Parameterized Expectations Approach; Some practical issues*[†]

Albert Marcet[‡]
and
Guido Lorenzoni[§]

May, 1998

*This research was partially supported by DGICYT (Spanish Ministry of Education), CIRIT (Catalan Conselleria of Education) and HCM (European Comission). All errors are our own. This chapter ows much to conversations with previous coauthors, mainly, Wouter den Haan, Ramon Marimon, David Marshall, Tom Sargent and Ken Singleton. We are in debt with the editors for their encouragement, their patience, and many helpful comments.

[†]This working paper is forthcoming as a chapter in *Computational Methods for the Study of Dynamic Economies*, Ramon Marimon and Andrew Scott, Editors), Oxford University Press, 1998.

[‡]Universitat Pompeu Fabra, CREI and London Business School.

[§]Massachussetts Institute of Technology and Universitat Pompeu Fabra.

Mailing address:

Albert Marcet
Departament d'Economia i Empresa
Universitat Pompeu Fabra
Ramon Trias Fargas, 25-27
08005 Barcelona
tel. 34-93-542-2740, fax -1746
e-mail: (marcet) albert.marcet@econ.upf.es, (lorenzoni) glorenzo@mit.edu

Abstract

We discuss some practical issues related to the use of the Parameterized Expectations Approach (PEA) for solving non-linear stochastic dynamic models with rational expectations. This approach has been applied in models of macroeconomics, financial economics, economic growth, contract theory, etc. It turns out to be a convenient algorithm, especially when there is a large number of state variables and stochastic shocks in the conditional expectations. We discuss some some practical issues having to do with the application of the algorithm, and we discuss a Fortran program for implementing the algorithm that is available through the internet. We discuss these issues in a battery of six examples.

Keywords: numerical algorithm, rational expectations, stochastic difference equations, simulation.
JEL classification: E10, C60, C63.

1 Introduction

We discuss some practical issues related to the use of the Parameterized Expectations Approach (PEA) for solving non-linear stochastic dynamic models with rational expectations. This approach is discussed formally in Marcet and Marshall (1994); it has been applied quite widely[1] as it turns out to be a convenient algorithm, especially when there is a large number of state variables and stochastic shocks in the conditional expectations. The contribution of this chapter is the detailed description of some practical issues; we discuss in detail the application of the algorithm, some tricks to speed up computation, some common pitfalls and possible solutions. We also discuss a Fortran program for implementing the algorithm. We discuss these issues in a battery of examples. Each example is described in three steps: first we show how each example fits in the general framework we consider (subsection 2.1); then we show how the algorithm is adapted to each example; third we show how the program is adapted to each example (section 4). The series of examples is as follows: Example I is the Lucas asset pricing model, and it is offered for comparison with the other chapters of this book; Example II is the simple stochastic growth model, which is the simplest model where many of the computational issues related to solving non-linear models arise. Examples III to VI are variations of the simple stochastic growth model, each of them selected to demonstrate a different issue.

We hope that this chapter will help applied economists to solve non-linear stochastic dynamic models, even if their knowledge of numerical analysis is almost non-existent; for this reason, we avoid discussing tools from numerical analysis as much as possible. The Fortran program is publicly available through the internet[2] and it contains some standardized subroutines that simplify enormously the task of applying PEA. However, the user will still have to do some programming in order to modify certain subroutines that are model-dependent. To guide the user in this task, we provide versions of the model-dependent subroutines that solve each one of the examples discussed in the text.

[1]An early description of the algorithm could be found in Marcet (1988). An early description of practical issues can be found in den Haan and Marcet (1990). Marcet and Marshall (1994) provide a long list of applications.

[2]http://www.econ.upf.es/~marcet and http://www.iue.it/Personal/Marimon/Welcome.html.

We discuss in detail how to solve models with Euler equations. Because Euler equations are easy to obtain in models with suboptimal equilibria, we can use the algorithm even in models where the equilibrium cannot be characterized as the solution to a maximization dynamic program. This is the case, for example, in models with distortionary taxes (see Example III below) or incomplete markets.

Having said what we do in the chapter, we now mention things we do *not* do. The basic version of the algorithm that we present only works to approximate the solution at the stationary distribution. The algorithm can be modified to compute the transition towards the stationary distribution from an arbitrary initial condition. Also, we only study models characterized by Euler equations. This means we only scratch the surface of the possible applications of PEA. Variations of the algorithm have been used to solve models with incentive constraints, value functions, conditional expectations of discounted sums, models with a large number of heterogeneous agents, present value budget constraints, incomplete information, etc.[3] Many of these extensions are easy to implement, but we simply do not have enough space. Finally, we only address in passing (section 5) issues related to the selection of the degree of approximation of the conditional expectation.

Technical issues are left out of the chapter. It is known that PEA can provide arbitrary accuracy if the approximation function is refined (e.g., if the degree of the polynomial approximation increases). Convergence to the correct solution is proved in Marcet and Marshall (1994). Also, we do not provide a detailed comparison with other approaches, but we only offer some informal comments on such comparisons.

The layout of the chapter is as follows. In the next section we discuss the algorithm and how it can be applied to six examples; section 3 discusses the Fortran program; section 4 the application of the program to each example; section 5 discusses some tricks, and section 6 concludes.

[3]For a more complete list of citations and to different applications see Marcet and Marshall (1994).

2 Parameterized Expectations Approach

We discuss the algorithm for models with a unique stationary and ergodic distribution[4], and we will discuss approximating the solution at the support of this distribution. In case that the model of interest is non-stationary, the current algorithm can be applied by transforming the model to a stationary one; for example, if the model has balanced growth, all variables need to be divided by the appropriate growth rate until all normalized variables have a stationary distribution. Parts of this section are taken from Marcet and Marshall (1994), including the notation, but they are offered here for completeness.

The reason that many dynamic models are difficult to solve is because conditional expectations often appear in the equilibrium conditions, as we shall show in this chapter using a series of examples. Often, we know these expectations are a time-invariant function $\mathcal{E}$ of some state variables. We can almost never derive formulas for this function analytically, but we can almost always derive formulas for the Euler equations. We know a key property of $\mathcal{E}$: under rational expectations, if agents use $\mathcal{E}$ to form their decisions, the series generated is such that $\mathcal{E}$ is precisely the best predictor of the future variables inside the conditional expectation. So, if we knew $\mathcal{E}$, we could easily simulate the model and check that this is actually the conditional expectation but, of course, we do not know $\mathcal{E}$ until we solve the model.

The basic idea in PEA to break this circle is to substitute the conditional expectations by parameterized functions of the state variables with arbitrary coefficients. Simulations of the model can then be easily obtained conditional on these coefficients and these simulations can be used to check if the pa-

[4]Often, the sufficient conditions for an equilibrium include a set of transversality conditions. For most models, stationarity of the solution is a sufficient condition for satisfying the transversality condition; therefore, since we are imposing stationarity, we can ignore the transversality condition. Since we just deal with models that have a unique solution of the first order conditions we do not address the issue that, due to non-convexities, the Euler equation might not be sufficient for the optimum (this could be a concern for example V below, which has a non-concave utility function). Similarly, we just assume existence of an equilibrium of a certain recursive form (this could be an issue in Example III below). These sins are committed by any Euler equation approach and they can be ammended in several ways that we do not discuss here. So we provide an algorithm to find the equilibrium (or the optimum) in case that it exists and that it can be described uniquely by a certain system of first order conditions.

rameterized expectation is close to the best predictor. If it is not, then we iterate on the coefficients until they deliver, approximately, the best possible prediction.

2.1 A general framework

We now discuss formally the general model to which we apply PEA. Consider an economy described by a vector of n variables z_t, and a vector of s exogenously given shocks u_t. The process $\{z_t, u_t\}$ is known to satisfy several Euler equations, feasibility constraints, equilibrium conditions, etc. which are summarized in a system

$$g(E_t[\phi(z_{t+1}, z_t)], z_t, z_{t-1}, u_t) = 0 \tag{1}$$

for all t, where $g : R^m \times R^n \times R^n \times R^s \to R^q$ and $\phi : R^{2n} \to R^m$. The vector z_t includes all endogenous variables, as well as those exogenous variables that appear inside the expectation. The process u_t is assumed to be Markov of order one. As usual, E_t denotes the conditional expectation given all information up to period t. It is assumed that, for given functional forms and parameter values for utilities, production functions, etc., the researcher knows the functions g and ϕ.

We consider solutions such that, in equilibrium, past information that is relevant for predicting $\phi(z_{t+1}, z_t)$ can be summarized in a finite-dimensional vector of state variables $x_t \in R^l$ satisfying

$$E_t[\phi(z_{t+1}, z_t)] = E[\phi(z_{t+1}, z_t)|x_t].$$

where x_t is a subset of (z_{t-1}, u_t). Furthermore, we compute solutions that satisfy a recursive framework in the sense that the conditional expectation is given by a time-invariant function $\mathcal{E}$ such that

$$\mathcal{E}(x_t) = E_t[\phi(z_{t+1}, z_t)]. \tag{2}$$

for a time-invariant function $\mathcal{E}$.

It turns out that, in many economic models of interest we can write down the functions g and ϕ that satisfy (1), and we know that (2) is satisfied for some function $\mathcal{E}$, but we do not know the form of this function.

Let us consider different examples that map into the framework (1) and (2). Since we are considering these models for simulation, we parameterize the functional form of all the fundamentals such as utility functions, production functions, etc.

- **Example I** (Lucas Asset Pricing Model)

  A representative agent maximizes the utility function $E_0 \sum_{t=0}^{\infty} \delta^t \frac{c_t^{\gamma+1}}{\gamma+1}$. There are J assets in the economy (J stocks) each in net supply equal to 1. Each stock is traded in a competitive market at price $p_{j,t}$ and pays an exogenous dividend flow $\{d_{j,t}\}$ forever. The exogenous process $\{d_t\}$ is Markov of order one, where d_t is the vector containing the dividends for all assets, and similarly for p_t. The Euler equation for the maximization problem of the consumer is

  $$p_{j,t} = \delta \; E_t \left(\frac{c_{t+1}^{\gamma}}{c_t^{\gamma}} \; (p_{j,t+1} + d_{j,t+1}) \right) . \tag{3}$$

  Market clearing implies $c_t = \sum_j d_{j,t}$. Still, we need to solve for the stock prices using (3). To map this model into the above framework let $z_t = (p_t, d_t)$, and $u_t = d_t$; the system of equations corresponding to (1) is given by (3) and $\phi_j(z', z) \equiv c^{\gamma} \; (p_j + d_j)$. Lucas' original paper shows that a time-invariant solution for the asset price can be found for which (2) is satisfied if we take $x_t \equiv d_t$. Imposing these state variables ensures that we calculate a bubble free solution.

