4
Parameterized Expectations

Wouter J. den Haan
University of Amsterdam
wdenhaan@uva.nl

4.1 Introduction and preliminaries

In this Chapter, I describe the Paramaterized Expectations Algorithm, which can be used to solve dynamic stochastic general equilibrium (DSGE) models. Two different versions will be discussed. The first is called stochastic PEA and the second non-stochastic PEA. The terminology does not refer to whether the model being solved is stochastic or not. That is, both procedures are meant to solve stochastic models. The terminology refers to whether the numerical procedure itself has stochastic elements. The stochastic version uses a simulated series for the innovations of the exogenous driving process and the outcome, thus, depends on the particular draw used. By increasing the sample size one can, of course, reduce the role of sampling variation. The solution obtained with the non-stochastic version does not have this dependence on a particular random realization.

Both versions allow for higher-order approximations quite easily. The advantage of stochastic PEA is that it is intuitive and easy to program. Non-stochastic PEA is a projection procedure and is much more efficient. That is, with non-stochastic PEA it takes typically less computing resources to get an accurate solution. For simple problems this may not matter very

much because by using a large enough sample of simulated data an accurate solution can be obtained with stochastic PEA as well. For more complex problems, however, computing efficiency is important.

4.1.1 Model and specification of the numerical problem

I will illustrate the numerical algorithms using the stochastic neo-classical growth model.

$$\begin{aligned}\max_{\{c_{t+j},k_{t+1+j}\}_{j=0}^{\infty}} & \mathrm{E}_t\left[\sum_{j=0}^{\infty}\beta^j\frac{c_{t+j}^{1-\gamma}-1}{1-\gamma}\right]\\ \text{s.t.}\quad & c_{t+j}+k_{t+1+j}=z_{t+j}k_{t+j}^{\alpha}+(1-\delta)k_{t+j}\\ & k_t \text{ is given.}\end{aligned}$$

The law of motion of aggregate productivity, z_t, is given by

$$\ln(z_t)=\rho\ln(z_{t-1})+\sigma\varepsilon_t,\quad \varepsilon_t\sim N(0,1) \tag{4.1}$$

A solution to this model consists of a policy function for consumption and captial. The arguments of these policy functions are the state variables, i.e., k_t and z_t. Thus, we are looking for a function $c_t=c(k_t,z_t)$ and a function $k_{t+1}=k(k_t,z_t)$. These two functions have to satisfy the first-order conditions

$$c_t^{-\gamma}=\mathrm{E}_t\left[\beta c_{t+1}^{-\gamma}\left(\alpha k_{t+1}^{\alpha-1}+1-\delta\right)\right] \tag{4.2}$$

$$c_t+k_{t+1}=z_tk_t^{\alpha}+(1-\delta)k_t \tag{4.3}$$

Numerical solutions of these functions are obtained for one specific set of values for the structural parameters. That is, at each point of the numerical algorithm, numerical values of the structural parameters are fixed and known. In fact, one of the first things one typically does in a program is to assign numerical values to all structural parameters.

Even though we are only looking for policy functions for one particular set of parameter values, this is still a tough problem. The reason is that we do not know the functional form of the policy function. And the space of functions is a large space to search for a solution.

4.1.2 Special case with known solution

Although one almost always has to rely on numerical procedures to solve DSGE models there is one model for which we known the solution in analytical form. This is the case when $\gamma=1$ (i.e., log utility) and $\delta=1$ (i.e., complete depreciation). In this case, the solutions for the consumption and the investment function are given by

$$c_t=c(k_t,z_t)=(1-\alpha\beta)z_tk_t^{\alpha} \tag{4.4}$$

$$k_{t+1} = c(k_t, z_t) = \alpha\beta z_t k_t^{\alpha} \tag{4.5}$$

Students often want to know *how to find* such a solution. The truth is that I do not know. It probably was discovered by luck or by trial and error. It is also not that important, because it is unlikely there are many interesting models that have an analytical solution. It is important to know, however, *how to check* whether a proposed solution is indeed a solution. A true solution satisfies the budget constraint and the Euler equation. It is trivial to see that the proposed solutions satisfy the budget constraint. To see whether they satisfies the Euler equation, we simply plug in these functions into the Euler equation.

$$c_t^{-1} = \mathrm{E}_t\left[\beta c_{t+1}^{-1}\alpha k_{t+1}^{\alpha-1}\right] \tag{4.6}$$

$$\frac{1}{(1-\alpha\beta)z_t k_t^{\alpha}} = \mathrm{E}_t\left[\beta\frac{\alpha z_{t+1}\left(\alpha\beta z_t k_t^{\alpha}\right)^{\alpha-1}}{(1-\alpha\beta)z_{t+1}k_{t+1}^{\alpha}}\right] \tag{4.7}$$

$$\frac{1}{z_t k_t^{\alpha}} = \beta\frac{\alpha\left(\alpha\beta z_t k_t^{\alpha}\right)^{\alpha-1}}{k_{t+1}^{\alpha}} \tag{4.8}$$

