Solving DSGE Models without a Grid

Oren Levintal

Bar-Ilan University
oren.levintal@biu.ac.il

October 20, 2013

Abstract

This paper presents a global solution method to DSGE models, which does not depend on a grid and hence does not suffer from the curse of dimensionality. The method enables to approximate the Taylor series of the policy function at any arbitrary point of the state space. Once the Taylor series is approximated at a given point, the constant term of the series provides the model solution at that point. Since the solution is not based on a grid, the computational costs are significantly lower compared to grid-based methods, because the model is solved only at points of interests (e.g. along a simulation path). Accuracy is high, compared to other methods, and it improves significantly by discretizing time into short periods.

Keywords: Taylor projection, DSGE, Taylor series, projection, perturbation, approximation, computational methods, numerical methods, tensors

JEL classification: C61,C68,E12,E13,E17.
1 Introduction

Methods for solving Dynamic Stochastic General Equilibrium (DSGE) models can be classified into two broad groups. The first group includes perturbation around a deterministic steady state, which is a local solution method. Perturbation solutions are easily implemented. They tend to be very accurate around the deterministic steady state in models with low volatility. However, their accuracy can deteriorate significantly away from the deterministic steady state. The second group includes global solution methods such as projection and value function iteration. These methods are in general very accurate on a large domain of the state space (Aruoba, Fernández-Villaverde and Rubio-Ramírez 2006). Their main drawback is that they require a grid. The size of the grid grows exponentially with the number of state variables, giving rise to the "curse of dimensionality." Consequently, global solution methods are very slow, and computation time can easily reach hours for large models (Kollmann, Maliar, Malin and Pichler 2011).

This paper presents a global solution method that does not depend on a grid and hence does not suffer from the curse of dimensionality. The method is called "Taylor projection", because it uses projection techniques to approximate the Taylor series of the policy function at any arbitrary point of the state space. The approximation

1A survey of the main methods can be found in Marimon and Scott (1999), Judd (1998) and Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006).
3Recent papers suggest several ways to reduce the size of the required grid, thereby weakening the curse, see Judd, Maliar, Maliar and Valero (2013), Judd, Maliar and Maliar (2012), Maliar, Maliar and Judd (2011), Malin, Krueger and Kubler (2011). Other papers propose ways to speed up the required computations at each node, e.g. see Barillas and Fernández-Villaverde (2007) and the references therein.
exploits local information on the Taylor series that comes from the model equations and the derivative conditions, evaluated at several points within a small neighbourhood. The number of points necessary for approximation is very small (few points only), and in some cases a single point is sufficient to get accurate results.

In contrast to perturbation methods that provide the Taylor series at the deterministic steady state, the proposed method enables to approximate the Taylor series at any arbitrary point of the state space. In this regard, ”Taylor projection” is a global solution method, because its accuracy does not depend on a specific point of the state space. The application of the method is fundamentally different from perturbation. Specifically, to approximate the policy function $g(x)$ at state $x_0$, we approximate the Taylor series of $g$ about $x_0$. The constant term of this series provides the approximated value of $g(x_0)$. When we want to approximate $g$ at a new state $x_1$, we do not use the Taylor series about $x_0$, but rather approximate a new Taylor series centered at $x_1$. The new constant term is the approximated value of $g(x_1)$. Hence, the coefficient of interest is the constant term of the Taylor series, whereas the other coefficients are auxiliary parameters.

Since Taylor projection does not require a grid, the curse of dimensionality does not apply. The solution at any given point of the state space is independent of other points. As a result, the policy function can be approximated at points of interests, rather than on an entire grid. For instance, to perform a simulation it is sufficient to solve the model along the simulation path. This property reduces computational costs to levels that are closer to perturbation than to grid-based methods. For instance, a standard Neo-Classical model is solved at one point of the state space.
in 0.1-0.8 seconds (depending on specification). Simulating the model for 10 periods would take about 1-8 seconds.

In terms of computational complexity, the algorithm requires to solve a nonlinear problem. However, a good initial guess is always available (provided that the model has a deterministic steady state), so the solution is obtained in few iterations only. Thus, the implementation of the method is straightforward and does not require sophisticated solution techniques.

The accuracy of Taylor projection is tested on the Neo-Classical model of Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006), and compared to other solution methods studied in their paper. Overall, the accuracy is in line with the existing grid-based methods, and in some cases it may be even better. Furthermore, it is shown that accuracy can increase by orders of magnitude when time is discretized into short periods. For instance, when the quarterly model of Arouba et al (2006) is calibrated as a daily model, Euler errors can be hundred times smaller (depending on specification). Hence, the computational gain that stems from the grid independence does not come at the cost of lower accuracy. On the contrary, Taylor projection enables to increase accuracy, simply by recalibrating the model.

The theory of Taylor projection builds on the special structure of the functional equations that emerge in DSGE models. The implicit functions in these equations are evaluated at multiple points. These points usually lie in a small neighbourhood. If the implicit functions are analytic, they can be replaced with Taylor series that converge in this neighbourhood. This observation stands at the core of Taylor projection and yields all the results of the paper. The theory and the conditions for its application
are presented in chapter 2. Since the method is new, it is first presented on a simple functional equation, and only later, in chapter 3, applied to a full DSGE model. Chapter 4 implements the method on a Neo-Classical growth model, and compares it to other solution methods. Chapter 5 shows that accuracy depends on the way time is discretized. Chapter 6 discusses computational costs, and chapter 7 concludes.

2 Taylor Projection

This chapter lays down the principles of the Taylor projection method. The first section discusses the type of functional equations that can be solved by this method. The second section develops the theory and discusses the condition for its application. The third section provides the solution algorithm and the fourth section implements the algorithm on a simple example. For exposition clarity, the chapter focuses on univariate implicit functions. The multivariate case is delayed to the next chapter which applies the method to a full DSGE model.

2.1 The functional equation

DSGE models generate functional equations with a special structure, which is exploited by the proposed method. To demonstrate this structure, it is easier to start with a simple example. Consider first a standard functional equation such as:

\[ g(x) - \sqrt{x} = 0. \] (1)
This equation defines the function \( g(x) \). Now, compare (1) with the following functional equation:

\[
g(x) + g(0.5x) - \sqrt{x} - \sqrt{0.5x} = 0.
\] (2)

The solution of these two examples is \( g(x) = \sqrt{x} \), but there is an important difference. In example (1), the function \( g \) is evaluated at a unique point \( x \). By contrast, in example (2) the function \( g \) is evaluated at two points: \( x \) and \( 0.5x \). These points will be called "the defining points of \( g \)" because \( g \) is defined by these two points. The collection of all defining points will be called "the defining set of \( g \)" and denoted \( D \).

By this terminology, the difference between examples (1) and (2) is that in example (1) there is only one defining point whereas in (2) there are two defining points. This difference is important, because (1) can be easily solved for a given \( x \) but (2) cannot. As shown below, DSGE models generate functional equations with multiple defining points.

To generalize (2), let \( u^1(x), u^2(x), \ldots, u^k(x) \) denote \( k \) known functions of \( x \), where \( u^k : \mathbb{R} \to \mathbb{R} \forall k \), and let \( f : \mathbb{R}^{k+1} \to \mathbb{R} \) denote a known function with \( k + 1 \) arguments. All functions are infinitely differentiable with respect to all their arguments. A functional equation in the unknown function \( g : \mathbb{R} \to \mathbb{R} \) with multiple defining points is defined by:
\[ f(v(x)) = 0, \quad (3) \]

\[ v(x) = \left( g(u^1(x)), g(u^2(x)), \ldots, g(u^k(x)), x \right), \]

where \( v(x) \) is a vector of size \( k + 1 \) that contains the arguments of \( f \) (which are functions of \( x \)). In this functional equation the implicit function \( g \) is evaluated at \( u^1(x), \ldots, u^k(x) \). Hence, the defining set of \( g \) is:

\[ D(x) = \{ u^1(x), u^2(x), \ldots, u^k(x) \}. \quad (4) \]

The notation \( D(x) \) denotes that the defining set may depend on \( x \), as in the above examples.

Lucas (1978) asset pricing model is an example of (3). Under logarithmic utility, the Euler condition is:

\[
\frac{1}{x_t} = \beta E_t \left( \frac{p_{t+1} + x_{t+1}}{p_t} \cdot \frac{1}{x_{t+1}} \right),
\]

where \( x_t \) and \( p_t \) denote dividends and the asset price ex dividends in period \( t \), \( \beta \) is the time discount factor, and \( E_t \) is the expectation operator. The state variable is \( x_t \) which is determined exogenously. Suppose that \( x_t \) follows an AR process:

\[ x_{t+1} = \rho x_t + \epsilon_{t+1}. \]

The solution to the model is the function \( p_t = g(x_t) \), where \( g \)
satisfies:

$$\frac{1}{x_t} = \beta E_t \left( g \left( \rho x_t + \epsilon_{t+1} \right) + \rho x_t + \epsilon_{t+1} \right) \cdot \frac{1}{\rho x_t + \epsilon_{t+1}}.$$ 

In this functional equation the implicit function $g$ is evaluated at $x_t$ and at $\rho x_t + \epsilon_{t+1}$ for all values of $\epsilon_{t+1}$ with nonzero probability. Hence, Lucas (1978) is a functional equation with multiple defining points as defined by (3).

