Lecture Notes 9

Minimum Weighted Residual Methods

The minimum weighted residual method has been introduced in economics by Judd [1992]. As for the PEA algorithm, it may receive an interpretation in terms of a generalized method of undetermined coefficients. However, here stops the comparison, as the philosophy of minimum weighted residual methods differs fundamentally from that of PEA as it will become clear in a moment. Nevertheless, as for the PEA algorithm, the basic idea of the method is to approximate either the decision rule — in Judd's original implementation — or the expectation function of the individuals — as proposed by Christiano and Fisher [2000] — by a smooth function, which in most of the cases will combine orthogonal polynomials. The parameters of the function are then revealed by imposing identifying restrictions dictated by economic theory.

9.1 The Basic idea

9.1.1 Stating the problem

The minimum weighted residual method may be implemented to solve a large set of models that admit the following general representation

$$E_t F(y_{t+1}, x_{t+1}, y_t, x_t, \varepsilon_{t+1}) = 0 \tag{9.1}$$

where $F : \mathbb{R}^{n_y} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_e} \longrightarrow \mathbb{R}^{n_y+n_x}$ describes the model. ε_t is the set of innovations of the structural shocks that hit the economy. The solution of the model is a set of decision rules for the control variables

$$y_t = g(x_t; \theta)$$

that define the next period state variables as

$$x_{t+1} \equiv h(x_t, y_t, \varepsilon_{t+1}) = h(x_t, g(x_t; \theta), \varepsilon_{t+1})$$

such that the model can be rewritten

$$E_t R(x_t, \varepsilon_{t+1}; g, \theta) = 0 \tag{9.2}$$

The idea of the minimum weighted residual method is to replace the true decision rule by a parametric approximation function, $\Phi(xt; \theta)$, of the current state variables x_t and a vector of parameters θ. Therefore, in its original implementation the minimum weighted residual method differs from the PEA algorithm in that we are not seeking directly an expectation function but a decision rule.[1] The problem is then to find a vector of parameters θ such that when the agents use the so defined "rule of thumb", $E_t F(x_t, \varepsilon_{t+1}; g, \theta)$ can be made as *"small"* as possible. But, *What do we mean by "small"?* In fact we want to find a vector of parameters $\widehat{\theta}$ such that

$$\| E_t R(x_t, \varepsilon_{t+1}; g, \{\theta\}_{i=0}^n) \|_\omega = 0$$

which corresponds to

$$\int_{\mathcal{X}} E_t R(x_t, \varepsilon_{t+1}; g, \theta) \omega_i(x) \mathrm{d}x = 0 \text{ for } i = 0, \ldots, n \tag{9.3}$$

where $\omega(x)$ is a weighting function, which operationalizes the notion of "small".

[1] Note however that we will present a variant of the method that attempts to reveal the expectation instead.

9.1.2 Implementation

Choosing an implementation of the minimum weighting residual method basically amounts to make 3 choices

1. the choice of a family of approximating functions,
2. the choice of the weighting function,
3. the choice of the method to approximate the integral involved by *(i)* the rational expectation and *(ii)* the identifying restriction (9.3).

Choosing a family of approximating functions: The choice of a family of approximating function defines the class of minimum weighting residual method we use. It is commonly admitted to range the method in two classes:

- Finite element methods,
- Spectral methods.

The *Finite element methods* can be viewed as a piecewise application of the minimum weighted residual method. This approach assumes that the decision rule can be accurately represented by a collection of small elements of finite dimensions called as *Finite Elements*. The original decision rule is then considered as an assemblage of these elements. These elements are connected to each other at joints called as nodes (or nodal points) to form the entire decision rule. Therefore, the first step involved in solving the model using finite elements is to divide the state space into disjoint intervals — the elements. Then the method will essentially amount to fit low order polynomials or spline functions on each subintervals. Once an approximation is obtained over each subinterval, all approximations are then pieced to form the decision rule over the whole domain of the state space we consider. Typical finite elements are

$$\Phi(x_i) = \begin{cases} \frac{x-x_{i-1}}{x_i-x_{i-1}} & \text{if } x \in [x_{i-1}; x_i] \\ \frac{x_{i+1}-x}{x_{i+1}-x_i} & \text{if } x \in [x_i; x_{i+1}] \\ 0 & \text{otherwise.} \end{cases}$$

we therefore recognize spline functions. McGrattan [1996] and McGrattan [1999] discuss and present the implementation of finite element methods in various economic models.

The *spectral methods* instead make use of higher order polynomials — usually orthogonal polynomials — but do not divide the domain of approximation into subintervals. The approximation is therefore conducted over the whole interval, which requires the decision rule to be continuous and differentiable. Therefore, in this case, the typical approximation will look like

$$y_t = \sum_{i=0}^{p} \theta_i \psi_i(x_t)$$

where $\{\psi_i(.)\}_{i=0}^{n}$ is a family of orthogonal polynomials.

Choosing a weighting function: The choice of the weighting function is extremely important in that it will define the method we will use. Traditionally, we may define 3 broad classes of method, depending on the weighting function we use:

1. The *Least square method* sets

$$\omega_i(x) = \frac{\partial R(x_t, \varepsilon_{t+1}; g, \theta)}{\partial \theta_i}$$

such that the problem amounts to find $\widehat{\theta}$ such that

$$\widehat{\theta} \in \underset{\theta}{\text{Argmin}} \int_{\mathcal{X}} E_t R(x_t, \varepsilon_{t+1}; \theta)^2 \mathrm{d}x_t$$

2. The *collocation method* uses the *Dirac function* as a weighting function, such that

$$\omega(x) = \delta(x - x_i) = \begin{cases} 1 \text{ if } x = x_i \\ 0 \text{ if } x \neq x_i \end{cases}$$

Therefore, by selecting this weighting function we set the residuals to zero on the nodes $x_1, \ldots, x_n$ — the collocation points — such that the

approximation is exact on these nodes. Note that nothing is imposed outside these nodes. When orthogonal polynomials are selected, these nodes are the roots of the polynomial of the highest order and the method is called *orthogonal collocation*. Note that in this approximation method, we only need $n+1$ collocation points to identify the $n+1$ elements of θ, otherwise the system is over–identified.

3. The *Galerkin method* uses the basis functions as weighting function. For instance, if the approximation is a linear combination of Chebychev polynomials of order 0 to n, the weighting functions are given by Chebychev polynomials of order 0 to n:

$$\omega_i(x) = \psi_t(x)$$

In other words, this amounts to impose the orthogonality of the residuals vis a vis the basis functions we are using. We are actually making use of the fact that a continuous function is identically zero if it is orthogonal to each element of a complete set of functions.

Choosing an integration method: We shall consider two problems here:

1. The problem of the rational expectation: In this case, everything depends on the form of the process for the shocks. If we are to use Markov chains, the integration problem is highly simplified as it only involves a discrete sum. If we use continuous support for the shocks, the choice of an integration method is dictated by the type of distribution we are assuming. In most of the cases, we will use gaussian shocks such that we will rely on Gauss–Hermite quadrature methods described in lecture notes # 4.

2. The problem of the inner product (equation (9.3)): In this case everything depends on the approximation method. If we are to use a collocation method, no integration method is to be used as collocation

amounts to impose that the residuals are zero at each node, such that (9.3) reduces to

$$E_t R(x_i, \varepsilon_{t+1}; g, \{\theta\}_{i=0}^n) = 0 \text{ for } i = 1, \ldots, n$$

When Least square methods are used, it is often the case that Legendre quadrature will do the job. When a Galerkin method is selected, this will often be the right choice too. However, when we are to use Chebychev polynomials, a Chebychev quadrature is in order provided each weighting function is defined as[2]

$$\omega_i(x) = \frac{T_i(\varphi(x))}{\sqrt{1-\varphi(x)^2}}$$

where $\varphi(x)$ is a linear function mapping the domain of x into [-1;1].

