^+ &= R_p(\omega^+, \vartheta^+) \cdot \frac{\tilde{a}^+}{\exp(\epsilon^+)} + w\tilde{h}^+ + \tilde{pen}^+,\end{aligned}$$

where due to normalization and independence assumptions, the expected value of \tilde{V} is independent of today's realization of labour productivity and interest-rate shocks. Plugging \tilde{X}^+ into the objective function and taking derivatives, we obtain the first-order condition

$$E \left[(\mu_r + \vartheta^+) \tilde{a}^+ \cdot (\exp(\epsilon^+) \cdot \tilde{c}(j+1, \tilde{X}^+))^{-\frac{1}{\gamma}} \right] = 0 \quad (10.23)$$

where we have used

$$\frac{\partial \tilde{X}^+}{\partial \omega^+} = (\mu_r + \vartheta^+) \frac{\tilde{a}^+}{\exp(\epsilon^+)}$$

and from the envelope theorem that

$$\tilde{V}_X(j+1, \tilde{X}^+) = u'(\tilde{c}(j+1, \tilde{X}^+)) = \tilde{c}(j+1, \tilde{X}^+)^{-\frac{1}{\gamma}}.$$

2. *Consumption-savings choice:* Given the optimal portfolio composition $\omega^+(j, \tilde{a}^+)$, the household decides about how much to consume and save by maximizing

$$\tilde{V}(j, \tilde{X}) = \max_{\tilde{c}, \tilde{a}^+} u(\tilde{c}) + \beta \psi_{j+1} \tilde{Q}(j, \tilde{a}^+) \quad \text{s.t. } \tilde{X} = \tilde{c} + \tilde{a}^+.$$

The respective first-order condition obviously is

$$u'(\tilde{c}) = \beta \psi_{j+1} \tilde{Q}_a(j, \tilde{a}^+) \quad \text{with}$$

$$\tilde{Q}_a(j, \tilde{a}^+) = E_j \left[R_p^+ (\exp(\epsilon^+) \cdot \tilde{c}(j+1, \tilde{X}^+))^{-\frac{1}{\gamma}} \right],$$

so that we finally have

$$\tilde{c}(j, \tilde{X}) = \left(\beta \psi_{j+1} E_j \left[R_p^+ (\exp(\epsilon^+) \cdot \tilde{c}(j+1, \tilde{X}^+))^{-\frac{1}{\gamma}} \right] \right)^{-\gamma} \quad (10.24)$$

with $R_p^+ = R_p(\omega^+(j, \tilde{a}^+), \vartheta^+) = 1 + r_f + \omega^+(j, \tilde{a}^+) (\mu_r + \vartheta^+)$.

Numerical implementation The solution to the above problem is implemented in Program 10.4. The parameters of the model are again directly set in the module `globals`. The remainder of the initialization process, most importantly the discretization of the state space and shock processes, is handled in the subroutine `initialize`, which is called up from the main program before we start the solution process. Approximated versions of the shock processes described in (10.21) consist of value pairs

$$\left(\hat{\zeta}_g, \pi_g^w\right) \text{ for } g = 1, \dots, m_w \quad \text{and} \quad \left(\begin{bmatrix}\hat{\epsilon}_i \\ \hat{\vartheta}_i\end{bmatrix}, \pi_i^{sr}\right) \text{ for } i = 1, \dots, m_{sr},$$

where $\hat{\zeta}_g$, $\hat{\epsilon}_i$, and $\hat{\vartheta}_i$ are discrete sets of potential realizations of shocks and π_g^w and π_i^{sr} the corresponding probabilities for the shock values to occur. The discretized versions of the first-order conditions in (10.23) and (10.24) consequently read

$$\sum_{g^+=1}^{m_w} \sum_{i^+=1}^{m_{sr}} \pi_{g^+}^w \cdot \pi_{i^+}^{sr} \cdot (\mu_r + \hat{\vartheta}_{i^+}) \tilde{a}^+ \cdot (\exp(\hat{\epsilon}_{i^+}) \cdot \tilde{c}(j+1, \tilde{X}^+))^{-\frac{1}{\gamma}} = 0 \quad (10.25)$$

with $\tilde{X}^+ = R_p(\omega^+, \hat{\vartheta}_{i^+}) \cdot \frac{\tilde{a}^+}{\exp(\hat{\epsilon}_{i^+})} + w e^+ \exp(\hat{\zeta}_{g^+}) + \tilde{p} \tilde{e} n^+$

and

$$\tilde{c}(j, \tilde{X}) = \left(\beta \psi_{j+1} \sum_{g^+=1}^{m_w} \sum_{i^+=1}^{m_{sr}} \pi_{g^+}^w \cdot \pi_{i^+}^{sr} \cdot \left[R_p^+ (\exp(\hat{\epsilon}_{i^+}) \cdot \tilde{c}(j+1, \tilde{X}^+))^{-\frac{1}{\gamma}} \right] \right)^{-\gamma} \quad (10.26)$$

with $R_p^+ = 1 + r_f + \omega^+(j, \tilde{a}^+) (\mu_r + \hat{\vartheta}_{i^+})$.

We use the subroutine `normal_discrete` from our toolbox to approximate the normally distributed shocks in our model as described in Chapter 2. In order to discretize the white noise productivity component ζ , we have to hand over two arrays `zeta` and `dist_zeta` in which the subroutine will store the realizations of the white noise component as well as the corresponding probabilities, respectively. The last two arguments we need are the mean and the variance of ζ . To keep later formulas simple, we already apply the exponential function to `zeta` at this stage of the program. The same subroutine can also be used to approximate the two-dimensional shock values ϵ and ϑ that feature a certain correlation ρ . Therefore the subroutine first receives the number of shock values that should be used in any of the two dimensions `NS` and `NR`, such that `NSR = NS*NR`. We then hand over an array `temp` of size `(NSR, 2)` in which the subroutine will store the respective realizations of the ϵ and ϑ shocks as well as an array `dist_epsvtheta` that will be filled up

Program 10.4 Initializing model variables

```

subroutine initialize
[.....]
! discretize zeta shocks
call normal_discrete(zeta, dist_zeta, 0d0, sigma_zeta)
zeta = exp(zeta)

! discretize eps-vtheta shocks
call normal_discrete((/NS, NR/), temp, dist_epsvtheta, &
(/0d0, 0d0/), (/sigma_eps, sigma_vtheta/), rho)
eps(:) = exp(temp(:, 1))
vtheta(:) = temp(:, 2)

! initialize asset grid
call grid_Cons_Grow(a, a_l, a_u, a_grow)

! endogenous lower and upper bound of cash-on-hand grid
X_l = min(w*minval(eff(1:JR-1))*minval(eps(:))*zeta(1), pen(JR))
X_u = (1d0 + r_f + mu_r + maxval(vtheta(:)))*a_u + &
      w*maxval(eff(1:JR-1))*maxval(eps(:))*zeta(NW)
call grid_Cons_Grow(X, X_l, X_u, X_grow)
[.....]
end subroutine

```

with the corresponding probabilities. Again we have to specify means and variances and this time also the correlation between the two random variables. To make things more accessible in later code, we extract the realizations of ϵ and ϑ into two separate arrays of dimension NSR, and again apply the exponential function to `eps`.

Next, we need to provide discrete grids for the continuous dimensions of the state space, that is sets of points $\{\hat{a}_v\}_{v=0}^{n_a}$ and $\{\hat{X}_k\}_{k=0}^{n_X}$. We start out with the grid for assets \hat{a}_v , that is used in determining the optimal portfolio composition. Having specified values for the lower and upper ends of this grid together with a growth rate, we can use the subroutine `grid_Cons_Grow` from the toolbox to fill up the array `a(:)` with corresponding grid values. As for the grid for cash-on-hand, it is quite easy to derive the potential minimum and maximum cash holdings once the asset dimension is specified. In particular for lower and upper values of the asset grid of $a_l = 0$ and a_u we have

$$X_l = \min[w h_j + pen_j] \quad \text{and} \quad X_u = \max[R_p(1, \vartheta)] \cdot a_u + \max[w h_j + pen_j].$$

Once we have calculated the lower and upper bound of the cash-on-hand grid as well as a growth rate, we can again use the toolbox to provide discrete gridpoints.

The household optimization problem The solution to the household optimization problem is organized in subroutine `solve_household`, which is shown in Program 10.4.a. Its structure is similar to the solution of the household problem in Program 10.1. We first compute the optimal behaviour in the last period of life where, owing to the absence of a bequest motive, the household just consumes all the cash it has available and

Program 10.4.a Solving the household optimization problem

```

subroutine solve_household()
    [.....]
    ! get decisions in last period of life
    omega_plus(JJ, :) = 0d0
    do ix = 0, NX
        a_plus(JJ, ix) = 0d0
        c(JJ, ix) = X(ix)
        V(JJ, ix) = valuefunc(0d0, c(JJ, ix), JJ)
    enddo

    do ij = JJ-1, 1, -1
        ! determine optimal portfolio choice for all others
        do ia = 1, NA
            call solve_portfolio(ij, ia)
        enddo

        ! set omega for zero savings consistent with next gridpoint
        omega_plus(ij, 0) = omega_plus(ij, 1)

        ! interpolate individual RHS and value function
        call interpolate(ij)

        ! determine consumption-savings solution
        do ix = 0, NX
            call solve_consumption(ij, ix)
        enddo

        write(*,'(a,i3,a)')'Age: ',ij,' DONE!'
    enddo

end subroutine

```

saves nothing into the next period. The corresponding value function can be computed using the function `valuefunc`, which takes as input the savings decision for tomorrow, a level of consumption, and the age j . Having set the terminal policy and value functions, we iterate backwards in time and successively solve the household problem at each age j . We do this in the two steps described above, namely, we first solve the portfolio optimization problem using the subroutine `solve_portfolio`, then interpolate the right-hand side of the first-order condition in (10.26) by means of the subroutine `interpolate` and finally determine the optimal consumption–savings decision in the subroutine `solve_consumption`. Note that the portfolio optimization problem obviously has no unique solution when individuals save zero assets. To make graphs look smooth, we hence just copy the portfolio share of the lowest positive asset value.

The optimal portfolio composition The subroutine `solve_portfolio`, which solves the household's portfolio optimization problem, is shown in Program 10.4.b. We solve for the optimal allocation of assets into riskless and risky investments at an age j for each potential asset value $\tilde{a}^+ = \hat{a}_v, v = 0, \dots, n_a$. The function `foc_port` evaluates the first-order condition – specifically the left-hand side of equation (10.25) – for some level of

Program 10.4.b Determining portfolio shares

```

subroutine solve_portfolio(ij, ia)
[.....]
! check for corner solutions
port0 = foc_port(0d0)
port1 = foc_port(1d0)

! use intermediate value theorem
if(port0*port1 > 0d0) then
    if(abs(port0) > abs(port1)) then
        omega_plus(ij, ia) = 1d0
    else
        omega_plus(ij, ia) = 0d0
    endif
    return
else

    ! get order of magnitude of foc
    tolerance = 1d-5*abs(port0-port1)
    tolerance = min(tolerance, 1d-8)
    call settol_root(tolerance)

    ! get best guess for the root of foc_port
    x_in = -port0/(port1-port0)
    check = .false.

    ! solve the household problem using root-finding
    call fzero(x_in, foc_port, check)
    [.....]
    omega_plus(ij, ia) = x_in

    ! reset tolerance level to original value
    call settol_root(1d-8)
endif

end subroutine

```

ω^+ .²⁰ The values of age and the respective asset grid point are communicated to this function by means of the communication variables `ij_com` and `ia_com`, respectively. Before we actually try to solve the first-order condition (10.25) using the root-finding routine from our toolbox, we first need to check whether the first-order condition has a root on the interval $\omega^+ \in [0, 1]$. To do this, we take the derivative of the left-hand side of equation (10.25) with respect to ω^+ and obtain

$$\begin{aligned}
& -\frac{1}{\gamma} \cdot \sum_{g^+=1}^{m_w} \sum_{i^+=1}^{m_{sr}} \pi_{g^+}^w \cdot \pi_{i^+}^{sr} \cdot \left[(\mu_r + \hat{\vartheta}_{i^+}) \tilde{a}^+ \right]^2 \cdot \\
& \quad \left(\exp(\hat{\epsilon}_{i^+}) \cdot \tilde{c}(j+1, \tilde{X}^+) \right)^{-\frac{1}{\gamma}-1} \cdot \frac{\partial c(j+1, \tilde{X}^+)}{\partial \tilde{X}^+} < 0.
\end{aligned}$$

²⁰ We do not show this function here since it should be self explanatory. We use linear interpolation to evaluate the consumption function at any future level of cash-on-hand X^+ .

Since consumption in the optimum should be an increasing function of cash-on-hand, i.e. $\frac{\partial c(j+1, \tilde{X}^+)}{\partial \tilde{X}^+} > 0$, this term is negative, which means that the left-hand side of (10.25) is strictly monotonically decreasing in ω^+ . Knowing this, we can check for the existence of an interior solution to the portfolio optimization problem by evaluating the first-order condition at the values $\omega^+ = 0$ and $\omega^+ = 1$. Let f_0 and f_1 denote the corresponding function values at these two corner points. If f_0 and f_1 do have the same sign—which we can formally check by verifying that their product $f_0 \cdot f_1$ is greater than zero—then we know from the monotonicity of the first-order condition that the value of the first-order condition will always be either positive or negative on the entire interval $[0, 1]$. Figure 10.6 illustrates this line of reasoning.

In case A in which $f_0 > 0$ and $f_1 > 0$, it is clear that the actual root of the first-order condition must lie to the right of the interval $[0, 1]$, meaning the unconstrained optimal ω^+ should be greater than 1. Consequently, the household can attain the highest possible utility from choosing the ω^+ that is closest to this unconstrained value. Hence the optimal constrained portfolio allocation must be $\omega^+ = 1$. The same but vice versa is true if $f_0 < 0$ and $f_1 < 0$. Put differently, the household will always choose the corner for which the function value f_0 or f_1 is closer to zero in absolute terms. Only if f_0 and f_1 have opposite signs, as in case B, the intermediate value theorem tells us that the first-order condition has a unique root on the interval $[0, 1]$. We can find this root using the root-finding subroutine `fzero` from the toolbox. The starting guess `x_in` we provide is a simple application of the secant method, see Section 2.2.2.

When applying the subroutine `fzero` there is one issue we need to take care of. Recall that the first-order condition is a function in $c^{-\frac{1}{\gamma}}$ where γ in our model takes on a relatively small value of 0.1, see the discussion about the calibration of the model below. Note further that for a value of consumption of $c = 10$ the scale of the first-order condition (10.25) will consequently be around 10^{-10} , which is actually quite small in

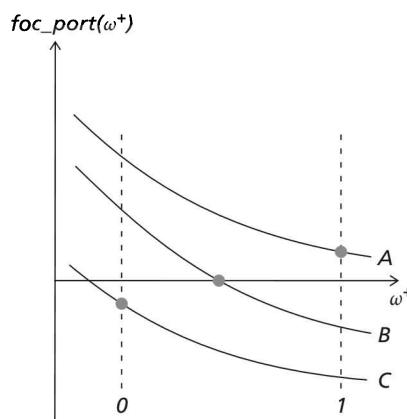


Figure 10.6 Determining corner solutions of portfolio shares

terms of a computer's order of magnitude. Our root-finding routine `fzero` will typically assume that the root of a function is accurately determined whenever the function value lies below 10^{-8} in absolute terms. When this subroutine finds a root of a function—in our case `foc_port`—that has an order of around 10^{-10} across the whole range of $\omega \in [0, 1]$, such an accuracy criterion is useless. To overcome this problem, we determine a ‘useful’ level of accuracy that the subroutine `fzero` should apply by taking the absolute of the difference between the function values of the first-order condition at the left and right interval endpoints. We have already calculated these and stored them in the variables `port0` and `port1`, respectively. Multiplying this difference with 10^{-5} should lead to a suitable tolerance level for the root-finding routine. Sticking with our example if both function values are in the order of 10^{-10} , the root-finder should use a tolerance level of 10^{-15} . We can communicate a new tolerance level to the root-finding routine through the subroutine `settol_root` that is provided by the toolbox. However, we should not forget to reset the tolerance level to its original value after we have solved for the optimal portfolio allocation.²¹

Intermediate interpolation Having determined the optimal portfolio allocations $\omega(j, \hat{a}_v)$ for each level of potential assets $\tilde{a}^+ = \hat{a}_v$, we can now start solving the consumption–savings problem. This involves finding the root of the first-order condition defined in (10.26), which means that we have to evaluate this first-order condition many times. To do this most efficiently, we again recognize that the right-hand side of the first-order condition is a function solely of age j and the level of future savings \tilde{a}^+ . Consequently, we can calculate

$$RHS(j, \hat{a}_v) = \left(\beta \psi_{j+1} \sum_{g^+=1}^{m_w} \sum_{i^+=1}^{m_{sr}} \pi_{g^+}^w \pi_{i^+}^{sr} \left[R_p^+ \left(\exp(\hat{\epsilon}_{i^+}) \tilde{c}(j+1, \tilde{X}^+) \right)^{-\frac{1}{\gamma}} \right] \right)^{-\gamma}$$

with $\tilde{X}^+ = R_p(\omega^+(j, \hat{a}_v), \hat{\vartheta}_{i^+}) \cdot \frac{\hat{a}_v}{\exp(\hat{\epsilon}_{i^+})} + w e^+ \exp(\hat{\zeta}_{g^+}) + \widetilde{pen}^+$

and $R_p^+ = 1 + r_f + \omega^+(j, \hat{a}_v) (\mu_r + \hat{\vartheta}_{i^+})$

for each level of $\hat{a}_v, v = 0, \dots, n_a$. This is done in the subroutine `interpolate`, which follows a very similar structure as the function `foc_port`. Note that in order to be able to calculate $RHS(j, \hat{a}_v)$, we need to evaluate the future consumption function $\tilde{c}(j+1, \tilde{X}^+)$ at some arbitrary level of cash-on-hand. We again use linear interpolation through the subroutine `linint_grow` provided in the toolbox to perform this task. Recall that during

²¹ Note that we will not run into such a problem when solving for the optimal consumption–savings decision by determining the root of equation (10.26), since this equation is formulated in consumption values c and not marginal utilities of consumption.

retirement we have both $\epsilon^+ = 0$ and $\zeta^+ = 0$. Finally, the subroutine `interpolate` also calculates the value of

$$\tilde{Q}(j, \hat{a}_v) = \sum_{g^+=1}^{m_w} \sum_{i^+=1}^{m_{sr}} \pi_{g^+}^w \cdot \pi_{i^+}^{sr} \cdot \exp(\hat{\epsilon}_{i^+})^{1-\frac{1}{\gamma}} \cdot \frac{\tilde{V}(j+1, \tilde{X}^+)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

It therefore linearly interpolates the transformed value function

$$\begin{aligned} \tilde{V}(j+1, \tilde{X}^+) &= \varphi \left[\left(1 - \frac{1}{\gamma}\right) \tilde{V}(j+1, \hat{X}_l) \right]^{\frac{1}{1-\frac{1}{\gamma}}} \\ &\quad + (1 - \varphi) \left[\left(1 - \frac{1}{\gamma}\right) \tilde{V}(j+1, \hat{X}_r) \right]^{\frac{1}{1-\frac{1}{\gamma}}} \end{aligned}$$

in the same way as the consumption function. The function values of $\tilde{Q}(j, \hat{a}_v)$ are ultimately used in the function `valuefunc`.

The consumption–savings problem Having calculated the right-hand side of the first-order condition at the grid points \hat{a}_v , we can finally solve the consumption–savings problem for each level of current cash-on-hand $\tilde{X} = \hat{X}_k, k = 0, \dots, n_X$. This is done in the subroutine `solve_consumption`, see Program 10.4.c. We first check whether the current level of cash-on-hand \hat{X}_k is smaller than 10^{-10} . If that is the case, we set both consumption and savings to zero and directly return to the subroutine `solve_household`. Otherwise we communicate the current age j and the current cash-on-hand gridpoint to the first-order condition `foc_cons` using communication variables. We then set the value of the first-order condition to zero using the root-finding subroutine `fzero` from the toolbox. The outcome of this procedure is an optimal level of savings \tilde{a}^+ stored in the variable `x_in`. Note that we again do not incorporate the borrowing constraint when solving the first-order conditions, but rather derive an unconstrained optimal savings level. If this unconstrained level of savings is negative, then the borrowing constraint is binding. Consequently we set the value of `x_in` to exactly the constraint of zero. The borrowing-constrained agent will obviously consume as much as she can, meaning her consumption level will equal the amount of cash-on-hand she has available. Note that the communication variable `cons_com` was already used in the function `foc_cons` to calculate the optimal unconstrained consumption level. Finally, we copy the optimal decisions into the storage variables for future savings and consumption and determine the value function that results from the optimal choices.

Module 10.4m finally shows how we calculate the first-order condition for the consumption–savings problem. The function `foc_cons` receives as input a level of future savings \tilde{a}^+ and using the communicated level of cash-on-hand directly calculates the resulting consumption level. We then use the subroutine `linint_Grow` to calculate both

Program 10.4.c Solving the consumption–savings problem

```

subroutine solve_consumption(ij, ix)
    [.....]
    ! determine decision for zero cash-on-hand
    if(X(ix) < 1d-10)then
        a_plus(ij, ix) = 0d0
        c(ij, ix) = 0d0
        V(ij, ix) = valuefunc(0d0, 0d0, ij)
        return
    endif

    ! set up communication variables
    ij_com = ij
    ix_com = ix

    ! get best initial guess from future period
    x_in = a_plus(ij+1, ix)
    check = .false.

    ! solve the household problem using root-finding
    call fzero(x_in, foc_cons, check)
    [.....]
    ! check for borrowing constraint
    if(x_in < 0d0)then
        x_in = 0d0
        cons_com = X(ix)
    endif

    ! copy decisions
    a_plus(ij, ix) = x_in
    c(ij, ix) = cons_com
    V(ij, ix) = valuefunc(x_in, cons_com, ij)

end subroutine

```

Module 10.4m The first-order condition of the consumption–savings problem

```

function foc_cons(x_in)
    [.....]
    ! calculate tomorrows assets
    a_plus = x_in

    ! calculate consumption
    cons_com = X(ix_com) - a_plus

    ! calculate linear interpolation for future part of foc
    call linint_Grow(a_plus, a_l, a_u, a_grow, NA, ial, iar, varphi)

    tomorrow = varphi*RHS(ij_com, ial)+(1d0-varphi)*RHS(ij_com, iar)

    ! calculate first-order condition for consumption
    foc_cons = cons_com - tomorrow

end function

```

indices of interpolation nodes \hat{a}_l and \hat{a}_r and an interpolation weight φ , which correspond to the level of assets \tilde{a}^+ . The right-hand side of the first-order condition in (10.26) can then be calculated from interpolating

$$RHS(j, \tilde{a}^+) = \varphi \cdot RHS(j, \hat{a}_l) + (1 - \varphi) \cdot RHS(j, \hat{a}_r).$$

The function `foc_cons` returns the difference between current consumption c and the interpolated right-hand side of the first-order condition, which should be equal to zero under the optimal choice of the household.

The distribution across the state space Solving the household optimization problem yields (discrete) policy functions $\tilde{c}(j, \hat{X}_k)$, $\tilde{a}^+(j, \hat{X}_k)$, and $\omega^+(j, \hat{a}_v)$. To be able to calculate meaningful life-cycle statistics from these policy functions, we need to determine the distribution of households over the state space at each age j . This is complicated by the fact that our model now has two state spaces, one for the beginning of period cash-on-hand X and one for the end of period savings a . We therefore determine distributions $\phi_X(j, X)$ and $\phi_a(j, a)$ for each of the state spaces, respectively. The subroutine `get_distribution` manages the calculation of these distributions. However, the actual codes are located in two separate subroutines. We derive the distribution across the cash-on-hand grid in `get_distribution_X`, parts of which are shown in Programs 10.4.d to 10.4.f.

Program 10.4.d Distribution of households over cash-on-hand grid (Part 1)

```

subroutine get_distribution_X(ij)
    [.....]
    if(ij == 1) then

        ! get initial distribution at age 1 of cash-on-hand
        do iw = 1, NW

            ! get initial cash-on-hand
            X_p = w*eff(1)*zeta(iw)

            ! derive interpolation weights
            call linint_Grow(X_p, X_l, X_u, X_grow, &
                NX, ixl, ixr, varphi)

            ! get distributional weight
            dist = dist_zeta(iw)

            ! initialize the distribution
            phi_X(1, ixl) = phi_X(1, ixl) + dist*varphi
            phi_X(1, ixr) = phi_X(1, ixr) + dist*(1d0-varphi)
        enddo

        elseif(ij <= JR-1) then
            [.....]
        else
            [.....]
        endif

    end subroutine

```

Program 10.4.e Distribution of households cross asset grid

```

subroutine get_distribution_a(ij)
    [.....]
    do ix = 0, NX

        ! interpolate asset decision
        call linint_Grow(a_plus(ij, ix), a_l, a_u, a_grow, &
                           NA, ial, iar, varphi)

        ! restrict values to grid just in case
        ial = min(ial, NA)
        iar = min(iar, NA)
        varphi = min(varphi, 1d0)

        ! get end of period asset distribution
        phi_a(ij, ial) = phi_a(ij, ial) + varphi*phi_X(ij, ix)
        phi_a(ij, iar) = phi_a(ij, iar) + (1d0-varphi)*phi_X(ij, ix)
    enddo

end subroutine

```

Program 10.4.f Distribution of households over cash-on-hand grid (Part 2)

```

! iterate over yesterdays asset distribution
do ia = 0, NA

    ! iterate over current shocks
    do iw = 1, NW
        do isr = 1, NSR

            ! get today's cash-on-hand
            R_port = 1d0 + r_f + omega_plus(ij-1, ia) &
                      * (mu_r + vtheta(isr))
            X_p = R_port*a(ia)/eps(isr) + w*eff(ij)*zeta(iw)

            ! derive interpolation weights
            call linint_Grow(X_p, X_l, X_u, X_grow, &
                               NX, ixl, ixr, varphi)

            ! get distributional weight
            dist = dist_zeta(iw)*dist_epsvtheta(isr)

            phi_X(ij, ixl) = phi_X(ij, ixl) + &
                             dist*varphi*phi_a(ij-1, ia)
            phi_X(ij, ixr) = phi_X(ij, ixr) + &
                             dist*(1d0-varphi)*phi_a(ij-1, ia)
        enddo
    enddo
enddo

```

We start with determining the initial distribution of households over cash-on-hand at age $j = 1$. Households start their life with zero assets. Hence, the only determinant of cash-on-hand is individual income. Normalized cash-on-hand \tilde{X}_1 is defined by the wage rate, the general labour efficiency level at age 1, and most importantly the white noise productivity shock $\hat{\xi}_g$. For each level of this white noise shock, we calculate interpolation nodes and weights \hat{X}_l , \hat{X}_r , and φ_X such that

$$\tilde{X}_1 = we_1 \exp(\hat{\xi}_g) = \varphi_X \cdot \hat{X}_l + (1 - \varphi_X) \cdot \hat{X}_r$$

and distribute a mass of π_g^w on these left and right interpolation nodes according to

$$\phi_X(1, \hat{X}_k) = \begin{cases} \phi_X(1, \hat{X}_k) + \pi_g^w \cdot \varphi_X & \text{if } k = l \\ \phi_X(1, \hat{X}_k) + \pi_g^w \cdot (1 - \varphi_X) & \text{if } k = r. \end{cases}$$

Knowing the distribution of the beginning of the period of cash-on-hand, we can use the savings policy function $\tilde{a}^+(j, \hat{X}_k)$ to derive the end-of-period distribution over savings $\phi_a(j, \hat{a}_v)$. This is done in the subroutine `get_distribution_a`. As shown in Program 10.4.e we again use linear interpolation for this. Specifically, for each level of cash-on-hand \hat{X}_k , we can derive interpolation nodes and weights \hat{a}_l , \hat{a}_r , and φ_a such that

$$\tilde{a}(j, \hat{X}_k) = \varphi_a \cdot \hat{a}_l + (1 - \varphi_a) \cdot \hat{a}_r.$$

The only thing to do then is to distribute a mass $\phi(j, \hat{X}_k)$ to these two points, i.e. set

$$\phi_a(j, \hat{a}_v) = \begin{cases} \phi_a(j, \hat{a}_v) + \varphi_a \cdot \phi_X(j, \hat{X}_k) & \text{if } v = l \\ \phi_a(j, \hat{a}_v) + (1 - \varphi_a) \cdot \phi_X(j, \hat{X}_k) & \text{if } v = r. \end{cases}$$

Last but not least, we need to calculate the distribution of the beginning of the period of cash-on-hand at the next age $j + 1$. We therefore use the policy function $\omega(j, \hat{a}_v)$ and calculate for every realization of the white noise productivity shock $\hat{\xi}_{g+}$, as well as the permanent productivity and interest-rate components $\hat{\epsilon}_{i+}$ and $\hat{\vartheta}_{i+}$, the corresponding level of cash-on-hand as

$$\tilde{X}^+ = R_p(\omega^+(j, \hat{a}_v), \hat{\vartheta}_{i+}) \cdot \frac{\hat{a}_v}{\exp(\hat{\epsilon}_{i+})} + we^+ \exp(\hat{\xi}_{g+}) + \tilde{pen}^+.$$

We can again derive interpolation nodes and weights \hat{X}_l , \hat{X}_r , and φ_X from this level of \tilde{X}^+ and finally distribute

$$\phi_X(j, \hat{X}_k) = \begin{cases} \phi_X(j, \hat{X}_k) + \varphi_X \cdot \pi_{g+}^w \cdot \pi_{i+}^{sr} \cdot \phi_a(j - 1, \hat{a}_v) & \text{if } k = l \\ \phi_X(j, \hat{X}_k) + (1 - \varphi_X) \cdot \pi_{g+}^w \cdot \pi_{i+}^{sr} \cdot \phi_a(j - 1, \hat{a}_v) & \text{if } k = r. \end{cases}$$

as shown in Program 10.4.f. We iterate on calculating the distribution of cash-on-hand and end-of-period savings over all ages $j = 1, \dots, J$. From the date of retirement, the calculation of cash-on-hand varies slightly, as we have $\epsilon_{i+} = \zeta_{g+} = 0$.

Life-cycle profiles Once we have computed the distribution of households over the state space, we can calculate average (normalized) age-specific profiles of income, consumption, savings, and the portfolio composition as

$$\begin{aligned}\tilde{\bar{y}}_j &= \sum_{g=1}^{m_w} \pi_g^w \cdot (w \cdot e_j \cdot \exp(\hat{\zeta}_g) + \tilde{pen}_j), \quad \tilde{\bar{c}}_j = \sum_{k=0}^{n_X} \tilde{c}(j, \hat{X}_k) \cdot \phi_X(j, \hat{X}_k) \\ \tilde{\bar{a}}_j &= \sum_{v=0}^{n_a} \hat{a}_v \cdot \phi_a(j-1, \hat{a}_v) \quad \text{and} \quad \tilde{\bar{\omega}}_j = \sum_{v=0}^{n_a} \omega^+(j-1, \hat{a}_v) \cdot \phi_a(j-1, \hat{a}_v).\end{aligned}$$

This is done in subroutine `aggregation`. Note that in order to be consistent with earlier programs in this chapter, we calculate the beginning of the period of the distribution of assets and portfolio composition and therefore have to use the end-of-period distribution of households over a of the previous age $j-1$. In the same way we can also calculate the age-specific variances of these variables. We are, however, not overly interested in the age profiles of normalized variables, but rather of the original variables

$$x_j = E[\exp(\eta_j) \cdot \tilde{x}_j] \quad \text{and} \quad \text{Var}[x_j] = \text{Var}[\exp(\eta_j) \cdot \tilde{x}_j].$$

To recover these we first need to think about the statistical properties of η_j . First, we again normalize $\eta_1 = 0$ and recall that $\epsilon_j = 0$ for $j \geq j_r$. For all ages $j = 2, \dots, J$ we consequently have

$$\eta_j = \eta_{j-1} + \epsilon_j = \sum_{k=2}^{\min(j, j_r-1)} \epsilon_k. \quad (10.27)$$

Since the ϵ_k are statistically independent across time by assumption, η_j is just the sum of independent normally distributed random variables. This makes η_j itself a normally distributed random variable with expected value

$$\mu_{\eta_j} = E[\eta_j] = E\left[\sum_{k=2}^{\min(j, j_r-1)} \epsilon_k\right] = \sum_{k=2}^{\min(j, j_r-1)} E[\epsilon_k] = 0,$$

where the second equality follows from the independence of the ϵ_k . In the same way we have

$$\sigma_{\eta_j}^2 = \text{Var}[\eta_j] = \sum_{k=2}^{\min(j, j_r-1)} \text{Var}[\epsilon_k] = [\min(j, j_r-1) - 1] \sigma_\epsilon^2.$$

Using the properties of the log-normal distribution we consequently have

$$\begin{aligned} E[\exp(\eta_j)] &= \exp(0.5\sigma_{\eta,j}^2) \quad \text{and} \\ \text{Var}[\exp(\eta_j)] &= \exp(\sigma_{\eta,j}^2) \cdot (\exp(\sigma_{\eta,j}^2) - 1). \end{aligned}$$

The last point to remember is that by construction the distribution of households over the state space (j, \tilde{X}) or (j, \tilde{a}) and over η are independent. Only this allowed us to normalize the optimization problem in the first place. Consequently, in order to recover the cohort average of some (non-normalized) variable x , we simply compute

$$\bar{x}_j = E[\exp(\eta_j) \cdot \tilde{x}_j] = E[\exp(\eta_j)] \cdot \tilde{x}_j.$$

For the variance, on the other hand, we can make use of the relationship

$$\begin{aligned} \text{Var}[\bar{x}_j] &= \text{Var}[\exp(\eta_j) \cdot \tilde{x}_j] \\ &= E[\exp(\eta_j)]^2 \text{Var}[\tilde{x}_j] + \text{Var}[\exp(\eta_j)] (\tilde{x}_j)^2 + \text{Var}[\exp(\eta_j)] \text{Var}[\tilde{x}_j]. \end{aligned}$$

All of these relationships result from the basic properties of the product of independent random variables. Note that there is another non-analytic way of calculating age-specific averages, variances, and other statistics which we will discuss later. For now, we want to stick with this *analytic* solution.

Calibration and outcomes Before solving the model and calculating life-cycle statistics, we need to specify parameter values. We follow the literature on life-cycle investments and choose a relatively high value for risk-aversion of 10. Hence, we need to set $\gamma = 0.1$. Since the expected interest rate will be higher than in previous models without risky investment options, we lower the time discount factor to $\beta = 0.96$. With respect to the transitory productivity shock we assume $\sigma_\zeta^2 = 0.0738$ and discretize this shock with $m_w = 7$ realizations. As for the permanent income risk component and the interest-rate shocks, we use $\sigma_\epsilon^2 = 0.0106$ and $\sigma_\vartheta^2 = (0.157)^2$. We discretize those as well with $m_s = 7$ and $m_r = 7$ shock realizations, respectively. In our benchmark simulation, we furthermore let the permanent productivity component and the portfolio return be uncorrelated. We pick a risk-free interest rate of $r_f = 0.02$ and an equity premium of $\mu_r = 0.04$. Finally the growth rates for both the asset and the cash-on-hand grid are set at $u_a = u_x = 0.07$ and the replacement rate of the pension system at $\kappa = 0.5$.

Financial planners routinely advise their clients to start saving with a high equity share and then reduce their equity exposure continuously as they get older. The black line in the left panel of Figure 10.7 indicates that such a rule of thumb could be close to optimal, given the specific assumptions we made about preferences and the risk structure of

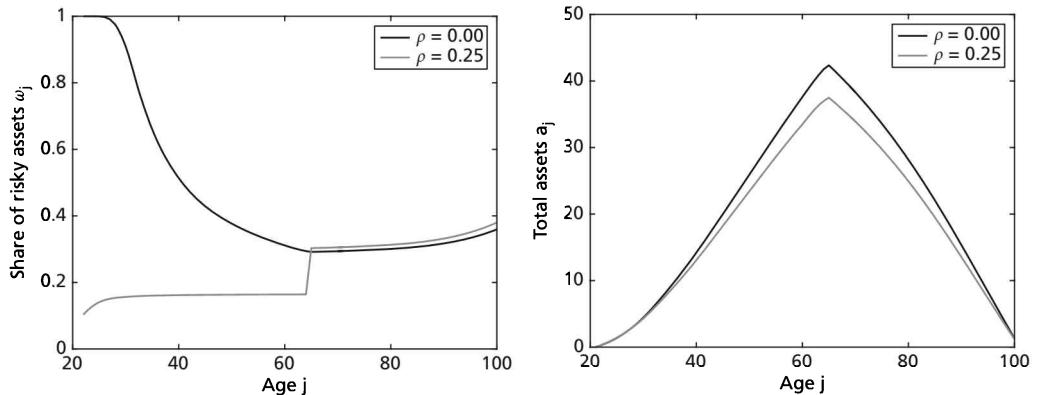


Figure 10.7 Optimal investment over the life cycle

labour income and interest-rate uncertainty. At the beginning of the life cycle, households optimally invest all their savings in stocks and then reduce their (average) risk exposure gradually to an equity level of around 30 to 40 per cent up to the date of retirement. Once they start receiving pensions, the equity share in their portfolio stays about constant. To explain this behaviour in economic terms, we have to realize that at the beginning of life only a small fraction of a household's overall wealth actually consists of financial assets, most of it is human capital (defined as the present value of future labour earnings and pension payments). Throughout working life, financial wealth successively increases—see the black line in the right panel of Figure 10.7—and at the same time the value of human capital declines. With constant relative risk-aversion preferences, the agent wants to keep her overall risk exposure in human capital and asset risk constant over the life cycle. With human capital being less risky than equity and labour-productivity risk and interest-rate risk being uncorrelated, an investment strategy with decreasing (relative) equity exposure over the life cycle can achieve this goal.