The defining set (4) contains known functions of $x$. This is not a necessary condition. A defining point can be also an implicit function of $x$, as in the following example:

$$g \left( g \left( x \right) \right) + g \left( 0.5 x \right) - x^{\frac{1}{4}} - \sqrt{0.5 x} = 0.$$ 

The solution is again $g \left( x \right) = \sqrt{x}$. Here, $g \left( x \right)$ is a defining point because in the first term on the LHS $g$ is evaluated at $g \left( x \right)$. This type of functional equations appears frequently in DSGE models. For instance, in a deterministic growth model with capital $k$, fixed labour supply, logarithmic utility and full depreciation (e.g. Brock and Mirman 1972), consumption is equal to $f \left( k_t \right) - k_{t+1}$ and the Euler condition is:

$$\frac{1}{f \left( k_t \right) - k_{t+1}} = \frac{\beta f_k \left( k_{t+1} \right)}{f \left( k_{t+1} \right) - k_{t+2}}.$$

where $f$ and $f_k$ are the production function and the marginal product of capital. The
solution of the model is the function \( k_{t+1} = g(k_t) \), where \( g \) is an implicit function that satisfies the Euler condition:

\[
\frac{1}{f(k_t) - g(k_t)} = \frac{\beta f_k(g(k_t))}{f(g(k_t)) - g(g(k_t))}.
\]

The term \( g(g(k_t)) \) on the RHS implies that \( g \) is evaluated at point \( g(k_t) \). Hence, the implicit function \( g(k_t) \) is itself a defining point. For exposition clarity, this chapter assumes that all the defining points are known functions of \( x \), but none of the results depend on this assumption. The application of the method to a full DSGE model is addressed in the next chapter.

### 2.2 Convergence assumption

Consider the functional equation (3) with the defining set (4) and the solution \( g(x) \). Let \( G_0, G_1, \ldots \) be the coefficients of the infinite Taylor series of \( g \) about \( c \). Assume that \( g \) is analytic at \( c \), so the Taylor series of \( g \) about \( c \) converges to \( g \) in the neighbourhood of \( c \). The following is an assumption on \( x \) and \( c \):

**Assumption. (Convergence)** Given \( x \) and \( c \), the defining set of \( g \), given in (4), lies within the radius of convergence of the Taylor series of \( g \) about \( c \). Formally:

\[
g(z) = \sum_{n=0}^{\infty} G_n (z - c)^n, \quad \forall z \in D(x).
\]
When this assumption holds (given \( x \) and \( c \)), the neighborhood of convergence of the Taylor series about \( c \) contains the defining set \( D(x) \). Namely, for each point in \( D(x) \) the function \( g \) is equal to its Taylor series about \( c \).

Substituting (5) in (3) yields:

\[
f(v(x)) = 0, \quad (6)
\]

\[
v(x) = \left( \sum_{n=0}^{\infty} G_n \left( u^1(x) - c \right)^n, \ldots, \sum_{n=0}^{\infty} G_n \left( u^k(x) - c \right)^n, x \right). \quad (7)
\]

Note that the implicit function \( g(x) \) is replaced with a parameterized function, whose parameters are the Taylor coefficients. If (5) holds in the neighborhood of \( x \), then the Taylor coefficients can be approximated by projection methods, because (6)-(7) hold in the neighborhood of \( x \). Since the Taylor series converges, a finite series can be estimated with a finite sample of points near \( x \) (see the next section). This result is the basis of the Taylor projection method.

The convergence assumption is crucial. It requires two conditions. First, that the function \( g \) is analytic at \( c \). Second, that the Taylor series of \( g \) about \( c \) converges for all the defining points (given \( x \)). The analyticity requirement usually holds in economic applications, because functions are in many cases smooth. The second condition requires that the implicit function is evaluated at points that are sufficiently close to one another. The economic interpretation of this requirement for dynamic models is that the state variables in period \( t + 1 \) are sufficiently close to their period \( t \) values. Namely, the model evolves slowly and gradually. This assumption is likely to
hold in many economic applications, because economic dynamics tend to be gradual. Moreover, it is shown in chapter 5 that this property can be controlled by proper calibration. The convergence assumption may fail to hold if the model generates big jumps in the state variables.

Additional information on the Taylor coefficients can be obtained by differentiating (6) with respect to $x$ an infinite number of times:

$$0 = f = \frac{\partial f}{\partial x} = \frac{\partial^2 f}{\partial x^2} = \frac{\partial^3 f}{\partial x^3} = \ldots$$

(8)

Given $x$ and $c$, this is a system of an infinite number of equations in the unknown variables $G_0, G_1, \ldots$. The solution to this system is the true Taylor coefficients. One particular example of (8) is known as "Taylor approximation" (Judd 1998, p. 196). Taylor approximation applies to functional equations of the form:

$$f \left( g(x), x \right) = 0,$$

(9)

where the defining set contains only one defining point. For this case, evaluating (8) at $x = c$ (i.e. at the center of the Taylor series) yields the familiar recursive procedure for calculating the Taylor series of $g(x)$ about $c$. The constant term of the Taylor series is calculated through $f = 0$, and the high order coefficients are obtained recursively through the zero derivative conditions. Note that Taylor approximation satisfies the convergence condition (5), because the Taylor series is centered at the
single defining point so all the terms of the Taylor series are zero except the constant.

System (8) is more general than Taylor approximation, because it applies to
functional equations with multiple defining points. Note also that in Taylor approx-
imation the center of the Taylor series must be equal to the defining point, whereas
in (8) it is not necessarily so. However, there is an important technical difference.
Taylor approximation is a special case where (8) is solved recursively, which is very
easy. In the general case, (8) needs to be solved simultaneously. Since the number
of equations is infinite, this is not feasible. However, due to the convergence of the
Taylor series, high order terms are negligible. This enables us to approximate the
solution through a finite system of equations.

2.3 Approximating the Taylor Series

The algorithm for approximating the solution to (8) is built on projection ideas,
hence the name "Taylor projection". To approximate a Taylor series about \( c_0 \), the
convergence assumption (5) must hold in the neighbourhood of some \( x = x_0 \). Under
this assumption, system (8) also holds in the neighbourhood of \( x_0 \). Evaluating the
system at \( x_0 \) and reducing it to the first \( N + 1 \) conditions gives:

\[
f(x_0) = \frac{\partial f(x)}{\partial x} \bigg|_{x_0} = \frac{\partial^2 f(x)}{\partial x^2} \bigg|_{x_0} = \ldots = \frac{\partial^N f(x)}{\partial x^N} \bigg|_{x_0} = 0. \tag{10}
\]

Express the infinite Taylor series of \( g(x) \) about \( c_0 \) as the sum of an \( N' \)th order finite
series and an error term:
\[ g(x) = \sum_{n=0}^{N} G_n (x - c_0)^n + \zeta(x), \quad (11) \]
\[ \zeta(x) = \sum_{n=N+1}^{\infty} G_n (x - c_0)^n. \]

As shown in detail in chapter 3, system (10) contains the error function \( \zeta(x) \) and its \( N \)'th derivatives evaluated at each point of the defining set:

\[ \zeta(z) = \sum_{n=N+1}^{\infty} G_n (z - c_0)^n, \quad (12) \]
\[ \zeta_z(z) = \sum_{n=N+1}^{\infty} nG_n (z - c_0)^{n-1} \]
\[ \zeta_{zz}(z) = \sum_{n=N+1}^{\infty} n(n-1)G_n (z - c_0)^{n-2} \]
\[ \vdots \]
\[ \zeta_{zzN}(z) = \sum_{n=N+1}^{\infty} n(n-1)\cdots(n-N+1)G_n (z - c_0)^{n-N} \]
\[ \forall z \in D(x_0), \]

where \( \zeta_z(z), \zeta_{zz}(z) \) and \( \zeta_{zzN}(z) \) denote the first, second and \( N \)'th derivatives of \( \zeta(z) \) with respect to \( z \), and \( D(x_0) \) is the defining set at \( x = x_0 \).

Note that the Taylor error \( \zeta(z) \) and all its derivatives are power series. It follows from (5) and (11) that \( \zeta_z(z), \ldots, \zeta_{zzN}(z) \) are finite for all \( z \in D(x_0). \)

\[ ^{\text{If any of these terms is infinite at some } z \in D(x_0) \text{ then (11) must also be infinite at that point,}} \]
some $M \geq 0$ we can approximate (12) by finite power series:

\[
\zeta(z) \approx \sum_{n=N+1}^{N+M} G_n (z - c_0)^n
\]

\[
\zeta_z (z) \approx \sum_{n=N+1}^{N+M} nG_n (z - c_0)^{n-1}
\]

\[
\zeta_{zz} (z) \approx \sum_{n=N+1}^{N+M} n (n - 1) G_n (z - c_0)^{n-2}
\]

\[
\vdots
\]

\[
\zeta_{z^N} (z) \approx \sum_{n=N+1}^{N+M} n (n - 1) \cdots (n - N + 1) G_n (z - c_0)^{n-N}
\]

\[
\forall z \in D(x_0)
\]

These assumptions reduce the number of unknown Taylor coefficients in (10) from infinite to $N + M + 1$. The number of equations is $N + 1$ so the system is under identified for $M \geq 1$. To identify the Taylor coefficients we need at least $M$ more conditions. Since (5) holds in the neighbourhood of $x_0$, the additional conditions are generated by evaluating (10) at other points near $x_0$.