  This is a very simple model to solve. Notice that no endogenous variables appear in the state vector, which greatly simplifies finding the solution. The only endogenous variable is the stock price. The model is offered here for comparability with all the other chapters in the book. We also use the model to demonstrate how to use different classes of approximating functions and different driving processes u_t in the Fortran program.

- **Example II** (Simple Stochastic Growth Model)

  Consider the simple growth model where an agent maximizes the same utility as in the previous example, subject to

  $$c_t + k_t - (1 - d)k_{t-1} = k_{t-1}^{\alpha} \theta_t \tag{4}$$

  k_{-1} given, where c_t denotes consumption, k_t is the capital stock and θ_t is an exogenous stochastic productivity shock, Markov of order one.

The first order condition for optimality is

$$c_t^\gamma = \delta \ E_t[\ c_{t+1}^\gamma \ (k_t^{\alpha-1}\alpha\theta_{t+1} + 1 - d)] \tag{5}$$

To map this model into the above framework, set $z_t = (c_t, k_t, \theta_t)$, $u_t = \theta_t$, and $x_t = (k_{t-1}, \theta_t)$. The function g is given by the resource constraint (4) and the Euler equation (5). The function $\phi(z', z)$ is given by $c^\gamma(k^{\alpha-1}\alpha\theta + 1 - d)$. Standard results from dynamic programming guarantee that the conditional expectation is a time invariant function of the state variables (k_{t-1}, θ_t).

- **Example III** (Stochastic Growth Model with Flexible Labor Supply and Proportional Taxes)

  Assume now that a representative consumer utility is given by

$$E_0 \sum_{t=0}^{\infty} \delta^t \left(\frac{c_t^{\gamma+1}}{\gamma+1} + b\frac{(1-l_t)^{\gamma_l+1}}{\gamma_l+1} \right)$$

where l_t represents hours worked.

A representative firm has a production function $k_{t-1}^\alpha l_t^{\alpha_l}\theta_t$. The firm maximizes profits. Both the firm and the consumers are price takers. The consumer owns the capital stock, receives income from renting the capital stock to the firm at the rental price r_t and from selling his labor to the firm at the wage w_t. The consumer pays a tax proportional to its labor income at the fixed tax rate τ^l and at a rate τ^k for its capital income.

If we assume constant returns to scale and set $\alpha_l = 1 - \alpha$, the first order conditions for the consumer are

$$c_t^\gamma = \delta \ E_t[\ c_{t+1}^\gamma(k_t^{\alpha-1}\alpha l_{t+1}^{1-\alpha}\theta_{t+1}(1-\tau^k) + 1 - d)] \tag{6}$$

$$c_t^\gamma = \frac{b(1-l_t)^{\gamma_l}}{\theta_t(1-\alpha)k_{t-1}^\alpha \, l_t^{-\alpha}(1-\tau^l)} \tag{7}$$

and the feasibility conditions are given by

$$c_t + k_t - (1-d)k_{t-1} = k_{t-1}^\alpha l_t^{1-\alpha}\theta_t \tag{8}$$

In the Euler equations we have introduced the fact that w and r are given by the marginal product of labor and capital.

Relative to Example II, we have introduced taxes and endogenous labor supply; obviously, Example II is a special case of the current example if we set $b = 0$, $\alpha_l = 0$ and $\tau^l = \tau^k = 0$.[5] The interest of this example is two-fold: first, it shows how to solve models with more decision variables than endogenous state variables (decision variables are c_t, l_t and endogenous state variable is k_t).

Second, the solution to this model is suboptimal (due to the distortionary tax) and it cannot be easily found with standard dynamic programming tools.

To map this model into the above framework, set $z_t = (c_t, l_t, k_t, \theta_t)$and $u_t = \theta_t$. The function g is given by the resource constraint (8) and the first order conditions (6), (7). The function $\phi(z', z) \equiv c^\gamma \, (k^{\alpha-1}\alpha l^{1-\alpha}(1-\tau^k)\theta + 1 - d)$. Results from Coleman (1991) guarantee that a time-invariant solution exists with $x_t = (k_{t-1}, \theta_t)$.

- **Example IV** (Simple Growth Model with non-negative Gross Investment)

  This example shows how inequality constraints can be easily handled by PEA, and it demonstrates the use of variables corresponding to Lagrange multipliers. Suppose we add to Example II a non-negativity constraint on gross investment:

$$k_t - (1-d)k_{t-1} \geq 0 \tag{9}$$

  With this restriction, the first order condition (5) is replaced by the Kuhn-Tucker conditions

$$c_t^\gamma - \lambda_t - \delta \; E_t \left[\; c_{t+1}^\gamma \; \left(k_t^{\alpha-1}\alpha\theta_{t+1} + (1-d)(1-\lambda_{t+1}) \right) \right] = 0 \tag{10}$$

$$(k_t - (1-d)k_{t-1})\lambda_t = 0 \tag{11}$$

$$\lambda_t \geq 0, i_t \geq 0, \tag{12}$$

  where λ_t denotes the Lagrange multiplier associated with constraint (9).

[5]For now, we ignore the budget constraint of the government; this is consistent with assuming a sufficiently low level of initial debt or with assuming that any surplus is returned as a lump sum transfer to the agents.

To map this model into framework (1) we set $z_t = (c_t, k_t, \lambda_t, \theta_t)$, $u_t = \theta_t$. The system g is given by (4), and the Kuhn-Tucker conditions (9)-(12). Note that, in this case, $\phi(z', z) \equiv c^{\gamma}(k^{\alpha-1}\alpha\theta + (1-d)(1-\lambda))$. Again, using standard dynamic programming, the model can be shown to be recursive with the same x as in Example II.

This example illustrates how to treat stochastically binding inequality constraints. Notice that the Lagrange multiplier is treated just like an additional variable by including it in z; we therefore have to solve for this multiplier jointly with all the other variables.

- **Example V** (Habits in Consumption)

  Modify Example II in order to introduce a per-period utility that depends on present and past consumption. More specifically, assume that utility at time t is given by $u(c_t, c_{t-1}) = \frac{(c_t+\eta c_{t-1})^{1+\gamma}}{1+\gamma}$. The Euler equation is

$$\begin{aligned}(c_t + \eta c_{t-1})^{\gamma} = \delta E_t\big[& -\eta(c_{t+1} + \eta c_t)^{\gamma} \\ & + \big((c_{t+1} + \eta c_t)^{\gamma} + \delta\eta(c_{t+2} + \eta c_{t+1})^{\gamma}\big)\left(\theta_{t+1}\alpha k_t^{\alpha-1} + 1 - d\right)\big]\end{aligned} \tag{13}$$

  To map this model into framework (1) we take z and u as in Example II. The system g is given by (13) and (4) and ϕ is given by

$$\begin{aligned}\phi(z_{t+2}, z_{t+1}, z_t) = & -\eta(c_{t+1} + \eta c_t)^{\gamma} + \\ & \big((c_{t+1} + \eta c_t)^{\gamma} + \delta\eta(c_{t+2} + \eta c_{t+1})^{\gamma}\big)\left(\theta_{t+1}\alpha k_t^{\alpha-1} + 1 - d\right)\end{aligned} \tag{14}$$

  Notice that, in this case, ϕ depends on two leads of the endogenous variables[6]. Again, the model can be shown to be recursive with standard dynamic programming techniques but where we add one additional state variable to obtain the vector $x_t = (k_{t-1}, c_{t-1}, \theta_t)$ in order to have (2).

  The example is used to illustrate three issues: first, the effects on computational costs of having an additional continuous state variable (namely c_{t-1}); second, the effects of having two shocks (namely $\theta_{t+1}, \theta_{t+2}$)

[6] Having two leads in ϕ does not match exactly the framework of (1), but it is a trivial extension.

inside the conditional expectation on the computational costs of the algorithm.

We will also solve the model by rewriting the above Euler equation separating out two expectations. This is obviously an inefficient approximation since it will involve computing two functions, but it will be done for pedagogical reasons since it demonstrates the issues related with having more than one conditional expectation to solve for.

- **Example VI** (Two Capital Goods)

  This extends Example II to include two capital goods. More precisely, assume that the production function is Cobb-Douglas with two capital goods, so that the feasibility conditions are:

$$c_t + i_{1,t} + i_{2,t} = \theta_t k_{1,t-1}^{\alpha_1} k_{2,t-1}^{\alpha_2} \tag{15}$$

$$k_{j,t} = k_{j,t-1}(1-d_j) + i_{j,t}\ , \qquad k_{j,-1} \text{ given,} \tag{16}$$

  for $j = 1, 2$. The Euler equations are:

$$c_t^{\gamma} = \delta E_t[c_{t+1}^{\gamma}\ (\theta_{t+1}\alpha_1 k_{1,t}^{\alpha_1 - 1} k_{2,t}^{\alpha_2} + 1 - d_1)] \tag{17}$$

$$c_t^{\gamma} = \delta E_t[c_{t+1}^{\gamma}\ (\theta_{t+1}\alpha_2 k_{2,t}^{\alpha_2 - 1} k_{1,t}^{\alpha_1} + 1 - d_2)]. \tag{18}$$

  Relative to Example II, we have to include the second capital stock both in z and in x and, by now, it is clear how we could set up the functions g and ϕ given the expressions above.

  This example will serve to show how to deal with several potential pitfalls with PEA. First of all, this example demonstrates the issue that, sometimes, the natural way to write system g does not allow the researcher to invert the system; this issue is quite delicate, as it does not arise in other algorithms. Also, this example will serve to discuss some problems that come up because of the addition of another state variable; we will see that, unless we are careful, the second state variable tends to generate non-stationary simulations, and the algorithm breaks down. Finally, the simulations of this model will be of interest in themselves.

  Given the assumptions of the model, if the depreciation rate is the same for the two types of capital goods the model can be easily transformed into a model with just one capital good, since the ratio of the two capital stocks will be constant over time[7]. We could therefore solve the case $d_1 = d_2$ directly from the solution of Example II. In order to avoid this trivial solution, we will consider different depreciation rates. We will, however, study the case of equal depreciation rates by solving it ignoring this theoretical knowledge. This will serve to demonstrate some problems that arise with PEA when the state variables are highly correlated and we will be able to perform an elementary consistency check on the functioning of the algorithm, just checking if it reproduces this known property of the exact solution.

