Chapter 3: Solution methods for macro models

Valerio Pieroni

Quantitative Macroeconomics

Introduction. This chapter provides an overview of commonly used solution methods in macroeconomics. I will consider as an example the neoclassical growth model with TFP shocks. However, the algorithms and techniques presented here are often used to solve more complex models with frictions, heterogeneous agents, and overlapping generations. In the next chapters we will cover some of these extensions.

I. Numerical methods

1 Numerical differentiation and integration

Numerical differentiation. Let $f : X \subseteq \mathbb{R} \to \mathbb{R}$ be a differentiable and integrable function. The key idea of numerical differentiation is to approximate a derivative by finite differences. We can use one-sided forward or backward differences

$$\frac{f(x+h)-f(x)}{h}, \quad \frac{f(x)-f(x-h)}{h},$$

or two-sided central difference,

$$\frac{f(x+h)-f(x-h)}{2h}.$$

A two-sided differential is more accurate. Second-order derivatives can be expressed as $\partial^2 f/\partial x^2 = (\partial/\partial x)(\partial f/\partial x)$. Thus, in finite differences

$$\frac{1}{2h}\left(\frac{f(x+h)-f(x)}{h} - \frac{f(x)-f(x-h)}{h}\right) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2}.$$

Analogously, numerical mixed partial derivatives are given by

$$\frac{1}{2k}\left(\frac{f(x+k,y+h)-f(x+k,y-h)}{2h} - \frac{f(x-k,y+h)-f(x-k,y-h)}{2h}\right).$$

Numerical integration. For numerical integration or quadrature the integrand is evaluated at a finite set of points and the integral is approximated by a weighted sum of these values.

$$\int_X f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i).$$

The choice of the weights w_i and evaluation points x_i characterizes different methods. Commonly used methods include the trapezoidal rule, the Simpson's rule, Gaussian quadrature, and Monte Carlo integration.

Consider the interval $X = [a, b]$. The trapezoidal rule constructs a uniform grid of nodes $x_i = a + (i-1)h$, for $i = 1, 2, ..., n$ where $h = (b-a)/(n-1)$ and set $w_1 = w_n = h/2$ and $w_i = h$ otherwise. In this way the function evaluations $f(x_i)$ are connected by linear segments and in each subinterval the trapezoid's area is computed as $h(f(x_i) + f(x_{i+1}))/2$. The Simpson's rule uses a quadratic function to connect evaluation points leading to weights $w_1 = w_n = h/3$ and $w_i = 4(h/3)$ if i is even and $w_i = 2(h/3)$ if i is odd.

Gaussian quadratrue rules construct numerical integrals for a given weight function $w(x)$,

$$\int_X f(x)w(x)dx \approx \sum_{i=1}^{n} w_i f(x_i).$$

The quadrature nodes x_i and weights w_i solve the moment-matching conditions

$$\int_X x^k w(x)dx = \sum_{i=1}^{n} x_i^k w_i, \;\; \forall k = 0, ..., 2n-1.$$

Typically $w(x)$ is either equal to 1 or is a density function.

Monte Carlo integration draws a random sample x_i from f_w and compute its mean. This is motivated by the strong law of large numbers. $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} f(x_i) = \int_X f(x)f_w(x)dx$, almost surely. Therefore, we simply have $w_i = 1/n, \forall i$. The random sample can be generated from the cumulative distribution function F_w associated to f_w. First, we draw a pseudorandom sample u_i from a uniform distribution on $[0, 1]$. Then, we find $x_i = F_w^{-1}(u_i)$.

2 Nonlinear systems

Let $f : \mathbb{R}^k \to \mathbb{R}^k$, we want to solve the nonlinear system $f(x) = 0$. Consider the case in which $k = 1$, f is continuous over the interval $[a, b]$, and $f(a), f(b)$ alternate the sign. Then, by the intermediate value theorem f must have at least one zero in $x \in [a, b]$. The bisection method start from an interval $[\underline{x}, \bar{x}]$ with $f(\underline{x}) < 0, f(\bar{x}) > 0$ and then evaluate f at the midpoint $x_n = (\underline{x} + \bar{x})/2$, if $f(x_n) = 0$ we are done, if $f(x_n) < 0$ set $\underline{x} = x_n$ and compute again the midpoint of this interval x_{n+1}, if $f(x_n) > 0$ set $\bar{x} = x_n$ and compute the midpoint x_{n+1}. We can stop when $|x_n - x_{n+1}| < \epsilon$ for $\epsilon \approx 0$ or $f(x_{n+1}) \leq \delta$ for $\delta \approx 0$. This procedure construct successive smaller intervals containing a zero of f.

The Newton-Raphson method finds the roots of a given function by linearly approximating the function in a given point x_n, and finding the root of this line x_{n+1}. Convergence to the solution is guaranteed as long as the function is globally convex or globally concave. Suppose that $k = 1$, $f(x)$ is at least once differentiable. Solving $f(x_n) + (x_{n+1} - x_n)f'(x_n) = 0$ for x_{n+1} yields

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

From an initial guess x_n we can iterate until convergence, i.e. $|x_{n+1} - x_n| < \epsilon$, for ϵ small. In general, $f : \mathbb{R}^k \to \mathbb{R}^k$ the iteration scheme in matrix notation becomes

$$x_{n+1} = x_n - J_f(x_n)^{-1} f(x_n).$$

This requires to invert the Jacobian $J_f(x_n)$. When the Jacobian is computationally too demanding or ill-conditioned, there are alternative methods, called Quasi-Newton methods, that simplify its computation and provides approximations to it.

Given a transformation $g : \mathbb{R}^k \to \mathbb{R}^k$ we can rewrite root finding problems $f(x) = x - g(x) = 0$ as fixed-point problems $x = g(x)$. This suggest the iteration $x_{n+1} = \theta x_n + (1 - \theta)g(x_n)$ until $|x_{n+1} - x_n| < \epsilon$. The parameter $\theta \in [0, 1]$ determines the size of each steps affecting the speed and convergence of the algorithm.

The bisection method always converges but it is often slow. The Quasi-Newton algorithms tend to be faster but the convergence is not guaranteed. If a function is flat around the zero, e.g. x^6, loose stopping rules can lead to convergence far from the true zero. Flat regions might also cause ill-conditioned matrices. Moreover, with irregular functions the solution can be very sensitive to starting values due to the presence of multiple roots, and there are cases in which these methods produce cycles. Other pitfalls are related to rounding errors and scale problems. Fixed point schemes also do not guarantee convergence with the exception of existence theory for the fixed points of contraction mappings.

3 Numerical optimization

Consider the optimization problem $\max_{x \in X \subseteq \mathbb{R}^k} f(x)$ where $f : \mathbb{R}^k \to \mathbb{R}$ is the objective function and X is the feasible set. Note that this is equivalent to minimize $-f(x)$. If f is differentiable the optimization problem reduces to solving a nonlinear system of fisrt-order conditions. Hence, we can use nonlinear solvers for these problems with all the caveats mentioned above. However, if f is not differentiable we need to rely on search methods. Search methods evaluate the objective function at several points on a grid to find a minimum. To check q values in each dimension we need to check q^k points. For example, if $q = 100$ and $k = 10$, there would be 100^{10} points to check. Given this "curse of dimensionality" search methods are feasible if k is small. Quasi-Newton and search algorithms may converge to a local minimum that is not optimal. Then, a common practice to find a global minimum is to try many starting values. Alternatively, global optimizers provide a systematic procedure for selecting restart points or to escape local minima.