Note that the expectations operator is no longer used because there are no stochastic elements on the right-hand side anymore.[1] Eliminating $z_t k_t^{\alpha}$ on both sides and using the solution one more time, we get

$$1 = \beta\frac{\alpha(\alpha\beta)^{\alpha-1}\left(z_t k_t^{\alpha}\right)^{\alpha}}{(\alpha\beta z_t k_t^{\alpha})^{a}} = 1 \tag{4.9}$$

4.1.3 Approximating polynomials

Underlying our numerical solution procedure is Weierstrass theorem. It says the following. If $f(x)$ is a continuous function on $[a, b]$, then there exists a polynomial that is arbitrarily close to $f(x)$. That is, for all $\varepsilon > 0$, there is an n^{th}-order polynomial, $p_n(x)$, such that[2]

$$\forall x \in [a, b],\ |f(x) - p_n(x)| \leq \varepsilon.$$

In other words, the class of polynomials is dense in the space of continuous functions. There are other functions that have the same property, for example, neural nets. But here we will use polynomials.

---

[1] This doesn't mean that outcomes are not stochastic. But z_{t+1} affects two outcomes next period, marginal utility of consumption and the marginal product of capital. Make you understand why z_{t+1} pushes these two variables in opposite directions so that it is at least in principle possible that z_{t+1} does not affect the product of these two variables.

[2] Note that the sup norm is used to evaluate whether $p_n(x)$ is "close" to $f(x)$. If weaker norms are used, such as the L^2 norm, then $f(x)$ doesn't have to be continuous.

How does this help us? First, we start with a low-order polynomial, typically a first-order polynomial. Now we have pinned down the functional form and have reduced the problem from finding an unknown functional form to finding the finite number of coefficients of this low-order polynomial. Next, we increase the order of the polynomial until we have found an accurate solution. Although Weierstrass theorem underlies the motivation for using polynomials, it actually is a bit misleading to refer to it here. At this point you shouldn't worry about it, but I'll return to it in Section 4.4.2.

4.2 Stochastic PEA

4.2.1 Which function to approximate?

Recall that we are looking for the policy functions $c(k_t, z_t)$ and $k(k_t, z_t)$. These two functions have to satisfy the first-order conditions

$$c_t^{-\gamma} = \mathrm{E}_t \left[\beta c_{t+1}^{-\gamma} \left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta \right) \right] \tag{4.10}$$

$$c_t + k_{t+1} = z_t k_t^{\alpha} + (1-\delta) k_t \tag{4.11}$$

Also, although I use symbols to indicate the structual parameters, the reader should keep in mind that we know the numerical value of all structural parameters.

Let's think about the conditional expectation in Equation (4.10). It is supposed to be the best forecast of $\beta c_{t+1}^{-\gamma} \left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta \right)$. What variables would be useful in making this forecast in period t? Those are the state variables, k_t and z_t. That is, we know that this conditional expectation is a function of the state variables, that is,

$$g(k_t, z_t) = \mathrm{E}_t \left[\beta c_{t+1}^{-\gamma} \left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta \right) \right] \tag{4.12}$$

Another way to realize that the conditional expectation must be a function of the state variables is that in equilibrium the conditional expectation (the right-hand side of Equation 4.10) is equal to the marginal utility of consumption (the left-hand side of Equation 4.10). The conditional expectation is, thus, simply a function of consumption and consumption is a function of the state variables.

If we make explicit that the consumption and capital choice are also functions of the state variables, then the model can be written as

$$c(k_t, z_t)^{-\gamma} = g(k_t, z_t) \tag{4.13}$$

$$c(k_t, z_t) + k(k_t, z_t) = z_t k_t^{\alpha} + (1-\delta) k_t \tag{4.14}$$

When solving a DSGE model one has to choose which function to approximate. Typically, one has a choice. In this model, the choices are the

conditional expectation, the consumption function, and the capital function. If one has solved for one, then the others are easily determined. If one knows the conditional expectation, $g(k_t, z_t)$, then one can solve for the consumption function from the Euler equation and then for the capital function from the budget constraint. If one chooses to approximate the consumption function, then the capital function is given by the budget contraint and the conditional expectation from the Euler condition. Note that in practice it makes a difference. For example, suppose that one uses a linear approximation for the consumption function. Then the capital choice is not linear because it is given by the budget constraint which has non-linear elements.

One should approximate the function that is most likely to be close to a low-order polynomial. For the neoclassical growth model it doesn't matter very much because all functions are approximated quite well with a low-order polynomial (except when risk aversion and the amount of uncertainty are high). But in other models the expectation may be a bit smoother than other functions in the model because it is an expectation. Anyway, the PEA algorithm puts the conditional expectation at the center of the solution algorithm and that is the function that is approximated with a flexible functional form. So let's get started.

