Journal of Mathematical Finance, 2018, 8, 317-334
http://www.scirp.org/journal/jmf
ISSN Online: 2162-2442
ISSN Print: 2162-2434

Projection Methods and the Curse of Dimensionality

Burkhard Heer[1], Alfred Maußner[2]

[1]University of Augsburg and CESIfo, Augsburg, Germany
[2]University of Augsburg, Augsburg, Germany
Email: Burkhard.Heer@wiwi.uni-augsburg.de, Alfred.Maussner@wiwi.uni-augsburg.de

How to cite this paper: Heer, B. and Maußner, A. (2018) Projection Methods and the Curse of Dimensionality. *Journal of Mathematical Finance*, **8**, 317-334.
https://doi.org/10.4236/jmf.2018.82021

Received: February 6, 2018
Accepted: May 5, 2018
Published: May 8, 2018

Abstract

We study the ability of three different projection methods to solve high-dimensional state space problems: Galerkin, collocation, and least squares projection. The curse of dimensionality can be reduced substantially for both Least Squares and Galerkin projection methods through the use of monomial formulas. Least Squares are shown to require a good initial value in order to give an accurate solution. Alternatively, we suggest a new ad hoc collocation method for complete polynomials that is fast and easy to implement.

Keywords

Projection Methods, Galerkin, Collocation, Least Squares, Heterogeneous Agents, Stochastic Dynamic General Equilibrium

1. Introduction

In this paper, we study projection methods which have become a standard tool in the analysis of business cycle models and, in particular, asset prices. These models are highly non-linear and cannot be solved with standard linearization methods. For example, [1], Section 6.3.4 and [2] use projection methods to compute equity premia in general equilibrium models of the business cycle, while [1], building on the idea of [3], also apply this method in order to solve models with occasionally binding constraints, e.g. in the form of non-negative investment.[1] More recently, [4] applies projection methods to a heterogeneous-firm model where firms face aggregate uncertainty and investments are lumpy.

A considerable difficulty in these models arises from the "curse of dimensionality". The curse of dimensionality[2] most generally describes problems

[1][3] use perturbation rather than projection methods.
[2]The term was originally coined by [5] in the context of dynamic optimization.

in the analysis of data that only arise in high-dimensional space. In the present context, we examine the phenomenon more narrowly in the consideration of numerical analysis (with an application to the solution of a stochastic general equilibrium model), in which the computational time rises exponentially with the state space dimension and constitutes a binding constraint even with current computer technology.

Three different kinds of this method are considered: Galerkin, Collocation, and Least Squares.[3] The three methods all try to find a good fit to a policy function of the state variable $x \in \mathbb{R}^n$ that is characterized by a parameter $\phi \in \mathbb{R}^m$. The policy function may take the form of a polynomial, for example, and the ϕ_i are simply the coefficients of the polynomial. Furthermore, the policy function is approximated over the bounded interval $D = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$ with $-\infty < a_i < b_i < \infty$ for all $i = 1, \cdots, n$. For this policy function, we are able to compute the so-called residual function that characterizes our problem. The residual function $R(x, \phi)$ may take the form of a first-order condition or an equilibrium condition and we try to choose ϕ so that $R(x, \phi)$ is close to zero over the domain *D*. The three different methods considered in the following differ with regard to the projection step *i.e.* how we choose the criterion of making $R(x, \phi)$ close to zero. The least squares method solves an optimization problem by minimizing the sum of the squared residuals over the domain *D*, while the collocation method finds the solution ϕ by setting the residual $R(\phi, x)$ to zero at exactly m points x_i, $i = 1, \cdots, m$. Galerkin projection, like collocation projection, solves a non-linear equations problem. With this method, the residual is multiplied by the basis of the policy function and integrated over the domain *D*. The values of ϕ are chosen so that the integrals for the m basis functions are equal to zero.

While Galerkin and collocation projection methods have been analyzed and applied extensively to the solution of stochastic dynamic general equilibrium models, least squares methods have only been given little emphasis in the solution of these problems. For example, Galerkin and collocation projection methods have been demonstrated to be a very useful tool in the solution of the standard stochastic growth model.[4] In particular, Chebyshev polynomials have been shown to provide very accurate approximation of the policy function in many examples.[5] On the other hand, the literature has only paid little attention to the solution of the stochastic growth model or any other dynamic stochastic general equilibrium model with the help of Least Squares projection.[6] This observation is somehow puzzling as a priori we would assume that it is easier to solve a minimization problem than a non-linear equations problem. In the former case, we are at least certain to find a solution, even if it may turn out to be only a local, but not a global minimum.

[3]For a description of various projection methods, please see [7], Chapter 11.

[4]Please see [6] [7] [8] and [9].

[5]See, e.g., [7] or [1].

[6]One of the few exception is [1]. In Chapter 4 of this book the stochastic growth model is solved employing both least squares and Chebyshev collocation.

In the following, we analyze if projection methods can successfully be applied to higher-dimensional state space problems that arise naturally in the context of heterogeneous-agent economies. In order to solve problems that are characterized by a state space of dimension *N*, we need to reduce the number of coefficients *m* in the approximating policy function. Assume that we approximate the policy function by a polynomial function of degree 2. If we use the tensor product, the number of coefficients for a 4-dimensional state space is equal to $3^4 = 81$. If we increase the dimension of the state space to 8, the number of coefficients is already equal to $3^8 = 6561$. This exponential growth of the number of coefficients, of course, becomes a binding constraint on computational time. [10] suggest to use complete polynomials instead. In this case, the number of coefficients only amounts to 15 and 45 in dimension 4 and 8, respectively. We, therefore, rather consider complete polynomials than tensor products.