To present the algorithm, let $\{x_1, \ldots, x_I\}$ denote a set of points close to $x_0$ (including $x_0$) and let $R_i$ denote a column vector of the residuals of (10) evaluated at $x_i$, for some $G_0, G_1, \ldots, G_{N+M}$.

contradicting (5).
\[ R_i = \left( f(x_i), \frac{\partial f(x)}{\partial x} \bigg|_{x_i}, \ldots, \frac{\partial^N f(x)}{\partial x^N} \bigg|_{x_i} \right). \]  

(14)

Under (13), the true Taylor coefficients satisfy:

\[ R_i \approx 0 \quad \forall i = \{1, 2, \ldots, I\}. \]

The approximated Taylor coefficients (denoted by *) are obtained by minimizing the squared residuals:

\[ G^*_0, G^*_1, \ldots, G^*_N = \arg \min \sum_{i=1}^{I} R^*_i R_i. \]  

(15)

The asymptotic properties of (15) are derived from the asymptotic properties of the system errors, which are implicit in (13). These errors are defined by the difference between the infinite series (12) and the finite series (13). Let \( \eta_k \) denote the \( k \)'th order system error. Specifically, \( \eta_0 \) is the difference between the true \( \zeta(z) \) and the finite series approximation in (13). Similarly, \( \eta_1 \) is the error of \( \zeta(z) \) and \( \eta_N \) is the error of \( \zeta_{z,N} \). The complete system errors are:

\[ \eta_k (z) = \sum_{n=N+M+1}^{\infty} \frac{g_n}{(n-k)!} (z - c_0)^{n-k} \]  

\[ \forall z \in D(x_i), i \in \{1, 2, \ldots, I\}, k \in \{0, 1, \ldots, N\}, \]

(16)
where $g_n = n! G_n$ is the $n$’th derivative of $g$ at $c_0$. It follows that the approximated coefficients converge to the true Taylor coefficients as $M$ increases:

**Theorem 1.** *(Convergence in $M$)* The solution to the least squares problem (15) converges to the true Taylor coefficients as $M$ goes to infinity.

*Proof.* Define $I = M+a$ where $a$ is a positive integer, so that (15) is overidentified for each $M$, and replace the target function in (15) with $\frac{1}{T} \sum_{i=1}^{I} R'_i R_i$. As $M$ goes to infinity (holding $N$ constant), the system error $\eta_k$ goes to zero for each $z$ in the convergence domain of $g$ and for each $k \in \{0, 1, \ldots, N\}$. Hence, the residuals $R_i$ evaluated at the true Taylor coefficients go to zero for each point in the convergence domain, including the sample $x_i = \{x_1, \ldots, x_I\}$. At the limit, the true Taylor coefficients satisfy $R_i = 0$ at each point in the convergence domain, so the minimum of $\frac{1}{T} \sum_{i=1}^{I} R'_i R_i$ is obtained at the true Taylor coefficients. ■

Convergence in $N$ requires a stronger assumption. Note that when $N$ increases (holding $M$ constant) $R_i$ does not necessarily go to zero, because new errors of higher order are added to (16). For instance, incrementing $N$ to $N+1$ introduces an error of order $N+1$, denoted $\eta_{N+1}$, which does not exist for $N$ order system. Convergence in $N$ requires that these higher order errors converge to zero.

In multivariate problems, memory and time constraints impose an upper bound on $N+M$, which is fairly low. Hence, asymptotic properties are less relevant, and the efficient choice of $N$ and $M$ is more important. Increasing $M$ is much easier than $N$, because obtaining high derivative conditions becomes computationally very costly for large $N$. Moreover, high derivative conditions add higher order errors to the
system, which may not be negligible. This adds noise that may reduce the accuracy of the result. On the other hand, dropping the zero derivative conditions entirely ($N = 0$) results in a loss of information that could be valuable for the approximation. Nevertheless, the results below suggest that projections of order $N = 0$ may be as accurate as projections with $N > 0$, holding $N + M$ constant.

The least squares problem (15) is a small residual problem. For these cases, Judd (1998, p. 119) recommends the Levenberg-Marquardt algorithm. Once a solution is obtained for some $x = x_0$, it can serve as a good initial guess for any point $x_1$ which is close to $x_0$. If we wish to change the center of the Taylor series from $c_0$ to $c_1$, then the initial guess is obtained by shifting the center of the solution at $x_0$ to $c_1$. Note that the center of the Taylor series can be different from $x_0$, though in general choosing $c = x$ is computationally easier.

In order to generate the first initial guess, the easiest way is to start at a point where the defining set contains a single defining point. At this point we can calculate the true Taylor series recursively by Taylor approximation. For instance, in a deterministic Neo-Classical growth model, the steady state is a point with a single defining point, and the true Taylor series can be easily calculated. The stochastic case requires some adjustment, which is explained in chapter 3. In general, a good initial guess is available for any model that has a deterministic steady state. Hence, the solution of the least squares problem is obtained with only few iterations.

The following algorithm summarizes the Taylor projection method:

**Algorithm. (Taylor Projection)** To approximate the Taylor series of $g(x)$, pick $x_0$, $c_0$, $N$ and $M$ such that (13) holds in the neighbourhood of $x_0$. Evaluate (14) at
multiple points around \(x_0\) to get an overidentified system, and choose \(G_0, G_1, \ldots, G_{N+M}\) that solve (15).

As in general projection methods (Judd 1992), Taylor projection uses a basis function to approximate an unknown function. The difference from general projection is that we know the basis function because our target function is a Taylor series. To approximate the coefficients we use local information that comes from the functional equation and the derivative conditions, evaluated in a small neighbourhood around \(x_0\). The main difference from standard projection is that the number of points required to identify the system is very small, because we are interested only in the local behaviour of the function. If \(M = 0\) is a reasonable assumption the system is exactly identified at one point.

2.4 A simple example

This section presents a simple example that demonstrates the implementation of the Taylor projection method. The functional equation to be solved here is:

\[
g(x) + g(1.5x) - x^{0.25} - (1.5x)^{0.25} = 0.\tag{17}
\]

The defining set is \(D(x) = \{x, 1.5x\}\), and the solution is the function \(g(x) = x^{0.25}\). Suppose we are interested in approximating the Taylor series of the solution about \(c_0 = 1\). The convergence of this particular Taylor series was studied by Judd (1998, p.198), who showed that the radius of convergence is 1. The first five coefficients of
the series are \( \{1, 0.25, -0.094, 0.055, -0.038\} \).

The implementation of Taylor projection requires four steps: (1) obtaining the \( N \) derivative conditions; (2) constructing \( N + M \) order Taylor series about \( c_0 \) and differentiating them \( N \) times; (3) substituting the Taylor series and their derivatives in the system; (4) evaluating the system at multiple points and solving the least squares problem.

Consider Taylor projection of order \( (N, M) = (1, 1) \). The first step is to differentiate (17) once (because \( N = 1 \)):

\[
g_x(x) + 1.5g_x(1.5x) - 0.25x^{-0.75} - 0.375(1.5x)^{-0.75} = 0. \tag{18}
\]

In the second step we construct a Taylor series of order \( N + M = 2 \) which approximates \( g(x) \). For consistency with the notation of the previous sections, the Taylor series is presented as a sum of an \( N \)'th order Taylor series and an \( M \)'th order error term:

\[
g(x) = G_0 + G_1(x - c_0) + \zeta(x) \tag{19}
\]

\[
\zeta(x) \approx G_2(x - c_0)^2
\]

The derivative of the Taylor series is:
\begin{align*}
g_x(x) &= G_1 + \zeta_x(x) \tag{20} \\
\zeta_x(x) &\approx 2G_2(x - c_0)
\end{align*}

The third step is to substitute (19) and (20) in (17) and (18):

\begin{align*}
2G_0 + G_1 (2.5x - 2c_0) + G_2 \left[ (x - c_0)^2 + (1.5x - c_0)^2 \right] - x^{0.25} - (1.5x)^{0.25} &\approx 0 \tag{21} \\
2.5G_1 + G_2 (6.5x - 5c_0) - 0.25x^{-0.75} - 0.375 (1.5x)^{-0.75} &\approx 0. \tag{22}
\end{align*}

This system has two conditions and three unknowns: $G_0, G_1, G_2$. In order to solve it, we need to evaluate the system at several points and find the coefficients that minimize the squared residuals. This particular problem is easy because the system is linear in the coefficients.