4.2.2 With what to approximate the conditional expectation

Let $P_n(s;\eta_n)$ stand for the n^{th}-order polynomial of the vector s with coefficients η_n. It is important to realize that by using monotone transformations one actually has a lot of flexibilities even if one is restricted to use polynomials. Here we approximate $g(k_t, z_t)$ as follows

$$g(k_t, z_t) \approx \exp(P_n(\ln(k_t), \ln(z_t); \eta_n)$$

That is, we approximate the log of the conditional expectation with a polynomial in the logs. There are several reasons for doing this. The main reason is that it has turned out to work very well in practice (and in more complex models as well). But this result may not have been completely unexpected. For example, by using the exponential of a polynomial we ensure that the contional expectation is always positive. Also, in many economic models, it is more likely that elasticities are constant than partial derivatives. This motivates using (natural) logarithms. Finally, the analytical solution for the special case discussed above has this form. To see why first note that the conditional expectation, i.e., the right-hand side, is equal to the marginal utility of consumption, i.e., the left-hand side. Thus,

$$\begin{aligned} g(k_t, z_t) = c_t^{-\gamma} = c_t^{-1} &= \tfrac{1}{(1-\alpha\beta)z_t k_t^{\alpha}} \\ = \exp\{-\ln(1-\alpha\beta) - \alpha\ln(k_t) - \ln(z_t)\} \\ = \exp\left\{P_1(\ln(k_t), \ln(z_t); [-\ln(1-\alpha\beta), -\alpha, -1]'\right\} \end{aligned} \tag{4.15}$$

4.2.3 How to simulate?

Suppose you are given

1. the values of the coefficients of the approximating polynomial, $\bar{\eta}_n$,
2. an initial value for aggregate productivity, z_1, and a sequence of realizations for the innovations of the aggregate productivity shock, $\{\varepsilon_t\}_{t=2}^{T}$, and
3. an initial capital stock, k_1.

Would it then be possible to generate a time series for c_t and k_{t+1}? Sure it is and it is very easy. First, given the innovations ε_t it is trivial to construct a time series for z_t. In period 1, use k_1 and z_1 together with $P_n(\ln(k_1), \ln(z_1); \bar{\eta}_n)$ to get an (approximating) value for the conditional expectation. From the Euler equation one can then solve for consumption. That is,

$$c_1 = [\exp(P_n(\ln(k_1), \ln(z_1); \bar{\eta}_n))]^{-1/\gamma} \tag{4.16}$$

Plugging this value for consumption in the budget constraint gives k_2. Iterating on this scheme gives the whole time series.

4.2.4 The preliminaries of the algorithm

The algorithm consists of an iterative scheme. Before starting the iterative scheme, we have to know the following:

1. The numerical values of the structural parameters.
2. A long time series of realizations of the aggregate productivity process $\{\varepsilon_t\}_{t=2}^{T}$ and an initial value for aggregate productivity, z_1. The draws for ε_t can be obtained using a numerical random number generator. A sensible value for z_1 would be the steady state value. Now you have enough information to generate a series $\{z_t\}_{t=1}^{T}$. This series should never change. Even if you change, for example, the order of the polynomial or a structural parameter, it is better to keep the exact same series. Moreover, you want to make sure you draw the same random numbers if you rerun your program. This can be accomplished by fixing the seed of the random number generator.
3. An initial capital stock, k_1. Like the initial value for z_t, it should be sensible. That is, it shouldn't be far from where the economy operates. Typically, the steady state value is fine.
4. An initial value for the coefficients of the approximating polynomial, η_n^1. The superscript refers to the number of the iteration, so it is equal to 1 in the first round. After one has obtained a numerical solution

for η_n, one may consider a higher-order solution. But in obtaining one particular solution the order of the approximation is fixed. The subscript of η refers to the order of the polynomial.

4.2.5 Iterative procedure

The algorithm uses the following iterative procedure

Step 1: Simulate.

Simulate the economy for T periods. In particular, calculate c_t, k_{t+1}, and the argument *inside* the conditional expectation, that is,

$$y_{t+1} = \beta c_{t+1}^{-\gamma} \left(\alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta\right). \tag{4.17}$$

Step 2: Update coefficients.

Update the coefficients of the approximating polynomial, that is, calculate η_n^2.

The algorithm repeats itself until the values of n_n^i have converged. How to simulate has already been explained above, so I only have to explain the second step. Besides describing the mechanics, I also want to give a little bit of an economy story behind the iterative procedure.