9.2 Practical implementation

In the sequel, we will essentially discuss the collocation and Galerkin implementations of a spectral method using Chebychev polynomials as they seem to be the most popular implementation of this method.[3]

9.2.1 The Collocation method

In this section, we present the collocation method in its simplest form. We will start by presenting the general algorithm, when Chebychev polynomials are used, and then present as an example the stochastic optimal growth model. For the moment, let us assume that we want to solve a rational expectation model that writes as

$$E_t R(x_t, \varepsilon_{t+1}; g, \theta) = 0$$

[2]Remember that Chebychev quadrature computes integrals of the form

$$\int_{-1}^{1} F(x) \frac{1}{\sqrt{1-x^2}} \mathrm{d}x$$

[3]The least square method can be straightforwardly implemented minimizing the implicit objective function we discussed.

and we want to find an approximating function for the decision rule $g(x_t)$ over the domain $[\underline{x}, \overline{x}]$. Assume that we take as approximating function

$$\Phi(x_t, \theta) \equiv \sum_{i=0}^{n} \theta_i T_i(\varphi(x_t))$$

where $T_i(.)$ is the Chebychev polynomial of order $i = 0, \ldots, n$. The problem is then to find the vector θ, such that

$$E_t R(x_i, \varepsilon_{t+1}; \Phi, \theta) = 0 \text{ for } i = 0, \ldots, n$$

The algorithm is then as follows[4]

1. Choose an order of approximation n, compute the $n+1$ roots of the Chebychev polynomial of order $n+1$ as

   $$z_i = \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i = 1, \ldots, n+1$$

   and formulate an initial guess for θ.

2. Compute x_i as

   $$x_i = \underline{x} + (z_i + 1)\frac{(\overline{x} - \underline{x})}{2} \text{ for } i = 1, \ldots, n+1$$

   to map [-1;1] into $[\underline{x}; \overline{x}]$.

3. Compute

   $$E_t R(x_i, \varepsilon_{t+1}; g, \theta) \text{ for } i = 1, \ldots, n+1$$

4. if it is close enough to zero then stop and form

   $$\Phi(x_t, \theta) \equiv \sum_{i=0}^{n} \theta_i T_i(\varphi(x_t))$$

   else update θ and go back to 3.

[4]We will discuss the one dimensional case. However, the multidimensional case will be illustrated in an example. You are also referred to lecture notes #3 which presented multidimensional approximation technics

The updating scheme for θ will typically be given by a non–linear solver of the type described in lecture notes #5. The computation of the integral involved by the rational expectation will depend on the process we assumed for the shocks. In order to better understand it, let us discuss the implementation of the collocation method on the stochastic optimal growth model both in the case of a Markov chain, and an AR(1) process.

The stochastic OGM (Markov chain) : We will actually consider a OGM in which the production function writes

$$y_t = \exp(a_t) k_t^{\alpha}$$

where a_t can take on two values $[\underline{a}, \overline{a}]$ with a transition matrix

$$\Pi = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$$

The Euler equation is given by

$$c_t^{-\sigma} - E_t \left[c_{t+1}^{-\sigma} \left(\alpha \exp(a_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta \right) \right] = 0$$

and the law of motion of capital

$$k_{t+1} = \exp(a_t) k_t^{\alpha} - c_t + (1-\delta) k_t$$

But, as a matter of fact, by taking the definition of the Markov chain into account, the model may be restated as the following system

$$\begin{aligned}
c_t(\underline{a})^{-\sigma} - \beta p \left[c_{t+1}(\underline{a})^{-\sigma} \left(\alpha \exp(\underline{a}) k_{t+1}(\underline{a})^{\alpha-1} + 1 - \delta \right) \right] & \\
-\beta(1-p) \left[c_{t+1}(\overline{a})^{-\sigma} \left(\alpha \exp(\overline{a}) k_{t+1}(\overline{a})^{\alpha-1} + 1 - \delta \right) \right] &= 0 \\
c_t(\overline{a})^{-\sigma} - \beta(1-p) \left[c_{t+1}(\underline{a})^{-\sigma} \left(\alpha \exp(\underline{a}) k_{t+1}(\underline{a})^{\alpha-1} + 1 - \delta \right) \right] & \\
-\beta p \left[c_{t+1}(\overline{a})^{-\sigma} \left(\alpha \exp(\overline{a}) k_{t+1}(\overline{a})^{\alpha-1} + 1 - \delta \right) \right] &= 0 \\
k_{t+1}(\underline{a}) - \exp(\underline{a}) k_t^{\alpha} + c_t(\underline{a}) - (1-\delta) k_t &= 0 \\
k_{t+1}(\overline{a}) - \exp(\overline{a}) k_t^{\alpha} + c_t(\overline{a}) - (1-\delta) k_t &= 0
\end{aligned}$$

where the notation $c_t(\underline{a})$ (resp. $c_t(\overline{a})$) stands for the fact that the consumption decision is contingent on the realization of the shock, $\underline{a}$ (resp. $\overline{a}$), likewise for $k_{t+1}(\underline{a})$ (resp. $k_{t+1}(\overline{a})$).

It therefore appears that we actually have only two decision rules to compute, one for $c_t(\underline{a})$ and one for $c_t(\overline{a})$, each taking the form[5]

$$c_t(\underline{a}) \simeq \Phi(k_t, \theta(\underline{a})) \equiv \exp\left(\sum_{j=0}^{n} \theta_j(\underline{a}) T_j(\varphi(\log(k_t)))\right) \tag{9.4}$$

$$c_t(\overline{a}) \simeq \Phi(k_t, \theta(\overline{a})) \equiv \exp\left(\sum_{j=0}^{n} \theta_j(\overline{a}) T_j(\varphi(\log(k_t)))\right) \tag{9.5}$$

where the notation $\theta(a)$ accounts for the fact that the form of the decision rule may differ depending on the realization of the shock, a. We may also select different order of approximation for the two rules, but in order to keep things simple we chose to impose the same order. The algorithm then works as follows.

1. Choose an order of approximation n, compute the $n+1$ roots of the Chebychev polynomial of order $n+1$ as

$$z_i = \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i = 1, \ldots, n+1$$

   and formulate an initial guess for $\theta(\underline{a})$ and $\theta(\overline{a})$.

2. Compute k_i as

$$k_i = \exp\left(\log(\underline{k}) + (z_i + 1)\frac{\log(\overline{k}) - \log(\underline{k})}{2}\right) \text{ for } i = 1, \ldots, n+1$$

   to map [-1;1] into $[\underline{k}; \overline{k}]$.

[5]The exponential form was imposed in order to guarantee positivity of consumption.

3. Compute

$$\begin{aligned} c_t(\underline{a}) &\simeq \Phi(k_t, \theta(\underline{a})) \equiv \exp\left(\sum_{j=0}^{n} \theta_j(\underline{a}) T_j(\varphi(\log(k_i)))\right) \\ c_t(\overline{a}) &\simeq \Phi(k_t, \theta(\overline{a})) \equiv \exp\left(\sum_{j=0}^{n} \theta_j(\overline{a}) T_j(\varphi(\log(k_i)))\right) \end{aligned}$$

and

$$\begin{aligned} k_{t+1}(k_i, \underline{a}) &= \exp(\underline{a}) k_i^{\alpha} - \Phi(k_i, \theta(\underline{a})) + (1-\delta) k_i \\ k_{t+1}(k_i, \overline{a}) &= \exp(\overline{a}) k_i^{\alpha} - \Phi(k_i, \theta(\overline{a})) + (1-\delta) k_i \end{aligned}$$

at each node k_i, $i = 1, \ldots, n+1$.

4. Then compute the possible levels of future consumption

a_t	a_{t+1}	Consumption
$\underline{a}$	$\underline{a}$	$\Phi(k_{t+1}(k_i, \underline{a}), \theta(\underline{a})) \equiv \sum_{j=0}^{n} \theta_j(\underline{a}) T_j(\varphi(k_{t+1}(k_i, \underline{a})))$
$\underline{a}$	$\overline{a}$	$\Phi(k_{t+1}(k_i, \underline{a}), \theta(\overline{a})) \equiv \sum_{j=0}^{n} \theta_j(\overline{a}) T_j(\varphi(k_{t+1}(k_i, \underline{a})))$
$\overline{a}$	$\underline{a}$	$\Phi(k_{t+1}(k_i, \overline{a}), \theta(\underline{a})) \equiv \sum_{j=0}^{n} \theta_j(\underline{a}) T_j(\varphi(k_{t+1}(k_i, \overline{a})))$
$\overline{a}$	$\overline{a}$	$\Phi(k_{t+1}(k_i, \overline{a}), \theta(\overline{a})) \equiv \sum_{j=0}^{n} \theta_j(\overline{a}) T_j(\varphi(k_{t+1}(k_i, \overline{a})))$

5. Evaluate the residuals

$$\begin{aligned} R(k_i, \underline{a}; \theta) &= \Phi(k_i, \theta(\underline{a}))^{-\sigma} - \beta p \Psi(k_i, \underline{a}, \underline{a}) - \beta(1-p)\Psi(k_i, \underline{a}, \overline{a}) \\ R(k_i, \overline{a}; \theta) &= \Phi(k_i, \theta(\overline{a}))^{-\sigma} - \beta(1-p)\Psi(k_i, \overline{a}, \underline{a}) - \beta p \Psi(k_i, \overline{a}, \overline{a}) \end{aligned}$$

where

$$\Psi(k_i, a_t, a_{t+1}) \equiv \Phi(k_{t+1}(k_i, a_t), \theta(a_{t+1}))^{-\sigma} \left(\alpha \exp(a_{t+1}) k_{t+1}(k_i, a_t)^{\alpha-1} + 1 - \delta\right)$$

for all $i = 1, \ldots, n$.