The grey lines in Figure 10.7 show that this result is very sensitive to assumptions about the risk structure of equity and labour productivity. To demonstrate this we re-run the model with a slightly positive correlation of stock market and permanent labour-productivity risk of $\rho = 0.25$. The resulting investment structure is fundamentally different from the original one. In fact, the household now optimally holds a much smaller and almost constant risky asset share of around 20 per cent over its working life and then increases equity exposure to about 35 to 40 per cent at the moment it enters retirement. The reason for these differences is that when equity and labour-income risk are correlated, equity investment drives up the riskiness of the total household portfolio much faster. This leads to a much lower optimal share of equity investment throughout working life. At the point of retirement, however, labour-productivity risk fades out immediately, so that the household can return to the investment structure that was already optimal in the baseline model. Of course, when the agent invests less in equity throughout her working life, she will yield a lower average return on her savings,

leading to less financial wealth accumulation up to the date of retirement. Remaining human capital (i.e. the present value of all pension payments), nevertheless, is the same as in the benchmark simulation. Consequently in the scenario with correlated risks the investment share throughout retirement needs to be slightly higher than under uncorrelated risk to guarantee the same overall risk exposure. Note that Figure 10.7 only shows the cohort averages of portfolio shares and asset holdings. In the output file of the program, we also show the coefficients of variation of these variables. They indicate that there actually is substantial variation around these average profiles, resulting from the fact that individuals receive different realizations of labour productivity shocks during their working lives.

The full distribution of households As discussed above, we calculate average profiles of savings, consumption, etc. and their variances using an analytical approach, which exploits the independence of the permanent labour-productivity shock and household choices. This approach, however, suffers from the drawback that we cannot go much beyond calculating age-specific first and second moments. Once we wanted to take a look at age-specific quantiles of the distribution of wealth, for example, we would run into problems immediately. In addition, calculating statistics from the overall distribution of households, like an economy-wide Gini index of wealth or consumption, would also be problematic. If we want to calculate more sophisticated distributional statistics, we have to determine the full distribution of households numerically. To this end, we have to provide a numerical approximation of the distribution of agents over different values of η_j . This is done in the subroutine `generate_eta`, which we show in Program 10.4.g.

In general, we proceed as follows: We discretize the distribution of households over η_j using age-specific shock realizations $\{\hat{\eta}_{j,q}\}_{q=0}^{n_e}$ and a distribution function

$$\phi_e(j, \hat{\eta}_{j,q}) \quad \text{with} \quad \sum_{q=0}^{n_e} \phi_e(j, \hat{\eta}_{j,q}) = 1.$$

This discrete approximation should be consistent with our discretized shock process for permanent labour productivity $(\hat{\epsilon}_i, \pi_i^{sr})$. In the subroutine `generate_eta` we therefore first provide age-specific lower and upper bounds $\underline{\eta}_j$ and $\bar{\eta}_j$ for the potential shock realizations at age j . Of course, since we normalize $\eta_1 = 0$, we necessarily have $\underline{\eta}_1 = \bar{\eta}_1 = 0$ and therefore also $\hat{\eta}_{1,q} = 0$ for all $q = 0, \dots, n_e$. For all subsequent ages, we recall that we can write η_j as the sum of all shock realizations ϵ_k between ages 2 and $\min(j, j_r - 1)$ like in equation (10.27). The lower and upper bound of $\hat{\eta}_{j,q}$ are consequently defined by the lowest and the highest shock realization of $\hat{\epsilon}_i$. Specifically, we can write

$$\underline{\eta}_j = \min \left[\sum_{k=2}^{\min(j, j_r - 1)} \epsilon_k \right] = \sum_{k=2}^{\min(j, j_r - 1)} \min[\hat{\epsilon}_i] = \min(j - 1, j_r - 2) \cdot \min[\hat{\epsilon}_i],$$

Program 10.4.g Computation of the permanent shock distribution

```

subroutine generate_eta()
[.....]
! set bounds and grid for working ages
eta_l(1) = 0d0
eta_u(1) = 0d0
eta(1, :) = 0d0

do ij = 2, JR-1
    eta_l(ij) = (ij-1)*(minval(log(eps)))
    eta_u(ij) = (ij-1)*(maxval(log(eps)))
    call grid_Cons_Equi(eta(ij, :), eta_l(ij), eta_u(ij))
enddo
[.....]
phi_e = 0d0

! initial distribution at age 1
phi_e(1, :) = 1d0/dble(NE+1)

! iterate over working different years
do ij = 2, JR-1

    ! last period's etas
    do ie = 0, NE

        ! new innovations
        do isr = 1, NSR

            ! distribute on new grid
            eta_temp = eta(ij-1, ie) + log(eps(isr))
            call linint_Equi(eta_temp, eta_l(ij), eta_u(ij), &
                NE, iel, ier, varphi)
            phi_e(ij, iel) = phi_e(ij, iel) + &
                dist_epsvtheta(isr)*varphi*phi_e(ij-1, ie)
            phi_e(ij, ier) = phi_e(ij, ier) + &
                dist_epsvtheta(isr)*(1d0-varphi)*phi_e(ij-1, ie)
        enddo
    enddo
[.....]
end subroutine

```

and in the same way for $\bar{\eta}_j$. Note that we already applied the exponential function to `eps` in the subroutine `initialize`, which is why we have to take the `log` here. Having defined a lower and upper bound, we simply let the shock realizations $\hat{\eta}_{j,q}$ form an equidistant grid between $\underline{\eta}_j$ and $\bar{\eta}_j$.

As for the distribution function $\phi_e(j, \hat{\eta}_{j,q})$ we assume an initial distribution of

$$\phi_e(1, \hat{\eta}_{j,q}) = \frac{1}{n_e + 1}.$$

In fact, we could use any distribution here, since all $\hat{\eta}_{1,q}$ values are identical. We continue by successively distributing households on different values of $\hat{\eta}_{j,q}$ according to the same linear interpolation scheme as above. As already mentioned, we use the

discretely approximated shock process $\hat{\epsilon}_i$ to perform this task. Note again that throughout retirement periods there are no new innovations to the random walk process η_j . Hence, the grid values and the distribution function can be copied simply from the last working age. Knowing the distribution ϕ_e we can now simply compute

$$\begin{aligned}\bar{y}_j &= \sum_{q=0}^{n_e} \sum_{g=1}^{m_w} \pi_g^w \cdot \left[w \cdot e_j \cdot \exp(\hat{\zeta}_g) + \widetilde{pen}_j \right] \cdot \exp(\hat{\eta}_{j,q}) \cdot \phi_e(j, \hat{\eta}_{j,q}), \\ \bar{c}_j &= \sum_{q=0}^{n_e} \sum_{k=0}^{n_X} \tilde{c}(j, \hat{X}_k) \cdot \exp(\hat{\eta}_{j,q}) \cdot \phi_X(j, \hat{X}_k) \cdot \phi_e(j, \hat{\eta}_{j,q}), \\ \bar{a}_j &= \sum_{q=0}^{n_e} \sum_{v=0}^{n_a} \hat{a}_v \cdot \exp(\hat{\eta}_{j,q}) \cdot \phi_a(j-1, \hat{a}_v) \cdot \phi_e(j, \hat{\eta}_{j,q}) \quad \text{and} \\ \bar{\omega}_j &= \sum_{q=0}^{n_e} \sum_{v=0}^{n_a} \omega^+(j-1, \hat{a}_v) \cdot \exp(\hat{\eta}_{j,q}) \cdot \phi_a(j-1, \hat{a}_v) \cdot \phi_e(j, \hat{\eta}_{j,q})\end{aligned}$$

and in the same way for the variances, see subroutine aggregation.

Quantiles of the distribution The full distribution of households allows us to determine age-specific (and also aggregate) quantiles of the distribution of wealth, portfolio shares, consumption, and income. In the following we restrict ourselves to wealth distribution. For all other variables of interest, the code can simply be adjusted. The subroutine `calculate_quantiles` organizes the quantile calculation and is shown in Program 10.4.h. We first define the quantile levels we want to compute in an array `thresholds`. The corresponding age-specific quantile values will later be stored in the array `quantiles`. At age 1, households by definition have zero assets such that all quantiles have to be equal to zero. For all other ages we write the asset positions $\hat{a}_v \cdot \exp(\hat{\eta}_{j,q})$ as well as the corresponding population shares $\phi_a(j-1, \hat{a}_v) \cdot \phi_e(j, \hat{\eta}_{j,q})$ into arrays `a_sort` and `a_dist`, respectively. Note that we only use asset levels with a population share of at least 10^{-12} . The total number of stored asset positions is recorded in the integer number `NC`. Following this, we sort the array `a_sort` in ascending order and record the sorting process in the integer array `iorder`. We can then calculate the cumulative distribution associated with the sorted array `a_sort` by successively adding up the values of the distribution in `a_dist`. Since `a_sort` is sorted and `a_dist` is not, we have to use the order provided in the array `iorder` when we cumulate the distribution. The result of this array is a sorted array of asset levels in the cohort of age j together with a cumulative distribution function stored in the array `a_cdist`. At this point it is relatively easy to read off the quantiles of the distribution. We just have to find the first entry in the array `a_cdist` for which the cumulative distribution is larger than the threshold we are looking for. We name this entry `a_cdist(ic)`. Typically, this value will not exactly correspond with the value of `threshold`, but rather the threshold value lies in between

Program 10.4.h Computation of the permanent shock distribution

```

subroutine calculate_quantiles()
[.....]
! define quantile thresholds
thresholds = (/0.05d0, 0.25d0, 0.50d0, 0.75d0, 0.95d0/)
quantiles = 0d0

! iterate over ages
do ij = 2, JJ
[.....]
ic = 1
do ie = 0, NE
do ia = 0, NA
if(phi_a(ij-1, ia)*phi_e(ij, ie) > 1d-12) then
    a_sort(ic) = a(ia)*eta(ij, ie)
    a_dist(ic) = phi_a(ij-1, ia)*phi_e(ij, ie)
    ic = ic + 1
endif
enddo
enddo
NC = ic -1

! sort array and distribution
call sort(a_sort(1:NC), iorder(1:NC))

! calculate cumulative distribution (attention ordering)
a_cdist(1) = a_dist(iorder(1))
do ic = 2, NC
    a_cdist(ic) = a_cdist(ic-1) + a_dist(iorder(ic))
enddo

! get quantiles
do it = 1, size(thresholds, 1)
    if(thresholds(it) <= a_cdist(1)) then
        quantiles(it, ij) = a_sort(1)
    else
        do ic = 2, NC
            if(thresholds(it) < a_cdist(ic)) then
                slope = (a_sort(ic)-a_sort(ic-1)) &
                    /(a_cdist(ic)-a_cdist(ic-1))
                quantiles(it, ij) = a_sort(ic-1) +
                    + slope*(thresholds(it)-a_cdist(ic-1))
                exit
            elseif(ic == NC) then
                quantiles(it, ij) = a_sort(NC)
            endif
        enddo
    endif
enddo
[.....]
end subroutine

```

`a_cdist(ic-1)` and `a_cdist(ic)`. We therefore simply linearly interpolate the values of `a_sort(ic-1)` and `a_sort(ic)` at the value of threshold to get the exact quantile of the distribution. The left panel of Figure 10.8 visualizes how we calculate quantiles using asset position and cumulative distribution. The right panel shows different age-specific quantiles of the asset distribution. We can see that there is a lot of heterogeneity in

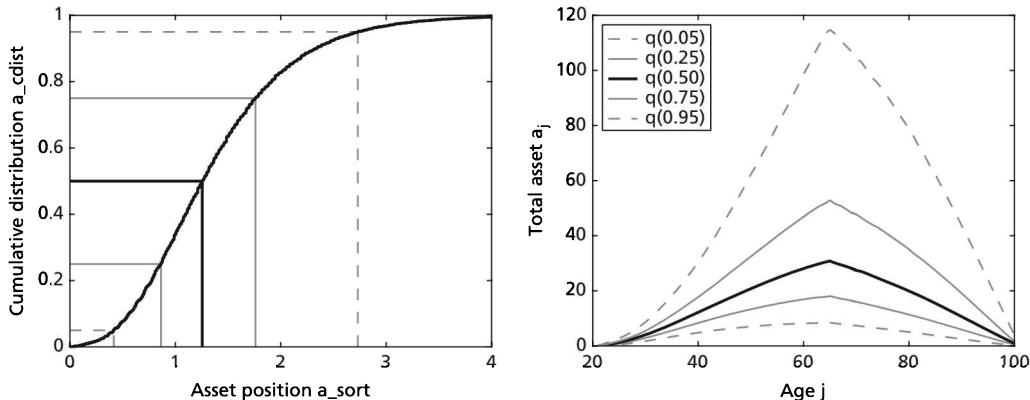


Figure 10.8 Quantiles of the asset distribution

wealth holdings. Asset holdings are most diverse at the date of retirement, where the 5 per cent poorest in terms of wealth hold assets worth only about 10 (model units) while the 5 per cent richest in terms of assets hold assets in excess of around 110. These differences in wealth reflect both the fact that labour incomes differ substantially across agents and that individuals differ in their exposure to capital market risk.

10.2.2 THE CHOICE TO BUY ANNUITIES

Lifespan in our baseline model is risky, but households can only self-insure against this risk through precautionary savings. In the following, we instead allow households to purchase annuities before they enter retirement in order to insure lifespan risk directly. An annuity is a product that pays a constant income stream throughout the whole retirement phase until the agent dies. We denote the purchase price of a product which generates an annuity income stream of 1 during retirement by p_a . Annuities are offered by risk-neutral insurers who pool all the survival risk completely in the economy. We assume that survival probabilities are known by the insurer and do not vary across individuals. In addition, insurers are only allowed to invest resources at the risk-free rate to avoid underfunding. Consequently, the price of an annuity bought at age $j_r - 1$ is

$$p_a = (1 + \xi) \sum_{j=j_r}^J \frac{\prod_{i=j_r}^j \psi_i}{(1 + r_f)^{j-(j_r-1)}} \quad (10.28)$$

where ξ denotes the load factor that the insurer charges on top of the actuarially fair value.

To incorporate the choice to buy annuities into the household optimization problem, we need to distinguish between (ordinary) liquid assets a_l and (non-liquid) retirement assets a_r as elements of the state space. By definition, retirement assets generate an

annuity income stream of $\frac{a_r}{p_a}$. We assume that households can choose to convert part of their wealth into retirement assets at the date prior to retirement and cannot purchase annuities at any other date. Throughout the retirement phase, annuitized assets need to be carried along in the state space. We hence write the optimization problem at each age j except for $j_r - 1$ in the same way to (10.22) as

$$\begin{aligned} V(j, X, a_r) &= \max_{c, a^+, \omega^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} \cdot V(j+1, X^+, a_r) \right] \\ \text{s.t. } X &= c + a^+, \quad a^+ \geq 0, \quad 0 \leq \omega^+ \leq 1 \\ X^+ &= R_p(\omega^+, \vartheta^+) \frac{a^+}{\exp(\epsilon^+)} + wh^+ + pen^+ + \frac{a_r}{p_a}. \end{aligned}$$

Note that we have dropped the \sim notation at this point to keep things neat and accessible. Nevertheless, we should keep in mind that our problem is still formulated in normalized variables and functions. Note further that we have $a_r = 0$ throughout the working phase, that is, whenever $j < j_r$. With the exception of having an additional element in the state space, this problem does not differ from the problem discussed in Section 10.2.1. Consequently the first-order conditions are exactly identical to the ones in (10.23) and (10.24).

The only difference to our baseline model arises at age $j_r - 1$, at which the household has to decide how to split total wealth a^+ into liquid assets $a_l^+ = (1 - \omega_r^+)a^+$ and retirement assets $a_r^+ = \omega_r^+a^+$. The corresponding optimization problem at this age reads

$$\begin{aligned} V(j_r - 1, X, 0) &= \max_{c, a^+, \omega^+, \omega_r^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j_r} E \left[V(j_r, X^+, \omega_r^+ a^+) \right] \\ \text{s.t. } X &= c + a^+, \quad a^+ \geq 0, \quad 0 \leq \omega^+, \omega_r^+ \leq 1 \\ X^+ &= R_p(\omega^+, \vartheta^+) (1 - \omega_r^+) a^+ + pen^+ + \frac{\omega_r^+ a^+}{p_a}, \end{aligned}$$

where we already used $\epsilon^+ = 0$ which is true for every age $j \geq j_r$. We solve this problem backwards in three successive steps:

1. *Equity exposure in liquid wealth:* Given a level of liquid savings a_l^+ and retirement assets a_r^+ , we can solve the household's portfolio optimization problem

$$\begin{aligned} Q(j_r - 1, a_l^+, a_r^+) &= \max_{0 \leq \omega^+ \leq 1} E \left[V(j_r, X^+, a_r^+) \right] \\ \text{s.t. } X^+ &= R_p(\omega^+, \vartheta^+) \cdot a_l^+ + pen^+ + \frac{a_r^+}{p_a}. \end{aligned}$$

The solution to this problem $\omega^+(j_r - 1, a_l^+, a_r^+)$ is again (see equation 10.23) defined by the first-order condition

$$E \left[(\mu_r + \vartheta^+) \cdot a_l^+ \cdot c(j_r, X^+, a_r^+)^{-\frac{1}{\gamma}} \right] = 0.$$

2. *The choice to buy annuities:* The choice of how to split total savings a^+ between liquid and retirement assets is new at this stage. Given that we already know the optimal equity exposure for a given split of assets from step 1, we can define the sub-optimization problem

$$S(a^+) = \max_{\omega_r^+} Q(j_r - 1, (1 - \omega_r^+)a^+, \omega_r^+ a^+)$$

Taking the derivative with respect to ω_r^+ immediately yields the first-order condition

$$Q_{a_l}(j_r - 1, a_l^+, a_r^+) = Q_{a_r}(j_r - 1, a_l^+, a_r^+), \quad (10.29)$$

where Q_{a_l} and Q_{a_r} are the derivatives of Q with respect to liquid and retirement assets, respectively. The first-order condition tells us that the marginal utility of investing one unit of wealth in liquid assets should be equal to the marginal utility of buying an annuity from it. Using the envelope theorem, it is easy to see that the first derivative is given by

$$\begin{aligned} Q_{a_l}(j_r - 1, a_l^+, a_r^+) &= E \left[R_p^+ \cdot c(j_r, X^+, a_r^+)^{-\frac{1}{\gamma}} \right] \\ \text{with } R_p^+ &= 1 + r_f + \omega^+(j_r - 1, a_l^+, a_r^+) (\mu_r + \vartheta^+) \\ \text{and } X^+ &= R_p^+ \cdot a_l^+ + p_e n^+ + \frac{a_r^+}{p_a}. \end{aligned} \quad (10.30)$$

Using the very same logic we obtain

$$Q_{a_r}(j_r - 1, a_l^+, a_r^+) = E \left[\frac{c(j_r, X^+, a_r^+)^{-\frac{1}{\gamma}}}{p_a} + V_{a_r}(j_r, X^+, a_r^+) \right]. \quad (10.31)$$

Given the definition $V(j, X, a_r)$ used above, it is clear that changes in a_r have an indirect impact on the future value function through future cash-on-hand X^+ and a direct impact throughout the whole retirement phase $j \geq j_r$, i.e.

$$V_{a_r}(j, X, a_r) = \beta \psi_{j+1} E \left[\frac{c(j+1, X^+, a_r)^{-\frac{1}{\gamma}}}{p_a} + V_{a_r}(j+1, X^+, a_r) \right]. \quad (10.32)$$

Since in the last period of life there is no future utility that the individual could influence with the annuity, the terminal condition $V_{a_r}(J, X, a_r) = 0$, allow us to compute the marginal value function $V_{a_r}(j, X, a_r)$ recursively at each age $j \geq j_r$. The solution to the first-order condition delivers an optimal choice $\omega_r^+(a^+)$ in the period before retirement that tells us which fraction of its wealth the household should optimally annuitize.

3. *The consumption–savings decision:* Finally, knowing how much wealth to annuitize and how much of the non-annuitized wealth to invest in risky assets, we can set up the consumption–savings problem as

$$V(j_r - 1, X, 0) = \max_{c, a^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j_r} S(a^+) \quad \text{s.t. } X = c + a^+$$

which yields the first-order condition

$$c = [\beta \psi_{j_r} S(a^+)]^{-\gamma}.$$

From the definition of $S(a^+)$ we find that

$$S_a(a^+) = (1 - \omega_r^+) Q_{a_l}(j_r - 1, a_l^+, a_r^+) + \omega_r^+ Q_{a_r}(j_r - 1, a_l^+, a_r^+), \quad (10.33)$$

where the two derivatives are defined in (10.30) and (10.31) and we used the optimal decision rule $\omega_r^+(a^+)$ derived in the previous step. Note that for interior solutions of $\omega_r^+(a^+)$, where the first-order condition $Q_{a_l} = Q_{a_r}$ actually holds, we could also use $S_a(a^+) = Q_{a_l}(j_r - 1, a_l^+, a_r^+)$. However, definition (10.33) can also be applied in corner solutions where either $\omega_r^+(a^+) = 0$ or $\omega_r^+(a^+) = 1$.²²

Numerical implementation To extend our baseline model to account for the choice of annuitization, we have to provide a discretization of the annuity state space a_r . We chose discrete gridpoints $\{\hat{a}_{r,q}\}_{q=0}^{n_{a_r}}$ which again form a growing grid. Since at retirement households can decide which part of their wealth to buy annuities from, we let the grid for liquid wealth and retirement wealth have the same bounds. The grid for annuities is constructed (like any other grid of the state space) in the subroutine `initialize`. Note that all individual choice variables are now three-dimensional arrays, with the first dimension being age, the second either cash-on-hand or end-of-period savings, and the last being the value of annuities $\hat{a}_{r,q}$. As individuals will not have bought any annuities prior to retirement, we don't have to solve the household problem throughout working life for any $\hat{a}_{r,q}$ other than $\hat{a}_{r,0}$.

²² The arguments here are the same as the ones we used in determining the optimal portfolio choice for the liquid asset, see again the discussion accompanying Figure 10.6.

The subroutine `solve_household` that organizes the calculation of policy functions is now divided into three distinct phases:²³

1. During retirement the portfolio optimization problem needs to be solved for each age j and every combination of gridpoints $(\hat{a}_v, \hat{a}_{r,q})$. After that, we use the subroutine `interpolate` do derive the right-hand side of the first-order condition of the consumption–savings problem as well as the future expected value function. Note that throughout retirement we have $a_r^+ = a_r$. Hence we only have to interpolate future values of the policy and value function along the cash-on-hand dimension. Knowing the savings decision, we can use the recursive formula (10.32) to determine the derivative $V_{a_r}(j, \hat{X}_k, \hat{a}_{r,q})$ at each gridpoint.
2. At the age directly before retirement $j_r - 1$, we proceed in exactly the three steps described above. Hence, we first solve the portfolio choice problem $\omega^+(j_r - 1, \hat{a}_v, \hat{a}_{r,q})$ of the household conditional on the amount of liquid and retirement savings held. We then determine the optimal fraction of wealth to annuitize $\omega_r^+(\hat{a}_v)$ which is a function solely of the amount of wealth the individual has available. Knowing how much the household wants to spend on annuities, we can use the subroutine `interpolate` to again calculate the right-hand side of the first-order condition for the consumption–savings problem. Last, we derive the households consumption–savings choice $a^+(j_r - 1, \hat{X}_k, 0)$ for each level of the beginning of the cash-on-hand period.
3. During the working phase the optimization problem of the household looks exactly as it did in Section 10.2.1.

The subroutine `interpolate` as well as the subroutine `get_distribution` follow the same logic.

The value of an annuity stream One of the key determinants of how much the agent should invest in retirement wealth is the marginal value of an annuity stream in comparison to having another unit of liquid wealth. Equation (10.32) shows a recursive formula for the computation of this marginal value. Module 10.5m shows the function `dv_dar`, which uses exactly this formulation. To calculate the value of V_{a_r} , we need to know how much an agent saves until tomorrow a^+ . In addition, the input variables `ij` and `ir` define the current age of the household as well as the value of her annuitized assets $\hat{a}_{r,q}$. We use the savings decision a^+ and determine the corresponding portfolio choice $\omega^+ = \omega(j, a^+, \hat{a}_{r,q})$ through linear interpolation. Knowing a^+ and ω^+ , we can iterate over all future realizations of the interest-rate shock ϑ_{i+} , calculate the resulting cash-on-hand X^+ , and determine the future consumption level again by linear interpolation. Recall that because in the last period of life the annuity payment is already included in cash-on-hand, we have $V_{a_r}(J, X^+, \hat{a}_{r,q}) = 0$. Therefore we can calculate the value of the annuity stream at age $J - 1$ just from the future marginal utility of consumption. In any

²³ We do not show this subroutine here because of space restrictions.

Module 10.5m The marginal value of an annuity

```

function dV_dar(a_plus, ij, ir)
[.....]
! interpolate asset decision
call linint_Grow(a_plus, a_l, a_u, a_grow, &
NA, ial, iar, varphi_a)

! get investment decision
omega_p = varphi_a * omega_plus(ij, ial, ir) + &
(1d0-varphi_a)*omega_plus(ij, iar, ir)

dV_dar = 0d0

do isr = 1, NSR

! get cash-on-hand
R_port = 1d0 + r_f + omega_p*(mu_r + vtheta(isr))
X_p = R_port*a_plus + pen(ij+1) + ar(ir)/p_a

! derive weights
call linint_Grow(X_p, X_l, X_u, X_grow, &
NX, ixl, ixr, varphi_X)

! get distributional weight
dist = dist_epsvtheta(isr)

! derive V_y
c_p = varphi_X * c(ij+1, ixl, ir) + &
(1d0-varphi_X)*c(ij+1, ixr, ir)
c_p = max(c_p, 1d-10)

if(ij < JJ-1)then
    Var_p = varphi_X * v_ar(ij+1, ixl, ir) + &
    (1d0-varphi_X)*v_ar(ij+1, ixr, ir)
    Var_p = max(Var_p, 1d-10)**(-1d0/gamma)
else
    Var_p = 0d0
endif

dV_dar = dV_dar + dist*(margu(c_p)/p_a + Var_p)
enddo
dV_dar = (beta*psi(ij+1)*dV_dar)**(-gamma)

end function

```

period prior to $J - 1$ we have to add the interpolated value of $V_{a_r}(j+1, X^+, \hat{a}_{r,q})$ as shown in equation (10.32). Last but not least, we multiply the resulting value with β and ψ_{j+1} . In addition we transform the value with the same transformation we apply to the right-hand side of the first-order condition. The reason is that $V_{a_r}()$ is the sum of marginal utilities of consumption, which should have the functional form $c^{-\frac{1}{\gamma}}$. Such a functional form, however, is hard to interpolate with linear interpolation. Again it is much better to use the monotone transformation $(\cdot)^{-\gamma}$ interpolate this and re-transform it to its original shape, see again the discussion in Chapter 9.

Annuity choice The annuity choice is carried out in the subroutine `solve_annuitization` which is shown in Program 10.5. The solution procedure for the

Program 10.5 The annuity choice

```

subroutine solve_annuitization(ia)
    [.....]
    ! check for corner solutions
    annu0 = foc_annu(0d0)
    annu1 = foc_annu(1d0)

    ! use intermediate value theorem
    if(annu0*annu1 > 0d0) then
        if(abs(annu0) > abs(annu1)) then
            omegar_plus(ia) = 1d0
        else
            omegar_plus(ia) = 0d0
        endif
        return
    else

        ! get best guess for the root of foc_port
        if(ia > 0) then
            x_in = omegar_plus(ia-1)
        else
            x_in = 0d0
        endif
        check = .false.

        ! solve the household problem using root-finding
        call fzero(x_in, foc_annu, check)
        [.....]
        omegar_plus(ia) = x_in
    endif
end subroutine

```

choice to buy annuities then is quite similar to that in subroutine `solve_portfolio`. We first check for the existence of corner solutions by evaluating the respective first-order condition `foc_annu` at the interval bounds $\omega_r^+ = 0$ and $\omega_r^+ = 1$ and check whether they have the same sign. For the same reasoning as above, (see again Figure 10.6,) we need a sign switch between the interval endpoints to obtain an interior solution. If we don't find such a switch, the optimal solution will be the corner in which the function value is (in absolute terms) closer to zero. In the case that an interior solution exists, we can use the subroutine `fzero` to find the root of the first-order condition. Note that the best starting guess we can provide for this root-finding procedure is the optimum at the previous asset gridpoint $\omega_r^+(\hat{a}_{\nu-1})$, which obviously only works for grid indices $\nu > 0$. Note that we do not have to adjust the tolerance level for the root-finder like we do in the subroutine `solve_portfolio`. The reason becomes clear when looking at the implementation of the first-order condition.

The first-order condition is shown in Module 10.5m.a. The function `func_annu` gets as input a fraction of wealth ω_r^+ that should be converted into an annuity stream. In addition, it receives the current index of the asset gridpoint \hat{a}_ν through the communication variable `ia_com`. Knowing these two variables, we can calculate the amount of future liquid wealth `al_p` and retirement wealth `ar_p` that results from this choice. We then

Module 10.5m.a The first-order condition for annuities

```

function foc_annu(x_in)
    [.....]
    ! store retirement asset
    omegar_p = x_in

    ! determine future liquid wealth and retirement assets
    al_p = (1d0-omegar_p)*a(ia_com)
    ar_p = omegar_p*a(ia_com)

    ! derive interpolation weights
    call linint_Grow(al_p, a_l, a_u, a_grow, NA, ial, iar, varphi_a)
    call linint_Grow(ar_p, ar_l, ar_u, ar_grow, &
                      NAR, irl, irr, varphi_r)

    ! get optimal investment strategy
    if(varphi_a <= varphi_r) then
        omega_p = varphi_a *omega_plus(JR-1, ial, irl) + &
                   (varphi_r-varphi_a)*omega_plus(JR-1, iar, irr) + &
                   (1d0-varphi_r) *omega_plus(JR-1, iar, irr)
    else
        omega_p = varphi_r *omega_plus(JR-1, ial, irl) + &
                   (varphi_a-varphi_r)*omega_plus(JR-1, iar, irr) + &
                   (1d0-varphi_a) *omega_plus(JR-1, iar, irr)
    endif

    Qal = 0d0
    Qar = 0d0
    do isr = 1, NSR

        ! get future cash-on-hand and interpolate
        R_port = 1d0 + r_f + omega_p*(mu_r + vtheta(isr))
        X_p = R_port*al_p + pen(JR) + ar_p/p_a
        call linint_Grow(X_p, X_l, X_u, X_grow, &
                           NX, ixl, ixr, varphi_X)

        ! get distributional weight
        dist = dist_epsvtheta(isr)

        ! get future consumption
        [.....]

        ! get future value of V_ar
        [.....]

        ! get Qal and Qar
        Qal = Qal + dist*R_port*margu(c_p)
        Qar = Qar + dist*(margu(c_p)/p_a + Var_p)
    enddo

    foc_annu = Qal/Qar - 1d0

end function

```

derive linear interpolation nodes and weights along both asset dimensions. Using multi-dimensional linear interpolation as described in Chapter 2, we can determine the risky investment share ω^+ that corresponds to the choice of liquid assets and annuities. Now we can compute the values of the derivatives Q_{a_l} and Q_{a_r} that we need to determine the first-order condition. To this end we iterate over all potential realizations of the interest-rates shock ϑ_{i+} – recall that $\epsilon^+ = \zeta^+ = 0$ – and calculate the amount of cash-on-hand X^+ that

corresponds to the amount of the two asset values, the portfolio choice and the interest-rate realization. Again we use multidimensional linear interpolation to determine both the value of the consumption policy function as well as the derivative V_{a_r} at this specific combination of cash-on-hand and retirement wealth. Knowing these, we can easily calculate Q_{a_l} and Q_{a_r} using the formulas in (10.30) and (10.31). We eventually derive the value of the first-order condition. However, we do not take the simple difference $Q_{a_l} - Q_{a_r}$, but rather the relative difference $\frac{Q_{a_l}}{Q_{a_r}} - 1$. The result is the same from an analytic point of view. Computationally, however, this makes a big difference. In fact, by using the relative difference we omit the accuracy problem we have with the first-order condition that governs the optimal portfolio choice. Since both derivatives Q_{a_l} and Q_{a_r} have the same order—which can of course be very small—their ratio will be in the order of 1 and, hence, a tolerance level of 10^{-8} will be sufficient for the root-finding subroutine.

Last but not least, we also need to adjust the subroutines `interpolate` and `get_distribution_a` so that they take into account the annuity choice directly before retirement. Since the mechanics are identical to those used in determining the first-order condition, the code looks quite similar to what we just saw. Hence, we will not show this subroutine at this stage.

Parameterization and simulation results There are in fact not many additional parameters we need to specify compared to our baseline model. We let the annuity value grid have exactly the same parameters as the liquid asset grid. Since the number of gridpoints increases substantially, as the grid space now has two dimensions, we reduce the number of cash-on-hand and asset grid points to $n_X = n_a = n_{a_r} = 100$, but therefore slightly increase the growth rates of the grids. Figure 10.9 shows average life-cycle profiles from simulations with different values of the annuity loading factor ξ , hence, with different costs of annuitization. The first panel shows the annuity choice policy function $\omega_r^+(\hat{a}_v)$ that tells us which fraction of its total wealth the household would like to convert into annuities prior to retirement. When annuities have a fair price, then only agents with a very small amount of wealth will not want to buy any annuities. As soon as wealth increases a bit, individuals convert up to 90 per cent of their wealth into an annuity stream. They don't transform their total savings into annuities as they want to have some risk exposure in the capital market and annuities only pay the risk-free interest rate. As the loading factor on annuities rises, the purchase price of annuities increases and individuals will buy less of these products. Note, however, that the value of buying insurance against longevity risk seems to be quite substantial so that individuals with a medium to high asset position still convert more than 50 per cent of their wealth into annuities even when the annuity-providing firm charges a 50 per cent surplus on the price. The upper right panel shows the corresponding average life-cycle consumption profiles. Since annuities provide longevity insurance, i.e. insurance against the risk of becoming very old, their major impact on the average consumption profile of household takes place at the latest stage of the life cycle. Individuals with a lot of annuities will realize a much higher consumption at old age. Hence, the consumption profile is much flatter throughout retirement in the case of $\xi = 0$. The lower two panels show the risky asset

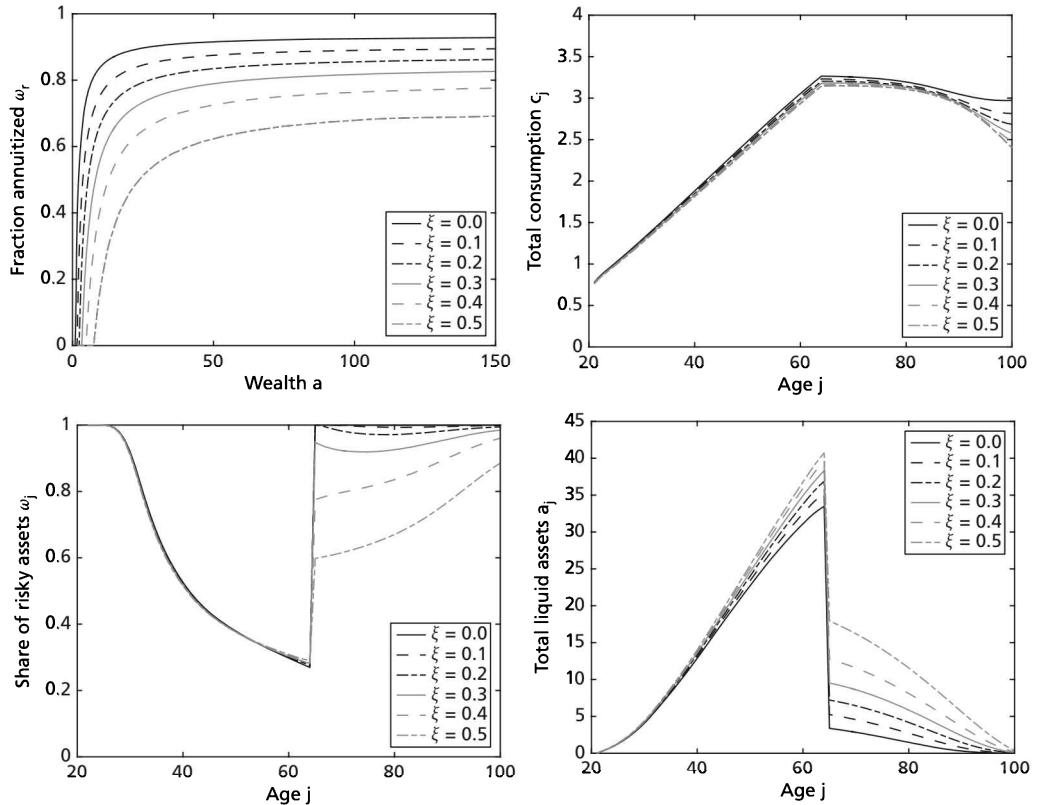


Figure 10.9 Annuity choice with different loading factors

share and the fraction of liquid assets over the life cycle in the different scenarios. The more annuities the household buys upon entering retirement, the fewer liquid savings are left over. Since annuities pay a fixed income stream and only yield the risk-free interest rate, the household will seek more risk in its liquid wealth the more annuities it bought. When buying annuities becomes more expensive and, hence, the household annuitizes less of its available wealth at retirement, the agent will be underinsured against longevity risk. To compensate for this underinsurance, the agent will accumulate more wealth up to retirement in order to hold a precautionary savings stock against the remaining risk of living up to a very old age.

10.2.3 RETIREMENT SAVINGS IN TAX-FAVoured SAVINGS VEHICLES

Deciding about how much of overall wealth to convert into an annuity stream is a rather abstract way of thinking about the retirement planning of a household. In reality savings for retirement typically are accumulated in special savings vehicles, like, for example, 401(k) accounts in the United States. Many of such retirement savings accounts enjoy special tax treatment. In reward for such tax credits, withdrawal from the account is

usually heavily restricted. In the following we want to extend our model to account explicitly for savings in tax-favoured, withdrawal-restricted retirement accounts.

Households can still invest resources a on the capital market and split these savings between bonds b and stocks s . We assume that interest payments as well as capital gains in this *regular savings account* are taxed at the income tax rate τ , which applies to both income from labour and capital.²⁴ In addition to investing in stocks and bonds, households can choose to put some of their savings in a tax-favoured retirement account a_r . Contributions s_r to the account can be fully deducted from income taxation, accrual of interest and capital gains is tax-free, but all withdrawals must be fully taxed. The retirement account is a life-cycle fund that follows a certain exogenous investment pattern $\bar{\omega}_j$. Hence, the equity exposure of the account can vary with age, but the household has no influence on the degree of riskiness. In addition to being tax-favoured, the retirement account does not allow individuals to withdraw money before the date of retirement—i.e. $s_r \geq 0$ must hold—and all the money saved in the account needs to be converted into an annuity at price p_a when the agent enters the retirement phase.