A key assumption that we will place on the system g is that it is invertible with respect to its second argument. Therefore, we will insist on writing the system of difference equations in such a way that, given a function $\mathcal{E}$ and past values of the series, we can then generate the current value for z by solving the system g. In this chapter, the term 'solving the model' refers to generating a stochastic process $\{z_t, u_t\}$ on the computer that satisfies, approximately, equations (1) and (2). Under the above invertibility requirement on g, we can back out the series $\{z_t\}$ from a candidate $\mathcal{E}$ and simulated values of u_t, so we will use the term 'solving the model' interchangeably with the term 'finding $\mathcal{E}$'.[8, 9]

2.2 Definition and Calculation of an Approximate PEA Solution

The general idea of PEA is to substitute the conditional expectations in (1) by flexible functions that depend on the state variables and some coefficients. Then (1) is used to generate simulations for z consistent with the parameterized expectations; with these simulations we can iterate on the parameterized expectations until they are consistent with the solutions they generate.

By a 'flexible functional form' we mean a function $\psi(\beta;x)$ for some coefficients $\beta \in R^{\nu}$ such that, as we increase the number of coefficients to let

[7]From the Euler equations it is clear that the ratio $k_{1,t}/k_{2,t}$ is determined by $\frac{f_{k_1}}{f_{k_2}} = 1$.

[8]As is always the case in numerical analysis, these 'solutions' are only approximate, so we often do not use the term 'approximate solution'.

[9]The usual restrictions apply to all the parameters in these examples, namely: $\gamma, \gamma_l < 0$; $0 < d, d_1, d_2, \alpha, \alpha_1, \alpha_2, \delta < 1$; and $b > 0$.

$\nu \to \infty$, we can approximate any function $f(x)$ arbitrarily well (e.g. polynomials, splines, etc.). By 'simulations for z consistent with the parameterized expectations' we mean a process $\{z_t(\beta), u_t\}_{t=0}^{\infty}$ that, for all t, solves the system

$$g\big(\psi(\beta; x_t(\beta)), z_t(\beta), z_{t-1}(\beta), u_t\big) = 0 \tag{19}$$

Obviously, the function $\psi(\beta; \cdot)$ could be quite different from $\mathcal{E}(\cdot)$ for an arbitrary β. We will ensure that $\psi(\beta; \cdot)$ is a good approximation by fixing an order of approximation ν (e.g., ν-degree polynomials, or splines with ν intervals ...), and choosing β in such a way that, within the approximating class for a given ν, the function is as close as possible to the conditional expectation function $\mathcal{E}$. For this purpose, we have to discuss how to find a good approximating β.[10]

We now describe a algorithm to solve for such a β. Here, we describe the structure of the algorithm. Many details will be discussed later and in the context of the examples.

- Step 1: Write the system g in (1) so that it is invertible with respect to its second argument; in other words, the system g has to be such that, given the first, third and fourth arguments the value for z_t can be uniquely determined from (1). Find a set of state variables x that satisfies (2). Replace the true conditional expectation by the parameterized function $\psi(\beta; \cdot)$ to obtain (19). Fix the initial conditions u_0 and z_0. Draw a series $\{u_t\}_{t=0}^{T}$ from a random number generator that obeys the distribution of u in the model, with T sufficiently large.

- Step 2: For a given $\beta \in R^{\nu}$ such that $\{z_t(\beta), u_t\}$ has a stationary distribution, recursively calculate $\{z_t(\beta), u_t\}_{t=0}^{T}$ using (19) and the realization for u drawn in the previous step.

- Step 3: Find $G(\beta)$ that satisfies

$$G(\beta) = arg \min_{\xi \in R^{\nu}} \frac{1}{T} \sum_{t=0}^{T} \| \phi(z_{t+1}(\beta), z_t(\beta)) - \psi(\xi; x_t(\beta)) \|^2 \tag{20}$$

  This minimization is easy to perform by computing a non-linear least squares regression with the sample $\{z_t(\beta), u_t\}_{t=0}^{T}$, taking $\phi\left(z_{t+1}(\beta), z_t(\beta)\right)$

[10] We do not discuss here the issues related to convergence with arbitrary accuracy as $\nu \to \infty$. See Marcet and Marshall (1994) for a convergence proof.

as the dependent variable, $\psi(\xi;\cdot)$ as the explanatory function, and ξ as the parameter vector to be estimated.

- Step 4: Find the fixed point

$$\beta_f = G(\beta_f) \tag{21}$$

Then, $\{z_t(\beta_f), u_t\}_{t=0}^{T}$ is our approximate solution. Or, equivalently, $\psi(\beta_f;\cdot)$ is our approximation to the conditional expectation $\mathcal{E}$ or, equivalently, the inverse of g in (19) with respect to its second argument (setting $\beta = \beta_f$) is the approximate equilibrium law of motion of the model. Clearly, in order to find the fixed point β_f one needs to iterate on Steps 2 to 4 (there is no need to redraw the u's or to change initial conditions) until a fixed point is found.

The non-linear regression of Step 3 is meant to approximate the best predictor in the steady state distribution of the model. This is why, in Step 2, the values of β are restricted to generate a stationary distribution for $\{z_t(\beta), u_t\}_{t=0}^{\infty}$; otherwise, there is no reason why $G(\beta)$ should deliver a good predictor and the solution might be explosive and give numerical errors.

We now discuss how to set up each example to apply the above steps. The issues related to algorithms for finding the non-linear regressions of Step 3 and the fixed point of Step 4 will be discussed later. Also, in the discussion of this section we use only a first degree approximation. In section 5 we discuss some issues related with increasing the degree of approximation.

- **Example I** (Lucas Asset Pricing Model)

  Consider now the simplest case where there is only one asset $J = 1$. In Step 1 we have to choose the functional form of ψ. Since the variable inside the expectation is positive, we use a polynomial of order one:

$$\psi(\beta; x_t) = \beta_1 + \beta_2 \; d_t \tag{22}$$

  In order to apply Step 2, notice that (3) delivers the following equation for the stock price:

$$p_t(\beta) = \delta \; \psi(\beta; d_t)$$

  which is used to generate simulations for $p_t(\beta)$.

  In order to apply Step 3, we just need to construct the 'dependent variable'

$$Y_t(\beta) \equiv \frac{c_{t+1}^{\gamma}}{c_t^{\gamma}}(p_{t+1}(\beta) + d_{t+1})$$

and run a regression on the equation:

$$Y_t(\beta) = \xi_1 + \xi_2 d_t + \eta_t$$

where η_t is the regression error. The vector of parameters ξ that minimizes the sum of squared residuals (in this case, simply the OLS estimator) is, precisely, $G(\beta)$.

Notice that, because we use a regular polynomial in ψ, it is possible for the approximate solution to deliver negative prices. This will not cause any problem in the implementation of the algorithm. Furthermore, under the assumption of no default, the correct solution might actually involve negative prices. This is in contrast with the other examples, where non-negativity of the expectation needs to be imposed.

The cases for $J = 2$ and positive dividends will be discussed in greater detail in Section 4.

- **Example II** (Simple Stochastic Growth Model)

  For the functional form of ψ, using a regular polynomial as in the previous example might be problematic because, eventually, it might generate a negative value for ψ and, since this number is raised to a real power in Step 2 (see equation (24) below), a numerical error would ensue. Furthermore, since the variable inside the expectation is positive, we know that the true $\mathcal{E}$ takes only positive values so, by imposing this feature on our approximating ψ, we are likely to get a better fit with fewer parameters.

  For these reasons, we use an exponentiated polynomial of order one:

$$\psi(\beta; x_t) = \beta_1 \exp(\beta_2 \ \log k_{t-1} + \beta_3 \ \log \theta_t) \tag{23}$$

  which is guaranteed to be positive and so generate a positive solution for consumption. Increasing the degree of the polynomial inside exp we can approximate the conditional expectation with arbitrary accuracy.

  In order to apply Step 2, notice that system (19) delivers the following equation for consumption:

$$c_t(\beta) = (\delta \ \psi(\beta; k_{t-1}(\beta), \theta_t))^{\frac{1}{\gamma}} \tag{24}$$

and capital can be found from

$$k_t(\beta) = k_{t-1}(\beta)^\alpha \theta_t - c_t(\beta) + (1-d)k_{t-1}(\beta)$$

These equations generate simulations for $(c_t(\beta), k_t(\beta))$.

For Step 3, we construct the 'dependent variable'

$$Y_t(\beta) \equiv \ c_{t+1}(\beta)^\gamma \ (k_t(\beta)^{\alpha-1}\alpha\theta_{t+1} + 1 - d)$$

and run a non-linear regression on the equation:

$$Y_t(\beta) = \xi_1 \exp(\xi_2 \log k_{t-1}(\beta) + \xi_3 \log \theta_t) + \eta_t$$

where η_t is the regression error. The vector of parameters ξ that minimizes the sum of squared residuals in this regression is, precisely, $G(\beta)$.

Notice that it would *not* be correct to take logs on both sides of the equation and run a *linear* regression of $\log Y_t(\beta)$ on $(\log k_{t-1}(\beta), \log \theta_t)$; this would only give the correct answer if the error were multiplicative (which is the case under log-normality), but it is not the case in general for conditional expectation errors. This is a reflection of the fact that this model is, in fact, non-linear, and linear functions are at best good approximations.

- **Example III** (Stochastic Growth Model with Flexible Labor Supply and Proportional Taxes)

  We choose ψ as in the previous example. As for Step 2, the corresponding system (19) delivers

$$\delta \ \psi(\beta; k_{t-1}(\beta), \theta_t) = \frac{b(1 - l_t(\beta))^{\gamma_l} l_t(\beta)^\alpha}{\theta_t \ (1-\alpha) \ k_{t-1}(\beta)^\alpha \ (1-\tau^l)} \qquad (25)$$

  which, given the state variables, is a non-linear equation in $l_t(\beta)$. This non-linear equation has to be solved numerically for each t and β. Consumption is found as in the previous example. Finally, from the resource constraint (corresponding to market clearing) we obtain $k_t(\beta)$.

  Clearly, the non-linear regression to be run in Step 3 is similar to Example II, the only difference is that labor and taxes enter in the calculation of $Y_t(\beta)$.

- **Example IV** (Simple Growth Model with Lower Bounds on Investment)

  We show how to use the Kuhn-Tucker conditions to find $\{c_t(\beta), k_t(\beta)\}$ while imposing inequalities (9) and (12. The system (19) is given by:

$$c_t(\beta)^{\gamma} - \lambda_t(\beta) = \delta \; \psi(\beta; k_{t-1}(\beta), \theta_t) \tag{26}$$

$$\big((c_t(\beta)^{\gamma} - \lambda_t(\beta) - \delta \; \psi(\beta; k_{t-1}(\beta), \theta_t)\big)\big(k_t(\beta) - (1-d)k_{t-1}(\beta)\big) = 0 \tag{27}$$

$$\lambda_t(\beta) \geq 0 \quad \text{and} \quad k_t(\beta) - (1-d)k_{t-1}(\beta) \geq 0 \tag{28}$$

  We can then proceed as follows: for each t,

  (a) compute $(c_t(\beta), k_t(\beta))$ from (26) under the conjecture that $\lambda_t(\beta) = 0$. In this case, the mechanics for computing consumption and capital are exactly like in Example II. If investment is positive, go to $t+1$. If investment is negative use the next step:

  (b) set $k_t(\beta) = (1-d)k_{t-1}(\beta)$, find $c_t(\beta)$ from the feasibility constraint, and then compute $\lambda_t(\beta)$ from (26).

