Modern Numerical Methods for Economists

Eric R. Young
Professor
Department of Economics
University of Virginia
Senior Research Economist
Research Department
Federal Reserve Bank of Cleveland

January 19, 2023

1 Introduction

> "A state of the art computation requires 100 hours of CPU time on a state of the art computer, independent of the current time." Teller's Law of Computing
>
> "Assume you currently have N errors in your code. Fix K of those errors. You now have N errors in your code." Dan Bernheim's Law of Programming
>
> "The good thing about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do." Ted Nelson

In economics, there are four main numerical tasks that you will typically perform. First, the computation of numerical derivatives and integrals; second, the numerical maximization or minimization of nonlinear functions, frequently with constraints; third, the computation of the roots of linear and nonlinear equations; and fourth, the approximation of functions. The goal of these notes is not just to give you enough of an introduction that you can apply these four basic operations to solve economic models; my goal is more ambitious, in that I want you to be close to the frontier where you can solve more economic models than your competitors.

For example, take the dynamic program from the optimal stochastic growth model:

$$v(k,z) = \max_{k'} \left\{ \log\left(\exp(z)\, k^{\alpha} + (1-\delta)\, k - k'\right) + \frac{\beta}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} v\left(k', z'\right) \exp\left(-\frac{\left(z' - \rho z\right)^2}{2\sigma^2}\right) dz' \right\}.$$

The first-order necessary and sufficient conditions for this problem are

$$\frac{1}{\exp(z)\, k^{\alpha} + (1-\delta)\, k - k'} = \frac{\beta}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{\alpha \exp(z')\, (k')^{\alpha-1} + 1 - \delta}{\exp(z')\, (k')^{\alpha} + (1-\delta)\, k' - k''} \exp\left(-\frac{\left(z' - \rho z\right)^2}{2\sigma^2}\right) dz'$$

$$\lim_{t\to\infty} \left\{ \beta^t \frac{k_{t+1}}{c_t} \right\} = 0;$$

we can generally check the transversality condition after solving the rest of the model, or use it to eliminate some potential solutions. We will also consider environments where transversality does not help eliminate solutions because it is always satisfied.

One method for solving this model involves iterating on the Bellman operator

$$v^1(k,z) =$$

$$\left(Tv^0\right)(k,z) = \max_{k'} \left\{ \log\left(\exp(z)\, k^{\alpha} + (1-\delta)\, k - k'\right) + \frac{\beta}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} v^0\left(k', z'\right) \exp\left(-\frac{\left(z' - \rho z\right)^2}{2\sigma^2}\right) dz' \right\}$$

or on the Euler operator defined implicitly by

$$g^1(k,z) = \left(Fg^0\right)(k,z):$$

$$\frac{1}{\exp(z)\, k^{\alpha} + (1-\delta)\, k - g^1(k,z)} =$$

$$\frac{\beta}{\sigma\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{\alpha \exp(z')\, g^1(k,z)^{\alpha-1} + 1 - \delta}{\exp(z')\, g^1(k,z)^{\alpha} + (1-\delta)\, g^1(k,z) - g^0\left(g^1(k,z), z'\right)} \exp\left(-\frac{\left(z' - \rho z\right)^2}{2\sigma^2}\right) dz'.$$

Implementing either operator involves approximating a function (v^0 or g^0) on the computer and then integrating this function (or a function of it) to obtain the conditional expectation. To implement T we have to solve a constrained maximization and to implement F we need to solve a nonlinear equation. More general models might require us to do both – solve maximizations to get demand functions, for example, then nonlinear equations to find market clearing prices.

Alternatively, we may also "directly" solve either equation by assuming some finite representation v or g; it is substantially easier to solve the Euler equation this way than the Bellman equation, because the 'max' operator must be approximated using inequalities. To directly solve the Bellman equation we can sometimes solve the problem

$$v(k,z) = \min_{V(k,z),k'(k,z)} \{V(k,z)\}$$

subject to

$$V(k,z) = \log\left(\exp(z)\,k^{\alpha} + (1-\delta)\,k - k'\right) + \frac{\beta}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty} V\left(k', z'\right)\exp\left(-\frac{\left(z'-\rho z\right)^2}{2\sigma^2}\right)dz'$$

$$V(k,z) \geq \log\left(\exp(z)\,k^{\alpha} + (1-\delta)\,k - \widetilde{k}'\right) + \frac{\beta}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty} V\left(\widetilde{k}', z'\right)\exp\left(-\frac{\left(z'-\rho z\right)^2}{2\sigma^2}\right)dz'$$

for all $\widetilde{k}'$ such that

$$\exp(z)\,k^{\alpha} + (1-\delta)\,k - \widetilde{k}' \geq 0.$$

For some models, iterative methods can fail while direct ones succeed; for others, the opposite. It is important for you to have programs that can solve models both ways, as it is not always clear in advance which one will work.

1.1 Does the Value Function Exist?

It is important to understand when solutions to the above problems exist; many a frustrated student has discovered their seemingly-unsolvable problem really is unsolvable, rendering all the time spent on trying to find the solution wasted. Take the generic Bellman equation

$$v(x) = \max_{a}\left\{r(x,a) + \beta v\left(x'\right) : a \in \Gamma(x), x' = t(x,a)\right\}$$

where x are the states, a are the actions, $\Gamma(x)$ is the feasible action correspondence (the set of actions that are feasible to the decision maker given the current state, $t(x,a)$ is the state transition equation, and β is the discount factor. The function $v(x)$ is the unknown of this equation. We know that inserting a function into the right-hand-side for $v(\cdot)$ and performing the maximization gives us a new function for the left-hand-side; these functions are not necessarily the same. The operator T takes a function $w : X \to \mathcal{R}$ and maps it to a function $Tw : X \to \mathcal{R}$ according to

$$(Tw)(x) = \max_{a}\left\{r(x,a) + \beta w\left(x'\right) : a \in \Gamma(x), x' = t(x,a)\right\}.$$

Differentiation and integration perform the exact same task – they associate two functions (or classes of functions) by "turning" one function into another. For example, let D be the differentiation operator; then $Dx^2 = 2x$, a new function which is related to the old one via the operator D. The Bellman operator works exactly the same way. We know that, if the true value function exists, it satisfies the Bellman equation. That is, if we feed the Bellman operator v we get v

back; $(Tv)(x) = v(x)$. In other words, the value function is a fixed point of the T operator in the space of functions, just as the functions $f(x) = 0$ and $f(x) = \exp(x)$ are fixed points of the D operator.

We need to first prove that the Bellman operator has at least one fixed point; that is a necessary but not sufficient condition for the value function to exist. The reason it is not sufficient is that we also need to rank-order the fixed points, which might not be possible; for example, how do we rank-order functions that cross over each other? This problem is equivalent to the problem of well-ordering sets of real numbers in more than one dimension, which is generally impossible without additional structure on the set. Second, we would like to be able to prove that the Bellman operator has only one fixed point; failing that, we need to prove that the fixed points can be well-ordered (formally, that they have a lattice structure), which in this case requires that they do not cross each other. Fortunately, the Bellman equation generally has the right properties for us to obtain a uniqueness condition (but not always).

Assuming that we have enough structure that solutions to the Bellman equation exist, there will be (at least) one action for each value of the state that is optimal. Denote this mapping the **policy function**:

$$a^* = g(x).$$

Policy functions are like demand functions – they tell you what to do in the event that the state becomes x; note that, in the infinite horizon case, policy functions are invariant (they are used in every period). In general g might be multi-valued, so we should call it the policy correspondence, but we will generally assume sufficient concavity to rule out multiple solutions.

Our goal is to find conditions under which the Bellman operator is a strict contraction on a complete normed vector space, because strict contractions have unique fixed points that are approached by iterations. We then have achieved a large part of our goal; we will have proven that the value function exists, is the unique solution to the Bellman equation, and that iterating on the Bellman operator will converge to the value function from any initial guess.

To examine the conditions under which the Bellman operator will be a strict contraction, define the 'sup' norm (also called the L^∞ norm) of a continuous function by

$$\|f\|_\infty = \sup_x \{|f(x)| : x \in X\}.$$

Denote by $C^b(X)$ the space of continuous functions for which this norm is finite (b for bounded); this subset of continuous functions equipped with the sup-norm $\left(C^b(X), ||\cdot||_\infty\right)$ is a complete normed vector space. Sufficient conditions for a mapping to be a contraction are that the map is monotone (if $w_1(x) \geq w_0(x)\ \forall x \in X$ then $(Tw_1)(x) \geq (Tw_0)(x)\ \forall x \in X$) and that it discounts constant functions (if A is a constant function then $(Tw + A)(x) \geq Tw(x) + \theta A$ for some $\theta < 1$); these conditions are often called Blackwell's Sufficiency Conditions. Because we will see a more general version of these conditions below, I am skipping their explicit statement for now.

We now show that the Bellman operator is a contraction provided we make some reasonable assumptions.

Theorem 1 *Assume the one-period return function r and the transition function t are continuous on $X \times A$; the feasible actions correspondence Γ is continuous and compact-valued; and there exists an $M < \infty$ with*

$$\max_{a \in \Gamma(x)} \{r(x,a)\} \leq M$$

$\forall x \in X$ and $\beta < 1$. Under these conditions the Bellman operator is a strict contraction map.

It is easy to see that $T : C^b(X) \to C^b(X)$; that is, if a function w is bounded then Tw will also be bounded:

$$\begin{aligned}
|(Tw)(x)| &= \left| \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a))\} \right| \\
&\leq \left| \max_{a \in \Gamma(x)} \{r(x,a)\} \right| + \left| \max_{a \in \Gamma(x)} \beta w(t(x,a)) \right| \\
&\leq \max_{a \in \Gamma(x)} \{|r(x,a)|\} + \beta \max_{a \in \Gamma(x)} \{|w(t(x,a))|\} \\
&\leq \|r\|_\infty + \|w\|_\infty .
\end{aligned}$$

Next, let $v(x) \geq w(x)\ \forall x \in X$. Then

$$\begin{aligned}
(Tv)(x) &= \max_{a \in \Gamma(x)} \{r(x,a) + \beta v(t(x,a))\} \\
&\geq \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a))\} \\
&= (Tw)(x).
\end{aligned}$$

Monotonicity is proven. Now, let A be a constant and let $w \in \Xi$. We have

$$\begin{aligned}(T(w + A\phi))(x) &= \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a)) + \beta A\} \\ &\leq \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a))\} + \beta A \\ &\leq (Tw)(x) + \beta A.\end{aligned}$$

Discounting constant functions is satisfied if $\beta < 1$. We have shown that the Bellman operator is a contraction under the stated assumptions, which seem fairly uncontroversial and are mainly aimed at guaranteeing the maximum exists.

We need only make one further step to show that there exist optimal paths that solve the sequence problem; after all, these are really the objects we are after. It may be the case that the value function is unattainable, that the max should be replaced by a sup. We are not interested in these types of problems – they do not solve the sequence problem. To this end, note that the value function must satisfy

$$v(x_0) \geq \sum_{t=0}^{\tau} \beta^t r(x_t, a_t) + \beta^\tau v(x_{\tau+1})$$

for any $\tau \geq 0$ and the any sequence of actions. To match up with our previous problem in sequence form, we need only show that the "size" of the last term goes to zero as $\tau \to \infty$. But we know that

$$|v(x_{\tau+1})| \leq \|v\|_\infty < \infty$$

because v is bounded, which implies

$$\beta^\tau |v(x_{\tau+1})| \leq \beta^\tau \|v\|_\infty .$$

The right hand side converges to zero because $\beta < 1$ and $\|v\|_\infty$ is finite; therefore we have

$$\beta^\tau |v(x_{\tau+1})| \to 0.$$

We have thus proven that the value function exists; that it can be obtained by iterating on the Bellman operator; and that optimal paths generated by the policy function solve the sequence problem. This last property implies that once we are on an optimal path it is optimal to remain there, so that "plans" announced at time t for time $t+k$ are carried out at $t+k$. This property is called the **Principle of Optimality**.

Having shown that the Bellman operator is a strict contraction, it is an immediate implication of the triangle inequality that

$$\frac{1}{1+\beta}\left\|T^{n+1}v_0 - T^n v_0\right\|_\infty \leq \left\|T^n v_0 - v\right\|_\infty \leq \frac{1}{1-\beta}\left\|T^{n+1}v_0 - T^n v_0\right\|_\infty;$$

that is, the difference between our current "iterate" $T^n v_0$ and the true solution v is bounded by the distance between the last two iterates.

When is the assumption that r is bounded without loss of generality? In the growth model, if $u\left(f\left(k\right)+\left(1-\delta\right)k\right)$ is bounded both above and below – if either function is bounded above and below this condition will hold – then boundedness of r is assured. Thus, all we need is $\beta < 1$ to ensure existence. We can also take r to be bounded above if f obeys the following "maximum sustainable stock" condition:

Definition 2 *f obeys the **maximum sustainable stock condition** if there exists $\overline{k} > 0$ such that $f\left(k\right) < k$ for all $k > \overline{k}$.*

This condition in effect bounds the function $u\left(f\left(k\right)+\left(1-\delta\right)k\right)$ by placing a strict upper bound on the capital stocks we need consider: $k \in [0, L]$ where $L = \max\left\{k_0, \overline{k}\right\}$. If we do not start with capital above $\overline{k}$ we will never observe it, and if we do start with a capital stock that high we will be forced to run it down. The Cobb-Douglas production function $f\left(k\right) = Ak^\alpha$ is one which satisfies the maximum sustainable stock condition provided $\alpha < 1$; two that do not are the CES production functions $f\left(k\right) = A\left(\alpha k^\nu + 1 - \alpha\right)^{\frac{1}{\nu}}$ for $\nu \in (0, 1]$ (Carroll and Young 2018; Krusell and Smith 2016) and the case of $\alpha = 1$ (Romer 1986; King and Rebelo 1993). Note that we still need to worry about the lower boundary as $k \to 0$. Certain important classes of return functions – polynomial return functions, and in particular quadratic functions – are always bounded above and below in any compact space.

Cases where the model is not bounded can be handled by defining a different norm (see Alvarez and Stokey 1998 or Boyd 1991). There are two possibilities here. One, the state space is unbounded and f does not obey a maximum sustainable stock condition; this case is simpler and we will discuss it first. The second case is more problematic, as it involves the utility function being unbounded below on the compact state space; for example, if $u\left(c\right) = \log\left(c\right)$, then as $c \to 0$ the utility function diverges to $-\infty$.

Let's start with the first case. Suppose ϕ is a function in the space of continuous functions Ξ that satisfies $\phi(x) > 0 \ \forall x \in X$. Let's call the following the "ϕ-norm" of a function:

$$\|f\|_\phi = \sup_x \left\{ \frac{|f(x)|}{\phi(x)} : x \in X \right\}. \tag{1}$$

That is, deflate the value of f by the positive function ϕ. Denote by Ξ_ϕ the subset of Ξ for which this norm is finite. This subset equipped with the ϕ-norm $\left(\Xi_\phi, \|\cdot\|_\phi\right)$ is a complete metric space. We can now state our general sufficiency conditions, the ones I skipped over earlier.

Theorem 3 *(**Blackwell-Boyd Sufficiency Conditions**) Let T be a mapping from $\left(\Xi_\phi, \|\cdot\|_\phi\right) \to \left(\Xi_\phi, \|\cdot\|_\phi\right)$ such that (a) T is **monotone** – $f, g \in \Xi$ and $f \geq g$ implies $Tf \geq Tg$ – and (b) T **discounts constant functions** – $\forall f \in \Xi$ and any constant function A, $T(f + A\phi) \leq Tf + \theta A\phi$ for some $\theta < 1$. Then T is a strict contraction map.*

Proof. Let $f, g \in \Xi_\phi$ and let T satisfy (a) and (b). By the definition of $\|\cdot\|_\phi$ we must have

$$\frac{|f - g|}{\phi} \leq \|f - g\|_\phi$$

so that

$$f \leq g + \|f - g\|_\phi \phi.$$

By monotonicity,

$$Tf \leq T\left(g + \|f - g\|_\phi \phi\right).$$

By discounting constant functions,

$$Tf \leq Tg + \theta \|f - g\|_\phi \phi.$$

That is,

$$Tf - Tg \leq \theta \|f - g\|_\phi \phi.$$

Since the RHS is positive, we have

$$\frac{|Tf(x) - Tg(x)|}{\phi(x)} \leq \theta \|f - g\|_\phi$$

$\forall x$. Therefore

$$\|Tf - Tg\|_\phi \leq \theta \|f - g\|_\phi.$$

That is, T is a contraction. ■

We now show that the Bellman operator is a contraction provided we make some reasonable assumptions.

Theorem 4 *Assume the one-period return function r and the transition function t are continuous on $X \times A$; the feasible actions correspondence Γ is continuous and compact-valued; and there is a function $\phi \in \Xi$ with $\phi(x) > 0$ such that (i) there exists an M with*

$$\max_{a \in \Gamma(x)} \{r(x,a)\} \leq M\phi(x) \tag{2}$$

and (ii) there exists $\theta < 1$ such that

$$\beta \max_{a \in \Gamma(x)} \{\phi(t(x,a))\} \leq \theta\phi(x) \tag{3}$$

$\forall x \in X$. Under these conditions the Bellman operator is a strict contraction map.

Basically, this assumption says that the return function must be bounded in the current period by ϕ and the state cannot evolve in such a way that it makes the ϕ function grow too fast. These are reasonable restrictions on any economy – a household should not be able to act in such a way as to get infinite current utility or infinite utility in the future (which are the same since $\beta^t > 0$), as otherwise everything is kind of pointless to study. We now proceed to demonstrate that the Bellman equation, under the conditions specified in the theorem, will satisfy the sufficiency conditions.

It is easy to see that $T : \Xi_\phi \to \Xi_\phi$; that is, if a function w is bounded by ϕ then Tw will also be bounded by ϕ, which follows from (i) of the condition. First, let $v(x) \geq w(x)\ \forall x \in X$. Then

$$\begin{aligned}
(Tv)(x) &= \max_{a \in \Gamma(x)} \{r(x,a) + \beta v(t(x,a))\} \\
&\geq \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a))\} \\
&= (Tw)(x).
\end{aligned}$$

Monotonicity is proven. Now, let A be a constant and let $w \in \Xi_\phi$. We have

$$\begin{aligned}
(T(w + A\phi))(x) &= \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a)) + \beta A\phi(t(x,a))\} \\
&\leq \max_{a \in \Gamma(x)} \{r(x,a) + \beta w(t(x,a))\} + \beta A \max_{a \in \Gamma(x)} \{\phi(t(x,a))\} \\
&\leq (Tw)(x) + \theta A\phi(x).
\end{aligned}$$

Discounting constant functions is proven if (ii) holds. We have shown that the Bellman operator is a contraction under the stated assumptions, which are almost entirely assumptions about the primitives of the model (except for the existence of ϕ). For bounded problems, we can choose $\phi(x) = 1$; for the unbounded versions of the growth model, we can pick

$$\phi(k) = 1 + u(f(k) + (1-\delta)k).$$

We need only make one final step to show that there exist optimal paths that solve the sequence problem. To this end, note that the value function must satisfy

$$v(x) \geq \sum_{t=0}^{\tau} \beta^t r(x_t, a_t) + \beta^\tau v(x_{\tau+1}) \tag{4}$$

for any $\tau \geq 0$ and any feasible sequence of actions. We need to show that the size of the last term goes to zero as $\tau \to \infty$. We know that

$$|v(x_{\tau+1})| = \frac{|v(x_{\tau+1})|}{\phi(x_{\tau+1})} \frac{\phi(x_{\tau+1})}{\phi(x_\tau)} \cdots \frac{\phi(x_1)}{\phi(x)} \phi(x) \tag{5}$$

since ϕ is always positive. Furthermore, we have

$$\frac{|v(x_{\tau+1})|}{\phi(x_{\tau+1})} \leq \|v\|_\phi < \infty \tag{6}$$

because v is ϕ-bounded. We also have, from our condition, that

$$\frac{\phi(x_{s+1})}{\phi(x_s)} = \frac{\phi(t(x_s, a_s))}{\phi(x_s)} \leq \frac{\theta}{\beta}. \tag{7}$$

Thus, we have

$$|v(x_{\tau+1})| \leq \left(\frac{\theta}{\beta}\right)^\tau \|v\|_\phi \phi(x) \tag{8}$$

or

$$\beta^\tau |v(x_{\tau+1})| \leq \theta^\tau \|v\|_\phi \phi(x). \tag{9}$$

The right hand side converges to zero because $\theta < 1$; therefore we have

$$\beta^\tau |v(x_{\tau+1})| \to 0. \tag{10}$$

Now we will examine how to handle the logarithmic case. It is straightforward to prove that the operator T remains a contraction; we need only worry about the fact that $\log(c)$ is unbounded

both above and below on the interval $[0, \infty)$. Therefore, we will modify our proof by considering the space of functions Ξ that satisfy the following condition:

$$f \in \Xi \Rightarrow f(x) = f\left(\frac{x}{\|x\|}\right) + \frac{\log(\|x\|)}{1-\beta}. \tag{11}$$

For any $a \in \mathcal{R}$ define the constant function

$$a(x) = a + \frac{\log(\|x\|)}{1-\beta}. \tag{12}$$

We can define addition and multiplication by scalars as

$$(f+g)(x) = f\left(\frac{x}{\|x\|}\right) + g\left(\frac{x}{\|x\|}\right) + \frac{\log(\|x\|)}{1-\beta} \tag{13}$$

and

$$\alpha f(x) = \alpha f\left(\frac{x}{\|x\|}\right) + \frac{\log(\|x\|)}{1-\beta}. \tag{14}$$

We have then that

$$\|f\| = \sup_{\|x\|=1, x \in X} \{|f(x)|\} \tag{15}$$

defines a norm on this space, where X is a cone (a cone is a set X such that if $x \in X$ then $\alpha x \in X$ $\forall \alpha \geq 0$). We will let Ξ_β be the space of continuous functions bounded in this norm, much like we defined Ξ_ϕ before. This space is complete. We state our new condition:

Condition 5 *Let the following restrictions hold:*

1. $X \subset \mathcal{R}^l$ *is a cone;*

2. *the feasible actions correspondence* Γ *is nonempty, compact-valued, and continuous and the graph of* Γ *is a cone;*

3. $\beta \in (0,1)$ *and there exists a continuous and homogeneous of degree one selection* g *from* Γ *and numbers* $0 < \zeta < \alpha < \infty$ *such that*

$$\|g(x)\| \geq \zeta \|x\| \text{ for all } x \in X$$

$$\|a\| \leq \alpha \|x\| \text{ for all } a \in \Gamma(x) \text{ and all } x \in X;$$

4. *$r : A\backslash\{0\} \to \mathcal{R}$ has the property that $r(x,a) = \ln(m(x,a))$ where $m : A\backslash\{0\} \to \mathcal{R}$ is continuous and homogeneous of degree one; $r(0,0) = -\infty$; and there exist $0 < b < B < \infty$ such that*

$$b\|x\| \leq m(x, g(x)) \text{ for all } x \in X$$

$$m(x,a) \leq B(\|x\| + \|a\|) \text{ for all } a \in \Gamma(x) \text{ and all } x \in X.$$

Basically, what we have assumed is that there are feasible paths which remain away from the origin and which do not converge to infinity. Also, we have assumed that m is bounded above and bounded away from the origin as well. We can obtain the following theorem from Alvarez and Stokey (1998):

Theorem 6 *Let (X, Γ, β, r) satisfy the condition. Then the Bellman operator is a contraction of modulus β and the supremum function v is the unique fixed point of T.*

The conditions we impose here are not very restrictive – we have only required that the consumer be able to consume positive amounts in every period and thus that utility does not converge to negative infinity for at least some feasible path (so that the optimal path stays away from zero), in addition to the usual requirement that returns cannot be infinite either now or at any point in the future. Other approaches are needed to deal with the more general case

$$u(c) = \frac{c^{1-\sigma}}{1-\sigma}$$

for $\sigma \neq 1$; these can be found in Alvarez and Stokey (1998).

1.2 Why Does Good Computation Matter?

We have seen that the value function and the policy function exist under some fairly mild conditions, and we can prove in the growth model some of their properties (monotonicity, concavity, differentiability). However, that the iterations we implement on the computer will not necessarily preserve the properties we need for this result – a perfectly-good model could fail on the computer if you aren't using the right tools. For example, in the risk-sharing case, we might "miss" the best fixed point if we do not accurately compute the value functions at each step, perhaps because we do not accurately compute the optimal decisions or because we inaccurately evaluate the V

functions given those choices; in that case, we could erroneously converge to autarky even from the appropriate initial condition (as mentioned above, this problem arose in Athreya, Tam, and Young 2012 and it took us many years to figure out that the problem was how we interpolated the pricing functions).

Furthermore, even if the model looks to be solved, poor methods can lead to misleading implications. There are many cases of high-profile papers where the solution method led to misleading answers. First, Tesar (1995) reports that international risk sharing welfare gains can be negative, which in her experiment (moving from incomplete to complete markets) would violate the First Welfare Theorem; Kim and Kim (2003) show that linearization is the problem – it leads to large approximation errors and ignores risk effects, which generate large errors in welfare calculations (van Wincoop 1991 also raised this possibility). Second, Aguiar and Gopinath (2006) solve a model of sovereign default and find that the model correctly predicts the observed high volatility of country interest rate spreads; Hatchondo, Martinez, and Sapriza (2010) show that this volatility is entirely the result of excessive space between grid points in a discrete environment, and a continuous version delivers essentially constant spreads. Third, Chang and Kim (2007) find that an incomplete-market business cycle model with heterogeneity in labor efficiency can produce a "labor wedge", a time-varying gap between the marginal product of labor and the marginal rate of substitution for a representative agent (who does not exist), and that this aggregation error accounts for the vast majority of the observed variation in this wedge (Hall 1997; Chari, Kehoe, and McGrattan 2008); Takahashi (2014) shows that their result is mainly dependent on their failure to properly account for a discontinuity caused by a borrowing constraint and their failure to clear the labor market, leading to a 66 percent decrease in the volatility of this aggregation error term. Fourth, Shimer (2005) identifies a quantitative failure of the basic Diamond-Mortensen-Pissarides search model: the implied volatility of unemployment and vacancies is far too small compared to movements in productivity. This result was so influential it became known as the Shimer puzzle. Petrosky-Nadeau and Zhang (2018) show that this implication disappears if one solves the model globally, rather than locally around a steady state, due to the appearance of a long tail in the distribution of unemployment.

1.3 Comments on Programming

I want to stress from the beginning that you should not be writing code to do operations like optimization or root-finding, especially in more than one dimension – people whose entire job is to write code to solve math problems (people like Andre Tits, Klaus Schittkowski, Richard Brent, and Michael Powell, whose algorithms we will encounter later) are going to do a much better job of it. Your job as an applied mathematician is to figure out what code to use and how to sequence it to solve your particular problem (your job as an economist is to figure out what problem you want to solve; once you write it down, you become a mathematician temporarily). You should not use "economic intuition" to solve your model; if good code gives you an unexpected answer, then either you have made a typo or your intuition is faulty (often both!).

I also want to stress that you should be learning a "fast" language that can be efficiently parallelized, like C or Fortran, and not other languages that are slower (like Matlab). Almost all example code will be written in Fortran, although I have some examples in Matlab as well, and you can compare the two (the Fortran code will typically be longer but the execution time will be a lot shorter). The price you pay is that you need to manage memory yourself, which is a useful skill anyway, and forces you to be careful about the size of variables and their type (integer, real, logical, character, etc); you will get unexpected answers if you think a number is real but you defined it to be an integer, for example, because the integer will get rounded down (which is a big problem if it gets rounded down to zero). That said, there is nothing in these notes that cannot be done in both languages, although you may want to take a nap while you wait if you're writing in Matlab and the programming in Matlab will be considerably less intuitive (everything must be vectorized, as loops take forever in Matlab).

Julia and Python are new, developing languages that impose considerably less burden on the programmer (or at least that is what they claim), but at the moment are not yet competitive in terms of speed with Fortran or C (Julia is much better than Python); furthermore, the libraries are open-source and therefore often full of errors and bugs. Use them at your discretion, and check the answers. While many Fortran libraries are also open-source, they are less prone to bugs because (i) the language has been around a lot longer and (ii) centralized repositories are available in which people have verified the results.

1.4 Free Computational Resources

Many of the solvers we will discuss are available for online use through the NEOS server at https://neos-server.org/neos/. To use NEOS you upload code that computes the value of your function and any constraints, select a solver, and come back later. The code is then distributed over a wide network of idle computer resources connected by HTCondor, a resource for distributing jobs on a massive scale. If you can write your problem as a single optimization problem then NEOS is a fantastic free resource for getting solutions to large problems; I have solved nonlinear optimization problems with over $25,000$ variables. NEOS requires you to submit your problem in one of two formats – GAMS (General Algebraic Modeling System) or AMPL (A Mathematical Programming Language). Both are high-level programming environments (black boxes with many canned routines).

1.5 Basics of Parallel Computing

Most computers today have multiple processors, which enables us to compute many tasks "in parallel"; that is, they are done simultaneously on different processors. Networks can tie multiple computers together, meaning we can create "clusters" or "supercomputers" that use many processors simultaneously. It is remarkably easy to access these resources, once you know some basic commands, so this section is dedicated to discussing these simple instructions.

There are two main kinds of parallel computing structures – OpenMP (Open Multi-Processing) and MPI (Message Passing Interface). OpenMP is a shared memory programming model, in which each processor has a local cache of memory but the main memory block is accessed directly by all processors. This sharing can create a data race problem, in which processors have to line up in order to read the memory locations they need to execute commands, and the scalability of the setup is therefore limited (with many processors, the queue to read a location may be rather long). MPI is a distributed memory programming model, in which each "slave" processor has a copy of the memory block and sends "messages" to a master processor to manage the interactions, which allows for larger systems.

OpenMP is easier to program: you can take a particular loop and instruct the program to distribute the elements over multiple processors (in Matlab, this instruction involves replacing your regular for loop with a parfor loop, but you don't need to do anything else; in Fortran,

you just invoke the OpenMP instruction for the compiler and specify a threshold size for loops that can be parallelized). Suppose you are doing dynamic programming; you can execute the maximization at each point in the state space in parallel, with the program sending a new point to a processor once it finishes its current point until we get all of them completed.

MPI is more complicated, since you need to tell the computer exactly what parts of the program are to be sent to each processor and how they are permitted to communicate. Here are some basic MPI commands:

1. MPI_BCAST sends the same message to all the slave processors;
2. MPI_SCATTER sends chunks of an array to different slave processors;
3. MPI_GATHER retrieves elements from many processors and sends them to the master processor, with "rank" specified by the MPI_RANK command (MPI_RANK gives a unique number/name to each process); this command is used to fill up a single array using the calculations from many processes;
4. MPI_ALLGATHER retrieves elements from all slaves and then sends them back the single process constructed from all of them; it is effectively a combination of MPI_GATHER and MPI_BCAST;
5. MPI_BARRIER blocks communication until all processors have reached a particular step (finished their calculations); this command is important if the communication by one thread may alter information used by other threads.

For example, suppose you are doing dynamic programming in parallel over a cluster. Divide up the current state across many processors using MPI_BCAST and MPI_SCATTER – you send the continuation value function via BCAST (since every process in principle needs the entire function) and the chunk of the state space to handle by MPI_SCATTER. Then each process returns the new value function and the policy function by MPI_GATHER. More sophisticated implementations may be possible in which you do not send the entire continuation function, but rather only those parts that are "likely" to be needed (for example, you don't need to send pieces that cannot be reached from a given state).

For programs you run entirely on one computer, the simplest option is to use OpenMP; the overhead cost is quite limited and implementation is simple. For programs you run over clusters or workstations with many processors, MPI is going to be your only option. However, you can nest them – use MPI to spread work over different computers and OpenMP within a computer to spread work across processors.

Note that you can actually make a program run *slower* by a poor parallel implementation; you introduce the cost of passing information back and forth (called the **overhead**) but, because each instance has to wait in the queue to get a particular piece of information, the overall execution time increases. Carefully think about what each subproblem needs to know, how much memory you need, and when messages are safe to pass before pushing ahead into parallel computing.

How much improvement can we realistically expect? The **latency** of a program is defined as the length of time required for it to execute. The improvement in latency from parallelization is governed by Amdahl's law: if a program requires n hours of computing time to complete and $m \leq n$ hours of that program cannot be implemented in parallel, then the maximum improvement in speed is given by

$$S_{latency} = \frac{1}{1 - \frac{m}{n} + \frac{m}{ns}} \leq \frac{1}{1 - \frac{m}{n}},$$

where s is the number of threads. Amdahl's law assumes the problem size remains fixed as we increase the resources dedicated to it; Teller's law (quoted at the beginning of these notes) recognizes that in practice researchers tend to expand the size of the problem to use up the extra resources instead of making the existing problem faster. That gives rise to Gustafson's law: the speedup S of using N processors for a task with serial portion s is

$$S_{latency} = 1 - p + sp,$$

where p is the fraction of the workload that can benefit from the extra resources. You can estimate m (or p) by tracking how long your serial program spends in different subroutines and noting which ones can be done in parallel and which ones cannot. If your code is inherently sequential (such as simulating the path of a dynamic system), or even mostly sequential (p is close to zero), then there is no point introducing the overhead needed for parallel computation. All you are doing is slowing things down. However, if your code consists of several disconnected calculations (say, solving an economic model for different sets of parameters), then the speed up

will be almost linear in the number of processors ($\frac{m}{n} = p \approx 1$).

1.6 Sparse Storage

In large problems, memory can become a constraint; as a result, computer scientists have developed "sparse" operations that reduce storage. A matrix B $(n \times m)$ is **sparse** if it contains "mostly" zeroes; to save on storage we can only keep track of which elements are nonzero and what value they take. There are two approaches to constructing sparse storage that minimizes the amount of information we need to keep in memory.

Compressed row storage (CRS) makes a $k \times 1$ vector of the nonzero elements A, a $n+1 \times 1$ vector of integers I_A that start from 0 and index the cumulative number of nonzero elements that occur in the first l rows, and a $n \times 1$ vector of integers I_B that index which column the particular element of A resides in

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 5 & 8 & 3 & 6 \end{bmatrix}$$

$$I_A = \begin{bmatrix} 0 & 0 & 2 & 3 & 4 \end{bmatrix}$$

$$I_B = \begin{bmatrix} 0 & 1 & 2 & 1 \end{bmatrix}$$

Compressed column storage (CCS) switches I_A and I_B:

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 5 & 8 & 3 & 6 \end{bmatrix}$$

$$I_A = \begin{bmatrix} 0 & 1 & 3 & 4 & 4 \end{bmatrix}$$

$$I_B = \begin{bmatrix} 1 & 1 & 2 & 3 \end{bmatrix}$$

CCS is more efficient for operations than CRS. Be sure that you are using the right storage protocol for your solver. In Fortran the Intel MK Library comes with a sparse solver, and sparse matrix operations are built into Matlab. Note also that some protocols start with $I_A(1,1) = 0$ (called zero syntax) and others with $I_A(1,1) = 1$ (called one or Fortran syntax).

Sparse matrix multiplication is straightforward. To remind ourselves, the matrix product AB is given by a matrix C such that

$$C_{ij} = \sum_k A_{ik} B_{kj}.$$

Thus, to compute C we can simply move through each entry (i, j) and increment the sum over k. If A is $n \times l$ and B is $l \times m$, then this operation is $O(nlm)$. With sparse matrices, we need only increment C_{ij} if both A_{ik} and B_{kj} are nonzero, and since we know precisely which elements are nonzero we can use this fact to eliminate some operations. The more sparse A or B are, the fewer actual computations we need.

2 Complexity Theory

Since I occasionally will use these terms later, I introduce some terms from complexity theory, the study of whether problems are solvable in some "reasonable" amount of time. They are helpful at identifying difficult problems that it might be better to avoid from the outset, rather than frustrating yourself by attempting to solve them. Note that these problems may well have solutions; in fact, the worst problems are ones that must have solutions, but they may be impossible to find quickly enough to be practical. Note also that these classifications apply to "large" problems; they are concerned with how the cost of solving the problem increases with its size. If you only care about small problems, then none of this really matters.

Definition 7 *A **Turing machine** is a seven-tuple $M = \{Q, \Gamma, b, \Sigma, \delta, q_0, F\}$ where Q is a finite non-empty set of states; Γ is a finite non-empty set of tape alphabet symbols; $b \in \Gamma$ is the blank symbol (the only symbol permitted to appear infinitely often at any step of the computation); $\Sigma \subseteq \Gamma \backslash \{b\}$ is the set of input symbols (the set of symbols allowed to appear on the initial tape contents); $q_0 \in Q$ is the initial state; $F \subseteq Q$ is the set of final or accepting states (the initial tape contents is accepted by M if it eventually halts in a state from F); and $\delta : (Q \backslash F) \times \Gamma \to Q \times \Gamma \times \{L, R\}$*

is the state transition function with L being the action "left shift" and R being the action "right shift".

Definition 8 *A problem L is in set P if there exists a deterministic Turing machine M such that (i) M runs in polynomial time ($T(n) = O\left(n^k\right)$ for $k > 0$), (ii) $\forall x \in L\ M(L) = 1$, and (iii) $\forall x \notin L\ M(L) = 0$.*

A Turing machine is an algorithm (it is literally a tape that shifts left and right according to the state transition function and, in each state, produces a symbol/takes an action); a deterministic Turning machine implies only one action is to be taken for any given situation, whereas a nondeterministic Turing machine allows for several possible actions to be chosen, picked at random. $T(n)$ denotes the time required to solve a problem of size n (the size of a problem is the number of bits required to describe the inputs). A Turing machine runs in polynomial time if $T(n)$ rises no faster than n^k for some finite integer k.

Definition 9 *A problem L is in set NP if there exist polynomials p and q and a deterministic Turing machine M such that (i) $\forall x, y$ the machine runs in polynomial time $p(|x|)$, (ii) $\forall x \in L$ there exists string y of length $q(|x|)$ such that $M(x, y) = 1$, and (iii) $\forall x \notin L$ and all strings y of length $q(|x|)\ M(x, y) = 0$.*

Problems in P can be **solved** in polynomial time, whereas potential solutions to problems in NP can be **verified** in polynomial time; NP is not the complement of P, so it does not mean "not in P", but rather "non-polynomial". It should be clear that $P \subset NP$, since the process of finding a solution also verifies that solution; one of the Millenium Problems that remains unsolved is whether $P = NP$, although it is generally believed not to be true.

Definition 10 *A problem H is NP-**hard** if $\forall L \in NP$ there exists a polynomial time many-one reduction from L to H.*

Essentially, H is NP-hard if it can be broken down into a sequence of one-step problems that can be used to solve any $L \in NP$. Therefore, in the worst-case scenario H is "as hard" as the hardest problems in NP; remember that "hard" here means slow, not difficult. Note that H does not have to be an element of NP. One classic example of an NP-hard problem is the traveling

salesman problem, in which we look for a shortest cyclic path through a given set of nodes on a graph. Following Danzig, Fulkerson, and Johnson, the traveling salesman problem can be written as the integer linear program

$$\min_x \left\{ \sum_{i=1}^{n} \sum_{j \neq i, j=1}^{n} c_{ij} x_{ij} \right\}$$

subject to

$$\begin{aligned} x_{ij} &\in \{0,1\} \quad i,j \in \{1,...,n\} \\ \sum_{i \neq j, i=1}^{n} x_{ij} &= 1 \quad j \in \{1,...,n\} \\ \sum_{j \neq i, j=1}^{n} x_{ij} &= 1 \quad i \in \{1,...,n\} \\ \sum_{i \in Q} \sum_{j \neq i, j \in Q} x_{ij} &\leq |Q| - 1 \quad \forall Q \subsetneq \{1,...,n\}, |Q| \geq 2, \end{aligned}$$

where $c_{ij} > 0$ is the cost of traveling from i to j and $x_{ij} = 1$ denotes that the salesman travels from i to j. The last constraint, which looks rather formidable, simply ensures that the solution is a single tour and not the union of multiple sub-tours (the traveling salesman is not allowed to revisit cities). As n increases, the cost of solving this problem increases rapidly due to the proliferation of terms in both the objective and the constraints.

Another example is the quadratic programming problem

$$\min_x \left\{ \frac{1}{2} x^T Q x + c^T x \right\}$$

subject to

$$Ax \leq b.$$

If Q is positive definite (so that the objective is strictly concave), the ellipsoid method can solve this problem in polynomial time; the ellipsoid method creates a sequence of ellipses with uniformly-shrinking volumes, in the end enclosing the minimizer in a smaller and smaller area. If Q is not positive definite (meaning it has at least one nonpositive eigenvalue), then this problem is NP-hard; the objective function is not strictly concave and might not even be concave, which creates problems just characterizing minima. As a result, finding the minimum will be difficult for large problems, and the burden will increase rapidly with the size.

Definition 11 *An NP-hard problem H is NP-**complete** if $H \in NP$.*

If we assume that the edge lengths in the traveling salesman problem are integers, then the problem is NP-complete. The quadratic program is NP-complete if Q is positive definite. There are problems that are NP-hard but not NP-complete. For example, the halting problem – determining whether a given program will ever terminate – is undecideable and therefore cannot be NP-complete.

Note that P does not mean "easy", as some $L \in P$ have extremely long runtimes or are extremely sensitive to numerical error; that is, the algorithm exists but for practical purposes it is useless. Similarly, $L \notin P$ does not mean "hard", as many such problems have known heuristic solutions that work quickly to achieve a "good enough" solution. However, we definitely prefer our problems to be in P, since we know at the minimum that some algorithm exists to get us a solution with a runtime that does not grow too quickly; in general knowing that a problem is in NP is not that useful, since we rarely have a potential solution to verify.

Figure 2 shows the relationship between P, NP, NP-complete, and NP-hard, under the assumption that $P \neq NP$; as noted already, this condition has not been proven, but it is generally believed to be true.

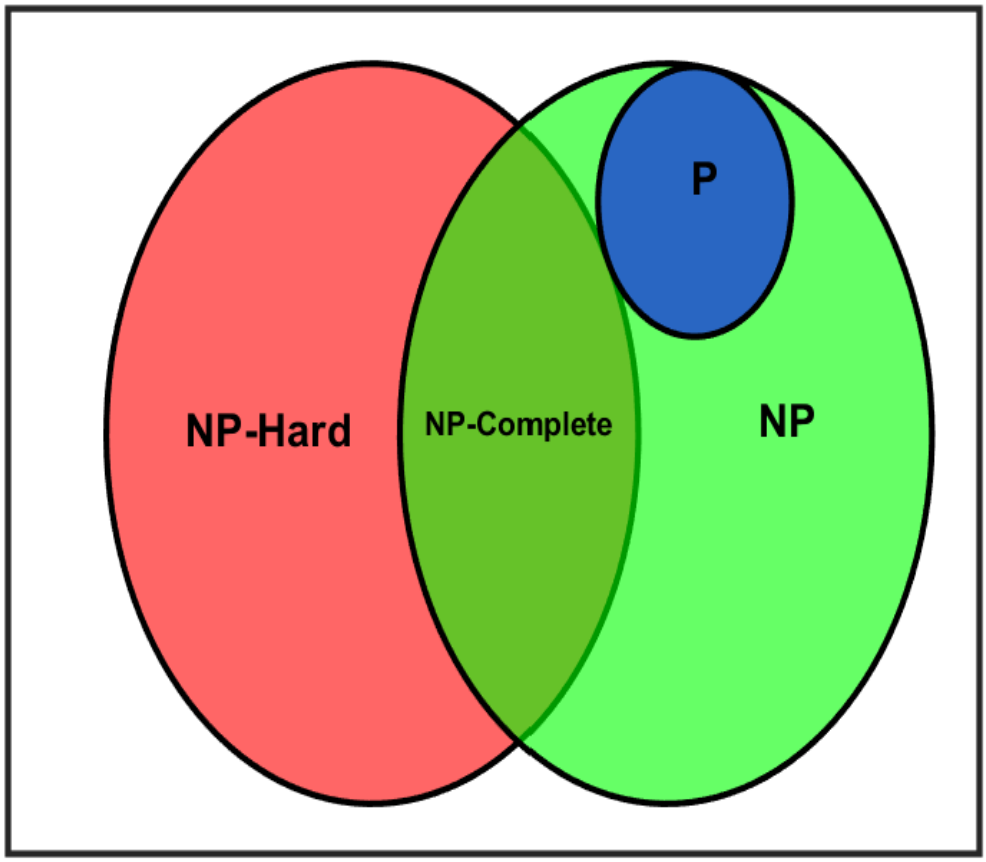
A surprisingly difficult problem is the computation of a mixed-strategy Nash equilibrium. We know that, for any finite game, there exists at least one mixed-strategy Nash equilibrium, which is a simple application of Kakutani's fixed-point theorem. For 2-player 2-option games, particularly those that are zero-sum like the chicken game

	Swerve	Straight
Swerve	$(0, 0)$	$(-1, 1)$
Straight	$(1, -1)$	$(-2, -2)$

,

the problem of locating the Nash equilibrium is easy. Since no strategy is weakly-dominated, we can simply set up the indifference conditions for each player. Suppose player 2 plays Swerve with probability q and Straight with probability $1 - q$; then payoffs to player 1 are then

$$\begin{aligned} \text{Swerve:} \quad & q * 0 + (1 - q) * (-1) \\ \text{Straight:} \quad & q * 1 + (1 - q) * (-2) \, . \end{aligned}$$


P
NP-Hard
NP-Complete
NP

Similarly, if player 1 plays Swerve with probability p, the payoffs to player 2 are

$$\begin{aligned} \text{Swerve:} \quad & p * 0 + (1-p) * (-1) \\ \text{Straight:} \quad & p * (1) + (1-p) * (-2) . \end{aligned}$$

For each player to actually mix their strategies, they must be indifferent, so we have the two linear equations

$$\begin{aligned} q * 0 + (1-q) * (-1) &= q * 1 + (1-q) * (-2) \\ p * 0 + (1-p) * (-1) &= p * 1 + (1-p) * (-2) \end{aligned}$$

which yield the mixed strategy Nash equilibrium

$$p^* = q^* = \frac{1}{2}.$$

Extending this setup to more than two players is not difficult.

What gets difficult is when there are many strategies, even if there are still only 2 players; the problem is that some strategies may be dominated, meaning that they will not be played with positive probability. If we take the payoffs to the players as matrices A and B, then a mixed strategy Nash equilibrium (p^*, q^*) must satisfy the inequalities

$$\begin{aligned} (p^*)^T A q^* &\geq p^T A q^* \\ (p^*)^T B q^* &\geq (p^*)^T B q \end{aligned}$$

for all alternatives (p, q), along with the conditions that both p and q sum to one. It seems likely, although not for certain known, that this problem does not live in P; it is known that finding Nash equilibria is $PPAD$- complete (see Chen and Deng 2006), which is a set of problems that are "not quite" as hard as NP-complete problems because we know a solution exists, even if we cannot find it in polynomial time. Formally, $PPAD$-complete problems have the property that a binary relation $\mathcal{P}(x, y)$ has a polynomial time algorithm that can determine whether $\mathcal{P}(x, y)$ holds for fixed x and y, given that we know for each x there exists a y such that $\mathcal{P}(x, y)$ is true (that is, we know there exists some y such that $\mathcal{P}(x, y)$ is true for each x, but we do not know whether the y we have is one that satisfies $\mathcal{P}(x, y)$ given x); it is currently unknown whether $PPAD = P$, but it is generally believed not to be true. Interestingly, the question of whether a

second Nash equilibrium exists is NP-complete, since we do not know whether it is true or not in a generic game, which implies that the problem of finding all Nash equilibria is also NP-complete.

You now have a language for expressing when problems should be expected to be impractical to "scale up". Again, remember that problems in P may well be costly to solve for a given n; whether a problem is in P or not is about how that time increases with n. For many tasks the question will be irrelevant because we are interested only in small values of n; for others, we want to study high n problems and those may or may not be practical. We now move to discussion of the main numerical problems encountered in economics – numerical differentiation and integration, optimization, root-finding, and functional approximation. Note that these problems are not independent; functional approximation will use tools from optimization and numerical integration, for example.

3 Differentiation

Our first task is to compute the derivative of a function that we have no way to analytically compute; for example, if the function can only be represented implicitly on the computer. Numerical differentiation computes an approximation to the derivative. While I'm sure you know the following definition, it will be helpful to write it out explicitly.

Definition 12 *The derivative Df of a function f is defined by*

$$Df(x) = \lim_{h \to 0} \left\{ \frac{f(x+h) - f(x)}{h} \right\}.$$

Equivalently, Df is implicitly defined by

$$\lim_{h \to 0} \left\{ \frac{f(x+h) - f(x) - Df(x)\, h}{h} \right\} = 0.$$

Note that D maps differentiable functions to functions (which are not necessarily differentiable). To see an example, consider the function

$$f(x) = \max\{0, x\}^2.$$

The derivative exists everywhere:

$$Df(x) = 2\max\{0, x\}.$$

But the derivative of $Df(x)$ does not exist at $x = 0$, as the left and right limits are not equal:

$$\lim_{x \uparrow 0} \left\{ D^2 f(x) \right\} = 0$$

$$\lim_{x \downarrow 0} \left\{ D^2 f(x) \right\} = 2.$$

Therefore, $Df(x)$ is not differentiable (we say that $f \in C^1$, which is that f has one continuous derivative).

3.1 Finite-Difference Methods

We could simply use the definition of the derivative to compute a numerical one:

$$Df(x) = \frac{f(x+h) - f(x)}{h}$$

for some small h. In general this calculation is inaccurate if h is too small; subtracting two terms that are nearly equal, then dividing them by a number close to zero, can lead to severe inaccuracies as the computer does not like anything close to $\frac{0}{0}$. On the other hand, h cannot be too big or the derivative calculation will not be close to the true value – essentially, we are trying to replace the tangent line with a secant line, and the displacement cannot be too large or it will not be accurate. In theory a "two-sided" approach is better. Note that the definition of derivative could equivalently be written

$$Df(x) = \lim_{h \to 0} \left\{ \frac{f(x+h) - f(x-h)}{2h} \right\}.$$

Mathematically, this expression is called the **symmetric derivative** of f, but the two objects are equal for differentiable functions; the symmetric derivative exists for more functions, like $f(x) = |x|$, which has a symmetric derivative at 0:

$$D(|0|) = \frac{|h| - |-h|}{2h} = 0.$$

Our approximate derivative is given by

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

The choice of h is optimally given by

$$h^* = \left(\frac{3\epsilon}{M_3} \right)^{1/3}$$

where ϵ is machine tolerance and M_3 is the upper bound on $\left|D^3 f\right|$ near x; by the Mean Value Theorem for this h we get the derivative exactly right up to the computer's ability to distinguish numbers. In practice, it is usually impossible to compute this term (since if we knew the third derivative we probably wouldn't need to approximate the first derivative), so we use something like $h = 10^{-6}x$ for first derivatives. The "two-sided" derivative is the first in a sequence of expressions called **stencils**; the next two stencils for first derivatives are

$$Df(x) = \frac{-f(x+2h) + 8f(x+h) - 8f(x-h) + f(x-2h)}{12h}$$
$$Df(x) = \frac{f(x+3h) - 9f(x+2h) + 45f(x+h) - 45f(x-h) + 9f(x-2h) - f(x-3h)}{60h}.$$

A higher-order stencil leads to a more accurate derivative at the cost of more function evaluations; note that we can do higher-order stencils for "one-sided" derivatives as well.

Higher order derivatives can be computed similarly:

$$D^2 f(x) = \frac{f(x+2h) - 2f(x) + f(x-2h)}{4h^2}$$

where we set $h = 10^{-3}x$. Keep in mind that, if evaluation is expensive, the two-sided case requires two additional evaluations (beyond x) while the one-sided only needs one, so if the extra accuracy is not needed you should not bother. The next two stencils for the second derivative are

$$D^2 f(x) = \frac{-f(x+2h) + 16f(x+h) - 30f(x) + 16f(x-h) - f(x-2h)}{12h^2}$$
$$D^2 f(x) = \frac{2f(x-3h) - 27f(x-2h) + 270f(x-h) - 490f(x) + 270f(x+h) - 27f(x+2h) + 2f(x+3h)}{180h^2}$$

In general you should be careful with derivatives above second-order – to get them accurately requires a stencil with relatively large n, and you can see that the coefficients start to get pretty large and alternate in sign so small inaccuracies in evaluating f could easily degrade the quality of your derivative estimate. However, if you are computing the first two derivatives, note that the "second order" stencils are essentially free – you need the $f(x \pm 2h)$ terms for the second derivative and $f(x \pm h)$ for the first derivative, so you might as well go ahead and use them all to get a better approximation.

For multivariate functions, we compute partial derivatives in a similar way:

$$D_i f(x) = \frac{f\left(x + h_i e^i\right) - f\left(x - h_i e^i\right)}{2h_i}$$
$$D^2_{ij} f(x) = \frac{f\left(x + h_i e^i + h_j e^j\right) - f\left(x + h_i e^i - h_j e^j\right) - f\left(x - h_i e^i + h_j e^j\right) + f\left(x - h_i e^i - h_j e^j\right)}{4h_i h_j}.$$

Note that the Hessian may not be symmetric as a result of the approximation error, but we can recover symmetry by adding our finite difference approximation to its transpose and dividing by 2; that is, for any matrix A the matrix

$$\frac{1}{2}\left(A + A^T\right)$$

is symmetric. The higher-order stencils are

$$D_i f(x) = \frac{-f\left(x + 2h_i e^i\right) + 8f\left(x + h_i e^i\right) - 8f\left(x - h_i e^i\right) + f\left(x - 2h_i e^i\right)}{12h_i}$$
$$D^2 f(x) = \frac{\begin{array}{c} -63\left(f_{1,-2} + f_{2,-1} + f_{-2,1} + f_{-1,2}\right) + 63\left(f_{-1,-2} + f_{-2,-1} + f_{1,2} + f_{2,1}\right) - \\ 44\left(f_{2,-2} + f_{-2,2} - f_{-2,-2} - f_{2,2}\right) + 74\left(f_{-1,-1} + f_{1,1} - f_{1,-1} - f_{-1,1}\right) \end{array}}{600h_1 h_2}$$

where

$$f_{i,j} = f\left(x + ih_1 e^1 + jh_2 e^2\right)$$

in the 2-d case (the n-d case is similar but notationally tedious).

Note that for some applications we don't need the derivative itself, but instead the derivative times a vector p. We can approximate this product by

$$Df(x)\, p \approx \frac{f(x + \epsilon p) - f(x - \epsilon p)}{2\epsilon}.$$

The cost saving can be significant since we avoid the matrix multiplication.

3.2 Interpolatory Rules

Suppose we know the value of the function at more than one point. We can exploit this information to compute derivatives with higher accuracy. Assume first that we have

$$f(x_1), f(x_2), f(x_3)$$

and want to compute

$$Df(x_1).$$

By Taylor's theorem we have

$$f(x_i) = f(x_1) + Df(x_1)(x_i - x_1) + \frac{1}{2}D^2 f(x_1)(x_i - x_1)^2 + \frac{1}{6}D^3 f(\xi)(x_i - x_1)^3.$$

We want to find constants (a, b, c) such that

$$af(x_1) + bf(x_2) + cf(x_3) = Df(x_1) + o\left((x_1 - x_2)^2 + (x_1 - x_3)^2\right).$$

Using the Taylor expansions at x_2 and x_3 we find

$$Df(x_1) = af(x_1) + b\left(f(x_1) + Df(x_1)(x_2 - x_1) + \frac{1}{2}D^2 f(x_1)(x_2 - x_1)^2\right) +$$
$$c\left(f(x_1) + Df(x_1)(x_3 - x_1) + \frac{1}{2}D^2 f(x_1)(x_3 - x_1)^2\right)$$

so that

$$\begin{aligned} a + b + c &= 0 \\ b(x_2 - x_1) + c(x_3 - x_1) &= 1 \\ b(x_2 - x_1)^2 + c(x_3 - x_1)^2 &= 0 \end{aligned}$$

which has solution

$$\begin{aligned} a &= \frac{x_3 - 2x_1 + x_2}{(x_3 - x_1)(x_1 - x_2)} \\ b &= \frac{x_1 - x_3}{(x_1 - x_2)(x_3 - x_2)} \\ c &= \frac{x_1 - x_2}{(x_2 - x_3)(x_1 - x_3)}. \end{aligned}$$

Therefore,

$$Df(x_1) = \frac{x_3 - 2x_1 + x_2}{(x_3 - x_1)(x_1 - x_2)} f(x_1) + \frac{x_1 - x_3}{(x_1 - x_2)(x_3 - x_2)} f(x_2) + \frac{x_1 - x_2}{(x_2 - x_3)(x_1 - x_3)} f(x_3).$$

One could extend this idea to more than 3 points or to higher-order derivatives. In general this procedure works best if $x_1 \in [x_2, x_3]$. If $x_1 \in [x_2, x_3]$ and $x_1 - x_2 = x_3 - x_1 = h$, then the interpolatory rule reduces to the two-sided (central) finite difference; the higher-order stencils are the symmetric versions of the interpolatory rules with more than 3 points.

If we can represent our function using some finite collection of basis functions (such as polynomials or splines), derivative calculations are straightforward. However, we then need the value at many points in order to construct the approximation. We'll discuss these issues later when we come to functional approximation.

3.3 Automatic Differentiation

Automatic differentiation is an efficient procedure for obtaining the value of a function and its derivatives at a given point. The idea is to break a function down into constituent simple operations and then use the chain rule to calculate the derivative; there can be an efficiency gain from computing common expressions only once. Define the set of **differential numbers** to be the pair (f, f') which we denote df. The rules for operating on differential numbers are just the usual rules for differentiation:

$$
\begin{aligned}
\pm dg &\doteq \left(\pm g, \pm g'\right) \\
dg \pm dh &\doteq \left(g \pm h, g' \pm h'\right) \\
dg * dh &\doteq \left(g * h, g' * h + g * h'\right) \\
dg/dh &\doteq \left(g/h, \frac{g' - \frac{g}{h}h'}{h}\right) \\
dg^n &\doteq \begin{cases} \left(g^{n-1} * g, n * g^{n-1} * g\right) & \text{if } n \geq 1 \\ \left(g^n, n * g^n * g'/g\right) & \text{otherwise} \end{cases} \\
dg^{dh} &\doteq \left(\exp\left(\log\left(g\right) * h\right), \exp\left(\log\left(g\right) * h\right) * \left(h' * \log\left(g\right) + h * g'/g\right)\right) \\
\sqrt{dg} &\doteq \left(\sqrt{g}, \frac{g'}{2\sqrt{g}}\right) \\
\exp\left(dg\right) &\doteq \left(\exp\left(g\right), \exp\left(g\right) * g'\right) \\
\log\left(dg\right) &\doteq \left(\log\left(g\right), g'/g\right) \\
abs\left(dg\right) &\doteq \left(abs\left(g\right), sign\left(g\right) * g\right) \\
\phi\left(dg\right) &\doteq \left(\phi\left(g\right), \phi'\left(g\right) * g'\right).
\end{aligned}
$$

To compute a function and its derivative at a point x_0, we simply calculate the differential number $(x_0, 1)$. For example, take the function

$$
f\left(x\right) = \frac{\left(x-1\right)\left(x+3\right)}{x+2}
$$

with derivative

$$Df(x) = \frac{x^2 + 4x + 7}{(x+2)^2}.$$

Thus, we have

$$f(3) = 2.4$$

$$Df(3) = 1.12.$$

Using differential calculus, we can obtain this result as

$$\begin{aligned}
\left(f, f'\right)(3) &= \frac{((3,1) - 1)((3,1) + 3)}{(3,1) + 2} \\
&= \frac{(2,1) * (6,1)}{(5,1)} \\
&= \frac{(12,8)}{(5,1)} \\
&= \left(2.4, \frac{8 - 2.4 * 1}{5}\right) \\
&= (2.4, 1.12)
\end{aligned}$$

To take a more complicated example, consider the function

$$f(x) = \sin\left(\exp\left(x^2 + \log(x)\right)\right)$$

with derivative

$$Df(x) = \left(\cos\left(x \exp\left(x^2\right)\right)\right) \exp\left(x^2\right)\left(2x^2 + 1\right).$$

We are interested in this function at the point $x = 2$:

$$f(2) = 0.68852$$

$$Df(2) = -356.36$$

Using differential calculus we get

$$\begin{aligned}
\left(f, f'\right)(2) &= \sin\left(\exp\left((2,1) * (2,1) + \log(2,1)\right)\right) \\
&= \sin\left(\exp\left((4,4) + \left(\log(2), \frac{1}{2}\right)\right)\right) \\
&= \sin\left(\exp\left(4 + \log(2), \frac{9}{2}\right)\right) \\
&= \sin\left(\exp\left(4 + \log(2), \frac{9}{2}\exp(4 + \log(2))\right)\right) \\
&= \left(\sin\left(\exp(4 + \log(2))\right), \frac{9}{2}\exp(4 + \log(2))\cos\left(\exp(4 + \log(2))\right)\right) \\
&= (0.68852, -356.36).
\end{aligned}$$

Higher-order derivatives work the same way, but the expressions get more complicated because of chain rules. Efficient implementations break the function into steps and use the chain rule step-by-step. For example,

$$f(x) = \sin\left(\exp\left(x^2 + \log(x)\right)\right)$$

can be broken down into

$$\begin{aligned}
w_1 &= x^2 + \log(x) \\
w_2 &= \exp(w_1) \\
f(w_2) &= \sin(w_2).
\end{aligned}$$

The individual derivatives are easy to compute:

$$\begin{aligned}
\frac{dw_1}{dx} &= 2x + \frac{1}{x} \\
\frac{dw_2}{dx} &= \frac{dw_2}{dw_1}\frac{dw_1}{dx} \\
&= \exp(w_1)\left(2x + \frac{1}{x}\right) \\
\frac{df(x)}{dx} &= \frac{df}{dw_2}\frac{dw_2}{dx} \\
&= \cos(w_2)\exp(w_1)\left(2x + \frac{1}{x}\right) \\
&= \cos\left(\exp\left(x^2 + \log(x)\right)\right)\exp\left(x^2 + \log(x)\right)\left(2x + \frac{1}{x}\right).
\end{aligned}$$

Simplifying this expression we get the same derivative as above.

Some implementations of automatic differentiation actually rewrite the code that evaluates the function to return both the function and the derivative; these tools can handle even functions that are defined piecewise via if-then statements. Other implementations do not provide the code rewriting option, but instead simply return the value of the function and all first and second derivatives at some point (because AD was created by physicists, who rarely require higher-order derivatives, they typically stop at second-order). The algorithm computes these expressions by "overloading" operators; that is, AD redefines a function (such as $\sin(x)$) to return not only the function value but the derivative:

$$\sin(x) = (\sin(x), \cos(x)).$$

It handles the chain rule by specifying the rules for various operations (which are the ones listed above). In Matlab the objects are called "structures" and are accessed using variables defined as

$$x.y;$$

in Fortran we can define a new variable "type" whose constituent parts are accessed via

$$x\%y.$$

These objects are like functions, but associate many things with the same operation; you can create irregularly-sized arrays (called **ragged arrays**) this way (for example, a "matrix" with a different number of columns in each row) without having to pack it with zeros (and then keep track of where the elements are "valid" and where they are just placekeepers). You can also create objects that mix different types (integers, real numbers, logical variables, strings) under one name, provided you are careful about how to manipulate it.

3.4 Complex Step Differentiation

If you are working with a programming language that handles operations on complex numbers, then you can estimate the derivative of a function without facing the problem of subtracting two very similar numbers; this procedure is called **complex step differentiation**. The idea is that

$$Df(x) = \lim_{h \to 0} \left\{ \frac{\mathrm{Im}(f(x + ih))}{h} \right\}$$

which can be approximated by using a small h. In this way we avoid the subtractive cancellation error so that h can be smaller than with the finite-difference method. This approach works for sufficiently smooth functions.

Definition 13 *A real-valued function $f : \mathcal{C} \to \mathcal{R}$ is **real analytic** on an open set D if, $\forall x_0 \in D$,*

$$T(x) = \sum_{n=0}^{\infty} \frac{D^n f(x_0)}{n!} (x - x_0)^n$$

converges pointwise to $f(x)$ in a neighborhood of x_0.

An analytic function is locally equal to its Taylor expansion at any point x_0, and is therefore infinitely-differentiable (smooth). However, there are functions that are smooth but not analytic, in particular those that are defined piecewise. One example is

$$f(x) = \begin{cases} \exp\left(-\frac{1}{x}\right) & x > 0 \\ 0 & x \leq 0 \end{cases}.$$

The derivative of order n is

$$D^n f(x) = \begin{cases} \frac{p_n(x)}{x^{2n}} f(x) & x > 0 \\ 0 & x \leq 0 \end{cases}$$

where $p_n(x)$ is the polynomial defined recursively as

$$p_{n+1}(x) = x^2 D p_n(x) - (2nx - 1) p_n(x).$$

Around 0, the Taylor series converges to the zero function:

$$T(0) = \sum_{n=0}^{\infty} \frac{D^n f(0)}{n!} = 0.$$

Since $T(x) \neq f(x)$ on a neighborhood of 0, the function is not analytic there. There are even more pathological examples of functions that have infinitely-many derivatives yet fail to be analytic anywhere, although it is highly unlikely you will encounter one in practice.

Consider a real analytic function $f(x)$ and a linear Taylor expansion along the imaginary axis:

$$f(x + ih) = f(x) + ihDf(x).$$

Take only the imaginary part and divide both sides by h:

$$Df(x) = \frac{\text{Im}\left(f\left(x + ih\right)\right)}{h}.$$

This approximation is accurate to order $O\left(h^2\right)$, so we can choose a very small h. A classic example of the power of complex-step differentiation is the function

$$f(x) = \frac{\exp(x)}{\sqrt{\sin^3(x) + \cos^3(x)}}.$$

The finite difference method never achieves floating-point accuracy, while the complex step is accurate even for $h = 10^{-15}$; the most accurate finite-difference value is $h = 10^{-5}$, which has an error twice the minimum for the complex-step case.

To obtain the second derivative, we can use a similar formula (the derivation is substantially more involved and therefore omitted; if we followed the logic of the first derivative we would end up with a subtraction, negating the benefit):

$$D^2 f(x) = \frac{\text{Im}\left(f\left(x + \sqrt{i}h\right) + f\left(x + \sqrt[5]{i}h\right)\right)}{h^2}.$$

Again, we avoid any subtractions, which allows us to use a much smaller h. It is not straightforward to extend this apprach to higher derivatives or higher dimensions, so we'll take an alternative and more general approach.

Multicomplex variables, an extension of complex variables to higher dimensions, can be used with overloading to obtain highly-accurate derivative calculations (Lantoine, Russell, and Dargent 2012); for example, a bicomplex number is represented as

$$z = x_0 + x_1 i_1 + x_2 i_2 + x_{12} i_1 i_2$$

where $i_n^2 = -1$ for each n, but $i_1 i_2 \neq -1$.[1] Note that this number can be represented in terms of complex numbers:

$$z = z_1 + z_2 i_2$$

[1]Bicomplex numbers are not the same as quaternions, the usual extension of complex numbers to "higher dimensions"; a quaternion takes the form

$$x = a + bi + cj + dk,$$

whereas a bicomplex number would restrict $k = ij$.

where

$$z_1 = x_0 + x_1 i_1$$

$$z_2 = x_2 + x_3 i_1.$$

Standard complex numbers can be represented as Cauchy-Riemann matrices:

$$a + bi = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}.$$

This representation also works for multicomplex numbers, but the dimension increases. For the biconvex case

$$z = x_0 + x_1 i_1 + x_2 i_2 + x_{12} i_1 i_2$$

the matrix representation is

$$i_1 = \begin{bmatrix} x_0 & -x_1 & -x_2 & x_{12} \\ x_1 & x_0 & -x_{12} & -x_2 \\ x_2 & -x_{12} & x_0 & -x_1 \\ x_{12} & x_2 & x_1 & x_0 \end{bmatrix}.$$

If we take a Taylor expansion, we get

$$\begin{aligned} f\left(x + h\left(i_1 + i_2\right)\right) &= f\left(x\right) + Df\left(x\right)\left(i_1 + i_2\right)h + \frac{1}{2}D^2 f\left(x\right)\left(i_1 + i_2\right)^2 h^2 \\ &= f\left(x\right) + Df\left(x\right)\left(i_1 + i_2\right)h + \frac{1}{2}D^2 f\left(x\right) i_1 i_2 h^2 - D^2 f\left(x\right) h^2. \end{aligned}$$

Rearranging we get

$$D^2 f\left(x\right) = \frac{\mathrm{Im}_{12}\left(f\left(x + h\left(i_1 + i_2\right)\right)\right)}{h^2},$$

a direct generalization of the first derivative, where Im_{jk} means to take the real coefficient in front of the $i_j i_k$ term. Partial derivatives are now straightforward:

$$\frac{\partial^2 f\left(x, y\right)}{\partial x \partial y} = \frac{\mathrm{Im}_{12}\left(f\left(x + i_1 h, y + i_2 h\right)\right)}{h^2}.$$

The matrix representation is useful for obtaining derivatives that involve linear algebra operations (like matrix multiplication).

4 Integration

We now examine numerical integration. In economics, uncertain outcomes are evaluated according to their expected value; expectations are simply the inner product of function values with probability density function values:

$$E\left[u\left(C,X\right)\right]=\int u\left(c,x\right)f\left(x\right)dx.$$

We need a numerical method to compute the integral, since many economic problems involve maximizing the expectation of some function. The essence of numerical integration is to replace the integral with a sum, but to carefully choose the location of the points in order to minimize approximation error at low computational cost.

4.1 Classical Rules

We start with formulae which should be familiar from undergraduate calculus – **Newton-Cotes** formulae. The basic idea here is to divide the domain into tiny pieces which are simple in shape, then just add up the areas. Assume that the domain is spaced according to

$$x_n=x_1+\left(n-1\right)h;$$

that is an evenly-spaced grid with interval size h. The simplest Newton-Cotes formula is the **trapezoidal rule**:

$$\int_{x_1}^{x_2}f\left(x\right)dx=h\left[\frac{f\left(x_1\right)+f\left(x_2\right)}{2}\right]+O\left(h^3D^2f\right).$$

Ignoring the last term gives us an integration formula that is exact up to an order 1 polynomial (linear); the error depends on the curvature of the function (measured by D^2f) and the distance between points (measured by h). We can extend this to **Simpson's rule**:

$$\int_{x_1}^{x_3}f\left(x\right)dx=h\left[\frac{f\left(x_1\right)+4f\left(x_2\right)+f\left(x_3\right)}{3}\right]+O\left(h^5D^4f\right);$$

this formula is exact for order 3 polynomials or smaller. Again, the error depends on the highest-order neglected derivative. Finally, we can use **Boole's rule** (sometimes, due to an old typo, incorrectly referred to as Bode's rule):

$$\int_{x_1}^{x_5}f\left(x\right)dx=h\left[\frac{14}{45}f\left(x_1\right)+\frac{64}{45}f\left(x_2\right)+\frac{24}{45}f\left(x_3\right)+\frac{64}{45}f\left(x_4\right)+\frac{14}{45}f\left(x_5\right)\right]+O\left(h^7D^6f\right)$$

which is exact for order 5 polynomials. It is possible to extend this approach to be exact for any odd polynomial class.

We can "extend" these formulae in the sense that we could use them over small intervals and "add them up" to get the integral over the entire domain; now we aren't assuming the function is roughly a polynomial over the entire range, but only over smaller ranges (in effect we allow for different polynomials to be strung together, which formally delivers a piecewise polynomial function). For example, we could use the trapezoidal rule $n-1$ times to integrate each segment separately, which results in the formula

$$\int_{x_1}^{x_n} f(x)\,dx = h\left[\frac{1}{2}f(x_1) + f(x_2) + \cdots + f(x_{n-1}) + \frac{1}{2}f(x_n)\right] + O\left(\frac{(x_n - x_1)^3}{n}D^2 f\right).$$

Doing this procedure to overlapping pairs of intervals yields the extended Simpson's rule:

$$\int_{x_1}^{x_n} f(x)\,dx = h\left[\begin{array}{c} \frac{1}{3}f(x_1) + \frac{4}{3}f(x_2) + \frac{2}{3}f(x_3) + \frac{4}{3}f(x_4) + \frac{2}{3}f(x_5) + \cdots + \\ \frac{2}{3}f(x_{n-4}) + \frac{4}{3}f(x_{n-3}) + \frac{2}{3}f(x_{n-2}) + \frac{4}{3}f(x_{n-1}) + \frac{1}{3}f(x_n) \end{array}\right] + O\left(\frac{1}{n^4}\right);$$

this formula is exact only up to order 2, so we have lost some accuracy due to the overlapping of intervals. However, note that the error is a function not of h but of n; large n reduces the error. Similarly, the composite Boole's rule divides the interval into subintervals of length $4h$ to obtain

$$\int_a^b f(x)\,dx = \frac{2h}{45}\left[\begin{array}{c} 7(f(a) + f(b)) + 32(f(a+h) + \cdots + f(a+(n-1)h)) + \\ 12(f(a+2h) + \cdots + f(a+(n-2)h)) + \\ 14(f(a+4h) + \cdots + f(a+(n-4)h)) \end{array}\right].$$

We can also employ adaptive rules, which check the approximation error over the interval and, if unacceptably large, subdivides once and repeats.

A more powerful version of the Newton-Cotes approach is called **Romberg's method**. Romberg's method starts with the trapezoid rule, then uses a procedure called Richardson extrapolation to reduce approximation errors. Suppose we have already computed a trapezoid rule with one and two subintervals:

$$T_{1,1} = \frac{b-a}{2}(f(a) + f(b))$$
$$T_{2,1} = \frac{b-a}{4}\left(f(a) + 2f\left(\frac{a+b}{2}\right) + f(b)\right).$$

A general n-interval composite trapezoidal rule would give us

$$\int_a^b f(x)\,dx = \frac{h}{2}\left(f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right) + \sum_{i=1}^{\infty} K_i h^{2i}$$

where K_i are constants that depend on the derivatives of f and

$$h = \frac{b-a}{n}$$
$$x_j = a + jh.$$

Note that

$$T_{1,1} = I(f) + K_1 h^2 + O\left(h^4\right)$$
$$T_{2,1} = I(f) + K_1\left(\frac{h}{2}\right)^2 + O\left(h^4\right)$$

where $I(f)$ is the value of the true integral. Ignoring the error term and solving these two equations yields

$$T_{2,2} = I(f) = T_{2,1} + \frac{T_{2,1} - T_{1,1}}{3};$$

this step is the Richardson extrapolation. Now suppose we compute another approximation, called $T_{3,1}$, using four subintervals; we can use Richardson extrapolation again to get $T_{3,2}$ and then again to get $T_{3,3}$:

$$T_{2,2} = I(f) + K_2 h^4 + O\left(h^6\right)$$
$$T_{3,2} = I(f) + K_2\left(\frac{h}{2}\right)^4 + O\left(h^6\right)$$
$$T_{3,3} = T_{3,2} + \frac{T_{3,2} - T_{2,2}}{15}.$$

The general recursion is

$$T_{j,1} = \frac{1}{2}T_{j-1,1} + h\sum_{j=1}^{2^{j-2}} f\left(a + (2j-1)h\right)$$
$$T_{j,k} = T_{j,k-1} + \frac{T_{j,k-1} - T_{j-1,k-1}}{4^{k-1} - 1}.$$

Given a desired error order $2J$, we need only let j run from 1 to J and k run from 2 to j.

An alternative approach is available if we have approximated our function as a polynomial (or piecewise polynomial). For example, if we have constructed a cubic spline interpolating function we have an exact formula for the integral:

$$\int f(x)\,dx = \sum_{i=1}^{n-1}\left[\frac{(x_{i+1}-x_i)\,(f(x_{i+1})+f(x_i))}{2} - \frac{(x_{i+1}-x_i)\,(x_{i+1}-x_i)\,(v(x_{i+1})+v(x_i))}{24}\right]$$

where the v terms are the constructed second derivatives. Similarly, if we have approximated our function as a linear combination of polynomials, integration is trivially easy; we will discuss these issues below.

Note that these formulae do not compute expectations, simply integrals. Expectations are more complicated because they involve an additional function, namely the density of the distribution one is using to take expectations. Of course, one could write

$$\int g(x)\,dx = \int f(x)\,\phi(x)\,dx$$

where $\phi(x)$ is the density function of the random variable. That is,

$$E[f(X)] = \int f(x)\,\phi(x)\,dx.$$

One simply applies the Newton-Cotes formulae to the product function. For a normal random variable we would then have

$$\begin{aligned} E[f(X)] &= \int_{-\infty}^{\infty} f(x)\,\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)dx \\ &= \int_{-\infty}^{\infty} g(x)\,dx. \end{aligned}$$

This approach is not preferred; it requires more nodes than certain specialized methods do to achieve a given level of accuracy, because the resulting functions are often not close to polynomials. All polynomials diverge to ∞ as their argument converges to ∞ (ignoring the sign; obviously it can go either way); the Gaussian density converges to 0, so that the behavior of $g(x)$ as $x \to \infty$ may or may not look polynomial depending on what f is doing (leaving aside the issue of the unbounded domain).

4.2 Quadrature Rules

We now move to more sophisticated and generally better methods for integrating. These methods, collectively known as **quadrature rules**, have the advantage that several of them are specialized

for computing expectations. One key difference is that the nodes – the points where the finite sum is defined – are no longer fixed in advance (say, by making them evenly-spaced), but rather chosen to improve the accuracy of the integral by demanding that our rule exactly integrate polynomials of a given order: an n-point rule will integrate exactly a polynomial of order $2n-1$, which is much better than we achieved using Newton-Cotes rules with fixed nodes.

Gauss-Legendre quadrature uses the weighting function 1 and uses the zeros of the Legendre polynomials; it is used for expectations of uniform random variables. The formula for integrating a function over the interval $[a,b]$ is

$$\int_a^b f(x)\,dx = \frac{b-a}{2}\sum_{i=1}^{n}\omega_i f\left(\frac{(x_i+1)(b-a)}{2}+a\right)+\frac{2^{2n+1}(n!)^4}{(2n+1)((2n)!)^3}D_{2n}f(\xi)$$

for some $\xi\in[a,b]$; for the standard uniform $[0,1]$ we get

$$\int_0^1 f(x)\,dx = \frac{1}{2}\sum_{i=1}^{n}\omega_i f\left(\frac{x_i+1}{2}\right).$$

The nodes and weights can be calculated using freely available code. Note that Gauss-Legendre quadrature can also be used for general integration over a finite interval, since the density is trivial. The proof that the error term goes to zero as $n\to\infty$ is tedious, and relies on the fact that $(2n)!$ diverges much faster than $n!$ to overwhelm the fact that 2^{2n+1} diverges faster than $2n$.

Gauss-Hermite quadrature is particularly useful for computing expectations of normal random variables because the weighting function is closely related to the density of the standard normal (it is only missing the $\frac{1}{2}$ term). The quadrature rule is given by

$$\int_{-\infty}^{\infty} f(x)\,e^{-x^2}dx = \sum_{i=1}^{n}\omega_i f(x_i)+\frac{n!\sqrt{\pi}}{2^n(2n)!}D_{2n}f(\xi);$$

as before we can find the nodes and weights using available code. As before, the elimination of the error term as $n\to\infty$ relies on the fact that $(2n)!$ diverges very quickly, although here we get the added bonus of 2^n in the denominator and no additional terms in the numerator to make convergence extremely rapid. To apply Gauss-Hermite quadrature to expectations of functions of normal random variables, we simply use the linear change of variables

$$x=\frac{y-\mu}{\sigma\sqrt{2}}$$

and compute

$$E\left[f\left(Y\right)\right] = \int_{-\infty}^{\infty} f\left(y\right) e^{-(y-\mu)^2/(2\sigma^2)} dy$$
$$= \frac{1}{\sqrt{\pi}} \sum_{i=1}^{n} \omega_i f\left(\sigma\sqrt{2}x_i + \mu\right).$$

Note the presence of the $\sqrt{\pi}$ term, which delivers

$$\sum_{i=1}^{n} \frac{\omega_i}{\sqrt{\pi}} = 1;$$

it is numerically more stable to simply renormalize the weights directly into

$$\widehat{\omega}_i = \frac{\omega_i}{\sum_{i=1}^{n} \omega_i}.$$

Another type of quadrature is useful for dealing with integrals with exponential discounting. This rule, called **Gauss-Laguerre quadrature**, computes

$$\int_0^{\infty} f\left(x\right) e^{-x} dx = \sum_{i=1}^{n} \omega_i f\left(x_i\right) + \frac{\left(n!\right)^2}{(2n)!} D_{2n} f\left(\xi\right);$$

as before, we rely on the fast convergence of the denominator term $(2n)!$, which now has to overwhelm the divergence of $(n!)^2$. To compute the general integral we use the linear change of variables

$$x = r\left(y - a\right)$$

and compute

$$\int_a^{\infty} e^{-ry} f\left(y\right) dy = \frac{e^{-ra}}{r} \sum_{i=1}^{n} \omega_i f\left(\frac{x_i}{r} + a\right).$$

Gauss-Laguerre integration is particularly useful in continuous time, as the weighting function is the continuous-time equivalent to geometric discouting at a constant rate, but it also computes expectations for exponentially-distributed variables.

If we have a beta distributed random variable we can use **Gauss-Jacobi quadrature**. The zeroes of the Jacobi polynomial $J_{N+1}^{\alpha,\beta}$ are the eigenvalues of the symmetric tridiagonal matrix

$$A_{N+1} = \begin{bmatrix} a_0 & \sqrt{b_1} & 0 & \cdots & 0 \\ \sqrt{b_1} & a_1 & \sqrt{b_2} & \ddots & \vdots \\ 0 & \sqrt{b_2} & a_2 & \ddots & 0 \\ \vdots & \ddots & \ddots & a_{N-1} & \sqrt{b_N} \\ 0 & \cdots & 0 & \sqrt{b_N} & a_N \end{bmatrix}$$

where

$$a_j = \frac{\beta^2 - \alpha^2}{(2j+\alpha+\beta)(2j+\alpha+\beta+2)}$$
$$b_j = \frac{4j(j+\alpha)(j+\beta)(j+\alpha+\beta)}{(2j+\alpha+\beta-1)(2j+\alpha+\beta)^2(2j+\alpha+\beta+1)}.$$

These eigenvalues are the nodes for the quadrature formula. The weights are

$$\omega_j = \frac{G_N^{\alpha,\beta}}{J_N^{\alpha,\beta}(x_j) DJ_{N+1}^{\alpha,\beta}(x_j)}$$
$$G_N^{\alpha,\beta} = \frac{2^{\alpha+\beta}(2N+\alpha+\beta+2)\Gamma(N+\alpha+1)\Gamma(N+\beta+1)}{(N+1)!\Gamma(N+\alpha+\beta+2)}.$$

To compute the expectation of a beta random variable, which takes the form

$$\frac{1}{\beta(\alpha,\beta)}\int_0^1 f(y) y^{\alpha-1}(1-y)^{\beta-1} dy = \frac{1}{2^{\alpha+\beta-1}\beta(\alpha,\beta)}\int_{-1}^1 f\left(\frac{1-x}{2}\right)(1-x)^{\alpha-1}(1+x)^{\beta-1} dx$$
$$= \sum_{j=0}^n \omega_j f\left(\frac{1-x_j}{2}\right) + \frac{\Gamma(n+\alpha+1)\Gamma(n+\beta+1)\Gamma(n+\alpha+\beta+1)}{(2n+\alpha+\beta+1)[\Gamma(2n+\alpha+\beta+1)]^2}\frac{2^{2n+\alpha+\beta+1}n!}{(2n)!}D_{2n}f(\xi),$$

we use the change of variables

$$y = \frac{1-x}{2}.$$

The Gamma function, defined as

$$\Gamma(z) = \int_0^\infty x^{z-1}\exp(-x)\, dx,$$

equals the factorial at the positive integers,

$$\Gamma(n) = (n-1)!$$

but is defined for all real numbers (it is the unique log-convex extension of the factorial function; obviously if we do not impose additional constraints, we would get infinitely-many functions that agree with the factorial at the positive integers). Thus, the convergence of the error term is again dependent on the fast divergence of $(2n)!$, which now appears twice. If you want to formally prove the convergence of the error terms in the above expressions, it is useful to note that the derivative of the Gamma function at the positive integers is

$$D\Gamma(n) = (n-1)!\left(-\gamma + \sum_{k=1}^{m-1}\frac{1}{k}\right)$$

where γ is the Euler-Mascheroni constant and the convention is $0! = 1$. You can then apply l'Hôpital's rule to the error terms.

Note that a beta distribution can be increasing, decreasing, multimodal, and have many different features. We can also adjust the range, as usual, to an arbitrary interval $[a, b]$ rather than $[0, 1]$ through the usual linear translation.

Finding the nodes and weights of the beta distribution illustrate the general approach to quadrature. Another example of this general approach is quadrature with a Student's t distribution. Let the degrees of freedom of the t random variable be $\nu > 0$, and require $\nu > n$ so that all the relevant moments exist and are finite. Then set

$$\begin{aligned} a_i &= 0 \\ b_0 &= 1 \\ b_i &= \frac{i\nu\,(\nu - 1 + i)}{(\nu - 2i)\,(\nu - 2i + 2)} \end{aligned}$$

and construct the J matrix

$$J = \begin{bmatrix} 0 & \sqrt{b_1} & 0 & \cdots & \cdots & 0 \\ \sqrt{b_1} & 0 & \sqrt{b_2} & 0 & \cdots & 0 \\ 0 & \sqrt{b_2} & \ddots & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \sqrt{b_{n-2}} & 0 \\ \vdots & \vdots & \ddots & \sqrt{b_{n-2}} & 0 & \sqrt{b_{n-1}} \\ 0 & 0 & \cdots & 0 & \sqrt{b_{n-1}} & 0 \end{bmatrix}.$$

Note there is an upper limit for n given ν, as $b_i \geq 0$ is required:

$$n < \frac{\nu}{2}.$$

Now compute the eigenvalues and normalized eigenvectors of J, where normalized means that they have length 1:

$$p^* = \frac{p}{\langle p|p\rangle} = \frac{p}{\sum_{i=1}^{n} p_i^2}.$$

The nodes are the eigenvalues, and the weights are the squared first elements of the eigenvectors. Note that one of the eigenvalues is zero if n is odd, which corresponds to the mean of the Student t distribution, and the nodes and weights will be symmetric. A non-zero mean μ is easily handled by shifting the nodes by μ.

If our distribution is not from these classes (maybe because we have no idea what it is), we may be able to use a 'mixture of normals' as a very flexible way of approximating it. To take a simple case, suppose that our distribution is a mixture of two normals with mixing weight α:

$$f(x) = \frac{\alpha}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) + \frac{1-\alpha}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right).$$

Mixtures can approximate a lot of different distributions; for example, a mixture of two normals can be unimodal or multimodal, and a mixture with other types is also possible (such as mixing a normal with a χ^2). More complicated mixture distributions can be constructed using copulas, which we will discuss below in the multidimensional subsection. Note that the convex combination of two normals is not distributed as a mixture of normals, but rather as a normal:

$$\alpha X_1 + (1-\alpha) X_2 \sim N\left(\alpha\mu_1 + (1-\alpha)\mu_2, \alpha^2\sigma_1^2 + (1-\alpha)^2\sigma_2^2\right).$$

In contrast, the mixture distribution has the same mean

$$E[X] = \alpha\mu_1 + (1-\alpha)\mu_2$$

but a different variance (unless $\mu_1 = \mu_2$):

$$\begin{aligned} E\left[(X-\mu)^2\right] &= E\left[X^2 - \mu^2\right] \\ &= \alpha\left(\sigma_1^2 + \mu_1^2\right) + (1-\alpha)\left(\sigma_2^2 + \mu_2^2\right) - (\alpha\mu_1 + (1-\alpha)\mu_2)^2 \\ &= \alpha\sigma_1^2 + (1-\alpha)\sigma_2^2 + \alpha(1-\alpha)(\mu_1-\mu_2)^2. \end{aligned}$$

Furthermore, the skewness of the convex combination is still zero, but again not for the mixture unless $\mu_1 = \mu_2$:

$$\begin{aligned} E\left[(X-\mu)^3\right] &= E\left[X^3 - 3X^2\mu + 2\mu^3\right] \\ &= \alpha\left(\mu_1^3 + 3\mu_1\sigma_1^2\right) + (1-\alpha)\left(\mu_2^3 + 3\mu_2\sigma_2^2\right) - \\ &\quad 3\left(\alpha\left(\sigma_1^2 + \mu_1^2\right) + (1-\alpha)\left(\sigma_2^2 + \mu_2^2\right)\right)(\alpha\mu_1 + (1-\alpha)\mu_2) + \\ &\quad 2(\alpha\mu_1 + (1-\alpha)\mu_2)^3 \\ &= \alpha(1-\alpha)(\mu_1-\mu_2)\left((1-2\alpha)(\mu_1-\mu_2)^2 + 3\left(\sigma_1^2 - \sigma_2^2\right)\right). \end{aligned}$$

Similarly, the excess kurtosis is

$$E\left[\frac{(X-\mu)^4}{\sigma^4}\right] = \frac{\alpha\left(\mu_1^4 + 6\mu_1^2\sigma_1^2 + 3\sigma_1^4\right) + (1-\alpha)\left(\mu_2^4 + 6\mu_2^2\sigma_2^2 + 3\sigma_2^4\right)}{\left(\alpha\left(\sigma_1^2 + \mu_1^2\right) + (1-\alpha)\left(\sigma_2^2 + \mu_2^2\right)\right)^2} - 3,$$

which can easily be seen to not necessarily be zero. Thus, the resulting distribution is not normal unless the means agree; just plotting them you will see that if $\mu_1 \neq \mu_2$ you will typically get two modes.

There are also formulas for truncated distributions, such as a truncated normal, which would arise in models with selection effects. As before, we start by constructing the moment matrix

$$M = \begin{bmatrix} \mu_0 & \cdots & \mu_n \\ \vdots & \ddots & \vdots \\ \mu_n & \cdots & \mu_{2n} \end{bmatrix}$$

where

$$\mu_i = \int_a^b x^i f(x)\, dx.$$

Then compute the Cholesky decomposition

$$M = R^T R$$

where R is upper triangular. From here we compute a vector α of length n and a vector β of length $n-1$ by the formula

$$\alpha_1 = \frac{R_{1,2}}{R_{1,1}}$$
$$\alpha_i = \frac{R_{i,i+1}}{R_{i,i}} - \frac{R_{i-1,i}}{R_{i-1,i-1}}$$

and

$$\beta_i = \frac{R_{i+1,i+1}}{R_{i,i}}.$$

We then form the tridiagonal matrix

$$J = \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ \beta_1 & \alpha_2 & \beta_2 & \ddots & \vdots \\ 0 & \beta_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \alpha_{n-1} & \beta_{n-1} \\ 0 & \cdots & 0 & \beta_{n-1} & \alpha_n \end{bmatrix}$$

and compute the eigenvalues and eigenvectors of J. The nodes are the eigenvalues, and the weights are the squares of the first component of the eigenvectors. For the truncated normal, we

can calculate the moments easily using the binomial expansion:

$$\mu_i = \sum_{j=0}^{i} \binom{i}{j} \sigma^j \mu^{i-j} L_j$$

where

$$\begin{aligned}
L_0 &= 1 \\
L_1 &= -\frac{\phi(0,1;\beta) - \phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)} \\
L_i &= -\frac{\beta^{i-1}\phi(0,1;\beta) - \alpha^{i-1}\phi(0,1;\alpha)}{\Phi(0,1;\beta) - \Phi(0,1;\alpha)} + (i-1) L_{i-2} \\
\alpha &= \frac{a-\mu}{\sigma} \\
\beta &= \frac{b-\mu}{\sigma}.
\end{aligned}$$

For other distributions, we can use the moment generating function:

$$m_X(t) = E[\exp(tX)] = 1 + tE[X] + \frac{t^2 E\left[X^2\right]}{2!} + \frac{t^3 E\left[X^3\right]}{3!} + ...;$$

note that

$$\left.\frac{d^k m_X(t)}{dt^k}\right|_{t=0} = E\left[X^k\right].$$

If we know the form of $m(t)$, we can simply compute these derivatives and fill up the moment matrix. Note: for some distributions $m(t)$ does not exist, but for those cases the moments also do not exist. For linear transformations, we have

$$\begin{aligned}
m_{\alpha X+\beta}(t) &= E[\exp(t(\alpha X + \beta))] = \exp(\beta t) m_X(\alpha t) \\
&= \exp(\beta t)\left(1 + \alpha t E[X] + \frac{\alpha^2 t^2}{2} E\left[X^2\right] + ...\right)
\end{aligned}$$

and

$$\left.\frac{d m_{\alpha X+\beta}(t)}{dt}\right|_{t=0} = \alpha E[X] + \beta$$

and so on. Note that these moments are not "centered"; for example, to get the variance we need to use the formula

$$Var[X] = E\left[X^2\right] - (E[X])^2.$$

Higher moments like skewness and kurtosis can be constructed in a similar fashion:

$$Skew\left[X\right] = E\left[X^3\right] - 3E\left[X^2\right]E\left[X\right] + 2E\left[X\right]^3$$
$$Kurt\left[X\right] = E\left[X^4\right] - 4E\left[X^3\right]E\left[X\right] + 6E\left[X^2\right]E\left[X\right]^2 - 3E\left[X\right]^4.$$

One unfortunate property of Gaussian quadrature rules is that the nodes and weights for an n-point rule are not contained in the nodes and weights for an m-point rule with $m > n$. This non-nestedness means that increasing accuracy is costly; Gauss-Kronrod integration rules fix this problem by picking $n+1$ points to be added to an existing n-point Gaussian quadrature rule that delivers a $3n+1$ rule (remember the n-point Gaussian rule is of order $2n-1$). The extra nodes for the Kronrod extension are the zeroes of the Stieljes polynomial E_{n+1}, which satisfies the integral equation

$$\int_{-1}^{1} P_n\left(x\right) E_{n+1}\left(x\right) w\left(x\right) x^k dx = 0$$

for $k \in \{0, ..., n\}$, where $w\left(x\right)$ is a weighting function and $P_n\left(x\right)$ is the relevant class of orthogonal polynomials. Unfortunately, for many cases the Kronrod extension does not exist as the resulting roots are not real or do not lie on the interval of interest, in particular for Gauss-Hermite quadrature with $n \notin \{1, 2, 4\}$, which limits the usefulness of this construction. Furthermore, note that it requires the same number of evalutions to compute the $2n+1$ order Gaussian rule as to obtain the Kronrod extension, and the Gaussian rule would be more accurate (order $4n+1$). That is, if you know how many nodes you need in advance, there is no need to do the extension; however, if you instead only know how accurate you need your integral computation to be, then the extension is useful – you can compute the embedded rule (the n-point rule) integral, then quickly improve it.

Finally, we have **Gauss-Chebyshev** quadrature. Here, the rule is

$$\int_{-1}^{1} f\left(x\right)\left(1-x^2\right)^{-\frac{1}{2}} dx = \frac{\pi}{n}\sum_{i=1}^{n} f\left(x_i\right).$$

Note that the weights do not vary. The weighting function is not the kernel of any density function, so this rule is not particularly useful for computing expectations; however, it will play an important role in functional approximation via projection methods.

One last point is to note that, because quadrature rules are designed to integrate polynomials and all polynomials diverge as $|x| \to \infty$, the weights drop off rapidly as $n \to \infty$ (since the range

of the nodes increases in n the weights must go to zero to cancel out the diverging terms). Thus, there is often no point in using a large n, since many of the weights will be negligible in size. Check your weights before wasting time computing terms that get effectively zero weight in the sum.

4.3 Multidimensional Integrals

For multidimensional integrals the problem gets much more difficult. The simplest method is the "product approach," where we simply multiply weights together. For example, take the integral

$$\int_{[-1,1]^d} f(x)\, dx$$

where x has dimension d. With quadrature the weighting function becomes

$$W(x) \equiv W(x_1, ..., x_d) = \prod_{l=1}^{d} w^l(x_l)$$

with the component weighting function $w^l(x_l)$ and nodes given by any rule we choose. This approach suffers from the curse of dimensionality as d gets large. Product versions of Newton-Cotes are of no practical value for this reason, and quadrature rules should be avoided for anything larger than 3-4 dimensions.

The better approach is to use a Gaussian rule over a smaller set of polynomials – these are called **monomial rules**. Before going into detail about how to obtain rules, I will first present some useful and simple ones. Let $e^j \equiv (0, ..., 1, ...0)$. There is a $2d$ rule that is exact over the complete set of third-order polynomials:

$$\int_{[-1,1]^d} f(x)\, dx = \frac{2^{d-1}}{d} \sum_{i=1}^{d} \left(f\left(\left(\frac{d}{3}\right)^{1/2} e^i \right) + f\left(-\left(\frac{d}{3}\right)^{1/2} e^i \right) \right).$$

For fifth-order polynomials we can use

$$\int_{[-1,1]^d} f(x)\, dx = \omega_1 f(0) + \omega_2 \sum_{i=1}^{d} \left(f\left(ue^i\right) + f\left(-ue^i\right) \right) + \omega_3 \sum_{\substack{1 \le i \le d \\ i < j \le d}} \left(f\left(u\left(e^i \pm e^j\right)\right) + f\left(-u\left(e^i \pm e^j\right)\right) \right)$$

where

$$\begin{aligned}
\omega_1 &= 2^d \left(25d^2 - 115d + 162\right) \\
\omega_2 &= 2^d \left(70 - 25d\right) \\
\omega_3 &= \frac{25}{324} 2^d \\
u &= \left(\frac{3}{5}\right)^{1/2}.
\end{aligned}$$

For $d \geq 3$ this approach suffers some convergence properties because the weights get large and alternate in sign, making them difficult to accurately add together. Another scheme for fifth-order complete polynomials is the Radon 7-point formula

$$\begin{aligned}
\int_{[-1,1]^2} f\left(x,y\right) dxdy = \omega_1 f\left(P_1\right) + \\
\omega_2 \left[f\left(P_2\right) + f\left(P_3\right) + f\left(P_4\right) + f\left(P_5\right)\right] + \\
\omega_3 \left[f\left(P_6\right) + f\left(P_7\right)\right]
\end{aligned}$$

where

$$\begin{aligned}
P_1 &= (0,0) \\
P_2, P_3, P_4, P_5 &= (\pm s, \pm t) \\
P_6, P_7 &= (\pm r, 0) \\
s &= \sqrt{1/3} \\
t &= \sqrt{3/5} \\
r &= \sqrt{14/15} \\
\omega_1 &= \frac{8}{7} \\
\omega_2 &= \frac{5}{9} \\
\omega_3 &= \frac{20}{63}.
\end{aligned}$$

We can also use symmetry conditions to simplify things. For example, consider the integral

$$\int_{\mathcal{R}^d} f\left(x\right) e^{-\sum_{i=1}^d x_i^2} dx;$$

these come up in the expectation of multivariate normal random variables. We have two choices; the degree 3 rule with $2d$ points

$$\int_{\mathcal{R}^d} f(x) e^{-\sum_{i=1}^d x_i^2} dx = \frac{V}{2d} \sum_{i=1}^d f\left(\pm re^i\right)$$
$$r = \sqrt{d/2}$$
$$V = \pi^{d/2}$$

and the degree 5 rules with $2d^2 + 1$ points

$$\int_{\mathcal{R}^d} f(x) e^{-\sum_{i=1}^d x_i^2} dx = Af(0, ..., 0) + B \sum_{i=1}^d \left(f\left(re^i\right) + f\left(-re^i\right)\right) +$$
$$D \sum_{i=1}^{d-1} \sum_{j=i+1}^d f\left(\pm se^i \pm se^j\right)$$
$$r = \sqrt{1 + \frac{d}{2}}$$
$$s = \sqrt{\frac{1}{2} + \frac{d}{4}}$$
$$V = \pi^{d/2}$$
$$A = \frac{2}{d+2} V$$
$$B = \frac{4-d}{2(d+2)^2}$$
$$D = \frac{V}{(d+2)^2}.$$

A third example is a degree 5 rule with $2^d + 2d + 1$ points:

$$\int_{[-1,1]^d} f(x)\,dx = Af(0,...,0) + B\sum_{i=1}^{d}\left(f\left(re^i\right) + f\left(-re^i\right)\right) + D\sum_{x\in C} f(x)$$

$$C = \{x|\forall i\,(x_i = \pm 1)\}$$
$$r = \sqrt{\frac{2}{5}}$$
$$V = 2^d$$
$$A = \frac{8-5d}{9}V$$
$$B = \frac{5}{18}V$$
$$D = \frac{1}{9}.$$

The key is that many of these formula increase in computational complexity with d not exponentially but polynomially, meaning that the curse of dimensionality is much less severe (the last rule is an exception, as it increases exponentially in d).

A theorem by Möller (1979) establishes the minimum number of nodes required to get a given order rule.

Theorem 14 *The number of nodes N of a cubature rule of order $2s-1$ in dimension d satisfies $N \geq N_{\min}$, where*

$$N_{\min} = \begin{cases} \binom{d+s-1}{d} + \sum_{k=1}^{d-1} 2^{k-d}\binom{k+s-1}{k} & s \text{ even} \\ \binom{d+s-1}{d} + \sum_{k=1}^{d-1}\left(1-2^{k-d}\right)\binom{k+s-2}{k} & s \text{ odd} \end{cases}.$$

For the most part, rules that satisfy this lower bound are not known. The degree 3 rule above satisfies the Möller bound of $2d$ points, but the degree 5 rules do not achieve the lower bound of $d^2 + d + 1$, and no rules are known that do so in general. Meng and Luo (2018) derive a degree 5 rule that achieves the lower bound for the special case of $d = 7$ over a spherically symmetric region. Many rules that can only be calculated numerically are better than the ones here, but they apply only for specific values of d.

When using these formulas, we must pay attention to accuracy if our integrand is not a polynomial of the appropriate degree. Some of the weights in cubature rules can be negative; this feature is generally undesirable, so rules with negative weights are viewed as "low quality".

The coefficient of stability is defined as

$$\rho \equiv \frac{\sum_{i=1}^{N} |w_i|}{\sum_{i=1}^{N} w_i} = \sum_{i=1}^{N} |w_i| \geq 1,$$

where we normalize the weights to sum to 1. If all the weights are positive, $C = 1$; if some weights are negative, $C > 1$, which indicates the potential for large errors (for some functions those negative weights can create delicate cancellation issues), and the larger the coefficient the larger the potential errors are. Note that the degree 5 rules above have negative weights for large enough d ($d > 4$ and $d > 1$, respectively).