Because all retirement wealth needs to be converted into an annuity, the optimization problem of the household throughout the retirement phase looks almost identical to the one in Section 10.2.2. The only difference is that we have to account for the taxation of capital income and income from the retirement savings account. Consequently we can write the value function of the household aged $j \geq j_r$ as

$$\begin{aligned} V(j, X, a_r) &= \max_{c, a^+, \omega^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E[V(j+1, X^+, a_r)] \\ \text{s.t. } X &= c + a^+, \quad a^+ \geq 0, \quad 0 \leq \omega^+ \leq 1 \\ X^+ &= R_p(\omega^+, \vartheta^+) a^+ + p e n^+ + (1 - \tau) \frac{a_r}{p_a} \end{aligned}$$

where we again used $\epsilon^+ = \zeta^+ = 0$. The portfolio return for the liquid savings account now reads

$$R_p(\omega^+, \vartheta^+) = 1 + (1 - \tau) [r_f + \omega^+ (\mu_r + \vartheta^+)],$$

since interest payments as well as capital gains are taxed at the rate τ .²⁵ The price of the annuity is again defined as in (10.28).

During the working phase of the household, the agent can decide how much of her wealth to save in liquid assets and how much to put into the retirement account. To define this problem correctly, we have to take a quick look at the budget constraint of the household at age $j < j_r$. With contributions to the retirement account s_r^+ being fully deductible from income tax, we can write this constraint as

²⁴ Pensions are exempt from taxation.

²⁵ Note that for simplicity we allow the full deduction of capital losses, i.e. the household gets a tax credit when it makes a negative return on risky stocks.

$$\begin{aligned}
& R_p(\omega, \vartheta) \frac{a_l}{\exp(\epsilon)} + wh - \tau(wh - s_r^+) = c + a_l^+ + s_r^+ \\
\Leftrightarrow & \underbrace{R_p(\omega, \vartheta) \frac{a_l}{\exp(\epsilon)}}_{=:X} + (1 - \tau)wh = c + \underbrace{a_l^+ + (1 - \tau)s_r^+}_{=:a^+} \\
\Leftrightarrow & X = c + a^+
\end{aligned}$$

so that total savings a^+ now consist of savings in liquid wealth a_l^+ and the net of tax contribution to the retirement account $(1 - \tau)s_r^+$. Given the above definitions we have

$$a_l^+ = (1 - \omega_r^+)a^+ \quad \text{and} \quad s_r^+ = \frac{\omega_r^+ a^+}{1 - \tau}.$$

Consequently we can write the household optimization problem during working years $j < j_r - 1$ as

$$\begin{aligned}
V(j, X, a_r) = & \max_{c, a^+, \omega^+, \omega_r^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} V(j+1, X^+, a_r^+) \right] \quad (10.34) \\
\text{s.t. } & X = c + a^+, \quad a^+ \geq 0, \quad 0 \leq \omega^+, \omega_r^+ \leq 1 \\
& X^+ = R_p(\omega^+, \vartheta^+) \frac{a_l^+}{\exp(\epsilon^+)} + (1 - \tau)wh^+ \\
& a_r^+ = \bar{R}_{j+1}(\vartheta^+) \cdot \frac{a_r + s_r^+}{\exp(\epsilon^+)}.
\end{aligned}$$

Since interest and capital gains are exempt from taxation in the retirement savings account, its return is defined as

$$\bar{R}_{j+1}(\vartheta^+) = 1 + r_f + \bar{\omega}_{j+1} (\mu_r + \vartheta^+).$$

Note that a special situation arises at the date $j_r - 1$ directly before retirement, since at this point all contributions to the retirement account $a_r^p = a_r + s_r^+$ will be converted into an annuity. This problem is similar to the annuity choice problem in Section 10.2.2:

$$\begin{aligned}
V(j_r - 1, X, a_r) = & \max_{c, a^+, \omega^+, \omega_r^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j_r} E \left[V(j_r, X^+, a_r^p) \right] \\
\text{s.t. } & X = c + a^+, \quad a^+ \geq 0, \quad 0 \leq \omega^+, \omega_r^+ \leq 1 \\
& X^+ = R_p(\omega^+, \vartheta^+) (1 - \omega_r^+) a^+ + (1 - \tau) \frac{a_r^p}{p_a} \\
& a_r^p = a_r + \frac{\omega_r^+ a^+}{1 - \tau}.
\end{aligned}$$

Note that in this last working period when all savings in the retirement account are converted into an annuity, the return also changes to r_f since all savings are now in riskless bonds.

The optimization problem of the working household has a similar structure as the annuity choice problem discussed in Section 10.2.2. Again we can separate it into three steps:

1. *Equity exposure in liquid wealth:* Given a level of liquid wealth a_l^+ as well as the pre-return balance $a_r^p = a_r + s_r^+$ of the retirement account, we can determine the optimal equity exposure $\omega^+(j, a_l^+, a_r^p)$ by solving

$$\begin{aligned} Q(j, a_l^+, a_r^p) &= \max_{0 \leq \omega^+ \leq 1} E \left[\exp(\epsilon^+)^{1-\frac{1}{r}} V(j+1, X^+, a_r^+) \right] \\ \text{s.t. } X^+ &= R_p(\omega^+, \vartheta^+) \cdot \frac{a_l^+}{\exp(\epsilon^+)} + (1 - \tau) w h^+ \\ a_r^+ &= \bar{R}_{j+1}(\vartheta^+) \frac{a_r^p}{\exp(\epsilon^+)} \end{aligned}$$

for each age $j < j_r - 1$. The definition of the pre-return balance a_r^p seems a bit cumbersome at this point. When forming the expectation of tomorrow's value function, however, we have to take into account that returns on the liquid portfolio and the retirement account are correlated because they are influenced by the common interest-rate shock ϑ^+ . By defining the pre-return value of the retirement account, this can easily be taken care of. With capital income in the regular savings account being taxed at rate τ , the first-order condition of this problem reads

$$E \left[(1 - \tau)(\mu_r + \vartheta^+) a_l^+ \cdot (\exp(\epsilon^+) \cdot c(j, X^+, a_r^+))^{-\frac{1}{r}} \right] = 0. \quad (10.35)$$

At age $j_r - 1$ the household solves the problem

$$\begin{aligned} Q(j_r - 1, a_l^+, a_r^p) &= \max_{0 \leq \omega^+ \leq 1} E \left[V(j_r, X^+, a_r^p) \right] \\ \text{s.t. } X^+ &= R_p(\omega^+, \vartheta^+) \cdot a_l^+ + (1 - \tau) \frac{a_r^p}{p_a} \end{aligned}$$

where p_a is defined in (10.28). The first-order condition of this problem is again (10.35) with $\epsilon^+ = 0$. From the solution of (10.35) we can derive $\omega^+(j, a_l^+, a_r^p)$.

2. *Investment in the retirement account:* Given a level of total savings a^+ and a balance a_r on the retirement account, the household chooses to save in the retirement account by solving the optimization problem

$$\begin{aligned} S(j, a^+, a_r) &= \max_{0 \leq \omega_r^+ \leq 1} Q(j, a_l^+, a_r^p) \\ \text{s.t. } a_l^+ &= (1 - \omega_r^+) a^+ \quad \text{and} \quad a_r^p = a_r + \frac{\omega_r^+ a^+}{1 - \tau}. \end{aligned}$$

The corresponding first-order condition reads

$$Q_{a_l}(j, (1 - \omega_r^+) a^+, a_r + \frac{\omega_r^+ a^+}{1 - \tau}) = \frac{Q_{a_r^p}(j, (1 - \omega_r^+) a^+, a_r + \frac{\omega_r^+ a^+}{1 - \tau})}{1 - \tau}, \quad (10.36)$$

where again the marginal utility of investing in either of the accounts should be equalized in the optimum. As before, we can write the derivative of Q with respect to liquid assets a_l^+ for each age $j < j_r - 1$ as

$$\begin{aligned} Q_{a_l}(j, a_l^+, a_r^p) &= E \left[R_p^+ \left(\exp(\epsilon^+) c(j+1, X^+, a_r^+) \right)^{-\frac{1}{\gamma}} \right] \\ \text{with } R_p^+ &= 1 + (1 - \tau) \left[r_f + \omega^+(j, a_l^+, a_r^p) (\mu_r + \vartheta^+) \right], \\ X^+ &= R_p^+ \cdot \frac{a_l^+}{\exp(\epsilon^+)} + (1 - \tau) w h^+ \quad \text{and} \quad a_r^+ = \bar{R}_{j+1}(\vartheta^+) \frac{a_r^p}{\exp(\epsilon^+)}. \end{aligned}$$

At age $j_r - 1$ the respective derivative of Q is defined by

$$\begin{aligned} Q_{a_l}(j_r - 1, a_l^+, a_r^p) &= E \left[R_p^+ \left(c(j_r, X^+, a_r^p) \right)^{-\frac{1}{\gamma}} \right] \\ \text{with } X^+ &= R_p^+ \cdot a_l^+ + p e n^+ + (1 - \tau) \frac{a_r^p}{p_a}. \end{aligned}$$

Turning to the derivative of Q with respect to savings in the retirement account, we have to remember that the consumption benefits of these savings are only realized from the date of retirement onwards. Hence we can write for each age $j < j_r - 1$

$$Q_{a_r^p}(j, a_l^+, a_r^p) = E \left[\bar{R}_{j+1}(\vartheta^+) \exp(\epsilon^+)^{-\frac{1}{\gamma}} V_{a_r}(j+1, X^+, a_r^+) \right],$$

where we derive from (10.34)

$$V_{a_r}(j, X, a_r) = \beta \psi_{j+1} Q_{a_r^p}(j, a_l^+, a_r^p). \quad (10.37)$$

Of course, at age $j_r - 1$ the derivative of Q with respect to retirement account savings also changes to

$$Q_{a_r^p}(j_r - 1, a_l^+, a_r^p) = E \left[(1 - \tau) \frac{c(j_r, X^+, a_r^p)^{-\frac{1}{\gamma}}}{p_a} + V_{a_r}(j_r, X^+, a_r^p) \right].$$

Substituting the definitions of the derivatives Q_{a_l} and $Q_{a_r^p}$ into the respective conditions (10.36) we then derive $\omega_r^+(j, a^+, a_r)$. Given optimal investment in the retirement account we see that

$$S_{a_r}(j, a^+, a_r) = Q_{a_r^p}(j_r, a_l^+, a_r^p) \quad (10.38)$$

which will be very helpful to compute $V_{a_r}(j, X, a_r)$ numerically.

3. *The consumption-savings decision:* Finally, knowing the future investment structure $\omega_r^+(j, a^+, a_r)$ and $\omega^+(j, a_l^+, a_r^p)$, we can set up the consumption-savings problem as

$$V(j, X, a_r) = \max_{c, a^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} S(j, a^+, a_r) \quad \text{s.t. } X = c + a^+$$

which again yields the first-order condition

$$c = [\beta \psi_{j+1} S_a(j, a^+, a_r)]^{-\gamma}. \quad (10.39)$$

The derivative of $S(j, a^+, a_r)$ with respect to a^+ can be computed from

$$S_a(j, a^+, a_r) = (1 - \omega_r^+) \cdot Q_{a_l}(j, a_l^+, a_r^p) + \frac{\omega_r^+}{1 - \tau} \cdot Q_{a_r^p}(j, a_l^+, a_r^p).$$

The solution of (10.39) then yields optimal savings $a^+(j, X, a_r)$.

- Numerical implementation** We discretize the state space of our model following the same logic and conventions as in Program 10.5. Note that we could be more precise and specify separate discretized grids for a_l^+ and a_r^p . For the sake of simplicity, however, we assume that the respective grids for these variables are identical to the ones for a^+ and a_r^+ . As a_l^+ and a^+ as well as a_r^p and a_r have the same order of magnitude, respectively, this should not generate any problems. Throughout the retirement phase, the structure of the program is very much the same as in Program 10.5. Hence, we will focus on how we solve for decisions throughout the working phase of the individual, where households have to decide how much to invest into the retirement account. Program 10.6 shows an excerpt of the subroutine `solve_household` that illustrates how the solution of the household problem during the working phase is organized. Here one can immediately see the three-step structure of the solution algorithm we already discussed above:

1. In the first step, given decisions in period $j + 1$, the subroutine `solve_portfolio` determines the optimal portfolio decisions $\omega^+(j, a_l^+, a_r^p)$ for each level of liquid savings and each pre-return balance of the retirement account. Remember that we use

Program 10.6 Organizing the solution of the household problem

```

subroutine solve_household()
[.....]
! now household problem during working life
do ij = JR-1, 1, -1

    ! the portfolio choice problem
do ia = 1, NA
    do ir = 0, NAR
        call solve_portfolio(ij, ia, ir)
    enddo
enddo
omega_plus(ij, 0, :) = omega_plus(ij, 1, :)

call interpolate(ij)

    ! the retirement account investment problem
do ia = 1, NA
    do ir = 0, NAR
        call solve_retaccount(ij, ia, ir)
    enddo
enddo
omegar_plus(ij, 0, :) = omegar_plus(ij, 1, :)

call interpolate_RHS(ij)

    ! the consumption-savings problems
do ix = 0, NX
    do ir = 0, NAR
        call solve_consumption(ij, ix, ir)
    enddo
enddo

write(*,'(a,i3,a)')'Age: ',ij,' DONE!'
enddo
end subroutine

```

the same grid definitions for a^+ and a_l^+ as well as a_r and a_r^p . Hence, the grid points $a(ia)$ and $ar(ir)$ map into levels of a_l^+ and a_r^p at this point. The follow up subroutine `interpolate` then derives the values of Q , Q_{a_l} , and $Q_{a_r^p}$ at these respective account balances.

2. In the second step, given optimal investment decisions for the liquid account, the subroutine `solve_retaccount` solves for the optimal split $\omega_r^+(j, a^+, a_r)$, where the grid points $a(ia)$ and $ar(ir)$ now stand in for total savings a^+ and the beginning of period balance of the retirement account a_r , respectively. Having determined the retirement investment plan, the subroutine `interpolate_RHS` calculates the right-hand side of the first-order condition (10.37), meaning $S_a(j, a^+, a_r)$, the actual value $S(j, a^+, a_r)$, and the derivative $S_{a_r}(j, a^+, a_r) = Q_{a_r^p}(j, a_l^+, a_r^p)$ at each grid point.
3. Last but not least, the consumption–savings decision is determined in the subroutine `solve_consumption`. Given the optimal savings $a^+(j, X, a_r)$ we derive the value function $V(j, X, a_r)$ using function `valuefunc` as well as the derivative $V_{a_r}(j, X, a_r)$

using function `vdaret` for any level of cash-on-hand X and any balance of the retirement account a_r .

Note that we will not be able to show all of the subroutines mentioned above, as this would clearly go beyond the constraints of this book. Nevertheless, many code fragments from the previous Program 10.5 can be rediscovered in the subroutines `solve_portfolio`, `solve_consumption`, and `interpolate`.

Solving for the retirement account investment decisions One particular concern in this program is how we solve for the retirement investment decision. Program 10.6.a shows the respective subroutine `solve_retaccount`. The subroutine is organized in the same way as the subroutine `solve_portfolio`. We first look for the existence of corner solutions and then apply the root-finder in subroutine `fzero` to determine the value of ω_r^+ that solves the first-order condition. However, there is one particular addition to the code that we haven't used so far. In a two-dimensional framework with linear interpolation on

Program 10.6.a Solving for the optimal retirement account investment

```

subroutine solve_retaccount(ij, ia, ir)
[.....]
! check for corner solutions
ret0 = foc_ret(0d0)
ret1 = foc_ret(1d0)

! use intermediate value theorem
if(ret0*ret1 > 0d0) then
    if(abs(ret0) > abs(ret1)) then
        omegar_plus(ij, ia, ir) = 1d0
    else
        omegar_plus(ij, ia, ir) = 0d0
    endif
    return
else
    ! get best guess for the root of foc_ret
    if(ia > 1) then
        x_in = omegar_plus(ij, ia-1, ir)
    else
        x_in = -ret0/(ret1-ret0)
    endif
    check = .false.

    ! solve the household problem using root-finding
    call fzero(x_in, foc_ret, check)

    ! double safety net
    if(check .or. x_in < 0d0 .or. x_in > 1d0) then
        [.....]
    endif

    omegar_plus(ij, ia, ir) = x_in
endif
end subroutine

```

triangles, it is sometimes hard to exactly identify the root of a function by means of a Newton or Newton-like method as the one provided in `fzero`. The reason for this is that the interpolated functions Q_a and Q_r exhibit a lot of non-differentiabilities. Hence, the root-finding routine of the toolbox might fail frequently in determining the root of the function `foc` which results in either the variable `check` being assigned a value `.true.` or the suggested solution in `x_in` lying outside of the interval $[0, 1]$. To deal with this issue, we make our root-finding procedure robust by adding a simple bisection search method in case the standard root-finder fails. The advantage of such a bisection search is that it is a super robust algorithm that almost always converges. The downside is that it is slow compared to a Newton-like method.

Making our code robust with a bisection search is described in Program 10.6.b. The idea of a bisection search was described in Program 2.5. Here we can use the left and right

Program 10.6.b Making the search for a root robust

```

if(check .or. x_in < 0d0 .or. x_in > 1d0) then

    ! perform a bisection search
    check = .false.
    x_l = 0d0
    x_r = 1d0
    f_l = reto
    it = 1
    do
        ! do the updating step
        x_new = (x_l + x_r)/2d0
        f_new = foc_ret(x_new)

        ! check for convergence
        if(abs(x_new-x_l) < 1d-8) then
            x_in = x_new
            exit
        endif

        ! check for maximum iterations
        if(it == 1000) then
            check = .true.
            x_in = x_new
            exit
        endif

        ! updating step
        if(f_l*f_new <= 0d0) then
            x_r = x_new
        else
            x_l = x_new
            f_l = f_new
        endif

        it = it + 1
    enddo

    ! if anything goes wrong then, print error message
    if(check .or. x_in < 0d0 .or. x_in > 1d0) then
        write(*,'(a, 3i4)')'ERROR IN ROOTFINDING RET : ', ij, ia, ir
    endif
endif
  
```

interval endpoints $\omega_r^+ = 0$ and $\omega_r^+ = 1$ to bracket the root of the function `foc_ret`. We have already obtained the respective function values and stored them in the variable `ret0` and `ret1`. We then iterate by calculating the function value at the midpoint of the respective bracketing interval and checking whether the root is lying in the left or the right subinterval. We continue this procedure until the width of the bracketing interval is smaller than a tolerance level of 10^{-8} . Only if our algorithm exceeds 1,000 steps, will we set the variable `check` to `.true.` indicating that something went badly wrong. Equipped with such a robust root-finding procedure, our model will almost certainly converge without problems. Note that we apply the same process to the subroutine `solve_portfolio` to make that robust.

The first-order condition for retirement savings Program 10.6.c shows how we calculate the first-order condition regarding retirement savings. This function departs from the function `foc_annu` in Module 10.5m in that it makes direct use of variables `Q_al` and `Q_ar`, which were computed in the subroutine `interpolate`. In the function `foc_annu` we still derived the values of Q_{al} and Q_{ar} ‘by hand’, in the sense that we calculated them

Program 10.6.c First-order condition for retirement savings

```

function foc_ret(x_in)
[.....]
omega_r = x_in

! determine liquid wealth and pre-return retirement wealth
al_p = (1d0-omega_r)*a(ij_com)
ar_p = ar(ir_com) + omega_r*a(ij_com)/(1d0-tau)

! derive interpolation weights
call linint_Grow(al_p, a_l, a_u, a_grow, NA, &
                  ial, iar, varphi_a)
call linint_Grow(ar_p, ar_l, ar_u, ar_grow, NAR, &
                  irl, irr, varphi_r)

! get interpolated values of Q_al and Q_ar
if(varphi_a <= varphi_r)then
    Qal = varphi_a *Q_al(ij_com, ial, irl) + &
          (varphi_r-varphi_a)*Q_al(ij_com, iar, irl) + &
          (1d0-varphi_r) *Q_al(ij_com, iar, irr)
    Qar = varphi_a *Q_ar(ij_com, ial, irl) + &
          (varphi_r-varphi_a)*Q_ar(ij_com, iar, irl) + &
          (1d0-varphi_r) *Q_ar(ij_com, iar, irr)
else
    Qal = varphi_r *Q_al(ij_com, ial, irl) + &
          (varphi_a-varphi_r)*Q_al(ij_com, ial, irr) + &
          (1d0-varphi_a) *Q_al(ij_com, iar, irr)
    Qar = varphi_r *Q_ar(ij_com, ial, irl) + &
          (varphi_a-varphi_r)*Q_ar(ij_com, ial, irr) + &
          (1d0-varphi_a) *Q_ar(ij_com, iar, irr)
endif
Qal = max(Qal, 1d-10)**(-1d0/gamma)
Qar = max(Qar, 1d-10)**(-1d0/gamma)

foc_ret = Qal/(Qar/(1d0-tau)) - 1d0

end function

```

by taking expectations over the future marginal utility of consumption and future values of V_{ar} . As we only had to solve for the annuitization decision at one stage of the life cycle, this procedure didn't result in too much of a computational efficiency loss. In the current program, however, we have to solve for retirement account investment decisions quite often. Hence, it is much more cost efficient to first calculate these expectations, store them in arrays Q_{a1} , and Q_{ar} and then just interpolate these arrays using multidimensional linear interpolation. Note that this procedure follows the same logic we apply when solving the consumption-savings problem using the variable RHS .

Marginal value of annuities In Module 10.5m we computed the marginal value of an V_{ar} . In the present model this value can be derived much more easily. From (10.37) and (10.38) we know that $V_{ar}(j, X, a_r) = \beta \psi_{j+1} S_{ar}(j, a^+, a_r)$. Since we computed the derivative s_{ar} in subroutine `interpolate_RHS`, we can directly use a linearly interpolated value of this derivative, as can be seen in Program 10.6.d.

Determining the distribution of households Consistent with our three optimization steps, we also determine three distributions of households at each age j . The distribution function $\phi_X(j, X, a_r)$ indicates how households are distributed on the cash-on-hand grid as well as the values of the retirement savings account. We then use a household's savings decision $a^+(j, X, a_r)$ to determine the distribution $\phi_a(j, a^+, a_r)$. The retirement account investment decision $\omega_r(j, a^+, a_r)$ leads us to a distribution $\phi_r(j, a_l^+, a_r^p)$ over liquid savings and pre-return balances of retirement accounts. Lastly, the portfolio decision $\omega^+(j, a_l^+, a_r^p)$ allows us to map ϕ_r into a next-period distribution over cash-on-hand and retirement account balances. The successive calculation of these distribution functions is again organized in the subroutine `get_distribution`, which now calls three other subroutines that actually carry out the calculation of the three distribution functions. An example of this is the subroutine `get_distirbution_r` that calculates the distribution ϕ_r and which is shown in Program 10.6.e. Throughout the retirement phase, the distributions ϕ_a and ϕ_r will coincide as $\omega_r = 0$ and the value of a_r stays constant over time. During working life, however, the distribution ϕ_r is a result of the retirement investment decision ω_r , which maps the values of a^+ and a_r into values a_l^+ and

Program 10.6.d Marginal value of an annuity

```
function dv_dar(a_plus, ij, ir)
[.....]
! interpolate asset decision
call linint_Grow(a_plus, a_1, a_u, a_grow, NA, &
                 ial, iar, varphi_a)

dv_dar = varphi_a*s_ar(ij, ial, ir) &
         + (1d0-varphi_a)*s_ar(ij, iar, ir)
dv_dar = (beta*psi(ij+1))**(-gamma)*dV_dar

end function
```

Program 10.6.e Calculating the end-of-period distribution of households

```

subroutine get_distribution_r(ij)
    [.....]
    ! phi_a and phi_r identical when agent is already retired
    if(ij >= JR) then
        phi_r(ij, :, :) = phi_a(ij, :, :)
    ! during working life
    else

        ! iterate over both wealth dimensions
        do ia = 0, NA
            do ir = 0, NAR

                ! determine future al_p and ar_p
                al_p = (1d0-omegar_plus(ij,ia,ir))*a(ia)
                ar_p = ar(ir) + omegar_plus(ij,ia,ir)*a(ia)/(1d0-tau)

                ! derive interpolation weights
                call linint_Grow(al_p, a_l, a_u, a_grow, &
                                  NA, ial, iar, varphi_a)
                call linint_Grow(ar_p, ar_l, ar_u, ar_grow, &
                                  NAR, irl, irr, varphi_r)

                ! get distribution over al_p and ar_p
                phi_r(ij, ial, irl) = phi_r(ij, ial, irl) + &
                    varphi_a*varphi_r*phi_a(ij, ia, ir)
                phi_r(ij, iar, irl) = phi_r(ij, iar, irl) + &
                    (1d0-varphi_a)*varphi_r*phi_a(ij, ia, ir)
                phi_r(ij, ial, irr) = phi_r(ij, ial, irr) + &
                    varphi_a*(1d0-varphi_r)*phi_a(ij, ia, ir)
                phi_r(ij, iar, irr) = phi_r(ij, iar, irr) + &
                    (1d0-varphi_a)*(1d0-varphi_r)*phi_a(ij, ia, ir)
            enddo
        enddo
    endif
end subroutine

```

a_r^p . From these values we can derive interpolation weights as usual. Note, however, that we no longer use multidimensional linear interpolation, but use what is called bilinear interpolation. Consequently, we distribute individuals on the four points that form a square around the actual point combination a_l^+ and a_r^p . At this stage of the program, using either multidimensional linear or bilinear interpolation is more a matter of taste than actually guided by numerical ideas. This is, however, not true when it comes to actually interpolating functions we want to use in a root-finding or optimization procedure, see Section 2.6.2 in Chapter 2.

Parameterization and simulation results We let the upper endpoints of the two savings grids be $a_u = 250$ and $a_{r,u} = 2 \cdot a_{r,u}$ since individuals tend to save more in the retirement account—at least for the specification of parameters we chose. As the problem is quite high-dimensional, we use $n_X = n_a = n_{a_r} = 80$ along the continuous dimensions of the state space and reduce the number of shocks to $n_w = m_s = m_r = 5$. We let the

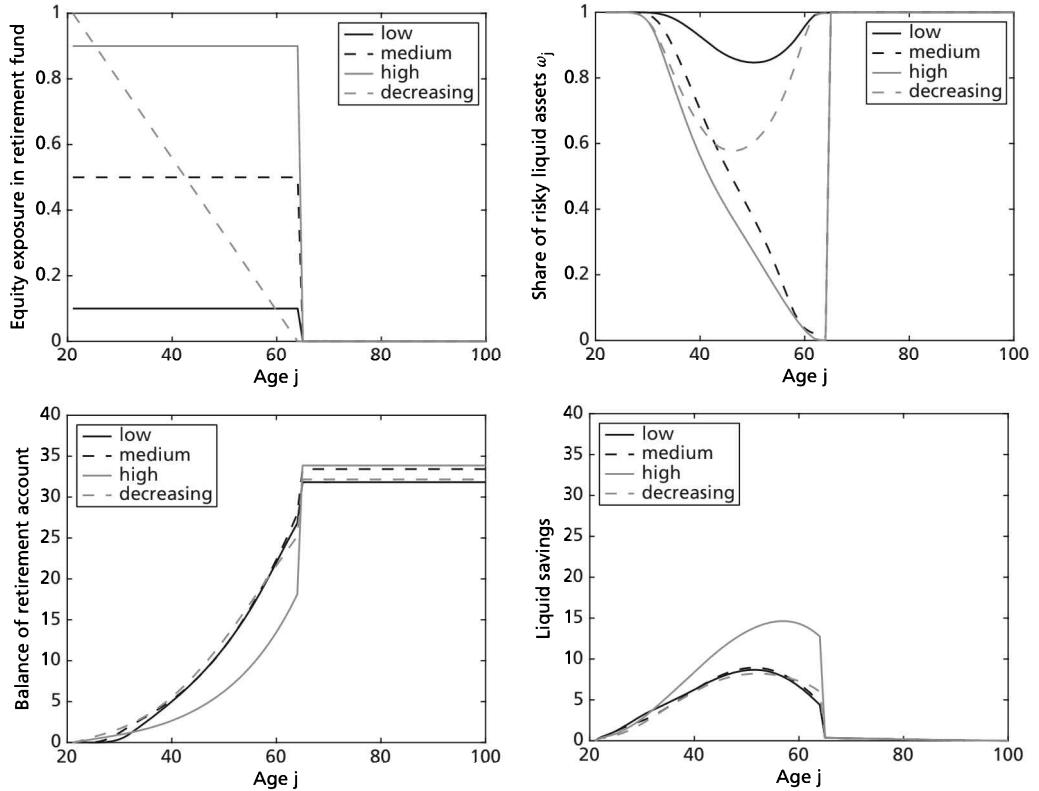


Figure 10.10 Retirement savings under different risk exposure

tax rate be $\tau = 0.20$ and assume that the annuity-providing firm charges a 5 per cent surcharge on the fair price of the annuity, i.e. $\xi = 0.05$. Like before, we assume that the annuity-providing firm only invests in risk-free assets when individuals are already retired. However, we consider four different scenarios for the (exogenous) investment strategies of the retirement savings fund. The first three scenarios assume a constant equity exposure over the working life of individuals at 10 (low), 50 (medium), and 90 (high) per cent. In the last scenario we assume that equity holdings amount to 100 per cent at age $j = 1$ and then successively decline to 0 until the date of retirement (decreasing). The first panel of Figure 10.10 summarizes these four investment strategies. The right panel of this figure shows the household's (average) investment strategy for its liquid assets in the four scenarios. The pattern that emerges is quite intuitive. The more risky the investments in the retirement savings account are, the more cautious the household will be with its liquid savings. While in the low equity exposure case the individual almost exclusively holds equity in her liquid account, equity exposure drops to almost zero towards the end of the working period in the case of a high equity exposure through retirement savings. The lower two panels describe the savings behaviour of the household in the retirement account (left) and in liquid savings (right). In general we find

that the household saves much more in the retirement account than in liquid savings. The reason for this is the preferential tax treatment of these accounts, meaning that the accumulation of interest payments is tax-free. The one scenario that stands out is the one where investment in the retirement fund is very risky. In this scenario, the household will accumulate much less in the retirement fund throughout the working period, foregoing preferential tax treatment on the one hand but gaining a lower risk exposure on the other. The agent then shifts the additional liquid wealth she built up throughout her working life into the retirement savings account upon retirement, where the riskiness of the account switches to zero and the account balance is fully annuitized.

Finally, we can ask which of the four investment strategies of the retirement fund would be preferred by the household or, put differently, which is the best investment strategy of the retirement fund. To this end, we use the low equity exposure scenario as a benchmark and compute by how much the welfare of the household changes when we move from this scenario to each of the other scenarios. We define welfare as an ex ante utility of a household and compute it according to

$$EV = E[V(1, X, 0)] = \sum_{g=1}^{m_w} \pi_g^w \cdot V(1, X, 0)$$

with $X = (1 - \tau) \cdot w \cdot e_1 \cdot \exp(\hat{\xi}_g)$.

Note that such a straightforward derivation of utility only makes sense at age 1, where by assumption we have $e_1 = 0$ and therefore $\eta_1 = 0$. Therefore we do not have to explicitly adjust utility for the normalization with the permanent income component, see again equation (10.22). With the definition of our utility function, we can use the concept of Hicksian equivalent variation in the same way as in Chapter 6. We consequently calculate

$$\Delta_i = \left[\frac{EV_i}{EV_{\text{low}}} \right]^{\frac{1}{1-\gamma}} - 1,$$

which informs us by which factor of consumption of the agent would have to increase or decrease in each state of the world in order to make this agent as well off under scenario $i \in \{\text{medium, high, decreasing}\}$ compared with the low equity exposure scenario. Table 10.1 shows that a constantly high equity exposure in the retirement account is bad for a household's welfare. The reason for this is that the household has to compensate for

Table 10.1 Welfare comparison

| Scenario | Welfare change |
|------------|----------------|
| Middle | -0.013 |
| High | -0.068 |
| Decreasing | 0.094 |

this high riskiness by saving less in the retirement account and therefore foregoing some of the tax benefits. The highest level of welfare is attained when the retirement fund has a decreasing equity exposure throughout the working life of the household. This is not too surprising in light of the fact that decreasing the riskiness of financial investments over time has already proven to be the optimal strategy of the household absent any form of retirement accounts. The magnitude of welfare gains and losses is however quite small, which can be explained by the fact that the household can (at least most of the time) undo a very high equity exposure in her retirement savings account by buying less risky assets in her liquid portfolio.

10.3 Further reading

The general idea of the analysis and the calibration of the earnings process in the baseline model of the first part is taken from Storesletten, Telmer, and Yaron (2004) and the GMM estimates from the respective NBER Working Paper version. Love (2006) and Pries (2007) extend the baseline model by considering tax-favoured retirement accounts in a model with fixed and variable labour supply, respectively. French (2005) estimates a life-cycle model with variable labour supply in which the future health status is uncertain. The female labour force-participation model is based on Attanasio, Low, and Sanchez-Marcos (2008). Low, Meghir, and Pistaferri (2010) provide a recent extension of this approach considering labour market search and disability risk, while Fernanadez and Wong (2014) analyse the impact of divorce risk on female labour supply.

The benchmark model for the portfolio choice approach of the second part is Cocco, Gomes, and Maenhout (2005), from where we also take most of the parameters. In recent years this model has been extended in numerous directions. Gomes, Kotlikoff, and Viceira (2008) introduce variable labour supply and investigate the welfare costs of constraining portfolio choices to mimic those of popular defined-contribution pension plans. In order to replicate the observed limited stock-market participation of specific population groups, studies such as Gomes and Michaelides (2005), Alan (2006), or Campanale (2009) model entry and/or transaction cost. Love (2010) considers the impact of marital-status transitions such as widowhood or divorce on portfolio choice.

Since governments in industrialized countries are downscaling pay-as-you-go pension schemes, the general awareness of longevity risk has increased worldwide. Therefore, recent life-cycle studies try to explain the observed thin capitalization on private annuity markets, search for optimal annuitization strategies, and compute the welfare losses from non-optimal investment behaviour. Our approach follows Horneff, Maurer, and Stamos (2008) who exclude equity-linked annuities and only consider annuities that can be bought at any age and yield an immediate and constant payout for the remaining

lifespan. Horneff et al. (2009) allows the investor to chose the proportion of stocks within the purchased annuities so that the payout becomes variable in each year. As Horneff, Maurer, and Rogalla (2010) demonstrate, so-called deferred annuities, where the payout starts at some advanced age, might be optimal when future mortality rates are uncertain. Chai et al. (2011) extend the model to endogenous labour supply and endogenous retirement choice. Some more recent papers study annuitization decisions in the presence of medical expenditure risk (Yogo (2016), Peijnenburg, Nijman, and Werker (2017)), inflation risk (Peijnenburg, Nijman, and Werker (2016)) and means-tested pensions (Iskhakov, Thorp, and Bateman (2015), Büttler, Peijnenburg, and Straubli (2017)). Finally, the last section which discusses retirement accounts is closely linked to Campbell et al. (2001). Two recent extensions of this model are provided by Gomes, Michaelides, and Polkovnichenko (2009) and Le Blanc and Scholl (2017). There investors can optimize stockholdings in the retirement account and are not forced to fully annuitize their wealth after retirement.

10.4 Exercises

- 10.1. Add a white noise component ζ_j to household labour productivity in the baseline model (see Storesletten, Telmer, and Yaron (2004)) so that

$$h_j = e_j \cdot \exp[\theta + \eta_j + \zeta_j] \quad \text{with} \quad \zeta_j \in N(0, \sigma_\zeta^2).$$

- a) Explain how this changes the dynamic programming problem and the numerical implementation. Use seven shock levels to approximate the white noise component.
- b) You should be able to generate the same model outcome as in the baseline model when you set $\sigma_\zeta = 0$. Then, simulate the model with $\sigma_\zeta = 0.057$ and therefore increase the asset grid to $a_u = 1,000$.
- c) What is the main difference between the model with the white noise component and the baseline model? Explain in economic terms.

- 10.2. The baseline model excludes a bequest motive so that households only care about their own consumption. We introduced the so-called ‘warm glow’ bequest motive in the exercise sections of Chapters 5 and 7, where the bequeather derives direct utility from the amount of assets left to descendants. Assume that household preferences are given by

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) \left\{ u(c_j) + \beta(1 - \psi_{j+1}) \mathcal{B}(Ra_{j+1}) \right\} \right]$$

using the bequest function

$$\mathcal{B}(Ra_j) = v_j \cdot \frac{\left(1 + \frac{Ra_j}{\chi}\right)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

The term v_j denotes the strength of the bequest motive at a specific age j , while χ reflects the extent to which bequests are luxury goods (see De Nardi (2004)).

- a) Set up the household's maximization problem, derive the first-order conditions and implement the bequest model numerically. Make sure that $v_j = 0$ generates the same results as the baseline model.
- b) Start out with specification

$$v_j = \begin{cases} 0 & \text{if } j < j_r \\ 10 & \text{if } j \geq j_r \end{cases} \quad \text{and}$$

as well as $\chi = 1$. Explain your results in economic terms.

- c) Successively increase v_j to a value of 10 for $j < j_r$. Then let χ rise to a value of 10. Explain the observed changes in behaviour.
- 10.3. Hyperbolic consumers have time-inconsistent preferences in the sense that they discount future consumption too much when making their consumption choices, see Angeletos et al. (2001). Since they regret their previous savings decisions when they grow older, one has to distinguish between their actual behaviour $c(z^+)$, $a^+(z^+)$, $V(z^+)$ and their belief $\hat{c}(z^+)$, $\hat{a}^+(z^+)$, $\hat{V}(z^+)$. In the following we consider so-called *naive* hyperbolic consumers who think that they will behave in a time-consistent manner in all future periods, despite the fact that they have consistently violated this belief in the past.