If we choose complete polynomials for the approximation of the policy function, however, we run into problems with the standard collocation method, where we simply set the residual function equal to zero at a number of points that is equal to the number of coefficients (and solve a system of non-linear equations). How should we choose the points, e.g. the 15 points in dimension 4? One possible solution is the Smolyak's algorithm presented by [11]. The Smolyak algorithm is a device how to pick the collocation points optimally. However, this algorithm suffers from its lack of universal applicability as it only works for certain combinations of the state space dimension and the degree of the complete polynomial in the approximating function.

In this paper, we propose three different projection techniques to address this problem that are also universally applicable. First, we consider the standard Galerkin projection method in Section 3. Second, we suggest a rather ad hoc collocation procedure in Section 4 that is found to perform rather well. Finally, we solve a least squares problem instead, *i.e.* we simply minimize the sum of the squared residuals. This method is presented in Section 5. The rest of the paper is organized as follows. In Section 2, the model is presented, and Section 6 concludes.

2. The Model

We study a simple social planner's problem in a *N*-country model. Time is discrete and denoted by *t*. The utility function of the representative agent in country *n* is $u^n\left(c_t^n, l_t^n\right)$ where c_t^n is consumption and l_t^n is labor supply in this country *n* at time *t*. We assume that country *n*'s production of the single good equals $f^n\left(k_t^n, l_t^n\right)$ where k_t^n is the capital stock. The social planner solves the following problem

$$\max_{\left\{\left(c_t^n, i_t^n, l_t^n\right)_{n=1}^{N}\right\}_{t=0}^{\infty}} \sum_{n=1}^{N} \tau^n \mathbb{E}_0\left(\sum_{t=0}^{\infty} \beta^t u^n\left(c_t^n, l_t^n\right)\right) \tag{1}$$

subject to

$$k_{t+1}^n = i_t^n + (1-\delta)k_t^n, \quad n = 1, 2, \cdots, N, \tag{2}$$

where δ is the depreciation rate of capital and i_t^n is gross investment in country n at time t, and where τ^n is the Negishi weight for country n. The world budget constraint is given by

$$\sum_{n=1}^{N} c_t^n + \sum_{n=1}^{N} i_t^n - \sum_{n=1}^{N} \delta k_t^n = \sum_{n=1}^{N} \left(a_t^n f^n\left(k_t^n, l_t^n\right) - \frac{\varphi}{2} k_t^n \left(\frac{i_t^n}{k_t^n} - \delta \right)^2 \right). \tag{3}$$

The productivity shocks are generated by the law of motion

$$\ln a_t^n = \rho \ln a_{t-1}^n + \sigma\left(e_t + e_t^n\right), \tag{4}$$

where $e_t^n \sim N(0,1)$ and $e_t \sim N(0,1)$ are i.i.d. normally distributed random variables. The production function is the standard CES specification (including the special case of Cobb-Douglas) where

$$f(k, l; \mu) = \begin{cases} \left(\alpha k^\mu + (1-\alpha) l^\mu\right)^{1/\mu}, & \mu \neq 0 \\ k^\alpha l^{1-\alpha}, & \mu = 0, \end{cases} \tag{5}$$

which implies that $1/(1-\mu)$ is the elasticity of substitution between capital and labor.

We will examine three kinds of utility functions. The first will be the separable utility function

$$u(c, l) = \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}} - b\frac{l^{1+\frac{1}{\eta}}}{1+\frac{1}{\eta}}, \tag{6}$$

where γ is the intertemporal elasticity of substitution and η is the elasticity of labor supply. A special case of this will be the inelastic labor supply case:

$$u(c) = \frac{c^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}}, \quad l \equiv 1.0. \tag{7}$$

The second utility function we will use is the Cobb-Douglas specification

$$u(c, l) = \frac{\left(c^\psi \left(L^e - l\right)^{1-\psi}\right)^{1-\frac{1}{\gamma}}}{1-\frac{1}{\gamma}} \tag{8}$$

where L^e is labor time endowment. The third utility function we will use is the CES specification

$$u(c, l) = \frac{\left(c^{1-\frac{1}{\chi}} + b\left(L^e - l\right)^{1-\frac{1}{\chi}}\right)^{\left(1-\frac{1}{\gamma}\right)\left(1-\frac{1}{\chi}\right)}}{1-\frac{1}{\gamma}}, \tag{9}$$

where χ is the elasticity of substitution between consumption and leisure.

2.1. Specification of Negishi Weights and Steady States

Each problem is specified in terms of the Negishi weights τ^n. These are tied down in general equilibrium models by endowments. However, we will specify Negishi weights instead of endowments since this allows us to focus on solving multidimensional dynamic models. If we did want to specify endowments instead and compute general equilibrium prices, the Negishi method would be the most natural way to proceed; that is, we would compute allocations (and the implied prices) conditional on the Negishi weights and then use a finite-dimensional nonlinear equation solver to find the weights that implied equilibrium. Therefore, we focus on the dynamic problem for fixed Negishi weights.

We use a simple rule to pin down the Negishi weights. For each problem, the Negishi weights are to be chosen so that the steady state consumption in each country would equal its net output if the productivity shocks were eliminated. This is a sensible choice since it implies that net foreign asset income is small, a rough approximation of reality.

We will also assume initially that all countries are the same size. Specifically, we will assume that the steady state capital stock and labor supply both equal unity for all countries. The parameter choices made below are intended to produce that result.

2.2. Common Parameter Values

Some parameters will be fixed at one value for all examples. We must choose a common β in order for the solution to be stationary. We choose $\beta = 0.99$ so that the period of time is about a quarter. We also fix the values for α and δ since these reflect standard choices and their variations will not present significant computational challenges. The adjustment cost parameter φ covers an empirically relevant range. The stochastic parameters will represent high and moderate persistence, and high and low productivity shocks. Table 1 summarizes our parameter choice.