For multivariate normals, we have to use a linear change of variables. The expectation of a function of a multivariate normal is given by

$$E\left(f\left(Y\right)\right) = \left|\Sigma\right|^{-\frac{1}{2}} \left(2\pi\right)^{-\frac{d}{2}} \int_{\mathcal{R}^d} f\left(y\right) \exp\left(-\frac{1}{2}\left(y-\mu\right)^T \Sigma^{-1}\left(y-\mu\right)\right) dy.$$

We need a notion of a matrix square root to implement the change of variables; while there are many such square roots, the Cholesky decomposition is of particular use here.

Theorem 15 *(Cholesky Decomposition) Suppose Σ is a real symmetric positive-definite $n \times n$ matrix. Then there exists a lower triangular $n \times n$ matrix Ω with positive diagonal elements such that $\Sigma = \Omega\Omega^T$. If Σ is only positive semi-definite with rank r, then there exists at least one permutation matrix P such that $P\Sigma P^T = \Omega\Omega^T$ with*

$$\Omega = \begin{bmatrix} \Omega_1 & 0 \\ \Omega_2 & 0 \end{bmatrix}$$

and Ω_1 is a lower triangular $r \times r$ matrix with a positive diagonal.

Suppose we have a full rank stochastic system. We can therefore use the Cholesky decomposition $\Sigma = \Omega\Omega^T$ which implies $\Sigma^{-1} = \left(\Omega^{-1}\right)^T \Omega^{-1}$. The linear transformation is

$$x = \Omega^{-1}\left(y-\mu\right)/\sqrt{2}$$

or

$$y = \sqrt{2}\Omega x + \mu.$$

Thus

$$\det \Sigma^{-\frac{1}{2}} \frac{1}{\sqrt{(2\pi)^d}} \int_{\mathcal{R}^d} f(y) \exp\left(-\frac{(y-\mu)^T \Sigma^{-1} (y-\mu)}{2}\right) dy = \int_{\mathcal{R}^d} f\left(\sqrt{2}\Omega x + \mu\right) \exp\left(-\sum_{i=1}^d x_i^2\right) |\det \Omega| \, 2^{\frac{d}{2}} dx$$

$$= \frac{1}{\sqrt{\pi^d}} \int_{\mathcal{R}^d} f\left(\sqrt{2}\Omega x + \mu\right) \exp\left(-\sum_{i=1}^d x_i^2\right) dx.$$

Then use the appropriate monomial formula. As before, we accomodate the $\frac{1}{\sqrt{\pi^d}}$ term by normalizing the weights to sum to 1.

There are specialized routines for computing integrals over non-rectangular domains such as simplicies, polyhedra, disks, and hyperspheres. More general distributions can be handled by modeling them as mixtures of one-dimensional marginal distributions.

Definition 16 *A function $C : [0,1]^d \to [0,1]$ is a d-dimensional **copula** if C is a joint cumulative distribution function of a random vector $X \in [0,1]^d$ with uniform marginal distributions.*

One can show that C is a copula iff (i) $C(u_1, ..., u_{i-1}, 0, u_{i+1}, ..., u_d) = 0 \; \forall i$, (ii) $C(1, ..., 1, u, 1, ..., 1) = u$; and (iii)

$$\int_B dC(u) = \sum_{z \in \prod_{i=1}^d \{x_i, y_i\}} (-1)^{N(z)} C(z) \geq 0$$

where $N(z) = \#\{k : z_k = x_k\}$. An easy description of a copula is as follows. Suppose $(X_1, ..., X_d)$ is a random vector with continuous marginal distributions $F_i(x) = \Pr\{X_i \leq x\}$. The random vector $(U_1, ..., U_d) = (F_1(X_1), ..., F_d(X_d))$ has uniform $[0,1]$ marginals. The copula of $(X_1, ..., X_d)$ is then defined as the joint cumulative distribution function of $(U_1, ..., U_d)$:

$$C(u_1, ..., u_d) = \Pr\{U_1 \leq u_1, ..., U_d \leq u_d\}.$$

C contains all the information about the dependence structure between the X components, whose behavior is described by the independent marginals. Note that we can write the copula as

$$C(u_1, ..., u_d) = \Pr\left\{X_1 \leq F_1^{-1}(u_1), ..., X_d \leq F_d^{-1}(u_d)\right\}.$$

There are two main classes of copulas. The Gaussian copula satisfies

$$C_\Sigma^{Gauss}(u) = \Phi_\Sigma\left(\Phi^{-1}(u_1), ..., \Phi^{-1}(u_d)\right)$$

where Φ is the cdf of a standard normal and Φ_Σ is the cdf of a normal with variance-covariance matrix Σ. The pdf is

$$c_\Sigma^{Gauss}(u) = \frac{1}{\sqrt{|\Sigma|}} \exp\left(\frac{1}{2} \begin{pmatrix} \Phi^{-1}(u_1) & \cdots & \Phi^{-1}(u_d) \end{pmatrix}^T \left(\Sigma^{-1} - I\right) \begin{pmatrix} \Phi^{-1}(u_1) \\ \vdots \\ \Phi^{-1}(u_d) \end{pmatrix} \right).$$

The Archimedean copulas satisfy the representation

$$C(u_1, ..., u_d; \theta) = \psi^{[-1]}(\psi(u_1; \theta) + \cdots + \psi(u_d; \theta); \theta)$$

where $\psi : [0,1] \times \Theta \to [0, \infty)$ is a continuous, strictly decreasing, and convex function such that $\psi(1; \theta) = 0$ and

$$\psi^{[-1]}(t; \theta) = \begin{cases} \psi^{-1}(t; \theta) & \text{if } 0 \leq t \leq \psi(0; \theta) \\ 0 & \text{otherwise} \end{cases}$$

is a pseudo-inverse function. Some members of this class include the Ali-Mikhail-Haq copula

$$\begin{aligned} C(u_1, ..., u_d) &= \frac{\prod_{i=1}^d u_i}{1 - \theta \prod_{i=1}^d (1 - u_i)} \\ \psi(t; \theta) &= \log\left(\frac{1 - \theta(1-t)}{t}\right) \\ \psi^{-1}(t; \theta) &= \frac{1-\theta}{\exp(t) - \theta} \end{aligned}$$

with $\theta \in [-1, 1]$, the Gumbel copula

$$\begin{aligned} C(u_1, ..., u_d) &= \exp\left(- \left(\sum_{i=1}^d (-\log(u_i))^\theta \right)^{\frac{1}{\theta}} \right) \\ \psi(t; \theta) &= (-\log(t))^\theta \\ \psi^{-1}(t; \theta) &= \exp\left(-t^{\frac{1}{\theta}}\right) \end{aligned}$$

with $\theta \in [1, \infty)$, and the Joe copula

$$\begin{aligned} C(u_1, ..., u_d) &= 1 - \left(\sum_{i=1}^d (1-u_i)^\theta - \prod_{i=1}^d (1-u_i)^\theta \right)^{\frac{1}{\theta}} \\ \psi(t; \theta) &= -\log\left(1 - (1-t)^\theta\right) \\ \psi^{-1}(t; \theta) &= 1 - (1 - \exp(-t))^{\frac{1}{\theta}} \end{aligned}$$

with $\theta \in [1, \infty)$. For the Ali-Mikhail-Haq copula with $\theta = 0$ or the Gumbel copula with $\theta = 1$, we get the independence copula

$$C(u_1, ..., u_d) = \prod_{i=1}^{d} u_i$$
$$\psi(t) = -\log(t)$$
$$\psi^{-1}(t) = \exp(-t).$$

Expectations of copulas satisfy

$$\begin{aligned} E[g(X_1, ..., X_d)] &= \int_{[0,1]^d} g\left(F_1^{-1}(u_1), ..., F_d^{-1}(u_d)\right) c(u_1, ..., u_d)\, du_1 \cdots du_d \\ &= \int_{\mathcal{R}^d} g(x_1, ..., x_d)\, c(F_1(x_1), ..., F_d(x_d))\, f_1(x_1) \cdots f_d(x_d)\, dx_1 \cdots dx_d. \end{aligned}$$

If the marginals are one of the distributions where we have formulas, we can apply them directly to this equation; for example, if all the marginals are normals, then we can use a product version of Gauss-Hermite quadrature applied to the joint function $g \cdot c$. We can even mix them together to get a mixture of normals and betas, for example, with correlations that are controlled by c.

4.4 Monte Carlo and Quasi-Monte Carol Integration

A completely different approach to integration is by Monte Carlo or simulation methods. From the ergodic theorem, we know that, under regularity conditions, it is the case that sample moments converge to their population equivalents:

$$\lim_{T\to\infty} \left\{ \frac{1}{T} \sum_{t=1}^{T} x_t \right\} = E(X)$$
$$\lim_{T\to\infty} \left\{ \frac{1}{T} \sum_{t=1}^{T} x_t^2 \right\} = E\left(X^2\right)$$

and so on; for any uniformly continuous function, we have

$$\lim_{T\to\infty} \left\{ \frac{1}{T} \sum_{t=1}^{T} g(x_t) \right\} = E[g(X)].$$

How then might we compute $E(X)$? We could calculate the LHS with "sufficiently large T." For example, what if we wanted to compute

$$E\left(X^2\right) = \int x^2 f(x)\, dx$$

where $f(x)$ is the pdf of X? We then draw a large number of x's from $f(x)$, square them, and average; note that in general you should divide each term by T before summing, to avoid creating a sum that overflows. This approach is good because it is simple; it is bad because it can take a very long time to converge and drawing the random numbers might be tedious and difficult to do, particularly if the pdf is difficult to sample from. How large do we need T to be? As is familiar from econometrics, the convergence rate is $O\left(\frac{1}{\sqrt{T}}\right)$, so if you want to reduce your sampling error in half you need to multiply T by 4; this convergence rate is quite bad. Unfortunately, for very large integrals Monte Carlo may be the only practical method. The numbers we obtain are called pseudo-random because they are deterministic if you know the "seed" value. There are good pseudo-random number generators available for nearly all distributions now, but for pedagogical purposes I will describe a transformation, called Box-Muller, that turns uniform random numbers into normals. Suppose u_1 and u_2 are independent draws from a uniform distribution on $[0, 1]$. Then

$$z_1 = \sqrt{-2\ln(u_1)}\cos(2\pi u_2)$$
$$z_2 = \sqrt{-2\ln(u_1)}\sin(2\pi u_2)$$

are two independent standard normals. Due to finite arithmetic, Box-Muller is limited in how far from 0 the normals can go; with 64-bit addressing, the standard used now, the limit is roughly 9.4 standard deviations, which seems adequate for most work.

In multivariate environments sampling can be difficult because of correlation between random variables of different types (imagine trying to sample a vector that consists of a normal and a beta with nonzero correlation; as noted above, we can model these kinds of distributions using copulas). In this case, we can use two Markov chain tools, Gibbs sampling and the Metropolis-Hastings algorithm, to get a sample from any distribution (these methods are used frequently in Bayesian estimation, where the posterior – the product of the prior and the likelihood – is usually intractable because we do not have a closed-form for it) provided we can sample from conditional distributions. Gibbs sampling starts with a given vector x^0. We then draw each component of x^1 sequentially:

$$x_1^1 \sim f\left(x_1 | x_2^0, ..., x_n^0\right)$$
$$x_2^1 \sim f\left(x_2 | x_1^1, x_3^0, ..., x_n^0\right)$$

and so on. We then generate a sample of vectors $\left(x^t\right)_{t=1}^T$ which satisfies

$$\lim_{T\to\infty}\left\{\frac{1}{T}\sum_{t=1}^{T} g\left(x^t\right)\right\} = E\left[g\left(X\right)\right].$$

If we know that some groups of variables have a joint distribution, say multivariate normal, then we can do a "block" Gibbs sampler that draws the components in groups or blocks. One example of the Gibbs sampler is the Bayesian estimation of a VAR

$$y_t = \sum_{i=1}^{q} A_i y_{t-i} + \Sigma^{\frac{1}{2}} e_t$$
$$e_t \sim \mathcal{N}\left(0, I\right).$$

Given the data $\{y_t\}_{t=1}^T$ the variance-covariance Σ terms have an inverse-Wishart posterior, while given the data *and* the variance-covariance matrix the autoregressive terms A have a normal posterior, so we can draw from the posterior by first drawing a variance-covariance matrix and then drawing the autogressive matrix given the draw for the variance-covariance matrix, then repeat many times. Suppose we have an uninformative prior (also called a diffuse or Jeffrey's prior); then the specific distributions are

$$\Sigma|y \sim \mathcal{IW}\left(\left(y-\widehat{A}x\right)^T\left(y-\widehat{A}x\right), T-k\right)$$
$$A|\Sigma, y \sim \mathcal{N}\left(vec\left(\widehat{A}\right), \Sigma\otimes\left(x^Tx\right)^{-1}\right)$$

where x is the vector of lagged y terms that show up on the RHS of the VAR (the "independent variables", including perhaps a constant term) and

$$\widehat{A} = \left(x^Tx\right)^{-1}x^Ty$$

is the MLE estimate (since VARs are estimated using OLS, getting $\widehat{A}$ is trivial). Drawing from normals we have already discussed, but how does one draw from an inverse Wishart distribution? The algorithm from Odell and Feiveson (1966) is one method.

Theorem 17 *Suppose that V_i $(1 \leq i \leq d)$ are independent χ^2 random variables with V_i having degrees of freedom $n-i+1$. Suppose that N_i are independent standard normal random variables.*

The random variables

$$b_{ii} = V_i + \sum_{r=1}^{i-1} N_{ri}^2$$

$$b_{ij} = N_{ij}\sqrt{V_i} + \sum_{r=1}^{i-1} N_{ri}N_{rj}$$

form a matrix $B = [b_{ij}]$ *with a Wishart distribution* $\mathcal{W}(I_d, d, n)$.

To get the inverse Wishart draw, we just invert B; remember that χ^2 random variables with n degrees of freedom are the sum of the squares of n independent standard normals, so we can obtain those variables by repeated use of Box-Muller. If for some reason you get an ill-conditioned matrix B (close to singular), just discard it and draw again.

A more general version of Gibbs sampling is the Metropolis-Hastings algorithm, which can be used even if we cannot sample from the conditional distributions, provided the kernel of f can be *evaluated* (that is, we do not know the normalizing constant that ensures f integrates to one). Take an initial x^0 and fix a "proposal density" $Q\left(x'|x^t\right)$ (use something easy to sample from, like a multivariate normal with mean equal to x^t and some specified variance). We draw a candidate x' from $Q\left(x'|x^0\right)$ and a uniform random number u from $[0, 1]$. Then

$$x^1 = \begin{cases} x' & \text{if } u \leq \frac{f(x')}{f(x^0)} \\ x^0 & \text{otherwise} \end{cases};$$

that is, we always accept a draw that implies a higher density (that is, the draw is "more likely" under f) but also sometimes accept a draw that implies a lower density. Note that the unknown normalizing constant would cancel out of this expression. One typically tunes Q to imply an acceptance rate of around 30 percent; if your acceptance rate is too high the chain will be trapped around local modes and if it is too low the chain will converge very slowly (require many draws). With enough draws the sample converges to a sample from f. As we will see later, Metropolis-Hastings was actually designed as a global optimization routine, but Chib and Greenberg (1987) discovered it generated samples with nice properties when applied to likelihoods – in particular, it allows us to sample from the posterior without having to know what it is. Bayes' rule says that the posterior equals the prior times the likelihood; then our f function is the product of the likelihood (assumed to be known) times the prior (fixed). In many cases the resulting posterior

would be otherwise intractable, since it does not have a known parametric form. This approach is now common in the estimation of structural macroeconomic models; the evaluation of the likelihood given the parameters is a filtering problem, and the MH algorithm is used to generate the parameter values. We will discuss filtering, which is the process of estimating hidden variables from limited or noisy observations, later in these notes.

We can improve our Monte Carlo integration by using some variance-reduction techniques. One such technique is to employ **antithetic variates**. Assume that the median of your distribution occurs at the vector $\left(x^{Med}\right)$. For each draw x, evaluate not only x but also all of its antithetes:

$$f\left(x^{Med} - x\right).$$

This process improves the sampling coverage of the Monte Carlo approach – if you happen to draw a very large value you also get to automatically draw a very small one. A simple example is the uniform $[0, 1]$ distribution; now for each draw u we also include $1 - u$. This procedure works because it generates negative covariance between the "drawn" sample and the "antithetic" sample, which reduces the variance of the average by more than just expanding the size of the sample:

$$Var\left(\frac{X_1 + X_2}{2}\right) = \frac{1}{2}Var\left(X_1\right) + \frac{1}{2}Cov\left(X_1, X_2\right) < \frac{1}{2}Var\left(X_1\right)$$

if $Cov\left(X_1, X_2\right) < 0$. Antithetic variates will work for any monotonic function of X (if the function is not monotonic, then we cannot guarantee that covariance is negative); for example, if we can use a monotonic transformation of the uniform to generate our random variables (such as Box-Muller), then we can apply antithetic variates. One way to use them is to invert the CDF of the distribution of interest; that is, we choose a uniform $[0, 1]$ random variable and compute

$$X_1 = F^{-1}\left(u\right)$$

and

$$X_2 = F^{-1}\left(1 - u\right).$$

These samples will then have negative covariance. If F is intractable, a reasonable approximation is

$$X_{2,t} = \frac{2}{T}\sum_{t=1}^{T} X_{1,t} - X_{1,t}.$$

Remember the goal is only to reduce variance, so speed definitely comes into play; wasting time inverting F, even if it is straightforward, may simply not be worth it compared to using the approximation.

Another useful tool is called **importance sampling**; it chooses the points by using a related distribution. What if there were a distribution $g(x)$ which "looked like" $f(x)$ but was easier to sample from? We could then choose points using $g(x)$ instead of $f(x)$. The trick is finding what $g(x)$ works well for your particular distribution; if g does not have a similar shape to f, then we could end up with a sample that poorly approximates one under f. We can then calculate the integral by using the fact that

$$\frac{1}{T}\sum_{t=1}^{T} x_t$$

sampled from f equals

$$\frac{1}{T}\sum_{t=1}^{T} x_t \frac{f(x_t)}{g(x_t)}$$

sampled from g. The second term is the "likelihood ratio" that weights each point according to how likely it would be under f relative to g. Note that we now don't need to sample from f, so we can use importance sampling as long as we can evaluate f.

Another useful variance reduction technique is called **stratified sampling**. Here we divide the space for x into segments and sample from each separately, guaranteeing that we get good coverage. This approach is simple if the underlying distribution is uniform; if not we need some method to guarantee that we sample "the right amount" from each segment. One way to ensure this is to compute the quantiles by inverting the CDF, as discussed above in our discussion of antithetic variates.

Finally, we always want to use **common random numbers** – if the integral is to be computed many times, say for different prices in a model, then we keep the same draws for each evaluation. This reduces "chatter", which is the discrepancy caused by different finite samples (having different random numbers can lead to a failure of stochastic equicontinuity, which would invalidate the convergence theorems).

A related tool called **rejection sampling** can help us integrate over an irregular domain – we pick points and toss out those that lie outside the domain of interest, and then keep drawing until we get T accepted draws. The obvious challenge is how to figure out if the draw is inside or

outside the domain of interest. For complicated domains, we approximate it as a polygon (the union of line segments between fixed nodes on the boundary) and use the "ray test": from each point draw a ray that diverges to infinity in some direction and count the number of times this ray crosses the boundary of the polygon. If the number of crossings is odd, the point is inside the polygon; if even, it is outside. For convex polyhedra the test is even easier; we can create a triangularization of the boundary nodes (also called a tesselation; see later in the notes) and systematically check the triangles.

4.4.1 Typical Set

There are some issues of concern with Monte Carlo integration, in particular concerns about coverage. Consider a set $\{x_1, ..., x_n\}$ drawn from an iid distribution X. The **typical set** A^n are those draws that satisfy

$$2^{-n(H(X)+\epsilon)} \leq p(x_1, ..., x_n) \leq 2^{-n(H(X)-\epsilon)}$$

where

$$H(X) = -\sum_y p(y) \log(p(y))$$

is the entropy of X. Even though a typical set may be relatively small, an iid sample is very likely to be a member. Oddly, the most likely sequences may not be members of the typical set. For example, suppose we have a Bernoulli random variable with $p(1) = 0.9$ and $p(0) = 0.1$. The most likely sequence of length n has all 1s. But the probability of this sequence satisfies

$$\frac{1}{n} \log(0.9^n) = 0.152$$

independent of n, whereas the entropy is

$$H(X) = 0.469.$$

The paradoxical result comes about because the probability of one specific sequence of 0s and 1s is very low, but the number of possible sequences with a given number of 0s and 1s is large, so the chance of getting one of these sequences is relatively high while the chance of getting any specific sequence is low.

Why do typical sets matter? Consider the expectation

$$E[\theta|y] = \int_{\Theta} \theta p(\theta|y)\, d\theta.$$

Since posterior events are integrals over the indicator function $\mathbf{1}(\theta \in A)$, we have

$$\Pr[A|y] = \int_{\Theta} \mathbf{1}(\theta \in A)\, p(\theta|y)\, d\theta \equiv \int_{T \subset \Theta} \mathbf{1}(\theta \in A)\, p(\theta|y)\, d\theta.$$

Almost every draw will lie in the typical set: $\forall \epsilon > 0, \exists n_0$ such that $\forall n \geq n_0$ we have

$$\Pr(T_{\epsilon}^{n}) \geq 1 - \epsilon,$$

where T_{ϵ}^{n} is the typical set for sequences of length n bounded by the inequalities given ϵ. Furthermore, as n gets large, the probability of being in the typical set gets larger. And as the dimension of x rises, the typical set becomes smaller – the reason is that, as the dimension increases, the volume of an ϵ-ball decreases relative to the size of the ambient space. Thus, as the dimension of our integral gets bigger, the less coverage we will expect to see.

For example, suppose we have a standard normal, which has a mode equal to 0. As we move away from 0, the density declines and the volume of the ball increases with the dimension. The product delivers the probability mass, which ends up concentrating around 10, not 0. What is remarkable is that the density at that distance from the mean is on the order of 10^{-23}!

4.4.2 Quasi-Monte Carlo Methods

Quasi-Monte Carlo methods use a particular sequence of realizations that has been pre-selected because of nice properties. A sequence is **equidistributed** if

$$\lim_{T \to \infty} \left\{ \frac{b-a}{T} \sum_{j=1}^{T} f(x_j) \right\} = \int_{a}^{b} f(x)\, dx$$

for any f. That is, an equidistributed sequence is ex post uniform. If we could construct actual random number sequences, then the law of large numbers would apply and we would get uniformity; however, the sequences generated by the computer are not really random and therefore may deviate ex post from uniformity (especially if the sample size is small). In this case, we might do better by "picking" a particular sequence, enabling us to keep T relatively small.

A good sequence is one that has "low discrepancy" from a uniform distribution. Formally, define the **discrepancy** of a sequence $X = \{x_1, ..., x_T\}$ as

$$D_T(X) = \sup_{B \in J} \left\{ \frac{A(B; X)}{T} - \lambda(B) \right\}$$

where λ is the standard Lebesgue measure, $A(B; X)$ is the number of points in X that lie in the set B, and J is the space of half-open Borel sets (boxes). Sequences have low discrepancy if the points in any given box are close to uniformly distributed. Three examples of low discrepancy sequences that work well for integration are the Halton sequence (for problems of dimension less than six) and the Faure and Sobol' sequences (for higher dimensional problems).

To generate the Halton sequence, suppose we are working in $[0, 1]^2$ so we need two sequences of numbers; we can use as a basis any two coprime numbers (coprime numbers have no common divisors except 1). Suppose we use 2 for the first dimension and 3 for the second. We first divide $[0, 1]$ into half, then each part into half again, and so on, generating the sequence

$$\left\{ \frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}, ... \right\}.$$

For the second dimension, we get

$$\left\{ \frac{1}{3}, \frac{2}{3}, \frac{1}{9}, \frac{4}{9}, \frac{7}{9}, \frac{2}{9}, \frac{5}{9}, ... \right\}.$$

The pairs

$$\left\{ \left(\frac{1}{2}, \frac{1}{3} \right), \left(\frac{1}{4}, \frac{2}{3} \right), \left(\frac{3}{4}, \frac{1}{9} \right) \right\}$$

constitute the first three terms in the Halton sequence. In higher dimensions the Halton sequence does not perform well, although a "leaped" Halton sequence in which we use only every nth entry works better (where n is a prime number not used in the sequence generation).

For Sobol' sequences consider the j^{th} component. Define the primitive polynomial

$$x^{s_j} + a_{1,j} x^{s_j - 1} + a_{2,j} x^{s_j - 2} + ... + a_{s_j - 1, j} x + 1,$$

where the a coefficients are either 0 or 1. Also define a sequence of positive integers $\{m_{1,j}, m_{2,j}, ...\}$ by

$$m_{k,j} = 2 a_{1,j} m_{k-1,j} \oplus 2^2 a_{2,j} m_{k-2,j} \oplus ... \oplus 2^{s_j} m_{k-s_j,j} \oplus m_{k-s_j,j}$$

where $\oplus$ is a bit-by-bit exclusive-or operator (equals 1 if the terms are different, 0 if they are the same), with initial values chosen arbitrarily with the restriction that $m_{k,j}$ is odd and less than 2^k for $1 \leq k \leq s_j$. The direction numbers are defined as

$$v_{k,j} = \frac{m_{k,j}}{2^k}$$

The jth component of the ith Sobol' number is then

$$x_{i,j} = i_1 v_{1,j} \oplus i_2 v_{2,j} \oplus ...,$$

where i_k is the kth digit from the right when i is written in binary form. At this point, I'm sure an example is useful. Consider $s_j = 3$, $a_{1,j} = 0$, $a_{2,j} = 1$, so that we have the primitive polynomial

$$x^3 + x + 1.$$

Starting with $m_{1,j} = 1$, $m_{2,j} = 3$, and $m_{3,j} = 7$, we then have

$$m_{4,j} = 5$$

$$m_{5,j} = 7$$

and so on. The direction numbers are

$$v_{1,j} = (0.1)_2$$

$$v_{2,j} = (0.11)_2$$

$$v_{3,j} = (0.111)_2$$

$$v_{4,j} = (0.0101)_2$$

$$v_{5,j} = (0.00111)_2$$

and so on. We then obtain the elements of the jth component as

$$x_{0,j} = 0$$

$$x_{1,j} = 0.5$$

$$x_{2,j} = 0.75$$

$$x_{3,j} = 0.25$$

$$x_{4,j} = 0.875$$

$$x_{5,j} = 0.375$$

and so on.

The Faure sequence starts with $p \geq d$ being the smallest prime number larger than the dimension. Define

$$c_j = \binom{i}{j} \bmod (p)$$

for $0 \leq j \leq i \leq m$, where p^m defines the upper bound on the sample size. The base p representation of n, for $n = \{0, 1, 2, ...\}$, is

$$n = \sum_{i=0}^{m-1} a_i (n) p^i,$$

where $a_i (n) \in [0, p)$ are integers. The first coordinate of the nth term is

$$x_n^1 = \sum_{j=0}^{m-1} a_j (n) p^{-j-1}$$

and the other coordinates satisfy

$$\overline{a}_j (n) = \sum_{l=j}^{m-1} c_{l,j} a_l (n) \bmod (p) \quad j \in \{0, 1, ..., m-1\}$$

$$a_j (n) = \overline{a}_j (n) \quad j \in \{0, 1, ..., m-1\}$$

$$x_n^i = \sum_{j=0}^{m-1} a_j (n) p^{-j-1}$$

in the order $i = \{2, ..., d\}$. As shown in Cheng and Druzdzel (2000), as the dimension increases the Sobol' sequence dominates the Faure sequence. Both clearly dominate (quasi)random sequences – their coverage is far more uniform.

To convert low-discrepancy sequences – which are uniform by design – into normal variables we can use Moro's "Full Monte" approximation to the inverse cdf of the normal (Moro 1995). Let

$$a_1 = 2.50662823884$$

$$a_2 = -18.61500062529$$

$$a_3 = 41.39119773534$$

$$a_4 = -25.44106049637$$

$$b_1 = -8.47351093090$$

$$b_2 = 23.08336743743$$

$$b_3 = -21.06224101826$$

$$b_4 = 3.13082909833$$

$$c_1 = 0.3374754822726147$$

$$c_2 = 0.9761690190917186$$

$$c_3 = 0.1607979714918209$$

$$c_4 = 0.0276438810333863$$

$$c_5 = 0.0038405729373609$$

$$c_6 = 0.0003951896511919$$

$$c_7 = 0.0000321767881768$$

$$c_8 = 0.0000002888167364$$

$$c_9 = 0.0000003960315187.$$

Then if $0.08 \leq x \leq 0.92$ we compute

$$y = x - \frac{1}{2}$$

$$z = y^2$$

$$\Phi^{-1}(x) = y\frac{a_1 + a_2 z + a_3 z^2 + a_4 z^3}{1 + b_1 z + b_2 z^2 + b_3 z^3 + b_4 z^4}$$

and if $x < 0.08$ or $x > 0.92$ we compute

$$\begin{aligned} y &= x - \frac{1}{2} \\ z &= \left\{ \begin{array}{ll} x & \text{if } y \leq 0 \\ 1-x & \text{if } y > 0 \end{array} \right\} \\ \kappa &= \log\left(-\log\left(z\right)\right) \\ q &= \left\{ \begin{array}{ll} y & \text{if } y > 0 \\ -y & \text{if } y \leq 0 \end{array} \right\} \\ \Phi^{-1}\left(x\right) &= q\left(\sum_{i=1}^{9} c_i \kappa^{i-1}\right). \end{aligned}$$

The Full Monte uses a rational polynomial approximation in the center of the distribution and a more flexible approximation in the tails to improve accuracy.

4.5 δ-Method

One last method that may occasionally be useful is the δ-method, generally used to derive the asymptotic distribution of functions of random variables. Suppose

$$\sqrt{n}\left|X_n - \theta\right| \xrightarrow{D} N\left(0, \Sigma\right).$$

Then for any differentiable function $g\left(x\right)$ with $Dg\left(\theta\right) \neq 0$, we have

$$\sqrt{n}\left|g\left(X_n\right) - g\left(\theta\right)\right| \approx \sqrt{n} Dg\left(\theta\right)^T \left|X_n - \theta\right|$$

so that

$$\sqrt{n}\left|g\left(X_n\right) - g\left(\theta\right)\right| \xrightarrow{D} N\left(0, Dg\left(\theta\right)^T \Sigma Dg\left(\theta\right)\right).$$

If $Dg\left(\theta\right) = 0$ but $D^2 g\left(\theta\right) \neq 0$, we can see that

$$\sqrt{n}\left|g\left(X_n\right) - g\left(\theta\right)\right| = \frac{1}{2}\sqrt{n}\left|X_n - \theta\right|^T D^2 g\left(\theta\right)\left|X_n - \theta\right|.$$

Similarly, we can use a quadratic expansion to get

$$\sqrt{n}\left|g\left(X_n\right) - g\left(\theta\right)\right| \approx \sqrt{n} Dg\left(\theta\right)^T \left|X_n - \theta\right| + \frac{1}{2}\sqrt{n}\left|X_n - \theta\right|^T D^2 g\left(\theta\right)\left|X_n - \theta\right|,$$

which is a weighted average of a normal and a χ^2, the weights being the first and second derivatives.

How do these facts help with numerical integration? Since

$$E\left[g\left(X\right)\right] \approx E\left[g\left(\theta\right) + Dg\left(\theta\right)\left(X-\theta\right) + \frac{1}{2}D^2 g\left(\theta\right)\left(X-\theta\right)^2\right] + ...,$$

we then have

$$E\left[g\left(X\right)\right] = g\left(\theta\right) + Dg\left(\theta\right)E\left[X-\theta\right] + \frac{1}{2}D^2 g\left(\theta\right)E\left[\left(X-\theta\right)^2\right] + ...,$$

which writes the expectation as a Taylor expansion in the moments. Provided the higher-order moments are small and/or the function is smooth, the δ-method can provide a quick and reasonably accurate numerical integration tool.

5 Markov Chains

A Markov chain is a finite-state process that can be characterized completely by two objects – the finite set of states $Z = \{z_1, ..., z_n\}$ and the transition matrix $\Pi = [\pi_{ij}]_{i,j=1,n}$ that describes the probability of leaving state i and entering state j. Π must have rows that sum to 1 and no element can be negative:

$$\pi_{ij} \geq 0$$
$$\sum\nolimits_j \pi_{ij} = 1 \quad \forall i.$$

That is, with probability 1 you must move from state i to some state j and transition probabilities are never negative. Π gives the probability of transitioning between states in t and $t+1$; correspondingly, Π^n gives the probability of transitioning between states in t and $t+n$. Let the elements of Π^n be denoted π_{ij}^n, which is not equal to π_{ij} raised to the nth power.

Suppose the transition probability matrix is time-invariant. An important characteristic of a Markov chain is the invariant distribution.

Definition 18 *An **invariant distribution** of a Markov chain is a vector π^* such that*

$$\pi^* = \Pi\pi^*.$$

Markov chains that have two properties possess unique invariant distributions.

Definition 19 *State i* ***communicates*** *with state j if $\pi_{ij}^n > 0$ and $\pi_{ji}^n > 0$ for some $n \geq 1$. A Markov chain is said to be* ***irreducible*** *if every pair (i, j) communicate.*

An irreducible Markov chain has the property that it is possible to move from any state to any other state in a countable number of periods (if N is finite, then n will be finite). Note that we do not require that such a movement is possible in one step, so $\pi_{ij} = 0$ is permitted.

Definition 20 *The* ***period*** *of state i is given by*

$$k = \gcd \{n : \pi_{ii}^n > 0\}$$

where gcd *is "greatest common divisor". If $k = 1$ for all i the Markov chain is* ***aperiodic****.*

A state has period k if any return to it occurs only in multiples of k steps. For example, take

$$\Pi = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix};$$

both states in this chain have period 2, since

$$\Pi^2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

has $\pi_{11} = \pi_{22} = 1$. A less trivial example is

$$\Pi = \begin{bmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{bmatrix}.$$

All states here are period 2 as well, since

$$\Pi^2 = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

$$\Pi^3 = \Pi.$$

For a Markov chain that is both irreducible and aperiodic, Π possesses only one eigenvalue of modulus 1 (that it has one such eigenvalue is a consequence of the Perron-Frobenius theorem). It also does not possess any **absorbing states** (a state i^* such that $\pi_{i^*i^*} = 1$) or **transient states** (a state i^* such that $\pi_{i^*j} = 0 \; \forall j \neq i^*$). If these two conditions are satisfied, then there exists a unique set of probabilities $[\pi_i^*]_{i=1,n}$ that define the unconditional probability of being in state i. As noted in the definition, this vector solves the eigenvector equation

$$\pi^* \Pi = \pi^*,$$

so that the invariant distribution is the eigenvector associated with the single unitary eigenvalue (assuming the eigenvectors have been orthonormalized); given that there is only one such eigenvector, the invariant distribution is unique. An alternative method is to compute

$$\Pi^* = \lim_{t \to \infty} \left\{ \Pi^t \right\}$$

which produces a matrix with identical columns, each of which is the invariant distribution. Under the assumptions of irreducibility and aperiodicity, this limit exists and is therefore obviously unique. In general, the second approach is easier and more numerically stable, unless Π possesses some special structure.

We can interpret the invariant distribution in two ways: (i) π_i^* is the unconditional probability that the chain is currently in state i; (ii) π_i^* is the probability that the chain will be in state i in t steps as t diverges to ∞. If the invariant distribution is not unique, these two interpretations would give different answers. To see why, take a particular degenerate case:

$$\Pi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

There are infinitely-many invariant distributions, corresponding to any value between 0 and 1 placed on state 1. But the limit distribution starting from state 1 is not the same as the one starting from state 2.

For a unique invariant distribution, it follows that

$$\pi_j^* = \sum\nolimits_i \pi_{ij} \pi_i^*$$

for each j, since taking one step further than ∞ is still ∞.

Taking conditional expectations of Markov chains just requires finite summation:

$$E\left[z'|z=z_i\right]=\sum\nolimits_j \pi_{ij}z_j.$$

Other moments are straightforward to compute as well, such as the conditional variance

$$\text{var}\left(z'|z=z_i\right)=\sum\nolimits_j \pi_{ij}z_j^2-\left(\sum\nolimits_j \pi_{ij}z_j\right)^2.$$

The long-run mean and variance can be obtained using the invariant distribution:

$$\begin{aligned}E\left[z\right]&=\sum\nolimits_i \pi_i^* z_i\\ \text{var}\left(z\right)&=\sum\nolimits_i \pi_i^* z_i^2-\left(\sum\nolimits_i \pi_i^* z_i\right)^2.\end{aligned}$$

We can also calculate statistics like the "return time," the expected number of periods before the chain revisits state i:

$$\begin{aligned}m_i&=E\left[\inf\left\{n>1:z^n=z_i|z=z_i\right\}\right]\\&=\frac{1}{\pi_i^*},\end{aligned}$$

and the persistence of a state, equal to the expected number of periods before the chain moves:

$$\begin{aligned}p_i&=\left(1-\pi_{ii}\right)+2\pi_{ii}\left(1-\pi_{ii}\right)+3\pi_{ii}^2\left(1-\pi_{ii}\right)+...\\&=\left(1-\pi_{ii}\right)\sum_{j=0}^{\infty}j\pi_{ii}^j\\&=\frac{1}{1-\pi_{ii}}.\end{aligned}$$

Another statistic of interest is the "mobility", which provides a measure of how quickly you move through the states of the chain. The typical measure is the value of the largest non-unitary eigenvalue. The larger this number, the less mobile the chain; for a two-state process the second largest eigenvalue equals the autocorrelation, although this condition is not true in general. Alternatively, we can look at the "average mixing time", which is

$$\min_t\left\{\pi_{ij}^t-\pi_j^*\right\}\leq\frac{1}{4}\ \forall i,j;$$

that is, how many periods ahead do we need to look before we are sufficiently close to the stationary distribution for state j no matter which state i we start in. Other mobility measures include the

trace (which captures how infrequently you leave a state), the Bartholomew measure

$$\sum_{i,j} |i-j|\,\pi_{ij},$$

which measures how much the chain moves if it moves, and the mean passage time (the average number of periods required for two chains that start in different states to "switch" places).

5.1 Rouwenhorst's Method of Approximation

Suppose that

$$z' = \rho z + \sigma\epsilon'$$

with $\epsilon' \sim N(0,1)$. Define

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

as the unconditional variance of z.

In keeping with finite-state dynamic programming, it may be convenient to "discretize" this process as a finite-state Markov chain. One straightforward approach is called Rouwenhorst's method (Rouwenhorst 1995), which recently was given some firmer theoretical foundation by Kopekcy and Suen (2010). Let

$$p = q = \frac{1+\rho}{2};$$

$p = q$ is a consequence of the AR process being symmetric. The N state transition matrix can be constructed using the recursion

1. For $N = 2$, set
$$\Pi_2 = \begin{bmatrix} p & 1-p \\ 1-q & q \end{bmatrix}.$$
2. For $N \geq 3$, construct
$$p\begin{bmatrix} \Pi_{N-1} & \mathbf{0}_N \\ \mathbf{0}_N^T & 0 \end{bmatrix} + (1-p)\begin{bmatrix} \mathbf{0}_N & \Pi_{N-1} \\ 0 & \mathbf{0}_N^T \end{bmatrix} + (1-q)\begin{bmatrix} \mathbf{0}_N^T & 0 \\ \Pi_{N-1} & \mathbf{0}_N \end{bmatrix} + q\begin{bmatrix} 0 & \mathbf{0}_N^T \\ \mathbf{0}_N & \Pi_{N-1} \end{bmatrix}.$$
3. Divide all rows except the top and bottom by 2 to obtain Π_N.

The realizations $\{z_1, ..., z_N\}$ are set to be evenly spaced between $z_1 = -\psi$ and $z_N = \psi$ with $\psi = \sigma_z\sqrt{N-1}$. Note that $p \neq q$ is perfectly fine and generates an asymmetric process (it will have nonzero skew). Rouwenhorst's method matches exactly the unconditional mean, unconditional variance, and autocorrelation of the underlying continuous process, and matches conditional moments better than many alternatives.

This method was extended by Gospodinov and Lkhagvasuren (2014) to approximate VARs:

$$\begin{bmatrix} z_1' \\ z_2' \end{bmatrix} = \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} + \begin{bmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{12} & \sigma_{22} \end{bmatrix} \begin{bmatrix} \epsilon_1' \\ \epsilon_2' \end{bmatrix}.$$

The basic idea is to orthogonalize the two rows using a spectral decomposition

$$\begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix} = PDP^{-1}$$

where D is the diagonal matrix of eigenvalues and P is the corresponding matrix of eigenvectors, using Rouwenhorst's method on each row separately, then undoing the orthogonalization (along with some extra moment-matching conditions to identify the covariance terms). We take a process with N states per row and convert it to a joint process with N^2 states. Since a higher-order AR process is equivalent to a VAR(1), their approach can also be used for those as well.

You might ask why we bother to approximate a continuous process, instead of simply starting with the Markov chain. The problem is that estimating a Markov chain is costly – there are N^2 values to figure out (the N realizations and $N(N-1)$ elements of the transition matrix), which requires lots of data, and the estimation method is nonlinear (although it is straightforward – you just count the number of transitions between bin i and bin j). In contrast, the AR(1) process has only three parameters, the mean, autocorrelation, and innovation variance, and these values can be easily estimated using OLS. Furthermore, increasing N would require reestimation of the entire process, whereas Rouwenhorst's method requires hardly any work to increase it. The downside is that we are restricted to Gaussian random variables, or we might as well be, since only the first two moments are being matched.

5.2 Tauchen's Method

A common but somewhat less accurate method is called Tauchen's method (Tauchen 1986). Choose $z_1 = -m\sigma_z$ and $z_n = m\sigma_z$ in order to cover m standard deviations of the unconditional

distribution, and evenly-space the grid points on this interval:

$$z_i = z_1 + \frac{z_n - z_1}{n-1}(i-1).$$

The elements of Π are chosen to match the probability of moving from z_i into the interval

$$\left(z_j - \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right), z_j + \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right)\right].$$

For the first interval we use

$$\left(-\infty, z_1 + \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right)\right]$$

and for the last interval we use

$$\left(z_n - \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right), \infty\right).$$

That is,

$$\begin{aligned}\pi_{ij} &= \Pr\left(\epsilon_{t+1} \leq z_j + \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right) - \rho z_i\right) - \Pr\left(\epsilon_{t+1} \leq z_j - \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right) - \rho z_i\right) \\ &= \Phi\left(\frac{z_j + \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right) - \rho z_i}{\sigma}\right) - \Phi\left(\frac{z_j - \frac{1}{2}\left(\frac{z_n - z_1}{n-1}\right) - \rho z_i}{\sigma}\right)\end{aligned}$$

where Φ is the cdf of the standard normal. Obviously we can accommodate other density functions as well, provided we can compute their cdf functions.

A better choice for the grid would be to use the zeros from an nth-order Hermite polynomial, since these polynomials are used to numerically integrate normal random variables, which delivers the Tauchen-Hussey method (Tauchen and Hussey 1991). That is, the approximation becomes a version of Gauss-Hermite quadrature:

$$\int_{-\infty}^{\infty} f(x) e^{-x^2} dx = \sum_{i=1}^{N} \omega_i f(x_i) + \frac{n!\sqrt{\pi}}{2^n} \frac{f^{(2n)}(\xi)}{(2n)!};$$

the weights and nodes can be obtained using any number of publicly-available routines. To apply Gauss-Hermite quadrature to expectations of functions of nonstandard normal random variables, we simply use the linear change of variables

$$x = \frac{y - \mu}{\sigma\sqrt{2}}$$

and compute

$$E\left[f\left(Y\right)\right] = \int_{-\infty}^{\infty} f\left(y\right) e^{-\left(y-\mu\right)^2/\left(2\sigma^2\right)} dy$$
$$= \frac{1}{\sqrt{\pi}} \sum_{i=1}^{n} \omega_i f\left(\sigma\sqrt{2}x_i + \mu\right).$$

Because the nodes in the integration $\sigma\sqrt{2}x_i+\mu$ will depend on the current value of the chain (if it is persistent), we need to adjust them to keep them on the fixed grid; otherwise, the dependence would propagate forward and the chain would nearly fill the real line. Therefore, we renormalize by conditioning on the current nodes:

$$z_j' = \sqrt{2}\sigma x_j + \left(1-\rho\right)\overline{z} + \rho z_i$$
$$\lambda_{i,j} = \frac{1}{\sqrt{\pi}}\omega_i \frac{\phi\left(z_j'|z_i\right)}{\phi\left(z_j'|\overline{z}\right)}$$
$$\overline{\lambda}_{i,j} = \frac{\lambda_{i,j}}{\sum_j \lambda_{i,j}}$$

and x_i and ω_i are the nodes and weights for the nth order Hermite polynomial and ϕ is the standard normal pdf. Essentially we create the transition matrix as

$$\Pi = \left[\overline{\lambda}_{i,j}\right]_{i,j=1}^{n}$$

and the realizations are the z_i. The benefit of this approach is that it provides a better approximation in the tails of the distribution, since the nodes are concentrated there. The Tauchen-Hussey programs also implement this procedure for vector processes using product quadrature rules.

For extremely persistent processes, Floden (2008) recommends using

$$\sigma_b = \omega\sigma^2 + \left(1-\omega\right)\sigma_z^2$$
$$\omega = 0.5 + 0.25\rho$$

as the "variance" that goes into the Markov chain approximation, based on numerical comparisons of the conditional moments (in particular the conditional means and variances at the extremal points 1 and n).

Tauchen's method handles VAR approximation in a straightforward manner using tensor grids (product grids). Gordon (2020) notes that this method is wasteful, since many extreme points will

get nearly zero weight. He suggests dropping low probability points in the invariant distribution and increasing the approximation to get a targeted number of points after pruning the state space, making sure to renormalize each row so that it continues to sum to one.

5.3 Farmer-Toda Method

Farmer and Toda present a method for approximating a nonlinear and/or non-Gaussian stochastic process as a Markov chain. One process where their method is useful is a process with stochastic volatility:

$$x_t = \rho x_{t-1} + \sigma_t \epsilon_t$$
$$\sigma_t = (1-\phi)\,\overline{\sigma} + \phi \sigma_{t-1} + \omega_t$$

where ϵ and ω are iid random variables. The resulting process has fatter tails than a normal. The idea is to match some low-order moments, similar to what GL do, but Farmer and Toda's approach is more flexible. To be specific, consider an arbitrary Markov process

$$P\left(x_t \leq x' | x_{t-1} = x\right) = F\left(x', x\right).$$

Take a set of initial grid points $D_N = \{x_1, ..., x_N\}$ and an initial transition matrix Q. The conditional moments given state x are a vector of length L:

$$\overline{T} = E\left[T\left(x_t\right)|x\right] = \int T\left(x'\right) dF\left(x', x\right);$$

we can use some numerical integration method to compute these moments to high accuracy, particularly if we only need do this step once (Farmer and Toda 2017 use evenly-spaced grids and trapezoid rules, but we could also use Gauss-Legendre integration in general and Gauss-Hermite quadrature if F is normal; quasi-Monte Carlo methods are available for processes whose distributions are not simple to evaluate). Then we solve the minimization problem

$$\min_{\{p_{ij}\}_{j=1}^N} \left\{ \sum_{j=1}^N p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right) \right\}$$

subject to

$$\sum_{j=1}^{N} p_{ij} T(x_j) = \overline{T}$$

$$\sum_{j=1}^{N} p_{ij} = 1$$

$$p_{ij} \geq 0.$$

This problem is a simple linearly-constrained convex programming problem; the objective function is the entropy of p relative to q (called the Kullback-Leibler divergence) and is a measure of how far apart two distributions are. The objective is convex,

$$\frac{d^2}{dp^2}\left(p\log(p) - p\log(q)\right) = \frac{1}{p} > 0,$$

and the constraint set is convex (it is a simplex) if we impose the equality constraint and eliminate p_{iN} for each N:

$$p_{i,N} = \sum_{j=1}^{N-1} p_{ij}$$

and impose that

$$\sum_{j=1}^{N-1} p_{ij} \leq 1.$$

Numerically, we have an issue with limits, however; while

$$\lim_{p\to 0}\{p\log(p)\} = \lim_{p\to 0}\left\{\frac{\log(p)}{\frac{1}{p}}\right\} = \lim_{p\to 0}\left\{\frac{\frac{1}{p}}{-\frac{1}{p^2}}\right\} = -\lim_{p\to 0}\{p\} = 0,$$

the computer is not good at applying l'Hôspital's rule so this calculation can easily diverge. With this divergence in mind, Farmer and Toda suggest solving the unconstrained dual problem

$$\min_{\lambda\in\mathcal{R}^L}\left\{\sum_{j=1}^{N} q_{ij}\exp\left(\lambda^T\left(T(x_j) - \overline{T}\right)\right)\right\}$$

where L is the number of moments; we can compute the probabilities as

$$p_{ij} = \frac{q_{ij}\exp\left(\lambda^T T(x_j)\right)}{\sum_{j=1}^{N} q_{ij}\exp\left(\lambda^T T(x_j)\right)}.$$

5.4 Markov Chain Approximation to a Random Walk

Consider an AR(1) process

$$x_{t+1} = \rho x_t + e_t$$

where e_t is iid and mean zero. If $\rho = 1$, we call this process a "random walk" – the expectation of all future states equals the current value,

$$E_t\left[x_{t+k}\right] = x_t$$

$\forall k$. This process does not have finite unconditional variance:

$$Var\left(x\right) = \frac{Var\left(e\right)}{1-\rho^2}$$

is not finite if $\rho = 1$. Random walks are an example of a more general class of processes called **martingales**, in which the expected change is zero conditional on the history:

$$E\left[x_{t+1} - x_t | x^t\right] = 0.$$

One cannot approximate a random walk using a Markov chain, because the space would be unbounded (because the random walk does not have finite variance). But an approximation to a random walk with a sufficiently-large number of states can be created, where reflecting barriers keep the process bounded and the zero expected change is enforced via permitting only one-state movements with equal probability of being 'up' or 'down'. The transition matrix has zeros everywhere except the once-displaced diagonals, which are both $\frac{1}{2}$ (except for the endpoints, which put $\frac{1}{2}$ on the diagonal). As an example, suppose we have a five-state process; then the transition matrix is

$$\Pi = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

With enough states, the distortion at the endpoints does not matter, and we get a binomial process that approximates a random walk on a bounded interval. The stationary distribution is

$\frac{1}{N}$ on each state for an N-state process. If we want to allow for the chain to remain in a given state, we can use

$$\Pi = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

6 Nonlinear Programming

Consider the general nonlinear programming problem

$$\min_x \{f(x)\}$$

subject to constraints

$$g(x) \leq 0$$
$$h(x) = 0;$$

g can contain bound constraints of the form

$$l \leq x \leq u$$

as well as both linear and nonlinear inequality constraints, and h can also have linear and nonlinear components.

There is a difference between *global* and *local* optima. Finding local optima is usually not very hard; most routines will converge to one of these, provided they do not diverge, and there exist "globally convergent" local optimization routines (that is, they converge to some local minimum from any initial condition, although not necessarily the same local minimum). Finding global optima is much harder; unless we can prove that the problem has a fixed number of local optima we cannot be certain that we found the best one. And we are generally only interested in global optima – for example, the consistency of maximum likelihood requires that the estimator is the global maximizer, not merely a local one. We will first discuss local methods and then global ones.

It is useful to remind ourselves of the mathematical conditions that characterize local and global optima. Some theorems that describe maxima (minima) will be useful, so we'll take a quick detour and present them.

Theorem 21 *(**Weierstrass Extreme Value Theorem**) Suppose that X is a compact set and $f : X \times P \to \mathcal{R}$ is continuous. Then $\exists x^* \in X$ such that $f(x^*) \geq f(x)\ \forall x \in X$.*

Proof. Step 1: Show that $\sup\{f(x) : x \in X\}$ exists, which will be true if it is bounded: $\exists M > 0$ such that $|f(x)| \leq M\ \forall x \in X$. Since X is compact and f is continuous, $f(X)$ is compact. Therefore $f(X)$ is bounded.

Step 2: $x_{\max}$ exists, where $f(x_{\max}) = \sup\{f(x) : x \in X\}$. Suppose not, so that $f(x) < \sup\{f(x) : x \in X\}$ for all $x \in X$. Then for each x there exists $\delta(x) > 0$ such that

$$f(x) + \delta(x) < \sup\{f(x) : x \in X\}.$$

Of course, $\left\{B_{\delta(x)}(f(x)), x \in X\right\}$ forms an open covering of $f(X)$ and $f(X)$ is compact, so

$$f(X) \subseteq \cup_{i=1}^{m} B_{\delta(x_i)}(f(x_i))$$

for finite m, where $x_i \in X$ for all i. Let $\overline{a} \equiv \max\{f(x_i) + \delta(x_i), i = 1, ..., m\}$; then $\overline{a} < \sup\{f(x) : x \in X\}$. But $\overline{a}$ is an upper boundary of $f(X)$ since it exceeds all members of $f(X)$ in value, which is a contradiction of the definition of supremum. ■

This theorem tells us whether a given maximization problem (such as utility maximization) has a solution. But it does not go far enough because it does not tell us anything about that solution. In particular, it may be multi-valued (there may be more than one $x \in X$ that achieve the maximum) and it may not be continuous in parameters. By putting more structure on f and X, we can make stronger statements.

First, a theorem that will be familiar from intro calculus.

Definition 22 *Let f be a real function defined on a metric space X. f has a **local maximum at** $p \in X$ if there exists $\delta > 0$ such that $f(q) \leq f(p)\ \forall q \in X$ such that $d(p, q) < \delta$.*

Theorem 23 *Let f be defined on $[a, b]$. If f has a local maximum at $x \in (a, b)$ and if $Df(x)$ exists, then $Df(x) = 0$.*

Proof. *Choose δ appropriately, so that*

$$a < x - \delta < x < x + \delta < b.$$

If $x - \delta < t < x$, then

$$\frac{f(t) - f(x)}{t - x} \geq 0.$$

Letting $t \to x$ we see $Df(x) \geq 0$.

If $x < t < x + \delta$, then

$$\frac{f(t) - f(x)}{t - x} \leq 0,$$

which shows that $Df(x) \leq 0$. Hence, $Df(x) = 0$. ■

With an appropriate change of notation we can apply this theorem to local minima: if f has a local minimum at $x \in (a, b)$ and if $Df(x)$ exists, then $Df(x) = 0$. This theorem also holds in multiple dimensions. Note that it does not say that $Df(x) = 0$ is sufficient for a local maximum (which is obvious given the existence of local minima and saddlepoints), nor does it say that every local maximum has $Df(x) = 0$ (since the derivative may not exist at that point, such as the maximum of $f(x) = -|x|$, or we may be on the boundary of X).

Theorem 24 *Let $f \in C^2$ and Ω be convex. Let $x \in \text{Int}(\Omega)$ be a local maximum point. Then $D^2 f(x)$ is negative semidefinite ($y^T D^2 f(x) y \leq 0 \ \forall y \in \Omega$).*

Proof. Suppose not and that $\exists y \in \Omega$ such that $y^T D^2 f(x) y > 0$. Let $x(\alpha) = x + \alpha y$ for $\alpha > 0$. Given that $f \in C^2$ we have

$$f(x(\alpha)) = f(x) + Df(x)(x(\alpha) - x) + \frac{1}{2}(x(\alpha) - x)^T D^2 f(\xi)(x(\alpha) - x)$$

where

$$\begin{aligned} \xi &= \theta x + (1 - \theta) x(\alpha) \\ &= x + \alpha(1 - \theta) y. \end{aligned}$$

We then have

$$f(x(\alpha)) = f(x) + \frac{\alpha^2}{2} y^T D^2 f(\xi) y$$

since $Df(x) = 0$ by previous result. Since $D^2 f(x)$ is continuous, for small enough α we must have $y^T D^2 f(\xi) y > 0$ so that $f(x(\alpha)) > f(x)$. Contradiction.

Theorem 25 *Let $f \in C^2$ and Ω be convex. Let $x \in \text{Int}(\Omega)$ and be a local minimum point. Then $D^2 f(x)$ is positive semidefinite ($y^T D^2 f(x) y \geq 0 \ \forall y \in \Omega$).*

■

Proof. Just alter the signs in the above proof. ■

These two theorems hold for any C^2 function. For concave functions we can do much better.

Theorem 26 *Let Ω be convex and $f \in C^1$ be concave. Then $x^* \in \Omega$ is a global maximum if*

$$Df(x^*)(x - x^*) \leq 0$$

$\forall x \in \Omega$. If $x^* \in \text{Int}(\Omega)$ then x^* is a global maximum if and only if $Df(x^*) = 0$.

Proof. *f is concave implies*

$$f(x) \leq f(x^*) + Df(x^*)(x - x^*)$$

$\forall x \in \Omega$. Combining this statement with the assumption we then have

$$f(x^*) \geq f(x).$$

The second statement is immediate. ■

Theorem 27 *Let Ω be convex and $f \in C^1$ be concave. (i) Any local maximum is a global maximum. (ii) The set of global maximum points is convex.*

Proof. *(i) Let x be a local maximum that is not a global maximum. Then $\exists y \in \Omega$ such that $f(y) > f(x)$. Let $x(\alpha) \equiv \alpha x + (1 - \alpha) y$ where $\alpha \in [0, 1]$. Then*

$$\begin{aligned} f(x(\alpha)) &= f(\alpha x + (1 - \alpha) y) \\ &\geq \alpha f(x) + (1 - \alpha) f(y) \\ &> \alpha f(x) + (1 - \alpha) f(x) \\ &= f(x) \end{aligned}$$

which is a contradiction. (ii) Let Γ denote the set of all global maximum points:

$$\Gamma = \{x \in \Omega : f(x) \geq f(y) \ \ \forall y \in \Omega\}.$$

Let x^* be a global maximum point. Then $\Gamma = \{x \in \Omega : f(x) = f(x^*)\}$, which is convex by the concavity of f. ■

Theorem 28 *Let Ω be convex and $f \in C^1$ be strictly concave. The set of global maximizers is a singleton.*

We now move to characterizing how the solution to an optimization problem depends on parameters – components of the problem that are held fixed.

Definition 29 *A **correspondence** (or **multifunction**) is a mapping $\Gamma : X \rightrightarrows 2^Y$ where 2^Y denotes the collection of all subsets of Y (called the **power set** of Y).*

Some familiar objects can be thought of as correspondences. For example, consider the budget set

$$B(p, m) = \left\{c \in \mathcal{R}^n_+ | pc \leq m\right\}. \tag{16}$$

B is a correspondence which maps the point $(p, m) \in \mathcal{R}^n_+ \times \mathcal{R}_+$ (prices and income) into a subset of $\mathcal{R}^n_+$, namely those consumption bundles that the household can afford to purchase with income m and market prices p. Since this set changes as (p, m) changes, we can view it as a correspondence.

We will now define some continuity concepts for correspondences – these will allow us to use the concept for maximization.

Definition 30 *Let $P \subset \mathcal{R}^n$ and $Y \subset \mathcal{R}^m$. A correspondence $\Gamma : P \rightrightarrows 2^Y$ is **upper hemicontinuous at** $p \in P$ if for each open set $U \subset Y$ that contains $\Gamma(p)$ there exists a neighborhood V of p such that $\Gamma(y) \subset U$ whenever $y \in V$. If this property holds for all $p \in P$ the correspondence is said to be **upper hemicontinuous**.*

Definition 31 *Let $P \subset \mathcal{R}^n$ and $Y \subset \mathcal{R}^m$. A correspondence $\Gamma : P \rightrightarrows 2^Y$ is **lower hemicontinuous at** $p \in P$ if for each open set $U \subset Y$ with $\Gamma(p) \cap U \neq \emptyset$, there exists a neighborhood V of p such that $\Gamma(y) \cap U \neq \emptyset$ for every $y \in V$. If this property holds for all $p \in P$ the correspondence is said to be **lower hemicontinuous**.*

Definition 32 *A correspondence that is both upper and lower hemicontinuous is called **continuous**.*

Loosely speaking, an upper hemicontinuous correspondence does not "collapse" at limit points and a lower hemicontinuous one does not "explode." These conditions ensure that the boundaries

of the image sets move in a continuous fashion; for example, they ensure that the affordable bundle set does not change in a discontinuous fashion when prices are changed continuously. It is straightforward to prove that a correspondence that is single-valued and either uhc or lhc is continuous. The definitions above are admittedly rather abstract; more intuitive definitions can be stated in terms of sequences (under some circumstances).

Definition 33 *Let $P \subset \mathcal{R}^n$ and $Y \subset \mathcal{R}^m$. A correspondence $\Gamma : P \rightrightarrows 2^Y$ is **compact-valued at** $p \in P$ if $\Gamma(p)$ is compact. If this property holds for all $p \in P$ the correspondence is said to be **compact-valued**. Similar statements define the terms **open-valued**, **closed-valued**, and **convex-valued**.*

Theorem 34 *$\Gamma : P \rightrightarrows 2^Y$ is lower hemicontinuous at $p \in P$ if and only if $\forall \{p_n\} \to p$ and $\forall x \in \Gamma(p)$, there exists $\{x_n\}$ with $x_n \in \Gamma(p_n)$ such that $\{x_n\} \to x$.*

Proof. *Let Γ be lhc at p and let $p_n \to p$ and $x \in \Gamma(p)$. For each integer r, let $B_r(x)$ denote the ball with radius $\frac{1}{r}$ and center x. Since Γ is lhc and p, there exists $\forall r$ a neighborhood $V_r(x)$ such that $z \in V_r$ implies $\Gamma(p) \cap B_r(x) \neq \emptyset$. Since $p_n \to p$ there is for each r an integer n_r such that $n \geq n_r$ implies $p_n \in V_r$. Clearly we can assume $n_r < n_{r+1}$.*

We now define the desired sequence x_n. For n with $n_r < n < n_{r+1}$ choose $x_n \in \Gamma(p_n) \cap B_r(x)$, which is nonempty from above. This sequence satisfies $x_n \to x$.

To prove the converse, assume that $\Gamma(p)$ is not lhc at p; that is, there exists an open set O with $O \cap \Gamma(p) \neq \emptyset$ such that every neighborhood $V(x)$ contains a point z_V with $\Gamma(z_V) \cap O = \emptyset$. Therefore, there exists a sequence $p_n \to p$ with $\Gamma(p_n) \cap O = \emptyset$. Let $x \in O \cap \Gamma(p)$. By assumption, there exists $x_{nq} \to x$ with $x_{nq} \in \Gamma(p_{nq})$. Since O is open and $x \in O$ we have for large enough q that $x_{nq} \in O$. Thus, $\Gamma(p_{nq}) \cap O \neq \emptyset$, which is a contradiction. ■

Theorem 35 *Let $\Gamma : P \rightrightarrows 2^Y$ be compact-valued. Then $\Gamma : P \rightrightarrows 2^Y$ is upper hemicontinuous at $p \in P$ if and only if $\forall \{p_n\} \to p$ and $\forall \{x_n\}$ with $x_n \in \Gamma(p_n)$, there exists a convergent subsequence with limit $x \in \Gamma(p)$.*

Proof. *Let Γ be uhc at $p \in P$. First we show that the sequence $\{x_n\}$ is bounded and therefore possesses a convergent subsequence. Then we show that the limit belongs to $\Gamma(p)$.*

Since $\Gamma(p)$ is bounded, there exists a bounded and open set B such that $\Gamma(p) \subset B$. By uhc, there exists neighborhood V of p such that $\Gamma(p) \subset B$ $\forall z \in V$. Since the sequence p_n converges

to p there exists N such that $p_n \in V\ \forall n \geq N$. Thus, x_n is bounded. Therefore there exists a convergent subsequence $\{x_{nq}\} \to x$.

Assume $x \notin \Gamma(p)$. Clearly, there exists a closed neighborhood of $\Gamma(p)$ not containing x, such as the closed ball B_ϵ around $\Gamma(p)$ with radius $\epsilon > 0$, where ϵ is smaller than the distance between x and any point $z \in \Gamma(p)$. Since Γ is uhc, it follows that for n large enough we have $\Gamma(p_n) \subset B_\epsilon$ so that $x_n \in B_\epsilon$. Since $x_{nq} \to x$ and B_ϵ is closed, $x \in B_\epsilon$. Contradiction.

To prove the converse, assume that $\Gamma(p)$ is not uhc at $p \in P$; that is, there exists an open set O containing $\Gamma(p)$ such that every neighborhood V of p contains a point z_V with $\Gamma(z_V) \nsubseteq O$. By choosing the sequence of neighborhoods $B_{1/n}(p)$ we obtain a sequence $p_n \to p$ and a sequence $x_n \in \Gamma(p_n)$ with $x_n \notin O$. We know there exists a subsequence of x_n whose limit belongs to $\Gamma(p)$. Since $2^Y \backslash O$ is closed, this is impossible. Therefore, $p_n \in 2^Y \backslash O\ \forall n$, implying that the limit point of no converging subsequence of x_n will belong to O. Since $\Gamma(p) \subset O$, we have proven the contradiction. ■

We now state a parametric optimization problem:

$$\max_{x \in X} \{f(x,p) : x \in \Gamma(p)\} \tag{17}$$

where p is a set of parameters (prices, income, etc.). The goal is to characterize how the solution depends on p. Let

$$v(p) = \max_{x \in X} \{f(x,p) : x \in \Gamma(p)\} \tag{18}$$

denote the maximized value of f at parameter vector p and

$$\begin{aligned} g(p) &= \{x^* \in \Gamma(p) : f(x^*,p) \geq f(x,p) \forall x \in \Gamma(p)\} \\ &\equiv \underset{x \in X}{\operatorname{argmax}} \{f(x,p) : x \in \Gamma(p)\} \end{aligned}$$

be the set of maximizers (the maximizer correspondence, also called the policy correspondence). We then have the following theorem:

Theorem 36 *(**Berge's Maximum Theorem**) Let $X \subseteq \mathcal{R}^l$ and $Y \subseteq \mathcal{R}^m$. Let $f : X \times Y \to \mathcal{R}$ be continuous. Let $\Gamma : X \rightrightarrows Y$ be a nonempty, continuous, and compact-valued correspondence. Let*

$$v(p) \equiv \max_{x \in \Gamma(p)} \{f(x,p)\}$$

denote the value function and

$$g(p) \equiv \{x \in \Gamma(p) : v(p) = f(x,p)\}$$

denote the maximizer correspondence. Then $v(p)$ is continuous and $g(p)$ is nonempty, compact-valued, and upper hemicontinuous.

Proof. First, we show that $g(p)$ is nonempty and compact-valued. Fix p. $g(p)$ is then nonempty because $f(x,p)$ is continuous and $\Gamma(p)$ is compact. To show compactness of $g(p)$ we need only show that it is both closed and bounded. $g(p)$ must be bounded because $g(p) \subseteq \Gamma(p)$ and $\Gamma(p)$ is compact. To show $g(p)$ is closed, let $\{x_n\} \subseteq g(p)$ be such that $x_n \to x$. Then $v(p) = f(x_n, p)$ $\forall n$. Since f is continuous, $v(p) = f(x,p)$. Therefore, $x \in g(p)$.

To show that $g(p)$ is upper hemicontinuous, we need to show that it is uhc at some fixed p. We apply the sequential definition of uhc. Since $x_n \in g(p_n) \subseteq \Gamma(p_n)$ and Γ is uhc at p, we have that $\exists \{x_{nk}\} \subseteq \{x_n\}$ such that $x_{nk} \to x \in \Gamma(p)$. We need only show that $x \in g(p)$, meaning that

$$f(x,p) \leq f(z,p) \ \ \forall z \in \Gamma(p).$$

Since $x_n \in g(p_n)$, we have

$$f(z,p_n) \leq f(x_n,p_n) \ \ \forall z \in \Gamma(p_n).$$

But f is continuous and $p_n \to p$, $x_n \to x$, so

$$f(z,p) \leq f(x,p) \ \ \forall z \in \Gamma(p).$$

Finally, we show that $v(p)$ is continuous. That is, we need to show that $v(p_n) \to v(p)$ $\forall \phi_n \to p$. Equivalently, we want to show that

$$v(x) = \underline{h} = \overline{h}$$

where

$$\underline{h} = \liminf \{v(p_k) : k \geq n\}$$
$$\overline{h} = \limsup \{v(p_k) : k \geq n\}.$$

Let $\{x_n\}$ be such that $x_n \in g(p_n)$ $\forall n$. Consider $\{f(x_n,p_n)\}$ (which is just $v(p_n)$). There exists $\{p_{nk}\} \subseteq \{p_n\}$ such that $f(x_{nk},p_{nk}) \to \overline{h}$ as $k \to \infty$. Since $x_{nk} \in g(p_{nk})$ and g is uhc, we have

$$\left\{p_{nk_q}\right\} \subset \{p_{nk}\}$$

such that $x_{nk_q} \to x \in g(p)$ as $q \to \infty$. Therefore, $\overline{h} = \lim \left\{ f\left(x_{nk_q}, p_{nk_q}\right) \right\} = f(x,p) = v(p)$. In the same way we can show that $\underline{h} = v(x)$. Therefore, v is continuous. ■

Therefore, the maximized value – the indirect utility function from microtheory – is continuous in the parameters of the problem and that the set of maximizers is at the very least upper hemi-continuous – it does not get nontrivially smaller as we move continuously through the parameter space (note: it may not be lower hemicontinuous, so it could explode).

Finally, two concepts concerning correspondences – the notions of a selection and the graph.

Definition 37 *The **graph** of a correspondence Γ is the set*

$$A = \{(x,y) \in X \times Y : y \in \Gamma(x)\}. \tag{19}$$

Definition 38 *A **selection** g from the correspondence Γ is a function $g : X \to A$ such that $g(x) \in \Gamma(x)$ for all $x \in X$.*

The graph is simply what you would draw on paper to represent the correspondence. A selection is simply a "path" running through the correspondence that picks out one value from $\Gamma(x)$ for each x. An important property of continuous correspondences is that there will always exist a continuous selection from them. In other words, even if our demand correspondence has many values for some given set of prices and income, we can always pick a continuous demand function from it.

Lemma 39 *Let $X \subset \mathcal{R}^l$ and $Y \subset \mathcal{R}^m$. Assume that the correspondence $\Gamma : X \rightrightarrows Y$ is nonempty, compact- and convex-valued, and continuous, and let A be the graph of Γ. Assume that $f : A \to \mathcal{R}$ is continuous and $f(x,\cdot)$ is strictly concave $\forall x \in X$. Define*

$$g(x) = \underset{y \in \Gamma(x)}{\operatorname{argmax}} \{f(x,y)\}.$$

Then $\forall \epsilon > 0$ and $x \in X$, there exists $\delta_x > 0$ such that $y \in \Gamma(x)$ and $|f(x,g(x)) - f(x,y)| < \delta_x$ implies $||g(x) - y|| < \epsilon$. If X is compact, then $\delta > 0$ is independent of x.

Proof. *Under the stated assumptions g is a continuous single-valued function. For the case where X is compact, A is compact as well. $\forall \epsilon > 0$ define*

$$A_\epsilon = \{(x,y) \in A : ||g(x) - y|| \geq \epsilon\}.$$

If $A_\epsilon = \emptyset\ \forall \epsilon > 0$, then Γ is single-valued and the result is trivial. Otherwise, there exists $\widehat{\epsilon} > 0$ sufficiently small such that $0 < \epsilon < \widehat{\epsilon}$, the set A_ϵ is nonempty and compact. For any such ϵ, let

$$\delta = \min_{(x,y) \in A_\epsilon} \{|f(x, g(x)) - f(x, y)|\}.$$

The function being minimized is continuous and the constraint set is compact, so δ exists. Since $(x, g(x)) \notin A_\epsilon\ \forall x \in X$, it follows that $\delta > 0$. Therefore, $y \in \Gamma(x)$ and $||g(x) - y|| \geq \epsilon$ implies $||f(x, g(x)) - f(x, y)|| \geq \delta$. If X is not compact, this argument is applied to each $x \in X$ separately. ■

Proposition 40 *Let X, Y, Γ, and A be defined as in the previous Lemma. Let $\{f_n\}$ be a sequence of continuous real-valued functions on A; assume that for each n and each $x \in X$, $f_n(x, \cdot)$ is strictly concave in the second argument. Assume that f has the same properties and that $f_n \to f$ uniformly in the* sup *norm. Define the functions*

$$g_n(x) = \underset{y \in \Gamma(x)}{\operatorname{argmax}} \{f_n(x, y)\}$$

and

$$g(x) = \underset{y \in \Gamma(x)}{\operatorname{argmax}} \{f(x, y)\}.$$

Then $g_n \to g$ pointwise. If X is compact, $g_n \to g$ uniformly.

Proof. *g_n is the unique maximizer of f_n on Γ and g is the unique maximizer for f. It follows that*

$$\begin{aligned} 0 &\leq f(x, g(x)) - f(x, g_n(x)) \\ &\leq f(x, g(x)) - f_n(x, g(x)) + f_n(x, g_n(x)) - f(x, g_n(x)) \\ &\leq 2\,||f - f_n|| \quad \forall x \in X. \end{aligned}$$

Since $f_n \to f$ uniformly, it follows that $\forall \delta > 0$ there exists $M_\delta \geq 1$ such that

$$0 \leq f(x, g(x)) - f(x, g_n(x)) \leq 2\,||f - f_n|| < \delta$$

holds for all $x \in X$ and all $n \geq M_\delta$.

To show that $g_n \to g$ pointwise, we must establish that $\forall \epsilon > 0$ and $x \in X$ there exists $N_x \geq 1$ such that

$$||g(x) - g_n(x)|| < \epsilon$$

holds for all $n \geq N_x$. *By the previous lemma, we need only show that* $\forall \delta_x > 0$ *and* $x \in X$ *there exists* $N_x \geq 1$ *such that*

$$|f(x, g(x)) - f(x, g_n(x))| < \delta_x$$

$\forall n \geq N_x$. *Any* $N_x \geq M_\delta$ *satisfies this condition.*

Suppose X *is compact. To establish* $g_n \to g$ *uniformly, we must show that for each* $\epsilon > 0$ *there exists* $N \geq 1$ *such that*

$$||g(x) - g_n(x)|| < \epsilon$$

holds for all $x \in X$. *Using the previous lemma, we again need only show that* $\forall \delta > 0$ *and* $x \in X$ *there exists* $N \geq 1$ *such that*

$$|f(x, g(x)) - f(x, g_n(x))| < \delta$$

$\forall n \geq N$. *Again, any* $N \geq M_\delta$ *will satisfy this condition.* ■

These final two results show that given a sequence of functions that converges uniformly, the sequence of maximizers for that function also converges. Compactness of the domain delivers uniform convergence.

Provided that f is differentiable we can obtain the derivative of the value function using the familiar envelope theorem.

Theorem 41 ***(Envelope Theorem)*** *Let* $f : X \times P \to \mathcal{R}$ *be continuous and differentiable and* X *and* P *be compact and convex. Let* $v(p)$ *be the maximized value of*

$$\max_{x \in X} \{f(x, p)\}. \tag{20}$$

Then

$$Dv(p) = D_p f(x, p). \tag{21}$$

Proof. *Assuming that the first-order conditions are necessary and sufficient for an optimum, we know that the solution must satisfy*

$$D_x f(x, p) = 0. \tag{22}$$

Using the implicit function theorem we assert that there is some function $x = g(p)$ *that solves this equation. Inserting this function into our objective we obtain*

$$v(p) = f(g(p), p). \tag{23}$$

Since this equation holds for any p we differentiate to obtain

$$Dv(p) = D_x f(g(p), p) Dg(p) + D_p f(g(p), p). \tag{24}$$

By the first-order conditions (which hold for each p) the first term is zero, yielding

$$Dv(p) = D_p f(g(p), p). \tag{25}$$

■

Most problems in economics are constrained optimization problems, so we proceed to define the problem

$$\max_{x \in X} \{f(x, p)\} \tag{(P)}$$

subject to

$$g(x, p) \geq 0 \tag{26}$$

where $f : X \times P \to \mathcal{R}$ and $g : X \times P \to \mathcal{R}^m$. We will assume that X is nonempty and convex. Note that it is always possible to rewrite problems in this fashion (for minima, note that $\min\{f(x)\} = \max\{-f(x)\}$ and everything goes through as before). Define the Lagrangian as

$$L : X \times P \times \mathcal{R}^m_+ \to \mathcal{R} \tag{27}$$

such that

$$L(x, p, \lambda) = f(x, p) + \lambda^T g(x, p). \tag{28}$$

The components of the vector λ are referred to as "Lagrange multipliers" and must be nonnegative in this case (for minima they must be nonpositive). At the optimum we have the **saddle-point-property**:

Theorem 42 *(**Lagrangian Saddle Point Theorem**) Let f, g^j, for $j = 1, 2, ..., m$ be concave in x and assume that there exists $x^+ \in X$ such that $g(x^+, p) \gg 0$ for all p. Then $x^* \in X$ is a solution to the problem if and only if there exists $\lambda^* \in \mathcal{R}^m_+$ such that (x^*, λ^*) satisfies*

$$L(x, \lambda^*) \leq L(x^*, \lambda^*) \leq L(x^*, \lambda) \tag{(S)}$$

for all $x \in X$, $\lambda \in \mathcal{R}^m_+$.

Proof. (a) First we show that if $\left(x^{i*}, \lambda^*\right)$ satisfies (S), then x^* solves (P).

Let

$$L\left(x, p, \lambda\right) = f\left(x, p\right) + \lambda g\left(x, p\right)$$

and assume that

$$f\left(x, p\right) + \lambda^* g\left(x, p\right) \leq f\left(x^*, p\right) + \lambda^* g\left(x^*, p\right) \leq f\left(x^*, p\right) + \lambda g\left(x^*, p\right)$$

$\forall x \in X$ and $\lambda \geq 0$. The second inequality implies

$$\lambda^* g\left(x^*, p\right) \leq \lambda g\left(x^*, p\right) \qquad ((*))$$

$\forall \lambda \geq 0$.

Observe that $g\left(x^*, p\right) \geq 0$. If not, there exists j with $g^j\left(x^*, p\right) < 0$. Then pick λ^j large enough such that $\lambda^* g\left(x^*, p\right) > \lambda g\left(x^*, p\right)$, which is a contradiction. Hence $\lambda^* g\left(x^*, p\right) \geq 0$ while $\lambda = 0$ in (*) implies $\lambda^* g\left(x^*, p\right) \leq 0$, so $\lambda^* g\left(x^*, p\right) = 0$.

The first inequality implies

$$f\left(x, p\right) + \lambda^* g\left(x, p\right) \leq f\left(x^*, p\right)$$

$\forall x \in X$. If $g\left(x, p\right) \geq 0$ then $\lambda^* g\left(x, p\right) \geq 0$ so that $f\left(x\right) \leq f\left(x^*\right)$ $\forall x \in X$ with $g\left(x\right) \geq 0$. Therefore, x^* solves (P).

(b) Now we show that if x^* solves (P) then there exists $\lambda^* \geq 0$ such that (x^*, λ^*) satisfies (S).

By concavity of f and g, the set

$$K = \left\{z \in \mathcal{R}^{m+1} : z \leq \left(f\left(x, p\right), g\left(x, p\right)\right) \ \forall x \in X\right\}$$

is convex. Consider the point $z^* = \left(f\left(x^*, p\right), 0\right)$. Then $z^* \in K$ but $z^* \notin Int\left(K\right)$, since $\left(f\left(x^*, p\right), 0\right) \leq \left(f\left(x^*, p\right), g\left(x^*, p\right)\right)$ by assumption. Hence, we can separate z^* and K: there exists $r \in \mathcal{R}^{m+1}$ such that $r \neq 0$ and $rz^* \geq rz$ $\forall z \in K$. Clearly $r > 0$, since if not then $r_i < 0$ and we can pick $z \in K$ with $z_i < 0$ such that $rz > rz^*$, which is a contradiction. Moreover, if $r = (r'', r')$ with $r'' \in \mathcal{R}_+$ then $r'' \neq 0$ (if not, necessarily $r' > 0$ and take x^+ with $g\left(x^+\right) >> 0$ and $z^+ = \left(f\left(z^+, p\right), g\left(z^+, p\right)\right)$; then $z^+ \in K$ and $rz^+ = r' g\left(x^+\right) > 0$ while $r' z' \leq 0$ $\forall z' \in K$, a contradiction).

Let $\lambda^* = r'/r'' \geq 0$. Rewriting $rz^* \geq rz \ \forall z \in K$ where $z = (f(x,p), g(x^*,p))$ and $x \in X$, we have

$$r''f(x^*,p) \geq r''f(x,p) + r'g(x^*,p)$$

or

$$f(x^*,p) \geq f(x,p) + \lambda^* g(x^*,p) \qquad ((**))$$

$\forall x \in X$. Now $\lambda^* \geq 0$ and $g(x^*,p) \geq 0$ implies $\lambda^* g(x^*,p) \geq 0$ while choosing $x = x^*$ above yields $\lambda^* g(x^*,p) \leq 0$, so $\lambda^* g(x^*,p) = 0$. We can rewrite (**) as

$$f(x^*,p) + \lambda^* g(x^*,p) \geq f(x,p) + \lambda^* g(x^*,p)$$

$\forall x \in X$. But $\lambda \geq 0$ and $g(x^*,p) \geq 0$ imply

$$f(x^*,p) + \lambda g(x^*,p) \geq f(x^*,p) + \lambda^* g(x^*,p)$$

$\forall \lambda \in \mathcal{R}^m_+$. Combining these expressions yields (S). ■

Intuitively the Lagrange multiplier captures the "tightness" of the constraint it is attached to – the more positive the multiplier the more binding the constraint is at the solution. Thus, the saddlepoint property says, at the optimum, changing x makes our objective smaller but relaxing a constraint (making λ_j smaller) increases it. The caveat is that the constraint set must contain an interior point; that is, it must be possible to move "inside" the constraint set. For example, in a plane if the constraint set were a line this theorem would not hold, because one cannot "relax" the constraint. This interiority requirement is called Slater's condition and it plays an important role below in the Kuhn-Tucker theorem.

One can therefore write the constrained optimization problem using the saddle-point representation

$$\max_x \left\{ \min_\lambda \{L(x,p,\lambda)\} \right\}. \qquad (29)$$

The envelope theorem goes through to this formulation (without loss of generality, we drop any constraints that do not bind).

Proposition 43 ***(Envelope Theorem)*** *Let* $x^*(p) \in \text{argmax}_{x \in X} \{f(x,p)\}$ *subject to* $g^i(x,p) = 0 \ \forall i \in \{1,...,m\}$. *Let*

$$v(p) \equiv f(x^*(p), p).$$

Then

$$Dv(p) = D_p f(x^*(p), p).$$

The Kuhn-Tucker conditions characterize a constrained local maximum under certain regularity conditions and are given by

$$\begin{aligned} \nabla f(x) + \sum_{j=1}^{r} \lambda_j \nabla g_j(x)^T + \sum_{j=1}^{s} \mu_j \nabla h_j(x)^T &= 0 \\ \lambda_j &\geq 0 \\ g_j(x) &\geq 0 \\ \lambda_j g_j(x) &= 0 \\ h_j(x) &= 0; \end{aligned}$$

note that μ_j is not restricted in sign. The regularity conditions are called "Constraint Qualification" conditions; there are a number of these conditions, but we will use a simple one called "Slater's Condition" that applies to concave programs.

Condition 44 *(Slater's Constraint Qualification) The problem must satisfy*

1. $\exists x^0 \in \mathcal{R}^n$ *such that*

$$\begin{aligned} g_j\left(x^0\right) &> 0 \quad \forall j = 1, ..., r \\ h_j\left(x^0\right) &= 0 \quad \forall j = 1, ..., s; \end{aligned}$$

   that is, the constraint set has an interior point;

2. $\forall x \in D$, *the vectors* $\nabla h_j(x)$, $j = 1, ..., s$ *are linearly independent.*

We also need the following definitions.

Definition 45 *A function* $f : \mathcal{R}^n \to \mathcal{R}$ *is* ***quasiconcave*** *if* $f(\lambda x + (1-\lambda) y) \geq \min\{f(x), f(y)\}$ $\forall x, y \in \mathcal{R}^n$ *and* $\lambda \in [0, 1]$. *A function* $f : \mathcal{R}^n \to \mathcal{R}$ *is* ***quasiconvex*** *if* $-f(x)$ *is quasiconcave.*

Definition 46 *A function* $f \in C^1(\mathcal{R}^n, \mathcal{R})$ *is* ***pseudoconcave*** *if it is quasiconcave and* $\forall x, y \in \mathcal{R}^n$ *with* $f(x) > f(y)$, $Df(y)(x - y) > 0$. *A function* $f \in C^1(\mathcal{R}^n, \mathcal{R})$ *is* ***pseudoconvex*** *if* $-f(x)$ *is pseudoconcave.*

Definition 47 *A function $f : \mathcal{R}^n \to \mathcal{R}$ is **quasilinear** if $f(x)$ and $-f(x)$ are quasiconcave. A function $f \in C^1(\mathcal{R}^n, \mathcal{R})$ is **pseudolinear** if $f(x)$ and $-f(x)$ are pseudoconcave.*

Pseudoconcave functions need not be concave, but they "behave like" concave functions with respect to local maxima; if f is pseudoconcave then every point where $Df(x) = 0$ is a local maximum and vice versa. For example, $f(x) = -x^3 - x$ is pseudoconcave but not concave (it has no points where $Df(x) = 0$ and no local maxima, so it trivially satisfies the condition). Similarly, quasiconcave functions need not be pseudoconcave. Consider $f(x) = -x^3$, which is quasiconcave (it is a univariate monotone function); $Df(0) = 0$ but that point is not a local maximum.

Now we can state the Kuhn-Tucker theorem.

Theorem 48 *(Kuhn-Tucker Theorem) Let f and g_j, $j = 1, ..., r$, be pseudoconcave and C^1 and the functions h_j, $j = 1, ..., s$, be pseudolinear. Suppose that Slater's CQ holds. If x^* is an optimal solution of the NLP then $\exists \lambda^* \in \mathcal{R}^r_+$ and $\mu^* \in \mathcal{R}^s$ such that the KT conditions hold at (x^*, λ^*, μ^*). Conversely, if (x^*, λ^*, μ^*) satisfies the KT conditions then x^* is an optimal solution to the NLP.*

The KT Theorem can be generalized slightly to **incave/invex** functions (invariant concave/convex functions).

Definition 49 *A function $f \in C^1(\mathcal{R}^n, \mathcal{R})$ is **incave** if there exists a vector function $\eta : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$ such that $\forall x, y \in \mathcal{R}^n$,*

$$f(x) - f(y) \leq \eta(x, y) Df(y).$$

*A function $f \in C^1(\mathcal{R}^n, \mathcal{R})$ is **invex** if $-f$ is incave.*

Definition 50 *A function $f \in C^1(\mathcal{R}^n, \mathcal{R})$ is a **Type 1 incave objective** at x_0 if there exists a vector function $\eta : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$ such that*

$$f(x) - f(x_0) \leq \eta(x) \cdot Df(x_0).$$

*A function $f \in C^1(\mathcal{R}^n, \mathcal{R})$ is a **Type 1 invex objective** at x_0 if $-f$ is a Type 1 incave objective at x_0. A function $g \in C^1(\mathcal{R}^n, \mathcal{R})$ is **Type 1 incave constraint** at x_0 if there exists a vector function $\eta : \mathcal{R}^n \times \mathcal{R}^n \to \mathcal{R}^n$ such that*

$$-g(x_0) \geq \eta \cdot Dg(x_0).$$

A function $g \in C^1(\mathcal{R}^n, \mathcal{R})$ is a ***Type 1 invex constraint*** *at x_0 if $-g$ is a Type 1 incave constraint at x_0.*

Similar to pseudoconcave functions, incave functions have the property that every stationary point (a point where the derivative is zero) is a global maximum; that is, $f(x^*) > f(x)$ iff $Df(x^*) = 0$. Type 1 incave functions are incave with respect to a particular point; the KT theorem then only requires (f, g) be Type 1 incave with respect to a point that satisfies the KT conditions, and does not require that property hold at other points. Note that pseudoconcave functions are incave since we can take $\eta(x, y) = x - y$, and therefore concave functions are incave as well.

A weaker constraint qualification condition is called Linear Independence Constraint Qualification (LICQ), which does not require concavity.

Condition 51 *(Linear Independence Constraint Qualification) Denote the active constraints at x^* by $g_j^*(x)$, $j = 1, ..., r^*$. The vectors $\nabla h_j(x)$, $j = 1, ..., s$ and $\nabla g_j^*(x)$, $j = 1, ..., r^*$, are linearly independent.*

Even weaker CQ conditions exist, but they are difficult to verify and the Lagrange multipliers may not be unique. One of the weakest is Quasi-Normal Constraint Qualification (QNCQ), which involves the conditions where LICQ does not apply (the gradients are linearly dependent).

Condition 52 *(Quasi-Normal Constraint Qualification) If the gradients of the active constraints at x^* are linearly dependent with multipliers $\lambda_j \geq 0$ and μ_j, then there does not exist a sequence $x_n \to x^*$ such that $\mu_j \neq 0$ implies $\mu_j h_j(x_n) > 0$ and $\lambda_j \neq 0$ implies $\lambda_j g_j(x_n) > 0$.*

If some CQ condition does not hold, then the *Fritz John conditions* can be applied to charac-

terize the optimum. They are

$$\lambda_0 \nabla f(x) + \sum_{j=1}^{r} \lambda_j \nabla g_j(x)^T + \sum_{j=1}^{s} \mu_j \nabla h_j(x)^T = 0$$

$$\lambda_j \geq 0$$

$$g_j(x) \geq 0$$

$$\lambda_j g_j(x) = 0$$

$$h_j(x) = 0$$

$$\lambda_0 \geq 0.$$

If $\lambda_0 > 0$ then it can be set to 1 without loss of generality, so the important case is when $\lambda_0 = 0$; CQ holding rules out the possibility that $\lambda_0 = 0$. Note that $\lambda_0 = 0$ implies that the FJ conditions would hold for any function $f(x)$. Principal-agent problems where the constraint set involves a maximization by the agent can often fail CQ.

Theorem 53 *(Fritz John's Theorem) Let f and g_j, $j = 1, ..., r$, be pseudoconcave and the functions h_j, $j = 1, ..., s$, be pseudolinear. If x^* is an optimal solution of the NLP then $\exists \lambda^* \in \mathcal{R}_+^{r+1}$ and $\mu^* \in \mathcal{R}^s$ such that (x^*, λ^*, μ^*) solve the FJ conditions. Conversely, if (x^*, λ^*, μ^*) satisfies the FJ conditions then x^* is an optimal solution to the NLP.*

As an example, consider the problem

$$\min_{x_1, x_2} \{-x_2\}$$

subject to

$$-x_1^2 - x_2 \leq 0$$

$$x_1^{12} + x_2^3 \leq 0.$$

The KT conditions are

$$-2\lambda_1 x_1 + 12 x_1^{11} = 0$$

$$-1 - \lambda_1 + 3\lambda_2 x_2^2 = 0$$

plus the complementarity slackness conditions

$$\lambda_1 \left(-x_1^2 - x_2\right) = 0$$
$$\lambda_2 \left(x_1^{12} + x_2^3\right) = 0.$$

Since the constraints require $x_2 \leq 0$ and the objective wants x_2 as large as possible, the solution is $(0,0)$. But at $(0,0)$ $\lambda_1 < 0$, so the KT conditions are not necessary for the solution. In contrast, the FJ conditions

$$-2\lambda_1 x_1 + 12x_1^{11} = 0$$
$$-\lambda_0 - \lambda_1 + 3\lambda_2 x_2^2 = 0$$

work because we can set $\lambda_0 = 0$, implying that $\lambda_1 = 0$. We have failed Slater's condition, because there is no point (x_1, x_2) such that

$$-x_1^2 - x_2 < 0$$
$$x_1^{12} + x_2^3 < 0;$$

instead, the two constraints have only the point $(0,0)$ in common, it (obviously) occurs on the boundary since the sets are closed, and also obviously must be the optimum. Therefore, CQ does not hold and the KT conditions are not necessary.

There are generalizations of the Kuhn-Tucker and Fritz John conditions to certain kinds nondifferentiable functions, namely those with kinks (a **kink** is a point at which the directional derivatives both exist but are not equal); these conditions require the notion of superdifferentials and subdifferentials.

Definition 54 *Suppose $f : \mathcal{R}^n \to \mathcal{R}$ is concave; a **supergradient** of f at x_0 is a vector v such that*

$$f(x) - f(x_0) \leq v \cdot (x - x_0)$$

*$\forall x \in \mathcal{R}^n$. The set of all supergradients at x_0 is called the **superdifferential** of f at x_0 and denoted $\partial f(x_0)$. Similarly, suppose $f : \mathcal{R}^n \to \mathcal{R}$ is convex; a **subgradient** of f at x_0 is a vector v such that*

$$f(x) - f(x_0) \geq v \cdot (x - x_0)$$

$\forall x \in \mathcal{R}^n$. *The set of all subgradients at* x_0 *is called the* ***subdifferential*** *of* f *at* x_0 *and denoted* $\partial f(x_0)$.

If the superderivative is a singleton, then f is differentiable at x_0; if not, then the superdifferential will be a convex set. For a concave function, a point x_0 is a global maximum if and only if $0 \in \partial f(x_0)$; this statement generalizes the result that the derivative is zero at the global maximum of a differentiable concave function. The generalized KT theorem then says that, under appropriate regularity conditions, an optimal point has zero in the superdifferential and any point where zero is an element of the superdifferential is an optimal point; that is, we replace the derivative condition with

$$0 \in \partial f(x) + \sum_{j=1}^{r} \lambda_j \partial g_j(x)^T + \sum_{j=1}^{s} \mu_j \partial h_j(x)^T .$$

There are still problematic functions, as the following example shows. Consider $f(x) = \sqrt[3]{x^2}$. At $x = 0$ the directional derivatives equal $-\infty$ (in the negative direction) and ∞ (in the positive direction); this kind of kink is called a **cusp**. While the minimum occurs at $x = 0$ and the subgradient is a singleton, the function is not Type I incave and it fails all CQ conditions (as far as I know). If you encounter such a function, and we will later in the course, it may need to be handled by special tools.

The KT conditions have a strange property that makes it unwise to use them as a criterion for convergence of your numerical solution – depending on how you converge to them, the error function may be close to discontinuous. The error function is given by

$$\left\| \nabla f(x) + \sum_{j=1}^{r} \lambda_j \nabla g_j(x)^T + \sum_{j=1}^{s} \mu_j \nabla h_j(x)^T \right\|$$

and it clearly should be zero at a solution; the question is then can we terminate a search if the error has gotten small? Another way to put it is – is the error a monotonic function of the distance to a KT point? As shown in Dutta, Deb, Tulshyan, and Arora (2010), the answer is

generally no. To understand why, let's define an ϵ-KT point as a value of x such that

$$\left\| \nabla f(x) + \sum_{j=1}^{r} \lambda_j \nabla g_j(x)^T + \sum_{j=1}^{s} \mu_j \nabla h_j(x)^T \right\| < \epsilon$$
$$\lambda_j \nabla g_j(x)^T = 0$$
$$\mu_j \nabla h_j(x)^T = 0.$$

That is, complementary slackness is satisfied exactly and the derivative condition is almost satisfied. They give an example where the error is actually increasing along a path converging to x, but arbitrarily close to it the error drops suddenly and rapidly:

$$\min_{x,y} \left\{ x^2 + y^2 - 10x + 4y + 2 \right\}$$

subject to

$$x^2 + y - 6 \leq 0$$
$$x - y \leq 0$$
$$-x \leq 0.$$

The global optimum is $(x^*, y^*) = (1.5, 1.5)$, and this point satisfies the KT conditions with the second constraint binding:

$$2x - 10 + \lambda_2 = 0$$
$$2y + 4 - \lambda_2 = 0$$
$$x - y = 0$$

implies $\lambda_2 = 7 > 0$. Since the objective is convex and the constraint set is also convex, this point is the global optimum.

Now let's consider the nearby point $(x^+, y^+) = (1.495, 1.505)$, at which no constraints are active and the error is $2.43 > 0$. Consider two paths from (x^+, y^+) to (x^*, y^*), one which follows the path $x = y$ and one that follows $x + y = 3$. For the first, the second constraint becomes active as we approach the solution and the error will decline smoothly to zero; for the second, all constraints are inactive and the error shows a discontinuity at the solution.

The problem is the behavior of the complementary slackness conditions; at points close to a boundary, but not on it, the multipliers must be zero, but once we hit the boundary the multiplier can be any non-negative number. As a result, the gradient of the constraint may suddenly contribute a lot to the error, leading to the discontinuity.

6.1 Turnpike Theorem

Next, I want to point out a property of certain programs with an infinity of variables in x called a turnpike (McKenzie 1975), that will be useful at solving for the maxima of such problems. Suppose our goal is to maximize a infinite sum of concave functions:

$$f(x) = \sum_{t=0}^{\infty} u_t(x_t),$$

where the initial value x_0 is fixed. Consider also an "approximate" finite horizon problem

$$\widehat{f}(x_0, ..., x_T) = \sum_{t=0}^{T} u_t(x_t)$$

where the initial and terminal values of x are fixed at x_0 and x_T. We will define some terms that will be useful at defining optimal paths.

Definition 55 *Let $\{x_t\}$ denote a path starting from x_0. An alternative path $\{\widehat{x}_t\}$* ***catches up*** *to $\{x_t\}$ if*

$$\limsup_T \left\{ \sum_{t=0}^{T} \left(u_t(\widehat{x}_t) - u_t(x_t) \right) \right\} \leq 0$$

as $T \to \infty$. The alternative path $\{\widehat{x}_t\}$ ***overtakes*** *$\{x_t\}$ if*

$$\limsup_T \left\{ \sum_{t=0}^{T} \left(u_t(\widehat{x}_t) - u_t(x_t) \right) \right\} \geq \epsilon$$

for some $\epsilon > 0$.

An optimal path is one that catches up to all other paths that start from the same x_0. A **weakly maximal path** is one that is not overtaken by any alternative path that starts from the same x_0.

Definition 56 *A point y is **reachable** from a path $\{x_t\}$ if there exists an alternative path $\{\widehat{x}_\tau\}_{\tau=t}^{t+n}$ for some $n < \infty$ with $\widehat{x}_t = x_t$ and $\widehat{x}_{t+n} = y$ and $\sum_{t+1}^{t+n} u_t(\widehat{x}_t) > U$, where n may depend on t but U does not. A path $\{y_t\}$ is reachable from a path $\{x_t\}$ if for any t there exists a path that reaches some point y_{t+n}, where again n may depend on t but U does not. A path is **uniformly reachable** if n can be chosen independent of t, and **freely reachable** if U can be chosen dependent on t such that $U(t) \to 0$ as $t \to \infty$.*

Theorem 57 *(Turnpike Theorem) Let $\{x_t\}$ be a weakly maximal path. Suppose also that u_t is strictly concave and the constraint set is convex, and further that $\forall t$ and $\forall \xi > 0$, there exists $|\zeta| < \infty$ such that $|x_t| < \xi$ implies $u_t < \zeta$. Let y be freely reachable from $\{x_t\}$ and $\{x_t\}$ be freely reachable from y. For any $\epsilon > 0$ and any τ_1 there exists $\tau_2 \leq T$ such that if $\{x_t^*\}_{t=0}^T$ is an optimal path with $x_0^* = x_0$ and $x_T^* = y$, then $|x_t - x_t^*| < \epsilon$ $\forall t \leq \tau_1$.*

What does the turnpike theorem imply? Suppose we want to solve the infinite program; obviously in practice we are confined to solving the finite horizon one and relying on some kind of limit argument. The turnpike theorem says that the solution to the finite program "hugs" the solution to the infinite program for some amount of time, independent of the imposed terminal condition x_T, provided T is large enough. Furthermore, as T increases the finite and infinite horizon solutions get closer together and stay close together longer. Thus, if we solve the finite problem for large enough T, we get an arbitrarily-good approximation to the path that solves the infinite horizon problem; the benefit is that we get to impose the terminal condition, rather than have to discover it through an opaque and numerically-troublesome transversality condition. The paths diverge after τ_2, but if this date is sufficiently distant then it plays no role in the behavior of the path for many periods.

Furthermore, if the infinite horizon problem has a stationary point x^* (a value of x_0 such that the optimal path is $x^* = x_0$ forever), then we can approximate this value independent of the values of both x_0 and x_T! The turnpike will move near to this constant value quickly, stay there for a long time, and diverge at the end, no matter what initial and terminal conditions we select. Obviously the transient dynamics are shorter if we pick x_0 and x_T close to the stationary point, and we can create problems if we choose them far from x^* and T is too small because the path never has a chance to get to the turnpike.

6.2 Mountain Pass Theorem

Sometimes we may want to find saddle-points (the optimality condition for a Lagrangian is a saddle-point). The mountain pass theorem provides some conditions under which a function has a saddle-point.

Theorem 58 *(Mountain Pass Theorem) Suppose $f : X \to \mathcal{R}$ is continuously differentiable. Suppose there exist two numbers $R > 0$ and a such that $f(x) \geq a$ for all $x \in S(R) \equiv \{x \in X : \|x\| = R\}$ and that $f(0) < a$ and $f(e) > a$ for some e with $\|e\| > R$. Finally, suppose that f is Palais-Smale compact (every sequence $\{x_n\} \in X$ with $|f(x_n)| < \infty$ and $Df(x_n) \to 0$ has a convergent subsequence in X). Then f has a critical point s not equal to either 0 or a with $f(s) \geq a$. If X has finite dimension, then the conditions can be weakened to $f(0) \leq a$ and $f(e) \geq 0$. If X has infinite dimension and we assume there exist numbers a, r, and R such that $0 < r < R$ and $f(x) \geq a$ for all $x \in S(r, R) \equiv \{x \in X : r < \|x\| < R\}$, then the weakened condition $f(0) \leq a$ and $f(e) \geq 0$ is sufficient. Furthermore, if $f(s) = a$, then s can be chosen in $S(r, R)$.*

An immediate corollary is that if f has two local minima it also has a third critical point, and that critical point is either a saddlepoint (the generic case) or a local maximum (the degenerate case).

The term 'mountain pass' comes from a geometric visualization of the assumptions; namely, that there are two 'valleys' separated by a high ridge of mountains. Somewhere in the mountains is a low elevation 'pass' that connects the two valleys, and it lies at a higher elevation than the valleys but lower than the rest of the ridge. There is a single point that is the "top" of the pass along the path between valleys and the "bottom" of the pass on a path across the ridge. Saddle points are difficult to locate in general, because one can "slide" off the ridge.