The naive hyperbolic consumer solves the problem

$$\begin{aligned} \max_{c, a^+} \quad & u(c) + \hat{\delta} \beta \psi_{j+1} E \left[\hat{V}(z^+) | \eta \right] \\ \text{s.t.} \quad & a^+ = (1 + r)a + wh + pen - c, \quad a^+ \geq 0, \end{aligned}$$

where $\hat{\delta} < 1$ represents an additional discount factor and $\hat{V}(\cdot)$ denotes the value function of a rational agent, i.e.

$$\begin{aligned} \hat{V}(z^+) = \max_{\hat{c}^+, \hat{a}^{++}} \quad & u(\hat{c}^+) + \beta \psi_{j+2} E \left[\hat{V}(z^{++}) | \eta^+ \right] \\ \text{s.t.} \quad & \hat{a}^{++} = (1 + r)\hat{a}^+ + wh + pen - \hat{c}^+, \quad \hat{a}^{++} \geq 0. \end{aligned}$$

We can use the actual behaviour to compute the agent's welfare

$$V(z) = u(c) + \beta \psi_{j+1} E \left[V(z^+) | \eta \right].$$

Implement the hyperbolic consumer model by solving the consumption path of the baseline model successively with $\hat{\delta} = 1$ and with $\hat{\delta} = 0.8$. Compare the consumption and the asset path between the rational and the hyperbolic consumer model and show how many agents are borrowing-constrained in each of these setups. Explain the differences in economic terms.

10.4. Consider the following time-separable expected utility function

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=2}^j \psi_i(m_{i-1}) \right) u_m(c_j) \right]$$

used in De Nardi, French, and Jones (2010). Instantaneous utility is given by

$$u_m(c_j) = \delta(m_j) \frac{c_j^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \quad \text{with} \quad \delta(m_j) = 1 - \delta m_j.$$

The indicator variable $m_j \in [0, 1]$ denotes the health (medical) status of an agent, which is either ‘good’ ($m_j = 0$) or ‘poor’ ($m_j = 1$). The preference shifter $\delta \in [-1; 1]$ determines how health status affects utility from consumption. In addition, the health status of the previous age affects survival probabilities so that $\psi_j(m_{j-1} = 1) = \chi \cdot \psi_j(m_{j-1} = 0)$ with $\chi < 1$. There are no health costs so the budget constraint is identical to the one in the baseline model. The transition probabilities from one health status to the other $\pi_{j,\theta,mm'} \in [0; 1]$ depend on age j and the permanent income class θ . We assume that the probabilities of suffering from poor health in the future when current health is good increase linearly from pg_1 to pg_J over the life cycle in both income classes. Similarly, the probabilities of suffering from poor health in the future when current health is poor increase linearly from pb_1 to pb_J . The state vector changes to $z = (j, a, \theta, m, \eta)$

- a) Set up the dynamic optimization problem and derive the first-order conditions. Then implement the model numerically with $\delta = 0.1$, $\chi = 1$, $[pg_1; pg_J] = [0; 0]$, and $[pb_1; pb_J] = [1; 1]$. Consequently, households are assigned a good and a poor health status in the first working year and then remain there forever. Compare the consumption and asset profiles of healthy and non-healthy households over the life cycle. How do changes in the parameters δ and χ affect these profiles? Explain in economic terms.
- b) Now consider transitions between health status by setting $[pg_1; pg_J] = [0.1; 0.4]$ and $[pb_1; pb_J] = [0.6; 0.9]$. How does this change the consumption and asset profiles of healthy and non-healthy households when we set $\delta = 0.1$ and $\chi = 1$. Provide an example of why even a preference shifter $\delta = -0.1$ might be justified.

10.5. Take the model from the previous exercise and successively implement the following extensions:

- a) Assume that the transitional probabilities depend on the income class by setting $[pg_1; pg_J] = [0.0; 0.2]$ for the higher income class.
- b) In addition, let labour productivity also be affected by the health status if $j < j_r$:

$$h_j(m_j) = e_j \cdot \exp [\theta + \eta_j - \varrho_\theta m_j]$$

where $\varrho_1 = 0.2$ and $\varrho_2 = 0.1$.

- c) Finally assume that poor health causes out-of-pocket expenses, which depend on a deterministic age profile k_j and a stochastic term ζ_j that is normally distributed:

$$hc_j = k_j \cdot \exp \zeta_j \quad \text{with} \quad \zeta_j \sim N(0, \sigma_\zeta^2).$$

Households therefore maximize expected utility subject to the (periodical) budget constraints

$$a_{j+1} + hc_j + c_j = (1 + r)a_j + wh_j(m_j) + pen_j + b_j$$

and the non-negativity constraint for savings $a_{j+1} \geq 0$. Resources on the right-hand side include the government transfer b_j , which provides a consumption floor \underline{c} , i.e.

$$b_j = \max[\underline{c} + hc_j - (1 + r)a_j - wh_j(m_j) - pen_j; 0].$$

The state space now increases to $z = (j, a, \theta, m, \eta, \zeta)$. Implement this model numerically with linearly increasing deterministic costs $[k_1; k_J] = [0.3; 0.9]$, a variance $\sigma_\zeta^2 = 0.2$, and a consumption floor $\underline{c} = 0.05$.

Compare the consumption, labour income, and asset profiles of low-skilled and highly-skilled households, as well those of healthy and non-healthy households. How do the different parameters affect these profiles?

- 10.6. Consider the following time-separable expected utility function

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) \left\{ \frac{c_j^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \alpha \frac{(n_j^c)^{1-\frac{1}{\chi}}}{1 - \frac{1}{\chi}} \right\} \right],$$

where n_j^c defines the number of children living in the household, α the respective utility weight, and χ the respective intertemporal elasticity of substitution, see Sommer (2016).

For simplicity we exclude different lifetime-income classes but include infertility shocks $f_j = F, I$, which arrive at the beginning of every period with probability p_j^f . We assume that the probability of an infertility shock increases with a constant growth rate $u_f = 0.2$ from $[0.02; 1.0]$ until at age 25 (i.e. real age 45) all households are infertile. Only fertile parents (where $f_j = F$) can choose to have another child, while parents once hit by the infertility shock remain infertile forever (if $f_j = I$ then $f_{j+i} = I$ for all $i > 0$). The number of children ever born to the household follows a deterministic process $n_{j+1}^b = n_j^b + k_j$, where $k_j \in \{0, 1\}$ determines the choice of the household to have a child ($k_j = 1$) or not ($k_j = 0$). The number of children who live in the households is distributed binomially with $n_{j+1}^c = B(n_j^c + k_j, p)$. $p = 0.9$ defines the time-invariant probability that a child remains at home. Parents only need to spend resources on their children as long as they are dependent and living in the household, so that the budget constraint is defined by

$$a_{j+1} + c_j = (1 + r)a_j + wh_j(1 - (n_j^c)^v l_c) + pen_j.$$

$l_c = 0.3$ defines the time cost of children and $v = 0.5$ determines the extent of economies of scale from having an additional child. Consequently the state space is $z = (j, a, n^c, \eta, f)$.

- Set up the dynamic optimization problem for fertile and infertile households and implement the model numerically with utility parameters $\alpha = 0.12$ and $\chi = 5$, a replacement rate $\kappa = 0.25$, an autocorrelation term of $\rho = 0.95$, the variance of the innovation term set at $\sigma_\epsilon = 0.22$, and the maximum possible number of children set at 10. Compute the number of (previous) births at age 30 and at age 45 as well as the average age at which the first and the second child are born. Finally, determine the number of children living in the household when the parents are aged 37.
- Compare the impact on birth numbers and birth ages when
 - the autocorrelation term is increased to $\rho = 0.99$;
 - the growth rate of infertility shocks is reduced to $u_f = 0.1$;
 - the replacement rate is increased to $\kappa = 0.4$.

Explain the changes in economic terms.

10.7. Consider the model with endogenous labour supply.

- Solve the model using value function iteration.
- Compare the Euler equation error (EEE) in the benchmark model with policy function iteration and the model with value function iteration. Note: The first-order condition only holds for $a^+ > 0$!

10.8. Consider the following time-separable expected utility function

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=2}^j \psi_i(m_{i-1}) \right) u(c_j, 1 - l_j, m_j) \right],$$

where instantaneous utility is

$$u(c_j, 1 - l_j, m_j) = \frac{\left[c_j^\gamma (1 - l_j - \phi_l m_j)^{1-\gamma} \right]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

As in exercise 10.4 the indicator variable $m_j \in [0, 1]$ denotes health (or medical) status, which affects survival probabilities but now also reduces the time available for work and leisure by $\phi_l \in [0; 1[$. We exclude out-of-the-pocket health cost so that the budget constraint does not change. Besides χ and ϕ_l we apply the same transition probabilities $\pi_{j,m,m'}$ as in exercise 10.4. Since we abstract from different skills, the state vector is $z = (j, a, m, \eta)$.

- a) Set up the dynamic optimization problem, derive the first-order condition and implement the model numerically. Compare the consumption, labour earnings, and asset profiles of healthy and non-healthy households over the life cycle when we either have $\phi_l = 0.2$ or $\chi = 0.8$. How do the two parameters affect these profiles? Explain in economic terms.
- b) Now assume that labour productivity is affected by the health status simply by assuming

$$h_j(m_j) = \begin{cases} e_j \cdot \exp[\eta_j - \varrho m_j] & \text{if } j < j_r \quad \text{and} \\ 0 & \text{if } j \geq j_r \end{cases}$$

where $\varrho = 0.2$. What are the consequences for the life-cycle profiles of households with good and poor health?

10.9. Consider an alternative form of preferences

$$u(c, l) = \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} - \nu \frac{l^{1+\frac{1}{\chi}}}{1 + \frac{1}{\chi}}$$

with $\gamma = 0.5$, $\chi = 0.6$ and $\nu = 25$.

- a) Set up the dynamic optimization problem, derive the first-order conditions, and implement the model numerically with policy function iteration.
- b) Now implement the model using the endogenous grid-point method.

- 10.10. Introduce a labour-income tax function $T(wl) = whl - (1 - \tau)(whl)^\varrho$ into the model with variable labour supply from exercise 10.9 where τ is (approximately) the average tax rate and ϱ a parameter for the progressiveness of the tax code.

- Set up the dynamic optimization problem, derive the first-order conditions, and implement the model numerically.
- Compare the tax revenues and aggregate labour supply with progressive ($\tau = 0.25, \varrho = 0.8$) and proportional ($\tau = 0.2, \varrho = 1$) taxes.

- 10.11. Attanasio, Low, and Sanchez-Marcos (2008) use the utility function

$$u(c, l) = \frac{1}{1 - \frac{1}{\gamma}} \left[\frac{c}{\sqrt{2 + n}} \right]^{1 - \frac{1}{\gamma}} \exp(v_1 \cdot l) - v_2 \cdot l$$

with $v_1 = 0.04$ in their model of female labour-force participation. How do such consumption utility costs of labour-force participation affect the life-cycle profiles of households?

- 10.12. Solve the model with female labour-force participation using value function iteration.

- 10.13. Assume that the survival probabilities of household members were independent. If one of the partners dies, the government pays for all remaining childcare costs, but the pension is reduced by 50 per cent. The total amount of household wealth is inherited by the surviving spouse.

- Formulate the optimization problem for different household types (married couple and widowed spouses) and implement the model numerically.
- Compare the aggregate savings behaviour with that from the corresponding benchmark model. Explain in economic terms!

Note: Instead of introducing a new state dimension it is more convenient to add a zero to the productivity state. We then let

$$z = (j, a, h_f, \theta, \eta_m, \eta_f), \quad z_f = (j, a, h_f, \theta, 0, \eta_f) \text{ and } z_m = (j, a, 0, \theta, \eta_m, 0)$$

define the states in which both members, only the female and only the male household member are living, respectively. You can check the consistency of your model by simulating both the benchmark model and the new one with a fixed and certain lifespan.

- 10.14. In the baseline model with female labour-force participation, we assumed that the wife's labour income has no impact on the size of the pension the household receives. Now we distinguish between the male pension $pen_{m,j}$ and the female pension $pen_{f,j,q}$. The former is computed as in the baseline model, but the replacement rate κ is reduced from 0.8 to 0.4. The latter depends on the stock of human capital of the woman in the last period, i.e.

$$pen_{f,j,q} = \chi \cdot w_f \cdot h_{f,j_r}$$

Implement such a separate pension for the wife and explain the resulting behavioural changes when the replacement rate is set to $\chi = 0$ initially. Then increase χ to 0.5 and 1.0 and discuss the economic consequences.

10.15. Compute

- a) the cohort-specific shares of investors who hold more than 50 per cent of their wealth in equity,
- b) the shares of these big equity investors in the 5th, 25th, 50th, 75th, and the 95th percentile of the wealth distribution, and
- c) the equity shares of the 5th, 25th, 50th, 75th, and the 95th percentile from the benchmark portfolio choice model and display them in a single graph.

10.16. Solve the portfolio choice model using value function iteration.

10.17. Solve the portfolio choice model with entry cost F . In this case the optimization problem changes to

$$V(j, X, f) = \max_{c, a^+, \omega^+, f^+} \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + \beta \psi_{j+1} E \left[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} \cdot V(j+1, X^+, f^+) \right]$$

s.t. $a^+ = X - c - (f^+ - f)F, \quad a^+ \geq 0, \quad 0 \leq \omega^+ \leq 1$

$$f^+ = \begin{cases} 1 & \text{if } f = 1 \text{ or if } f = 0 \text{ and } \omega^+ > 0 \\ 0 & \text{otherwise.} \end{cases}$$

$$X^+ = R_p(\omega^+, \vartheta^+) \frac{a^+}{\exp(\epsilon^+)} + wh^+ + pen^+$$

so that the (normalized) state vector changes to $z = (j, X, f)$ and the optimal portfolio share is given by $\omega^+(j, a^+, f^+)$. The state $f \in (0, 1)$ is an indicator variable which is zero as long as the investor has not invested in stocks. Consequently $\omega^+(j, a^+, 0) = 0$. As soon as the investor makes the first investment in stocks, he has to pay a fixed cost F and f^+ changes from zero to 1. After the first investment in stocks has been made, f^+ will remain at 1 forever.

Implement the model with entry cost $F = e_1$, then double the entry cost and discuss the share of stock investors in the different asset quintiles.

10.18. Bremus and Kuzin (2014) extend the benchmark portfolio choice model by considering the possibility of short- and long-term unemployment. Consequently the state space $z = (j, X, s)$ includes the current employment state $s \in S = \{e, u_s, u_l\}$. If $s = e$, the household supplies labour inelastically, if $s = u_s$ or $s = u_l$, the household

is short-term or long-term unemployed. Employment opportunities are assumed to follow a first-order Markov-chain with the transition matrix

$$\Pi(s', s) = \begin{bmatrix} \pi_{ee} & \pi_{eu_s} & \pi_{eu_l} \\ \pi_{u_se} & \pi_{u_su_s} & \pi_{u_su_l} \\ \pi_{u_le} & \pi_{u_lu_s} & \pi_{u_lu_l} \end{bmatrix} = \begin{bmatrix} 0.95 & 0.05 & 0.00 \\ 0.89 & 0.09 & 0.02 \\ 0.15 & 0.00 & 0.85 \end{bmatrix}$$

Assume that all households start in employment. When they get unemployed at some age, they receive an unemployment benefit. The replacement rate χ_k for this benefit falls from $\chi_s = 0.64$ to $\chi_l = 0.36$. Unemployment benefits are computed as a fraction of permanent income of that specific age j , i.e. $\chi_k w e_j \exp(\eta_j)$ and pension benefits of the unemployed are computed as a fraction of the last unemployment benefit.

- a) Set up the optimization problem and solve the model in two steps as in the benchmark case.
- b) Check whether the model replicates the benchmark solution with $\pi_{ee} = 1$.
- c) Introduce the transition possibilities towards unemployment and compare the equity shares of employed and unemployed households.

10.19. Gomes, Kotlikoff, and Viceira (2008) analyse optimal portfolio choice in a model with variable labour supply, where individual preferences are given by

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) \frac{[c^\nu (1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right].$$

Since labour hours are now a choice variable, ‘cash-on-hand’ is defined by $X = R_p(\omega, \vartheta)a + pen$ and the (already normalized) state space is defined by $z = (j, X, \zeta)$, where ζ defines the white noise term of labour productivity.

- a) Set up the optimization problem and solve the model in two steps as in the benchmark case. Note: The consumption–labour–savings decision is solved quite similarly as in Module 10.2m.
- b) Simulate the model with $\nu = 0.4$ and $\nu = 0.5$ and explain the observed changes in the equity share.

10.20. Analyse the portfolio choice model when household preferences are given by

$$V(j, X) = \left\{ c^{1-\frac{1}{\gamma}} + \beta \psi_{j+1} E \left[[\exp(\epsilon^+) V(j+1, X^+)]^{1-\chi} \right]^{\frac{1-\frac{1}{\gamma}}{1-\chi}} \right\}^{\frac{1}{1-\frac{1}{\gamma}}}.$$

χ denotes the risk-aversion, see Epstein and Zin (1989). Specify the optimization problem and derive the solution in two steps as in the benchmark. Implement the model numerically. If $\chi = 10$ the model should generate the same solution as the benchmark model. Set $\chi = 5$ and explain the changes in behaviour in economic terms.

- 10.21. Consider a case in which households are allowed to buy annuities at each age j and not only at the age prior to retirement. Consequently household split their total wealth a into retirement assets $a_r = \omega_r a$ and liquid wealth $a_l = (1 - \omega_r)a$. The latter is again split into stocks ωa_l and bonds $(1 - \omega)a_l$. Retirement assets bought at age j generate an annuity stream $\frac{a_r^+}{p_{a,j}}$ starting in the next period $j + 1$. Total annuity income for the following period can therefore be computed from

$$y_a^+ = \left[y_a + \frac{a_r^+}{p_{a,j}} \right].$$

Household preferences are given by

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) \left\{ u(c_j) + \beta(1 - \psi_{j+1}) \mathcal{B}(a_{l,j+1}) \right\} \right]$$

where

$$\mathcal{B}(a_l) = \nu \frac{(R_p(\omega, \vartheta) \cdot a_l)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

defines a bequest motive with ν capturing its strength. The state space changes now to $z = (j, X, y_a)$.

- a) Specify the normalized optimization problem and derive the model solution in three steps.
 - b) Implement the annuity model numerically and simulate it without a bequest motive. Discuss the annuity demand over the life cycle.
 - c) Now set $\nu = 10$ and explain the observed changes in behaviour.
- 10.22. In this exercise, we consider the case of a housing asset, which is consumed as durable good. Households split their total wealth a into housing wealth $a_h = \omega_h a$ and liquid wealth $a_l = (1 - \omega_h)a$. The latter is again distributed over stocks ωa_l and bonds $(1 - \omega)a_l$. Houses depreciate at rate δ_h in every period. Household preferences are

$$E \left[\sum_{j=1}^J \beta^{j-1} \left(\prod_{i=1}^j \psi_i \right) \left\{ u(c_j, a_{h,j}) + \beta(1 - \psi_{j+1}) \mathcal{B}(a_{l,j+1}, a_{h,j+1}) \right\} \right],$$

where

$$u(c, a_h) = \theta \frac{c^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} + (1 - \theta) \frac{a_h^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

defines the instantaneous utility function with γ as intertemporal elasticity of substitution for ordinary and housing consumption and θ as a share parameter for ordinary consumption. In addition,

$$\mathcal{B}(a_l, a_h) = \nu \frac{(R_p(\omega, \vartheta) \cdot a_l + (1 - \delta_h) \cdot a_h)^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}$$

defines the bequest function with ν capturing the strength of the bequest motive. The state vector changes to $z = (j, X, a_h)$. Households are restricted to a minimum house size h_{min} and to a maximum loan-to-value ratio ξ . In the event that they borrow against the value of the house, agents face a fixed interest rate of $r_f + 0.02$. Consequently the (normalized) optimization problem changes to

$$\begin{aligned} V(z) = & \max_{c, a^+, \omega^+, \omega_h^+} u(c, a_h) + \beta \psi_{j+1} E[\exp(\epsilon^+)^{1-\frac{1}{\gamma}} V(z^+)] \\ & + \beta(1 - \psi_{j+1}) E[\mathcal{B}(a_l^+, a_h^+)] \end{aligned}$$

subject to

$$X = c + a^+, \quad a_h^+ \geq h_{min}, \quad a_l^+ \geq -\xi a_h^+, \quad 0 \leq \omega^+, \omega_h^+ \leq 1,$$

$$X^+ = \frac{R_p(\omega^+, \vartheta^+) a_l^+}{\exp(\epsilon^+)} + w h^+ + pen^+ + \frac{(1 - \delta_h) a_h^+}{\exp(\epsilon^+)}$$

and

$$R_p(\omega^+, \vartheta^+) = \begin{cases} 1 + r_f + \omega^+(j, a_l^+, a_h^+) \cdot (\mu_r + \vartheta^+) & \text{if } a_l^+ > 0 \\ 1 + r_f + 0.02 & \text{otherwise.} \end{cases}$$

- a) Derive the normalized optimization problem as well as the first-order conditions in three steps.
- b) Implement the housing model numerically with a housing grid $[h_l, h_u] = [a_l, a_u]$ and the parameters values $\theta = 0.35$, $\gamma = 0.1$, $\delta_h = 0.01$, $\kappa = 0.5$, as well as $h_{min} = \xi = \nu = 0$.

- c) Now allow for borrowing with a loan-to-value ratio $\xi = 0.7$. Discuss the resulting housing demand over the life cycle.
- d) In addition to $\xi = 0.7$ introduce a minimum house size of $h_{min} = e_1$ and explain the observed changes in behaviour.
- e) Finally introduce a bequest motive with $\nu = 10$ and explain the observed changes compared to exercise 10.21.

11 Dynamic macro II: The stochastic OLG model

In Chapters 6 and 7 we discussed how to compute overlapping generations models and how to use them for policy analysis. The models developed there are completely *deterministic* in that they exclude both income and investment risk. While this ensured analytical tractability of the household problem and greatly facilitated computation, it certainly limit the scope of policy analysis. Consequently these chapters centred around clarifying the impact of public policy on the labour-supply and savings decisions of households and around evaluating its consequences for intergenerational redistribution.

In practice, however, households face all kinds of risks that cannot be insured perfectly by the market. This opens up an additional channel through which the government could increase households' welfare, namely by providing public insurance. In Chapter 10, we studied individual behaviour in an uncertain world, where individuals face idiosyncratic labour income and mortality risk as well as aggregate capital-market risk. The models therein are partial equilibrium models, meaning that prices are fixed and there is no need for the government to operate a balanced budget. In this chapter, we embed a household's decision model with idiosyncratic labour-productivity risk and endogenous labour-supply decisions into a general equilibrium framework, which leads us to the stochastic OLG model. In this setup, factor prices respond to changes in individual behaviour and the government will be an explicit entity that collects revenue from taxes to finance its expenditure. Such a setup allows us to analyse both the distortionary and the *risk-sharing* effects of public policies.

This chapter is organized in three main sections. The first two closely follow the setup of Chapter 6. We first explain the general structure of the stochastic OLG model with all its actors and conduct some steady-state policy analysis. We then discuss how to extend the model to include a transition path between steady states and to compute aggregate efficiency effects. In the last section, Section 11.3, we provide some policy applications where we analyse optimal tax schedules and the optimal size of the social-security system in more detail.

11.1 General structure and long-run equilibrium

In the following, we extend the life-cycle model with variable labour supply from Section 10.1.2 to a full general equilibrium setup with overlapping generations. This section describes the theoretical structure of the model, how we implement it, and

how we calibrate parameters. Eventually, we conduct some long-run analysis of policy reforms. Many parts of this section will already sound familiar. Individual decision-making works (almost) exactly like in the previous chapter. The macroeconomic setup, on the other hand, is quite similar to that in Chapters 6 or 9. Hence, we will be brief on things we have discussed before, but as detailed as necessary to paint a full picture of the model.

11.1.1 DEMOGRAPHICS, BEHAVIOUR, AND MARKETS

Demographics At each point in time t , the economy is populated by J overlapping generations indexed by $j = 1, \dots, J$. Individuals live up to age J and die with certainty afterwards. Hence, in contrast to Chapter 10, there is no uncertain survival. Cohort sizes grow over time at the constant rate n_p . Let N_t denote the size of the cohort that enters the labour market at time t , then we have

$$N_t = (1 + n_p)N_{t-1},$$

see Figure 6.1. As the population size is growing over time at rate n_p , on a balanced growth path all aggregate variables grow at rate n_p . As in Chapters 6 and 7, we therefore normalize aggregate variables at time t by the size of the youngest cohort living in this period. On a balanced growth path these normalized aggregates are then constant. Note that since population growth is constant over time, so are the relative population shares

$$m_j = \frac{N_{t-j+1}}{N_t} = (1 + n_p)^{1-j}.$$

Preferences and labour-productivity risk Individuals have preferences over streams of consumption $c_{j,t}$ and leisure $\ell_{j,t}$. Like in earlier chapters, we assume a time endowment of 1. With $l_{j,t}$ denoting the amount of labour hours supplied to the market in period t , we have $\ell_{j,t} + l_{j,t} = 1$. The utility function of the household reads

$$U_0 = E \left[\sum_{j=1}^J \beta^{j-1} u(c_{j,t}, 1 - l_{j,t}) \right], \quad (11.1)$$

where the expectation is from an ex ante perspective, i.e. before any information about labour productivity has been revealed to the individual. We again use the specific preferences

$$u(c_{j,t}, 1 - l_{j,t}) = \frac{[(c_{j,t})^\nu (1 - l_{j,t})^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}}.$$

Individuals differ with respect to their labour productivity $h_{j,t}$, which depends on a (deterministic) age profile of earnings e_j , a fixed productivity effect θ that is drawn at the beginning of the life cycle, and an autoregressive component $\eta_{j,t}$ that evolves over time. At the mandatory retirement age j_r , labour productivity falls to zero and households receive a flat pension benefit $pen_{j,t}$ computed as a fraction κ of average labour income in period t . Households maximize the expected utility function (11.1) subject to the (periodical) budget constraints

$$a_{j+1,t} = (1 + r_t^n) a_{j,t} + w_t^n h_{j,t} l_{j,t} + pen_{j,t} - p_t c_{j,t} \quad (11.2)$$

where $w_t^n = w_t(1 - \tau_t^w - \tau_t^P)$, $r_t^n = r_t(1 - \tau_t^r)$, and $p_t = 1 + \tau_t^c$ denote the net wage rate, the net interest rate, and the consumer price, respectively. τ_t^c , τ_t^w , τ_t^P , and τ_t^r are consumption, labour-income, payroll, and capital-income tax rates. Finally, the household has to respect a non-negativity constraint on savings $a_{j+1,t} \geq 0$ at all ages in every period.

The dynamic programming problem The optimization problem of households reads

$$\begin{aligned} V_t(z) &= \max_{c, l, a^+} u(c, 1 - l) + \beta E[V_{t+1}(z^+) | \eta] \\ \text{s.t. } &a^+ + p_t c = (1 + r_t^n)a + w_t^n h l + pen, \quad a^+ \geq 0, \quad l \geq 0 \\ &\text{and } \eta^+ = \rho \eta + \epsilon^+ \quad \text{with } \epsilon^+ \sim N(0, \sigma_\epsilon^2), \end{aligned} \quad (11.3)$$

where $z = (j, a, \theta, \eta)$ again is the vector of individual state variables. Note that we put a time index on the value function and on prices. This will be necessary as soon as we compute transitional dynamics of the model. The terminal condition for the value function is

$$V_T(z) = 0 \quad \text{for } z = (J+1, a, \theta, \eta),$$

which means we assume that the household doesn't value what is happening after death.

Applying the same steps as in Chapter 10, we can formulate the solution to the household problem by recognizing that we can write labour hours and consumption as functions of a^+ as

$$l = l(a^+) = \min \left\{ \max \left[\nu + \frac{1-\nu}{w_t^n h} (a^+ - (1 + r_t^n)a - pen), 0 \right], 1 \right\} \quad (11.4)$$

$$c = c(a^+) = \frac{1}{p_t} [(1 + r_t^n)a + w_t^n h l(a^+) + p e n - a^+]. \quad (11.5)$$

The household problem then reduces to solving the first-order condition

$$\frac{\nu [c(a^+)^v (1 - l(a^+))^{1-v}]^{1-\frac{1}{\gamma}}}{p_t c(a^+)} = \beta (1 + r_{t+1}^n) \cdot E \left[\frac{\nu [c_{t+1}(z^+)^v (1 - l_{t+1}(z^+))^{1-v}]^{1-\frac{1}{\gamma}}}{p_{t+1} c_{t+1}(z^+)} \middle| \eta \right], \quad (11.6)$$

where a^+ is the unknown.

Aggregation In order to aggregate individual decisions at each element of the state space to economy-wide quantities, we need to determine the distribution of households $\phi_t(z)$ across the state space. For the sake of simplicity, we assume that we have already discretized the state space. Then we can apply exactly the same procedure as in Chapter 10. Specifically, we know that at age $j = 1$ households hold zero assets, experience a permanent productivity shock $\hat{\theta}_i$ with probability π_i^θ , as well as a transitory productivity shock of $\eta_1 = 0$. Hence, we have

$$\phi_t(1, 0, \hat{\theta}_i, \hat{\eta}_g) = \begin{cases} \pi_i^\theta & \text{if } g = \frac{m+1}{2} \text{ and} \\ 0 & \text{otherwise.} \end{cases}$$

Knowing the distribution of households over the state space at age 1, we can compute the distribution at any successive age–year combination using the policy function $a_t^+(z)$. Specifically, for each element of the state space z at age j and time t , we compute the left and right interpolation nodes \hat{a}_l and \hat{a}_r , as well as the corresponding interpolation weight φ . The nodes and the weight satisfy

$$a_t^+(z) = \varphi \cdot \hat{a}_l + (1 - \varphi) \cdot \hat{a}_r.$$

Taking into account the transition probabilities for the transitory productivity shock η_{gg^+} , we then distribute the mass of individuals at state z to the state space at the next age $j+1$ and year $t+1$ according to

$$\phi_{t+1}(z^+) = \begin{cases} \phi_{t+1}(z^+) + \varphi \cdot \pi_{gg^+} \cdot \phi_t(z) & \text{if } v = l, \\ \phi_{t+1}(z^+) + (1 - \varphi) \cdot \pi_{gg^+} \cdot \phi_t(z) & \text{if } v = r \end{cases}$$

with $z^+ = (j+1, \hat{a}_v, \hat{\theta}_i, \hat{\eta}_g)$.

Note that the distributional measure $\phi_t(z)$ satisfies

$$\sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi_t(z) = 1$$

for any age j and time t . We can hence use it to calculate cohort-specific aggregates

$$\begin{aligned}\bar{c}_{j,t} &= \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi_t(z) \cdot c_t(z), & \bar{l}_{j,t} &= \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi_t(z) \cdot h_t(z) l_t(z), & \text{and} \\ \bar{a}_{j,t} &= \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m \phi_t(z) \cdot \hat{a}_v.\end{aligned}$$

From these cohort values, we can in turn generate economy-wide quantities. We have only to weight the cohort variables with by the respective relative cohort sizes m_j . Consequently we get

$$C_t = \sum_{j=1}^J m_j \cdot \bar{c}_{j,t}, \quad L_t^s = \sum_{j=1}^J m_j \cdot \bar{l}_{j,t}, \quad \text{and} \quad A_t = \sum_{j=1}^J m_j \cdot \bar{a}_{j,t}. \quad (11.7)$$

Firms Firms' behaviour is identical to the RBC or the heterogeneous agent model of Chapter 9. Consequently firms hire capital K_t and labour L_t on perfectly competitive factor markets to be transformed into a single output good Y_t according to the Cobb-Douglas production technology

$$Y_t = \Omega K_t^\alpha L_t^{1-\alpha},$$

with Ω being the technology level which is constant over time. Capital depreciates at rate δ , so that the capital stock evolves as

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

Under the assumption of perfect competition, the (inverse) demand functions of the firm for capital and labour are

$$r_t = \alpha \Omega \left[\frac{L_t}{K_t} \right]^{1-\alpha} - \delta \quad \text{and} \quad w_t = (1 - \alpha) \Omega \left[\frac{K_t}{L_t} \right]^\alpha. \quad (11.8)$$

The government The last actor in our model is the government. In our model it runs two separate systems: the tax system and the pension system, which both operate on

a balanced-budget basis. The government collects taxes on consumption expenditure, labour income, and interest income in order to finance public expenditure G_t and payments related to the stock of debt B_t . In the initial equilibrium, we let government expenditure be a constant share of GDP, that is, $G = g_y Y$. In later periods, the level of public goods is kept constant (per capita), meaning we have $G_t = G$. The same applies to public debt, where the initial share is denoted by b_y . At any point in time the budget of the tax system is balanced if

$$\tau_t^c C_t + \tau_t^w w_t L_t^s + \tau_t^r r_t A_t + (1 + n_p) B_{t+1} = G_t + (1 + r_t) B_t. \quad (11.9)$$

In addition to revenue from taxation, the government finances expenditure from issuing new debt $(1 + n_p) B_{t+1}$. However it has to repay current debt including interest payments so we have to add $(1 + r_t) B_t$ to government consumption on the expenditure side. Therefore, in a steady-state equilibrium, expenditure $(r - n_p) B$ reflects the cost needed to keep the debt level constant. Note that we do not make any a priori restriction about which tax rate has to adjust in order to balance the budget over time.

The pension system operates on a pay-as-you-go basis, meaning that it collects contributions from working-age generations and directly redistributes them to current retirees. There is no capital accumulation process involved. The budget-balance equation of the pension system then reads

$$\tau_t^p w_t L_t^s = \overline{pen}_t \cdot N^R \quad \text{with} \quad N^R = \sum_{j=j_r}^J m_j \quad (11.10)$$

where N^R denotes the (fixed) number of retirees. Since payments are made to all retirees in a lump-sum fashion, one simply has to add up the relative cohort sizes of the retired generations on the expenditure side and multiply this number with the respective benefit. For the evolution of pension payments over time we assume that they are linked to the average labour earnings of the previous period, i.e.

$$\overline{pen}_t = \kappa_t \cdot \frac{w_{t-1} L_{t-1}^s}{N^L} \quad \text{with} \quad N^L = \sum_{j=1}^{j_r-1} m_j,$$

where κ_t is the replacement rate of the pension system and N^L is the (fixed) size of working-age cohorts. We assume that the replacement rate κ_t is given exogenously while the contribution rate τ_t^p adjusts in order to balance the budget.

Markets There are three markets in our economy: the factor markets for capital and labour and the goods market. On the factor markets, prices for capital r_t and labour w_t adjust so that markets clear, that is

$$K_t + B_t = A_t \quad \text{and} \quad L_t = L_t^s. \quad (11.11)$$

Note that there are two sectors that demand savings from the households. The firm sector employs savings as capital in the production process, the government uses its public debt in order to finance expenditure. By assumption the government and the firm sector perfectly compete on the capital market. On the goods market, all output produced must be used either as consumption of the private sector or the government or as investment into the future capital stock. The goods market equilibrium therefore reads

$$Y_t = C_t + G_t + I_t. \quad (11.12)$$

Using Walras' law we know that if the factor markets of the economy clear, then the goods market will also be in equilibrium.

Equilibrium definition Last but not least we want to formally define an equilibrium path of our OLG economy.

Definition (Equilibrium path) *Given a path for government expenditure and debt $\{G_t, B_t\}_{t=0}^\infty$, a path for tax rates $\{\tau_t^c, \tau_t^w, \tau_t^r\}_{t=0}^\infty$, and a characterization of the pension system, $\{\tau_t^p, \kappa_t\}_{t=0}^\infty$, a recursive competitive equilibrium of the economy is a set of policy functions*

$$\{c_t(z), l_t(z), a_t(z)\}_{t=0}^\infty$$

for the households, a set of input choices $\{K_t, L_t\}_{t=0}^\infty$ for the firms, prices $\{r_t, w_t\}_{t=0}^\infty$ and a measure of households $\{\phi_t\}_{t=0}^\infty$ that are such that

1. [Household maximization]: Given prices $\{r_t, w_t\}_{t=0}^\infty$, the policy functions solve the household optimization problem as stated in (11.3).
2. [Firms maximization]: Given prices, the firms' factor inputs satisfy their demand equations in (11.8).
3. [Government budget constraints]: The government budget constraints (11.9) and (11.10) are satisfied.
4. [Market clearing]: The market-clearing equations (11.11) and (11.12) hold with aggregate quantities being computed from (11.7).
5. [Consistency of the measure of households]: The measure of households is consistent with the assumptions about stochastic processes and individual decisions.

The above definition applies to any general equilibrium path of the economy. We can furthermore specify a long-run equilibrium of the model or a steady state.

Definition (Long-run equilibrium) *A long-run equilibrium of the economy is an equilibrium path on which prices, tax rates, and all individual variables are constant over time and aggregate quantities all grow at the rate of the population n_p .*

11.1.2 NUMERICAL IMPLEMENTATION OF STEADY-STATE EQUILIBRIUM

To solve our stochastic OLG model numerically, we combine the policy function iteration of Chapter 10 for the life-cycle model with the Gauss-Seidel iteration procedure of Chapter 6 for the macroeconomic equilibrium. We have already discussed all the details of the policy function iteration method in Chapter 10, including the discretization of the state space, how to solve the first-order condition, and how to determine the distribution of households across the state space. As there is literally no change to the method, we shall not repeat this. At this stage the important thing is that the policy function iteration method allows us to determine cohort-specific averages, which can be aggregated to economy-wide quantities C , L^s , and A , see (11.7). In a steady-state equilibrium, these (normalized) quantities are, of course, constant over time. Note that the solution to the household problem depends on prices and tax rates p , r^n , w^n , as well as the pension payment $\bar{p}en$. The same is then obviously true for aggregate quantities.