6. if all residuals are close enough to zero then stop, else update θ and go back to 3.

From a practical point of view, the last step is performed using a Newton algorithm. Initial conditions can be obtained from a linear approximation of the model (see the matlab codes in directory `growth/collocmc`).

MATLAB CODE: COLLOCATION METHOD (OGM, MAIN CODE)

```
clear all
global kmin ksup XX kt;             % need to set some parameters
global nstate nbk ncoef XX XT PI;   % globally

nbk     = 4;            % Degree of polynomials
nodes   = nbk+1;        % # of Nodes
nstate  = 2;            % # of possible states for technology shock
ncoef   = nbk+1;        % # of coefficients

delta   = 0.1;      % depreciation rate
beta    = 0.95;     % discount factor
alpha   = 0.3;      % capital elasticity
sigma   = 1.5;      % parameter of utility
%
% Steady state
%
ysk     =(1-beta*(1-delta))/(alpha*beta);
ksy     = 1/ysk;
ys      = ksy^(alpha/(1-alpha));
ks      = ys^(1/alpha);
is      = delta*ks;
cs      = ys-is;
%
% Markov Chain: technology shock (Tauchen Hussey)
%
rho     = 0.8;
se      = 0.2;
ma      = 0;
[agrid,wmat] = hernodes(nstate);
agrid        = agrid*sqrt(2)*se;
PI           = transprob(agrid,wmat,0,rho,se);
at           = agrid+ma;
%
% grid for the capital stock
%
kmin    = log(0.1);
ksup    = log(6);
rk      = rcheb(nodes);                 % roots
kt      = exp(itransfo(rk,kmin,ksup));  % grid
XX      = cheb(rk,[0:nbk]);             % Polynomials
%
% Initial Conditions
```

```
%
a0=repmat([-0.2 0.65 0.04 0 0]',nstate,1);
a0=a0(:);
%
% Main loop
%
param   = [alpha beta delta sigma]';
th      = fcsolve('residuals',a0,[],param,at);
th      = reshape(th,ncoef,nstate);
```

MATLAB CODE: RESIDUALS FUNCTION (OGM)

```
function res=residuals(theta,param,at);
global nbk kmin ksup XX kt PI nstate;

alpha   = param(1);
beta    = param(2);
delta   = param(3);
sigma   = param(4);
lt      = length(theta);
theta   = reshape(theta,lt/nstate,nstate);

RHS=[];
LHS=[];
for i=1:nstate;
   ct   = exp(XX*theta(:,i));                        % C(t)
   k1   = exp(at(i))*kt.^alpha+(1-delta)*kt-ct;      % k(t+1)
   rk1  = transfo(log(k1),kmin,ksup);                % k(t+1) in kmin,ksup
   xk1  = cheb(rk1,[0:nbk]);                         % polynomials
   %
   % Euler equation for all states
   %
   aux  = 0;
   for j=1:nstate;
    c1    = exp(xk1*theta(:,j));                     % c(t+1)
    %
    % Cumulates the left hand side of the Euler equation for all states
    %
    resid = beta*(alpha*exp(at(j))*k1.^(alpha-1)+1-delta).*c1.^(-sigma);
    aux   = aux+PI(i,j)*resid;
   end;
   RHS  = [RHS -sigma*log(ct)];
   LHS  = [LHS log(aux)];
end;

res=LHS-RHS;        % Note that residuals are taken in logs (easier)
res=res(:);
```

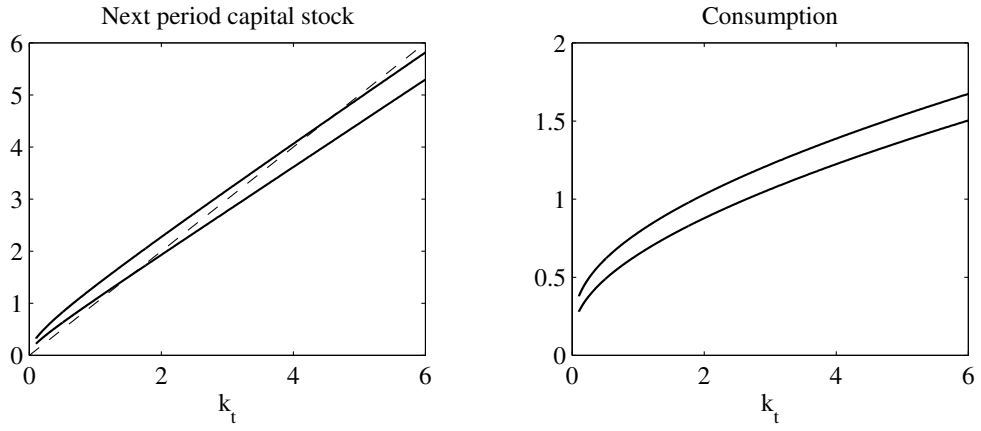
For the parameterization we use in the codes, and using 4–th order Chebychev polynomials, we obtain the decision rules reported in table 9.1. Figure

Table 9.1: Decision rules (OGM, Collocation)

θ_0	θ_1	θ_2	θ_3	θ_4
-0.2807	0.7407	0.0521	0.0034	-0.0006
-0.4876	0.8406	0.0546	0.0009	-0.0006

(9.1) reports the decision rules of consumption and next period capital stock. As can be seen, they are (hopefully) alike those obtained from the value iteration method (see lecture notes #7).

Figure 9.1: Decision rules (OGM, Collocation)



The AR(1) process : In this case, we assume that the technology shock is modelled as

$$a_{t+1} = \rho a_t + \varepsilon_{t+1}$$

where $\varepsilon \rightsquigarrow \mathcal{N}(0, \sigma_\varepsilon^2)$. The Euler equation is given by

$$c_t^{-\sigma} - E_t \left[c_{t+1}^{-\sigma} \left(\alpha \exp(a_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta \right) \right] = 0$$

and the law of motion of capital

$$k_{t+1} = \exp(a_t)k_t^{\alpha} - c_t + (1-\delta)k_t$$

But, as a matter of fact, by taking the definition of AR(1) process, the model may be restated as the following system

$$c_t^{-\sigma} - \frac{1}{\sqrt{2\pi\sigma_\varepsilon^2}} \int_{-\infty}^{\infty} \left[c_{t+1}^{-\sigma} \left(\alpha \exp(\rho a_t + \varepsilon_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta\right)\right] \exp\left(-\frac{\varepsilon_{t+1}^2}{2\sigma_\varepsilon^2}\right) \mathrm{d}\varepsilon_{t+1} = 0$$

$$k_{t+1} - \exp(a_t)k_t^{\alpha} + c_t - (1-\delta)k_t = 0$$

We therefore have to compute the integral in the Euler equation. However, since we have assumed a gaussian distribution for the shock we may use the Gauss–Hermite quadrature. Therefore, we make the following change of variables: $z = \varepsilon/\sqrt{2\sigma_\varepsilon}$, such that the Euler equation rewrites as

$$c_t^{-\sigma} - \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \left[c_{t+1}^{-\sigma} \left(\alpha \exp(\rho a_t + z\sqrt{2\sigma_\varepsilon}) k_{t+1}^{\alpha-1} + 1 - \delta\right)\right] \exp\left(-z^2\right) \mathrm{d}z = 0$$

which makes more explicit the need for a Gauss–Hermite quadrature. We then just have to compute the nodes and weight for the quadrature, such that the Euler equation rewrites

$$c_t^{-\sigma} - \frac{1}{\sqrt{\pi}} \sum_{j=1}^{q} \omega_j \left[c_{t+1}^{-\sigma} \left(\alpha \exp(\rho a_t + z_j\sqrt{2\sigma_\varepsilon}) k_{t+1}^{\alpha-1} + 1 - \delta\right)\right] = 0$$

We now have to formulate a guess for the decision rule. Note that since we have not specified particular values for the technology shock, we can introduce it into the approximating rule, such that we will take

$$c_t \simeq \Phi(k_t, a_t, \theta) \equiv \exp\left(\sum_{j_k=0}^{n_k} \sum_{j_a=0}^{n_a} \theta_{j_k j_a} T_{j_k}(\varphi(\log(k_t))) T_{j_a}(\varphi(a_t))\right)$$

Note that since we use collocation, the numbers of nodes should be equal to the total numbers of coefficients. It may then be much easier to work with tensor basis rather than complete polynomials in this case. The algorithm then works as follows.