One can think of the polynomial that approximates the conditional expectation in the i^{th}-iteration, $P_n(\ln(k_t), \ln(z_t); \eta_n^i)$ as the beliefs of the agents about the expected value of $\beta c_{t+1}^{-\gamma} \left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta\right)$. Based on these beliefs agents make the consumption and capital decision (step 1 of the algorithm). In step 1, the program generates a long time series according to these beliefs. Now suppose that agents ask themselves whether they can update their beliefs using the available time series. It is important to remember, that we restrict agents to use a particular functional form, namely an n^{th}-order polynomial. So the question is what the best values for the coefficients of the approximating polynomial are. But this is a typical econometric problem. By definition of a conditional expectation we have that

$$y_{t+1} = g(k_t, z_t) + u_{t+1}, \tag{4.18}$$

where u_{t+1} is a prediction error and, thus, satisfies all the standard assumptions of a regression error. The idea is to use the same logic for the approximating polynomial. Thus,

$$y_{t+1} = \exp(P_n(\ln(k_t), \ln(z_t); \eta_n)) + \widetilde{u}_{t+1} \tag{4.19}$$

Recall that in step #1 we have calculated time series for y_{t+1}, k_t, and z_t. In step #2, we first use least-squares to find the value of η that gives the best fit to Equation (4.19). That is,

$$\widehat{\eta}_n = \arg\min_{\eta_n} \sum_{t=\bar{T}+1}^{T-1} \left(y_{t+1} - \exp(P_n(\ln(k_t), \ln(z_t); \eta_n))\right)^2 \tag{4.20}$$

Note that the first $\bar{T}$ observations are discarded. By doing this we can get rid of any dependence on the initial values chosen for z_1 and k_1.[3] For the standard growth model, sensible choices for T and $\bar{T}$ are 5,000 and 500.

One could stop here and simply let $\eta_n^{i+1} = \widehat{\eta}_n$. If it works then this is great but with this choice the algorithm could very well diverge. That is, the steps taken by the algorithm are too big. In general, one should use

$$\eta_n^{i+1} = \lambda\widehat{\eta}_n + (1-\lambda)\eta_n^i \quad \text{with } 0 < \lambda \leq 1 \tag{4.21}$$

For well behaved problems one can use values for λ close to 1 and for tougher problems one should use lower values.

Stopping rule

One can use the following stopping rule[4]

$$\max(\widehat{\eta}_n(j) - \eta_n^i(j)) < 0.0001. \tag{4.22}$$

4.3 Non-Stochastic PEA

4.3.1 Key difference with stochastic PEA

Both versions of PEA approximate the conditional expectation with a polynomial. Nevertheless there is a key difference. Stochastic PEA finds the coefficients of the approximating polynomial using the following regression

$$y_{t+1} = \exp(P_n(\ln(k_t), \ln(z_t); \eta_n)) + \widetilde{u}_{t+1}, \tag{4.23}$$

with the dependent variable defined as

$$y_{t+1} = \beta c_{t+1}^{-\gamma}\left(\alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta\right). \tag{4.24}$$

This means that $\widetilde{u}_{t+1}$ contains (in addition to possible numerical errors) true random (prediction) errors.

---

[3] Discarding a set of initial values is especially important if your choices for z_1 and/or k_1 were not that sensible, but is a good practice in general.

[4] Determining the stopping rule is not trivial. In principle it would be better to have a stopping rule on the function itself. If there is multicollinearity then coefficients may change quite a bit even though the function values do not. Also, if coefficients are large then using the percentage change may be better.

Non-stochastic PEA differs from stochastic PEA in two key aspects. The first is that it replaces y_{t+1} in Equation 4.23 with an explicitly calculated conditional expectation of y_{t+1}. This would imply running the regression

$$\mathrm{E}_t\left[y_{t+1}\right] = \exp(P_n(\ln(k_t), \ln(z_t); \eta_n)) + \bar{u}_{t+1}, \tag{4.25}$$

where $\bar{u}_{t+1}$ now may still capture numerical errors but does not contain random noise. Consequently, Equation 4.25 is a more efficient system to obtain estimates for η_n. A second difference is that non-stochastic PEA does not use a simulated series to generate the observations for the regression. Instead it constructs a grid. By controlling the location of the nodes used to find η_n one typically can gain efficiency.

I will first describe the procedure when I use an arbitrary grid in the untransformed state variables, $\ln(k_t)$ and $\ln(z_t)$. This will hopefully show that non-stochastic PEA is not much more difficult than stochsatic PEA. Later I will show how the algorithm can be modified to use Chebyshev nodes and Chebyshev polynomials. This is only a minor (but very useful) modification but you may get lost in the notation if I immediately start with it.

4.3.2 Preliminaries of the algorithm

Calculating the conditional expectation using numerical integration

The goal is to calculate

$$\mathrm{E}_t\left[\beta c_{t+1}^{-\gamma}\left(\alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta\right)\right]$$

given (i) values for k_t and z_t and (ii) a value for η_n with which each endogenous element in the conditional expectation can be calculated.