The macroeconomic iteration procedure We use the Gauss-Seidel iteration procedure discussed in Chapters 2 and 6 to determine the macroeconomic equilibrium of our economy. Program 11.1 shows how this procedure is applied. After initializing the model parameters and providing initial guesses for aggregate capital and labour input as well

Program 11.1 Gauss-Seidel iteration to solve for the steady state

```

subroutine get_SteadyState()
[.....]
! iterate until value function converges
do iter = 1, itermax

    ! compute prices
    call prices()

    ! solve the household problem
    call solve_household()

    ! calculate the distribution of households over state space
    call get_distribution()

    ! aggregate individual decisions over cohorts
    call aggregation()

    ! determine the government parameters
    call government()

    write(*, '(i4,5f8.2,f12.5)')iter, (/5d0*KK, CC, &
                                         II//)/YY*100d0, r, w, DIFF/YY*100d0
    if(abs(DIFF/YY)*100d0 < sig)then
        call toc
        call output()
        return
    endif
enddo
[.....]
end subroutine

```

as the pension payment in subroutine `initialize`, the algorithm proceeds according to the following steps:

1. Given guesses for capital and labour input as well as tax rates, compute factor and consumer prices using the subroutine `prices`.
2. Given prices and public pension payments, determine household policy functions using subroutine `solve_household`.
3. Compute the distribution of households over the state space using subroutine `get_distribution`.
4. Aggregate household decisions to quantities with subroutine `aggregation`.
5. Determine new taxes and the pension level with subroutine `government`.
6. Calculate the absolute value of the relative difference between demand $C + G + I$ and supply Y of goods. If this difference is small enough, we have found the equilibrium and can stop the iteration procedure. If not, start again at point 1.

There are two subroutines involved in this process that we haven't discussed so far. The aggregation procedure and the subroutine that determines the parameters of the tax and pension system.

An excerpt of the aggregation procedure is shown in Program 11.1.a. First of all this routine stores the current value for labour supply in a variable `LL_old`. It then calculates cohort averages for important variables such as consumption, assets, and labour input.

Program 11.1.a Aggregating individual decisions to quantities

```
subroutine aggregation()
[.....]
LL_old = LL

! calculate cohort aggregates
c_coh(:) = 0d0
[.....]

! calculate aggregate quantities
CC = 0d0
[.....]
do ij = 1, JJ
    CC = CC + c_coh(ij)*m(ij)
    [.....]
enddo

! damping and other quantities
KK = damp*(AA-BB) + (1d0-damp)*KK
LL = damp*LL + (1d0-damp)*LL_old
II = (n_p+delta)*KK
YY = Omega * KK**alpha * LL** (1d0-alpha)
[.....]
! get difference on goods market
DIFF = YY-CC-GG-II

end subroutine
```

In the next step, we aggregate cohort averages to economy-wide quantities using the relative population shares stored in the array m . Having fully aggregated household decisions, we can calculate an updated level of the capital stock and aggregate labour supply using the factor-market-clearing conditions. Note that the Gauss-Seidel procedure involves using a damping factor $damp$, which ensures that the new levels of capital and labour are some linear combination of the values from the previous iteration step and the newly calculated aggregate quantities. Having derived new values for aggregate capital and labour, we can determine investment and total output. Last but not least, we calculate the difference between supply and demand on the goods market.

Finally we need to compute the budget-balancing tax rates and the pension-system parameters. This is done in the subroutine `government` in Program 11.1.b. This subroutine starts with determining the expenditure side of the tax system. The fractions of government consumption and government debt to GDP are stored in the variables gy and by . Note that we only derive values for G and B when we calculate an initial equilibrium. The logical variable `reform_on` controls this calculation procedure and can be set to a value of `.true.` if the values for government consumption and debt are not updated.

Program 11.1.b Government parameters

```

subroutine government()
[.....]
if (.not. reform_on) then
    GG = gy*YY
    BB = by*YY
endif

! calculate government expenditure
expend = GG + (1d0+r)*BB - (1d0+n_p)*BB

! get budget balancing tax rate
if(tax == 1)then
    tauc = (expend - (tauw*w*LL + taur*r*AA))/CC
    p   = 1d0 + tauc
elseif(tax == 2)then
    tauw = (expend - tauc*CC) / (w*LL + r*AA)
    taur = tauw
elseif(tax == 3)then
    tauw = (expend - (tauc*CC + taur*r*AA)) / (w*LL)
else
    taur = (expend - (tauc*CC + tauw*w*LL)) / (r*AA)
endif
[.....]

! get budget balancing social security contribution
pen(JR:JJ) = kappa*INC
PP = 0d0
do ij = JR, JJ
    PP = PP + pen(ij)*m(ij)
enddo

taup = PP/(w*LL)

end subroutine

```

The total expenditure of the tax system that needs to be financed by taxes is the difference between government consumption GG plus debt repayment $(1+r) * BB$ and newly issued debt $(1+n_p) * BB$. Next we calculate the budget-balancing tax rate. The variable `tax` indicates which tax rate should be used to balance the budget.¹ There are four different options: the consumption tax τ^c , an income tax $\tau^w = \tau^r$, the labour-earnings tax τ^w , and the capital-income tax τ^r . In each case, we subtract the revenue from all other taxes from public expenditure to obtain the government's financing gap. We then divide this value by the respective tax base. The result is the budget-balancing tax rate. In addition to this, we have to compute the budget-balancing contribution rate of the pension system. To this end, we first derive the pension level `pen` as a fraction `kappa` of average labour earnings `INC`. We then aggregate total pension expenditure in the variable `PP` and divide it by total labour income to obtain the balancing contribution rate.

Program 11.1.c shows the main executable part of our program. The program starts with calling the subroutine `initialize`, which assigns values to all variables of the program that have not already been set as parameters in the module `globals`. We then open a file in which the subroutine `output` can write the model outcome. The logical variable `reform_on` is set to a value of `.false.` as we want to determine the initial equilibrium of our economy. This is done in the subroutine `get_SteadyState`. Having calculated the initial equilibrium of our economy, we can open a new file and calculate a

Program 11.1.c Organization of the main program

```
program SOLG_LR
[.....]
! initialize variables
call initialize()

open(21, file='output_initial.out')

! calculate initial equilibrium
reform_on = .false.
call get_SteadyState()

close(21)

open(21, file='output_final.out')

! set reform variables
reform_on = .true.
kappa = 0d0

! calculate final equilibrium
call get_SteadyState()

close(21)
[.....]
end program
```

¹ All other tax rates are set exogenously and remain constant throughout the iteration process.

new counterfactual long-run steady state which results from changing some parameter of public policy. In the case presented here we set the replacement rate of the public pension system to zero. Note that we have to switch the value of `reform_on` to `.true.` in order to keep government expenditure G and public debt B constant in the reform equilibrium.

11.1.3 MODEL PARAMETRIZATION AND CALIBRATION

One thing we haven't talked about so far—and on which we also were very brief in the last chapter—is how to parameterize this model, that is, how to map the model into the data. There are two types of model parameters. Those that can be directly observed in the data, such as life expectancy, capital shares, etc., and those parameters that influence the model outcome but cannot be observed directly. To parameterize the latter the literature typically uses a calibration procedure. The idea behind that procedure is to relate each model parameter to a specific target output value of the model, the real-world counterpart of which can be observed in the data. Then the parameter value is adjusted until the respective target output value is sufficiently close to the value taken from the data. There are a couple of things to note when applying such a calibration strategy. First, it is not only one parameter that influences one outcome variable of the model. Instead when we change one parameter usually all variables of the model will change. Therefore the process of calibration involves first getting a feeling for how different parameters shape model outcomes and which parameter we can actually identify from which model outcome. This means one has to run the model many times. Second, the calibration process is not an analytically defined procedure, but a rather vague process in which the researcher decides which fit of the model is good enough to suit her needs. Third, especially for elasticities, it is often useful to get estimates from the literature and directly feed them into the model.

Exogenous parameter values In order to limit the running time of our model, we let one model period cover five real years. The model could easily be expanded so that one period covers one real year by just increasing the maximum age JJ .² If we assume that households start their economic life at age 20 ($j = 1$) and face a life expectancy (at birth) of 80 years, the model's life cycle has to cover $JJ=12$ periods. Assuming a statutory retirement age of 65 years yields a model retirement age of $JR=10$, so that households spend the last three periods in retirement. The last demographic parameter is the population growth rate. We assume an annual growth rate of 1 per cent, which needs to be converted into a periodic growth rate $n_p = 1.01^5 - 1 \approx 0.05$.³

² However, more model periods lead to a longer running time and more computer memory consumption as the size of arrays to store individual decision rules increases.

³ Note that we report annualized values wherever possible as they are much easier to interpret. The adjustment to model periods is done in the code.

The capital share in production is set at $\alpha = 0.36$, which is a quite common value and leads the labour-income share to be equal to 0.64. On the household side we choose an intertemporal elasticity of substitution of $\gamma = 0.5$, which implies an individual relative risk-aversion of 2. Note that a risk-aversion of 2 is quite common in the macroeconomic literature, but fairly low compared to values used in finance. The age-productivity profile e_j is taken from the literature. We normalize it to a value of 1 at the first working age $j = 1$. From there on labour productivity roughly doubles until age 45–55. Afterwards it slightly falls again until the date of retirement. Finally, we fix total government expenditure as a fraction of GDP at $gy=0.19$, while we specify a government debt to GDP ratio of $by=0.6$.

Calibrated parameter values The remaining parameters of the model need to be pinned down by calibration. We discuss these parameters and their targets by economic sector. On the production side we have to specify the aggregate technology level Ω as well as the capital depreciation rate δ . The former is pinned down by normalizing the wage rate for effective labour to $w = 1$. This requires setting $\Omega = 1.6$. The depreciation rate of capital mostly impacts on the size of aggregate investment. In a long-run equilibrium, aggregate investment relative to GDP is defined as

$$\frac{I}{Y} = (n_p + \delta) \cdot \frac{K}{Y} \quad \text{which implies} \quad \delta = \frac{I/Y}{K/Y} - n_p.$$

Assuming an average investment to GDP ratio of 24 per cent, an annual capital-to-output ratio of 3—meaning a five-year ratio of $\frac{3}{5} = 0.6$ —and taking into account the (5-year) population growth rate of 5.1 per cent, we arrive at a five-year depreciation rate of 0.349, or an annual value of 8.23 per cent.

We want the capital output ratio to be equal to 3 and use this value to pin down the intertemporal discount factor β in the household utility function.⁴ The higher is β , the more individuals are willing to save and the higher will be the aggregate capital stock of the economy. The consumption share parameter ν governs the individual preference for consumption goods purchased on the market in relation to leisure consumption. The larger ν is, the more consumption goods a household buys in the market and the less leisure the household consumes. Therefore ν has a strong influence on the number of hours a household works in the market. We adjust ν to target an average share of working time in the total time endowment of around 33 per cent.⁵

Next we calibrate the wage process of the model by targeting the variance of the log of household labour income over the life cycle. Empirical studies suggest that around age 25 this variance has a value of 0.3, which then increases almost linearly to a value of 0.9 at the age of 60. The variance of log labour earnings in our model is determined by two

⁴ Note that estimates of country-specific capital stocks vary substantially depending on whether housing and other non-productive capital is included or not.

⁵ This share is derived from assuming a maximum weekly working-time endowment of 110 hours as well as 50 working weeks per year. We relate this to average annual working hours per employee of around 1,800.

components: the exogenous processes for idiosyncratic labour productivity θ and η_j , as well as the individual decision about how many hours of labour to supply in the market. We do not have a closed-form answer to how labour hours react to labour-productivity shocks and therefore shape the earnings process over the life cycle as labour hours are also influenced by the amount of wealth and the age of the individual. We do, however, have some information about the structure of the labour-productivity process and how it may influence the variance of log labour earnings. The log of labour earnings of an individual reads

$$\log(w_t h_j l_j) = \log(w_t) + \log(e_j) + \theta + \eta_j + \log(l_j).$$

The first two components are deterministic for each age group, so that their variance is equal to zero. The variance of log labour earnings at age j consequently is

$$\begin{aligned} \text{Var}[\log(w_t h_j l_j)] &= \text{Var}[\theta] + \text{Var}[\eta_j] + \text{Var}[\log(l_j)] + 2 \cdot \text{Cov}[\theta, \log(l_j)] \\ &\quad + 2 \cdot \text{Cov}[\eta_j, \log(l_j)]. \end{aligned}$$

Note that the term $2 \cdot \text{Cov}[\theta, \eta_j]$ is equal to zero since θ and η_j are independent random variables. For our calibration strategy the first two components of the variance are the most interesting, since they can be written more explicitly. First recall that the initial condition for the transitory process is $\eta_1 = 0$, which implies $\text{Var}[\eta_1] = 0$. Knowing this, we can calculate the variances of the stochastic component of log labour productivity for each potential age j as

$$\begin{aligned} \text{Var}[\theta] + \text{Var}[\eta_1] &= \text{Var}[\theta] &= \sigma_\theta^2 \\ \text{Var}[\theta] + \text{Var}[\eta_2] &= \text{Var}[\theta + \epsilon_2] &= \sigma_\theta^2 + \sigma_\epsilon^2 \\ \text{Var}[\theta] + \text{Var}[\eta_3] &= \text{Var}[\theta + \rho\epsilon_2 + \epsilon_3] &= \sigma_\theta^2 + (1 + \rho^2)\sigma_\epsilon^2 \\ \text{Var}[\theta] + \text{Var}[\eta_4] &= \text{Var}[\theta + \rho^2\epsilon_2 + \rho\epsilon_3 + \epsilon_4] &= \sigma_\theta^2 + (1 + \rho^2 + \rho^4)\sigma_\epsilon^2 \\ &\vdots \end{aligned}$$

The above calculation tells us a lot about the variance of log labour productivity over the life cycle. First of all, since the initial transitory component is normalized to zero, the variance of log labour productivity at the youngest age $j = 1$ is solely due to variations in the fixed effect σ_θ^2 . Second, a strongly increasing variance over the life cycle indicates a high autocorrelation parameter ρ . To see this, just assume that persistence was $\rho = 0$. In this case the variance of log labour earnings would simply be $\sigma_\theta^2 + \sigma_\epsilon^2$ in each period $j > 1$. The variance of log labour productivity would therefore jump between ages $j = 1$ and $j = 2$ and then stay constant over the remainder of the life cycle. With an extreme autocorrelation of $\rho = 1$, on the other hand, the variance of labour productivity at age j was $\sigma_\theta^2 + (j - 1) \cdot \sigma_\epsilon^2$, i.e. it would increase linearly with age. Since in our model

Program 11.1.d Calculating the variance of log labour earnings

```

exp_y = 0d0 ; var_y = 0d0 ; mas_y = 0d0
do ij = 1, JJ
  do ia = 0, NA
    do ip = 1, NP
      do is = 1, NS
        [.....]
        if(l(ij, ia, ip, is) > 0.01d0)then
          [.....]
          ! earnings
          temp = log(w*eff(ij)*theta(ip)*eta(is)*l(ij, ia, ip, is))
          exp_y(ij) = exp_y(ij) + temp*phi(ij, ia, ip, is)
          var_y(ij) = var_y(ij) + temp**2*phi(ij, ia, ip, is)
          mas_y(ij) = mas_y(ij) + phi(ij, ia, ip, is)
        endif
      enddo
    enddo
  enddo
exp_y = exp_y/max(mas_y, 1d-4) ; var_y = var_y/max(mas_y, 1d-4)
var_y = var_y - exp_y**2

```

the variance of log labour earnings is influenced by both labour productivity and hours worked, we choose an autocorrelation of $\rho = 0.98$, which makes the variance of log labour productivity rise quite substantially but not linearly. We will see that this value is enough to create a strongly rising profile for the variance of log labour earnings. Having chosen a value for ρ , the remaining parameters of the labour productivity process are the variances σ_θ^2 and σ_ϵ^2 . We calibrate values for these parameters to target a variance of the log of labour earnings of 0.3 at the age of 25 ($j = 2$) and of 0.9 at the age of 60 ($j = 9$).

Program 11.1.d shows how to calculate the age-specific variance of log labour earnings. We iterate over the state space at a specific age, add up the log of labour earnings as well as its square and weight it by the respective distribution of households at a certain grid point. Note that we only select households that work at least 1 per cent of their total time endowment in order to avoid that the program tries to take the log of a number that is close to or equal to zero. Since this selection causes the sum of households to not necessarily add up to a value of 1, we have to normalize the variables `exp_1` and `var_1` by the respective mass of households we used.

Finally, we have to calibrate the remaining government parameters. We have already defined values for government consumption and public debt as a fraction of GDP. What remains to be specified are the tax rates on consumption and income from capital and labour as well as the parameters of the pension system. Assuming that the revenue from the taxation of goods and services is about 4.5 per cent of GDP and aggregate consumption amounts to roughly 60 per cent of GDP, a consumption tax rate of 7.5 per cent is our favorite choice. There are not many more free parameters in the tax system, as the expenditure side is already determined. If we furthermore assume that the government levies a uniform income tax on income from both labour and capital, the income tax rate needs to be the endogenous tax rate in our model. Consequently it is

Table 11.1 Calibrated parameters in the OLG model

| Parameter | Value | Target | Value |
|---------------------|--------|--------------------------------------------|-------|
| Ω | 1.60 | w | 1.00 |
| δ | 0.0823 | $\frac{l}{Y}$ | 0.24 |
| β | 0.998 | $\frac{K}{Y}$ | 3.00 |
| v | 0.335 | average hours worked | 0.33 |
| ρ | 0.98 | linearly increasing $\text{Var}[wh_j/l_j]$ | |
| σ_θ^2 | 0.23 | $\text{Var}[wh_1/l_1]$ | 0.30 |
| σ_ϵ^2 | 0.05 | $\text{Var}[wh_9/l_9]$ | 0.90 |
| τ^C | 0.075 | $\frac{\tau_{\text{af}}^C}{Y}$ | 0.045 |
| κ | 0.50 | τ^P | 0.12 |

calculated such that the government budget constraint is satisfied. Last but not least, we have to specify a replacement rate κ for the pension system. We want to target a pension contribution rate of around 12.4 per cent, which leads us to a replacement rate of 0.50. Table 11.1 summarizes our parameter choices and their respective targets.

Parameters for computational purposes The tax system parameters include a variable `tax` that informs the subroutine `government` about which tax rate should be the endogenous one. We set this variable to a value of 2, indicating that a uniform income tax system on labour and capital income should balance the tax system's budget. Next to the model-related parameters we also have to specify the numerical parameters at this stage. These include a damping factor of `damp=0.3` for the Gauss-Seidel iteration procedure, a level of tolerance for the relative difference between demand and supply on the goods market `sig=1-d4`, and the maximum number of iterations that should be used to calculate the equilibrium values `itermax=50`. The remainder of the setup procedure happens in subroutine `initialize`, in which we discretize the state space, specify labour productivities e_j , and initialize prices and quantities.

Our parameter choice for the asset grid deserves a more detailed discussion. Obviously the lower end of the state space $a_l = 0$ is defined through the borrowing constraint of households. Specifying the growth rate of the asset grid u_a and the maximum asset level a_u remains to be done. We choose a growth rate u_a such that the difference between the first and the second asset grid point is in the order of 10^{-2} . Note that a larger density of points around the lower bound of the asset grid is desirable as the borrowing constraint might hit individuals in this range of the asset space and therefore the policy functions might have a kink. We suggest the following calibration procedure for the growth rate. Start out with a growth rate of $u_a = 0$ and simulate the model. Then successively increase the growth rate by 0.01 and simulate the model again. This will cause both the macro variables as well as the average life-cycle paths of the household to change slightly. As soon as these variables stay constant by increasing the growth rate, you have found a suitable growth rate for the asset grid.⁶ Note that distributional variables will always

⁶ Alternatively, one can try finding a good growth rate by computing Euler equation errors and trying to minimize them with the choice of u_a , see the exercise section of Chapter 9.

change slightly when the growth rate is adjusted, especially so at later ages. So don't pay too much attention to these. Note further that a similar procedure can be used to determine what is a good number of discretization points n for the asset grid space. Finally, we have to come up with a suitable maximum asset level a_u . In order to calibrate this, we identify the maximum asset gridpoint that is used at each age. The subroutine `check_grid(iamax)`, which is not shown here, iterates over the whole state space at each age j . If the share of households at a specific gridpoint (ij , ia , ip , is) is greater than 10^{-8} , the asset gridpoint is stored in the array `iamax(ij)`. Due to the specific structure of our do-loops, this procedure yields the highest asset gridpoint used at each age. We chose a_u such that only about 95 per cent of gridpoints are effectively used. This leaves some upward space in case we want to calculate some reform scenario in which households start increasing their savings significantly.

11.1.4 THE INITIAL EQUILIBRIUM

Having calibrated the model, we can now run the program and write some output. The output file in which we write the initial equilibrium variables is called `output_initial.out` and contains all the necessary information. The output file is structured into two parts. The first shows the outcomes of the model on the macroeconomic level, the second part summarizes the average life-cycle profiles of individual variables as well as the variances of their logs. Note that many of the macro variables are shown both in absolute values and as a fraction of GDP in annual terms.⁷ The life-cycle averages are expressed as a fraction of the working households' average labour earnings, so that they can easily be compared to real-life data.

Table 11.2 summarizes the macroeconomic data generated by our calibrated OLG model. Most of these data have already been discussed. We find that we can match all our targets pretty well. Note that the model generates an interest rate of 4.55 per cent per year, which implies $r > n_p$ and therefore the economy is not falling within the golden rule. Rather, it is on the dynamically efficient side of the golden rule, which will be important in understanding the welfare effects of policy reforms simulated below.

Figure 11.1 shows the life-cycle profiles of households as a fraction of average working households' labour income. We can make a couple of observations here. First of all, since the time discount factor of the household is $\frac{1}{0.998} - 1 = 0.002$ and therefore much smaller than the interest rate of 4.55 per cent, the household favours an increasing consumption path over the life cycle.⁸ Second, labour hours and therefore labour earnings are hump-shaped over the life cycle. They first increase in the initial years as labour productivity increases. However, in the same way that they demand an increasing consumption path, households also want leisure consumption to increase. Therefore hours worked succes-

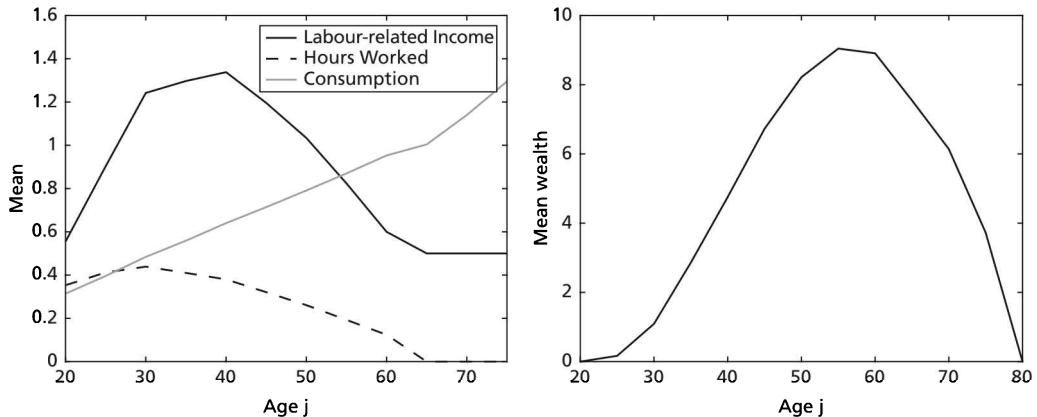
⁷ Where the stock variables as well as the interest rate are adjusted to an annual basis.

⁸ Note that in the heterogeneous agent model this was not possible since, with the interest rate being larger than the time discount factor, aggregate savings would diverge to infinity.

Table 11.2 The initial equilibrium of the OLG model

| Variable | Value | Variable | Value |
|---------------------------|--------|--------------------------------------------------|-------|
| <i>Capital market:</i> | | | |
| Private assets | 360.82 | <i>Labour market:</i> | |
| Capital | 300.82 | Average hours worked (in % of time endowment) | 33.21 |
| Public debt | 60.00 | Wage rate (absolute) | 1.00 |
| Interest rate (in % p.a.) | 4.55 | | |
| <i>Goods market:</i> | | | |
| Private consumption | 56.93 | <i>Pension system:</i> | |
| Public consumption | 19.00 | Payroll tax rate (in %) | 12.27 |
| Investment | 24.07 | Replacement rate (in %) | 50.00 |
| | | Total pension payments | 7.86 |
| <i>Tax rates:</i> | | | |
| Consumption (in %) | 7.50 | <i>Tax revenue:</i> | |
| Labour earnings (in %) | 20.87 | Consumption tax | 4.27 |
| Capital income (in %) | 20.87 | Labour-earnings tax | 13.36 |
| | | Capital-income tax | 3.75 |

In % of GDP if not indicated otherwise.

**Figure 11.1** Average life-cycle profiles

sively decrease again with age. The combination of an increasing consumption path and a hump-shaped labour-related income profile⁹ leads households to save quite a substantial amount. These savings are meant to finance consumption in the retirement period, where labour-related income is especially low. The households wealth profile therefore follows a typical hump shape over the life cycle. Note that in peak times average household wealth amounts to roughly 10 times average labour income. Beneath this life-cycle savings motive, however, there is also a precautionary savings motive in this model, which comes into play mostly in early stages of the life-cycle. Like in the standard heterogeneous agent and the life-cycle models discussed in Chapters 9 and 10, households build a buffer stock

⁹ The labour-related income profile is made up of gross labour earnings in the working phase and gross pension payments in the retirement phase.



Figure 11.2 Variance of logs over the life cycle

of assets in order to insure against transitory fluctuations in labour productivity. Yet, savings are not the only means to insure against productivity shocks. Since households can adjust their labour supply freely, they can work more hours in times with low labour productivity to generate more income. Consequently, in comparison to models with fixed labour supply, the amount of precautionary savings is limited.

That households actually do smooth consumption over different states of productivity can be seen from Figure 11.2. This figure plots the model-simulated variance of log labour earnings (of individuals with at least 0.01 hours of work) as well as the variance of consumption over the working years of the household. We find that the variance of consumption is much smaller than the variance of earnings. The reason is that households can in fact quite effectively self-insure against shock-driven fluctuations in income by using precautionary savings.

11.1.5 LONG-RUN ANALYSIS OF POLICY REFORMS

We now want to use our simulation model to conduct and evaluate reforms of government policy. As a first step, we only compute new long-run equilibria of the model that result from adjusting certain policy parameters. To this end, we keep public consumption G as well as the stock of debt B fixed (per capita) by setting the logical variable `reform_on` to a value of `.true.`. Table 11.3 shows our results.¹⁰ To facilitate comparison, the first line of the table again summarizes the public policy parameters and central macroeconomic outcomes of the initial equilibrium.

¹⁰ Note that these results are qualitatively quite similar to the ones reported in Table 6.1, in which we show the results of similar reforms in a deterministic OLG model.

Table 11.3 Long-run policy analysis

| # | b_y | κ | τ^W | τ^I | τ^C | C/Y | K/Y | L | w | E(U) |
|-----|-------|----------|----------|----------|----------|------|-------|------|------|--------|
| (0) | 0.60 | 0.5 | 0.209 | 0.209 | 0.075 | 56.9 | 300.8 | 33.2 | 1.00 | -19.62 |
| (1) | 0.53 | | 0.000 | 0.000 | 0.326 | 55.0 | 352.7 | 34.2 | 1.10 | -18.98 |
| (2) | 0.60 | | 0.262 | 0.000 | | 55.8 | 316.0 | 32.7 | 1.03 | -19.81 |
| (3) | 0.48 | 0.0 | 0.157 | 0.157 | | 55.3 | 369.2 | 37.2 | 1.12 | -18.79 |

$$\gamma = 0.5, \nu = 0.335, \beta = 0.998, \alpha = 0.36, \delta = 0.0823.$$

In simulation (1) we completely eliminate the income tax and move to a pure consumption tax system. The consumption tax rate that is needed to fully finance public expenditure is equal to 33.2 per cent. As in contrast to income taxes, a large share of consumption taxes is paid by the elderly population, households increase their working hours and savings to prepare for a higher tax burden in old age. The consumption share in GDP falls to 55 per cent, while the investment share increases to maintain a long-run capital-to-output ratio of 352.2. As the expansion of private savings outweighs the rise in labour supply, long-run wages are around 10 per cent higher than in the initial equilibrium and the interest rate in turn falls. Like in Chapter 6, households are better off with a move from income to consumption taxation in the long run. Remember that in the initial equilibrium, the economy is on the dynamically efficient side of the golden rule. Hence, an increase in capital and a corresponding decline in the interest rate lead to a gain in long-run welfare.

In simulation (2) we substitute the income tax with a pure wage-earnings tax, meaning that we eliminate capital-income taxation. As a consequence, the tax burden on labour income increases to 26.2 per cent. This depresses labour supply. The elimination of capital-income taxation, on the other hand, creates incentives for higher savings. Hence, the capital-to-output ratio increases to 316 per cent. Despite a higher capital stock, higher wages, and a decline in the interest rate, this reform causes long-run welfare to fall below its initial equilibrium value. This somehow contrasts our intuition from the deterministic OLG model. We will return to this phenomenon again at a later point.

In simulation (3), we finally maintain the tax system of the initial equilibrium, but eliminate the pay-as-you-go pension system. Since payroll taxes distort labour supply, households increase their working hours. On the other hand, they have to compensate for the loss in old-age income through additional savings. Again, the higher capital stocks weighs heavier than the rise in labour hours, so that wages increase by 12 per cent and the interest rate falls. This time our golden rule intuition again proves valid, meaning that the expansion of productive capital causes a rise in long-run welfare.

Overall, despite the fact that our model now features idiosyncratic uncertainty, the long-run effects of policy reforms are qualitatively quite similar to those derived under certainty in Chapter 6. This is not too surprising, since the line of reasoning we followed is quite similar. The question is whether the same also applies to the efficiency effects of public policy reforms, which is what we deal with next.

11.2 Transitional dynamics and welfare analysis

Chapters 6 and 7 already showed that ignoring transitional dynamics in the OLG model is potentially problematic, as a reform's impact on transitional generations can be fundamentally different from what we observe in the long run. Hence, we want to discuss how to calculate transition paths in the stochastic OLG economy, how to measure welfare effects of different generations, and how to calculate a measure of the aggregate efficiency impact of policy reforms.

11.2.1 COMPUTATION OF TRANSITIONAL DYNAMICS

Adding transitional dynamics into the OLG model is not very difficult. It basically requires putting a time index on any individual and aggregate variable. As before, we call this time index `it` and the maximum number of periods `TT`. Program 11.2 shows the resulting structure of the main program. Like in the steady-state calculations of Section 11.1 we use the subroutine `get_SteadyState` to calculate the initial long-run equilibrium with the calibrated parameters discussed above. The respective values for macroeconomic variables as well as policy and value functions are stored in the time index `it = 0`. We then change one or more parameters of the model; in the case shown here we set the replacement rate of the pension system to zero starting in period $t = 1$. The elimination of pension benefits disturbs the economy so that it is not in a long-run equilibrium anymore. Instead, per capital quantities and prices start moving and slowly converge to a new long-run equilibrium. This path of convergence is called the *transition path*, see Chapter 6 for further details. Calculating a transition path requires some initial conditions for all stock variables, in our case the capital stock K_1 and the public debt level B_1 . We let these stocks be equal to the initial equilibrium values K_0

Program 11.2 Solving the OLG model with transitional dynamics

```
program SOLG_TR
[.....]
! calculate initial equilibrium
call get_SteadyState()

! set reform parameters
kappa(1:TT) = 0.0d0

! calculate transition path without lsra
lsra_on = .false.
call get_transition

! calculate transition path with lsra
lsra_on = .true.
call get_transition
[.....]
end program
```

and B_0 . This also implies that the distribution of households over the state space $\phi_1(\cdot)$ in period 1 of the transition should be identical to the distribution of households in the initial equilibrium $\phi_0(\cdot)$.

We store the quantities, prices, policy, and value functions along the transition path in the indexes $1 : \text{TT}$, where TT denotes the new long-run equilibrium. Note that the length T of the transition path needs to be large enough so that the economy has a chance to actually converge to the new long-run equilibrium. In our model setup with a life length of $J = 12$ periods, $T = 40$ transition periods should be enough for this convergence to take place. When looking at the simulation results, however, one should always check that this is true. The subroutine `get_transition` that manages the calculation of transition paths is structured as follows:

1. It first uses the subroutine `initialize_trn` that initializes all variables with indexes $1 : \text{TT}$. This initialization process is fairly simple. The subroutine just takes the values of the respective variables in the initial equilibrium and copies them into all periods $1 : \text{TT}$. In the absence of any parameter adjustments, the model economy will therefore be in equilibrium directly after one iteration of the macroeconomic iteration algorithm.
2. The subroutine `get_transition` then kicks off the very same macroeconomic iteration procedure that we also used to find the initial long-run equilibrium. This time, however, we need to solve the household problem for many more generations. In fact, we call the subroutine `solve_household` with two input arguments. A time index `it` and an age index `ij`. The reason for this is that we not only need to solve the household problem for any generation that starts their life at some date `it` along the transition path. We also have to re-optimize the decisions for any generation living in the initial equilibrium and caught by surprise by the policy change at some date in the middle of their life. Having solved the household optimization problem, the subroutine `get_transition` determines the distribution of households in each year `it`, aggregates decisions to macroeconomics quantities, and determines budget-balancing tax rates. This is done until the relative differences between demand and supply on the goods market in any period `it` is below the tolerance level `sig`.¹¹
3. If the iteration process has converged or a maximum number of iterations is reached, the subroutine writes the results into the output files `output.out` and `summary.out`. While the former contains detailed information on macroeconomic and life-cycle variables in each period `it`, the latter write some summary output for prices and quantities along the transition path and in the new long-run equilibrium. This file will also contain the information on welfare and aggregate efficiency effects, see section 11.2.2.

¹¹ Note that for one of the goods markets along the transition, we use a less strict convergence criterion of `sig*100d0`. Typically this is the market with index $\text{TT} - 1$, which doesn't converge with such precision as we force the market with index TT to be the final long-run equilibrium.

Module 11.2m Time indexes in the OLG model

```

function year(it, ij, ijp)
    [.....]
    year = it + ijp - ij

    if(it == 0 .or. year <= 0)year = 0
    if(it == TT .or. year >= TT)year = TT

end function

```

Note that the subroutine `get_transition` is actually called up twice. The first time, it just calculates the transition path. The second time, it applies the concept of a Lump Sum Redistribution Authority to calculate the aggregate efficiency gain or loss of a reform.

The last thing that needs our attention is the management of time indexes. This is done in the function `year` shown in Module 11.2m. This function calculates the year in which a household who at time `it`, aged `ij` turns `ijp` years old. With this function one can, for example, simply calculate the next year by calling `year(it, 1, 2)` or the previous year by calling `year(it, 2, 1)`. There are two things to note here. First, the years need to be limited to the minimum and maximum periods 0 and `TT`. Second, and more importantly, if the current year `it` is a long-run equilibrium year (initial or final), then the time index does not change. This means that when we calculate the initial equilibrium, the current time index is 0 and the future time index is 0 as well. The same applies to the final long-run equilibrium.

11.2.2 GENERATIONAL WELFARE AND AGGREGATE EFFICIENCY

We use the concept of *Hicksian equivalent variation* (HEV) to measure the welfare effects of a specific policy reform, which we applied in Chapters 6 and 7. Assume a household at age j in period t had some level of wealth a and is experiencing labour-productivity shocks θ and η . Such a household's value function is an expected discounted sum of utilities from instantaneous flows of consumption and leisure that respect the individual budget constraint and the laws of motion of the transitory shocks. Consequently we can write

$$V_t(z) = E_j \left[\sum_{s=j}^J \beta^{s-j} \frac{[c_{t+s-j}(\tilde{z})^\nu \cdot (1 - l_{t+s-j}(\tilde{z}))^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right]$$

with $z = (j, a, \theta, \eta)$ and $\tilde{z} = (s, \cdot, \theta, \cdot)$. The HEV compares the household at state z at time t with a household at the same state in the initial equilibrium. It measures how much additional consumption and leisure we would have to give to the household in the initial equilibrium in order to make her equally well off as in the reform path. This means

we want to know by what factor $\Delta_t(z)$ we have to increase consumption and leisure at each remaining age $s = j, \dots, J$ and in each state of the world \tilde{z} in order to get

$$\begin{aligned} V_t(z) &\stackrel{!}{=} E_j \left[\sum_{s=j}^J \beta^{s-j} \frac{[(1 + \Delta_t(z))c_0(\tilde{z})]^\nu \cdot [(1 + \Delta_t(z))(1 - l_0(\tilde{z}))]^{1-\nu}]^{1-\frac{1}{\gamma}}}{1 - \frac{1}{\gamma}} \right] \\ &= [1 + \Delta_t(z)]^{1-\frac{1}{\gamma}} \cdot V_0(z). \end{aligned}$$

The HEV is then given by

$$\Delta_t(z) = \left[\frac{V_t(z)}{V_0(z)} \right]^{\frac{1}{1-\frac{1}{\gamma}}} - 1.$$

We can say that the policy reform we are studying is making an individual at state z at time t exactly $\Delta_t(z) \cdot 100$ per cent better off in terms of consumption and leisure compared to her initial equilibrium counterpart.

We calculate the welfare effects in subroutine `output_summary`, see Program 11.2.a. The calculation procedure follows two steps that are related to the structure of the population in the reform path. To understand this we have to remember what the different generations living in the reform year $t = 1$ are, see Figure 11.3.¹² Before

Program 11.2.a Calculation of welfare effects

```

HEV = 0d0
mas = 0d0
do ij = JJ, 2, -1
  do ia = 0, NA
    do ip = 1, NP
      do is = 1, NS
        if(ij >= JR .and. ia == 0 .and. (kappa(0) <= 1d-10 &
          .or. kappa(1) <= 1d-10)) then
          cycle
        endif
        HEV_help = ((VV(ij, ia, ip, is, 1) / &
          max(VV(ij, ia, ip, is, 0), -1d10))** (1d0/egam) - 1d0) * 100d0
        HEV(-(ij-2)) = HEV(-(ij-2)) + HEV_help * phi(ij, ia, ip, is, 1)
        mas(-(ij-2)) = mas(-(ij-2)) + phi(ij, ia, ip, is, 1)
      enddo
    enddo
  enddo
HEV(-(JJ-2):0) = HEV(-(JJ-2):0) / mas

! calculate ex ante welfare of future generations
do it = 1, TT
  HEV(it) = ((VV_coh(1, it) / VV_coh(1, 0))** (1d0/egam) - 1d0) * 100d0
enddo

```

¹² For readability we let the maximum number of periods be $J = 6$ in this case.