Table 1 presents the results for different $N, M$ values. These results were obtained by evaluating the system at 4 points in the interval $[.79, .81]$ and applying linear least squares. Since the defining set is $D(x) = \{x, 1.5x\}$, the distance of the defining points from the Taylor center ($c_0 = 1$) is approximately 0.2 for $x \in [.79, .81]$. This distance is smaller than the radius of convergence (which is 1), so the convergence assumption holds. The table presents the $\log_{10}$ of the approximation errors, defined by the absolute difference between the true and the approximated coefficients. In general, accuracy increases with $N + M$ except for few cases. Interestingly, accuracy does not change with $N$ and $M$ separately, holding $N + M$ constant. For instance,
order $N, M = 0, 3$ is as accurate as $N, M = 2, 1$. Finally, the constant term ($G_0$) is much more accurate than the high order terms. This is a good result because the constant term is the main parameter of interest.

3 Application to DSGE Models

This chapter applies the Taylor projection method to a full DSGE model. This requires to extend (3) to multivariate implicit functions and to allow the defining points to depend on the implicit functions. The first section of this chapter presents the DSGE model using the notation of Schmitt-Grohé and Uribe (2004) with some modifications. The second section shows how to form the system of conditions. The third section constructs the Taylor series, the Taylor errors and the required derivatives. Section four evaluates the system and discusses how to reduce the dimensions of the least squares problem by exploiting symmetry of mixed derivatives. The last section shows how to obtain the initial guess. Results are presented in the next chapter.

3.1 The model

The state variables are denoted by vector $x_t = \begin{pmatrix} x^1_t \\ x^2_t \end{pmatrix}$, where $x^1_t$ denotes a vector of $n^1_x$ pre-determined variables and $x^2_t$ denotes a vector of $n^2_x$ exogenous variables. The total size of $x_t$ is $n_x = n^1_x + n^2_x$. The evolution of $x^2_t$ is:
\[ x_{t+1}^2 = \Phi (x_t, \epsilon_{t+1}), \quad (23) \]

where \( \Phi \) is a known function and \( \epsilon_{t+1} \) is a vector of \( n_{\epsilon} \) independent shocks. The model is defined by a set of \( n_f = n_y + n_x^1 \) expectational conditions:

\[ E f (y_{t+1}, y_t, x_{t+1}, x_t) = 0. \quad (24) \]

Note that (23) is not included in (24), which is different from the formulation of Schmitt-Grohé and Uribe (2004).

The solution to the model is the policy functions \( g \) and \( \tilde{h} \), which provide the optimal response of \( y_t \) and \( x_{t+1}^1 \), given the state of the economy:

\[ y_t = g (x_t), \quad (25) \]
\[ x_{t+1}^1 = \tilde{h} (x_t). \]

Let the function \( h (x_t, \epsilon_{t+1}) \) denote the evolution of all the state variables (\( x_t^1 \) and \( x_t^2 \)), that is:
With this notation, it should be clear that only the first $n^1_x$ rows of $h$ are unknown.

To reduce notation, time subscripts are dropped and next period variables are denoted by superscript $p$. In addition, let $v = (y^p, y, x^p, x)$ denote a vector of size $n_v = 2(n_y + n_x)$ that contains the model variables, and write (24)-(26) as follows:

\[
Ef(v) = 0, \quad (27)
\]
\[
v = \left( g(x^p) , g(x) , x^p , x \right), \quad (28)
\]
\[
x^p = h(x, e^p). \quad (29)
\]

Note that the function $g$ is evaluated at $x$ and $x^p$, so the defining set of $g$ contains multiple points. $x^p$ is given by the implicit function $h(x, e^p)$, hence the defining set contains implicit functions. Note that in a stochastic model the defining set contains $h(x, e^p)$ for each $e^p$ with nonzero probability.

To distinguish between the value of $g$ at $x$ and at $x^p$, I use superscript $p$ in the following way:
\[ g \equiv g(x), \quad (30) \]
\[ g^p \equiv g(x^p). \quad (31) \]

Similarly, the derivative of \( g \) with respect to \( x \), evaluated at \( x \) and \( x^p \), are denoted:

\[ g_x \equiv \frac{\partial g(x)}{\partial x} \bigg|_x \quad (32) \]
\[ g^p_x \equiv \frac{\partial g(x^p)}{\partial x} \bigg|_{x^p} \quad (33) \]

Higher derivatives are denoted in a similar way.

### 3.2 The system of conditions

The first step of the Taylor projection is to form the system of conditions (10). This is done by differentiating (27) \( N \) times with respect to \( x \) and letting the derivatives be equal to zero:

\[ f = 0, \quad \frac{\partial f}{\partial x} = 0, \quad \frac{\partial^2 f}{\partial x^2} = 0, \ldots, \quad \frac{\partial^N f}{\partial x^N} = 0. \quad (34) \]

Each of these \( N + 1 \) conditions is a tensor with different dimensions. \( f \) is an \( n_f \times 1 \) vector, \( \frac{\partial f}{\partial x} \) denotes an \( n_f \times n_x \) matrix of the first derivatives of \( f \) with respect to \( x \),
\[ \frac{\partial^2 f}{\partial x^2} \] denotes an \( n_f \times n_x \times n_x \) tensor of the second derivatives, where the \( i, j, k \) element of this tensor is the derivative of the \( i \)'th row of \( f \) with respect to the \( j \)'th row of \( x \) and the \( k \)'th row of \( x \).

Since \( f(v(x)) \) is a composition of the functions \( f \) and \( v \), (34) is obtained by the chain rule. In this paper I focus on \( N \leq 2 \). For \( N = 2 \) system (34) looks as follow, where the tensor notation follows Schmitt-Grohe and Uribe (2004)\(^5\):

\[
E[f]^i = 0 \quad \forall i \in \{1, \ldots, n_f\} \tag{35}
\]
\[
E[fv]_\alpha [vx]^\alpha_j = 0 \quad \forall i \in \{1, \ldots, n_f\}, j \in \{1, \ldots, n_x\} \tag{36}
\]
\[
E[fvv]_{\alpha\beta} [vx]_k^\alpha [vx]^\beta_j + E[fv]_\alpha [vx]_{jk} = 0 \quad \forall i \in \{1, \ldots, n_f\}, j \in \{1, \ldots, n_x\}, k \in \{1, \ldots, n_x\} \tag{37}
\]

To get \( v_x \) and \( v_{xx} \), differentiate (28) twice with respect to \( x \):

\[
v_x = \begin{pmatrix} \frac{\partial g(h(x, \epsilon) p)}{\partial x} \\ g_x \\ h_x \\ I_{n_x \times n_x} \end{pmatrix}, \quad v_{xx} = \begin{pmatrix} \frac{\partial^2 g(h(x, \epsilon) p)}{\partial x^2} \\ g_{xx} \\ h_{xx} \\ 0_{n_x \times n_x \times n_x} \end{pmatrix}. \tag{38}
\]

\( v_x \) is a \( n_v \times n_x \) matrix and \( v_{xx} \) is a \( n_v \times n_x \times n_x \) tensor. To save space, it is presented here as a vertical concatenation of four tensors (in an equivalent way to matrix

---

\(^5\)Summation indices are denoted by Greek letters.
concatenation). Note that the first $n_y$ rows of $v_x$ and $v_{xx}$ are obtained by applying first and second order multivariate chain rules to the composite function $g(h(x, e^p))$. The structure of these derivatives is similar to (36) and (37):

\[
\left[ \frac{\partial g(h(x, e^p))}{\partial x} \right]_{ij}^{\alpha} = \left[ g_p^{x\alpha} [h_x]_{j}^{\alpha} \right] \quad \forall i \in \{1, \ldots, n_y\}, j \in \{1, \ldots, n_x\} \quad (39)
\]

\[
\left[ \frac{\partial^2 g(h(x, e^p))}{\partial x^2} \right]_{ijk}^{\alpha\beta} = \left[ g_p^{xx\alpha\beta} [h_x]_{k}^{\beta} [h_x]_{j}^{\alpha} + g_p^{x\alpha} [h_{xx}]_{jk}^{\alpha} \right] \quad (40)
\]

\forall i \in \{1, \ldots, n_y\}, j \in \{1, \ldots, n_x\}, k \in \{1, \ldots, n_x\}

### 3.3 The Taylor series

Next, we express the unknown policy functions as sums of $N$ order Taylor series and $M$ order Taylor errors:

\[
g(x) = \hat{g}(x) + \zeta^G(x) \quad (41)
\]

\[
h(x, e^p) = \begin{pmatrix} \hat{h}(x) + \zeta^H(x) \\ \Phi(x, e^p) \end{pmatrix} \quad (42)
\]

where $\hat{g}$ and $\hat{h}$ are the Taylor series, and $\zeta^G$ and $\zeta^H$ are the Taylor errors. Note that $\hat{h}$ approximates the first $n_x$ rows of $h$ (denoted earlier $\tilde{h}$), because the other rows are known functions. For the case $N, M = 2, 1$ we get:
\[
\hat{g}(x)^i = [G_0]^i + [G_1]_{\alpha}^i [x - c_0]^\alpha + [G_2]_{\alpha\beta}^i [x - c_0]^\beta [x - c_0]^\alpha
\] (43)

\[
[\zeta^G(x)]^i = [G_3]_{\alpha\beta\gamma}^i [x - c_0]^\gamma [x - c_0]^\beta [x - c_0]^\alpha
\] (44)

\[
[\hat{h}(x)]^i = [H_0]^i + [H_1]_{\alpha}^i [x - c_0]^\alpha + [H_2]_{\alpha\beta}^i [x - c_0]^\beta [x - c_0]^\alpha
\] (45)

\[
[\zeta^H(x)]^i = [H_3]_{\alpha\beta\gamma}^i [x - c_0]^\gamma [x - c_0]^\beta [x - c_0]^\alpha
\] (46)

Here, \( \hat{g} \) and \( \hat{h} \) have the same order and the same center, but in principle the orders and the centers may differ.