1. Choose an order of approximation n_k and n_a for each dimension, compute the n_k+1 and n_a+1 roots of the Chebychev polynomial of order n_k+1 and n_a+1 as

$$\begin{aligned} z_k^i &= \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i=1,\ldots,n_k+1 \\ z_a^i &= \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i=1,\ldots,n_a+1 \end{aligned}$$

and formulate an initial guess for θ.

2. Compute k_i as

$$k_i = \exp\left(\log(\underline{k}) + (z_k^i+1)\frac{\log(\overline{k})-\log(\underline{k})}{2}\right) \text{ for } i=1,\ldots,n_k+1$$

to map [-1;1] into $[\underline{k};\overline{k}]$ and

$$a_i = \log(\underline{a}) + (z_a^i+1)\frac{\overline{a}-\underline{a}}{2} \text{ for } i=1,\ldots,n_a+1$$

to map [-1;1] into $[\underline{a};\overline{a}]$ and

3. Compute

$$c_t \simeq \Phi(k_i,a_j,\theta) = \exp\left(\sum_{j_k=0}^{n_k}\sum_{j_a=0}^{n_a}\theta_{j_kj_a}T_{j_k}(\varphi(\log(k_i)))T_{j_a}(\varphi(a_j))\right)$$

and

$$k_{t+1}(k_i,a_j) = \exp(a_j)k_i^\alpha - \Phi(k_i,a_j,\theta) + (1-\delta)k_i$$

at each node (k_i,a_j), $i=1,\ldots,n_k+1$ and $j=1,\ldots,n_a+1$.

4. Then, for each state (k_i,a_j), compute the possible levels of future consumption needed to compute the integral

$$\begin{aligned} c_{t+1} &\simeq \Phi(k_{t+1}(k_i,a_j),\rho a_j + z_\ell\sqrt{2\sigma_\varepsilon},\theta) \\ &\simeq \exp\left(\sum_{j_k=0}^{n_k}\sum_{j_a=0}^{n_a}\theta_{j_kj_a}T_{j_k}(\varphi(\log(k_{t+1}(k_i,a_j))))T_{j_a}(\varphi(\rho a_j + z_\ell\sqrt{2\sigma_\varepsilon}))\right) \end{aligned}$$

for $\ell=1,\ldots,q$.

5. Evaluate the residuals

$$R(k_i, a_j; \theta) = \Phi(k_i, a_j, \theta)^{-\sigma} - \frac{\beta}{\sqrt{\pi}} \sum_{\ell=1}^{q} \omega_\ell \Psi(k_i, a_j, z_\ell; \theta)$$

where

$$\Psi(k_i, a_j, z_\ell; \theta) \equiv \Phi(k_{t+1}(k_i, a_j), \rho a_j + z_\ell \sqrt{2\sigma_\varepsilon})^{-\sigma} \left(\alpha \exp(\rho a_j + z_\ell) k_{t+1}(k_i, a_j)^{\alpha-1} + 1 - \delta\right)$$

for all $i = 1, \ldots, n_k + 1$ and $i = 1, \ldots, n_a + 1$.

6. if all residuals are close enough to zero then stop, else update θ and go back to 3.

From a practical point of view, the last step is performed using a Newton algorithm. Initial conditions can be obtained from a linear approximation of the model (see the matlab codes in directory `growth/collocg`).

MATLAB CODE: COLLOCATION METHOD (STOCHASTIC OGM, MAIN CODE)

```
clear all
global nbk kmin ksup XK kt;             % parameters that will be common to
global nba amin asup XA at;             % several subfunctions
global nstate nodea nodek wmat wgrid;   %

nbk=4;              % Degree of polynomials (capital)
nba=2;              % Degree of polynomials (technology shock)
nodek=nbk+1;        % # of nodes for capital stock
nodea=nba+1;        % # of nodes for technology shock
nstate=12;          % # of nodes for Gauss-Hermite quadrature

delta   = 0.1;      % depreciation rate
beta    = 0.95;     % discount factor
alpha   = 0.3;      % capital elasticity
sigma   = 1.5;      % parameter of utility
rho     = 0.8;      % persistence of AR(1)
se      = 0.2;      % standard deviation of innovation
ma      = 0;        % average of the process
%
% deterministic steady state
%
ysk     =(1-beta*(1-delta))/(alpha*beta);
ksy     = 1/ysk;
ys      = ksy^(alpha/(1-alpha));
ks      = ys^(1/alpha);
```

```
is      = delta*ks;
cs      = ys-is;
ab      = 0;
%
% grid for the technology shocks
%
[wgrid,wmat]=hernodes(nstate);          % weights and nodes for quadrature
wgrid   = wgrid*sqrt(2)*se;
amin    = (ma+wgrid(nstate));
asup    = (ma+wgrid(1));
ra      = rcheb(nodea);                 % roots
at      = itransfo(ra,amin,asup);       % grid
XA      = cheb(ra,[0:nba]);             % Polynomials
%
% grid for the capital stock
%
kmin    = log(.1);
ksup    = log(6);
rk      = rcheb(nodek);                 % roots
kt      = exp(itransfo(rk,kmin,ksup));  % grid
XK      = cheb(rk,[0:nbk]);             % Polynomials
%
% Initial Conditions
%
a0      = [
  -0.23759592487257
   0.60814488103911
   0.03677400318790
   0.69025680170443
  -0.21654209984197
   0.00551243342828
   0.03499834613714
  -0.00341171507904
  -0.00449139656933
   0.00085302605779
   0.00285737302122
  -0.00002348542016
  -0.00011606672164
  -0.00003323351559
   0.00018045618825];
a0      = a0(:);
%
% main part!
%
param   = [alpha beta delta sigma ma rho]';
th      = fcsolve('residuals',a0,[],param);
```

MATLAB CODE: RESIDUALS FUNCTION (STOCHASTIC OGM)

```
function res=residuals(theta,param);

global nbk kmin ksup XK kt;
global nba amin asup XA at;
global nstate nodea nodek wmat wgrid;

alpha   = param(1);
beta    = param(2);
delta   = param(3);
sigma   = param(4);
ma      = param(5);
rho = param(6);

RHS=[];
LHS=[];
XX=[];
for i   = 1:nodek;
  for j=1:nodea;
    XX0 = makepoly(XA(j,:),XK(i,:));
    ct  = exp(XX0*theta);                                % consumption
    XX  = [XX;XX0];
    k1  = exp(at(j))*kt(i).^alpha+(1-delta)*kt(i)-ct;   % k(t+1)
    rk1 = transfo(log(k1),kmin,ksup);                    % log(k1) to [-1;1]
    xk1 = cheb(rk1,[0:nbk]);                             % Cheb. polynomials
    a1  = rho*at(j)+(1-rho)*ma+wgrid;                    % next period shock
    ra1 = transfo(a1,amin,asup);                         % a(t+1) to [-1;1]
    XA1 = cheb(ra1,[0:nba]);                             % Cheb. Polynomials
    XX1 = makepoly(XA1,xk1);
    c1  = exp(XX1*theta);                                % c(t+1)
    %
    % computes the integral
    %
    tmp = beta*(alpha*exp(a1)*k1.^(alpha-1)+1-delta).*c1.^(-sigma);
    aux = wmat'*tmp/sqrt(pi);
    RHS = [RHS;log(aux)];
    LHS = [LHS;log(ct.^(-sigma))];
  end
end;
res = LHS-RHS;
res = res(:);
```

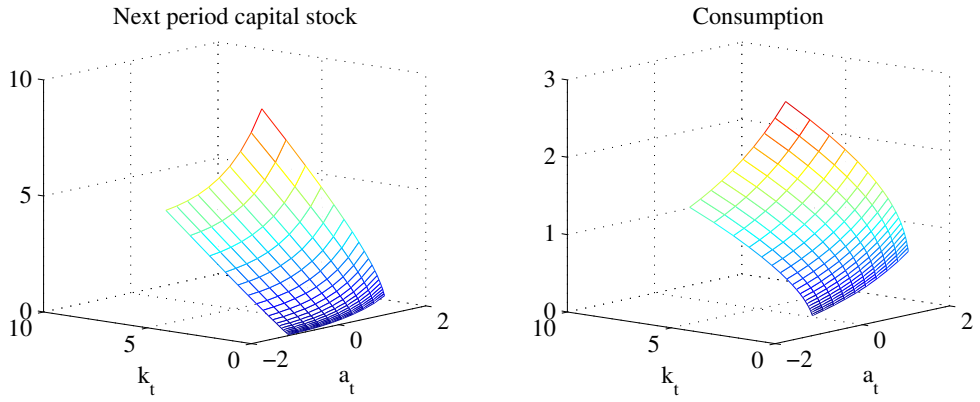
For the parameterization we use in the codes, and using 4–th order Chebychev polynomials for the capital stock and second order polynomials for the technology shock, we obtain the decision rules reported in table 9.2.