Having laid out the mathematical details that describe critical points, we now start the process of finding them on the computer.

6.3 One Dimension

We first discuss one-dimensional unconstrained minimization problems; in many cases, one dimension will be easy and two (or more) will be much harder. The first method is called **golden**

section. To begin, we bracket our minimum:

Definition 59 *A minimum* $f(x^*) \leq f(x)$ *with* $f : \mathcal{R} \to \mathcal{R}$ *is said to be* ***bracketed*** *if* $\exists\,(a, b, c)$ *such that*

$$a < b < c$$

and

$$f(b) < f(a)$$
$$f(b) < f(c).$$

We need three points to bracket a minimum; in this case the minimum occurs somewhere between a and c. We start by choosing a point $d \in (a, c)$; suppose for concreteness that $d \in (b, c)$. We then evaluate $f(d)$; if $f(d) > f(b)$ the new bracket is (a, b, d); otherwise it is (b, d, c). We continue this process until the distance is small; about the best we can do is the square root of machine precision.

What is our rule for generating the point d? Start by noting that b will be some fraction of the interval size away from a:

$$\frac{b-a}{c-a} = w$$
$$\frac{c-b}{c-a} = 1 - w.$$

Our trial point d is also a fraction of the interval size away from b:

$$\frac{d-b}{c-a} = z.$$

Relative to the current bracket size, the new one will either be of length $w + z$ (if the new bracket is (a, b, d)) or length $1 - w$ (if the new bracket is (b, d, c)). To minimize the worst case scenario, we choose z to make these equal. To avoid the situation where the interval is not shrinking at a constant rate, we want to make the spacing proportional; that is, the bracket (a, b, d) or (b, d, c) should remain in constant proportion to (a, b, c). The result is that we need to solve the system of equations

$$\frac{z}{w} = \frac{w}{1-w}$$
$$\frac{z}{1-w-z} = \frac{w}{1-w}$$

We have two equations in the two unknowns (w, z); the solution is

$$w^* = \frac{3-\sqrt{5}}{2} \approx 0.38197.$$

The optimal bracketing interval has its middle point (not its midpoint) b 0.38197 from a and 0.61803 from c; these fractions are the so-called golden ratios. We implement this procedure by testing a new point d which is 0.38197 into the larger of the two intervals $(a, b), (b, c)$. This step shrinks the size of the bracket by 0.61803 each time. We call this 'linear convergence'. If we define the uncertainty about the location of the root by $\epsilon = |c - a|$, then golden section search implies

$$\epsilon_{n+1} = \frac{3-\sqrt{5}}{2}\epsilon_n,$$

so that the size of the bracket shrinks linearly across iterations. Note: we could have used any point in the interval (a, c) as d, but the golden section search test point ensures that the worse-case scenario is "least bad".

If the function has a 'flat spot' at its minimum, then golden section will find one of the points; unfortunately, which one it finds is a function of the initial bracket and is not easily predicted, and it will not find the boundaries of the set (those points will fail the bracketing condition). We can find the boundaries as a constrained optimization problem, but that is for a later section.

Golden section is the worst case scenario – it cannot fail but is quite slow (linear convergence is rather poor, as it produces only one additional '0' digit each iteration). We can improve it using **Brent's method**, in which parabolic interpolation is used to locate test points. Take three points $f(a)$, $f(b)$, and $f(c)$. The critical point of the unique parabola which passes through these points occurs at

$$d = b - \frac{1}{2}\frac{(b-a)^2 [f(b) - f(c)] - (b-c)^2 [f(b) - f(c)]}{(b-a)[f(b) - f(c)] - (b-c)[f(b) - f(a)]}; \tag{30}$$

note that this point may be a minimum or a maximum. Brent's method takes this approach and combines it with golden section search if the minimum candidate d does not lie inside the bracket interval (a, c) or implies a movement from the best value x that is more than half the previous movement (so that the process is converging to something); it also takes a golden section search if the implied critical point d is a maximum. The convergence rate is faster than linear for any parabolic interpolation step, so the actual convergence rate depends on how often we are stuck with golden section steps.

If we have derivative information, we can exploit it using a variant of Brent's method. That is, the derivative at the central point b determines whether the next test point should be taken in (a, b) or (b, c); we take this derivative and the derivative at the second best point and extrapolate to zero using the secant method. We then reject the test point under the same conditions as before.

Faster methods fall under the general category of Newton methods. Suppose we currently have a guess x^n and want to update it towards the minimum; at a local minimum we have $Df(x^*) = 0$, so you can consider the goal to find the zero of the derivative (we will deal with root-finding later). **Newton-Raphson** takes the new guess for the zero to be the zero of a linear approximation to Df at the current guess. By Taylor's theorem, we have

$$Df(x+\delta) \approx Df(x) + D^2 f(x)\,\delta + \frac{D^3 f(x)}{2}\delta^2 + o(|\delta|^3). \tag{31}$$

For small enough δ, we can ignore the quadratic terms so that

$$Df(x+\delta) = 0 \tag{32}$$

implies

$$\delta = -\frac{Df(x)}{D^2 f(x)}. \tag{33}$$

We then construct the iterative procedure

$$x^{n+1} = x^n - \frac{Df(x^n)}{D^2 f(x^n)} \tag{34}$$

to search for the root, where the "Newton step" $-\frac{Df(x^n)}{D^2 f(x^n)}$ consists of a direction $-Df(x^n)$ and a magnitude $\frac{1}{D^2 f(x^n)}$; note that the step is "locally downhill" since it points in the direction away from the gradient. The convergence rate close to the zero can be calculated as follows. Around the root x^* we have

$$f(x^*) = f(x^n) + Df(x^n)(x^* - x^n) + \frac{1}{2}Df^2(\xi)(x^* - x^n)^2$$

where ξ is either in (x^n, x^*) or (x^*, x^n). Since $f(x^*) = 0$, we can rearrange to get

$$\frac{f(x^n)}{Df(x^n)} + (x^* - x^n) = -\frac{D^2 f(x^n)}{2Df(x^n)}(x^* - x^n)^2.$$

Using the definition of x^{n+1}, we then get

$$|\epsilon_{n+1}| = -\frac{\left|D^2 f(x^n)\right|}{2\left|Df(x^n)\right|}\epsilon_n^2$$

where

$$\epsilon_n = x^* - x^n.$$

That is, if we are close enough to a root the convergence is quadratic.

We can use a golden section step if the Newton step leaves the bracket. We need that safeguard because Newton-Raphson can diverge if started sufficiently far from a root; think about what would happen to x^{n+1} at points where $D^2 f(x^n)$ is very close to zero. Furthermore, locally downhill does not mean downhill in general; because it is not a good idea to take infinitessimal steps, moving in the Newton direction may not reduce the objective. Having a golden section backstop prevents these kinds of issues from derailing convergence entirely. Furthermore, Newton-Raphson can cycle indefinitely, even if f is globally concave. If f is doubly concave (that is, $D^2 f$ is concave), then Newton-Raphson can only take a non-monotonic step once (although the first step can be arbitrarily bad).

Also, note that the term "close enough" is not precise – how close we need to be depends on the behavior of the function at potentially many points. Note also that "degenerate" minima – those where $D^2 f(x^*) = 0$, such as $f(x) = x^4$ – slow the convergence of Newton-Raphson to linear. There are modifications that we will see later (in the root-finding section) that improve the behavior of Newton-Raphson in these cases.

6.4 Derivative-Free Methods in Many Dimensions

Optimization in many dimensions is much harder than in one dimension, as we lose the ability to "trap" a minimum. One very simple and fairly robust method is the Nelder-Mead downhill simplex method. This method does not use any information about the shape of the function, it simply looks at function values and infers the right direction to proceed. As a result it can handle, at least in principle, basically any function, including those with kinks, cusps, discontinuities, and other irregularities, although perhaps not quickly.

Definition 60 *A **simplex** $\Delta \in \mathcal{R}^n$ is a subspace consisting of $n+1$ noncollinear points and their connecting facets and edges.*

In one dimension, Δ^1 is a line segment. In two dimensions, Δ^2 is a triangle. In three dimensions Δ^3 is a tetrahedron. Taken to be nondegenerate (in two dimensions, the three points

are not colinear; in three dimensions, the four points are not coplanar), a simplex forms a basis for the space if vectors are defined from one given point.

We start our algorithm off with an initial simplex, denoted Δ_0^n; for robustness Δ_0^n needs to be large, for speed it needs to be small, so there will be a tradeoff. Define the centroid of the simplex by x_o. There are four types of movements – reflections, expansions, contractions, and shrinkages. A **reflection** evaluates the point

$$x_r = x_o + (x_o - x_{n+1}),$$

where x_{n+1} is the point in the current simplex with the highest (worst) function value. If

$$f(x_1) < f(x_r) < f(x_{n+1}),$$

then construct a new simplex by replacing x_{n+1} with x_r. If

$$f(x_r) < f(x_1),$$

then we conduct an **expansion** by checking the point

$$x_e = x_o + 2(x_r - x_o);$$

in this case since the reflection point is better than any other existing point, it may be advantageous to take a large step in that direction. If

$$f(x_e) < f(x_r)$$

then form a new simplex by replacing x_{n+1} with x_e, otherwise replace x_{n+1} with x_r. If $f(x_r) \geq f(x_{n+1})$, then we conduct a **contraction** by checking the point

$$x_c = x_o + \frac{1}{2}(x_{n+1} - x_o).$$

If

$$f(x_c) < f(x_{n+1})$$

then form a new simplex by replacing x_{n+1} with x_c. On rare occasions we need to **shrink** the entire simplex, which is accomplished by replacing all points except x_1 with

$$x_i = x_1 + \frac{1}{2}(x_i - x_1);$$

this type of move only occurs around degenerate minima (where the Hessian of the function would be singular). This process repeats itself until the simplex has contracted sufficiently around a local minimum.

The Nelder-Mead algorithm is very slow, unfortunately, since it does not use the shape of the function to help choose a profitable direction (the only directions it can move are dictated by the location of the points in the simplex currently, and these change slowly because they can change only one at a time); for this reason one really shouldn't use it unless other options have not worked. Other DFO (derivative-free optimization) methods use line searches to find directions of descent (two examples are Powell's method and the Hooke-Jeeves algorithm). The idea of line search algorithms is to take a candidate x_k and a direction vector d, minimize the expression $f(x_k + \lambda d)$ by choosing the scalar λ, then updating the guess to $x_{k+1} = x_k + \lambda^* d$. The various different algorithms essentially only differ by how the line search direction d is selected. For Powell's method, begin with a set of directions equal to the standard basis vectors

$$d_i = e_i.$$

The algorithm then takes the following steps. Given x_0, for $i = 1, ..., n$ (where n is the dimension of x), move x_{i-1} to the minimum in direction d_i and call it x_i. For $i = 1, ..., n-1$, set $d_i = d_{i+1}$ and set $d_n = x_n - x_0$. Move x_n to the minimum of direction d_n and call this new point x_0. Repeat until the function stops decreasing. Some modifications to the basic approach can improve convergence rates for functions with "narrow twisty valleys" that slow down search algorithms because steps are small and keep changing directions.

Hooke-Jeeves uses a sequence of two kinds of moves, exploratory and pattern. An **exploratory move** evaluates the function starting from a base point x_k by moving each coordinate up and down by ϵ; any coordinate move that succeeds at reducing the function value will be retained and the new base point x_{k+1} is constructed. If $x_{k+1} = x_k$ then the step size ϵ is reduced and a new set of exploratory moves are considered. If $x_{k+1} \neq x_k$, then a **pattern step** is attempted, by constructing the pattern point

$$p_k = x_k + 2(x_{k+1} - x_k)$$

and doing exploratory moves around p_k; the idea is that if moving in this direction is good, then moving further may be better. If we get a point better than x_{k+1}, we drop x_{k+1} and construct

x_{k+2} otherwise we abandon the pattern move and do exploratory moves around x_{k+1}. This method is also sometimes called "pattern search".

A better DFO method is NEWUOA (NEW Unconstrained Optimization Algorithm), which constructs a quadratic approximation Q_k by interpolating across different points at which we have already evaluated f and solves it over a trust region (it was also developed by Powell; trust regions are defined below). The key is that, for large dimensional problems, NEWUOA only constructs the quadratic approximation using a small number of evaluations; instead of the $(n+1)(n+2)/2$ points required for a full quadratic approximation, NEWUOA only uses $m = 2n+1$ points. The rest of the required coefficients are computed by minimizing the Frobenius (matrix) norm of $D^2Q_k - D^2Q_{k-1}$, where the Frobenius norm of $n \times m$ matrix A is given by

$$\|A\|_F = \left(\sum_{i=1}^{n} \sum_{j=1}^{m} A_{ij}^2 \right)^{\frac{1}{2}}.$$

It then minimizes the quadratic approximation as usual.

6.5 Derivative-Based Methods

The extension to a multivariate case of Newton-Raphson is straightforward:

$$x^{n+1} = x^n - H^{-1}(x^n) \nabla f(x^n)$$

where H is the Hessian and ∇ is the gradient. As before, we refer to the term $H^{-1}(x^n) \nabla f(x^n)$ as the "Newton step" consisting of a size $H^{-1}(x^n)$ and a direction $-\nabla f(x^n)$. Here, I want to point out two useful properties of the gradient – it is the direction of steepest descent locally and is locally perpendicular to the level set.

Theorem 61 *Let $f \in C^1$ and $\nabla f(x_0) \neq 0$. Then $\nabla f(x_0)$ points in the direction that increases f most rapidly.*

Proof. *Suppose h is a unit vector. Then*

$$\nabla f(x_0) h = \|\nabla f(x_0)\| \cos(\theta),$$

where θ is the angle between $\nabla f(x_0)$ and h. This function is maximized at $\theta = 0$, so that $\nabla f(x_0)$ and h are parallel. ■

Theorem 62 *Let $f \in C^1$ and $x_0 \in \{x : S(x) = k\}$ for some fixed k. Then $\nabla f(x_0) \cdot h = 0 \ \forall h$ that are tangent vectors at x_0 to paths $c(t) \in S$ that start at $c(0) = x_0$ (that is, $\nabla f(x)$ is **normal** to the set S at x_0).*

Proof. *Let $c(t)$ be a path in S. Then $f(c(t)) = k \ \forall t$. Let h be the tangent vector of c at $t = 0$:*

$$\left.\frac{dc}{dt}\right|_{t=0} = h.$$

Then

$$\left.\frac{d}{dt} f(c(t))\right|_{t=0} = \nabla f(x_0)\, h.$$

The first term is zero because we are moving along a level set. ■

Since the Newton step is always locally downhill (in the sense of reducing f) as it points in the opposite direction of the gradient, moving in that direction is always a good idea locally, and we just saw that locally the function decreases most rapidly in that direction; unfortunately the size of the step is not local (as a local step would lead nowhere) and therefore can lead us uphill, so we need to make some step-size adjustments. Note that we did this adjustment in the univariate case by enforcing the bracket, which is not possible in two or more dimensions. Here we will apply a method called a **trust-region with dogleg**. Consider the approximation

$$m_k(p) = f(x_k) + \nabla f(x_k)^T p^T + \frac{1}{2} p^T H(x_k)\, p$$

subject to

$$\|p\| \leq \Delta_k;$$

Δ_k is called the "trust-region radius" and defines the area over which we think the function is sufficiently well-approximated by the quadratic Taylor expansion, and m_k is called the "model function" (it models or approximates the behavior of f in a neighborhood of x_k); trust regions are the next best thing to brackets and exist in all dimensions. If $H(x_k)$ is positive definite and $\left\| H(x_k)^{-1} f(x_k) \right\| \leq \Delta_k$, then the vector that minimizes the model function m_k is the unconstrained one

$$p_k^B = -B(x_k)^{-1} \nabla f(x_k).$$

If these conditions are not satisfied, we will look for approximate solutions that are easy to find; wasting time finding an exact solution only to discard it at the next step is not sensible (see

below). Given p_k define

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)},$$

which is the ratio of the actual reduction to the predicted reduction (which is always nonnegative since we could choose $p_k = 0$). Thus, we know immediately that if $\rho_k < 0$ the step leads uphill and should be rejected. If ρ_k is close to 1, we can expand Δ_k because the function is closely approximated by the model function and we can be more aggressive in our search.

The trust-region algorithm is then to start with $\overline{\Delta} > 0$, $\Delta_0 \in \left(0, \overline{\Delta}\right)$, and $\eta \in \left[0, \frac{1}{4}\right)$. Calculate p_0 by minimizing the model function and evaluate ρ_0. If $\rho_0 < \frac{1}{4}$, so that the actual decrease is small compared to the expected one, set $\Delta_1 = \frac{1}{4}||p_0||$, otherwise if $\rho_0 > \frac{3}{4}$ and $||p_0|| = \Delta_0$ set $\Delta_1 = \min\left\{2\Delta_0, \overline{\Delta}\right\}$ or else set $\Delta_1 = \Delta_0$. We therefore adjust the size of the trust region to adapt to whether the function is declining faster or slower than expected. Finally, to compute the step we set $x_1 = x_0 + p_0$ if $\rho_0 > \eta$, otherwise we set $x_1 = x_0$.

To find an approximate solution to the constrained optimization, we start by using the Cauchy point, defined as the solution to

$$p_k^s = \underset{p}{\operatorname{argmin}} \left\{ f(x_k) + \nabla f(x_k)^T p \right\}$$

subject to the trust-region restriction

$$||p|| \leq \Delta_k.$$

Also find

$$\tau_k = \underset{\tau > 0}{\operatorname{argmin}} \left\{ m_k(\tau p_k^s) \right\}$$

subject to

$$||\tau p_k^s|| \leq \Delta_k.$$

The Cauchy point is then

$$p_k^C = \tau_k \frac{\Delta_k}{||\nabla f(x_k)||} \nabla f(x_k)$$

$$\tau_k = \begin{cases} 1 & \text{if } \nabla f(x_k)^T H(x_k) \nabla f(x_k) \leq 0 \\ \min\left\{ ||\nabla f(x_k)||^3 / \left(\Delta_k \nabla f(x_k)^T H(x_k) \nabla f(x_k) \right), 1 \right\} & \text{otherwise} \end{cases}.$$

We then use a dogleg to improve convergence. We will restate the trust-region problem as

$$\min_p \left\{ f + g^T p + \frac{1}{2} p^T H p : ||p|| \leq \Delta \right\}.$$

If H is positive definite, then we set

$$p^H = -H^{-1}g,$$

which is obviously the solution if feasible:

$$p^*(\Delta) = p^H \text{ if } \Delta \geq ||p^H||.$$

If Δ is small, then the quadratic term in the objective function plays little role, so that the solution is approximately the same as minimizing the linear part:

$$p^*(\Delta) \approx -\Delta \frac{g}{||g||}.$$

For larger values of Δ p^* follows a curved trajectory that is driven by the quadratic corrections, which makes it difficult to "follow". The dogleg replaces that complicated trajectory with a piecewise linear path. The first segment runs from the origin x_k to the unconstrained minimizer along the steepest descent direction:

$$p^U = -\frac{g^T g}{g^T H g} g;$$

we have already seen that the locally steepest descent is the (negative) of the gradient. The second segment runs from p^U to p^H. We can define the entire path then by

$$\widetilde{p}(\tau) = \begin{cases} \tau p^U & 0 \leq \tau \leq 1 \\ p^U + (\tau - 1)\left(p^H - p^U\right) & 1 \leq \tau \leq 2 \end{cases}.$$

It is easy to see that the dogleg path either intersects the boundary of the trust-region once if $||p^H|| \geq \Delta$ and not at all otherwise, and we can find an approximate value for τ by solving the scalar quadratic equation

$$\left|\left|p^U + (\tau - 1)\left(p^H - p^U\right)\right|\right|^2 = \Delta^2$$

and choosing the root that lies in $[0, 2]$.

Note the calculation of the Hessian may be problematic. BFGS (Broyden-Fletcher-Goldfarb-Shanno) uses the iteration scheme

$$x^{n+1} = x^n - A^{n+1}\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)$$

where the approximation to the inverse of the Hessian matrix A^n follows

$$A^{n+1} = A^n + \frac{\left(x^{n+1} - x^n\right)\left(x^{n+1} - x^n\right)^T}{\left(x^{n+1} - x^n\right)\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)} - \frac{\left(A^n\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)\right)\left(A^n\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)\right)^T}{\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)A^n\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)} + \left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)A^n\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)mm^T$$

with

$$m = \frac{x^{n+1} - x^n}{\left(x^{n+1} - x^n\right)\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)} - \frac{A^n\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)}{\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)A^n\left(Df\left(x^{n+1}\right) - Df\left(x^n\right)\right)}.$$

This iteration avoids encountering singular or ill-conditioned Hessians, and can be combined with trust-regions and doglegs to get a local minimizer with global convergence (it converges to a local minimum from any starting value; it does not converge to a global minimum necessarily). A related approximation scheme is called Davidon-Fletcher-Powell (DFP), which is identical to BFGS except it is missing the last term. In general, BFGS dominates DFP, but in my experience the difference is not so significant that it really matters either way.

For maximum-likelihood problems, where the objective is the sum of a log-likelihoods, the Hessian has structure that we can exploit to improve convergence. We can iterate on

$$\beta^{n+1} = \beta^n + \lambda B^{-1} g^n$$

where λ is a step-size parameter, g^n is the gradient evaluated at β^n, and

$$s_m\left(\beta^n\right) = \left.\frac{\partial L}{\partial \beta}\right|_{\beta^n}$$
$$B = \frac{1}{M}\sum_m s_m\left(\beta^n\right) s_m\left(\beta^n\right)^T.$$

The s function is an observation-specific gradient and commonly called the **score**; the gradient is then the average score

$$g^n = \frac{1}{M}\sum s_m\left(\beta^n\right).$$

This iterative scheme is called Berndt, Hall, Hall, and Hausman (BHHH), based on their 1974 paper. Computing B is much faster than computing the Hessian, and since the scores are needed

in g anyway we can save substantial time in problems with lots of observations. Furthermore, B is positive definite. BHHH is generally dominated by BFGS as well.

For least squares problems performance is better if we use a routine specialized to solve them; the main routine is called the Levenburg-Marquardt algorithm. Consider the nonlinear least-squares problem

$$\min_{\beta} \left\{ \sum_{t=1}^{m} \left(y_t - f\left(x_t, \beta \right) \right)^2 \right\}.$$

The idea is to replace current estimate β with new estimate $\beta + \delta$ and determine the best step δ. Locally approximate the nonlinear function f at $\beta + \delta$ as

$$f\left(x_t, \beta + \delta \right) = f\left(x_t, \beta \right) + J_t \delta$$

where

$$J_t = D_\beta f\left(x_t, \beta \right)$$

is the gradient at point (x_t, β). We now have the approximate quadratic problem

$$S\left(\beta + \delta \right) = \sum_{t=1}^{m} \left(y_t - f\left(x_t, \beta \right) - J_t \delta \right)^2$$

which has first-order condition

$$\left(J^T J \right) \delta = J^T \left(y - f\left(\beta \right) \right).$$

As before, Newton steps that satisfy this equation are locally but not globally downhill, so we introduce a "regularization factor" $\lambda \operatorname{diag}\left(J^T J \right)$ that reduces the size of steps, but allows for larger steps in directions along which the gradient is smaller:

$$\left(J^T J + \lambda \operatorname{diag}\left(J^T J \right) \right) \delta = J^T \left(y - f\left(\beta \right) \right).$$

Note in econometrics the regularization factor leads to a "ridge regression". The step taken is therefore

$$\delta = \left(J^T J + \lambda \operatorname{diag}\left(J^T J \right) \right)^{-1} J^T \left(y - f\left(\beta \right) \right),$$

and the update for the minimum goes from β to $\beta + \delta$. The regularization parameter λ is chosen to be some initial value λ_0 and then λ_0 / ν for some $\nu > 1$; if both of these lead to solutions that are worse than the current guess β, we progressively check $\lambda_0 \nu^k$ for values of k until we get a better residual, and reset λ_0 to its new value.

6.6 Constrained Optimization

All of the above are tools for doing unconstrained optimization; generally, however, problems in economics are constrained optimizations. For non-smooth problems, Powell gives us COBYLA (Constrained Optimization BY Linear Approximation), which handles general problems, and the specialized BOBYQA (Bound Optimization BY Quadratic Approximation) and LINCOA (LINearly Constrained Optimization Algorithm) in which only bound or linear constraints are present; of course COBYLA can handle them too, but at an efficiency cost. All of these algorithms are adaptations of NEWUOA.

The state-of-the-art for smooth local optimization with constraints are SQP algorithms, which stands for sequential quadratic programming (or sometimes successive quadratic programming). The idea is to solve the nonlinear optimization problem as a sequence of quadratic programs, which are individually very easy to solve, keeping track of constraints by dividing them into "active" (will be binding at the solution and therefore is currently treated as binding) and "inactive" (will be slack at the solution and therefore is currently treated as slack); of course we don't know at the outset which constraints go in which set, so we will have to update our active set along with our guess at the solution. Suppose we have a current guess (x_k, λ_k, μ_k) for the solution (iterate k), where (λ_k, μ_k) are the multipliers of the Lagrangian

$$L(x, \lambda, \mu) = f(x) - \lambda^T g(x) - \mu^T h(x).$$

Then we solve the approximate quadratic program

$$d_k = \underset{d}{\operatorname{argmin}} \left\{ f(x_k) + Df(x_k)^T d + \frac{1}{2} d^T D^2 f(x_k) d \right\}$$

subject to

$$g(x_k) + Dg(x_k)^T d \geq 0$$
$$h(x_k) + Dh(x_k)^T d = 0.$$

In this problem constraints that are in the active set are imposed as equalities, those in the inactive set are ignored entirely (for now), which means we can solve the simpler linear-quadratic program

$$d_k = \underset{d}{\operatorname{argmin}} \left\{ f(x_k) + Df(x_k)^T d + \frac{1}{2} d^T D^2 f(x_k) d \right\}$$

subject to

$$g^*(x_k) + Dg^*(x_k)^T d = 0$$
$$h^*(x_k) + Dh^*(x_k)^T d = 0;$$

g^* is the vector of active inequality constraints (constraints that we think will bind at the solution). Since this problem is concave, the Kuhn-Tucker conditions deliver a solution (d_k, q_k) that determines the update:

$$x_{k+1} = x_k + d_k$$
$$\mu^*_{k+1} = \mu^*_k + q_k$$

where μ^* is the vector of Lagrange multipliers on the linearized active set constraints. Sophisticated implementations use the same kinds of adjustments to ensure the steps (d_k, δ_k) are not so large that they lead "uphill" in the original nonlinear problem, including trust region and dogleg adaptations and not using the actual Hessian matrix. The next step is to investigate how to change the active set. For some constraints in the active set the multiplier μ^*_{k+1} will be negative; these constraints will be moved into the inactive set, as the objective can be decreased by allowing them to go slack (that is, they are forcing us to do "worse" by keeping them binding). For constraints in the inactive set, some of them will be violated by our new point; we then retreat to the boundary of these **blocking constraints** and enter them into the active set for the next step. FSQP, or "feasible SQP", places bounds not only on the solution but also the search region, in case there are points where the objective function is not well-defined (such as negative consumption with log utility); these algorithms then need to generate an initial point that is feasible, if you do not provide one that works, in order to get started.

The most sophisticated (freely available) SQP algorithm I have seen is NLPQLP. Here the approximate subproblem is

$$d_k = \underset{d,\delta}{\operatorname{argmin}} \left\{ f(x_k) + Df(x_k)^T d + \frac{1}{2} d^T D^2 f(x_k)\, d + \frac{1}{2}\sigma_k^2 \delta^2 \right\}$$

subject to

$$(1-\delta)\, h^*(x_k) + Dh^*(x_k)^T d = 0;$$

the additional variable δ is introduced to deal with infeasible linearized constraints (it displaces them towards the trust-region), and σ_k is a term that penalizes large values of δ (because large

displacements ultimately have to be "undone" to enforce the actual constraints). The algorithm also keeps track of a "working set" of constraints, rather than the entire active set, meaning the routine can handle very large problems (in particular ones with lots of constraints, such as constraints that contain integrals or other functional expressions, sometimes called semidefinite programs) because it does not consider all the constraints at any time; this approach is also useful when the inequality constraints are not all linearly independent, because having linearly dependent constraints binding makes the quadratic problem impossible to solve. Given the search direction d_k, the algorithm optimizes over the step size α_k to obtain the iteration

$$x_{k+1} = x_k + \alpha_k d_k$$
$$\mu^*_{k+1} = \mu^*_k + \alpha_k \left(q_k - \mu^*_k\right).$$

The performance of SQP algorithms is improved if you can provide the derivatives analytically, both in terms of speed and accuracy. Since evaluating derivatives can be time-consuming, the best option is automatic differentiation; for some code, particularly older code, that option is not available. For example, FSQP can use automatic differentiation, but because the function values and derivatives are computed in separate subroutines that are called separately the efficiency gain is smaller; NLPQLP can efficiently use AD, because it computes both the values and the derivatives in the same call to the same subroutine.

FSQP and NLPQLP can also solve a particular class of min-max (saddlepoint) problems, namely

$$\min_x \left\{ \max_{i \in \{1,\ldots,I\}} \left\{ f_i\left(x\right) \right\} \right\}$$

subject to

$$g_i\left(x\right) \leq 0$$
$$h_i\left(x\right) = 0;$$

that is, we look to minimize the maximum value of I different functions through the choice of x. With some effort, it is possible to recast the recursive saddlepoint problem in the first chapter as a problem of this form.

An alternative to SQP are the class of algorithms called interior point (IP) methods. Here, we transform the problem into an unconstrained problem that "avoids" violating constraints. Define

a barrier function by

$$b(x;\gamma) = \gamma \sum_{i=1}^{n} \log(g_i(x))$$

where γ is a fixed parameter; note that $b(x;\gamma)$ goes to $-\infty$ as we approach the boundary of the region $g(x) \geq 0$. Consider the objective function

$$F(x;\gamma) = f(x) - b(x;\gamma);$$

as $g_i(x)$ goes to zero F diverges to infinity at a rate defined by γ. Note that the gradient of F will be very poorly behaved as γ gets small (near the optimum where the constraint is binding), but the solution is distorted by large γ since we cannot get close to the constraint; the interior-point method solves a sequence of problems with decreasing γ and stops when the constraint is within some tolerance of binding. IPOPT is a publicly-available interior-point algorithm that works quite well.

Related to barrier methods are penalty methods, which use the function

$$p(x;\gamma) = -\frac{\gamma}{2} g(x)^2;$$

for γ large enough the optimizer will experience a large enough cost of violating the constraint that it refuses to do so. These methods use a sequence of γ values that get larger to keep the problem well-behaved; the tradeoff is again the enforcement of the constraint (large γ) versus a well-behaved gradient (small γ). The augmented Lagrangian method adds another constraint of the form $\lambda g(x) \geq 0$, and updates λ along with γ as

$$\lambda \to \lambda + \gamma g(x(\gamma))$$

where g_k is the solution to the unconstrained problem given γ. The advantage of this formulation is that γ will not need to get so large to avoid violating the constraint, which helps keep the gradient under control.

Which method dominates seems to depend on the particular problem, but I use SQP methods personally as I've found them to be fast and robust for the kinds of problems generally found in dynamic programming. KNITRO, the best overall minimizer on the market today, permits a choice of SQP or IP depending on the problem at hand; unfortunately KNITRO is not free.

For least squares problems, note that Levenberg-Marquadt is an unconstrained optimization method. A new method, called Derivative-Free BOx-constrained Least Squares (DFBOLS),

combines Levenberg-Marquadt ideas with SQP methods to obtain a very robust minimizer that can accommodate bound constraints. Describing all the steps that go into this algorithm would be a challenge.

Remember that for convex problems we have a set of necessary and sufficient conditions for a minimum – the Kuhn-Tucker conditions mentioned above:

$$\begin{aligned} Df(x) + \lambda^T Dg(x) + \mu^T Dh(x) &= 0 \\ \lambda &\leq 0 \\ g(x) &\leq 0 \\ h(x) &= 0 \\ \lambda g(x) &= 0 \\ \mu h(x) &= 0. \end{aligned}$$

Unfortunately, this system is not a system of equations, since it has inequalities. But through a clever change of variables suggested by Garcia and Zangwill (1981) we can turn it into a system of equations. Let

$$\lambda = -\max\{\eta, 0\}^2.$$

Then the solution to the KT conditions is the same as the solution to the system of equations

$$\begin{aligned} Df(x) - \max\{\eta, 0\}^2 Dg(x) - \mu Dh(x) &= 0 \\ -\max\{-\eta, 0\}^2 - g(x) &= 0 \\ h(x) &= 0. \end{aligned}$$

Note first that

$$\lambda \leq 0;$$

if $\eta > 0$ then $\lambda < 0$, if $\eta < 0$ then $\lambda = 0$. Note next that

$$g(x) \leq 0;$$

if $\eta > 0$ then $g(x) = 0$, if $\eta < 0$ then $g(x) < 0$. And now also that

$$\lambda g(x) = 0$$

since either $\eta < 0$ or $\eta > 0$. We can now use a nonlinear equation solver to find the solution to the system of equations (see below); while in general this approach is not recommended simply for optimization (as it is generally easier to minimize a function than to find a solution to a system of equations), it can be useful for finding competitive equilibria that are not the solutions to optimization problems and where constraints occasionally bind.

An alternative to the Garcia-Zangwill approach is to use the Fischer-Burmeister function

$$\varphi(\lambda, x) = \lambda + g(x) - \sqrt{\lambda^2 + g(x)^2};$$

note that $\varphi(\lambda, x) = 0$ if and only if $\lambda \geq 0$, $g(x) \geq 0$, and $\lambda g(x) = 0$. The equations to be solved are

$$\begin{aligned} Df(x) - \lambda Dg(x) - \mu Dh(x) &= 0 \\ \varphi(\lambda, x) &= 0 \\ h(x) &= 0. \end{aligned}$$

A smoothed version uses an extra parameter $\tau > 0$, and allows $\tau \to 0$:

$$\varphi(\lambda, x; \tau) = \lambda + g(x) - \sqrt{\lambda^2 + g(x)^2 + 2\tau}$$

where

$$\tau^{k+1} = \beta \tau^k$$

and $\beta \in (0, 1)$ is updated along with the solutions $\left(\lambda^k, x^k\right)$ as we search for a solution. Using the smoothed version requires you to have control over the algorithm, which makes it difficult to use with good solvers without knowing exactly what variable counts the number of iterations.

What is the difference between the Garcia-Zangwill and Fischer-Burmeister methods? That is, which one should we use? Unfortunately, I have not found any systematic comparisons of the two methods; the GZ approach is common in economics while physicists seem to use the FB method. The answer, as with many such questions, seems to be "it depends".

6.7 Global Optimization

The First Rule of Global Optimization (Dirty Harry Rule): "You should never feel lucky".

All of these multidimensional routines are "local" in the sense that they are designed to converge to local optima (although they often converge to local optima globally, so watch your language – globally convergent means convergent to a local optimum from any initial starting value); for some problems we need to find global optima with some degree of confidence. Global optimizers must have a method for escaping local minima by occasionally moving "uphill", which as noted above we do not permit in local optimization methods through adjustment of the step size and/or the trust region. But we do not want to allow uphill movements too often, or we will never get to the minimum. The best global optimizers are stochastic as a result – they sometimes allow things to get worse before they get better.

The first algorithm (and one of the oldest) is called simulated annealing. This approach is related to the Nelder-Mead simplex algorithm in that we start with a simplex Δ^n and implement the same set of potential "moves." However, to each point $x_i \in \Delta^n$ we add a logarithmically normal random variable proportional to a value T (called the temperature) and subtract a similar random variable to each value tested during a move. This approach will always accept any downward step but also sometimes accepts upward ones that are not too large. In the limit $T \to 0$ we converge to a local minimum. The advantage comes at relatively large values of T. The area of exploration is on a scale proportional to T; it executes a form of Brownian motion within this area with an efficiency independent of size. That is, the algorithm generally explores all regions equally well. Thus, as we decrease T slowly we maximize the likelihood that the algorithm shrinks into the region that contains the best local minimum (which is by definition the global minimum), as these points will remain in the simplex.

What "sufficiently slowly" means is unfortunately very model specific. There are several options:

- Reduce T to $T(1-\epsilon)$ after every m moves, where ϵ/m is determined by trial and error;
- Budget a total of K moves and reduce T after every m moves to a value

$$T = T_0 \left(1 - \frac{k}{K}\right)^{\alpha}$$

  where k is the number of moves so far and α is a constant larger than 1;
- After every m moves set T to $\beta(f_1 - f_b)$ where β is a constant of order 1, f_1 is the smallest

value currently in the simplex, and f_b is the best point so far. Never reduce T by more than γ.

If the algorithm appears stuck, we can restart from the best point encountered (unless it is in the simplex currently). Simulated annealing is not recommended as it is dominated by more recent tools; my experience with it is not positive.

One better method is the Metropolis-Hastings algorithm combined with a Monte Carlo Markov Chain. The idea is to start with a candidate minimizer x_0, then draw a new candidate x^* (say, from a multivariate normal with mean equal to x_0 and some specified variance-covariance matrix, called the proposal density), and a uniform random variable u. Then the search moves according to the rule

$$x_1 = \begin{cases} x^* & \text{if } u \leq \frac{f(x_0)}{f(x^*)} \\ x_0 & \text{otherwise} \end{cases}.$$

Note that the algorithm always takes "downhill steps" (if $f(x^*) < f(x_0)$ then u is always less than the ratio since it is confined to $[0, 1]$) but sometimes takes "uphill steps"; a good rule of thumb is to choose the variance of the proposal density to obtain a rejection rate (where x^* is discarded) equal to somewhere around $25 - 30$ percent so as to prevent slow mixing. We then take as the global maximum the largest value of $f(x_t)$ in the sample (the mode if we interpret the chain as coming from a distribution). The problem with the MH algorithm is the undirected random proposals – it would be better if these candidates could be directed towards more promising ones from the outset, rather than waiting for the rejection.

A more powerful class of algorithms are called genetic algorithms, for this very reason – there will be some intelligence in the choice of new vectors. This approach generates a "population" of candidate vectors for the minimum. It tests each one for "fitness" by evaluating the function at that point. Members of the population are then selected for "breeding" according to their fitness; in the manner of Darwinian evolution, more fit members of the population should be more likely to breed. Breeding occurs according to specific rules, generating a set of "children" who are compared with their parents for relative fitness. We then "mutate" the population randomly to generate diversity, evaluate the fitnesses of the new population, and continue. Results from measure theory suggest that this approach generates a sequence of vectors which are monotonic

in the following sense:

$$\Pr\left\{f\left(x^{n+1}\right) \leq f\left(x\right) \ \forall x \in X\right\} \geq \Pr\left\{f\left(x^{n}\right) \leq f\left(x\right) \ \forall x \in X\right\}$$

across successive generations n and $n+1$. That is, the population at generation $n+1$ is more likely to contain the global minimum over the set X than generation n. As we approach our solution we gradually reduce the mutation rate and the mutation size, treating these like temperature in the simulated annealing algorithm.

The nature of breeding is one again in which there are no optimal rules, only useful guidelines. Here is one that was found to be good at solving least-squares problems.

1. Take the fitness function to be the value of the the function. The goal is to solve

$$\min_{x}\left\{f\left(x\right)\right\}.$$

2. Set an initial population Ω which consists of n vectors x.

3. Evaluate the fitness of each member of the initial population.

4. From the population select n pairs with replacement. These vector-pairs will be candidates for breeding. The selection criterion weights each member by its fitness according to the rule

$$1 - \frac{f\left(x_j\right)}{\sum_{j=1}^{n} f\left(x_j\right)}$$

so that more fit specimens are more likely to breed. Note that you have to adjust this step for problems in which f can be negative; for least-squares problems this issue obviously does not arise.

5. From each breeding pair generate one offspring according to the BLX-α crossover routine. This routine generates a child in the following fashion. Denote the parent pair by $\left(x_i^1, x_i^2\right)_{i=1}^{9}$. The child is then given by

$$\left(h_i\right)_{i=1}^{9} \tag{35}$$

where $h_i \sim UNI(c_{\min} - \alpha I, c_{\max} + \alpha I)$, $c_{\min} = \min\left\{x_i^1, x_i^2\right\}$, $c_{\max} = \max\left\{x_i^1, x_i^2\right\}$, and $I = c_{\max} - c_{\min}$. Our choice is $\alpha = 0.5$, which was found to be the most efficient by Herrera, Lozano, and Verdegay (1998) in their horse-race of genetic algorithms.

6. Next, introduce mutation in the children. With probability $\mu_G = 0.15 + \frac{0.33}{t}$, where t is the current generation number, we mutate a particular element of the child vector. This mutation involves two random numbers, r_1 and r_2, which are $UNI(0,1)$, and one random number s which is $N(0,1)$. The element, if mutated, becomes

$$h_i = \begin{cases} h_i + s\left[1 - r_2^{\left(1-\frac{t}{T}\right)^{\delta}}\right] & \text{if } r_1 > 0.5 \\ h_i - s\left[1 - r_2^{\left(1-\frac{t}{T}\right)^{\delta}}\right] & \text{if } r_2 < 0.5 \end{cases}; \tag{36}$$

set $\delta = 2$ following Duffy and McNelis (2001). Note that both the rate of mutation and the size shrinks as time progresses, allowing us to zero in on potential solutions.

7. Evaluate the fitness of the children.

8. From each family trio, discard the least fit member. There are now exactly n members of the population again.

9. Compare the most fit member of the last generation, if not selected for breeding, with the least fit member of the new generation. Keep the better of the two vectors. If the most fit member of generation $t-1$ is selected for breeding this step should not be executed. This step is called **elitism** and ensures that the best member of the population is never discarded.

10. Return to step 4 unless the population's best N vectors have not changed significantly.

It is frequently a good idea to "polish" a solution using some other, faster algorithm once the algorithm appears to have settled into a solution. A good publicly-available genetic algorithm is PIKAIA, named after some weird flatworm from 530 million years ago. PIKAIA allows for full generational replacement, steady-state-delete-random, and steady-state-delete-worst generational movements (with elitism), with uniform one-point crossover and one-point-uniform mutation with a rate controlled by the difference between the best point and the median in the current population. It is also written to be accessible and the code can be inspected, but be aware that, unusually for optimiazation routines, PIKAIA is written to maximize a function. It also requires that the objective be positive, to avoid problems with the selection lottery, so instead of using

$$\min_x \{f(x)\} = \max_x \{-f(x)\}$$

you need to use

$$\min_x \{f(x)\} = \max_x \left\{ \frac{1}{f(x)} \right\}.$$

Algorithms related to genetic algorithms are particle swarm methods, bee colony methods, ant colony methods, fish school search, and intelligent water drops. These differ from genetic algorithms merely in terms of the heuristics used to guide the movements of each member of the population, but all have the same key ingredients. For particle swarm methods, the idea is to randomly move particles around through the state space and keeping track of the best locations of each particle and the swarm as a whole, while bee colony methods use previously-located "food sources" as the starting point for random local searches.

One of the best routines I have encountered is the Boender-Timmer-Rinnoy Kan algorithm (unfortunately the authors named the Fortran routine Global, which obviously creates some difficulties with Googling the code). Start with two sets, X^* (the set of local minima that have been encountered so far) and $X^{(1)}$ (the set of points from which local searches delivered convergence to a known local minimum); initially both sets are empty. Draw N^* random points and add them to the sample (which is initially empty as well). For the points in the sample with the lowest g values (where g is an input parameter that is smaller than N^*) initiate a steepest descent step and replace them with the result of the local optimization, then discard the remainder of the sample; call this set the transformed sample. Apply a clustering procedure using as seeds first the elements of X^* and then the elements of $X^{(1)}$. If some points are not assigned to a cluster because they are distinct, initiate the steepest descent local line search to obtain a new local minimum x^* and add that point to X^* (if it is new) or $X^{(1)}$ (if it is not new), then draw another sample of N^* random points and repeat. Once all points are assigned to a cluster in the first step stop and select the element of X^* with the lowest value as the global minimum.

The specific clustering method used is called a "single linkage method" which takes an initial partition of single elements and fuses subsets that are closest to each other to form new sets, using the distance measure

$$d\left(x, x'\right) = \sqrt{\left(x - x'\right)^T H\left(x^*\right)\left(x - x'\right)}$$

where x^* is a seed point in X^* and H is an estimate of the Hessian of f (which we can get as in the quasi-Newton methods above). The procedure is terminated when the only unclustered points lie too far from a given cluster. If the seed point is in $x^{(1)}$ we use I as the weighting

matrix. Before adding any point to a cluster the gradient is checked to make sure it points away from the local minimum x^*.

The algorithm also delivers a "confidence interval". Let p be a probability and y^* be the candidate global minimum. Let $y^{(1)}$ and $y^{(2)}$ be the two lowest points in the original sample. Define

$$y(p) = y^{(1)} - \frac{y^{(2)} - y^{(1)}}{p^{-2/n} - 1}$$

where n is the dimension of x. Then the interval

$$[y(p), y^*]$$

does not contain the true global minimum with probability $1 - p$. Note that these confidence intervals do not have any meaning if the units of the objective function are not known. If the interval is not acceptable, additional points can be added to the sample and the process repeated. GLOBAL is an unconstrained minimizer, so if you have constraints you may need to consider adding penalty functions to the problem to enforce them. It is not clear how well that works, though, so other algorithms may be needed.

One useful method for global optimization under bound constraints is called DiRect, for Dividing Rectangles; this algorithm is globally convergent to the global optimum, but can only handle relatively small problems (and all variables need to be bounded). The algorithm first converts a rectangular search space defined by the bounds into the unit hypercube; the centroid of this hypercube is denoted c_1 and we compute $f(c_1)$. We then divide the hypercube by evaluating the points $c_1 \pm \delta e_i$ where δ is $\frac{1}{3}$ of the side-length of the hypercube and e_i is the ith unit vector. Define

$$w_i = \min\{f(c_1 + \delta e_i), f(c_1 - \delta e_i)\}$$

and divide the dimension with the smallest w_i into thirds, so that $c_1 \pm \delta e_i$ are the centers of new hyperrectangles. The pattern is repeated on the center hyperrectangle for each dimension.

The next step is to identify potentially optimal hyperrectangles. Let $\epsilon > 0$ and let $f_{\min}$ be the current best function value. A hyperrectangle j is said to be **potentially optimal** if there exists some $\widehat{K} > 0$ such that

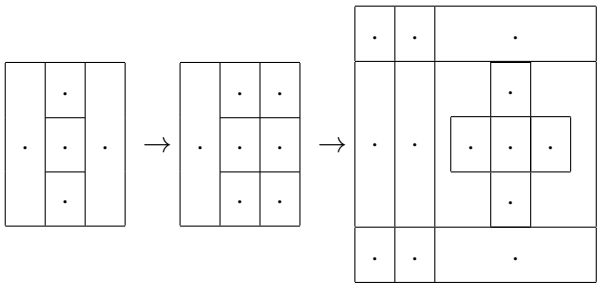
$$f(c_j) - \widehat{K} d_j \leq f(c_i) - \widehat{K} d_i$$

for all i and

$$f(c_j) - \widehat{K} d_j \leq f_{\min} - \epsilon |f_{\min}| .$$

c_j is the center of hyperrectangle j and d_j is the measure (the distance from c_j to the vertices). Experiments indicate $\epsilon = 10^{-4}$ is a reasonable value. Note that if i is potentially optimal then $f(c_i) \leq f(c_j)$ for all hyperrectangles with $d_j = d_i$. If $d_i \geq d_k$, for all k hyperrectangles and $f(c_i) \leq f(c_j)$ with $d_j = d_i$, then i is potentially optimal. And if $d_i \leq d_k$ and i is potentially optimal, then $f(c_i) = f_{\min}$. In general there may be more than one potentially optimal hyperrectangle at each iteration; the next step is to subdivide the potentially-optimal hyperrectangles along the longest dimensions. The process terminates after a given number of steps, and one would normally then implement a local search to "polish" the solution (which is a good strategy for any global method). DiRDFN adds general constraints to DiRect.

Here is a figure that illustrates the behavior of DiRect. Starting with the leftmost box, we evaluate the red points. Given these values, the routine selects the right long rectangle for subdivision, and we evaluate the two new red points. The result of this step is to subdivide the left long rectangle (which was skipped over before) and the central right box. The process would then continue until we get small enough boxes.



With global optimization we need to accept one fact – it takes a long time, particularly if the function is costly to evaluate, because we need to explore a lot of territory. Therefore, using good routines that avoid unnecessary calculations is critical to good performance. Note also that the common approach of "start local searches at a couple of different points" is not a robust method for finding global optima.

A systematic approach to "start at different points" can be effective, as shown by the TikTak algorithm (Arnoud, Guvenen, and Kleineberg 2019), which is related to BTRK but uses quasi-Monte Carlo integration ideas:

1. Generate N Sobol' points and evaluate $(f_i)_{i=1}^{N}$; keep the best $N^* < N$ points $(s_i)_{i=1}^{N^*}$.

2. Set $i = 1$. Choose s_1 and run a local search, storing the local solution $\left(x_1^{low}, f_1^{low}\right)$.

3. For each $i > 1$, start a local search from

$$S_i = (1 - \theta_i)\, s_i + \theta_i x_{i-1}^{low}$$

where

$$\theta_i = \min\left\{\max\left\{0.1, \sqrt{\frac{i}{N^*}}\right\}, 0.995\right\}.$$

Denote the local solution as (f_i^*, x_i^*). If $f_i^* < f_{i-1}^{low}$, then $f_i^{low} = f_i^*$ and $x_i^{low} = x_i^*$. Increment i and repeat.

This algorithm can be parallelized easily using a common save file; create two files done.dat and best.dat. done.dat contains the largest value of i that has been completed; the new local search then chooses s_{i+1} from the list of Sobol' points. best.dat contains $\left(f^{low}, x^{low}\right)$. Each computer, once it finishes, updates done.dat, then checks best.dat for the value of S_i and overwrites it if the result f_i^* is better than f^{low}. Note that this "dirty parallelization" can be done using any computers with access to a common folder (such as Box or Dropbox), and so can be done without knowledge of parallel computing tools or in languages that do not effectively support OpenMP or MPI.

Testing global optimizers is done using a set of well known nasty functions – these functions combine flat spots with many local minima that are at the bottom of narrow valleys and ridges that surround local optima. Here are some standard ones.

1. Griewank function

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{a} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 2$$
$$a = 200$$
$$x^* = (0, ..., 0)$$
$$f(x^*) = 1;$$

2. Levi No. 13 function

$$f(x) = \sin^2(3\pi x_1) + (x_n - 1)^2\left(1 + \sin^2(2\pi x_n)\right) + \sum_{i=1}^{n-1}(x_i - 1)^2\left(1 + \sin^2(3\pi x_{i+1})\right) + 1$$

$$x^* = (1, ..., 1)$$

$$f(x^*) = 1;$$

3. Rastrigin function

$$f(x) = An + \sum_{i=1}^{n}\left(x_i^2 - A\cos(2\pi x_i)\right) + 1$$

$$A = 10$$

$$x^* = (0, ..., 0)$$

$$f(x^*) = 1;$$

4. Rosenbrock function

$$f(x) = \sum_{i=1}^{n-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (1 - x_i)^2\right) + 1$$

$$x^* = (1, ..., 1)$$

$$f(x^*) = 1;$$

5. Ackley function

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + e + 20$$

$$x^* = (0, ..., 0)$$

$$f(x^*) = 0;$$

6. Shekel function

$$f(x) = \sum_{i=1}^{m}\left(c_i + \sum_{j=1}^{n}(x_j - a_{ji})^2\right)^{-1}.$$

7. PIKAIA test function

$$f(x) = -\cos(nr\pi)^2 \exp\left(-\frac{r^2}{\sigma}\right)$$
$$r = \sqrt{\sum_{i=1}^{m}\left(x_i - \frac{1}{2}\right)^2}$$
$$n = 9$$
$$\sigma = 0.15$$
$$x^* = \left(\frac{1}{2}, ..., \frac{1}{2}\right)$$
$$f(x^*) = -1;$$

8. Branin-Hoo test function with three global minima

$$f(x) = a\left(x_2 - bx_1^2 + cx_1 - r\right)^2 + s(1-t)\cos(x_1) + s$$
$$a = 1$$
$$b = \frac{5.1}{16\pi^2}$$
$$c = \frac{5}{\pi}$$
$$r = 6$$
$$s = 10$$
$$t = \frac{1}{8\pi}$$
$$x^* = (-\pi, 12.275) \text{ or } x^* = (\pi, 2.275) \text{ or } x^* = (9.42478, 2.475)$$
$$f(x^*) = 0.397887.$$

Many nonlinear least-squares problems are going to be littered with local minima (and possibly flat spots and ridges), and therefore should only be done using solid global minimizers.

6.8 Mathematical Programming with Equilibrium Constraints

The next four subsections consider special kinds of mathematical programs and discuss specialized algorithms that solve them. First, we consider the complementarity problem

$$\min_x \{f(x)\}$$

subject to

$$g(x) \geq 0$$
$$h(x) = 0$$
$$0 \leq G(x) \perp H(x) \geq 0;$$

the last constraint says that there are functions of choice variables that must be orthogonal. Common problems that fit into this setup are principal-agent problems where the agent's internal maximization problem shows up as a constraint on the choices of the planner:

$$\min_{x,a} \{f(x,a)\}$$

subject to

$$a = \min_{h(x,b)\geq 0} \{g(x,b)\}.$$

The necessary and sufficient conditions for the second minimization will generally involve orthogonality conditions (think about the Kuhn-Tucker theorem); if the "first-order" approach is valid, then we can replace that constraint with the FOC

$$D_b g(x,b) - \lambda^T D_b h(x,b) = 0$$

and the complementary slackness conditions

$$0 \leq \lambda^T \perp h(x,b) \geq 0.$$

These problems are difficult because constraint qualification conditions generally fail, which are central to the convergence properties of the SQP methods – solving the local QP problem relies on the use of the FONC, and without some version of constraint qualification these conditions are not generally necessary (that is, the solution need not satisfy them). However, we can reformulate the problem to make SQP methods work. First, introduce slack variables s, such that the MPEC problem becomes

$$\min_{x,a} \{f(x,a)\}$$

subject to

$$g(x,a) \geq 0$$
$$h(x,a) - s = 0$$
$$a \geq 0, s \geq 0, as = 0;$$

note the last set of constraints takes the form of complementary slackness conditions. We then "smooth" this problem by rewriting it as

$$\min_{x,a,s,t} \left\{ f(x,a) + \rho e^T t \right\}$$

subject to

$$\begin{aligned} g(x,a) - t &\geq 0 \\ h(x,a) - s &= 0 \\ \Phi_\mu(a,s) &= 0 \end{aligned}$$

where

$$\Phi_\mu(a,s) = \begin{cases} \varphi_\mu(a_1, s_1) \\ \vdots \\ \varphi_\mu(a_m, s_m) \end{cases}$$

$$\varphi_\mu(a,b) = a + b - \sqrt{a^2 + b^2 + 2\mu}$$

for $\mu > 0$; the $\varphi_\mu(a,b)$ function is a smoothed Fischer-Burmeister function and t is called an 'elastic variable' that penalizes constraint violations.

Another approach is to rewrite the problem as

$$\min_x \{ f(x) \}$$

subject to

$$\begin{aligned} g(x) &\geq 0 \\ h(x) &= 0 \\ G(x) &\geq 0 \\ H(x) &\geq 0 \\ G(x) \circ H(x) &\leq 0 \end{aligned}$$

where '$\circ$' denotes the Hadamard (element-by-element) product. Because the constraint set is still not particularly well-behaved, the first approach is preferred.

6.9 Linear Programming

Next, we consider another special class of optimization problems – linear programs. Linear programming solves problems of the form

$$\min_x \left\{ \sum_{i=1}^{n} a_i x_i \right\} \tag{37}$$

subject to

$$Cx \leq b \tag{38}$$

$$x \geq 0; \tag{39}$$

that is, both the objective and the constraints are linear. The standard algorithm for solving linear programs is called the simplex method (which is not related to the Nelder-Mead downhill simplex method). The essence of the simplex method is that, for linear programs, solutions must occur at points where constraints intersect (the vertices). Furthermore, if we define the number of variables to be n and the number of constraints to be m (not including the nonnegativity constraints, which are also called primary constraints), then we will have at most m variables which are nonzero, making at least $n - m$ variables equal to zero at the solution. We can formalize this idea in the fundamental theorem of linear programming.

Definition 63 *A vector that is feasible and satisfies n of the (non-primary) constraints with equality is a* ***feasible basic vector****.*

Theorem 64 *(Fundamental Theorem of Linear Programming) If an optimal feasible vector exists (that is, it satisfies the constraints and it maximizes the linear function), then there is a feasible basic vector that is optimal.*

That is, any solution to the linear program yields the same objective value as one of the vertices in n-space. Under certain circumstances one of the constraints is collinear with the level sets of the objective, meaning that the solution may not be unique. In that case, it is permissible to simply select a vertex – that is why the theorem states existence rather than uniqueness of the optimal basic vector. These cases will occur with probability zero and generally can be ignored in practical problems. The simplex method is a systematic way of ordering the vertices so that we test as few as possible before concluding we have found the optimal point.

Large linear programs (meaning ones with hundreds of thousands of variables and constraints) are too cumbersome to solve with the simplex method, so these are usually solved using an interior-point method. Certain kinds of problems have a special structure permitting them to be decomposed into smaller problems linked by only a small number of constraints; in these cases (which arise in problems with private information, for example) the Dantzig-Wolfe decomposition can be used to solve much larger problems. The specific structure is that the "coupling constraints" are relatively few and described by the matrices D and the rest can be put into a block-diagonal structure:

$$C = \begin{bmatrix} D_1 & D_2 \\ F_1 & 0 \\ 0 & F_2 \end{bmatrix}.$$

The method then determines an appropriate objective for each subproblem – the ones constrained by the F matrices – and adjusts the values of the variables to accommodate the coupling constraints, and repeats until the objective cannot be improved any further.

6.10 Integer Programming

A particularly challenging optimization problem is the mixed-integer program, in which some variables are constrained to be integers. These problems are NP-hard. Specialized code to solve integer programs works by finding solutions to the "relaxed" problem where solutions are not integer-constrained, then efficiently checking the nearby integers to see which one is best. The procedure for this checking is called 'branch and bound' and will be discussed below.

Consider the optimization problem

$$\min_{x,y} \{f(x,y)\}$$

subject to

$$\begin{aligned} g_j(x,y) &= 0 \quad j = 1, ..., m_e \\ g_j(x,y) &\leq 0 \quad j = m_e + 1, ..., m_e + m \\ x_l &\leq x \leq x_u \\ y_l &\leq y \leq y_u \end{aligned}$$

where x is a real vector and y is an integer vector. MISQP applies SQP to this problem, using a branch-and-bound approach to solving for the integer variables. The idea is to start at the solution to the relaxed problem in which y variables are not integers. Obviously, this problem gives a lower bound to our objective; forcing the y variables to be integers can only increase the objective function value. To obtain an upper bound, we can round each y variable toward the nearest integers that deliver a feasible solution.

Next, we select a particular integer variable, y_k, and solve two relaxed problems that impose either an upper bound on y_k by rounding down to $\lfloor y_k \rfloor$ or a lower bound on y_k by rounding up to $\lfloor y_k \rfloor + 1$.

The solution to the relaxed problem serves as the root of the binary tree. We then take a variable in y, solve the two subproblems, and keep the one that delivers the lower objective value. We then repeat this process until all variables in y have been bounded by integers and we obtain the solution (which need not be unique).

How do we choose the "branch" points (which y_k to choose at each step)? At the first step (branching from the root), we choose the y_k with the largest non-integer part (that is, the coordinate that is furthest from $\lfloor y_k \rfloor$). At subsequent steps, we either employ the "best of all" strategy, in which we branch from the free node that has the smallest lower bound, or the "depth first" strategy which branches from the most recently-created node. The benefit of the depth first method is that it simplifies the solution to the relaxed problem, which is very similar to the most recently solved subproblem, but the cost is that it will often require the solution to more subproblems than the best of all. We can also take a **fathoming** of the tree – if the relaxed problem at any step is infeasible or delivers a higher objective value than the current upper bound, we need never branch from that node again.

6.11 Fractional Programming

Closely related to linear programming is linear-fractional programming, where we try to solve a problem of the form

$$\min_x \left\{ \frac{c^T x + \alpha}{d^T x + \beta} \right\}$$

subject to

$$Ax \leq b.$$

By a clever change of variables introduced by Charnes and Cooper (????), we can rewrite this problem as a linear program. Define

$$y = \frac{1}{d^T x + \beta} x$$
$$t = \frac{1}{d^T x + \beta};$$

now we have

$$\min_{y,t} \left\{ c^T y + \alpha t \right\}$$

subject to

$$Ay \leq bt$$
$$d^T y + \beta t = 1$$
$$t \geq 0.$$

We solve this LP and recover our initial solution as

$$x = \frac{y}{t}.$$

There is one additional restriction we need to impose, which is that the constraint set must imply the denominator term $d^T x + \beta$ is strictly positive at every feasible x.

A similar transformation works for a general fractional program under some additional restrictions. Suppose we have

$$\min_x \left\{ \frac{f(x)}{g(x)} \right\}$$

subject to

$$h(x) \leq 0;$$

if f is convex and nonnegative, g is positive and concave, and the constraint set is convex, then we have a concave program. The transformation is

$$y = \frac{x}{g(x)}$$
$$t = \frac{1}{g(x)}$$

so that we have

$$\min_{y,t} \left\{ tf\left(\frac{y}{t}\right) \right\}$$

subject to

$$h\left(\frac{y}{t}\right) \leq 0$$
$$tg\left(\frac{y}{t}\right) \leq 1$$
$$t \geq 0.$$

We can again recover the original solution as $x = \frac{y}{t}$. If $g(x)$ is affine, we do not need to restrict the sign of $f(x)$ and we get the constraint

$$tg\left(\frac{y}{t}\right) = 1.$$

6.12 Geometric Programming

Another special class of programs are geometric programs – these programs put restrictions on the objective and the constraints. A real-valued function is called a **monomial function** if it takes the form

$$f(x) = cx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$$

with $c > 0$; note that this definition of monomial differs from the one used in approximation theory. A sum of monomial functions is a **posynomial function**:

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}}\cdots x_n^{a_{nk}}.$$

Note that the exponents do not need to be integers. Posynomials are closed under addition, multiplication, and positive scaling. Note that if f is a posynomial and g is a monomial, then f/g is a posynomial, and if γ is a nonnegative integer then f^γ is a posynomial.

A geometric program is a problem of the form

$$\min_x \{f_0(x)\}$$

subject to

$$f_i(x) \leq 1 \; i \in \{1, ..., m\}$$
$$g_i(x) = 1 \; i \in \{1, ..., p\}$$

where the f functions are posynomials and the g functions are monomials.

This class is broader than it may seem. With a clever trick, we can accomodate objectives and constraints that do not appear to be posynomials. For example, consider the problem

$$\min_{x,y,z} \left\{ \sqrt{1+x^2} + \left(1+\frac{y}{z}\right)^{3.1} \right\}$$

subject to

$$\begin{aligned} \frac{1}{x} + \frac{z}{y} &\leq 1 \\ \left(\frac{x}{y} + \frac{y}{z}\right)^{2.2} + x + y &\leq 1. \end{aligned}$$

The objective and second constraint are not posynomials. If we define new variables (t_1, t_2, t_3), we can rewrite this problem as

$$\min_{t_1,t_2,t_3,x,y,z} \left\{ t_1^{0.5} + t_2^{3.1} \right\}$$

subject to

$$\begin{aligned} 1 + x^2 &\leq t_1 \\ 1 + \frac{y}{z} &\leq t_2 \\ \frac{1}{x} + \frac{z}{y} &\leq 1 \\ t_3^{2.2} + x + y &\leq 1 \\ \frac{x}{y} + \frac{y}{z} &\leq t_3. \end{aligned}$$

This program is a GP. Similarly, we can handle constraints of the form

$$\max\left\{f_1(x), f_2(x)\right\} + f_3(x) \leq 1$$

by changing it to

$$\begin{aligned} t + f_3(x) &\leq 1 \\ f_1(x) &\leq t \\ f_2(x) &\leq t. \end{aligned}$$

Any function that can transformed in this manner is called a **generalized posynomial**, since it is effectively a posynomial in an expanded space. They are closed under addition, multiplication,

positive powers, and maximization, as well as operations that can be derived from these operations (such as division by monomials). They are also closed under composition of a particular form. If f_0 is a generalized posynomial of k variables with no negative exponents and $\{f_1, ..., f_k\}$ are generalized posynomials, then

$$f_0\left(f_1\left(x\right), ..., f_k\left(x\right)\right)$$

is a generalized posynomial. With some manipulation, negative terms, negative powers, minimization, and right-hand-side terms can also be accomodated, although it isn't always obvious how to do it. We can simply replace the functions in the GP with generalized posynomials and we still have a geometric program.

GPs are easy to solve – they can be transformed into nonlinear convex programming problems (convex objective, convex inequality constraints, and linear equality constraints). Since they are convex, we also know that any local solution is the global solution. We first transform the problem by changing the variables to logs:

$$y_i = \log\left(x_i\right).$$

We then minimize the log of the objective $\log\left(f_0\right)$ subject to the constraints

$$\log\left(f_i\right) \leq 0$$

$$\log\left(g_i\right) = 0.$$

That is, we solve

$$\min_y \left\{\log\left(f_0\left(\exp\left(y\right)\right)\right)\right\}$$

subject to

$$\log\left(f_i\left(\exp\left(y\right)\right)\right) \leq 0$$

$$\log\left(g_i\left(\exp\left(y\right)\right)\right) = 0.$$

A more general but harder class of problems are signomial programs:

$$\min_x \left\{g_0\left(x\right)\right\}$$

subject to

$$f_i\left(x\right) = 1 \quad i \in \{1, ..., m\}$$

$$g_i\left(x\right) - h_i\left(x\right) \leq 1 \quad i \in \{1, ..., n\}$$

where each f is a monomial and each g and h are posynomials; signomial programs are NP-hard. A **signomial** is similar to a posynomial, but the c terms need not be positive, which means they cannot be log-transformed. Signomial programs are solved using a sequence of approximate geometric programs (analogous to SQP methods); since the term $g_i(x) - h_i(x)$ can be written in many ways, it is generally recommended that the programmer maximize the number of terms in $g_i(x)$, with priority given to those variables that also appear in $g_0(x)$, and then prioritize variables that appear in other constraints. Variables that appear in only the ith constraint should be placed in $h_i(x)$. Equality constraints are particularly difficult in signomial programs, since they restrict the feasible space of the GP approximation.

We can solve the growth model using geometric programming (Tsener 2020). Consider the problem

$$\max_{\{c_t,k_{t+1}\}_{t=0}^{T}} \left\{ \sum_{t=0}^{T} \beta^t \frac{c_t^{1-\gamma}}{1-\gamma} \right\}$$

subject to the resource constraints

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

with k_0 given; turnpike theorems imply that for large enough T this program can approximate an infinite horizon as well (the solution $(c_0, k_1)(k_0)$ approximates the stationary infinite-horizon solution $(c, k')(k)$ as T goes to infinity). If $\gamma > 1$, then the objective is the sum of monomials with negative coefficients; then we can rewrite the problem as a minimization and the objective is a posynomial. To handle the resource constraint, we can use a "condensation" method that approximates the right hand side as

$$\begin{aligned} M(k_t) &\equiv \left(\frac{(1-\delta) k_t}{\omega_{1,t}} \right)^{\omega_{1,t}} \left(\frac{k_t^{\alpha}}{\omega_{2,t}} \right)^{\omega_{2,t}} \\ \omega_{1,t} &= \frac{(1-\delta) k_t}{k_t^{\alpha} + (1-\delta) k_t} \\ \omega_{2,t} &= 1 - \omega_{1,t}. \end{aligned}$$

Now the resource constraint is

$$c_t M(k_t)^{-1} + k_{t+1} M(k_t)^{-1} \leq 1.$$

We then iterate on the $(\omega_{1,t}, \omega_{2,t})$ terms; given a sequence of weights, solve the GP program, then use the solution to update the weights and repeat. This procedure converges under very mild

restrictions. It is trivial to add a nonnegativity constraint on investment:

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

is easily transformed into

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

6.13 Semi-Definite Programming

Closely related to geometric programs are semi-definite programs, which involve the choice of matrices that are required to be positive semi-definite. Take the program

$$\min_x \left\{ c^T x \right\}$$

subject to the matrix inequality

$$F(x) \succeq 0,$$

where we can decompose F as

$$F(x) = F_0 + \sum_{i=1}^{m} x_i F_i.$$

Each F matrix is $n \times n$ and symmetric (hence the decomposition), and the constraint states that $F(x)$ must be positive semidefinite (that is, $z^T F(x) z \geq 0 \; \forall z$). This type of inequality is called a **linear matrix inequality**. This program is easily see to be convex: if $F(x) \geq 0$ and $F(y) \geq 0$ then

$$F(\lambda x + (1-\lambda) y) \geq 0.$$

The boundary of the constraint set is called a **spectrahedron** (roughly speaking, it is a polyhedron but the edges are not required to be straight lines; more formally, a spectrahedron is the intersection of a convex cone with a linear affine subspace). Solutions to semidefinite programs, like linear programs, are always on the boundary of the feasible set. In fact, linear programs are easily seen to be special cases of semi-definite programs; the constraint set

$$Ax + b \geq 0$$

can be rewritten as

$$F_0 = \operatorname{diag}(b)$$

$$F_i = \operatorname{diag}(a_i)$$

where a_i is the ith row of A.

Since you know the definition of positive semidefiniteness, you might ask why we don't simply use the constraint

$$z^T A z \geq 0;$$

the reason is that this constraint is actually an infinite number of constraints, once for each z, and thus is not computationally tractable.

As with geometric programming, tricks can convert nonlinear programs into SDPs. For example, consider

$$\min_x \left\{ \frac{\left(c^T x\right)^2}{d^T x} \right\}$$

subject to

$$Ax + b \geq 0,$$

with the restriction that $d^T x > 0$ for all feasible x. We then change the program to

$$\min_{x,t} \{t\}$$

subject to

$$\begin{aligned} Ax + b &\geq 0 \\ \frac{\left(c^T x\right)^2}{d^T x} &\leq t \end{aligned}$$

and then build the LMI as

$$\begin{bmatrix} \operatorname{diag}(Ax + b) & 0 & 0 \\ 0 & t & c^T x \\ 0 & c^T x & d^T x \end{bmatrix} \succeq 0.$$

The trick here is to use the Schur complement. The 2×2 matrix inequality

$$\begin{bmatrix} t & c^T x \\ c^T x & d^T x \end{bmatrix} \succeq 0$$

is equivalent to the principal minors conditions

$$\begin{aligned} d^T x &\geq 0 \\ t - \frac{\left(c^T x\right)^2}{d^T x} &\geq 0; \end{aligned}$$

the second term is called the Schur complement of $d^T x$.

We can even take quadratic constraints and make them LMIs. For example, consider the constraint

$$(Ax+b)^T (Ax+b) - c^T x - d \geq 0;$$

it can be written as

$$\begin{bmatrix} I & Ax+b \\ Ax+b & c^T x + d \end{bmatrix} \succeq 0,$$

so that

$$F_0 = \begin{bmatrix} I & b \\ b^T & d \end{bmatrix}$$

$$F_i = \begin{bmatrix} 0 & a_i \\ a_i^T & c_i \end{bmatrix};$$

therefore any program with a quadratic objective and quadratic constraints can be rewritten as a semi-definite program. Another useful SDP is the problem of minimizing the sum of the r largest eigenvalues of a matrix:

$$\min_{x,t} \{rt + \text{trace}\{X\}\}$$

subject to

$$tI + X - A(x) \succeq 0$$

$$X \succeq 0,$$

which arises combinatorics, and the minimization of the norm of a matrix

$$\min_{x,t} \{t\}$$

subject to

$$\begin{bmatrix} tI & A(x) \\ A(x)^T & tI \end{bmatrix} \succeq 0.$$

Another application is called "pattern separation" in which we try to separate two sets of points $\{x_1, ..., x_K\}$ and $\{y_1, ..., y_L\}$. A separating hyperplane $a^T x + b = 0$ exists if

$$a^T x_i + b \leq 0 \ \forall i$$

$$a^T y_j + b \geq 0 \ \forall j;$$

separating hyperplanes are only guaranteed to exist if the points define two disjoint convex sets. This problem is a standard linear program. If a separating hyperplane does not exist, we may be able to find a quadratic surface that separates the points. That is, we look for a quadratic function

$$f(x) = x^T Ax + b^T x + c$$

such that

$$x_i^T Ax_i + b^T x_i + c \leq 0 \quad \forall i$$
$$y_j^T Ay_j + b^T y_j + c \geq 0 \quad \forall j.$$

This problem can also be cast as a linear program. If we put restrictions on the separating surface such that it must be an ellipse (so that it can represent confidence intervals, for example), which imposes a restriction that $A \succeq 0$, we can find an ellipsoid that contains all of x but none of y (or prove that no such ellipsoid exists). If we further insist that we find the "most spherical" such ellipsoid, we are adding the constraints

$$I \leq A \leq \gamma I$$

for some variable γ, and minimize γ over the constraint set. If there is a sphere that separates the two sets, then $\gamma = 1$ is optimal.

Semidefinite programs are easy to solve. The usual algorithm is to follow a particular path, called the center path, to the boundary of the feasible set; that intersection is the optimal choice. First, we use a barrier function to enforce the constraints:

$$\phi(x) = \log\left(\det\left(F(x)^{-1}\right)\right),$$

which is defined obviously only if F is positive definite. If the set X is bounded, then the **analytic center** of the LMI $F(x) \succeq 0$ is the solution to

$$x^* = \arg\min_x \{\phi(x)\};$$

this problem is easily solved using Newton's method, which converges quadratically and works even if the Newton direction must be computed approximately.

Now return to the original SDP:

$$\min_x \left\{c^T x\right\}$$

subject to

$$F(x) \succeq 0.$$

Consider instead the LMI

$$F(x) \succ 0$$
$$c^T x = \gamma,$$

with

$$p^* < \gamma < \overline{p} = \sup\left\{c^T x : F(x) \succ 0\right\}.$$

The analytic center of this feasible set exists for all γ, so we can parameterize a path through these centers using γ, called $x^*(\gamma)$. At $\gamma = \overline{p}$, we have that $x^*(\gamma)$ maximizes $c^T x$ subject to $F(x) \succ 0$; as $\gamma \to p^*$, $x^*(\gamma)$ approaches the optimal point for the SDP. The interior point method then just follows the central path from $\overline{p}$ to p^*, using a predictor-corrector approach – we predict where the path will go as γ decreases, then correct back to the central path by finding the appropriate $x^*(\gamma)$.

One application of SDP methods is to solve the "attention allocation problem": given a vector of noisy Gaussian random variables that represent signals, how can one allocate optimally attention (defined as noise reduction) across these variables given that we only have a finite amount of attention. This problem arises in rational inattention models in which information flow is constrained by an entropy condition. We will examine this problem later.

6.14 Minmax Problems

We now turn to solving saddlepoint problems. The simplest one is called the minmax problem:

$$\min_{x \in X} \max_{i \in \{0,\ldots,I\}} \{f_i(x)\}$$

subject to

$$g(x) \geq 0$$
$$h(x) = 0.$$

The minmax problem seeks to minimize the maximum of a finite number of functions, subject to constraints (note that the constraints apply to all functions; having separate constraints for each problem makes this a different problem entirely). To apply SQP methods, consider adding an additional scalar variable z and additional constraints

$$z - f_i(x) \geq 0.$$

Then we can solve the problem

$$\min_{z \in \mathcal{R}, x \in X} \{z\}$$

subject to

$$\begin{aligned} g(x) &\geq 0 \\ h(x) &= 0 \\ z - f_i(x) &\geq 0. \end{aligned}$$

Solving this problem is equivalent to solving the minmax problem.

A more difficult problem involves the pure saddlepoint problem, which can thought of as permitting an infinite number of $f_i(x)$ functions; this problem is often called the semi-infinite minmax problem:

$$\min_{x \in X} \{\psi(x)\}$$

where

$$\psi(x) \equiv \max_{y \in Y} \{\phi(x, y)\}.$$

One approach is to reformulate this problem as a semi-infinite program

$$\min_{x \in X, z \in \mathcal{R}} \{z\}$$

subject to

$$\phi(x, y) - z \leq 0 \quad \forall y \in Y.$$

Exchange algorithms solve this problem by iterating on (x, z), with the constraints exchanged at each iteration using the maximizer with respect to y:

$$\widehat{y}_i \in \arg\max_{y \in Y} \{\phi(x_i, y)\}$$

generates a new set of constraints

$$\phi\left(x_i, \widehat{y}_i\right) - z_i \leq 0.$$

Local reduction problems create semi-infinite programs locally. Discretization methods convert the SMX to a simple finite minmax problem by finely-discretizing the Y space; the most efficient ones use adaptive methods that put lots of points in the region near the minmax solution, but the disadvantage is that the number of points in the discretization tends to increase across iterations.

There are a number of applications of saddlepoints, including the risk-sharing economy in the introduction and optimization in the presence of ambiguity aversion (see the chapter on extensions of preferences).

7 Root Finding

In this part of the course we will discuss how to solve nonlinear equations, namely obtaining the set of values $x \in X$ such that

$$F(x) = 0 \ \forall x \in X. \tag{40}$$

7.1 One-Dimensional Root-Finding

The easiest case is one-dimensional:

$$f : \mathcal{R} \to \mathcal{R}. \tag{41}$$

For this case, we will employ a strategy which first "traps" the solution between two values and then progressively shrinks the trap down. For multidimensional cases, the problem is much more difficult, as one cannot trap a solution in more than 1 dimension.

The first task is to "bracket" a candidate zero. Formally,

Definition 65 *An interval* $[a, b]$ ***brackets*** *a zero of the continuous function* $f : X \to \mathcal{R}$ *if* $f(a) \cdot f(b) < 0$.

That is, we say that the interval contains a zero if the function changes sign over the interval, which is simply an application of the Intermediate Value Theorem. With computers, it is possible that we might think a zero is within the interval when there is no root there because the function

might be discontinuous, as in the case of

$$f(x) = \frac{1}{x - c}. \tag{42}$$

Simple evaluation of the function will rule out mistaking this point for an actual zero. Note that some "degenerate" zeroes (such as $x^2 = 0$) cannot be bracketed, while others ($x^3 = 0$) can.

Bracketing a zero is easy: pick an interval and look outward until you find a bracket. This step is relatively low-tech, but important: one should not just start searching for a zero without some good reason to expect it to be there. By trapping our zero, we ensure ourselves that the search will not be pointless.

We next take steps to "refine" our guess. **Bisection** is a method which cannot fail to locate a bracketed zero in a finite number of steps; think of bisection as the root-finding equivalent of golden section search. Start with an initial bracket $[a, b]$ and evaluate the function f at the midpoint, $\frac{a+b}{2}$. Replace the endpoint of the interval that agrees in sign with the midpoint with the midpoint value. After each step the interval is bisected into a part which contains the zero and a part which does not. That is, the root is known to be in an interval of size

$$\frac{b - a}{2} \tag{43}$$

after the first step. Repeating this process converges to within ϵ of the root in $\log_2\left(\frac{b-a}{\epsilon}\right)$ steps. Note that bisection cannot fail – it will return either a root or a singularity.

The problem with bisection is speed. Bisection converges linearly: the size of the interval containing the root converges to zero according to

$$\epsilon_{n+1} = \frac{\epsilon_n}{2}. \tag{44}$$

Bisection is simple to program:

1. Choose bracket $[a, b]$ and compute $f(a)$ and $f(b)$ (which might have been done during the bracketing process, or perhaps you know the basic shape of the function);
2. Assume without loss of generality that $f(a) < 0$;
3. Compute $f\left(\frac{a+b}{2}\right)$;
4. If $f\left(\frac{a+b}{2}\right) < 0$ then reset $a = \frac{a+b}{2}$, otherwise reset $b = \frac{a+b}{2}$;

5. Return to 1 until $|b - a| < \epsilon$.

Brent's method is a hybrid that combines aggressive methods based on interpolation with the security of bisection. We begin with a bracket $[a, b]$. We then compute a new point c which lies between a and b and evaluate the function at all three points. The initial guess is usually taken to be the midpoint

$$c = \frac{a+b}{2}. \tag{45}$$

Our estimate of the value for the root x is then taken as an inverse quadratic interpolant according to the updating rule

$$x = \frac{[y - f(a)]\,[y - f(b)]\,c}{[f(c) - f(a)]\,[f(c) - f(b)]} + \frac{[y - f(b)]\,[y - f(c)]\,a}{[f(a) - f(b)]\,[f(a) - f(c)]} + \frac{[y - f(a)]\,[y - f(c)]\,b}{[f(b) - f(c)]\,[f(b) - f(a)]}.$$

Setting $y = 0$ yields our updated root guess d as

$$d = b + \frac{P}{Q} \tag{46}$$

where

$$R = \frac{f(b)}{f(c)}$$
$$S = \frac{f(b)}{f(a)}$$
$$T = \frac{f(a)}{f(c)}$$

and

$$P = S\left[T\,(R - T)\,(c - b) - (1 - R)\,(b - a)\right]$$
$$Q = (T - 1)\,(R - 1)\,(S - 1)\,.$$

If the algorithm chooses a d that lies outside the bracket $[a, b]$ or too close to one endpoint it conducts a bisection step instead. In this way it maintains a strict bound on the solution but takes superlinear steps whenever possible. The inverse quadratic interpolation has a convergence rate of $m = 1.839$, so the convergence rate of Brent's method depends on how often it is forced to bisect. Brent showed that there exists N_0 such that, for iterations $n > N_0$, there are no further bisection steps. Note that there is one special case that creates problems – what happens if

$f(a) = f(b)$, so that we cannot fit an inverse parabola? In this case, Brent's method takes a secant step; it projects the secant line that connects $f(a)$ and $f(c)$ to the x-axis, checks whether this step is permissible (inside the bracket), and then either takes it or bisects appropriately.

There exist functions in which Brent's method only takes bisection steps (N_0 is sufficiently large given the initial bracket that it is never violated), although they are not easy to find.

If we can calculate derivatives for the function we can improve the speed of convergence. **Newton-Raphson** will use this information by taking the new guess for the zero to be the zero of a linear approximation to f at the current guess. By Taylor's theorem, we have

$$f(x+\delta) \approx f(x) + Df(x)\,\delta + \frac{D^2 f(x)}{2}\delta^2 + o(|\delta|^3). \tag{47}$$

For small enough δ, we can ignore the quadratic terms so that

$$f(x+\delta) = 0 \tag{48}$$

implies

$$\delta = -\frac{f(x)}{Df(x)}. \tag{49}$$

We then construct the iterative procedure

$$x^{n+1} = x^n - \frac{f(x^n)}{Df(x^n)} \tag{50}$$

to search for the root. Define

$$\epsilon^n = x^n - x^*.$$

Using Taylor's theorem we have

$$f(x^n - \epsilon^n) = f(x^n) - \epsilon^n Df(x^n) + \frac{(\epsilon^n)^2}{2} D^2 f(\xi^n)$$

for some ξ^n between x^n and x^*. Since $f(x^*) = 0$, we have

$$0 = f(x^n) - (x^n - x^*)\,Df(x^n) + \frac{(\epsilon^n)^2}{2} D^2 f(\xi^n).$$

Given that $Df(x^*) \neq 0$, then for x^n close enough to x^* we also have $Df(x^n) \neq 0$. Therefore,

$$0 = \frac{f(x^n)}{Df(x^n)} - (x^n - x^*) + \frac{(\epsilon^n)^2}{2}\frac{D^2 f(\xi^n)}{Df(x^n)}.$$

Using the Newton iteration, we get

$$x^{n+1} - x^* = \frac{(\epsilon^n)^2}{2} \frac{D^2 f(\xi^n)}{Df(x^n)}.$$

Therefore,

$$\left|\epsilon^{n+1}\right| \leq \frac{1}{2} \frac{\left|D^2 f(\xi^n)\right|}{|Df(x^n)|} |\epsilon^n|^2,$$

That is, we have quadratic convergence near a simple (isolated) root.

Near a repeated root of order m (meaning that $D^m f(x^*) = 0$), the convergence is linear at rate $\frac{m-1}{m}$, but the modified Newton's method

$$x^{n+1} = x^n - m \frac{f(x^n)}{Df(x^n)}$$

will converge quadratically (provided we know m). To see why, note that we can write

$$f(x) = (x - x^*)^m g(x)$$

where $g(x^*) \neq 0$. Then

$$Df(x) = m(x - x^*)^{m-1} g(x) + (x - x^*)^m Dg(x).$$

The Newton step implies

$$\begin{aligned} x^{n+1} - x^* &= (x^n - x^*) - \frac{(x - x^*)^m g(x)}{m(x - x^*)^{m-1} g(x) + (x - x^*)^m Dg(x)} \\ &= \frac{(m-1) g(x^n) + (x^n - x^*) Dg(x^n)}{m g(x^n) + (x^n - x^*) Dg(x^n)} (x^n - x^*). \end{aligned}$$

Therefore,

$$\left|\epsilon^{n+1}\right| = \frac{m-1}{m} |\epsilon^n| + \left|\frac{Dg(x^*)}{m g(x^*)}\right| |\epsilon^n|^2,$$

which is dominated by the linear term. For the modified Newton method, we note that

$$\begin{aligned} x^{n+1} - x^* &= (x^n - x^*) - m \frac{(x^n - x^*)^m g(x)}{m(x^n - x^*)^{m-1} g(x^n) + (x^n - x^*)^m Dg(x)} \\ &= \left(\frac{Dg(x^n)}{m g(x^n) + (x^n - x^*) Dg(x^n)}\right) (x^n - x^*)^2, \end{aligned}$$

so that

$$\left|\epsilon^{n+1}\right| \leq \left|\frac{Dg(x^*)}{m g(x^*)}\right| |\epsilon^n|^2,$$

which is quadratic convergence.

We can combine Newton-Raphson with bisection: if the Newton step is permissible (inside the bracket) then we take it, otherwise we bisect. Newton-Raphson is not globally convergent, so the bracket enforcement is critical for convergence.

A two-point Newton method from Tiruneh-Ndlela-Nkambule

$$\begin{aligned} x_{n+1} &= x_{n-1} - \frac{x_{n-1} - x_n}{1 - \left(\frac{f(x_n)}{f(x_{n-1})}\right)\left(\frac{f(x_n)-f(x_{n-1})}{x_n - x_{n-1}}\right)\frac{1}{Df(x_n)}} \\ &= \left(1 - \frac{1}{r}\right) x_{n-1} + \frac{1}{r} x_n \end{aligned}$$

converges superquadratically and has better convergence properties than standard NR. Note that we must keep track of two iterates; initialization is done using a standard Newton-Raphson step.

If the derivatives must be computed numerically, Newton-Raphson becomes a related method called the secant method:

$$x^{n+1} = x^n - f\left(x^n\right) \frac{x^n - x^{n-1}}{f\left(x^n\right) - f\left(x^{n-1}\right)};$$

note the slope of the tangent line is replaced by the slope of a secant line determined by the last two iterates, which means the convergence rate is lower (it equals the golden ratio constant $\frac{1+\sqrt{5}}{2} \approx 1.618$). As with Newton-Raphson, the secant method is not globally convergent, so we need a bracketing procedure to guarantee we get to a root.

The secant method combined with bracketing is called **regula falsi** (false position), and oddly it can be slower than bisection alone. The regula falsi algorithm for current bracket $[a_k, b_k]$ generates the new point according to

$$c_k = b_k - f\left(b_k\right) \frac{b_k - a_k}{f\left(b_k\right) - f\left(a_k\right)}$$

and then updates the bracket to either $[a_k, c_k]$ or $[c_k, b_k]$. The Anderson-Björck method deals with the slow convergence problem that occurs when $f\left(c_k\right)$ has the same sign as $f\left(c_{k-1}\right)$; there it chooses

$$c_k = \frac{f\left(b_k\right) a_k - m f\left(a_k\right) b_k}{f\left(b_k\right) - m f\left(a_k\right)}$$

where

$$m = \left\{ \begin{array}{c} 1 - \frac{f(c_k)}{f(b_k)} \text{ if positive} \\ \frac{1}{2} \text{ otherwise} \end{array} \right\}.$$

For simple roots, Galdino found that the AB algorithm is the clear winner; for multiple roots (like $x^3 = 0$), the winner was Galdino's own method with

$$m = \frac{f(a_k) - f(c_k)}{\left(2 + \frac{f(c_k)}{f(b_k)}\right)^2}.$$

Ridder's method is closely related to regula falsi. Given a bracket $[a, b]$, we compute the midpoint $c = \frac{a+b}{2}$ and evaluate the function at all three points. Then we compute the exponential function $\exp(\phi x)$ such that $h(x) = f(x)\exp(\phi x)$ satisfies

$$h(c) = \frac{h(a) + h(b)}{2},$$

with

$$\exp(\phi(b-a)) = \frac{f(c) - \operatorname{sign}(f(a))\sqrt{f(c)^2 - f(a)f(b)}}{f(b)}.$$

Regula falsi is then applied to the points $(a, f(a))$ and $(b, f(b))$ to obtain

$$d = c + (c-a)\frac{\operatorname{sign}(f(a))f(b)}{\sqrt{f(c)^2 - f(a)f(b)}},$$

which is one of the bracket values for the next step. The other bracket value is either c (if $f(c)f(d) < 0$) or whichever of a or b delivers a bracket with d. If the function is well-behaved, the order of convergence is $\sqrt{2}$ because we must evaluate the function twice to update; otherwise convergence is linear.

Steffenson's method combines Newton-Raphson with an acceleration technique for sequences called Aitken's delta-squared method. Aitken's method takes a sequence $\{x_n\}$ and converts it to a sequence $\{a_n\}$ defined by

$$\begin{aligned} a_n &= x_n - \frac{(\Delta x_n)^2}{\Delta^2 x_n} \\ &= x_{n+2} - \frac{(x_{n+2} - x_{n+1})^2}{(x_{n+2} - x_{n+1}) - (x_{n+1} - x_n)}; \end{aligned}$$

a_n converges to the same limit as x_n, but faster. This method is often used for evaluating functions that require power series expansions, especially those that converge very slowly. Using this conversion we get the root-finding iteration

$$x^{n+1} = x^n - \frac{f(x^n)}{\frac{f(x^n + f(x^n))}{f(x^n)} - 1};$$

convergence here is quadratic, but we have to do another function evaluation to get it.

If we are worried about $Df(x) = 0$, we can alter Newton-Raphson to

$$x^{n+1} = x^n - \frac{f(x^n)}{Df(x^n) - \alpha f(x^n)}$$

for some α chosen such that $f(x^n)$ and $Df(x^n)$ have opposite sign. The resulting error satisfies

$$\epsilon^{n+1} = -\left(\frac{f(x^n)}{Df(x^n)} - \alpha\right)(\epsilon^n)^2;$$

for larger α the error converges more slowly.

A version of Newton's method will converge cubicly:

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

Note that this procedure requires an extra function evaluation relative to the quadratic convergence version. Another cubic variant, called the **leapfrog Newton method** (Kasturiarachi 2002), combines the secant method with Newton:

$$y^n = x^n - \frac{f(x^n)}{Df(x^n)}$$
$$x^{n+1} = x^n - \frac{(f(x^n))^2}{Df(x^n)(f(x^n) - f(y^n))}.$$

The leapfrog Newton method can even converge for cases where standard Newton-Raphson fails, such as the pathological function $f(x) = x^{\frac{1}{3}}$ which has a simple root at $x = 0$. Leapfrog methods are frequently used to solve ODEs (see below). Note that, if $y^n = x^n$, the second step will generate a zero in the denominator, so we need to check convergence at each step and not just at each iteration.

We can also get cubic convergence for some iterative schemes that involve second derivatives, such as Halley's method

$$x^{n+1} = x^n - \frac{f(x^n)}{Df(x^n) - f(x^n)\frac{D^2 f(x^n)}{2Df(x^n)}}$$

and the Schröder (or super-Newton) method

$$x^{n+1} = x^n - \frac{f(x^n)}{Df(x^n)} - \frac{(f(x^n))^2 D^2 f(x^n)}{2(Df(x^n))^3}.$$

The Schröder method can be extended to higher-order convergent iterations, but these require even higher-order derivatives; for example, the quartically convergent version is

$$x^{n+1} = x^n - \frac{f\left(x^n\right)}{Df\left(x^n\right)} - \frac{\left(f\left(x^n\right)\right)^2 D^2 f\left(x^n\right)}{2\left(Df\left(x^n\right)\right)^3} - \frac{\left(f\left(x^n\right)\right)^3}{6}\left(\frac{3\left(D^2 f\left(x^n\right)\right)^2 - Df\left(x^n\right) D^3 f\left(x^n\right)}{\left(Df\left(x^n\right)\right)^5}\right),$$

which requires the third derivative. Automatic differentiation will obviously be useful here, since we can avoid costly repeated calculations.

There are methods that have even higher-order convergence. For example, here is a sixth-order convergent method that only requires the first derivative, but multiple function evaluations:

$$\begin{aligned} y^n &= x^n - \frac{f\left(x^n\right)}{Df\left(x^n\right)} \\ z^n &= x^n - \left(\frac{5}{8} + \frac{3}{8}\left(\frac{Df\left(x^n\right)}{Df\left(y^n\right)}\right)^2\right)\frac{f\left(x^n\right)}{Df\left(x^n\right)} \\ x^{n+1} &= z^n - \left(\frac{Df\left(x^n\right) + Df\left(y^n\right)}{-2Df\left(x^n\right) + 4Df\left(y^n\right)}\right)\frac{f\left(z^n\right)}{Df\left(z^n\right)}. \end{aligned}$$

Whether the extra cost of each step is outweighed by the reduction in the number of steps will be problem-specific; with automatic differentiation we can reduce this particular case down to three function calls (since we get f and Df at the same time). There are other formulations of the sixth-order algorithm, such as

$$\begin{aligned} y^n &= x^n - \frac{f\left(x^n\right)}{Df\left(x^n\right)} \\ z_n &= x^n - \frac{f\left(x^n\right)\left(Df\left(x^n\right) + Df\left(y^n\right)\right)}{Df\left(x^n\right)^2 + Df\left(y^n\right)^2} \\ x^{n+1} &= z^n - \frac{f\left(z^n\right)\left(Df\left(x^n\right)^2 + Df\left(y^n\right)^2\right)}{2Df\left(x^n\right)Df\left(y^n\right)}. \end{aligned}$$

7.2 Multidimensional Root-Finding

For multidimensional problems, things are more complicated because there is no such thing as a bracket (the intermediate value theorem does not apply). One choice is to do a sequence of one-dimensional root findings along with an updating scheme. To take a two-dimensional case for clarity, assume that we seek the solution to the system

$$\begin{aligned} F_1(x_1, x_2) &= 0 \\ F_2(x_1, x_2) &= 0. \end{aligned}$$

There are two main methods for this approach: **Gauss-Jacoby** and **Gauss-Seidel**. For both, we guess an initial value $\left(x_1^0, x_2^0\right)$. For Gauss-Jacobi, we then solve the equation

$$F_1\left(x_1^*, x_2^0\right) = 0$$

and the equation

$$F_2\left(x_1^0, x_2^*\right) = 0$$

for x_1^* and x_2^* separately, and then update our guess as

$$x_1^1 = (1-\theta)\, x_1^0 + \theta x_1^*$$
$$x_2^1 = (1-\theta)\, x_2^0 + \theta x_2^*$$

for some $\theta \in (0,1)$. We stop when the iterates are sufficiently close together. This procedure is relatively inefficient, because it ignores the information about x contained in the solution to

$$F_1\left(x_1^*, x_2^0\right) = 0.$$

Gauss-Seidel fixes this inefficiency by using x_1^* in F_2 instead of x_1^0:

$$F_1\left(x_1^*, x_2^0\right) = 0$$
$$F_2\left(x_1^*, x_2^*\right) = 0.$$

Both methods are sensitive to initial guesses – there is no guarantee that you can find solutions to each equation for a given initial guess, or for any subsequent updated value. And the updating scheme may not converge either, so you may need to use a θ close to zero which makes the process very slow. These approaches should be used only as a last resort.

The second choice is multidimensional Newton-Raphson; this algorithm uses the recursion

$$x^{n+1} = x^n - J^{-1}(x^n)F(x^n) \tag{51}$$

where $J(x^n)$ is the Jacobian matrix of first derivatives evaluated at the current guess x^n. As with one-dimensional Newton-Raphson, this routine has superlinear convergence sufficiently near a root (in fact it is a contraction on a small enough neighborhood, so convergence is also guaranteed), but globally it can diverge; here we can't use bisection to halt the divergence. We therefore are typically forced to use a dampening term ξ:

$$x^{n+1} = x^n - \xi J^{-1}\left(x^n\right) F\left(x^n\right) \tag{52}$$

where $\xi > 0$ and typically close to 0. What this dampening does is try to ensure that our step moves towards the root (in the sense of minimizing the squared residual function $\langle F, F \rangle$); while the Newton direction is always downhill locally, the Newton step is not local and therefore could result in an upward movement if taken completely. We will see more sophisticated variants of the dampening approach below.

However, if calculating the Jacobian is costly or inaccurate or if it can become singular, figuring out what direction the Newton steps even roughly point may be difficult. Broyden's method (an extension of the secant method) avoids computing the Jacobian directly, instead using a sequence of approximations that converge to the true as we converge to the solution. Here we use the standard Newton step but where the Jacobian is replaced by a matrix J_n that satisfies the recursion

$$\left(J^{n+1}\right)^{-1} = \left(J^n\right)^{-1} + \frac{\Delta x^{n+1} - \left(J^n\right)^{-1} \Delta F^{n+1}}{\left(\Delta x^{n+1}\right)^T \left(J^n\right)^{-1} \Delta x^{n+1}} \left(\Delta x^{n+1}\right)^T \left(J^n\right)^{-1}$$

where

$$\Delta x^{n+1} = x^{n+1} - x^n$$
$$\Delta F^{n+1} = F\left(x^{n+1}\right) - F\left(x^n\right).$$

As with Newton methods, Broyden's method may not go downhill for the full step size, and so some adjustment must be done to get global convergence. One method of ensuring downhill progression is the line search approach that is essentially what was presented above; use the Newton direction but backtrack until a sufficiently-downhill step is guaranteed. The condition for whether a step is sufficiently downhill is called the Armijo-Goldstein condition:

$$F\left(x + \alpha p\right) \leq F\left(x\right) + \alpha c \nabla F\left(x\right)^T p$$

where $c \in (0, 1)$ is called the control parameter, p is the direction, and α is the step size. Armijo used $c = \frac{1}{2}$; if the initial step fails, reduce c by a factor of 2 and repeat.

An alternative method is the trust-region algorithm. The merit function is the criterion for deciding whether a step goes downhill – the least-squares merit function

$$f\left(x\right) = \frac{1}{2} \left|\left|F\left(x\right)\right|\right|^2 = \frac{1}{2} \sum_{i=1}^{n} F_i\left(x\right)^2$$

is a common and sensible choice (for minimization the merit function was the function itself, which is why I didn't mention it before). Consider the model function

$$m_k(p) = \frac{1}{2}\left|\left|F\left(x^k\right) + J\left(x^k\right)p\right|\right|^2 = f\left(x^k\right) + p^T J\left(x^k\right)^T F\left(x^k\right) + \frac{1}{2}p^T J\left(x^k\right)^T J\left(x^k\right)p,$$

which is a quadratic approximation to the merit function (rather than the function itself). We choose the step p^k by minimizing the model function subject to a constraint

$$||p|| \leq \Delta_k,$$

where Δ_k is the radius of the trust region; because the model function is quadratic this problem is easy to solve. Then compute the ratio of actual to predicted reduction in the merit function:

$$\rho^k = \frac{\left|\left|F\left(x^k\right)\right|\right|^2 - \left|\left|F\left(x^k + p^k\right)\right|\right|^2}{\left|\left|F\left(x^k\right)\right|\right|^2 - \left|\left|F\left(x^k\right) + J\left(x^k\right)p^k\right|\right|^2}.$$

The trust-region algorithm is then to start with $\overline{\Delta} > 0$, $\Delta_0 \in \left(0, \overline{\Delta}\right)$, and $\eta \in \left[0, \frac{1}{4}\right)$. Calculate p^0 by minimizing the model function and evaluate ρ^0. If $\rho^0 < \frac{1}{4}$, so that the actual decrease is small compared to the expected one, set $\Delta_1 = \frac{1}{4}\left|\left|p^0\right|\right|$, otherwise if $\rho^0 > \frac{3}{4}$ and $\left|\left|p^0\right|\right| = \Delta_0$ set $\Delta_1 = \min\left\{2\Delta_0, \overline{\Delta}\right\}$ or else set $\Delta_1 = \Delta_0$. We therefore adjust the size of the trust region to adapt to whether the function is declining faster or slower than expected. Finally, to compute the step we set $x^1 = x^0 + p^0$ if $\rho^0 > \eta$, otherwise we set $x^1 = x^0$.

As with minimization, we then employ a dogleg to improve performance. Define the Cauchy point

$$c^k = -\tau^k \left(\frac{\Delta_k}{\left\|J\left(x^k\right)^T F\left(x^k\right)\right\|}\right) J\left(x^k\right)^T F\left(x^k\right)$$

where

$$\tau^k = \min\left\{1, \frac{\left\|J\left(x^k\right)^T F\left(x^k\right)\right\|^3}{\Delta_k F\left(x^k\right) J\left(x^k\right)\left(J\left(x^k\right)^T J\left(x^k\right)\right) J\left(x^k\right)^T F\left(x^k\right)}\right\};$$

the Cauchy point is the minimizer of the model function along the direction of steepest descent. Define the unconstrained minimizer of the model function by $p_*^k = -J\left(x^k\right)^{-1} F\left(x^k\right)$ (which is a full Newton step). The dogleg procedure starts by calculating c^k. If $\left|\left|c^k\right|\right| = \Delta_k$, then $p^k = c^k$. Otherwise, $p^k = c^k + z\left(p_*^k - c^k\right)$, where z is the largest value in $[0, 1]$ such that $\left|\left|p^k\right|\right| \leq \Delta_k$. The dogleg minimizes over a piecewise linear segment that extends from the original

point to the Cauchy point and then to the unconstrained minimizer, but confined to the trust region (it is straightforward to show that the dogleg path intersects the boundary of the trust region at most one time). The reduction in the model function is therefore at least as good as using the Cauchy point and often much better. The hybrid-Powell method uses the dogleg to generate global convergence to a local minimum of the merit function; this root may not be a solution to $F(x) = 0$, unfortunately, but the algorithm does detect this failure (since evaluating F immediately reveals we have converged to a local minimum).