To calculate (38) we need the derivatives of \( g \) and \( h \) with respect to \( x \) up to order \( N \). For the case \( N = 2 \) differentiating (41) and (42) twice yields:

\[
g_x(x) = \hat{g}_x(x) + \zeta^G_x(x) \quad g_{xx}(x) = \hat{g}_{xx}(x) + \zeta^G_{xx}(x)
\]

\[
h_x(x) = \begin{pmatrix}
\hat{h}_x(x) + \zeta^H_x(x) \\
\Phi_x(x, \epsilon^p)
\end{pmatrix} 
\quad h_{xx}(x) = \begin{pmatrix}
\hat{h}_{xx}(x) + \zeta^H_{xx}(x) \\
\Phi_{xx}(x, \epsilon^p)
\end{pmatrix}
\] (47)

where the derivatives of \( \hat{g} \) and \( \hat{h} \) in tensor notation are:
\[ [\hat{g}_x]^{ij} = [G_1]^{ij} + 2 [G_2]^{ij} [x - c_0]^\alpha \]  
\[ [\hat{g}_{xx}]^{ijk} = 2 [G_2]^{ijk} \]  
\[ [\hat{h}_x]^{ij} = [H_1]^{ij} + 2 [H_2]^{ij} [x - c_0]^\alpha \]  
\[ [\hat{h}_{xx}]^{ijk} = 2 [H_2]^{ijk} \]

and the derivatives of \( \zeta^G \) and \( \zeta^H \) are:

\[ [\zeta^G_x \begin{pmatrix} x \end{pmatrix}]^{ij} = 3 [G_3]^{ij} [x - c_0]^\alpha [x - c_0]^\beta \]
\[ [\zeta^G_{xx} \begin{pmatrix} x \end{pmatrix}]^{ijk} = 6 [G_3]^{ijk} [x - c_0]^\alpha \]
\[ [\zeta^H_x \begin{pmatrix} x \end{pmatrix}]^{ij} = 3 [H_3]^{ij} [x - c_0]^\beta [x - c_0]^\alpha \]
\[ [\zeta^H_{xx} \begin{pmatrix} x \end{pmatrix}]^{ijk} = 6 [H_3]^{ijk} [x - c_0]^\alpha \]

The last required expressions are \( g^p, g_x^p, g_{xx}^p, \ldots \) up to order \( N \), which denote the function \( g \) and its \( N \) derivatives evaluated at \( x^p \). \( g^p \) appears in \( f \) and its derivatives, and \( g_x^p \) and \( g_{xx}^p \) appear in \( v_x \) and \( v_{xx} \), given in (38)-(40). Hence, after \( h \begin{pmatrix} x, \epsilon^p \end{pmatrix} \) is obtained through (42), \( x^p \) is calculated by (29) and substituted for \( x \) in (41) to get \( g^p \). The derivatives \( g_x^p \) and \( g_{xx}^p \) are obtained by substituting \( x^p \) for \( x \) in (47).
3.4 Evaluating the system and applying least squares

We are now in a position to evaluate (34) at $x_0$. For the case $N = 2$ the system has $n_f (1 + n_x + n_x^2)$ conditions. However, due to symmetry of mixed derivatives, the number of unique conditions is only $n_f (1 + n_x + 0.5n_x (n_x + 1))$. This is also the number of unique elements in the first three coefficient tensors $G_0, G_1, G_2, H_0, H_1, H_2$. Thus, the dimensions of the problem are reduced by extracting the unique conditions of the system and the unique elements of the coefficient tensors. This speeds up the solution algorithm, because the size of the Jacobian is smaller. It also contributes to the accuracy of the solution by imposing symmetry on the coefficient tensors.

Symmetric tensors occur repeatedly in this context, because the system conditions and the Taylor coefficients are symmetric derivatives. In general, the number of unique elements in a symmetric tensor with $k$ dimensions and $n^k$ elements is the number of $k$ unique combinations from the set $\{1, 2, \ldots, n\}$ with repetitions. This is given by the binomial coefficient ${n+k-1 \choose k}$. Hence, the number of unique conditions in (34) is $n_f \sum_{k=0}^{N} {n_x+k-1 \choose k}$, and the number of unique Taylor coefficients is $n_f \sum_{k=0}^{N+M} {n_x+k-1 \choose k}$. It follows that in order to get an identifiable problem, the system should be evaluated at $I$ distinct points close to $x_0$, where:

$$I > \frac{\sum_{k=0}^{N+M} {n_x+k-1 \choose k}}{\sum_{k=0}^{N} {n_x+k-1 \choose k}}.$$

---

6 The unique elements of tensor $T$ that is symmetric in its $i_1, i_2, \ldots, i_n$ dimensions are defined as the elements that satisfy $i_1 \leq i_2 \leq \ldots \leq i_n$.

7 Namely, combinations such 1, 2, 2, 3 are allowed and considered identical to 2, 1, 2, 3.
With equality the system is exactly identified, provided that $I$ is an integer. For $M = 0$ the system is exactly identified for $I = 1$.

To obtain the solution, a least squares procedure is applied. Since this is a nonlinear least squares problem, a good initial guess is required. The next section discusses this issue.

3.5 Initial guess

If the model has a deterministic steady state, then a good initial guess is always available. Other cases require to apply general methods, see Judd (1998, p. 176). Let $x^{ss}$ denote the deterministic steady state, provided that it exists. An initial guess can be obtained in one of three ways. The simplest way is to calculate a perturbation solution around a deterministic steady state and use it as an initial guess for $x = x^{ss}$. This can be a good initial guess in many applications. However, if the stochastic model is significantly different from its deterministic version, perturbation methods may not provide a sufficiently good initial guess. In these cases there are two alternatives.

The first option is to add an an exogenous state variable $\sigma_t$ such that the evolution law of the exogenous variables becomes:

$$
\begin{pmatrix}
x_{t+1}^2 \\
\sigma_{t+1}
\end{pmatrix} = \begin{pmatrix}
\Phi(x_t, \sigma_t \epsilon_{t+1}) \\
\sigma_t
\end{pmatrix}.
$$

Here, $\sigma_t$ plays the same role as a perturbation parameter. When $\sigma_t = 0$ the model
is deterministic, and it stays at the steady state when \( \left( \frac{x_1}{x_2} \right) = x^{ss} \). At this point the defining set contains only one defining point. Hence, the true Taylor coefficients can be calculated recursively. Having the first solution, we proceed to adjacent points of the state space, while treating \( \sigma_t \) as a regular state variable. Eventually, we will be interested in points with \( \sigma_t = 1 \).

An alternative option is to use the parameterization \( \Phi(x_t, \sigma_{t+1}) \) where \( \sigma \) is treated as a parameter. For \( \sigma = 0 \) the model is deterministic. At \( x_t = x^{ss} \) the Taylor coefficients can be calculated recursively. Increasing \( \sigma \) gradually from zero to one produces at each iteration a good initial value for the next iteration (as in Homotopy continuation methods). This procedure is applied in the examples below.

4 Results

This section applies the Taylor projection method on the Neo-Classical growth model studied by Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006). Aruoba et al (2006) solve the model with six different methods: first, second and fifth order perturbation, projection with finite elements, projection with Chebyshev polynomials and value function iteration. Their results are taken as a benchmark for the accuracy standards of the existing solution methods and compared to Taylor projection.

The model is a standard Neo-Classical model with elastic supply of labour. The period utility function is \( u(c_t, l_t) = (1 - \tau)^{-1} \left( c^\theta_t (1 - l_t)^{(1 - \theta)} \right)^{(1 - \tau)} \) and the production function is \( y_t = e^{z_t} k^\alpha_t l^{1-\alpha}_t \), where \( z_t \) is an AR technology shock, \( k_t \) is capital, \( l_t \) is labour and \( c_t \) is consumption. The infinitely lived representative agent chooses con-
consumption, labour and capital that maximize her lifetime utility. For further details see Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006). To save space, I give here only the equilibrium conditions:

\begin{align*}
    u_c(c_t, l_t) &= \beta E u_c(c_{t+1}, l_{t+1}) \left( \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} + 1 - \delta \right) \quad (55) \\
    u_l(c_t, l_t) &= u_c(c_t, l_t) (1 - \alpha) e^{z_t} k_t^{\alpha} l_t^{-\alpha} \quad (56) \\
    c_t + k_{t+1} &= e^{z_t} k_t^{\alpha} l_t^{1-\alpha} + (1 - \delta) k_t \quad (57)
\end{align*}

where \( \beta \) and \( \delta \) are the time discount rate and the depreciation rate, respectively, and \( u_c \) and \( u_l \) are the marginal utility from consumption and labour, respectively. (55) is the Euler condition, (56) is the labour supply condition, and (57) is the resource constraint. Together they form the function \( f \) in (27). For a proper performance of the least squares algorithm, it is important to transform \( f \) into a unit free form.