3. Galerkin Projection

In this section, we will first introduce you to the projection method and to the Galerkin method in more detail. Specifically, we will describe how we implemented the method for the computation of the model in section 2. Second, we will present accuracy results for the Galerkin projection.

Table 1. Parameter settings.

Preferences	$\beta = 0.99$	
Production	$\alpha = 0.36$	
Capital Accumulation	$\delta = 0.025$	$\phi = \{0.5, 2.0, 10.0\}$
Schock Process	$\rho = \{0.8, 0.95\}$	$\sigma = \{0.001, 0.01\}$

3.1. The Numerical Method

With the help of projection methods, we want to approximate an unknown function $f : X \to Y$, where X and Y are subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$, respectively:

$$\hat{f}(x) = \sum_{i=0}^{p} \phi_i \psi_i(x),\, x \in X \subset \mathbb{R}^n. \tag{10}$$

This function is implicitly defined by the functional equation $F(f) = 0$, where $F : C_1 \to C_2$. C_1 and C_2 are given spaces of functions, e.g., the set of all continuously differentiable functions on $[a,b]$. Examples of functional equations are the Bellman equation or the Euler equation of the stochastic growth model. The functions may also take the form of equilibrium conditions, for example, such as supply equals demand. In the model of section 2, the functional equations are represented by the first-order conditions with respect to current-period consumption and labor supply and the next-period capital stock, respectively:

$$\forall n \neq m, n, m = 1, \cdots, N:$$

$$0 = \tau_n \frac{\partial u^n(c_t^n, l_t^n)}{\partial c_t^n} - \tau_m \frac{\partial u^m(c_t^m, l_t^m)}{\partial c_t^m} \tag{11a}$$

$$\forall n = 1, \cdots, N:$$

$$0 = \frac{\partial u^n(c_t^n, l_t^n)}{\partial l_t^n} - \frac{\partial u^n(c_t^n, l_t^n)}{\partial c_t^n} a_t^n \frac{\partial f^n(k_t^n, l_t^n)}{\partial l_t^n}, \tag{11b}$$

$$0 = \frac{\partial u^n(c_t^n, l_t^n)}{\partial c_t^n} \left[1 + \varphi\left(\frac{n_t^n}{k_t^n} - \delta\right)\right] - \beta \mathbb{E}_t \frac{\partial u^n(c_{t+1}^n, l_{t+1}^n)}{\partial c_{t+1}^n} \tag{11c}$$

$$\times \left[1 + a_{t+1}^n \frac{\partial f^n(k_{t+1}^n, l_{t+1}^n)}{\partial k_{t+1}^n} + \varphi\left(\frac{i_{t+1}^n}{k_{t+1}^n} - \delta\right)\left[1 + \frac{1}{2}\left(\frac{i_{t+1}^n}{k_{t+1}^n} - \delta\right)\right]\right] \tag{11d}$$

together with the world budget constraint (3). These equations can be rewritten as implicit functions of the state variables x_t that consist of the individual capital stocks k_t^n and technology levels a_t^n, $x_t = (k_t^1, \cdots, k_t^N, a_t^1, \cdots, a_t^N)$. In our example, we will try to approximate the next-period capital stock k_{t+1} and labor supply l_t by a function *f*.

The residual function is obtained by substituting $\hat{f}$ into the functional equations:

$$R(\phi, x) := F(\hat{f}(\phi, x)), \quad \phi := (\phi_0, \cdots, \phi_p).$$

For the true solution *f*, the residual function is equal to zero. In addition, we also normalize the residual function to one so that deviation from zero can be interpreted as percentage deviations. For example, the residual function that represents the first-order condition with respect to current-period consumption is formulated as follows:

$$R\left(x_t;k_{t+1}(.),l_t(.)\right)=\frac{\tau_m\dfrac{\partial u^m\left(c_t^m,l_t^m\right)}{\partial c_t^m}}{\tau_n\dfrac{\partial u^n\left(c_t^n,l_t^n\right)}{\partial c_t^n}}-1. \tag{12}$$

Suppose there is a set of test functions $\{g_i(x)\}_{i=0}^p$ and a weight function $w(x)$. Together with R they define an inner product given by

$$\int_X w(x)R(\phi,x)g_i(x)\mathrm{d}x.$$

On a function space, this inner product induces a norm on this space and we choose the vector of parameters ϕ such that

$$\int_X w(x)R(\phi,x)g_i(x)\mathrm{d}x=0,\quad \forall i=0,1,\cdots,p. \tag{13}$$

The three different solutions considered in this paper are derived for special choices of g_i and w.

- The Galerkin solution chooses $g_i\equiv\psi_i$ and $w\equiv 1$.
- The collocation method uses the Dirac delta function as weight function,

$$w(x)=\begin{cases}0 & \text{if } x\neq x_i,\\ 1 & \text{if } x=x_i,\end{cases}$$

and puts $g_i\equiv 1$.

- The least squares solution puts $g_i\equiv\partial R/\partial\phi_i$ and $w\equiv 1$.

We summarize the general procedure that underlies projection methods in the following algorithm that is adapted from [7].

Algorithm (Projection Method)

Purpose: Approximate the solution f: $X\to Y$ of a functional equation F(f) = 0.

Steps:

Step 1: Choose a bounded state space $X\subset\mathbb{R}^n$ and a family of functions $\psi_i(x):X\to Y$, $i=0,1,\cdots$.