Table 9.2: Decision rule (Consumption, OGM, Collocation, AR(1))

	$T_0(k_t)$	$T_1(k_t)$	$T_2(k_t)$	$T_3(k_t)$	$T_4(k_t)$
$T_0(a_t)$	-0.3609	0.7992	0.0487	0.0019	-0.0003
$T_1(a_t)$	0.6474	-0.2512	-0.0096	0.0048	0.0001
$T_2(a_t)$	0.0338	0.0103	-0.0058	-0.0004	0.0004

Figure (9.2) reports the decision rules of consumption and next period capital stock.

Figure 9.2: Decision rules (OGM, Collocation, AR(1))



9.2.2 The Galerkin method

In this section, we present the Galerkin method. As in the previous section, we will start by presenting the general algorithm, when Chebychev polynomials are used, and then present as an example the stochastic optimal growth model. Assume that we want to solve a rational expectation model that writes as

$$E_t R(x_t, \varepsilon_{t+1}; g, \theta) = 0$$

We therefore want to find an approximating function for the decision rule $g(x_t)$ over the domain $[\underline{x}, \overline{x}]$. Assume that we take as approximating function

$$\Phi(x_t, \theta) \equiv \sum_{i=0}^{n} \theta_i T_i(\varphi(x_t))$$

where $T_i(.)$ is the Chebychev polynomial of order $i = 0, \ldots, n$. The problem is then to find the vector θ, such that

$$\int_{-1}^{1} E_t R(x, \varepsilon_{t+1}; \Phi, \{\theta_i\}_{i=0}^{n}) \omega_i(\varphi(x)) \mathrm{d}\varphi(x) = 0 \text{ for } i = 0, \ldots, n$$

where

$$\omega_i(x) = \frac{T_i(\varphi(x))}{\sqrt{1 - \varphi(x)^2}}$$

In fact, the integral may be evaluated using a Gauss–Chebychev quadrature, such that, as the weights of the Gauss–Chebychev quadrature are constant, the problem amounts to solve

$$\sum_{j=1}^{m} E_t R(x_j, \varepsilon_{t+1}; \Phi, \{\theta_i\}_{i=0}^{n}) T_i(\varphi(x_j)) = 0 \text{ for } i = 0, \ldots, n$$

which rewrites

$$\mathcal{T}(\varphi(x)) R(x, \varepsilon_{t+1}; \Phi, \theta) = 0$$

where

$$\mathcal{T}(x) = \begin{pmatrix} T_0(\varphi(x_1)) & \ldots & T_0(\varphi(x_m)) \\ \vdots & \ddots & \vdots \\ T_n(\varphi(x_1)) & \ldots & T_n(\varphi(x_m)) \end{pmatrix}$$

The algorithm then works as follows[6]

1. Choose an order of approximation n, compute the $m > n$ roots of the Chebychev polynomial of order $m > n$ as

$$z_i = \cos\left(\frac{(2i-1)\pi}{2m}\right) \text{ for } i = 1, \ldots, m$$

and formulate an initial guess for θ.

[6]We will discuss the one dimensional case. However, the multidimensional case will be illustrated in an example. You are also referred to lecture notes #3 which presented multidimensional approximation technics

2. Compute the matrix $\mathcal{T}(x)$ as

$$\mathcal{T}(x) = \begin{pmatrix} T_0(z_1) & \dots & T_0(z_m) \\ \vdots & \ddots & \vdots \\ T_n(z_1) & \dots & T_n(z_m) \end{pmatrix}$$

3. Compute x_i as

$$x_i = \underline{x} + (z_i + 1)\frac{(\overline{x} - \underline{x})}{2} \text{ for } i = 1, \dots, n+1$$

to map [-1;1] into $[\underline{x}; \overline{x}]$.

4. Compute

$$E_t R(x_i, \varepsilon_{t+1}; g, \theta) \text{ for } i = 1, \dots, n+1$$

and evaluate

$$\mathcal{T}(\varphi(x)) R(x, \varepsilon_{t+1}; \Phi, \theta) = 0$$

5. if it is close enough to zero then stop and form

$$\Phi(x_t, \theta) \equiv \sum_{i=0}^{n} \theta_i T_i(\varphi(x_t))$$

else update θ and go back to 3.

As before, the updated θ in steps 4 and 5 will typically be obtained from a non–linear solver of the type described in lecture notes #5. Likewise, the computation of the integral involved by the rational expectation will depend on the process we assumed for the shocks. In order to better understand it, and as in the previous section, we will discuss the implementation of the method on the stochastic optimal growth model both in the case of a Markov chain, and an AR(1) process.

The stochastic OGM (Markov chain) : We consider the same OGM as in the previous section, such that the production function writes

$$y_t = \exp(a_t) k_t^{\alpha}$$

where a_t can take on two values $[\underline{a}, \overline{a}]$ with a transition matrix

$$\Pi = \begin{pmatrix} p & 1-p \\ 1-p & p \end{pmatrix}$$

The Euler equation is given by

$$c_t^{-\sigma} - E_t \left[c_{t+1}^{-\sigma} \left(\alpha \exp(a_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta \right) \right] = 0$$

and the law of motion of capital

$$k_{t+1} = \exp(a_t) k_t^{\alpha} - c_t + (1-\delta) k_t$$

In fact, the algorithm is exactly the same up to some slight differences that I will just mention:

1. Rather than evaluating the residuals on $n+1$ nodes, we evaluate them on $m > n$ nodes.

2. The functions we want to set to zero are not the residuals anymore, as we have to compute the integral. Therefore, the vector θ solves

$$\begin{aligned} \mathcal{T}(z) R(k, \underline{a}; \theta) &= 0 \\ \mathcal{T}(z) R(k, \overline{a}; \theta) &= 0 \end{aligned}$$

The matlab codes associated to the Galerkin method are basically the same as the ones for the collocation method, up to some minor differences. We were using a greater number of nodes (20) such that in the main code, the line `nodes=nbk;` now reads `nodes=20;`, and the residuals are projected on the chebychev matrix, such that the codes for the residuals is slightly modified, as the line `res=LHS-RHS;` is now replaced by `res=XX'*(LHS-RHS);`. Applying these modifications, we end up with the decision rules reported in table 9.3. These decision rules do not differ much from those obtained from collocation at the 4 digits precision, they actually differ if we report numbers with a greater precision. This actually illustrates the remarkable capability of the collocation method to capture the main features of the decision rule in this particular example (this is actually related to the smoothness of the decision rules). This would not be the case for other more sophisticated models.

Table 9.3: Decision rules (OGM, Galerkin)

θ_0	θ_1	θ_2	θ_3	θ_4
-0.2807	0.7407	0.0521	0.0034	-0.0005
-0.4876	0.8406	0.0546	0.0009	-0.0006

The AR(1) process : Like in the case of a Markov chain, the matlab codes associated to the Galerkin method are basically the same as the ones for the collocation method, up to some minor differences. We were using a greater number of nodes (20 for capital and 10 for the shock) such that in the main code, the line `nodek=nbk+1;` and `nodea=nba+1;` now read `nodek=20;` and `nodea=10;`, and the residuals are projected on the chebychev matrix, such that the codes for the residuals is slightly modified, as the line `res=LHS-RHS;` is now replaced by `res=XX'*(LHS-RHS);`. Further, we used *complete basis* rather than tensor basis (see the `makepoly.m` function)

Table 9.4: Decision rule parameters

	$T_0(k)$	$T_1(k)$	$T_2(k)$	$T_3(k)$	$T_4(k)$
$T_0(a)$ -0.359970	0.647039	0.032769	0.799127	-0.252257	
$T_1(a)$ 0.009890	0.048143	-0.009799	-0.005171	–	
$T_2(a)$ 0.001764	0.004738	-0.000247	–	–	

9.3 The PEA modification

The PEA modification to minimum weighted residual method has recently been proposed by Christiano and Fisher [2000]. The basic idea to this modification is basically to apply exactly the same technic as the one we just described to the expectation function rather than the decision rule. One very attractive feature of this modification is that it will enable us to handle, as easily as in the case of PEA, problems in which there is a possibly binding constraint. Since, this modification is only a change in the function we are approximating, I will not elaborate any further on the method — which is the exact application of the previously described methods to the expectation function— but will rather provide detailed examples of application, focusing on the Galerkin method.