The state variables are \( k_t \) and \( z_t \), where \( k_t \) is predetermined and \( z_t \) is exogenous. The evolution of \( z_t \) is given by:

\[ z_{t+1} = \rho z_t + \epsilon_{t+1}, \quad (58) \]

where \( \epsilon_{t+1} \sim N(0, \sigma^2) \). This is the function \( \Phi \). The control variables are \( l_t \) and \( c_t \). The mapping of these variables to the model notation is \( x^1 = k_t, x^2 = z_t, y = (l_t, c_t) \) and \( \Phi = \rho z_t + \epsilon_{t+1} \).

I study two versions of the model. The first version is the case of logarithmic
utility, full depreciation and fixed labour supply \((\tau = 1, \theta = 1, \delta = 1)\). This case has a closed form solution, so the approximated Taylor coefficients can be compared to the true Taylor coefficients. The second version uses two alternative sets of parameters that are studied in Aruoba et al (2006): a benchmark calibration and an extreme calibration (see Table 2). For these calibrations, accuracy is measured by the unit free Euler errors defined by Aruoba et al (2006). To calculate expectations, the technological innovation \(\epsilon_t\) is discretized by the method of Tauchen (1986) over a grid of 40 points equally spaced between \([-3\sigma, 3\sigma]\).\(^8\)

4.1 Model with a closed form solution

This section assumes \(\tau = 1, \theta = 1, \delta = 1\). The other parameters follow the benchmark calibration of Aruoba et al (2006), presented in Table 2. For these parameter values there is a closed form solution, which is \(l_t = 1, c_t = (1 - \alpha\beta)e^{z_t}k_t^\alpha\) and \(k_{t+1} = \alpha\beta e^{z_t}k_t^\alpha\). Hence, we can calculate the true Taylor coefficients and compare them to the results of the Taylor projection method.

Three specifications are studied, where each specification transforms the model variables into different units. The purpose is to see how the results depend on the specification of the model. The first specification expresses \(c_t\) and \(k_t\) in logs. In this case the policy functions are linear in the state variables, so any combination of \(N\) and \(M\) such that \(N + M \geq 1\) should give an exact solution, because the system errors are zero. This specification should be the most accurate. The second

\(^8\)Aruoba et al (2006) discretize \(z_t\) over a grid of the same size in the interval \([-3\sigma_z, 3\sigma_z]\), where \(\sigma_z\) is the unconditional standard deviation of \(z_t\). Instead, I discretize the technological shock \(\epsilon_t\), because the state variables are assumed to be continuous. The differences are negligible.
specification takes all variables at their original unit. In this case, the policy functions are not Polynomials so the solution is subject to approximation errors. The third specification takes $k_t^\alpha$ and $e^{zt}$ as the state variables and leaves $c_t$ in levels. In this case, the policy function of $c_t$ is a second order Polynomial, but the policy function of $k_{t+1}$ is not. Hence, approximation errors still exist in this specification.

The solution procedure is the following. The center of the Taylor series is taken to be equal to the point at which the model is solved. For instance, if we solve the model at the steady state $x^{ss}$ then $c_0 = x^{ss}$. The system is evaluated at the minimum number of points required for identification.\(^9\) The model is first solved at the deterministic steady state $x^{ss}$. The initial guess is obtained by the third method (continuation) discussed in section 3.5. From $x^{ss}$ we proceed to an adjacent point $x_1$, where the solution at $x^{ss}$ serves as the initial guess. This way, the model is solved successively at adjacent points.

To measure accuracy, a $50 \times 50$ rectangular grid is constructed around the deterministic steady state.\(^10\) The model is solved at all nodes, and the difference between the true Taylor coefficients and the approximated coefficients is calculated for each node. Table 3 presents the $\log_{10}$ of the maximum absolute error across all nodes and all policy functions. The results are presented for each order of coefficients, separately. For instance, column (2) presents the error of the constant terms of the Taylor series, columns (3) shows the first order coefficients and so on. Since the units

---

\(^9\)If one point is required, the system is evaluated only at $c_0$. If $I > 1$ points are required, the system is evaluated at $I$ points equally spaced on a line from $c_0 - \nu$ to $c_0 + \nu$, where $\nu$ is a $n_x \times 1$ vector with all entries equal to $10^{-6}$.

\(^10\)The bounds of the grid are $[0.7k^{ss}, 1.3k^{ss}]$ for capital, where $k^{ss}$ is the steady state level, and $[-3\sigma_z, 3\sigma_z]$ for technology, where $\sigma_z = \frac{\sigma}{\sqrt{1-\rho^2}}$ is the unconditional standard deviation of technology.
of the variables change across specifications, the meaning of the Taylor coefficients also change. To enable comparison across specifications, the table presents in the last column a unit free error, which is the deviation rate of the approximated policy from the true policy.\footnote{Specifically, at each node of the grid the unit free error of capital is $k^{\text{approximate}}/k^{\text{true}} - 1$ and a similar error is calculated for consumption.}

As expected, the approximation errors of the first specification are practically zero, given the computer rounding errors. This result serves also as an accuracy test for the matlab code used in this paper.\footnote{Another test was employed by solving equation (17) with the DSGE code and comparing the results to simple linear least squares.} The second specification exhibits larger errors, because the policy functions are not Polynomials. The third specification is an intermediate case, because the policy function of consumption is polynomial but the policy function of capital is not. Hence, approximation errors are lower for this specification. These results demonstrate the impact of the model specification on the approximation errors. The errors tend to be smaller, when the functional form of the policy function is closer to a Polynomial.

The interesting result is the high accuracy of projections with $N = 0$. For instance, in specification 3 the unit free error of $N, M = 0, 2$ is higher than 1, 1 (but lower from 2, 0). However, if we look at the accuracy across coefficient orders, we see that projections with $N = 0$ consistently underperform. For instance, compare $N, M = 0, 2$ with $N, M = 2, 0$ in specification 3. The constant of the Taylor series is approximated very accurately in both cases, but the first order coefficient exhibits very large differences (in favour of $N, M = 2, 0$). Similar (but weaker) differences arise in specification 2 and by comparing $N, M = 0, 1$ to $N, M = 1, 0$. These differences
reflect the information loss incurred by dropping the zero derivative conditions in $N = 0$ projections. Without these restrictions, it is difficult to estimate accurately the high order Taylor coefficients because the power functions are collinear. Nevertheless, the constant term, which is the parameter of interest, is approximated relatively accurately even for $N = 0$.


This section studies the calibration of Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006), which is presented in Table 2. Two calibrations are considered. The first is the benchmark calibration which follows US data. The second is an extreme calibration where risk aversion is 25 times larger and technological innovations are 5 times more volatile. The extreme calibration is particularly challenging for applying Taylor projection, because the state of the economy can change significantly from period to period. As already explained, in this case the convergence assumption may not hold for all defining points, or alternatively, the speed of convergence may be low. Hence, the approximation errors are expected to be larger for this case.

The accuracy of the results is measured by the unit free Euler Errors defined by Aruoba et al (2006). Substituting $u_c(c_t, l_t) = \theta c_t^{(1-\tau)-1} (1 - l_t)^{(1-\theta)(1-\tau)}$ on the LHS of (55) yields:

$$EE = 1 - \frac{\beta En_c(c_{t+1}, l_{t+1})[\alpha e^{z_t+1} k_{t+1}^{\alpha-1} l_{t+1}^{1-\alpha} + 1 - \delta]}{\theta(1-l_t)^{(1-\theta)(1-\tau)} c_t^{(1-\tau)-1}}.$$  

(59)
The advantage of this form is that the error is unit free and expressed in percent of consumption. For the economic interpretation of this error see Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006).

To calculate the Euler error for a given state $x_t$, we need to calculate the control variables for each possible state $x_{t+1}$. In grid-based methods the approximated policy functions are used to interpolate/extrapolate the control variables at $x_{t+1}$. Since Taylor projection does not depend on a grid, it is possible to proceed in three different ways. The first way is to use the approximated Taylor coefficients at $x_t$ to estimate the control variables at $x_{t+1}$, as done in perturbation solutions. The second option is to use Taylor projection to solve the model again for each point in $x_{t+1}$. The third option is to construct a grid, solve the model at each node, and interpolate/extrapolate the solution at other points. Here, I follow the third option with linear interpolation.\footnote{Linear interpolation is implemented by matlab codes written by Richter, Throckmorton and Walker (2013).}

Aruoba et al (2006) calculate Euler errors over a grid that ranges in $[0.7k^*, 1.3k^*]$ for capital and $[-0.065, 0.065]$ for technology. For comparison, I use a grid with the same bounds and calculate the log of the maximum absolute error. The results are presented in Table 4, where column (1) presents the benchmark calibration and column (2) the extreme calibration. $TP(N, M)$ denotes Taylor projection of order $N, M$. The table shows again that the accuracy of the solution increases with the order of the Taylor series, as found also in the previous examples.