A commonly used search algorithm is the Nelder-Mead simplex method. This method construct a sequence of evaluation points x_i in $\mathbb{R}^k$ as vertices of a simplex to minimize $f(x)$. If $k = 1$ the simplex is a segment, if $k = 2$ the simplex is a triangle. We start specifying the starting points by choosing one point and creating a regular simplex from it or by specifying all the points together. Then, we order the evaluation points $x_1, x_2, ..., x_{k+1}$ so that $f(x_1) \leq f(x_2) \leq ... \leq f(x_{k+1})$. Let $x = k^{-1} \sum_{i=1}^{k} x_i$ be the centroid of the best k points, i.e. all points but x_{k+1}. Then, compute a reflection point $x_r = x + \rho(x - x_{k+1})$ where $\rho > 0$. If $f(x_1) \leq f(x_r) \leq f(x_k)$ the reflected point replaces x_{k+1}. If $f(x_r) < f(x_1)$ we calculate an expansion point $x_e = x + \chi(x_r - x)$ with $\chi > 1$. If $f(x_e) < f(x_r)$ then x_e replaces x_{k+1} otherwise we use x_r. If $f(x_k) \leq f(x_r)$ we perform a contraction with the better of x_r, x_{k+1}. If $f(x_k) \leq f(x_r) \leq f(x_{k+1})$ we compute an outside contraction $x_c = x + \gamma(x_r - x)$ with $0 < \gamma < 1$, and if $f(x_c) \leq f(x_r)$ we use x_c otherwise we go to the shrink step described below. If $f(x_k) \leq f(x_{k+1}) \leq f(x_r)$ we compute an inside contraction $x_c = x + \gamma(x_{k+1} - x)$, and if $f(x_c) \leq f(x_{k+1})$ we use x_c otherwise we go to the shrink step. If a shrink step is performed, a set of k new points together with x_1 form the updated simplex. The new points are given by $x_i = x_1 + \sigma(x_i - x_1)$ with $0 < \sigma < 1$. The iteration continues until $\max_i |x_i - x_1| < \epsilon$ or $\max_i f(x_i) < \delta$. In general, $\rho = 1, \chi = 2, \gamma = \sigma = .5$. The algorithm requires 1 or 2 function evaluations in each iteration, and $k + 2$ if we shrink.

A global search algorithm is simulated annealing. This method randomly selects evaluation points by perturbating the ith component of a guess $x_n \in \mathbb{R}^k$, that is $x_{in+1} = x_{in} + \alpha_i r_i$ where α_i specify the steplenght and r_i is a random draw from a uniform distribution in $(-1, 1)$. We accept the new guess if it decreases f or if it increases f but

$\exp((f(x_n) - f(x_{n+1}))/T_n) > u$ where u is a random draw from a uniform $(0, 1)$ and T_n is a scaling parameter called temperature. By accpeting some points that increase the objective function the algorithm to escape from local maxima. By varying T_n the probability that negative move is accepted reduces and the algorithm focuses on the most promising region.

An alternative global algorithm is TikTak. Developed by Arnoud, Guvenen, and Kleineberg (2019), TikTak belongs to the class of multistart algorithms. These algorithms typically have two stages: (i) a global stage, which selects the starting points for new local searches, and (ii) a local stage, which implements local searches, by choosing a local search algorithm and local stopping criteria. TikTak works as follows. First, generate k quasi-random points x_i using the Sobol' sequence to uniformly cover the search space. Select the best k^* points with lowest $f(x_i)$ value and sort these points $x_1, x_2, ..., x_{k^*}$ so that $f(x_1) \leq f(x_2) \leq ... \leq f(x_{k^*})$. Then, compute a minimum z_1 using a local search from the starting point $y_1 = x_1$. For the next local search use the starting point $y_{j+1} = (1 - \theta_j)x_{j+1} + \theta_j z_j^*$ where $\theta_j \in [0, 1], \forall j = 1, 2, ..., k^* - 1$ and z_j^* is the best minimum so far. Starting from θ_j very close to zero θ_j is gradually increased. We stop when $|z_{j-1} - z_j| < \epsilon$.

4 The functional equation problem

Function approximation. Quantitative macro models do not have analytical solutions available and we need to rely on numerical solutions. In this chapter I present the solution methods for a simple Dynamics Stochastic General Equilibrium (DSGE) model. The neoclassical growth model with TFP shocks. The solution techniques can be divided in local methods and global methods. Local methods require few coding and running time but we obtain local solutions, accurate only around the approximation point. Global methods deliver a solution of the fully-nonlinear model and can attack complex problems with occasionally binding constraints, but they require more computational effort.

We can cast macroeconomic models as functional equations. These problems are difficult to solve because the unknown is a function. Let J^1, J^2 be two functional spaces, $X \subseteq \mathbb{R}^n$ the state space, $H : J^1 \to J^2$. A functional equation problem is to find a unknown function $d : X \to \mathbb{R}^m$ where $\{d\} \subseteq J^1$ such that

$$H(d) = 0.$$

Perturbation and projections algorithms replace the unknown function d for an approximation $d_j(x, \theta)$ which is a function of the state variables x, a vector of numerical coefficients θ and a degree of approximation j.

Neoclassical model with aggregate risk. As an example I will consider the neoclassical growth model with aggregate TFP shocks. To save on notation we omit the argument of $c_t(z^t), n_t(z^t), k_{t+1}(z^t)$ and ignore long run growth. Note that capital is not a state-contingent asset. Since the welfare theorems hold we consider the planner's problem

$$\max_{\{c_t,n_t,k_{t+1}\}} \mathrm{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t, n_t),$$

$$\text{s.t.} \quad c_t + k_{t+1} = z_t k_t^{\alpha} n_t^{1-\alpha} + (1-\delta)k_t, \ \forall t.$$

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

Preferences are given by a log-log utility, $\ln c_t + \chi \ln(1-n_t)$. Other widely used functional forms are $(c(1-n)^{\chi})^{1-\gamma}/(1-\gamma)$, $\ln c - \chi(n^{1+\nu})/(1+\nu)$, $c_t^{1-\gamma}/(1-\gamma) - \chi(n^{1+\nu})/(1+\nu)$. where χ, γ, ν^{-1} are respectively the labor disutility coefficient, the relative risk aversion coefficient, and the Frish elasticity of labor supply. Sometimes, we will also consider for simplicity the inelastic case, i.e. $u(c_t) = c_t^{1-\gamma}/(1-\gamma), n_t = 1$. We could also write the TFP process directly in log terms z_t and then take $\exp(z_t)$ in the market clearing condition.

Solving the model. In the inelastic case given k_0, z_0 the equilibrium solution of the model can be characterized as decision rules $g : \mathbb{R}^2_+ \to \mathbb{R}^2_+$:

$$\begin{pmatrix} c_t \\ k_{t+1} \end{pmatrix} = g(k_t, z_t),$$

mapping k_t, z_t into c_t, k_{t+1}. Denote this solution as $g^c(k_t, z_t), g^k(k_t, z_t)$. To solve the model we need to solve a functional equation problem $H(d) = 0$ that can be characterized by a Bellman equation or by a system of first order conditions and other equilibrium conditions.

In particular, the equilibrium conditions $\forall t, \forall z^t$ are given by

$$H(g) = \begin{cases} u'(g^c(k_t, z_t)) - \beta \mathrm{E}_t[u'(g^c(g^k(k_t, z_t), z_{t+1})(z_{t+1} f'(g^k(k_t, z_t)) + 1 - \delta)] \\ g^c(k_t, z_t) + g^k(k_t, z_t) - z_t f(k_t) - (1-\delta)k_t \end{cases} = 0,$$

together with a transversality condition. The Bellman equation is

$$H(v) = v(k_t, z_t) - \max_{k_{t+1}} [u(z_t f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta \mathrm{E}_t v(k_{t+1}, z_{t+1})] = 0.$$

II. Local Methods

Local methods are the usual way to solve macro models with representative agent. A widely used approach consists of linearizing the equilibrium conditions and solving the resulting system of stochastic linear difference equations. There are many toolkits, e.g. Blanchard and Kahn (1980), King and Watson (1998), Uhlig (1999), Klein (2000), Sims (2002), Dynare, that do so very efficiently.