Step 2: Choose a degree of approximation p and let

$$\hat f(\phi,x)=\sum_{i=0}^p\phi_i\psi_i(x).$$

Step 3: Define the residual function:

$$R(\phi,x):=F\left(\hat f(\phi,x)\right).$$

Step 4: Choose a projection function g_i, a weight function w and compute the inner product:

$$G_i:=\int_X w(x)R(\phi,x)g_i(x)\mathrm{d}x,\ i=0,\cdots,n.$$

Find the value of ϕ that solves $G_i=0$, or, in the case of least squares projection ($g_i=\partial R/\partial\phi_i$ and $w\equiv 1$), minimize

$$\int_X R(\phi,x)^2\,\mathrm{d}x$$

with respect to ϕ.

Step 5: Verify the quality of the candidate solution ϕ. If necessary, return to step 2 and increase the degree of approximation n or even return to step 1 and choose a different family of basis functions.

In step 1, the boundaries are found with some trial and error. In the following, we will concentrate on the symmetric case ($\tau_i = \tau_j \equiv 1$) with $N = 4$ countries and the benchmark calibration $\alpha = 0.36$, $\beta = 0.99$, $\delta = 0.025$, $\varphi = 0.5$, $\rho = 0.95$, and $\sigma = 0.001$. The labor supply is fixed, $l \equiv 1$, and the steady state capital stock is normalized to one, $k = 1$. Furthermore, we use a Cobb-Douglas production function ($\mu = 0$), and the utility function (6) with $\gamma = 1.0$. For this benchmark case, we approximated the policy function for the capital stock on the sets $[0.90, 1.10]$ for the individual capital stocks. For the technology level a_t^i, we assumed that it does not deviate from its steady state level 1.0 by more than six times its unconditional standard deviation, $\rho \big/ \sqrt{1-\sigma^2}$. In step 2, we picked ψ_i from the family of the Chebyshev polynomials. We report the results for a degree of approximation $p = 2$.[7] In the symmetric case, the residual function is one-dimensional. We obtain the residual function from the intertemporal first-order condition (11c) after having inserted the Equation (2) for the accumulation of the capital stock and the world budget constraint (3). Again, we normalized the residual function to one by dividing (11c) through the marginal utility of current-period consumption.

Step 4 deserves some discussions. In particular, we applied various devices that are non-standard in the literature in order 1) to make it faster and 2) to keep it from breaking down. The first adjustment is necessary in the presence of high-dimensional state spaces, while the second adjustment is found to be useful in dynamic stochastic equilibrium models in general. 3) A third adjustment was necessary for the Galerkin and least squares projection due to the numerical accuracy of the hardware.

1) Consider the projection step 4 where we compute a high-dimensional integral in the case of both Galerkin and least squares projection.[8] As one possible way, one can apply Gaussian formulas to compute this integral, for example Gauss-Chebyshev integration. However, we find this method to be too time-consuming in the present application once we have a state space of dimension 8 or higher (even if we only pick three nodes in each dimension). For this reason, we rather compute the integral (13) with the help of a monomial formula as suggested by [7].[9] In particular, we use the following monomial formula for the cube which is taken from [12]:

[7]For a linear approximation ($p = 1$), the projection methods did not provide accurate solution for the case with $N \geq 6$ countries. In particular, the dynamic behavior of the capital stock k became unstable when we simulated over many periods.

[8]With collocation, we only compute a finite number of points.

[9][9] also applies monomial formula in his study of Galerkin projection methods. His integration step, however, differs from ours in the choice of the grid points and the integration formula. In addition, we also provide a new algorithm for the collocation projection that is efficient and fast.

$$\int_{[-1,1]^n} f(x)\mathrm{d}x \approx Af(0,\cdots,0) + B\sum_{i=1}^{n}\left(f\left(re^i\right) + f\left(-re^i\right)\right) + D\sum_{x\in C} f(x), \tag{14}$$

where $C = \{x \mid \forall(x_i = \pm 1)\}$ and

$$r = \sqrt{\frac{2}{5}}, V = 2^n, A = \frac{8-5n}{9}V, B = \frac{5}{18}V, D = \frac{1}{9}.$$

By using this rule,[10] we reduce the number of computations of $f(x)$ from y^n in the case of the Chebyshev integration with y nodes to $2^n + 2n + 1$. For our problem, we find the Chebyshev integral and the monomial rule to coincide for the first four digits.

We also applied monomial rules to the computation of the expectation on the right-hand side of the first-order condition (11c). In order to compute the residual, we, again use a monomial formula from [12]. If $x \in \mathbb{R}^n$ has a multivariate standard normal distribution, the expectation $E(f(x))$ can be computed by the following:

$$(2\pi)^{-n/2}\int_{R^n} f(x)\mathrm{e}^{-\sum_{i=1}^{n} x_i^2}\mathrm{d}x \simeq \frac{1}{2n}\sum_{i=1}^{n} f\left(\pm\sqrt{n/2}e_i\right), \tag{15}$$

where $e_i = (0,\cdots,0,1,0,\cdots,0)$ denotes the i-th unit vector. In order to apply this rule to the general case of a random normal vector z with mean μ and covariance matrix Σ, we use the change of variable technique.[11] The linear transformation

$$x = \frac{\Sigma^{-1/2}}{\sqrt{2}}(z-\mu)$$

implies that $E(f(z))$ can be expressed as an integral function of x:

$$\begin{aligned} E(f(z)) &= (2\pi)^{-n/2}|\Sigma|^{-1/2}\int_{\mathbb{R}^n} f(z)\mathrm{e}^{\frac{-1}{2}(z-\mu)'\Sigma^{-1}(z-\mu)}\mathrm{d}z \\ &= \pi^{-n/2}\int_{\mathbb{R}^n} f\left(\sqrt{2}\Sigma^{1/2}x + \mu\right)\mathrm{e}^{-\sum_{i=1}^{n} x_i^2}\mathrm{d}x. \end{aligned}$$

This integral can be approximated by formula (15). Again, we find this formula to perform rather well compared to standard Gauss-Hermite integration. For the expectation in the first-order condition of (11c) in the symmetric benchmark case, the first four digits of the monomial formula integral (15) and the Gauss-Hermite integral with 4 nodes coincide.