Overall, the accuracy of Taylor projection meets the standards of the grid-based methods studied by Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006), which
include finite elements, Chebyshev polynomials and value function iteration. For the benchmark calibration and the specific grid chosen, Taylor projection with \( N + M \geq 3 \) has a maximum error which is even lower than value function iteration. The results for the extreme calibration are less accurate, which is expected because the defining points are further away from the center of the Taylor series. Nevertheless, the results are still reasonable when compared to other solution methods, particularly for \( N, M = 2, 2 \). The next chapter shows that the accuracy level can improve by orders of magnitude, simply by discretizing time into shorter periods.

5 Controlling Convergence

The fundamental requirement of Taylor projection is the convergence assumption (5). This assumption is likely to hold when the state variables change slowly so that all the defining points are located within a small neighbourhood. This property is affected by the way time is discretized. For instance, if \( t \) is a day then capital changes very little from \( t \) to \( t + 1 \). But when \( t \) is a decade, the small changes accumulate to large quantities, so that convergence may not hold. But even if the convergence assumption holds, the accuracy of Taylor projection depends on the size of the Taylor errors. These errors increase with the dispersion of the defining set. Hence, Taylor projection should be more accurate when time is discretized into shorter periods, because the defining points are closer to one another.

To demonstrate this point, I adjust the calibration of Aruoba et al (2006) to shorter periods. In the original calibration each period is a quarter of a year. To
discretize time differently, let $\tilde{t} \equiv t/T$ denote a new time period for some $T > 1$. For instance, if $T = 10$ and $t$ is a quarter, then $\tilde{t}$ is 12 days. Throughout, I use $\sim$ to denote calibration with $\tilde{t}$ periods. In this calibration, discount, depreciation and decay parameters are calibrated such that their cumulative effect after $T$ periods is the same as in the original calibration:

$$\tilde{\beta}^T = \beta, \quad \left(1 - \tilde{\delta}\right)^T = 1 - \delta, \quad \tilde{\rho}^T = \rho.$$ 

The standard deviation of technological innovations is calibrated so that the unconditional variance of technology is held constant. The unconditional variance of $z_t$ is $\frac{\sigma^2}{1 - \rho^2}$. Hence, the standard deviation of technological innovations in the $\sim$ calibration satisfies:

$$\frac{\tilde{\sigma}^2}{1 - \tilde{\rho}^2} = \frac{\sigma^2}{1 - \rho^2}.$$ 

Production in the $\sim$ calibration is $\frac{1}{T} e^{\tilde{z}_t} k^\alpha l^{1-\alpha}\tilde{t}$. The share of labour income remains the same. If capital, labour and technology are held constant, aggregate production over $T$ periods is identical to production in the quarterly calibration. Since the constant $1/T$ does not affect the results, it is ignored.

The results are presented in Table 4. The table presents two time discretizations. In columns (3) and (4) the time period is a tenth of a quarter ($T = 10$) and in columns (5) and (6) a period is a percent of a quarter ($T = 100$), which is roughly a day. The results are presented for the benchmark and the extreme calibrations.
Since the Euler errors are in percent of consumption, they are comparable across different time discretizations. As expected, the results are now much more accurate. Comparison between columns (1) and (5) shows a remarkable increase in accuracy that in some cases amounts to two orders of magnitude. Particularly surprising is the high accuracy levels achieved in the extreme calibration. These results show that Taylor projection can be extremely accurate, with just a simple adjustment of the model calibration.

6 Computational Costs

The main advantage of the Taylor projection method is that it does not depend on a grid. The method enables to solve a DSGE model at any given point of the state space. This property reduces significantly the computational costs, compared to grid-based methods, because the model is solved only at points of interest. Table 5 presents the time required to solve the benchmark calibration in Table 4 column (1). Time is presented in seconds per each point of the state space. Depending on specification, the solution at one point was obtained in 0.1-0.8 seconds. If we are interested in simulating the model for 10 periods, then the model is solved at 10 points, which would take about 1-8 seconds for the benchmark calibration. If the goal is to estimate the model econometrically, then the model is solved at one point for each iteration. Hence, computation time of Taylor projection is closer in magnitude to perturbation methods, than to grid-based methods.

14For example, suppose that the Euler error in a daily model as a ratio of daily consumption is $\epsilon$. The accumulated errors over a week can reach $7\epsilon$ of daily consumption, which is $\epsilon$ of weekly consumption. Hence, $\epsilon$ is the unit free error of both the daily and the weekly calibrations.
The computational complexity involves a non-linear least squares problem. However, since we always have a good initial value, the solution is obtained within few iterations only. This property is a big advantage, because finding a good initial guess is often a challenging task that consumes a lot of research time (Marcet and Lorenzoni 1999).

In this paper I use the Levenberg-Marquandt algorithm to solve the least squares problem. To speed up computations, the Jacobian was calculated analytically. The formula of the Jacobian is derived in the appendix. The $N$ order projections used in this paper are limited to $N \leq 2$. To increase $N$ further, higher order multivariate chain rules should be used. However, the examples shown in the present paper suggest that the accuracy of the solution depends mainly on $N + M$. From a computational point of view, solving a system with large $N$ and small $M$ is much more costly than a system with small $N$ and large $M$ (holding $N + M$ constant). The tradeoff is between calculating high derivative conditions (when $N$ is large) and evaluating the system at multiple points (when $M$ is large). For instance, Table 5 shows that $TP(1,2)$ was almost five times faster than $TP(2,1)$. Note also that raising $M$ may reduce computational time, as in $TP(1,1)$ and $TP(1,2)$, because approximation errors are smaller, so the least squares problem is solved faster.

The programming costs of Taylor projection can be significant if $N$ is large. However, most of the programming costs are fixed costs, so a code that implements the method on one model can be easily adjusted to other models. To facilitate the use of Taylor projection, this paper provides a Matlab package that produces
all the necessary codes automatically. The user has to define the model as in Schmitt-Grohé and Uribe (2004), choose the algorithm parameters (e.g. \( N \) and \( M \)) and provide an initial guess. The package performs the algorithm and provides the approximated Taylor coefficients.

7 Conclusion

This paper presents a new solution method to DSGE models, called ”Taylor projection”. This method solves functional equations with multiple defining points, provided that the points are located within a small neighbourhood. The advantage of the method is that it does not depend on a grid and hence does not suffer from the curse of dimensionality. Consequently, computational costs are low because the model is solved only at points of interest. An interesting property of this method is that its accuracy depends on the model calibration. Specifically, accuracy is higher when time is discretized into shorter periods.

The main requirement for applying the method is the convergence assumption. However, from a practical viewpoint, the speed of convergence also matters. It is shown that the results are more accurate when the policy functions have a structure that is closer to polynomial, and when the change in the state variables over time is more moderate. In these cases convergence is faster. A possible extension of the method is to use other basis functions with different convergence properties. For instance, Padé series may perform better than Taylor series, because their convergence properties are superior (Judd 1998). This is left for future research.

\[\text{15 The codes are available from the author upon request.}\]
References


The table presents the approximation errors of Taylor projection applied to (17) with $c_0 = 1$. The system of conditions is evaluated at 4 points in $[.79, .81]$ and solved by linear least squares. The table presents the $\log_{10}$ of the absolute approximation error, defined by the difference between the true Taylor coefficients and the approximated coefficients.

Table 2: Calibration of the Neo Classical Growth Model

<table>
<thead>
<tr>
<th>Β</th>
<th>τ</th>
<th>θ</th>
<th>α</th>
<th>δ</th>
<th>ρ</th>
<th>σ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benchmark</td>
<td>0.9896</td>
<td>2.0</td>
<td>0.357</td>
<td>0.4</td>
<td>0.0196</td>
<td>0.95</td>
</tr>
<tr>
<td>Extreme</td>
<td>0.9896</td>
<td>50</td>
<td>0.357</td>
<td>0.4</td>
<td>0.0196</td>
<td>0.95</td>
</tr>
</tbody>
</table>