We can also solve systems of equations $F(x) = 0$ using minimization, by casting the problem as

$$\min_x \{1\}$$

subject to

$$F(x) = 0;$$

as we have noted already, nonlinear equality constraints are a challenge for optimization routines, but there are circumstances where this formulation is useful. Note that we generally do not want to solve the least-squares problem

$$\min_x \{\langle F(x), F(x)\rangle\}$$

because it likely has many local solutions for which $F(x^*) \neq 0$. While testing the "solution" clearly reveals whether we have found the global minimum, it may not be easy to actually find it (P versus NP!). The solutions to these two problems can serve as good initial guesses, however, especially with systems that have "hidden constraints" (points where the function is not well-defined).

7.3 Tensor Methods

Suppose we consider a function $F : \mathcal{R}^n \to \mathcal{R}^m$ with $m \geq n$. If $m = n$, then we look for a solution to the system of equations $F(x) = 0$, whereas if $m > n$ then we look to minimize the norm $\|F(x)\|_2$. Tensor methods work with the model function

$$M(x_c + d) = F(x_c) + DF(x_c)\, d + \frac{1}{2} T_c dd$$

where x_c is the current iterate and T_c is a three-dimensional tensor (a tensor is a generalization of matrices to more than 2 indices; they can be thought of as linear maps with certain transformational rules). Rather than using second-derivative information to construct T_c, as in Broyden's method, tensor methods build it by requiring that the model interpolates up to $\sqrt{n}$ past function values in a way that minimizes storage and execution costs. That is, we require that

$$F(x_{-k}) = F(x_c) + DF(x_c)\, s_k + \frac{1}{2} T_c s_k s_k$$

for $k \in \{1, ..., p\}$, where

$$s_k = x_{-k} - x_c.$$

The past points $\{x_{-1}, ..., x_{-p}\}$ are selected so that the set of directions $\{s_k\}$ is strongly linearly independent (each s_k is required to make an angle of at least 45 degrees with the subspace spanned by the other directions); it is common that $p \leq 2$ is all that is needed to get good performance.

We then choose T_c as the solution to

$$\min_{T \in \mathcal{R}^{m \times n \times n}} \{\|T\|_F\}$$

subject to

$$T_c s_k s_k = 2\left(F(x_{-k}) - F(x_c) - DF(x_c)\, s_k\right),$$

where

$$||T||_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{k=1}^{n} \left(T_c[i,j,k]\right)^2$$

is the Frobenius matrix norm. The solution is a sum of p rank-one tensors with symmetric horizontal faces:

$$T_c = \sum_{k=1}^{p} a_k s_k s_k$$

where a_k is the kth column of $A \in \mathcal{R}^{m \times p}$. A is defined as $A = ZM^{-1}$, where Z is an $(m \times p)$ matrix with columns

$$Z_j = 2\left(F(x_{-j}) - F(x_c) - DF(x_c)\, s_j\right)$$

and M is a $(p \times p)$ matrix defined by

$$M_{ij} = \left(s_i^T s_j\right)^2.$$

Using the solution for T_c, we can write the model function as

$$M(x_c + d) = F(x_c) + DF(x_c)\, d + \frac{1}{2}\sum_{k=1}^{p} a_k \left(d^T s_k\right)^2 .$$

Once we have the tensor model, we look for its root:

$$\min_{d \in \mathcal{R}^n} \left\{ \| M(x_c + d) \|_2 \right\} .$$

The minimizer is used to update $x_c \to x_+ = x_c + d^*$, and then standard global convergence tools (either line searches or trust regions) can be added to improve convergence. Tensor methods are almost always more efficient than Newton-based methods, especially for roots at which the Jacobian is close to singular.

7.4 Solving Hard Systems of Nonlinear Equations – Homotopy

Sometimes nonlinear equation systems have solutions that are "hard" to find, often because their Jacobians become singular and therefore the methods converge to local minima of the merit function. Homotopy and path-following algorithms can be useful in these situations. They take a system with a solution that can be easily found – we'll call this $G(x) = 0$ – and turns it into the system of interest, called $F(x) = 0$, using a homotopy parameter t:

$$H(x, t) = (1 - t)\, G(x) + tF(x) .$$

As t ranges from 0 to 1, the solution goes from the solution to $G(x) = 0$ to the solution to $F(x) = 0$. The hope is that the solution to $H(x, t)$ for a given t is close to the one for a t slightly smaller, generating good initial conditions. The best methods construct a differential equation system in t and thus can follow a path that is not monotone; that is, a path that requires t to sometimes decrease along the way to the solution. One useful choice is $G(x) = x - a$ for some constant vector a; obviously the solution is uniquely $x = a$.

The mathematics of the homotopy method are discussed in several papers; an easy introduction is contained in Eaves and Schmedders (1999). Let A,B be open subsets of $\mathcal{R}^m$ and $g : A \to B$ be a smooth function. The derivative Dg is an $m \times m$ matrix. A point $y \in B$ is called **regular** if the row rank of $Dg(y)$ is equal to m for every $x \in A$ such that $g(x) = y$. Let $F : X \to Y$ be smooth and assume X and Y are dimension m. The goal is to compute $F(x) = 0$.

Some terms are needed. A **path** is a continuous function of t as it ranges over $[0,1]$; a **homotopy** is a path of functions. A **route** is a homeomorphism of a convex subset of the real line containing at least two points; routes can have 0, 1, or 2 endpoints, and the ends can be open. A **primary route** has 2 endpoints; a primary route is a homeomorphism of a line. A **loop** is a route with no endpoints or open ends which intersects itself only once; loops are homeomorphisms of circles.

1. Select a homotopy $H : X \times [0,1] \to Y$ with $H(\cdot, 1) = F(\cdot)$. The function F is smooth and we choose H to be smooth as well. In practice there are several possible choices for H, but a typical one is the "convex combination"

$$H(x,t) = tF(x) + (1-t)G(x)$$

for some fixed function G.

2. Check that $H(x,0) = 0$ is an easy system – that is, it has a unique solution that is robust to small deformations. Since $H(x,0) = G(x)$, and we are free to pick G, we can ensure that the system has a unique, easily located zero (perhaps because we have already solved it!). One particularly clever choice is

$$G(x) = x - a$$

which naturally has a solution $x = a$. Recently Jong and Kim argue for

$$G(x) = F(x) - F(a) + A(x-a)$$

for some symmetric positive definite matrix A such that $DF(x) + A$ is nonsingular. They show that this choice can also solve complementarity problems of the form

$$\begin{aligned} x &\geq 0 \\ f(x) &\geq 0 \\ xf(x) &= 0, \end{aligned}$$

which, as we have seen with optimization, create a number of numerical problems.

3. Check that a primary route exists. This step requires proving that H is regular at every y in the range space. If it is regular, then open ends of routes tend to infinity and loops never touch either boundary (though they may appear and disappear along the way).

4. Check that the primary route is trapped – it does not tend to infinity.

5. Follow the primary route to a solution to $F(x) = 0$. One method for following is called the predictor-corrector method, related to Newton's method of solving zeros and iterative schemes for solving differential equations. The two steps are also sometimes called the Euler step and the Newton step.

   (a) Predictor step: Given a current point (x_0, t_0) in the path, we take the step

   $$(x_0, t_0) + sDH(x,t)\left(x', t'\right)$$

   where s is the step size and $DH(x,t)$ is the step direction.

   (b) Corrector step: Since the path may not be linear, the predictor step takes us off of it. The corrector step moves back to the primary route, by fixing either x or t (depending on which one has been changing most rapidly) and solving for the intersection with the path. If this step is not feasible (the solver cannot find the path), the step size in the predictor step will be shortened and the corrector step attempted again.

A second method for following the primary route is to parameterize both x and t as a function of the "distance travelled" s: $H(x(s), t(s)) = 0$ must be satisfied for all $s \geq 0$. The total derivative is then

$$D_x H(x,t)\,\dot{x} + D_t H(x,t)\,\dot{t} = 0$$

$$\dot{x} = \frac{\partial x}{\partial s}$$

$$\dot{t} = \frac{\partial t}{\partial s}.$$

The vector $\left(\dot{x}, \dot{t}\right)$ is the tangent to the primary route, which can be calculated in a straightforward fashion, and we impose the normalization criterion

$$||\dot{x}(s)||^2 + \left|\dot{t}(s)\right|^2 = 1$$

to determine the length, along with a procedure for selecting the sign (a good heuristic is to keep the angle between this tangent and the previous tangent smaller than $\pi/2$). Since we can now compute the "time derivative" of the system we can use any decent ODE solver to trace the primary route (such as Runge-Kutta; see below).

6. As t approaches 1 x will approach a zero of $F(x) = 0$.

Note that path-following methods are slow, since they solve many problems along the way to the solution of interest. But no method is slower than not finding the solution at all.

7.5 The Roots of Polynomials

Polynomials are special functions that require some particular care; generally, we are interested in all the solutions of a polynomial system, rather than simply one of them. Specialized tools are available to solve these kinds of systems.

Consider the polynomial system

$$\begin{aligned} f_1(x_1, ..., x_n) &= 0 \\ &\vdots \\ f_n(x_1, ..., x_n) &= 0. \end{aligned}$$

If f_i is linear for all i, then there are only three possibilities: unique solution, no solution, or infinitely many solutions. Almost all linear systems will have a unique solution.

For other polynomials, there exist generically only finitely many solutions.

Definition 66 *Given a set of polynomials in n variables $\{f_1, ..., f_m\}$, the **ideal** generated by $\{f_1, ..., f_m\}$ is the set of all polynomials of the form*

$$p(x_1, ..., x_n) = g_1(x_1, ..., x_n) f_1(x_1, ..., x_n) + \cdots + g_m(x_1, ..., x_n) f_m(x_1, ..., x_n)$$

where g_i are polynomials in n variables.

Ideals are denoted

$$\langle f_1, ..., f_m \rangle;$$

an ideal is essentially a basis for polynomials in which the weights are themselves polynomials. Ideals are useful because all members share the same zeros: the point $(a_1, ..., a_n)$ is a zero of the polynomials $\{f_1, ..., f_m\}$ if and only if it is a zero of every polynomial in $\langle f_1, ..., f_m \rangle$.

A polynomial in $(x_1, ..., x_n)$ is the sum of terms of the form $cx_1^{k_1} x_2^{k_2} \cdots x_n^{k_n}$ for nonnegative integers k_j and $c \neq 0$. The product term is a **monomial**. The **degree** of a term is the sum of the k_j. We call the term of a polynomial with the highest degree the **leading term**; if there are more than one with the same degree, we just choose some ordering within the set of leading terms (commonly one uses the lexicographic ordering starting from x_1).

Definition 67 *The set $\{h_1, ..., h_k\}$ is a **Grobner basis** for the ideal $\langle f_1, ..., f_m \rangle$ if (1) $\langle f_1, ..., f_m \rangle = \langle h_1, ..., h_k \rangle$ and (ii) the leading term of any polynomial in $\langle f_1, ..., f_m \rangle$ is divisible by the leading term of h_i for some i.*

Define $\overline{f}^F$ as the operation of dividing f by F, where

$$F = \{f_1, f_2, ..., f_s\} \subset k[x_1, x_2, ..., x_n]$$
$$f \in k[x_1, x_2, ..., x_n].$$

Polynomial division is defined by $f/g = h$ satisfying

$$f = qg + h$$

for some polynomial q such that the leading term of f is eliminated. Dividing by a set F is division by each term sequentially. The definition implies that F is a Grobner basis of I iff $\overline{S(f_i, f_j)}^F = 0$ for all pairs i, j, where the S-polynomial $S(f_i, f_j)$ is defined by

$$S(f_i, f_j) = \frac{x^g}{LM(f_i)} \frac{f_i}{LC(f_i)} - \frac{x^g}{LM(f_j)} \frac{f_j}{LC(f_j)}$$
$$x^g = \frac{LM(f) LM(g)}{GCD(LM(f), LM(g))}.$$

$LT(f)$ is the leading term of f, $LM(f)$ is the leading monomial of f, and $LC(f)$ is the leading coefficient of f. $GCD(f, g)$ is the greatest common divisor of f and g. All of these terms are defined according to some ordering.

Buchberger's algorithm is used to construct a Grobner basis. First choose a pair f_i, f_j and compute $h = \overline{S(f_i, f_j)}^F$. If $h = 0$, go to the next pair. If $h \neq 0$, adjoin h to F. Then restart using the enlarged F. If the S polynomials are all zero, stop.

Let I be an ideal with two generators $\langle f_1, f_2 \rangle$ and use the lexicographic ordering $x > y$; that is, first take the term with the highest power of x, then if there are two take the term with the highest power of y. Suppose

$$f_1[x, y] = xy - x$$
$$f_2[x, y] = x^2 - y.$$

Then

$$LT(f_1) = xy$$
$$LT(f_2) = x^2$$
$$LM(f_1) = xy$$
$$LM(f_2) = x^2$$
$$LC(f_1) = 1$$
$$LC(f_2) = 1$$
$$GCD(LM(f_1), LM(f_2)) = x$$
$$x^g = \frac{x^3 y}{x} = x^2 y.$$

Then

$$\begin{aligned} S(f_1, f_2) &= \frac{x^g}{LM(f_1)} f_1 - \frac{x^g}{LM(f_2)} f_2 \\ &= \frac{x^2 y}{xy}(xy - x) - \frac{x^2 y}{x^2}\left(x^2 - y\right) \\ &= -x^2 + y^2. \end{aligned}$$

Now divide by f_2:

$$\begin{aligned} \left(-x^2 + y^2\right) &= q\left(x^2 - y\right) + h \\ &= -1\left(x^2 - y\right) + h \\ y^2 &= y + h \\ h &= y^2 - y. \end{aligned}$$

Since y^2 is not divisible by $LM(f_1)$ or $LM(f_2)$, we now have basis $\{f_1 = xy - x, f_2 = x^2 - y, f_3 = y^2 - y\}$ and

$$\begin{aligned} -x^2 + y^2 &= q\left(y^2 - y\right) + h \\ &= y^2 - y + h \\ h &= -x^2 + y \\ -x^2 + y &= q\left(x^2 - y\right) + h \\ &= -\left(x^2 - y\right) + h \\ h &= 0. \end{aligned}$$

Also

$$\begin{aligned} S(f_1, f_3) &= \frac{xy^2}{xy}(xy - x) - \frac{xy^2}{y^2}\left(y^2 - y\right) = 0 \\ S(f_2, f_3) &= \frac{x^2y^2}{x^2}\left(x^2 - y\right) - \frac{x^2y^2}{y^2}\left(y^2 - y\right) = x^2y - y^3. \end{aligned}$$

Then

$$\begin{aligned} x^2y - y^3 &= q\left(x^2 - y\right) + h \\ &= y\left(x^2 - y\right) + h \\ h &= -y^3 + y^2 \\ -y^3 + y^2 &= q\left(y^2 - y\right) + h \\ &= -y\left(y^3 - y\right) + h \\ h &= 0. \end{aligned}$$

Therefore, $\{f_1, f_2, f_3\}$ is a Grobner basis.

If we can write the Grobner basis in the form

$$\{x_1 - q_1(x_n), ..., x_{n-1} - q_{n-1}(x_n), q_n(x_n)\}$$

where q_i is a univariate polynomial, we have a **shape basis**. The shape basis system is not only triangular but is basically already solved – first, find all the solutions to $q_n(x_n) = 0$, then substitute each of them into the other equations to get every possible solution. Not every Grobner

basis will come out as nice as a shape basis; for example, the system we have above does not deliver a shape basis, but nevertheless is triangular and easy to solve:

$$y^2 - y = 0$$

implies

$$y \in \{0, 1\}$$

and

$$x^2 - y = 0$$

implies

$$x = \{0\}$$

or

$$x \in \{-1, 1\}.$$

Therefore, we have three distinct solutions $\{(0,0), (-1,1), (1,1)\}$. However, almost all polynomial systems do have shape bases.

Lemma 68 *(Shape Lemma) There exists an open dense subset S of all polynomial systems of n equations in n unknowns such that, for every system*

$$\begin{aligned} f_1(x_1, ..., x_n) &= 0 \\ &\vdots \\ f_n(x_1, ..., x_n) &= 0, \end{aligned}$$

there exists a polynomial system of the form

$$\begin{aligned} x_1 - q_1(x_n) &= 0 \\ &\vdots \\ x_{n-1} - q_{n-1}(x_n) &= 0 \\ q_n(x_n) &= 0. \end{aligned}$$

Reducing polynomial systems to shape bases is an efficient way to solve them, since they reduce the last polynomial to a univariate one. The next step is to find a robust computational method

for obtaining the roots of a univariate polynomial. Linear univariate systems are obviously rather easy to solve:

$$ax + b = 0 \Rightarrow x^* = -\frac{b}{a}.$$

Provided a and b are not too close to zero, this calculation is simple; if they are both small, a simple renormalization fixes the problem (just multiply them by big numbers!). Quadratic equations are also easy to solve using the formula:

$$ax^2 + bx + c = 0 \Rightarrow x^* = -\frac{b}{2a} \pm \frac{1}{2a}\sqrt{b^2 - 4ac};$$

this expression can be numerically sensitive, so generally you should use the alternative but equivalent expression

$$x^* = -\frac{2c}{b \pm \sqrt{b^2 - 4ac}}.$$

There are analytical formulas for cubic (Cardano's formula) and quartic equations (Ferraro's formula), and a theorem (the Abel-Ruffini theorem) that shows no such formula exists for quintic equations and above. These formulas are usually more trouble than they are worth, as they involve repeated square roots. Therefore, rather than use them, we will proceed directly to study routines that will give us all the roots of general polynomials; remember that often these roots will have non-zero imaginary components, so your program should be prepared to handle that possibility. If your coefficients are real, these roots will come in conjugate pairs; if the coefficients are complex, the roots will generically also be complex.

The starting point for finding all the roots is the fundamental theorem of algebra.

Theorem 69 *(Fundamental Theorem of Algebra) Every non-zero, univariate, degree-n polynomial with complex coefficients has exactly n complex roots, including multiplicities.*

This theorem allows us to determine exactly how many solutions exist; once we find that many, we can quit. General nonlinear systems do not have this feature, and therefore finding all the solutions is generically impossible.

The state of the art at solving polynomial equations is the Aberth-Ehrlich method, which converges cubically (although it only converges linearly to repeated roots, as usual for methods based on Newton-Raphson). Let

$$p(x) = p_n x^n + p_{n-1} x^{n-1} + \cdots + p_0$$

be the polynomial of interest (the coefficients can be complex). Then we can factor this polynomial into

$$p(x) = p_n (x - z_1^*) \cdots (x - z_n^*)$$

where the z_i^* terms are the complex roots that we are trying to compute. First, we can establish upper and lower bounds on their values. The most common bound is due to Cauchy:

$$|z_i^*| \leq 1 + \max \left\{ \left| \frac{p_{n-1}}{p_n} \right|, \left| \frac{p_{n-2}}{p_{n-1}} \right|, \cdots, \left| \frac{p_0}{p_1} \right| \right\};$$

this bound is not particularly tight, and for some cases tighter bounds are known. For example, if 1 is larger than the sum of all $\left| \frac{p_i}{p_n} \right|$ except the largest such term, then Lagrange obtained a tighter bound:

$$|z_i^*| \leq \max \left\{ 1, \sum_{i=0}^{n-1} \left| \frac{p_i}{p_n} \right| \right\};$$

this situation is relatively rare. Because these bounds are not invariant to rescaling the polynomial, it is sometimes useful to use an alternative that is invariant: $|z_i^*|$ is smaller than the sum of the two largest terms in the sequence

$$\left\{ \left| \frac{p_{n-1}}{p_n} \right|, \left| \frac{p_{n-2}}{p_n} \right|^{\frac{1}{2}}, \cdots, \left| \frac{p_0}{p_n} \right|^{\frac{1}{n}} \right\}.$$

Lower bounds are simply the reciprocal of the upper bounds.

If your coefficients are real, we can reduce Cauchy's formula to a bound on the positive real roots

$$|z_i^*| \leq 1 + \max_i \left\{ -\frac{p_i}{p_n} \right\}.$$

The bounds on the negative roots are obtained by substituting $p(-x)$ for $p(x)$, which amounts to changing the sign of the odd coefficients. If this bound is not larger than one, all nonzero coefficients have the same sign and therefore no positive roots exist.

We can also obtain the number of roots on a particular interval $[a, b]$, using Sturm's theorem. A **Sturm sequence** for polynomial f satisfies

$$f_0 = f$$

$$f_1 = Df$$

$$f_n = -rem(f_{n-2}, f_{n-1})$$

where $rem\,(f,g)$ is the remainder obtained by Euclidean division of f_{n-2} by f_{n-1}, which computes (q_n, r_n) such that

$$f_{n-2} = f_{n-1}q_n + r_n$$

with $\deg(r) < \deg(f_{n-1})$ and sets $f_n = -r_n$. An algorithm to compute the Euclidean division step is to input (a,b) and then set

$$\begin{aligned} q &= 0 \\ r &= a \\ d &= \deg(b) \\ c &= lc(b), \end{aligned}$$

where $lc(b)$ is the leading coefficient of b (coefficient on term of degree d). While the condition $\deg(r) \geq d$ holds, iterate on the following steps:

$$\begin{aligned} s &= \frac{lc(r)}{c} x^{\deg(r)-d} \\ q &= q + s \\ r &= r - sb. \end{aligned}$$

Once the condition $\deg(r) < d$ is satisfied, return (q,r).

Theorem 70 *(Sturm's Theorem) Let f be a polynomial with Sturm sequence $\{f_0, ..., f_k\}$ and real numbers $a < b$, neither of which are roots of f. The number of roots on the interval $[a,b]$ equals the number of sign changes of $f_0(a), ..., f_k(a)$ minus the number of sign changes of $f_0(b), ..., f_k(b)$.*

Sturm's theorem tells us precisely how many real roots of a polynomial lie in a particular interval; for example, if we choose the interval $[-1,1]$, we can determine how many stable real roots we have, although we would not have the number of stable complex roots. To do an example, suppose we have the polynomial

$$f(x) = x^4 + x^3 - x - 1;$$

the real roots of this polynomial are $\{-1, 1\}$. The Sturm sequence is

$$\begin{aligned}
f_0 &= x^4 + x^3 - x - 1 \\
f_1 &= 4x^3 + 3x^2 - 1 \\
f_2 &= \frac{3}{16}x^2 + \frac{3}{4}x + \frac{15}{16} \\
f_3 &= -32x - 64 \\
f_4 &= -\frac{3}{16}.
\end{aligned}$$

To find the number of roots in $[-3, 0]$, we look at the sign changes at -3 and 0:

$$\begin{aligned}
f_0(-3) &= (-3)^4 + (-3)^3 - (-3) - 1 = 56 \\
f_1(-3) &= 4(-3)^3 + 3(-3)^2 - 1 = -82 \\
f_2(-3) &= \frac{3}{16}(-3)^2 + \frac{3}{4}(-3) + \frac{15}{16} = \frac{3}{8} \\
f_3(-3) &= -32(-3) - 64 = 32 \\
f_4(-3) &= -\frac{3}{16}
\end{aligned}$$

has 3 sign changes, and

$$\begin{aligned}
f_0(0) &= (0)^4 + (0)^3 - (0) - 1 = -1 \\
f_1(0) &= 4(0)^3 + 3(0)^2 - 1 = -1 \\
f_2(0) &= \frac{3}{16}(0)^2 + \frac{3}{4}(0) + \frac{15}{16} = \frac{15}{16} \\
f_3(0) &= -32(0) - 64 = -64 \\
f_4(0) &= -\frac{3}{16}
\end{aligned}$$

has 2 sign changes, so f has $3 - 2 = 1$ root on $[-3, 0]$. Similarly, on $[0, 3]$ we have

$$\begin{aligned}
f_0(3) &= (3)^4 + (3)^3 - (3) - 1 = 104 \\
f_1(3) &= 4(3)^3 + 3(3)^2 - 1 = 134 \\
f_2(3) &= \frac{3}{16}(3)^2 + \frac{3}{4}(3) + \frac{15}{16} = \frac{39}{8} \\
f_3(3) &= -32(3) - 64 = -160 \\
f_4(3) &= -\frac{3}{16},
\end{aligned}$$

so f has $2-1=1$ root on $[0,3]$.

Another point to make before proceeding to the algorithm involves the separation of roots; multiple roots create problems for all root-finders, and I have already mentioned that they reduce the convergence rate from cubic to linear for the Aberth-Ehrlich method. For the computer, "nearly equal" may be treated as equal, so it is useful to have some idea of the minimal separation between roots. We have a bound due to Mignotte:

$$\min_{i,j,i\neq j}\left\{\left|z_i^*-z_j^*\right|\right\}>\frac{\sqrt{3}\Delta\left(p\right)}{n^{\frac{n}{2}+1}\left(\left\|p\right\|_2\right)^{n-1}}$$

where Δ is the discriminant of p and $\left\|\cdot\right\|_2$ is the Euclidean norm. If this distance is too small, our root-finding operations will be slow.

To begin the root-finding operation, we take an initial guess inside the bounds (in the absence of additional information, uniformly spread guesses seem reasonable; however, you should avoid symmetric guesses because they can inhibit convergence in the rare case where the roots themselves are symmetric). We then compute the offset terms

$$w_i=\frac{\frac{p(z_i)}{Dp(z_i)}}{1-\frac{p(z_i)}{Dp(z_i)}\sum_{j\neq i}\frac{1}{z_i-z_j}}$$

and update the guesses as

$$z_i-w_i.$$

This formula is basically a version of Newton-Raphson adapted for polynomials. One can update the coefficients in a Gauss-Jacobi manner (all at once) or in a Gauss-Seidel manner (sequentially updating each root using the new guesses obtained in the current iteration); performance does not seem to depend on which option you choose. The Aberth-Ehrlich method will identify simple and multiple roots.

Note that the location of polynomial roots can be very sensitive to the coefficients; while they are continuous, that is of no comfort if the slope is very high. The classic example is Wilkinson's 20th order polynomial:

$$w\left(x\right)=\left(x-1\right)\left(x-2\right)\cdots\left(x-20\right).$$

This polynomial is horrendously ill-conditioned; changing coefficient on the 19th term from -210 to -210.0000001192 changes the location of the $x=20$ root to roughly 20.8 and causes the $x=18$

and $x = 19$ roots to merge into a repeated root at roughly 18.62, despite the fact that the original roots are separated by 1. A slightly larger change then changes this repeated real root into a complex conjugate pair around $19.5 \pm 1.9i$. While it is unlikely you will encounter a 20th order polynomial, this sensitivity is present in a lot of polynomials and may not be apparent ex ante. We can get a sense of the sensitivity by adding a polynomial $\epsilon q(x)$ to $p(x)$ and using a Taylor expansion:

$$\widetilde{x}_j = x_j^* + \frac{dx_j^*}{d\epsilon}\epsilon,$$

where

$$\frac{dx_j^*}{d\epsilon} = -\frac{q\left(x_j^*\right)}{Dp\left(x_j^*\right)}.$$

One expects to see problems if $|\epsilon|$ exceeds the radius of convergence for this expansion, which in turn is given by the smallest value of $|\epsilon|$ needed to create a double root at $\widetilde{x}_j$ (which puts a zero in the denominator of the derivative term unless $q\left(x_j^*\right) = 0$, and in that case creates an equally-bad $\frac{0}{0}$ problem). A crude rule-of-thumb for this radius is half the distance from x_j^* to the nearest root divided by the derivative term. For Wilkinson's polynomial, the small roots are not sensitive, but the large ones are; for other polynomials the reverse can be true. One way to avoid the problem is to express the polynomial in an alternative basis; we will see that there exist classes of polynomials for which the roots have known formulas and that are very stable later in the course. To obtain these polynomials one simply needs to regroup the terms.

Matrix polynomial equations are more complicated than systems of polynomials (see Ceria and Ríos-Rull 1993). To be concrete, take the quadratic matrix equation

$$P(D) \equiv AD^2 + BD + C = 0,$$

where D is an $n \times n$ matrix; a matrix that satisfies this equation is called a **solvent**. The **quadratic eigenvalue problem** is to find a scalar λ such that

$$\det(P(\lambda I)) = \det\left(A\lambda^2 + B\lambda + C\right) = 0,$$

and any such scalar is called a **latent root** of $P(\lambda I)$ and a vector b such that

$$P(\lambda I)\, b = 0$$

is called a **latent vector**. $P(\lambda I)$ has at most $2n$ latent roots, and strictly fewer if A is singular. Unfortunately, there is no matrix analogue to the fundamental theorem of algebra, as $P(D) = 0$ can have any number of solvents, including zero and infinitely-many. However, for certain kinds of problems we can establish exactly how many solvents exist.

First, we note that every eigenvalue of a solvent D is a latent root of $P(\lambda I)$. Lancaster (1966) shows the following useful result.

Theorem 71 *If $P(\lambda I)$ has n linearly independent latent vectors, then $Q\Lambda Q^{-1}$ is a solvent of P, where $Q = [b_1, ..., b_n]$ and Λ is a diagonal matrix of the latent roots.*

In fact, one can show that any diagonalizable solvent takes this form. Since almost every matrix is diagonalizable, almost every solvent takes this form. Then we have the following statement about the number of solutions.

Theorem 72 *If $P(\lambda I)$ has $2n$ distinct latent roots and every set of n latent vectors are linearly independent, then there are exactly $\binom{2n}{n}$ different solvents of $P(D) = 0$.*

One can take the derivative of the solvent:

$$DP(D) = (AD + B) \otimes I + A \otimes D,$$

which is a $n^2 \times n^2$ matrix. If DP is singular for some D, then small changes in the (A, B, C) matrices may cause unbounded changes in the solvent. Davis (1983) describes a solution method, called SQUINT (Solving the Quadratic by Interating Newton Triangularizations), that can find solvents by iterating from an initial guess; if we know how many solutions there are, we can find all of them by starting from different initial guesses.

SQUINT iterates on

$$D_{k+1} = D_k - T_k$$

where

$$T_k = DP(D_k)^{-1} P(D_k),$$

which is clearly a Newton-type iteration scheme; T_k solves

$$(AD_k + B) T_k + AT_k D_k = P(D_k)$$

and can be computed using a combination of the Schur (QZ) decomposition and a QR decomposition. Specifically, we find unitary matrices (Q, Z) such that

$$Q\left(AD_k + B\right)Z = U$$
$$QAZ = V$$

are both upper triangular and a matrix R such that

$$R^H D_k R = L$$

is lower triangular. Then we can write the matrix polynomial as

$$\left(Q^H U Z^H\right) T_k + \left(Q^H V Z^H\right) T_k \left(RLR^H\right) = P\left(D_k\right)$$

We then compute

$$Y = Z^H T R$$

in the system

$$UY + VYL = QP\left(D_k\right)R$$

and translate the solution back to

$$T_k = ZYR^H.$$

While you can provide an initial guess, it is not always easy to come up with one, so SQUINT can provide a decent one for you:

$$D_0 = \left(\frac{\|B\| + \sqrt{\|B\|^2 + 4\,\|A\|\,\|C\|}}{2\,\|A\|}\right) I$$

where $\|\cdot\|$ is the sup-norm; this matrix, which is obviously related to the quadratic formula, gives roughly the right magnitude for a possible solvent. As with all Newton methods, SQUINT can fail to converge.

7.6 Interval Arithmetic and Newton's Method

While in general we cannot find all the roots of nonlinear functions other than polynomials, we may be able to find all of them within an interval; the key is to use interval arithmetic, an extension

of standard operations to work directly on intervals. An **interval** $X = [\underline{x}, \overline{x}]$ consists of a lower bound $\underline{x}$, an upper bound $\overline{x}$, and all the points x such that $\underline{x} \leq x \leq \overline{x}$. An **interval vector** $\mathbf{X} = [X_1, ..., X_n]$ is an n-dimensional hyperrectangle where each X_i is an interval. Similarly, an $n \times m$ **interval matrix** $\mathbb{X} = [\mathbf{X}_1, ..., \mathbf{X}_m]$ consists of m interval vectors of length n.

Interval arithmetic works element-by-element. For example, take the usual arithmetic operators defined on sets X and Y:

$$\begin{aligned} X + Y &= \{x + y : x \in X, y \in Y\} \\ X - Y &= \{x - y : x \in X, y \in Y\} \\ X \times Y &= \{xy : x \in X, y \in Y\} \\ X \div Y &= \left\{ \frac{x}{y} : x \in X, y \in Y \right\}. \end{aligned}$$

Specializing to intervals we get

$$\begin{aligned} [x_1, x_2] + [y_1, y_2] &= [x_1 + y_1, x_2 + y_2] \\ [x_1, x_2] - [y_1, y_2] &= [x_1 - y_2, x_2 - y_1] \\ [x_1, x_2] \times [y_1, y_2] &= [\min\{x_1y_1, x_2y_1, x_1y_2, x_2y_2\}, \max\{x_1y_1, x_2y_1, x_1y_2, x_2y_2\}] \end{aligned}$$

for the first three cases. To handle division involving intervals that contain zero, we have to use an extension of interval arithmetic to include infinities:

$$[x_1, x_2] \div [y_1, y_2] = [x_1, x_2] \times \frac{1}{[y_1, y_2]}$$

where

$$\frac{1}{[y_1, y_2]} = \begin{cases} \left[\frac{1}{y_2}, \frac{1}{y_1}\right] & \text{if } 0 \notin [y_1, y_2] \\ \left[-\infty, \frac{1}{y_1}\right] \cup \left[\frac{1}{y_2}, \infty\right] & \text{if } 0 \in [y_1, y_2] \end{cases}.$$

Interval division can therefore produce more than one interval, both of which are treated as valid outcomes.

Other functions, such as logarithms, exponentiation, and powers work the same way (element-by-element):

$$\begin{aligned} a^{[x_1,x_2]} &= [\min\{a^{x_1}, a^{x_2}\}, \max\{a^{x_1}, a^{x_2}\}] \\ \log([x_1, x_2]) &= [\log(x_1), \log(x_2)] \text{ for positive intervals} \\ [x_1, x_2]^n &= [x_1^n, x_2^n] \text{ for odd } n. \end{aligned}$$

Even n requires some extra care, as the following example shows. Consider $[-1,1]^n$ for even n; we should get

$$[-1,1]^2 = [0,1]$$

but a straightforward use of repeated interval multiplication would yield $[-1,1]^2 = [-1,1]$, which is twice as big as it should be, and using the formula for odd n would deliver the degenerate interval $[1,1]$.

What is the right way to proceed? For monotonic functions, we need only consider the endpoints to obtain the interval outcome. For piecewise monotonic functions (those that are monotonic only on a subset of the interval), we need to include in our calculations the critical points where the function's slope changes sign; for $n = 2$ this requirement means we need to consider 0 in addition to $\{-1,1\}$. Doing so yields

$$[-1,1]^2 = \left[\min\left\{-1^2, 0^2, 1^2\right\}, \max\left\{-1^2, 0^2, 1^2\right\}\right] = [0,1]$$

as desired. For higher powers we need to include more critical points; for sines and cosines, we will need to deal with up to five points per interval.

For a function $f(x)$, the **interval extension** $F(x)$ encloses the range of $f(x)$:

$$F(x) \supseteq \{f(x) : x \in X\}.$$

We can compute $F(x)$ by using interval operations on f. The reason that $F(x)$ is not necessarily equal to the range of f, which might seem natural, is that interval arithmetic may have a 'dependency problem' in which expressions that appear more than once in an expression are evaluated at different values. To see what this problem involves, consider the function

$$f(x) = x^2 + x.$$

The range of this function on $[-1,1]$ is $\left[-\frac{1}{4}, 2\right]$. But the 'natural extension' is computed as

$$[-1,1]^2 + [-1,1] = [0,1] + [-1,1] = [-1,2],$$

which is substantially larger; the problem is that we really have computed the minimum and maximum of the function

$$g(x,y) = x^2 + y$$

over $(x, y) \in [-1, 1] \times [-1, 1]$. Sometimes we can fix this problem by rewriting the function to have x only appearing once:

$$f(x) = x^2 + x = \left(x + \frac{1}{2}\right)^2 - \frac{1}{4}.$$

Now we get

$$\begin{aligned} F(x) &= \left([-1, 1] + \frac{1}{2}\right)^2 - \frac{1}{4} \\ &= \left[-\frac{1}{2}, \frac{3}{2}\right]^2 - \frac{1}{4} \\ &= \left[0, \frac{9}{4}\right] - \frac{1}{4} \\ &= \left[-\frac{1}{4}, 2\right]. \end{aligned}$$

We can use interval methods to find multiple solutions to nonlinear equations. Suppose we are looking to solve

$$f(x) = 0$$

and f is at least C^1 on the interval X. Suppose also we have an interval extension of Df, denoted DF, and that $0 \notin DF(X)$ so that f has at most one root on X. The interval Newton operator is

$$N(X) = m - \frac{f(m)}{DF(X)} = \left\{m - \frac{f(m)}{z} : z \in DF(X)\right\}$$

where $m \in X$ (typically one uses the midpoint). We generate the sequence of intervals

$$\begin{aligned} &X_0 \\ &X_{n+1} = X_n \cap N(X_n). \end{aligned}$$

Theorem 73 *(Mean Value Theorem) Let $f : [a, b] \to \mathcal{R}$ be continuous on $[a, b]$ and C^1 on (a, b). There exists $c \in (a, b)$ such that*

$$Df(c) = \frac{f(b) - f(a)}{b - a}.$$

If $f(a) = f(b)$, then there exists c such that

$$Df(c) = 0.$$

The mean value theorem ensures that if f has a root in X_n, then that root remains in X_{n+1}; the interval X_{n+1} is no more than half the size of X_n. We also immediately obtain that if $X_0 \cap N(X_0) = \emptyset$, then there is no root in X_0.

If DF contains zero, we use extended interval division to obtain a union of intervals for $N(X_n)$; this procedure then automatically separates and identifies multiple roots and we can proceed to isolate all of them iteratively (the number of intervals could continue to expand depending on how many roots f has on X_0).

Let's do an explicit example:

$$f(x) = x^2 - 2.$$

We know that we have two roots on $[-2, 2]$, equal to $\pm\sqrt{2}$. Set $m = 0$ and do an interval Newton step to obtain

$$\begin{aligned}[-2,2] \cap \left(0 - \frac{1}{2 \times [-2,2]}(0-2)\right) &= [-2,2] \cap ([-\infty, -0.5] \cup [0.5, \infty]) \\ &= [-2, -0.5] \cup [0.5, 2].\end{aligned}$$

If we then use a separate Newton interval iteration on each of these subintervals, they will individually converge to $-\sqrt{2}$ and $\sqrt{2}$. Thus, if you can bound all the roots in the initial interval, we can efficiently hunt them all down; for polynomials at least we know how to do this bounding.

Why couldn't we just use regular-old Newton from different initial points? The problem is that Newton's method produces basins of attraction that are quite irregular at the boundaries. Let's consider the problem of finding all the real roots of the polynomial system

$$f(x, y) \equiv \begin{bmatrix} x^3 - y \\ y^3 - x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix};$$

the real roots are $\{(-1,-1), (0,0), (1,1)\}$. Now consider the following experiment – over the (x, y) plane, initialize Newton's method from every possible value (or at least a wide enough interval), and keep track of which root the process converges to (or if it diverges or converges to a complex root). We know that Newton's method converges to a root if "close enough", but what does close enough mean? What if we are not "close enough" to any root? In the figure you can see each root (the isolated black dots in each of the colored regions) is surrounded by a well-behaved basin of attraction, but the boundaries between these well-behaved areas are rather

complicated; the resulting basins of attraction are composed of many disconnected regions. That is, a small change in the initial condition can lead to a very different root; notice also that many initial conditions also lead to one of the six complex roots

$$\begin{aligned}
(x,y) &= (-0.70711 - 0.70711i, 0.70711 - 0.70711i) \\
(x,y) &= (-0.70711 + 0.70711i, 0.70711 + 0.70711i) \\
(x,y) &= (0.70711 - 0.70711i, -0.70711 - 0.70711i) \\
(x,y) &= (0.70711 + 0.70711i, -0.70711 + 0.70711i) \\
(x,y) &= (-i, i) \\
(x,y) &= (i, -i),
\end{aligned}$$

and some initial conditions will diverge. This behavior is not specific to Newton-Raphson either; with the exception of Halley's method, all the iterative root-finders have fractal boundaries for their basins of attraction (Kneisl 2001).

The INTLIB Fortran library implements interval arithmetic using overloaded operators, just as the automatic differentation methods did before. Giving a function f an input of type Interval produces an array with lower and upper bounds (a univariate function produces two numbers).

8 Perturbation Methods

One method of approximating unknown functions is to build them up as Taylor expansions, which are polynomials with the coefficients derived from the derivatives of the function at a single point.

We will need two theorems.

Theorem 74 *(Taylor's Theorem) If $f \in C^{n+1}[a,b]$ and x, $x_0 \in [a,b]$, then*

$$f(x) = f(x_0) + Df(x_0)(x - x_0) + \frac{1}{2}D^2 f(x_0)(x - x_0)^2 + ... + \frac{1}{n!}D^n f(x_0)(x - x_0)^n + \frac{1}{(n+1)!}D^{n+1} f(\xi)(x - x_0)^{n+1}$$

for some ξ between x and x_0.

Theorem 75 *(Implicit Function Theorem) If $H(x,y) : \mathcal{R}^n \times \mathcal{R}^m \to \mathcal{R}^m$ is C^k, $H(x_0, y_0) = 0$, and $D_y H(x_0, y_0)$ is nonsingular, there exists a unique C^k function $h : \mathcal{R}^n \to \mathcal{R}^m$ such that*

$y_0 = h(x_0)$ and for x near x_0 $H(x, h(x)) = 0$ and the derivatives of h are obtained by implicit differentiation of $H(x, h(x)) = 0$ and satisfy

$$Dh(x_0) = -\frac{D_x H(x_0, h(x_0))}{D_y H(x_0, h(x_0))}.$$

A **perturbation** is a small displacement along a fixed vector; a derivative is simply a perturbation of a function at a given point (in a principal direction):

$$(Df)(x) = \lim_{h \to 0} \left\{ \frac{f(x+h) - f(x)}{h} \right\}.$$

Take the Euler equation from the standard growth model:

$$Du(f(k_t) + (1-\delta)k_t - k_{t+1}) = \beta Du(f(k_{t+1}) + (1-\delta)k_{t+1} - k_{t+2})(Df(k_{t+1}) + 1 - \delta).$$

Implicitly this equation (plus tranversality) defines k_{t+1} as a function of k_t, which we will denote by $g(k_t)$. k_t defines the "state" and k_{t+1} is the "control" or "jump"; the terminology will reappear later, so it is useful to begin getting used to it now. Note that $k_{t+2} = g(k_{t+1}) = g(g(k_t))$, as the logic of the Euler equation requires that your anticipated saving tomorrow use the same optimal rule as your saving today.

Note that the Euler equation holds for every k_t; in particular, it must hold for perturbations of k_t, xo we can differentiate it. To simplify expressions, we will denote the Euler equation (as a function of k_t) by G:

$$G(k_t) = Du(f(k_t) + (1-\delta)k_t - g(k_t)) - \beta Du(f(g(k_t)) + (1-\delta)g(k_t) - g(g(k_t)))(Df(g(k_t)) + 1 - \delta).$$

This equation is called a "functional equation" because obtaining a solution means figuring out what function $g(k_t)$ sets $G(k_t) = 0$ at each k_t; a functional equation is an infinite number of equations "strung together" (try thinking of a function as just a very long vector of numbers). Now note that

$$\begin{aligned} G(k_t + \epsilon) &= 0 \\ G(k_t + \epsilon) - G(k_t) &= 0 \\ \frac{G(k_t + \epsilon) - G(k_t)}{\epsilon} &= 0 \end{aligned}$$

and take the limit to get

$$DG(k_t) = 0.$$

The only unknown in this equation is the value of the decision rule $g(k_t)$. However, since we have the composition term $g(g(k_t))$, in general we can't simply solve it for g; finding $g(k)$ at a given k requires knowing g at other values of k'. We have already seen a few special cases of the model exist where g can be constructed analytically, but for most models there does not exist an analytical representation of g (which does not mean that it doesn't exist, merely that we can't write it down). Instead, we will build an approximation to g using Taylor's theorem:

$$g(k_t) \approx g(k^*) + Dg(k^*)(k - k^*) + \frac{1}{2}D^2 g(k^*)(k - k^*)^2 + ...$$

To proceed, we need a particular value of k_t where a solution is readily available; fortunately, the steady state k^* provides us such a value, since $k^* = g(k^*) = g(g(k^*))$. The steady state is the solution to

$$G(k^*) = 1 - \beta(Df(k^*) + 1 - \delta).$$

The solution is then

$$k^* = (Df)^{-1}\left(\frac{1}{\beta} - 1 + \delta\right).$$

We will call this a "zero-order" solution:

$$g(k_t) = g(k^*).$$

Obviously, it is only correct exactly at the steady state, and the accuracy of our solution will deteriorate as we move away from k^*. We will discuss accuracy after constructing the higher-order terms of $g(k_t)$.

The next step is to compute the linear term:

$$g(k_t) = g(k^*) + Dg(k^*)(k_t - k^*).$$

To do so we will differentiate G; remember that $G(k_t) = 0$ will still hold at $k_t + \varepsilon$, so that

$$\begin{aligned}
DG(k_t) = {} & D^2u(f(k_t) + (1-\delta)k_t - g(k_t))(Df(k_t) + 1 - \delta - Dg(k_t)) - \\
& \beta D^2u(f(g(k_t)) + (1-\delta)g(k_t) - g(g(k_t)))(Df(g(k_t)) + 1 - \delta) \times \\
& ((Df(g(k_t)) + 1 - \delta)Dg(k_t) - Dg(g(k_t))Dg(k_t)) - \\
& \beta Du(f(g(k_t)) + (1-\delta)g(k_t) - g(g(k_t)))D^2 f(g(k_t))Dg(k_t) \\
& = 0.
\end{aligned}$$

Now suppose we set $k_t = k^*$ so that $g(k^*) = g(g(k^*)) = k^*$ and then we eliminate the arguments for clarity:

$$\begin{aligned} DG(k^*) = & -\beta D^2 u \cdot (Df + 1 - \delta) \cdot (Dg)^2 - \\ & \left(D^2 u + \beta D^2 u \cdot (Df + 1 - \delta)^2 + \beta Du \cdot D^2 f\right) Dg + D^2 u \cdot (Df + 1 - \delta) \\ & = 0. \end{aligned}$$

Thus, $DG(k^*)$ defines a quadratic equation in $Dg(k^*)$. Transversality eliminates one of the roots (in this model we always have a saddle point, so one root of the quadratic will be larger than one and can be discarded), so now we have the unique stable linear solution:

$$k_{t+1} = g(k_t) = k^* + Dg(k^*)(k_t - k^*).$$

Note that we built our policy function $g(k_t)$ using the true value of consumption,

$$c_t = h(k_t) = f(k_t) + (1 - \delta) k_t - g(k_t);$$

we could have simply treated this equation as another structural equation and built a joint approximation to $(g, h)(k_t)$. With multiple equations, we use the same idea in more dimensions to construct the linear system, which we can solve in a number of ways; we will discuss the most widely used, which are based on the spectral decomposition and the QZ decomposition of matrices. It is problem-specific which approach is more accurate, which is perhaps a bit surprising.

8.1 Spectral Decomposition

To begin, we need some definitions from linear algebra. We are after the solution to

$$AE[x_{t+1}|\mathcal{F}_t] = Bx_t + Cz_t$$

with conformable matrices A, B, and C and an adapted filtration $\{\mathcal{F}_t\}$ (a filtration is a nested sequence of σ-algebras that represent the flow of information over time, $\mathcal{F}_t \subset \mathcal{F}_{t+1}$). x and z are vectors and assume that

$$z_{t+1} = \Phi z_t + \epsilon_{t+1}.$$

Assume that A is nonsingular; then we can rewrite this equation as

$$E[x_{t+1}|\mathcal{F}_t] = A^{-1}Bx_t + A^{-1}Cz_t.$$

Nonsingularity of A requires that there are no purely static equations; if the model has these, we need to separately solve that subblock and eliminate them from the equation, leaving only dynamic equations. Now we will be more formal about the matrix operations we used to solve linear difference equations in the last chapter. Define the Hermitian transpose by A^H (transpose and take complex conjugates if necessary).

Definition 76 *Let A be a square matrix. A is **symmetric** if $A = A^T$. A is **normal** if $A^H A = AA^H$. A is **Hermitian** if $A^H = A$. A is **unitary** if $A^H = A^{-1}$.*

Definition 77 *Let A be a square matrix. Then A is **diagonalizable** if the eigenvector matrix P is linearly independent (equivalently there are no repeated eigenvalues). Matrices that are not diagonalizable are called **defective**.*

Theorem 78 *(Spectral Decomposition Theorem) Let A be a diagonalizable matrix; then $A = P\Lambda P^{-1}$, where Λ is a diagonal matrix of eigenvalues and P is the matrix of eigenvectors.*

If A is symmetric, then the decomposition takes the form

$$A = P\Lambda P^T.$$

If A is normal, then

$$A = P\Lambda P^H$$

where P is unitary. If in addition A is Hermitian, then Λ has only real entries. Normal matrices are never defective. Note also that diagonalizability is not connected to invertibility; the matrix

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

is obviously diagonalizable but is not invertible.

To begin solving the model, let's decompose $A^{-1}B$:

$$A^{-1}B = P\Lambda P^{-1}.$$

Sort the eigenvalues in increasing order, and decompose the vector x_t into "stable" s_t and "unstable" u_t components, according to whether the eigenvalues are larger or smaller than 1 in modulus.

We now have

$$E\left[\left.\begin{array}{c} s_{t+1} \\ u_{t+1} \end{array}\right| \mathcal{F}_t\right] = P\Lambda P^{-1}\left[\begin{array}{c} s_t \\ u_t \end{array}\right] + \left[\begin{array}{c} Q_1 \\ Q_2 \end{array}\right] z_t.$$

Now multiply by P^{-1} and define

$$\left[\begin{array}{c} \widetilde{s}_t \\ \widetilde{u}_t \end{array}\right] = P^{-1}\left[\begin{array}{c} s_t \\ u_t \end{array}\right].$$

Our linear system is now

$$E\left[\left[\begin{array}{c} \widetilde{s}_{t+1} \\ \widetilde{u}_{t+1} \end{array}\right]\middle| \mathcal{F}_t\right] = \left[\begin{array}{cc} \Lambda_1 & 0 \\ 0 & \Lambda_2 \end{array}\right]\left[\begin{array}{c} \widetilde{s}_t \\ \widetilde{u}_t \end{array}\right] + \left[\begin{array}{c} Q_1 \\ Q_2 \end{array}\right] z_t.$$

The two rows are decoupled, since Λ is diagonal. We can therefore solve the unstable row forward by repeated substitution:

$$\begin{aligned} \widetilde{u}_t &= \Lambda_2^{-1} E\left[\widetilde{u}_{t+1}|\mathcal{F}_t\right] - \Lambda_2^{-1} Q_2 z_t \\ &= \Lambda_2^{-1} E\left[\Lambda_2^{-1}\widetilde{u}_{t+2} - \Lambda_2^{-1} Q_2 z_{t+1}|\mathcal{F}_t\right] - \Lambda_2^{-1} Q_2 z_t \\ &= -\Lambda_2^{-1}\sum_{j=0}^{\infty} \Lambda_2^{-j} Q_2 \Phi^j z_t \end{aligned}$$

since Λ_2 has only eigenvalues larger than 1 and Φ has only eigenvalues smaller than 1. Taking $t = 0$ without loss of generality, we therefore have

$$\widetilde{u}_0 = -\Lambda_2^{-1}\sum_{j=0}^{\infty} \Lambda_2^{-j} Q_2 \Phi^j z_0.$$

Assuming the Blanchard-Kahn conditions hold, we can take the initial condition for the stable part as given; s_0 (not $\widetilde{s}_0$) is known. Then we have that

$$\begin{aligned} \widetilde{s}_0 &= P_{11}^* s_0 + P_{12}^* u_0 \\ \widetilde{u}_0 &= P_{21}^* s_0 + P_{22}^* u_0 \end{aligned}$$

where P_{ij}^* is a partition of P^{-1}. Since we know $\widetilde{u}_0$, we can solve for u_0:

$$u_0 = \left(P_{22}^*\right)^{-1}\widetilde{u}_0 - \left(P_{22}^*\right)^{-1} P_{21}^* s_0.$$

Now we can simulate the economy from this initial condition. We get a unique stable solution if our model satisfies the Blanchard-Kahn condition that the number of stable eigenvalues equals

the number of predetermined variables; if the number of stable eigenvalues is larger, we have local indeterminacy in which there are an infinity of stable paths. This case will be dealt with later.

You might be asking what do we do with systems that have repeated eigenvalues? First, if A is normal, we can still diagonalize it. If not, then we can use the Jordan decomposition, which is "almost diagonal" for defective matrices; we write

$$A = PJP^{-1}$$

where the matrix J which differs from diagonal only by having ones in some entries above the main diagonal. For a four-dimensional system with eigenvalues $\{\lambda_1, \lambda_1, \lambda_2, \lambda_3\}$, the Jordan matrix is

$$J = \begin{bmatrix} \lambda_1 & 1 & 0 & 0 \\ 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & \lambda_3 \end{bmatrix}.$$

Raising the Jordan matrix to an integer power is still easy:

$$J^n = \begin{bmatrix} \lambda_1^n & \binom{n}{1}\lambda_1^{n-1} & 0 & 0 \\ 0 & \lambda_1^n & 0 & 0 \\ 0 & 0 & \lambda_2^n & 0 \\ 0 & 0 & 0 & \lambda_3^n \end{bmatrix}$$

with the binomial coefficients

$$\binom{n}{k} = \prod_{i=1}^{k} \frac{n+1-i}{i};$$

if $n < 0$ we can use the identity

$$\binom{-n}{k} = (-1)^k \binom{n+k-1}{k}.$$

More generally, the Jordan canonical form of a matrix is divided into Jordan blocks:

$$J = \begin{bmatrix} J_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & J_n \end{bmatrix}$$

where each J_i has the above form. The number of blocks associated with a given eigenvalue is called the **geometric multiplicity**, which corresponds to the number of linearly-independent

eigenvectors for that eigenvalue, and the number of times an eigenvalue is repeated is called the **algebraic multiplicity**. To see the difference in the two numbers, consider the matrix

$$A = \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix}.$$

Both eigenvalues of this matrix are equal to 1, so $\lambda = 1$ has an algebraic multiplicity of 2; however, the eigenspace of $\lambda = 1$ is spanned by a single eigenvector, so it has a geometric multiplicity of 1. Since they are not equal, the matrix is defective; the Jordan canonical form is

$$J = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

We then therefore proceed as before; be aware that computing the Jordan canonical form is substantially less numerically-stable than the spectral decomposition. To see why, consider the matrix

$$A = \begin{bmatrix} 1 & 1 \\ \varepsilon & 1 \end{bmatrix}$$

with $\varepsilon \geq 0$. The Jordan form for $\varepsilon \neq 0$ is

$$J(\varepsilon) = \begin{bmatrix} 1 + \sqrt{\varepsilon} & 0 \\ 0 & 1 - \sqrt{\varepsilon} \end{bmatrix}$$

as the eigenvalues $\{1 \pm \sqrt{\varepsilon}\}$ are distinct; if $\varepsilon = 0$, the Jordan form is

$$J(0) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

The limit of $J(\varepsilon)$ as $\varepsilon \to 0$ is not $J(0)$; thus, we can get very different answers if the computer deems two eigenvalues to be equal or simply very close together, which in turn will generally depend on the conditioning number of the entire matrix. We therefore consider a better-behaved alternative, the QZ or generalized Schur decomposition.

8.2 QZ Decomposition

As before, we are after the solution to

$$AE[x_{t+1}|\mathcal{F}_t] = Bx_t + Cz_t$$

with conformable matrices A, B, and C and an adapted filtration $\{\mathcal{F}_t\}$. x and z are vectors and assume that

$$z_{t+1} = \Phi z_t + \epsilon_{t+1}.$$

For there to exist a stable solution to this equation, we need the condition

$$|Az - B| = 0$$

for some z in complex Euclidean space; otherwise, a solution will still exist but it may not be stable. Our model comes with predetermined variables k_t and jump variables d_t, such that

$$x_t = \begin{bmatrix} k_t \\ d_t \end{bmatrix};$$

a generalized version of the Blanchard-Kahn conditions gives us uniqueness if the number of predetermined variables k_t equals the number of stable generalized eigenvalues, which is what we need to define next.

Definition 79 *Let $P : \mathcal{C} \to \mathcal{C}^{n \times n}$ be a matrix-valued function of a complex variable (called a **matrix pencil**). The set of **generalized eigenvalues** $\lambda(P)$ is defined as*

$$\lambda(P) = \{z \in \mathcal{C} : |P(z)| = 0\}.$$

Generalized eigenvalues solve the problem

$$|Az - B| = 0$$

instead of

$$|Az - I| = 0.$$

We may also write them as $\lambda(A, B)$. The collection of eigenvalues is called the **spectrum** of the pencil.

Definition 80 *Let $P(z)$ be a matrix pencil. Then P is **regular** if $\exists x \neq 0 \in \mathcal{C}$ such that $Bx = \lambda Ax$.*

Regular matrix pencils have at least one nonzero eigenvector (similar to regular matrices); note that zero eigenvalues are perfectly fine.

Definition 81 *A square matrix Q is **orthogonal** if all columns are orthogonal: $q_i^T q_j = 0 \ \forall i, j$.*

Definition 82 *A square matrix Q is **orthonormal** if it is orthogonal and $q_i^T q_i = 1 \ \forall i$.*

Definition 83 *A complex matrix Q with orthonormal columns is called **unitary** and satisfies $Q^H Q = I$, where Q^H is the Hermitian transpose of Q (the Hermitian transpose is the regular transpose combined with conjugation of all complex entries).*

We then can state our decomposition theorem.

Theorem 84 *(Complex Schur Decomposition) Let A and B be $n \times n$ matrices of complex numbers such that $P(z) = Az - B$ is a regular matrix pencil. Then there exist unitary $n \times n$ matrices of complex numbers Q and Z such that*

1. *$QAZ = S$ is upper triangular;*
2. *$QBZ = T$ is upper triangular;*
3. *$\forall i$, s_{ii} and t_{ii} are not both zero;*
4. *$\lambda(A, B) = \left\{ \frac{t_{ii}}{s_{ii}} \right\}$;*
5. *The pairs (s_{ii}, t_{ii}) can be arranged in any order.*

We call any $|\lambda_i| > 1$ "unstable" (including ones that are infinite because s_{ii} can be zero), and we call $|\lambda_i| < 1$ "stable" (including zero). We will ignore the possibility of $|\lambda_i| = 1$; some models, in particular those with heterogeneous agents, can produce unit eigenvalues, but dealing with them requires a lengthy diversion into center manifolds.

To begin solving systems using this decomposition, define

$$y_t = Z^H x_t,$$

where Z^H is the conjugate transpose of Z (take the transpose and replace any complex numbers with their conjugates). Partition y such that

$$y_t = \begin{bmatrix} s_t \\ u_t \end{bmatrix};$$

these are the stable and unstable parts and have size n_s and n_u. Substitute in y_t for x_t and premultiply by Q:

$$SE\left[y_{t+1}|\mathcal{F}_t\right] = Ty_t + QCz_t.$$

Since S is upper-triangular we can write the partitioned system as

$$\begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix} E\left[\left.\begin{bmatrix} s_{t+1} \\ u_{t+1} \end{bmatrix}\right| \mathcal{F}_t\right] = \begin{bmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{bmatrix} \begin{bmatrix} s_t \\ u_t \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} Cz_t.$$

We then solve the unstable forward and the stable backward; note that the unstable part is detached from the stable part since S and T are upper triangular. Unlike the $P\Lambda P^{-1}$ spectral decomposition, the equations are still linked; thus the solution must proceed by solving the unstable part first and then using that solution to get the stable part.

The solution for the unstable part is

$$u_t = -\sum_{k=0}^{\infty} \left[T_{22}^{-1}S_{22}\right]^k T_{22}^{-1}Q_2CE\left[z_{t+k}|\mathcal{F}_t\right];$$

we do not need to worry about the invertibility of A for this solution to hold (remember in the co-state case we needed to have no leading coefficients on the $t+1$ terms). Proceeding recursively through the rows of this solution, we get

$$u_{it} = -\frac{1}{t_{ii}} \sum_{k=0}^{\infty} \left(\frac{s_{ii}}{t_{ii}}\right)^k r_i^T \Phi^k z_t$$

where

$$r_i^T = \sum_{j=i+1}^{n_u} \left(t_{ij}m_j^T - s_{ij}m_j^T\Phi\right) + g_i^T$$

and g is a row in Q_2C and m_j are rows in an unknown matrix M. It then follows that

$$\begin{aligned} u_{it} &= r_i^T \left(s_{ii}\Phi - t_{ii}I_n\right)^{-1} z_t \\ &= m_i z_t \end{aligned}$$

and therefore

$$u_t = Mz_t$$

Note: we could have computed M directly using the formula

$$vec\left(M\right) = \left[\left(\Phi^T \otimes S_{22}\right) - I \otimes T_{22}\right]^{-1} vec\left(Q_2C\right),$$

but as noted by Klein (2000) this formula is time-consuming to use because of the large matrix that needs to be inverted. Note: $vec(X)$ is the vector obtained from matrix X by stacking the columns and is equivalent to reshaping an $n \times n$ matrix into a $n^2 \times 1$ vector.

We now solve for s_t using the solution for u_t. The first row is

$$S_{11}E\left[s_{t+1}|\mathcal{F}_t\right] + S_{12}E\left[u_{t+1}|\mathcal{F}_t\right] = T_{11}s_t + T_{12}u_t + Q_1Cz_t$$

or

$$E\left[s_{t+1}|\mathcal{F}_t\right] = S_{11}^{-1}T_{11}s_t + S_{11}^{-1}T_{12}u_t - S_{11}^{-1}S_{12}E\left[u_{t+1}|\mathcal{F}_t\right] + S_{11}^{-1}Q_1Cz_t.$$

The expectation error for the stable or predetermined variables is exogenous and given by ξ_{t+1}:

$$\xi_{t+1} = k_{t+1} - E\left[k_{t+1}|\mathcal{F}_t\right].$$

We have

$$k_{t+1} = \begin{bmatrix} Z_{11} & Z_{12} \end{bmatrix} \begin{bmatrix} s_{t+1} \\ u_{t+1} \end{bmatrix}$$

which gives us

$$Z_{11}\left(s_{t+1} - E\left[s_{t+1}|\mathcal{F}_t\right]\right) + Z_{12}\left(u_{t+1} - E\left[u_{t+1}|\mathcal{F}_t\right]\right) = \xi_{t+1};$$

we will assume that Z_{11} is square (giving us the uniqueness result) and invertible. We then solve row-by-row again to obtain

$$s_{t+1} = S_{11}^{-1}T_{11}s_t + S_{11}^{-1}\left[T_{12}M - S_{12}M\Phi + Q_1C\right]z_t - Z_{11}^{-1}Z_{12}M\epsilon_{t+1} + Z_{11}^{-1}\xi_{t+1}.$$

Using the predetermined values k_0 and z_0 we obtain

$$s_0 = Z_{11}^{-1}\left[k_0 - Z_{12}Mz_0\right].$$

Converting back we obtain

$$\begin{aligned} d_t &= Z_{21}Z_{11}^{-1}k_t + Nz_t \\ k_{t+1} &= Z_{11}S_{11}^{-1}T_{11}Z_{11}^{-1}k_t + Lz_t + \xi_{t+1} \end{aligned}$$

where

$$\begin{aligned} N &= \left(Z_{22} - Z_{21}Z_{11}^{-1}Z_{12}\right)M \\ L &= -Z_{11}S_{11}^{-1}T_{11}Z_{11}^{-1}Z_{12}M + Z_{11}S_{11}^{-1}\left[T_{12}M - S_{12}M\Phi + Q_1C\right] + Z_{12}M\Phi. \end{aligned}$$

These expressions define the current controls d and the new states k.

Note the difference between the spectral and QZ decomposition methods – for the QZ we do not need to eliminate purely static equations and the decomposition is numerically robust to the presence of (nearly) repeated eigenvalues. However, the generalized Schur decomposition is more costly than the spectral decomposition, so the speedup depends critically on how many such equations need to be eliminated and how close the eigenvalues are to each other.

Lee and Park (2020a) note that the solution method still works if we use the standard Schur decomposition

$$A^{-1}B = Q\Omega Q^T$$

where

$$\Omega = \begin{bmatrix} \Omega_{11} & \Omega_{12} \\ 0 & \Omega_{22} \end{bmatrix}$$

is upper block-triangular (that is, Ω_{11} and Ω_{22} are not upper triangular) and Q is unitary. We can then apply the forward solution for the unstable part Ω_{22} as before, but working on matrices rather than individual rows. The savings in computing time is substantial. To use this method, we again need an invertible A, as with the spectral decomposition, but we can handle repeated eigenvalues.

The same authors (Lee and Park 2020b) note that the dynamics of the system do not actually depend on the behavior of the unstable block, because it gets zeroed out by the appropriate choice of initial condition. For certain questions, that observation implies that we can reduce the size of the system substantially. To understand their argument, I will write the system in the Sims (2002) notation

$$\Gamma_0 s_t = \Gamma_1 s_{t-1} + C + \Psi z_t + \Pi \eta_t,$$

where s_t are the models variables, z_t are the exogenous shocks, and η_t are the expectation errors. We can use real Schur decomposition to obtain

$$Q^T \Lambda Z^T = \Gamma_0$$
$$Q^T \Omega Z^T = \Gamma_1$$

where Λ is upper block-triangular and Ω is upper triangular. Multiplying by Q yields

$$\Lambda w_t = \Omega w_{t-1} + QC + Q\Psi z_t + Q\Pi \eta_t$$

which is partitioned into

$$\begin{bmatrix} \Lambda_{11} & \Lambda_{12} \\ 0 & \Lambda_{22} \end{bmatrix} \begin{bmatrix} w_{1,t} \\ w_{2,t} \end{bmatrix} = \begin{bmatrix} \Omega_{11} & \Omega_{12} \\ 0 & \Omega_{22} \end{bmatrix} \begin{bmatrix} w_{1,t-1} \\ w_{2,t-1} \end{bmatrix} + \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} (C + \Psi z_t + \Pi \eta_t).$$

The unstable block can be solved forward as usual,

$$w_{2,t} = (\Lambda_{22} - \Omega_{22})^{-1} Q_2 C,$$

which delivers the solution for the stable block:

$$w_{1,t} = \Lambda_{11}^{-1}\Omega_{11} w_{1,t-1} + (Q_1 - \Phi Q_2)\Psi z_t + \Big((Q_1 - \Phi Q_2) - (\Lambda_{12} - \Phi\Lambda_{22} - \Omega_{12} - \Phi\Omega_{22})(\Lambda_{22} - \Omega_{22})^{-1} Q_2\Big) C$$

where Φ satisfies

$$Q_1 \Pi = \Phi Q_2 \Pi.$$

We can then recover the dynamics of the system as a whole by $s_t = Z w_t$:

$$s_t = Z \begin{bmatrix} \Lambda_{11}^{-1}\Omega_{11} & 0 \\ 0 & 0 \end{bmatrix} Z^T s_{t-1} + Z \begin{bmatrix} Q_1 - \Phi Q_{22} \\ 0 \end{bmatrix} \Psi z_t + Z \begin{bmatrix} (Q_1 - \Phi Q_2) - (\Lambda_{12} - \Phi\Lambda_{22} - \Omega_{12} - \Phi\Omega_{22})(\Lambda_{22} - \Omega_{22})^{-1} Q_2 \\ (\Lambda_{22} - \Omega_{22})^{-1} Q_2 \end{bmatrix} C.$$

Suppose we intend to estimate the model using a state-space representation with observation equation

$$y_t = H s_t + R\varepsilon_t.$$

Note we can rewrite this equation in terms of w_t:

$$y_t = H Z_1 w_{1,t} + H Z_2 w_{2,t} + R\varepsilon_t.$$

Since $w_{2,t}$ is constant, we can drop it from the dynamic system completely, leaving us with the smaller system

$$\begin{aligned} y_t &= H Z_1 w_{1,t} + R\varepsilon_t \\ w_{1,t} &= \Lambda_{11}^{-1}\Omega_{11} w_{1,t-1} + (Q_1 - \Phi Q_2)\Psi z_t; \end{aligned}$$

I have rewritten the system to suppose the constant is zero (say, by writing all variables in terms of deviations from the steady state). Provided the elements of $w_{1,t}$ are all treated as latent

(unobservable), which can always be done by an appropriate choice of H and y_t (if some states are intended to be observable, we can define a new control variable that always equals that state and have a unit vector row in H), the system is substantially smaller and thus the solution is faster.

8.3 Second-Order Solution

Having obtained the linear approximation, we move on to the quadratic term. To keep things as simple as possible, I will return to the univariate system of the deterministic growth model.

Since $DG(k_t) = 0$ also holds for all k_t, we can differentiate it. The second derivative is extremely complicated, so I will skip directly to the "sans argument" version:

$$\begin{aligned}
D^2G(k^*) = {} & D^3u \cdot (Df + 1 - \delta - Dg)^2 + D^2u \cdot \left(D^2f - D^2g\right) - \\
& \beta D^3u \cdot (Df + 1 - \delta)\left((Df + 1 - \delta)Dg - Dg \cdot Dg\right)^2 - \\
& \beta D^2u \cdot D^2F \cdot Dg \cdot \left((Df + 1 - \delta)Dg - Dg \cdot Dg\right) - \\
& \beta D^2u \cdot (Df + 1 - \delta)\left(D^2F \cdot Dg \cdot Dg + (Df + 1 - \delta)D^2g - D^2g \cdot Dg \cdot Dg - Dg \cdot D^2g\right) - \\
& \beta D^2u \cdot D^2f \cdot Dg\left((Df + 1 - \delta)Dg - Dg \cdot Dg\right) - \beta Du \cdot D^3f \cdot Dg \cdot Dg - \beta Du \cdot D^2f \cdot D^2g.
\end{aligned}$$

This expression is linear in the unknown $D^2g(k^*)$, as we already know $g(k^*)$ and $Dg(k^*)$, and all subsequent derivatives will produce linear equations in the new derivative unknown $D^ig(k^*)$. Our decision rule is now

$$g(k_t) = k^* + Dg(k^*)(k_t - k^*) + \frac{1}{2}D^2g(k^*)(k_t - k^*)^2$$

We can solve systems by a notationally-burdensome but conceptually-straightforward extension of this approach; those of you who know tensor notation will appreciate the notational difficulties associated with derivatives of systems of equations (Gomme and Klein 2014 have a version that does not use tensors but instead "stacks" the derivative matrices).

Higher-order terms are straightforward extensions, with notational burdens increasing at each step. The key is that all of them are linear. One can also show (at least for the growth model) that the higher-order equations are "solvable" under very mild conditions; that is, the matrix A in the linear system

$$AD^{(k)}g(k^*) = b$$

is such that the solution is unique.

8.4 Stochastic Difference Equations

Now suppose that we have a random shock in our model, so that the model is stochastic (we'll use a productivity shock as is common):

$$\begin{aligned}
&Du\left(\exp\left(z_t\right)f\left(k_t\right)+\left(1-\delta\right)k_t-g\left(k_t\right)\right)- \\
&\beta E_t\left[Du\left(\exp\left(z_{t+1}\right)f\left(g\left(k_t\right)\right)+\left(1-\delta\right)g\left(k_t\right)-g\left(g\left(k_t\right)\right)\right)\left(\exp\left(z_{t+1}\right)Df\left(g\left(k_t\right)\right)+1-\delta\right)\right] \\
&=0.
\end{aligned}$$

It will be easier if we make explicit assumptions about the process for z_t (although we don't need to), so let's suppose

$$z_{t+1}=\rho z_t+\sigma\epsilon_{t+1}$$

with $\epsilon_{t+1}\sim N\left(0,1\right)$; note that the steady state is $z^*=0$. Now we can write out the expectation:

$$\begin{aligned}
&Du\left(\exp\left(z_t\right)f\left(k_t\right)+\left(1-\delta\right)k_t-g\left(k_t\right)\right)- \\
&\frac{\beta}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}\left[\begin{array}{c}Du\left(\exp\left(z_{t+1}\right)f\left(g\left(k_t\right)\right)+\left(1-\delta\right)g\left(k_t\right)-g\left(g\left(k_t\right)\right)\right)\times \\ \left(\exp\left(z_{t+1}\right)Df\left(g\left(k_t\right)\right)+1-\delta\right)\end{array}\right]\exp\left(-\left(\frac{z_{t+1}-\rho z_t}{2\sigma}\right)^2\right)d\epsilon_{t+1} \\
&=0
\end{aligned}$$

where we have substituted out for ϵ_{t+1} using the expression for z_{t+1}. Now clearly the "state" involves both k_t and z_t; to capture some "risk" effects we will also treat σ as a state, although one that does not change over time (note also that the only case we can solve explicitly is one where $\sigma=0$ since otherwise even if $z_t=0\ \forall t$ we have an integral over future values of z_{t+1}). Therefore, we can write the Euler equation in the form

$$G\left(k_t,z_t,\sigma\right)=0$$

and note that the steady state is

$$G\left(k^*,0,0\right)=0.$$

Note that the distribution of the error terms can be arbitrary; as long as it has a density function that we can differentiate and possesses a sufficient number of finite moments, we can solve the model using the perturbation approach.

This equation can also be differentiated assuming that all the functions have bounded derivatives in a neighborhood of the steady state; we take the derivatives with respect to k, z, and σ. σ becomes a measure of uncertainty; for each value of σ we get a different decision rule that applies to an economy with that level of variance in technology. It will be the case that first-order terms in σ will be zero, so that the decision rule takes the form

$$g(k_t, z_t, \sigma) \approx g(k^*, 0, 0) + D_1 g(k^*, 0, 0)(k_t - k^*) + D_2 g(k^*, 0, 0) z_t + \frac{1}{2} D_{11}(k^*, 0, 0)(k_t - k^*)^2 + D_{12} g(k^*, z^*, 0)(k_t - k^*) z_t + \frac{1}{2} D_{22}(k^*, 0, 0) z_t^2 + \frac{1}{2} D_{33} g(k^*, 0, 0) \sigma^2.$$

To see why take $D_\sigma G(k_t, z_t, \sigma)$ and evaluate it at $(k^*, 0, 0)$, which yields

$$D^2 u(c^*) D_\sigma g(k^*, 0, 0) - \beta D^2 u(c^*) \begin{pmatrix} Df(k^*) D_\sigma g(k^*, 0, 0) + (1-\delta) D_\sigma g(k^* 0, 0) - \\ Dg_k(k^*, 0, 0) D_\sigma g(k^*, 0, 0) - Dg_\sigma(k^*, 0, 0) \end{pmatrix} (Df(k^*) + 1 - \delta) = 0.$$

Collecting terms we get

$$D^2 u(c^*) - \beta D^2 u(c^*)(Df(k^*) - \delta - Dg_k(k^*, 0, 0))(Df(k^*) + 1 - \delta) = D^2 u(c^*) \left(D_k g(k^*, 0, 0) - \frac{1}{\beta} \right) > 0$$

using the fact that $\beta(Df(k^*) + 1 - \delta) = 1$ and $|D_k g(k^*, 0, 0)| < 1$; therefore, the only solution is $D_\sigma g(k^*, 0, \sigma) = 0$.

You can verify through direct calculation that we again get a quadratic in $D_1 g(k^*, 0, 0)$, but we get linear equations for all the other derivatives. Under the 'log-Cobb' setup where

$$u(c) = \log(c)$$
$$f(k) = Ak^\alpha$$
$$\delta = 1,$$

we can state the solvability condition for the second order term as

$$\rho^2 \neq \frac{1}{\alpha\beta}$$
$$\rho\alpha \neq \frac{1}{\alpha\beta}.$$

Note that stationarity ($|\rho| < 1$) is sufficient but not necessary for solveability (since $\alpha\beta < 1$ we have $\frac{1}{\alpha\beta} > 1$ and $\rho^2 < 1$); by continuity it would also be satisfied for depreciation rates close enough to 1. Note also that the solveability problem has nothing to do with the actual model,

only with the approximate solution; the model has a solution for any ρ, so these restrictions do not have economic content, but they may matter as we try to find those solutions. We can now define the "stochastic steady state" as the value $\overline{k}$ such that

$$\overline{k} = g\left(\overline{k}, 0, \sigma\right) = g\left(k^*, 0, 0\right) + D_1 g\left(k^*, 0, 0\right)\left(\overline{k} - k^*\right) + \frac{1}{2} D_{11}\left(k^*, 0, 0\right)\left(\overline{k} - k^*\right)^2 + \frac{1}{2} D_{33} g\left(k^*, 0, 0\right)\sigma^2;$$

$\overline{k}$ is the value of capital that would obtain in a model with variance σ if the shock realizations were always $z_t = 0$ but agents did not know that in advance.

There is nothing special about treating σ as a state; we can use any parameter. The following growth model has a closed-form solution:

$$u\left(c\right) = \log\left(c\right)$$
$$F\left(k\right) = Ak^{\alpha};$$

that solution is

$$k' = \alpha\beta A k^{\alpha}$$
$$v\left(k\right) = a + \frac{\alpha}{1 - \alpha\beta}\log\left(k\right).$$

The generic growth model that we consider instead takes the functional forms

$$u\left(c\right) = \frac{c^{1-\psi} - 1}{1 - \psi}$$
$$F\left(k\right) = Ak^{\alpha} + \left(1 - \delta\right)k;$$

the two coincide if $\psi = 1$ and $\delta = 1$. Therefore, to solve this model we could take an expansion in (k, ψ, δ) and evaluate everyting at $(k^*, 1, 1)$. We then obtain policy functions of the form $g\left(k, \psi, \delta\right)$ which are Taylor expansion polynomials. Why would we ever want to do this, since it seems to add extra terms? For some models, even the steady state cannot be solved except under special parameter values; for these models this approach is a useful method.

Clearly, perturbation is a bit of a pain to do by hand, as the derivative expressions get more and more complicated as we add orders or variables. There are several ways around having to do the derivatives by hand. First, we could use symbolic methods to construct the derivative expressions; Matlab, Maple, and Mathematica all have symbolic packages that can be used to generate expressions for derivatives of any order, and they can export these expressions in C

or Fortran syntax as well. Second, we can use one of the derivative approximation methods, although one needs to be careful about accuracy past first order; the most efficient method here is to use automatic differentiation.

8.5 Using the Solution

Having obtained an approximate solution, we can derive implications from the model. In particular, we can calculate moments relating to the "ergodic" distribution of the model, which is the space of (k_t, z_t) that is visited with positive probability (from an initial condition inside the set); if we start outside the set, we will experience a transient period of convergence, which can be eliminated by dropping some number of initial observations (sometimes called a "burn in" period). It is straightforward to compute moments using the linear solution, so we'll start there:

$$\begin{bmatrix} k_{t+1} \\ \widetilde{z}_{t+1} \end{bmatrix} = \begin{bmatrix} k^* - D_1 g\left(k^*, 0\right) k^* \\ 0 \end{bmatrix} + \begin{bmatrix} D_1 g\left(k^*, 0, 0\right) & D_2 g\left(k^*, 0\right) \\ 0 & \rho \end{bmatrix} \begin{bmatrix} k_t \\ \widetilde{z}_t \end{bmatrix} + \begin{bmatrix} 0 \\ \sigma \end{bmatrix} \epsilon_{t+1}.$$

If we draw a sequence for $\{\epsilon_t\}_{t=1}^T$ of sufficient length (using the randn function in Matlab), we can use this system (along with an initial condition, say $k_1 = k^*$ and $z_1 = 0$) to simulate a long series for (k_t, z_t). We then use the fact that sample moments converge to population moments; now we can compute the unconditional variance of k_t, for example. We can also note that the variance-covariance matrix for (k_t, z_t) must solve the discrete Lyapounov equation

$$\begin{bmatrix} \sigma_k^2 & \rho_{kz}\sigma_k\sigma_z \\ \rho_{kz}\sigma_k\sigma_z & \sigma_z^2 \end{bmatrix}$$
$$= \begin{bmatrix} D_1 g\left(k^*, 0, 0\right) & D_2 g\left(k^*, 0\right) \\ 0 & \rho \end{bmatrix} \begin{bmatrix} \sigma_k^2 & \rho_{kz}\sigma_k\sigma_z \\ \rho_{kz}\sigma_k\sigma_z & \sigma_z^2 \end{bmatrix} \begin{bmatrix} D_1 g\left(k^*, 0, 0\right) & D_2 g\left(k^*, 0\right) \\ 0 & \rho \end{bmatrix}^T + \begin{bmatrix} 0 & 0 \\ 0 & \sigma^2 \end{bmatrix}.$$

Simplifying the RHS gives

$$\begin{bmatrix} \sigma_k^2 & \rho_{kz}\sigma_k\sigma_z \\ \rho_{kz}\sigma_k\sigma_z & \sigma_z^2 \end{bmatrix} = \begin{bmatrix} \sigma_k^2 \left(D_1 g\right)^2 + 2\rho_{kz}\sigma_k\sigma_z \left(D_1 g\right)\left(D_2 g\right) + \sigma_z^2 \left(D_2 g\right)^2 & \rho\sigma_z^2 \left(D_2 g\right) + \rho\sigma_k\rho_{kz}\sigma_z \left(D_1 g\right) \\ \rho\sigma_z^2 \left(D_2 g\right) + \rho\sigma_k\rho_{kz}\sigma_z \left(D_1 g\right) & \sigma^2 + \rho^2\sigma_z^2 \end{bmatrix}$$

Matching terms yields

$$\begin{aligned}
\sigma_k^2 &= \sigma_k^2 \left(D_1 g\right)^2 + 2\rho_{kz}\sigma_k\sigma_z \left(D_1 g\right)\left(D_2 g\right) + \sigma_z^2 \left(D_2 g\right)^2 \\
\rho_{kz}\sigma_k\sigma_z &= \rho\sigma_z^2 \left(D_2 g\right) + \rho\sigma_k\rho_{kz}\sigma_z \left(D_1 g\right) \\
\sigma_z^2 &= \sigma^2 + \rho^2\sigma_z^2.
\end{aligned}$$

We can get

$$\sigma_z^2 = \frac{\sigma^2}{1-\rho^2},$$

which naturally matches the long-run variance of z_t. Substituting into the first two equations yields

$$\begin{aligned}
\sigma_k^2 &= \sigma_k^2 \left(D_1 g\right)^2 + 2\left(\frac{\sigma^2}{1-\rho^2}\right)^{\frac{1}{2}} \left(D_1 g\right)\left(D_2 g\right)\rho_{kz}\sigma_k + \frac{\sigma^2}{1-\rho^2}\left(D_2 g\right)^2 \\
\rho_{kz}\sigma_k\left(\frac{\sigma^2}{1-\rho^2}\right)^{\frac{1}{2}} &= \rho\frac{\sigma^2}{1-\rho^2}\left(D_2 g\right) + \rho\left(\frac{\sigma^2}{1-\rho^2}\right)^{\frac{1}{2}} \left(D_1 g\right)\sigma_k\rho_{kz}
\end{aligned}$$

which determines $\left(\sigma_k^2, \rho_{kz}\right)$. Analytical solutions exist to these equations:

$$\begin{aligned}
\sigma_k^2 &= \sigma^2 \left(D_2 g\right)^2 \frac{1+\rho D_1 g}{\left(1-\rho\right)\left(1+\rho\right)\left(1-D_1 g\right)\left(1+D_1 g\right)\left(1-\rho D_1 g\right)} \\
\rho_{kz} &= \sigma\rho\sqrt{1-\rho^2}\frac{D_2 g}{\sigma_k\left(1-\rho\right)\left(1+\rho\right)\left(1-\rho D_1 g\right)}.
\end{aligned}$$

Clearly they are not particularly enlightening, so normally one would simply solve these equations on the computer. One point to note is that the sign of the correlation between k and z depends only on $D_2 g$; if a positive TFP shock leads to more capital tomorrow, it will generate a positive correlation in the time series (which I suppose is pretty obvious). Note also that, since $\sigma_k^2 \geq 0$, we must have

$$\frac{1+\rho D_1 g}{1-\rho D_1 g} > 0;$$

we know that $1 > D_1 g > 0$ and $|\rho| < 1$, so that either

$$D_1 g < \rho < 1$$

or

$$-1 < \rho < D_1 g,$$

which pretty much exhausts all the possibilities. For general discrete Lyapounov equations of the form

$$\Sigma = A\Sigma A^T + Q$$

we solve it by "vectorizing" and inverting. First, use the vec operator and noting that

$$\text{vec}\left(ABC\right) = \left(C^T \otimes A\right)\text{vec}\left(B\right),$$

we obtain

$$\left(I - A^T \otimes A\right)\text{vec}\left(\Sigma\right) = \text{vec}\left(Q\right).$$

The equation has a unique positive semidefinite solution provided that A has only eigenvalues inside the unit circle, and it is given by

$$\begin{aligned}\text{vec}\left(\Sigma\right) &= \left(I - A^T \otimes A\right)^{-1}\text{vec}\left(Q\right) \\ &= \sum_{k=0}^{\infty} A^k Q\left(A^T\right)^k.\end{aligned}$$

We would normally not invert this matrix but instead do a QR decomposition, in which a matrix A is decomposed as

$$A = QR$$

for an orthogonal matrix Q and an upper triangular matrix R; nonsingular matrices have a unique QR decomposition if we impose the restriction that the diagonal elements of R must be positive. We can then compute the solution to

$$Ax = b$$

as

$$x = Q\left(R^T\right)^{-1} b.$$

Undoing the vectorization gives us the matrix Σ. We could also iterate on the summation representation, but that typically takes many more operations.

For second-order solutions we can do the same steps, but the expressions are more complicated. It is generally easier to get them by simulation (but not necessarily faster).

If we are interested in welfare, we can use the Bellman equation

$$v\left(k,z,\sigma\right) = u\left(\exp\left(z\right)f\left(k\right) + \left(1-\delta\right)k - g\left(k,z,\sigma\right)\right) + \frac{\beta}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty} v\left(g\left(k,z,\sigma\right),z',\sigma\right)\exp\left\{\frac{1}{2\sigma^2}\left(z'-\rho z\right)^2\right\}d\epsilon'$$

to obtain a second-order approximation to v as

$$v(k,z,\sigma) \approx v(k^*,0,0) + v_1(k^*,0,0)(k-k^*) + v_2(k^*,0,0)z + \frac{1}{2}v_{11}(k^*,0,0)(k-k^*)^2 + v_{12}(k^*,0,0)(k-k^*)z + \frac{1}{2}v_{22}(k^*,0,0)z^2 + \frac{1}{2}v_{33}(k^*,0,0)\sigma^2.$$

Average utility is therefore equal to $v(k^*,0,0) + \frac{1}{2}v_{33}(k^*,0,0)\sigma^2$.

8.5.1 Accuracy

For notational simplicity, let's define the solution to consumption as

$$c_t \equiv h(k_t) = f(k_t) + (1-\delta)k_t - g(k_t).$$

The next step is to determine how accurate our solution is. Since the Euler equation must hold for every k_t, we can simply check whether it does. Define the Euler error function $e(k_t)$ as

$$e(k_t) = \left| 1 - \frac{(Du)^{-1}(\beta Du(h(g(k_t)))(Df(g(k_t)) + 1 - \delta))}{h(k_t)} \right|;$$

expressing it in these terms means it can be interpreted in terms of the percentage loss of consumption equivalent, in welfare terms, to using the approximate decision rule instead of the true (and unknown) one that actually sets the Euler equation exactly to zero at every k_t. It is common to plot $\log_{10}(e(k_t))$, which results in the y-axis being in terms of powers.

With shocks to evaluate the approximation error we need to compute the integral, which we can do using Gauss-Hermite quadrature:

$$e(k_t,z_t,\sigma) = \left| 1 - (Du)^{-1} \left(\beta \sum_{i=1}^{n} \omega_i Du\left(h\left(g(k_t,z_t,\sigma), \sqrt{2}\sigma x_i + \rho z_t, \sigma\right)\right)(Df(g(k_t,z_t,\sigma)) + 1 - \delta) \right) \frac{1}{h(k_t,z_t,\sigma)} \right|$$

Sometimes we want a summary accuracy measure (one number that we can use to compare solution methods or models). To aggregate our errors we have several options. One is the maximum absolute error

$$EE_\infty = \max_{k_t \in \mathcal{K}, z_t \in Z} \{e(k_t,z_t,\sigma)\}$$

evaluated over a grid $\mathcal{K} \times \mathcal{Z}$. This approach yields the uniform approximation (L^∞ norm) error. Average deviations might be preferable if we are concerned more with a specific region:

$$EE_2 = \left[\sum_{i=1}^{\#\mathcal{K}} \sum_{j=1}^{\#\mathcal{Z}} e(k_i,z_i,\sigma)^2 \right]^{\frac{1}{2}}.$$

This approach yields the L^2 norm error. Ideally both would be small, but note that the L^2 norm could be small and the L^∞ be large if large errors are made in "small" areas of the state space. An alternative way to express the error is weighted relative to the invariant distribution:

$$EE_2^* = \left[\int_{\mathcal{K}}\int_{\mathcal{Z}} e\left(k,z,\sigma\right)^2 \Gamma\left(k_t,z_t\right)\right]^{\frac{1}{2}}.$$

in this case we concentrate our attention to points that the model will visit in the stationary distribution.

One might ask whether we know that the errors in the policy function are small if the Euler equation errors are small; Santos (2000) proves if the maximum Euler equation error is ε, then

$$\left\|g-\widehat{g}\right\|_\infty \leq \frac{2}{\eta\left(\frac{1}{\sqrt{\beta}}-1\right)\left(1-\sqrt{\beta}\right)}\sqrt{\frac{L}{\eta}}\varepsilon$$

where L is the sup-norm of the second derivative of the value function and η is the lower bound of that derivative. Thus, the errors of the Euler equation will be of the same order as the errors in our computed policy function. However, note also that if β is close to 1, the denominator will be large.

8.6 Problems with Higher-Order Perturbation

Using second-order solutions (or higher-order ones) in simulations can create problems, because second-order solutions have unstable dynamics sufficiently far from the steady state; remember that a parabola cannot be monotonic unless it is linear and therefore will have more than one steady state, and the "false" steady state may attract the system if it gets sufficiently far from the true steady state. Third-order solutions may actually make these problems worse; the false steady states (there can now be more than one) may actually get closer rather than further away, even though the accuracy of the solution near the true steady state improves. **Pruning** is one method for eliminating these unstable dynamics; see Kim, Kim, Schaumburg, and Sims (2003).

To see how pruning works, take the system

$$y_t = g\left(x_t,\sigma\right)$$
$$x_{t+1} = h\left(x_t,\sigma\right)+\sigma\eta\varepsilon_{t+1}$$

where ε is a standard multivariate normal (although it need not be) and η is the Cholesky decomposition of the variance-covariance matrix. The first-order solution is denoted

$$\begin{aligned} y_t^{(1)} &= g_x x_t^{(1)} \\ x_{t+1}^{(1)} &= h_x x_t^{(1)} + \sigma\eta\varepsilon_{t+1} \end{aligned}$$

and is identical to what we had before. However, we will build the second-order solution in a different way. Define the second-order solution as

$$x_t^{(2)} = x_t^f + x_t^s;$$

where x_t^f is the "first-order component" and x_t^s is the "second-order component". The second-order solution $x_t^{(2)}$ is

$$\begin{aligned} x_{t+1}^{(2)} &= h_x x_t^{(2)} + \frac{1}{2}H_{xx}\left(x_t^{(2)} \otimes x_t^{(2)}\right) + \frac{1}{2}h_{\sigma\sigma}\sigma^2 + \sigma\eta\varepsilon_{t+1} \\ &= h_x\left(x_t^f + x_t^s\right) + \frac{1}{2}H_{xx}\left(\left(x_t^f + x_t^s\right) \otimes \left(x_t^f + x_t^s\right)\right) + \frac{1}{2}h_{\sigma\sigma}\sigma^2 + \sigma\eta\varepsilon_{t+1} \\ y_t^{(2)} &= g_x x_t^{(2)} + \frac{1}{2}G_{xx}\left(x_t^{(2)} \otimes x_t^{(2)}\right) + \frac{1}{2}g_{\sigma\sigma}\sigma^2 \\ &= g_x\left(x_t^f + x_t^s\right) + \frac{1}{2}G_{xx}\left(\left(x_t^f + x_t^s\right) \otimes \left(x_t^f + x_t^s\right)\right) + \frac{1}{2}g_{\sigma\sigma}\sigma^2 \end{aligned}$$

where

$$\begin{aligned} x_{t+1}^f &= h_x x_t^f + \sigma\eta\varepsilon_{t+1} \\ y_t^f &= g_x x_t^f \end{aligned}$$

are the "first-order terms" (which are the same as the first-order solution), and

$$\begin{aligned} x_{t+1}^s &= h_x x_t^s + \frac{1}{2}H_{xx}\left(x_t^f \otimes x_t^f\right) + \frac{1}{2}h_{\sigma\sigma}\sigma^2 \\ y_t^s &= g_x x_t^s + \frac{1}{2}G_{xx}\left(x_t^f \otimes x_t^f\right) + \frac{1}{2}g_{\sigma\sigma}\sigma^2 \end{aligned}$$

are the "second-order terms". Note we have dropped the quadratic terms in x_t^s since these are fourth-order and the product terms in x_t^f and x_t^s as these are third-order; that is, we have "pruned" the system. These dropped terms are exactly those terms that cause problems in the unpruned

system. The state space is now expanded to be $\left[x_t^f, x_t^s\right]$:

$$\begin{aligned}
x_{t+1}^f &= h_x x_t^f + \sigma\eta\varepsilon_{t+1} \\
x_{t+1}^s &= h_x x_t^s + \frac{1}{2}H_{xx}\left(x_t^f \otimes x_t^f\right) + \frac{1}{2}h_{\sigma\sigma}\sigma^2 \\
y_t^s &= g_x\left(x_t^f + x_t^s\right) + \frac{1}{2}G_{xx}\left(x_t^f \otimes x_t^f\right) + \frac{1}{2}g_{\sigma\sigma}\sigma^2.
\end{aligned}$$

Third-order solutions are obtained as

$$\begin{aligned}
x_{t+1}^t &= h_x x_t^t + H_{xx}\left(x_t^f \otimes x_t^s\right) + \frac{1}{6}H_{xxx}\left(x_t^f \otimes x_t^f \otimes x_t^f\right) + \frac{1}{2}h_{\sigma\sigma x}\sigma^2 x_t^f + \frac{1}{6}h_{\sigma\sigma\sigma}\sigma^3 \\
y_t^t &= g_x\left(x_t^f + x_t^s + x_t^t\right) + \frac{1}{2}G_{xx}\left(x_t^f \otimes x_t^f + 2\left(x_t^f \otimes x_t^s\right)\right) + \\
&\quad \frac{1}{6}G_{xxx}\left(x_t^f \otimes x_t^f \otimes x_t^f\right) + \frac{1}{2}g_{\sigma\sigma}\sigma^2 + \frac{1}{2}g_{\sigma\sigma x}\sigma^2 x_t^f + \frac{1}{6}g_{\sigma\sigma\sigma}\sigma^3.
\end{aligned}$$

Note that the higher-order components are deterministic, the shock only enters into the first-order terms.

As a handy hint, the matrix H_{xx} can be obtained from the derivative array h_{xx} (which is three-dimensional) using

Hxx=reshape(hxx,nx,nx*nx).

The same obviously holds for G_{xx}. Similarly, the matrix H_{xxx} is

Hxxx=reshape(hxxx,nx,nx*nx*nx).

There are potential problems with pruning. First, we have increased the size of the state space, which is generally not a big deal unless we are trying to estimate the model and need to solve it many times. Second, we may have compromised accuracy, which is potentially a bigger problem; for example, even if the true function g is cubic, the pruned cubic solution will not be exact. Third, the control variables are no longer functions of the original state variables (they are multivalued since the components may differ for a given value of x_t), which makes interpretation a bit more challenging.

8.7 Stable Alternatives to Pruning

de Wind and den Haan (2012) suggest using a "perturbation plus" method to improve accuracy and keep stability. Take a standard functional equation

$$E_t\left[f\left(x_{t+1}, x_t, x_{t-1}, z_{t+1}, z_t\right)\right] = 0.$$

The goal is to solve for

$$x_t = g\left(x_{t-1}, z_t\right).$$

The problem is the presence of x_{t+1}. de Wind and den Haan (2012) suggest we eliminate x_{t+1} using the first-order solution

$$x_{t+1} = h_x x_t + h_z z_{t+1}.$$

Then we have the single nonfunctional equation

$$E_t\left[f\left(h_x x_t + h_z z_{t+1}, x_t, x_{t-1}, z_{t+1}, z_t\right)\right] = 0,$$

which can be solved using standard numerical methods for

$$x_t = g_1\left(x_{t-1}, z_t\right)$$

using numerical integration to evaluate the expectation. We can extend this approach by rolling forward one period, obtaining the pair of equations

$$\begin{aligned} E_t\left[f\left(x_{t+1}, x_t, x_{t-1}, z_{t+1}, z_t\right)\right] &= 0 \\ E_{t+1}\left[f\left(h_x x_{t+1} + h_z z_{t+2}, x_{t+1}, x_t, z_{t+2}, z_{t+1}\right)\right] &= 0 \end{aligned}$$

which can be solved for

$$\begin{aligned} x_t &= g_1\left(x_{t-1}, z_t\right) \\ x_{t+1} &= g_2\left(x_{t-1}, z_t\right). \end{aligned}$$

To simulate we only use $x_t = g_1\left(x_{t-1}, z_t\right)$; we ignore the perturbation solution and the second-period function g_2. This approach can be extended to a large number of periods, and thus the approximation should get better, but it becomes difficult because we have to solve a larger system of equations and compute larger integrals. Note that there is nothing special about using the first-order approximation; we could use any order solution we want. A modified version of their method rearranges the equation to obtain

$$x_t = E_t\left[f\left(x_{t+1}, x_t, x_{t-1}, z_{t+1}, z_t\right)\right]$$

and then substitutes out both x_{t+1} and x_t (only on the RHS) using the perturbation solution:

$$\begin{aligned} x_t &= E_t\left[f\left(h_x x_t + h_z z_{t+1}, x_t, x_{t-1}, z_{t+1}, z_t\right)\right] \\ &= E_t\left[f\left(h_x^2 x_{t-1} + h_x h_z z_t + h_z z_{t+1}, h_x x_{t-1} + h_z z_t, x_{t-1}, z_{t+1}, z_t\right)\right]. \end{aligned}$$

Now all we need to do is numerically integrate and we are done.

Another option is to use what den Haan (201?) calls "perturbation consistent weighting". Consider instead of the standard second-order expansion

$$x_t = g\left(x_{t-1}\right) = \eta_0 + \eta_1\left(x_{t-1} - x^*\right) + \eta_2\left(x_{t-1} - x^*\right)^2$$

we use an expansion of the form

$$x_t = g\left(x_{t-1}\right) = \omega_0 + \omega_1\left(x_{t-1} - x^*\right) + \omega_2\left(x_{t-1} - x^*\right)^2 \exp\left(-\left(x_{t-1} - x^*\right)^2\right).$$

What is remarkable is that (i) $\eta_i = \omega_i$ and (ii) the second approximation is both more accurate and globally stable (no pruning required here). You can even add a "weight" to the exponential to control the deviation from the standard second-order expansion. To do this, we take our initial equation (simplified)

$$f\left(x_{t-1}, x_t, x_{t+1}\right) = 0$$

and add a new equation and a new unknown y_t

$$x_t = y_t \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right) + \left(\eta_0 + \eta_1 x_{t-1}\right)\left(1 - \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right)\right)$$

where (η_0, η_1) come from the perturbation expansion of f. We solve both equations for a new expansion, and use

$$\begin{aligned} y_t &= \omega_0 + \omega_1 x_{t-1} \\ x_t &= \left(\omega_0 + \omega_1 x_{t-1}\right) \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right) + \left(\eta_0 + \eta_1 x_{t-1}\right)\left(1 - \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right)\right). \end{aligned}$$

You can vary α to get a good (small) error. An extension to quadratic yields

$$\begin{aligned} y_t &= \omega_0 + \omega_1 x_{t-1} + \omega_2 x_{t-1}^2 \\ x_t &= \left(\omega_0 + \omega_1 x_{t-1} + \omega_2 x_{t-1}^2\right) \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right) + \left(\eta_0 + \eta_1 x_{t-1} + \eta_2 x_{t-1}^2\right)\left(1 - \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right)\right). \end{aligned}$$

A recent paper by Blasques and Nientker (201?) develops a similar idea and explains how to choose α. They consider the expansion

$$x_t = g\left(x_{t-1}\right) = \omega_0 + \omega_1\left(x_{t-1} - x^*\right) + \omega_2\left(x_{t-1} - x^*\right)^2 \exp\left(-\alpha\left(x_{t-1} - x^*\right)^2\right).$$

They argue that one should choose

$$\exp\left(-\alpha\left\|\frac{x_{t-1}}{x^*}\right\|_2^2\right)$$

where

$$\alpha = -\log\left(1-\rho\left(h_x\right)\right) > 0$$

and $\rho(h)$ is the spectral radius (that is, the largest eigenvalue).

8.8 Singularities and Bifurcations

There is a problem that is often encountered when trying to solve portfolio problems – the portfolio decision is indeterminate if the variance ot the risky return is zero. Since all assets must have the same expected return, if there is no risk there can be no difference in real returns across assets, meaning that any asset is equally desireable. That is, any portfolio is a solution. But not any portfolio is the limit of portfolios as variance goes to zero, and perturbation methods will allow us to find that limiting portfolio. As an illustration, I will present the model from Judd and Guu (2001). The key definition is that of a bifurcation:

Definition 85 *Suppose that $H: X\times R\to X$ and $H^0\left(x^0,\epsilon_0\right)=0$. Then $\left(x^0,\epsilon_0\right)$ is a **bifurcation point** if there exist two sequences (y^n,ϵ_n) and (z^n,ϵ_n) such that $H\left(y^n,\epsilon_n\right)=H\left(z^n,\epsilon_n\right)=0$, $\lim_{n\to\infty}y^n=\lim_{n\to\infty}z^n=x^0$ and $\lim_{n\to\infty}\epsilon_n=\epsilon_0$ but $y^n\neq z^n\ \forall n$.*

The portfolio problem we will solve has a bifurcation when variance is zero, since all portfolios zero the first-order condition. The bifurcation theorem gives us a constructive method to locate these points.