2) In order to apply our integration routine (14), we, however, need to be careful. Assume we approximate a one-dimensional function $f(x)$ on $[a,b]$. As you can see from the formula (14), we also evaluate the function $f(x)$ at the boundaries $x = a$ and $x = b$.[12] Now consider our problem introduced in section 2. In order to compute the integral over the squared residuals, we have to evaluate the residual for a capital stock k in period t at the boundaries of the approximation interval $[k_{\min}, k_{\max}]$. In the likely case that our initial guess is not

[10]In order to apply the monomial formula (17), we need to transform the variables into the canonical form. This procedure is described in [7] Ch. 7.5.

[11]See, e.g., Theorem 7.5.3 in [7].

[12]Similarly, if we apply Chebyshev integration instead, for y>1, two of the Chebyshev nodes will be close to the lower and upper boundary limit.

very close to the true solution, we might choose a next-period capital stock k' in $t+1$ that is outside the approximation interval $[k_{\min}, k_{\max}]$. We, however, also need to compute k'' in period $t+2$ in order to compute c_{t+1} in (11c) with the help of the world budget constraint (3). Therefore, we might have to compute an approximation of the policy function that lies outside the approximation interval *D*. Outside *D*, however, the approximation accuracy might deteriorate quickly so that our computation might break down because of a variable overflow or a negative value for c_{t+1}. Indeed, we encountered this problem in almost all our applications. In order to circumvent the problem, we suggest the following two procedures that we also apply for the other two projections methods below. a) First choose an interval *D* for the approximation of the policy function. Next choose a subset $D_1 \subset D$ and compute the sum of projected residuals (squared residuals) over this smaller interval. In all our applications, this device worked rather well. Typically, we choose the length of the edges of D_1 to be approximately 50% - 70% of the corresponding ones in the set *D*.

b) Routines that solve non-linear equations, as the modified Newton-Raphson method with line search, require a good starting value. In our applications, we started with the solution from the log-linear model for the computation of the Chebyshev approximation with a degree $p=1$. For a degree $p>1$, we simply used the solution for the approximation with the degree $p-1$. Even with this values, however, the Newton-Rhapson algorithm was trying values for the vector ϕ where it is impossible to evaluate the residual function. For this reason, the routine that evaluates the residual function must return an error flag that signalizes the calling program to stop. Otherwise, the program will crash because of overflows, underflows, or other run-time errors arising from undefined numerical operations. Yet, standard software usually assumes that it is possible to evaluate a given non-linear system everywhere and there is no way to tell the program to do otherwise. Therefore, we wrote our own non-linear equations solver where the step size in the Newton-Rhapson algorithm is reduced as long as the evaluation of the residual function returns an error flag.[13]

3) In the cases of the Galerkin and the least squares projection, we also find a third device to be very helpful. Typically in the computation of stochastic dynamic general equilibrium models, we would like to find a solution with residuals that are small and below 10^{-4} or even 10^{-5}. Therefore, the value of the squared residual is of the typical order 10^{-8} or even less. If we integrate the projections of the (squares) residuals over the n-dimensional interval $D=[a_1,b_1]\times[a_2,b_2]\cdots\times[a_n,b_n]$, for example, the result is typically in the range of the order $\prod_{i=1}^{n}(b_i-a_i)10^{-6}$ or even lower. If the parameters of the problem are chosen such that the lengths of the intervals b_i-a_i are small, we end up

[13]The algorithm is downloadable from our homepage. The link is:
http://www.uni-augsburg.de/vwl/maussner/englisch/chair/maussner/pap/mnr.for
In order to decrease computational time, we also rather used forward than central differences in order to compute the Jacobian.

computing an integral value that is close to the accuracy of our numerical software (we used FORTRAN 95 with double precision data type). Indeed, this is the case for our state space where the typical integral (13) amounts to less than 10^{-16} in the symmetric 4-country benchmark case, for example. We, therefore, adjusted the scale of our sum of squared residual in (13) and normalized it to one by dividing the integral by the volume of the interval D_1. In this regard, our procedure is equivalent to the standard OLS approach where the total weight of all squared residuals is set equal to one.

3.2. Accuracy Check

In order to evaluate the performance of the projection methods, we use two different tests:

- **Accuracy test 1**: We compute a sample of 100 points x_i at radius r from the deterministic steady state. We compute the absolute value of the residual $R(x_i)$ for each point in the sample. The maximum of these values for each radius r are reported in Table 2.[14]
- **Accuracy test 2**: We simulate our solution to produce a sequence of values for the state, $\{x_t\}_{t=1}^{T}$. For $t = 1, 10, 20, \cdots, T$, we compute the maximum and the mean of the (absolute) residual.[15]

Table 2 presents the results for the Galerkin projection. Naturally, the accuracy declines with increasing radius from the steady state in the accuracy test 1 (please see the first two columns of the table). Remember that we only approximated the policy function inside the radius $r = 0.10$. For this region (which is also the relevant region during the simulation), the maximum deviation of the residual function deviates amounts to only 0.20%. The good performance of the algorithm is also reflected in the values of the accuracy test 2. Even after 100 simulation, the maximum absolute value of the residual did not

Table 2. Galerkin projection.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t}\lvert R(x_t)\rvert$
0.01	0.20E−5	1	0.41E−6	0.41E−6
0.02	0.31E−5	10	0.70E−6	0.42E−6
0.05	0.18E−4	20	0.70E−6	0.27E−6
0.10	0.21E−3	30	0.70E−6	0.22E−6
0.15	0.71E−3	40	0.11E−5	0.32E−6
0.20	0.18E−2	50	0.11E−5	0.45E−6
0.30	0.73E−2	100	0.14E−5	0.45E−6

Notes: The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 4 minutes 49 seconds and the program needed 8 iterations over ϕ.