  In applying this procedure, we have to make sure that the Kuhn-Tucker conditions are satisfied. Notice that, for a fixed value of $\psi(\beta; k_{t-1}(\beta), \theta_t)$, the left side of (28) is increasing in $k_t(\beta)$ in step (a). Hence, if step (a) delivers a negative investment, consumption will be lower when we go to step (b), so $c_t(\beta)^{\gamma}$ will be higher (relative to (a)), and (26) delivers a positive λ in Step (b), so (28) is satisfied. Clearly, (27) is satisfied, since either one or the other large parenthesis is automatically set to zero.

  Notice that the Lagrange multiplier is treated just like an additional variable in our problem. Its value is derived from solving a system of equalities and inequalities corresponding to the g system. In the present model the Lagrange multiplier affecting investment appears also inside the expectation, and so its realized values are needed to compute the ϕ expression; this is how the possibility of having future binding constraints affects today's investment.

  It must be noticed that there are some ways of writing the Kuhn-Tucker condition that would be inappropriate. For example, by multiplying both sides of (10) by $c_t^{-2\gamma}$, the Kuhn-Tucker condition can be correctly expressed as,

$$c_t^{-\gamma} - \lambda_t c_t^{-2\gamma} = \qquad (29)$$
$$\delta \; E_t \left[c_{t+1}^{\gamma} \; c_t^{-2\gamma} \; (k_t^{\alpha-1} \alpha \; \theta_{t+1} + 1 - d) - (1-d)(1-\lambda_{t+1}) \right]$$

However, (29) would be an inappropriate choice for g if PEA is used as a solution algorithm, because the function $c^{-\gamma}$ is decreasing in today's k. Then, step (b) would deliver a negative value for the multiplier λ. In terms of the step-by-step description of the algorithm, (29) violates step 1, since it implies a representation for g that is not invertible with respect to its second element.

To summarize, one can simply simulate by putting the Lagrange multiplier to zero, checking the inequality constraint, if it is not satisfied, impose the inequality constraint and obtain the multiplier from the Euler equation. Due to our previous derivations, the researcher can be confident that the relevant inequalities for the Kuhn & Tucker conditions are satisfied, but this should be checked for a different example, in order to avoid situations such as the one discussed with (29).

Clearly, the non-linear regression to be run is as in Example II, except that now λ enters in the calculation of $Y_t(\beta)$.

- **Example V** (Habits in Consumption)

We choose the same kind of ψ as in Example II, but now we need to include an additional state variable:

$$\psi(\beta; x_t(\beta)) = \beta_1 \exp(\beta_2 \; \log k_{t-1}(\beta) + \beta_3 \; \log \theta_t + \beta_4 \; \log c_{t-1}(\beta)) \quad (30)$$

In order to apply Step 2 we obtain consumption from

$$c_t(\beta) = (\delta \; \psi(\beta; k_{t-1}(\beta), \theta_t, c_{t-1}(\beta))^{\frac{1}{\gamma}} - \eta c_{t-1}(\beta);$$

then we solve for capital as in Example II. As in all previous examples, we run the regression of Step 3 with $Y_t(\beta)$ given by the right hand side of (14). Notice that inside the conditional expectation of the Euler equation (13) we now have random variables affected by the two shocks, θ_{t+1} and θ_{t+2}. However, the number of computations required by the algorithm has barely increased because of this fact. This is in contrast for algorithms that use explicit integration of the conditional expectation (for example, algorithms that use quadrature to evaluate the conditional expectation at each point), adding stochastic shocks with continuous distributions leads to a large increase in the cost of computing the integral. Of course, the number of computations will certainly increase because of the fact that now we have to solve for a fixed point of a four-coefficient vector, but this is common to all algorithms that approach this problem.

We use this example to discuss the issues that arise when there is more than one conditional expectation in the model. Notice that (13) could have been written as

$$\begin{aligned}(c_t + \eta c_{t-1})^\gamma = \delta E_t\Big[\big((c_{t+1} + \eta c_t)^\gamma + \delta\eta(c_{t+2} + \eta c_{t+1})^\gamma\big) \\ \Big(\theta_{t+1}\alpha k_t^{\alpha-1} + 1 - d\Big)\Big] - \delta\eta E_t(c_{t+1} + \eta c_t)^\gamma\end{aligned} \qquad (31)$$

and we can approximate each of these expectations with two different functions $\psi(\beta^1; x)$ and $\psi(\beta^2; x)$ where each β^i has four elements, and ψ is as in (30). Clearly, this option is computationally inefficient, because we now have to compute a fixed point of eight coefficients $\beta \equiv (\beta^1, \beta^2) \in R^{4\times2}$, but it should give a correct approximation to the true solution[11]. Clearly, now the simulation generated in step 2 is given by

$$c_t(\beta) = \Big(\delta\ \psi(\beta^1; k_{t-1}(\beta), \theta_t, c_{t-1}(\beta)) - \delta\eta\ \psi(\beta^2; k_{t-1}(\beta), \theta_t, c_{t-1}(\beta))\Big)^{\frac{1}{\gamma}} - \eta c_{t-1}(\beta)$$

Notice that the variables depend on the whole vector β, but the coefficient that appears as an argument of each ψ depends on $i = 1, 2$.

Now, in order to apply Step 3, we need to run *two* separate non-linear regressions, with the same right hand side but with the explanatory variables:

$$\begin{aligned}Y_t^2(\beta) &= (c_{t+1}(\beta) + \eta c_t(\beta))^\gamma \\ Y_t^1(\beta) &= (Y_t^2(\beta) + \delta\eta Y_{t+1}^2(\beta))(\theta_{t+1}\alpha k_t(\beta)^{\alpha-1} + 1 - d)\end{aligned}$$

Now, letting $G^i(\beta)$ be the result of regression $i = 1, 2$, we define $G(\beta) \equiv (G^1(\beta), G^2(\beta))$, so that $G : R^{4\times2} \to R^{4\times2}$, and the fixed point of Step 4 involves eight coefficients.

[11]Assuming T and ν go to infinity.

- **Example VI** (Two Capital Goods)

  We have two endogenous state variables and, unavoidably, two parameterized expectations. Each parameterized expectation is of the form $\psi(\beta^i; k_{1,t-1}(\beta), k_{2,t-1}(\beta), \theta_t,)$, and the same issues discussed at the end of the previous example apply.

  Consider the Euler equations (17) and (18) with $\psi(\beta^i; \cdot)$ in the right hand side. We can obtain a value for $c_t(\beta)$ from any one of the two Euler equations, but we have no way to compute $k_{1,t}$ and $k_{2,t}$. This system is underdetermined for the capital stocks. But if we want both Euler equations to be satisfied, the system is overdetermined for consumption. This is the sense in which the system g in this example is, in principle, not invertible. We simply can not proceed to Step 2 in the way that the Euler equations have been initially written.

  In order to proceed, we have to rearrange the Euler equations in a way that allows us to compute the simulated series. There are many alternatives; we should choose one where the series are easily solved and, of course, we have to be careful that the new system still delivers a sufficient condition for an equilibrium. We could proceed by premultiplying both sides of the Euler equation by $k_{2,t}$ to obtain

$$c_t^{\gamma}\ k_{2,t} = \delta E_t[c_{t+1}^{\gamma}\ (\theta_{t+1}\ \alpha_2\ k_{2,t}^{\alpha_2}\ k_{1,t}^{\alpha_1} + (1-d_2)k_{2,t})]$$

  Clearly, since $k_{2,t}$ is never zero in equilibrium, this equation is satisfied if and only if the original Euler equation is satisfied, and it can replace sufficient condition for the optimum. We can determine $c_t(\beta)$ from the first Euler equation as in Example II, and then obtain $k_{2,t}$ from the second one by setting:

$$k_{2,t}(\beta) = \delta \frac{\psi(\beta^i; k_{1,t-1}(\beta), k_{2,t-1}(\beta), \theta_t)}{c_t(\beta)^{\gamma}} \qquad (32)$$

2.3 An Algorithm to Find β_f and the Use of Homotopy

Having discussed how to find G for a given β, we now turn to the problem of finding the fixed point of Step 4. Our comments here apply to all examples.

Although we could use a standard hill-climbing algorithm for solving nonlinear systems of equations this may not be the best alternative. First of all, hill climbing algorithms are relatively complicated, often unstable, and they require the knowledge of some numerical analysis; second, these algorithms work by calculating the gradient of G, which can become very expensive in models with many coefficients.

Applications of PEA often use the following algorithm, based on modified successive approximations:

$$\beta(\tau + 1) = (1 - \mu)\, \beta(\tau) + \mu\, G(\beta(\tau)) \tag{33}$$

for some $\mu > 0$ and some fixed initial condition $\beta(0)$; here, $\beta(\tau)$ represents the τ-th iteration of the algorithm. The algorithm is extremely easy to program and each iteration is calculated very fast. The most obvious point at which to stop the algorithm is when all the elements of the matrix $\beta(\tau)$ are sufficiently close to the corresponding element of $G(\beta(\tau))$.

There are two possible shortcomings of the above algorithm. First of all, (33) will typically need more iterations to converge than a hill-climbing algorithm[12]; because of that, it could happen that the gain in speed from avoiding computation of the gradient is lost in the additional iterations. Second, hill-climbing algorithms are (typically) locally stable, but the above algorithm could be locally unstable. Nevertheless, it can be shown that the above algorithm is locally unstable only in models where the rational expectations algorithm can*not* be learnt by agents in the economy. More precisely, consider a model of learning where agents, instead of having rational expectations, form their forecasts with $\psi(\beta_t; x_t)$, where β_t is the parameter that agents use on the basis of today's information. Assume that agents incorporate the new information that arrives at every period and they update their beliefs on β by using least squares estimators using all past observations. It can be shown that such a learning model converges to rational expectations (locally) if and only if iterations on (33) are locally stable for μ sufficiently small. Therefore, it is possible to use the above algorithm in all models where the rational expectations equilibrium can be justified as the limit of a learning model. As many economists believe that rational expectations equilibria are only interesting if they can be learnt by agents in the model, this works

[12]This is because hill-climbing algorithms converge in one step when they are in the neighborhood of the fixed point, and this is not the case in (33).

precisely for models that, in this sense, are of interest[13]. In any case, it turns out that least squares learning is locally stable in most applications, and the above algorithm can be used to compute the fixed point of Step 4 in most applications.