9.3.1 The optimal growth model

Let us first recall the type of problem we have in hand. We are about to solve the set of equations

$$\begin{aligned}
\lambda_t - \beta E_t \left[\lambda_{t+1}\left(\alpha \exp(a_{t+1})k_{t+1}^{\alpha-1} + 1 - \delta\right)\right] &= 0 \\
c_t^{-\sigma} - \lambda_t &= 0 \\
k_{t+1} - \exp(a_t)k_t^{\alpha} + c_t - (1-\delta)k_t &= 0 \\
a_{t+1} - \rho a_t - \varepsilon_{t+1} &= 0
\end{aligned}$$

Our problem will therefore be to get an approximation for the expectation function:

$$\beta E_t \left[\lambda_{t+1}\left(\alpha \exp(a_{t+1})k_{t+1}^{\alpha-1} + 1 - \delta\right)\right]$$

In this problem, we will deal with a continuous AR(1) process,[7] such we have 2 state variables: k_t and a_t, such that $\Phi(.)$ should be a function of both k_t and

[7] See directory `growth/peagalmc` for the matlab code with a Markov chain.

a_t. Like in the standard Galerkin procedure, we will use a guess of the form

$$\Phi(k_t, a_t, \theta) \equiv \exp\left(\sum_{\substack{j_k=0,\ldots,n_k \\ j_a=0,\ldots,n_a \\ j_a+j_k \leqslant \max(n_k,n_a)}} \theta_{j_k j_a} T_{j_k}(\varphi(\log(k_t))) T_{j_a}(\varphi(a_t))\right)$$

therefore imposing a *complete basis* of polynomials. Then the algorithm is pretty much the same as the one for standard Galerkin procedure with an AR(1) process.

1. Choose an order of approximation n_k and n_a for each dimension, compute the $m_k > n_k$ ($n_k + 1$ if you use collocation) and $m_a > n_a$ ($n_a + 1$ for collocation) roots of the Chebychev polynomial of order m_k and m_a as

$$\begin{aligned} z_k^i &= \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i = 1, \ldots, m_k \\ z_a^i &= \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i = 1, \ldots, m_a \end{aligned}$$

   and formulate an initial guess for θ.

2. Compute k_i as

$$k_i = \exp\left(\log(\underline{k}) + (z_k^i + 1)\frac{\log(\overline{k}) - \log(\underline{k})}{2}\right) \text{ for } i = 1, \ldots, m_k$$

   to map [-1;1] into $[\underline{k}; \overline{k}]$ and

$$a_i = \log(\underline{a}) + (z_a^i + 1)\frac{\overline{a} - \underline{a}}{2} \text{ for } i = 1, \ldots, m_a$$

   to map [-1;1] into $[\underline{a}; \overline{a}]$ and

3. Compute the approximating expectation function at each node (k_i, a_j), $i = 1, \ldots, m_k$ and $j = 1, \ldots, m_a$:

$$\Phi(k_i, a_j, \theta)$$

4. Deduce the level of consumption, which, from the Euler equation, is given by

$$c_t(k_i, a_j, \theta) = \Phi(k_i, a_j, \theta)^{-1/\sigma}$$

and the capital stock

$$k_{t+1}(k_i, a_j) = \exp(a_j)k_i^{\alpha} - c_t(k_i, a_j, \theta) + (1-\delta)k_i$$

5. Then, for each state (k_i, a_j), compute the possible levels of future consumption needed to compute the integral by computing the expectation function in $t+1$

$$\Phi(k_{t+1}(k_i, a_j), \rho a_j + z_\ell\sqrt{2\sigma_\varepsilon}, \theta)$$

and the consumption in $t+1$

$$c_t(k_{t+1}(k_i, a_j), \rho a_j + z_\ell\sqrt{2\sigma_\varepsilon}, \theta) = \Phi(k_{t+1}(k_i, a_j), \rho a_j + z_\ell\sqrt{2\sigma_\varepsilon}, \theta)^{-1/\sigma}$$

for $\ell = 1, \ldots, q$

6. Evaluate the residuals ($\Phi(k_i, a_j, \theta)$ minus the expectation function)

$$R(k_i, a_j; \theta) = \Phi(k_i, a_j, \theta) - \frac{\beta}{\sqrt{\pi}}\sum_{\ell=1}^{q}\omega_\ell\Psi(k_i, a_j, z_\ell; \theta)$$

where

$$\Psi(k_i, a_j, z_\ell; \theta) \equiv \Phi(k_{t+1}(k_i, a_j), \rho a_j + z_\ell\sqrt{2\sigma_\varepsilon})^{-\sigma}\left(\alpha\exp(\rho a_j + z_\ell)k_{t+1}(k_i, a_j)^{\alpha-1} + 1 - \delta\right)$$

for all $i = 1, \ldots, m_k$ and $i = 1, \ldots, m_a$.

7. if all residuals (projected on polynomials in the case of standard Galerkin) the are close enough to zero then stop, else update θ and go back to 3.

The implementation of this version of the problem is extremely close to the standard Galerkin (collocation) procedure, such that you are left to the matlab code (in `/growth/peagalg` and `/growth/peagalmc` directories).

At this point, it is not clear why the PEA modification to the minimum weighted residual method could be interesting, as it does not differ so much from its standard implementation. However, it has the attractive feature of being able to deal with binding constraints extremely easily. As an example of implementation, we shall now study the stochastic optimal growth model with an irreversibility in investment decision.[8]

9.3.2 Irreversible investment

We now consider a variation to the previous model, in the sense that we restrict gross investment to be positive in each and every period:

$$i_t \geqslant 0 \Longleftrightarrow k_{t+1} \geqslant (1-\delta)k_t \tag{9.6}$$

This assumption amounts to assume that there does not exist a second hand market for capital. In such a case the problem of the central planner is to determined consumption and capital accumulation, such that utility is maximum:

$$\max_{\{c_t,k_{t+1}\}_{t=0}^{\infty}} E_0 \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}-1}{1-\sigma}$$

s.t.

$$k_{t+1} = \exp(a_t)k_t^{\alpha} - c_t + (1-\delta)k_t$$

and

$$k_{t+1} \geqslant (1-\delta)k_t$$

Forming the Lagrangean associated to the previous problem, we have

$$\mathcal{L}_t = E_t \sum_{\tau=0}^{\infty} \beta^{\tau} \left[\frac{c_{t+\tau}^{1-\sigma}-1}{1-\sigma} + \lambda_{t+\tau}\left(\exp(a_{t+\tau})k_{t+\tau}^{\alpha} - c_{t+\tau} + (1-\delta)k_{t+\tau} - k_{t+\tau+1}\right) + \mu_{t+\tau}\left(k_{t+1} - (1-\delta)k_t\right) \right]$$

[8] You will also find in the matlab codes a version of the problem of a consumer facing a borrowing constraint in the directory `borrow`.

which leads to the following set of first order conditions

$$c_t^{-\sigma} = \lambda_t \tag{9.7}$$

$$\lambda_t - \mu_t = \beta E_t \left[\lambda_{t+1} \left(\alpha \exp(a_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta \right) - \mu_{t+1}(1-\delta) \right] \tag{9.8}$$

$$k_{t+1} = \exp(a_t) k_t^{\alpha} - c_t + (1-\delta) k_t \tag{9.9}$$

$$\mu_t \left(k_{t+1} - (1-\delta) k_t \right) \tag{9.10}$$

The main difference with the previous example is that now the central planner faces a constraint that may be binding in each and every period. Therefore, this complicates a little bit the algorithm, and we have to find a rule for both the expectation function

$$E_t \left[\Upsilon_{t+1} \right]$$

where

$$\Upsilon_{t+1} \equiv \beta \left(\lambda_{t+1} \left(\alpha \exp(a_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta \right) - \mu_{t+1}(1-\delta) \right)$$

and μ_t. We then proceed as we did in the PEA algorithm, that is we will compute an approximation to the expectation function and to the Lagrangean multiplier, checking at each computed node whether the constraint is binding or not. In order to keep things simple, we will discuss the case where the shock follows a Markov Chain,[9] Therefore, the rule of thumb will be such that

$$E_t \left[\Upsilon_{t+1} \right] \simeq \Phi(k_t, a_\ell; \theta) \equiv \exp \left(\sum_{j=0}^{n_k} \theta(a_\ell) T_j(\varphi(\log(k_i))) \right) \text{ for each } a_\ell,\ \ell = 1, \ldots, n_a$$

where n_a denotes the number of possible states for a.