[14]This is the style of error used in [6].

[15]This style of error was used in [13].

exceed 0.0014% and the mean was only 0.00045%.[16]

The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 4 minutes 49 seconds and the program needed 8 iterations over ϕ. Our algorithm, therefore, is reasonably fast and accurate considering that we compute a policy function for an 8-dimensional state space (in the case $N = 4$, we have four individual capital stocks k_t^i and four individual productivities a_t^i). If we increase the number of countries to 6 or 8, we run into the limits for the computational time on our Pentium III, 846 MHz computer. As can be seen from Table 3, the runtime increases quickly to multiple days. Therefore, for the computer technology at our current disposal, 16 state variables (corresponding to $N = 8$) has been found to be the limit.

4. Collocation Projection

In the collocation projection, we use the Dirac delta function as projection function with a weight identical to one. Therefore, we only compute the projection at single points and circumvent the computation of integrals (except for the computation of the expectation). Of course, this will save a lot of computer time if the state space is large. Instead, the task is to solve the non-linear equation system

$$R\left(\phi, x_j\right) = 0, \quad j = 0, 1, \cdots, p, \quad \phi = \left(\phi_1, \phi_2, \cdots, \phi_p\right).$$

But at which set of points x_j should the residual function equal zero? It is well known from the so called Chebyshev interpolation theorem[17] that the Chebyshev zeros minimize the maximum interpolation error. For this reason, one should use the Chebyshev nodes of the Chebyshev polynomial of order $p+1$ if we approximate the function by a Chebyshev polynomial of order p. This particular projection method is called Chebyshev collocation.

The procedure is easy to apply in small dimensional space where we can use the tensor product of the basis functions in order to approximate the policy functions. Let d denote the dimension of the state space ($d = 8$ for the

Table 3. Computational Time.

N	Galerkin	Collocation	Least Squares
4	4 min 49 sec (8)	0 min 10 sec (5)	11 min 4 sec (40)
6	2 h 53 min 44 sec (6)	3 min 4 sec (5)	
8	14 days 16 h 32 min (6)	30 min 47 sec (5)	

Notes: The solutions are computed with a Pentium III, 846 MHz. The number in brackets presents the number of iterations over ϕ in the modified Newton-Rhapson algorithm with line search (Galerkin, Collocation) and in the Quasi-Newton algorithm (Least Squares).

[16]The accuracy decreases with increasing number of the state space dimension. For a higher number of shocks e_t^i, the likelihood for a larger deviation from the steady-state values increases and we have to pick a wider interval for the approximation of the policy functions. Naturally, the goodness of fit decreases.

[17]See, e.g., [7], Theorem 6.7.2, p. 221.

4-country model with $N = 4$). In this case, we simply take the $m = (p+1)^d$ (transformed) Chebyshev nodes $\left(z_1^{i_1}, z^{i_2}, \cdots, z_d^{i_d}\right)$, where $z_1^{i_1}$ for example denotes the i_1-th Chebyshev zero in the first dimension of the state space and $i_1, i_2, \cdots, i_d = 1, \cdots, p+1$. This procedure, however, is no longer feasible if the dimension of the state space increases to values of $d = 6$ or above with current computer technology. For this reason, we have to use complete polynomials as approximating functions with a lower number of coefficients and a lower number of points at which we evaluate the residual function and try to set it equal to zero.

In the following, we suggest a very easy-to implement method for high-dimensional state spaces. Assume that the number of coefficients is equal to m. Just pick any m Chebyshev nodes and compute the solution. In very few cases, we find that this procedure does not converge and the modified Gauss-Newton algorithms fails. In this case, restart the program for another random pick of the Chebyshev nodes. The accuracy of this procedure is reported in Table 4 for the 4-country case.[18] Notice that the accuracy, at least for the relevant range of the state space, is less than in the case of Galerkin projection. However, the gain in speed is tremendous, especially if we consider the case for 8 countries with a corresponding number of 16 state variables. While Galerkin takes several days, our ad hoc collocation algorithm is able to compute a very accurate solution with a mean deviation of 0.016% within minutes.[19]

In summary, the proposed collocation procedure is less accurate than Galerkin. However, for the researcher who is time constrained, it may provide a valuable alternative. In addition, the solution from the Collocation projection can be used as an initial guess for the Galerkin method. By this procedure, the number of iterations over ϕ in the Newton Raphson algorithm can be reduced to

Table 4. Collocation projection.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t} \lvert R(x_t)\rvert$
0.01	0.18E−4	1	0.14E−4	0.14E−4
0.02	0.25E−4	10	0.14E−4	0.14E−4
0.05	0.35E−4	20	0.14E−4	0.13E−4
0.10	0.22E−3	30	0.15E−4	0.14E−4
0.15	0.71E−3	40	0.15E−4	0.14E−4
0.20	0.16E−2	50	0.15E−4	0.14E−4
0.30	0.41E−2	100	0.16E−4	0.15E−4

Notes: The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 10 seconds and the program needed 5 iterations over ϕ.

[18] Table 3 reports the maximum values from 10 runs of the collocation algorithm.

[19] We have also found this result to hold in other problems. In [1], Section 6.3.4 for example, we consider an equity-premium model with 3 states. The proposed ad hoc collocation procedure is found to perform equally well.

one or two and the computational time declines significantly. This savings in time comes at almost no costs as the additional programming is very little.