In order to use the above algorithm successfully, one needs good initial conditions; formally we need $\beta(0)$ to be not too far away from β_f. This is needed for two reasons; first, because iterations are more likely to get 'lost' and never converge if the initial condition is too far away from the limit[14]. Second, a good initial condition is needed because in Step 2 we need to consider β's that generate $\{z_t(\beta)\}$ that possess a stationary distribution. One could ensure stationarity by writing a general test for existence of a stationary and ergodic distribution[15], and then impose the restriction that only β's that satisfy such a condition are considered in the algorithm that looks for the fixed point of Step 4. This is a valid but rather cumbersome alternative since solving non-linear systems of equations subject to restrictions is a tricky business. An alternative solution that works in practice is to use the ideas of homotopy, which amounts to imposing good initial conditions in a systematic way.

The idea of homotopy is very simple - start with a version of the model that is easy to solve, then modify these parameters slowly to go to the desired solution. As long as the model goes from the known to the desired solution in a smooth way (formally, as long as the solutions are continuous with respect to the parameter that drives the model from the known to the desired solution), we are always solving models with good initial conditions. It is often possible to find such 'known' solutions and to build a bridge that goes to the desired solution. For example, den Haan and Marcet (1990) explained how to solve the simple growth model of Example II by starting the solution at the case of $d = 0$ and $\gamma = -1$, which happens to have an analytic solution. The 'bridge' to the desired model is obtained by slowly setting depreciation to the desired level of, say, d=.9 . In this case, d is the parameter that drives

[13]One caveat to this claim is that the model of learning we outline in the main text is not the only way that a learning model could be specified.

[14]This is also a common problem with hill-climbing algorithms and with most algorithms for solving non-linear systems, where only local stability is guaranteed.

[15]Some tests are provided by Duffie and Singleton (1993) and Domowitz and el Gamal (1993). The chapter by Novales et al. in this book has an explicit discussion of how to impose stationarity.

the model from the known solution to the desired solution. Also, one can go from Example II to Example III by starting at $b = 0$ and zero taxes. One can go from Example II to Example IV by adding to Example II the constraint $k_t - (1-d)k_{t-1} \geq K$; for a very low (and negative) value of K, the solution is as in Example II; by increasing K to zero we approach the solution to Example III. This exercise can even be performed when we change the functional forms of the fundamentals. For example, if we wanted to solve the simple growth model with the CARA instantaneous utility function -$e^{-\kappa c_t}$, we could consider solving the growth model with a utility function

$$-e^{-\kappa c_t}\zeta + \frac{c_t^{\gamma+1}}{\gamma+1}(1-\zeta).$$

Clearly, we know the solution for $\zeta = 0$ (since this is just Example II) and to get to the desired solution with CARA utility we need to increase until $\zeta = 1$.

In each case, we would start at the solution that is known and change the relevant parameter slightly. We would solve the model for this slightly different value of the parameter, using as initial condition the solution to the model under the previous parameter. For example, assume that we have already solved Example II and we would like to calculate the solution to Example IV. Let's say that gross investment in Example II is never less than -50. Let β_f^K be the solution for a given K; clearly, we know the solution for β_f^{-50} (this is just the solution of Example II), and we want to have the solution for β_f^0 (which is the solution to Example IV). To use a homotopy algorithm we could increase K by, for example, one unit at a time, using the previous fixed point as the initial condition in the algorithm to find the new fixed point. More precisely, letting G^K be the mapping of Step 3, for each K between -50 and 0 we would use the algorithm

$$\beta^K(\tau+1) = (1-\mu)\,\beta^K(\tau) + \mu\, G^K(\beta^K(\tau)) \tag{34}$$

with initial condition $\beta^K(0) = \beta_f^{K-1}$.

In this way, we only need local stability of the algorithm that solves for the fixed point, since we always have a good initial condition. Also, since the algorithm is never too far from the fixed point, all the β's we consider generate $\{z_t(\beta)\}$ with a stationary distribution, so that there is no need to impose the stationarity requirement on β explicitly.

In our personal experience, we have found that many researchers try to avoid the use of homotopy. Perhaps this is because homotopy slows down the algorithm and because it looks like a unsophisticated way to make an unstable algorithm converge[16]. In fact, using homotopy often ends up saving time to the researcher: researchers often spend hours in front of the computer (even though this activity is never discussed in papers) watching their hill-climbing algorithms drift all over the place or even diverge for no apparent reason. Then they spend even more hours trying to guess what initial condition could possibly work. By contrast, homotopy finds the good initial condition in a systematic way. So, even if it consumes more computing time, it is likely to save on researcher's time.

Here we have just discussed homotopy informally and in its most trivial form although there is a large mathematics literature on it. This literature discusses many tricks that can be used to speed up the algorithm. For example, along the homotopy it is not necessary to require high accuracy of the fixed point, since the intermediate models are only accessory. Also, this literature studies some pitfalls of the procedure. For example, if the model has a region of parameters where the solution is undefined, it is likely that the homotopy may cross it sooner or later, so one has to be careful about the existence and the continuity of the homotopy path. For example, we will show in section 4 that setting up a *continuous* homotopy path for Example III is possible, but not trivial. We do not discuss this literature in detail for lack of space, but we hope to arise some interest. In the program we have a built in homotopy (only the most simple minded homotopy), and in the examples below we show how this can be applied.

3 General Description of the Program

The fortran program open can be used to compute the steady state dynamics of non-linear stochastic models fitting into the general framework discussed above. In the following we will discuss issues related to the practical implementation of the algorithm, making reference to this program, and describing how to apply it to our examples.

[16]A very good expression of this sentiment was provided by a referee to a previous PEA paper, who referred to the homotopy discussion as 'making a big deal out of finding good initial conditions, which is something we all know how to do'.

The source code `open.for` (and its variants used in some of the examples) includes all the subroutines needed to perform the algorithm (matrix inversion, random numbers generator, etc.). Some parts of the program have been written on separate short files, called `psis.for`, `der.for`, `ginvert.for` [17]. These are the parts of the programs that are specific to the economic model at hand and to the functional form used to approximate the expectation terms. In principle, the user will need only to write these program files, and to fill the files containing the parameters (`alpha.dat`, `par.dat` and `pini.dat`) in order to use the main program `open.for` for different economic applications. On the web there is a directory for each example, each containing the file `open.for` (or some variant of it) and all the auxiliary and parameter files needed to solve that example.

In this section we refer to variables in the program in italics; most variables in the program are named as in the chapter. For example, *beta* is the coded version of β, *phi* is the coded version of ϕ, and so on. The index t always refers to time-period.

Let us illustrate the functioning of the program following steps 1 to 4. First of all, the program reads the control parameters for the algorithm (in `par.dat`), the parameters of the model (in `alpha.dat`) and the initial parameters *beta* for the functions *psi* (in `pini.dat`). The program generates a random sequence of shocks $theta(l,t)$ (l is the index for the exogenous shock) that will be used throughout the computations[18]. All the model parameters are stored in a vector *alpha*, the parameters controlling the disturbance process are stored in the arrays *am*, *cm*, and *dm* and the series are stored in the vector Z.

Then, the iterative process to compute the *beta* is started.

- Subroutine `simul` (Step 2): The program simulates a time series $Z(i,t)$.

  a) In order to compute $psi(j,t)$ it uses the initial values for the state variables set in `par.dat` and the parameterized expressions defined in `psis.for` with the initial *beta* set in `pini.dat`.

[17]Common fortran compilers will automatically include the text of these auxiliary files into the main program, during the compiling phase.

[18]The program includes a subroutine that generates i.i.d. standard normal disturbances, and a subroutine that can be adapted to generate processes with different autocovariance structure. The parameters controlling the process for the shocks are also set in `alpha.dat`.

b) With these data available the program can solve, for every time period, the system corresponding to equation (19), in order to obtain the state variables in the current period $Z(i,t)$. The algebraic steps corresponding to the inversion of (19) are defined in `ginvert.for` by the user, according to the model at hand. This part of the program makes use of the parameters of the model set in `alpha.dat`. If a numerical approximation is needed for this purpose (as in Example III) a subroutine for this task can be easily appended to the main program `open.for`.

c) The program computes the realized values of the expressions that appear inside the expectations —that is, the $phi(t - nfor, j)$ in the same loop that computes the values for the $Z(i,t)$ variables. $nfor$ is the maximum number of leads appearing inside the expectations, and at period t is the first time at which we are able to compute all the expressions, whose value we were trying to predict $t - nfor$ periods before using *psi*. If $nfor$ is greater than 1 you can start to compute *phi* only after $nfor - 1$ periods. This is not a problem in the examples discussed since we are computing the steady state parameters. In this case the program drops the first observations (150 in the programs described) in the nonlinear regression and then it is set to compute *phi* just from the 150th period on.

- Subroutine `Gbeta` (Step 3): The parameters *Gbeta* are estimated with a nonlinear regression of $phi(t,j)$ on the state variables appearing in *psi*. The nonlinear regression is performed following a common iterative procedure described, for example, in Pindyck and Rubinfeld (1981) in which, at every step, the residuals and the derivatives of the ψ function, using the parameters from the previous step, are used to perform a linear regression that gives the coefficients for next step. Here the program needs to use both the form of *psi* defined by the user in `psis.for` and the derivatives of *psi* defined in `der.for`. After convergence is achieved for *Gbeta* we pass to step 4. The convergence criterion for the non-linear regression *acclnr* and the maximum number of iterations *maxitnl* are both set in `par.dat`.

- Convergence check and updating (Step 4): The parameters *Gbeta* of the regression in step 3 are compared with the *beta* parameters, which we used to performed step 2. If the distance between them is smaller than *acc* the algorithm stops. Clearly, this distance can be evaluated in many different ways. In most of the examples discussed below it is computed simply as a sum of absolute differences, while in the program we use to solve Example VI it is evaluated computing the distance (sum of squared differences) between the simulated series obtained using *beta* and the simulated series using *Gbeta* (see next section for the latter convergence criterion).

The parameters are updated according to: $beta' = mu \cdot Gbeta + (1 - mu) \cdot beta$. The value for mu is set in `par.dat`. In general an mu closer to 1 makes the algorithm faster but is more likely to result in non-convergence. If the algorithm converges the updated values for the beta are stored in '`pini.dat`' and the old parameters are discarded. In this way, the next time that the program is used, it automatically uses the fixed point of the last run (keep this is mind if you want to store a fixed point for future use, in that case, you need to save the contents of `pini.dat` in a different file).

After the iterations on *beta* converge (hopefully to β_f), a simulation of the series $Z(i,t)$ is stored in the output file `openout.dat`, along with descriptive statistics of the series. It is very easy to adapt this part of the program so as to obtain additional aggregate information on the series. Moreover, along the program (in particular in `ginvert.for`) the user can compute whichever auxiliary variable is needed, and store it as an additional series ($Z(t,i)$), obtaining other useful information on the model.