The algorithm is very close to the one we used in the case of PEA and closely follows ideas suggested in Marcet and Lorenzoni [1999] and Christiano and Fisher [2000]. In the case of a Markov chain and for the Galerkin procedure, it can be sketched as follows.

[9] You are left to study the matlab code located in `/irrev/peagalg` for the continuous AR(1) case.

1. Choose an order of approximation n_k and a number of possible states n_a for the shock, compute the $m_k > n_k$ roots of the Chebychev polynomial of order m_k as

$$z_i = \cos\left(\frac{(2i-1)\pi}{2(n+1)}\right) \text{ for } i = 1, \ldots, m_k$$

Compute a parameters of the Markov chain (Transition matrix $\Pi = [\pi_{\ell\tau}]$ and formulate an initial guess for θ.

2. Compute the matrix $\mathcal{T}(z)$ as

$$\mathcal{T}(z) = \begin{pmatrix} T_0(z_1) & \ldots & T_0(z_m) \\ \vdots & \ddots & \vdots \\ T_n(z_1) & \ldots & T_n(z_m) \end{pmatrix}$$

3. Compute k_i as

$$k_i = \exp\left(\log(\underline{k}) + (z_i + 1)\frac{\log(\overline{k}) - \log(\underline{k})}{2}\right) \text{ for } i = 1, \ldots, n_k + 1$$

to map [-1;1] into $[\underline{k}; \overline{k}]$.

4. Compute the approximating expectation function at each node k_i, $i = 1, \ldots, m_k$, and for each possible state a_ℓ, $\ell = 1, \ldots, n_a$:

$$\Phi(k_i, a_\ell; \theta) = \exp\left(\sum_{j=0}^{n_k} \theta(a_\ell) T_j(\varphi(\log(k_i)))\right)$$

5. Deduce the level of consumption, which, from the Euler equation, is given by

$$c_t(k_i, a_\ell, \theta) = \Phi(k_i, a_\ell, \theta)^{-1/\sigma}$$

the level of investment

$$i_t(k_i, a_\ell; \theta) = \exp(a_\ell) k_i^\alpha - c_t(k_i, a_\ell, \theta)$$

the level of next period capital stock

$$k_{t+1}(k_i, a_\ell; \theta) = i_t(k_i, a_\ell) + (1 - \delta) k_i$$

6. Check whether the constraint is binding for each k_i, $i = 1, \ldots, n_k$ and each a_ℓ, $\ell = 1, \ldots, n_a$

   - If the constraint is not binding then keep computed values for $c_t(k_i, a_\ell, \theta)$ and $k_{t+1}(k_i, a_\ell; \theta)$ and set

   $$\mu_t(k_i, a_\ell; \theta) = 0$$

   - If it is binding then set

   $$\begin{aligned} k_{t+1}(k_i, a_\ell; \theta) &= (1-\delta)k_i \\ c_t(k_i, a_\ell, \theta) &= \exp(a_\ell)k_i^\alpha \\ \lambda_t(k_i, a_\ell; \theta) &= c_t(k_i, a_\ell; \theta)^{-\sigma} \\ \mu_t(k_i, a_\ell; \theta) &= \lambda_t(k_i, a_\ell; \theta) - \Phi(k_i, a_\ell; \theta) \end{aligned}$$

7. Then, for each state (k_i, a_ℓ), $i = 1, \ldots, n_k$ and $\ell = 1, \ldots, n_a$, compute the possible levels of future consumption without taking the constraint into account

   $$c_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) = \Phi(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta)^{-1/\sigma}$$

   for $\tau = 1, \ldots, n_a$, the level of next period investment

   $$i_t(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) = \exp(a_\tau)k_{t+1}(k_i, a_\ell; \theta)^\alpha - c_t(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta)$$

8. Check whether the constraint is binding at each position

   - If the constraint is not binding then keep computed values for $c_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta)$ and set

   $$\mu_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) = 0$$

   - If it is binding then set

   $$\begin{aligned} c_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) &= \exp(a_\tau)k_{t+1}(k_i, a_\ell; \theta)^\alpha \\ \lambda_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) &= c_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta)^{-\sigma} \\ \mu_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) &= \lambda_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) - \Phi(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta) \end{aligned}$$

9. Evaluate the residuals ($\Phi(k_i, a_\ell; \theta)$ minus the expectation function)

$$R(k_i, a_\ell; \theta) = \Phi(k_i, a_\ell; \theta) - \beta \sum_{\tau=1}^{n_a} \pi_{\ell\tau} \Psi(k_i, a_\ell, a_\tau; \theta)$$

where

$$\begin{aligned} \Psi(k_i, a_\ell, a_\tau; \theta) \equiv\ & c_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta)^{-\sigma} \times \\ & \left[\alpha \exp(a_\tau) k_{t+1}(k_i, a_\ell; \theta)^{\alpha-1} + 1 - \delta\right] \\ & -\mu_{t+1}(k_{t+1}(k_i, a_\ell; \theta), a_\tau; \theta)(1-\delta) \end{aligned}$$

for all $i = 1, \ldots, m_k$ and $\ell = 1, \ldots, n_a$.

10. Compute the inner product involved by the Galerkin procedure

$$\mathcal{T}(\varphi(k)) R(k, a; \theta)$$

11. if all inner products are close enough to zero then stop, else update θ and go back to 3.

Note that, as in the standard PEA algorithm, we treat μ_t as an technical variable which is just used to compute the residuals. We therefore do not need to compute explicitly its interpolating function.

MATLAB CODE: PEA GALERKIN (IRREVERSIBLE INVESTMENT)

```
clear all
global kmin ksup XX kt;           % parameters that will be common to
global nstate nbk ncoef XX XT PI; % several subfunctions

nbk=10;             % Degree of polynomials
nodes=30;           % # of Nodes
nstate=5;           % # of possible states for the shock
ncoef=nbk+1;        % # of coefficients

delta   = 0.1;      % depreciation rate
beta    = 0.95;     % discount factor
alpha   = 0.3;      % capital elasticity
sigma   = 1.5;      % CRRA Utility
ysk     =(1-beta*(1-delta))/(alpha*beta);
ksy     = 1/ysk;
```

```
ys      = ksy^(alpha/(1-alpha));
ks      = ys^(1/alpha);
is      = delta*ks;
cs      = ys-is;
%
% Markov Chain: (Tauchen-Hussey 1991) technology shock
%
rho     = 0.8;      % persistence
se      = 0.075;    % volatility
ma      = 0;
[agrid,wmat]=hernodes(nstate);
agrid       = agrid*sqrt(2)*se;
PI          = transprob(agrid,wmat,0,rho,se);
at          = agrid+ma;
%
% grid for the capital stock
%
kmin    = log(1);
ksup    = log(7);
rk      = rcheb(nodes);                 % roots
kt      = itransfo(rk,kmin,ksup);       % grid
XX      = cheb(rk,[0:nbk]);             % Polynomials
%
%  Initial Conditions
%
a0=[
-0.32518704    -0.22798091    -0.15021991    -0.12387075    -0.23608269
-0.61535041    -0.64879442    -0.70031744    -0.84056683    -1.23998020
-0.02892469    -0.03369096    -0.05570433    -0.15063263    -0.45685868
-0.00565378    -0.00941495    -0.02618621    -0.09526710    -0.32031660
-0.00296230    -0.00547789    -0.01692339    -0.06102182    -0.21319594
-0.00145337    -0.00274142    -0.00988611    -0.03461623    -0.13635186
-0.00065225    -0.00115070    -0.00536657    -0.01776343    -0.09027963
-0.00025245    -0.00038302    -0.00291504    -0.00838555    -0.06431768
-0.00004845    -0.00001360    -0.00162292    -0.00365588    -0.04689460
 0.00001150     0.00013398    -0.00088759    -0.00124601    -0.03294968
-0.00003248     0.00013595    -0.00055706    -0.00013116    -0.02163670
];
%
% Solves the problem effectively
%
param   = [alpha beta delta sigma]';
th      = fcsolve('residuals',a0,[],param,at);
th      = reshape(th,ncoef,nstate);
%
% Evaluate the approximation
%
lt      = length(th);
nb      = 100;
```

```
kt      = [kmin:(ksup-kmin)/(nb-1):ksup]';
rk      = transfo(kt,kmin,ksup);
XX      = cheb(rk(:),[0:nbk]);
kt      = exp(kt);
kp      = zeros(nb,nstate);
it      = zeros(nb,nstate);
ct      = zeros(nb,nstate);
mu      = zeros(nb,nstate);
for i=1:nstate;
   Upsilon  = exp(XX*th(:,i));
   y        = exp(at(i))*kt.^alpha;
   it(:,i)  = max(y-Upsilon.^(-1/sigma),0);
   ct(:,i)  = y-it(:,i);
   kp(:,i)  = it(:,i)+(1-delta)*kt;
   mu(:,i)  = ct(:,i).^(-sigma)-Ephi;
end;
```