5. Least Squares

The least squares method minimizes the (weighted) sum of squared residuals:

$$\min_{\gamma} \int_{D_1} R(\phi, x)^2 \, dx. \tag{16}$$

One possible way to implement this method is to simply compute the sum of squared residuals over an n-dimensional grid of the state space. As a natural choice of the grid points in each dimension of the state space, we pick the Chebyshev nodes (Chebyshev integration uses constant weights). As, however, the state space dimension increases, we rather use monomial formulas to compute the integral (19). For this procedure, our results are discouraging if we use the log-linear solution as the initial guess. Typically, the mean residuals in our simulation of the economy are of the magnitude 10^{-1}, which, of course, is not satisfactory. Acoordingly, as the first important result of our analysis, we find that least squares do not work well if the initial solution is not close to the true solution. As the nature of the model in Section 2 is rather complex and number of parameters for the policy coefficients is rather large, the minimization problem (19) displays many local minima.[20]

We could only conceive a least squares algorithm with reasonably accurate results when we used the solution from our ad hoc collocation procedure as an initial guess for ϕ. The performance of the algorithm in this case is summarized in Table 5. As you can see, the accuracy is much higher than in the case of collocation projection and is comparable to the one with Galerkin projection. However, the computational time is much higher than with Galerkin as the

Table 5. Least Squares projection.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t} \lvert R(x_t)\rvert$
0.01	0.23E−5	1	0.61E−6	0.61E−6
0.02	0.38E−5	10	0.74E−6	0.66E−6
0.05	0.36E−4	20	0.96E−6	0.71E−6
0.10	0.33E−3	30	0.96E−6	0.65E−6
0.15	0.71E−3	40	0.96E−6	0.61E−6
0.20	0.13E−2	50	0.96E−6	0.60E−6
0.30	0.52E−2	100	0.96E−6	0.58E−6

Notes: The runtime for the Quasi-Newton algorithm with line search on a Pentium III, 846 MHz, amounted to 11 minutes 4 seconds and the program needed 40 iterations over ϕ.

[20]We also applied a genetic search algorithm in order to provide a more sophisticated guess for ϕ. In particular, we used a simple and fast genetic search algorithm based on [14] that is described in detail in [1], Chapter 8. However, results did not improve much and computational time became a limit for $N = 6$ already.

algorithm needs more iterations over ϕ in the minimization step.

6. Conclusions

Modern business cycle analyses are based on complex high-dimensional general equilibrium models. Monetary policy, for example, uses New Keynesian models such as those presented by [15] or [16] that introduce nominal frictions along with habit formation, costs of adjustment in capital accumulation, and variable capacity utilization. In these models, the number of state variables increases significantly, in particular if frictions in financial markets are added such as in [17] [18]. The curse of dimensionality is further aggravated if the model is extended to include heterogeneous firms or households as in the business cycle model of [2] [19] or [20]. These models, in particular in the presence of binding constraints or asymmetric behavior (for example, if households are subject to loss aversion), need to be solved with the help of non-linear methods such as projection methods, while the numerical accuracy of standard perturbation methods may prove to be insufficient.

We analyzed three projection methods and their ability to solve dynamic stochastic general equilibrium models that are characterized by a high dimension of the state space. Galerkin projection is shown to work remarkably well, accurately, and fast, if we use monomial integration formulas. Computational time only becomes a binding constraint for a state space dimension exceeding 15 - 20, depending on the current computer technology. We also suggested an alternative, rather ad hoc Chebyshev collocation method for incomplete polynomials which is found to work surprisingly well for the present problem and that is also found to be much faster than Galerkin. Even though this method is less accurate than Galerkin, it may help to economize on computational time as it provides a very good initial value so that the number of iteration steps in the Newton-Rhapson algorithm of the Galerkin method can be reduced substantially, typically to one or two iterations. Finally, we studied the least squares projection method. Our results are most discouraging. Without a very good initial value, least squares is unable to find an accurate solution. It takes a considerable effort to find a good starting value and, even then, the method is much slower than Galerkin projection. We, therefore, cannot recommend the use of least squares for the solution of dynamic stochastic general equilibrium models. More promising, instead, is our new collocation algorithm that picks the collocation nodes randomly.

References

[1] Heer, B. and Maußner, A. (2009) Dynamic General Equilibrium Models: Computation and Applications. 2nd Edition, Springer. https://doi.org/10.1007/978-3-540-85685-6

[2] Chen, Z. (2016) Time-to-Produce, Inventory, and Asset Prices. *Journal of Financial Economics*, **120**, 330-345. https://doi.org/10.1016/j.jfineco.2016.01.006

[3] Christiano, L.J. and Fisher, J.D.M. (2000) Algorithms for Solving Dynamic Models

with Occasionally Binding Constraints. *Journal of Economic Dynamics and Control*, **24**, 1179-1232. https://doi.org/10.1016/S0165-1889(99)00016-0

[4] Terry, S.J. (2017) Alternative Methods for Solving Heterogeneous Firm Models. *Journal of Money, Credit and Banking*, **49**, 1081-1111. https://doi.org/10.1111/jmcb.12414

[5] Bellman, R.E. (1957) Dynamic Programming. Princeton University Press, Princeton, NJ.

[6] Judd, K.L. (1992) Projection Methods for Aggregate Growth Models. *Journal of Economic Theory*, **58**, 410-452. https://doi.org/10.1016/0022-0531(92)90061-L

[7] Judd, K.L. (1998) Numerical Methods in Economics. MIT Press, Cambridge, MA.

[8] Heer, B. and Maußner, A. (2008) Computation of Business Cycle Models: A Comparison of Numerical Methods. *Macroeconomic Dynamics*, **12**, 641-663. https://doi.org/10.1017/S1365100508070363

[9] Pichler, P. (2011) Solving the Multi-Country Real Business Cycle Model Using a Monomial Rule Galerkin Method. *Journal of Economic Dynamics and Control*, **35**, 240-251. https://doi.org/10.1016/j.jedc.2010.09.009

[10] Judd, K. and Gaspar, J. (1997) Solving Large-Scale Rational-Expectations Models. *Macroeconomic Dynamics*, **1**, 45-75.