It is advisable to set the initial conditions of the endogenous state variables at levels that are often visited in the stationary distribution. This is because, with the algorithm described in section 2, there is no guarantee that our approximation is any good outside the support of the stationary distribution. The output file may also be useful to choose an appropriate initial value for the state variable (e.g. k_{-1} in example II) when the parameters have been modified and hence the stationary distribution of the state variables has changed.

Notice the convention that the time subscript for $phi(t,j)$ refers to the period in which the expected value of $phi(t,j)$ is needed to solve the system (19) (and the approximated value $psi(t,j)$ is used instead). In step 2 of the program this notation gives the natural result that $phi(t,j)$ is the dependent variable to be explained by the state variables at time t in the parameterized expression *psi*, but $phi(t,j)$ depends on random variables whose value is realized in the future. This is why, in `ginvert.for`, we compute the past value $phi(t-nfor,j)$ at period t. As noticed above it is impossible to compute the value of $phi(t,j)$ unless $nfor$ periods later when we have all the needed realized values.[19] $nfor$ is another parameter to be set in `par.dat`.

Parameter files. The parameters controlling the algorithm are listed in the file `par.dat`. In particular in this file the user will set *neq*, the number of parameterized expressions corresponding to so many *psi* functions, and $np(j)$ for $j = 1, \dots, neq$, the number of parameters in expression j (i.e. the number of *beta* coefficients in each *psi*), and *nz* the number of Z variables used, and the control parameters for the various iterative procedures performed by the program. Another file `alpha.dat` contains the parameters of the model and also parameters controlling the automatic homotopy steps.[20]

Program files. Here we summarize the content of program files that are model dependent and have to be written by the user. The parameterized forms (*psi*) have to be written in the file `psis.for` and the derivatives of *psi* in file `der.for`. In this version there is not a predefined functional form, the user has to provide it in `psis.for` and to provide analytical derivatives for *psi* in `der.for`. Clearly, as far as you are satisfied with an exponentiated polynomial, you can simply adjust the expressions in the `psis.for` and `der.for` used in the examples II to VI.

In `ginvert.for` there are the algebraic or numerical steps to solve the equation (19) and the expressions to compute the ex-post values for *phi*[21]. Finally, to avoid repeated computation of logarithms we have introduced the auxiliary variables $Zlog(i,t)$ that may be used (but need not to) along the program to store the values of the log of $Z(i,t)$. Usually the logarithms are computed in `ginvert.for` immediately after the computation of the $Z(i,t)$. The general form of the parameter files is described in the program, along with a list of all the variable names.

[19]This is also the reason why we have to drop the last $nfor$ observations when performing the regression, simply because the last values of *phi* are not available.

[20]See the detailed description of the files in the program.

[21]Notice also that in psis.for and `der.for` the parameters are named $b(h,j)$ (h-th parameter of the j-th expression), so that the program can use the same definition –substituting $beta(h,j)$ or $Gbeta(h,j)$ for $b(h,j)$– in different steps of the algorithm.

Built-in Homotopy. As discussed above, in order to ensure convergence of the PEA it is often a good idea to change the parameters in small steps along a homotopy path, from a model already solved to the target model. Since this procedure is indispensable in many models, the program has been set up to perform it automatically. The built-in homotopy steps are activated setting 1 in the third line of `alpha.dat` (after the list of the *alpha* parameters), and adding a line with the following control parameters: the identifier i of the parameter $alpha(i)$ to be changed, the target value for $alpha(i)$ and the size of the step of adjustment. As initial value the program takes the $alpha(i)$ in the list at the beginning of `alpha`, notice that at the end of the program the file `alpha.dat` is updated automatically. In this way, once the program has finished the computations, the files `alpha.dat` and `pini.dat` are always in line. The file `alpha.dat` is not changed if the homotopy is not activated. Also, if the program finds the same target value as $alpha(i)$ the homotopy steps are omitted. If you do not want to activate the homotopy procedure just leave a zero in the third line of `alpha.dat`.

4 Solutions to Examples

In this section we illustrate by means of Examples I to VI the use of the general program `open`. We also discuss some of the results that we can observe in the computations. Along with the Examples, we will describe some common problems and will give some practical suggestions for the use and the extension of the program. In the programs available on-line the modifications to the basic model `open.for` are highlighted with two rows of comments ('C') at the beginning and at the end of each insertion.

4.1 Example I. Lucas Asset Pricing

See files in the directory `~\ex1luc`. We begin with the case of one asset, we compute it first with a regular polynomial and later with an exponentiated polynomial. Then we compute the case for two stocks ($J = 2$). The program file `open.for`, allows for a general vector AR(1) structure of the shocks u_t (each component of the shock is labeled $theta(t, j)$), and can be easily modified to use a different distribution for this process. The current example is the only one in which we will consider more than one series of random shocks;

all the others will use a one-dimensional log AR(1) exogenous process. We will take advantage of this example to show that the algorithm deals very easily also with fairly rich stochastic structures.

The labeling of model variables is as follows.

i	1	2
$Z(i,t)$	$p_{1,t}$	$p_{2,t}$
j	1	2
$alpha(j)$	δ	γ

The shocks *theta* follow a normal AR(1) process

$$thetalog(t) = D + A \cdot thetalog(t-1) + C \cdot q(t) \tag{35}$$

where $q(t)$ is a vector of dimension *nthe* of standard normal independent shocks, A and C are matrices (with components $am(j,k)$, $cm(j,k)$) and D is a vector determining the means, covariances and autocovariances of the *theta*'s. The values of these parameters are also set in the file `alpha.dat`.

Consider first the case of one asset, serially independent dividends and log utility. In order to consider the case where we allow for negative values inside the conditional expectation, we set $D = 1, A = 0$ and $C = .4$, and we take as the dividends $d_t = thetalog(t)$. In this case we obtain an analytical solution for the asset price corresponding to

$$p_{j,t} = \frac{\delta}{1-\delta} d_{j,t} \tag{36}$$

Clearly, for this case, a first degree polynomial for ψ with $beta(1,1) = 0$ and $beta(2,1) = 20$ gives the precise solution solution. We introduce a first degree polynomial in the files `psis.for` and the derivatives of *psi* defined in `der.for` Then, we can move the autocorrelation parameter to some other level, say $A = .8$ and $\gamma = -2$ to find a solution that is not analytic.

Next, take as the dividend process $d_t = \exp(thetalog(t))$. Since the expectation is now positive, we can use an exponentiated polynomial and, with a similar reasoning as before, we see that a first order exponentiated polynomial with $beta(1,1) = 20$ and $beta(2,1) = 1$ gives the precise solution for the logarithmic case. To move to the case of $A = .8$, $\gamma = -2$, start with the appropriate initial conditions, modify `psis.for` and `der.for`, and set up the homotopy.

Let us now consider a model with two stocks so that $J = 2$, and let us maintain strictly positive dividends $d_t = \exp\ thetalog(t)$. We can start the homotopy from the case of logarithmic utility and perfectly correlated dividends (the latter feature makes the model identical to one with a single asset). We now set

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad C = \begin{bmatrix} .4 & 0 \\ .4 & 0 \end{bmatrix} \tag{37}$$

Thus, in this case, an exponentiated first order polynomial in the current *thetalog*'s gives the exact solution. Since $theta(t,1) = theta(t,2)$, letting the parameters *beta* be (20, 0.5, 0.5), we have parameters to start the homotopy. Clearly, if the two *theta* are perfectly correlated the non-linear regression will fail because of perfect collinearity, thus we better start the homotopy with a small departure from the C matrix above (say $c(2,2) = 0.0001$). We also have to set the number of expectations $neq = 2$.

Then, with a sequence of homotopy steps, we can change the parameters of the dividend process to compute a model with two different risky assets by changing D, A, B and we can also change the risk aversion γ. Moreover if we make the second row of C go to (0, 0) we approximate the case of a riskless console bond that pays one unit of consumption with certainty in all periods from now into the future. If we proceed like this, at a certain point we have to eliminate $theta(t,2)$ from the *psi*, since it gets close to be a constant, and the non-linear regression would fail. Notice that even when $theta(t,2)$ is a constant still we will have two *psi* expressions, and perform two nonlinear regression, because the riskless asset's price is still a non-degenerate random variable.

4.2 Example II. Simple Stochastic Growth Model

See files in the directory `~\ex2grow`

The *psi* and its derivatives are defined in `psis.for` and `der.for`.

The variables and parameters are labeled as follows:

i	1	2		
$Z(i,t)$	k_t	c_t		
j	1	2	3	4
$alpha(j)$	δ	γ	α	$1-d$

The piece of program `ginvert.for` solves for c_t in the equation

$$c_t = (\delta \ psi(1,t))^{\frac{1}{\gamma}} \tag{38}$$

and then solves for k_t using the resource constraint. Then it computes

$$phi(t-1,1) = c_t^{\gamma} \cdot (\theta_t \ \alpha k_{t-1}^{\alpha-1} + 1 - d)$$

The homotopy is started with the Brock-Mirman solution that applies in the case of depreciation 1 and logarithmic utility. To do that we start with an economy with parameters:

j	1	2	3	4
$alpha(j)$	δ	γ	α	$1-d$
	0.95	-1.00	0.33	0.00

In this case an appropriate exponentiated polynomial corresponds exactly to the optimal policy. Using the formulas derived by Brock and Mirman, we see that the relation between the expression for the optimal policy function and those in the (exact) expression for the expectation can be easily derived substituting $c_t = \alpha \ \delta \ \theta_t \ k_{t-1}^{\alpha-1}$ in (38).

In particular, given the parameters fixed above, we have an equilibrium characterized by:

$$\begin{aligned} beta(1,1) &= \delta(1-\delta \ \alpha) - 1 = 1.533 \\ beta(2,1) &= -\alpha = -0.330 \\ beta(3,1) &= -1.000 \end{aligned}$$

We can run the program with these parameters to check that this is, indeed, the solution, and then change the parameters *alpha* manually or using the built-in homotopy procedure to compute an approximate solution for other sets of parameters; recall that, when the program ends, the last fixed point calculated is stored in `pini.dat`, and the initial condition is scratched. It is quite common that when there is full depreciation ($d = 1$) the series for capital becomes easily non-stationary for small changes in *beta* away from the optimal path, so that the algorithm may fail to converge. In the case we are discussing a solution is to set a value for *mu* smaller than one (say .5) as long as $(1-d)$ is close to 0. This simple correction is enough to prevent *beta* from exiting the stable region (i.e. the asymptotic unit circle). For example using twice the automatic homotopy we can compute an approximate solution for a model with depreciation, $1-d=0.8$, and $\gamma=-2.0$, and with an AR(1) process for $\log\theta$ with parameters $\rho=.9$ and $\sigma=.03$.