First, replace $c_{t+1}^{-\gamma}$ by its approximation. This gives

$$\mathrm{E}_t\left[\begin{array}{c} \beta P_n\left(\begin{array}{c}\ln(k_{t+1});\\ \ln(z_{t+1})\end{array}\ ;\eta_n\right) \\ \times\left(\alpha\left(\begin{array}{c} z_{t+1}\times \\ k_{t+1}^{\alpha-1}\end{array}\right)+1-\delta\right)\end{array}\right]$$

Using that the value for k_{t+1} is given by

$$k_{t+1} = z_t k_t^{\alpha} + (1-\delta)k_t - \left[\exp(P_n(\ln(k_t), \ln(z_t); \eta_n))\right]^{-1/\gamma}$$

we get

$$\mathrm{E}_t\left[\begin{array}{c} \beta P_n\left(\begin{array}{c}\ln\left(\begin{array}{c} z_t k_t^{\alpha} + (1-\delta)k_t \\ -\left[\exp(P_n(\ln(k_t), \ln(z_t); \eta_n))\right]^{-1/\gamma}\end{array}\right),\ ;\eta_n\\ \ln(z_{t+1})\end{array}\right) \\ \times\left(\alpha\left(\begin{array}{c} z_{t+1}\times \\ \left(\begin{array}{c} z_t k_t^{\alpha} + (1-\delta)k_t \\ -\left[\exp(P_n(\ln(k_t), \ln(z_t); \eta_n))\right]^{-1/\gamma}\end{array}\right)^{\alpha-1}\end{array}\right)\right) \\ +1-\delta\end{array}\right].$$

Finally, we make explicit that the value over which we are integrating is ε_{t+1}. Thus, we are trying to calculate

$$\mathrm{E}\left[\begin{array}{c} \beta P_n\left(\begin{array}{c} \ln\left(\begin{array}{c} z_t k_t^{\alpha}+(1-\delta)k_t \\ -\left[\exp(P_n(\ln(k_t),\ln(z_t);\eta_n))\right]^{-1/\gamma} \end{array}\right); \\ \ln(\exp(\rho\ln(z_t)+\varepsilon_{t+1})) \end{array};\eta_n\right) \\ \times\left(\begin{array}{c} \alpha\left(\begin{array}{c} (\exp(\rho\ln(z_t)+\varepsilon_{t+1}))\times \\ \left(\begin{array}{c} z_t k_t^{\alpha}+(1-\delta)k_t \\ -\left[\exp(P_n(\ln(k_t),\ln(z_t);\eta_n))\right]^{-1/\gamma} \end{array}\right)^{\alpha-1} \end{array}\right) \\ +1-\delta \end{array}\right) \end{array}\right].$$

But this is a standard numerical integration problem. Note that the shock, ε_{t+1}, shows up in two places. It affects next period's productivity but it also affect consumption because z_{t+1} is a state variable. Since ε_{t+1} has a normal distribution, it is appropriate to use Gaussian Hermite quadrature. Let $\{\xi_h,\omega_h\}_{h=1}^{H}$ be the set of Hermite nodes and weights. The conditional expectation can then be calculated using

$$\sum_{h=1}^{H}\left[\begin{array}{c} \beta P_n\left(\begin{array}{c} \ln\left(\begin{array}{c} z_t k_t^{\alpha}+(1-\delta)k_t \\ -\left[\exp(P_n(\ln(k_t),\ln(z_t);\eta_n))\right]^{-1/\gamma} \end{array}\right), \\ \ln(\exp(\rho\ln(z_t)+\sqrt{2}\sigma\xi_h)) \end{array};\eta_n\right)\times \\ \left(\begin{array}{c} \alpha\left(\begin{array}{c} (\exp(\rho\ln(z_t)+\sqrt{2}\sigma\xi_h))\times \\ \left(\begin{array}{c} z_t k_t^{\alpha}+(1-\delta)k_t \\ -\left[\exp(P_n(\ln(k_t),\ln(z_t);\eta_n))\right]^{-1/\gamma} \end{array}\right)^{\alpha-1} \end{array}\right) \\ +1-\delta \end{array}\right) \\ \times\frac{1}{\sqrt{\pi}}\omega_h \end{array}\right].$$

Grid

Let $\{k_j,z_j\}_{j=1}^{J}$ be the set of grid point for the two state variables, $\ln(k)$ and $\ln(z)$. It may be a bit confusing that we now use a subscript, namely j, to indicate the node and we use a subscript, namely t, to indicate the time period. From the context it should be clear though whether we are on a grid or in the time domain. Figuring out a sensible range for z_t is easy. The standard deviation of $\ln(z_t)$, σ_z, is equal to $\sqrt{\sigma^2/(1-\rho^2)}$. A sensible grid would locate points in the interval $[\exp(-3\sigma_z),\exp(3\sigma_z)]$, although one may want to start with a smaller range, say $[\exp(-2\sigma_z),\exp(2\sigma_z)]$. What a sensible range is for capital is a bit more difficult. I often found the interval $[0.8k_{ss},1.2k_{ss}]$ a good starting point. This range covers values from 20% below to values 20% above the steady state capital stock. If your initial value is bad and leads, for example, to negative capital stock choices, then one may want to start again with a somewhat smaller range.[5]

[5] Note that consumption is positive by construction but if consumption exceeds total resources then the budget constraint would imply a negative capital choice.