1 Perturbation methods

Equilibirum conditions. The equilibrium of a model can be characterized as a functional equation problem together with an exogenous stochastic process for the exogenous states:

$$\mathrm{E}_t H(y_t, y_{t+1}, x_t, x_{t+1}) = 0,$$

$$x_{2,t+1} = Ax_{2,t} + B\varepsilon_{t+1}$$

where y, x are the controls and states vectors of dimension n_y and n_x, $n = n_y + n_x$. Moreover, partitioning the state variables in a n_1-dimensional vector of endogenous state variables and a n_2-dimensional vector of exogenous state variables we have $x_t = [x'_{1,t}, x'_{2,t}]'$ and $n_x = n_1 + n_2$. The function $H : \mathbb{R}^{n_y} \times \mathbb{R}^{n_y} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^n$ stacks all the equilibrium conditions. The vector $\varepsilon_t \overset{iid}{\sim} N(0, \sigma I_{n_2})$, where $\sigma > 0$. A, B are $n_2 \times n_2$ matrices, A has all eigenvalues with modulus less than 1. The solution of the model d is given by a set of decision rules

$$y_t = g(x_t),$$

$$x_{t+1} = h(x) + \eta\varepsilon_{t+1},$$

where $\eta = [0, B]'$ is a $n_x \times n_2$ matrix, $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}, h : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$. Therefore,

$$F(x;\sigma) := \mathrm{E}_t H(g(x;\sigma), g(h(x;\sigma) + \eta\varepsilon';\sigma), x, h(x;\sigma) + \eta\varepsilon') = 0.$$

Assuming that $F : \mathbb{R}^{n_x} \to \mathbb{R}^n$ is differentiable the perturbation solution takes a Taylor series expansion of the functions g, h around the steady state.

Steady states. Stochastic models have two types of steady states. The deterministic steady state is the steady state in which we set the variance of the shocks equal to zero, i.e. $\sigma = 0$. This is the fixed point $(\hat{x}, \hat{y})$ of $H(\hat{x}, \hat{x}, \hat{y}, \hat{y}) = 0$. The stochastic steady state is the steady state in which the realization of the shocks are always zero $\varepsilon_t = 0, \forall t$. This is

the fixed point $(\bar{x}, \bar{y})$ of $\mathrm{E}_t H(\bar{x}, \bar{x}, \bar{y}, \bar{y}) = 0$. It's common to use $(\hat{x}; 0)$ as approximation point since we can easily compute analytically $\hat{y} = g(\hat{x}; 0), \hat{x} = h(\hat{x}; 0)$. In case we have growth in the model we rescale the growing variables z_t by their trend μ_t to make them stationary $\tilde{z}_t = z_t/\mu_t$.

Taylor expansions. The first order perturbation yields a linear approximation

$$g(x; \gamma) = g(\hat{x}; 0) + D_x g(\hat{x}; 0)(x - \hat{x})' + D_\gamma g(\hat{x}; 0)\gamma + o(1),$$

$$h(x; \gamma) = h(\hat{x}; 0) + D_x h(\hat{x}; 0)(x - \hat{x})' + D_\gamma h(\hat{x}; 0)\gamma + o(1).$$

$D_x g(.), D_x h(.)$ are the $n_x \times 1$ vectors of partial derivatives of g, h with respect to x, while D_σ stands for the partial derivative with respect to σ. The numerical coefficients or perturbation coefficients $D_x g(\hat{x}; 0), D_x h(\hat{x}; 0), D_\sigma g(\hat{x}; 0), D_\sigma h(\hat{x}; 0)$ can be found by the implicit function theorem. To ease notation I omitt the argument of the functions. Since $F = 0$ then $D_x g(\hat{x}; 0), D_x h(\hat{x}; 0)$ solve the $n \times n_x$ system

$$D_x F(\hat{x}; 0) = D_{y'} H D_x g (D_x h)' + D_y H D_x g + D_{x'} H D_x h + D_x H = 0.$$

$D_\sigma g(\hat{x}; 0), D_\sigma h(\hat{x}; 0)$ solve the n equations

$$D_\sigma F(\hat{x}; 0) = D_{y'} H D_x g D_\sigma h + D_{y'} H D_\sigma g + D_y H D_\sigma g + D_{x'} f D_\sigma h = 0.$$

Because the derivatives of H evaluated at $(\hat{y}, \hat{y}, \hat{x}, \hat{x})$ are known we can solve the systems. Notice that in the second case the equations are linear and homogeneous in $D_\sigma g, D_\sigma h$ therefore if a unique solution exists it is given by $D_\sigma g = 0, D_\sigma h = 0$. Hence, we have the solution

$$g(x; \sigma) - \hat{y} = D_x g(\hat{x}; 0)(x - \hat{x})'$$

$$h(x; \sigma) - \hat{x} = D_x h(\hat{x}; 0)(x - \hat{x})'.$$

One drawback of a first order perturbation is that the approximate solution displays certainty equivalence, that is the variance of the disturbances does not appears in the solution of the model only because of the solution method we are using, implying that the approximate solution coincide with the one we obtained under perfect foresight i.e. $\sigma = 0$. Therefore, the agents in the model do not take any precautionary behavior and the solution cannot generate any risk premia for assets or be useful for analyzing changes in volatility. It is easy to check that this is not the case for higher order perturbations. To this end, we can iterate the previous procedure taking higher order Taylor expansions,

plugging in the already found terms and solving for the remaining ones with the implicit function theorem.

There are many packages implementing these computations. Dynare is a program written for Matlab. In the preamble we can specify endogenous and exogenous variables, parameters, and equilibrium equations. Dynare provides routines to solve the model using perturbation algorithms.

2 Linearization

Log-linearization is an efficient method to linearize equilibrium conditions by hand that does not require to compute derivatives. However, the linearization of the equilibrium conditions is equivalent to a first order perturbation. Moreover, even linearizations with analytical derivatives are easily implemented by existing softwares, e.g. one could use the symbolic math toolbox in Matlab.

With a log-linearization we can approximate the solution of the model in terms of the log-deviations of the variables with respect to their steady state value

$$\hat{x}_t := \ln\left(\frac{x_t}{x}\right) = \ln x_t - \ln x = \ln\left(1 + \frac{x_t - x}{x}\right) \simeq \frac{x_t - x}{x}$$

Thus, $\hat{x} = 0$. Log-linear solutions have a nice interpretations in terms of percentage deviations from the deterministic steady state. Moreover, computing the derivatives is not necessary to log-linearize equilibrium conditions. Notice that $x_t = \exp(\ln x_t)$, $\ln x_t = \hat{x}_t + \ln x$ then taking the exponential of both sides we have

$$x_t = x \exp(\hat{x}_t).$$

By a first-order Taylor expansion around the steady state $\hat{x}_t = 0$ we have $x \exp(\hat{x}_t) = x \exp(0) + x \exp(0)(\hat{x}_t - 0)$. Thus, $\exp(\hat{x}_t) = (1 + \hat{x}_t)$. Therefore, we have the following useful formulae $x_t = x(1 + \hat{x}_t), x_t y_t = xy(1 + \hat{x}_t + \hat{y}_t), x_t/y_t = (x/y)(1 + \hat{x}_t - \hat{y}_t), x_t^n = x^n(1 + n\hat{x}_t), x_t^n = x^n(1 + \hat{x}_t)^n, \exp(x_t) = \exp(x)(1 + x\hat{x}_t)$.