In general the order in which we change the variables should not affect the results obtained, but in models where there is a depreciation rate it seems that it is better to move the depreciation rate away from 100%, after that we can safely set *mu* closer to 1 and achieve faster convergence.

4.3 Example III. A Growth Model with Endogenous Labor Supply and Proportional Taxes

See files in the directory `~\ex3lab`.

We now have to address the issue of the numerical solution of the system g at each period t. Also, it will turn out that the setup of the homotopy in order to have a continuous path is non-trivial.

As we discussed before, to complete the inversion of the g system we use a simple numerical procedure to compute hours worked. Rewriting (7), we see that solving for $l_t(\beta)$ amounts to finding the zero of the function

$$f(l) \equiv (1-\tau^l)\;\delta\;\psi(\beta;x_t(\beta))\;\theta_t\;(1-\alpha)\;k_t^{\alpha}\;l_t^{-\alpha}-(1-l_t)^{\gamma_l}b \tag{39}$$

in the interval [0,1]. A numerical algorithm to solve (39) is appended to the main program as a subroutine named `federzero`. Also, at the end of the main program there are the definitions of the function f and of its derivative.[22]

To save computing time along the simulation we use as the starting point for the numerical procedure at a given time period the solution found at the preceding period (see discussion in the next section of this issue). This solution is stored as *whold* at the end of `ginvert.for` and retrieved at its beginning. Clearly, the very first period there is no *whold*, so at the beginning of the `simul` subroutine the value of the labor supply is initialized at the value

[22]Notice that in order to allow the subroutine and the program to share the parameters *alpha* we have redefined the common commands. Moreover the external functions f and $fder$ have been declared in the main program and in the subroutine `simul`.

$Z(1,2) = l_1$ written, for convenience, in `par.dat` as if l_t were a state variable (the number of 'state' variables $nxini$ is accordingly set to 2).

The other interesting question here is how to set up the homotopy path to move from the model already solved (Example II) down to the model we want to solve (Example III). We need a general model that encompasses both the initial model and the 'new' model, with some parameters controlling the movement from one model to the other. Also, we need that as we change these parameters a little bit the solution moves only slightly.

It is useful, though, to realize that all intermediate steps along the homotopy can correspond to a mechanical model where the equations do not necessarily correspond to rational forward-looking behavior. The starting and the ending point clearly will be fully fledged models and in some cases the intermediate models can be interesting too. However from a computational standpoint it is often easier to just see them as dummy models connecting one model to the other.

This is clear if we consider only the case of endogenous labor supply and, for now, set tax rates to zero. Here we could try to specify an economic model in which if a parameter takes on a particular value the labor supply is identically equal to 1 in equilibrium, thus replicating the basic model of growth. For example, we could let b in the utility function move from 0 to 1.

Unfortunately this procedure may bring additional problems from a computational point of view, since $(1-l)^{\gamma_l}$ will typically have a derivative that limits either to infinity or to 0 as l_t approaches 1. The subroutine `federzero` makes largely use of this derivative, and will easily fail to converge under these conditions. Formally, what happens is that the homotopy path defined in this way is discontinuous, precisely, at $b = 0$.

We can bypass this complication in the following way. We can solve at any period a model where actual working hours are given by $l_t^* = s\ l_t + (1-s)$ (with $0 \leq s \leq 1$), where l_t solves the first order condition of the original optimization problem. We use l_t^* to compute the return on capital, total production, and so on. In this way when $s = 0$ (and zero tax rates) the problem will be identical to the basic growth problem. For $0 < s < 1$ we cannot give an economic interpretation to our simulations, but eventually, as we reach $s = 1$, we have a correct solution to the model with endogenous labor supply. This is the procedure we adopted when first running the homotopy. Once we have an approximate solution for $s = 1$, we can as well eliminate it from our program. From then on it is straightforward to modify the tax rates from 0 to any level of interest, and in the same way to modify all the other parameters. The order in which the parameter values are changed should not matter for the final result, but sometimes the algorithm is faster changing some parameters first. As noticed above, if we start from the Brock-Mirman parameters, it is usually better to start by moving the depreciation rate away from 1, and then modify the other parameters (including the s).

Here we summarize the variable and parameter labels used by the program for this example. [23]

i	1	2	3					
$Z(i,t)$	k_t	l_t	c_t					
j	1	2	3	4	5	6	7	8
$alpha(j)$	δ	γ	α	γ_l	d	τ^l	τ^k	s

4.4 Example IV. Non-negative Gross Investment.

See files in the directory ~\ex4inv.

The naming of variables and parameters, and the baseline parameters values are:

i	1	2	3	4	5
$Z(t,1)$	k_t	c_t	λ_t	i_t	bind.ind.
j	1	2	3	4	
$alpha(j)$	δ	γ	α	$(1-d)$	
	0.95	-1.00	0.33	0.9	

Notice that we introduced some auxiliary variables $Z(t,i)$ that are not necessary from the purely computational point of view. The first is the investment level, $Z(t,4)$, that may be useful in the simulated series stored in `openout.dat`. The second is an indicator variable $Z(t,5)$, assuming value 1 whenever the non-negativity constraint is binding. The mean of the $Z(t,5)$

[23]The program uses an additional variable $Z(4,t)$ corresponding to the number of steps needed for `federzero` to converge. This kind of auxiliary variable is useful to check the performance of the numerical subroutine used. Also, an additional parameter used by the program is $alpha(9)$ that controls the speed of adjustment in the subroutine `federzero`.

variable represents the fraction of periods in which the non-negativity constraint is binding.[24]

The program `opent.for` is a slight modification of `open.for`. The computation of the summary statistics is performed not only at the end of the algorithm but also along the homotopy steps, producing a table with the values obtained. This program illustrates how the homotopy steps can be used to perform a comparative dynamics exercise. That is, along the homotopy we can see how changes in parameters affect the summary statistic of some simulated variable. In all cases in which the models along the homotopy have economic meaning we can immediately store the values of some statistic of interest. In this way the table we obtain displays the relation between a parameter of the model and some statistical features of the equilibrium path obtained.

For example, a simple qualitative result that we can expect from the model with non-negative investment is that lower depreciation rate are associated with a higher probability of being up against the constraint, and the greater is the number of times the multiplier λ_t is positive.

With the appropriate modification of the homotopy steps we obtain a table displaying the relation between a parameter and a summary statistic of the simulated model. For example, Table 1 displays the relation between the depreciation rate and the percent of times in which the constraint binds in the simulated economies, confirming our intuition.

We can also study the effect of the elasticity of the instantaneous marginal utility γ on investment and on the probability of having a binding constraint. Again just performing the homotopy we obtain Table 2 below ($1 - d$ is now kept fixed at 0.975).

The relation we found between risk aversion (absolute value of γ) and average investment is increasing, while the relation with the frequency of the binding constraint is non monotonic. A possible intuition for this result is the following. If the constraint binds often, this introduces higher volatility in consumption, since agents can not smooth consumption by eating their capital; the way to avoid this unpleasantly volatile consumption is to keep, on average, a larger investment and capital stock, so that hitting the constraint

[24]To increase copmutational efficiency all the variables that are not strictly needed for the iterations, such as $Z(t,4)$ and $Z(t,5)$ in this case, would be computed after the fixed point has been found. In the current model, however, the gain in efficiency would be minimal.

$(1-d)$	Binding fraction	$E(i_t)$	$E(\mu_t)$
.9000	.0000	.3273	.00000
.9100	.0000	.3260	.00000
.9200	.0000	.3232	.00000
.9300	.0000	.3181	.00000
.9400	.0000	.3098	.00000
.9500	.0010	.2969	.00000
.9600	.0032	.2770	.00004
.9700	.0203	.2455	.00032
.9800	.0998	.1915	.00208
.9900	.3677	.0952	.01216

Table 1: Frequency of binding constraint and $(1-d)$

is less likely. With higher risk aversion $-\gamma$, agents are more willing to settle, in steady state, to a high level of the capital stock in order to reduce the probability of being up against the constraint. This, in turn, has ambiguous implications on the observed frequency of the binding constraint. When risk aversion is higher there is more need to buffer against bad shocks eating the capital stock, but at the same time the capital stock is endogenously kept higher making this event less likely, as a consequence the frequency with which the lower bound is hit may be larger or smaller. Computation helps us to see these two opposite effects are balancing each other in our specific case, so that the binding frequency oscillates around 5 percent.

4.5 Example V. Habits in Consumption

See the directory `~\ex5hco`.

We only discuss here the case with two expectations.

The routine `Gbeta` now is performing two nonlinear regressions with four parameters each. The program lines in `ginvert.for` substitute the ψ's for the corresponding expectations in the Euler equation and use the transition for k to solve for c_t and k_t, given the shock process and the past values of capital and consumption. Since now c_{t-1} is a state variable the number of x to be initialized ($nxini$) is 2, and there is a starting value c_1 for consumption

γ	Binding fraction	$E(i_t)$	$E(\lambda_t)$
-.7	.056	.219	.0009
-1.1	.048	.221	.0009
-1.5	.047	.223	.0009
-1.9	.047	.225	.0009
-2.3	.048	.228	.0009
-2.7	.049	.231	.0009
-3.1	.049	.235	.0009
-3.5	.049	.239	.0008
-3.9	.049	.243	.0008
-4.2	.049	.246	.0007
-4.4	.048	.249	.0007
-4.6	.048	.251	.0006
-4.8	.047	.254	.0006
-5.0	.047	.256	.0005

Table 2: Binding constraint and γ

in the file `par.dat`. The variables and parameters labeling is exactly as in Example II with the addition of $alpha(5) = \eta$.

Notice that when $\eta = 0$ the model coincides with the one in Example II. Hence, the homotopy steps are taken in a natural way starting with the *beta*'s computed for that case and $beta(i, 4) = 0$. Let $\eta = 0$ in `alpha.dat`, and set to 0 the β corresponding to $\log c_{t-1}$ in `pini.dat`. All the β's for the second expression ($psi(t, 2)$) can be set freely at the first stage of the homotopy, because when $\eta = 0$ they do not affect the simulated path and so they cannot cause non-stationarity (at this stage the problem for the series $phi(t, 2)$ is just a problem of estimation, that does not feed back into the simulation part). After estimating the β's for the two parameterized expressions in the case $\eta = 0$, we can safely proceed with the homotopy and move η toward the range we are interested in.