Definition 86 *A function $f:\mathcal{R}^n\to\mathcal{R}^m$ is **diffeomorphic** to $g:\mathcal{R}^n\to\mathcal{R}^m$ at $z=z_0$ if and only if there exists a differentiable function $h:\mathcal{R}^n\to\mathcal{R}^n$ which is invertible at $z=z_0$ and $f\left(h\left(z\right)\right)=g\left(z\right)$ in some open neighborhood of z_0.*

Theorem 87 *(Bifurcation Theorem) Suppose that $H:\mathcal{R}\times\mathcal{R}\to\mathcal{R}$ and $H\left(x,\epsilon\right)=0$ for all x if $\epsilon=0$. Suppose that*

$$D_xH\left(x,0\right)=0=D_\epsilon H\left(x_0,0\right)$$

$$D_{x\epsilon}H\left(x_0,0\right)\neq 0$$

for some x_0. *Then (i) if* $D_{\epsilon\epsilon}H(x_0,0) \neq 0$ *there exists an open neighborhood* N *of* $(x_0,0)$ *and a function* $h(\epsilon)$ *with* $h(\epsilon) \neq 0$ *for* $\epsilon \neq 0$ *such that* $H(h(\epsilon),\epsilon) = 0$ *and locally* $H(x,\epsilon)$ *is diffeomorphic to* $\epsilon(\epsilon - x)$ *or* $\epsilon(\epsilon + x)$*; (ii) if* $D_{\epsilon\epsilon}H(x_0,0) = 0 \neq D_{\epsilon\epsilon\epsilon}H(x_0,0)$ *there exists an open neighborhood* N *of* $(x_0,0)$ *and a function* $h(\epsilon)$ *with* $h(\epsilon) \neq 0$ *for* $\epsilon \neq 0$ *such that* $H(h(\epsilon),\epsilon) = 0$ *and locally* $H(x,\epsilon)$ *is diffeomorphic to either* $\epsilon^2(\epsilon - x)$ *or* $\epsilon^2(\epsilon + x)$*; (iii) in either case* $(x_0,0)$ *is a bifurcation point.*

Roughly speaking, bifurcation points are parameter values at which the number or stability of solutions to a system qualitatively changes. For example, consider the quadratic equation

$$x^2 + a = 0;$$

if $a < 0$, there are two real solutions; if $a = 0$ there is one repeated solution; and if $a > 0$ there are no real solutions. Therefore, $a = 0$ is a bifurcation point. Similarly, consider the dynamic system

$$x_{t+1} = \mu x_t (1 - x_t)$$

on $[0,1]$ with $\mu > 0$. If $\mu < 1$, there is only one fixed point, $x^* = 0$, and it is globally stable:

$$\left.\frac{dx_{t+1}}{dx_t}\right|_{x^*=0} = \mu < 1.$$

If $\mu \in (1,3)$, there are two fixed points, 0 and $\frac{\mu-1}{\mu} > 0$, and the zero steady state is unstable while the positive one is stable:

$$\left.\frac{dx_{t+1}}{dx_t}\right|_{x^*=\frac{\mu-1}{\mu}} = 2 - \mu.$$

Therefore, $\mu = 1$ is a bifurcation point. For $\mu > 3$, the positive steady state becomes unstable, so $\mu = 3$ is also a bifurcation point; at $\mu > 3$ stable limit cycles appear. It might also be the case that the bifurcation requires you to vary more than one parameter at a time.

Suppose that an investor has wealth W to invest in a risky and a riskless asset with returns Z and R, respectively. Final wealth is given by

$$Y = W((1 - \omega)R + \omega Z)$$

and the goal is to maximize

$$E(u(Y)),$$

where $u(c)$ is C^∞, strictly increasing, and strictly concave. Assume that

$$Z = R + \epsilon z + \epsilon^2 \pi,$$

where $E[z] = 0$ and $\epsilon^2\pi$ is the risk premium (proportional to the variance $E\left[(\epsilon z)^2\right] = \epsilon^2\sigma_z^2$). The first-order condition for optimality is

$$0 = \epsilon W E\left\{Du\left(WR + \omega W\left(\epsilon z + \epsilon^2\pi\right)\right)(z + \epsilon\pi)\right\}$$

which I define as $G(\omega, \epsilon)$. Note that we have $G(\omega, 0) = 0\ \forall\omega$ but the solution is unique for all $\epsilon > 0$; that is, there is a bifurcation at $\epsilon = 0$ for some unknown ω^0.

Differentiating yields

$$G_\omega \cdot D\omega(\epsilon) + G_\epsilon = 0,$$

which we can do as long as everything is bounded (we need to pass a differentiation operator through an integration one, which only works if the function is uniformly continuous, that it has bounded derivatives on a closed interval). We therefore have

$$\begin{aligned} G_\omega &= E\left\{D^2u(Y)\,W(z + \epsilon\pi)^2\,\epsilon\right\} \\ G_\epsilon &= E\left\{D^2u(Y)\,W(\omega z + 2\omega\epsilon\pi)(z + \epsilon\pi) + Du(Y)\,\pi\right\}. \end{aligned}$$

At $\epsilon = 0$, we have $G_\omega = 0\ \forall\omega$. We therefore search for a point ω^0 that will allow us to use L'Hospital's rule to compute

$$D\omega = -\frac{G_\epsilon}{G_\omega},$$

which we can do if $\frac{G_\epsilon}{G_\omega} = \frac{0}{0}$. At $\epsilon = 0$ the second condition is

$$G_\epsilon = D^2u(Y)\,W\omega z + Du(Y)\,\pi$$

which implies

$$\omega^0 = -\frac{\pi}{\sigma_z^2}\frac{Du(WR)}{WD^2u(WR)}.$$

ω^0 is the limiting portfolio weight as $\epsilon \to 0$. Note that the limiting porfolio weight depends negatively on the absolute risk aversion coefficient $-\frac{D^2u(R)}{Du(R)}$ for normalized $W = 1$, which is a sensible result – more risk averse agents will place less of their wealth in risky assets. It also sensibly depends negatively on variance and positively on the risk premium.

To obtain $D\omega$ we differentiate to obtain

$$0 = G_{\omega\omega}\left(D\omega\right)^2 + 2G_{\omega\epsilon}\cdot D\omega + G_{\omega}\cdot D^2\omega + G_{\epsilon\epsilon}$$

where we set $W = 1$ and

$$\begin{aligned}
G_{\epsilon\epsilon} &= E\left\{D^3u\left(Y\right)\left(\omega z + 2\omega\epsilon\pi\right)^2\left(z+\epsilon\pi\right) + D^2u\left(Y\right)2\omega\pi\left(z+\epsilon\pi\right) + 2D^2u\left(Y\right)\left(\omega z + 2\omega\epsilon\pi\right)\pi\right\}\\
G_{\omega\omega} &= E\left\{D^3u\left(Y\right)\left(z+\epsilon\pi\right)^3\epsilon\right\}\\
G_{\omega\epsilon} &= E\left\{D^3u\left(Y\right)\left(\omega z + 2\omega\epsilon\pi\right)\left(z+\epsilon\pi\right)^2\epsilon + D^2u\left(Y\right)\left(z+\epsilon\pi\right)2\pi\epsilon + D^2u\left(Y\right)\left(z+\epsilon\pi\right)^2\right\}.
\end{aligned}$$

At $\epsilon = 0$ we have

$$\begin{aligned}
G_{\epsilon\epsilon} &= D^3u\left(R\right)\omega_0^2 E\left[z^3\right]\\
G_{\omega\omega} &= 0\\
G_{\omega\epsilon} &= D^2u\left(R\right)E\left[z^2\right]
\end{aligned}$$

so we can solve for

$$D\omega = -\frac{1}{2}\frac{D^3u\left(R\right)}{D^2u\left(R\right)}\frac{E\left[z^3\right]}{\sigma_z^2}\omega_0^2;$$

this term depends positively on **absolute prudence** $-\frac{D^3u(R)}{D^2u(R)}$ (from Kimball 1990) and the skewness of z. Under CRRA utility $D^3u > 0$, so that if z has a positive third moment it will be the case that higher ϵ leads to more investment in the risky asset; positive skewness implies that very high returns are more likely than very low ones, so households who are prudent will buy more risky assets.

To find $D^2\omega$ we differentiate again:

$$0 = G_{\omega\omega\omega}\left(D\omega\right)^3 + 3G_{\omega\omega\epsilon}\left(D\omega\right)^2 + 3G_{\omega\omega}\left(D\omega\right)\left(D^2\omega\right) + 3G_{\omega\epsilon\epsilon}D\omega + 3G_{\omega\epsilon}D^2\omega + G_{\omega}D^3\omega + G_{\epsilon\epsilon\epsilon}.$$

Note that at $\epsilon = 0$ we have $G_{\omega\omega\omega} = G_{\omega\omega} = G_{\omega} = 0$, since

$$G_{\omega\omega\omega} = E\left\{D^4u\left(Y\right)\left(z+\epsilon\pi\right)^4\epsilon\right\},$$

so that

$$D^2\omega = -\frac{3G_{\omega\omega\epsilon}\left(D\omega\right)^2 + 3G_{\omega\epsilon\epsilon}D\omega + G_{\epsilon\epsilon\epsilon}}{3G_{\omega\epsilon}}.$$

The new terms $G_{\omega\omega\epsilon}$, $G_{\omega\epsilon\epsilon}$, and $G_{\epsilon\epsilon\epsilon}$ depend on the fourth derivative of the utility function:

$$G_{\omega\omega\epsilon} = E\left\{D^4u\left(Y\right)\left(z+\epsilon\pi\right)^3\epsilon\left(\omega z+2\omega\epsilon\pi\right)+3D^3u\left(Y\right)\left(z+\epsilon\pi\right)^2\epsilon\pi+D^3u\left(Y\right)\left(z+\epsilon\pi\right)^3\right\}$$

$$G_{\omega\epsilon\epsilon} = E\left\{\begin{array}{c} D^4u\left(Y\right)\left(\omega z+2\omega\epsilon\pi\right)^2\left(z+\epsilon\pi\right)^2\epsilon+4D^3u\left(Y\right)\left(\omega z+2\omega\epsilon\pi\right)\left(z+\epsilon\pi\right)\epsilon\pi+ \\ 2D^3u\left(Y\right)\left(\omega z+3\omega\epsilon\pi\right)\left(z+\epsilon\pi\right)^2+2D^2u\left(Y\right)\left(3\pi^2\epsilon+2z\pi\right) \end{array}\right\}$$

$$G_{\epsilon\epsilon\epsilon} = E\left\{\begin{array}{c} D^4u\left(Y\right)\left(\omega z+2\omega\epsilon\pi\right)^3\left(z+\epsilon\pi\right)+6D^3u\left(Y\right)\left(\omega z+2\omega\epsilon\pi\right)\left(z+\epsilon\pi\right)\omega\pi+ \\ 3D^3u\left(Y\right)\left(\omega z+2\omega\epsilon\pi\right)^2\pi+6D^2u\left(Y\right)\omega\pi^2 \end{array}\right\}.$$

For CRRA functions $D^4u<0$, representing an aversion to kurtosis (fat tails in the distribution of consumption); in the literature the term $-\frac{D^4u}{D^3u}$ is called the coefficient of **absolute temperance**. At $\epsilon=0$ these coefficients become

$$\begin{aligned} G_{\omega\omega\epsilon} &= D^3u\left(Y\right)E\left\{z^3\right\} \\ G_{\omega\epsilon\epsilon} &= 2D^3u\left(Y\right)\omega_0E\left\{z^3\right\} \\ G_{\epsilon\epsilon\epsilon} &= D^4u\left(Y\right)\omega_0^3E\left\{z^4\right\}+3\left(2+\pi\right)D^3u\left(Y\right)\omega_0^2E\left\{z^2\right\}+6D^2u\left(Y\right)\omega_0\pi^2 \end{aligned}$$

which can be used to compute $D^2\omega$. We therefore have constructed the quadratic approximation to the decision rule

$$\omega\left(\epsilon\right)=\omega_0+D\omega\left(\epsilon\right)\epsilon+D^2\omega\left(\epsilon\right)\epsilon^2.$$

One can continue and obtain a term involving $-\frac{D^5u}{D^4u}$, which is called the coefficient of **absolute edginess**. I do not think anyone has gone further.

We can extend this method to investigate market equilibria as well. Suppose there are two traders, identified by $i\in\{1,2\}$, with wealth constraint

$$Y_i=\theta_iR\left(1+\epsilon z\right)+B_iR$$

where θ_i is the risky asset position and B_i is the risk-free bond position. The goal is to maximize

$$E\left[u_i\left(Y_i\right)\right]$$

subject to the budget constraint

$$\theta_ip+B_i=a_i+\theta_i^ep;$$

a and θ^e are endowments of dollars and shares, respectively. Market clearing requires

$$\theta_1+\theta_2=1.$$

As with the portfolio problem, we will look for a Taylor expansion local to $\epsilon = 0$, but now for both θ and p.

The first-order condition for θ_i is

$$H^i\left(\theta\left(\epsilon\right), p\left(\epsilon\right), \epsilon\right) \equiv E\left[Du_i\left(Y_i\right)\left(1 + \epsilon z - p\left(\epsilon\right)\right)\right] = 0;$$

stacking them and imposing $\theta_2 = 1 - \theta_1$ gives us two functional equations in $\theta_1\left(\epsilon\right)$ and $p\left(\epsilon\right)$. As before, we cannot apply the implicit function theorem directly, so we look for an expansion that will permit us to use the bifurcation theorem. However, things are more complicated now; implicitly differentiating with respect to ϵ produces

$$D_\theta H\left(\theta\left(\epsilon\right), p\left(\epsilon\right), \epsilon\right)\theta'\left(\epsilon\right) + D_p H\left(\theta\left(\epsilon\right), p\left(\epsilon\right), \epsilon\right)p'\left(\epsilon\right) + D_\epsilon H\left(\theta\left(\epsilon\right), p\left(\epsilon\right), \epsilon\right) = 0.$$

Note that $p\left(0\right) = 1$ by no arbitrage and $D_\theta H\left(\theta\left(0\right), p\left(0\right), 0\right) = 0$; thus, we must have

$$\begin{aligned} 0 &= D_p H\left(\theta\left(0\right), p\left(0\right), 0\right)p'\left(0\right) + D_\epsilon H\left(\theta\left(0\right), p\left(0\right), 0\right) \\ &= \left(E\left[z\right] - p'\left(0\right)\right)Du\left(Y_i\right). \end{aligned}$$

Therefore, we must have $p'\left(0\right) = 0$ to apply the bifurcation theorem; unfortunately, we have now pinned down both $p\left(0\right)$ and $p'\left(0\right)$, so that the Jacobian $D^2_{(\theta,p)}H$ is not a zero matrix. To fix this issue, we can write

$$p\left(\epsilon\right) = 1 + \epsilon E\left[z\right] + \frac{\epsilon^2}{2}\pi\left(\epsilon\right) = 1 + \frac{\epsilon^2}{2}\pi\left(\epsilon\right)$$

and solve the system

$$0 = E\left[Du_i\left(Y_i\right)\left(z - \frac{\epsilon}{2}\pi\left(\epsilon\right)\right)\right]$$

for $\left(\theta, p\right)\left(\epsilon\right)$. This system is degenerate at $\left(\theta, \pi\right)\left(0\right)$; in effect we are endogenously determining the risk premium rather than the price. The solution is

$$\begin{aligned} \theta\left(0\right) &= \frac{\tau_1}{\tau_1 + \tau_2} \\ \pi\left(0\right) &= -2\frac{R\sigma^2}{\tau_1 + \tau_2} \end{aligned}$$

where

$$\tau_i = -\frac{Du_i\left(Y_i\right)}{D^2u_i\left(Y_i\right)}$$

is risk tolerance. The risk premium is the ratio of the variance of risky wealth to the total amount of risk tolerance, and the portfolio share for agent 1 is just his share of total risk tolerance. Continuing we get

$$\theta'(0) = R\frac{\tau_1}{\tau_1+\tau_2}\frac{\tau_2}{\tau_1+\tau_2}(\rho_1-\rho_2)\frac{E\left[z^3\right]}{\sigma_z^2}$$
$$\pi'(0) = \frac{2R^2}{(\tau_1+\tau_2)^2}\left(\frac{\tau_1}{\tau_1+\tau_2}\rho_1+\frac{\tau_2}{\tau_1+\tau_2}\rho_2\right)\frac{E\left[z^3\right]}{\sigma_z^2}$$

where

$$\rho_i = -\frac{D^2u_i(Y_i)}{D^3u_i(Y_i)}$$

is skew tolerance. The risk premium increases with ϵ if skewness is negative and the amount depends on the skew tolerance relative to risk tolerance; the portfolio share of agent 1 increases with ϵ if agent 1 is more skew tolerant and skewness is positive or less skew tolerant and skewness is negative.

Dynamic versions of this model have been derived by Judd and Guu (2001) and Tallarini and Zin (2004); the latter is interesting because it involves an expansion not around the deterministic model, but around a model with stochastic labor income and no asset uncertainty. They are able to do this because their assumption of quadratic preferences means the permanent income model they consider has a closed-form solution even with stochastic labor income. Devereux and Sutherland (2011) and Tille and van Wincoop (200?) adopt related approaches to study international models with portfolio problems; they use second-order expansions to identify the steady-state portfolios and linear expansions for the rest of the model.

I will illustrate the Devereux-Sutherland approach. Consider a two-country model with preferences represented by in the home country

$$U_t = E_t\left[\sum_{\tau=0}^{\infty}\theta_\tau\frac{C_\tau^{1-\rho}}{1-\rho}\right]$$

where the discount factor evolves as

$$\theta_{\tau+1} = \theta_\tau\omega C_{A\tau}^{-\eta}$$

with C_A being aggregate consumption in the country and

$$0 \leq \eta < \rho$$
$$0 < \omega\overline{C}_A^{-\eta} < 1$$

with $\overline{C}_A$ being steady state aggregate consumption. The budget constraint is

$$\alpha_{1,t} + \alpha_{2,t} = \alpha_{1,t-1} r_{1,t} + \alpha_{2,t-1} r_{2,t} + Y_t - C_t,$$

where $\alpha_{i,t}$ is the portfolio holdings of country i equity. Defining

$$W_t = \alpha_{1,t} + \alpha_{2,t},$$

we can write the budget constraint as

$$\begin{aligned} W_t &= \alpha_{1,t-1} r_{xt} + \alpha_{2,t-1} W_t + Y_t - C_t \\ r_{x,t} &= r_{1,t} - r_{2,t}. \end{aligned}$$

Foreign variables have '$*$' superscripts, such as C_t^*; note that in equilibrium we must have $W_t = -W_t^*$. Assets are in zero supply:

$$\begin{aligned} \alpha_{1,t} + \alpha_{1,t}^* &= 0 \\ \alpha_{2,t} + \alpha_{2,t}^* &= 0. \end{aligned}$$

Endowments are the sum of capital and labor income:

$$\begin{aligned} Y_t &= Y_{K,t} + Y_{L,t} \\ \log\left(Y_{K,t}\right) &= \log\left(\overline{Y}_K\right) + \varepsilon_{K,t} \\ \log\left(Y_{L,t}\right) &= \log\left(\overline{Y}_L\right) + \varepsilon_{L,t} \end{aligned}$$

and the equivalent expressions for the foreign country.

Equity claim returns are given by

$$\begin{aligned} r_{1,t} &= \frac{Y_{K,t}}{Z_{E,t-1}} \\ r_{2,t} &= \frac{Y_{K,t}^*}{Z_{E,t-1}^*} \end{aligned}$$

where Z_E is the price of the claim. Goods market clearing requires

$$C_t + C_t^* = Y_t + Y_t^*.$$

In a nonstochastic environment, we must have

$$r_{1,t+1} = r_{2,t+1},$$

which implies the portfolio is irrelevant; this condition also holds at first-order, given certainty equivalence:

$$E_t r_{1,t+1} = E_t r_{2,t+1}.$$

Now consider a second-order approximation to the portfolio choice equations

$$E_t\left[\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1} + \frac{1}{2}\left(\widehat{r}_{1,t+1}^2 - \widehat{r}_{2,t+1}^2\right) - \rho\widehat{C}_{t+1}\left(\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1}\right)\right] = 0$$
$$E_t\left[\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1} + \frac{1}{2}\left(\widehat{r}_{1,t+1}^2 - \widehat{r}_{2,t+1}^2\right) - \rho\widehat{C}^*_{t+1}\left(\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1}\right)\right] = 0.$$

We can rearrange these equations to get

$$E_t\left[\left(\widehat{C}_{t+1} - \widehat{C}^*_{t+1}\right)\left(\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1}\right)\right] = 0$$

and

$$E_t\left[\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1}\right] = -\frac{1}{2}E_t\left[\widehat{r}_{1,t+1}^2 - \widehat{r}_{2,t+1}^2\right] + \frac{\rho}{2}E_t\left[\left(\widehat{C}_{t+1} + \widehat{C}^*_{t+1}\right)\left(\widehat{r}_{1,t+1} - \widehat{r}_{2,t+1}\right)\right].$$

To evaluate the first equation, we note that second-order accurate solutions of products can be obtained using first-order accurate solutions of the individual terms. We also note that, to first order, the only component of the portfolio that affects consumption and excess returns is the steady state portfolio $\overline{\alpha}$.

The key is then to rearrange things to obtain

$$\widehat{r}_{x,t+1} = R_1\xi_{t+1} + R_2\varepsilon_{t+1}$$
$$\widehat{C}_{t+1} - \widehat{C}^*_{t+1} - \frac{\widehat{Q}_{t+1}}{\rho} = D_1\xi_{t+1} + D_2\varepsilon_{t+1} + D_3\begin{bmatrix} x_t \\ s_{t+1} \end{bmatrix},$$

where $\xi_t = \widetilde{\alpha}^T\widehat{r}_{x,t}$ is an iid vector random variable with $\widetilde{\alpha} \equiv \frac{\overline{\alpha}}{\overline{\beta Y}}$ (which is temporarily taken as exogenous), ε_t is a vector of structural shocks, x_t are the exogenous states, and s_t are the endogenous states. We can then solve for the steady state portfolio given ξ:

$$\widetilde{\alpha} = \left(R_2\Sigma D_2^T R_1 - D_1 R_2 \Sigma R_2^T\right)^{-1} R_2\Sigma D_2^T;$$

the steady state portfolio is therefore

$$\overline{\alpha} = \widetilde{\alpha}\overline{\beta Y}.$$

This solution may not be unique. If markets are complete, the expression is simpler:

$$\widetilde{\alpha} = \left(D_2^T R_1 - D_1 R_2\right)^{-1} D_2^T.$$

Note that this expression does not involve Σ, the variance-covariance matrix of the shocks.

More generally, suppose we have an approximate linear system

$$A_1 \begin{bmatrix} s_{t+1} \\ E_t\left[c_{t+1}\right] \end{bmatrix} = A_2 \begin{bmatrix} s_t \\ c_t \end{bmatrix} + A_3 x_t + B\xi_t$$

$$x_t = N x_{t-1} + \varepsilon_t;$$

B is a matrix that is unity in the rows corresponding to the excess returns $r_{x,t+1}$ and zero otherwise, and ξ_t is the innovation to wealth driven by portfolio choice. The solution to this system is

$$s_{t+1} = F_1 x_t + F_2 s_t + F_3 \xi_t$$
$$c_t = P_1 x_t + P_2 s_t + P_3 \xi_t.$$

Extracting the appropriate rows we get

$$\widehat{r}_{x,t+1} = R_1 \xi_{t+1} + R_2 \varepsilon_{t+1}.$$

Next, we impose the condition

$$\xi_{t+1} = \widetilde{\alpha}^T \widehat{r}_{x,t+1}$$

for some to-be-determined portfolio weight vector $\widetilde{\alpha}$. Therefore, we can obtain

$$\xi_{t+1} = \widetilde{H} \varepsilon_{t+1}$$
$$\widehat{r}_{x,t+1} = \widetilde{R} \varepsilon_{t+1}$$

where

$$\widetilde{H} = \frac{\widetilde{\alpha}^T R_2}{1 - \widetilde{\alpha}^T R_1}$$
$$\widetilde{R} = R_1 \widetilde{H} + R_2.$$

Similarly, we can obtain

$$\widehat{C}_{t+1} - \widehat{C}^*_{t+1} - \frac{\widehat{Q}_{t+1}}{\rho} = \widetilde{D} \varepsilon_{t+1} + D_3 \begin{bmatrix} x_t \\ s_{t+1} \end{bmatrix}$$

where

$$\widetilde{D} = D_1 \widetilde{H} + D_2.$$

Thus,

$$E_t \left[\left(\widehat{C}_{t+1} - \widehat{C}^*_{t+1} - \frac{\widehat{Q}_{t+1}}{\rho} \right) \widehat{r}_{x,t+1} \right] = \widetilde{R} \Sigma \widetilde{D}^T.$$

Thus, the equilibrium $\widetilde{\alpha}$ must solve

$$\widetilde{R} \Sigma \widetilde{D}^T = 0,$$

which after expanding terms yields

$$\widetilde{\alpha} = \left(R_2 \Sigma D_2^T R_1 - D_1 R_2 \Sigma R_2^T \right)^{-1} R_2 \Sigma D_2^T.$$

We can obtain the dynamics of portfolios by expanding the portfolio Euler equations to third-order, as in Devereux and Sutherland (2010).

8.8.1 Tallarini and Zin (2004)

Preferences are quadratic:

$$U_t = -\frac{1}{2} E_t \left[\sum_{j=0}^{\infty} \beta^j \left(c_{t+j} - b \right)^2 \right].$$

Each agent receives a stochastic stream of labor income $\{y_t\}$, which can be serially correlated but is required to be covariance stationary. At date t, the consumer decides wealth to be carried into the next period a_t; the return on this wealth depends on the allocation of assets between riskless and risky assets. The budget constraint is

$$a_t = R_t^p a_{t-1} + y_t - c_t$$

where

$$R_t^p = \sum_{n=0}^{N} \omega_{t-1}^n R_t^n$$

is the portfolio return. R^0 is the return on a riskless asset, and assume $\beta R^0 = 1$.

Risky returns and labor income are jointly normal:

$$\begin{bmatrix} y_t \\ R_t^1 \\ \vdots \\ R_t^N \end{bmatrix} = Hv_t$$

$$v_t = A_v v_{t-1} + G_v w_t.$$

The first-order conditions for saving and portfolio weight (1 risky asset and 1 riskless asset) are

$$\begin{aligned} b - c_t &= \beta E_t \left[(b - c_{t+1}) \left(\omega_t \left(R_{t+1}^e - R^f \right) + R^f \right) \right] \\ 0 &= \beta E_t \left[(b - c_{t+1}) \left(R_{t+1}^e - R^f \right) \right]. \end{aligned}$$

Parameterize the risky return as

$$R_t^e = R^f + \epsilon z_t + \epsilon^2 \pi.$$

Taking the second foc note that

$$H\left(\omega\left(x_t, \epsilon\right), \epsilon\right) = \beta E_t \left[\left(b - c\left(x_{t+1}, \epsilon\right) \right) \left(z_{t+1} + \epsilon\pi \right) \epsilon \right] = 0.$$

Rewriting a bit yields

$$H\left(\omega_t\left(\epsilon\right), \epsilon\right) = \beta E_t \left[\left(b - \left[\omega_t\left(\epsilon\right) \left(\epsilon z_t + \epsilon^2 \pi \right) + R^f \right] a_t\left(\epsilon\right) - y_{t+1} + a_{t+1}\left(\epsilon\right) \right) \left(z_{t+1} + \epsilon\pi \right) \right] = 0.$$

Indeterminacy arises because $H\left(\omega, 0\right) = 0 \ \forall \omega$.

Constraining z_t to be iid, we can apply the bifurcation approach to get

$$\omega_t^0 = \frac{\beta E_t \left[a_{t+1}^1 z_{t+1} \right] + \pi \beta E_t \left[b - c_{t+1}^0 \right]}{a_t^0 \sigma_z^2},$$

where c_{t+1}^0 comes from the budget constraint in the unperturbed model. Since $\beta R^f = 1$, we have

$$b - c_t^0 = \beta E_t \left[R^f \left(b - c_{t+1}^0 \right) \right] = E_t \left[b - c_{t+1}^0 \right]$$

so that

$$\omega_t^0 = \frac{\beta E_t \left[a_{t+1}^1 z_{t+1} \right] + \pi \beta \left(b - c_t^0 \right)}{a_t^0 \sigma_z^2}.$$

a^0 comes from the unperturbed PI model and a^1 is the first-order coefficient in a Taylor expansion for a (to be constructed).

Similar steps yield the first-order coefficient

$$\omega_t^1 = \frac{\pi\left(E_t\left[a_{t+1}^1\right] - R^f a_t^1\right) - \omega_t^0 a_t^1 \sigma_z^2 + \frac{1}{2} E_t\left[a_{t+1}^2 z_{t+1}\right]}{a_t^0 \sigma_z^2}.$$

And so on.

To get the term a_t^0 we use the unperturbed PI model; it is linear in a_t. Using the Euler equation we can get

$$\begin{aligned} & b - \left[\omega_{t-1}\left(R_t^e - R^f\right) + R^f\right] a_{t-1} - y_t + a_t \\ & = \beta E_t\left[\left(\omega_t\left(R_{t+1}^e - R^f\right) + R^f\right)\left(b - \left(\omega_t\left(R_{t+1}^e - R^f\right) + R^f\right) a_t - y_{t+1} + a_{t+1}\right)\right]. \end{aligned}$$

Substitute a Taylor expansion for a, ω, $R^e - R^f$ into this expression, differentiate wrt ϵ, and set $\epsilon = 0$:

$$R^f a_{t-1}^1 - \left(1 + \beta\left(R^f\right)^2\right) a_t^1 x_t + \beta R^f E_t\left[a_{t+1}^1\right] = \delta_t$$

where

$$\delta_t = -\omega_{t-1}^0 a_{t-1}^0 z_t + 2R^f \omega_t^0 a_t^0 E_t\left[z_{t+1}\right] - \beta b \omega_t^0 E_t\left[z_{t+1}\right] + \beta \omega_t^0 E_t\left[y_{t+1} z_{t+1}\right] - \beta \omega_t^0 E_t\left[a_{t+1}^0 z_{t+1}\right].$$

This expression is a second-order difference equation in a^1. We can factor it to obtain

$$\left(1 - L\right) a_t^1 = -\frac{1}{R^f}\left(1 - \frac{1}{R^f} L^{-1}\right)^{-1} \delta_t$$

or

$$a_t^1 - a_{t-1}^1 = -\beta E_t \sum_{j=0}^{\infty} \beta^j \delta_{t+j}.$$

Since z_t is iid we can evaluate the sum to obtain

$$\delta_t = -\omega_{t-1}^0 a_{t-1}^0 z_t + \beta \omega_t^0 E_t\left[\left(y_{t+1} - a_{t+1}^0\right) z_{t+1}\right].$$

Consolidating our approximations yields

$$\begin{aligned} \omega_t^0 &= v \\ \omega_t^1 &= \frac{\pi\left[\left(1 - \beta\right)\delta_t - \left(R^f - 1\right) a_{t-1}^1\right] - \omega_t^0 a_t^1 \sigma_z^2}{a_t^0 \sigma_z^2} \\ &= -\beta\left(\frac{\pi}{\sigma_z^2}\right)^2 \frac{\left(1 - \beta\right) a_t^0\left(b - c_{t-1}^0\right) z_t + a_t^1\left(b - c_t^0\right)\sigma_z^2}{\left(a_t^0\right)^2} - \left(R^f - 1\right)\frac{\pi}{\sigma_z^2}\frac{a_{t-1}^1}{a_t^0}. \end{aligned}$$

That is, the portfolio weight is

$$\begin{aligned}\omega_t &= \omega_t^0 + \omega_t^1 \epsilon \\ &= \frac{\pi}{\sigma_z^2}\frac{\beta\left(b - c_t^0\right)}{a_t^0} - \left(\beta\left(\frac{\pi}{\sigma_z^2}\right)^2 \frac{(1-\beta)\, a_t^0 \left(b - c_{t-1}^0\right) z_t + a_t^1 \left(b - c_t^0\right)\sigma_z^2}{\left(a_t^0\right)^2} + \left(R^f - 1\right)\frac{\pi}{\sigma_z^2}\frac{a_{t-1}^1}{a_t^0}\right)\epsilon.\end{aligned}$$

8.8.2 Gauge Function Expansions

Standard perturbation expansions use Taylor series, implying that the unknown function would be written as

$$f(x_0) + Df(x_0)\,\epsilon + \frac{1}{2}D^2 f(x_0)\,\epsilon^2 + ...$$

around some point x_0 for small ϵ. For some problems this expansion does not exist. For example, the function

$$f(x) = e^{x^{\frac{1}{3}}}$$

does not have a Taylor expansion around 0 because it does not have a finite derivative there:

$$f'(x) = x^{-\frac{2}{3}} e^{x^{\frac{1}{3}}}$$

is infinite at $x = 0$. But an alternative expansion does exist:

$$e^{x^{\frac{1}{3}}} = 1 + x^{\frac{1}{3}} + \frac{1}{2}\left(x^{\frac{1}{3}}\right)^2 + \frac{1}{6}\left(x^{\frac{1}{3}}\right)^3 + ...$$

is an expansion at $x = 0$. The difference is the gauge functions – the component functions used to build the approximation – in the Taylor expansion take the form x^k for integer k, but sometimes we might want k to not be an integer, like here.

Let's work with the portfolio problem, where the risky return is modified to be

$$Z = R + \epsilon z + \epsilon^\nu \pi$$

for some unknown ν. The first-order condition is

$$0 = E\left[Du\left(R + \left(\epsilon z + \epsilon^\nu \pi\right)\omega\right)\left(z + \epsilon^{\nu-1}\pi\right)\epsilon\right]$$

which defines the implicit function $G(\omega, \epsilon; \nu)$. Clearly $G(\omega, 0; \nu) = 0$ for any ω and $\nu > 0$. Differentiating implies

$$0 = G_\omega D\omega + G_\epsilon$$

where

$$G_\omega = E\left[D^2u\left(Y\right)\left(z+\epsilon^{\nu-1}\pi\right)^2\epsilon^{\nu-1}\right]$$
$$G_\epsilon = E\left[D^2u\left(Y\right)\left(\omega z+\nu\omega\epsilon^{\nu-1}\pi\right)\left(z+\epsilon^{\nu-1}\pi\right)+Du\left(Y\right)\epsilon^{\nu-2}\left(\nu-1\right)\pi\right]$$

and

$$Y = R+\left(\epsilon z+\epsilon^\nu\pi\right)\omega.$$

Searching for a bifurcation point means finding ω_0 such that

$$0 = \lim_{\epsilon\to 0}\left\{G_\epsilon\left(\omega_0,\epsilon;\nu\right)\right\}$$

so that we can use l'Hôspital's rule. We know that $\nu = 2$ works. We show now that $\nu = 1$ does not.

With $\nu = 1$ we have

$$0 = G_\omega D\omega + G_\epsilon$$

reducing to

$$0 = \left(D\omega+\omega\right)E\left[D^2u\left(Y\right)\left(z+\pi\right)^2\right].$$

Since the expectation is never zero, we must have

$$0 = D\omega+\omega.$$

Furthermore, the bifurcation expansion equation becomes

$$0 = E\left[D^2u\left(R\right)\left(z+\pi\right)^2\omega\right]$$

or

$$0 = D^2u\left(R\right)\omega_0\left(\sigma_z^2+\pi^2\right).$$

Solving this equation obviously requires $\omega_0 = 0$, which is the initial condition for the differential equation

$$0 = D\omega+\omega.$$

But then the solution is $\omega\left(\epsilon\right) = 0\ \forall\epsilon$, which cannot be true because it cannot satisfy the first-order condition unless $\pi = 0$:

$$0 = Du\left(R\right)\epsilon E\left[\left(z+\pi\right)\right]$$

would only be true if $\pi = 0$ (given $E(z) = 0$ and $\epsilon > 0$). Thus $\nu = 1$ clearly does not work.

What about $\nu \in (1, 2)$? In this case we obtain

$$G_\epsilon = E\left[D^2u(Y)\left(\omega z + \nu\omega\epsilon^{\nu-1}\pi\right)\left(z + \epsilon^{\nu-1}\pi\right) + Du(Y)\,\epsilon^{\nu-2}(\nu-1)\,\pi\right]$$

with a dominant term

$$E[Du(Y)]\,\epsilon^{\nu-2}(\nu-1)\,\pi.$$

As $\epsilon \to 0$ $\epsilon^{\nu-2}$ diverges to infinity, making satisfying the bifurcation equation impossible to satisfy. For $\nu > 2$ we get

$$\lim_{\epsilon\to 0}\{G_\epsilon\} = E\left[D^2u(R)\,\omega z^2\right] = 0$$

requires $\omega_0 = 0$, which we have already ruled out. Thus, pinning down ω_0 actually requires the gauge expansion to be in the correct powers, but fortunately this term is identified because it is the only one that works.

8.9 Sunspots and Limit Cycles

We have up to this point simply assumed the Blanchard-Kahn conditions are satisfied. What if they are not? We will start with the case in which the model has too many stable eigenvalues. For the growth model with one state and one control, we get a sink-stable outcome in this case; it cannot happen if the economy is efficient, but inefficiencies can produce these kinds of results. If the economy has a sink-stable solution, TVC does not pin down the root uniquely. Rather, all roots are permitted since they all satisfy TVC, so we can arbitrarily designate a "sunspot" that picks from them. As an example, let's consider the following economy with external production effects, taken from Benhabib and Farmer (1994). Assume that the production function of an individual firm is given by

$$y_t = S_t k_t^m l_t^{1-m}$$

and that technology is a function of total inputs:

$$S_t = \left[\int k_t^m l_t^{1-m}\right]^\theta.$$

Combining the two yields aggregate production as

$$Y_t = K_t^{m(1+\theta)} L_t^{(1-m)(1+\theta)}$$

which can display increasing returns to scale. Let the utility function be

$$u\left(c_t, l_t\right) = \log\left(c_t\right) - \frac{l_t^{1+\chi}}{1+\chi}.$$

The equations that describe the economy are then

$$\begin{aligned}
K_{t+1} &= K_t^{m(1+\theta)} L_t^{(1-m)(1+\theta)} + \left(1-\delta\right) K_t - C_t \\
C_t L_t^{\chi} &= \left(1-m\right) K_t^{m(1+\theta)} L_t^{(1-m)(1+\theta)-1} \\
\frac{1}{C_t} &= \beta E_t \left[\frac{1}{C_{t+1}} \left(1 - \delta + m K_{t+1}^{m(1+\theta)-1} L_{t+1}^{(1-m)(1+\theta)}\right)\right].
\end{aligned}$$

We can linearize this system and define a series of expectational errors w_{t+1} such that

$$\left[\begin{array}{c} \widehat{K}_t \\ \widehat{C}_t \end{array}\right] = J \left[\begin{array}{c} \widehat{K}_{t+1} \\ \widehat{C}_{t+1} \end{array}\right] + R \left[\begin{array}{c} 0 \\ w_{t+1} \end{array}\right]$$

where

$$w_{t+1} = \left[\begin{array}{c} E_t \widehat{K}_{t+1} - \widehat{K}_{t+1} \\ E_t \widehat{C}_{t+1} - \widehat{C}_{t+1} \end{array}\right].$$

L_t has been eliminated using the static condition.

Suppose that J has two eigenvalues outside the unit circle, meaning that J^{-1} has two eigenvalues inside the unit circle. This economy is sink-stable – from any initial condition it converges to the steady state. Since $\widehat{C}_0$ is not predetermined, we have an infinite number of solution paths that correspond to a different choice of initial consumption. Let e_{t+1} be a random variable that is uncorrelated with any variable at time t; we refer to e_t as extrinsic uncertainty. Rearranging the above equation we get

$$\left[\begin{array}{c} \widehat{K}_{t+1} \\ \widehat{C}_{t+1} \end{array}\right] = J^{-1} \left[\begin{array}{c} \widehat{K}_t \\ \widehat{C}_t \end{array}\right] + \left[\begin{array}{c} 0 \\ e_{t+1} \end{array}\right].$$

One can interpret e_t as 'animal spirits' or 'sunspots' or any bubble-like solution to the fundamental difference equation of the economy. The economy's path is then affected by these disturbances. If the economy is saddle-stable, these paths cannot occur because the approach to the steady state is unique; since the social planning solution never involves sunspot shocks, their presence is suboptimal in the Paretian sense. Note that the same code can be used here, as long as you redefine the state vector to include C_t.

In the increasing returns model, the labor demand curve can be upward-sloping, implying that increases in the capital stock lead to declines in hours. Since more capital means higher marginal products of labor, labor demand shifts upward. If labor demand is more upward-sloping than supply, total hours will fall and the equilibrium will be indeterminate.

Other distortions – income taxes, monopoly power, incomplete markets – can also lead to sunspot equilibria, as can interactions between monetary and fiscal policy (Leeper 1991). For higher-order systems, we get one dimension of indeterminacy – meaning a sunspot – for each of the excess stable eigenvalues; in three-dimensional systems, we might get a saddle with a plane of convergence (one sunspot) or a sink (two sunspots).

Note that we do not need to stop at linear solutions; we can build second-order terms as before. Wen and Wu (2011) show that, near the region where indeterminacy occurs, second-order terms may become very important; I checked third-order terms and they are large as well (but remember they get multiplied by $\frac{1}{k!}(x - x^*)^k$, so the product term might still be small).

Now we turn to the other option – too few stable eigenvalues. In this case, it is possible that the unstable steady state is surrounded by stable limit cycles (Beaudry, Galizia, and Portier 2016 argue that this representation is a good one for the US). Local solutions to this economy would be very misleading as a result, since one would typically discard this parameterization as explosive even though the global equilibria are not explosive and the transversality condition is satisfied.

Galizia (2019) shows standard local perturbation methods cannot solve this kind of economy. To understand what to do, we follow that paper. Consider an economic model

$$\Gamma(x_t, x_{t+1}) = 0$$

for some locally smooth function Γ. We suppose that there is a unique steady state satisfying

$$\Gamma(x^*, x^*) = 0,$$

for any x_t there is a locally unique solution for x_{t+1}, and $D_{x_{t+1}}\Gamma(x^*, x^*)$ is nonsingular. For convenience, let's define variables such that $x^* = 0$. As usual, partition the variables into predetermined variables y and jump variables z, with sizes n_y and n_z. Then the solution takes

the form of two functions,

$$z_t = \phi(y_t)$$
$$y_{t+1} = \pi(y_t).$$

That is, for an open set Θ that contains zero,

$$\Gamma(y_t, \phi(y_t), \pi(y_t), \pi(\phi(y_t))) = 0$$

and

$$\limsup_{t\to\infty} \{\|x_t\|\} < \infty.$$

Define

$$A = \left(D_{x_{t+1}}\Gamma(0,0)\right)^{-1} D_{x_t}\Gamma(0,0);$$

the linearized system is then

$$x_{t+1} = Ax_t.$$

Let Λ denote the spectrum of A, and for each $\lambda \in \Lambda$, let $e(\lambda)$ be the generalized eigenspace (the invariant manifold associated with λ), and

$$r(\lambda) \equiv \mathrm{Re}(e(\lambda)) \oplus \mathrm{Im}(e(\lambda)),$$

which we call the RGE (real generalized eigenspace) associated with λ. Note that $r(\lambda) = r\left(\overline{\lambda}\right)$, $r(\lambda)$ is an A-invariant (if $x \in M$ then $Ax \in M$) linear subspace, and the dimension of $r(\lambda)$ is either the same as $e(\lambda)$ (if λ is real) or twice the dimension of $e(\lambda)$ (if λ is complex). Finally, define $\mathcal{W}$ as the set of A-invariant linear subspaces of $\mathcal{R}^n$ (excluding the origin) and $\mathcal{S}$ as the set of analytic solutions (π, ϕ), regardless of whether they satisfy TVC. Again for convenience, we will also assume that A has no repeated eigenvalues.

For a pair (π, ϕ), we must have

$$D_{y_t}\Gamma(0,0) + D_{z_t}\Gamma(0,0)\,D\phi(0) + D_{y_{t+1}}\Gamma(0,0)\,D\pi(0) + D_{z_{t+1}}\Gamma(0,0)\,D\phi(0)\,D\pi(0) = 0.$$

There are typically multiple solutions to this equation associated with different Schur orderings; given each solution, we can find higher-order terms uniquely. The usual procedure would examine the solutions, discard any that are locally divergent (since the linear system would therefore not

satisfy TVC), and obtain higher-order solutions for only those locally stable solutions. However, that would be misleading if there are stable limit cycles lurking far from the steady state. Rather than throw them away ex ante and potentially miss valid stable solutions, Galizia (2019) suggests that we simulate the dynamic behavior of each higher-order solution, independent of whether the first-order solution is stable, and only discard those solutions that globally violate TVC. Fortunately, he also provides us a rule for discarding some of the potential solutions, which is important for a large system; any Schur ordering with a real unstable eigenvalue cannot have a stable limit cycle and therefore cannot satisfy TVC.

I want to point out that we are establishing the stability of the approximation, not the model; as noted with pruning, low-order nonlinear approximations may display behavior qualitatively inconsistent with the true solution.

8.10 What to Do If Steady States Are Not Solvable

As mentioned above, in some models the steady state is not "solvable" in the sense that we need the entire policy function in order to determine it, instead of being able to simply find the fixed point. Here I give two examples (which are actually closely related) and show how to use parameters as states to deal with them. The key is that the steady state is not solvable for all parameter values but is for special cases.

There is empirical evidence that households discount the future versus the present, not merely in a geometric way but by adding an extra factor to the entire future. That is, there is a term θ which applies to the entire future relative to the present, and geometric discounting applies between periods; see Laibson (1994) for the initial investigation of this model. The resulting sequence of discount factors is

$$\left\{1, \theta\beta, \theta\beta^2, \theta\beta^3, ...\right\}.$$

If we think of lifetime utility, we get

$$U_0 + \theta\beta U_1 + \theta\beta^2 U_2 + ...$$

at time 0 and

$$U_1 + \theta\beta U_2 + ...$$

at time 1. The result is that the household changes their discount factor between future periods, depending on when they are asked about it. From the perspective of period 0, the marginal rate of substitution between 1 and 2 is

$$\frac{DU_1}{\beta DU_2}$$

but in period 1 it becomes

$$\frac{DU_1}{\theta\beta DU_2}.$$

Only when $\theta = 1$ does the agent have time-invariant marginal rates of substitution. In other words, quasi-geometric discounters will change their minds about the future optimality of currently optimal plans.

What this means for decision problems is that behavior looks like there is a game being played between the current self and future selves; these future selves (who will change their mind relative to what the current self wants) are not under the control of the current self, so any plans are recalculated period-by-period – therefore the current self must accurately anticipate what future selves will do. We will look for a subgame perfect Markov differentiable equilibrium to this game. The decision problem for the current self is

$$v_0(k) = \max_{k'} \left\{ u\left(F(k) - k'\right) + \theta\beta v\left(k'\right) \right\}$$

subject to the subgame perfection requirement that the value function of the future selves satisfy the recursion

$$v(k) = u\left(F(k) - g(k)\right) + \beta v\left(g(k)\right)$$

for the policy function $g(k)$ that will be used by the future selves. Note that nothing under the control of the current self appears in this equation, and that the current self problem is not a functional equation because v_0 and v are not the same functions. Krusell and Smith (2003) and Judd (200?) both show that the objective function here is very poorly-behaved.

The first-order condition is

$$Du\left(F(k) - k'\right) = \theta\beta Dv\left(k'\right);$$

for subgame perfection to hold we will have $k' = g(k)$. Since the future selves do not agree with the current self, the standard envelope theorem does not apply. But we still need to rid ourselves

of the term $Dv(k')$, which we do as follows. Differentiate the equilibrium condition to obtain

$$Dv(k) = Du(F(k) - g(k))(DF(k) - Dg(k)) + \beta Dv(g(k))Dg(k)$$

and solve this for $Dv(g(k))$:

$$Dv(g(k)) = \frac{Dv(k) - Du(F(k) - g(k))(DF(k) - Dg(k))}{\beta Dg(k)}.$$

We then insert this into the first-order condition and solve for $Dv(k)$:

$$\begin{aligned} Du(F(k) - g(k)) &= \theta \frac{Dv(k) - Du(F(k) - g(k))(DF(k) - Dg(k))}{Dg(k)} \\ Dv(k) &= \frac{1}{\theta} Du(F(k) - g(k))Dg(k) + Du(F(k) - g(k))(DF(k) - Dg(k)). \end{aligned}$$

Advancing this one period and inserting it back into the first-order condition we get

$$Du\left(F(k) - k'\right) = \beta Du\left(F\left(k'\right) - g\left(k'\right)\right)Dg\left(k'\right) + \theta\beta Du\left(F\left(k'\right) - g\left(k'\right)\right)\left(DF\left(k'\right) - Dg\left(k'\right)\right).$$

Finally imposing $k' = g(k)$ we obtain the generalized Euler equation

$$Du(F(k) - g(k)) = \beta Du(F(g(k)) - g(g(k)))(\theta DF(g(k)) + (1-\theta)Dg(g(k))).$$

Notice that the derivative of the policy function appears in the Euler equation, reflecting the ability of the current self to influence the future self by saving more. Since the two selves do not agree on the value of this saving, that term does not disappear – that is, the usual envelope theorem condition fails. The second self, which discounts between periods 2 and 3 more than the first self does, will not save enough and therefore is manipulated by the first self through the Dg term. Note: for some parameterizations, this game has more than one continuously-differentiable Markov solution (for example, if $F(k) = Ak$, $u(c) = \frac{c^{1-\sigma}}{1-\sigma}$, and σ is sufficiently high).

Solution methods to solve the functional equations which define an equilibrium of this game,

$$\begin{aligned} Du(F(k) - g(k)) &= \beta Du(F(g(k)) - g(g(k)))(\theta DF(g(k)) + (1-\theta)Dg(g(k))) \\ v(k) &= u(F(k) - g(k)) + \beta v(g(k)) \\ g(k) &= \underset{k'}{\operatorname{argmax}} \left\{u\left(F(k) - k'\right) + \theta\beta v\left(k'\right)\right\}, \end{aligned}$$

are not readily apparent. The operator that defines a recursion on these equations – solve the third and use it in the second to update a guess for the value function – is not a contraction

and therefore does not converge to a fixed point in general (it may if the scheme is numerically inaccurate and θ is sufficiently close to 1). Furthermore, many numerical methods designed to solve the Euler equation iteratively also do not work; it appears that they run afoul of the discontinuous local solutions discussed in Krusell and Smith (2003), which are step functions and not global equilibria; see Cao and Werning (2019). Perturbation methods work perfectly fine, once we deal with the determination of the steady state.

In the steady state we have

$$1 = \beta \left(\theta DF \left(k^* \right) + \left(1 - \theta \right) Dg \left(k^* \right) \right),$$

which cannot be solved for k^* unless we know $g(k)$, which we don't. But this expression can be solved if $\theta = 1$, since that last term drops out, so we can treat θ as a state and use 1 as the "steady state" value, similar to the approach of dealing with volatility. This procedure produces a policy function that depends explicitly on θ and the capital stock of the form

$$\begin{aligned} g\left(k,\theta\right) \approx{}& g\left(k^*,1\right) + D_1 g\left(k^*,1\right)\left(k-k^*\right) + D_2 g\left(k^*,1\right)\left(\theta-1\right) + \frac{1}{2} D_{11} g\left(k^*,1\right)\left(k-k^*\right)^2 + \\ & D_{12} g\left(k^*,1\right)\left(k-k^*\right)\left(\theta-1\right) + \frac{1}{2} D_{22} g\left(k^*,1\right)\left(\theta-1\right)^2 + ...; \end{aligned}$$

we can check the Euler equation errors to see how accurate our solution is away from $\theta = 1$.

Young (2006) extended the basic model to multiple-period deviations from geometric discounting. The general case has a discount factor sequence

$$\left\{ \beta\theta_1, \beta^2\theta_1\theta_2, \cdots, \beta^n \prod_{i=1}^{n} \theta_i, \beta^{n+1} \prod_{i=1}^{n} \theta_i, \cdots \right\}$$

and leads to the generalized Euler equation

$$\begin{aligned} Du\left(c_t\right) = {}& \beta\theta_1 Du\left(c_{t+1}\right)\left(Df\left(k_{t+1}\right) + 1 - \delta\right) - \\ & \sum_{j=2}^{n} \beta^j \left(\prod_{i=1}^{j} \theta_i \right) \left(\frac{1}{\theta_j} - 1 \right) Du\left(c_{t+j}\right) Df\left(k_{t+j}\right) Dg^{(j-1)}\left(k_{t+1}\right) + \\ & \sum_{j=2}^{n} \beta^j \left(\prod_{i=1}^{j} \theta_i \right) \left(\frac{1}{\theta_j} - 1 \right) Du\left(c_{t+j}\right) Dg^{(j)}\left(k_{t+1}\right) \end{aligned}$$

where

$$\begin{aligned} Dg^{(2)}\left(k_{t+1}\right) &= Dg\left(g\left(k_{t+1}\right)\right) Dg\left(k_{t+1}\right) \\ Dg^{(3)}\left(k_{t+1}\right) &= Dg\left(g\left(g\left(k_{t+1}\right)\right)\right) Dg\left(g\left(k_{t+1}\right)\right) Dg\left(k_{t+1}\right) \end{aligned}$$

and so on. Now the perturbation solution is taken at the point $\theta_i = 1\ \forall i \in \{1, ..., n\}$. The interpretation of this nasty equation is slightly more complicated – it now involves the incentive of the current self to influence future selves directly (the third term) and indirectly (by manipulating the intervening selves, which is the second term).

Another example is the so-called temptation model, in which agents are tempted to discount the future excessively (just an example); Krusell, Kuruşçu, and Smith (2010) show that this model reduces to the quasi-geometric one as a special case. The Bellman equation is

$$v(k) = (1+\gamma) \max_{k'} \left\{ u\left(f(k) + (1-\delta)k - k'\right) + \beta \frac{1+\theta\gamma}{1+\gamma} v\left(k'\right) \right\} -$$
$$\gamma \max_{\widetilde{k}'} \left\{ u\left(f(k) + (1-\delta)k - \widetilde{k}'\right) + \beta\theta v\left(\widetilde{k}'\right) \right\};$$

the last term reflects the cost of self-control. The first-order conditions are

$$Du\left(f(k) + (1-\delta)k - g(k)\right) = \beta \frac{1+\theta\gamma}{1+\gamma} Dv\left(g(k)\right)$$
$$Du\left(f(k) + (1-\delta)k - h(k)\right) = \beta\theta Dv\left(h(k)\right).$$

The envelope conditions are

$$Dv(k) = (1+\gamma) Du\left(f(k) + (1-\delta)k - g(k)\right)\left(Df(k) + 1 - \delta - Dg(k)\right) -$$
$$\gamma Du\left(f(k) + (1-\delta)k - h(k)\right)\left(Df(k) + 1 - \delta - Dh(k)\right).$$
$$Dv(g(k)) = (1+\gamma) Du\left(f(g(k)) + (1-\delta)g(k) - g(g(k))\right)\left(Df(g(k)) + 1 - \delta - Dg(g(k))\right) -$$
$$\gamma Du\left(f(g(k)) + (1-\delta)g(k) - h(g(k))\right)\left(Df(g(k)) + 1 - \delta - Dh(g(k))\right)$$
$$Dv(h(k)) = (1+\gamma) Du\left(f(h(k)) + (1-\delta)h(k) - g(h(k))\right)\left(Df(h(k)) + 1 - \delta - Dg(h(k))\right) -$$
$$\gamma Du\left(f(h(k)) - h(h(k))\right)\left(Df(h(k)) + 1 - \delta - Dh(h(k))\right).$$

The Euler equations are therefore

$$Du\left(f(k) + (1-\delta)k - g(k)\right) = \beta \frac{1+\theta\gamma}{1+\gamma} \left[\begin{array}{r} (1+\gamma) Du\left(f(g(k)) + (1-\delta)g(k) - g(g(k))\right)\left(Df(g(k)) + 1 - \delta - \right. \\ \gamma Du\left(f(g(k)) + (1-\delta)g(k) - h(g(k))\right)\left(Df(g(k)) + 1 - \delta - D\right. \end{array} \right.$$
$$Du\left(f(k) + (1-\delta)k - h(k)\right) = \beta\theta \left[\begin{array}{r} (1+\gamma) Du\left(f(h(k)) + (1-\delta)h(k) - g(h(k))\right)\left(Df(h(k)) + 1 - \delta - Dg(h\right. \\ \gamma Du\left(f(h(k)) - h(h(k))\right)\left(Df(h(k)) + 1 - \delta - Dh(h(k))\right) \end{array} \right.$$

In a steady state we have

$$k^* = g(k^*).$$

There also exists a fixed point for h:

$$\widetilde{k}^* = h\left(\widetilde{k}^*\right).$$

It will generally not be the case, however, that

$$k^* = \widetilde{k}^*,$$

so that we cannot easily evaluate terms like

$$g\left(\widetilde{k}^*\right)$$
$$h\left(k^*\right)$$

let alone

$$h\left(g\left(\widetilde{k}^*\right)\right)$$
$$g\left(h\left(k^*\right)\right).$$

Fortunately, we know that this problem reduces to the standard growth model if $\theta = 1$ for any value of γ, so we could use a perturbation approach around the point $(k, \theta) = (k^*, 1)$ similar to the method for solving the quasi-geometric discounting game. It also reduces to the growth model if $\gamma = 0$, so we might find it better to build our approximation there, or even to use both, depending on what improves accuracy the most.

If we let $\gamma \to \infty$, then the decision rules for this economy (and the value functions) converge to those for the Laibson version. To see how this convergence works, consider the Euler equations again:

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-g\left(k\right)\right)=\beta\frac{1+\theta\gamma}{1+\gamma}\left[\begin{array}{r}\left(1+\gamma\right)Du\left(f\left(g\left(k\right)\right)+\left(1-\delta\right)g\left(k\right)-g\left(g\left(k\right)\right)\right)\left(Df\left(g\left(k\right)\right)+1-\delta-\right.\\ \gamma Du\left(f\left(g\left(k\right)\right)+\left(1-\delta\right)g\left(k\right)-h\left(g\left(k\right)\right)\right)\left(Df\left(g\left(k\right)\right)+1-\delta-D\right.\end{array}\right.$$

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-h\left(k\right)\right)=\beta\theta\left[\begin{array}{r}\left(1+\gamma\right)Du\left(f\left(h\left(k\right)\right)+\left(1-\delta\right)h\left(k\right)-g\left(h\left(k\right)\right)\right)\left(Df\left(h\left(k\right)\right)+1-\delta-Dg\left(h\right.\right.\\ \gamma Du\left(f\left(h\left(k\right)\right)+\left(1-\delta\right)h\left(k\right)-h\left(h\left(k\right)\right)\right)\left(Df\left(h\left(k\right)\right)+1-\delta-Dh\left(h\right.\right.\end{array}\right.$$

Suppose there is a terminal point T, at which

$$g_T\left(k\right)=h_T\left(k\right)=0.$$

At date $T-1$, we have the Euler equations

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-g_{T-1}\left(k\right)\right)=\beta\frac{1+\theta\gamma}{1+\gamma}\left[Du\left(f\left(g_{T-1}\left(k\right)\right)+\left(1-\delta\right)g_{T-1}\left(k\right)\right)\left(Df\left(g_{T-1}\left(k\right)\right)+1-\delta\right)\right]$$

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-h_{T-1}\left(k\right)\right)=\beta\theta\left[Du\left(f\left(h_{T-1}\left(k\right)\right)+\left(1-\delta\right)h_{T-1}\left(k\right)\right)\left(Df\left(h_{T-1}\left(k\right)\right)+1-\delta\right)\right].$$

As $\gamma\to\infty$, $\frac{1+\theta\gamma}{1+\gamma}\to\theta$, so the equations require $g_{T-1}=h_{T-1}$. Iterating backward until g and h converge yields the desired result.

We could assume many other forms of temptation utility, such as a temptation to behave less risk averse, overly optimistic, or perhaps many others (DeJong and Ripoll 2008).

Competitive equilibria with quasi-geometric discounting are not Pareto efficient in general, and generate additional complications. Consider the household problem

$$V\left(k,K\right)=\max_{k'}\left\{u\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-k'\right)+\theta\beta v\left(k',G\left(K\right)\right)\right\}$$

where

$$v\left(k,K\right)=u\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)+\beta v\left(g\left(k,K\right),G\left(K\right)\right).$$

The first-order condition is

$$Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-k'\right)=\theta\beta D_1v\left(k',G\left(K\right)\right).$$

The envelope condition is

$$D_1v\left(k,K\right)=Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)\left(r\left(K\right)+1-\delta-D_1g\left(k,K\right)\right)+\beta D_1v\left(g\left(k,K\right),G\left(K\right)\right)D_1$$

Rearranging we get

$$D_1v\left(g\left(k,K\right),G\left(K\right)\right)=\frac{D_1v\left(k,K\right)-Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)\left(r\left(K\right)+1-\delta-D_1g\left(k,K\right)\right)}{\beta D_1g\left(k,K\right)}.$$

Then

$$Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)=\theta\frac{D_1v\left(k,K\right)-Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)\left(r\left(K\right)+1\right.}{D_1g\left(k,K\right)}$$

which implies

$$D_1v\left(k,K\right)=\left(\frac{1}{\theta}-1\right)Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)D_1g\left(k,K\right)+Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\right.$$

so that the Euler equation is

$$
\begin{aligned}
&Du\left(\left(r\left(K\right)+1-\delta\right)k+w\left(K\right)-g\left(k,K\right)\right)\\
&=\theta\beta\left(\begin{array}{c}\left(\frac{1}{\theta}-1\right)Du\left(\left(r\left(G\left(K\right)\right)+1-\delta\right)g\left(k,K\right)+w\left(G\left(K\right)\right)-g\left(g\left(k,K\right),G\left(K\right)\right)\right)D_1g\left(g\left(k,K\right),G\left(K\right)\right)+\\ Du\left(\left(r\left(G\left(K\right)\right)+1-\delta\right)g\left(k,K\right)+w\left(G\left(K\right)\right)-g\left(g\left(k,K\right),G\left(K\right)\right)\right)\left(r\left(G\left(K\right)\right)+1-\delta\right)\end{array}\right).
\end{aligned}
$$

Now impose equilibrium $k=K$, $g\left(K,K\right)=G\left(K\right)$, $r\left(K\right)=D_1F\left(K\right)$ and $w\left(K\right)=F\left(K\right)-D_1F\left(K\right)K$:

$$
\begin{aligned}
&Du\left(F\left(K\right)+\left(1-\delta\right)K-G\left(K\right)\right)\\
&=\theta\beta\left(\begin{array}{c}\left(\frac{1}{\theta}-1\right)Du\left(F\left(G\left(K\right)\right)+\left(1-\delta\right)G\left(K\right)-G\left(G\left(K\right)\right)\right)D_1g\left(G\left(K\right),G\left(K\right)\right)+\\ Du\left(F\left(G\left(K\right)\right)+\left(1-\delta\right)G\left(K\right)-G\left(G\left(K\right)\right)\right)\left(D_1F\left(G\left(K\right)\right)+1-\delta\right)\end{array}\right).
\end{aligned}
$$

Note the difference from the central planner solution above: since in general

$$D_1g\left(G\left(K\right),G\left(K\right)\right)\neq DG\left(K\right),$$

we will get different answers. To solve for the equilibrium, we need to explicitly retain the individual state and solve both Euler equations at once for the functions

$$k'=g\left(k,K\right)$$

and

$$K'=G\left(K\right),$$

where by construction it will hold that

$$G\left(K\right)=g\left(K,K\right).$$

8.11 OccBin and DynareOBC

The OccBin algorithm (Guerrieri and Iacoviello 2010) is a method for imposing occasionally-binding constraints in perturbation solutions. The idea is to ignore precautionary effects and work with a piecewise-linear solution that is constructed in a particular way.

First, linearize the model around the deterministic steady state for both the binding and nonbinding regimes. Regime 1 is the regime that applies at the steady state, Regime 2 is the

other regime. Regime 1 is locally described by

$$AE_t\left[X_{t+1}\right] + BX_t + CX_{t-1} + F\varepsilon_t = 0$$
$$g\left(E_t\left[X_{t+1}\right], X_t, X_{t-1}\right) < 0$$

and Regime 2 by

$$A^*E_t\left[X_{t+1}\right] + B^*X_t + C^*X_{t-1} + D^* + F^*\varepsilon_t = 0;$$

the constant term D^* appears because this equation has not been linearized around its steady state.

The solution is represented by

$$X_1 = P_1X_0 + R_1 + Q_1\varepsilon_1$$
$$X_t = P_tX_{t-1} + R_t;$$

that is, OccBin assumes there are no shocks beyond period 1, then guesses the length of time each regime is in place and verifies ex post using the function g and the sample implied by the guesses.

Specifically, suppose Regime 2 applies from period 1 to T. Then the linear approximation for $t \geq T$ is

$$X_t = PX_{t-1},$$

so that

$$X_T = PX_{T-1},$$

which further implies

$$X_{T-1} = -\left(A^*P + B^*\right)^{-1}\left(C^*X_{T-2} + D^*\right).$$

Thus we get

$$P_{T-1} = -\left(A^*P + B^*\right)^{-1}C^*$$
$$R_{T-1} = -\left(A^*P + B^*\right)^{-1}D^*.$$

We can then solve for X_{T-2} given X_{T-3}, choosing the appropriate matrices based on our guess for which regime is in place. In period 1 we have either

$$Q_1 = -\left(A^*P + B^*\right)^{-1}F$$

or

$$Q_1 = -\left(A^* P + B^*\right)^{-1} F^*.$$

This solution is used to check whether the guess was correct, by calculating the value of g in each period. If the guess is wrong, we update and repeat.

For a stochastic simulation, we use this result to get X_1 given X_0, and then repeat the process to get X_2 given X_1. The result is a simulation that implies, at each point in time, agents ignore the possibility of future shocks when calculating what decisions to make today. While OccBin is better than another smooth method – the barrier function we see next – it still implies rather large errors; for large models, however, it is not clear there is another method that is both (i) accurate and (ii) fast.

8.11.1 Holden and Paetz

An alternative to OccBin is based on Holden and Paetz (2013), in which we use "news" shocks to enforce the constraints. Suppose the model has one nonlinear equation

$$x_{1,t} = \max\left\{0, \mu_1 + \phi_{-1}^T x_{t-1} + \phi_0^T x_t + \phi_1^T E_t x_{t+1} - \left(\phi_{-1}^T + \phi_0^T + \phi_1^T\right)\mu\right\}$$

with the rest linear

$$x_{i,t} = \mu_i + \psi_{-1}^T x_{t-1} + \psi_0^T x_t + \psi_1^T E_t x_{t+1} - \left(\psi_{-1}^T + \psi_0^T + \psi_1^T\right)\mu.$$

Note that if we need an equation of the form

$$x_t = \max\left\{\widehat{x}_t, y_t\right\}$$

with $x_t > y_t$ in the steady state, we rewrite as

$$x_t = y_t + \max\left\{0, \widetilde{x}_t - y_t\right\}.$$

Consider a shock that drives $x_{1,t}$ to zero as being a combination of the true shock and a "news" shock that $x_{1,t}$ will be higher than expected for some time. Let T^* denote the length of time the constraint is expected to bind. IRF vectors are of length $T \geq T^*$. Now replace the bounded equation with

$$x_{1,t} = \mu_1 + \phi_{-1}^T x_{t-1} + \phi_0^T x_t + \phi_1^T E_t x_{t+1} - \left(\phi_{-1}^T + \phi_0^T + \phi_1^T\right)\mu + \sum_{s=0}^{T^*-1} \epsilon_{s,t-s}^{SP};$$

$\epsilon_{s,t}^{SP}$ is known at time t but does not affect anything directly until time $t+s$. We then look for $\left\{\epsilon_{s,t}^{SP}\right\}$ such that

$$x_{1,t} = 0$$

for all $t \leq T^*$ and

$$x_{1,t} > 0$$

for all $t > T^*$, where $\epsilon_{s,t} = 0 \; \forall t > 0$. Suppose that

$$\epsilon_t = 1$$
$$\epsilon_{s,0}^{SP} = \alpha_s$$

and define $m_{j,s}$ to be the IRF of variable j to shock $\epsilon_{s,t}^{SP}$ and M_j to be a horizontally-stacked matrix of these IRFs. Let M^* be the upper $T^* \times T^*$ submatrix of M_1. Then the complementary slackness condition is

$$\alpha^T \left(\mu^* + v^* + M^*\alpha\right) = 0$$

where ν^* is the first T^* elements of the IRF to ϵ_t (not $\epsilon_{s,t}^{SP}$) and $\mu^* = \mu_1 1_{T^*}$. We can then solve the quadratic program

$$\alpha^* = \arg \min_{\substack{\alpha \geq 0 \\ \mu^* + v^* + M^*\alpha \geq 0}} \left\{ \alpha^T \left(\mu^* + v^*\right) + \frac{1}{2}\alpha^T \left(M^* + \left(M^*\right)^T\right)\alpha \right\}.$$

We will get a unique solution if $M^* + \left(M^*\right)^T$ is positive definite, which may or may not be true. An admissible solution requires

$$\alpha^T \left(\mu^* + v^*\right) + \frac{1}{2}\alpha^T \left(M^* + \left(M^*\right)^T\right)\alpha = 0,$$

which in turn implies the complementary slackness condition holds. In effect, we find the "shadow prices" such that the constraint is exactly binding and convert them into anticipated changes in future variables associated with leaving the constraint. Note that there need not be an admissible solution.

An equivalent method for finding these shocks is to cast it as a Linear Complementarity Problem (LCP): find $y \geq 0$ such that

$$y \cdot (q + My) = 0$$
$$q + My \geq 0.$$

LCPs have a unique solution for all q if M is a **P-matrix** (all principal submatrices have positive derivatives), otherwise it may have multiple solutions for some q. $\forall q >> 0$, the LCP has a unique solution iff M is **semi-monotone** ($\forall x \geq 0$ and $x \neq 0$, there exists k such that the kth entries satisfy $x_k > 0$ and $(Mx)_k \geq 0$). $\forall q \geq 0$, the LCP has a unique solution iff M is **strictly semi-monotone** ($\forall x \geq 0$ and $x \neq 0$, there exists k such that the kth entries satisfy $x_k > 0$ and $(Mx)_k > 0$). With multiple solutions, we can have a weird situation where news causes the constraint to be violated if it exists, but the condition would not be violated if the constraint did not exist.

The LCP is **feasible** if there exists $y \geq 0$ such that $q + My \geq 0$. LCPs are feasible iff M is an **S-matrix** ($\exists x \geq 0$ such that $Mx > 0$). A matrix is **row-sufficient** if

$$x_i \left(M^T x\right)_i \leq 0 \Rightarrow x_i \left(M^T x\right)_i = 0.$$

The LCP is **solvable** if it is feasible and M is row-sufficient. A matrix is **co-positive** if $x^T M x \geq 0$ $\forall x \geq 0$. The LCP is solvable if it is feasible and M is co-positive.

Collecting: an LCP (q, M) is solvable $\forall q$ if any of the following hold: (i) M is an S-matrix and either M is row-sufficient or co-positive; (ii) M is co-positive with no zero principal minors; (iii) M is a P-matrix, a strictly co-positive matrix, or a strictly semi-monotone matrix.

If M has only nonnegative entries, then the LCP (q, M) is solvable $\forall q$ iff M has a strictly positive diagonal.

Define

$$\varsigma \equiv \sup_{\substack{y \in [0,1]^{\mathcal{N}_+} \\ \exists T \in \mathcal{N}: \forall t > T, y_t = 0}} \inf_{t \in \mathcal{N}_+} \left\{ M_{t,1:T} y_t \right\}.$$

M is an S-matrix for large enough T iff $\varsigma > 0$.

In general, LCP is NP-complete (it must be or it could be used to solve the QP problem, which we know is NP-complete). Suppose $\widetilde{\omega} > 0$. Set up a mixed-integer program: find α, $\widehat{y}$, $z \in \{0,1\}^T$ to maximize α subject to

$$\begin{aligned} &\alpha \geq 0 \\ &0 \leq \widehat{y} \leq z \\ &0 \leq \alpha q + M\widehat{y} \leq \widetilde{\omega}\left(1 - z\right). \end{aligned}$$

If $(\alpha, \widetilde{y}, z)$ solve the MI problem and $\alpha = 0$, the LCP has no solution; if $\alpha \neq 0$, then the LCP is solved by $y = \frac{\widetilde{y}}{\alpha}$. As $\widetilde{\omega} \to 0$, we get the solution to the LCP that minimizes $\|q + My\|_\infty$; as $\widetilde{\omega} \to \infty$, we get the solution to the LCP that minimizes $\|y\|_\infty$.

8.12 Barrier Functions – A Smooth Way to get Occasionally Binding Constraints

Models with occasionally binding constraints are not analytic: the unknown functions are not globally the limit of a power series expansion at every point (they typically are not differentiable at the points where the constraint begins to bind, and there is little to no information about the constrained region that can be gained from expansions in the unconstrained region). One approach to dealing with this problem is to replace the actual constraint with a "smooth constraint" that tries to mimic the effects. Specifically, we can define a bnrrier function for a set D, just as we discussed in the interior-point method of optimization.

Definition 88 *A function $f : X \to \mathcal{R}$ is a **barrier function for the set** D if f is close to zero for x in the interior of D and converges to infinity as x converges to a point in ∂D.*

One example of a barrier function is the log-barrier:

$$f(x) = -\log(b - x)$$

for $x < b$; clearly as x goes to b this function diverges and the function is not well defined for $x > b$. Another version does not require the function to go to infinity, but only to increase rapidly (like penalty functions); one example is

$$f(x) = \frac{\eta_1}{\eta_0} \exp(-\eta_0 x) - \eta_2 a.$$

Since these functions are smooth, we can differentiate them; they adjust the cost of moving toward the constraint (gets bigger) or away from the constraint (gets smaller), and therefore if severe enough can enforce a soft version of a constraint. Unfortunately, for problems where the point of approximation is far from the point where the constraint binds, the use of barrier functions often badly distorts the answer. To see how much, consider the growth model with an irreversibility constraint on investment:

$$v(k) = \max_{k', c} \left\{ \log(c) + \beta v\left(k'\right) : c + k' \leq k^\alpha + (1 - \delta) k, k' \geq (1 - \delta) k \right\}.$$

We choose a version of the log-barrier function used in Kim, Kim, and Kollman (2010), yielding

$$v(k) = \max_{k'} \left\{ \log\left(k^{\alpha} + (1-\delta)k - k'\right) + \beta v\left(k'\right) - \mu \log\left(\frac{k' - (1-\delta)k}{\delta k^*}\right) \right\};$$

note that at the deterministic steady state k^* the penalty is zero. The first-order condition is

$$-\frac{1}{k^{\alpha} + (1-\delta)k - k'} + \beta Dv\left(k'\right) - \frac{\mu}{k' - (1-\delta)k} = 0,$$

the envelope condition is

$$Dv(k) = \frac{\alpha k^{\alpha-1} + 1 - \delta}{k^{\alpha} + (1-\delta)k - k'} + \frac{\mu(1-\delta)}{k' - (1-\delta)k},$$

which can be combined to obtain the Euler equation

$$-\frac{1}{k^{\alpha} + (1-\delta)k - k'} + \beta\left(\frac{\alpha(k')^{\alpha-1} + 1 - \delta}{(k')^{\alpha} + (1-\delta)k' - k''} + \frac{\mu(1-\delta)}{k'' - (1-\delta)k'}\right) - \frac{\mu}{k' - (1-\delta)k} = 0.$$

If $\mu = 0$, we get the unconstrained case; if $\mu \neq 0$, however, this solution is distorted by the term

$$\mu\left(\frac{\beta(1-\delta)}{k'' - (1-\delta)k'} - \frac{1}{k' - (1-\delta)k}\right).$$

A first-order expansion around k^* of this term yields

$$\frac{\mu}{(\delta k^*)^2}\left(-\beta(1-\delta)\left(k'' - k^*\right) + \left(\beta(1-\delta)^2 + 1\right)\left(k' - k^*\right) - (1-\delta)\left(k - k^*\right)\right),$$

which will be the size of the distortion introduced into the linear decision rules.

De Wind (2018) showed that barrier functions distort some aspects of the solution rather severely, in particular the tails of the invariant distribution.

8.13 Change of Variables

A standard Taylor expansion of a function is

$$f(x) = f(a) + Df(a)(x-a) + \frac{1}{2}D^2 f(a)(x-a)^2 + \frac{1}{6}D^3 f(a)(x-a)^3.$$

A change of variables can be helpful in that it will build in nonlinearity without adding terms. Let

$$y = Y(x)$$

$$x = X(y)$$

$$g(y) = f(X(y)).$$

That is, we will now build an expansion of g (a nonlinear transformation of f) in terms of y (a nonlinear transformation of x). In some ways this method is similar to the gauge expansion, in that we construct an polynomial expansion in a different variable.

We use the point

$$y = b = Y(a).$$

The Taylor expansion is

$$\begin{aligned} g(y) = f(X(y)) = f(X(b)) + Df(X(b))\,DX(b)(y-b) + \\ \frac{1}{2}\left(D^2 f(X(b))(DX(b))^2 + Df(X(b))\,D^2X(b)\right)(y-b)^2 + \\ \frac{1}{6}\left(D^3 f(X(b))(DX(b))^2 + 3D^2 f(X(b))\,DX(b)\,D^2X(b) + Df(X(b))\,D^3X(b)\right)(y-b)^3. \end{aligned}$$

Using the facts that

$$f(x) = g(Y(x))$$
$$b = Y(a)$$

we therefore have that our expansion of f is given by

$$\begin{aligned} f(x) = g(Y(x)) = f(a) + Df(a)\,DX(b)(Y(x)-b) + \\ \frac{1}{2}\left((DX(b))^2 D^2 f(a) + Df(a)\,D^2X(b)\right)(Y(x)-b)^2 + \\ \frac{1}{6}\left(DX(b)\,D^2 f(a)\,D^2X(b) + 2(DX(b))^3 D^3 f(a) + Df(a)\,D^3X(b)\right)(Y(x)-b)^3. \end{aligned}$$

We can also do an expansion in a nonlinear transformation of both the domain and the range. That is, we look for h such that

$$h(f(x)) = g(Y(x))$$
$$h(f(X(y))) = g(y)$$

which leads to

$$h(f(x)) = h(f(a)) + Dg(b)(Y(x)-b) + \frac{1}{2}D^2 g(b)(Y(x)-b)^2 + \frac{1}{6}D^3 g(b)(Y(x)-b)^3.$$

All of this looks to be a giant pain, but it turns out that there are simple relations we can use between the standard and nonlinear expansions. For example, a log-log approximation would

give us

$$
\begin{aligned}
Dg(b) &= Dh(f(X(b)))\,Df(X(b))\,DX(b)\\
&= \frac{1}{f(x_0)}Df(x_0)\,x_0\\
&= \eta_1\\
D^2g(b) &= D^2h(f(X(b)))\,(Df(X(b))\,DX(b))^2 + Dh(f(X(b)))\,D^2f(X(b))\,(DX(b))^2 +\\
&\quad Dh(f(X(b)))\,Df(X(b))\,D^2X(b)\\
&= -\frac{(Df(x_0)\,x_0)^2}{(f(x_0))^2} + \frac{D^2f(x_0)\,x_0^2}{f(x_0)} + \frac{Df(x_0)\,x_0}{f(x_0)}\\
&= \frac{Df(x_0)\,x_0}{f(x_0)}\left(1 - \frac{Df(x_0)\,x_0}{f(x_0)}\right) + \frac{D^2f(x_0)\,x_0^2}{f(x_0)}\\
&= \eta_1(1-\eta_1) + \eta_2\\
D^3g(b) &= D^3h(f(X(b)))\,(Df(X(b))\,DX(b))^3 + 2D^2h(f(X(b)))\,(DX(b))^3\,Df(X(b))\,D^2f(X(b)) +\\
&\quad 2D^2h(f(X(b)))\,(Df(X(b)))^2\,DX(b)\,D^2X(b) +\\
&\quad D^2h(f(X(b)))\,D^2f(X(b))\,(DX(b))^3\,Df(X(b)) + Dh(f(X(b)))\,D^3f(X(b))\,(DX(b))^3 +\\
&\quad 2Dh(f(X(b)))\,D^2f(X(b))\,DX(b)\,D^2X(b) +\\
&\quad D^2h(f(X(b)))\,(Df(X(b)))^2\,DX(b)\,D^2X(b) + Dh(f(X(b)))\,D^2f(X(b))\,DX(b)\,D^2X(b) +\\
&\quad Dh(f(X(b)))\,Df(X(b))\,D^3X(b)\\
&= \frac{2\,(Df(x_0))^3\,x_0^3}{f(x_0)^3} - 2\frac{Df(x_0)\,D^2f(x_0)\,x_0^3}{f(x_0)^2} - 2\frac{Df(x_0)^2\,x_0^2}{f(x_0)^2} - \frac{Df(x_0)\,D^2f(x_0)\,x_0^3}{f(x_0)^2} + \frac{D^3f(x_0)\,x_0^3}{f(x_0)} +\\
&\quad 2\frac{D^2f(x_0)\,x_0^2}{f(x_0)} - \frac{(Df(x_0))^2\,x_0^2}{f(x_0)^2} + \frac{D^2f(x_0)\,x_0^2}{f(x_0)} + \frac{Df(x_0)\,x_0}{f(x_0)}\\
&= \left(\frac{D^3f(x_0)\,x_0^3}{f(x_0)} + 3\frac{D^2f(x_0)\,x_0^2}{f(x_0)} + \frac{Df(x_0)\,x_0}{f(x_0)}\right) -\\
&\quad 3\left(\frac{Df(x_0)\,x_0}{f(x_0)}\frac{D^2f(x_0)\,x_0^2}{f(x_0)} + \frac{Df(x_0)\,x_0}{f(x_0)}\frac{Df(x_0)\,x_0}{f(x_0)}\right) +\\
&\quad 2\frac{Df(x_0)\,x_0}{f(x_0)}\frac{Df(x_0)\,x_0}{f(x_0)}\frac{Df(x_0)\,x_0}{f(x_0)}\\
&= \eta_3 + 3\eta_2 + \eta_1 - 3\eta_1(\eta_1+\eta_2) + 2\eta_1^3.
\end{aligned}
$$

where

$$\eta_1 = \frac{Df(x_0)\,x_0}{f(x_0)}$$
$$\eta_2 = \frac{D^2 f(x_0)\,x_0^2}{f(x_0)}$$
$$\eta_3 = \frac{D^3 f(x_0)\,x_0^3}{f(x_0)}.$$

Thus, we have a third-order expansion

$$\log(f(x)) - \log(f(x_0)) = \eta_1(\log(x) - \log(x_0)) + \frac{1}{2}(\eta_1(1-\eta_1) + \eta_2)(\log(x) - \log(x_0))^2 + \frac{1}{6}\left(\eta_3 + 3\eta_2 + \eta_1 - 3\eta_1(\eta_1 + \eta_2) + 2\eta_1^3\right)(\log(x) - \log(x_0))^3;$$

getting the log-linear expansion is simple once we have the linear one, as it involves only some simple manipulations involving terms we already have computed.

Another example is a power transformation:

$$Y(x) = x^\alpha$$
$$X(y) = y^{\frac{1}{\alpha}}$$
$$h(x) = x^\gamma$$
$$h^{-1}(x) = x^{\frac{1}{\gamma}}.$$

The second-order expansion is

$$f(x)^\gamma = f(x_0)^\gamma + \eta_1\left(\frac{\gamma}{\alpha} f(x_0)^\gamma x_0^{-\alpha}\right)(x^\alpha - x_0^\alpha) + \frac{1}{2}\eta_1\left(\gamma\left(f\left(x_0^{\frac{1}{\alpha}}\right)\right)^\gamma\left((\gamma-1)\,x_0^{\frac{2}{\alpha}(1-2\alpha)}\eta_1 + x_0^{\frac{2}{\alpha}(1-2\alpha)}\eta_2 + \frac{1-\alpha}{\alpha}x_0^{\frac{1-3\alpha}{\alpha}}\right)\right)(x^\alpha - x_0^\alpha)^2$$

where

$$\begin{aligned}
Dg(b) &= Dh(f(X(b)))\,Df(X(b))\,DX(b)\\
&= \frac{\gamma}{\alpha} f(x_0)^{\gamma} x_0^{-\alpha} \frac{Df(x_0)\,x_0}{f(x_0)}\\
&= \eta_1 \left(\frac{\gamma}{\alpha} f(x_0)^{\gamma} x_0^{-\alpha}\right)\\
D^2 g(b) &= D^2 h(f(X(b)))\,(Df(X(b))\,DX(b))^2 + Dh(f(X(b)))\,D^2 f(X(b))\,(DX(b))^2 +\\
&\quad Dh(f(X(b)))\,Df(X(b))\,D^2 X(b)\\
&= \gamma(\gamma-1)\left(f\left(x_0^{\frac{1}{\alpha}}\right)\right)^{\gamma-2}\left(Df\left(x_0^{\frac{1}{\alpha}}\right)x_0^{\frac{1-\alpha}{\alpha}}\right)^2 + \gamma\left(f\left(x_0^{\frac{1}{\alpha}}\right)\right)^{\gamma-1} D^2 f\left(x_0^{\frac{1}{\alpha}}\right)\left(x_0^{\frac{1-\alpha}{\alpha}}\right)^2 +\\
&\quad \gamma\left(f\left(x_0^{\frac{1}{\alpha}}\right)\right)^{\gamma-1} Df\left(x_0^{\frac{1}{\alpha}}\right)\frac{1-\alpha}{\alpha} x_0^{\frac{1-2\alpha}{\alpha}}\\
&= \gamma\left(f\left(x_0^{\frac{1}{\alpha}}\right)\right)^{\gamma}\frac{Df\left(x_0^{\frac{1}{\alpha}}\right)x_0}{f\left(x_0^{\frac{1}{\alpha}}\right)}\left(\begin{array}{l}(\gamma-1)\,x_0^{\frac{2}{\alpha}(1-2\alpha)}\left(\frac{Df\left(x_0^{\frac{1}{\alpha}}\right)x_0}{f\left(x_0^{\frac{1}{\alpha}}\right)}\right)+\\ x_0^{\frac{2}{\alpha}(1-2\alpha)}\frac{D^2 f\left(x_0^{\frac{1}{\alpha}}\right)x_0}{Df\left(x_0^{\frac{1}{\alpha}}\right)} + \frac{1-\alpha}{\alpha}x_0^{\frac{1-3\alpha}{\alpha}}\end{array}\right)\\
&= \eta_1\left(\gamma\left(f\left(x_0^{\frac{1}{\alpha}}\right)\right)^{\gamma}\left((\gamma-1)\,x_0^{\frac{2}{\alpha}(1-2\alpha)}\eta_1 + x_0^{\frac{2}{\alpha}(1-2\alpha)}\eta_2 + \frac{1-\alpha}{\alpha}x_0^{\frac{1-3\alpha}{\alpha}}\right)\right).
\end{aligned}$$

For the stochastic growth model we can construct the linear solutions

$$\begin{aligned}
k' - k^* &= a_1(k-k^*) + b_1 z\\
c - c^* &= c_1(k-k^*) + d_1 z.
\end{aligned}$$

where we set $z^* = 0$ and the loglinear solutions

$$\begin{aligned}
\log(k') - \log(k^*) &= a_2(\log(k) - \log(k^*)) + b_2 z\\
\log(c) - \log(c^*) &= c_2(\log(k) - \log(k^*)) + d_2 z
\end{aligned}$$

or

$$\begin{aligned}
\widehat{k}' &= a_2\widehat{k} + b_2 z\\
\widehat{c} &= c_2\widehat{k} + d_2 z;
\end{aligned}$$

here we use the common notation that

$$\widehat{x} = \frac{x_t - x^*}{x^*} \approx \log(x_t) - \log(x^*)$$

represents the percentage deviation from the steady state, which is approximately the log deviation if the deviation is small.

We now want to change our variables' domains and ranges in order to relate the linear solution to the log-linear one. Define

$$\begin{aligned} h^1 &= \log\left(f^1\right) \\ h^2 &= \log\left(f^2\right) \\ Y^1(x) &= \log\left(x^1\right) \\ Y^2(x) &= x^2 \end{aligned}$$

which imply the inverse maps are

$$\begin{aligned} X^1 &= \exp\left(y^1\right) \\ X^2 &= y^2. \end{aligned}$$

That is, we change the domain of x^1 from the space of capital stocks to the space of the log of the capital stocks and we leave the domain of the shock unchanged. We also change the range of the capital decision rule from capital to log capital and change the range of the labor decision from consumption to log consumption.

Applying the formula yields

$$\begin{bmatrix} \log(k')(\log(k), z) \\ \log(c)(\log(k), z) \end{bmatrix} = g(\log(k), z)$$

$$= \begin{bmatrix} \log f^1(k^*, 0, 0) \\ \log f^2(k^*, 0, 0) \end{bmatrix} + \begin{bmatrix} \frac{dh^1}{df^1}\left(f_1^1(k^*, 0, 0), f_2^1(k^*, 0, 0)\right) \begin{pmatrix} k^* \\ 1 \end{pmatrix} \\ \frac{dh^2}{df^2}\left(f_1^2(k^*, 0, 0), f_2^2(k^*, 0, 0)\right) \begin{pmatrix} k^* \\ 1 \end{pmatrix} \end{bmatrix} \begin{bmatrix} \log(k) - \log(k^*) & z \\ \log(k) - \log(k^*) & z \end{bmatrix}$$

$$= \begin{bmatrix} \log f^1(k^*, 0, 0) \\ \log f^2(k^*, 0, 0) \end{bmatrix} + \begin{bmatrix} \frac{1}{k^*}\left(f_1^1(k^*, 0, 0), f_2^1(k^*, 0, 0)\right) \begin{pmatrix} k^* \\ 1 \end{pmatrix} \\ \frac{1}{l^*}\left(f_1^2(k^*, 0, 0), f_2^2(k^*, 0, 0)\right) \begin{pmatrix} k^* \\ 1 \end{pmatrix} \end{bmatrix} \begin{bmatrix} \log(k) - \log(k^*) & z \\ \log(k) - \log(k^*) & z \end{bmatrix}$$

or

$$\log\left(k'\right) - \log\left(k^*\right) = f_1^1\left(k^*,0,0\right)\left(\log\left(k\right) - \log\left(k^*\right)\right) + \frac{1}{k^*}f_2^1\left(k^*,0,0\right)z$$
$$\log\left(c\right) - \log\left(c^*\right) = \frac{k^*}{l^*}f_1^2\left(k^*,0,0\right)\left(\log\left(k\right) - \log\left(k^*\right)\right) + \frac{1}{l^*}f_2^2\left(k^*,0,0\right)z,$$

where

$$k' - k^* = f_1^1\left(k^*,0,0\right)\left(k - k^*\right) + f_2^1\left(k^*,0,0\right)z$$
$$l - l^* = f_1^2\left(k^*,0,0\right)\left(k - k^*\right) + f_2^2\left(k^*,0,0\right)z$$

is the linear solution. Equating coefficients yields

$$a_2 = a_1$$
$$b_2 = \frac{1}{k^*}b_1$$
$$c_2 = \frac{k^*}{c^*}c_1$$
$$d_2 = \frac{1}{c^*}d_1.$$

Note that we could obtain the same result by simply doing the following. Take the linear solution and divide by k^*:

$$\frac{k' - k^*}{k^*} = a_1\frac{k - k^*}{k^*} + \frac{b_1}{k^*}z$$
$$\frac{c - c^*}{k^*} = c_1\frac{k - k^*}{k^*} + \frac{d_1}{k^*}z.$$

The first equation is

$$\widehat{k}' = a_1\widehat{k} + \frac{b_1}{k^*}z$$

which is the log-expansion. The second equation requires some manipulation:

$$\frac{c - c^*}{c^*} = \frac{k^*}{c^*}c_1\frac{k - k^*}{k^*} + \frac{d_1}{c^*}z,$$

which is

$$\widehat{c} = \frac{k^*}{c^*}c_1\widehat{k} + \frac{d_1}{c^*}z.$$

It is also straightforward to obtain the quadratic solution in logs:

$$k' - k^* = a_1\left(k - k^*\right) + b_1 z + c_1\left(k - k^*\right)^2 + d_1\left(k - k^*\right)z + e_1 z^2 + f_1\sigma^2$$
$$c - c^* = a_2\left(k - k^*\right) + b_2 z + c_2\left(k - k^*\right)^2 + d_2\left(k - k^*\right)z + e_2 z^2 + f_2\sigma^2$$

becomes

$$\frac{k'-k^*}{k^*} = a_1\left(\frac{k-k^*}{k^*}\right) + \frac{b_1}{k^*}z + c_1 k^*\left(\frac{k-k^*}{k^*}\right)^2 + d_1\left(\frac{k-k^*}{k^*}\right)z + \frac{e_1}{k^*}z^2 + \frac{f_1}{k^*}\sigma^2$$
$$\frac{c-c^*}{c^*} = \frac{a_2}{c^*}k^*\left(\frac{k-k^*}{k^*}\right) + \frac{b_2}{c^*}z + \frac{c_2}{c^*}\left(k^*\right)^2\left(\frac{k-k^*}{k^*}\right)^2 + \frac{d_2}{c^*}k^*\left(\frac{k-k^*}{k^*}\right)z + \frac{e_2}{c^*}z^2 + \frac{f_2}{c^*}\sigma^2$$

or

$$\widehat{k'} = a_1\widehat{k} + \frac{b_1}{k^*}z + c_1 k^*\widehat{k}^2 + d_1\widehat{k}z + \frac{e_1}{k^*}z^2 + \frac{f_1}{k^*}\sigma^2$$
$$\widehat{c} = a_2\frac{k^*}{c^*}\widehat{k} + \frac{b_2}{c^*}z + c_2\frac{\left(k^*\right)^2}{c^*}\widehat{k}^2 + d_2\frac{k^*}{c^*}\widehat{k}z + \frac{e_2}{c^*}z^2 + \frac{f_2}{c^*}\sigma^2.$$

The power transformation yields

$$\left(k'\right)^\gamma - \left(k^*\right)^\gamma = a_3\left(k^\zeta - \left(k^*\right)^\zeta\right) + b_3 z$$
$$c^\mu - \left(c^*\right)^\mu = c_3\left(k^\zeta - \left(k^*\right)^\zeta\right) + d_3 z;$$

note that the power on k' differs from the one on k and the one on l. The change of variables for this family of functions is given by

$$h^1 = \left(f^1\right)^\gamma$$
$$h^2 = \left(f^2\right)^\mu$$
$$Y^1 = \left(x^1\right)^\zeta$$
$$Y^2 = x^2$$

with inverse maps

$$X^1 = \left(y^1\right)^{\frac{1}{\zeta}}$$
$$X^2 = y^2.$$

The relationship between the coefficients of the linear solution and the coefficients of this solution are

$$a_3 = \frac{\gamma}{\zeta}\left(k^*\right)^{\gamma-\zeta}a_1$$
$$b_3 = \gamma\left(k^*\right)^{\gamma-1}b_1$$
$$c_3 = \frac{\mu}{\zeta}\left(c^*\right)^{\mu-1}\left(k^*\right)^{1-\zeta}c_1$$
$$d_3 = \mu\left(c^*\right)^{\mu-1}d_1,$$

so that it is immediate to obtain a nonlinear solution of this form. Even more general transformations which do not require the same domain for each equation can be considered (that is, we could transform the k in the first equation in a manner different from the k in the second equation); obviously the resulting expressions will be complicated.

The second-order solutions take the form

$$\begin{aligned} k' - k^* &= a_1 (k - k^*) + b_1 z + c_1 (k - k^*)^2 + d_1 z^2 + e_1 \sigma^2 + f_1 (k - k^*) z \\ c - c^* &= a_2 (k - k^*) + b_2 z + c_2 (k - k^*)^2 + d_2 z^2 + e_2 \sigma^2 + f_2 (k - k^*) z \end{aligned}$$

and

$$\begin{aligned} \left(k'\right)^\gamma - (k^*)^\gamma &= a_3 \left(k^\zeta - (k^*)^\zeta\right) + b_3 z + c_3 \left(k^\zeta - (k^*)^\zeta\right)^2 + d_3 z^2 + e_3 \sigma^2 + f_3 \left(k^\zeta - (k^*)^\zeta\right) z \\ c^\mu - (c^*)^\mu &= a_4 \left(k^\zeta - (k^*)^\zeta\right) + b_4 z + c_4 \left(k^\zeta - (k^*)^\zeta\right)^2 + d_4 z^2 + e_4 \sigma^2 + f_4 \left(k^\zeta - (k^*)^\zeta\right) z. \end{aligned}$$

The coefficients are related by

$$\begin{aligned} a_3 &= \frac{\gamma}{\zeta} (k^*)^{\gamma-\zeta} a_1 \\ b_3 &= \gamma (k^*)^{\gamma-1} b_1 \\ c_3 &= \frac{\gamma}{\zeta^2} (k^*)^{\gamma-2\zeta} (c_1 k^* + a_1 (1 + a_1 (\gamma - 1) - \zeta)) \\ d_3 &= \gamma (k^*)^{\gamma-2} \left(d_1 k^* + b_1^2 (\gamma - 1)\right) \\ e_3 &= e_1 (k^*)^{\gamma-1} \gamma \\ f_3 &= \frac{\gamma}{\zeta} (k^*)^{\gamma-\zeta-1} (f_1 k^* + a_1 b_1 (\gamma - 1)) \end{aligned}$$

and

$$\begin{aligned} a_4 &= \frac{\mu}{\zeta} (c^*)^{\mu-1} (k^*)^{1-\zeta} a_2 \\ b_4 &= \mu (c^*)^{\mu-1} b_2 \\ c_4 &= \frac{\mu}{\zeta^2} (k^*)^{1-2\zeta} (c^*)^{\gamma-2} (c_1 k^* c^* + a_2 (c^* (1 - \zeta) + k^* a_2 (\mu - 1))) \\ d_4 &= \mu (c^*)^{\mu-2} \left(d_2 c^* + b_2^2 (\mu - 1)\right) \\ e_4 &= e_2 (c^*)^{\mu-1} \mu \\ f_4 &= \frac{\mu}{\zeta} (k^*)^{1-\zeta} (c^*)^{\mu-2} (f_2 c^* + a_2 b_2 (\mu - 1)). \end{aligned}$$

Nonlinear expansions can be handled in a straightforward manner by making the appropriate substitutions into the Schmitt-Grohe/Uribe/Ruge-Murcia programs. For example, in their code it is easy to obtain the log-linear solution by using the command

f=subs(f,[x,y,xp,yp],exp([x,y,xp,yp]))

which replaces all variables in the vector x with e^x and so on; now x should be interpreted as $\log(x)$. For the power functions, we make the transformation

f=subs(f,x1,x1^(1/zeta))

to replace the variable x^1 with y^1; do not forget to change the steady state calculations to the new variable definitions or you will get the wrong solutions.

An important consideration is how to choose $(\gamma, \mu, \zeta, \varphi)$ in the power transformation. Fernández-Villaverde and Rubio-Ramírez choose these coefficients to minimize the Euler errors over a grid of points, finding that for a plausible calibration the capital equation remains close to linear ($\gamma = \zeta \approx 1$) but the labor supply function deviates significantly from linearity (μ differs from 1 by a large amount); the Euler errors fall by a factor of three. In the second-order case they find $\gamma = \zeta$ but not close to 1 and μ again well above 1, and they cut the errors in half. For more general models, a hybrid approach that combines perturbation with projection methods (to be discussed below) can be used to compute these coefficients; unfortunately, we do not currently have any general results.

An important implication of the change of variables approach is that the solution remains linear in a nonlinear transformation of the data if $\gamma = \zeta$ so that linear estimation techniques like Kalman filtering can still be applied; numerical experiments suggest that such an equality is warranted, but in any case it could be used even if it wasn't strictly optimal (as long as it does "better"). In such an environment the likelihood becomes a function of those coefficients; you could estimate them by penalizing the Euler equation errors at the data points.

8.14 Expansion Around the Risky Steady State

Many models do not have a deterministic steady state, or (as in portfolio problems) they may have many. One commonly-used model that has this problem is the small open model economy

with a fixed discount factor β and exogenous world interest rate r. Here, there are either a continuum of deterministic steady states (if $\beta r = 1$) or none (if $\beta r < 1$). The third case, $\beta r > 1$, leads to explosive behavior and will not be considered. However, these models may possess a "stochastic steady state" if $\beta r < 1$, where precautionary savings motives are balanced exactly by impatience. Note that this approach is very similar to the cases where the deterministic steady state is not known; here, the center of approximation is the stochastic steady state where the "displacement" caused by precautionary savings is not known because it depends on the entire policy function. The difference is that the class of models under consideration here do not have a deterministic steady state at all.

To fix notation, suppose we have a state space consisting on an exogenous vector X_t and an endogenous vector Y_t, which yields the state space representation

$$\begin{aligned} X_t &= (1-\rho)\,\overline{X} + \rho X_{t-1} + \epsilon_t \\ Y_t &= g\left(Y_{t-1}, X_t\right) \\ S_t &= \left(Y_{t-1}, X_t\right). \end{aligned}$$

We are looking for a solution to the functional equation

$$E_t\left[f\left(Y_{t+1}, Y_t, Y_{t-1}, X_{t+1}, X_t, X_{t-1}\right)\right] = 0.$$

To find the "risky steady state", we take second-order expansion around $E_t V_{t+1}$ where

$$\begin{aligned} V_{t+1} &= \left(Y_{t+1}, Y_t, Y_{t-1}, X_{t+1}, X_t, X_{t-1}\right) \\ E_t V_{t+1} &= \left(E_t Y_{t+1}, Y_t, Y_{t-1}, (1-\rho)\,\overline{X} + \rho X_t, X_t, X_{t-1}\right). \end{aligned}$$

Define this function as $\Phi\left(S_t\right)$:

$$\Phi\left(S_t\right) = f\left(E_t V_{t+1}\right) + E_t\left[D^2 f\left(E_t V_{t+1}\right)\left(V_{t+1} - E_t V_{t+1}\right)^2\right].$$

We are looking for a linear decision rule

$$Y_t = \overline{Y} + \pi_Y\left(S_t - \overline{S}\right)$$

where $\overline{S}$ is the risky steady state, which differs from the deterministic steady state S^* due to precautionary effects (we defined the stochastic steady state in an earlier section; risky steady

state is just another name). To find the coefficients $\left(\overline{Y}, \pi_Y\right)$, we use the Φ function and its gradient:

$$\Phi\left(\overline{S}\right) = 0$$
$$\frac{\partial \Phi}{\partial S_t}\left(\overline{S}\right) = 0.$$

An example will be helpful; we use a standard consumption-savings model which has first-order conditions

$$1 = \beta E_t\left[\left(\frac{c_{t+1}}{c_t}\right)^{-\gamma} r_{t+1}\right]$$
$$c_t = y_t + w_{t-1} r_t - w_t;$$

The deterministic steady state satisfies

$$c^* = \overline{y} + w^*\left(\overline{r} - 1\right)$$
$$\beta \overline{r} = 1$$

where

$$\overline{r} = E\left[r\right]$$
$$\overline{y} = E\left[y\right]$$

but w^* is undetermined. Thus, this model possesses either infinitely many steady states or none, depending on whether the interest rate satisfies the required condition.