Suppose that we want to linearize the Euler equation

$$\mathrm{E}_t\left[\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\sigma}\left(\frac{1}{\Pi_{t+1}}\right)\right] = q_t,$$

where $\Pi_{t+1} = p_{t+1}/p_t$ is the gross inflation rate and $\pi_t = \Pi_t - 1$ is the inflation rate. In the steady state $q = \beta, \Pi = 1, \pi = 0$. Let's start with the direct Taylor expansion. In this case the direct computation of derivatives and Taylor expansion by hand can be costly.

We need

$$\frac{\partial q_t}{\partial c_{t+1}} = -\sigma\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\sigma-1}\frac{1}{c_t}\frac{1}{\Pi_{t+1}},$$
$$\frac{\partial q_t}{\partial c_t} = \sigma\beta\left(\frac{c_{t+1}}{c_t}\right)^{-\sigma-1}\left(\frac{c_{t+1}}{c_t^2}\right)\left(\frac{1}{\Pi_{t+1}}\right),$$
$$\frac{\partial q_t}{\partial \Pi_{t+1}} = \beta\left(\frac{c_{t+1}}{c_t}\right)^{-\sigma}\frac{1}{\Pi_{t+1}^2}.$$

Hence, the Taylor expansion

$$q\hat{q}_t = -\sigma\beta c(1/c)\mathrm{E}_t\hat{c}_{t+1} + \sigma\beta c(c/c^2)\hat{c}_t - \beta\mathrm{E}_t\pi_{t+1}.$$

Simplyfing and rearranging terms yields

$$\hat{c}_t = \mathrm{E}_t\hat{c}_{t+1} - \sigma^{-1}(\hat{i}_t - \mathrm{E}_t\pi_{t+1}).$$

Now use the exponential formulae.

$$\mathrm{E}_t\left[\beta\left(\frac{ce^{c_{t+1}}}{ce^{c_t}}\right)^{-\sigma}\frac{1}{e^{\hat{\Pi}_{t+1}}}\right] = qe^{\hat{q}_t}.$$

Rearraging, $\beta\mathrm{E}_t[\exp(-\sigma\hat{c}_{t+1}+\sigma\hat{c}_t-\pi_{t+1}-\hat{q}_t)] = q$. From the Taylor expansion $\exp(-\sigma\hat{c}_{t+1}+\sigma\hat{c}_t-\pi_{t+1}-\hat{q}_t) = 1-\sigma\hat{c}_{t+1}+\sigma\hat{c}_t-\pi_{t+1}-\hat{q}_t$ we have $\mathrm{E}_t[1-\sigma\hat{c}_{t+1}+\sigma\hat{c}_t-\pi_{t+1}-\hat{q}_t] = 1$ and rearranging we obtain

$$\hat{c}_t = \mathrm{E}_t\hat{c}_{t+1} - \sigma^{-1}(\hat{i}_t - \mathrm{E}_t\pi_{t+1}).$$

If we want to keep track of some constant we can use a third approach. Rewriting all variables as $x_t = \exp(\ln x_t)$ and taking a Taylor expansion around the steady state lead us to the Euler equation $1 = \mathrm{E}_t[\exp(i_t - \sigma\Delta c_{t+1} - \pi_{t+1} - \rho)]$. In the steady state $i - \rho - \pi - \sigma g_c = 0$,

$$\begin{aligned}\exp(i_t - \sigma\Delta c_{t+1} - \pi_{t+1} - \rho) &= 1 + (i_t - i) - \sigma(\Delta c_{t+1} - g_c) - (\pi_{t+1} - \pi)\\ &= 1 + i_t - \sigma\Delta c_{t+1} - \pi_{t+1} - \rho,\end{aligned}$$

and we obtain

$$c_t = \mathrm{E}_t c_{t+1} - \frac{1}{\sigma}(i_t - \mathrm{E}_t\pi_{t+1} - \rho).$$

3 Solving linear dynamic systems with aggregate risk

Step 1: equilibrium conditions. Consider the neoclassical growth model with TFP shocks and log-log utility. The equilibrium conditions $\forall t, \forall z^t$ are given by

$$c_t^{-1} = \beta \mathrm{E}_t c_{t+1}^{-1}(1 + \alpha e^{z_{t+1}} k_{t+1}^{\alpha-1} n_{t+1}^{1-\alpha} - \delta),$$

$$\chi c_t = (1 - n_t)(1 - \alpha) e^{z_t} k_t^{\alpha} n_t^{-\alpha},$$

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

$$\ln z_t = \rho \ln z_{t-1} + \sigma \varepsilon_t,$$

$$\lim_{t \to \infty} \beta^t u'(c_t) k_{t+1} = 0.$$

Step 2: steady state. Evaluating the equilibrium system at the deterministic steady state $\sigma = 0, z = 1$ yields a system of 3 equations in 3 unknown

$$1 = \beta[\alpha k^{\alpha-1} n^{1-\alpha} + 1 - \delta],$$

$$\chi c = (1 - \alpha) k^{\alpha} n^{-\alpha} (1 - n),$$

$$c + k = k^{\alpha} n^{1-\alpha} + (1 - \delta) k.$$

We can take n from the data and solve the system as follows $k/n = [\alpha(\beta^{-1} - 1 + \delta)^{-1}]^{1/(1-\alpha)}, \chi = (1 - \alpha)(k/n)^{\alpha} c^{-1}(1 - n), c = (k/n)^{\alpha} n - \delta(k/n) n$.

Step 3: undetermined coefficients. Let's log-linearize the model by hand.

$$\hat{c}_t = \mathrm{E}_t \hat{c}_{t+1} - \alpha_1 \hat{k}_{t+1} - \alpha_2 \mathrm{E}_t \hat{n}_{t+1} - \alpha_3 \mathrm{E}_t \hat{z}_{t+1},$$

$$\hat{n}_t = \alpha_4 \hat{c}_t - \alpha_4 \hat{z}_t + \alpha_5 \hat{k}_t,$$

$$\hat{k}_{t+1} = \alpha_6 \hat{z}_t + \alpha_7 \hat{k}_t + \alpha_8 \hat{n}_t - \alpha_9 \hat{c}_t,$$

$$\hat{z}_t = \rho \hat{z}_{t-1} + \sigma \varepsilon_t$$

Substituting for $\hat{n}_t$ we have a linear system of difference equations.

$$A \begin{bmatrix} \mathrm{E}_t \hat{c}_{t+1} \\ \hat{k}_{t+1} \\ \hat{z}_{t+1} \end{bmatrix} = B \begin{bmatrix} \hat{c}_t \\ \hat{k}_t \\ \hat{z}_t \end{bmatrix} + C \varepsilon_{t+1} \tag{1}$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 & 0 \\ b_{21} & b_{22} & b_{23} \\ 0 & 0 & \rho \end{bmatrix}, \quad C = \begin{bmatrix} 0 \\ 0 \\ \sigma \end{bmatrix}.$$

Now we solve the model numerically using the method of undetermined coefficients. This method is a simple guess and verify procedure. Assume linear policy functions

$$\hat{c}_t = \eta_1 \hat{k}_t + \eta_2 \hat{z}_t,$$

$$\hat{k}_{t+1} = \eta_3 \hat{k}_t + \eta_4 \hat{z}_t.$$

Substituting this guess in the model and using $\mathrm{E}_t \hat{z}_{t+1} = \rho \hat{z}_t$ we have

$$(1 - \alpha_2\alpha_4)[\eta_1(\eta_3 \hat{k}_t + \eta_4 \hat{z}_t) + \eta_2 \rho \hat{z}_t] = (\alpha_1 + \alpha_2\alpha_5)(\eta_3 \hat{k}_t + \eta_4 \hat{z}_t) + \rho(\alpha_3 - \alpha_2\alpha_4)\hat{z}_t + \eta_1 \hat{k}_t + \eta_2 \hat{z}_t,$$

$$(\eta_3 \hat{k}_t + \eta_4 \hat{z}_t) = (\alpha_6 - \alpha_8\alpha_4)\hat{z}_t + (\alpha_7 + \alpha_5\alpha_8)\hat{k}_t - (\alpha_9 - \alpha_8\alpha_4)(\eta_1 \hat{k}_t + \eta_2 \hat{z}_t).$$