4.6 Example VI.

Before discussing the use of the algorithm it is useful to describe the closed form solution that can be obtained in the simple case in which $d_1 = d_2 = 1$ and utility is logarithmic. Using the result stated above of a constant ratio of $k_{2,t}$ to $k_{1,t}$ on the optimal path, this case can be made to fit into the Brock-Mirman framework if $\alpha_1 = \alpha_2 = \alpha$. In this case, the form of the optimal policy for consumption and investment can be shown to be

$$c_t = (1 - 2\delta\alpha)\theta k_{1,t}^{\alpha} k_{2,t}^{\alpha} \tag{40}$$

$$k_{1,t} = k_{2,t} = \delta\alpha\theta k_{1,t}^{\alpha} k_{2,t}^{\alpha} \tag{41}$$

In turn, the first of these equations implies the following (exact) form for the expectation in (17) and (18).

$$\delta E_t \ldots = (1 - 2\delta\alpha)^{-1} \exp(\log\theta_t + \alpha\log k_{1,t} + \alpha\log k_{2,t}) \tag{42}$$

This solution will be useful both as a starting point for the homotopy steps and as a reference in order to discuss the convergence of the algorithm.

As we try to implement the algorithm using the equations (16), (17) and (32), the first problem we face is the choice of the state variables to be included in the parameterized expressions *psi*. The two capital stocks are natural candidates. But if we try, for example, to specify the function ψ as an exponentiated first order polynomial in $k_{1,t-1}$, $k_{2,t-1}$ and θ_t , we will be in trouble with the non-linear regression. The obvious reason is that the two capital stocks are highly collinear. Actually in the case already mentioned of $d_1 = d_2$ the two capital stocks are perfectly collinear, since $\frac{k_{2,t}}{k_{1,t}} = \frac{\alpha_2}{\alpha_1}$ on the optimal path.

On the other hand if we use only one capital level as a state variable we incur in a different type of problem. For example, if we tried simply to drop one of the capital stocks from the specification of *psi*, it turns out that, the simulated series for consumption and capital stocks either diverge or take negative values, and the algorithm often fails to converge.

Now consider the two capital goods model and in particular consider the case $d = 1$, in which the Brock-Mirman solution applies and consumption and investment are proportional to current production. Suppose we omit the first capital good from the state variables and we use only $k_{2,t-1}$ to compute the *psi*'s. At the simulation stage every small numerical error in the ratio of the capital stocks is amplified along the simulation, for example, if $k_{2,t-1}/k_{1,t-1} = 1 + \epsilon$ the production will be overstated, c_t and $k_{2,t}$ will be larger, while $k_{1,t}$ is computed as a residual, the implied difference equation is explosive, and so the ratio will tend to increase. A similar problem of non-stationary behavior of the simulated series arises also with different specification of the *psi*'s.

It seems that a source of this problem was the fact that one of the two capital stocks was computed as a residual, so we have chosen a different transformation of the second Euler equations, namely

$$c_t^{\gamma} \frac{k_{2,t}}{k_{1,t}} = \delta E_t c_{t+1}^{\gamma} \left[\theta_{t+1} \alpha_2 k_{1,t}^{\alpha_1 - 1} k_{2,t}^{\alpha_2} + (1 - d_2) \frac{k_{2,t}}{k_{1,t}} \right] \tag{43}$$

Then, we have set the `ginvert.for` file to invert the system given by (16), (17), and (43), recovering the $k_{2,t}$ to $k_{1,t}$ ratio from (43), and computing investment in the two capital goods according to this ratio and the total production available for capital (i.e. $y_t + (1 - d_1)k_{1,t-1} + (1 - d_2)k_{2,t-1} - c_t$).

Setting up the system (19) in this way we are able to overcome the problems of non-convergence mentioned above. Thus, we can compute an approximate solution of the model for different sets of parameter values, using the built-in homotopy steps. Moreover as we move away from the case of identical depreciation we can add to the explanatory variables in the exponentiated polynomial the log of $k_{2,t-1}$, without having problems with the regression subroutine. Clearly the two variables are still highly correlated, but this is not a problem given that we are not interested in the values of the coefficients *beta* but only in the predictions. Nevertheless the high correlation between two explanatory variables may be a practical problem if it generates spurious movements in the computed *beta* along the algorithm (movements that the program interprets as non-convergent behavior). For this reason, it is safer in these cases to use the convergence criterion based on the distance of the implied series. This is discussed more formally in the next section. The program `opens.for` is designed for this purpose.

As noted above this model differs in an interesting way from the growth model in Example II only insofar as the depreciation rates are different. A qualitative question we may ask is how investment is distributed between the two capital stocks with different depreciation rates over the cycle.

We have computed a solution for the parameters values reported below, together with the labels of the variables.

i	1	2	3	4		
$Z(t,i)$	$k_{1,t}$	$k_{2,t}$	c_t	$k_{2,t}/k_{1,t}$		
j	1	2	3	4	5	6
$alpha(j)$	δ	γ	α_1	α_2	$(1-d_1)$	$(1-d_2)$
	0.96	-1.00	0.4	0.2	0.3	0.8

For the two *psi* forms we have specified exponentiated polynomials in θ, $\log k_{1,t}$ and $\log k_{2,t}$ (the last *beta* was set equal to zero in the region close to the identical depreciation case). The program output shows a correlation between $Z(t,3)$ and $Z(t,4)$ of .148, indicating a positive relation between consumption and the proportion of capital good $k_{2,t}$ to the total stock. This economy displays booms in which accumulation is concentrated on capital goods with low depreciation rates, while during recessions the capital stock composition is costlessly modified in favor of $k_{1,t}$, in order to increase productivity in the short term (as $\alpha_1 > \alpha_2$).

The following table (obtained as the tables in Example IV) displays the relation between the risk aversion parameter and some statistics on the simulated series. As *gamma* increases the comovement of capital composition with the cycle is enhanced. That is, during booms more capital is shifted towards low-depreciation/low-return capital stocks. As a consequence of this policy consumption variability is slightly decreased for larger risk aversion.

If we let the homotopy steps go back to a case of identical depreciation rate (e.g. move d_1 to 0.8) we obtain the comforting result that the simulated

γ	corr(c,ratio)	std.dev(c)	std.dev(ratio)
-1.0000000	.1481591	.0538640	.0652414
-1.1000000	.1683820	.0537597	.0676530
-1.2000000	.1882749	.0536681	.0700300
-1.3000000	.2081062	.0535875	.0723799
-1.4000000	.2279308	.0535159	.0747565
-1.5000000	.2470898	.0534498	.0770776
-1.6000000	.2657092	.0533886	.0793931
-1.7000000	.2837222	.0533309	.0817056
-1.8000000	.3013010	.0532761	.0840440
-1.9000000	.3177685	.0532227	.0863246
-2.0000000	.3335457	.0531707	.0886010

Table 3: Risk aversion and the commovement of consumption with capital composition

ratio $k_{2,t}/k_{1,t}$ is on average 0.5056 with standard deviation of 0.0086, which is pretty close to the theoretical result of a ratio constant over time and equal to 0.5 ($=\frac{\alpha_2}{\alpha_1}$).

5 Conclusion

In this chapter we discuss some very practical aspects of solving non-linear stochastic dynamic models with PEA. As we mentioned in the introduction, our purpose is not to describe the full scope of all the applications that can be performed with PEA, and we have limited our discussion to solving Euler equations at the stationary distribution. Restricted to this case, we have discussed many practical issues and the use of a publicly available fortran program. We hope the reader will see that the ideas here are easy to apply. We have shown how the computational costs do not increase exponentially if more state variables or stochastic shocks are introduced, how one can refine the solution by introducing more flexible functional forms, and how the algorithm can be used to calculate suboptimal equilibria.

Missing from the chapter is a comparison with other methods, both at the theoretical and practical level. The reader of the book will probably realize that this is an important issue, as each method has advantages and disadvantages, there is no algorithm that is the best one for all models, and there is no general framework that encompasses a large number of algorithms. Here, we can offer some informal comments on how PEA compares to other algorithms. It should be clear to the careful reader of this chapter that a grid was never introduced in the state space and that an integral was never explicitly calculated (although Step 3 does calculate an integral implicitly). This means that all the problems with grids and integration in multi-dimensional spaces are avoided here. There is no 'curse of dimensionality' as we introduce more state variables, and models with three or four state variables can be easily solved taking into account of the non-linearities in the solution. Furthermore, the algorithm endogenously selects the values of the state variables at which to fit the solution, by performing the simulation of Step 2. This is important because the user does not have to specify a range of relevant values for the state variables, and we need a lower degree of the polynomial to achieve a reasonable fit, and the number of computations does not increase exponentially as the state space increases. For these reasons, this approach to solving non-linear models is relatively more efficient when there are several state variables and stochastic shocks with continuous distributions.

The simple-minded-numerical-analysis approach that we have taken here is overly simplistic and could be improved upon. There are better ways of computing integrals in Step 3 than just running the long run simulation of Step 2. Certain forms of hill-climbing to find the fixed point might be a good idea. And the possibilities for homotopy have only been scratched. But before we complicate our lives introducing these techniques, it seems interesting to see how far we can go solving models just by computing a few simulations and running a few regressions.

References

- Christiano, L. and J. Fisher (1994); "Prototype Algorithms for Solving Dynamic Models with Occasionally Binding Constraints", working paper, Northwestern University and University of Western Ontario.
- Coleman, W.J. (1991) "Equilibrium in a Production Economy with an Income Tax", *Econometrica*, 59, pp 1091-1104.
- den Haan, W. and Marcet A. (1994); "Accuracy in Simulations", *Review of Economic Studies*, January.
- den Haan and Marcet (1990); "Solving a Simple Growth Model by Parameterizing Expectations" *Journal of Business and Economic Statistics*, January, 31-34.
- Domowitz I. and M. A. el Gamal (1993); "A Consistent Test of Stationarity-Ergodicity" *Econometric Theory*, vol. 9, no. 4, pp 589-601
- Duffie, D., and K.J. Singleton (1993); "Simulated Moments Estimation of Markov Models of Asset Prices," *Econometrica,* vol. 61, 929-952.
- Judd, K."Projection Methods in Optimal Growth Models", *Journal of Economic Theory*, December pp. 410-453.
- Marcet, A. (1988); "Solving Non-linear Stochastic Models by Parameterizing Expectations", working paper, Carnegie Mellon University.
- Marcet, A. and Marimon, R. (1992); "Communication, Commitment and Growth", *Journal of Economic Theory*, December, pp. 219-250.
- Marcet, A. and D.A. Marshall (1994); "Convergence of Approximate Model Solutions to Rational Expectations Equilibria Using the Method of Parameterized Expectations," Working Paper No. 73, Department of Finance, Kellogg Graduate School of Management, Northwestern University.
- Marcet, A. and K.J. Singleton (1990) "Equilibrium Asset Prices and Savings of Heterogeneous Agents with Portfolio Constraints" Working Paper, Carnegie Mellon University.

- Pindyck, R.S. and D.L. Rubinfeld (1981) *Econometric Models and Economic Forecasts* (2nd. ed.), New York, McGraw-Hill.