4.3.3 Non-stochastic PEA as a projection method

A true rational expectations solution satisfies the Euler equations at all points in the state space. The idea behind projection methods is to choose the value of η_n to get a good fit on the grid points. Consider a grid point (k_j, z_j). Using the formula for the conditional expectation derived above we can check whether at this point the Euler equation holds for a particular choice of η_n. In particular, the Euler equation error at the j^{th}-node is defined as

$$u_j = +P_n(\ln(k_j), \ln(z_j); \eta_n)$$

$$- \sum_{h=1}^{H} \left[\begin{array}{c} \beta P_n \left(\begin{array}{c} \ln \left(\begin{array}{c} z_j k_j^{\alpha} + (1-\delta) k_j \\ - [\exp P_n(\ln(k_j), \ln(z_j); \eta_n))]^{-1/\gamma} \end{array} \right), \\ \ln(\exp(\rho \ln(z_J) + \sqrt{2}\sigma\xi_h)) \end{array} ; \eta_n \right) \times \\ \left(\begin{array}{c} \alpha \left(\begin{array}{c} (\exp(\rho \ln(z_j) + \sqrt{2}\sigma\xi_h)) \times \\ \left(\begin{array}{c} z_j k_j^{\alpha} + (1-\delta) k_j \\ - [\exp(P_n(\ln(k_j), \ln(z_j); \eta_n))]^{-1/\gamma} \end{array} \right)^{\alpha - 1} \end{array} \right) \\ + 1 - \delta \end{array} \right) \\ \times \frac{1}{\sqrt{\pi}} \omega_h \end{array} \right]$$

So the idea is to choose η_n such that the error terms u_j are close to zero. If the number of elements of η_n is equal to the number of grid point then one can use an equation solver to find the value of η_n that sets u_j equal to zero at all nodes. If one has more nodes then elements to solve for one could find η_n with the following minimization problem

$$\eta_n^* = \arg\min_{\eta_n} \sum_{j=1}^{J} u_j(\eta_n)^2$$

This simply gives each grid point equal weight. The projection literature has thought carefully on how to choose efficient weigthing functions but I found this unweighted procedure to often work well.

4.3.4 Iterative procedure for non-stochastic PEA

A simple way to find the solution η_n^* is to use the following iterative procedure. All aspects of the algorithm, such as construction of the grid and choice of approximating function, are identical to those outlined above. The only difference is in how the value of η_n^* is found. Let η_n^1 be the initial value for the vector of coefficients.

Step I: Calculate conditional expectation explicitly

In the i^{th}-iteration the value of η_n^i is given. In this step we want to calculate at each node $\{k_j, z_j\}$ the conditional expectation explicitly when η_n^i is used. Denote the value calculated at the j^{th}-node by y_j^e. The superscript e

indicates that this is an expectation in contrast to the realization y_{t+1} that was used in stochastic PEA. Thus, we want to calculate

$$y_j^e = \sum_{h=1}^{H} \left[\begin{array}{c} \beta P_n \left(\begin{array}{c} \ln \left(\begin{array}{c} z_j k_j^\alpha + (1-\delta)k_j \\ -\left[\exp P_n(\ln(k_j), \ln(z_j); \eta_n^i)\right]^{-1/\gamma} \end{array} \right), \\ \ln(\exp(\rho \ln(z_J) + \sqrt{2}\sigma\xi_h)) \end{array} ; \eta_n^i \right) \times \\ \left(\begin{array}{c} \alpha \left(\begin{array}{c} (\exp(\rho \ln(z_j) + \sqrt{2}\sigma\xi_h)) \times \\ \left(\begin{array}{c} z_j k_j^\alpha + (1-\delta)k_j \\ -\left[\exp(P_n(\ln(k_j), \ln(z_j); \eta_n^i))\right]^{-1/\gamma} \end{array} \right)^{\alpha-1} \end{array} \right) \\ +1-\delta \end{array} \right) \\ \times \frac{1}{\sqrt{\pi}} \omega_h \end{array} \right]. \tag{4.26}$$

One could just blindly program this formula but it may be useful to understand the key steps you are taking at each node.

- Given values for k_j and z_j and using η_n^i calculate c_j from

$$c_j = \left[P_n(\ln(k_j), \ln(z_j); \eta_n^i)\right]^{-1/\gamma}. \tag{4.27}$$

  Use this value of c_j and the budget constraint to solve for k_j'. The value of c_j is only used to calculate k_j' and can now be discarded.