[11] Krueger, D. and Kubler, F., (2004) Computing Equilibrium in OLG Models with Stochastic Production. *Journal of Economic Dynamics and Control*, **28**, 1411-1436. https://doi.org/10.1016/S0165-1889(03)00111-8

[12] Stroud, A.H. (1971) Approximate Calculation of Multiple Integrals. Academic Press, New York.

[13] Jin, H. and Judd, K. (2002) Perturbation Methods for General Dynamic Stochastic Equilibrium. Stanford University, Stanford.

[14] Duffy, J. and McNelis, P.D. (2002) Approximating and Simulating the Stochastic Growth Model: Parameterized Expectations, Neural Networks, and the Genetic Algorithm. *Journal of Economic Dynamics and Control*, **25**, 1273-1303. https://doi.org/10.1016/S0165-1889(99)00077-9

[15] Christiano, L.J. and Eichenbaum, M. and Evans, C.L. (2005) Nominal Rigidities and the Dynamic Effects of a Shock to Monetary Policy, Journal of Political Economy, **113**, 1-45. https://doi.org/10.1086/426038

[16] Smets, F. and Wouters, R. (2003) An Estimated Dynamic Stochastic General Equilibrium Model of the Euro Area. *Journal of the European Economic Association*, **1**, 1123-1175. https://doi.org/10.1162/154247603770383415

[17] Carlstrom, C.T. and Fuerst, T.S. (1997) Agency Costs, Net Worth, and Business Fluctuations: A Computable General Equilibrium Analysis, American Economic Review, **87**, 893-910.

[18] Carlstrom, C.T. and Fuerst, T.S. (2007) Asset Prices, Nominal Rigidities, and Monetary Policy. *Review of Economic Dynamics*, **10**, 256-275. https://doi.org/10.1016/j.red.2006.11.005

[19] Heer, B. and Maußner, A. (2012) The Burden of Unanticipated Inflation: Analysis of an Overlapping Generations Model with Progressive Income Taxation and Staggered Prices. *Macroeconomic Dynamics*, **16**, 278-308. https://doi.org/10.1017/S1365100510000490

[20] Heer, B. and Scharrer, C. (2018) The Age-Specific Burdens of Short-Run Fluctuations in Government Spending. *Journal of Economic Dynamics and Control*, **90**, 45-75. https://doi.org/10.1016/j.jedc.2018.01.041

Appendix

In this appendix we provide the results for the 6-country and 8-country model in the symmetric case.

Table S1. Galerkin projection, 6 countries.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t}\lvert R(x_t)\rvert$
0.01	0.32E−5	1	0.15E−6	0.15E−6
0.02	0.58E−5	10	0.38E−6	0.22E−6
0.05	0.15E−4	20	0.13E−5	0.37E−6
0.10	0.15E−3	30	0.20E−5	0.67E−6
0.15	0.82E−3	40	0.20E−5	0.77E−6
0.20	0.17E−2	50	0.20E−5	0.69E−6
0.30	0.18E−2	100	0.20E−5	0.70E−6

Notes: The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 2 hours 53 minutes 44 seconds and the program needed 6 iterations over ϕ.

Table S2. Collocation projection, 6 countries.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t}\lvert R(x_t)\rvert$
0.01	0.26E−4	1	0.22E−4	0.22E−4
0.02	0.30E−4	10	0.23E−4	0.20E−4
0.05	0.29E−4	20	0.23E−4	0.21E−4
0.10	0.19E−3	30	0.23E−4	0.22E−4
0.15	0.69E−3	40	0.23E−4	0.22E−4
0.20	0.11E−2	50	0.23E−4	0.22E−4
0.30	0.15E−2	100	0.23E−4	0.20E−4

Notes: The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 3 minutes 4 seconds and the program needed 5 iterations over ϕ.

Table S3. Galerkin projection, 8 countries.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t}\lvert R(x_t)\rvert$
0.01	0.48E−4	1	0.39E−5	0.39E−5
0.02	0.10E−3	10	0.27E−4	0.13E−4
0.05	0.19E−2	20	0.40E−4	0.16E−4
0.10	0.36E−2	30	0.40E−4	0.14E−4
0.15	0.14E−1	40	0.40E−4	0.15E−4
0.20	0.37E−1	50	0.40E−4	0.16E−4
0.30	0.14E−0	100	0.53E−4	0.19E−4

Notes: The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 14 days 16 hours 32 minutes and the program needed 6 iterations over ϕ.

Table S4. Collocation projection, 8 countries.

Accuracy Test 1		Accuracy Test 2		
r	$\max_i \lvert R(x_i)\rvert$	Period t	$\max_t \lvert R(x_t)\rvert$	$\frac{1}{t}\sum_{i=1}^{t}\lvert R(x_t)\rvert$
0.01	0.17E−3	1	0.15E−3	0.15E−3
0.02	0.24E−3	10	0.15E−3	0.13E−3
0.05	0.70E−2	20	0.15E−3	0.13E−3
0.10	0.49E−1	30	0.15E−3	0.12E−3
0.15	0.17E−0	40	0.15E−3	0.12E−3
0.20	0.22E−0	50	0.15E−3	0.11E−3
0.30	0.55E−0	100	0.15E−3	0.11E−3

Notes: The runtime for the modified Newton-Rhapson algorithm with line search on a Pentium III, 846 MHz, amounted to 30 minutes 47 seconds and the program needed 5 iterations over ϕ.