### Table 3: Model with a closed form solution

<table>
<thead>
<tr>
<th>N,M</th>
<th>$G_0, H_0$</th>
<th>$G_1, H_1$</th>
<th>$G_2, H_2$</th>
<th>$G_3, H_3$</th>
<th>Unit-Free Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(5)</td>
</tr>
<tr>
<td>Specification 1</td>
<td>0,1</td>
<td>-15.2</td>
<td>-15.5</td>
<td>-15.1</td>
<td>-15.1</td>
</tr>
<tr>
<td></td>
<td>0,2</td>
<td>-15.2</td>
<td>-15.5</td>
<td>-15.4</td>
<td>-15.2</td>
</tr>
<tr>
<td></td>
<td>1,0</td>
<td>-15.0</td>
<td>-15.6</td>
<td>-14.9</td>
<td>-15.1</td>
</tr>
<tr>
<td></td>
<td>1,1</td>
<td>-15.2</td>
<td>-15.6</td>
<td>-15.2</td>
<td>-15.1</td>
</tr>
<tr>
<td></td>
<td>1,2</td>
<td>-15.2</td>
<td>-15.8</td>
<td>-15.9</td>
<td>-15.1</td>
</tr>
<tr>
<td></td>
<td>2,0</td>
<td>-15.0</td>
<td>-15.6</td>
<td>-14.9</td>
<td>-14.9</td>
</tr>
<tr>
<td></td>
<td>2,1</td>
<td>-15.2</td>
<td>-15.7</td>
<td>-15.3</td>
<td>-15.1</td>
</tr>
<tr>
<td>Specification 2</td>
<td>0,1</td>
<td>-2.4</td>
<td>-0.7</td>
<td>-1.7</td>
<td>-1.7</td>
</tr>
<tr>
<td></td>
<td>0,2</td>
<td>-3.2</td>
<td>-1.3</td>
<td>-0.1</td>
<td>-2.5</td>
</tr>
<tr>
<td></td>
<td>1,0</td>
<td>-2.6</td>
<td>-1.5</td>
<td>-1.9</td>
<td>-2.5</td>
</tr>
<tr>
<td></td>
<td>1,1</td>
<td>-3.1</td>
<td>-2.1</td>
<td>-0.2</td>
<td>-2.5</td>
</tr>
<tr>
<td></td>
<td>1,2</td>
<td>-3.5</td>
<td>-2.5</td>
<td>-0.5</td>
<td>-2.8</td>
</tr>
<tr>
<td></td>
<td>2,0</td>
<td>-3.2</td>
<td>-2.0</td>
<td>-1.2</td>
<td>-2.5</td>
</tr>
<tr>
<td></td>
<td>2,1</td>
<td>-4.3</td>
<td>-3.2</td>
<td>-1.4</td>
<td>-3.6</td>
</tr>
<tr>
<td>Specification 3</td>
<td>0,1</td>
<td>-2.8</td>
<td>-1.3</td>
<td>-2.2</td>
<td>-2.2</td>
</tr>
<tr>
<td></td>
<td>0,2</td>
<td>-7.2</td>
<td>-2.3</td>
<td>-1.2</td>
<td>-6.5</td>
</tr>
<tr>
<td></td>
<td>1,0</td>
<td>-3.9</td>
<td>-3.2</td>
<td>-3.2</td>
<td>-3.2</td>
</tr>
<tr>
<td></td>
<td>1,1</td>
<td>-6.8</td>
<td>-5.8</td>
<td>-1.2</td>
<td>-6.1</td>
</tr>
<tr>
<td></td>
<td>1,2</td>
<td>-7.7</td>
<td>-6.7</td>
<td>-2.0</td>
<td>-0.9</td>
</tr>
<tr>
<td></td>
<td>2,0</td>
<td>-9.4</td>
<td>-9.1</td>
<td>-9.3</td>
<td>-9.0</td>
</tr>
<tr>
<td></td>
<td>2,1</td>
<td>-7.8</td>
<td>-6.7</td>
<td>-5.0</td>
<td>-0.9</td>
</tr>
</tbody>
</table>

The table presents approximation errors of the Neo Classical growth model of Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006), in a version with a closed form solution. The units of the model variables differ across specifications. Specification 1 uses log transformation, so the state variables are $\log(k_t), z_t$ and the control variable is $\log(c_t)$. In specification 2 all variables are in their original units, so the state variables are $k_t, z_t$ and the control variable is $c_t$. In specification 3 the state variables are $k_t^*, e^t,$ and the control variable is $c_t$. The first column presents the $N, M$ order of the Taylor projection. Columns (2)-(5) present the difference between the true Taylor coefficient and the approximated coefficient, for different orders. For instance, column (2) presents the errors of the constant, columns (3) presents the errors of the first derivatives and so on. For each order the table presents the $\log_{10}$ of the maximum absolute error across a rectangular grid of 2,500 points around the deterministic steady state. To compare errors across specifications, use the unit-free error which is the deviation rate of the approximated policy variables from their true values.
Table 4: Euler errors for the benchmark and extreme calibrations

<table>
<thead>
<tr>
<th>Method</th>
<th>Quarterly model Benchmark</th>
<th>Tenth of a quarter Benchmark</th>
<th>Percent of a quarter Benchmark</th>
<th>Extreme</th>
<th>Extreme</th>
<th>Extreme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(1)</td>
<td>(2)</td>
<td>(3)</td>
<td>(4)</td>
<td>(5)</td>
<td>(6)</td>
</tr>
<tr>
<td>TP(0,1)</td>
<td>-2.9</td>
<td>-2.5</td>
<td>-3.0</td>
<td>-2.2</td>
<td>-3.1</td>
<td>-2.5</td>
</tr>
<tr>
<td>TP(0,2)</td>
<td>-4.2</td>
<td>-2.5</td>
<td>-4.1</td>
<td>-2.7</td>
<td>-6.2</td>
<td>-3.8</td>
</tr>
<tr>
<td>TP(0,3)</td>
<td>-5.2</td>
<td>-2.7</td>
<td>-6.0</td>
<td>-3.7</td>
<td>-6.8</td>
<td>-3.7</td>
</tr>
<tr>
<td>TP(1,0)</td>
<td>-3.0</td>
<td>-1.5</td>
<td>-4.0</td>
<td>-2.3</td>
<td>-4.0</td>
<td>-2.7</td>
</tr>
<tr>
<td>TP(1,1)</td>
<td>-4.1</td>
<td>-2.5</td>
<td>-5.0</td>
<td>-1.7</td>
<td>-4.6</td>
<td>-4.5</td>
</tr>
<tr>
<td>TP(1,2)</td>
<td>-4.9</td>
<td>-2.8</td>
<td>-4.7</td>
<td>-3.8</td>
<td>-6.7</td>
<td>-4.5</td>
</tr>
<tr>
<td>TP(2,0)</td>
<td>-4.4</td>
<td>-2.3</td>
<td>-5.3</td>
<td>-3.3</td>
<td>-6.1</td>
<td>-4.3</td>
</tr>
<tr>
<td>TP(2,1)</td>
<td>-5.2</td>
<td>-2.6</td>
<td>-5.3</td>
<td>-3.2</td>
<td>-5.8</td>
<td>-4.8</td>
</tr>
<tr>
<td>TP(2,2)</td>
<td>-5.2</td>
<td>-2.9</td>
<td>-6.0</td>
<td>-3.6</td>
<td>-6.8</td>
<td>-4.5</td>
</tr>
<tr>
<td>Perturbation1</td>
<td>-2.8</td>
<td>-1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perturbation2</td>
<td>-3.3</td>
<td>-1.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Perturbation5</td>
<td>-3.3</td>
<td>-1.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finite elements</td>
<td>-3.4</td>
<td>-2.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chebyshev</td>
<td>-3.3</td>
<td>-2.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Value function</td>
<td>-4.4</td>
<td>-4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table presents the log\(_{10}\) of the maximum absolute Euler error of the Neo Classical growth model of Aruoba, Fernández-Villaverde and Rubio-Ramírez (2006). The Euler errors are calculated on a grid of 50 \times 50 nodes. The bounds of the grid in column (1) are [0.7k\(**\), 1.3k\(**\)] for capital and [−0.65, 0.65] for technology. The bounds in column (2) are [0.9k\(**\), 2.4k\(**\)] for capital and [−0.33, 0.33] for capital. These bounds correspond to the grids in Aruoba et al (2006). Columns (3)-(4) use a different calibration, where the time period represents a tenth of a quarter, and columns (5)-(6) calibrates time as percent of a quarter. The bounds of the grids for capital maintain the same ratio to the steady state capital level as in columns (1) and (2), and the bounds for technology maintain the same ratio to the unconditional standard deviation of technology. The maximum Euler error for perturbation solutions, finite elements, Chebyshev polynomials and value function iteration are taken from tables 5 and 6 in Aruoba et al (2006).

Table 5: Computation time for the benchmark calibration (seconds per state)

<table>
<thead>
<tr>
<th>Method</th>
<th>TP(0,1)</th>
<th>TP(0,2)</th>
<th>TP(0,3)</th>
<th>TP(1,0)</th>
<th>TP(1,1)</th>
<th>TP(1,2)</th>
<th>TP(2,0)</th>
<th>TP(2,1)</th>
<th>TP(2,2)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.13</td>
<td>0.16</td>
<td>0.22</td>
<td>0.09</td>
<td>0.38</td>
<td>0.17</td>
<td>0.30</td>
<td>0.82</td>
<td>0.52</td>
</tr>
</tbody>
</table>

The table presents the average time (in seconds) required to solve the model at one point of the state space, for the benchmark calibration. Average time is calculated by dividing the time taken to solve the model on the entire grid used in Table 4 column (1) by the size of the grid. The algorithm was implemented in MATLAB R2013a on a desktop computer with Intel(R) Core(TM) i3-2100 CPU, 3.10GHz, 4.00 GB RAM.