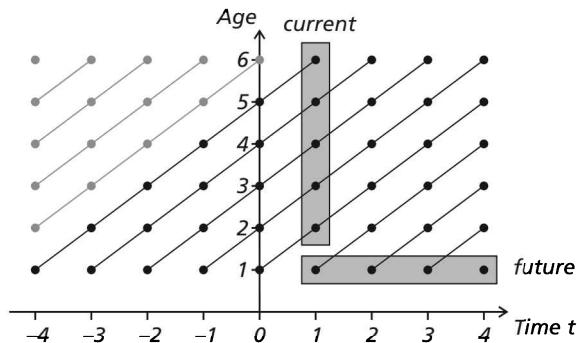


Figure 11.3 Generational structure

the reform period the economy is in its initial equilibrium and per capita quantities, prices as well as policy and value functions are time-invariant. Therefore it is enough to assume a representative generation in the program, the values of which are stored in the time index $it = 0$. For illustration purposes, however, it is valuable to also show how the generations evolve over time in the initial equilibrium. When the reform is implemented in period $t = 1$, we distinguish between *current* generations that were already living the initial equilibrium and *future* or *young* generations that first enter the economy along the transition path. The calculation of welfare effects differs slightly for these two types of generations.

The distributions of the current generations on the state space in the initial equilibrium and the reform period are identical, meaning that $\phi_0(z) = \phi_1(z)$. Consequently it is easy to compare the reform generation with utility level $V_1(z)$ for $j \geq 2$ with their initial equilibrium counterpart $V_0(z)$ and compute the Hicksian equivalent variation $\Delta_1(z)$ according to the above formula. This is done in the variable `HEV_help` in the first part of Program 11.2.a.¹³ We then aggregate the individual specific HEVs for each cohort using the distributional measure of households to get a summary statistic for the generation as a whole. Note that $2 - j$ indicates the date at which a generation with age j in the reform year $t = 1$ first entered the economy, see Figure 11.3.

For future generations we do things slightly differently. For these individuals we calculate the HEV from a so-called *ex ante perspective*, or behind the Rawlsian veil of ignorance. The idea behind this is that when the reform hits in, these individuals do not

¹³ Note that we do not calculate the welfare effects for generations that do not have any resources left and would therefore realize a consumption level of zero. This would be the case if no pension system was in place and an individual reaches retirement without assets. For technical reasons such an outcome could not be ruled out for a tiny fraction of the population. As we drop such households from the HEV calculation, we have to keep track of the mass of agents we actually sum up in the variable `mas` and normalize the variable `HEV` afterwards.

even know the value of their persistent shock θ . To include this in the welfare calculations, we compute the individuals' ex ante value function

$$EV_t = E_0 [V_t(1, 0, \theta, \eta)] = \sum_{i=1}^2 \sum_{g=1}^m V_t(1, 0, \hat{\theta}_i, \hat{\eta}_g) \cdot \phi_t(1, 0, \hat{\theta}_i, \hat{\eta}_g).$$

which means that the utility levels of the different $\hat{\theta}_i$ and $\hat{\eta}_g$ combinations are simply weighted with the respective likelihoods for such combinations to arise. Note that since the only source of heterogeneity in the first period of life are the exogenous shocks θ and η , the distribution functions $\phi_t(\cdot)$ in the above equation are identical over all periods t . The concept of HEV applies likewise. The second part of Program 11.2.a computes exactly this expression. It uses the ex ante value function of a cohort that we already calculated in the aggregation process in the variable `VV_coh`.

Measuring aggregate efficiency By only looking at the welfare impact on different generations, it is hard to provide a comprehensive economic evaluation of a government policy. The reason is that there are many different generations affected by a reform, some of which might win while others lose. In Chapter 6, we discussed the issue of such *intergenerational redistribution* and showed a way of dealing with it. To this end, we introduced the (hypothetical) concept of a Lump-Sum Redistribution Authority (LSRA), which levies lump-sum taxes or pays lump-sum transfers to individuals in order to neutralize intergenerational redistribution induced by a reform experiment. In the stochastic OLG model, we can essentially apply the same concept. However, the situation is more complicated for two reasons: First, it is impossible to derive the expected present value of an individual's remaining lifetime resources $W_{j,s}$ analytically, which complicates the calculation of lump-sum taxes and transfers. Second, welfare effects not only vary across different cohorts, but also within the cross-section of current generations. Hence, in order to isolate the aggregate efficiency consequences of a reform, LSRA payments need to undo both inter- *and* intragenerational redistribution.

In the stochastic OLG model the LSRA acts as follows:

1. For each member of current generations with state z it pays a lump-sum transfer $v_1(z)$ so as to make these individuals exactly as well off as in the initial equilibrium, i.e. the transfer has to satisfy

$$V_1(j, a + v_1(z), \theta, \eta) = V_0(z).$$

As a consequence of this exercise, the welfare effects of all current generations are zero *after compensation*, i.e. after having received LSRA transfers.

2. In a second step, the LSRA pays lump-sum transfers v_t to all future generations $t = 1, \dots, \infty$ such that these future generations all face an identical utility level,¹⁴ that is

$$EV_t(v_t) = E_0 [V_t(1, v_t, \theta, \eta)] = EV^*.$$

Since the LSRA is a self-financing institution, the discounted present value of lump-sum transfers needs to be equal to zero, meaning that

$$\underbrace{\sum_{j=2}^J \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m m_j \cdot \phi_1(z) \cdot v_1(z)}_{=SV^c \text{ (current)}} + \underbrace{\sum_{t=1}^{T-1} R_t \cdot v_t}_{\text{transitional}} + \underbrace{R_T \cdot \frac{(1+r_T)v_T}{r_T - n_p}}_{\text{new long-run}} = 0$$

where $R_t = \prod_{s=2}^t \frac{1+n_p}{1+r_s}$ is the intertemporal discount factor. The first term specifies aggregate LSRA transfers to current generations, the second term adds up transfer payments to all transitional cohorts, while the third term denotes the present value of transfer payments in the new long-run equilibrium.

The LSRA is a fullyfledged actor on the capital market, meaning that it has to issue debt or build up assets in order to shuffle around resources in a lump-sum way between different generations. We can calculate the LSRA debt level as $B_1^a = 0$,

$$(1+n_p)B_2^a = v_1 + \sum_{j=2}^J \sum_{v=0}^n \sum_{i=1}^2 \sum_{g=1}^m m_j \cdot \phi_1(z) \cdot v_1(z) \quad (11.13)$$

and

$$(1+n_p)B_{t+1}^a = (1+r_t)B_t^a + v_t. \quad (11.14)$$

Finally, with LSRA lump-sum transfer payments the capital market equilibrium reads

$$A_t = K_t + B_t + B_t^a.$$

The result of all of this is that after compensation all current generations face a utility change of zero from the reform and future generations all face an identical utility level $EV_t(v_t) = EV^*$. As the present value of all LSRA transfers needs to be zero, the LSRA

¹⁴ Note that due to the ex ante perspective each future generation receives one specific transfer, meaning that the transfer is independent of the individual's state.

has then neutralized the effects of intra- and intergenerational redistribution. The welfare effect associated with the utility level EV^* is

$$\Delta^* = \left[\frac{EV^*}{EV_0} \right]^{\frac{1}{1-\gamma}} - 1$$

and reflects the pure efficiency gain or loss associated with a reform. Since current generations are equally well off and all future generations face a common utility gain (or loss), we can call a certain reform *Pareto improving* (or *deteriorating*) after compensation.

In our program we calculate the efficiency effect Δ^* in a separate simulation after we have calculated the transition path. The logical variable `lsra_on` tells the subroutine `get_transition` to include LSRA payments into the model. The necessary transfers and the LSRA debt levels are calculated in the subroutine `LSRA`. The subroutine acts in the two steps illustrated above. We will now discuss these steps in more detail.

Transfers to current generations Calculating transfers to individuals of the current generations such that

$$V_1(j, a + v_1(z), \theta, \eta) = V_0(z)$$

is not straightforward since the value function is a complicated object and there is no analytical solution to the household optimization problem. In order to at least get an approximate solution for this transfer, we can make use of a simple first-order Taylor approximation of the value function

$$V_1(j, a + v_1(z), \theta, \eta) \approx V_1(z) + v_1(z) \cdot \frac{\partial V_1(j, a + v, \theta, \eta)}{\partial v}.$$

The key ingredient to this approximation is the partial derivative

$$\frac{\partial V_1(j, a + v, \theta, \eta)}{\partial v} =: V_{1,v}(z).$$

From the discussion in the previous chapters we already know that we can apply the envelope theorem and obtain¹⁵

$$V_{1,v}(z) = \frac{v [c_1(z)^v \cdot (1 - l_1(z))^{1-v}]^{1-\frac{1}{\gamma}}}{p_1 \cdot c_1(z)}.$$

¹⁵ Note that we excluded the factor $1 + r$ on purpose to avoid additional complexity. By doing so, we implicitly assume that the transfer already includes interest payments.

Summing all of this up, we can approximate the transfer $v_1(z)$ that is needed to bring an individual at state z in period $t = 1$ of the transition from the utility level $V_1(z)$ to the utility level $V_0(z)$ by

$$v_1(z) \approx \frac{V_0(z) - V_1(z)}{V_{1,v}(z)}.$$

Since the above transfer is only an approximation, we cannot expect the utility level of the agent to be equal to $V_0(z)$ after we have given her this transfer. However, when we recalculate the agent's new value function $V_1(\tilde{z})$ with $\tilde{z} = (j, a + v_1(z), \theta, \eta)$, her utility level will be closer to $V_0(z)$ than before. Therefore we can calculate an additional transfer level

$$\tilde{v}_1(z) = \frac{V_0(z) - V_1(\tilde{z})}{V_{1,v}(\tilde{z})}$$

and update the original transfer level $v_1(z)$ by

$$v_1(z) = v_1(z) + \tilde{v}_1(z).$$

If we repeat this iterative procedure just long enough, the utility level $V_1(\tilde{z})$ should converge towards the utility level $V_0(z)$, so that $\tilde{v}_1(z)$ converges to zero. Note that this iteration procedure is basically a Newton root-finding method, see Section 2.2.2. We integrate this iteration procedure into the macroeconomic algorithm, so that for each iteration in which the household problem is solved and prices, quantities, and tax rates are updated we compute the additional transfer levels $\tilde{v}_1(z)$ exactly once. Given the updated transfer levels, we again solve the household problem and compute prices, etc. The calculation of the additional transfer $\tilde{v}_1(z)$ is done in the first part of the subroutine `LSRA` shown in Program 11.2.b.

We iterate over the whole state space for each generation $j = 2, \dots, J$ in year 1 of the transition. The respective LSRA transfers to individuals are stored in the variable `v()`. Before we calculate the approximate transfer, we check whether the individual does have positive consumption both in the initial equilibrium and in year 1 of the transition. If this is not the case, we set the transfer level of the individual to zero and directly turn to the next one. In the event that the household does have positive consumption, we calculate the approximate transfer from the above equation. For that we make use of the function `margu` that comes from the module `globals`. We already used this function to solve the household optimization problem. Having calculated the additional transfer `v_tilde` we make sure that the transfer is not too negative, meaning that it does not lead to negative consumption. We then calculate which fraction of the current population already was has been successfully compensated.¹⁶ We therefore check whether the

¹⁶ This is done mainly for reasons of printing it to the screen and therefore for the user to observe the convergence process.

Program 11.2.b LSRA transfers to current generations

```

do ij = 2, JJ
  do ia = 0, NA
    do ip = 1, NP
      do is = 1, NS

        ! do not do anything for agent without resources
        if(ij >= JR .and. ia == 0 .and. &
           (kappa(0) <= 1d-10 .or. kappa(1) <= 1d-10))then
          v(ij, ia, ip, is, 1) = 0d0
          cycle
        endif

        ! get today's utility
        VV_1 = VV(ij, ia, ip, is, 1)

        ! get target utility
        VV_0 = VV(ij, ia, ip, is, 0)

        ! get derivative of the value function
        dVV_dv = margu(c(ij, ia, ip, is, 1),l(ij, ia, ip, is, 1), 1)

        ! calculate change in transfers
        v_tilde = (VV_0-VV_1)/dVV_dv

        ! restrict z_tilde to income maximum
        v_tilde = max(v_tilde, -((1d0+rn(1))*a(ia) + pen(ij, 1) &
          + wn(1)*eff(ij)*theta(ip)*eta(is)*0.99d0 &
          + v(ij, ia, ip, is, 1)))

        ! check whether individual is already compensated
        lsra_all = lsra_all + phi(ij, ia, ip, is, 1)*m(ij, 1)
        if(abs((VV_1-VV_0)/VV_0)*100d0 < sig) &
          lsra_comp = lsra_comp + phi(ij, ia, ip, is, 1)*m(ij, 1)

        ! calculate total transfer
        v(ij, ia, ip, is, 1) = v(ij, ia, ip, is, 1) + damp*v_tilde

        ! aggregate transfers by cohort
        v_coh(ij, 1) = v_coh(ij, 1) + &
          v(ij, ia, ip, is, 1)*phi(ij, ia, ip, is, 1)

      enddo
    enddo
  enddo
enddo

```

percentage difference between the utility level $V_1(z)$ and the target level is smaller than our general level of tolerance `sig`. If this is the case, we add the mass of the individual to the variable `lsra_comp` which will later be divided by the mass of the total population. Last but not least we add the additional transfer $\tilde{v}_1(z)$ to the total transfer $v_1(z)$. Note that we damp the additional transfer level $\tilde{v}_1(z)$ with the damping factor we also use for the macroeconomic algorithm. This introduces some stability into the LSRA algorithm.

Simultaneously, we also aggregate the transfers $v_t(z)$ to cohort aggregates using the distribution of households stored in the array `phi()`.

Transfers to future generations The computation of transfers to future generation is a little more tricky since we have to identify the utility level EV^* to which we can compensate each individual and which respects the aggregate budget constraint of the LSRA. In order to determine this level, we use the same trick of approximation as with the current generations. Specifically, we write

$$EV_t(v_t) \approx EV_t(0) + v_t \cdot EV_{t,v}$$

and

$$EV_{t,v} = E_0 \left[\frac{v [c_t(1, 0, \theta, \eta)^v \cdot (1 - l_t(1, 0, \theta, \eta))^{1-v}]^{1-\frac{1}{\gamma}}}{p_t \cdot c_t(1, 0, \theta, \eta)} \right].$$

The approximate transfer level that brings individuals to the utility level EV^* is then

$$v_t \approx \frac{EV^* - EV_t(0)}{EV_{t,v}} = \frac{\Lambda^* EV_0 - EV_t(0)}{EV_{t,v}}. \quad (11.15)$$

Now let's plug this transfer level into the intertemporal budget constraint of the LSRA. This yields

$$SV^c + \sum_{t=1}^{T-1} R_t \cdot \frac{\Lambda^* EV_0 - EV_t(0)}{EV_{t,v}} + R_T \cdot \frac{1+r_T}{r_T - n_p} \cdot \frac{\Lambda^* EV_0 - EV_T(0)}{EV_{T,v}} = 0$$

which directly leads to

$$\Lambda^* = \frac{\sum_{t=1}^{T-1} R_t \cdot \frac{EV_t(0)}{EV_{t,v}} + R_T \cdot \frac{1+r_T}{r_T - n_p} \cdot \frac{EV_T(0)}{EV_{T,v}} - SV^c}{\sum_{t=1}^{T-1} R_t \cdot \frac{EV_0}{EV_{t,v}} + R_T \cdot \frac{1+r_T}{r_T - n_p} \cdot \frac{EV_0}{EV_{T,v}}}. \quad (11.16)$$

Having determined the utility gain Λ^* , we can calculate the individual transfer levels v_t using the above approximation (11.15).

Note that we can again apply this concept in an iterative procedure as we did for the current generations. Consequently, in each iteration step we calculate the *additional* utility gain (or loss) necessary to balance the budget, given the transfers v_t from the previous iteration. The budget of the LSRA therefore reads

$$SV^c + \sum_{t=1}^{T-1} R_t \cdot \left[\frac{\Lambda^* EV_0 - EV_t(\nu_t)}{EV_{t,v}} + \nu_t \right] \\ + R_T \cdot \frac{1+r_T}{r_T - n_p} \cdot \left[\frac{\Lambda^* EV_0 - EV_T(\nu_T)}{EV_{T,v}} + \nu_T \right] = 0,$$

from which we can derive Λ^* similarly as in (11.16). Given Λ^* , we can compute the *additional* transfers $\tilde{\nu}_t$ and update the actual transfer level to

$$\nu_t = \nu_t + \tilde{\nu}_t.$$

Note that the difference $\Lambda^* EV_0 - EV_t(\nu_t)$ becomes smaller in each iteration step. Hence, the additional transfer $\tilde{\nu}_t$ converges to zero during the iteration process. In equilibrium, the intertemporal budget of the LSRA then reads

$$SV^c + \sum_{t=1}^{T-1} R_t \cdot \nu_t + R_T \cdot \frac{1+r_T}{r_T - n_p} \cdot \nu_T = 0. \quad (11.17)$$

Program 11.2.c shows the second part of the subroutine `LSRA`, in which we calculate the utility gain Λ^* . The utility gain is stored in the variable `Lstar`. There are two important things to note here. First, instead of using the utility levels EV_0 and EV_t directly, we damp them with the damping factor of the macroeconomic iteration algorithm. By doing so, in each iteration step of the LSRA we only try to close a fraction `damp` of the difference $\Lambda^* EV_0 - EV_t$. Since `damp` is smaller than 1 this means that the transfer payments to individuals will be much smaller in the initial iteration steps as if we tried to close the complete gap at once. It turns out that this produces a much more stable convergence on the macro level. Especially in the first couple of iterations the LSRA debt levels tend to vary a lot along the transition due to the inaccuracy of the utility gain `Lstar`. These variations may be so large that the capital stock of the economy actually turns negative which leads to non-convergence of the macroeconomic iteration algorithm. By applying the damping factor, the variation in debt levels is much smaller and the macro algorithm almost always converges. The second thing to note is that in order to derive the additional transfers $\tilde{\nu}_t$ we have to determine the present value of current transfers in the initial period $t = 1$

$$PV_1 = \sum_{t=1}^{\infty} R_t \nu_t = \sum_{t=1}^{T-1} R_t \cdot \nu_t + R_T \cdot \frac{1+r_T}{r_T - n_p} \cdot \nu_T$$

where we have already taken into account that in the long-run equilibrium (i.e. after period T) all variables are constant per capita. We therefore can compute the

Program 11.2.c Calculating future generations' utility gain

```

! initialize present value variables
PV_t = 0d0
PV_0 = 0d0
PV_trans = 0d0

! calculate present value of utility changes (in monetary values)
do it = TT, 1, -1

    ! get today's ex ante utility
    EVV_t = damp*VV_coh(1, it)

    ! get damped target utility
    EVV_0 = damp*VV_coh(1, 0)

    ! get derivative of expected utility function
    dEVV_dv = 0d0
    do ip = 1, NP
        do is = 1, NS
            dEVV_dv = dEVV_dv + margu(c(1, 0, ip, is, it), &
                l(1, 0, ip, is, it), it)*phi(1, 0, ip, is, it)
        enddo
    enddo

    ! calculate present values
    if(it == TT)then
        PV_t      = EVV_t/dEVV_dv      *(1d0+r(it))/(r(it)-n_p)
        PV_0      = EVV_0/dEVV_dv      *(1d0+r(it))/(r(it)-n_p)
        PV_trans = v(1, 0, 1, 1, it)*(1d0+r(it))/(r(it)-n_p)
    else
        PV_t      = PV_t      *(1d0+n_p)/(1d0+r(it+1)) + EVV_t/dEVV_dv
        PV_0      = PV_0      *(1d0+n_p)/(1d0+r(it+1)) + EVV_0/dEVV_dv
        PV_trans = PV_trans*(1d0+n_p)/(1d0+r(it+1)) + v(1, 0, 1, 1, it)
    endif
    enddo

    ! calculate the constant utility gain/loss for future generations
    Lstar = (PV_t-PV_trans-SV(1))/PV_0

```

infinite sum backwards starting in period T where we define $PV_T = \frac{1+r_T}{r_T-n_p} \cdot v_T$ as the present value of transfers in the steady state (discounted to the initial year T of the steady state!). We then compute the present value of transfers recursively for periods $t = T - 1, \dots, 1$ as

$$PV_t = PV_{t+1} \cdot \frac{1+n_p}{1+r_{t+1}} + v_t,$$

which finally yields PV_1 . Of course, we can apply the same procedure for the infinite sums of the utility levels divided by the derivative of the value function. Having calculated the respective present values, we finally determine the utility gain Λ^* according to the (adjusted) formula in (11.16).

Knowing the utility gain we can finally calculate the payments each generation receives from or has to pay to the LSRA and determine the LSRA debt levels in each period of the

Program 11.2.d Determining future transfers and LSRA debt levels

```

! calculate compensation payments for future cohorts
do it = TT, 1, -1

    ! get today's ex ante utility
    EVV_t = damp*VV_coh(1, it)

    ! get target utility
    EVV_0 = damp*VV_coh(1, 0)*Lstar

    ! get derivative of expected utility function
    dEVV_dv = 0d0
    do ip = 1, NP
        do is = 1, NS
            dEVV_dv = dEVV_dv + margu(c(1, 0, ip, is, it), &
                l(1, 0, ip, is, it), it)*phi(1, 0, ip, is, it)
        enddo
    enddo

    ! compute change in transfers (restricted)
    v_tilde = (EVV_0-EVV_t)/dEVV_dv

    ! calculate cohort transfer level
    v(1, 0, :, :, it) = v(1, 0, :, :, it) + v_tilde

    ! aggregate transfers
    v_coh(1, it) = v(1, 0, 1, 1, it)
    SV(it) = SV(it) + v_coh(1, it)*m(1, it)

enddo

! determine sequence of LSRA debt/savings
BA(2) = SV(1)/(1d0+n_p)
do it = 3, TT
    BA(it) = ((1d0+r(it-1))*BA(it-1) + SV(it-1))/(1d0+n_p)
enddo

```

transition and in the new long-run equilibrium, see Program 11.2.d. For this purpose we again use the damped utility levels as in the calculations of the utility gain L_{star} . Note that since all households of one future cohort receive the same transfer level, the $j = 1$ cohort aggregate transfers level for each generation is equal to the individual transfer v_t . From the cohort aggregate we can directly determine the aggregate transfer level SV_t by weighting the cohort aggregate with the relative population share and adding it to $sv(it)$.¹⁷ Finally, the LSRA debt levels are determined according to equations (11.13) and (11.14) which completes the computational algorithm of the LSRA. Given that $EV^* = \Lambda^* \cdot EV_0$, we can derive the aggregate efficiency effect from

$$\Delta^* = [\Lambda^*]^{\frac{1}{1-\gamma}} - 1.$$

¹⁷ We cannot write $SV(it) = v_coh(1, it) * m(1, it)$ since $SV(1)$ already contains the transfers that are paid to the current generations.

11.3 Comprehensive analysis of policy reforms

We now want to use this model to analyse several policy experiments. Our policy experiments are related to the pension system as well as the taxation of income from labour and capital. In each policy experiment we study its effects on the macroeconomy, household behaviour, the risk properties of individual consumption and leisure as well as on the welfare of different generations and aggregate efficiency. Then we follow a popular route in Macro Public Finance and determine the optimal policy that maximizes aggregate efficiency. The word ‘optimality’ has to be used with caution since it is restricted only to the respective system we are looking at and constrained by how we parameterize the system. A more general optimality analysis may want to include all policy instruments of the government and eliminates all (or at least most) parameter restrictions. However, such an exercise is probably not feasible in a rich model such as the present one.

11.3.1 THE OPTIMAL SIZE OF THE PENSION SYSTEM

In Chapter 7 we have seen that once we account for variable labour supply, introducing an unfunded pension system into the deterministic OLG model leads to major losses in aggregate efficiency. The reason for this is that the contributions to the pension system are perceived as a pure tax by the individuals. Consequently, the pension system distorts the labour-supply decision of households and decreases aggregate efficiency. In the present model, we already have a pension system with a replacement rate of $\kappa = 0.5$ in place. Consequently, by the same argument, shutting down the pension system by reducing the replacement rate to $\kappa = 0$ should increase aggregate efficiency. However, as we will show in the following, this is not necessarily the case in the stochastic OLG model.

Macroeconomic implications Table 11.4 shows the macroeconomic effects of decreasing the replacement rate of the pension system to zero. If not otherwise indicated the numbers shown are expressed as percentage deviations from initial equilibrium values.¹⁸ There are two immediate consequences when setting $\kappa_t = 0$. First of all, the pension payments individuals received in the initial equilibrium immediately drop down to zero. As a consequence, individuals have to privately prepare for retirement by building up adequate private wealth. As a result assets as well as the capital stock increase significantly throughout the transition. The other immediate consequence is that the pension contribution rate directly falls by 12.27 percentage points to zero in period 1 of the transition. While the pension contribution rate distorted labour supply in the initial equilibrium, this distortion is gone from the first period of the transition onwards which

¹⁸ Note that this is the same reform as simulation (3) in Table 11.3. However, we now report the results differently.

Table 11.4 Shutting down the pension system

| t | τ_p^* | τ_w^* | C | A | K | L | r^* | w |
|----------|------------|------------|-------|-------|-------|-------|-------|-------|
| 1 | -12.27 | -0.79 | -5.23 | 0.00 | 0.00 | 9.67 | 0.60 | -3.27 |
| 2 | -12.27 | -2.16 | 1.17 | 11.95 | 14.34 | 9.85 | -0.25 | 1.45 |
| 3 | -12.27 | -3.19 | 7.08 | 21.61 | 25.92 | 10.55 | -0.81 | 4.80 |
| 4 | -12.27 | -3.97 | 12.87 | 28.97 | 34.75 | 11.29 | -1.18 | 7.13 |
| 5 | -12.27 | -4.38 | 15.87 | 33.63 | 40.33 | 11.53 | -1.41 | 8.62 |
| : | : | : | : | : | : | : | : | : |
| ∞ | -12.27 | -5.15 | 21.53 | 44.70 | 53.62 | 11.54 | -1.92 | 12.21 |

*Change in percentage points.

causes aggregate labour supply to increase quite substantially by 9.67 per cent. Along the transition path labour supply rises even more as the government can lower tax rates on income from labour and capital. In fact the income tax rate declines by more than 5 percentage points which leads to an additional 2 per cent increase in overall labour supply. Prices along the transition and in the new long-run equilibrium perfectly reflect the movement in aggregate capital and labour. With aggregate labour supply increasing substantially and the capital stock being fixed in the first period of the transition, the wage rate initially drops and the interest rate increases. As the capital stock increases over time, the capital-to-labour ratio in the economy starts to rise and the interest rate again falls. In the long-run, the per annum interest rate is almost 2 percentage points lower than in the initial equilibrium. This effect is often referred to as the *crowding-out* effect of the pension system. By providing individuals with income in old age which is directly transferred to them from the young generations, a pay-as-you-go pension system reduces the need for private savings and therefore crowds out part of the capital stock, which in turn causes the interest rate to rise. Finally, with a steadily increasing net labour income, consumption increases by more than 20 per cent in the long run. Note however, that this long-run increase comes with a quite significant short-run decline in private consumption of about 5 per cent. Of course, the increase in the capital stock needs to be financed by additional investment in the short run, which is achieved at the expense of lower consumption.

Life-cycle effects Figure 11.4 shows the implications of the reform experiment for the life-cycle profiles of wealth and labour hours.¹⁹ The increase in retirement savings after the shutdown of the pension system can be seen by directly comparing the average wealth profile of households over the life cycle in the initial and the new long-run equilibrium in the left panel of Figure 11.4. The change in wealth between initial and final equilibrium peaks exactly at the date of retirement $j_r = 10$. Afterwards individuals quickly draw down their remaining wealth until they die with certainty after period $j = 12$. The right panel

¹⁹ As before, we express all life-cycle variables that are measured in monetary units as a fraction of average labour earnings of the working-age generations in the initial equilibrium.

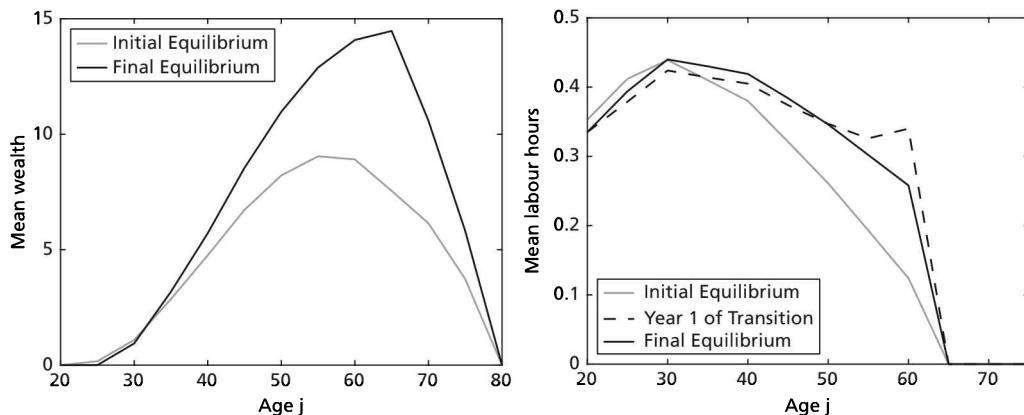


Figure 11.4 Life-cycle profiles before and after reform

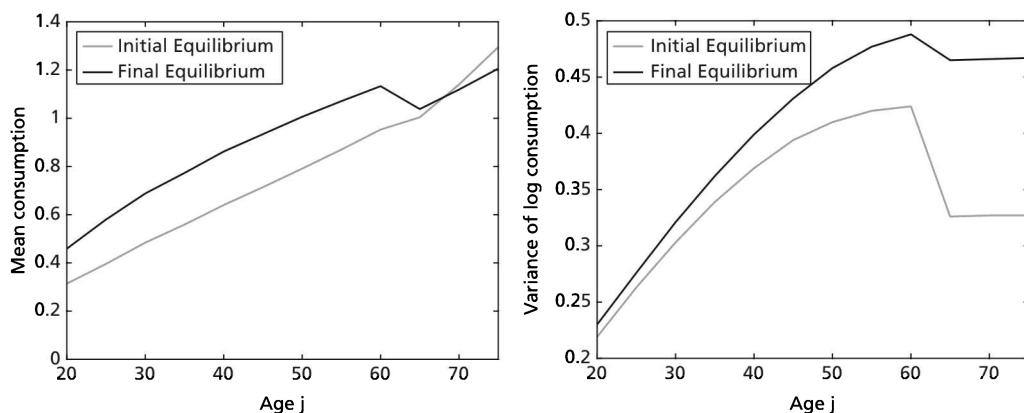


Figure 11.5 Life-cycle profiles of consumption

of Figure 11.4 reports the changes in hours over the life cycle. Labour hours increase at almost all ages both in the short and the long run. However the reaction in labour hours is strongest for older workers in the first year of the transition. This is because generations shortly before retirement in the reform year suddenly lose all their pension claims. As this change is completely unanticipated and these cohorts only have a few working periods left, they have to massively increase their labour supply in order to compensate for the loss in income.

Figure 11.5 shows the life-cycle profiles for both mean consumption and the variance of its logs. The elimination of the pension system leads to a significant long-run increase in capital and labour hours so that consumption increases quite substantially at almost all ages. Interestingly in the new long-run equilibrium the consumption profile exhibits a remarkable fall when individuals enter retirement which wasn't the case in the initial equilibrium. The reason for this fall can be found in the labour-supply profile. In the initial equilibrium labour supply gradually declined as individuals approached

retirement, i.e. they smoothly transit into retirement. In the final equilibrium households work significantly more until the date of retirement and labour supply suddenly drops to zero. This in turn means that leisure suddenly increases as individuals are forced to retire. Since leisure and consumption are substitutes owing to the choice of utility function, in reaction to this households will decrease consumption. The right-hand side of Figure 11.5 reports the variance of log consumption before and after the reform. The reform considered increases the variance of consumption quite substantially, especially during retirement years. Of course, the pension system in the initial equilibrium was very progressive since it levied contributions proportional to labour earnings but payed out benefits in a lump-sum fashion. This redistribution effect, which is now eliminated, was especially strong during the retirement years. As a consequence we observe an increase in consumption variance, especially after retirement. Throughout the working years, the increase in consumption inequality is induced by the elimination of the pension contribution rate.

Welfare effects Having understood the macroeconomic and life-cycle implications of the reform, we can turn to its welfare effects. The resulting HEV_t 's along the transition path and in the new long-run equilibrium are depicted in the left panel of Figure 11.6. Recall that for current generations we show the average equivalent variation per cohort while for future generations we report the ex ante equivalent variation. Obviously the generations who are retired at the date when the reform hits, meaning those with entry years $-10, -9$ and -8 , necessarily lose in terms of welfare from this reform. While in the initial equilibrium these cohorts still received some significant pension payments, these payments are taken away from them in the first year of the transition. For the working-age generations in the period of the reform, i.e. those with entry years -7 to 0 , the loss in pension entitlements also reduces their welfare. However, the younger the cohort in the reform year, the easier it is to make up for the loss in pension benefits by higher net

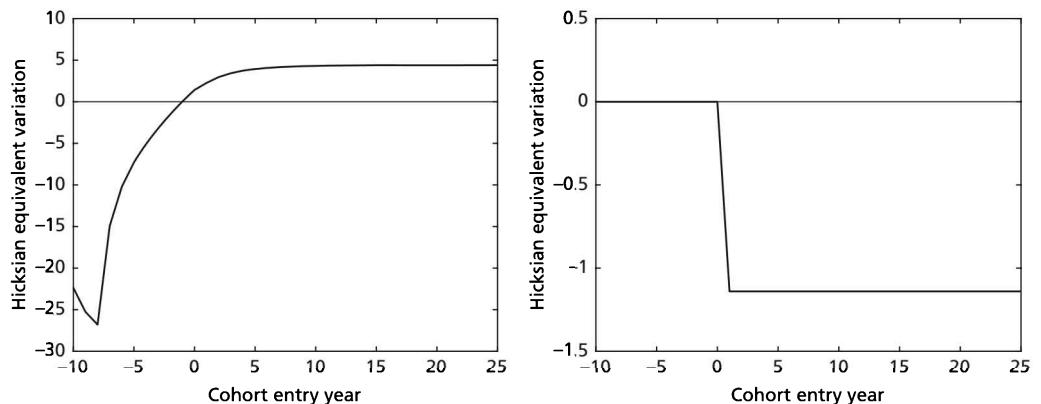


Figure 11.6 Welfare effects of shutting down the pension system

income, more working and higher private savings. Hence, the welfare loss for these young generations is much less pronounced and the welfare effect ultimately becomes positive. Owing to the elimination of pension contributions, labour supply distortions for these cohorts have substantially declined. Last but not least, future generations benefit from this reform as the capital stock rises, which causes the wage rate to grow substantially during the transition. Note that in the initial equilibrium the economy is on the dynamically efficient side of the golden rule $r_0 > n_p$, i.e. the capital stock is too low and the interest rate is too high. The reform of the pension system actually increases the capital stock and lowers the interest rate so that the economy moves closer towards the golden rule. This must be welfare-improving for future generations.

Aggregate efficiency The welfare effects shown in the left panel of Figure 11.6 suggest that there is a lot of intergenerational redistribution going on when we shut down the pension system. Hence, we are in a classical situation in which some generations benefit and some lose from a reform. In order to judge whether the elimination of the pay-as-you-go system is favourable or not overall, we simulate the reform again, this time employing the transfers from the LSRA. The respective welfare effects are represented in the right panel of Figure 11.6. As already discussed above, the LSRA pays lump-sum transfers to (or levies taxes from) all households from current cohorts so as to make them equally well off as in the initial equilibrium. Their compensated welfare change is therefore equal to zero. It then redistributes the remaining resources to make all future generations equally well off. We see that in our present reform scenario this causes all future generations to actually suffer from a significant welfare loss of -1.16 per cent of resources. This means that after compensations all future generations will realize 1.16 per cent less consumption and leisure at each age in any state of labour productivity. Hence, eliminating the pension system is probably not a valuable reform option.

But why is there such an aggregate efficiency loss from this reform although the economy is obviously expanding? In order to answer this question we have to distinguish two competing efficiency effects that come along with the elimination of the pension system. On the one hand, the lower contribution rate alleviates labour-supply distortions and therefore improves aggregate efficiency. On the other hand, the pension system was strongly progressive since it redistributed income from those who were lucky in the labour market to those who were unlucky. Expressing this in economic terms means that the pension system (i) redistributes between ex ante homogeneous but ex post heterogeneous types, that is it moves income from individuals with a high-productivity fixed effect to those with a low-productivity fixed effect and (ii) it provides insurance against individual fluctuations in labour productivity over the life cycle. Note that the former effect is a redistributive effect between different individuals, while the latter effect insures one individual as it moves through time. This redistribution provides an insurance for those who are unlucky in the labour market and the elimination of this insurance provision reduces economic efficiency. Therefore, for our specified parametrization, the negative efficiency effect caused by the loss in redistribution and insurance clearly

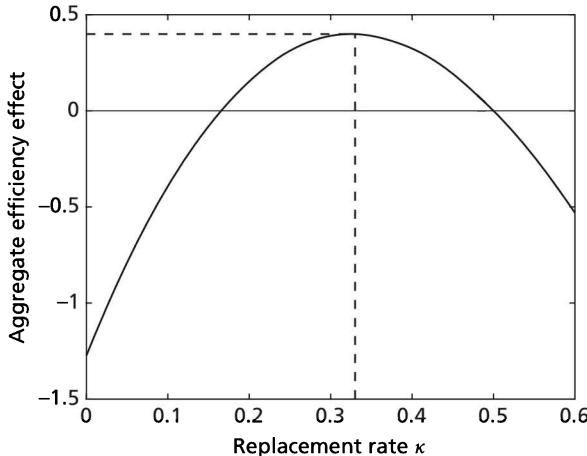


Figure 11.7 The optimal size of the pension system

dominates the positive effect through decreased labour-supply distortions, so that overall aggregate efficiency declines by 1.16 per cent of initial equilibrium resources.

Optimal size The fact that the elimination of the pension system reduces economic efficiency does not tell us that the status quo pension system is the most desirable in terms of economic efficiency. We know that there is a trade-off between labour-supply distortions and insurance benefits. However, it is not clear at which replacement rate this trade-off is actually at its optimum. To study the optimal size of the pension system, we rerun the above model with different values for κ along the reform path. Figure 11.7 illustrates the resulting aggregate efficiency effects. In this figure we first observe the negative efficiency effect of -1.16 for a replacement rate of $\kappa = 0$. As we increase the contribution rate, starting from this point, we see that aggregate efficiency improves again, the reason being that the positive insurance effect weighs more than higher labour-supply distortions. The aggregate efficiency effect then peaks at a value of $\kappa = 0.33$ with an aggregate efficiency effect of roughly 0.40 . For higher replacement rates—which also means higher contribution rates—the distortions of labour supply start to weigh more heavily than the additional insurance benefits such that our economic efficiency measure declines again. Note that the efficiency effect has to be exactly zero at the status quo level of the pension system $\kappa = 0.5$. Summing up we find the optimal replacement rate level for our pension system to be smaller than the status quo level, namely at about 33 per cent of average labour earnings, but significantly larger than zero.