Since these equations need to hold for any value of $\hat{k}_t, \hat{z}_t$ we have to equate each coefficient to zero. This yields to a system of 4 equations in 4 unknown $\eta_1, \eta_2, \eta_3, \eta_4$.

$$(1 - \alpha_2\alpha_4)\eta_1\eta_3 - (\alpha_1 + \alpha_2\alpha_5)\eta_3 - \eta_1 = 0,$$

$$(1 - \alpha_2\alpha_4)[\eta_1\eta_4 + \eta_2\phi] - (\alpha_1 + \alpha_2\alpha_5)\eta_4 - \phi(\alpha_3 - \alpha_2\alpha_4) - \eta_2 = 0,$$

$$\eta_3 - (\alpha_7 + \alpha_5\alpha_8) + (\alpha_9 - \alpha_8\alpha_4)\eta_1 = 0,$$

$$\eta_4 - (\alpha_6 - \alpha_8\alpha_4) + (\alpha_9 - \alpha_8\alpha_4)\eta_2 = 0.$$

Substituting the last two in the first two we have

$$a\eta_1^2 + b\eta_1 - c = 0,$$

$$d\eta_1 + e\eta_2 + f\eta_1\eta_2 + g = 0.$$

The first is a quadratic equation in η_1 with two solutions $\eta_1 < 1$ and $\eta_1 > 1$. We can rule out the latter since it produces an explosive behavior violating the transversality condition. Once we know η_1 it's easy to solve for η_2, η_3, η_4.

$$\begin{aligned} k_t &= \eta_1 \hat{k}_{t-1} + \eta_2 \hat{z}_{t-1} = \eta_1[\eta_1 \hat{k}_{t-2} + \eta_2 \hat{z}_{t-2}] + \eta_2 \hat{z}_{t-1} \\ &= ... = \sum_{j=1}^{\infty} \eta_1^{j-1} \eta_2 \hat{z}_{t-j} + \lim_{t\to\infty} \eta_1^t k_0 = +\infty. \end{aligned}$$

Step 3: Blanchard-Khan conditions. We can also solve the model (1) decoupling the dynamics. This is the Blanchard-Khan approach. Rewrite the system

$$A \begin{bmatrix} \mathrm{E}_t \hat{c}_{t+1} \\ \hat{k}_{t+1} \end{bmatrix} = B \begin{bmatrix} \hat{c}_t \\ \hat{k}_t \end{bmatrix} + C \hat{z}_t$$

where

$$A = \begin{bmatrix} a_{11} & a_{12} \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 0 \\ b_{21} & b_{22} \end{bmatrix}, \quad C = \begin{bmatrix} -\rho a_{13} \\ b_{23} \end{bmatrix}.$$

Inverting A we have

$$\begin{bmatrix} \mathrm{E}_t \hat{c}_{t+1} \\ \hat{k}_{t+1} \end{bmatrix} = A^{-1} B \begin{bmatrix} \hat{c}_t \\ \hat{k}_t \end{bmatrix} + A^{-1} C \hat{z}_t.$$

Let $A_1 = A^{-1}B, A_2 = A^{-1}C$. The canonical spectral decomposition $A_1 = Q\Lambda Q^{-1}$ allow us to define new variables so that we can write the model as a diagonal system of stochastic difference equations

$$\begin{bmatrix} \mathrm{E}_t \hat{x}_{t+1} \\ \hat{\kappa}_{t+1} \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \hat{x}_t \\ \hat{\kappa}_t \end{bmatrix} + Q^{-1} A_2 \hat{z}_t.$$

The model has a forward-looking variable c_t and a backward-looking variable k_t. Since A_1 has an unstable eigenvalue $\lambda_1 > 1$ and a stable eigenvalue $\lambda_2 < 1$, i.e. the Blanchard-Khan condition, we know that this new system admits a saddle path solution.

The solution is given by

$$\hat{x}_t = -\sum_{j=0}^{\infty} \lambda_1^{-j} (Q^{-1}A_2)_1 \mathrm{E}_t \hat{z}_{t+j} = -\sum_{j=0}^{\infty} \lambda_1^{-j} (Q^{-1}A_2)_1 \rho^j \hat{z}_t = -\frac{\lambda_1}{\lambda_1 - \rho} (Q^{-1}A_2)_1 \hat{z}_t,$$

$$\hat{\kappa}_t = \sum_{j=0}^{\infty} \lambda_2^{j} (Q^{-1}A_2)_2 \hat{z}_{t-j}.$$

We can recover capital and consumption by

$$\begin{bmatrix} \hat{c}_t \\ \hat{k}_t \end{bmatrix} = Q \begin{bmatrix} \hat{x}_t \\ \hat{\kappa}_t \end{bmatrix}.$$

Step 3: First-order perturbation. The functional equation $F(k_t, z_t; \sigma) = 0$ is given by

$$\mathrm{E}_t \begin{bmatrix} \dfrac{1}{c(k_t, z_t; \sigma)} - \beta \dfrac{1 - \delta + \alpha(\rho \ln z_t + \sigma \varepsilon_{t+1}) k(k_t, z_t; \sigma)^{\alpha - 1} n(k_t, z_t; \sigma)^{1-\alpha}}{c(k(k_t, z_t; \sigma), \rho \ln z_t + \sigma \varepsilon_{t+1}; \sigma)} \\ \dfrac{\chi c(k_t, z_t; \sigma)}{1 - n(k_t, z_t; \sigma)} - (1 - \alpha) z_t k(k_t, z_t; \sigma)^{\alpha} n(k_t, z_t; \sigma)^{-\alpha} \\ c(k_t, z_t; \sigma) + k(k_t, z_t; \sigma) - z_t k(k_t, z_t; \sigma)^{\alpha} n(k_t, z_t; \sigma)^{1-\alpha} - (1 - \delta) k(k_t, z_t; \sigma) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

or in the implicit form

$$\mathrm{E}_t H(c(k_t, z_t; \sigma), c(k(k_t, z_t; \sigma), z_{t+1}; \sigma), n(k_t, z_t; \sigma), n(k(k_t, z_t; \sigma), z_{t+1}; \sigma), k(k_t, z_t; \sigma), k_t, z_t; \sigma).$$

Omitting the evaluation point $(k, 1; 0)$ we have the approximate solution

$$c_t = c + c_k(k_t - k) + c_z z_t + c_\sigma \sigma,$$

$$n_t = n + n_k(k_t - k) + n_z z_t + n_\sigma \sigma,$$

$$k_{t+1} = k + k_k(k_t - k) + k_z z_t + k_\sigma \sigma.$$

From the steady state we already have the first three terms c, n, k. Next we obtain a system deriving $F(k_t, z_t; \sigma) = 0$ with respect to k_t, z_t, σ and then evaluate it at $(k, 1; 0)$. Let H_i be the partial derivative of H with respect to the ith argument, dropping the evaluation point we have a system of 9 equation in 9 unknown $c_k, c_z, c_\sigma, k_k, k_z, k_\sigma, n_k, n_z, n_\sigma$:

$$F_k = H_1 c_k + H_2 c_k k_k + H_3 n_k + H_4 n_k k_k + H_5 k_k + H_6 = 0,$$

$$F_z = H_1 c_z + H_2(c_k k_z + c_z \phi) + H_3 n_z + H_4(n_k k_z + n_z \phi) + H_5 k_z + H_7 = 0,$$

$$F_\sigma = H_1 c_\sigma + H_2(c_k k_\sigma + c_\sigma) + H_3 n_\sigma + H_4(n_k k_\sigma + n_\sigma) + H_5 k_\sigma + H_8 = 0.$$