- Given the value of k_j' calculate the conditional expectation. To calculate the value of next period's marginal utility one can use the approximating function with η_n^i as its coefficients. In fact, given k_j' the expression for y_j^e can be made a bit more transparent. Thus,

$$y_j^e = \sum_{h=1}^{H} \left[\begin{array}{c} \beta \exp\left(P_n \left(\begin{array}{c} \ln(k_j'); \\ \ln(\exp(\rho \ln(z_J) + \sqrt{2}\sigma\xi_h)) \end{array} ; \eta_n^i \right) \right) \\ \times \left(\alpha \left(\begin{array}{c} (\exp(\rho \ln(z_j) + \sqrt{2}\sigma\xi_h)) \times \\ \left(k_j'\right)^{\alpha-1} \end{array} \right) + 1 - \delta \right) \frac{1}{\sqrt{\pi}} \omega_h \end{array} \right]. \tag{4.28}$$

Step II: Update coefficients of approximating function

In the second step we get an updated value for η_n by projecting the values of y_j^e on P_n. One possibility would be to use

$$\widehat{\eta}_n = \arg\min_{\eta_n} \sum_{j=1}^{J} \left(y_j^e - \exp\left(P_n(\ln(k_j), \ln(z_j); \eta_n)\right) \right)^2.$$

But recall that with non-stochastic PEA the difference between y_j^e is *not* a stochastic error term. Consequently, we take transformations of both terms.

That is, we can also use

$$\widehat{\eta}_n = \arg\min_{\eta_n} \sum_{j=1}^{J} \left(\ln(y_j^e) - P_n(\ln(k_j), \ln(z_j); \eta_n)\right)^2, \tag{4.29}$$

but this is equivalent to a linear regression. Note that we can use this regression procedure when J is equal to the number of elements in η_n or when it is bigger. It is often better to slowly update the value of η_n. Thus,

$$\eta_n^{i+1} = \lambda\widehat{\eta}_n + (1-\lambda)\eta_n^i \quad \text{with } 0 < \lambda \leq 1. \tag{4.30}$$

Discussion

One can think of the iterative procedure as a numerical algorithm to find the value of η_n that gives the best fit on the nodes. For many problems it would be inefficient numerical procedure. The reason is that it doesn't use any derivatives to figure out what the best direction is to change η_n. Nevertheless, it may be useful because it is easy to program and because you can control what values of η_n^i the algorithm tries. The problem may not be well defined for all possible choices of η_n^i. For example, some choices of η_n^i, the capital choice may be negative. By choosing λ effectively one can often avoid such problems. Equation solvers and minimization routines are often black boxes and one cannot always easily control what values of η_n they are trying.

4.3.5 Non-stochastic PEA with Chebyshev grid & Chebyshev polynomials

In the chapter on function approximation we learned two important things that are relevant for the discussion here. First, by using Chehyshev nodes as grid point we ensure uniform convergence. Second, higher-order approximations are sometimes difficult to implement because of the multicollinearity between the explanatory variables. Using Chebyshev polynomials are useful for this problem because they ensure that evaluated at the nodes the basis functions of Chebyshev polynomials are orthogonal to each other.

But Chebyshev nodes are in between -1 and 1. So to use Chebyshev polynomials we have to do some scaling. Suppose we would like to have a grid such that $\ln(k_j)$ is in between $k^{\min}$ and $k^{\max}$ and $\ln(z_j)$ is in between $z^{\min}$ and $z^{\max}$. We can then adopt the procedure above as follows.

- Generate Chebyshev nodes for the two state variables, ζ^k and ζ^z.
- The mapping between Chebyshev nodes and the actual values of the state variables is given by

$$\ln(k_j) = a^k + b^k\zeta_j^k$$

with

$$a^k = \frac{k^{\max} + k^{\min}}{2} \text{ and } b^k = \frac{k^{\max} - k^{\min}}{2}$$

and a similar equation to scale $\ln(z_j)$. There are different ways to proceed but the easiest may be to actually define the grid using the scaled up Chebyshev nodes. This way the value at a node is simply the actual value of the state variable.

- To ensure that the discrete orthogonality property works, however, one has to use the following modified procedure to generate the Chebyshev basis functions.

$$\begin{aligned} T_0^c(\ln k_j) &= 1 \\ T_1^c(\ln k_j) &= \frac{(\ln k_j - a^k)}{b^k} \\ T_{i+1}^c(\ln k_j) &= \frac{(\ln k_j - a^k)}{b^k} T_i^c(\ln k_j) - T_{i-1}^c(\ln k_j) \end{aligned}$$

This way of defining the Chebyshev basis function undoes the scaling. Consequently, we have the discrete orthogonality property again. That is, if X is the matrix with the regressors of the linear regression associated with the problem in Equation 4.29, then $X'X$ is a diagonal matrix which is always invertible even if the rows of X have many higher-order polynomial terms.