We will look for a linear policy function of the form

$$w_t = \overline{w} + \pi_w\left(w_{t-1} - \overline{w}\right) + \pi_y\left(y_t - \overline{y}\right) + \pi_r\left(r_t - \overline{r}\right)$$

and and solve for $(\overline{w}, \overline{c}, \pi_w, \pi_y, \pi_r)$ where

$$\overline{c} = \overline{y} + \overline{w}\left(\overline{r} - 1\right)$$

is risky steady state consumption. Take a quadratic approximation to the Euler equation

$$\frac{1}{\beta}\left(\frac{E_t c_{t+1}}{c_t}\right)^{-\gamma} = E_t\left[r_{t+1}\right]\left(1 + \gamma\left(1+\gamma\right)\frac{Var_t c_{t+1}}{\left(E_t c_{t+1}\right)^2}\right) - \gamma\frac{Cov_t\left(c_{t+1}, r_{t+1}\right)}{E_t c_{t+1}}.$$

and evaluate it at the risky steady state to obtain

$$\frac{1}{\beta} = \overline{r}\left(1+\gamma\left(1+\gamma\right)\frac{\overline{var}\left(c_{t+1}\right)}{\overline{c}^2}\right) - \gamma\frac{\overline{cov}\left(c_{t+1},r_{t+1}\right)}{\overline{c}}.$$

Since

$$\begin{aligned}
w_{t+1} - E_t w_{t+1} &= \overline{w} + \pi_w\left(w_t - \overline{w}\right) + \pi_y\left(y_{t+1} - \overline{y}\right) + \pi_r\left(r_{t+1} - \overline{r}\right) - E_t\left[\overline{w} + \pi_w\left(w_t - \overline{w}\right) + \pi_y\left(y_{t+1} - \overline{y}\right) + \pi_r\left(r_{t+}\right.\right. \\
&= \pi_y y_{t+1} + \pi_r r_{t+1} - \pi_y E_t y_{t+1} - \pi_r E_t r_{t+1} \\
&= \pi_y \epsilon_{y,t+1} + \pi_r \epsilon_{r,t+1}
\end{aligned}$$

we can compute the second-order moments of c_{t+1} as

$$\begin{aligned}
Var_t c_{t+1} &= E_t\left(c_{t+1} - E_t c_{t+1}\right)^2 \\
&= E_t\left(y_{t+1} + w_t r_{t+1} - w_{t+1} - E_t\left(y_{t+1} + w_t r_{t+1} - w_{t+1}\right)\right)^2 \\
&= E_t\left(y_{t+1} - E_t y_{t+1} + w_t\left(r_{t+1} - E_t r_{t+1}\right) - \left(w_{t+1} - E_t w_{t+1}\right)\right)^2 \\
&= E_t\left(\left(1-\pi_y\right)\epsilon_{y,t+1} + w_t\left(1-\pi_r\right)\epsilon_{r,t+1}\right)^2 \\
&= \left(1-\pi_y\right)^2\sigma_y^2 + w_t^2\left(1-\pi_r\right)^2\sigma_r^2 \\
\overline{var}\left(c_{t+1}\right) &= \left(1-\pi_y\right)^2\sigma_y^2 + \overline{w}^2\left(1-\pi_r\right)^2\sigma_r^2 \\
Cov_t\left(c_{t+1,}r_{t+1}\right) &= E_t\left(c_{t+1} - E_t c_{t+1}\right)\left(r_{t+1} - E_t r_{t+1}\right) \\
&= E_t\left(\left(1-\pi_y\right)\epsilon_{y,t+1} + w_t\left(1-\pi_r\right)\epsilon_{r,t+1}\right)\epsilon_{r,t+1} \\
&= w_t\left(1-\pi_r\right)\sigma_r^2 \\
\overline{cov}\left(c_{t+1},r_{t+1}\right) &= \overline{w}\left(1-\pi_r\right)\sigma_r^2.
\end{aligned}$$

Using these expressions we get

$$
\begin{aligned}
0 &= \frac{1}{\beta}\left(\frac{E_t c_{t+1}}{c_t}\right)^{-\gamma} - E_t\left[r_{t+1}\right]\left(1+\gamma\left(1+\gamma\right)\frac{\left(1-\pi_y\right)^2\sigma_y^2 + w_t^2\left(1-\pi_r\right)^2\sigma_r^2}{\left(E_t c_{t+1}\right)^2}\right) + \gamma\frac{w_t\left(1-\pi_r\right)\sigma_r^2}{E_t c_{t+1}} \\
w_t &= \overline{w} + \pi_w\left(w_{t-1}-\overline{w}\right) + \pi_y\left(y_t-\overline{y}\right) + \pi_r\left(r_t-\overline{r}\right) \\
c_t &= y_t + w_{t-1}r_t - w_t \\
E_t c_{t+1} &= E_t\left(y_{t+1} + w_t r_{t+1} - w_{t+1}\right) \\
&= \left(1-\rho_y\right)\overline{y} + \rho_y y_t + w_t\left(\left(1-\rho_r\right)\overline{r} + \rho_r r_t\right) - \overline{w} - \pi_w\left(w_t-\overline{w}\right) - \pi_y\left(E_t y_{t+1}-\overline{y}\right) - \pi_r\left(E_t r_{t+1}-\overline{r}\right) \\
&= \left(1-\rho_y\right)\overline{y} + \rho_y y_t + w_t\left(\left(1-\rho_r\right)\overline{r} + \rho_r r_t\right) - \overline{w} - \pi_w\left(w_t-\overline{w}\right) - \pi_y\left(-\rho_y\overline{y} + \rho_y y_t\right) - \pi_r\left(-\rho_r\overline{r} + \rho_y r_t\right) \\
&= \left(1-\rho_y+\pi_y\rho_y\right)\overline{y} - \left(1-\pi_w\right)\overline{w} + \pi_r\overline{r} + \left(1-\pi_y\right)\rho_y y_t - \pi_r\rho_y r_t + w_t\left(\left(1-\rho_r\right)\overline{r} - \pi_w + \rho_r r_t\right) \\
&= \left(1-\rho_y+\pi_y\rho_y\right)\overline{y} - \left(1-\pi_w\right)\overline{w} + \pi_r\overline{r} + \left(1-\pi_y\right)\rho_y y_t - \pi_r\rho_y r_t + \\
&\left(\overline{w} + \pi_w\left(w_{t-1}-\overline{w}\right) + \pi_y\left(y_t-\overline{y}\right) + \pi_r\left(r_t-\overline{r}\right)\right)\left(\left(1-\rho_r\right)\overline{r} - \pi_w + \rho_r r_t\right).
\end{aligned}
$$

Therefore, $\Phi\left(\overline{S}\right) = 0$ yields

$$
\begin{aligned}
\frac{1}{\beta} &= \overline{r}\left(1+\gamma\left(1+\gamma\right)\frac{\left(1-\pi_y\right)^2\sigma_y^2 + \overline{w}^2\left(1-\pi_r\right)^2\sigma_r^2}{\overline{c}^2}\right) - \gamma\frac{\overline{w}\left(1-\pi_r\right)\sigma_r^2}{\overline{c}} \\
\overline{c} &= \overline{y} + \overline{w}\left(\overline{r}-1\right).
\end{aligned}
$$

and then we take derivatives with respect to (w_{t-1}, y_t, r_t), which we can easily do using automatic differentiation.

In models without a deterministic steady state, this approach turns out to be quite inaccurate, both for small shocks (where precautionary effects are weak) and for large shocks (where unstable dynamics often appear because the local approximation is poor). The reason is that a risky steady state only exists if the precautionary savings effect is balanced by the impatience effect ($\beta r < 1$), and that balancing is difficult to obtain using information only at the steady state itself. Den Haan, Kobielarz, and Rendahl (2016) find that solving this model globally is a challenge as well, as it frequently displays "escape dynamics" that blow wealth up to arbitrarily-high values.

8.15 Padé Approximation

Padé approximation uses rational functions – functions that are the ratio of polynomials – to construct the approximation. They are easy to construct once we have a Taylor expansion.

Suppose we have the third-order expansion

$$f(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3.$$

We can then choose two polynomials

$$P(x) = a_0 + a_1 x + a_2 x^2$$
$$Q(x) = 1 + b_1 x$$

such that

$$\frac{P(x)}{Q(x)} = f(x)$$

by matching coefficients. That is, we rewrite this equation as

$$(1 + b_1 x)\left(c_0 + c_1 x + c_2 x^2 + c_3 x^3\right) = a_0 + a_1 x + a_2 x^2.$$

The coefficients are then

$$b_1 = -\frac{c_3}{c_2}$$
$$a_2 = c_2 - \frac{c_1 c_3}{c_2}$$
$$a_1 = c_1 - \frac{c_0 c_3}{c_2}$$
$$a_0 = c_0,$$

where we ignore the x^4 term on the left-hand-side. It can be the case that the Padé approximant is more accurate than the Taylor approximant, and it is trivial to obtain the coefficients from the Taylor expansion so it should always be checked. Note that the order of the Taylor expansion should equal the sum of the orders of P and Q.

The extension of the Padé approximant to multiple dimensions is called a Canterbury approximant. For two-dimensional expansions, a Canterbury approximant takes the form

$$f(x,y) = \frac{P(x,y)}{Q(x,y)} = \frac{\sum_{i=0}^{m_1}\sum_{j=0}^{m_2} p_{ij} x^i y^j}{\sum_{i=0}^{n_1}\sum_{j=0}^{n_2} q_{ij} x^i y^j}$$

with the usual normalization that $q_{00} = 1$. The procedure is identical to the Padé approximant – match the coefficients and discard any higher order terms. For example, consider the Taylor

expansion in two variables

$$\begin{aligned} &a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3 \\ &= \frac{p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 + p_{03}y^3}{1 + q_{10}x + q_{01}y} \end{aligned}$$

The Canterbury approximation satisfies

$$\begin{aligned} &(1 + q_{10}x + q_{01}y)\left(a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3\right) \\ &= p_{00} + p_{10}x + p_{01}y + p_{20}x^2 + p_{11}xy + p_{02}y^2 + p_{30}x^3 + p_{21}x^2y + p_{12}xy^2 + p_{03}y^3 \end{aligned}$$

which requires that

$$\begin{aligned} a_{00} &= p_{00} \\ a_{00}q_{10} + a_{10} &= p_{10} \\ a_{00}q_{01} + a_{01} &= p_{01} \\ a_{10}q_{10} + a_{20} &= p_{20} \\ q_{10}a_{01} + q_{01}a_{10} + a_{11} &= p_{11} \\ q_{01}a_{01} + a_{02} &= p_{02} \\ q_{10}a_{20} + a_{30} &= p_{30} \\ q_{01}a_{02} + a_{03} &= p_{03} \\ q_{10}a_{11} + a_{21} &= p_{21} \\ q_{01}a_{11} + a_{12} &= p_{12}; \end{aligned}$$

as before we discard any terms of the RHS that are of higher order than cubic. The solution is

$$\begin{aligned}
p_{00} &= a_{00} \\
q_{10} &= -\frac{a_{30}}{a_{20}} \\
q_{01} &= -\frac{a_{03}}{a_{02}} \\
p_{10} &= -\frac{a_{30}}{a_{20}} a_{00} + a_{10} \\
p_{01} &= -\frac{a_{03}}{a_{02}} a_{00} + a_{01} \\
p_{20} &= -\frac{a_{30}}{a_{20}} a_{10} + a_{20} \\
p_{11} &= -\frac{a_{30}}{a_{20}} a_{01} - \frac{a_{03}}{a_{02}} a_{10} + a_{11} \\
p_{02} &= -\frac{a_{03}}{a_{02}} a_{01} + a_{02}.
\end{aligned}$$

Since Padé and Canterbury approximants are trivial to construct (at least in relatively low dimensions), one should always check them.

8.16 Markov-Switching Perturbation

Perturbation can be applied to so-called Markov-switching models, where some exogenous states are discrete and evolve as a Markov chain, while all the others are continuous. Here, we follow Foerster, Rubio-Ramirez, Waggoner, and Zha (2016).

Let $s_t \in \{1, ..., n_s\}$ be the discrete Markov process with transition matrix $P = \left[p_{s_t, s_{t+1}}\right]$; $p_{s_t, s_{t+1}}$ is the probability of switching from s_t at time t to s_{t+1} at time $t+1$. Collect all the switching states (think of them as parameters of the underlying model) in $\theta(s_t)$. Denote the invariant probabilities of the process by $\overline{p}$ and the mean by $\overline{\theta}$.

The model can now be written as

$$E_t\left[f\left(y_{t+1}, y_t, x_t, x_{t-1}, \varepsilon_{t+1}, \varepsilon_t, \theta\left(s_{t+1}\right), \theta\left(s_t\right)\right)\right] = 0.$$

y_t are the jump variables, x_t are the states (both endogenous and exogenous), ε_t are the shocks with $E_t\left[\varepsilon_{t+1}\right] = 0$ and $E_t\left[\varepsilon_{t+1}\varepsilon_{t+1}^T\right] = I$. The function f then maps points in $\mathcal{R}^{2(n_y+n_x+n_\varepsilon+n_\theta)}$ into points in $\mathcal{R}^{n_y+n_x}$. Suppose that f is infinitely differentiable, uniformly continuous (so we can swap differentiation and integration), and has steady state values x_{ss} and y_{ss} that satisfy

$$f\left(y_{ss}, y_{ss}, x_{ss}, x_{ss}, 0, 0, \overline{\theta}, \overline{\theta}\right) = 0.$$

As a simple example, suppose we have the stochastic growth model with a shock process whose parameters switch:

$$\max_{\{c_t,k_t\}_{t=0}^{\infty}} \left\{ E_0 \left[\sum_{t=0}^{\infty} \beta^t \frac{c_t^{\upsilon}}{\upsilon} \right] \right\}$$

subject to

$$\begin{aligned} c_t + k_t &= z_t^{1-\alpha} k_{t-1}^{\alpha} + (1-\delta) k_{t-1} \\ \log\left(\frac{z_t}{z_{t-1}}\right) &= (1-\rho(s_t))\,\mu(s_t) + \rho(s_t)\log\left(\frac{z_{t-1}}{z_{t-2}}\right) + \sigma(s_t)\,\varepsilon \\ & k_0, z_0, s_0 \text{ given.} \end{aligned}$$

Using the standard normalization method we get

$$E_t[f] = E_t \left[\begin{array}{c} \widetilde{c}_t^{\upsilon-1} - \beta \widetilde{z}_t^{\upsilon-1} \widetilde{c}_{t+1}^{\upsilon-1} \left(\alpha \exp\left[(1-\rho(s_{t+1}))\,\mu(s_{t+1}) + \rho(s_{t+1}) \log(\widetilde{z}_t) + \sigma(s_{t+1})\,\varepsilon_{t+1} \right] (1-\alpha)\, \widetilde{k}_t^{\alpha-1} + 1 - \right. \\ \widetilde{c}_t + \widetilde{k}_t - \widetilde{z}_t^{1-\alpha} \widetilde{k}_t^{\alpha} - (1-\delta)\,\widetilde{k}_{t-1} \\ \log(\widetilde{z}_t) - (1-\rho(s_t))\,\mu(s_t) - \rho(s_t)\log(\widetilde{z}_{t-1}) - \sigma(s_t)\,\varepsilon_t \end{array} \right]$$

Therefore

$$\begin{aligned} y_t &= \widetilde{c}_t \\ x_t &= \left[\widetilde{z}_t, \widetilde{k}_t \right]. \end{aligned}$$

The steady state solves

$$\begin{aligned} \widetilde{z}_{ss} &= \exp(\overline{\mu}) \\ \widetilde{k}_{ss} &= \left(\frac{\alpha\beta \exp((1-\upsilon)\,\overline{\mu})}{\exp((\alpha-1)\,\overline{\mu})\,(1-\beta(1-\delta))} \right)^{\frac{1}{1-\alpha}} \\ \overline{c}_{ss} &= \left(1 - \delta - \exp(\overline{\mu}) + \frac{\beta^{-1} \exp((1-\upsilon)\,\overline{\mu}) - 1 + \delta}{\alpha} \right) \widetilde{k}_{ss}. \end{aligned}$$

Note that the steady state does not depend on $\overline{\rho}$ or $\overline{\sigma}$; therefore, we will not generate perturbation expansions in those variables. So we apply the FRWZ "Partition Principle" and divide θ into

$$\theta = \left[\begin{array}{c} \theta_1 \\ \theta_2 \end{array} \right] = \left[\begin{array}{c} \mu \\ (\rho, \sigma) \end{array} \right].$$

We will build our approximation using the perturbation function

$$\theta(k, \chi) = \left[\begin{array}{c} \overline{\theta}_1 + \chi\left(\theta_1(k) - \overline{\theta}_1\right) \\ \overline{\theta}_2 \end{array} \right]$$

for $k \in \{1, ..., n_s\}$. When $\chi = 0$ we have $\theta(k, \chi) = \overline{\theta}_1$ (we center here) and when $\chi = 1$ we have $\theta(k, \chi) = \theta_1(k)$ (we evaluate here).

Now define

$$F_{s_t}(y_t, x_t, x_{t-1}, \varepsilon_t, \chi) = \sum_{s_{t+1}=1}^{n_s} p_{s_t, s_{t+1}} \int_{\mathcal{R}^{n_\varepsilon}} f\left(g_{s_{t+1}}(x_t, \chi\varepsilon_{t+1}, \chi), y_t, x_t, x_{t-1}, \chi\varepsilon_{t+1}, \varepsilon_t, \theta(s_{t+1}, \chi), \theta(s_t, \chi)\right) d\mu(\varepsilon_{t+}$$

where

$$y_t = g_{s_t}(x_{t-1}, \varepsilon_t, \chi)$$
$$x_t = h_{s_t}(x_{t-1}, \varepsilon_t, \chi)$$

solve

$$F_{s_t}(y_t, x_t, x_{t-1}, \varepsilon_t, \chi) = 0.$$

Now stack the regime dependent solutions into

$$Y_t = G(x_{t-1}, \varepsilon_t, \chi) = \begin{bmatrix} g_1(x_{t-1}, \varepsilon_t, \chi) \\ \vdots \\ g_{n_s}(x_{t-1}, \varepsilon_t, \chi) \end{bmatrix}$$
$$X_t = H(x_{t-1}, \varepsilon_t, \chi) = \begin{bmatrix} h_1(x_{t-1}, \varepsilon_t, \chi) \\ \vdots \\ h_{n_s}(x_{t-1}, \varepsilon_t, \chi) \end{bmatrix}.$$

Then we have

$$\mathbf{F}(Y_t, X_t, x_{t-1}, \varepsilon_t, \chi) = 0.$$

A first-order solution would be of the form

$$G(x_{t-1}, \varepsilon_t, \chi) = G(x_{ss}, 0, 0) + DG(x_{ss}, 0, 0) \begin{bmatrix} x_{t-1} - x_{ss} \\ \varepsilon_t \\ \chi \end{bmatrix}$$
$$H(x_{t-1}, \varepsilon_t, \chi) = H(x_{ss}, 0, 0) + DH(x_{ss}, 0, 0) \begin{bmatrix} x_{t-1} - x_{ss} \\ \varepsilon_t \\ \chi \end{bmatrix}.$$

Solving for the derivative terms $D_{x_{t-1}}$ is harder than in the no-regime-switching case, as we end up with a system of $n_s\left(n_y+n_x\right)n_x$ quadratic polynomial equations in the same number of unknowns. We can then solve a linear system of size $n_s\left(n_y+n_x\right)n_\varepsilon$ to get D_{ε_t} and a linear system of size $n_s\left(n_y+n_x\right)$ to get D_χ, which may not be zero as they would be in a standard non-regime-switching model. The second order solutions are again linear in all derivatives, despite also varying with the regime. Note that the derivative terms are functions of

$$u_{ss}=\left[y_{ss},y_{ss},x_{ss},x_{ss},0,0,\theta\left(s_{t+1},0\right),\theta\left(s_t,0\right)\right],$$

and so change with the regime (that is, they change with the elements of θ_2). A standard perturbation approach would not have this feature because it treats them as variables, and it turns out to improve accuracy.

We know that finding all the solutions to a system of polynomial equations is hard, and we need all of them to determine how many are stable. FRWZ use a Grobner basis to find all the solutions, specifically the one implied by the Shape Lemma, which makes it easy to obtain all the solutions. We can then determine which solutions are stable and therefore constitute an equilibrium. Stability in Markov-switching models is more complicated than in standard models, so FRWZ advocate using the mean-square stability criterion rather than the usual boundedness requirement. To be concrete, we have the following definitions.

Definition 89 *An n-dimensional process x_t is **mean-square stable** (MSS) iff there exists an n-vector μ and an $n\times n$ positive semi-definite matrix Σ such that*

$$\lim_{t\to\infty}\left\{E_0\left[x_t\right]\right\}=\mu$$
$$\lim_{t\to\infty}\left\{E_0\left[x_tx_t^T\right]\right\}=\Sigma.$$

If in addition there exists a real number N such that

$$\|x_t\|\leq N$$

*$\forall t$, then x_t is **boundedly stable**.*

For linear systems with bounded shocks, the two notions are equivalent (MSS implies bounded stability). In general systems, that is not true, and in particular for regime switching it does not

generally hold. The main reason that we need a different definition is that the system can be unstable in one regime; provided that regime does not occur too frequently, the system overall can be mean-square stable. Note also that a system can be unbounded even if the system in each regime is stable: consider the example in Farmer, Waggoner, and Zha (2009)

$$x_t = A_{s_t} x_{t-1}$$

with

$$A_1 = \begin{bmatrix} 0 & 2 \\ 0 & 0.5 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} 0.5 & 0 \\ 2 & 0 \end{bmatrix}.$$

s_t switches according to some transition matrix; note that the eigenvalues of both A_1 and A_2 are inside the unit circle, but the product matrix

$$A_1 A_2 = \begin{bmatrix} 4 & 0 \\ 1 & 0 \end{bmatrix}$$

has eigenvalues $\{1, 4\}$ and is therefore not boundedly stable. If the system would remain in one regime it would be boundedly stable, but since it switches back and forth it is not. However, it can be MSS, if the transition matrix s_t is

$$\Pi = \begin{bmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{bmatrix},$$

as the transitions do not occur too frequently. The key to MSS are the eigenvalues of

$$\begin{bmatrix} \pi_{11} A_1 \otimes A_1 & \pi_{21} A_2 \otimes A_2 \\ \pi_{12} A_1 \otimes A_1 & \pi_{22} A_2 \otimes A_2 \end{bmatrix};$$

if they are inside the unit circle, the system is MSS.

In the DSGE case, MSS requires that the eigenvalues of the $n_s n_x^2 \times n_s n_x^2$ block-diagonal matrix

$$\left(P^T \otimes I_{n_x^2}\right) \operatorname{diag}\left(D_{x_{t-1}} h_1 \otimes D_{x_{t-1}} h_1 \cdots D_{x_{t-1}} h_{n_s} \otimes D_{x_{t-1}} h_{n_s}\right)$$

are all inside the unit circle.

We'll now do a simple example. Take the two equations

$$R_t = r + E_t\left[\pi_{t+1}\right]$$
$$R_t = R + \phi\left(s_t\right)\left(\pi_t - \pi\right) + \sigma\left(s_t\right)\varepsilon_t,$$

a Fisher equation and a regime-switching Taylor rule. Let

$$\widehat{\pi}_t = \pi_t - \pi$$

and we get the single equation

$$\phi\left(s_t\right)\widehat{\pi}_t + \sigma\left(s_t\right)\varepsilon_t = E_t\left[\widehat{\pi}_{t+1}\right].$$

Let $s_t \in \{1, 2\}$. Define $\pi_t^* = \pi_t$ and let $y_t = \pi_t^*$ and $x_t = \pi_t$. Then $y_{ss} = \pi$ and $x_{ss} = \pi$. Since the steady state is independent of all regime-switching parameters, we have

$$\theta\left(k, \chi\right) = \begin{bmatrix} \phi\left(k\right) \\ \sigma\left(k\right) \end{bmatrix}.$$

Now the equations can be written as

$$E_t\begin{bmatrix} \left(1 - \phi\left(s_t\right)\right)\pi + \phi\left(s_t\right)\pi_t - \pi_{t+1}^* - \sigma\left(s_t\right)\varepsilon_t \\ \pi_t^* - \pi_t \end{bmatrix} = 0.$$

The partition perturbation solution is

$$\widehat{\pi}_t = -\frac{\sigma\left(s_t\right)}{\phi\left(s_t\right)}\varepsilon_t,$$

which is exact and representative of a more general result for linear systems with regime-switching parameters. In contrast, doing perturbation "the old way" would deliver

$$\pi_t = -\frac{\overline{\sigma}}{\overline{\phi}}\varepsilon_t;$$

this solution is clearly not exact and therefore less accurate. If we set

$$p_{1,1} = 0.95$$
$$p_{2,2} = 0.85$$
$$\phi\left(1\right) = 1.25$$
$$\phi\left(2\right) = 0.96$$
$$\sigma\left(1\right) = 0.1$$
$$\sigma\left(2\right) = 0.6,$$

we find four solutions to the polynomial equations for the slope term, but only one is stable (in the sense of mean-square). The stable solution is

$$\widehat{\pi}_t = -0.08\varepsilon_t$$

in state 1 and

$$\widehat{\pi}_t = -0.625\varepsilon_t$$

in state 2. The wrong way gives us

$$\widehat{\pi}_t = -0.191083\varepsilon_t$$

in both states for the linear solution and

$$\widehat{\pi}_t = 0.0447610\varepsilon_t$$

$$\widehat{\pi}_t = -0.8986170\varepsilon_t$$

in the quadratic solution; clearly neither are close and the first regime even has the wrong sign.

Note that the Markov-switching probability can be endogenous.

8.17 Solving Linear Models by Time Iteration

Let $u_t = x_t - \overline{x}$ and obtain the linearized system around $\overline{u} = 0$

$$Au_{t-1} + Bu_t + Cu_{t+1} = 0;$$

any shocks have been eliminated. The solution takes the form

$$u_t = Fu_{t-1}.$$

We can solve this model by iterating in the time domain. That is, if we guess

$$u_{t+1} = F_0 u_t,$$

we have the equation

$$Au_{t-1} + Bu_t + CF_0 u_t$$

which can be solved for u_t. That is, guess F_0 then update using

$$F_{n+1} = (B + CF_n)^{-1}(-A).$$

If the eigenvalues of F are all smaller than one in modulus, this procedure converges. To check that there are no other stable solutions, iterate on

$$S_{n+1} = (B + AS_n)^{-1}(-C).$$

If $|\lambda_s| < 1$ then F is the unique solution.

We can also produce local solutions around other points. To do so, linearize around $\widetilde{u} \neq \overline{u}$ to obtain

$$Au_{t-1} + Bu_t + Cu_{t+1} + D = 0.$$

The solution now takes the form

$$u_t = E + Fu_{t-1}.$$

We can obtain this solution as

$$\begin{aligned} E_{n+1} &= (B + CF_n)^{-1}(-(D + CE_n)) \\ F_{n+1} &= (B + CF_n)^{-1}(-A). \end{aligned}$$

Example: growth model:

$$\begin{aligned} c_t^{-\sigma} &= \beta c_{t+1}^{-\sigma}\left(\alpha k_t^{\alpha-1} + 1 - \delta\right) \\ k_t &= k_{t-1}^{\alpha} + (1-\delta)k_{t-1} - c_t. \end{aligned}$$

The steady state is

$$\begin{aligned} \overline{k} &= \left(\frac{\alpha\beta}{1 - \beta(1-\delta)}\right)^{\frac{1}{1-\alpha}} \\ \overline{c} &= \overline{k}^{\alpha} - \delta\overline{k}. \end{aligned}$$

Linearizing around $\left(\overline{k}, \overline{c}\right)$ yields

$$\begin{aligned} -\sigma\overline{c}^{-\sigma-1}(c_t - \overline{c}) + \sigma\beta\overline{c}^{-\sigma-1}\left(\alpha\overline{k}^{\alpha-1} + 1 - \delta\right)(c_{t+1} - \overline{c}) - \beta\overline{c}^{-\sigma}\alpha(\alpha-1)\overline{k}^{\alpha-2}\left(k_t - \overline{k}\right) = 0 \\ k_t - \overline{k} - \alpha\overline{k}^{\alpha-1}\left(k_{t-1} - \overline{k}\right) - (1-\delta)\left(k_{t-1} - \overline{k}\right) + (c_t - \overline{c}) = 0 \end{aligned}$$

Therefore, we have

$$\begin{bmatrix} 0 & 0 \\ -\frac{1}{\beta} & 0 \end{bmatrix}\begin{bmatrix} k_{t-1} - \overline{k} \\ c_{t-1} - \overline{c} \end{bmatrix} + \begin{bmatrix} -\beta\overline{c}\alpha(\alpha-1)\overline{k}^{\alpha-2} & -\sigma \\ 1 & 1 \end{bmatrix}\begin{bmatrix} k_t - \overline{k} \\ c_t - \overline{c} \end{bmatrix} + \begin{bmatrix} 0 & \sigma \\ 0 & 0 \end{bmatrix}\begin{bmatrix} k_{t+1} - \overline{k} \\ c_{t+1} - \overline{c} \end{bmatrix} = 0.$$

Then

$$A\left(\overline{k},\overline{c}\right)=\left[\begin{array}{cc} 0 & 0 \\ -\frac{1}{\beta} & 0 \end{array}\right]$$

$$B\left(\overline{k},\overline{c}\right)=\left[\begin{array}{cc} -\beta\overline{c}\alpha\left(\alpha-1\right)\overline{k}^{\alpha-2} & -\sigma \\ 1 & 1 \end{array}\right]$$

$$C\left(\overline{k},\overline{c}\right)=\left[\begin{array}{cc} 0 & \sigma \\ 0 & 0 \end{array}\right].$$

Linearizing around (k,c) yields

$$-\sigma\overline{c}^{-\sigma-1}\left(c_t-\overline{c}+\overline{c}-c\right)+\sigma\beta c^{-\sigma-1}\left(\alpha k^{\alpha-1}+1-\delta\right)\left(c_{t+1}-\overline{c}+\overline{c}-c\right)-\beta c^{-\sigma}\alpha\left(\alpha-1\right)k^{\alpha-2}\left(k_t-\overline{k}+\overline{k}-k\right)=0$$

$$k_t-\overline{k}+\overline{k}-k-\alpha k^{\alpha-1}\left(k_{t-1}-\overline{k}+\overline{k}-k\right)-\left(1-\delta\right)\left(k_{t-1}-\overline{k}+\overline{k}-k\right)+\left(c_t-\overline{c}+\overline{c}-c\right)=0$$

so we have

$$\left[\begin{array}{cc} 0 & 0 \\ -\alpha k^{\alpha-1}-(1-\delta) & 0 \end{array}\right]\left[\begin{array}{c} k_{t-1}-\overline{k} \\ c_{t-1}-\overline{c} \end{array}\right]+\left[\begin{array}{cc} \beta c\alpha\left(\alpha-1\right)k^{\alpha-2} & -\sigma \\ 1 & 1 \end{array}\right]\left[\begin{array}{c} k_t-\overline{k} \\ c_t-\overline{c} \end{array}\right]+$$

$$\left[\begin{array}{cc} 0 & \sigma\beta\left(\alpha k^{\alpha-1}+1-\delta\right) \\ 0 & 0 \end{array}\right]\left[\begin{array}{c} k_{t+1}-\overline{k} \\ c_{t+1}-\overline{c} \end{array}\right]+$$

$$\left[\begin{array}{c} -\sigma\left(\overline{c}-c\right)+\sigma\beta\left(\alpha k^{\alpha-1}+1-\delta\right)\left(\overline{c}-c\right)-\beta c\alpha\left(\alpha-1\right)k^{\alpha-2}\left(\overline{k}-k\right) \\ \overline{k}-k+\overline{c}-c-\alpha k^{\alpha-1}\left(\overline{k}-k\right)-\left(1-\delta\right)\left(\overline{k}-k\right) \end{array}\right]$$

$$=0.$$

That is,

$$A\left(k,c\right)=\left[\begin{array}{cc} 0 & 0 \\ -\alpha k^{\alpha-1}-(1-\delta) & 0 \end{array}\right]$$

$$B\left(k,c\right)=\left[\begin{array}{cc} \beta c\alpha\left(\alpha-1\right)k^{\alpha-2} & -\sigma \\ 1 & 1 \end{array}\right]$$

$$C\left(k,c\right)=\left[\begin{array}{cc} 0 & \sigma\beta\left(\alpha k^{\alpha-1}+1-\delta\right) \\ 0 & 0 \end{array}\right]$$

$$D\left(k,c\right)=\left[\begin{array}{c} -\sigma\left(\overline{c}-c\right)+\sigma\beta\left(\alpha k^{\alpha-1}+1-\delta\right)\left(\overline{c}-c\right)-\beta c\alpha\left(\alpha-1\right)k^{\alpha-2}\left(\overline{k}-k\right) \\ \overline{k}-k+\overline{c}-c-\alpha k^{\alpha-1}\left(\overline{k}-k\right)-\left(1-\delta\right)\left(\overline{k}-k\right) \end{array}\right].$$

This approach is generally going to be very fast compared to even the QZ decomposition, especially if we are smart about the matrix inversion step (one possibility might be to use an approximation that converges to the inverse of that matrix, similar to Broyden's method for solving nonlinear equations).

9 Linear Quadratic Gaussian Regulator Problem

An alternative to locally approximating the equilibrium conditions is to locally approximate the underlying optimization problem; one tractable method is to turn the optimization into a Linear-Quadratic-Gaussian Regulator problem. Imagine first that we have a problem of the following form:

$$\max_{\{a_t,x_t\}_{t=0}^{\infty}} \left\{ \sum_{t=0}^{\infty} \beta^t \left(x_t^T Q x_t + a_t^T R a_t + 2 a_t^T W x_t \right) \right\} \tag{53}$$

where x_t is a vector of states and a_t is a vector of controls (actions). I will assume that there are n states where the first one is a constant and there are k controls – the matrices are then dimensioned as $Q_{n\times n}$, $R_{k\times k}$, and $W_{n\times k}$. States evolve according to the transition function

$$x_{t+1} = Ax_t + Ba_t. \tag{54}$$

Basically, we are maximizing a quadratic objective with linear constraints. The Bellman equation is

$$x_t^T P x_t = \max_{a_t} \left\{ x_t^T Q x_t + a_t^T R a_t + 2 a_t^T W x_t + \beta x_{t+1}^T P x_{t+1} \right\} \tag{55}$$

subject to

$$x_{t+1} = Ax_t + Ba_t. \tag{56}$$

I have used the fact that the value function will be quadratic; it is trivial to show this fact holds. The value function, $x^T P x$, is symmetric (the P matrix is symmetric; from now on I will abuse notation a bit and refer to P as the value function). Substituting the constraint into the right-hand-side of the Bellman equation yields

$$x^T Q x + a^T R a + 2 a^T W x + \beta \left(x^T A^T + a^T B^T \right) P \left(Ax + Ba \right). \tag{57}$$

To take a first-order condition we need to use the matrix differentiation rules

$$D_x\left(y^T Ax\right) = A^T y$$
$$D_y\left(y^T Ax\right) = Ax.$$

Maximizing with respect to a yields the solution

$$a = -(R + \beta B^T PB)^{-1}(W + \beta B^T PA)x. \tag{58}$$

Substituting this into the Bellman equation and matching coefficients yields the recursive relationship known as the **matrix Ricatti equation**:

$$x^T Px = x^T(Q + \beta A^T PA - (W^T + \beta A^T PB)(R + \beta B^T PB)^{-1}(W + \beta B^T PA))x. \tag{59}$$

This equation can be used to solve for the value function by iteration. Think of the RHS as the result of the Bellman operator; it takes the function characterized by P and maps it into one characterized by

$$Q + \beta A^T PA - (W^T + \beta A^T PB)(R + \beta B^T PB)^{-1}(W + \beta B^T PA). \tag{60}$$

One way to solve this equation for P is to iterate: start with $P_0 = 0$, calculate a new P_1 by

$$P_1 = Q + \beta A^T P_0 A - (W^T + \beta A^T P_0 B)(R + \beta B^T P_0 B)^{-1}(W + \beta B^T P_0 A) \tag{61}$$

and repeat until the P are sufficiently close together. The Contraction Mapping Theorem guarantees that only one matrix P can solve the matrix Ricatti equation. Note: there are subspace methods that can solve the Ricatti equation more quickly.

The optimal decision rules are then recovered by

$$\begin{aligned} a^* &= -Fx \\ &= -(R + \beta B^T PB)^{-1}(W + \beta B^T PA)x. \end{aligned}$$

Pseudo-code for this method is then

1. Set initial P_0 (a good choice is either the zero matrix or a matrix which is a small negative number times an identity matrix);

2. Compute P_1 according to the Ricatti equation;

3. Compute the norm of $P_1 - P_0$ using the built-in 'norm' function;

4. Set up a 'while' loop: while norm$(P_1 - P_0) > \epsilon$ set $P_0 = P_1$ and recompute P_1.

The while loop will continue until the difference between the two matrices is small. After the loop terminates you can compute F.

Some minor points are of some importance here. We require that the value function be concave; this condition was needed to ensure differentiability and many other desirable features such as the transversality condition. In the quadratic case, we need P to be negative definite – it must possess only negative eigenvalues (except for one related to the constant, which typically will be large and positive). A matrix A is called **negative definite** if and only if $x^T Ax < 0$ $\forall x \neq 0$. The eigenvalue condition is necessary and sufficient for this condition. We also want our system to be saddle-stable; this will require that the matrix $A - BF$ have eigenvalues within the unit circle since

$$\begin{aligned} x_{t+1} &= Ax_t + Ba_t^* \\ &= (A - BF)x_t. \end{aligned}$$

Thus, iterating on the decision rule produces a sequence that converges to the steady state, so transversality is satisfied.

A more general quadratic environment would be the Bellman equation

$$\begin{bmatrix} z^T & s^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} z \\ s \end{bmatrix} = \max_d \left\{ \begin{array}{c} \begin{bmatrix} z^T & s^T & d^T \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \end{bmatrix} \begin{bmatrix} z \\ s \\ d \end{bmatrix} + \\ \beta \begin{bmatrix} (z')^T & (s')^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} z' \\ s' \end{bmatrix} \end{array} \right\}$$

subject to

$$\begin{aligned} z' &= Az \\ s' &= B_1 z + B_2 s + B_3 d. \end{aligned}$$

Constants are absorbed into the exogenous state vector z.

The first-order condition for this problem is

$$Q_{13}z + Q_{23}s + Q_{33}d + \beta B_3^T P_{12} A z + \beta B_3^T P_{22} B_1 z + \beta B_3^T P_{22} B_2 s + \beta B_3^T P_{22} B_3 d = 0$$

(where we exploit the symmetry of the Q and P matrices), which can be rearranged to obtain

$$\begin{aligned} d^* = &- \left(Q_{33} + \beta B_3^T P_{22} B_3\right)^{-1} \left(Q_{13} + \beta B_3^T P_{12} A + \beta B_3^T P_{22} B_1\right) z - \\ &\left(Q_{33} + \beta B_3^T P_{22} B_3\right)^{-1} \left(Q_{13} + \beta B_3^T P_{22} B_2\right) s \\ &= D_1 z + D_2 s. \end{aligned}$$

Construct the matrix that describes the state evolution at the optimal control as

$$\begin{aligned} \begin{bmatrix} z' \\ s' \end{bmatrix} &= \begin{bmatrix} A & 0 \\ B_1 + B_3 D_1 & B_2 + B_3 D_2 \end{bmatrix} \begin{bmatrix} z \\ s \end{bmatrix} \\ &= \begin{bmatrix} A & 0 \\ F_1 & F_2 \end{bmatrix} \begin{bmatrix} z \\ s \end{bmatrix} \end{aligned}$$

and then make substitutions to obtain

$$\begin{aligned} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} = &\begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} + \begin{bmatrix} Q_{13} \\ Q_{23} \end{bmatrix} \begin{bmatrix} D_1 & D_2 \end{bmatrix} + \\ &\begin{bmatrix} D_1^T \\ D_2^T \end{bmatrix} \begin{bmatrix} Q_{31} & Q_{32} \end{bmatrix} + \begin{bmatrix} D_1^T \\ D_2^T \end{bmatrix} Q_{33} \begin{bmatrix} D_1 & D_2 \end{bmatrix} + \\ &\beta \begin{bmatrix} A^T & F_1^T \\ 0 & F_2^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} A & 0 \\ F_1 & F_2 \end{bmatrix}. \end{aligned}$$

We can iterate on this equation until convergence, which is guaranteed by the Contraction Mapping Theorem from any (negative definite) initial guess for P. While we technically do not need to make the initial guess negative definite, it will create problems with the first-order condition if we do not.

In general the growth model is not quadratic. Therefore, now we will examine how to approximate it locally around the steady state as an LQ regulator problem – this approach was used by Kydland and Prescott (1982) in their time-to-build paper and first discussed in the appendix to their rules vs. discretion (Kydland and Prescott 1977) paper (although their actual

procedure is slightly different, in that they identify the coefficients of the LQ approximation not through a Taylor expansion but rather by interpolating the function through three points; the results are indistinguishable from a Taylor expansion, as shown in Christiano 1987). The growth model has a Bellman equation like this:

$$v\left(k\right)=\max_{k'}\{u(f(k)+(1-\delta)k-k')+\beta v(k')\}. \tag{62}$$

Let's take the steady state of this model to be k^*. We know that the model is saddle-stable, at least locally, around this point – it will tend to this value over time. Therefore, what if we choose to approximate the utility function around this point by a quadratic? We know that this steady state is given by the solution to

$$Df\left(k^*\right)+1-\delta=1/\beta. \tag{63}$$

From Taylor's theorem we have

$$\begin{aligned}u(f(k)+(1-\delta)k-k')=u\left(f\left(k^*\right)-\delta k^*\right)+\\ Du(f\left(k^*\right)-\delta k^*)(Df(k^*)+1-\delta)(k-k^*)-Du(f\left(k^*\right)-\delta k^*)(k'-k^*)+\\ \frac{1}{2}\left(D^2u\left(f\left(k^*\right)-\delta k^*\right)\left(Df\left(k^*\right)+1-\delta\right)^2+Du\left(f\left(k^*\right)-\delta k^*\right)D^2f\left(k^*\right)\right)\left(k-k^*\right)^2-\\ D^2u\left(f\left(k\right)+\left(1-\delta\right)k-k'\right)\left(Df\left(k\right)+1-\delta\right)\left(k-k^*\right)\left(k'-k^*\right)+\\ \frac{1}{2}D^2u\left(f\left(k^*\right)-\delta k^*\right)\left(k'-k^*\right)^2.\end{aligned}$$

We can use either numerical or symbolic derivatives to compute these terms for more complicated models (numerical derivatives work just fine, so I tend to use them).

Our goal is to approximate the return function

$$u(f(k)+(1-\delta)k-k')\equiv H(y) \tag{64}$$

where

$$y=\begin{bmatrix}k\\k'\end{bmatrix}=\begin{bmatrix}x\\a\end{bmatrix} \tag{65}$$

where we do not augment x with a constant. Let the steady state be denoted y^*. Then the approximation around the steady state yields

$$H(y)=H(y^*)+f^T(y-y^*)+(y-y^*)^T\frac{1}{2}V(y-y^*). \tag{66}$$

Now define

$$\frac{1}{2}V = \begin{bmatrix} S & T^T \\ T & R \end{bmatrix} \tag{67}$$

$$l_k^T = \frac{1}{2}[f_k^T - 2\left(x^*\right)^T T^T - 2\left(a^*\right)^T R] \tag{68}$$

$$l_n^T = \frac{1}{2}[f_n^T - 2\left(x^*\right)^T S - 2\left(a^*\right)^T T] \tag{69}$$

$$G = H(y^*) + \left(y^*\right)^T \frac{1}{2} V y^* - f^T y^* \tag{70}$$

where

$$f = \begin{bmatrix} f_n \\ f_k \end{bmatrix} \tag{71}$$

where f_n refers to the first derivatives with respect to states and f_k to controls. Now define

$$W = \begin{bmatrix} l_k & T \end{bmatrix} \tag{72}$$

and

$$Q = \begin{bmatrix} G & l_n^T \\ l_n & S \end{bmatrix}. \tag{73}$$

From here we can use the Ricatti equation above to find P and F. Q now embeds the constant term G as well as the linear terms associated with states times the constant.

We can produce a log-quadratic-linear approximation by replacing each variable by its "exponential" and then computing the same expressions – thus, instead of x we have $\exp(x)$ and instead of a we have $\exp(a)$. Then x and a are interpreted to be logs.

Adding uncertainty in this environment is very simple, because it satisfies something called certainty equivalence. Imagine the following two problems:

$$\max_a \{E[f(x,a)]\} \tag{1}$$

where x is a random variable and

$$\max_a \{f(E[x],a)\} \tag{2}$$

where all the randomness is gone. Under what conditions are these two problems equivalent? By equivalent, I mean that they generate the same α; the value functions for the two problems will differ by a constant. We state and prove the following theorem due to Simon (1948):

Theorem 90 *(**Certainty Equivalence Theorem**) If f is quadratic in x and a, these two problems will have the same a^* where*

$$a^* = \underset{a}{\operatorname{argmax}} \{E[f(x,a)]\} = \underset{a}{\operatorname{argmax}} \{f(E(x),a)\}.$$

Proof. For problem (1) the first-order conditions are

$$\frac{\partial}{\partial \alpha} E[f(x,a)] = 0. \tag{74}$$

Assume that

$$f(x,a) = A + Bx + Ca + Dx^2 + Fa^2 + Gxa. \tag{75}$$

Then the first-order condition implies

$$E[c + 2Fa + Gx] = 0 \Rightarrow c + 2Fa + GE(x) = 0. \tag{76}$$

For problem (2) the first-order conditions are

$$\frac{\partial}{\partial x} f(E(x),a) = 0 \Rightarrow c + 2Fa + GE(x) = 0. \tag{77}$$

Thus, the problems are equivalent. ■

We can use the certainty equivalence principle to derive the Ricatti equation for a quadratic problem with uncertainty. The Bellman equation is

$$v(x_t) = \max_{a_t} \left\{ x_t^T Q x_t + a_t^T R a_t + 2 a_t^T W x_t + \beta E_t[v(x_{t+1})|x_t] \right\} \tag{78}$$

subject to the transition equation

$$x_{t+1} = A x_t + B a_t + \epsilon_{t+1} \tag{79}$$

where $\epsilon_t \sim N(0,\Sigma)$ is a vector of random variables. The form of the value function is now

$$x_t^T P x_t + d \tag{80}$$

where d is a constant. Substituting into the right-hand-side of the Bellman equation and taking expectations yields

$$\begin{aligned} E_t(x_{t+1}^T P x_{t+1} + d) &= E_t\left[(Ax_t + Ba_t + \epsilon_{t+1})^T P (Ax_t + Ba_t + \epsilon_{t+1}) + d\right] \\ &= E_t[\cdot] + E_t\left[a_t B^T P \epsilon_{t+1}\right] + E_t\left[\epsilon_t^T P A x_t\right] + E_t\left[\epsilon_t^T P B a_t\right] + \\ &\quad E_t\left[x_{t+1}^T A^T P \epsilon_{t+1}\right] + E_t\left[\epsilon_{t+1}^T P \epsilon_{t+1}\right] + d \end{aligned}$$

where $[\cdot]$ consists of terms that do not involve ϵ_{t+1}. This implies

$$E_t(x_{t+1}^T P x_{t+1}) = [\cdot] + E_t\left[\epsilon_{t+1}^T P \epsilon_{t+1}\right]. \tag{81}$$

Clearly we will have the same decision rules as before. The matrix Ricatti equation is then identical to the one before. Matching up constants yields

$$d = \beta d + \beta E_t\left[\epsilon_{t+1}^T P \epsilon_{t+1}\right] \Rightarrow d = \frac{\beta}{1-\beta}\,\text{trace}(P\Sigma). \tag{82}$$

If P is negative definite this term is negative; uncertainty about the future lowers lifetime utility – note however that if some states are exogenous, then P may not be negative definite. What we have done here is shown that the problem "separates" forecasting – the forming of conditional expectations – and control – the optimal decisions: we forecast first and then make decisions based on the conditional expected value of future variables, without taking into account the uncertainty that surrounds those forecasts.

In the previous section where we used local approximations on first-order conditions, we could break the certainty equivalence property by using higher-order approximations. That approach does not work here, unfortunately, because the value function of a regulator problem with either cubic returns or quadratic constraints is not cubic; it is not of known form, in fact, so that approach is not productive. However, it is possible to extend the linear quadratic regulator problem to break the certainty equivalence property. Hansen and Sargent (1995) formulate a model with "risk sensitivity" which alters the expectations of the household in a way that lets Σ, the variance-covariance matrix of the stochastic terms, affect decisions but keeps the problem quadratic. The distortion takes the form of an operator D that alters P:

$$D(P) = P - \sigma P\Sigma\left(I + \sigma\Sigma P\Sigma\right)^{-1}\Sigma P \tag{83}$$

where σ is a preference parameter with the property that if σ increases the risk aversion of the agent with respect to gambles over future wealth also increases. Note that if $\sigma = 0$ (agents obey certainty equivalence) or $\Sigma = 0$ (the model is deterministic) this expression collapses to P. Assuming that the return function is quadratic the Bellman equation is

$$V(x) = \max_a \left\{ x^T Q x + a^T R a + 2x^T W a + \beta\left(\frac{2}{\sigma}\right)\log\left(E_t\left[\exp\left(-\frac{\sigma}{2}V(x')\right)\middle| x\right]\right)\right\};$$

this equation is a particular form of Epstein-Zin preferences which we will discuss later.

The Bellman equation can be computed by evaluating the expectation under the assumption of normal errors and linear state transition equations:

$$x^T Px + d = \max_a \left\{ x^T Qx + a^T Ra + 2x^T Wa + \beta \left(x^T A^T + a^T B^T \right) D\left(P\right)\left(Ax + Ba\right) + U\left(P, d\right) \right\} \tag{84}$$

where U is some function that does not involve a. This result comes from exploiting the moment generating function of a normal:

$$E\left[\exp\left(tx\right)\right] = \exp\left(\mu t + \frac{1}{2} t^2 \sigma^2\right).$$

Since x' is normal if ϵ' is normal, it has moments

$$\begin{aligned} E\left[x' | x, a\right] &= Ax + Ba \\ E\left[\left(x' - E\left[x' | x, a\right]\right)^2\right] &= \Sigma. \end{aligned}$$

The optimal control is

$$a = -(R + \beta B^T D\left(P\right) B)^{-1} (W + \beta B^T D\left(P\right) A) x. \tag{85}$$

Substitution back into the Bellman equation yields the Ricatti equation

$$x^T Px + d = x^T (Q + \beta A^T D\left(P\right) A - (W^T + \beta A^T D\left(P\right) B)(R + \beta B^T D\left(P\right) B)^{-1} (W + \beta B^T D\left(P\right) A)) x + U\left(P, d\right). \tag{86}$$

This expression again defines a Contraction Mapping so iterations converge to a unique P. Since Σ appears in the expression for $D\left(P\right)$, agents in this economy change the coefficients of their linear decision rules in response to different variance-covariance matrices for the shocks.

Since the economy no longer displays certainty equivalence, the deterministic and stochastic steady states are no longer the same. As a result, the decision rules, if centered at the deterministic steady state, may be inaccurate around the average values, particularly for large σ and Σ. Centering the approximation then involves solving a single equation that is, heuristically at least, defined by guessing a stochastic steady state value $\overline{x}$, computing the decision rule matrix $-F$, computing the fixed point of the transition equation implied by this rule

$$x^* = \left(A - BF\left(\overline{x}\right)\right) x^*$$

(which can be done via iteration or eigenvalue methods, since x^* is just the eigenvector associated with the unit eigenvalue of $A - BF$), and then checking to see if $x^* = \overline{x}$. If it does, the approximation is centered around the average value in the economy. If not, solve this equation for $\overline{x}$ and use it. Tallarini (2000) uses a fixed point algorithm to solve this equation, but treating it as a nonlinear function of one variable and solving it using a standard nonlinear equation solver will generally get the answer more quickly. Note that this is similar to perturbing around the risky steady state.

9.1 Eliminating Cross-Terms and Discounting

Suppose we have the problem

$$\sum_{t=0}^{\infty} \beta^t \left(x_t^T Q^* x_t + u_t^T R u_t + 2 u_t^T W x_t \right) = \sum_{t=0}^{\infty} \beta^t \left[\begin{array}{cc} x_t^T & u_t^T \end{array} \right] \left[\begin{array}{cc} Q^* & W^T \\ W & R \end{array} \right] \left[\begin{array}{c} x_t \\ u_t \end{array} \right]$$

subject to

$$x_{t+1} = A^* x_t + B u_t.$$

The problem can be simplified by eliminating the cross-terms. Define

$$a_t = u_t + R^{-1} W x_t.$$

Then

$$a_t^T R a_t = \left[\begin{array}{cc} x_t^T & u_t^T \end{array} \right] \left[\begin{array}{cc} W^T R^{-1} W & W^T \\ W & R \end{array} \right] \left[\begin{array}{c} x_t \\ u_t \end{array} \right]$$

and

$$\left[\begin{array}{cc} x_t^T & u_t^T \end{array} \right] \left[\begin{array}{cc} Q^* & W^T \\ W & R \end{array} \right] \left[\begin{array}{c} x_t \\ u_t \end{array} \right] = x_t^T Q x_t + a_t^T R a_t$$

where

$$Q = Q^* - W^T R^{-1} W$$

and also

$$\begin{aligned} x_{t+1} &= A^* x_t + B u_t \\ &= A^* x_t + B a_t - B R^{-1} W x_t \\ &= A x_t + B a_t. \end{aligned}$$

Now we have the standard LQ regulator problem

$$\sum_{t=0}^{\infty} \beta^t \left(x_t^T Q x_t + a_t^T R a_t \right)$$

subject to

$$x_{t+1} = A x_t + B a_t.$$

We can also eliminate discounting. Define

$$\widetilde{x}_t = \sqrt{\beta} x_t$$
$$\widetilde{a}_t = \sqrt{\beta} a_t.$$

Then we have

$$\sum_{t=0}^{\infty} \beta^t \left(x_t^T Q x_t + a_t^T R a_t \right) = \sum_{t=0}^{\infty} \left(\widetilde{x}_t^T Q \widetilde{x}_t + \widetilde{a}_t^T R \widetilde{a}_t \right)$$

subject to

$$\widetilde{x}_{t+1} = \widetilde{A} \widetilde{x}_t + \widetilde{B} \widetilde{a}_t$$

where

$$\widetilde{A} = A\sqrt{\beta}$$
$$\widetilde{B} = B\sqrt{\beta}.$$

The solution is

$$\widetilde{u}_t = -\widetilde{F} \widetilde{x}_t$$

where

$$\widetilde{F} = -\left(R + \widetilde{B}^T \widetilde{P} \widetilde{B} \right)^{-1} \widetilde{B}^T \widetilde{P} \widetilde{A}$$
$$\widetilde{P} = Q + \widetilde{A}^T \widetilde{P} \widetilde{A} - \widetilde{A}^T \widetilde{P} \widetilde{B} \left(R + \widetilde{B}^T \widetilde{P} \widetilde{B} \right)^{-1} \widetilde{B}^T \widetilde{P} \widetilde{A}$$

and

$$x_{t+1} = \left(A - B\widetilde{F} \right) x_t.$$

Given the speed of current computers, these modifications no longer seem particularly important, but many problems in the engineering literature are formulated this way so it may be convenient if you are trying to adapt some results.

9.2 Linear-Quadratic Competitive Equilibrium

Imagine the generic competitive equilibrium growth model formulation

$$v(k,K) = \max_{k'} \left\{ u(r(K)k + w(K) - k') + \beta v(k', K') \right\} \tag{87}$$

subject to the law of motion for K:

$$K' = G(K). \tag{88}$$

From the firm's problem we get the conditions

$$r(K) = MPK \tag{89}$$

and

$$w(K) = f(K) - MPK * K \tag{90}$$

(check this for yourself). The following algorithm was suggested by Kydland (1989) to solve these problems:

1. Guess an initial value function $v^0(k,K)$.
2. Insert the factor conditions into the consumer's problem.
3. Obtain the first-order condition for the RHS of the Bellman equation and solve for the decision rule $k' = g(k,K)$.
4. Aggregate the first-order condition to obtain $K' = g(K,K) \equiv G(K)$.
5. Update the guess for the value function by

$$v^1(k,K) = u(r(K)k + w(K) - g(k,K)) + \beta v^0(g(k,K), G(K)). \tag{91}$$

6. Repeat until the value function has converged.

Note that we do not have to guess the form $K = G(K)$; we obtain it at each step as the aggregation of the individual decision rule $g(k,K)$. This algorithm works very nicely in a linear-quadratic environment where all decision rules and transition equations are linear. We will

examine this case explicitly as it is instructive on how to solve these problems. The linear-quadratic problem faced by the consumer is

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} P \begin{bmatrix} z \\ S \\ s \end{bmatrix} = \max_d \left\{ \begin{bmatrix} z^T & S^T & s^T & D^T & d^T \end{bmatrix} Q \begin{bmatrix} z \\ S \\ s \\ D \\ d \end{bmatrix} + \beta \begin{bmatrix} (z')^T & (S')^T & (s')^T \end{bmatrix} P \begin{bmatrix} z' \\ S' \\ s' \end{bmatrix} \right\} \tag{92}$$

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \tag{93}$$

$$Q = \begin{bmatrix} Q_{11} & \cdots & Q_{15} \\ \vdots & \ddots & \vdots \\ Q_{51} & \cdots & Q_{55} \end{bmatrix} \tag{94}$$

subject to the transition equations

$$z' = Az \tag{95}$$

$$s' = B_1 z + B_2 S + B_3 s + B_4 D + B_5 d \tag{96}$$

and

$$S' = C_1 z + C_2 S + C_3 D. \tag{97}$$

In the notation used here, z are exogenous variables, S is the vector of aggregate state variables, s is the individual vector of state variables, d is the individual vector of choice variables, and D is the aggregate version of d. Equilibrium for this model demands that

$$B_1 z + (B_2 + B_3) S + (B_4 + B_5) D = C_1 z + C_2 S + C_3 D. \tag{98}$$

The first-order condition for the household's choice of d is

$$\begin{aligned} d^* = & - \left[Q_{55} + \beta B_5^T P_{33} B_5 \right]^{-1} \left[Q_{51} + \beta B_5^T P_{31} A + \beta B_5^T P_{32} C_1 + \beta B_5^T P_{33} B_1 \right] z \\ & - \left[Q_{55} + \beta B_5^T P_{33} B_5 \right]^{-1} \left[Q_{52} + \beta B_5^T P_{32} C_2 + \beta B_5^T P_{33} B_2 \right] S \\ & - \left[Q_{55} + \beta B_5^T P_{33} B_5 \right]^{-1} \left[Q_{53} + \beta B_5^T P_{33} B_3 \right] s \\ & - \left[Q_{55} + \beta B_5^T P_{33} B_5 \right]^{-1} \left[Q_{54} + \beta B_5^T P_{32} C_3 + \beta B_5^T P_{33} B_4 \right] D. \end{aligned}$$

We can rewrite this more compactly as

$$d^* = F_1 z + F_2 S + F_3 s + F_4 D. \tag{99}$$

We now impose that $s = S$ and $d = D$ to obtain the equilibrium expression

$$\begin{aligned} D^* &= (I - F_4)^{-1} F_1 z + (I - F_5)^{-1}(F_2 + F_3) S \\ &= R_1 z + R_2 S. \end{aligned}$$

From here we have that

$$\begin{aligned} d^* &= (F_1 + F_4 R_1) z + (F_2 + F_4 R_2) S + F_3 s \\ &= J_1 z + J_2 S + J_3 s. \end{aligned}$$

Also, we can insert this rules into the transition equations to obtain

$$\begin{aligned} S' &= (C_1 + C_3 R_1) z + (C_2 + C_3 R_2) S \\ &= W_1 z + W_2 S \end{aligned}$$

and

$$\begin{aligned} s' &= (B_1 + B_4 R_1 + B_5 J_1) z + (B_2 + B_4 R_2 + B_5 J_2) S + (B_3 + B_5 J_3) s \\ &= X_1 z + X_2 S + X_3 s. \end{aligned}$$

These decision rules are inserted into the Bellman equation to obtain the recursion

$$\begin{bmatrix} P_{11}^{n+1} & P_{12}^{n+1} & P_{13}^{n+1} \\ P_{21}^{n+1} & P_{22}^{n+1} & P_{23}^{n+1} \\ P_{31}^{n+1} & P_{32}^{n+1} & P_{33}^{n+1} \end{bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} \\ Q_{21} & Q_{22} & Q_{23} \\ Q_{31} & Q_{32} & Q_{33} \end{bmatrix} + \begin{bmatrix} Q_{14} & Q_{15} \\ Q_{24} & Q_{25} \\ Q_{34} & Q_{35} \end{bmatrix} \begin{bmatrix} R_1 & R_2 & 0 \\ J_1 & J_2 & J_3 \end{bmatrix} +$$
$$\begin{bmatrix} R_1^T & J_1^T \\ R_2^T & J_2^T \\ 0 & J_3^T \end{bmatrix} \begin{bmatrix} Q_{41} & Q_{42} & Q_{43} \\ Q_{51} & Q_{52} & Q_{53} \end{bmatrix} +$$
$$\begin{bmatrix} R_1^T & J_1^T \\ R_2^T & J_2^T \\ 0 & J_3^T \end{bmatrix} \begin{bmatrix} Q_{44} & Q_{45} \\ Q_{54} & Q_{55} \end{bmatrix} \begin{bmatrix} R_1 & R_2 & 0 \\ J_1 & J_2 & J_3 \end{bmatrix} +$$
$$\beta \begin{bmatrix} A^T & W_1^T & X_1^T \\ 0 & W_2^T & X_2^T \\ 0 & 0 & X_3^T \end{bmatrix} \begin{bmatrix} P_{11}^n & P_{12}^n & P_{13}^n \\ P_{21}^n & P_{22}^n & P_{23}^n \\ P_{31}^n & P_{32}^n & P_{33}^n \end{bmatrix} \begin{bmatrix} A & 0 & 0 \\ W_1 & W_2 & 0 \\ X_1 & X_2 & X_3 \end{bmatrix}.$$

We iterate on this equation until it converges as before – this is the "distorted Ricatti equation." Unlike the iterative version of the planning problem, there is no guarantee that this process will converge, but it does seem to work remarkably often. Some proofs of convergence are available for special cases (Coleman 1989,1997 and Greenwood and Huffman 1995).

However, because there is no guarantee that this procedure will converge, it is useful to have an alternative approach – this one is provided by Ceria and Ríos-Rull (1994). Denote by $Q_{s',\cdot}$ the submatrix of the quadratic approximation matrix Q that contains row s'. The Euler equation for the approximate quadratic problem can then be written

$$Q_{s',\cdot} \begin{bmatrix} z \\ S \\ s \\ S' \\ s' \end{bmatrix} + \beta Q_{s,\cdot} \begin{bmatrix} Az \\ S' \\ s' \\ S'' \\ s'' \end{bmatrix}.$$

The solutions for D and d then must satisfy

$$Q_{s',\cdot}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ C_1 & C_2 & 0 \\ B_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} z \\ S \\ s \end{bmatrix} + \beta Q_{s,\cdot}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ D_1 & D_2 & 0 \\ d_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} A & 0 & 0 \\ D_1 & D_2 & 0 \\ d_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} z \\ S \\ s \end{bmatrix} = 0.$$

We can rewrite this expression as

$$\begin{bmatrix} R_{s',z} & R_{s',S} & R_{s',s} \end{bmatrix}\begin{bmatrix} z \\ S \\ s \end{bmatrix} + \begin{bmatrix} R_{s',S'} & R_{s',s'} \end{bmatrix}\begin{bmatrix} D_1 & D_2 & 0 \\ d_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\beta\begin{bmatrix} R_{s,z} & R_{s,S} & R_{s,s} \end{bmatrix}\begin{bmatrix} A & 0 & 0 \\ D_1 & D_2 & 0 \\ d_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\beta\begin{bmatrix} R_{s,S'} & R_{s,s'} \end{bmatrix}\begin{bmatrix} D_1 & D_2 & 0 \\ d_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} A & 0 & 0 \\ D_1 & D_2 & 0 \\ d_1 & d_2 & d_3 \end{bmatrix}\begin{bmatrix} z \\ S \\ s \end{bmatrix}$$

$$= 0.$$

Using the equilibrium condition that aggregate and individual coincide we can obtain

$$\begin{bmatrix} R_{s',z} & R_{s',S} + R_{s',s} \end{bmatrix}\begin{bmatrix} z \\ S \end{bmatrix} + \left[R_{s',S'} + R_{s',s'}\right]\begin{bmatrix} D_1 & D_2 \end{bmatrix}\begin{bmatrix} z \\ S \end{bmatrix} +$$

$$\beta\begin{bmatrix} R_{s,z} & R_{s,S} + R_{s,s} \end{bmatrix}\begin{bmatrix} A & 0 \\ D_1 & D_2 \end{bmatrix}\begin{bmatrix} z \\ S \end{bmatrix} +$$

$$\beta\left[R_{s,S'} + R_{s,s'}\right]\begin{bmatrix} D_1 & D_2 \end{bmatrix}\begin{bmatrix} A & 0 \\ D_1 & D_2 \end{bmatrix}\begin{bmatrix} z \\ S \end{bmatrix}$$

$$= 0.$$

Since this must hold for all (z, S), the following two equations must be satisfied:

$$\left(R_{s',S} + R_{s',s}\right) + \left[\left(R_{s',S'} + R_{s',s'}\right) + \beta\left(R_{s,S} + R_{s,s}\right)\right] D_2 + \beta\left(R_{s,S'} + R_{s,s'}\right) D_2 D_2 = 0$$

$$\left[\left(R_{s',S'} + R_{s',s'}\right) + \beta\left[\left(R_{s,S} + R_{s,s}\right) + \left(R_{s,S'} + R_{s,s'}\right) D_2\right]\right] D_1 + \left(R_{s',z} + \beta R_{s,z} A\right) + \beta\left(R_{s,S'} + R_{s,s'}\right) D_1 A = 0.$$

The first is a second-order matrix polynomial, which from our discussion of solvents we know how many solutions we can expect. Given D_2, the second equation is a linear equation in D_1, which will have a unique solution.

To solve the equation for $P(D)$ we can use SQUINT. But we must determine which solutions constitute competitive equilibria; many of them will fail to satisfy the equilibrium requirement

$$d_1 z + d_2 S + d_3 S = D_1 z + D_2 S.$$

This means that only solutions which generate well-defined solutions to the household problem need to be considered. For example, if a solution implies nonstationary behavior, it can be discarded. Once we have all the solvents we have computed all the equilibria.

9.3 Dynamic Stackelberg Games

Ambler and Paquet (1997) show that one can solve for the equilibrium of dynamic Stackelberg games in a straightforward manner using a variant of Kydland's algorithm. While the notation will look formidable, the idea is simple; after computing the competitive equilibrium at iteration n, use the leader's first-order equation to solve for their choice variables as functions of the aggregate state; with the exception of this extra step, the procedure is the same as the previous subsection. To make things more general, I will present here some new ideas about how to introduce robustness and constitutional limits into a Stackelberg LQG game, based on some conversations with Jun Nie, and speculate about the existence of a tradeoff between flexibility and misaligned incentives (as in Young 2012).

The general model has six types of variables. z denotes exogenous states (and includes a constant), S denotes aggregate endogenous states, s denotes the individual state vector corresponding to S, G denotes government policy variables, d denotes individual decision variables, and D denotes the aggregate equivalent of d. We study a Stackelberg game between the government and the competitive private sector, and for simplicity suppose the government is restricted to time-consistent Markov strategies that arise as part of a recursive structure. We will approximate the return function as a quadratic and permit only linear constraints and Gaussian shocks.

There are two ways to handle nonlinear equilibrium conditions. The first, following Cooley and Hansen (1989), is to substitute them into the objective function of the household before taking

the quadratic approximation. This method leaves some equations of the model fully-nonlinear, but may not be possible in all contexts (Cooley and Hansen 1989 have to work pretty hard to get rid of the resource constraint since it depends on aggregate consumption and aggregate labor supply). We take a second approach, discussed in Kydland, Rupert, and Sustek (2014), where the equations are added as constraints on the household problem and we take a quadratic approximation to the Lagrangian; this method is more general and can in principle handle any model, although we may need to verify that the Lagrange multipliers have the right signs.

The basic linear-quadratic-Gaussian Stackelberg game begins with the private sector solving the problem

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix}$$

$$= \max_d \left\{ \begin{bmatrix} z^T & S^T & s^T & G^T & D^T & d^T \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} & Q_{15} & Q_{16} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} & Q_{25} & Q_{26} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} & Q_{35} & Q_{36} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} & Q_{45} & Q_{46} \\ Q_{51} & Q_{52} & Q_{53} & Q_{54} & Q_{55} & Q_{56} \\ Q_{61} & Q_{62} & Q_{63} & Q_{64} & Q_{65} & Q_{66} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \\ D \\ d \end{bmatrix} + \right.$$

$$\left. \beta \begin{bmatrix} z_+^T & S_+^T & s_+^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} z_+ \\ S_+ \\ s_+ \end{bmatrix} \right\}$$

subject to

$$\begin{aligned} z_+ &= Az + \Omega e_+ \\ S_+ &= B_1 z + B_2 S + B_3 G + B_4 D \\ s_+ &= C_1 z + C_2 S + C_3 s + C_4 G + C_5 D + C_6 d, \end{aligned}$$

taking as given aggregates (S, D) and policy G. Q and P are symmetric negative definite matrices, and we assume that the eigenvalues of A are bounded below one in modulus. P_{11} is therefore the square submatrix of P that involves only products of elements of z, and so on.

The individual private sector optimality condition has solution

$$\begin{aligned}
d^* = & -\left(Q_{66}+\beta C_6^T P_{33} C_6\right)^{-1}\left(Q_{61}+\beta C_6^T P_{31} A+\beta C_6^T P_{32} B_1+\beta C_6^T P_{33} C_1\right) z \\
& -\left(Q_{66}+\beta C_6^T P_{33} C_6\right)^{-1}\left(Q_{62}+\beta C_6^T P_{32} B_2+\beta C_6^T P_{33} C_2\right) S \\
& -\left(Q_{66}+\beta C_6^T P_{33} C_6\right)^{-1}\left(Q_{63}+\beta C_6^T P_{33} C_3\right) s \\
& -\left(Q_{66}+\beta C_6^T P_{33} C_6\right)^{-1}\left(Q_{64}+\beta C_6^T P_{32} B_3+\beta C_6^T P_{33} C_4\right) G \\
& -\left(Q_{66}+\beta C_6^T P_{33} C_6\right)^{-1}\left(Q_{65}+\beta C_6^T P_{32} B_4+\beta C_6^T P_{33} C_5\right) D \\
= & F_1 z+F_2 S+F_3 s+F_4 G+F_5 D.
\end{aligned}$$

To eliminate D we note that $s=S$ and $d=D$ to obtain

$$\begin{aligned}
D^* &= \left(I-F_5\right)^{-1} F_1 z+\left(I-F_5\right)^{-1}\left(F_2+F_3\right) S+\left(I-F_5\right)^{-1} F_4 G \\
&= H_1 z+H_2 S+H_3 G \\
S_+^* &= \left(B_1+B_4 H_1\right) z+\left(B_2+B_4 H_2\right) S+\left(B_3+B_4 H_3\right) G \\
&= J_1 z+J_2 S+J_3 G.
\end{aligned}$$

We can then obtain individual decisions from

$$\begin{aligned}
d^* &= \left(F_1+F_5 H_1\right) z+\left(F_2+F_5 H_2\right) S+F_3 s+\left(F_4+F_5 H_3\right) G \\
&= M_1 z+M_2 S+M_3 s+M_4 G \\
s_+^* &= \left(C_1+C_5 H_1\right) z+\left(C_2+C_5 H_2\right) S+C_3 s+\left(C_4+C_5 H_3\right) G \\
&= N_1 z+N_2 S+N_3 s+N_4 G.
\end{aligned}$$

Therefore, the RHS of the functional equation, given G, is

$$\left\{ \begin{array}{l} \left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array} \right] \left[\begin{array}{cccc} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} \end{array} \right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array} \right] + \\ \left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array} \right] \left[\begin{array}{cc} Q_{15} & Q_{16} \\ Q_{25} & Q_{26} \\ Q_{35} & Q_{36} \\ Q_{45} & Q_{46} \end{array} \right] \left[\begin{array}{cccc} H_1 & H_2 & 0 & H_3 \\ M_1 & M_2 & M_3 & M_4 \end{array} \right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array} \right] + \\ \left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array} \right] \left[\begin{array}{cc} H_1^T & M_1^T \\ H_2^T & M_2^T \\ 0 & M_3^T \\ H_3^T & M_4^T \end{array} \right] \left[\begin{array}{cccc} Q_{51} & Q_{52} & Q_{53} & Q_{54} \\ Q_{61} & Q_{62} & Q_{63} & Q_{64} \end{array} \right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array} \right] + \\ \left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array} \right] \left[\begin{array}{cc} H_1^T & M_1^T \\ H_2^T & M_2^T \\ 0 & M_3^T \\ H_3^T & M_4^T \end{array} \right] \left[\begin{array}{cc} Q_{55} & Q_{56} \\ Q_{65} & Q_{66} \end{array} \right] \left[\begin{array}{cccc} H_1 & H_2 & 0 & H_3 \\ M_1 & M_2 & M_3 & M_4 \end{array} \right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array} \right] \\ \beta \left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array} \right] \left[\begin{array}{ccc} A^T & J_1^T & N_1^T \\ 0 & J_2^T & N_2^T \\ 0 & 0 & N_3^T \\ 0 & J_3^T & N_4^T \end{array} \right] \left[\begin{array}{ccc} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{array} \right] \left[\begin{array}{cccc} A & 0 & 0 & 0 \\ J_1 & J_2 & 0 & J_3 \\ N_1 & N_2 & N_3 & N_4 \end{array} \right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array} \right] \end{array} \right\}$$

or more compactly as

$$\left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array} \right] \left[\begin{array}{cccc} R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{array} \right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array} \right]$$

Now the Stackelberg leader (the government) is going to choose G to maximize this expression

at $s = S$, which gives us

$$\begin{bmatrix} z^T & S^T & G^T \end{bmatrix} \begin{bmatrix} R_{11} & R_{12}+R_{13} & R_{14} \\ R_{21}+R_{31} & R_{22}+R_{23}+R_{32}+R_{33} & R_{24}+R_{34} \\ R_{41} & R_{42}+R_{43} & R_{44} \end{bmatrix} \begin{bmatrix} z \\ S \\ G \end{bmatrix}.$$

The first-order condition is

$$R_{14}z + (R_{24}+R_{34})\,S + R_{44}G = 0$$

so that

$$\begin{aligned} G^* &= -R_{44}^{-1}R_{14}z - R_{44}^{-1}\,(R_{24}+R_{34})\,S \\ &= U_1 z + U_2 S. \end{aligned}$$

Now plug that into the RHS and match to the LHS:

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix}$$

$$= \begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} R_{14} \\ R_{24} \\ R_{34} \end{bmatrix} \begin{bmatrix} U_1 & U_2 & 0 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \\ 0 \end{bmatrix} \begin{bmatrix} R_{41} & R_{42} & R_{43} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \\ 0 \end{bmatrix} [R_{44}] \begin{bmatrix} U_1 & U_2 & 0 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix}.$$

The matrix P solves this equation.

To add robustness, define the operator $\mathcal{D}(P)$ before:

$$\mathcal{D}(P) = P - \sigma P\Sigma \left(I + \sigma \Sigma P \Sigma\right)^{-1} \Sigma P$$

where

$$\Sigma = \begin{bmatrix} \Omega\Omega^T & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Also suppose that this adjustment applies only to the government's problem, so that the objective becomes

$$\left\{ \begin{array}{c} \begin{bmatrix} z^T & S^T & s^T & G^T \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \end{bmatrix} + \\ \begin{bmatrix} z^T & S^T & s^T & G^T \end{bmatrix} \begin{bmatrix} Q_{15} & Q_{16} \\ Q_{25} & Q_{26} \\ Q_{35} & Q_{36} \\ Q_{45} & Q_{46} \end{bmatrix} \begin{bmatrix} H_1 & H_2 & 0 & H_3 \\ M_1 & M_2 & M_3 & M_4 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \end{bmatrix} + \\ \begin{bmatrix} z^T & S^T & s^T & G^T \end{bmatrix} \begin{bmatrix} H_1^T & M_1^T \\ H_2^T & M_2^T \\ 0 & M_3^T \\ H_3^T & M_4^T \end{bmatrix} \begin{bmatrix} Q_{51} & Q_{52} & Q_{53} & Q_{54} \\ Q_{61} & Q_{62} & Q_{63} & Q_{64} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \end{bmatrix} + \\ \begin{bmatrix} z^T & S^T & s^T & G^T \end{bmatrix} \begin{bmatrix} H_1^T & M_1^T \\ H_2^T & M_2^T \\ 0 & M_3^T \\ H_3^T & M_4^T \end{bmatrix} \begin{bmatrix} Q_{55} & Q_{56} \\ Q_{65} & Q_{66} \end{bmatrix} \begin{bmatrix} H_1 & H_2 & 0 & H_3 \\ M_1 & M_2 & M_3 & M_4 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \end{bmatrix} \\ \beta \begin{bmatrix} z^T & S^T & s^T & G^T \end{bmatrix} \begin{bmatrix} A^T & J_1^T & N_1^T \\ 0 & J_2^T & N_2^T \\ 0 & 0 & N_3^T \\ 0 & J_3^T & N_4^T \end{bmatrix} \mathcal{D}\left(\begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \right) \begin{bmatrix} A & 0 & 0 & 0 \\ J_1 & J_2 & 0 & J_3 \\ N_1 & N_2 & N_3 & N_4 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \end{bmatrix} \end{array} \right\}$$

which can be written compactly as

$$\left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array}\right] \left[\begin{array}{cccc} V_{11} & V_{12} & V_{13} & V_{14} \\ V_{21} & V_{22} & V_{23} & V_{24} \\ V_{31} & V_{32} & V_{33} & V_{34} \\ V_{41} & V_{42} & V_{43} & V_{44} \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array}\right];$$

it is easy to also accomodate risk-sensitivity on the part of the private sector. The first-order condition is

$$V_{14}z + (V_{24} + V_{34})\, S + V_{44}G = 0$$

so that

$$\begin{aligned} G^* &= -V_{44}^{-1}V_{14}z - V_{44}^{-1}\,(V_{24} + V_{34})\, S \\ &= W_1 z + W_2 S. \end{aligned}$$

Now plug that into the RHS and match to the LHS, using R instead of V:

$$\left[\begin{array}{ccc} z^T & S^T & s^T \end{array}\right] \left[\begin{array}{ccc} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \end{array}\right]$$

$$= \left[\begin{array}{ccc} z^T & S^T & s^T \end{array}\right] \left[\begin{array}{ccc} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \end{array}\right] + \left[\begin{array}{ccc} z^T & S^T & s^T \end{array}\right] \left[\begin{array}{c} R_{14} \\ R_{24} \\ R_{34} \end{array}\right] \left[\begin{array}{ccc} W_1 & W_2 & 0 \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \end{array}\right] +$$

$$\left[\begin{array}{ccc} z^T & S^T & s^T \end{array}\right] \left[\begin{array}{c} W_1^T \\ W_2^T \\ 0 \end{array}\right] \left[\begin{array}{ccc} R_{41} & R_{42} & R_{43} \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \end{array}\right] + \left[\begin{array}{ccc} z^T & S^T & s^T \end{array}\right] \left[\begin{array}{c} W_1^T \\ W_2^T \\ 0 \end{array}\right] [R_{44}] \left[\begin{array}{ccc} W_1 & W_2 & 0 \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \end{array}\right].$$

The matrix P solves this equation under robustness. Note that, in general, $U_i \neq W_i$, so that decisions by the "robust" government will not align with those of the "nonrobust" government. Two features of general robust decision making may come into play here: (i) precautionary effects appear, altering the intercept term in W_1; and (ii) slopes generally get steeper, changing W_1 and W_2 in a way that involves more aggressive responses to changes in the state.

The final piece is to determine the "constitution" that the government operates under. Suppose we can "penalize" the government by subtracting a term that depends on size of policy

intervention:

$$\left[\begin{array}{cccc} z^T & S^T & s^T & G^T \end{array}\right] \left[\begin{array}{cccc} V_{11} & V_{12} & V_{13} & V_{14} \\ V_{21} & V_{22} & V_{23} & V_{24} \\ V_{31} & V_{32} & V_{33} & V_{34} \\ V_{41} & V_{42} & V_{43} & V_{44} \end{array}\right] \left[\begin{array}{c} z \\ S \\ s \\ G \end{array}\right] - G^T \Theta G$$

for some matrix Θ. To see how this term might work, suppose we have a model with capital and labor taxes, and we want to consider the effects of having a preference for uniform income taxes (that is, tax systems where $\tau_k = \tau_l$). Then we can compute the Θ matrix as the solution to

$$\left[\begin{array}{cc} \tau_k & \tau_l \end{array}\right] \left[\begin{array}{cc} \Theta_{11} & \Theta_{12} \\ \Theta_{21} & \Theta_{22} \end{array}\right] \left[\begin{array}{c} \tau_k \\ \tau_l \end{array}\right] = (\tau_k - \tau_l)^2$$

which yields

$$\begin{aligned} \Theta_{11} &= \Theta_{22} = 1 \\ \Theta_{12} &= \Theta_{21} = -1. \end{aligned}$$

Thus, a wide range of possible restrictions can be embedded in Θ (we can scale the cost up by multiplying Θ_{ij} by some constant if we are dissatisfied with the resulting policy functions, just as in penalty and interior point methods).

Now the first order condition is

$$V_{41} z + (V_{42} + V_{43}) S + V_{44} G - \Theta G = 0$$

so that

$$\begin{aligned} G^* &= (\Theta - V_{44})^{-1} V_{41} z + (\Theta - V_{44})^{-1} (V_{42} + V_{43}) S \\ &= X_1 z + X_2 S. \end{aligned}$$

Private sector welfare is then the solution to

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix}$$

$$= \begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} R_{14} \\ R_{24} \\ R_{34} \end{bmatrix} \begin{bmatrix} X_1 & X_2 & 0 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} X_1^T \\ X_2^T \\ 0 \end{bmatrix} \begin{bmatrix} R_{41} & R_{42} & R_{43} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix} +$$

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} X_1^T \\ X_2^T \\ 0 \end{bmatrix} [R_{44}] \begin{bmatrix} X_1 & X_2 & 0 \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix},$$

which depends on Θ.

The ex ante optimal Θ can be found by solving

$$\Theta^* = \underset{\Theta}{\text{argmax}} \left\{ \begin{bmatrix} z^T & S^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} + P_{13} \\ P_{21} + P_{31} & P_{22} + P_{23} + P_{32} + P_{33} \end{bmatrix} \begin{bmatrix} z \\ S \end{bmatrix} \right\}.$$

The term $G^T \Theta G$ must be positive semidefinite for this problem to have a solution (at least, to guarantee one that we can find in polynomial time; remember the quadratic program is NP-hard).

For some problems, there will be endogenous variables that are not the aggregate equivalent of individual ones (such as the price level in Cooley and Hansen 1989). To handle these models,

we add another variable, Y, and suppose the individual problem is given by

$$\begin{bmatrix} z^T & S^T & s^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \end{bmatrix}$$

$$= \max_d \left\{ \begin{bmatrix} z^T & S^T & s^T & G^T & D^T & d^T & Y^T \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} & Q_{13} & Q_{14} & Q_{15} & Q_{16} & Q_{17} \\ Q_{21} & Q_{22} & Q_{23} & Q_{24} & Q_{25} & Q_{26} & Q_{27} \\ Q_{31} & Q_{32} & Q_{33} & Q_{34} & Q_{35} & Q_{36} & Q_{37} \\ Q_{41} & Q_{42} & Q_{43} & Q_{44} & Q_{45} & Q_{46} & Q_{47} \\ Q_{51} & Q_{52} & Q_{53} & Q_{54} & Q_{55} & Q_{56} & Q_{57} \\ Q_{61} & Q_{62} & Q_{63} & Q_{64} & Q_{65} & Q_{66} & Q_{67} \\ Q_{71} & Q_{72} & Q_{73} & Q_{74} & Q_{75} & Q_{76} & Q_{77} \end{bmatrix} \begin{bmatrix} z \\ S \\ s \\ G \\ D \\ d \\ Y \end{bmatrix} + \right.$$
$$\left. \beta \begin{bmatrix} z_+^T & S_+^T & s_+^T \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} \begin{bmatrix} z_+ \\ S_+ \\ s_+ \end{bmatrix} \right\}$$

subject to

$$\begin{aligned} z_+ &= Az + \Omega e_+ \\ S_+ &= B_1 z + B_2 S + B_3 G + B_4 D + B_5 Y \\ s_+ &= C_1 z + C_2 S + C_3 s + C_4 G + C_5 D + C_6 d + C_7 Y \\ 0 &= F_1 z + F_2 S + F_3 D + F_4 Y + F_5 G. \end{aligned}$$

The last equation are the market clearing conditions that are needed to obtain values for Y. The

first-order condition is

$$\begin{aligned}
d^* = &- \left(Q_{66} + \beta C_6^T P_{33} C_6\right)^{-1} \left(Q_{61} + \beta C_6^T P_{31} A + \beta C_6^T P_{32} B_1 + \beta C_6^T P_{33} C_1\right) z \\
&- \left(Q_{66} + \beta C_6^T P_{33} C_6\right)^{-1} \left(Q_{62} + \beta C_6^T P_{32} B_2 + \beta C_6^T P_{33} C_2\right) S \\
&- \left(Q_{66} + \beta C_6^T P_{33} C_6\right)^{-1} \left(Q_{63} + \beta C_6^T P_{33} C_3\right) s \\
&- \left(Q_{66} + \beta C_6^T P_{33} C_6\right)^{-1} \left(Q_{64} + \beta C_6^T P_{32} B_3 + \beta C_6^T P_{33} C_4\right) G \\
&- \left(Q_{66} + \beta C_6^T P_{33} C_6\right)^{-1} \left(Q_{65} + \beta C_6^T P_{32} B_4 + \beta C_6^T P_{33} C_5\right) D \\
&- \left(Q_{66} + \beta C_6^T P_{33} C_6\right)^{-1} \left(Q_{67} + \beta C_6^T P_{32} B_5 + \beta C_6^T P_{33} C_7\right) Y \\
&= H_1 z + H_2 S + H_3 s + H_4 G + H_5 D + H_6 Y
\end{aligned}$$

Now we set $s = S$, $d = D$, and solve the system for (D, Y):

$$\begin{aligned}
\begin{bmatrix} D^* \\ Y^* \end{bmatrix} = &\begin{bmatrix} I - H_5 & -H_6 \\ -F_3 & -F_4 \end{bmatrix}^{-1} \begin{bmatrix} H_1 \\ F_1 \end{bmatrix} z + \\
&\begin{bmatrix} I - H_5 & -H_6 \\ -F_3 & -F_4 \end{bmatrix}^{-1} \begin{bmatrix} H_2 + H_3 \\ F_2 \end{bmatrix} S + \\
&\begin{bmatrix} I - H_5 & -H_6 \\ -F_3 & -F_4 \end{bmatrix}^{-1} \begin{bmatrix} H_4 \\ F_5 \end{bmatrix} G \\
D^* = &\, J_1 z + J_2 S + J_3 G \\
Y^* = &\, K_1 z + K_2 S + K_3 G.
\end{aligned}$$

We then obtain

$$\begin{aligned}
S_+^* &= \left(B_1 + B_4 J_1 + B_5 K_1\right) z + \left(B_2 + B_4 J_2 + B_5 K_2\right) S + \left(B_3 + B_4 J_3 + B_5 K_3\right) G \\
&= \Psi_1 z + \Psi_2 S + \Psi_3 G \\
d^* &= \left(H_1 + H_5 J_1 + H_6 K_1\right) z + \left(H_2 + H_5 J_2 + H_6 K_2\right) S + H_3 s + \left(H_4 + H_5 J_3 + H_6 K_3\right) G \\
&= \Phi_1 z + \Phi_2 S + \Phi_3 s + \Phi_4 G \\
s_+^* &= \left(C_1 + C_5 J_1 + C_6 K_1\right) z + \left(C_2 + C_5 J_2 + C_6 K_2\right) S + C_3 s + \left(C_4 + C_5 J_3 + C_6 K_3\right) G \\
&= \Pi_1 z + \Pi_2 S + \Pi_3 s + \Pi_4 G.
\end{aligned}$$

Now we update the P matrix and repeat to convergence.

9.4 Information-Constrained LQG Programming

The literature on rational inattention studies the information-attention problem – if agents can only process a limited amount of information, how will they choose to collect it? Sims (1998) first suggested this idea as a mechanism for producing stickiness in decisions, and developed a tractable LQG version in Sims (2003); Miao, Wu, and Young (2020) provide a complete investigation of the properties of this problem in the LQG framework (see also Raginsky, Shafieepoorfard, and Meyn 2015). For obvious reasons, I will choose to follow MWY in the presentation.

We start with a finite-horizon linear-quadratic control problem under rational inattention. Let the n_x-dimensional state vector x_t follow the linear dynamics

$$x_{t+1} = A_t x_t + B_t u_t + \epsilon_{t+1},\ t = 0, 1, ..., T, \tag{100}$$

where u_t is an n_u dimensional control variable and ϵ_{t+1} is a serially independent Gaussian random vector with mean zero and covariance matrix W_t. The matrix W_t is positive semidefinite, denoted by $W_t \succeq 0$. The state transition matrix A_t and the control coefficient matrix B_t are deterministic and conformable. The state vector x_t may contain both exogenous states such as AR(1) shocks and endogenous states such as capital.

Suppose that the decision maker does not observe the state x_t perfectly, but observes a multi-dimensional noisy signal s_t about x_t given by

$$s_t = C_t x_t + v_t,\ t = 0, 1, ..., T, \tag{101}$$

where C_t is a conformable deterministic matrix and v_t is a serially independent Gaussian random variable with mean zero and covariance matrix $V_t \succ 0$. Assume that x_0 is a Gaussian random variable with mean $\bar{x}_0$ and covariance matrix Σ_0. The random variables ϵ_t, v_t, and x_0 are all mutually independent for all t. The decision maker's information set at date t is generated by $s^t = \{s_0, s_1, ..., s_t\}$. The control u_t is measurable with respect to s^t.

Suppose that the decision maker is boundedly rational and has limited information-processing capacity. He faces the following information-flow constraint

$$\sum_{t=0}^{T} I\left(x_t; s_t | s^{t-1}\right) \leq \kappa, \tag{102}$$

where $\kappa > 0$ denotes the information-flow rate or channel capacity and $I\left(x_t; s_t | s^{t-1}\right)$ denotes the conditional (Shannon) mutual information between x_t and s_t given s^{t-1},

$$I\left(x_t; s_t | s^{t-1}\right) \equiv H\left(x_t | s^{t-1}\right) - H\left(x_t | s^t\right).$$

Here $H(\cdot|\cdot)$ denotes the conditional entropy operator. Let $s^{-1} = \varnothing$. Intuitively, entropy measures uncertainty. At each time t, given past information s^{t-1}, observing s_t reduces uncertainty about x_t. The total uncertainty reduction from time 0 to time T is measured by the expression on the left-hand side of the inequality in (102). The decision maker can process information by choosing the information structure represented by $\{C_t, V_t\}_{t=0}^T$ for the signal s_t, but the rate of uncertainty reduction is limited by an upper bound κ. For example, if C_t is equal to the identity matrix and $V_t = 0$ for all t, then the decision maker fully observes x_t and hence $I\left(x_t; s_t | s^{t-1}\right) = \infty$, violating the information-flow constraint (102).

Notice that the choice of $\{C_t, V_t\}_{t=0}^T$ implies that the dimension of the signal vector s_t and the correlation structure of the noise v_t are endogenous and may vary over time. One may imagine the decision maker makes decisions sequentially. He first chooses the information structure $\{C_t, V_t\}_{t=0}^T$ and then selects a control $\{u_t\}_{t=0}^T$ adapted to $\left\{s^t\right\}$ to maximize an objective function. Suppose that the objective function is quadratic. We are ready to formulate the decision maker's problem as follows:

Problem 91 *(Finite-horizon LQG problem under RI)*

$$\max_{\{u_t\},\{C_t\},\{V_t\}} \left\{ -E\left[\sum_{t=0}^{T} \beta^t \left(x_t^T Q_t x_t + u_t^T R_t u_t + 2 x_t^T S_t u_t \right) + \beta^{T+1} x_{T+1}^T P_{T+1} x_{T+1} \middle| s_0 \right] \right\}$$

subject to (100), (101), and (102).

The parameter $\beta \in (0, 1]$ represents the discount factor. The deterministic matrices Q_t, R_t, and S_t for all t and P_{T+1} are conformable and exogenously given. For the infinite-horizon stationary case, we set $T \to \infty$ and remove the time index for all exogenously given matrices A_t, B_t, Q_t, R_t, and S_t. We also replace (102) by

$$\limsup_{T \to \infty} \left\{ \frac{1}{T} \sum_{t=0}^{T} I\left(x_t; s_t | s^{t-1}\right) \right\} \leq \kappa, \tag{103}$$

where

$$s_t = Cx_t + v_t \tag{104}$$

and v_t is a serially independent Gaussian random variable with mean zero and covariance matrix $V \succ 0$. The interpretation is that the average rate of uncertainty reduction per period is limited by $\kappa > 0$. We now formulate the infinite-horizon problem as follows:

Problem 92 *(Infinite-horizon stationary LQG problem under RI)*

$$\max_{\{u_t\},C,V} \left\{ -E\left[\sum_{t=0}^{\infty} \beta^t \left(x_t^T Q x_t + u_t^T R u_t + 2 x_t^T S u_t \right) \right] \right\}$$

subject to (103), (104), and

$$x_{t+1} = Ax_t + Bu_t + \epsilon_{t+1}, \tag{105}$$

for $t \geq 0$. Here the expectation is taken with respect to the unconditional stationary distribution.

In applications, it may also be more convenient to consider the following relaxed problems.

Problem 93 *(Finite-horizon relaxed LQG problem under RI)*

$$\max_{\{u_t\},\{C_t\},\{V_t\}} \left\{ -E\left[\sum_{t=0}^{T} \beta^t \left(x_t^T Q_t x_t + u_t^T R_t u_t + 2 x_t^T S_t u_t \right) + \beta^{T+1} x_{T+1}^T P_{T+1} x_{T+1} \middle| s_0 \right] - \lambda \sum_{t=0}^{T} I\left(x_t; s_t | s^{t-1} \right) \right\}$$

subject to (100) and (101).

Problem 94 *(Infinite-horizon stationary relaxed LQG problem under RI)*

$$\max_{\{u_t\},C,V} \left\{ -E\left[\sum_{t=0}^{\infty} \beta^t \left(x_t^T Q x_t + u_t^T R u_t + 2 x_t^T S u_t \right) \right] - \lambda \limsup_{T\to\infty} \left\{ \frac{1}{T} \sum_{t=0}^{T} I\left(x_t; s_t | s^{t-1} \right) \right\} \right\}$$

subject to (104) and (105) for $t = 0, 1, ...,$ where the expectation is taken with respect to the unconditional stationary distribution.

In these two problems $\lambda > 0$ can be interpreted as the Lagrange multiplier associated with the information-flow constraint or the shadow price (cost) of the information flow. We will focus our analysis on Problem 91. The solution to Problem 92 is the limit of that to Problem 91. Problems 93 and 94 can be similarly analyzed and are easier to solve because the information-flow constraint is removed from the optimization.

9.5 Full Information Case

Before analyzing Problem 91, we first present the solution in the full information case, in which the decision maker observes x_t perfectly. Suppose that $P_{T+1} \succeq 0$, $R_t \succ 0$, and

$$\begin{bmatrix} Q_t & S_t \\ S_t^T & R_t \end{bmatrix} \succeq 0$$

for all $t = 0, 1, ..., T$. Then the value function takes the form

$$v_t^{FI}(x_t) = -x_t^T P_t x_t - g_t, \tag{106}$$

where $P_t \succeq 0$ and satisfies

$$\begin{aligned} P_t = {} & Q_t + \beta A_t^T P_{t+1} A_t \\ & - \left(\beta A_t^T P_{t+1} B_t + S_t\right) \left(R_t + \beta B_t^T P_{t+1} B_t\right)^{-1} \left(\beta B_t^T P_{t+1} A_t + S_t^T\right), \end{aligned} \tag{107}$$

and g_t satisfies

$$g_t = \beta \operatorname{trace}(P_{t+1} W_t) + \beta g_{t+1}, \; g_{T+1} = 0,$$

for $t = 0, 1, ..., T$. Here $\operatorname{tr}(\cdot)$ denotes the trace operator.

The optimal control is

$$u_t = -F_t x_t, \tag{108}$$

where

$$F_t = \left(R_t + \beta B_t^T P_{t+1} B_t\right)^{-1} \left(S_t^T + \beta B_t^T P_{t+1} A_t\right). \tag{109}$$

For the infinite horizon case, all exogenous matrices are time invariant. As $T \to \infty$, we obtain the infinite-horizon solution under some standard stability conditions. The value function is given by

$$v^{FI}(x_t) = -x_t^T P x_t - g,$$

where $P \succeq 0$ and satisfies

$$P = Q + \beta A' P A - \left(\beta A^T P B + S\right) \left(R + \beta B^T P B\right)^{-1} \left(\beta B^T P A + S^T\right),$$

and

$$g = \frac{\beta}{1-\beta} \operatorname{trace}(PW).$$

The optimal control is given by

$$u_t = -Fx_t, \tag{110}$$

where

$$F = \left(R + \beta B^T P B\right)^{-1} \left(S^T + \beta B^T P A\right).$$

9.6 Tracking Problem

We solve Problem 91 in three steps. In the first step we observe that Problem 91 is a standard LQG problem under partial information for fixed $\{C_t, V_t\}_{t=0}^T$. Thus the usual certainty equivalence principle holds. This implies that the optimal control is given by

$$u_t = -F_t \widehat{x}_t, \tag{111}$$

where $\widehat{x}_t \equiv \mathbb{E}\left[x_t | s^t\right]$ denotes the estimate of x_t given information s^t. The state under the optimal control satisfies

$$x_{t+1} = A_t x_t - B_t F_t \widehat{x}_t + \epsilon_{t+1}. \tag{112}$$

By the Kalman filtering formula, $\widehat{x}_t$ follows the dynamics

$$\widehat{x}_t = \widehat{x}_{t|t-1} + \Sigma_{t|t-1} C_t' \left(C_t \Sigma_{t|t-1} C_t' + V_t\right)^{-1} \left(s_t - C_t \widehat{x}_{t|t-1}\right), \tag{113}$$

$$\widehat{x}_{t|t-1} = \left(A_{t-1} - B_{t-1} F_{t-1}\right) \widehat{x}_{t-1}, \tag{114}$$

where $\widehat{x}_{t|t-1} \equiv \mathbb{E}\left[x_t | s^{t-1}\right]$ with $\widehat{x}_{0|-1} = \bar{x}_0$ and $\Sigma_{t|t-1} \equiv \mathbb{E}\left[\left(x_t - \widehat{x}_{t|t-1}\right)\left(x_t - \widehat{x}_{t|t-1}\right)' | s^{t-1}\right]$ with $\Sigma_{0|-1} = \Sigma_0$. Moreover

$$\Sigma_{t|t-1} = A_{t-1} \Sigma_{t-1|t-1} A_{t-1}' + W_{t-1}, \tag{115}$$

$$\Sigma_{t|t} = \left(\Sigma_{t|t-1}^{-1} + \Phi_t\right)^{-1}, \tag{116}$$

for $t = 0, 1, ..., T$, where $\Sigma_{t|t} \equiv \mathbb{E}\left[\left(x_t - \widehat{x}_t\right)\left(x_t - \widehat{x}_t\right)' | s^t\right]$ and Φ_t denotes the "signal-to-noise ratio" defined by

$$\Phi_t = C_t' V_t^{-1} C_t \succeq 0$$

for $t = 0, 1, ..., T$.

We now solve for $\{C_t, V_t\}_{t=0}^T$ in the remaining two steps. The next step is to show that $\{C_t, V_t\}_{t=0}^T$ can be computed by solving a tracking problem.

Proposition 95 *Suppose that the assumptions in Section 2.1 are satisfied. Then the optimal control for Problem 91 is given by (111) and the optimal* $\{C_t, V_t\}_{t=0}^T$ *solves the following problem:*

$$\min_{\{C_t\},\{V_t\}} \left\{ E\left[\sum_{t=0}^{T} \left(x_t - \widehat{x}_t\right)^T \Omega_t \left(x_t - \widehat{x}_t\right) \middle| s_0 \right] \right\} \tag{117}$$

subject to (101), (102), and (112), where

$$\Omega_t = \beta^t F_t^T \left(R_t + \beta B_t^T P_{t+1} B_t\right) F_t \succeq 0. \tag{118}$$

Since the information-flow constraint cannot help in the optimization, we know that $E\left[v_t^{FI}\left(x_t\right) \middle| s^t\right] > \widehat{v}_t\left(\widehat{x}_t\right)$. Following Sims (2003), we solve for $\{C_t, V_t\}_{t=0}^T$ by minimizing $E\left[v_0^{FI}\left(x_0\right) \middle| s_0\right] - \widehat{v}_0\left(\widehat{x}_0\right)$. In other words, we choose information structure so as to bring expected utility from the current date onward as close as possible to the expected utility value under full information. The proposition shows that $E\left[v_0^{FI}\left(x_0\right) \middle| s_0\right] - \widehat{v}_0\left(\widehat{x}_0\right)$ is equal to the expression in (117).