The first three equations in $F_k = 0$ implies a quadratic equation in k_k with two solutions $k_k < 1$ and $k_k > 1$. As before we can rule out the latter since

$$\begin{aligned} k_t &= k + k_k(k_{t-1} - k) + k_z z_t + k_\sigma \sigma = k + k_k[k_k(k_{t-2} - k) + k_z z_{t-1} + k_\sigma \sigma] + k_z z_t + k_\sigma \sigma \\ &= ... = \sum_{j=1}^{\infty} k_k^{j-1}[k_z z_{t-j} + k_\sigma \sigma] + \lim_{t \to \infty} k_k^t (k_0 - k) = +\infty. \end{aligned}$$

We can find k_k, c_k, n_k solving the other equations. Then, we can solve $F_z = 0$ for k_z, c_z, n_z. Finally, the last block $F_\sigma = 0$ is a linear homogeneous system in $k_\sigma, n_\sigma, c_\sigma$. Hence, $k_\sigma = 0, n_\sigma = 0, c_\sigma = 0$. Again, we can do all this algebra easily with a computer.

III. Global methods

Local solutions are accurate only around the approximation point and cannot handle large changes in the economy. Moreover, they do not work well with non-differentiabilities and discontinuities such as borrowing limits and defaults. Global solution allow us to solve more interesting economic problems. Global function approximation algorithms can be discrete or continuous. In the discrete case we replace a possibly continuous state space X with a discrete one G_X and the approximate solution $\hat{d}$ is a mapping from a grid $G_X \in X$ into a grid $G_d \in \mathbb{R}^m$. In the continuous case we approximate d on X using projections.

1 Projection methods

Given the functional equation problem

$$H(d) = 0$$

where $d : X \subseteq \mathbb{R}^n \to \mathbb{R}^m$. We approximate $d \in J^1$ by $\hat{d}(x;\theta) = (\hat{d}_1(x;\theta_i), ..., \hat{d}_m(x;\theta_m))'$,

$$\hat{d}_i(x;\theta_i) = \sum_{j=1}^{n} \theta_{ij}\phi_j(x), i = 1, ..., m.$$

where $\theta_i = (\theta_{i1}, ..., \theta_{in})' \in \mathbb{R}^n$ for $i = 1, ..., m$ and $\theta = (\theta_1', ..., \theta_m')'$. The projection method is given by the following steps:

Algorithm 1. (Projection).

1. Choose basis functions $\phi_j : X \to \mathbb{R}$ for $j = 1, ..., n$ given θ.
2. Compute $\hat{d}(x;\theta)$ and the residual function $R(x;\theta) = H(\hat{d}(x;\theta))$.
3. Given some metric $\rho : J^2 \to \mathbb{R}_+$ compute $\hat{\theta} = \text{argmin}_{\theta \in \mathbb{R}^{m \times n}} \rho(R(x;\theta))$.

We compute the approximation to the true solution by making R as close to $0 \in J^2$ as possibile. Projection methods differ by the choice of the basis $\phi_1(x), ..., \phi_n(x)$ and metric function. Spectral methods use global basis functions, i.e. ϕ_j are nonzero for most of the state space X, e.g. Chebyshev polynomials. Finite element methods use local basis functions, i.e. ϕ_j are nonzero on only small subsets of X, e.g. splines.

Metric functions. For simplicity let $m = 1, J^2 := \mathbb{R}$. We can use least squares $\min_{\theta \in \mathbb{R}^n} \int_X R(x;\theta)^2 dx$ with first order conditions

$$\int_X D_{\theta_j} R(x;\theta) R(x;\theta) dx = 0, \ \ \forall j = 1, ..., n.$$

An alternative is collocation where the approximation is exact at n collocation points

$$R(x_j;\theta) = 0, \forall j = 1, ..., n.$$

In orthogonal collocation we pick the collocation points as roots of the n-th-order Chebyshev polynomial or extrema of the $n-1$-th order Chebyshev polynomial in each dimension of x. Finally, in the Garlekin projection we impose an orthogonality condition

$$\int_X \phi_j(x) R(x;\theta) dx = 0, \ \ \forall j = 1, ..., n.$$

Chebyshev polynomials. A first simple basis is the monomial basis $\{1, x, x^2, x^3, ...\}$. Given an approximation of order n the $m \times (n-1)$ matrix containing monomials is ill-conditioned. The monomials are collinear, i.e. almost linearly dependent and vary considerably in size leading to large rounding errors and scaling problems. We need an orthogonal basis. Chebyshev polynomials are defined recursively by $T_0(x) = 1, \ T_1(x) = x, \ T_2(x) = 2x^2 - 1, \ ..., \ T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$ or

$$T_n(x) = \cos(n \arccos(x))$$

where $x \in [-1, 1]$. We can map $x \in [-1, 1]$ into $z \in [a, b]$ by trasforming the data linearly $z = a + .5(b-a)(x+1)$. Chebyshev polynomials are orthogonal in the sense that $\int_X w(x) T_j(x) T_k(x) dx = 0$ with weight function $w(x) = (1 - x^2)^{0.5}$. There are n distinct roots of $T_n(x) = 0$ on $x \in [-1, 1]$ given by

$$x_k = -\cos\left(\frac{\pi(2k-1)}{2n}\right)$$

for $k = 1, 2, ..., n$, where the minus puts them in ascending order. We can approximate $f : [-1, 1] \to \mathbb{R}$ using $\hat{f}(x) = \sum_{j=1}^n \theta_j T_{j-1}(x)$ for $j = 1, ..., n$. We find the coefficients by least squares $\theta_j = \text{argmin} \sum_{k=1}^m (f(x_k) - \sum_{j=1}^n \theta_j \phi_j(x_k))^2$ using the $m \geq n$ roots of $T_m(x)$ as nodes $\{x_k\}_{k=1}^m$.

Splines. The chebyshev polynomials work well for well-behaved smooth functions. The idea of a spline interpolation is to divide the domain into nonintersecting subdomains called elements and then approximate the function on each element by some polynomials. A function $\phi : [a, b] \subseteq \mathbb{R} \to \mathbb{R}$ is a spline of order k if $\phi \in C^{k-2}([a, b])$ and there is a grid of nodes $a = x_0 < x_1 < ... < x_n = b$ such that ϕ is a polynomial of order at most $k - 1$ on each subinterval $[x_i, x_{i+1}], i = 0, 1, ..., n - 1$. For example, a spline of order 2 or linear spline is a picewise linear function. A spline of order 4 or cubic spline is given by $\phi_i(x) = a_i + b_i x + c_i x^2 + d_i x^3, i = 0, 1, ..., n - 1$ with $4(n - 1)$ coefficients to be computed. We get the coefficients for each polynomial making sure they satisfy the constraints imposed by the n-knots and some continuity and smoothness conditions: for example $f(x_i) = \phi_i(x_i)$, by continuity $\phi_i(x_{i+1}) = \phi_{i+1}(x_{i+1})$ these are $n+n-2$ equations and similarly we can impose conditions on the derivatives. The natural spline satisfies $\phi''(x_0) = \phi''(x_n) = 0$. The hermite spline satisfies $\phi'(x_0) = f'(x_0), \phi'(x_n) = f'(x_n)$. The not-a-knot end condition requires that $\phi'''(x) \in C(\{x_1, x_{n-1}\})$.

2 Value function and Euler equation algorithms

Global methods, both discrete and continuous, lead to iterate on the value function using the Bellman equation or on the policy functions using an equilibrium system that usually has an Euler equation at its core, and sometimes on both. In this section we will cover the most common schemes. However, combinations and extensions of these are possible.