11.3.2 THE OPTIMAL PROGRESSIVITY OF THE LABOUR-INCOME TAX

The same line of reasoning can be applied to study the optimal progressivity of the labour-income tax code. There are various definitions of a progressive tax schedule. The most straightforward one is that a tax code is progressive if the average tax rate of an

individual strictly increases with her labour earnings. Suppose labour earnings were $y = w_t e_j \exp(\theta + \eta_j) l_j$ and the tax schedule is $T_t(y)$. Then a progressive tax system is defined by

$$\frac{\partial(T_t(y)/y)}{\partial y} > 0.$$

Up to now we have applied a proportional tax schedule, i.e. $T_t(y) = \tau_t^w \cdot y$. In order to make this tax system progressive we simply add a lump-sum transfer Tr_t received by each working-age household, so that²⁰

$$T_t(y) = \tau_t^w \cdot y - Tr_t \quad \Rightarrow \quad \frac{T_t(y)}{y} = \tau_t^w - \frac{Tr_t}{y} \quad \text{and} \quad \frac{\partial(T_t(y)/y)}{\partial y} = \frac{Tr_t}{y^2} > 0.$$

In the following we show how to adapt the model in order to account for the progressive tax schedule defined above. There are mainly two passages in the program code where the transfer Tr_t comes into place. The first-order condition of the household problem needs to take into account that working-age households receive a lump-sum transfer of size $Tr(it)$. This is done in the function `foc` by adding `Tr(it_com)` to available resources of working cohorts. We do not show the relevant program code here. The more important part is the subroutine `government` that calculates budget-balancing tax rates. The newly adjusted subroutine that includes the lump-sum transfer payment is shown in Program 11.3. Up to the part in which the tax rates are computed, this subroutine doesn't change at all. However, when it comes to the calculation of the endogenous tax

Program 11.3 The government routine with lump-sum transfers

```
subroutine government(it)
[.....]
! calculate government expenditure
expend = GG(it) + (1d0+r(it))*BB(it) - (1d0+n_p)*BB(itp)

if(tax(it) == 1)then
    tauc(it) = (expend + Tr(it)*workpop(it) &
                - (tauw(it)*w(it)*LL(it) + taur(it)*r(it)*AA(it)))/CC(it)
    p(it) = 1d0 + tauc(it)
elseif(tax(it) == 2)then
[.....]
else
    Tr(it) = -(expend - (tauc(it)*CC(it) + tauw(it)*w(it)*LL(it) &
                           + taur(it)*r(it)*AA(it)))/workpop(it)
endif
[.....]
taxrev(4, it) = -Tr(it)*workpop(it)
taxrev(5, it) = sum(taxrev(1:4, it))
[.....]
end subroutine
```

²⁰ Note that throughout this section we make the assumption that retired individuals do not have to pay taxes on their pension income, but also do not receive any lump-sum benefits from the tax system.

rate, we need to account for the total amount of lump-sum transfer payments. With N^L denoting the total size of all working-age cohorts, this amount is $Tr_t \cdot N^L$. When the endogenous component of the government budget is not the lump-sum transfer itself, we simply add this amount to total government expenditure. In addition, we introduce a fifth balancing rule for the government that makes the transfer Tr_t endogenous. To calculate the budget-balancing transfer payment, we need to subtract all revenue the government attains from all tax sources from total government expenditure and divide it by the size of the population that receives the transfer, meaning all working-age households. We store the size of the working population in the variable `workpop(it)`. Note that we need a negative sign, because the lump-sum transfer is expenditure not revenue. Last but not least, we can calculate the revenue and expenditure for the government that it generates through taxation including the lump-sum transfer.

Optimal progressivity Having implemented a rudimentary progressive tax schedule in the stochastic OLG model, we can now analyse the optimal progressivity of the labour-income tax. We therefore proceed as follows. We first simulate the initial equilibrium of the economy the same as we did before. We then fix the original marginal tax rate on capital income and chose a marginal tax rate on labour earnings τ_t^w for all $t = 1, 2, \dots, \infty$. By setting the variable `tax(1:TT)` to a value of 5, the program will use the lump-sum transfer Tr_t to balance the budget of the government in each period t . For any potential marginal tax rate on labour earnings we can then compute the resulting LSRA efficiency gain Δ^* . Figure 11.8 plots these efficiency gains as a function of the marginal tax rate on labour earnings. In terms of economic efficiency we find that the optimal tax rate on labour earnings is equal to 0.15. The associated lump-sum transfer to each working-age individual is actually a lump-sum tax and amounts to 4.47 per cent of average labour earnings of the working-age population in the long run. Consequently in the optimal tax system the average tax rate falls with rising income, so that the tax system

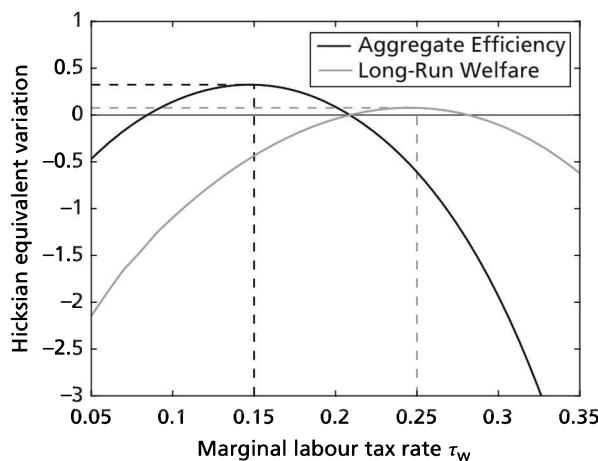


Figure 11.8 The progressivity of the labour-income tax schedule

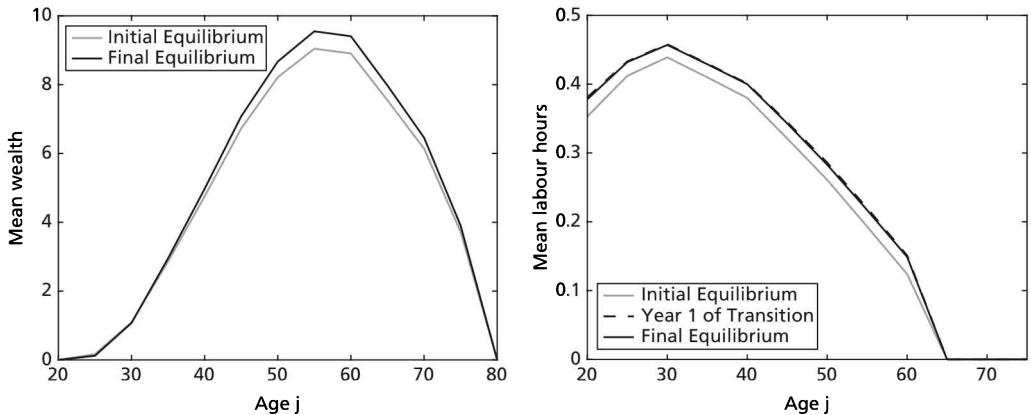
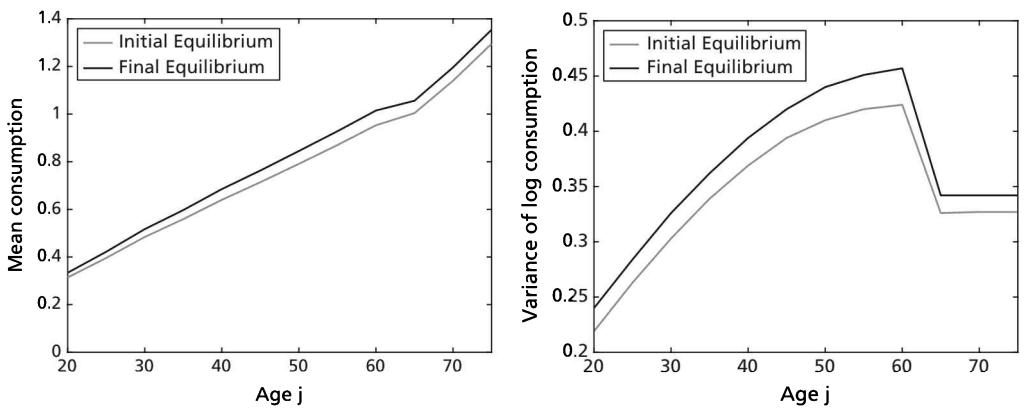
Table 11.5 Implementing a regressive labour-income tax ($\tau^w = 0.15$)

| t | $\bar{\tau}_t^b$ | C | A | K | L | r^a | w |
|----------|------------------|------|------|------|------|-------|-------|
| 1 | -5.06 | 3.39 | 0.00 | 0.00 | 5.15 | 0.33 | -1.79 |
| 2 | -4.87 | 4.23 | 1.76 | 2.12 | 4.67 | 0.16 | -0.89 |
| 3 | -4.73 | 4.86 | 2.85 | 3.42 | 4.56 | 0.07 | -0.39 |
| 4 | -4.64 | 5.32 | 3.59 | 4.30 | 4.49 | 0.01 | -0.06 |
| 5 | -4.58 | 5.60 | 4.06 | 4.87 | 4.44 | -0.03 | 0.15 |
| : | : | : | : | : | : | : | : |
| ∞ | -4.46 | 6.16 | 4.95 | 5.94 | 4.38 | -0.10 | 0.54 |

^a Change in percentage points, ^b As fraction of average income in $t = 0$

is actually regressive and not progressive. Given the result from the previous subsection that a reduced size of the progressive pension system is optimal, the present result is not too surprising.

Macroeconomic implications But what drives the optimality of a regressive tax on labour earnings in our model. To understand this in detail, we again look at the macroeconomic and life-cycle implications of decreasing the marginal tax rate on labour earnings from around 21 per cent in the initial equilibrium to 15 per cent. Table 11.5 reveals that a regressive tax comes along with a significant expansion of economic activity. A lower marginal tax rate on the labour earnings alleviates distortions on labour supply of households. Consequently aggregate labour supply increases by about 5 per cent in the period of the reform. Afterwards it slightly falls again, so that the long-run equilibrium labour-supply change amounts to 4.38 per cent. This fall in labour supply during the transition is due to the reduction of the lump-sum tax over time caused by higher revenues from capital-income and consumption taxes. To deal with the higher burden from lump-sum taxation in the short term, individuals will have to work a little more. The increase in labour supply obviously augments the available resources of the household. As these additional resources are only generated throughout the working phase and individuals want to smooth consumption over time, private savings and therefore also the capital stock have to increase. The rise in labour supply directly after the reform causes capital to become a scarcer resource. As a result the wage rate for effective labour falls and the interest rate rises directly after the reform by 0.33 percentage points per year. As the capital stock gradually increases throughout the transition, the relative scarcity of capital is ultimately reversed, so that the interest rate is actually lower in the long run than in the initial equilibrium. The overall effect on long-run factor prices is, however, modest. With a growing economy, consumption, investment, and aggregate output also grow over time. Summing up, a decline in the marginal tax rate on labour earnings paired with a lump-sum tax component propagates economic growth and leads to a higher level of long-run consumption and labour supply.

**Figure 11.9** Life-cycle profiles before and after reform**Figure 11.10** Life-cycle profiles of consumption

Life-cycle effects Next let's take a look at the impact of the reform on the mean life-cycle profiles as well as the variance of consumption. Figure 11.9 shows the life-cycle profiles of wealth and labour hours over the life cycle. Labour hours increase pretty evenly over the whole working life. This is not very surprising as the marginal tax rate on labour supply is lowered by the same amount irrespective of a household's age. Private wealth mostly rises throughout the middle of working life. This is the time where labour productivity is the highest and therefore households generate most of their labour earnings. This is also the time where the savings motive for retirement is the strongest. In the initial years of working life the household mostly saves for precautionary reasons, that is to self-insure against labour-productivity fluctuations. These savings are barely influenced by changes in marginal tax rates. Having built up a certain buffer stock, individuals start saving in order to smooth consumption over the life cycle. This smoothing motive causes consumption to increase quite evenly in each period of working life, see left panel of Figure 11.10. Importantly the increase in average consumption is accompanied by an

increase in its variance. A rising variance of consumption from lower marginal tax rates on labour earnings is exactly the reason why the government should not rely fully on lump-sum taxation, but maintain an earnings-related component in the tax schedule. As with the optimal pension system, our optimal labour-earnings tax trades off the negative effects of labour-supply distortions and resulting consumption losses with the positive insurance effects. A marginal tax rate on labour supply of 0.15 constitutes the optimal trade-off between these opposing forces.

Welfare effects Having identified the source of utility gains and losses, we can take a closer look at the welfare effects on different generations that result from the reform experiment. The welfare effects are summarized in the left panel of Figure 11.11. Again we see that the welfare effects for different generations are quite distinct. First of all, we find that the older generations gain from the reform in the short term. This is due to two effects. On the one hand, the interest rate increases directly after the reform which causes private wealth, especially of the retired generations ($t = -10, -9, -8$), to appreciate. On the other hand, for elderly workers most of the uncertainty regarding their labour productivity is already resolved. Hence, for these generations the reduction in insurance does not matter that much compared to the reduced labour-supply distortions, which explains their utility gain. As generations get younger, however, the remaining labour-productivity risk becomes more and more important and they value insurance much more. Owing to that, the welfare effect quickly turns negative. Note that there is quite some downward jump in the welfare effect between the youngest current generation $t = 0$ and the first future generation $t = 1$. The difference between these generations is mainly that future generations' welfare is evaluated behind the Rawlsian veil of ignorance, i.e. before these generations know their persistent labour-productivity shock θ . Consequently, for cohorts born in $t = 1$ and later, the reduction in insurance is more costly than for cohorts that already know the persistent shock. Finally, the gradual reduction in welfare losses for cohorts born after period 1 is due to rising wages along the transition (remember the

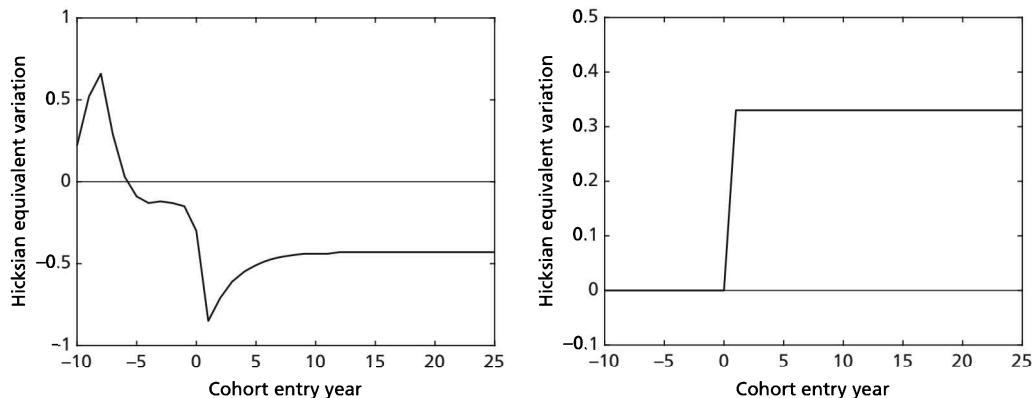


Figure 11.11 Welfare effects of regressive labour-earnings taxation

golden rule). Overall the positive welfare effects on the current elder generations more than outweigh the negative welfare effects of all future cohorts. Hence, the welfare change of future generations after compensation through the LSRA is positive, as shown in the right panel of Figure 11.11.

Note finally that only future generations value the benefits of ex ante redistribution through the tax system. This explains why the optimal tax system features a higher marginal tax rate on labour earnings if we maximize long-run welfare instead of aggregate efficiency. This is shown in Figure 11.8 where the resulting marginal tax rate on labour earnings rises to 0.25 per cent if long-run welfare is maximized. The desired tax system is then really progressive, meaning that there is a positive lump-sum transfer to every working-age individual. Section 11.3.3 sheds more light on the difference between aggregate efficiency and long-run welfare maximization.

11.3.3 SHOULD CAPITAL INCOME BE TAXED?

The question whether capital income should be taxed or not has a long tradition in both the macroeconomics and the public finance literature. In the following, we want to analyse this issue by varying both the tax rate on capital income as well as the marginal tax rate on labour earnings in our economy and calculate the associated reform path. As in the previous subsection, 11.3.2, the lump-sum transfer Tr_t is used to balance the government's budget. For each combination of τ^r and τ^w we then determine the aggregate efficiency effect Δ^* as well as the change in long-run welfare Δ_∞ . We vary the tax rates up to the point where we cannot increase efficiency or long-run welfare any further. Table 11.6 summarizes the optimal tax system when aggregate efficiency or long-run welfare is taken as a measure of optimality, respectively.

We find that the optimal income tax system that maximizes aggregate efficiency is very close to the tax system of the previous section, 11.2. In fact, the system features the same marginal tax rate on labour income and a capital-income tax rate of $\tau^r = 0.20$, while in the initial equilibrium and therefore, like in Section 11.2, the capital-income tax rate was $\tau^r = 0.2088$. The associated gain in aggregate efficiency is 0.33 per cent of initial equilibrium resources.²¹ However in the long-run, households would lose about

Table 11.6 Optimal capital-income tax rate

| Criterion | τ^r | τ^w | Tr_∞ | Δ^* | Δ_∞ |
|------------------|----------|----------|-------------|------------|-----------------|
| Efficiency | 0.20 | 0.15 | -4.64 | 0.33 | -0.48 |
| Long-run welfare | 0.49 | 0.19 | 4.13 | -1.20 | 0.74 |

²¹ As one would expect, this gain is slightly higher than the one derived in Section 11.3.2, where we only optimized the labour-income tax rate.

0.5 per cent of their resources from such a reform scenario. If we take long-run welfare as a measure of optimality, the optimal tax system looks quite different. In fact it comprises a much higher tax rate on capital income of $\tau^r = 0.49$ and a mildly higher tax rate on labour earnings of 0.19. Yet, the tax rate on labour earnings is lower than in the initial equilibrium. Since the government generates a lot of revenue from the taxation of capital income, it can also pay a lump-sum transfer that amounts to a little more than 4 per cent of average labour income. The tax system that maximizes aggregate efficiency is very close to the tax system of the previous section, 11.2. Hence, we do not want to discuss the effects it has on the macroeconomy and on life-cycle allocations in too much detail. Instead, in the following we will contrast the two tax systems—the one that maximizes efficiency and the one that maximizes long-run welfare—and try to figure out why they are so distinct and where their optimality comes from.

Macroeconomic implications Table 11.7 summarizes the macroeconomic effects of moving from the status quo tax system to the one that maximizes aggregate efficiency. The macroeconomic effects of such a reform are basically identical to the ones in Table 11.5. Since a reduction in the marginal tax rate alleviates distortions on households' labour supply, economic performance is enhanced. Labour supply increases substantially in the first period of the transition while the capital stock is slowly built up over time. Factor prices, aggregate consumption, and production as well as the lump-sum tax move accordingly.

Table 11.8 reports what happens to the macroeconomy when we implement the tax system that maximizes long-run welfare. As the tax rate on capital income increases substantially, individuals will start reducing their saving compared to the initial equilibrium. Consequently the capital stock declines throughout the transition and converges to a level that is -18.39 per cent lower than in the initial equilibrium. Labour supply is also negatively affected by the reform. The reason for this may be twofold. On the one hand, the reduced incentive to save (and consume in the future) also reduces the incentive to earn income at younger ages, when labour hours are greater. In addition

Table 11.7 Moving to the efficient capital-income tax ($\tau^r = 0.20$, $\tau^w = 0.15$)

| <i>t</i> | $T\tau_t^b$ | <i>C</i> | <i>A</i> | <i>K</i> | <i>L</i> | r^a | <i>w</i> |
|----------|-------------|----------|----------|----------|----------|-------|----------|
| 1 | -5.31 | 3.36 | 0.00 | 0.00 | 5.30 | 0.33 | -1.84 |
| 2 | -5.10 | 4.28 | 1.91 | 2.29 | 4.80 | 0.16 | -0.87 |
| 3 | -4.94 | 4.96 | 3.10 | 3.71 | 4.67 | 0.06 | -0.33 |
| 4 | -4.84 | 5.45 | 3.90 | 4.68 | 4.59 | -0.01 | 0.03 |
| 5 | -4.77 | 5.75 | 4.42 | 5.30 | 4.53 | -0.05 | 0.26 |
| : | : | : | : | : | : | : | : |
| ∞ | -4.64 | 6.37 | 5.41 | 6.49 | 4.46 | -0.12 | 0.70 |

^a Change in percentage points, ^b As fraction of average income in $t = 0$.

Table 11.8 Moving to the long-run optimal capital-income tax ($\tau^r = 0.49$, $\tau^w = 0.19$)

| t | $\bar{T}r_t^b$ | C | A | K | L | r^a | w |
|----------|----------------|----------|----------|----------|----------|----------|----------|
| 1 | 5.36 | 2.36 | 0.00 | 0.00 | -3.16 | -0.21 | 1.17 |
| 2 | 5.05 | -0.17 | -4.46 | -5.35 | -2.80 | 0.17 | -0.95 |
| 3 | 4.83 | -1.87 | -7.56 | -9.07 | -2.45 | 0.46 | -2.49 |
| 4 | 4.67 | -2.96 | -9.72 | -11.66 | -2.17 | 0.67 | -3.60 |
| 5 | 4.53 | -3.85 | -11.32 | -13.58 | -2.00 | 0.83 | -4.43 |
| \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots | \vdots |
| ∞ | 4.13 | -6.26 | -15.36 | -18.42 | -1.56 | 1.25 | -6.54 |

^a Change in percentage points, ^b As fraction of average income in $t = 0$.

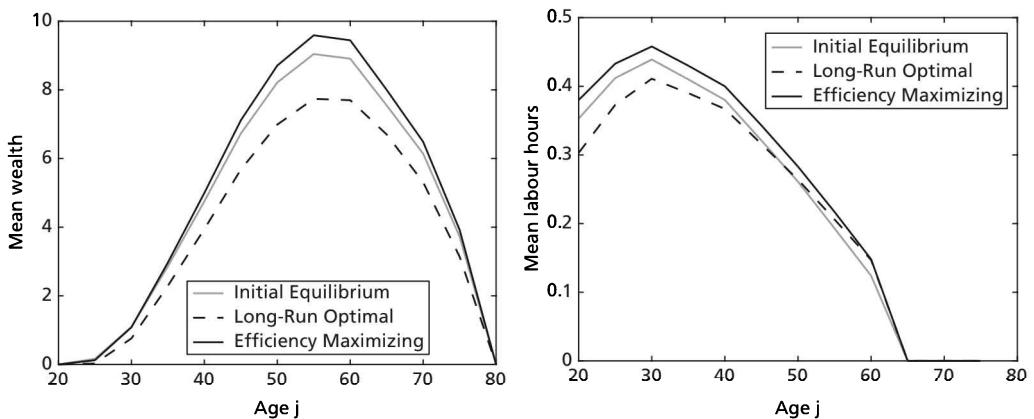


Figure 11.12 Life-cycle profiles before and after reform

to this implicit taxation mechanism, working-age households now receive a lump-sum transfer from the tax system. This transfer causes an income effect that increases both consumption and leisure, therefore resulting in lower labour supply. As the lump-sum transfer declines throughout the transition (because of falling tax revenues induced by a shrinking economy), aggregate labour supply somewhat recovers and ends up at -1.56 percent of initial equilibrium labour supply. Prices naturally adjust according to the movement in aggregate capital and labour. Finally, due to reduced savings and lower investment, aggregate consumption increases in the first period of the transition. In later periods when the economy has contracted further, consumption falls again and reaches a level that is 6.24 per cent below the initial equilibrium.

Life-cycle effects Figure 11.12 summarizes the wealth and labour hours profiles over the life cycle in the initial equilibrium as well as in the long-run equilibria resulting from the two tax reforms discussed above. The differences in the reaction in terms of private wealth are obvious. In the efficiency-maximizing tax system, household labour-earnings substantially increase. Owing to the consumption-smoothing motive of the household,

private retirement savings need to rise as well. When we implement a much higher capital-income tax rate, capital accumulation is damped and the resulting wealth profile lies below that of the initial equilibrium.

The reaction in terms of labour hours in the right panel of Figure 11.12 is much more interesting. Here we can see directly the implicit taxation effect of a capital-income tax. While under a capital-income tax rate of $\tau^r = 0.20$, labour supply mostly reacts to the lower marginal tax rate on earnings and therefore labour hours increase unanimously over the whole working life, things look different when we choose a higher capital-income tax rate. Since the latter burdens the returns on savings made from previous labour income, it implicitly burdens labour supply in early years of the life cycle. Consequently, it is somewhat equivalent to decreasing the marginal tax rate on labour income over the life cycle. This is reflected in the labour-hours profile. At early stages of the life cycle, labour supply decreases quite a bit while it actually rises over initial equilibrium levels shortly before retirement. This is not too surprising, since the marginal tax rate on labour earnings is lower after the reform than in the initial equilibrium. The question at hand then is why taxing labour supply at the beginning of the life cycle is welfare improving for households. This will most certainly only be the case if the elasticity of labour supply with respect to (implicit) tax rates is low at the beginning of working life and high at the end. So in order to make a judgement about the optimality of positive capital-income taxes in our model, we have to calculate the elasticity of labour supply with respect to the wage rate by age. We will therefore use the concept of the Frisch compensated elasticity, which is a very common indicator of a compensated elasticity and easy to calculate in the present setup. The Frisch compensated elasticity is the elasticity of labour supply with respect to the wage rate keeping the marginal utility of consumption constant. In order to calculate this elasticity, recall that the first-order conditions of the model with respect to consumption and leisure were

$$\nu \frac{[c^\nu(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{c} = \lambda p \quad \text{and} \quad (1-\nu) \frac{[c^\nu(1-l)^{1-\nu}]^{1-\frac{1}{\gamma}}}{1-l} = \lambda w^n$$

with w^n being the net wage rate. From these two conditions we can immediately derive that $c = \frac{\nu}{1-\nu} \cdot \frac{w^n}{p} \cdot (1-l)$. Plugging this into the first-order condition for leisure we obtain after some calculation

$$l = 1 - \frac{(\nu/p)^{\nu(\gamma-1)} \cdot (1-\nu)^{\gamma-\nu(\gamma-1)}}{(w^n)^{\gamma-\nu(\gamma-1)} \cdot \lambda^\gamma}.$$

If we now want to derive the elasticity of l with respect to w^n we have to calculate

$$\eta_{l,w} = \frac{\partial l}{\partial w^n} \cdot \frac{w^n}{l} = [\gamma - \nu(\gamma - 1)] \cdot \frac{1-l}{l}.$$

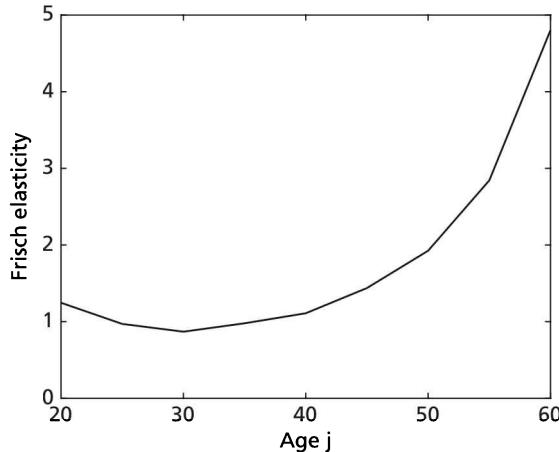


Figure 11.13 Frisch elasticity of labour supply

In our parameterization we have

$$\gamma - \nu(\gamma - 1) = 0.5 \cdot 0.36 \cdot (0.5 - 1) = 0.68$$

so that the Frisch elasticity of labour supply over the life cycle is

$$\eta_{l_j,w} = 0.68 \cdot \frac{1 - l_j}{l_j}.$$

Figure 11.13 shows how the Frisch elasticity evolves over the life cycle. To calculate the elasticity we used the average labour supply of a cohort. Note that since the variance in labour supply within a cohort is small, we do not make much of an error here. The figure shows that the Frisch elasticity of labour supply increases with age. This finding confirms that (at least in the long run) a positive capital-income tax rate should be welfare improving.

Last but not least, Figure 11.14 shows the average consumption profile over the life cycle as well as the variance of its logs. While the efficiency-maximizing tax system increases mean consumption and its variance, the tax system that maximizes long-run welfare does exactly the opposite although the marginal tax rate on labour earnings is lower than in the initial equilibrium. However, in this tax system the government generates a lot of revenue out of the taxation of capital income, which it again distributes to households in a lump-sum fashion. This lump-sum transfer that is not tied to individual income causes the variance of consumption to decline.

Welfare effects Figure 11.15 finally shows the welfare effects for different cohorts resulting from the two reform scenarios. Under the efficiency-maximizing tax reform, current retirees gain from the short-run increase in the interest rate. Older workers' welfare

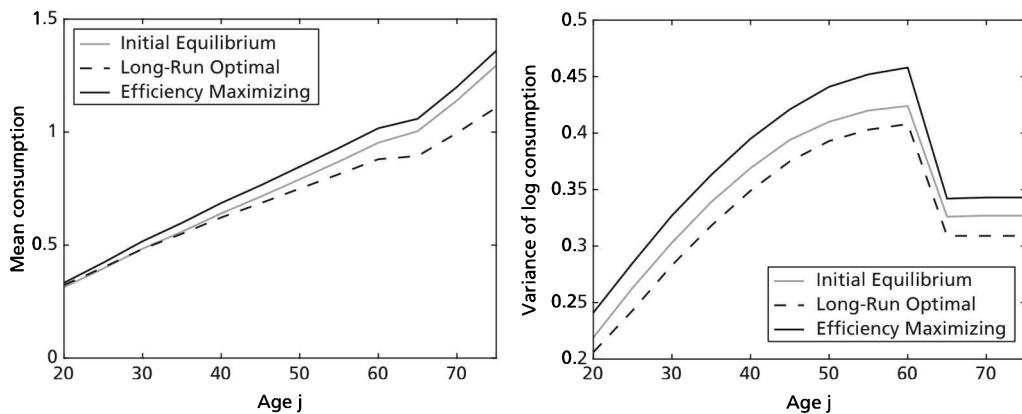


Figure 11.14 Life-cycle profiles of consumption

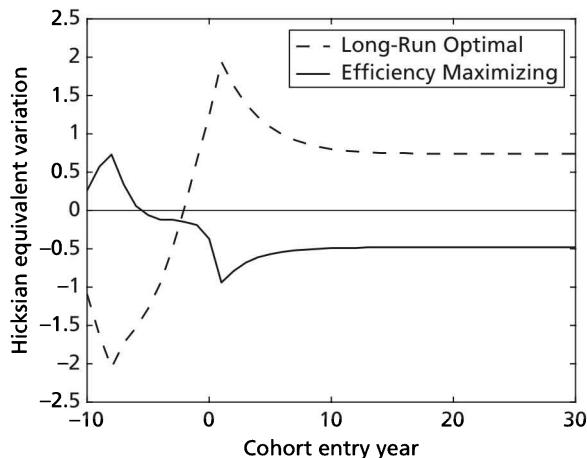


Figure 11.15 Welfare effects of different capital-income tax reforms

rises, since distortions on labour supply are alleviated and these generations do not value insurance very highly. Younger generations, who on the other hand do value insurance, experience a negative welfare effect. Throughout the transition, future generations benefit slightly from rising wages. However, the long-run welfare effects remain negative.

In order to understand the welfare effects resulting from implementing the long-run optimal tax system, we have to recall that this system generates insurance against consumption fluctuations and changes tax burdens across generations. The increased tax rate on capital income especially burdens retirees as well as older workers. These cohorts not only experience a devaluation of their wealth, they also suffer from a major distortion of their future savings. Since they hardly benefit from improved insurance provision, they realize severe welfare losses. Younger current generations pay capital-income taxes

later in life, so that their overall tax burden is smaller in present value. In addition, they benefit from the fact that a capital-income tax implicitly burdens labour earnings that emerge early in life. As they also value insurance highly, those generations experience a positive welfare effect. In this scenario, we can again see a jump in welfare effects between the youngest current and the first future cohort, which is explained by the latter valuing ex ante redistribution. Welfare gains decline throughout the transition path owing to the contraction of the aggregate economy and the resulting decline in wages. Yet, the welfare effect still remains positive in the long-run equilibrium.

In terms of aggregate efficiency, the long-run optimal tax system creates a trade-off between increased savings distortions and improve insurance provision against fluctuations in consumption. As the negative distortion effect weighs more than the positive insurance effect, the reform induces an aggregate efficiency loss. The efficiency-maximizing tax system, on the other hand, creates an ideal balance between distortions and insurance, so that aggregate welfare rises by 0.33 per cent.

11.4 Further reading

The stochastic OLG model is by now a state-of-the-art workhorse model for all kinds of tax and transfer policy analyses. There is a wealth of studies that deal with such issues, so we can only discuss a very limited selection. With respect to the calibration of our model, we borrowed a lot from the work of Storesletten, Telmer, and Yaron (2004) on the wage process. Estimates for the productivity profile are taken from Hansen (1993) and finally we also used a lot of information from Conesa, Kitao, and Krueger (2009), who calibrate a similar model setup to the US economy. As an alternative to this simple calibration procedure, one could also apply a *generalized method of moments* as introduced by Hansen (1982) or its extension the *method of simulated moments* that has first been suggested by McFadden (1989). These methods are more formal in that they specify a set of parameters and a larger set of targets. The goal then is to numerically find a parameter combination that minimizes a weighted sum of squared differences between the model outcomes and the respective target values.

Imrohoroglu, Imrohoroglu, and Joines (1995) were among the first to simulate the long-run impact of pay-as-you-go-financed pension systems in an OLG model with uncertain labour earnings. They model earnings risk using a binary state variable that either takes a value of 1 if the agent is given the opportunity to work in a specific period or zero if the agent is unemployed. This fairly ad hoc procedure was refined by Storesletten, Telmer, and Yaron (1999), who introduced three persistent ability levels as well as serially correlated and transitory shocks to labour income. They also estimated the parameters of the specified stochastic earnings process using data from the Panel Study on Income Dynamics (PSID). While Storesletten, Telmer, and Yaron (1999) partially considered

transitional generations, Huang, Imrohoroglu, and Sargent (1997) were among the first to compute the transition path between steady states in order to separate intergenerational redistribution from efficiency effects. To do so, they aggregated wealth-equivalent welfare gains and losses of all cohorts along the transition path. The concept of LSRA transfers was first applied in a stochastic OLG framework by Nishiyama and Smetters (2007) and Fehr, Habermann, and Kindermann (2008).

It should be clear that the desired degree of income redistribution and insurance through one government system (e.g. the pension system) depends on the amount of insurance already provided by all other tax and transfer systems (like the progressive tax system). A comprehensive analysis of the optimal degree of redistribution and insurance in society would enable varying the full set of government parameters and therefore reveal through which system insurance can be provided most effectively.

The above analysis was restricted to a fully progressive pension system, which pays lump-sum transfers to pensioners that are independent of their prior labour earnings. Yet most pension systems in operation include an income-related component. Such systems then come with smaller distortions, but also provide fewer insurance benefits. Fehr, Kallweit, and Kindermann (2013) study the optimal mix between earnings-related and lump-sum pension systems in an OLG economy with labour-productivity and disability risk. Another strong assumption of the model presented in this chapter is the absence of lifespan uncertainty for which the pension system can also provide insurance. Fehr and Habermann (2010) include uncertain survival into the stochastic OLG model and simulate a move towards funded pension systems. They consider two situations. One in which annuitized private retirement accounts are available to households, and one in which they are not. Following this strategy, they quantify the benefits from implicit longevity insurance in social security.

Our analysis of tax progressivity features the most simple structure of a progressive earnings tax, namely a constant marginal tax rate on all individuals paired with a lump-sum transfer. While this specification is often used in the public finance literature to ensure analytical tractability, the quantitative macroeconomic literature typically uses more elaborate tax structures. Conesa, Kitao, and Krueger (2009) and Fehr and Kindermann (2015), for example, adopt the tax function proposed by Gouveia and Strauss (1994). Benabou (2002), on the other hand, proposes a relatively simple progressive tax function that causes households' disposable income to be a log-linear function of labour earnings. That positive taxes on capital income enhance both long-run welfare and aggregate efficiency in stochastic OLG economies can nowadays be considered a standard result. Nishiyama and Smetters (2005) and Conesa, Kitao, and Krueger (2009) are quite influential studies in this direction, which challenged the seminal works of Chamley (1986), Judd (1985), and Atkeson, Chari, and Kehoe (1999) who argued in favor of zero capital-income taxation. Erosa and Gervais (2002) discuss how varying elasticities of labour supply and consumption over the life cycle can shape the optimality result regarding capital-income taxation. Fehr and Kindermann (2015) offer a comparison of different government objectives and their implications for the optimal income

tax schedule. Kindermann and Krueger (2015) analyse a model with specific income dynamics that allow us to generate realistic tails of the wealth and income distribution.

Finally, we want to mention some obvious extensions of the model structure. First, in the present model initial abilities are of purely stochastic nature, meaning that there is no upfront investment involved that might increase individual labour productivity. Kindermann (2012), Krueger and Ludwig (2013), and Badel and Huggett (2014) relax this assumption by considering an education investment decision and an on-the-job human-capital investment, respectively. Second, since a normative reason for some government programs can be to protect short-sighted individuals, one can also alter the assumption of full rationality. Imrohoroglu, Imrohoroglu, and Joines (2003) and Fehr and Kindermann (2010) augment the standard life-cycle model with hyperbolic discounting consumers, while Kumru and Thanopoulos (2008) use preferences according to which agents are tempted to consume their entire endowment in every period. Third, we can relax the assumption that cohorts are populated only by one-earner single households by considering intergenerational linkages as Fuster, Imrohoroglu, and Imrohoroglu (2007) or distinguishing singles and couples as Kaygusuz (2015) or Fehr, Kallweit, and Kindermann (2017).