Notice that the matrix Ω_t is positive semidefinite because $R_t \succ 0$ and $P_{t+1} \succeq 0$. Since F_t is an n_u by n_x dimensional matrix, $\text{rank}\left(\Omega_t\right)$ does not exceed the minimum of the dimension n_x of the state vector and the dimension n_u of the control vector. Thus it is possible that Ω_t is singular. If $n_x \geq n_u$ and F_t has full column rank, then $\text{rank}\left(\Omega_t\right) = n_u$. If $n_x < n_u$ and F_t has full row rank, then $\text{rank}\left(\Omega_t\right) = n_x$.

In the infinite-horizon stationary case, suppose that the initial state is the stationary distribution so that the decision-maker will use the stationary Kalman filter (one could also suppose that the initial state is fixed at some arbitrary prior so we need to allow time-dependence in the filter; that is more notationally burdensome and requires us to solve a sequence of attention-allocation decisions). We can then compute the limit

$$\lim_{T\to\infty} \left\{ E\left[\sum_{t=0}^{T} \left(x_t - \widehat{x}_t\right)^T \Omega_t \left(x_t - \widehat{x}_t\right) \right] \right\} = \frac{1}{1-\beta} E\left[\left(x_t - \widehat{x}_t\right)^T \Omega \left(x_t - \widehat{x}_t\right) \right],$$

where the expectation is taken using the stationary distribution and

$$\Omega = F^T \left(R + \beta B^T P B\right) F. \tag{119}$$

Miao, Wu, and Young (2020) show the following result.

Proposition 96 *Suppose that the assumptions in Section 2.1 hold. The optimal control for Problem 92 is given by* $u_t = -F\widehat{x}_t$*, and the optimal* C *and* V *solve the following problem*

$$\min_{C,V} \left\{ E\left[(x_t - \widehat{x}_t)^T \Omega (x_t - \widehat{x}_t) \right] \right\}$$

subject to (103), (104), and

$$x_{t+1} = Ax_t - BF\widehat{x}_t + \epsilon_{t+1}, \tag{120}$$

where $\Omega \succeq 0$ *is given by (119).*

The previous two propositions state that an optimal information structure in the control problem under RI can be found by solving a tracking problem. It is important that the weighting matrices Ω_t and Ω must be positive semidefinite, which ensure that the tracking problems are well-defined.

9.7 The Rate Distortion Function

In the final step of our solution procedure, we transform the tracking problem into a simpler equivalent problem. We will draw connection to the engineering literature on information theory (Cover and Thomas 2006). In this literature the objective function in (117) measures the distance between sequences x^T and $\widehat{x}^T$. It is a distortion measure for the source random sequence x^T and its estimate $\widehat{x}^T$. Define the function $D_T(\kappa)$ as the minimized distortion for all rates $\kappa > 0$ such that $D_T(\kappa)$ is finite. This function is called a **distortion rate function** in information theory and is equal to the minimized objective function in (117) in our context.

A closely related concept is the (information) **rate distortion function** $\kappa_T(D)$, defined as the minimized information-flow rate for all distortions $D > 0$ such that $\kappa_T(D)$ is finite. It is given by the solution to the following problem.

Problem 97 *(Finite-horizon rate distortion problem)*

$$\kappa_T(D) \equiv \min_{\{C_t, V_t\}_{t=0}^T} \left\{ \sum_{t=0}^{T} I\left(x_t; s_t | s^{t-1}\right) \right\} \tag{121}$$

subject to (100), (101), and

$$E\left[\sum_{t=0}^{T} (x_t - \widehat{x}_t)^T \Omega_t (x_t - \widehat{x}_t) \middle| s_0 \right] \leq D. \tag{122}$$

It can be checked that both $\kappa_T(D)$ and $D_T(\kappa)$ are decreasing and convex functions. Thus, for any $\kappa > 0$, we can find a $D > 0$ such that the solution to the rate distortion problem in (121) gives the solution to the distortion rate problem or the tracking problem in (117) (Cover and Thomas (2006)). The rate distortion problem is easier to solve numerically because the complex information-flow constraint is moved into the objective function so that the problem has a semidefinite programming representation as shown in Section 3.[2] We thus focus our analysis on the problem in (121).

To solve this problem, we compute the mutual information[3]

$$\begin{aligned} I\left(x_t; s_t | s^{t-1}\right) &= H\left(x_t | s^{t-1}\right) - H\left(x_t | s^t\right) \\ &= \frac{1}{2}\log\det\left(A_{t-1}\Sigma_{t-1|t-1}A_{t-1}' + W_{t-1}\right) - \frac{1}{2}\log\det\left(\Sigma_{t|t}\right) \end{aligned}$$

for $t = 1, 2, .., T$, and

$$I\left(x_0; s_0 | s^{-1}\right) = H(x_0) - H(x_0|s_0) = \frac{1}{2}\log\left(\det\left(\Sigma_0\right)\right) - \frac{1}{2}\log\left(\det\left(\Sigma_{0|0}\right)\right)$$

for $t = 0$, where the functions $H(\cdot)$ and $H(\cdot|\cdot)$ denote the entropy and conditional entropy operators, and $\det(\cdot)$ denotes the determinant operator. Moreover, we compute

$$E\left[\sum_{t=0}^{T}\left(x_t - \widehat{x}_t\right)^T \Omega_t \left(x_t - \widehat{x}_t\right) \middle| s_0\right] = \sum_{t=0}^{T}\operatorname{trace}\left(\Omega_t \Sigma_{t|t}\right).$$

Thus the rate distortion problem becomes

$$\kappa_T(D) = \min_{\{\Phi_t \succeq 0\}, \{\Sigma_{t|t}\}} \left\{ \begin{array}{c} \frac{1}{2}\log\det\left(\Sigma_0\right) - \frac{1}{2}\log\det\left(\Sigma_{0|0}\right) + \\ \sum_{t=1}^{T}\frac{1}{2}\log\det\left(A_{t-1}\Sigma_{t-1|t-1}A_{t-1}' + W_{t-1}\right) - \frac{1}{2}\log\det\left(\Sigma_{t|t}\right) \end{array} \right\} \quad (123)$$

subject to

$$\Sigma_{t|t} = \left[\left(A_{t-1}\Sigma_{t-1|t-1}A_{t-1}' + W_{t-1}\right)^{-1} + \Phi_t\right]^{-1}, \quad t = 1, 2, ..., T, \quad (124)$$

$$\Sigma_{0|0} = \left(\Sigma_0^{-1} + \Phi_0\right)^{-1}, \quad (125)$$

$$\sum_{t=0}^{T}\operatorname{trace}\left(\Omega_t \Sigma_{t|t}\right) \leq D. \quad (126)$$

[2]The relaxed Problems 93 and 94 are also easier to solve than the original Problems 91 and 92.

[3]The usual base for logarithm in the entropy formula is 2, in which case the unit of information is a "bit." In this paper we adopt natural logarithm, in which case the unit is called a "nat."

Equation (124) follows from (115) and (116). Instead of choosing $\{C_t\}$ and $\{V_t\}$ directly, we view problem (123) as a standard optimal control problem with $\Sigma_{t|t}$ as the state variable and Φ_t as the control variable. After obtaining a solution for $\{\Phi_t\}_{t=0}^T$ to this problem, we can recover $\{C_t\}$ and $\{V_t\}$ from the following result.

Proposition 98 *Given an optimal sequence $\{\Phi_t\}_{t=0}^T$ determined from problem (123), an optimal information structure $\{C_t, V_t\}_{t=0}^T$ satisfies $\Phi_t = C_t' V_t^{-1} C_t$. A particular solution is that $V_t = diag\left(\varphi_{it}^{-1}\right)_{i=1}^{m_t}$ and the m_t columns of $n_x \times m_t$ matrix C_t' are orthonormal eigenvectors for all positive eigenvalues of Φ_t, denoted by $\{\varphi_{it}\}_{i=1}^{m_t}$. The optimal dimension of the signal vector s_t is equal to $rank\,(\Phi_t) = m_t \leq n_x$.*

This proposition shows that the optimal information structure $\{C_t, V_t\}_{t=0}^T$ is not unique and can be computed by the singular-value decomposition. The optimal signal can always be constructed such that the components in the noise vector v_t of the signal s_t are independent. However, the optimal signal components are in general not independent in the sense that the matrix C_t may not be diagonal or invertible.

The next step is to transform the problem into one that permits the use of semi-definite programming tools, which we discussed earlier.

9.8 Finite-Horizon Problems

We follow the procedure in Tanaka et al (2017) closely, which consists of the following three steps. The first step is to transform (123) into an optimization problem in terms of $\left\{\Sigma_{t|t}\right\}$ only. We eliminate the control $\{\Phi_t\}$ and replace (124) and (125) by linear inequality constraints

$$0 \prec \Sigma_{t|t} \preceq A_{t-1}\Sigma_{t-1|t-1}A_{t-1}' + W_{t-1},\ t = 1, 2, ..., T, \tag{127}$$

and

$$0 \prec \Sigma_{0|0} \preceq \Sigma_0. \tag{128}$$

Equations (127) and (128) give the no-forgetting constraints discussed by Sims (2003, 2011). Intuitively, observing signals s_t over time will reduce uncertainty about the state. Sims (2003, 2011) recommends to use a Cholesky decomposition as the new choice variable to handle the

no-forgetting constraints for the infinite-horizon model, but the resulting problem is often rather hard to solve.

Now we obtain the following equivalent problem

$$\min_{\{\Sigma_{t|t}\}} \left\{ \frac{1}{2}\log\left(\det\left(\Sigma_0\right)\right) - \frac{1}{2}\log\left(\det\left(\Sigma_{0|0}\right)\right) + \frac{1}{2}\sum_{t=1}^{T}\log\left(\det\left(A_{t-1}\Sigma_{t-1|t-1}A_{t-1}^T + W_{t-1}\right)\right) - \frac{1}{2}\log\left(\det\left(\Sigma_{t|t}\right)\right) \right\} \tag{129}$$

subject to (126), (127), and (128). The eliminated variable Φ_t can be recovered through

$$\Phi_t = \Sigma_{t|t}^{-1} - \left(A_{t-1}\Sigma_{t-1|t-1}A_{t-1}^T + W_{t-1}\right)^{-1},\ t = 1, 2, ..., T, \tag{130}$$

and

$$\Phi_0 = \Sigma_{0|0}^{-1} - \Sigma_0^{-1}. \tag{131}$$

The second step is to rewrite the objective function in (129) by regrouping terms as a sum of the initial cost $\frac{1}{2}\log\det\left(\Sigma_0\right)$, the final cost $\frac{1}{2}\log\left(\det\left(\Sigma_{T|T}\right)\right)$, and period costs

$$\frac{1}{2}\log\det\left(A_t\Sigma_{t|t}A_t' + W_t\right) - \frac{1}{2}\log\det\left(\Sigma_{t|t}\right),$$

for $t = 0, 1, ..., T-1$. The matrix determinant lemma (Theorem 18.1.1 in Harville 1997) implies that the preceding expression is equal to

$$\frac{1}{2}\log\left(\det\left(W_t\right)\right) - \frac{1}{2}\log\left(\det\left(\Sigma_{t|t}^{-1} + A_t'W_t^{-1}A_t\right)^{-1}\right).$$

Due to the monotonicity of the determinant function, this expression is equal to the optimal value of

$$\min_{\Pi_t}\left\{\frac{1}{2}\log\left(\det\left(W_t\right)\right) - \frac{1}{2}\log\left(\det\left(\Pi_t\right)\right)\right\}$$

subject to

$$0 \prec \Pi_t \preceq \left(\Sigma_{t|t}^{-1} + A_t^TW_t^{-1}A_t\right)^{-1}. \tag{132}$$

In the final step we apply the matrix inversion formula to rewrite (132) as

$$0 \prec \Pi_t \preceq \Sigma_{t|t} - \Sigma_{t|t}A_t^T\left(W_t + A_t\Sigma_{t|t}A_t^T\right)^{-1}A_t\Sigma_{t|t},$$

which is equivalent to

$$\begin{bmatrix} \Sigma_{t|t} - \Pi_t & \Sigma_{t|t}A_t^T \\ A_t\Sigma_{t|t} & W_t + A_t\Sigma_{t|t}A_t^T \end{bmatrix} \succeq 0$$

$$\Pi_t \succ 0,$$

by exploiting the Schur complement. Note that this is a linear matrix inequality condition. We summarize the analysis above in the following result.

Proposition 99 *Suppose that $W_t \succ 0$ and $\Omega_t \succeq 0$, for $t = 0, 1, ..., T$. Then the optimal information structure $\{C_t, V_t\}_{t=0}^T$ for problem (121) or (123) can be constructed by solving the following determinant maximization problem with decision variables $\left\{\Sigma_{t|t}, \Pi_t\right\}_{t=0}^T$:*

$$\kappa_T(D) = \min_{\left\{\Sigma_{t|t}, \Pi_t\right\}_{t=0}^T} \left\{ -\sum_{t=0}^{T} \frac{1}{2} \log\det \Pi_t + c \right\} \tag{133}$$

subject to (126), (127), (128),

$$\begin{bmatrix} \Sigma_{t|t} - \Pi_t & \Sigma_{t|t} A_t^T \\ A_t \Sigma_{t|t} & W_t + A_t \Sigma_{t|t} A_t^T \end{bmatrix} \succeq 0, \quad t = 0, 1, ..., T-1, \tag{134}$$

$$\Sigma_{T|T} = \Pi_T, \quad \Pi_t \succ 0, \ t = 0, 1, ..., T, \tag{135}$$

where

$$c = \frac{1}{2} \log\left(\det\left(\Sigma_0\right)\right) + \sum_{t=1}^{T} \frac{1}{2} \log\left(\det\left(W_{t-1}\right)\right).$$

The optimal sequence $\{\Phi_t\}_{t=0}^T$ is obtained from (130) and (131). The optimal information structure $\{C_t, V_t\}_{t=0}^T$ is given in Proposition 98.

The key difference between the problem above and that in Tanaka et al (2017) is that we have a single distortion constraint in (122) or (126), while they have a sequence of distortion constraints. The constraint (128) is also absent in their optimization problem. Moreover, our distortion constraint is derived from a LQG control problem, while their constraints are imposed exogenously from information theory. Following their arguments, we can show that there always exists an optimal solution to the problem in Proposition 99. Moreover, it is a strictly convex optimization problem and hence the optimal solution is unique.

The assumption of $W \succ 0$ can be restrictive in economic applications. This assumption implies that there must be a nontrivial random shock to each state transition equation (100). It is possible that there is no random shock to the state transition equation for some state variables. For example, we typically assume that the capital stock k_t follows the law of motion $k_{t+1} = (1-\delta) k_t + I_t$, where $\delta > 0$ denotes the depreciation rate and I_t denotes investment. To get

around this issue, one may introduce a depreciation or capital destruction shock, or one can allow $W \succeq 0$ and derive a result similar to Proposition 99 under the assumptiont that A is invertible. This assumption can also be restrictive. For example, it rules out the case in which an iid shock is used as a state variable.

For the relaxed problem under RI in Problem 93, we can use a similar three-step procedure. In particular, the optimal control is given by $u_t = -F_t \widehat{x}_t$ in step 1 by the certainty equivalence principle. In step 2 we derive the relaxed tracking problem under RI in which the information-flow constraint is removed. In step 3 we use the semidefinite programming approach to transform this problem into the following determinant maximization problem:

$$\min_{\{\Sigma_{t|t}, \Pi_t\}_{t=0}^{T}} \left\{ \sum_{t=0}^{T} \operatorname{trace}\left(\Omega_t \Sigma_{t|t}\right) - \lambda \sum_{t=0}^{T} \frac{1}{2} \log\left(\det\left(\Pi_t\right)\right) + \ \lambda c \right\}$$

subject to (127), (128), (134), and (135). Using the solution to this problem and equations (130) and (131), we determine the optimal sequence $\{\Phi_t\}_{t=0}^{T}$. Proposition 98 gives the optimal information structure $\{C_t, V_t\}_{t=0}^{T}$.

We apply the software package SDPT3, version 4.0, which can solve semidefinite programming problems up to 100 dimensions efficiently and robustly (Toh, Todd, and Tutuncu 1999 and Tutuncu, Toh, and Todd 2003). As we noted in the section on SDPs, some tricks are required to get optimization problems into the form needed by the solvers; here, I will go through the steps needed to apply this particular solver. The required form is

$$\max_{y \in \mathcal{R}^m,\ Z_j^s,\ Z^l} \left\{ b^T y + \sum_{j=1}^{n_s} \nu_j^s \log\left(\det\left(Z_j^s\right)\right) \right\} \tag{136}$$

subject to

$$\left(\mathcal{A}_j^s\right)^T y + Z_j^s = c_j^s, \quad Z_j^s \in K_s^{s_j}, \quad j = 1, 2, ..., n_s, \tag{137}$$

$$\left(A^l\right)^T y + Z^l = c^l, \quad Z_l \in \mathcal{R}_+^{n_l}, \tag{138}$$

where constraints (137) and (138) correspond to the linear matrix inequality and the linear vector inequality in applications.

The choice variables are an m-dimensional real vector y, an s_j-dimensional positive semidefinite matrix Z_j^s, and an n_l-dimensional nonnegative vector Z^l. The set $K_s^{s_j}$ denotes the cone of

positive semidefinite symmetric matrices of dimension s_j. In our application, Z_j^s and Z^l are slack variables that transform the matrix inequality constraints and linear inequalities into equalities. All other variables are exogenous input parameters. In particular, b is an m-dimensional vector, $\nu_s^j \geq 0$ for all j, c_j^s is an s_j-dimensional matrix, A^l is an m by n_l dimensional matrix, and c^l is an n_l-dimensional vector.

Let $\mathcal{S}^n$ denote the set of all symmetric matrices. Define a vectorization operator for a symmetric matrix $X \in \mathcal{S}^n$ as $\mathbf{svec} : \mathcal{S}^n \longmapsto \mathbb{R}^{n(n+1)/2}$,

$$\mathbf{svec}(X) = \left[X_{11}, \sqrt{2}X_{12}, X_{22}, \sqrt{2}X_{13}, ..., \sqrt{2}X_{1n}, ..., \sqrt{2}X_{n-1,n}, X_{n,n}\right]^T,$$

where the $\sqrt{2}$ is to make the operation isometric. We use the Matlab notation $[U; V]$ to denote the matrix obtained by appending V below the last row of U. Then we identfiy $\mathcal{A}_j^s$ with the following m by the $s_j\left(s_j+1\right)/2$ matrix

$$A_j^s = \left[\mathbf{svec}\left(a_{j,1}^s\right)^T; \mathbf{svec}\left(a_{j,2}^s\right)^T; ....; \ \mathbf{svec}\left(a_{j,m}^s\right)^T\right],$$

where $a_{j,1}^s, ..., a_{j,m}^s$ are model specific input symmetric coefficient matrices of dimension s_j associated with each element in y. The operation $\left(\mathcal{A}_j^s\right)^T y$ is then defined as

$$\left(\mathcal{A}_j^s\right)^T y = \sum_{k=1}^{m} a_{j,k}^s y_k.$$

To use the software SDPT3, we need to transform our optimization problem in Proposition **??** into the form in (136), (137), and (138) by the following four steps.

Step 1. Set the choice variable $y = [\mathbf{svec}\left(\Sigma\right); \mathbf{svec}\left(\Pi\right)]$ so that $m = n_x\left(n_x+1\right)$. The other choice variables are the slack variables Z_1^s, Z_2^s, and Z_3^s defined next. Set $s_j = n_x$.

Step 2. To derive the constraint (138), we consider (**??**) and write

$$\mathrm{tr}\left(\Omega\Sigma\right) = \left(\mathbf{svec}\left(\Omega\right)\right)^T \mathbf{svec}\left(\Sigma\right).$$

Set $n_l = 1$, $A^l = [\mathbf{svec}(\Omega); \mathbf{0}]$, and $c^l = D$. Introduce $Z^l \geq 0$. Then (**??**) is in the form of (138).

Step 3. Derive the constraint (137). There are three linear matrix inequalities in our optimization problem. Set $n_s = 3$. We introduce three slack variables Z_1^s, Z_2^s, and Z_3^s. First, write the semidefinite constraint $\Pi \succ 0$ as

$$-\Pi + Z_1^s = 0, \ \ Z_1^s \in K_s^{n_x}.$$

and

$$-\Pi = \sum_{k=1}^{m} a_{1,k}^{s} y_k,$$

where $a_{1,k}^{s}$, $k = 1, ..., m$, are coefficient matrices. To understand the last equation, consider the simple 2 by 2 case:

$$-\Pi = -\begin{bmatrix} \pi_{12} & \pi_{12} \\ \pi_{12} & \pi_{22} \end{bmatrix} = -\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \pi_{11} - \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \left(\sqrt{2}\pi_{12} \right) - \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \pi_{22},$$

where $\left[\pi_{11}, \sqrt{2}\pi_{12}, \pi_{22}\right]^T$ are in the vector y. Set $c_1^s = \mathbf{0}$.

Second, we can similarly write the no-forgetting constraint (**??**) as

$$\Sigma - A\Sigma A' + Z_2^s = W, \;\; Z_2^s \in K_s^{n_x},$$

and

$$\Sigma - A\Sigma A' = \sum_{k=1}^{m} a_{2,k}^{s} y_k,$$

where $a_{2,k}^{s}$, $k = 1, ..., m$, are coefficient matrices. Set $c_2^s = W$.

Third, we write (**??**) as

$$\begin{bmatrix} \Pi & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \Sigma & \Sigma A^T \\ A\Sigma & A\Sigma A^T + W \end{bmatrix} + Z_3^s = \begin{bmatrix} 0 & 0 \\ 0 & W \end{bmatrix}, \;\; Z_3^s \in K_s^{2n_x},$$

and

$$\begin{bmatrix} \Pi & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \Sigma & \Sigma A^T \\ A\Sigma & A\Sigma A^T + W \end{bmatrix} = \sum_{k=1}^{m} a_{3,k}^{s} y_k,$$

where $a_{3,k}^{s}$, $k = 1, 2, ..., m$, are coefficient matrices. Set

$$c_3^s = \begin{bmatrix} 0 & 0 \\ 0 & W \end{bmatrix}.$$

Step 4. Set input parameters in the objective function (136) as $b = \mathbf{0}$, $\nu_1^s = 0.5$, $\nu_2^s = \nu_3^s = 0$. After solving for optimal $y = [\mathbf{svec}(\Sigma); \mathbf{svec}(\Pi)]$, we use the inverse operator of **svec**, **smat**, to obtain optimal Σ and Π. Since $\Pi = Z_1^s$ enters $\log\det$ in the objective function, optimal $\Pi = Z_1^s$ will be positive definite because the singular case will lead the objective function to approach negative infinity. After solving for the optimal Π, we obtain the rate distortion function

$$\kappa(D) = -\frac{1}{2}\log\left(\det(\Pi)\right) + \frac{1}{2}\log\left(\det(W)\right).$$

10 Projection Methods

Projection methods exploit the "projection theorem" to construct the best approximation to an unknown function from a particular class of functions that are known to be close in some sense. The key theorem is well known – you use it all the time in least-squares regression.

Definition 100 *$(X, \langle \cdot, \cdot \rangle)$ is a **Hilbert space** if $\langle \cdot, \cdot \rangle$ is an inner product and X is a Banach space (a complete metric space) with metric equal to the norm induced by the inner product.*

Theorem 101 *(Projection Theorem) Let H be a Hilbert space and $G \subset H$ be a Hilbert subspace. If $x \in H$, there exists a unique element $\widehat{x} \in G$ (called the projection of x onto G) such that*

$$\|x - \widehat{x}\| = \inf_{y \in G} \{\|x - y\|\}$$

and $x - \widehat{x}$ is orthogonal to all elements of G. If $x \in G$, then $\widehat{x} = x$.

To work through a simple example, let's take ordinary least-squares regressions

$$y_t = \alpha + \beta x_t + \epsilon_t.$$

We are "projecting" the unknown data generating process for y_t onto the space of affine functions of x_t, with ϵ_t being the approximation error. The projection coefficients (α, β) minimize the Euclidean norm

$$\|y - \widehat{y}\|_2 = \sum_{t=1}^{T} (y_t - \alpha - \beta x_t)^2$$

and satisfy the normal equations

$$\left(X^T X\right) \widehat{\beta} = X^T y$$

where

$$X = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_T \end{bmatrix}$$

$$y = \begin{bmatrix} y_1 \\ \vdots \\ y_T \end{bmatrix}$$

$$\widehat{\beta} = \begin{bmatrix} \widehat{\alpha} \\ \widehat{\beta} \end{bmatrix}.$$

We can invert the matrix to obtain an explicit solution provided the covariance matrix is not singular:

$$\widehat{\beta} = \left(X^T X\right)^{-1} X^T y.$$

We can use the Euclidean L_2 norm of the error vector as a measure of our accuracy:

$$\|\epsilon\|_2 = \sqrt{\sum_{t=1}^{T} \left(y_t - \widehat{\alpha} - \widehat{\beta} x_t\right)^2}.$$

We can also use the L_∞ norm

$$\|\epsilon\|_\infty = \max_t \left\{ \left| y_t - \widehat{\alpha} - \widehat{\beta} x_t \right| \right\}.$$

Each of these represent different requirements for our projections; the first requires the approximation be good "on average" while the second requires it to be good "everywhere".

We will consider a number of different options for the target space of functions which are finitely parameterized and therefore can be represented on the computer.

10.1 Polynomial Approximation

Polynomial approximation uses polynomials to fit the function

$$f(x) = \sum_{i=0}^{n} a_i x^i \tag{139}$$

for some $n > 0$. This procedure can be quickly implemented by linear least squares to obtain the coefficients a_i, given the values of x that you use for the projection. Note that evaluating the

polynomial directly is inefficient, since you repeatedly recompute powers of x; instead, Horner's method is preferred:

$$f(x) = a_0 + x(a_1 + x(a_2 + ...)),$$

which can be computed recursively as

$$f^k = a_k + xf^{k-1}.$$

Unfortunately, Horner's method is sequential rather than parallel, so we generally prefer to use Estrin's scheme, which rewrites the polynomial into one in higher powers of x. Here are the steps:

1. For $i \in \{0, ..., \lfloor n/2 \rfloor\}$, compute $b_i = a_{2i} + a_{2i+1}x$ and $y = x^2$; if n is odd, $b_{n/2} = a_n$;
2. Evaluate $g(y) = \sum_{i=0}^{\lfloor n/2 \rfloor} b_i y^i$.

While we do more operations in Estrin's scheme than Horner's method, we can do Step 1 in parallel.

If you are not interested in the coefficients but only want to evaluate the function you can use Neville's algorithm to compute the values of the unique $n+1$-order polynomial that passes through n points; if you intend to evaluate the same polynomial many times this approach is not efficient. Neville's algorithm is recursive. Let $p_{i,j}$ denote the polynomial of degree $j-i$ that passes through the points (x_k, y_k) for $k = i, i+1, ..., i+j$. Then

$$\begin{aligned} p_{i,i}(x) &= y_i \qquad 0 \leq i \leq n \\ p_{i,j}(x) &= \frac{(x - x_j) p_{i,j-1}(x) - (x - x_i) p_{i+1,j}(x)}{x_i - x_j} \qquad 0 \leq i < j \leq n. \end{aligned}$$

Continuing until $p_{0,n}(x)$ gives us the value of the polynomial at x.

If instead we want the coefficients, we need to construct them using the projection theorem. The Stone-Weierstrass theorem is the theoretical foundation of polynomial approximation.

Theorem 102 *(Stone-Weierstrass Theorem) Let (X, ρ) be a compact metric space and $C(X)$ be the space of continuous functions $f : X \to \mathcal{R}$ endowed with the* sup*-norm. Let $A \subseteq C(X)$ be such that (i) A contains all the constant functions on X; (ii) A is an* ***algebra*** *(if $f, g \in A$ then $f + g \in A$ and $fg \in A$); (iii) A* ***separates points*** *($\forall x, y \in X$ and $x \neq y$, $\exists f \in A$ such that*

$f(x) \neq f(y)$). *Then the closure of* $A = C(X)$ *(*A *is* ***dense*** *in* $C(X)$*, meaning* $\forall \epsilon > 0$ *and* $\forall f \in A$, $\exists g \in A$ *such that* $||f - g|| < \epsilon$*).*

It is straightforward to see that polynomials are an algebra (the sum and product of polynomials are polynomials), they separate points (just take the linear polynomial $f(z) = az$ for $a \neq 0$), and they contain the constant functions. Therefore, the polynomials are dense in the space of continuous functions over a compact space, so that any continuous function is arbitrarily well approximated by a polynomial; the order of that polynomial may be prohibitively large, however, especially if the function has points of non-differentiability (kinks or cusps).

Note that the Stone-Weierstrass theorem does not say that increasing the order of a polynomial leads necessarily to a better approximation, particularly for a fixed set of points. For example, consider attempting to approximate f using the polynomial constructed via Neville's method. **Runge's phenomenon** occurs if this polynomial diverges from f as the number of points goes to infinity, with the typical divergence occuring as uncontrolled oscillations near the boundary of X. To see an example, consider the Witch of Agnesi function

$$f(x) = \frac{1}{1 + 25x^2}.$$

If we attempt to interpolate this function at the $n + 1$ equidistant points on $[-1, 1]$ given by

$$x_i = \frac{2i}{n} - 1$$

for $i \in \{0, 1, ..., n\}$, then

$$\lim_{n \to \infty} \left\{ \max_{-1 \leq x \leq 1} \{|f(x) - p_n(x)|\} \right\} = \infty.$$

The key to avoiding Runge's phenomenon is to choose the nodes carefully; we will address that concern after discussing the appropriate choice of basis for the approximating polynomials.

The class of polynomials used in the constructive proof of the Stone-Weierstrass theorem are called the Bernstein polynomials.

Definition 103 *Define* $l_{n,m}(x) = \binom{n}{m} x^m (1 - x)^{n-m}$, $0 \leq m \leq n$. *The* n*th* ***Bernstein polynomial*** *of* $f(x)$ *on* $(0, 1)$ *is*

$$B_n(x; f) = \sum_{m=0}^{n} f\left(\frac{m}{n}\right) l_{n,m}(x).$$

$B_n(x)$ *has order at most* n*.*

One can show that $\{B_n(x)\}$ converges uniformly to $f(x)$ as $n \to \infty$. Note that the n-th order Bernstein polynomial approximation of a polynomial function may not be itself:

$$\begin{aligned} B_2\left(x; x^2\right) &= \sum_{m=0}^{2} \left(\frac{m}{2}\right)^2 \binom{2}{m} x^m (1-x)^{2-m} \\ &= 0 + \left(\frac{1}{2}\right)^2 \binom{2}{1} x(1-x) + \binom{2}{2} x^2 \\ &= \frac{1}{2} x(1+x) \\ &\neq x^2. \end{aligned}$$

We can continue to higher-order polynomials:

$$\begin{aligned} B_3\left(x; x^2\right) &= \sum_{m=0}^{3} \left(\frac{m}{3}\right)^2 \binom{3}{m} x^m (1-x)^{3-m} \\ &= \frac{1}{3} x(1+2x) \\ B_4\left(x; x^2\right) &= \sum_{m=0}^{4} \left(\frac{m}{4}\right)^2 \binom{4}{m} x^m (1-x)^{4-m} \\ &= \frac{1}{4} x(1+3x). \end{aligned}$$

The pattern is

$$B_n\left(x; x^2\right) = \frac{1}{n} x(1+(n-1)x).$$

As $n \to \infty$ we have

$$\lim_{n\to\infty} \left\{ \frac{x(1+(n-1)x)}{n} \right\} = x^2.$$

Thus, we do recover the correct function in the limit, but it is obviously inefficient to approximate a quadratic polynomial with a different quadratic polynomial.

Clearly, the Bernstein polynomials are not necessarily the best approximation of a given order; all we know is that, if n is sufficiently large, the Bernstein polynomial is arbitrarily close to f. In fact, the convergence rate is generally very slow:

$$|B_n(x; f) - f| \leq \frac{8306 + 837\sqrt{6}}{5832} \omega\left(n^{-\frac{1}{2}}\right)$$

where the modulus of continuity ω means that the condition

$$|f(x) - f(y)| \leq \omega |x - y|$$

holds $\forall x, y$ (this condition is a bound on the slope of the function); therefore, Bernstein polynomials converge at rate $\sqrt{n^{-1}}$, which is rather poor. However, Bernstein polynomials will preserve the shape of a function – if f is increasing and/or concave, then so is B. Furthermore, they are easy to evaluate using de Casteljau's recursion. For a Bernstein polynomial of the form

$$B(x) = \sum_{i=0}^{n} \beta_i b_{i,n}(x)$$
$$b_{i,n}(x) = \binom{n}{i} (1-x)^{n-i} x^i,$$

we compute

$$\beta_i^{(0)} = \beta_i \quad i \in \{0, 1, ..., n\}$$
$$\beta_i^{(j)} = \beta_i^{(j-1)} (1-x) + \beta_{i+1}^{(j-1)} x \quad i \in \{0, ..., n-j\}, j \in \{1, ..., n\}.$$

The value is $B(x) = \beta_0^{(n)}$; this procedure requires $\binom{n}{2}$ operations. Using Bernstein polynomials therefore involves a tradeoff between shape-preservation, a property not shared by other polynomials in general, and the slow rate of convergence. Given that we typically want use relatively low-order polynomials to conserve on unknowns, the slow convergence is a bigger problem and we should look for other polynomial families.

To begin thinking about accuracy (what is the best polynomial of a given order?), we need to define the subspace of polynomials we plan to work in. In the space of polynomials, the monomial basis functions $\left\{1, x, x^2, ...\right\}$ are not mutually orthogonal: all of these polynomials are monotone increasing in the positive reals and, on the negative reals, they are all either monotone increasing or monotone decreasing. That is, these polynomials are "multicollinear" which, as you know, is really bad for projections (in empirical work, it leads to large standard errors; in numerical work, it leads to inaccurate calculations). We therefore will use a different basis, one that has nice orthogonality properties. Think about running a regression on orthogonal regressors; the matrix $X^T X$ will be diagonal and easy to invert accurately. Constructing an orthogonal basis is our first goal.

Definition 104 *A **weighting function** $w(x)$ on $[a, b]$ is any function which is positive almost everywhere and satisfies*

$$\int_a^b w(x)\, dx < \infty. \tag{140}$$

Definition 105 *An **inner product** with respect to $w(x)$ is a operator $\langle \cdot, \cdot \rangle$ such that*

$$\langle f, g \rangle = \int_a^b f(x) g(x) w(x) dx. \tag{141}$$

Definition 106 *A family of polynomials $\{\phi_n(x)\}$ is **mutually orthogonal** with respect to weighting function $w(x)$ if*

$$\langle \phi_n(x), \phi_m(x) \rangle = 0 \tag{142}$$

$\forall n \neq m$.

Definition 107 *A family of polynomials $\{\phi_n(x)\}$ is **mutually orthonormal** if they are mutually orthogonal and satisfy*

$$\langle \phi_n(x), \phi_n(x) \rangle = 1 \tag{143}$$

$\forall n$.

There are five classes of orthogonal polynomials that we will find useful: Legendre, Chebyshev, Laguerre, Hermite, and Jacobi. We have seen these before (although they were hidden) in the Gaussian quadrature rules for integration; for example Gauss-Hermite quadrature uses the Hermite polynomial class to approximate the integrand and obtain the resulting nodes and weights. Table 1 presents these polynomials and their respective domains and weighting functions.

Table 1

Orthogonal Families

Family	$w(x)$	$[a, b]$	Definition
Legendre	1	$[-1, 1]$	$P_n(x) = \frac{(-1)^n}{2^n n!} \frac{d^n}{dx^n} \left[\left(1 - x^2\right)^n \right]$
Chebyshev	$\left(1 - x^2\right)^{-\frac{1}{2}}$	$[-1, 1]$	$T_n(x) = \cos(n \arccos(x))$
Laguerre	e^{-x}	$[0, \infty)$	$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x})$
Hermite	e^{-x^2}	$(-\infty, \infty)$	$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} \left(e^{-x^2}\right)$
Jacobi	$(1 - x)^\alpha (1 + x)^\beta$	$[-1, 1]$	$J_{n+1}(x) = \frac{d_n + e_n x}{c_n} J_n(x) - \frac{f_n}{c_n} J_{n-1}(x)$

where

$$\begin{aligned}
c_j &= 2\left(j+1\right)\left(j+\alpha+\beta+1\right)\left(2j+\alpha+\beta\right) \\
d_j &= \left(2j+\alpha+\beta+1\right)\left(\alpha^2-\beta^2\right) \\
e_j &= \left(2j+\alpha+\beta\right)\left(2j+\alpha+\beta+1\right)\left(2j+\alpha+\beta+2\right) \\
f_j &= 2\left(j+\alpha\right)\left(j+\beta\right)\left(2j+\alpha+\beta+2\right)
\end{aligned}$$

are the Jacobi coefficients. Admittedly some of these do not resemble polynomials, but each admits a recursive definition that makes it clear they are in fact polynomial expressions. First, we have

$$P_0\left(x\right) = T_0\left(x\right) = L_0\left(x\right) = H_0\left(x\right) = J_0\left(x\right) = 1. \tag{144}$$

Next, we have

$$\begin{aligned}
P_1\left(x\right) &= T_1\left(x\right) = x \\
L_1\left(x\right) &= 1-x \\
H_1\left(x\right) &= 2x \\
J_1\left(x\right) &= \frac{\left(2+\alpha+\beta+1\right)\left(\alpha^2-\beta^2\right)+\left(2+\alpha+\beta\right)\left(2+\alpha+\beta+1\right)\left(2+\alpha+\beta+2\right)x}{2\left(1+1\right)\left(1+\alpha+\beta+1\right)\left(2+\alpha+\beta\right)}.
\end{aligned}$$

Then, for any n, the polynomials obey

$$\begin{aligned}
P_{n+1}\left(x\right) &= \frac{2n+1}{n+1}xP_n\left(x\right) - \frac{n}{n+1}P_{n-1}\left(x\right) \\
T_{n+1}\left(x\right) &= 2xT_n\left(x\right) - T_{n-1}\left(x\right) \\
L_{n+1}\left(x\right) &= \frac{1}{n+1}\left(2n+1-x\right)L_n\left(x\right) - \frac{n}{n+1}L_{n-1}\left(x\right) \\
H_{n+1}\left(x\right) &= 2xH_n\left(x\right) - 2nH_{n-1}\left(x\right) \\
J_{n+1}\left(x\right) &= \frac{\left(d_n+e_nx\right)}{c_n}J_n\left(x\right) - \frac{f_n}{c_n}J_{n-1}\left(x\right).
\end{aligned}$$

As an example, take the Chebyshev polynomials:

$$\begin{aligned}
T_0\left(x\right) &= 1 \\
T_1\left(x\right) &= x \\
T_2\left(x\right) &= 2x^2-1 \\
T_3\left(x\right) &= 4x^3-3x.
\end{aligned}$$

They are mutually orthogonal on $[-1,1]$ with respect to their weighting function:

$$\int_{-1}^{1} x\left(2x^2-1\right)\left(1-x^2\right)^{-\frac{1}{2}}dx=0$$
$$\int_{-1}^{1}\left(2x^2-1\right)\left(4x^3-4x\right)\left(1-x^2\right)^{-\frac{1}{2}}dx=0.$$

Note that the derivatives of the Chebyshev polynomials also follow a recursive structure:

$$\begin{aligned}DT_0(x)&=0\\DT_1(x)&=1\\DT_{n+1}(x)&=2T_n(x)+2xDT_n(x)-DT_{n-1}(x)\end{aligned}$$

and

$$\begin{aligned}D^2T_0(x)&=0\\D^2T_1(x)&=0\\D^2T_{n+1}(x)&=4DT_n(x)+2xD^2T_n(x)-D^2T_{n-1}(x).\end{aligned}$$

We get

$$\begin{aligned}DT_2(x)&=2x+2x=4x\\DT_3(x)&=2\left(2x^2-1\right)+2x(4x)-1=12x^2-3\\DT_2(x)&=4\\DT_3(x)&=4(4x)+2x(4)=24x.\end{aligned}$$

Evaluating orthogonal polynomials is easy given their recursive structure. Let the function be represented by n Chebyshev polynomials according to

$$f(x)=\sum_{i=0}^{n}a_iT_i(x);\tag{145}$$

the a_i are scalar coefficients which are known. To do so, you nest two loops and keep track of two running sums. One sum is the next polynomial to be evaluated; the other is the function itself.

1. Set $T(0)=1$ and $T(1)=x$; the current sum is $\Sigma=a_0+a_1x$;

2. Compute $T(2) = 2xT(1) - T(0)$; the current sum is $\Sigma = \Sigma + a_2 T(2)$;

3. For each n compute $T(n) = 2xT(n-1) - T(n-2)$; the current sum becomes $\Sigma = \Sigma + a_n T(n)$.

Similar tricks can be employed for any of the families of orthogonal polynomials using their recursive formulae.

The next question is how to choose between the classes of polynomials – which one works best? To begin, we must consider what we mean by "work well." One way to determine our fit is the L^2 norm:

$$\min_{\deg(p) \leq n} \left\{ \int_a^b (f(x) - p(x))^2 w(x) \, dx \right\}. \tag{146}$$

This norm is global in that it measures the average deviation; however, it says nothing about the approximation at any given x. If instead we wish to approximate $f(x)$ uniformly well, we are using the stronger requirement that

$$\lim_{n \to \infty} \left\{ \max_{x \in [a,b]} |f(x) - p(x)| \right\} = 0 \tag{147}$$

for some sequence $p_n(x)$. Uniform approximations are good at every x; L^2 approximations are good only on average. The Stone-Weierstrass theorem assures us that a sequence that works uniformly exists; we are left with how to find it.

Definition 108 *The **minmax polynomial** is the polynomial of fixed degree n which solves*

$$\inf_{\deg(p) \leq n} \{\|f - p\|_\infty\}. \tag{148}$$

Minmax polynomials are by definition the best uniform approximators of a given degree within the class of polynomials. In general, finding the minmax polynomial is quite difficult since we must minimize at every point in the state space. It is possible to construct low-order minmax polynomials for some cases. For example, suppose we are trying to approximate the function $f(x) = \exp(x)$ on $[1, 3]$; the minmax polynomial of degree $p \leq 1$ is

$$p_1^*(x) = mx + \frac{e - m \log(m)}{2}$$

where

$$m = \frac{e^3 - e}{2}.$$

This approximation is not particularly good: $\max_{x\in[1,3]}\{|f(x)-p_1^*(x)|\}\approx 2$.

The algorithm for computing the uniformly-best polynomial approximator is called the Remez exchange algorithm (Remez 1934). To use this algorithm, suppose our goal is to solve

$$\min_p\{\|f-p\|_\infty\}$$

using some polynomial p. To begin, first solve the linear system

$$b_0+b_1x_i+\cdots+b_nx_i^n+(-1)^iE=f(x_i)$$

for $(b_0,...,b_n,E)$ at some set of points $X=(x_i)_{i=1}^{n+2}$. Using the b coefficients, construct a polynomial

$$P_n(x)=b_0+b_1x+\cdots+b_nx^n.$$

Next, find all the M local maxima of the error

$$e(x)=|P_n(x)-f(x)|;$$

if the local maxima are of equal size and alternate in sign, then P_n is the minmax polynomial approximation of f. Otherwise, replace X with M and repeat. A good initial X are the zeros of the $n+2$th order Chebyshev polynomial. The convergence criterion is typically

$$\max\{|P_n(x_i)-f(x_i)|\}-\min\{|P_n(x)-f(x)|\}<\epsilon.$$

The challenge here is finding all the local extrema of the error function, which can be difficult even in low dimensional problems (it is not a polynomial and therefore we do not even know how many such extrema there are).

Fortunately, we have the following result using Chebyshev polynomials:

Theorem 109 *If $T_n(x)$ is the nth degree Chebyshev L^2 approximation to $f\in C^1[-1,1]$, then*

$$\inf_{\deg(p)\leq n}\{\|f-p\|_\infty\}\leq\|f-T_n\|_\infty\leq\left(4+\frac{4}{\pi^2}\ln(n)\right)\inf_{\deg(p)\leq n}\{\|f-p\|_\infty\}. \tag{149}$$

That is, least-squares Chebyshev approximation is almost as good as minmax approximation and obviously much easier to compute. As $n\to\infty$ the LHS goes to zero by application of the

Stone-Weierstrass theorem. It turns out that the RHS also goes to zero, despite the presence of the $\ln(n)$ term, because the minmax error goes to zero faster:

$$\inf_{\deg(p)\leq n}\{\|f-p\|_\infty\} \leq \frac{(n-k)!}{n!}\left(\frac{\pi}{2}\right)^k\left(\frac{b-a}{2}\right)^k\left\|D^k f\right\|_\infty$$

where $f \in C^k$. Note that for $k=1$, we have convergence at rate n^{-1}, since n grows faster than $\ln(n)$:

$$\lim_{n\to\infty}\left\{\frac{\ln(n)}{n}\right\} = \lim_{n\to\infty}\left\{\frac{1}{n}\right\} = 0.$$

Smoother functions will have even faster convergence: given $f \in C^k$, the convergence rate is $\frac{1}{n(n-1)\cdots(n-k)}$. Clearly, this rate is much faster than for Bernstein polynomials, and the Stone-Weierstrass theorem applies to any polynomial since we can simply rearrange the terms to create a Bernstein polynomial.

To avoid Runge's phenomenon, we use the zeros of the Chebyshev polynomials as the interpolation nodes; these points are given by

$$z_i = -\cos\left(\frac{2i-1}{2m}\pi\right) \in (-1,1).$$

These points are not equidistant, but rather are bunched up near the endpoints of the interval; as a result, we control more effectively the oscillations that generate divergence. Note that extrapolation – evaluating the function outside the interval $(-1,1)$ – is hazardous, as polynomials diverge and Chebyshev polynomials diverge particularly rapidly; that divergence is the price paid for controlling oscillations on $(-1,1)$.

Are these points the best? Actually, no, we can do a little better at essentially no cost: the linear transformation

$$s_i = \frac{z_i}{\cos\left(\frac{\pi}{2(n+1)}\right)}$$

reduces the approximation error by a small amount. These nodes are actually optimal if the underlying function f is n times differentiable with uniformly-bounded derivatives. However, if n is reasonably large, this correction has little effect because the denominator approaches 1; in fact, even for $n=2$ the denominator is already 0.87, so it isn't clear that this modification is ever necessary.

10.2 Polynomial Approximation in the Growth Model

We now implement Chebyshev approximation in the growth model. We pick our grid to be the zeroes of an mth order Chebyshev polynomial

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

and translate them to the interval $[k_l, k_h]$

$$k_i = (z_i + 1)\left(\frac{k_h - k_l}{2}\right) + k_l.$$

Next, make a guess at the policy function on these points:

$$k'(k_i) = g_i.$$

We can now compute the coefficients of the Chebyshev approximation to the g function:

$$a_i = \frac{\sum_{k=1}^{m} g_k T_i(z_k)}{\sum_{k=1}^{m} T_k(z_k)}$$

so that our policy function is

$$\widehat{g}(k) = \sum_{i=0}^{n} a_i T_i\left(2\frac{k - k_l}{k_h - k_l} - 1\right)$$

where clearly we need $m \geq n$.

There are two ways to proceed, iterative and noniterative. The iterative approach guesses $\widehat{g}^0(k')$, inserts the guess $\widehat{g}^0(k')$ in place of k'', and solves for k' in terms of k:

$$Du\left(f(k_i) + (1-\delta)k_i - k'\right) = \beta Du\left(f\left(k'\right) + (1-\delta)k' - \widehat{g}^0\left(k'\right)\right)\left(Df\left(k'\right) + 1 - \delta\right)$$

can be solved to obtain

$$k'(k_i)$$

at each grid point k_i. To solve this we need to solve a single nonlinear equation in a single unknown; under standard assumptions the LHS is strictly increasing in k' and the RHS is strictly decreasing, so we get a unique solution that Brent's method can solve quickly (the obvious bracket is $[0, f(k_i) + (1-\delta)k_i]$ but with Inada conditions you can be a bit more aggressive). We then use these points to "estimate" a new polynomial $\widehat{g}^1(k)$ using projection, and repeat until it converges.

The noninterative method inserts $\widehat{g}$ into the expressions for k' and k'':

$$Du\left(f\left(k_i\right)+\left(1-\delta\right)k_i-\widehat{g}^0\left(k_i\right)\right)=\beta Du\left(f\left(\widehat{g}^0\left(k_i\right)\right)+\left(1-\delta\right)\widehat{g}^0\left(k_i\right)-\widehat{g}^0\left(\widehat{g}^0\left(k_i\right)\right)\right)\left(Df\left(\widehat{g}^0\left(k_i\right)\right)+1-\delta\right).$$

We now have a system of n equations in n unknowns (a_i). If $n=m$, we have a collocation method, and we solve a nonlinear system of equations. If $n<m$ we have a weighted residual method, and we cannot set all of the Euler equations to zero (in general). We then define the Euler residual

$$\begin{aligned}R\left(k_i\right)=&Du\left(f\left(k_i\right)+\left(1-\delta\right)k_i-\widehat{g}^0\left(k_i\right)\right)-\\&\beta Du\left(f\left(\widehat{g}^0\left(k_i\right)\right)+\left(1-\delta\right)\widehat{g}^0\left(k_i\right)-\widehat{g}^0\left(\widehat{g}^0\left(k_i\right)\right)\right)\left(Df\left(\widehat{g}^0\left(k_i\right)\right)+1-\delta\right)\end{aligned}$$

and choose (a_i) to minimize the weighted sum of squares

$$\sum_{i=1}^{m}R\left(k_i\right)^2\omega_i$$

for some set of weights ω_i. The best choice for the weights is constructed using the Chebyshev weighting function

$$\omega_i=w\left(k_i\right)=\left(1-\left(\frac{2k_i-k_l-k_h}{k_h-k_l}\right)^2\right)^{-\frac{1}{2}},$$

which are constant and equal to $\frac{\sqrt{\pi}}{m}$ at the Chebyshev zeros. With this choice, the finite sum is the quadrature approximation to

$$\int R\left(k\right)^2w\left(k\right)dk;$$

that is, we are actually setting the Euler equation to zero as closely as possible over the entire space of capital stocks, not just the ones at the nodes.

Note that we can use the Chebyshev polynomial to construct a Padé approximant as well, since after all the Chebyshev polynomial is just a polynomial. The coefficients for the approximant are a little different since the terms are grouped differently. Suppose we want

$$\sum_{i=0}^{3}a_iT_i\left(k\right)=\frac{c_0+c_1k+c_2k^2}{1+b_1k};$$

then the solution for the coefficients are

$$\begin{aligned}
c_0 &= a_0 - a_2 \\
c_1 &= -\frac{1}{a_2}\left(2a_0a_3 - a_1a_2 + a_2a_3\right) \\
c_2 &= \frac{1}{a_2}\left(2a_2^2 + 6a_3^2 - 2a_1a_3\right) \\
b_1 &= -\frac{2}{a_2}a_3.
\end{aligned}$$

It is straightforward to then check which variant is better, as we did before. For computational purposes, if the expansion is high order it is useful to transform the numerator into the Chebyshev form; for the above example we have

$$f(k) = z_0 + z_1 k + z_2\left(2k^2 - 1\right)$$

where

$$\begin{aligned}
z_0 &= c_0 + \frac{c_2}{2} \\
z_1 &= c_1 \\
z_2 &= \frac{c_2}{2}.
\end{aligned}$$

Projection methods can also easily handle the quasi-geometric discounting problem discussed in the perturbation section (see Judd 2006). The Euler equation for that economy is

$$Du\left(f(k) + (1-\delta)k - g(k)\right) = \beta Du\left(f\left(g(k)\right) + (1-\delta)g(k) - g\left(g(k)\right)\right)\left(Df\left(g(k)\right) + 1 - \delta + \left(\frac{1}{\theta} - 1\right)Dg\left(g(k\right.\right.$$

if we let $g(k) = \sum_{i=0}^{n} a_i T_i(k)$ we can easily compute the term $Dg\left(g(k)\right)$ and solve this problem in the usual way. Note however that the iterative method does not seem to work, for reasons already discussed above, so you should only use the direct method.

10.3 Multidimensional Polynomial Approximation

We can extend projection to multiple dimensions. A function of two variables could be approximated using a product of Chebyshev polynomials:

$$f(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{m} a_{ij} T_i(x) T_j(y).$$

Note that the highest power would be nm. It turns out that we often can achieve nearly the same accuracy with less points by using a "complete polynomial" which drops all terms of order higher than n (similar to pruning). That is, a complete quadratic would take the form

$$f(x,y) = a_{00} + a_{10}T_1(x) + a_{01}T_1(y) + a_{11}T_1(x)T_1(y) + a_{20}T_2(x) + a_{02}T_2(y).$$

We then use a product grid of points. Why can we get away with discarding the other points? The answer is essentially Taylor's theorem. A Taylor expansion of order two would not have the product of second-order terms with first or second-order terms, as those would be third-order, and the second-order Taylor expansion is locally the best polynomial approximation.

Obviously the "curse of dimensionality" is going to show up here – the product grid grows exponentially in the number of dimensions. There are two methods to deal with this problem, both of which selectively remove points from the grid. One, based on simulations designed to identify the ergodic set of the model, will be discussed later. The other one, called the Smolyak sparse grid method, uses an algorithm to toss out points that, in some sense, are likely to be less useful; see Krueger, Kubler, and Malin (2010) for an early application.

Consider our unknown function

$$f : [-1,1]^d \to \mathcal{R}$$

The goal is to find finite set of points $\mathcal{H} \subset [-1,1]^d$ and function $\widehat{f}$ that approximates f well on $[-1,1]^d$. Of course since $[-1,1]^d$ is isomorphic to any hypercube (including hypercubes with sides of different lengths), the domain restriction is harmless – we can always translate each dimensional domain $[a_i, b_i]$ to $[-1,1]$. However, note that the basic shape of the domain must be rectangular.

Suppose we use

$$p_{n-1}(x) = \sum_{j=1}^{n} \theta_j T_j(x)$$

as the class of approximating functions of order $n-1$. Let

$$\mathcal{G}^1 = \{0\}$$
$$\mathcal{G}^n = \{\zeta_1, ..., \zeta_n\} \subset [-1,1]$$

be the extrema of the Chebyshev polynomials

$$\zeta_j = -\cos\left(\frac{\pi(j-1)}{n-1}\right).$$

Let

$$m(1) = 1$$
$$m(i) = 2^{i-1} + 1$$

and note that

$$\mathcal{G}^{m(i)} \subset \mathcal{G}^{m(i+1)}$$

so that the sequence of points will be nested.

Now define a grid in three dimensions with "level" μ:

$$\mathcal{H}^{3,\mu} = \bigcup_{(i_1,i_2,i_3)\in\mathcal{Z}^3_{++}} \mathcal{G}^{m(i_1)} \times \mathcal{G}^{m(i_2)} \times \mathcal{G}^{m(i_3)}.$$

Remember that

$$\mathcal{G}^{m(1)} = \{0\}$$
$$\mathcal{G}^{m(2)} = \{-1, 0, 1\}$$
$$\mathcal{G}^{m(3)} = \left\{-1, -\frac{1}{\sqrt{2}}, 0, \frac{1}{\sqrt{2}}, 1\right\}$$

and so on.

$$\begin{aligned}\mathcal{H}^{3,1} &= \mathcal{G}^{m(2)} \times \mathcal{G}^{m(1)} \times \mathcal{G}^{m(1)} \cup \mathcal{G}^{m(1)} \times \mathcal{G}^{m(2)} \times \mathcal{G}^{m(1)} \cup \mathcal{G}^{m(1)} \times \mathcal{G}^{m(1)} \times \mathcal{G}^{m(2)} \\ &= \{(-1,0,0), (0,0,0), (1,0,0), (0,-1,0), (0,1,0), (0,0,-1), (0,0,1)\}.\end{aligned}$$

Similarly,

$$\begin{aligned}\mathcal{H}^{3,2} &= \mathcal{G}^{m(3)} \times \mathcal{G}^{m(1)} \times \mathcal{G}^{m(1)} \cup \mathcal{G}^{m(1)} \times \mathcal{G}^{m(3)} \times \mathcal{G}^{m(1)} \cup \mathcal{G}^{m(1)} \times \mathcal{G}^{m(1)} \times \mathcal{G}^{m(3)} \cup \\ &\quad \mathcal{G}^{m(2)} \times \mathcal{G}^{m(2)} \times \mathcal{G}^{m(1)} \cup \mathcal{G}^{m(2)} \times \mathcal{G}^{m(1)} \times \mathcal{G}^{m(2)} \cup \mathcal{G}^{m(1)} \times \mathcal{G}^{m(2)} \times \mathcal{G}^{m(2)} \\ &= \left\{ \begin{array}{c} (-1,0,0), \left(-\frac{1}{\sqrt{2}},0,0\right), (0,0,0), \left(\frac{1}{\sqrt{2}},0,0\right), (1,0,0), (0,-1,0), \left(0,-\frac{1}{\sqrt{2}},0\right), \left(0,\frac{1}{\sqrt{2}},0\right), \\ (0,1,0), (0,0,-1), \left(0,0,-\frac{1}{\sqrt{2}}\right), \left(0,0,\frac{1}{\sqrt{2}}\right), (0,0,1), \\ (-1,-1,0), (-1,1,0), (1,1,0), (1,-1,0), (1,0,1), (1,0,-1), (-1,0,1), \\ (-1,0,-1), (0,1,1), (0,-1,-1), (0,-1,1), (0,1,-1) \end{array} \right\}\end{aligned}$$

and so on. In general,

$$\mathcal{H}^{d,\mu} = \bigcup_{\mathbf{i}:|\mathbf{i}|=d+\mu} \left(\mathcal{G}^{m(i_1)} \times \cdots \mathcal{G}^{m(i_d)}\right)$$

where $|\mathbf{i}| = i_1 + \cdots + i_d$.

To do the interpolation, the polynomial constructed as

$$p_\theta^{\mathbf{i}}(s) = \sum_{l_1=1}^{m(i_1)} \cdots \sum_{l_d=1}^{m(i_d)} \theta_{l_1 \ldots l_d} T_{l_1}(s_1) \ldots T_{l_d}(s_d),$$

the tensor product of Chebyshev polynomials. The coefficients are easy to compute:

$$\theta_{l_1 \ldots l_d} = \frac{2^d}{(k_1 - 1) \cdots (k_d - 1)} \frac{1}{c_{l_1} \cdots c_{l_d}} \sum_{j_1=1}^{k_1} \cdots \sum_{j_d=1}^{k_d} \frac{T_{l_1}(\zeta_{j1}) \cdots T_{l_d}(\zeta_{jd}) f(\zeta_{j_1}, ..., \zeta_{j_d})}{c_{j_1} \cdots c_{j_d}}$$

with $c_1 = c_{k_d} = 2$ and $c_j = 1$ otherwise, where k_i is the number of points in the ith dimension. The Smolyak interpolant function is a weighted average of these polynomials:

$$\widehat{f}^{d,\mu}(x) = \sum_{q \leq |\mathbf{i}| \leq d+\mu} (-1)^{d+\mu-|\mathbf{i}|} \begin{pmatrix} d-1 \\ d+\mu-|\mathbf{i}| \end{pmatrix} p^{\mathbf{i}}(x)$$

where $q = \max\{d, \mu + 1\}$.

As an example, suppose $d = 3$ and $\mu = 2$, so that we are using $\mathcal{H}^{3,2}$ as the grid of 25 points. Then $q = 3$ and $\widehat{f}$ is a weighted sum of the following polynomials.

If $|\mathbf{i}| = 3$

$$p_\theta^{1,1,1} = \theta_{1,1,1}.$$

If $|\mathbf{i}| = 4$

$$\begin{aligned}
p_\theta^{2,1,1} &= \theta_{1,1,1} + \theta_{2,1,1} T_2(s_1) + \theta_{3,1,1} T_3(s_1) \\
p_\theta^{1,2,1} &= \theta_{1,1,1} + \theta_{1,2,1} T_2(s_2) + \theta_{1,3,1} T_3(s_2) \\
p_\theta^{1,1,2} &= \theta_{1,1,1} + \theta_{1,1,2} T_2(s_3) + \theta_{1,1,3} T_3(s_3) \\
p_\theta^{3,1,1} &= \theta_{1,1,1} + \theta_{2,1,1} T_2(s_1) + \theta_{3,1,1} T_3(s_1) + \theta_{4,1,1} T_4(s_1) + \theta_{5,1,1} T_5(s_1) \\
p_\theta^{1,3,1} &= \theta_{1,1,1} + \theta_{1,2,1} T_2(s_2) + \theta_{1,3,1} T_3(s_2) + \theta_{1,4,1} T_4(s_2) + \theta_{1,5,1} T_5(s_2) \\
p_\theta^{1,1,3} &= \theta_{1,1,1} + \theta_{1,1,2} T_2(s_3) + \theta_{1,1,3} T_3(s_3) + \theta_{1,1,4} T_4(s_3) + \theta_{1,1,5} T_5(s_3) \\
p_\theta^{2,2,1} &= \theta_{1,1,1} + \theta_{2,1,1} T_2(s_1) + \theta_{3,1,1} T_3(s_1) + \theta_{1,2,1} T_2(s_2) + \theta_{1,3,1} T_3(s_2) + \\
&\quad \theta_{2,2,1} T_2(s_1) T_2(s_2) + \theta_{3,2,1} T_3(s_1) T_2(s_2) + \theta_{2,3,1} T_2(s_1) T_3(s_2) + \\
&\quad \theta_{3,3,1} T_3(s_1) T_3(s_2)
\end{aligned}$$

If $|\mathbf{i}| = 5$

$$\begin{aligned}
p_\theta^{2,1,2} &= \theta_{1,1,1} + \theta_{2,1,1}T_2(s_1) + \theta_{3,1,1}T_3(s_1) + \theta_{1,1,2}T_2(s_3) + \theta_{1,1,3}T_3(s_3) + \\
&\quad \theta_{2,1,2}T_2(s_1)T_2(s_3) + \theta_{3,1,2}T_3(s_1)T_2(s_3) + \theta_{2,1,3}T_2(s_1)T_3(s_3) + \\
&\quad \theta_{3,1,3}T_3(s_1)T_3(s_3) \\
p_\theta^{1,2,2} &= \theta_{1,1,1} + \theta_{1,2,1}T_2(s_2) + \theta_{1,3,1}T_3(s_3) + \theta_{1,1,2}T_2(s_3) + \theta_{1,1,3}T_3(s_3) + \\
&\quad \theta_{1,2,2}T_2(s_2)T_2(s_3) + \theta_{1,3,2}T_3(s_2)T_2(s_3) + \theta_{1,2,3}T_2(s_2)T_3(s_3) + \\
&\quad \theta_{1,3,3}T_3(s_2)T_3(s_3)
\end{aligned}$$

There are 25 unknown coefficients here, and the weights are 1 on $|\mathbf{i}| = 3$ and $|\mathbf{i}| = 5$ and -2 on $|\mathbf{i}| = 4$.

Notice that the sets of points are nested; the role of the negative coefficients is to cancel out the terms that appear in lower sets so that we don't count them twice. Of course, doing so is inefficient, since we calculate these coefficients just to eliminate them. We can implement Smolyak's method more efficiently by constructing non-nested sets:

$$\begin{aligned}
\mathcal{F}^{m(1)} &= \{0\} \\
\mathcal{F}^{m(2)} &= \{-1, 1\} \\
\mathcal{F}^{m(3)} &= \left\{-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right\}.
\end{aligned}$$

Now, since we don't have to cancel out terms that have already been used, we can solve for the coefficients by simple matrix inversion. Concretely, let

$$f(x,y) = b_{11} + b_{12}x + b_{31}\left(2x^2 - 1\right) + b_{12}y + b_{13}\left(2y^2 - 1\right)$$

on the five Smolyak points defined at level 2:

$$\{(0,0), (-1,0), (1,0), (0,-1), (0,1)\}.$$

We can find the coefficients by solving

$$\begin{bmatrix} 1 & 0 & -1 & 0 & -1 \\ 1 & -1 & 1 & 0 & -1 \\ 1 & 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & -1 & 1 \\ 1 & 0 & -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} f(0,0) \\ f(-1,0) \\ f(1,0) \\ f(0,-1) \\ f(0,1) \end{bmatrix}$$

which has solution

$$\begin{bmatrix} b_{11} \\ b_{21} \\ b_{31} \\ b_{12} \\ b_{13} \end{bmatrix} = \begin{bmatrix} \frac{1}{4}\left(f\left(-1,0\right)+f\left(1,0\right)+f\left(0,-1\right)+f\left(0,1\right)\right) \\ \frac{1}{2}\left(f\left(1,0\right)-f\left(-1,0\right)\right) \\ \frac{1}{4}\left(f\left(-1,0\right)+f\left(1,0\right)\right)-\frac{1}{2}f\left(0,0\right) \\ \frac{1}{2}\left(f\left(0,1\right)-f\left(0,-1\right)\right) \\ \frac{1}{4}\left(f\left(0,-1\right)+f\left(0,1\right)\right)-\frac{1}{2}f\left(0,0\right) \end{bmatrix}$$

which agrees with the other method of course.

Smolyak methods have been successfully applied to a multi-country growth model with 10 countries and therefore 20 state variables (capital stock and TFP shock in each country), as well as single country problems where the distribution over capital is a state variable (this one has 200 state variables but is close to linear in most of them).

The use of the Smolyak sparse grid approach is justified only if the function to be approximated is sufficiently smooth – the error of a level l Smolyak approximant on N_l points for a function with r continuous derivatives in d dimensions is $O\left(N_l^{-\frac{r}{d}}\right)$.

10.3.1 Shape Preservation

In the growth model we know that the decision rule $k' = g\left(k\right)$ is monotonically increasing. Cheybshev polynomials do not generally preserve the shape of the data, whether that shape is monotonicity or convexity. Not preserving shape can create problems – the unique solution to the Euler equation $k' = g^1\left(k\right)$ can be lost if our guess $k'' = g^0\left(k'\right)$ is non-monotonic, and during iterations we can encounter this problem even if we do not start with it (shape loss often occurs when functions have sharp changes in their slopes, which would happen in the growth model with high curvature in the utility function or if we have high capital adjustment costs); we already know that Bernstein polynomials do preserve shape, but at the cost of poor approximations with a small number of terms.

One approach is to change the minimization problem used to estimate the coefficients to impose monotonicity and concavity on the solution. The coefficients solve the linear program

$$\min_{\left(a_j^+,a_j^-\right)_{j=0}^n} \left\{\sum_{j=0}^{n}\left(\frac{1}{1+j}\right)^2\left(a_j^+ + a_j^-\right)\right\}$$

subject to

$$\sum_{j=0}^{n} a_j DT_j(k_i) > 0 \text{ for } i \in \{1, ..., m'\}$$

$$\sum_{j=0}^{n} a_j D^2 T_j(k_i) < 0 \text{ for } i \in \{1, ..., m'\}$$

$$\sum_{j=0}^{n} a_j T_j(k_i) = g_i \text{ for } i \in \{1, ..., m\}$$

$$a_j^+ \geq 0$$

$$a_j^- \geq 0$$

and setting $a_j = a_j^+ - a_j^-$. We set $m' > m > n$ to ensure stability and we can use standard linear programming tools (like the simplex method) to solve the program. Note that we *impose* this structure on the problem – we must therefore be sure that the functions being approximated actually have the desired shape, as otherwise we will compromise the accuracy of our solution; in contrast we will see methods later that *preserve* shape if it is present. This procedure extends in a straightforward way to multiple dimensions, and linear programming scales quite nicely so the curse of dimensionality does not bite particularly hard (at least, no harder than it already did without imposing shape).

10.4 Hybrid Perturbation-Projection Methods

Both perturbation and projection methods construct solutions to systems of functional equations that take the form

$$f(x) = \sum_{i=0}^{n} a_i \varphi_i(x)$$

for some coefficients a_i and some basis functions $\varphi_i(x)$. In regular perturbation methods, the coefficients are fixed as $a_i = \varepsilon^i$ and the functions are constructed using repeated differentiation. When ε is small, perturbation methods are asymptotically the best approximators (Taylor's Theorem). When ε is not small, they may not work as well.

Projection methods fix the basis functions (say, as Chebyshev polynomials) and then solve for the coefficients. The coefficients are the best given the basis functions, but these functions may not be the best ones for a given problem; generally, they are chosen to be good for a generic

function (Stone-Weierstrass Theorem). It would be better if we could combine the two approaches into one. The hybrid approach first obtains an approximation

$$f(x) = \sum_{i=1}^{n} \delta_i(\varepsilon)\varphi_i(x)$$

for some sequence of gauge functions $\delta_i(\varepsilon)$. We then use a projection method to obtain coefficients such that

$$\widehat{f}(x) = \sum_{i=1}^{n} a_i\varphi_i(x)$$

best solves the functional equation. The key is to use the perturbation method to get the functions and the projection method to get the coefficients.

As an example, we'll solve the continuous time growth model

$$\max_c \left\{ \int_0^\infty e^{-\rho t} \frac{c(t)^{1+\gamma}}{1+\gamma} dt \right\}$$

subject to

$$\dot{k}(t) = f(k(t)) - c(t)$$
$$k(0) = k_0.$$

Using the Hamiltonian for this problem leads to the Euler equation

$$\gamma DC(k)(f(k) - C(k)) - C(k)(\rho - Df(k)) = 0.$$

Parameterizing the problem by $\varepsilon = -\gamma^{-1}$ implies that the solution at $\varepsilon = 0$ is $C(k,0) = f(k)$, since

$$D_k C(k,\varepsilon)(f(k) - C(k,\varepsilon)) - \varepsilon C(k,\varepsilon)(Df(k) - \rho) = 0$$

only works at $\varepsilon = 0$ if that is true. The first perturbation yields

$$D_{k\varepsilon}C(k)(f(k) - C(k)) + D_k C(k)(-D_\varepsilon C(k)) - C(k)(Df(k) - \rho) - \varepsilon D_\varepsilon C(k)(Df(k) - \rho) = 0$$

so that

$$D_\varepsilon C(k,0) = f(k)\left(\frac{\rho}{Df(k)} - 1\right).$$

The second yields

$$D_{\varepsilon\varepsilon}C(k,0) = \frac{2\rho(Df(k) - \rho)D^2 f(k)}{(Df(k))^4}.$$

If we assume that $f(k) = Ak^{\alpha}$ then the first three basis functions are

$$\begin{aligned} C(k,0) &= Ak^{\alpha} \\ D_{\varepsilon}C(k,0) &= Ak^{\alpha}\left(\frac{\rho}{\alpha Ak^{\alpha-1}} - 1\right) \\ &= \frac{\rho}{\alpha}k - Ak^{\alpha} \\ D_{\varepsilon\varepsilon}C(k,0) &= \frac{2\rho\left(\alpha Ak^{\alpha-1} - \rho\right)\alpha(\alpha-1)Ak^{\alpha-2}}{\left(\alpha Ak^{\alpha-1}\right)^4} \\ &= \frac{2\rho(\alpha-1)}{\alpha^2A^2}k^{1-2\alpha} - 2\frac{\rho^2}{\alpha^3}\frac{\alpha-1}{A^3}k^{2-3\alpha}. \end{aligned}$$

We therefore look for a projection solution that takes the form

$$C(k) = a_0 + a_1k + a_2k^{\alpha} + a_3k^{1-2\alpha} + a_4k^{2-3\alpha}.$$

If $\alpha = 1/3$ then the function we use has the form

$$C(k) = a_0 + a_1k + a_2k^{\frac{1}{3}},$$

since

$$\begin{aligned} \alpha &= 1 - 2\alpha \\ 1 &= 2 - 3\alpha, \end{aligned}$$

so that the problem reduces to a very simple projection equation. Since the above basis is not orthogonal, it may pose numerical problems, but these can be avoided by applying the Gram-Schmidt orthogonalization procedure. The Gram-Schmidt procedure takes a vector $\{v_1, v_2, ...\}$ and orthonormalizes it by setting

$$\begin{aligned} u_1 &= v_1 \\ u_2 &= v_2 - \frac{\langle v_2, u_1\rangle}{\langle u_1, u_1\rangle}u_1 \\ u_3 &= v_3 - \frac{\langle v_3, u_1\rangle}{\langle u_1, u_1\rangle}u_1 - \frac{\langle v_3, u_2\rangle}{\langle u_2, u_2\rangle}u_2 \end{aligned}$$

and so on, where $\langle x, y\rangle$ is an inner product.

Let's assume the range of capital is $[0, 1]$, or has been translated there. To apply the Gram-Schmidt procedure we note that the inner product of elements in function spaces is defined by

$$\langle f, g\rangle = \int_a^b f(x)\, g(x)\, dx$$

and set

$$\begin{aligned}
u_1 &= 1 \\
u_2 &= k - \frac{\langle k, 1\rangle}{\langle 1, 1\rangle} 1 \\
&= k - \int_0^1 k dk \\
&= k - \frac{1}{2} \\
u_3 &= k^{\frac{1}{3}} - \frac{\left\langle k^{\frac{1}{3}}, 1\right\rangle}{\langle 1, 1\rangle} 1 - \frac{\left\langle k^{\frac{1}{3}}, k - \frac{1}{2}\right\rangle}{\left\langle k - \frac{1}{2}, k - \frac{1}{2}\right\rangle}\left(k - \frac{1}{2}\right) \\
&= k^{\frac{1}{3}} - \int_0^1 k^{\frac{1}{3}} dk - \frac{\int_0^1 k^{\frac{1}{3}}\left(k - \frac{1}{2}\right) dk}{\int_0^1 \left(k - \frac{1}{2}\right)^2 dk}\left(k - \frac{1}{2}\right) \\
&= k^{\frac{1}{3}} - \frac{3}{7} - \frac{9}{14}k.
\end{aligned}$$

We therefore can search for a solution that takes the form

$$C(k) = a_0 + a_1\left(k - \frac{1}{2}\right) + a_2\left(k^{\frac{1}{3}} - \frac{3}{4} - \frac{9}{7}k\right)$$

easily, and the resulting functions are orthogonal:

$$\begin{aligned}
\int_0^1 1\left(k - \frac{1}{2}\right) dk &= 0 \\
\int_0^1 \left(k^{\frac{1}{3}} - \frac{3}{7} - \frac{9}{14}k\right) dk &= 0 \\
\int_0^1 \left(k - \frac{1}{2}\right)\left(k^{\frac{1}{3}} - \frac{3}{7} - \frac{9}{14}k\right) dk &= 0.
\end{aligned}$$

We can even construct these orthogonal terms over an arbitrary interval $[0, K]$:

$$\begin{aligned}
u_1 &= 1 \\
u_2 &= k - \frac{\langle k, 1\rangle}{\langle 1, 1\rangle} 1 \\
&= k - \int_0^K k dk \\
&= k - \frac{1}{2} K^2 \\
u_3 &= k^{\frac{1}{3}} - \frac{\left\langle k^{\frac{1}{3}}, 1\right\rangle}{\langle 1, 1\rangle} 1 - \frac{\left\langle k^{\frac{1}{3}}, k - \frac{1}{2}\right\rangle}{\left\langle k - \frac{1}{2}, k - \frac{1}{2}\right\rangle}\left(k - \frac{1}{2}\right) \\
&= k^{\frac{1}{3}} - \int_0^K k^{\frac{1}{3}} dk - \frac{\int_0^K k^{\frac{1}{3}}\left(k - \frac{1}{2}\right) dk}{\int_0^K \left(k - \frac{1}{2}\right)^2 dk}\left(k - \frac{1}{2}\right) \\
&= \frac{\left(112K^2 - 168K + 84\right) k^{\frac{1}{3}}}{28\left(4K^2 - 6K + 3\right)} + \frac{84K^{\frac{10}{3}} - 126K^{\frac{7}{3}} - 9K^{\frac{4}{3}} + 63K^{\frac{1}{3}}}{28\left(4K^2 - 6K + 3\right)} - \\
&\frac{\left(144K^{\frac{4}{3}} - 126K^{\frac{1}{3}}\right) k}{28\left(4K^2 - 6K + 3\right)}
\end{aligned}$$

although clearly it is not as pleasant looking the only difference is in the constant terms.

10.5 Taylor Projection and Exact-Today Approximation

Taylor projection is an alternative method for generating the projection equations – rather than using information at several points, it uses information about the Euler equation and its derivatives at a single point k^*; unlike perturbation, however, this point need not be the deterministic steady state. Levintal (2018) is the original source of this approach, although it has connections to the hybrid perturbation-projection method discussed Judd (1998) and used in Chen and Fowler (2016).

To begin, suppose we are looking for a second-order solution of the form

$$g\left(k_t\right) = g_0 + g_1\left(k_t - k^*\right) + \frac{g_2}{2}\left(k_t - k^*\right)^2.$$

We will use projection equations to solve for (g_0, g_1, g_2), where these equations are defined to be

$$\left[\begin{array}{c} G\left(k_t\right) \\ DG\left(k_t\right) \\ D^2G\left(k_t\right) \end{array}\right] = 0.$$

For the growth model we can write them out explicitly:

$$G(k_t) = Du(h(k_t)) - \beta Du(h(g(k_t)))(Df(g(k_t)) + 1 - \delta)$$
$$h(k_t) = f(k_t) + (1-\delta)k_t - g(k_t)$$

$$DG(k_t) = D^2u(h(k_t))Dh(k_t) - \beta D^2u(h(g(k_t)))(Df(g(k_t)) + 1 - \delta)Dh(g(k_t))Dg(k_t) - \beta Du(h(g(k_t)))D^2f(g(k_t))Dg(k_t)$$
$$Dh(k_t) = Df(k_t) + 1 - \delta - Dg(k_t)$$

$$\begin{aligned} D^2G(k_t) = {} & D^3u(h(k_t))(Dh(k_t))^2 + D^2u(h(k_t))D^2h(k_t) - \\ & \beta D^3u(h(g(k_t)))(Df(g(k_t)) + 1 - \delta)(Dh(g(k_t))Dg(k_t))^2 - \\ & \beta D^2u(h(g(k_t)))D^2f(g(k_t))Dh(g(k_t))(Dg(k_t))^2 - \\ & \beta D^2u(h(g(k_t)))(Df(g(k_t)) + 1 - \delta)D^2h(g(k_t))(Dg(k_t))^2 - \\ & \beta D^2u(h(g(k_t)))(Df(g(k_t)) + 1 - \delta)Dh(g(k_t))D^2g(k_t) - \\ & \beta D^2u(h(g(k_t)))D^2f(g(k_t))Dh(g(k_t))(Dg(k_t))^2 - \\ & \beta Du(h(g(k_t)))D^3f(g(k_t))(Dg(k_t))^2 - \\ & \beta Du(h(g(k_t)))D^2f(g(k_t))D^2g(k_t) \end{aligned}$$
$$Dh^2(k_t) = D^2f(k_t) - D^2g(k_t).$$

Note that we can use automatic differentiation to obtain the values of these equations at k_t given the coefficients (g_0, g_1, g_2), and therefore do not need to actually differentiate them. To do a third-order approximation, given that most automatic differentiation code only gives you up to second derivatives, you can simply work with the expression for DG instead of G, but naturally you have to construct DG yourself (which may be a pain).

Relative to standard projection, Taylor projection is faster but is not global as it uses information only at one point; implicitly, we are assuming that the state today is not too far from the optimal state tomorrow. However, we can "tie" solutions at different points k_i using Shepard's

method of inverse-distance weighting:

$$\begin{aligned} g_i(k_t) &= g_{0,i} + g_{1,i}(k_t - k_i) + \frac{g_{2,i}}{2}(k_t - k_i)^2 \\ g(k_t) &= \frac{\sum_{i=1}^m w_i(k_t) g_i(k_t)}{\sum_{i=1}^m w_i(k_t)} \\ w_i(k_t) &= \frac{1}{\|k_t - k_i\|^p} \\ p &\geq 2; \end{aligned}$$

this approach is used in Bacchetta, van Wincoop, and Young (2020) to solve a model with 15 state variables. Using many points instead of only one leads to significant improvement in the Euler equation errors, and it is much faster than standard projection because we are solving a bunch of small systems of equations rather than one big system. The errors are a little smaller if one uses a high value for p; as $p \to \infty$ the weighting scheme approaches a 'nearest neighbor' spline, so very large values are not desireable. In n dimensions if $p < n$ the weighting scheme puts more weight on more distant points, rather than nearby ones, so we also cannot use small values for p. We'll discuss inverse distance weighting below.

Closely related to Taylor projection is a method by den Haan, Kobielarz, and Rendahl (2016) called Exact-Today and Consistent Tomorrow Approximation (ET). Suppose that the current choice for capital is k_{t+1}; then use a first-order perturbation expansion for tomorrow around the current state today to obtain

$$k_{t+2} = k_{t+1} + \phi_1(k_{t+1} - k_t) + \phi_2(z_{t+1} - z_t).$$

The Euler equation is then

$$F(k_t, k_{t+1}, z_t) = (\exp(z_t) k_t^\alpha + (1-\delta) k_t - k_{t+1})^{-\sigma} -$$
$$\beta \sum_{i=1}^n \omega_i \left(\begin{array}{c} \left(\exp(\rho z_t + \epsilon_i) k_{t+1}^\alpha + (1-\delta) k_{t+1} - (k_{t+1} + \phi_1(k_{t+1} - k_t) + \phi_2(\rho z_t + \epsilon_i - z_t))\right)^{-\sigma} \times \\ \left(\alpha \exp(\rho z_t + \epsilon_i) k_{t+1}^{\alpha-1} + 1 - \delta\right) \end{array} \right).$$

Take the derivatives of F with respect to k_t, k_{t+1}, and z_t:

$$D_{k_t}F\left(k_t,k_{t+1},z_t\right)=-\sigma\left(\exp\left(z_t\right)k_t^{\alpha}+\left(1-\delta\right)k_t-k_{t+1}\right)^{-\sigma-1}\left(\alpha\exp\left(z_t\right)k_t^{\alpha-1}+1-\delta\right)+$$

$$\sigma\beta\sum_{i=1}^{n}\omega_i\left(\begin{array}{c}\left(\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha}+\left(1-\delta\right)k_{t+1}-\left(k_{t+1}+\phi_1\left(k_{t+1}-k_t\right)+\phi_2\left(\rho z_t+\epsilon_i-z_t\right)\right)\right)^{-\sigma-1}\times\\ \left(\alpha\exp\left(\rho z_t+\epsilon_j\right)k_{t+1}^{\alpha-1}+1-\delta\right)\phi_1\end{array}\right)$$

$$D_{z_t}F\left(k_t,k_{t+1},z_t\right)=-\sigma\left(\exp\left(z_t\right)k_t^{\alpha}+\left(1-\delta\right)k_t-k_{t+1}\right)^{-\sigma-1}\exp\left(z_t\right)k_t^{\alpha}+$$

$$\sigma\beta\sum_{i=1}^{n}\omega_i\left(\begin{array}{c}\left(\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha}+\left(1-\delta\right)k_{t+1}-\left(k_{t+1}+\phi_1\left(k_{t+1}-k_t\right)+\phi_2\left(\rho z_t+\epsilon_i-z_t\right)\right)\right)^{-\sigma-1}\times\\ \left(\alpha\exp\left(\rho z_t+\epsilon_j\right)k_{t+1}^{\alpha-1}+1-\delta\right)\left(\rho\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha}-\phi_2\left(\rho-1\right)\right)\end{array}\right)-$$

$$\alpha\rho\beta\sum_{i=1}^{n}\omega_i\left(\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha}+\left(1-\delta\right)k_{t+1}-\left(k_{t+1}+\phi_1\left(k_{t+1}-k_t\right)+\phi_2\left(\rho z_t+\epsilon_j-z_t\right)\right)\right)^{-\sigma}\exp\left(\rho z_t+\epsilon_j\right)k_{t+1}^{\alpha-1}$$

$$D_{k_{t+1}}F\left(k_t,k_{t+1},z_t\right)=\sigma\left(\exp\left(z_t\right)k_t^{\alpha}+\left(1-\delta\right)k_t-k_{t+1}\right)^{-\sigma-1}+$$

$$\sigma\beta\sum_{i=1}^{n}\omega_i\left(\begin{array}{c}\left(\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha}+\left(1-\delta\right)k_{t+1}-\left(k_{t+1}+\phi_1\left(k_{t+1}-k_t\right)+\phi_2\left(\rho z_t+\epsilon_j-z_t\right)\right)\right)^{-\sigma-1}\times\\ \left(\alpha\exp\left(\rho z_t+\epsilon_j\right)k_{t+1}^{\alpha-1}+1-\delta\right)\left(\alpha\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha-1}+1-\delta-\left(1+\phi_1\right)\right)\end{array}\right)-$$

$$\alpha\left(\alpha-1\right)\beta\sum_{i=1}^{n}\omega_i\left(\begin{array}{c}\left(\exp\left(\rho z_t+\epsilon_i\right)k_{t+1}^{\alpha}+\left(1-\delta\right)k_{t+1}-\left(k_{t+1}+\phi_1\left(k_{t+1}-k_t\right)+\phi_2\left(\rho z_t+\epsilon_j-z_t\right)\right)\right)^{-\sigma}\times\\ \exp\left(\rho z_t+\epsilon_j\right)k_{t+1}^{\alpha-2}\end{array}\right);$$

it is obviously easier to use automatic differentiation here.

The goal is to solve for the variables (k_{t+1},ϕ_1,ϕ_2) using the equations

$$\begin{aligned}F\left(k_t,k_{t+1},z_t\right)&=0\\ D_{k_t}F\left(k_t,k_{t+1},z_t\right)+D_{k_{t+1}}F\left(k_t,k_{t+1},z_t\right)\frac{dk_{t+1}}{dk_t}&=0\\ D_{z_t}F\left(k_t,k_{t+1},z_t\right)+D_{k_{t+1}}F\left(k_t,k_{t+1},z_t\right)\frac{dk_{t+1}}{dz_t}&=0\end{aligned}$$

with

$$\begin{aligned}\frac{dk_{t+1}}{dk_t}&=\phi_1\\ \frac{dk_{t+1}}{dz_t}&=\phi_2;\end{aligned}$$

these equations impose that the linear approximation of k_{t+2} has the same slope as the function we uncover that links k_{t+1} to k_t and z_t. As with Taylor projection, this approximation can be done at any (k_t,z_t) point, so we can do a "simulation-based" approach (see more on this stuff

below) by solving the equations along a sample path (note that the approximate solution for k_{t+2} is never used).

Note also there is nothing special about a linear approximation for k_{t+2}; we can use a higher-order approximation. Suppose that

$$k_{t+2} = k_{t+1}+\phi_1 \left(k_{t+1} - k_t\right)+\phi_2 \left(z_{t+1} - z_t\right)+\frac{\phi_3}{2} \left(k_{t+1} - k_t\right)^2+\phi_4 \left(k_{t+1} - k_t\right) \left(z_{t+1} - z_t\right)+\frac{\phi_5}{2} \left(z_{t+1} - z_t\right)^2.$$

Now we have to take the second-order derivatives of F, which generates the extra three equations we need:

$$D_{k_t k_t} F\left(k_t, k_{t+1}, z_t\right) + 2D_{k_t k_{t+1}} F\left(k_t, k_{t+1}, z_t\right) \phi_1 + D_{k_{t+1} k_{t+1}} F\left(k_t, k_{t+1}, z_t\right) \phi_1^2 + D_{k_{t+1}}$$

$$D_{k_t z_t} F\left(k_t, k_{t+1}, z_t\right) + D_{k_t k_{t+1}} F\left(k_t, k_{t+1}, z_t\right) \phi_2 + D_{k_{t+1} z_t} F\left(k_t, k_{t+1}, z_t\right) \phi_1 + D_{k_{t+1} k_{t+1}} F\left(k_t, k_{t+1}, z_t\right) \phi_1 \phi_2 + D_{k_{t+1}}$$

$$D_{z_t z_t} F\left(k_t, k_{t+1}, z_t\right) + 2D_{z_t k_{t+1}} F\left(k_t, k_{t+1}, z_t\right) \phi_2 + D_{k_{t+1} k_{t+1}} F\left(k_t, k_{t+1}, z_t\right) \phi_2^2 + D_{k_{t+1}}$$

We can also "extend" the method by assuming the perturbation solution applies at $t+2$ rather than $t+1$, so that we have the $n+3$ system of equations

$$\begin{aligned}
&F_0\left(k_t, k_{t+1}, k_{t+2}, z_t\right) \\
&= \left(\exp\left(z_t\right) k_t^\alpha + \left(1-\delta\right) k_t - k_{t+1}\right)^{-\sigma} - \\
&\beta \sum_{i=1}^{n} \omega_i \left(\exp\left(\rho z_t + \epsilon_i\right) k_{t+1}^\alpha + \left(1-\delta\right) k_{t+1} - k_{t+2,i}\right)^{-\sigma} \left(\alpha \exp\left(\rho z_t + \epsilon_i\right) k_{t+1}^{\alpha-1} + 1 - \delta\right) \\
&F_i\left(k_t, k_{t+1}, k_{t+2,i}, z_t, \epsilon_i\right) \\
&= \left(\exp\left(z_{t+1}\right) k_{t+1}^\alpha + \left(1-\delta\right) k_{t+1} - k_{t+2,i}\right)^{-\sigma} - \\
&\beta \sum_{j=1}^{n} \omega_j \left(\exp\left(\rho^2 z_t + \rho\epsilon_i + \epsilon_j\right) k_{t+2,i}^\alpha + \left(1-\delta\right) k_{t+2,i} - \left(k_{t+2,i} + \phi_1\left(k_{t+2,i} - k_{t+1}\right) + \phi_2\left(\rho^2 z_t + \rho\epsilon_i + \epsilon_j - z_t\right)\right)\right)^{-\sigma} \times \\
&\left(\alpha \exp\left(\rho^2 z_t + \rho\epsilon_i + \epsilon_j\right) k_{t+2,i}^{\alpha-1} + 1 - \delta\right)
\end{aligned}$$

to solve for $\left(k_{t+1}, \left(k_{t+2,i}\right)_{i=1}^n, \phi_1, \phi_2\right)$. While more costly, it also limits the dependence on the perturbation solution, and therefore should be more accurate. Again, we ignore the actual solutions for $k_{t+2,i}$.

The method used in Chen and Fowler (2016) is a more general variant of Taylor projection, in that they generate the Taylor expansion in a nonlinear transformation of the state (think of it as subjecting Taylor projection to the Fernandez-Villaverde and Rubio-Ramirez 2006 adjustments).

In fact, this approach offers a way to compute the adjustment parameters – we use the projection to find the change of variables that reduces the Euler equation errors as much as possible. Conceptually, think of solving the projection equations in the growth model

$$G\left(k_t; a, p\right) = 0$$

$$DG\left(k_t; a, p\right) = 0$$

where

$$k_{t+1} = \sum_{i=0}^{n} a_i T_n\left(Z\left(k_t; p\right)\right)$$

where Z is the transformation parameterized by p. For example, if we use the power transformation $k \mapsto k^\gamma$ then we get

$$Z\left(k; \gamma\right) = k^\gamma - \left(k^*\right)^\gamma;$$

for a quadratic we then have

$$k_{t+1} = a_0 + a_1\left(k^\gamma - \left(k^*\right)^\gamma\right) + a_2\left(2\left(k^\gamma - \left(k^*\right)^\gamma\right)^2 - 1\right).$$

We then choose both a and γ to minimize the errors. Because we can always choose $\gamma = 1$, this approach will always weakly improve fit, and we can use the solution for $\gamma = 1$ as a starting value.

10.6 Nonlinear Certainty-Equivalent Approximation

Another alternative is to think of solving a certainty equivalent version of the sequence problem. Fix an initial value (k_0, z_0) and construct a deterministic sequence

$$z_t = \rho z_{t-1} - \frac{1}{2}\sigma^2;$$

the second term is to correct for a certainty equivalence bias and may (or may not) improve accuracy. Then solve the problem (for some fixed large T)

$$V_0\left(k_0, z_0\right) = \max_{\left\{k_{t+1}\right\}_{t=0}^{T-1}} \left\{ \sum_{t=0}^{T-1} \beta^t \frac{\left(\exp\left(z_t\right) k_t^\alpha + \left(1-\delta\right) k_t - k_{t+1}\right)^{1-\sigma}}{1-\sigma} + \beta^T V_T\left(k_T, z_T\right) \right\}$$

where

$$V_T\left(k_T, z_T\right) = \frac{\left(\exp\left(z_T\right) k_T^\alpha + \left(1-\delta\right) k_T - k\right)^{1-\sigma}}{\left(1-\beta\right)\left(1-\sigma\right)}$$

or alternatively V_T is a quadratic approximation around the steady state $(k, 0)$

$$V_T(k_T, z_T) = A + B \begin{bmatrix} k_T - k \\ z_T \\ k_{T+1} - k \end{bmatrix} + \frac{1}{2} \begin{bmatrix} k_T - k & z_T & k_{T+1} - k \end{bmatrix} C \begin{bmatrix} k_T - k \\ z_T \\ k_{T+1} - k \end{bmatrix};$$

for large enough T obviously it doesn't matter which option you choose, but larger T is more costly. Set the policy function equal to

$$\pi_0(k_0, z_0) = k_1(k_0, z_0).$$

Now repeat for a number of different (k_0, z_0) and approximate (V, π) (the true value function and policy function) by fitting some nonlinear approximator (such as a polynomial). One can either choose the nodes (k_0, z_0) using some kind of quadrature method (Gauss-Chebyshev) or by constructing a set of covering points from a simulation of a perturbation solution (see below).

If the equilibrium is not the solution to an optimization problem, one can approach this problem by solving the problem

$$\min_{\{k_{t+1}\}_{t=0}^{T-1}} \{1\}$$

subject to the equilibrium conditions

$$\left(\exp(z_t) k_t^{\alpha} + (1-\delta) k_t - k_{t+1}\right)^{-\sigma} - \beta \left(\exp(z_{t+1}) k_{t+1}^{\alpha} + (1-\delta) k_{t+1} - k_{t+2}\right)^{-\sigma} \left(\alpha \exp(z_{t+1}) k_{t+1}^{\alpha-1} + 1 - \delta\right) = 0$$

$$\exp(z_t) k_t^{\alpha} + (1-\delta) k_t - k_{t+1} \geq 0$$

along with an appropriate terminal condition.

A variant of this approach is to allow for risk in period 1 only – using Gauss-Hermite quadrature we can construct a node of potential productivity values tomorrow:

$$z_{1,i} = \rho z_0 + e_i.$$

From period 2 onward z evolves deterministically:

$$z_{t+1,i} = \rho z_{t,i}.$$

The choice variables are then the capital stock for period 1, k_1, and the sequences starting from each $z_{1,i}$. The objective is

$$V(k_0, z_0) = \max_{k_1, \{k_{t,i}\}_{t=2,i=1}^{T,n}} \left\{ \begin{array}{c} \frac{1}{1-\sigma} \left(\exp(z_0) k_0^{\alpha} + (1-\delta) k_0 - k_1\right)^{1-\sigma} + \\ \beta \sum_{i=1}^{n} \omega_i \left[\sum_{t=1}^{T-1} \frac{\beta^{t-1}}{1-\sigma} \left(\exp(z_{t,i}) k_{t,i}^{\alpha} + (1-\delta) k_{t,i} - k_{t+1,i}\right)^{1-\sigma} + \beta^T V_T(k_{T,i}, z_{T,i}) \right] \end{array} \right\}$$

where $k_{1,i} = k_1$. It is an open question whether this approach delivers a better approximation compared to the increase in computational burden.

Closely related to this procedure is one that uses projection methods to solve both the dynamic program and the approximation problem at the same time. Consider the generic dynamic program

$$\begin{aligned} v(x) &= \max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta v\left(x'\right) \right\} \\ x' &= t(x,a). \end{aligned}$$

Define two problems

$$\begin{aligned} a^*(x) &= \arg\max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta v\left(t(x,a)\right) \right\} \\ \alpha^* &= \arg\min_{\alpha} \left\{ \sum_{i=1}^{n} \left(r\left(x_i, a^*(x_i)\right) + \beta v\left(t\left(x_i, a^*(x_i)\right); \alpha\right) - v\left(x_i; \alpha\right) \right)^2 \right\}; \end{aligned}$$

the first is the definition of the policy function and the second is the functional approximation problem for the value function. Assume that

$$\begin{aligned} a^* &\in \mathcal{R}^n \\ \alpha^* &\in \mathcal{R}^m. \end{aligned}$$

Define

$$F(z) \perp l \leq z \leq u$$

as notation for

$$\begin{aligned} 0 &\leq F(z) \text{ and } z = l \\ 0 &= F(z) \text{ and } l \leq z \leq u \\ 0 &\geq F(z) \text{ and } z = u. \end{aligned}$$

Thus, for the optimization problem

$$\min_{x \geq 0} \{ f(x) \}$$

subject to

$$g(x) \geq 0,$$

we can write the KT conditions as

$$0 \leq \nabla f(x) - \nabla g(x) \lambda \perp x \geq 0$$
$$0 \leq g(x) \perp \lambda \geq 0.$$

The equations we need to satisfy to jointly solve the two problems are

$$-\left(\frac{\partial r(x_i, a_i)}{\partial a_i} + \beta \frac{\partial V(x_i'; \alpha)}{\partial a_i}\right) + \frac{\partial t(x_i, a_i)}{\partial a_i} p_i \perp a_l \leq a_i \leq a_u \quad \forall i \in \{1, ..., m\}$$
$$U_i = r(x_i, a_i) + \beta V\left(x_i'\right) \perp U_i \text{ free} \quad \forall i \in \{1, ..., m\}$$
$$\frac{\partial \left(\sum_{i=1}^{n} \left(U_i - V(x_i)\right)^2\right)}{\partial \alpha_l} = 0 \perp \alpha_l \text{ free} \quad \forall l \in \{1, ..., m\}$$
$$x_i' = h(x_i, a_i) \perp p_i \text{ free} \quad \forall i \in \{1, ..., m\}.$$

To take a simple example, consider the deterministic growth model with Chebyshev polynomial approximation:

$$g(k) = \arg\max_{k'} \left\{ \log\left(k^\alpha + (1-\delta)k - k'\right) + \beta \sum_{j=0}^{m} a_j T_j\left(k'\right) \right\}$$
$$a = \arg\max_{\alpha} \left\{ \sum_{i=1}^{n} \left(\sum_{j=0}^{m} \alpha_j T_j(k_i) - \log\left(k_i^\alpha + (1-\delta)k_i - g(k_i)\right) - \beta \sum_{j=0}^{m} \alpha_j T_j(g(k_i)) \right)^2 \right\}.$$

The first-order conditions are

$$0 = \frac{1}{k^\alpha + (1-\delta)k - k'} - \beta \sum_{j=0}^{m} a_j T_j'\left(k'\right) \quad \forall i$$
$$0 = \sum_{i=1}^{n} \left(\sum_{j=0}^{m} \alpha_j T_j(k_i) - \log\left(k_i^\alpha + (1-\delta)k_i - k_i'\right) - \beta \sum_{j=0}^{m} \alpha_j T_j\left(k_i'\right) \right) \left(T_j(k_i) - \beta T_j\left(k_i'\right)\right) \quad \forall j.$$

If we assume that the approximation is quadratic ($m = 2$) and $n = 3$, then we are looking for the

solution to the equations

$$
\begin{aligned}
0 &= \frac{1}{k_1^{\alpha} + (1-\delta) k_1 - k_1'} - \beta \left(a_1 + 4 a_2 k_1'\right) \\
0 &= \frac{1}{k_2^{\alpha} + (1-\delta) k_2 - k_2'} - \beta \left(a_1 + 4 a_2 k_2'\right) \\
0 &= \frac{1}{k_3^{\alpha} + (1-\delta) k_3 - k_3'} - \beta \left(a_1 + 4 a_2 k_3'\right) \\
0 &= \sum_{i=1}^{3} z_i \left(1 - \beta\right) \\
0 &= \sum_{i=1}^{3} z_i \left(k_i - \beta k_i'\right) \\
0 &= \sum_{i=1}^{3} z_i \left(\left(2k_i^2 - 1\right) - \beta \left(2 \left(k_i'\right)^2 - 1\right)\right)
\end{aligned}
$$

where

$$
z_i = \left(a_0 + a_1 k_i + a_2 \left(2k_i^2 - 1\right) - \log\left(k_i^{\alpha} + (1-\delta) k_i - k_i'\right) - \beta \left(a_0 + a_1 k_i' + a_2 \left(2 \left(k_i'\right)^2 - 1\right)\right)\right)
$$

is the value function approximation error. We then can simply use a nonlinear equation solver to get (a, k').

10.7 Splines

Splines are piecewise polynomial functions – clearly this space of functions contains all polynomials, but also contains many more functions as well. Splines are a very flexible form of approximator, although they are not "low dimensional" in the sense that they require many pieces of data to describe. We will assume that the state space has been partitioned into a grid $\{x_1, ..., x_n\}$ (which need not be uniformly-spaced but must be rectangular, which does not matter in one dimension); we call the grid points **nodes** and the segments between them **elements**. The spline takes a function and replaces it with a vector (representing the function values at the nodes) and an "interpolation rule" that describes the polynomial that represents the function in between the nodes.

For future reference, if we know the nodes and the value of the function at the nodes (x, y), we have "Lagrange" data. If we also know the slope of the function at the nodes (x, y, s), we have "Hermite" data.

10.7.1 Linear Spline

The idea behind linear interpolation is to connect the grid points with straight lines, leading to a continuous function. The value of the function at a point between two points on the grid is given by

$$y = Ay_j + By_{j+1}$$

where

$$A = \frac{x_{j+1} - x}{x_{j+1} - x_j}$$
$$B = 1 - A = \frac{x - x_j}{x_{j+1} - x_j}.$$

Storage requirements for a linear spline at equal to the number of nodes.

Derivatives can be calculated easily as

$$y' = \frac{y_{j+1} - y_j}{x_{j+1} - x_j};$$

the derivative will change discontinuously at the nodes. To solve an Euler equation we again have iterative and noniterative methods. The iterative method makes a guess at the policy function for k'' at the nodes, then solves the nonlinear equation for k' at each k, just as with orthogonal polynomials. The noniterative method solves for the value of the function at the nodes, which can be very costly if the number of nodes is large but is done only once.