MATLAB CODE: RESIDUALS FUNCTION (IRREVERSIBLE INVESTMENT)

```
function res=residuals(theta,param,at);
global nbk kmin ksup XX kt PI nstate;

alpha   = param(1);
beta    = param(2);
delta   = param(3);
sigma   = param(4);
lt      = length(theta);
theta   = reshape(theta,lt/nstate,nstate);
RHS=[];LHS=[];                                  % lhs and rhs of Euler equation
for i=1:nstate;
   Upsilon  = exp(XX*theta(:,i));               % evaluate expectation
   y        = exp(at(i))*exp(kt).^alpha;        % Output
   iv       = max(y-Ephi.^(-1/sigma),0);        % takes care of the constraint
   k1       = (1-delta)*exp(kt)+iv;             % next period capital stock
   rk1      = transfo(log(k1),kmin,ksup);       % maps it into [-1;1]
   xk1      = cheb(rk1,[0:nbk]);                % Computes the polynomials
   aux      = 0;
   for j=1:nstate;
      Upsilon1 = exp(xk1*theta(:,j));           % used for the consumption
      y1       = exp(at(j))*k1.^alpha;          % next period output
      mu1      = max(y1.^(-sigma)-Ephi1,0); % mu>0 <=> lambda=(c=y)^(-sigma)
      tmp      = beta*((alpha*y1./k1+1-delta).*Upsilon1-mu1*(1-delta));
      aux      = aux+PI(i,j)*tmp;
   end;
   RHS  = [RHS (aux)];
   LHS  = [LHS (Ephi)];
end;
res = XX'*(LHS-RHS);                             % Galerkin's projection
res = res(:);
```

Once again an important issue in this problem is that of initial conditions. The initial conditions reported in the matlab codes are actually extremely close to the solution. They were obtained using a *homotopy* type of approach. In praxis, we started from the deterministic optimal growth model without any irreversibility, and get a decision rule using an approximation of order 10. We then assign the decision rule we obtained in this case to a stochastic optimal growth model, with 5 possible states but low volatility (0.01), and obtained a new solution. Then we introduced the irreversibility constrained, and starting from the set of initial conditions we had, we get an approximate solution for the optimal growth model with irreversibility but with a low volatility. Starting from this new solution, we solved the same problem with higher volatility (0.02) and applied again the process until we reached the desired volatility (`se=0.075`). At a first sight, this may seem quite a long and useless process, but it saves a lot of time as it is much easier to go from a problem for which we have a solution to the problem we actually want to solve than trying to guess initial conditions.

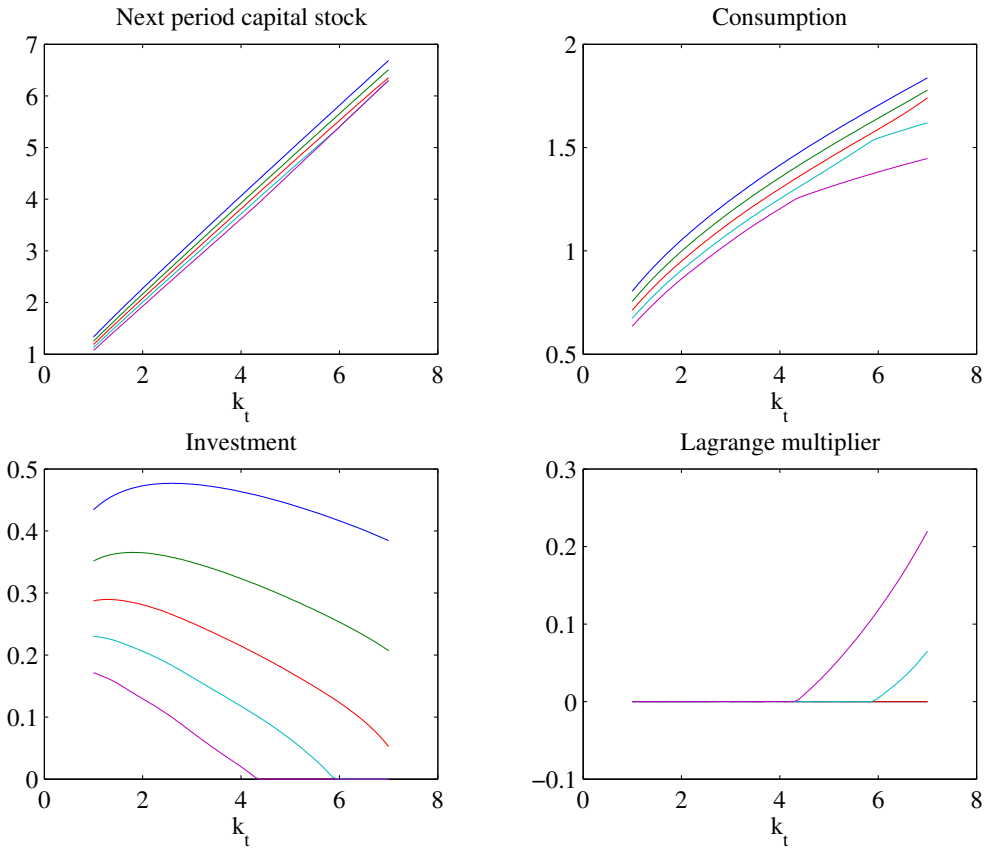
Table 9.5: Decision rules parameters (Irreversible investment, PEA–Galerkin)

	a_1	a_2	a_3	a_4	a_5
$T_0(\log(k_t))$	-0.271633	-0.198195	-0.131885	-0.076585	-0.044720
$T_1(\log(k_t))$	-0.617972	-0.639585	-0.662475	-0.701972	-0.786584
$T_2(\log(k_t))$	-0.020999	-0.021617	-0.024964	-0.042518	-0.094447
$T_3(\log(k_t))$	-0.000673	-0.001298	-0.004093	-0.017748	-0.053895
$T_4(\log(k_t))$	-0.000403	-0.000913	-0.002900	-0.012222	-0.032597
$T_5(\log(k_t))$	-0.000251	-0.000670	-0.002002	-0.007493	-0.015666
$T_6(\log(k_t))$	-0.000073	-0.000312	-0.001192	-0.004215	-0.005548
$T_7(\log(k_t))$	-0.000031	-0.000103	-0.000519	-0.001893	-0.000922
$T_8(\log(k_t))$	-0.000042	-0.000104	-0.000363	-0.000923	0.000202
$T_9(\log(k_t))$	-0.000028	-0.000093	-0.000359	-0.000890	-0.000995
$T_10(\log(k_t))$	-0.000008	-0.000032	-0.000182	-0.000557	-0.001279

Table 9.5 and Figure 9.3 report the approximate decision rules for an economy where $\alpha = 0.3$, $\beta = 0.95$, $\delta = 0.1$, $\sigma = 1.5$, the persistence of the shock

is $\rho = 0.8$ and the volatility $\sigma_\varepsilon = 0.075$. As can be seen, only low values for the shock make the constraint binds. This is illustrated by the fact that the Lagrange multiplier becomes positive, and investment becomes negative. Therefore, the next period capital stock is given by what is left by the depreciation and output is fully consumed.

Figure 9.3: Decision rules (Irreversible investment, PEA–Galerkin)



Note that up to now, we have not address the important issue of the evaluation of the rule we obtained using any of the method we have presented so far. This is the object of the next lecture notes.

Bibliography

Christiano, L.J. and J.D.M. Fisher, Algorithms for solving dynamic models with occasionally binding constraints, *Journal of Economic Dynamics and Control*, 2000, *24* (8), 1179–1232.

Judd, K., Projection Methods for Solving Aggregate Growth Models, *Journal of Economic Theory*, 1992, *58*, 410–452.

Marcet, A. and G. Lorenzoni, The Parameterized Expectations Approach: Some Practical Issues, in M. Marimon and A. Scott, editors, *Computational Methods for the Study of Dynamic Economies*, Oxford: Oxford University Press, 1999.

McGrattan, E., Solving the Stochastic Growth Model with a Finite Element Method, *Journal of Economic Dynamics and Control*, 1996, *20*, 19–42.

——, Application of Weighted Residual Methods to Dynamic Economic Models, in M. Marimon and A. Scott, editors, *Computational Methods for the Study of Dynamic Economies*, Oxford: Oxford University Press, 1999.

Contents

List of Figures

List of Tables