Value function iteration. Consider the following iteration scheme at the n-th iteration

$$v_{n+1}(x) = \max_{x' \in G(x)} \{f(x, x') + \beta \mathrm{E} v_n(x')\} \tag{2}$$

with $f : X \to \mathbb{R}$ for some constraint correspondence $G : X \rightrightarrows \mathbb{R}_+$.

Algorithm 2. (value function iteration).

1. Start with an initial guess $v_n(x)$.
2. Given $v_n(x)$ solve the maximization on the right of $v_{n+1}(x)$ and get $x' = g_{n+1}(x)$.
3. Update $v_{n+1}(x) = f(x, g_{n+1}(x)) + \beta \mathrm{E} v_n(g_{n+1}(x))$ and iterate until convergence.

By the Banach fixed point theorem the sequences $\{v_n(x)\}_{n=0}^{\infty}, \{g_n(x)\}_{n=1}^{\infty}$ converge to the solution $v(x), g(x)$ of (2) for any guess v_0. We can implement value function iteration using discrete or continuous methods.

Discrete value function iteration. Consider the stochastic neoclassical growth model with inelastic labor supply. Let $k \in A = \mathbb{R}_+$ and $z \in Z$ be a discrete Markov process with transition probabilities T. Therefore, Z is a discrete set but A is not discrete. We can discretize the state space $A \times Z$ using a grid for the states. Hence,

$$v(k_i, z_j) = \max_{k' \geq 0} \left[u(k_i, z_j, k') + \beta \sum_{j'=1}^{J} v(k', z'_{j'}) T(z'_{j'}) \right].$$

for $i = 1, 2, ..., I$ and $j = 1, 2, ..., J$.

Algorithm 3. (Discrete VFI).

1. Define a grid for states $G_A = \{k_1, k_2, ..., k_I\}, G_Z = Z = \{z_1, z_2, ..., z_J\}$.
2. Compute the $S \times I$ payoff matrix M with $S = IJ$ and generic element

$$M_{s,i'} = u(z_j f(k_i) + (1 - \delta)k_i - k_{i'}), \quad \forall s = 1, ..., S, \ \ \forall i' = 1, ..., I.$$

   where $s = (j-1)I + i, \forall j, \forall i$ runs over all possible states (k_i, z_j).
3. Set $M_{s,i'} = \omega << 0$ if $z_j f(k_i) + (1 - \delta)k_i - k_{i'} < 0$.
4. Guess a $v^n \in \mathbb{R}^S$ say $v^n_s := v^n(k_i, z_j) = 0, \forall i, \forall j$.
5. Compute the $J \times I$ matrix W with generic element

$$W_{j,i'} = \textstyle\sum_{j'=1}^{J} T(z_{j'}; z_j) v^n(k_{i'}, z_{j'}).$$

6. Compute the $S \times I$ matrix A for each Bellman operator $A_{s,i'} = M_{s,i'} + W_{j,i'}$.
7. Compute $v^{n+1} = (\max_{i'}(A_{1,i'}), ..., \max_{i'}(A_{S,i'}))'$.
8. Iterate from step 4 until $\max(|v^{n+1}_s - v^n_s|) < \varepsilon$.

In step 2 the matrix M is such that for each z on the grid we have a submatrix with I rows while the columns contain a possible value for k' on the grid for each (k_i, z_j). In step 3 to avoid negative consumption we replace the cell with an very negative value so $k_{i'}$ will not be optimal. In step 6 let ι be a vector of 1, in matrix notation $A = M + \iota_{I \times 1} \otimes W$. In step 7 we find the maximum value in each row and the associated maximizer k' on the

grid. There are several ways to speed up the algorithm. For example, in the iteration n given $v_{n_0}(k,z)$ and $g_{n_0}(k,z)$ iterate M more times fixing the policy function g_{n_0}

$$v_{n_m}(k,z) = u(k,z,g_{n_0}(k,z)) + \beta \mathrm{E} v_{n_{m-1}}(g_{n_0}(k,z)), \;\; m = 1,...,M.$$

Practice suggets $M = 10$ and few policy iterations in the middle of the value iterations. Other examples exploit monotonicity and concavity of policy functions and value function. If z_t follows a continuous Markov process, for example $z_t \sim$ AR(1), we have

$$v(k,z) = \max_{k' \in G(k,z)} \left[u(k,z,k') + \beta \int_Z v(k',z') dT(z') \right].$$

To solve the functional equation problem on a grid we need to discretize also Z. There are two popular ways to do it. We need $G_Z = \{z_1,...,z_J\}$ and the markov transition probabilities $\hat{T}$. Consider the AR(1) exogenous process

$$z_{t+1} = \mu(1-\rho) + \rho z_t + \varepsilon_{t+1}, \varepsilon_t \overset{iid}{\sim} N(0,\sigma^2).$$

Recall that $\mathrm{E}(z_t) = \mu, \mathrm{Var}(z_t) = \sigma^2(1-\rho)^{-1}$. We want to construct a discrete Markov chain to approximate $\{z_t\}$. We can discretize z_t using the Tauchen (1986) method. Choose J equally spaced points $z_J\mu + 3\sigma, z_1 = \mu - 3\sigma, z_j = z_1 + (z_n - z_1)(J-1)^{-1}(j-1)$. Construct the midpoints $\{d_j\}_{j=1}^{J-1}$ given by $d_j = (z_{j+1} + z_j)/2$ and intervals $I_1 = (-\infty, d_1), I_2 = (d_1, d_2), ..., I_{J-1} = (d_{J-2}, d_{J-1}), I_J = (d_{J-1}, +\infty)$. Find the transition probabilities with

$$\begin{aligned}
\pi_{j'j} &= \Pr(z' \in I_{j'} | z = z_j) = \Pr(z_{j'} - d_{j'-1} < z' \leq z_{j'} + d_{j'} | z = z_j) \\
&= \Pr(z_{j'} - d_{j'-1} < \mu(1-\rho) + \rho z_j + \varepsilon' \leq z_{j'} + d_{j'} | z = z_j) \\
&= \Phi(a_1^{j'}) - \Phi(a_2^{j'}).
\end{aligned}$$

where $a_1^{j'} = \sigma^{-1}[z_{j'} - d_{j'-1} - (1-\rho)\mu - \rho z_j], a_2^{j'} = \sigma^{-1}[z_{j'} + d_{j'} - (1-\rho)\mu - \rho z_j]$ and $\pi_{j'j} = 1 - \Phi(a_2^{j'})$ if $j' = J$ and $\Phi(a_1^{j'})$ if $j = 1$.

We can extend this method to multivariate exogenous processes by diagonalization. Given the VAR process $z_t = \Phi z_{t-1} + \varepsilon_t$ with nondiagonal covariance matrix Σ using the spectral decomposition of $\Sigma = QDQ'$ yields

$$Q'z_t = Q'\Phi QQ'z_{t-1} + Q'\varepsilon_t$$

where $Q'\Sigma Q = D$. We can extend the method to higher-order AR processes writing them as multivariate first order systems.

A better alternative when ρ is very close to 1 is Rouwenhorst (1995). Set $z_1 = -\lambda\sigma, z_J = \lambda\sigma$ where $\lambda = 3$ or $\lambda = \sqrt{J-1}$. Compute a linear grid $z_j = z_1 + (z_J - z_1)(j-1)(J-1)^{-1}$. For the transition matrix when $J = 2$ we use

$$P_2 = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}$$

where $p = q = (1+\rho)/2$. When $J \geq 3$ we construct the transition matrix recursively by

$$P_J = p\begin{bmatrix} P_{J-1} & 0 \\ 0 & 0 \end{bmatrix} + (1-p)\begin{bmatrix} 0 & P_{J-1} \\ 0 & 0 \end{bmatrix} + (1-q)\begin{bmatrix} 0 & 0 \\ P_{J-1} & 0 \end{bmatrix} + q\begin{bmatrix} 0 & 0 \\ 0 & P_{J-1} \end{bmatrix}$$

where each block matrix is $J \times J$ with P_{J-1} of dimension $(J-1) \times (J-1)$ and zeros elsewhere. Dividing all but the top and bottom rows by 2 ensures that the sum of the elements of each row is 1.