11.5 Exercises

- 11.1. Try to verify the arguments brought forward in the last two sections, Sections 11.3 and 11.4. Therefore set the variances of all shocks to zero and recalculate the optimal tax systems. What does the optimal tax structure look like. Now assume that the government can only pay lump-sum transfers but not levy lump-sum taxes. How does this change the optimal tax system when the government can freely choose the capital-income tax rate? Explain your results.
- 11.2. Implement a small open economy in the tax model and figure out what the optimal capital-income tax rate looks like when capital is perfectly mobile across countries. What are your assumptions about the tax treatment of home and foreign capital?
- 11.3. Calculate Euler equation errors in the OLG model. How accurate is our solution?
- 11.4. Introduce Epstein-Zin preferences and simulate the above reforms with alternative assumptions about risk-aversion.
- 11.5. Solve the OLG model using value function iteration.
- 11.6. Assume that the elimination of the pension system is announced with a one period time lag. Does the policy announcement increase or decrease aggregate efficiency losses?
- 11.7. Introduce uncertain lifespan into the OLG model and compute the optimal size of the pension system. Why is it different compared to what we find in the benchmark model?

- 11.8. Introduce a pension benefit that is a function of the earnings history of the household instead of a flat transfer (see Fehr, Kallweit, and Kindermann (2013)). What is the optimal size of the pension system?
- 11.9. Implement a flat income tax with a basic allowance $T_t(y, a) = \tau_t \cdot [y + ra - Tr_t]$ and compute the optimal combination of τ and Tr . Compare your results with those from the benchmark model and explain the difference.

■ BIBLIOGRAPHY

- Adda, J. and R. W. Cooper. 2003. *Dynamic Economics: Quantitative Methods and Applications*. Cambridge, MA: MIT Press.
- Aiyagari, S. R. 1994. "Uninsured Idiosyncratic Risk and Aggregate Savings." *Quarterly Journal of Economics* 109(3):659–84.
- Alan, S. 2006. "Entry Cost and Stock Market Participation over the Life Cycle." *Review of Economic Dynamics* 9(4):588–611.
- Angeletos, G. M., D. Laibson, A. Repetto, J. Tobacman, and S. Weinberg. 2001. "The Hyperbolic Consumption Model: Calibration, Simulation, and Empirical Evaluation." *Journal of Economic Perspectives* 15(3): 47–68.
- Arratia, A. 2014. *Computational Finance: An Introductory Course with R*. Paris: Atlantis Press.
- Aruoba, S. B. and J. Fernandez-Villaverde. 2015. "A Comparison of Programming Languages in Macroeconomics." *Journal of Economic Dynamics and Control* 58:265–73.
- Aruoba, S. B., J. Fernandez-Villaverde, and J. F. Rubio-Ramirez. 2006. "Comparing Solution Methods for Dynamic Equilibrium Economies." *Journal of Economic Dynamics and Control* 30:2477–508.
- Atkeson, A., V. V. Chari, and P. J. Kehoe. 1999. "Taxing Capital Income: A Bad Idea." *Federal Reserve Bank of Minneapolis Quarterly Review* 23(3):3–17.
- Attanasio, O., H. Low, and V. Sanchez-Marcos. 2008. "Explaining Changes in Female Labor Supply in a Life-Cycle Model." *American Economic Review* 98(4):1517–52.
- Attanasio, O. and G. Weber. 2010. "Consumption and Saving: Models of Intertemporal Allocation and their Implications for Public Policy." *Journal of Economic Literature* 48(3):693–751.
- Auerbach, A. J. and L. J. Kotlikoff. 1987. *Dynamic Fiscal Policy*. Cambridge: Cambridge University Press.
- Badel, A. and M. Huggett. 2014. "Taxing Top Earners: A Human Capital Perspective." Federal Reserve Bank of St. Louis Working Paper 2014-017A.
- Baranzini, M. 2005. "Modigliani's Life-Cycle Theory of Savings Fifty Years Later." *Banco Nazionale del Lavoro Quarterly Review* 58(233/234):109–72.
- Barillas, F. and J. Fernandez-Villaverde. 2007. "A Generalization of the Endogenous Grid Method." *Journal of Economic Dynamics and Control* 31:2698–712.
- Beaudry, P. and F. Portier. 2007. "When Can Changes in Expectations Cause Business Cycle Fluctuations in Neo-Classical Settings?" *Journal of Economic Theory* 135(1):458–77.
- Benabou, R. 2002. "Tax and Education Policy in a Heterogeneous Agent Economy: What Levels of Redistribution Maximize Growth and Efficiency?" *Econometrica* 70(2):481–517.
- Bencivenga, V. R. 1992. "An Econometric Study of Hours and Output Variation with Preference Shocks." *International Economic Review* 33(2):449–71.
- Benninga, S. 2014. *Financial Modeling*. 4th ed. Cambridge, MA: MIT Press.
- Bewley, T. F. 1986. Stationary Monetary Equilibrium with a Continuum of Independently Fluctuating Consumers. In *Contributions to Mathematical Economics in Honor of Gerard Debreux*, ed. W. Hildenbrand and A. Mas-Colell. Amsterdam: North-Holland pp. 79–102.
- Bouzahzah, M., D. De la Croix, and F. Docquier. 2002. "Policy Reforms and Growth in Computable OLG Economies." *Journal of Economic Dynamics and Control* 26(12):2093–113.

- Brandimarte, P. 2002. *Numerical Methods in Finance: A Matlab-Based Introduction*. New York: Wiley.
- Bremus, F. M. and V. Kuzin. 2014. "Unemployment and Portfolio Choice: Does Persistence Matter?" *Journal of Macroeconomics* 40:99–113.
- Broer, P. and J. Lassila. 1997. *Pension Policies and Public Debt in Dynamic CGE Models*. Heidelberg: Physica Verlag.
- Browning, M. and T. F. Crossley. 2001. "The Life-cycle Model of Consumption and Saving." *Journal of Economic Perspectives* 15(3):3–22.
- Burfischer, M. E. 2011. *Introduction to Computable General Equilibrium Models*. Cambridge: Cambridge University Press.
- Bütler, M., K. Peijnenburg, and S. Straubli. 2017. "How Much Do Means-tested Benefits Reduce the Demand for Annuities?" *Journal of Pension Economics and Finance* 16(4):419–49.
- Buyse, T., F. Heylen, and R. van de Kerckhove. 2017. "Pension Reform in an OLG Model with Heterogeneous Abilities." *Journal of Pension Economics and Finance* 16(3):144–72.
- Campanale, C. 2009. "Life-cycle Portfolio Choice: The Role of Heterogeneous Under-diversification." *Journal of Economic Dynamics and Control* 33(9):1682–98.
- Campbell, J. Y., J. F. Cocco, F. J. Gomes, and P. J. Maenhout. 2001. Investing Retirement Wealth—A Life-Cycle Model. In *Risk Aspects of Social Security Reform*, ed. J. Y. Campbell and J. F. Cocco. Chicago and London: University of Chicago Press pp. 439–82.
- Capinski, M. J. and T. Zastawniak. 2012. *Numerical Methods in Finance with C++*. Cambridge: Cambridge University Press.
- Cardenete, M. A., A.-I. Guerra, and F. Sancho. 2012. *Applied General Equilibrium: An Introduction*. Berlin: Springer.
- Carroll, C. 2006. "The Method of Endogenous Gridpoints for Solving Dynamic Stochastic Optimization Problems." *Economics Letters* 91(3):312–20.
- Chai, J. J., W. Horneff, R. Maurer, and O. S. Mitchell. 2011. "Optimal Portfolio Choice over the Life-Cycle with Flexible Work, Endogenous Retirement, and Lifetime Payouts." *Review of Finance* 15(4):875–907.
- Chamley, C. 1986. "Optimal Taxation of Capital Income in General Equilibrium with Infinite Lives." *Econometrica* 54(3):607–22.
- Chapman, S. J. 2004. *Fortran 95/2003 for Scientists and Engineers*. 3rd ed. Boston: McGraw Hill.
- Chivers, I. and J. Sleightholme. 2012. *Introduction to Programming with Fortran*. 2nd ed. London: Springer.
- Cocco, J. F., F. J. Gomes, and P. J. Maenhout. 2005. "Consumption and Portfolio Choice over the Life Cycle." *Review of Financial Studies* 18(2):491–533.
- Conesa, J. C., S. Kitao, and D. Krueger. 2009. "Taxing Capital? Not a Bad Idea after All!" *American Economic Review* 99(1):25–48.
- Cooley, T. F. and E. C. Prescott. 1995. Economic Growth and Business Cycles. In *Frontiers of Business Cycle Research*, ed. T. F. Cooley. Princeton: Princeton University Press pp. 1–38.
- Cox, S. H., Y. Lin, R. Tian, and L. F. Zuluaga. 2013. "Mortality Portfolio Risk Management." *Journal of Risk and Insurance* 80(4):853–90.
- De Nardi, M. 2004. "Wealth Inequality and Intergenerational Links." *Review of Economic Studies* 71(3): 743–68.
- De Nardi, M., E. French, and J. B. Jones. 2010. "Why Do the Elderly Save? The Role of Medical Expenses." *Journal of Political Economy* 118(1):39–75.
- De Waegenaere, A., B. Melenberg, and R. Stevens. 2010. "Longevity Risk." *De Economist* 158(2):151–92.
- Deaton, A. 2005. "Franco Modigliani and the Life-Cycle Theory of Consumption." *Banco Nazionale del Lavoro Quarterly Review* 58(233/234):91–107.

- Epstein, L. G. and S. E. Zin. 1989. "Substitution, Risk Aversion, and the Temporal Behavior of Consumption and Asset Returns: A Theoretical Framework." *Econometrica* 57:937–69.
- Erosa, A. and M. Gervais. 2002. "Optimal Taxation in Life-Cycle Economies." *Journal of Economic Theory* 105(2):338–69.
- Fehr, H. 1999. *Welfare Effects of Dynamic Tax Reforms*. Tübingen: Mohr Siebeck.
- Fehr, H. and C. Habermann. 2010. "Private Retirement Savings: The Structure of Tax Incentives and Annuitization." *International Tax and Public Finance* 17:640–61.
- Fehr, H., C. Habermann, and F. Kindermann. 2008. "Social Security with Rational and Hyperbolic Consumers." *Review of Economic Dynamics* 11:884–903.
- Fehr, H., M. Kallweit, and F. Kindermann. 2013. "Should Pensions Be Progressive?" *European Economic Review* 63:94–116.
- Fehr, H., M. Kallweit, and F. Kindermann. 2017. "Families and Social Security." *European Economic Review* 91:30–56.
- Fehr, H. and F. Kindermann. 2010. "Pension Funding and Individual Accounts in Economies with Lifecyclers and Myopes." *CESifo Economic Studies* 56(3):404–43.
- Fehr, H. and F. Kindermann. 2015. "Taxing Capital along the Transition—Not A Bad Idea after All?" *Journal of Economic Dynamics and Control* 51:64–77.
- Fehr, H., C. Rosenberg, and W. Wiegard. 1995. *Welfare Effects of Value-Added Tax Harmonization in Europe*. Berlin: Springer.
- Fernandez, R. and J. C. Wong. 2014. "Divorce Risk, Wages and Working Wives: A Quantitative Life-Cycle Analysis of Female Labor Force Participation." *Economic Journal* 124(576):319–58.
- French, E. 2005. "The Effects of Health, Wealth, and Wages on Labour Supply and Retirement Behaviour." *Review of Economic Studies* 72(2):395–427.
- Fuster, L., A. Imrohoroglu, and S. Imrohoroglu. 2007. "Elimination of Social Security in a Dynastic Framework." *Review of Economic Studies* 74(1):113–45.
- Galí, J. 1994. "Government Size and Macroeconomic Stability." *European Economic Review* 38(1):117–32.
- Gatzert, N. and H. Wesker. 2014. "Mortality Risk and its Effect on Shortfall and Risk Management in Life Insurance." *Journal of Risk and Insurance* 81(1):57–90.
- Gomes, F. J., L. J. Kotlikoff, and L. M. Viceira. 2008. "Optimal Life-Cycle Investing with Flexible Labor-Supply: A Welfare Analysis of Life-Cycle Funds." *American Economics Review P&P* 98(2):297–303.
- Gomes, F. J. and A. Michaelides. 2005. "Optimal Life-Cycle Asset Allocation: Understanding the Empirical Evidence." *Journal of Finance* 60(2):869–904.
- Gomes, F. J., A. Michaelides, and V. Polkovnichenko. 2009. "Optimal Savings with Taxable and Tax-deferred Accounts." *Review of Economic Dynamics* 12(4):718–35.
- Gottfried, P. and W. Wiegard. 1991. "Exemption versus Zero Rating: A Hidden Problem of VAT." *Journal of Public Economics* 46(3):307–28.
- Gouveia, M. and R. Strauss. 1994. "Effective Federal Individual Tax Functions: An Exploratory Empirical Analysis." *National Tax Journal* 47(2):317–39.
- Greenwood, J., Z. Hercowitz, and G. W. Huffman. 1988. "Investment, Capacity Utilization, and the Real Business Cycle." *American Economic Review* 78(3):402–17.
- Hansen, G. D. 1985. "Indivisible Labor and the Business Cycle." *Journal of Monetary Economics* 16(1):309–27.
- Hansen, G. D. 1993. "The Cyclical and Secular Behaviour of the Labour Input: Comparing Efficiency Units and Hours Worked." *Journal of Applied Econometrics* 8(1):71–80.
- Hansen, L. P. 1982. "Large Sample Properties of Generalized Method of Moments Estimators." *Econometrica* 50(4):1029–54.

- Harrison, G. W., S. E. H. Jensen, L. H. Pedersen, and T. F. Rutherford, eds. 2000. *Using Dynamic General Equilibrium Models for Policy Analysis*. Amsterdam: North-Holland.
- Heer, B. and A. Maussner. 2009. *Dynamic General Equilibrium Modeling*. 2nd ed. Berlin: Springer.
- Heijdra, B. J., J. O. Mierau, and L. S. M. Reijnders. 2014. "A Tragedy of Annuitization? Longevity Insurance in General Equilibrium." *Macroeconomic Dynamics* 18(7):1607–34.
- Horneff, W. J., R. H. Maurer, O. S. Mitchell, and M. Z. Stamos. 2009. "Asset Allocation and Location over the Life-Cycle with Investment-linked Survival-contingent Payouts." *Journal of Banking and Finance* 33(9):1688–99.
- Horneff, W. J., R. H. Maurer, and R. Rogalla. 2010. "Dynamic Portfolio Choice with Deferred Annuities." *Journal of Banking and Finance* 34(11):2652–64.
- Horneff, W. J., R. H. Maurer, and M. Z. Stamos. 2008. "Life-Cycle Asset Allocation with Annuity Markets." *Journal of Economic Dynamics and Control* 32(11):3590–612.
- Hosoe, N., K. Gasawa, and H. Hashimoto. 2010. *Textbook of Computable General Equilibrium Modelling*. Hounds mills: Palgrave Macmillan.
- Huang, H., S. Imrohoroglu, and T. J. Sargent. 1997. "Two Computations to Fund Social Security." *Macroeconomic Dynamics* 1(1):7–44.
- Huffman, G. W. and M. A. Wynne. 1999. "The Role of Intratemporal Adjustment Costs in a Multisector Economy." *Journal of Monetary Economics* 43(2):317–50.
- Huggett, M. 1993. "The Risk-Free Rate in Heterogeneous-Agent Incomplete-Insurance Economies." *Journal of Economic Dynamics and Control* 17(5-6):953–69.
- Huggett, M. 1997. "The One-Sector-Growth Model with Idiosyncratic Shocks: Steady States and Dynamics." *Journal of Monetary Economics* 39(3):385–403.
- Hull, J. C. 2009. *Options, Futures, and other Derivatives*. 7th ed. New Jersey: Pearson.
- Imrohoroglu, A. 1989. "Cost of Business Cycles with Indivisibilities and Liquidity Constraints." *Journal of Political Economy* 97(6):1364–83.
- Imrohoroglu, A., S. Imrohoroglu, and D. H. Joines. 1995. "A Life-cycle Analysis of Social Security." *Economic Theory* 6:83–114.
- Imrohoroglu, A., S. Imrohoroglu, and D. H. Joines. 2003. "Time-inconsistent Preferences and Social Security." *Quarterly Journal of Economics* 118:745–84.
- Iskhakov, F., S. Thorp, and H. Bateman. 2015. "Optimal Annuity Purchases for Australian Retirees." *Economic Record* 91(293):139–54.
- Jappelli, T. 2005. "The Life-cycle Hypothesis, Fiscal Policy and Social Security." *Banco Nazionale del Lavoro Quarterly Review* 58(233/234):173–86.
- Johansen, L. A. 1960. *A Multi-Sectoral Study of Economic Growth*. Amsterdam: North-Holland.
- Jokisch, S. 2006. *The Developed World's Demographic Transition*. Tübingen: Mohr Siebeck.
- Judd, K. L. 1985. "Redistributive Taxation in a Simple Perfect Foresight Model." *Journal of Public Economics* 28(1):59–83.
- Judd, K. L. 1998. *Numerical Methods in Economics*. Cambridge, MA: MIT Press.
- Kaygusuz, R. 2015. "Social Security and Two-earner Households." *Journal of Economic Dynamics and Control* 59:163–78.
- Kendrick, D. A. and H. M. Amman. 1999. "Programming Languages in Economics." *Computational Economics* 14(1–2):151–81.
- Kendrick, D. A., P. R. Mercado, and H. M. Amman. 2006. *Computational Economics*. Princeton and Oxford: Princeton University Press.
- Kindermann, F. 2012. "Welfare Effects of Privatizing Public Education When Human Capital Investments Are Risky." *Journal of Human Capital* 6(2):87–123.

- Kindermann, F. and D. Krueger. 2015. "High Marginal Tax Rates on the Top 1%?" NBER Working Paper No. 20601.
- King, R. G., C. I. Plosser, and S. T. Rebelo. 1988. "Production, Growth and Business Cycles." *Journal of Monetary Economics* 21(2–3):195–232.
- Kopecky, K. A. and R. M. H. Suen. 2010. "Finite State Markov-Chain Approximations to Highly Persistent Processes." *Review of Economic Dynamics* 13(3):701–14.
- Kotlikoff, L. J. 2000. The A-K OLG Model: Its Past, Present, and Future. In *Using Dynamic General Equilibrium Models for Policy Analysis*, ed. G. W. Harrison, S. E. H. Jensen, L. H. Pedersen, and T. F. Rutherford. Amsterdam: North-Holland pp. 13–52.
- Krueger, D. and A. Ludwig. 2013. "Optimal Progressive Labor Income Taxation and Education Subsidies When Education Decisions and Intergenerational Transfers Are Endogenous." *American Economic Review* 103(2):496–501.
- Krusell, P. and A. A. Smith. 1998. "Income and Wealth Heterogeneity in the Macroeconomy." *Journal of Political Economy* 106(5):867–96.
- Kumru, C. S. and A. C. Thanopoulos. 2008. "Social Security and Self Control Preferences." *Journal of Economic Dynamics and Control* 32:757–78.
- Kydland, F. E. and E. C. Prescott. 1982. "Time to Build and Aggregate Fluctuations." *Econometrica* 50(6):1345–70.
- Le Blanc, J. and A. Scholl. 2017. "Optimal Savings for Retirement: The Role of Individual Accounts." *Macroeconomic Dynamics* 21:1361–88.
- Love, D. A. 2006. "Buffer Stock Saving in Retirement Accounts." *Journal of Monetary Economics* 53(7): 1473–92.
- Love, D. A. 2010. "The Effect of Marital Status and Children on Savings and Portfolio Choice." *Review of Financial Studies* 23(1):385–432.
- Low, H., C. Meghir, and L. Pistaferri. 2010. "Wage Risk and Employment Risk over the Life Cycle." *American Economic Review* 100(4):1432–67.
- Marcet, A., F. Obiols-Homs, and F. Weil. 2007. "Incomplete Markets, Labor Supply and Capital Accumulation." *Journal of Monetary Economics* 54(8):2621–35.
- McCandless, G. T. and N. Wallace. 1991. *Introduction to Dynamic Macroeconomic Theory: An Overlapping Generations Approach*. Cambridge, MA: Harvard University Press.
- McFadden, D. 1989. "A Method of Simulated Moments for Estimation of Discrete Response Models without Numerical Integration." *Econometrica* 57(5):995–1026.
- McGrattan, E. R. 1994. "The Macroeconomic Effects of Distortionary Taxation." *Journal of Monetary Economics* 33(3):573–601.
- Miao, J. 2014. *Economic Dynamics in Discrete Time*. Cambridge, MA: MIT Press.
- Miranda, M. J. and P. L. Fackler. 2002. *Applied Computational Economics and Finance*. Cambridge, MA: MIT Press.
- Nishiyama, S. and K. Smetters. 2005. "Consumption Taxes and Economic Efficiency with Idiosyncratic Wage Shocks." *Journal of Political Economy* 113(5):1088–115.
- Nishiyama, S. and K. Smetters. 2007. "Does Social Security Privatization Produce Efficiency Gains?" *Quarterly Journal of Economics* 122(4):1677–719.
- Peijnenburg, K., T. Nijman, and B. J. M. Werker. 2016. "The Annuity Puzzle Remains a Puzzle." *Journal of Economic Dynamics and Control* 70:18–35.
- Peijnenburg, K., T. Nijman, and B. J. M. Werker. 2017. "Health Cost Risk: A Potential Solution to the Annuity Puzzle." *Economic Journal* 127(603):1598–625.
- Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. 2001. *Numerical Recipes in Fortran 77: The Art of Scientific Computing*. Cambridge: Cambridge University Press.

- Pries, M. J. 2007. "Social Security Reform and Intertemporal Smoothing." *Journal of Economic Dynamics and Control* 31(1):25–54.
- Ramaswamy, S. 2004. *Managing Credit Risk in Corporate Bond Portfolios: A Practitioner's Guide*. Hoboken: Wiley.
- Ramsey, F. P. 1928. "A Mathematical Theory of Saving." *Economic Journal* 38(152):543–59.
- Rendahl, P. 2014. "Inequality Constraints and Euler Equation-based Solution Methods." *Economic Journal* 125(585):1110–35.
- Rosenthal, R. E. 2012. *GAMS: A User's Guide*. Washington, DC: GAMS Development Corporation.
- Rouwenhorst, K. G. 1995. Asset Pricing Implications of Equilibrium Business Cycle Models. In *Frontiers of Business Cycle Research*, ed. T. F. Cooley. Princeton: Princeton University Press pp. 294–330.
- Rutherford, T. 1999. "Applied General Equilibrium Modeling with MPSGE as a GAMS Subsystem: An Overview of the Modeling Framework and Syntax." *Computational Economics* 14(1–2):1–46.
- Shoven, J. B. and J. Whalley. 1984. "Applied General Equilibrium Models of Taxation and International Trade: An Introduction and Survey." *Journal of Economic Literature* 22(3):1007–51.
- Shoven, J. B. and J. Whalley. 1992. *Applying General Equilibrium*. Cambridge: Cambridge University Press.
- Sommer, K. 2016. "Fertility Choice in a Life-cycle Model with Idiosyncratic Uninsurable Earnings Risk." *Journal of Monetary Economics* 83:27–38.
- Stokey, N. L. and R. E. Lucas. 1989. *Recursive Methods in Economic Dynamics*. Cambridge, MA: Harvard University Press.
- Storesletten, K., C. I. Telmer, and A. Yaron. 1999. "The Risk-sharing Implications of Alternative Social Security Arrangements." *Carnegie-Rochester Conference Series on Public Policy* 50:213–59.
- Storesletten, K., C. I. Telmer, and A. Yaron. 2004. "Consumption and Risk Sharing over the Life Cycle." *Journal of Monetary Economics* 51(3):609–33.
- Tauchen, G. 1986. "Finite State Markov-Chain Approximations to Univariate and Vector Autoregressions." *Economics Letters* 20(2):177–81.
- Wälde, K. 2012. *Applied Intertemporal Optimization*. <http://www.waelde.com/aio.html>.
- White, M. N. 2015. "The Method of Endogenous Gridpoints in Theory and Practice." *Journal of Economic Dynamics and Control* 60:26–41.
- Yogo, M. 2016. "Portfolio Choice in Retirement: Health Risk and the Demand for Annuities, Housing, and Risky Assets." *Journal of Monetary Economics* 80:17–34.
- Zodrow, G. R. and J. W. Diamond. 2013. Dynamic Overlapping Generations Computable General Equilibrium Models and the Analysis of Tax Policy: The Diamond-Zodrow Model. In *Handbook of Computable General Equilibrium Modeling*, ed. P. B. Dixon and D. W. Jorgenson. Vol. 1A Amsterdam: North-Holland pp. 743–813.

■ INDEX

aggregate efficiency 241, 530
aggregate fluctuations 341
aggregate productivity risk 341
aggregation 227, 385, 419, 426, 462, 508
all-in-one solution 290
American option 151, 155
announcement effect 267
annuity insurance 189, 219, 279, 469, 471, 479
annuity puzzle 218, 274
annuity value 473
antithetic variates 176
approximant 85
approximation 85
array 16, 39
Asian option 161
autarky 132
auto-correlation 341
autoregressive process 341, 407

balanced growth path 394, 506
basis coefficient 85
basis function 85
Bellman equation 295
Bellman's principle of optimality 295
benefit-price ratio 194
bequests 275
binomial option pricing approach 152
birth 431
bisection search 48, 486
Black-Scholes formula 155
bond 444
bond portfolio 173
borrowing constraint 376, 380, 392, 410
Brent's method 63
Broyden's method 56
budget constraint
 dynamic 226, 254, 293, 325, 395, 408, 423,
 432, 507
 government 121, 334, 510
 household 116, 123, 206, 226, 254, 290, 325,
 395, 408, 423, 432, 507
 intertemporal 226, 254
budget incidence 122
business cycle statistics 362

cake eating problem 289
calibration 130, 420, 426, 441, 463, 517
call (subroutine) 21
canonical form 103

capital imports 132
capital market 229, 377, 510
cash-on-hand 447
cdf 78
centralized representation 327
CES preferences 229, 290, 407
Chebychev nodes 89
child care 431
children 430
Cholesky factorization 44
closed economy 113, 225, 253, 505
Cobb-Douglas
 preferences 113, 262, 355, 423, 506
 technology 113, 230, 328,
 355, 509
coefficient of variation 361
cohort 226
command optimum 113
compensation 241, 530
compiler 5ff
complementarity constraint 147
complementarity problem 147
computable general equilibrium model 134
conditional value at risk 182, 195
constant relative risk aversion 229, 407
constraint optimization 60
consumer price 120
consumption function error 305
consumption good 113, 227, 324, 407, 506
consumption path 290
contains statement 19
contraction mapping theorem 300
control flow diagram 11
control flow statement 11
control variable 295
convergence speed 54
 linear 50
 quadratic 53
corporate bond 164
coupon payment 164
coupon rate 164
Cox-Ross-Rubinstein 152
credit rating 165
credit risk 164
credit spread 165
crowding out effect 540
cumulative distribution function 78
curve fitting problem 87
CVaR 182, 195

- data type 7
 - derived 169, 192
- death probability 185, 218, 274
- declaration of variables 7
- default 165
- default mode 166
- default probability 165
- deficit spending 228
- demand function 116, 227
- demographics 225, 249, 324, 506
- depreciation 324, 509
- derived data type 169, 192
- discretization 380
 - random variable 451
 - state space 299, 410
 - stochastic process 345, 347, 410
- distribution 77
 - Bernoulli 80, 167
 - beta 80, 177
 - binomial 81, 153
 - Gamma 80
 - log-normal 80, 155, 198, 210, 214
 - normal 74, 79, 341, 407, 446
 - uniform 79
- distribution of households 386, 417, 439, 459, 488, 508
- diversification 144
- do-loop 15
- drift 187
- dynamic general equilibrium model 225, 253, 505
- dynamic optimization problem 290
- dynamic programming 291
- economies of scale 431
- economy
 - closed 113, 225, 253, 505
 - open 130
 - small open 131
- education 264
- education investment 265
- education subsidy 264, 268, 273
- efficiency 241, 530
- efficiency frontier 144
- efficiency locus 114
- efficient allocation 113
- efficient portfolio 141
- endogenous gridpoint 317
- endogenous growth 270
- endogenous labour supply 119, 253, 422, 506
- envelope theorem 326, 409
- equidistant nodes 88, 299, 300
- equivalence scale 431
- Euler equation error 333, 363
- Euler equation residual 333
- European option 151, 153
- ex-ante welfare 529
- excess return 145
- expected loss 166, 168
- expected utility 208
- experience effect 430
- expiry date 151
- externality 265, 269
- face value 166
- factor price equalization theorem 132
- factor prices 116, 228, 324, 375, 509
- female labour force participation 429
- fertility 249, 430
- final demand 126
- firm 116, 228, 324, 375, 509
- first order autoregressive process 341
- Fischer-Burmeister function 147, 197
- fixed-point iteration 54, 300
- formatter 9
- Fortran 3
- Frisch elasticity 553
- function 21
- functional equation 295
- Gauss-Hermite quadrature 73, 451
- Gauss-Jacobi method
 - linear 45
 - non-linear 58
- Gauss-Legendre quadrature 72
- Gauss-Seidel method 512
 - linear 45
 - non-linear 58, 237, 256
- Gaussian elimination 41
- Gaussian quadrature 72
- general equilibrium model
 - dynamic 225, 253, 505
 - static 113
- general linear interpolation 319
- generation 226
- global minimum 67
- global variable 25
- GNU plot 31
- golden rule 524
- Golden-Search method 61
- goods market 229, 377, 511
- government 120, 228, 334, 396, 509
- grid 299
 - equidistant 299
 - growing 381
- grid search 301
- gridpoint 299
 - endogenous 317
- growth 261
- hand-to-mouth consumer 393
- heterogeneous agent model 374
- Hicksian equivalent variation 241, 397, 491, 527
- high-level programming 4, 5
- histogram 84, 179
- homogeneity 448

- household distribution 386
 human capital 254, 264, 430
 profile 258
 transmission 270, 272

 idiosyncratic mortality risk 218, 274, 407
 idiosyncratic shock 374, 407
 if-statement 12
 impatience 289
 imperative programming 4, 6, 8
 implicit none 7
 impulse response function 342
 include statement 24
 infinite horizon model 323
 initial guess 300
 initialization effect 385
 innovation 341
 input output table 126
 input variable 20
 instantaneous utility function 289, 363, 407
 insurance
 annuity 189, 219, 279,
 469, 471, 479
 life 190
 premium 191
 integral 68
 integration 68
 integration nodes 68
 intent statement 21
 intergenerational redistribution 241, 530
 intermediate good 126
 intermediate input 126
 intermediate value theorem 48
 international trade 130
 interpolant 86
 interpolation 85
 bilinear 96, 489
 cubic spline 91, 307, 314, 330, 353, 356
 general linear 319
 multidimensional linear 98, 476
 multidimensional splines 99
 piecewise linear 91, 319, 383, 388, 413, 437,
 454, 456
 piecewise polynomial 91
 polynomial 88
 two-dimensional 95
 intertemporal budget constraint 226, 254
 intertemporal elasticity of substitution 206, 290
 intertemporal optimization 205
 investment 325, 375
 net 332
 replacement 332
 investment risk 139, 212, 444
 IO-Table 126
 iteration in policy space 313

 Kuhn-Tucker condition 147, 196

 labour force participation 429
 labour income risk 207, 374, 407, 445
 labour market 123, 229, 377
 labour productivity 254, 407, 430
 permanent 447
 profile 258
 risk 407, 506
 labour supply 119, 253, 354, 422, 506
 least-squares approximation 87
 Lee-Carter model 186
 leisure consumption 119, 253, 354,
 422, 506
 Leontief technology 127
 life cycle 205, 406, 462
 life expectancy 185, 279, 469
 life insurance 190
 life-cycle fund 479
 linear equation system 39, 40, 128
 linear homogeneity 324
 linear programming 100
 canonical form 103
 standard form 102
 liquid assets 469
 loading factor 192, 477
 local minimum 67
 logical expression 12, 13
 logical operator 14
 long-run equilibrium 229, 511
 longevity risk 184, 218, 274
 loss given default 168
 LU-decomposition 43
 LU-factorization 43, 129
 Lump-Sum Redistribution Authority
 242, 530

 market equilibrium 117, 229, 327,
 377, 511
 market system 116
 markets 116, 229
 matrix 39
 matrix inversion 45, 142
 maturity 164
 maxloc 303
 MCM 149
 mean reverting 342
 mean-variance diagram 144
 mean-variance portfolio theory 139
 method of endogenous gridpoints 316
 migration mode 168
 minimization 60, 114, 206, 210, 216, 309
 dynamic 290
 intertemporal 205
 multi-dimensional 64
 one-dimensional 61
 minimum variance portfolio 142
 module 23
 Monte Carlo minimization 149

- Monte Carlo simulation 81, 149, 156, 161, 176, 187, 360, 385
- mortality risk 184, 218, 274
 - idiosyncratic 218, 274, 407
 - systematic 186
- mortality table 185
- natural spline 93
- neoclassical growth model 323
- Newton method 51
- Newton-Cotes method 69
- nodes
 - Chebyshev 89
 - equidistant 88
 - integration 68, 210, 212, 219, 345, 347, 451
 - interpolation 85, 308
- nominal exposure 164
- non-linear equation 47
- non-linear equation system 47, 56
- normalization by productivity 447
- open economy 130
- optimization and interpolation 306
- option
 - American 151, 155
 - Asian 161
 - call 151
 - European 151, 153
 - pricing 151
 - put 151
- option pricing 151
- optional argument 28
- orthogonal matrix 44
- oscillation 307
- output 324
- overaccumulation 393
- overlapping generations model 225, 253, 505
- parameter 8
- Pareto improvement 242, 532
- participation decision 430
- pay-as-you-go 229, 247, 260, 277, 510
- pdf 78
- pension 219, 228, 247, 260, 277, 407, 430, 445, 507, 510, 524
 - optimal size 539
- persistence 341
- plotting 28
 - three-dimensional 31
 - two-dimensional 28
- policy announcement 267
- policy function 297, 349
- policy function iteration 313
- policy reform 232, 523
- population growth 225, 506
- portfolio
 - bonds 173
 - choice 139, 214, 446, 449, 470, 481
- efficient 141
- minimum variance 142
- mortality 191
- optimal 196, 450
- optimization 139, 214
- return 140, 445
- tangent 144
- Powell's algorithm 64
- precautionary savings 208, 376
- primary factor input 126
- probability density function 78
- probability distribution 77
- probability measure 77
- probability of default 165
- procedural programming 4, 19
- producer price 120
- productivity growth 394
- productivity shock 341
- profit 324
- profit maximization 116, 228
- public debt 228, 247, 334, 510
 - optimal 394
- public expenditure 369
- public good 120, 228, 334, 396, 510
- public sector 120, 228, 334
- pure endowment insurance 190
- quadrature methods 68
- quantile 467
- Quasi-Newton Method 56
- Ramsey model 323
- random number generator 82
- random seed 82
- random variable 77
 - continuous 78
 - discrete 77
- random walk 187, 342, 445
- rating migration 166
- rating symbol 165
- Rawlsian veil of ignorance 529
- RBC 354
- read 8
- real business cycle 354
- recovery rate 166
- rectilinear grid 95
- relational operator 13
- representative agent 114
- representative agent model 323
- retirement 407, 430, 445, 507
- retirement asset 469
- risk management
 - credit risk 164
 - mortality risk 184
- risk premium 145, 215, 445
- risk sharing 505
- risk-free rate 143, 214, 444
- risky asset 139, 212, 444

- risky income 207, 374, 407, 445
root 47
root-finding 47, 51, 118, 147, 196, 232, 314, 331, 348, 358, 378, 382, 415, 437, 455, 457, 485
multi-dimensional 56
rootfinding and interpolation 314
Rouwenhorst method 345
Runge's function 89
- savings
liquid 469
old age 205, 227, 469
precautionary 207, 376
tax-favored 479
- savings function 227
- secant method 54
- secondary market 165
- seed 82
- self-insurance 393
- separation theorem 144
- short-selling constraint 146
- simplex algorithm 103
- simulation 77, 81, 304, 311, 350, 358, 385, 426, 441
- slack variable 104
- small open economy 131
- social planner 113, 328
- sorting 467
- spillover 270, 430
- spline
coefficients 93, 310
interpolation 91
natural 93
- standard form 102
- state space 299, 408
- state variable 295
- static general equilibrium model 113
- stationary distribution 342
- stationary equilibrium 328
- steady state 229, 328, 375, 511, 521
stochastic 360
- stochastic growth model 341
- stochastic steady state 360
- stock 444
- strike price 151
- subroutine 19
- summed Newton-Cotes method 69
- summed rectangle rule 69
- summed Simpson rule 71
- summed trapezoid rule 70
- systematic mortality risk 186
- tangent portfolio 144
- tax incidence 120
differential 122
- tax revenue 228
- tax system 228, 509
- tax-favored retirement account 479
- taxes
capital income 246, 334, 396, 550
consumption 120, 228, 524
income 120, 228, 245, 479, 524
labour income 246, 259, 267, 273, 334, 396, 524, 544
lump-sum 242, 369, 545
optimal progressivity 546
- Taylor approximation 363
- technological change 261
productivity 394
time augmenting 261
- technology 324
- time discount factor 206, 289, 407
- time preference rate 325
- tolerance level 300
- toolbox 27
- total factor productivity 341
- trade balance 132
- training decision 264
- transformation curve 114
- transition 234, 329, 525
- transition equation 293
- transition matrix 345, 347
- transitional dynamics 234, 329, 332, 525
- triangular matrix 41
- unconditional survival probability 188
- unemployment 123
- unemployment benefit 124
- unemployment rate 123
- unexpected loss 166, 168
- unintended bequests 275
- update 300, 313
- use (module) 25
- value assignment 11
- value at risk 181, 195
- value function 294, 416
- value function iteration 298, 300
- Vandermonde matrix 88
- VaR 181, 195
- variable declaration 7
- variable type 7
- vector 39
- wage curve 123
- Walras' law 117, 229, 377, 511
- website 27
- Weierstrass Theorem 88
- weight function 68
- weights 68, 210, 212, 219, 451
- welfare 240, 337, 363, 372, 397, 491, 527
- worforce 226
- write 8