4.4 Concluding comments

4.4.1 Rational expectations equilibrium

Suppose that the program converges. Let η_n^* stand for the value of η_n in the last iteration, that is, the one that satisfies the stopping rule. What can we say about a solution that is based on $P_n(\ln(k_t), \ln(z_t); \eta_n^*)$? Suppose we would simulate a time series based on this belief, that is, based on this approximation of the conditional expectation. Suppose we would ask agents whether—faced with this simulation—they would like to update the value of η_n. By construction the answer is no. That is, the series generated are consistent with the beliefs on which they are based. This sounds a lot like a true rational expectations equilibrium. But not completely! There are two reasons. The first reason is that the simulation may be too short to pin down the true value of η_n^*.[6] The second (and more important) reason

[6] This reason (and the next) also apply when we use the non-stochastic version of PEA. That is, η_n^* may be the best value on the chosen grid, but this does not mean that it would be best value if more grid points would be considered.

it is not exactly like the definition of a rational expectations equilibrium is that in the numerical procedure we restrict agents to use a particular functional form to form beliefs. Suppose, we use a first-order approximation and we have found a fixed point. *Conditional on using a linear forecasting rule* agents are perfectly happy with the coefficients of this rule. But that doesn't mean that they couldn't do better with a higher-order polynomial. For a true rational expectations solution, agents do not want to update the coefficients of the function they are using *and* they do not want to use any alternative functional form.

4.4.2 Convergence to a rational expectations equilibrium?

In practice, we start with a first-order approximation and when we have solved for a first-order solution, then we solve for a second-order approximation. We continue until the properties of the policy functions, or the properties we are interested in, do not change anymore. The big question is whether we indeed converge towards the true rational expectations solution if we follow this procedure. Weierstrass theorem seems promising but there are two problems. First, Weierstrass theorem doesn't give us a roadmap on how to find the polynomial that is arbitrarily close. It just says there is one. We already encountered this problem in the chapter on function approximation. If equidistant nodes are used then polynomials will not converge when the sup norm is used, in fact they would diverge. To guarantee convergence one had to use other nodes such as Chebyshev nodes. But the problem here is a lot trickier. When we discussed function approximation then the true function values at the nodes were given. That is not the case here. That is, here we cannot evaluate the true values of $g(k_t, z_t)$ at a set of nodes (or time series observations). We have to use our approximation to construct the values of what we are trying to approximate. Thus, if we go from a first-order to a second-order approximation then the target changes as well. That is, there is feedback from the approximation (i.e., the beliefs) onto what is being approximated.

This doesn't mean that it cannot be proven that convergence to the rational expectation will take place. In fact, Marcet and Marshall (1994) show exactly this for a certain class of functions. But it is a lot trickier to prove than to invoke the standard convergence results from function approximation.

4.4.3 Comparison of stochastic and non-stochastic PEA

There are two key differences between stochastic and non-stochast PEA. Those are the numerical integration procedure and the construction of the grid. Let's discuss those in turn.

Numerical integration procedure

Stochastic PEA (implicitly) uses pseudo Monte-Carlo integration[7] to calculate the conditional expectation. This is a really inefficient integration procedure and there is no good reason to use it. Above we saw that it is very easy to explicitly calculate the conditional expectation. Note that this can also be done on a simulated time path. Calculating the conditional expectation explicitly not only gets rid of the (slowly disappearing) sampling noise, it also allows one to take a monotone transformation of the regressand and approximating polynomial and change a non-linear projection problem into a linear one. If the model has many stochastic shocks then at some point Gaussian quadrature methods would become unfeasible because of the curse of dimensionality. The most logical alternative would then be to use quasi Monte-Carlo procedures. The terminology "quasi Monte-Carlo" is somewhat misleading because the idea of these procedures is not to mimic random numbers. The idea of these algorithms is to fill high-dimensional spaces in an efficient way. These procedures resemble (pseudo) Monte Carlo procedures in that they calculate the integral as a simple average of the integrand at the generated observations.

Construction of the grid

The grid in stochastic PEA is constructed endogeneously. Especially if a bad initial guess is used then the points used in the projection step may be not at in the relevant range. Probably worse is that by using simulated points we get points that are clustered around the mean, whereas precision increases by spreading out the nodes towards the boundary. This was exactly the reason why Chebyshev nodes ensure uniform convergence and equidistrant nodes do not. There is one case, however, where it could very well be advantageous not to use spread out points in the projection step. And that is when the approximating function is grossly misspecified. For example, suppose that the truth has substantial nonlinearities but that one uses a linear approximation. Then clearly one will do poorly somewhere. By using points close to the average one may do better where one would like to do well namely where the economy spends most of the time. Personally I do not find this argument too convincing for my own work because it is rare in macroeconomics that functions cannot be approximated well with relatively low-order polynomials.

There is one other reason why rectilinear grids may not work well. That is when variables are not defined everywhere. For example, it is possible that at some grid points the capital choice is negative using the true consumption function. This can still be internally consistent if the economy never gets to such a grid point. But especially on the transition path towards the

[7] Pseudo Monte-Carlo integration procedures are procedures that rely on a computer algorithm to generate random numbers as opposed to true random numbers.

fixed point, one may easily encounter problematic grid points. Working with spheres instead of rectilinear grids may help but may not be enough to avoid such problems. In my experience such points are characterized by unlikely combinations of the state variables such as high capital stocks and for example low productivity or (in matching models) low unemployment. It is possible that the simulated data endogenously avoid such problematic nodes.