Continuous value function iteration. Consider again the neoclassical growth models with TFP shocks and inelastic labor supply. The continuous version of the value function iteration uses a projection algorithm.

Algorithm 4. (Continuous VFI).

1. Set the approximation order p and the number of points $q+1$ in each dimension $\{k_i, z_j\}$ where $i, j = 1, 2, ..., q+1$.
2. Guess a solution $v^m(k,z) = \sum_{s=1}^{p} \theta_s^m \phi_s(k,z)$, i.e. guess $\theta^m \in \mathbb{R}^p$.
3. Find $k^m(k_i, z_j) = \text{argmax}_{k' \in [k_1, k_{q+1}]}\{u(z_j f(k_i) + (1-\delta)k_i - k') + \beta \text{E} v^m(k', z')\}, \forall i, j$
4. Update $v^{m+1}(k_i, z_j) = u(z_j f(k_i) + (1-\delta)k_i - k^m(k_i)) + \beta \text{E} v^m(k^m(k_i, z_j), z'), \forall i, j.$
5. Given $\{v^{m+1}(k_i, z_j)\}$ solve for the projection coefficients θ_q^{m+1} the linear system

$$v^{m+1}(k_i, z_j) = \sum_{s=1}^{p} \theta_s^{m+1} \phi_s(k_i, z_j), \forall i, j.$$

6. Iterate from step 2 until $\|\theta^{m+1} - \theta^m\| < \varepsilon$.

The points $k_1 < k_2 < ... < k_{q+1}$ and $z_1 < z_2 < ... < z_{q+1}$ in step 1 are either nodes or knots depending if we are using Chebyshev polynomials or splines. Notice that

we can evaluate v^m over the entire state space. If we use Chebyshev polynomials then $v^m(k,z) = \sum_{s=1}^{p} \theta_s^m T_{s-1}(g(k,z))$ where $g : [a,b] \to [-1,1]$. In step 3 we need to use numerical integration and numerical maximization or a nonlinear solver for the first order conditions. In step 5 we have $(q+1)^2$ equations in p unknown. If $p = (q+1)^2$ we are using collocation. If $(q+1)^2 > p$ we are using least squares.

Extensions. Value function iteration can be extended to the case of elastic labor supply. In this case we have an additional static variable $a(x)$ in the optimization problem and an associated intratemporal optimal condition. In this case we want to solve

$$v(x) = \max_{x', a' \in G(x,a)} \{f(x, x', a) + \beta \mathrm{E} v(x')\}.$$

In finite horizon the Bellman equation is of the form

$$v_t(x) = \max_{x' \in G(x)} \{f(x, x') + \beta \mathrm{E} v_{t+1}(x')\}.$$

This suggest a backward solution $\{v_t\}_{t=0}^{T}$ starting from a guess in the final period $v_{T+1} = 0$. So, there is no need to iterate until convergence.

VFI in practice. We can always use value function iteration to solve partial equilibrium models. In general equilibrium models with a representative agent value function iteration is used mostly to solve centralized economies. If the welfare theorems hold value function iteration is also useful to find the competitive allocations solving the social planner problem. In decentralized economies agents need to know the law of motion of aggregate states to know, at least in expectations, future prices. However, we know this object only after we solved the Bellman equation. Therefore, we would need an additional outer loop iterating over aggregate states and their dynamics. In this case convergence is not guaranteed. As we will see in Chapter 5, value function iteration can be used in heterogeneous agent models for the computation of competitive equilibria. In particular, to solve the households' consumption-saving optimization problem. In the case of a representative agent to find global solutions in decentralized economies with frictions we might want to consider alternative iteration schemes that solve directly the equilibrium system, which typically has an Euler equation at its core.

Discrete and continuous policy function iteration. Consider the neoclassical growth model with TFP shocks and inelastic labor supply. The equilibrium of the centralized or decentralized model is characterized by an Euler equation.

$$u'(g^c) = \beta \mathrm{E}_t[u'(g^c(g^k, z_{t+1}))(z_{t+1} f'(g^k) + 1 - \delta)],$$
$$g^c = z_t f(k_t) + (1-\delta)k_t - g^k.$$

A natural alternative to value function iteration is to iterate over the policy functions on the Euler equation. The iteration scheme at the n-th iteration is given by

$$u'(zf(k)+(1-\delta)k - g^k_{n+1}) = \\ \beta \mathrm{E}[u'(z' f(g^k_{n+1}) + (1-\delta)g^k_{n+1} - g^k_n)(z' f'(g^k_{n+1}) + 1 - \delta)].$$

In infinite horizon we iterate untile convergence, in finite horizon we can solve this backward using $g^k_{T+1} = 0$. We can implement policy function iteration using discrete or continuous methods. However, in contrast with value function iteration in general convergence is not guaranteed. Moreover, given the nonlinearity of the Euler equation we need a root-finder algorithm to compute g^k_{n+1}. We can also use a plain projection algorithm iterating over projection coefficients with residual function at the n-th iteration

$$u'(zf(k)+(1-\delta)k - g^k_n) - \\ \beta \mathrm{E}[u'(z' f(g^k_n) + (1-\delta)g^k_n - g^k_n(g^k_n, z'))(z' f'(g^k_n) + 1 - \delta)] = 0.$$

3 Error analysis

Informal accuracy tests are always a good practice in quantitative work. Change the parameters values, both structural and computational, and understand how the properties of the solution change. A formal way to evaluate the approximation error is to use Euler equation errors. We construct a new grid G'_A different from the grid on which we approximated the consumption policy function G_A. For example G'_A might contain the midpoints between nodes of G_A. For each j interpolate $(k_i, g^c(k_i, z_j))$ where $k_i \in G_A, \forall i$ over G'_A to get $g^c(k_m, z_j)$ where $k_m \in G'_A, \forall m$. The Euler equation errors are given by

$$e_{m,j} = u'(g^c(k_m, z_j)) - \beta \mathrm{E}[u'(g^c(g^k(k_m, z_j), z'))(z' f'(g^k(k_m, z_j)) + 1 - \delta)]$$

This might require numerical integration that should be done accurately. We can investigate the pattern of the errors, their maximum, and average.

References

Arnoud, Antoine, Fatih Guvenen, and Tatjana Kleineberg (2019). "Benchmarking Global Optimizers". In: *NBER Working Paper No. 26340*.

Blanchard, Olivier and Charles Kahn (1980). "The Solution of Linear Difference Models under Rational Expectations". In: *Econometrica* 48 (5), pp. 1305–1311.

King, Robert and Mark Watson (1998). "The Solution of Singular Linear Difference Systems under Rational Expectations". In: *International Economic Review* 39 (4), pp. 1015–1026.

Klein, Paul (2000). "Using the generalized Schur form to solve a multivariate linear rational expectations model". In: *Journal of Economic Dynamics and Control* 24, pp. 1405–1423.

Rouwenhorst, Geert (1995). *Asset Pricing Implications of Equilibrium Business Cycle Models*.

Sims, Christopher (2002). "Solving Linear Rational Expectations Models". In: *Computational Economics* 20, pp. 1–20.

Tauchen, George (1986). "Finite state markov-chain approximations to univariate and vector autoregressions". In: *Economics Letters* 20 (2), pp. 177–181.

Uhlig, Harald (1999). "A Toolkit for Analysing Nonlinear Dynamic Stochastic Models Easily". In: *Working Paper*.