10.7.2 Cubic Spline

Cubic splines involve "connecting the dots" in a smooth fashion. The basic idea is to piece together a function out of cubic portions such that the function goes through each of the points and that the derivatives also "connect" in a smooth way. The result is a C^2 function over the range of points $\{x_1, ..., x_n\}$. Denote the values of the function at the grid points by $\{y_1, ..., y_n\}$. A cubic spline interpolation says that the value of the function at a point between the grid points

is given by

$$
\begin{aligned}
y &= Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1} \\
A &= \frac{x_{j+1} - x}{x_{j+1} - x_j} \\
B &= 1 - A = \frac{x - x_j}{x_{j+1} - x_j} \\
C &= \frac{1}{6}\left(A^3 - A\right)\left(x_{j+1} - x_j\right)^2 \\
D &= \frac{1}{6}\left(B^3 - B\right)\left(x_{j+1} - x_j\right)^2.
\end{aligned}
$$

Taking derivatives we obtain

$$
\begin{aligned}
y' &= -\frac{y_j}{x_{j+1} - x_j} + \frac{y_{j+1}}{x_{j+1} - x_j} - \frac{1}{6}\left(3A^2 - 1\right)\left(x_{j+1} - x_j\right) y''_j + \frac{1}{6}\left(3B^2 - 1\right)\left(x_{j+1} - x_j\right) y''_{j+1} \\
y'' &= Ay''_j + By''_{j+1}.
\end{aligned}
$$

However, we do not currently know the values for y''. To solve for them, we impose smoothness conditions on the segments. It is also potentially important to remember that these values are not the second derivatives of the true function, but rather artificial ones constructed to satisfy our smoothness requirements; they may or may not accurately represent the second derivatives at the nodes. The same caveat applies to the first derivatives.

We set the first derivative equation (evaluated at x_j in the interval (x_{j-1}, x_j)) equal to the first derivative equation, but evaluated at x_j in the interval (x_j, x_{j+1}). We obtain a system of $N - 2$ equations of the form

$$
\frac{x_j - x_{j-1}}{6} y''_{j-1} + \frac{x_{j+1} - x_j}{3} y''_j + \frac{x_{j+1} - x_j}{6} y''_{j+1} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}};
$$

this system contains the N unknowns y''_j. Thus, we have a 2-parameter family of solutions. One way to fully-specify the solution is to add conditions about first-derivatives at the boundaries,

$$
y'_1 = y'_N = 0,
$$

which yields the "natural spline" formulation. For economic problems, where concavity can often be proven, a natural spline is generally a poor choice. A better choice is the "not-a-knot" condition which sets first-derivatives at the boundaries to be multiples of first-derivatives at the

next interior nodes:

$$y'_1 = \alpha y'_2$$
$$y'_N = \omega y'_{N-1};$$

these solutions then satisfy montonicity of the first derivatives even on the boundary. For a globally concave function we require $\alpha > 1$ and $\omega < 1$. We will see below that we can impose any side conditions we want, provided they are linear.

The above system is "tridiagonal"; it is a linear system of the form

$$\begin{bmatrix} b_1 & c_1 & 0 & \cdots & 0 \\ a_2 & b_2 & c_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} y''_1 \\ y''_2 \\ \vdots \\ y''_{n-1} \\ y''_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_{n-1} \\ r_n \end{bmatrix}$$

and can be efficiently solved using a specialized routine. Note that storage requirements have been doubled relative to linear splines, since we need to store the values and the artificial second derivatives.

Note that we can rewrite the spline interpolation in polynomial form:

$$\begin{aligned} y &= a_j + b_j x + c_j x^2 + d_j x^3 \\ a_j &= \frac{1}{x_{j+1} - x_j} \left(x_{j+1} y_j - x_j y_{j+1} - \frac{z_j}{6} \left(x_j^2 x_{j+1} - 2 x_j x_{j+1}^2 \right) - \frac{z_{j+1}}{6} \left(2 x_j^2 x_{j+1} - x_j x_{j+1}^2 \right) \right) \\ b_j &= \frac{1}{x_{j+1} - x_j} \left(y_{j+1} - y_j + \frac{z_j}{6} \left(x_j^2 - 2 x_j x_{j+1} - 2 x_{j+1}^2 \right) + \frac{z_{j+1}}{6} \left(2 x_j^2 + 2 x_j x_{j+1} - x_{j+1}^2 \right) \right) \\ c_j &= \frac{1}{2} \frac{y''_j x_{j+1} - y''_{j+1} x_j}{x_{j+1} - x_j} \\ d_j &= \frac{1}{6} \frac{y''_{j+1} - y''_j}{x_{j+1} - x_j}; \end{aligned}$$

for some applications this form may be more convenient.

There is nothing special about cubic splines, however, other than they are the lowest order spline that is C^2. One can have higher order splines as well, such as quintic splines. These splines deliver a "pentadiagonal" linear system and a C^4 interpolant; we now need four additional conditions, which makes these high-order splines hard to use, but if the second derivative is of

particular interest (say, because we're trying to accurately capture risk aversion measures) then they may come in handy. For example, if we need to compute the Schwarzian derivative

$$S(x) = \frac{D^3 f(x)}{Df(x)} - \frac{3}{2}\left(\frac{D^2 f(x)}{Df(x)}\right)^2,$$

which appeared in our discussion of stable limit cycles in dynamic systems, higher-order splines would obviously be critical. Similarly, the coefficient of relative prudence

$$P(c) = -\frac{D^3 u(c)}{D^2 u(c)} c,$$

which measures the strength of the precautionary savings effect, would require a smooth second derivative.

We do not need to use the standard side conditions; in fact, we can impose any side conditions we want on the cubic spline construction. Suppose there are $n+1$ nodes defining n elements. At each node $i \in \{2, ..., n+1\}$ we must have

$$y_i = a_i + b_i x_i + c_i x_i^2 + d_i x_i^3$$

and for $i \in \{1, ..., n\}$ we must have

$$y_i = a_{i+1} + b_{i+1} x_i + c_{i+1} x_i^2 + d_{i+1} x_i^3.$$

We now have $2n$ conditions. To make the approximation C^2 we need the first and second derivatives to be continuous:

$$b_i + 2c_i x_i + 3d_i x_i^2 = b_{i+1} + 2c_{i+1} x_i + 3d_{i+1} x_i^2$$

for each node $i \in \{1, ..., n\}$ and

$$2c_i + 6d_i x_i = 2c_{i+1} + 6d_{i+1} x_i$$

for each node $i \in \{1, ..., n\}$.

Now add two boundary conditions. First, suppose that we want the derivative at x_1 to be $\omega > 1$ times the slope at x_2; the extra condition is

$$b_2 + 2c_2 x_2 + 3d_2 x_2^2 = \omega\left(b_1 + 2c_1 x_1 + 3d_1 x_1^2\right).$$

If we also want the slope at x_n to be $\alpha < 1$ times the slope at x_{n-1}, then we impose

$$b_n + 2c_n x_n + 3d_n x_n^2 = \alpha \left(b_{n-1} + 2c_{n-1} x_{n-1} + 3d_{n-1} x_{n-1}^2 \right).$$

We now stack the coefficients

$$\begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ \vdots \\ a_n \\ b_n \\ c_n \\ d_n \end{bmatrix}.$$

The first condition then yields

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 & 0 & \cdots & 0 \end{bmatrix}$$

with constant term y_1. There are n of these. The second condition yields

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & x_1 & x_1^2 & x_1^3 & 0 & \cdots & 0 \end{bmatrix}$$

with constant term y_1. Again, there are n of these. The third condition has

$$\begin{bmatrix} 0 & 1 & 2x_1 & 3x_1^2 & 0 & -1 & -2x_1 & -3x_1^2 & 0 & \cdots & 0 \end{bmatrix}$$

with no constant term and there are $n-1$ of these. The fourth condition has

$$\begin{bmatrix} 0 & 0 & 2 & 6x_1 & 0 & 0 & -2 & -3x_1^2 & 0 & \cdots & 0 \end{bmatrix}$$

with no constant term and there are $n-1$ of these. Finally, the last two terms are

$$\begin{bmatrix} 0 & \omega & 2\omega x_1 & 3\omega x_1^2 & 0 & -1 & -2x_2 & -3x_2^2 & 0 & \cdots & 0 \end{bmatrix}$$

and

$$\begin{bmatrix} 0 & \cdots 0 & 0 & 0 & -\alpha & -2\alpha x_{n-1} - & 3\alpha x_{n-1}^2 & 0 & 1 & 2x_n & 3x_n^2 \end{bmatrix}$$

with zero constant terms.

Thus, the constant vector is

$$\begin{bmatrix} y \\ y \\ 0 \\ 0 \end{bmatrix}$$

with each part of length n. Now assemble the entire thing into the linear equation

$$A \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} = \begin{bmatrix} y \\ y \\ 0 \\ 0 \end{bmatrix}$$

and solve for z.

This approach can be extended to higher-order splines in a straightforward way, and can accomodate other boundary conditions provided they are linear. One such method would be to fix the slopes using extra information (such as an envelope condition); we can exactly match this condition at any two nodes.

10.7.3 B-splines

Given a set of knots $\{t_i\}$, we can construct a basis spline (B-spline) through them; a B-spline is a linear combination of polynomials defined only over a small number of elements. The ith 0-order B-spline satisfies

$$B_i^0(x) = \begin{cases} 1 & \text{if } t_i \le x \le t_{i+1} \\ 0 & \text{otherwise} \end{cases}.$$

A simple recursion computes the higher-order B-splines:

$$B_i^d(x) = \frac{x - t_i}{t_{i+d-1} - t_i} B_i^{d-1}(x) + \frac{t_{i+d-1} - x}{t_{i+d-1} - t_{i+1}} B_{i+1}^{d-1}(x).$$

To interpolate a set of points $(x_j, y_j)_{j=1}^n$, we first compute the coefficients in the linear system

$$y_j = \sum_{i=1}^{n} c_i^d B_i^d(x_j);$$

these coefficients define the B-spline of order d. Note that the B-spline basis functions are only nonzero on $d+1$ elements. We then have

$$f(x) = \sum_{i=1}^{n} c_i^d B_i^d(x).$$

The derivatives of a B-spline over n distinct knots are continuous to order $n-2$; if a knot occurs r times only the first $n-r-1$ derivatives are continuous across that knot. B-splines do not preserve shape in general.

B-splines are common in finite element methods (McGrattan 1996), because they lead to a sparse linear system.

10.7.4 Cubic Hermite Spline

The problem with cubic and B-splines is that they may not preserve shape. Linear splines will generate a function that is monotonic and concave (or convex) if the function values at the nodes display these properties; cubic splines, which use higher-order polynomials, may not because cubics are not generally monotonic or concave functions. But of course linear splines are not smooth, which can cause problems for optimization, so we would like some way to get both. Piecewise cubic Hermite splines are the answer – they are C^1 and will be monotone or concave if the data support it. Note that differently from the polynomial shape preservation method, if the data are not monotone or concave the resulting spline will not force those features on them.

The piecewise cubic Hermite spline is given by the formula

$$p(t) = h_{00}(t)\, p_0 + h_{10}(t)\, hm_0 + h_{01}(t)\, p_1 + h_{11}(t)\, hm_1$$

with

$$\begin{aligned} h_{00}(t) &= 2t^3 - 3t^2 + 1 \\ h_{10}(t) &= t^3 - 2t^2 + t \\ h_{01}(t) &= -2t^3 + 3t^2 \\ h_{11}(t) &= t^3 - t^2, \end{aligned}$$

where p is the level, m is the slope, '0' is the left endpoint of the current element, '1' is the right

endpoint, and

$$h = x_{k+1} - x_k$$
$$t = \frac{x - x_k}{h}$$
$$x \in [x_k, x_{k+1}].$$

Shape-preservation comes in through selection of the slope coefficients. If we have the slopes directly, then we are done. If not, let

$$\Delta_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k}$$

for data (x_k, y_k), $k = 1, ..., n-1$. Then set

$$m_k = \frac{\Delta_{k-1} + \Delta_k}{2}$$

for $k = 2, ..., n-1$, and

$$m_1 = \Delta_1$$
$$m_n = \Delta_{n-1}.$$

For $k = 1, ..., n-1$, if $\Delta_k = 0$ then $m_k = m_{k+1} = 0$. Otherwise, let $\alpha_k = \frac{m_k}{\Delta_k}$ and $\beta_k = \frac{m_{k+1}}{\Delta_k}$. If $\alpha_k^2 + \beta_k^2 > 9$, reset $m_k = \tau_k \alpha_k \Delta_k$ and $m_{k+1} = \tau_k \beta_k \Delta_k$, where

$$\tau_k = \frac{3}{\sqrt{\alpha_k^2 + \beta_k^2}}.$$

Storage requirements are the same as for cubic splines – we store the values and the artificial first derivatives. In general I prefer to use piecewise cubic Hermite splines – storage is the same as cubic splines (values plus slopes) and the shape preservation is usually more important than the extra degree of smoothness. There do exist methods for constructing shape-preserving splines that are C^2, like regular cubic splines, but these splines do not exist for every set of monotone and concave nodes.

If the data is oscillatory, pchip splines are "over-flattened" around local extrema relative to cubic splines, implying that all local extrema occur only at nodes. Note also that, if you know the value of the derivatives at the nodes, you do not need to construct them; that is, the pchip spline will interpolate Hermite data.

10.7.5 Modified Akima Splines

An effective spline for oscillatory data is a modified Akima spline. Given data (x_i, y_i), define

$$m_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$$

and impose that the slope of the spline is

$$s_i = \frac{|m_{i+1} - m_i| \, m_{i-1} + |m_{i-1} - m_{i-2}| \, m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|}.$$

The spline on $[x_i, x_{i+1}]$ is then uniquely determined by

$$\begin{aligned} P(x_i) &= y_i \\ P(x_{i+1}) &= y_{i+1} \\ DP(x_i) &= s_i \\ DP(x_{i+1}) &= s_{i+1}. \end{aligned}$$

That is, we solve the linear system

$$\begin{bmatrix} 1 & x_i & x_i^2 & x_i^3 \\ 1 & x_{i+1} & x_{i+1}^2 & x_{i+1}^3 \\ 0 & 1 & 2x_i & 3x_i^2 \\ 0 & 1 & 2x_{i+1} & 3x_{i+1}^2 \end{bmatrix} \begin{bmatrix} a_i \\ b_i \\ c_i \\ d_i \end{bmatrix} = \begin{bmatrix} y_i \\ y_{i+1} \\ s_i \\ s_{i+1} \end{bmatrix}$$

for the coefficients that apply on each element. Akima splines reduce unwanted oscillations on the interior of the elements relative to cubic splines and are less aggressively "flattened" near local extrema relative to pchip splines. Note that to obtain the coefficients on a given element we do not need any information from other elements, unlike cubic splines; the cost of this simpler system is that the splines are only C^1, not C^2. Akima splines do not preserve shape in general, but they do allow local extrema to occur between nodes.

10.7.6 Rational Hermite Spline

If we know the value of the derivatives at the nodes, we can implement an alternative shape-preserving method called rational Hermite approximation; this approximation method will also

interpolate the derivatives of f. Given Hermite data (x_i, v_i, s_i), the function $f(x)$ can be approximated by

$$f(x) = c_{i1} + c_{i2}(x - x_i) + \frac{c_{i3}c_{i4}(x - x_i)(x - x_{i+1})}{c_{i3}(x - x_i) + c_{i4}(x - x_{i+1})}$$

$$x \in [x_i, x_{i+1}]$$

$$c_{i1} = v_i$$

$$c_{i2} = \frac{v_{i+1} - v_i}{x_{i+1} - x_i}$$

$$c_{i3} = s_i - c_{i2}$$

$$c_{i4} = s_{i+1} - c_{i2}.$$

This approximation is C^1 with first derivative

$$Df(x) = c_{i2} + c_{3i}c_{4i}\frac{c_{3i}(x - x_i)^2 + c_{4i}(x - x_{i+1})^2}{(c_{3i}(x - x_i) + c_{4i}(x - x_{i+1}))^2};$$

note that

$$Df(x_i) = c_{i2} + c_{3i} = s_i,$$

so that the approximation method interpolates the first derivative function as well. As with other shape-preserving splines, rational Hermite splines are only C^1.

10.7.7 Shape-Preserving Quadratic Spline

An alternative method of using Hermite data is to construct quadratic shape-preserving splines, sometimes called Schumaker splines. The goal is to construct a piecewise quadratic function $s_i \in C^1([x_i, x_{i+1}])$ such that

$$s(x_i) = f(x_i)$$

$$s(x_{i+1}) = f(x_{i+1})$$

$$Ds(x_i) = Df(x_i)$$

$$Ds(x_{i+1}) = Df(x_{i+1})$$

for each grid point. Since in general we need three points to pin down our quadratic function, we will add a **knot** to the interval; denote the knot by $\xi_i \in (x_i, x_{i+1})$. For every knot on the

interval there is a unique quadratic that satisfies these four conditions given by

$$s(x) = \begin{cases} A_1 + B_1(x - x_i) + C_1(x - x_i)^2, & x \in [x_i, \xi_i] \\ A_2 + B_2(x - \xi_i) + C_2(x - \xi_i)^2, & x \in [\xi_i, x_{i+1}] \end{cases}$$

where

$$\begin{aligned} A_1 &= f(x_i) \\ B_1 &= Df(x_i) \\ C_1 &= \frac{\overline{f} - Df(x_i)}{2\alpha} \\ A_2 &= A_1 + \alpha B_1 + \alpha^2 C_1 \\ B_2 &= \overline{f} \\ C_2 &= \frac{Df(x_{i+1}) - \overline{f}}{2\beta} \\ \overline{f} &= \frac{2(f(x_{i+1}) - f(x_i) - (\alpha Df(x_i) + \beta Df(x_{i+1})))}{x_{i+1} - x_i} \\ \alpha &= \xi_i - x_i \\ \beta &= x_{i+1} - \xi_i. \end{aligned}$$

We then need a strategy for choosing the value of the knot ξ_i. We will select ξ_i to preserve the concavity or convexity of the function we are approximating. Let

$$\delta = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}.$$

If

$$(Df(x_i) - \delta)(Df(x_{i+1}) - \delta) \geq 0$$

then set

$$\xi_i = \frac{1}{2}(x_i + x_{i+1});$$

else if

$$|Df(x_{i+1}) - \delta| < |Df(x_i) - \delta|$$

then set

$$\begin{aligned} \xi_i &= \frac{1}{2}\left(x_i + \overline{\xi}\right) \\ \overline{\xi} &= x_i + \frac{2(x_{i+1} - x_i)(Df(x_{i+1}) - \delta)}{Df(x_{i+1}) - Df(x_i)}; \end{aligned}$$

else if

$$|Df(x_{i+1}) - \delta| \geq |Df(x_i) - \delta|$$

then set

$$\xi_i = \frac{1}{2}\left(x_{i+1} + \underline{\xi}\right)$$
$$\underline{\xi} = x_{i+1} + \frac{2(x_{i+1} - x_i)(Df(x_i) - \delta)}{Df(x_{i+1}) - Df(x_i)}.$$

Using this approach guarantees that our numerical function will be concave or convex. This method will also interpolate the derivatives correctly, but is again only C^1.

10.7.8 Extrapolation

As a general rule you should avoid extrapolation (evaluation of the spline at values of x below x_1 or above x_n) – linear splines extrapolate safely (they do not diverge rapidly) but there is no guarantee they are accurate, and higher-order splines (and polynomials for that matter) diverge rapidly outside the set of nodes (and Chebyshev polynomials diverge extremely rapidly). Extend your grid if necessary to eliminate the chance of having to extrapolate; for some problems this extension may require special accommodations for certain points (such as extending the growth model to zero capital) but that is the price we have to pay to get good approximations.

Extrapolating splines constrain the behavior of the spline approximation outside the range of the grid. In this case, you specify the order of the polynomial that computes the function beyond the grid, which should be of lower order than the spline used on the interior of the grid. Then we fit that polynomial to the first (or last) element, depending on the direction of extrapolation; for example, for a cubic spline on $\{(x_i, f_i)\}$ we can extrapolate below x_1 using a quadratic $g(x)$ that satisfies

$$g(x_1) = f_1$$
$$g(x_2) = f_2$$

and that equals the spline interpolant at

$$x = \frac{x_1 + x_2}{2}.$$

The resulting coefficients solve

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & \frac{x_1+x_2}{2} & \left(\frac{x_1+x_2}{2}\right)^2 \end{bmatrix} \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ s_{1,2} \end{bmatrix}.$$

These splines are not guaranteed to be accurate (since we have no information about the function beyond the grid), and they still diverge, but because the order is lower they diverge less quickly. Thus, we get the accuracy on the grid of the cubic spline and the ability to extrapolate a small amount.

10.7.9 Multidimensional Splines

Multidimensional extensions exist for all types of splines. However, there is no easy way to preserve shape in more than one dimension (and no guaranteed way in more than two), so a lot of the benefits of splines disappear. Bilinear interpolation is the easiest to use, and extends quite naturally:

$$f(x,y) = (1-A)(1-B) f(x_i, y_j) + A(1-B) f(x_{i+1}, y_j) + (1-A) B f(x_i, y_{j+1}) + AB f(x_{i+1}, y_{j+1})$$

where

$$x \in [x_i, x_{t+1}]$$
$$y \in [y_j, y_{j+1}]$$

for some "rectangular" grid in x and y and

$$A = \frac{x - x_i}{x_{i+1} - x_i}$$
$$B = \frac{y - y_i}{y_{j+1} - y_j}.$$

A more efficient approach is to interpolate the function as

$$f(x,y) = C_{ij} \begin{bmatrix} 1 \\ x \\ y \\ xy \end{bmatrix}$$

where

$$C_{ij} = \begin{bmatrix} 1 & x_i & y_j & x_i y_j \\ 1 & x_i & y_{j+1} & x_i y_{j+1} \\ 1 & x_{i+1} & y_j & x_{i+1} y_j \\ 1 & x_{i+1} & y_{j+1} & x_{i+1} y_{j+1} \end{bmatrix}^{-1} \begin{bmatrix} f(x_i, y_j) \\ f(x_i, y_{j+1}) \\ f(x_{i+1}, y_j) \\ f(x_{i+1}, y_{j+1}) \end{bmatrix}.$$

If we have to evaluate the function many times, this method can be significantly faster. This procedure extends to higher dimensions, but the conditioning number of the matrix that needs to be inverted gets worse as the dimension gets larger.

What is a conditioning number? Take a matrix A and the ordered spectrum of eigenvalues $\{\lambda_1, ..., \lambda_n\}$ from smallest to largest. The **conditioning number** is

$$\Lambda = \frac{|\lambda_n|}{|\lambda_1|},$$

the ratio of the largest eigenvalue to the smallest. If this value is large, the matrix is numerically singular, even if it is nonsingular in reality. Rescaling can sometimes fix conditioning issues, but not always.

Bicubic interpolation assumes the function, the first derivatives, and the cross-derivative are known at the nodes. We then interpolate the function as

$$f(x, y) = \begin{bmatrix} 1 & x & x^2 & x^3 \end{bmatrix} D_{ij} \begin{bmatrix} 1 \\ y \\ y^2 \\ y^3 \end{bmatrix}$$

where

$$D_{ij} = FV_{ij}F^T$$

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

$$V_{ij} = \begin{bmatrix} f(x_i, y_j) & f(x_i, y_{j+1}) & D_y f(x_i, y_j) & D_y f(x_i, y_{j+1}) \\ f(x_{i+1}, y_j) & f(x_{i+1}, y_{j+1}) & D_y f(x_{i+1}, y_j) & D_y f(x_{i+1}, y_{j+1}) \\ D_x f(x_i, y_j) & D_x f(x_i, y_{j+1}) & D_{xy} f(x_i, y_j) & D_{xy} f(x_i, y_{j+1}) \\ D_x f(x_{i+1}, y_j) & D_x f(x_{i+1}, y_{j+1}) & D_{xy} f(x_{i+1}, y_j) & D_{xy} f(x_{i+1}, y_{j+1}) \end{bmatrix}.$$

Two-dimensional cubic splines are usually done "one direction at a time" – we construct the second derivatives in one direction and store them. When we need to evaluate the spline, we calculate the value at y (for example) for each node in the x direction, construct a second-derivative vector through those points, and then evaluate the spline in the x direction. Clearly this approach is slow, and the shape problems are even worse in two dimensions; as noted, in general they cannot be corrected without simply adding more nodes. Plus, while the grid need not be uniformly-spaced, it must have the same nodes in each direction at each point in every other direction (rectangular), which is a serious limitation when we want to deal with high dimensions.

B-splines extend to higher dimensions using product rules:

$$f(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_i^{d_1} g_j^{d_2} B_i^{d_1}(x) B_j^{d_2}(y).$$

There is a method of constructing a 2-d shape-preserving spline interpolation based on Bernstein polynomials developed by Costantini (see Wang and Judd 1999). This method chooses the order of the spline for you, in order to impose monotonicity and/or concavity. There does not exist a general method to preserve shape in more than two dimensions beyond merely adding additional points into the grid to limit the gaps over which the polynomials can oscillate.

10.8 Meshless Interpolation

In high dimensions, the ratio of the volume of a sphere to the volume of a bounding hypercube goes to zero as the dimension goes to infinity, so solving models on a sphere (where the set of

nodes cannot be rectangular) has huge advantages (note that a sphere is simply a transformation of an ellipse). For a proof, take the ratio of the volume of a hypersphere to that of a hypercube:

$$\frac{V\left(S^d\right)}{V\left(C^d\right)} = \frac{\left(\frac{\sqrt{\pi}}{2}\right)^d}{\frac{d}{2}\Gamma\left(\frac{d}{2}\right)}.$$

Now take the dimension to infinity and use Stirling's approximation to the factorial function

$$n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

to obtain

$$\lim_{d\to\infty}\left(\frac{\left(\frac{\sqrt{\pi}}{2}\right)^d}{\frac{d}{2}\Gamma\left(\frac{d}{2}\right)}\right) \approx \lim_{d\to\infty}\left(\frac{\left(\sqrt{\pi}\right)^{d-1} e^{\frac{d}{2}}}{2^{\frac{d}{2}}\left(\sqrt{d}\right)^{d+1}}\right) = 0.$$

Thus, if we solve a model on a high-dimensional rectangular grid, most of our points will be "wasted" computing the model on regions of the state space it will not visit. How relevant is this calculation – that is, how quickly does it converge to zero? The answer is very quickly: for $d = 10$ the relative volume is 0.0025, so anything larger than 10 the volume may as well be zero.

In addition, the endogeneous grid method, which has speed advantages and will be discussed later, generally will not produce a rectangular grid. Therefore we now discuss interpolation approaches that do not require rectangular grids (technically they deal with "scattered data"). Polynomial projection methods are one such method, but we have already discussed their strengths and weaknesses. Splines do not interpolate scattered data, but some similar functions do.

Note that scattered interpolators have advantages over methods that convert certain irregular domains (spheres, for example) into rectangular ones, because those methods generally lead to very tightly packed grid points. For example, consider the following procedure that maps a set of rectangular points into a simplex. Take

$$X \times X = [0,1] \times [0,1]$$

as the relevant rectangle. We want to map each point in $X \times X$ into a point in the simplex

$$\Sigma = \left\{(\omega_1, \omega_2, \omega_3) : 0 \leq \omega_i \leq 1, \omega_1 + \omega_2 + \omega_3 = 1\right\}.$$

Define the variables

$$z_1 = g\left(x_1\right) = -\log\left(x_1\right)$$

$$z_2 = g\left(x_2\right) = -\log\left(x_2\right).$$

Then define

$$\begin{aligned}\omega_1 &= h_1(z_1, z_2) = \frac{z_1}{1 + z_1 + z_2} \\ \omega_2 &= h_2(z_1, z_2) = \frac{z_2}{1 + z_1 + z_2} \\ \omega_3 &= h_3(z_1, z_2) = \frac{1}{1 + z_1 + z_2}.\end{aligned}$$

Then the mapping f defined by

$$\begin{aligned}\omega_i &= h_i(g(x_1), g(x_2)) \\ &= \frac{z_i}{1 + z_1 + z_2} \\ &= \frac{-\log(x_i)}{1 - \log(x_1) - \log(x_2)}\end{aligned}$$

takes points in the unit rectangle and maps them into the unit simplex. Thus, you can think of "starting" with a rectangular set of points in an "unnatural" set of state variables, then this procedure maps these state variables into the natural ones. If you look at these points, they are bunched up in the simplex, which leads to wasteful calculations (see Figure).

A better alternative is the following shearing map (but note that it is not differentiable). Take $X \times X = [0, 1] \times [0, 1]$. For each point (x_1, x_2) compute

$$m = \max\{x_1, x_2\}$$

$$s = x_1 + x_2.$$

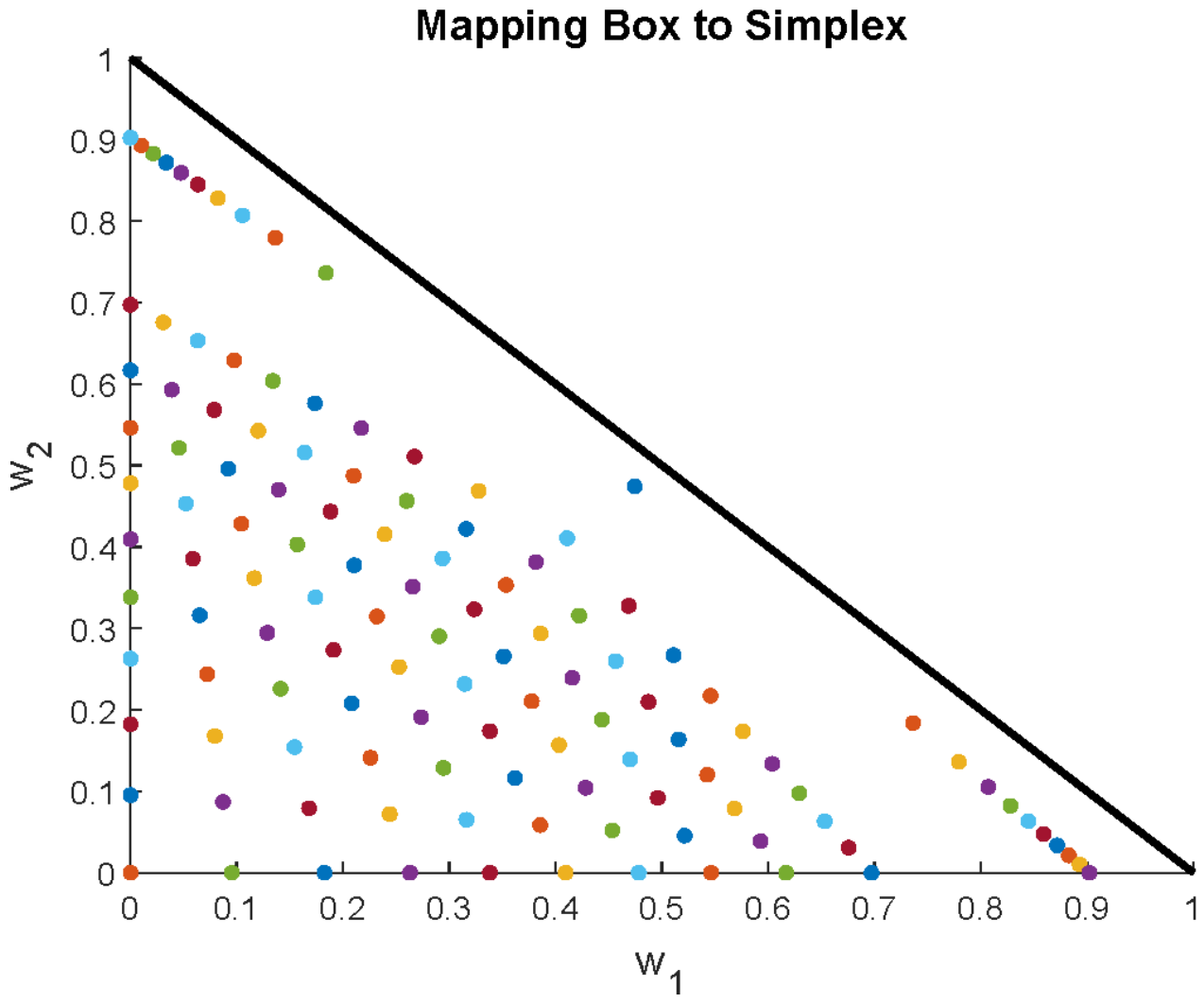
Then we map $(x_1, x_2) \rightarrow \left(x_1 \frac{m}{s}, x_2 \frac{m}{s}\right)$. These points are more uniformly distributed. Note however that the origin must be dealt with separately by defining it to be a fixed point.

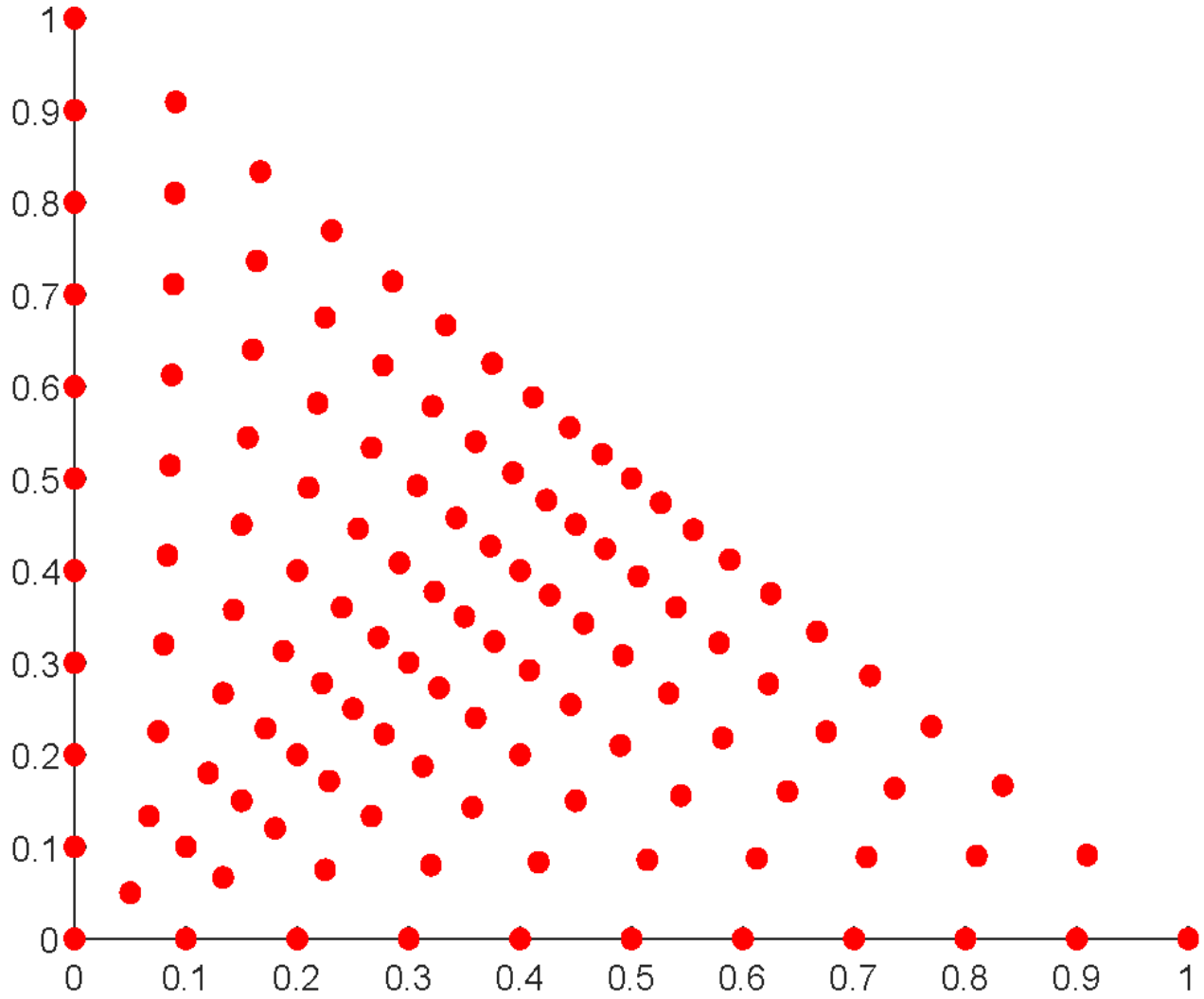
We also can map rectangles into ellipses, which is a useful approach given that many models will generally live on ellipses. This mapping has two stages: first we take points in $[-1, 1] \times [-1, 1]$ and map them to the unit circle:

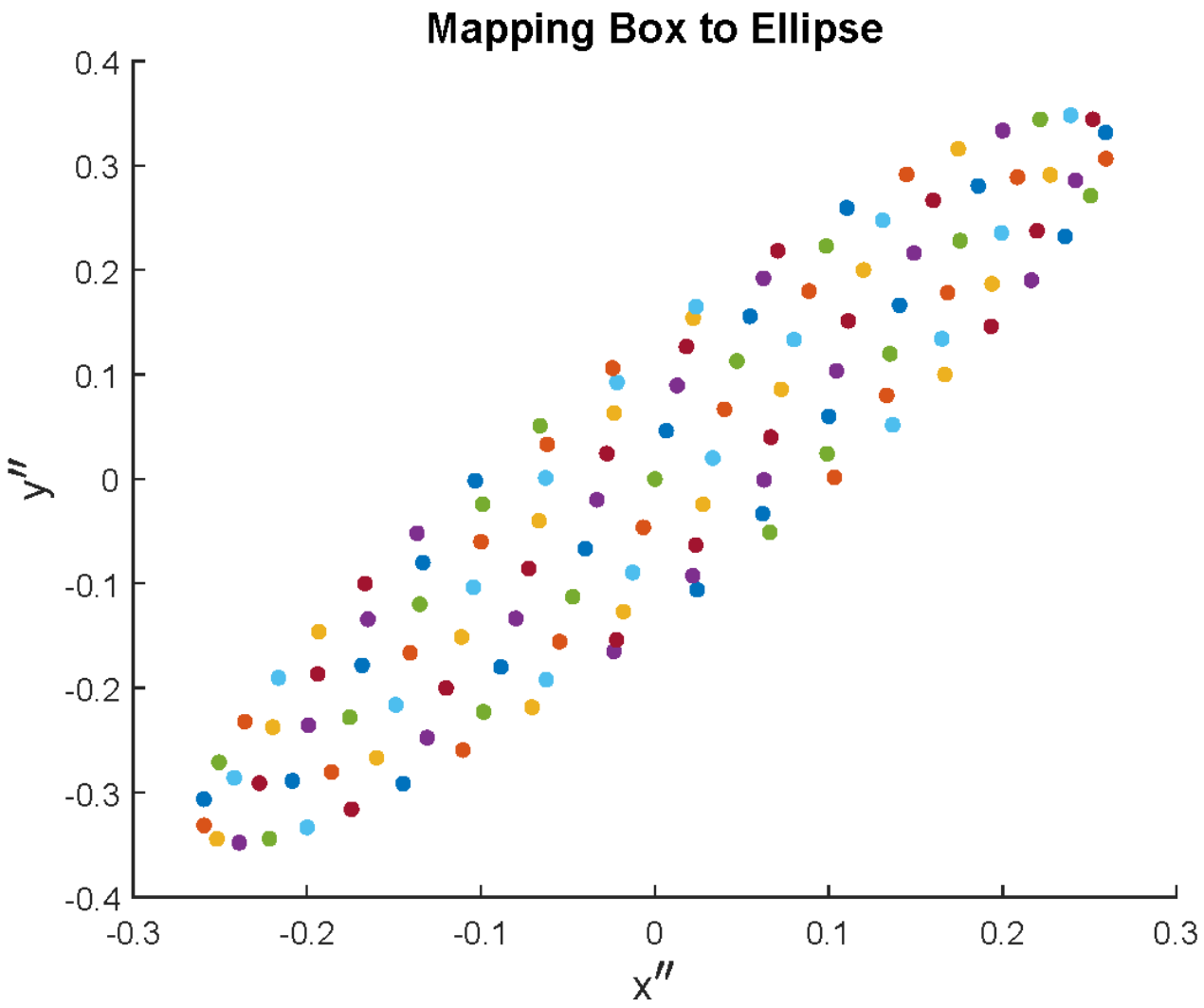
$$(x, y) \longmapsto (x', y') = \left(x\sqrt{1 - \frac{y^2}{2}}, y\sqrt{1 - \frac{x^2}{2}}\right).$$

We then "stretch" these points:

$$(x', y') \longmapsto (x'', y'') = (p_{xx}x' + p_{xy}y', p_{yx}x' + p_{yy}y')$$


Mapping Box to Simplex
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
w2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
w1


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



for some set of weights $(p_{xx}, p_{xy}, p_{yx}, p_{yy})$. Note that this mapping has better "coverage" – the points are not as bunched up – but the shape is regular (see the figure). Trying to map into less regular shapes is what becomes challenging.

10.8.1 Shepard's Inverse-Weighting Method

Shepard's method does not require a rectangular grid; instead, it will construct approximating functions over any set of points. Given a set of "data" $\{x_1, ..., x_n\}$ we build our approximation at each point by "averaging" the value of the function at nearby points (just like a linear spline does), but we will use more than simply the vertices of the element the point is located on (because we never construct a grid, Shepard's method and similar methods are called meshless). At x the

value of the function is approximated by

$$f(x) = \frac{\sum_{i=1}^{n} w_i(x) f(x_i)}{\sum_{i=1}^{n} w_i(x)}$$

where

$$w_i(x) = \frac{1}{\|x - x_i\|^p}$$

is the "inverse distance weight" and $\|\cdot\|^p$ is the L^p norm. Values that are closer to x get more weight than those more distant when we average the function values. One usually defines a "radius" R that completely excludes points that are too far away, in order to conserve on computational time, leading to the modified Shepard's method with weights

$$w_i(x) = \frac{\max\{0, R - \|x - x_i\|^p\}^2}{R\|x - x_i\|^p},$$

where R is chosen to include a fixed number of points (usually 19). In N dimensions we must set $p \geq N$ to avoid perverse results that give far away points more weight. The most advanced versions assume that the function value is given by a Taylor expansion around x_k with constant value f_k, denoted P_k, and then formulates the least squares problem

$$\min\left\{\sum_{i=1, i\neq k}^{n} \omega_{ik}\left(P_k(x_i) - f_k\right)^2\right\}$$

with weights

$$\omega_{ik} = \frac{\max\{0, R_k - \|x_k - x_i\|^p\}^2}{R_k\|x_k - x_i\|^p}$$
$$R_k = \frac{\max_{i,j}\{\|x_i - x_j\|^p\}}{2}\sqrt{\frac{13}{n}}.$$

Depending on the order of the Taylor expansion we get linear, quadratic, or cubic interpolants. Costs become prohibitively high for nonlinear versions if N is large (note N is the dimension of the space, not the number of points, which is given by n), so linear interpolants are generally the only option for relatively large N.

10.8.2 Tesselation and Barycentric Coordinates

Rectangular grid methods "tile" the space of nodes using hypercubes. An alternative method for tiling a space is a tesselation, which is covering the space with simplices instead of hypercubes

(remember, a simplex in n dimensional space is a collection of $n+1$ points, so in two dimensions we have a triangle and in three we have a tetrahedron). Given a set of simplices, barycentric points are used to weight the value of the function at each node to interpolate values inside the particular simplex of interest. In two dimensions, any point (x, y) can be written as

$$x = \lambda_1 x_1 + \lambda_2 x_2 + (1 - \lambda_1 - \lambda_2)\, x_3$$
$$y = \lambda_1 y_1 + \lambda_2 y_2 + (1 - \lambda_1 - \lambda_2)\, y_3.$$

Rearranging we have

$$\lambda_1 (x_1 - x_3) + \lambda_2 (x_2 - x_3) + (x_3 - x) = 0$$
$$\lambda_1 (y_1 - y_3) + \lambda_2 (y_2 - y_3) + (y_3 - y) = 0$$

which can be written as

$$T\lambda = r - r_3.$$

Therefore

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = T^{-1} \left(\begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x_3 \\ y_3 \end{bmatrix} \right)$$
$$\lambda_3 = 1 - \lambda_1 - \lambda_2.$$

Then point $\begin{bmatrix} x \\ y \end{bmatrix}$ lies in a given triangle if and only if

$$0 < \lambda_i < 1$$

for $i \in \{1, 2, 3\}$. Point $\begin{bmatrix} x \\ y \end{bmatrix}$ is on a edge if and only if

$$0 \leq \lambda_i \leq 1$$

for $i \in \{1, 2, 3\}$ with equality for some i. These results are just an application of Carathéodory's theorem that any member of the convex hull of $d+1$ points in $\mathcal{R}^d$ can be written as a convex combination of those $d+1$ points.

The function interpolated at $\begin{bmatrix} x \\ y \end{bmatrix}$ is then

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \lambda_1 f\left(\begin{bmatrix} x_1 \\ y_1 \end{bmatrix}\right) + \lambda_2 f\left(\begin{bmatrix} x_2 \\ y_2 \end{bmatrix}\right) + (1-\lambda_1-\lambda_2) f\left(\begin{bmatrix} x_3 \\ y_3 \end{bmatrix}\right).$$

Higher dimensions are straightforward generalizations.

The next step is to construct our set of triangles. Since there are many possible triangularizations for any set of points, we want to use a "good one". Specifically, we want our triangles to have wide angles so that interpolation is done generally using the closest set of points; thin triangles lead to points in other triangles being closer to some interior points and therefore lead to interpolation inaccuracies. The most useful triangularization is called a Delaunay tesselation.

Definition 110 *Take two triangularizations of S, denoted T_1 and T_2. Let $\{\alpha_{i,1}, ..., \alpha_{i,3n}\}$ be the ordered sequence of angles in T_i. If there exists $k \in \{1, ..., 3n\}$ such that $\alpha_{1,j} = \alpha_{2,j}\ \forall j < k$ and $\alpha_{1,k} > \alpha_{2,k}$, then we say T_1 is **fatter than** T_2 (written $T_1 > T_2$).*

Definition 111 *Let e be an edge of a triangularization T_1 and let Q be the quadrilateral in T_1 formed by the two triangles sharing e. If Q is convex, let T_2 be the triangularization after flipping edge e in T_1 (flipping an edge means replacing e with the other diagonal from the quadrilateral). e is a **legal edge** if $T_1 \geq T_2$ and e is an **illegal edge** if $T_1 < T_2$.*

Definition 112 *A **Delaunay tesselation** $D(S)$ has only legal edges.*

Every collection of points has a unique Delaunay tesselation (provided the points are in **general position**, which means that no four points lie on the same circle; that situation would typically not happen and could easily be fixed by slightly moving some nodes). Delaunay tesselations have the property that no point is in the interior of the circumcircle of any triangle, which means that triangles are "not skinny" (in fact Delaunay tesselations are the fattest possible triangularization). This property generally implies that the triangle that will be used to interpolate a point has vertices that are the three closest ones to that point (formally this type of triangularization is called a **Pitteway triangularization** and may not exist, but when it exists it coincides with the Delaunay tesselation). To understand this fact, we need to use some results in geometry that link Delaunay tesselations to objects called Voronoi regions.

Definition 113 *The* ***Voronoi region*** $V(p)$ *of a point* p *in a collection* S *is*

$$V(p) = \left\{x \in \mathcal{R}^2 : \|x - p\| \leq \|x - q\| \ \ \forall q \in S\right\}.$$

That is, a Voronoi region is the region of points in the plane that are closer to p than any other element of S. All Voronoi regions are convex, and their 'dual graph' (the dual graph of S is the collection of edges that connect each point in S with the points in S located in the adjacent Voronoi regions) is the Delaunay tesselation. This result is behind the idea that the Delaunay tesselation links function values to the nearest vertices; as noted already, it does not always imply that the closest points are used, but it does whenever such an arrangement constitutes a triangularization.

Definition 114 *The set* $\{V(p_i)\}_{i=1}^{n}$ *is called a* ***Voronoi tesselation.***

Voronoi tesselations can have irregularly-shaped elements, which is why we prefer to use the Delaunay tesselation.

Note that the Delaunay tesselation is not the triangularization that involves the shortest total length of edges (this triangularization, called the **minimum weight triangularization**, is NP-hard to compute). It turns out that the **minimum spanning tree**, which is the tree that connects all points and has the least total edge length, is a subset of the Delauanay tesselation but may not be a triangularization itself.

To construct and interpolate using a Delaunay tesselation, first construct the triangles (which needs only to be done once, provided the nodes stay fixed), locate the triangle for a given point, and then calculate the barycentric coordinates; note that some routines return the barycentric coordinates without normalizing them to sum to one, so you need to check them carefully (it is harmless to normalize them automatically). An efficient search mechanism through a tesselation follows a stochastic visibility walk path. The idea of a visibility walk is to start with a triangle t that does not currently contain the desired point $p = (x, y)$, then move to a triangle whose shared edge e has a supporting line that separates t from p; if this edge is not unique, then we pick randomly among all the qualifying edges. For Delaunay tesselations the stochastic visibility walk terminates in a finite number of steps, while for other triangularizations it can cycle, which is another reason why Delaunay tesselations are good. The worst case for length of the stochastic

visibility walk is $O(\sqrt{n})$, where n is the number of points, and we can improve performance by carefully selecting the initial triangle (for example, if we are evaluating a lot of points we may be choose the order smartly so that we need only make small numbers of moves, such as remembering the index of the previous point).

Note that our barycentric interpolant is "piecewise linear" and therefore has a derivative that is discontinuous on the boundaries of the triangles. It is possible to compute a C^1 interpolant using the Cendes-Wong algorithm, which places "control points" on and in the triangles to subdivide them into six subtriangles and then uses piecewise quadratic functions that fit together smoothly on the boundaries, provided we know the gradient (derivative vector) at the meshpoints. To begin constructing the surface, we take a Delaunay triangularization and subdivide each triangle and then fit the subtriangles as quadratic Bezier triangles to ensure derivative continuity. Take the points (b_2, b_4, b_6) to be the points on a given triangle, and b_0 is the inscribed center (the intersection of lines that bisect each angle). (b_1, b_3, b_5) are the intersections of each edge with a segment that connects b_0 to the center of each adjacent triangle (or the midpoint if the triangle

is on the boundary of the mesh). The subdivision points then satisfy

$$
\begin{aligned}
b_0 &= \delta_0 b_2 + \delta_1 b_4 + \delta_2 b_6 \\
b_1 &= \gamma_0 b_2 + \gamma_2 b_6 \\
b_3 &= \alpha_0 b_2 + \alpha_1 b_4 \\
b_5 &= \beta_1 b_4 + \beta_2 b_6 \\
b_7 &= \frac{b_1 + b_2}{2} \\
b_8 &= \frac{b_2 + b_3}{2} \\
b_9 &= \frac{b_3 + b_4}{2} \\
b_{10} &= \frac{b_4 + b_5}{2} \\
b_{11} &= \frac{b_5 + b_6}{2} \\
b_{12} &= \frac{b_1 + b_6}{2} \\
b_{13} &= \gamma_0 b_{14} + \gamma_2 b_{18} \\
b_{14} &= \frac{b_0 + b_2}{2} \\
b_{15} &= \alpha_0 b_{14} + \alpha_1 b_{16} \\
b_{16} &= \frac{b_0 + b_4}{2} \\
b_{17} &= \beta_1 b_{16} + \beta_2 b_{18} \\
b_{18} &= \frac{b_0 + b_6}{2}
\end{aligned}
$$

where

$$
\begin{aligned}
\delta_0 + \delta_1 + \delta_2 &= 1 \\
\alpha_0 + \alpha_1 &= 1 \\
\beta_1 + \beta_2 &= 1 \\
\gamma_0 + \gamma_2 &= 1
\end{aligned}
$$

are the barycentric coordinates for the inscribed center and the intersection of each edge and the segment that connects the inscribed center to the inscribed center of the adjacent triangle (or the

midpoint if the edge is a boundary), respectively. The inscribed center is given by the expression

$$b_0 = \frac{|b_4 - b_6|}{|b_4 - b_6| + |b_2 - b_6| + |b_2 - b_4|} b_2 + \frac{|b_2 - b_6|}{|b_4 - b_6| + |b_2 - b_6| + |b_2 - b_4|} b_4 + \frac{|b_2 - b_4|}{|b_4 - b_6| + |b_2 - b_6| + |b_2 - b_4|} b_6.$$

To find the intersection points, consider the inscribed center $\widetilde{b}_0$ of the triangle that shares edge (b_2, b_6); the intersection point satisfies the equations

$$(1 - s)\, b_0 + s\widetilde{b}_0 = (1 - t)\, b_2 + t b_6$$

which can be solved for the weights s and t, and then setting $\gamma_0 = 1 - t$ and $\gamma_2 = t$. Obviously if we have a boundary edge then the midpoint satisfies $\gamma_0 = \gamma_2 = \frac{1}{2}$. The same procedure finds the (α, β) coefficients.

Let the value of the function at the mesh points be $(\phi_i)_{i=0}^{18}$; we already know the values (ϕ_2, ϕ_4, ϕ_6) as well as the derivatives $(D\phi_2, D\phi_4, D\phi_6)$ (we can often obtain the derivatives from envelope conditions). Note that if the triangle index is i for $1 \leq i \leq 6$, then the control points are

$$\{0, 12 + i, 13 + i \bmod 6, i, 6 + i, 1 + i \bmod 6\}.$$

We now need to specify values at the other sixteen mesh points to guarantee continuity of the

derivative. These values must satisfy

$$\begin{aligned}
\phi_7 &= \phi_2 + D\phi_2\,(b_7 - b_2) \\
\phi_8 &= \phi_2 + D\phi_2\,(b_8 - b_2) \\
\phi_{14} &= \phi_2 + D\phi_2\,(b_{14} - b_2) \\
\phi_9 &= \phi_4 + D\phi_4\,(b_9 - b_4) \\
\phi_{10} &= \phi_4 + D\phi_4\,(b_{10} - b_4) \\
\phi_{16} &= \phi_4 + D\phi_4\,(b_{16} - b_4) \\
\phi_{11} &= \phi_6 + D\phi_6\,(b_{11} - b_6) \\
\phi_{12} &= \phi_6 + D\phi_6\,(b_{12} - b_6) \\
\phi_{18} &= \phi_6 + D\phi_6\,(b_{18} - b_6) \\
\phi_3 &= \alpha_0\phi_8 + \alpha_1\phi_9 \\
\phi_5 &= \beta_1\phi_{10} + \beta_2\phi_{11} \\
\phi_1 &= \gamma_0\phi_7 + \gamma_2\phi_{12} \\
\phi_{15} &= \alpha_0\phi_{14} + \alpha_1\phi_{16} \\
\phi_{17} &= \beta_1\phi_{16} + \beta_2\phi_{18} \\
\phi_{13} &= \gamma_0\phi_{14} + \gamma_2\phi_{18} \\
\phi_0 &= \delta_0\phi_{14} + \delta_1\phi_{16} + \delta_2\phi_{18}.
\end{aligned}$$

Evaluation of the function first locates the Delaunay triangle that contains the point, then computes the function using the unique quadratic defined by the six points, which satisfies the six linear equations

$$\phi_i = a_0 + a_1 x_i + a_2 y_i + a_3 x_i^2 + a_4 x_i y_i + a_5 y_i^2$$

for $b_i = (x_i, y_i)$ and $i \in \{1, ..., 6\}$.

Higher order continuity is feasible only if we have derivative information at non-node points, which we may have to locally estimate. In this case we should use Renka's method instead.

10.9 Renka's Method

Renka's method can construct either C^1 or C^2 interpolants over two-dimensional spaces using a Delaunay tesselation. The C^1 version uses quintic polynomials and the C^2 version uses nonic polynomials. Derivatives at the nodes are estimated locally – the first derivative for the C^1 interpolant and the first and second derivatives for the C^2 version – by fitting a quadratic at a given point using weighted least squares, where the weights are given by

$$\overline{w}_i = \frac{w_i}{\sum_{j=1}^{N} w_j}$$
$$w_i = \frac{\max\{R_k - d_i, 0\}}{R_k d_i}$$

where R_k is a radius about node k and d_i is the distance between nodes i and k; experimentation yielded that the best method had the closest nine nodes used in the least squares fitting exercise (note the similarity with the modified Shepard's method).

10.10 Natural Neighbor Splines

We can also use the Delaunay tesselation to construct a smoother interpolant using a procedure called 'natural neighbor interpolation'. First, we construct the circumcircles for each triangle; these circles are defined by the centroids

$$c_i = \left(-\frac{b_x}{2a}, -\frac{b_y}{2a}\right)$$

and radii

$$r_i = \frac{1}{|2a|}\sqrt{b_x^2 + b_y^2 - 4ac}$$

where

$$a = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}$$

$$b_x = -\begin{vmatrix} x_1^2 + y_1^2 & y_1 & 1 \\ x_2^2 + y_2^2 & y_2 & 1 \\ x_3^2 + y_3^2 & y_3 & 1 \end{vmatrix}$$

$$b_y = \begin{vmatrix} x_1^2 + y_1^2 & x_1 & 1 \\ x_2^2 + y_2^2 & x_2 & 1 \\ x_3^2 + y_3^2 & x_3 & 1 \end{vmatrix}$$

$$c = \begin{vmatrix} x_1^2 + y_1^2 & x_1 & y_1 \\ x_2^2 + y_2^2 & x_2 & y_2 \\ x_3^2 + y_3^2 & x_3 & y_3 \end{vmatrix}.$$

We then interpolate as

$$f(x,y) = \sum_{i=1}^{3} \lambda_i(x,y) f(x_i, y_i)$$

where

$$\lambda_i(x,y) = \frac{1}{\sum_{j=1}^{k} \psi_j(x,y)} \sum_{j=1}^{k} \psi_j(x,y) b_i(x,y)$$

for the usual barycentric coordinates $b_i(x,y)$ and blending function

$$\psi_j(x,y) = \begin{cases} \left(\|x - c_j\|^2 - r_j^2\right)^4 & \text{if } \|x - c_j\| < r_i \\ 0 & \text{otherwise} \end{cases}.$$

k equals the number of triangles whose circumcircles contain the point (x,y); everywhere except at the vertices the resulting interpolant is C^k. This smoothness is much better than barycentric interpolation, which is linear with discontinuities in the derivatives along each edge (not just at the nodes).

10.10.1 Adaptive Grids

Note that all tesselation methods suffer from the curse of dimensionality – creating the tesselation is burdensome in high dimensions, and so is searching for the target simplex. But since they

use a scattered grid, we can mitigate this problem by adapting the grid to our problem (Brumm and Grill 2019). Start with a relatively coarse grid G. After approximating the function, take a uniform sample from the state space and compute the approximation errors. Where the errors are too large, add that point from the sample to the grid and recompute the approximation.

10.11 Kriging

Kriging is a method of interpolation built on the idea that we can view the observed values of a function at the nodes as the realizations of a Gaussian process, and we can "predict" the missing values using forecasting techniques.

Definition 115 *A stochastic process* $\{X_t : t \in T\}$ *is a* ***Gaussian process*** *iff* $\forall (t_1, ..., t_k) \in T$ *and* $\forall k < \infty$, $(X_{t_1}, ..., X_{t_k})$ *is jointly a multivariate normal random vector. Alternatively, every linear combination of* $(X_{t_1}, ..., X_{t_k})$ *is a univariate normal random variable.*

A Gaussian process is the extension of a multivariate normal to infinitely-many random variables. Note that the variance of a Gaussian process is finite. If the process is stationary, then we can represent it as

$$X_t = \cos(at)\,\xi_1 + \sin(at)\,\xi_2$$

where (ξ_1, ξ_2) are independent standard normal random variables and a is a constant. If we assume that $E[X_t] = 0$, then the process is determined entirely by the covariance functions. Some common covariance functions are the constant

$$K_C(x, x') = C,$$

the linear

$$K_L(x, x') = x^T x',$$

the white Gaussian noise

$$K_{GN}(x, x') = \sigma^2 \delta_{x,x'},$$

the squared exponential

$$K_{SE}(x, x') = \exp\left(-\frac{|x - x'|^2}{2l^2}\right),$$

the Ornstein-Uhlbeck

$$K_{OU}\left(x,x'\right)=\exp\left(-\frac{\left|x-x'\right|}{l}\right),$$

the Matérn

$$K_{M}\left(x,x'\right)=\frac{2^{1-\nu}}{\Gamma\left(\nu\right)}\left(\frac{\sqrt{2\nu}\left|x-x'\right|}{l}\right)^{\nu}K_{\nu}\left(\frac{\sqrt{2\nu}\left|x-x'\right|}{l}\right),$$

the periodic

$$K_{P}\left(x,x'\right)=\exp\left(-\frac{2\sin^{2}\left(\frac{d}{2}\right)}{l^{2}}\right),$$

and the rational quadratic

$$K_{RQ}\left(x,x'\right)=\left(1+\frac{\left|x-x'\right|^{2}l^{2}}{\alpha}\right)^{-\alpha},\alpha\geq0.$$

l is called the characteristic length-scale, which governs how close two variables have to be (in time) to influence each other, $\delta_{x,x'}$ is the Kronecker-δ function

$$\delta_{i,j}=\begin{cases}1 & \text{if } i=j\\ 0 & \text{if } i\neq j\end{cases},$$

and K_{ν} is the modified Bessel function of order ν which satisfies

$$I_{\nu}\left(z\right)=\sum_{m=0}^{\infty}\frac{1}{m!\Gamma\left(m+\nu+1\right)}\left(\frac{z}{2}\right)^{2m+\nu}$$
$$K_{\nu}\left(z\right)=\frac{\pi}{2}\frac{I_{-\nu}\left(z\right)-I_{\nu}\left(z\right)}{\sin\left(\nu\pi\right)}.$$

Note that any finite sum of covariance functions is also a valid covariance function, so we can form weighted sums (the sum of products of covariance functions with the constant function) to generate rather complicated covariance structures. We can also form products in higher dimensions:

$$\widetilde{K}\left(x,y,x',y'\right)=K\left(x,x'\right)K\left(y,y'\right)$$

is a proper covariance function in two dimensions. If we have symmetry, then we can form our covariance function as

$$\widehat{K}\left(x,x'\right)=K\left(x,x'\right)+K\left(-x,x'\right).$$

If we want independence from the order of arguments, then we can use

$$\widehat{K}\left(x,y,x',y'\right)=\widetilde{K}\left(x,y,x',y'\right)+\widetilde{K}\left(y,x,x',y'\right).$$

For high-dimensional problems, we can also do a dimension-reduction using a low-rank matrix A:

$$\widetilde{K}\left(x,x'\right)=K\left(Ax,Ax'\right).$$

In general people use either K_{SE} or K_{RQ}, so those are a good place to start.

Kriging predicts the function values using Bayesian regression, which amounts to computing the function values as weighted averages of the function values at the nodes in a neighborhood of the given point x. The weights satisfy

$$\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} K\left(x_1,x_1\right) & \cdots & K\left(x_1,x_n\right) \\ \vdots & \ddots & \vdots \\ K\left(x_n,x_1\right) & \cdots & K\left(x_n,x_n\right) \end{bmatrix}^{-1} \begin{bmatrix} K\left(x_1,x\right) \\ \vdots \\ K\left(x_n,x\right) \end{bmatrix},$$

so that the function value at x is given by

$$\widehat{f}\left(x\right)=\begin{bmatrix} f\left(x_1\right) & \cdots & f\left(x_n\right) \end{bmatrix} \begin{bmatrix} K\left(x_1,x_1\right) & \cdots & K\left(x_1,x_n\right) \\ \vdots & \ddots & \vdots \\ K\left(x_n,x_1\right) & \cdots & K\left(x_n,x_n\right) \end{bmatrix}^{-1} \begin{bmatrix} K\left(x_1,x\right) \\ \vdots \\ K\left(x_n,x\right) \end{bmatrix}.$$

Note that the Kriging "estimate" $\widehat{f}\left(x\right)$ is unbiased

$$E\left[\widehat{f}\left(x_i\right)\right]=E\left[f\left(x_i\right)\right],$$

interpolates the data

$$\widehat{f}\left(x_i\right)=f\left(x_i\right),$$

and is the best linear unbiased estimator with error

$$e\left(x\right)=K\left(x,x\right)-\begin{bmatrix} K\left(x_1,x\right) & \cdots & K\left(x_n,x\right) \end{bmatrix} \begin{bmatrix} K\left(x_1,x_1\right) & \cdots & K\left(x_1,x_n\right) \\ \vdots & \ddots & \vdots \\ K\left(x_n,x_1\right) & \cdots & K\left(x_n,x_n\right) \end{bmatrix}^{-1} \begin{bmatrix} K\left(x_1,x\right) \\ \vdots \\ K\left(x_n,x\right) \end{bmatrix},$$

which is just

$$Var\left(\widehat{f}\left(x\right)-f\left(x\right)\right)=Var\left(f\left(x\right)\right)-Var\left(\widehat{f}\left(x\right)\right).$$

Since Kriging interpolates the data no matter the values for the hyperparameters, we can choose them to improve the fit at selected points that are not nodes (or on average). Note also that the "variance" matrix need only be computed once; it does not depend on the value of x.

10.12 Wavelets

Wavelets approximate functions using a different basis. First, consider the **Fourier transform** of a function, which decomposes it into pieces that are localized by frequency. That is, a function can be written as

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx)$$

with coefficients

$$\begin{aligned} a_0 &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\,dx \\ a_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)\,dx \\ b_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)\,dx, \end{aligned}$$

which decomposes the function into a sum of sines and cosines (which are an orthogonal basis for continuous functions). Fourier series are useful for approximating functions that are periodic. Note that the Fourier series approximation suffers from a version of Runge's phenomenon, called the Gibbs phenomenon, in which the error can explode at the endpoints.

Wavelets decompose a stochastic process (data sampled continuously) into pieces that are localized in time *and* frequency; Fourier transforms localize the process only in frequency and therefore struggle to approximate functions with sharp spikes. A wavelet takes the process f and decomposes it as

$$\begin{aligned} f &= c_0 + \sum_{j\geq 0}^{\infty}\sum_{k=1}^{2^j} \langle f, \psi_{j,k}\rangle \psi_{j,k} \\ c_0 &= \int_0^1 f(t)\,dt \\ \theta_{j,k} &= \langle f, \psi_{j,k}\rangle = \int_0^1 f(t)\,\psi_{j,k}(t)\,dt. \end{aligned}$$

$\psi_{j,k}(t)$ is called the wavelet at scale 2^{-j} and position $k2^{-j}$.

One common wavelet is called the Haar wavelet:

$$\psi_{j,k}(t) = 2^{j/2}\left(\mathbf{1}\left\{t \in \left[2^{-j}(k-1), 2^{-j}\left(k-\frac{1}{2}\right)\right]\right\} - \mathbf{1}\left\{t \in \left[2^{-j}\left(k-\frac{1}{2}\right), 2^{-j}k\right]\right\}\right)$$

which satisfies

$$\int_0^1 \psi_{j,k}(t)\,dt = 0$$

$$\int_0^1 \psi_{j,k}^2(t) = 1$$

$$\int_0^1 \psi_{j,k}(t)\,\psi_{l,m}(t)\,dt = 0 \text{ unless } j = l \text{ and } k = m.$$

If f is constant on $\left[2^{-j}(k-1), 2^{-j}k\right]$ then

$$\int_0^1 f\psi_{j,k}(t) = 0.$$

Suppose f is piecewise constant with at most m discontinuities. Let

$$f_J = c_0 + \sum_{j=0}^{J-1}\sum_{k=1}^{2^j}\theta_{j,k}\psi_{j,k};$$

then f_J has at most mJ nonzero wavelet coefficients, so that f_J is piecewise constant with discontinuities occuring only at the endpoints of the intervals $\left[2^{-J}(k-1), 2^{-J}k\right]$. Thus,

$$||f - f_J||_{L_2}^2 = O\left(2^{-J}\right).$$

Daubechies-N wavelet

$$\int_0^1 t^l\psi_{j,k}(t) = 0 \text{ for } l \in \{0, 1, ..., N-1\}.$$

Haar wavelets have 1 vanishing moment, Daubechies-N wavelets have N, so Haar = Daubechies-1. Thus, if f is a piecewise polynomial of degree N, then

$$||f - f_J||_{L_2}^2 = O\left(2^{-J}\right).$$

That is, any continuous function on a compact set can be approximated uniformly by linear combinations of the ψ terms.

For discrete time processes, suppose

$$f = \left\{f\left(\frac{1}{n}\right), ..., f\left(\frac{n}{n}\right)\right\}.$$

Then

$$f = c_0 + \sum_{j=0}^{\log_2(n-1)}\sum_{k=1}^{2^j}\langle f, \psi_{j,k}\rangle\,\psi_{j,k}$$

satisfies

$$\sum_{i=1}^{n} i^l \psi_{j,k}(i) = 0 \text{ for } l \in \{0, 1, ..., N-1\};$$

that is, f_J has at most mJ nonzero coefficients.

10.13 Neural Nets

The next method of approximation we will examine are **neural nets**; these are flexible nonlinear approximators that combine linear combinations with nonlinear transformations to form an approximating function that can fit a wide range of functions. Here, we start with a set of nodes $\{x_i\}$ and the function values $\{f(x_i)\}$. We then fit the following nonlinear least squares problem to this data:

$$\min_{\beta} \left\{ \sum\nolimits_{i=1}^{m} \left(f(x_i) - F(x_i; \beta) \right)^2 \right\}$$

for a **single-layer network** and

$$\min_{\beta,\gamma} \left\{ \sum\nolimits_{i=1}^{m} \left(f(x_i) - F(x_i; \beta, \gamma) \right)^2 \right\}$$

for a **single hidden-layer network**. The function $F(\cdot)$ takes the form

$$F(x; \beta) = h\left(\sum\nolimits_{i=1}^{m} \beta_i g(x_i) \right)$$

or

$$F(x; \beta, \gamma) = l\left(\sum\nolimits_{j=1}^{n} \gamma_i h\left(\sum\nolimits_{i=1}^{m} \beta_i^j g(x_i) \right) \right).$$

The particular functions $l(\cdot)$, $g(\cdot)$, and $h(\cdot)$ that we choose determine the accuracy of our approximation. If we let $g(\cdot)$ and $l(\cdot)$ be linear and $h(\cdot)$ be a "squashing function" (essentially a cdf)

$$h : \mathcal{R} \to [0, 1], \; h \text{ nondecreasing}, \; \lim_{x \to \infty} \{h(x)\} = 1, \text{ and } \lim_{x \to 0} \{h(x)\} = 0,$$

there exists a representation theorem showing there is a member of this class which satisfies

$$\sup_{x \in K} \{|f(x) - F(x)|\} \leq \epsilon;$$

that is, these functions are dense in the space of continuous functions. Furthermore, we know that, for a single-hidden layer neural network with n hidden nodes, convergence is at rate $O\left(n^{-1}\right)$

compared to $O\left(n^{-\frac{2}{m}}\right)$ for series approximators of dimension m; that is, we can do very well with fairly-simple networks.

Let's take a specific example, the softlog function

$$h(x) = \log(1 + \exp(x))$$

and the neural net takes the form

$$F(x; \beta, \gamma) = \gamma_0 + \sum_{j=1}^{n} \gamma_j \log\left(1 + \exp\left(\beta_{0,j} + \sum_{k=1}^{m} \beta_{k,j} x_k\right)\right).$$

Another option is the logsigmoid function

$$\begin{aligned} h(x) &= \frac{1}{1 + \exp(-x)} \\ Dh(x) &= \frac{\exp(-x)}{1 + \exp(-x)} \end{aligned}$$

so that the neural net is

$$F(x; \beta, \gamma) = \gamma_0 + \sum_{j=1}^{n} \left(\frac{\gamma_j}{1 + \exp\left(-\beta_{0,j} - \sum_{k=1}^{m} \beta_{k,j} x_k\right)}\right).$$

For smooth functions, the hyperbolic tangent function works quite well:

$$\begin{aligned} h(x) &= \tanh(x) \\ &= \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \\ Dh(x) &= \frac{4\exp(-2x)}{(1 + \exp(-2x))^2}, \end{aligned}$$

which leads to the neural net

$$F(x; \beta, \gamma) = \gamma_0 + \sum_{j=1}^{n} \left(\gamma_j \tanh\left(-\beta_{0,j} - \sum_{k=1}^{m} \beta_{k,j} x_k\right)\right).$$

A nondifferentiable version is the rectified linear function

$$\begin{aligned} h(x) &= \max\{0, x\} \\ Dh(x) &= \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0 \end{cases}, \end{aligned}$$

which produces the neural net

$$F(x;\beta,\gamma)=\gamma_0+\sum_{j=1}^{n}\left(\gamma_j\max\left\{0,\beta_{0,j}+\sum_{k=1}^{m}\beta_{k,j}x_k\right\}\right).$$

Depending on the problem, any of these can be useful.

The nonlinear least-squares problem used to fit the neural net parameters can be challenging, however, so neural nets are usually "fit" using a machine learning mechanism. The essence of the learning mechanism is to run inputs (values for x) through the network for a given set of weights (β,γ) and update them according to the difference between the expected output y and the output of the network $F(x;\beta,\gamma)$. A technique called **backpropagation** can make this updating process relatively easy. Here I am going to use more generic notation. Define the input vector as

$$a^0$$

The first layer has the linear representation

$$n_i^1=\sum_{j=1}^{N}w_{ij}^1a_j^0+b_i^1.$$

We then use the squashing transfer function

$$a_i^1=f^1\left(n_i^1\right);$$

think of a_i^1 as the output of the first layer.

We then continue to next layer

$$n_i^2=\sum_{j=1}^{N}w_{ij}^2a_j^1+b_j^2$$
$$a_i^2=f^2\left(n_i^2\right)$$

and repeat until we reach a^M, the output of the entire network (that is, the function we are trying to use as an approximator).

Remember the goal is to minimize the least squares criterion

$$F=\min_{a^0}\left\{\left(a^M-t\right)^2\right\}$$

given the points $t=g(x)$. One simple method to solve this problem is to use the steepest gradient descent algorithm; recall from earlier in the notes that the steepest descent algorithm follows the

gradient downhill, since locally the gradient delivers the fastest decline in the objective. We adjust the parameters of the network following the recursion

$$w_{ij}^m = w_{ij}^m - \alpha \frac{\partial F}{\partial w_{ij}^m}$$
$$b_i^m = b_i^m - \alpha \frac{\partial F}{\partial b_i^m},$$

where α is a hyperparameter that controls how aggressive our steps are (because the steps are not infinitessimal in size, they may not go downhill). Define

$$s_i^m = \frac{\partial F}{\partial n_i^m}$$

so that

$$\frac{\partial F}{\partial w_{ij}^m} = s_i^m a_j^{m-1}$$
$$\frac{\partial F}{\partial b_i^m} = s_i^m.$$

Then we need only figure out how to compute s_i^m for each m and each i. Back propagation will give us a simple rule for calculating these values iteratively:

$$s_i^M = -2\left(t_i - a_i^M\right) \frac{\partial f^M\left(n_i^M\right)}{\partial n_i^M}$$
$$s_i^m = \frac{\partial f^m\left(n_i^m\right)}{\partial n_i^m} \sum_{j=1}^{N^{m-1}} s_j^{m+1} w_{ji}^{m+1}.$$

To get the gradient with respect to the coefficients, we average these values over the training sample, either using the entire sample or using a "batch" (a subsample) if the number of points is prohibitively large.

Here is a simple two-dimensional example with one layer and a softplus transfer function:

$$\begin{aligned}
a^0 &= \left(a_1^0, a_2^0\right) = (x_1, x_2) \\
n_1^1 &= w_{11}^1 a_1^0 + w_{12}^1 a_2^0 + b_1^1 \\
a_1^1 &= f^1\left(n_1^1\right) \\
&= \log\left(1 + \exp\left(n_1^1\right)\right) \\
s_1^1 &= -2\left(g\left(x_1, x_2\right) - a_1^1\right) \frac{\exp\left(n_1^1\right)}{1 + \exp\left(n_1^1\right)} \\
w_{11}^1 &= w_{11}^1 - \alpha s_1^1 a_1^0 \\
w_{12}^1 &= w_{12}^1 - \alpha s_1^1 a_2^0 \\
b_1^1 &= b_1^1 - \alpha s_1^1.
\end{aligned}$$

We first run every point (x_1, x_2) through this process (the order is not important); one time through the data is called an **epoch**. The gradient term is computed using an average of the derivatives at each of the sample points. Then we repeat until the coefficients stop changing.

In contrast, batch gradient descent uses the entire training sample to construct the gradient before updating, which can be very slow and even infeasible if the data set is too large. Mini-batch gradient descent "mixes" the two; instead of using every point, it uses batches of n points, where n is much smaller than the data set.

The current state of the art learning algorithm is called ADAM (Adaptive Moment Estimation), which is similar to stochastic gradient descent but uses the "momentum vector" to improve speed:

$$\begin{aligned}
m^{n+1} &\leftarrow \beta_1 m^n - \left(1 - \beta_1\right) \nabla_\theta J^n\left(\theta\right) \\
s^{n+1} &\leftarrow \beta_2 s^n + \left(1 - \beta_2\right) \nabla_\theta J^n\left(\theta\right) \otimes \nabla_\theta J^n\left(\theta\right) \\
\widehat{m}^{n+1} &\leftarrow \frac{m^{n+1}}{1 - \beta_1^{n+1}} \\
\widehat{s}^{n+1} &\leftarrow \frac{s^{n+1}}{1 - \beta_2^{n+1}} \\
\theta^{n+1} &\leftarrow \theta^n - \alpha \widehat{m}^{n+1} \oslash \sqrt{\widehat{s}^{n+1} + \epsilon}
\end{aligned}$$

where common choices for the hyperparameters are $\beta_1 = 0.9$ as the momentum decay hyperparameter, $\beta_2 = 0.999$ as the scaling decay hyperparameter, and $\epsilon = 10^{-8}$; $\otimes$ is component-wise

multiplication (.∗ in Matlab), and ⊘ is component-wise division (./ in Matlab). Both m and s are initialized to zero, and α is problem-specific. By not allowing the algorithm to take the steepest descent direction, ADAM prevents deviations into "side valleys" that slow down convergence to the global minimum of the error function. The backpropagation method still is used to compute the gradient terms $\nabla_\theta J^n(\theta)$.

Maliar, Maliar, and Winant (2018) show how to use neural nets to "learn" the solution to an economic model. For example, suppose we want to solve the life cycle consumption model

$$E_{\{e_0,\ldots,e_T\}}\left[\sum_{t=0}^{T}\beta^t u(c_t)\right]$$

$$\begin{aligned} c_t &= Ra_t + \exp(z_t) - a_{t+1} \\ a_0 &= 0 \\ a_{t+1} &\geq 0 \\ z_t &= \rho z_{t-1} + \sigma e_t \\ z_{-1} &= 0. \end{aligned}$$

Suppose we approximate the decision rules $a_{t+1} = f_t(a_t, z)$ with a neural network; given random samples from the stochastic variables, we will learn the optimum using penalty functions to enforce the constraints.

For illustration, let $T = 2$ so that

$$x^0 = (e_0, e_1, e_2)$$

$$f(\theta) = -\frac{1}{N}\sum_{n=1}^{N}\left[\begin{array}{c} \log(\exp(e_{0,n}) - a_{1,n}) + \beta\log(Ra_{1,n} + \exp(\rho e_{0,n} + e_{1,n}) - a_{2,n}) + \\ \beta^2\log\left(Ra_{2,n} + \exp\left(\rho^2 e_{0,n} + \rho e_{1,n} + e_{2,n}\right)\right) - \frac{\gamma_1}{3}\min\{0, a_{1,n}\}^3 - \frac{\gamma_2}{3}\min\{0, a_{2,n}\}^3 \end{array}\right].$$

The neural net is given by

$$\begin{aligned} a_1 &= \varphi_0(e_0) \\ &= \theta_{021} + \theta_{121}\log(1 + \exp(\theta_{011} + \theta_{111}e_0)) \\ a_2 &= \varphi_1(a_1, \rho e_0 + e_1) \\ &= \theta_{022} + \theta_{122}\log(1 + \exp(\theta_{012} + \theta_{112}a_1 + \theta_{212}(\rho e_0 + e_1))); \end{aligned}$$

that is, it has one hidden layer and a softlog activation function. As above, we evaluate this function using its component pieces:

$$\begin{aligned}
l_{1,1} &= \theta_{011} + \theta_{111} e_0 \\
l_{1,2} &= \theta_{012} + \theta_{112} a_1 + \theta_{212} \left(\rho e_0 + e_1\right) \\
m_{1,1} &= \log\left(1 + \exp\left(l_{1,1}\right)\right) \\
m_{1,2} &= \log\left(1 + \exp\left(l_{1,2}\right)\right) \\
l_{2,1} &= \theta_{021} + \theta_{121} m_{1,1} \\
l_{2,2} &= \theta_{022} + \theta_{122} m_{1,2} \\
a_1 &= l_{2,1} \\
a_2 &= l_{2,2}.
\end{aligned}$$

The coefficient vector is

$$\theta = \left\{\theta_{011}, \theta_{111}, \theta_{012}, \theta_{112}, \theta_{212}, \theta_{021}, \theta_{121}, \theta_{022}, \theta_{122}\right\}.$$

Using gradient descent we can update the guess by

$$\theta^{k+1} = \theta^k - \alpha \frac{\partial f\left(\theta^k\right)}{\partial \theta}$$

where

$$\frac{\partial f}{\partial \theta_{011}} = -\frac{1}{N}\left(\begin{array}{r} -\frac{1}{\exp(e_{0,n})-a_{1,n}} + \frac{\beta R}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} - \\ \frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} \frac{\partial a_{2,n}}{\partial a_{1,n}} - \gamma_1 \min\{0, a_{1,n}\}^2 - \gamma_2 \min\{0, a_{2,n}\}^2 \frac{\partial a_{2,n}}{\partial a_{1,n}} \end{array} \right) \frac{\partial a_{1,n}}{\partial \theta_{011}}$$

$$\frac{\partial f}{\partial \theta_{111}} = -\frac{1}{N}\left(\begin{array}{r} -\frac{1}{\exp(e_{0,n})-a_{1,n}} + \frac{\beta R}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} - \\ \frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} \frac{\partial a_{2,n}}{\partial a_{1,n}} - \gamma_1 \min\{0, a_{1,n}\}^2 - \gamma_2 \min\{0, a_{2,n}\}^2 \frac{\partial a_{2,n}}{\partial a_{1,n}} \end{array} \right) \frac{\partial a_{1,n}}{\partial \theta_{111}}$$

$$\frac{\partial f}{\partial \theta_{021}} = -\frac{1}{N}\left(\begin{array}{r} -\frac{1}{\exp(e_{0,n})-a_{1,n}} + \frac{\beta R}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} - \\ \frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} \frac{\partial a_{2,n}}{\partial a_{1,n}} - \gamma_1 \min\{0, a_{1,n}\}^2 - \gamma_2 \min\{0, a_{2,n}\}^2 \frac{\partial a_{2,n}}{\partial a_{1,n}} \end{array} \right) \frac{\partial a_{1,n}}{\partial \theta_{021}}$$

$$\frac{\partial f}{\partial \theta_{121}} = -\frac{1}{N}\left(\begin{array}{r} -\frac{1}{\exp(e_{0,n})-a_{1,n}} + \frac{\beta R}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} - \\ \frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} \frac{\partial a_{2,n}}{\partial a_{1,n}} - \gamma_1 \min\{0, a_{1,n}\}^2 - \gamma_2 \min\{0, a_{2,n}\}^2 \frac{\partial a_{2,n}}{\partial a_{1,n}} \end{array} \right) \frac{\partial a_{1,n}}{\partial \theta_{121}}$$

$$\frac{\partial f}{\partial \theta_{012}} = -\frac{1}{N}\left(-\frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} + \frac{\beta^2 R}{Ra_{2,n}+\exp(\rho^2 e_{0,n}+\rho e_{1,n}+e_{2,n})} - \gamma_2 \min\{0, a_{2,n}\}^2 \right) \frac{\partial a_{2,n}}{\partial \theta_{01}}$$

$$\frac{\partial f}{\partial \theta_{112}} = -\frac{1}{N}\left(-\frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} + \frac{\beta^2 R}{Ra_{2,n}+\exp(\rho^2 e_{0,n}+\rho e_{1,n}+e_{2,n})} - \gamma_2 \min\{0, a_{2,n}\}^2 \right) \frac{\partial a_{2,n}}{\partial \theta_{11}}$$

$$\frac{\partial f}{\partial \theta_{212}} = -\frac{1}{N}\left(-\frac{\beta}{Ra_{1,n}+\exp(\rho e_{0,n}+e_{1,n})-a_{2,n}} + \frac{\beta^2 R}{Ra_{2,n}+\exp(\rho^2 e_{0,n}+\rho e_{1,n}+e_{2,n})} - \gamma_2 \min\{0, a_{2,n}\}^2 \right) \frac{\partial a_{2,n}}{\partial \theta_{21}}$$

and

$$\begin{aligned} \frac{\partial a_{2,n}}{\partial a_{1,n}} &= \frac{\partial a_2}{\partial l_{2,2}} \frac{\partial l_{2,2}}{\partial m_{1,2}} \frac{\partial m_{1,2}}{\partial l_{1,2}} \frac{\partial l_{1,2}}{\partial a_1} \\ &= \theta_{122} \frac{\exp(l_{1,2})}{1+\exp(l_{1,2})} \theta_{112} \end{aligned}$$

$$\begin{aligned} \frac{\partial a_{1,n}}{\partial \theta_{011}} &= \frac{\partial a_1}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial m_{1,1}} \frac{\partial m_{1,1}}{\partial l_{1,1}} \frac{\partial l_{1,1}}{\partial \theta_{011}} \\ &= \theta_{121} \frac{\exp(l_{1,1})}{1+\exp(l_{1,1})} \\ \frac{\partial a_{1,n}}{\partial \theta_{111}} &= \frac{\partial a_1}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial m_{1,1}} \frac{\partial m_{1,1}}{\partial l_{1,1}} \frac{\partial l_{1,1}}{\partial \theta_{011}} \\ &= \theta_{121} \frac{\exp(l_{1,1})}{1+\exp(l_{1,1})} e_0 \\ \frac{\partial a_{1,n}}{\partial \theta_{021}} &= \frac{\partial a_1}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial \theta_{021}} \\ &= 1 \\ \frac{\partial a_{1,n}}{\partial \theta_{121}} &= \frac{\partial a_1}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial \theta_{121}} \\ &= m_{1,1} \end{aligned}$$

$$
\begin{aligned}
\frac{\partial a_{2,n}}{\partial \theta_{012}} &= \frac{\partial a_2}{\partial l_{2,2}} \frac{\partial l_{2,2}}{\partial m_{2,2}} \frac{\partial m_{2,2}}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial \theta_{012}} \\
&= \theta_{122} \frac{\exp\left(l_{2,1}\right)}{1+\exp\left(l_{2,1}\right)} \\
\frac{\partial a_{2,n}}{\partial \theta_{112}} &= \frac{\partial a_2}{\partial l_{2,2}} \frac{\partial l_{2,2}}{\partial m_{2,2}} \frac{\partial m_{2,2}}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial \theta_{112}} \\
&= \theta_{122} \frac{\exp\left(l_{2,1}\right)}{1+\exp\left(l_{2,1}\right)} a_1 \\
\frac{\partial a_{2,n}}{\partial \theta_{212}} &= \frac{\partial a_2}{\partial l_{2,2}} \frac{\partial l_{2,2}}{\partial m_{2,2}} \frac{\partial m_{2,2}}{\partial l_{2,1}} \frac{\partial l_{2,1}}{\partial \theta_{212}} \\
&= \theta_{122} \frac{\exp\left(l_{2,1}\right)}{1+\exp\left(l_{2,1}\right)} \left(\rho e_0 + e_1\right) \\
\frac{\partial a_{2,n}}{\partial \theta_{022}} &= \frac{\partial a_2}{\partial l_{2,2}} \frac{\partial l_{2,2}}{\partial \theta_{022}} \\
&= 1 \\
\frac{\partial a_{2,n}}{\partial \theta_{122}} &= \frac{\partial a_2}{\partial l_{2,2}} \frac{\partial l_{2,2}}{\partial \theta_{122}} \\
&= m_{2,2}.
\end{aligned}
$$

Things are a bit more complicated if we aim to solve a system of "Euler equations" that describe an economy's evolution over time. Take the stochastic growth model, which has the Euler equation

$$\left(\exp\left(z\right)k^{\alpha}+\left(1-\delta\right)k-k'\right)^{-\gamma}=\beta E_{e'}\left[\left(\exp\left(\rho z+e'\right)\left(k'\right)^{\alpha}+\left(1-\delta\right)k'-k''\right)^{-\gamma}\left(\alpha\exp\left(\rho z+e'\right)\left(k'\right)^{\alpha-1}+1-\delta\right.\right.$$

The loss function is the squared error of this equation, evaluated at a set of points chosen via simulation.

1. Guess parameters $\theta = (a, b, c)$ of neural net

$$k'\left(k,z\right)=a_0+a_1\max\left\{b_1+b_2k+b_3z,0\right\}+a_2\max\left\{c_1+c_2k+c_3z,0\right\};$$

2. Simulate $\{k_t, z_t\}_{t=1}^{T}$ using these parameters and random draws for z;

3. Compute the Euler equation errors at each point (k_t, z_t):

$$e_t = 1-\frac{\left(\beta\sum\omega_i\left(\exp\left(\rho z_t+\sqrt{2}\sigma x_i\right)k'\left(k_t,z_t\right)^{\alpha}+\left(1-\delta\right)k'\left(k_t,z_t\right)-k'\left(k'\left(k_t,z_t\right),\rho z_t+\sqrt{2}\sigma x_i\right)\right)^{-\gamma}\left(\alpha\exp\right.\right.}{\exp\left(z_t\right)k_t^{\alpha}+\left(1-\delta\right)k_t-k'\left(k_t,z_t\right)}$$

4. Update the neural net parameters to reduce

$$l = \sum_{t=1}^{T} e_t^2$$

as

$$\theta^{new} = \theta^{old} - a\nabla l\left(\theta^{old}\right)$$

for learning parameter $a > 0$. Evaluating the gradient is the difficult part here, due to the composition term.

10.14 Radial Basis Functions

Another flexible nonlinear approximator class are called radial basis functions (in econometrics these are often called kernel smoothers). The idea is to approximate the unknown function using basis functions that satisfy

$$\phi(r) = \phi(\|r\|);$$

that is, only the distance from the point to zero matters. Some simple radial basis functions are the Gaussian kernel

$$\phi(r) = \exp\left(-\epsilon^2 r^2\right),$$

the inverse multiquadric

$$\phi(r) = \sqrt{\frac{1}{r^2 + \epsilon^2}},$$

and the thin plate spline

$$\phi(r) = r^2 \log\left(\frac{r}{\epsilon}\right);$$

$\epsilon > 0$ is a scale parameter that must be adjusted on a case-by-case basis; the value of ϵ does affect the behavior of the interpolating functions. The function f is then approximated as

$$f(x) = \sum_{i=1}^{n} w_i \phi(\|x - x_i\|)$$

for some weights w_i and nodes x_i, where the weights are chosen via linear least squares and the nodes are given. It is standard practice to add a polynomial to the interpolating function to improve stability of the interpolant, in which case we get

$$f(x) = \sum_{i=1}^{n} w_i \phi(\|x - x_i\|) + \sum_{j=1}^{m} b_j p_j(x).$$

In this case the coefficients solve the linear system

$$Aw + Pb = f$$
$$P^T w = 0$$

or

$$\begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w \\ b \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}.$$

Provided we have $rank\,(P) = m \leq n$ this system has a unique solution given by

$$\begin{aligned} \begin{bmatrix} w^* \\ b^* \end{bmatrix} &= \begin{bmatrix} A & P \\ P^T & 0 \end{bmatrix}^{-1} \begin{bmatrix} f \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} A^{-1} - A^{-1}P\left(P^T A^{-1} P\right)^{-1} P^T A & A^{-1}P\left(P^T A^{-1}P\right)^{-1} \\ \left(P^T A^{-1}P\right)^{-1} P^T A^{-1} & -\left(P^T A^{-1}P\right)^{-1} \end{bmatrix} \begin{bmatrix} f \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \left(A^{-1} - A^{-1}P\left(P^T A^{-1} P\right)^{-1} P^T A\right) f \\ \left(P^T A^{-1}P\right)^{-1} P^T A^{-1} f \end{bmatrix}. \end{aligned}$$

A useful variant of the RBF are called the 'compactly-supported radial basis functions', which are generated recursively using the formula

$$\varphi_{s,k} = \mathcal{I}^k \varphi_{\lfloor s/2 \rfloor + k + 1}$$

where

$$(\mathcal{I}\varphi)(r) = \int_r^\infty t\varphi(t)\,dt$$

for $r \geq 0$ and $\lfloor \cdot \rfloor$ is the "floor" function (rounded down to the integer below $s/2$). The first four CS-RBF functions are

$$\begin{aligned} \varphi_{s,0}(r) &= (1-r)_+^l \\ \varphi_{s,1}(r) &= (1-r)_+^{l+1}\left((l+1)r + 1\right) \\ \varphi_{s,2}(r) &= (1-r)_+^{l+2}\left(\left(l^2 + 4l + 3\right)r^2 + (3l+6)r + 3\right) \\ \varphi_{s,3}(r) &= (1-r)_+^{l+3}\left(\left(l^3 + 9l^2 + 23l + 15\right)r^3 + \left(6l^2 + 36l + 45\right)r^2 + (15l+45)r + 15\right) \end{aligned}$$

where $l = \lfloor s/2 \rfloor + k + 1$ for $k = 0, 1, 2, 3$. s is the dimension and k governs smoothness.

We can also do "anisotropic" radial basis approximation, where we distort the coordinate system locally in order to preserve the local geometry of the problem. Suppose that $y = Mx$ where M is some nonsingular matrix. Then the radial basis interpolating function on y instead of x is

$$f(y) = \sum_{i=1}^{n} \omega_i \phi\left(\|y - y_i\|_M\right) + \sum_{j=1}^{m} \beta_j p_j(y)$$

where

$$\|x\|_M = \|Mx\|.$$

We then associate a different M_i matrix with each node x_i; it is unclear how one would apply this approach to an economics problem at this point, however, since we generally want to preserve global rather than merely local shape.

Related to the thin plate spline is the polyharmonic spline approximation

$$f(x) = \sum_{i=1}^{N} w_i \varphi\left(\|x - c_i\|\right) + v^T \begin{bmatrix} 1 \\ x \end{bmatrix}$$

where $\{c_i\}_{i=1}^{N}$ are the nodes the function must interpolate in d dimensions, $\{w_i\}_{i=1}^{N}$ are the spline weights, $\{v_i\}_{i=1}^{d+1}$ are the polynomial weights (for the linear case), and

$$\varphi(r) = \left\{ \begin{array}{c} r^k \quad k \text{ odd} \\ r^{k-1} \ln(r^r) \quad k \text{ even} \end{array} \right\}$$
$$r = \sqrt{(x-c)^T (x-c)}.$$

The weights solve

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}$$

where

$$A_{i,j} = \varphi\left(|c_i - c_j|\right)$$
$$B = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ c_1 & c_2 & \cdots & c_N \end{bmatrix}$$
$$f = \begin{bmatrix} f_1 & f_2 & \cdots & f_N \end{bmatrix}.$$

Generically we can invert the matrix to solve uniquely for the weights:

$$\begin{bmatrix} w \\ v \end{bmatrix} = \begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix}^{-1} \begin{bmatrix} f \\ 0 \end{bmatrix}.$$

11 Projection on Simulated Data

Consider the simple stochastic growth model

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

subject to

$$c_t + k_{t+1} \leq z_t k_t^{\alpha} + (1-\delta) k_t.$$

The Euler equation for this economy is

$$\left(z_t k_t^{\alpha} + (1-\delta) k_t - k_{t+1}\right)^{-\sigma} = \beta E_t \left[c_{t+1}^{-\sigma} \left(\alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta \right) \right].$$

The solution is a function $k_{t+1} = g(k_t, z_t)$ that satisfies this equation for any (k_t, z_t) pair.

Let the entire right-hand side be parametrized by the exponential log-polynomial

$$RHS = F(k_t, z_t) = \exp\left(a_0 + a_1 \log(k_t) + a_2 [\log(k_t)]^2 + a_3 \log(z_t) + a_4 [\log(z_t)]^2 + a_5 \log(k_t) \log(z_t) \right);$$

for this reason the method has been called the Parametrized Expectations Algorithm or PEA. Guess values for the coefficients $(a_i)_{i=0}^5$. We then solve the equation for k_{t+1} as a function of k_t and z_t:

$$k_{t+1}^* = \left(F(k_t, z_t)\right)^{-\frac{1}{\sigma}} - z_t k_t^{\alpha} + (1-\delta) k_t.$$

Using this rule and the stochastic process for z_t, we can generate an artificial sample (k_t, z_t, c_t). Next, we compute a time series for the RHS of the Euler equation:

$$m_{t+1} = c_{t+1}^{-\sigma} \left(\alpha z_{t+1} k_{t+1}^{\alpha-1} + 1 - \delta \right).$$

Finally we regress m_{t+1} on $F(k_t, z_t)$. We compare the coefficients obtained from the regression with those we guessed, then update and continue until they agree. Generally the updating would proceed by taking a linear combination of the guessed coefficients and the resulting ones and

using this as the new guess. That is, define $(\widehat{a}_i)_{i=1}^5$ as the coefficients obtained by the nonlinear regression, given a current guess $(a_i^n)_{i=1}^5$. The new guess for the coefficients would be

$$\left(a_i^{n+1}\right)_{i=1}^5 = \theta \left(a_i^n\right)_{i=1}^n + (1-\theta)\left(\widehat{a}_i\right)_{i=1}^5.$$

Generally you will need θ close to 1, particularly with a bad initial guess, and the regression step may be ill-conditioned.

Good starting guesses (such as ones derived from a local perturbation solution) help, since the PEA does not have good numerical convergence properties. Another improvement are artificial bounds on the realizations of z_t that are gradually eliminated, which helps avoid explosive simulation paths caused by initial guesses that are poor. Finally, we can instead use $E[m_{t+1}]$, which we compute using a numerical integration routine (more on this below); furthermore, we can do linear least squares as it is valid to regress $\log(E[m_{t+1}])$ on the log polynomial. We can also use orthogonal polynomials (such as Hermite polynomials) instead of monomials as our basis, for reasons we already know, and we can generate our grid using only every I realizations (which limits multicollinearity).

The example above had the property that we could solve exactly for decision variables; of course in many cases this solution is not possible. Then we simply need to solve a nonlinear system of equations. For example, if we have elastic labor supply then the equilibrium is given by two equations:

$$\left(z_t k_t^\alpha h_t^{1-\alpha} + (1-\delta)k_t - k_{t+1}\right)^{-\sigma} = \beta E_t\left[c_{t+1}^{-\sigma}\left(\alpha z_{t+1} k_{t+1}^{\alpha-1} h_{t+1}^{1-\alpha} + 1 - \delta\right)\right]$$
$$\left(z_t k_t^\alpha h_t^{1-\alpha} + (1-\delta)k_t - k_{t+1}\right)^{-\sigma}(1-\alpha) z_t k_t^\alpha h_t^{-\alpha} = \theta(1-h_t)^{-\gamma}.$$

Again parameterizing the expectational term, we can solve the resulting two equations for (k_{t+1}, h_t). There are several clever tricks that can be used to rewrite the equation to make the solution easier, one of which we will use below.

One important aspect of using the PEA is that we must evaluate how good our approximation is. That is, does our functional form accurately capture the behavior of the conditional expectation of the marginal utility of future consumption? If it does not, we cannot claim to have approximated the solution. There are several ways to compute a measure of fit: we could use simple linear R^2 measure, we could use the standard deviation of the one-step ahead forecast $\widehat{\sigma}$,

or we can use a derivative of Hansen's J-statistic suggested by den Haan and Marcet (1994). The J-statistic for our economy is given by

$$J = TB_T' A_T^{-1} B_T$$

where

$$B_T = \frac{1}{T}\sum e_t \otimes h\left(G_t\right)$$
$$A_T = \frac{1}{T}\sum e_t^2 \otimes h\left(G_t\right) h\left(G_t\right)'$$
$$e_t = R_t - \exp\left(a_0 + a_1 \log\left(k_{t-1}\right) + a_2 \left[\log\left(k_{t-1}\right)\right]^2 + a_3 \log\left(z_{t-1}\right) + a_4 \left[\log\left(z_{t-1}\right)\right]^2 + a_5 \log\left(k_{t-1}\right)\log\left(z_{t-1}\right)\right)$$
$$R_t = \beta\left(z_{t+1}k_{t+1}^{\alpha} + \left(1-\delta\right)k_{t+1} - k_{t+2}\right)^{-\sigma}\left(\alpha z_{t+1}k_{t+1}^{\alpha-1} + 1 - \delta\right)$$

and $h\left(G_t\right)$ is some vector function of variables dated t and before. That is, e_t is the forecast error caused by using the functional form and not the exact function; note that we are effectively computing the expectation in the Euler equation using one point, the realized value, and then using an unconditional expectation to condition it down via the law of iterated expectations. To give the test power, $h\left(\cdot\right)$ should contain a relatively large number of terms, including potentially fractional powers of state variables and other flexible approximators. This test statistic is χ^2 with degrees of freedom equal to the number of coefficients used in $h\left(\cdot\right)$. The den Haan-Marcet statistic tests whether elements of the information set are correlated with the Euler equation errors – furthermore, it tells us exactly how often such correlations should occur randomly. It is generally also useful to state the maximum error encountered, as this contains some useful information about where the approximation is worst, and to plot the errors as functions of the state variables.

The basic PEA algorithm is not particularly good – the iterations are often unstable and take a long time, and the use of the entire sample leads to multicollinearity in the "data" used for the projection step. A number of papers have created ad hoc mechanisms to induce the PEA to behave, but they seem to be problem-specific. We therefore consider how to improve on the basic PEA in general; the methods we will explore were developed to handle high-dimensional problems that multiply the bad behavior of the PEA, but that also help at low dimensions.

Let's take the growth model and rewrite the Euler equation as

$$k_{t+1} = E_t\left[\beta\frac{Du\left(c_{t+1}\right)}{Du\left(c_t\right)}\left(\exp\left(a_{t+1}\right)Df\left(k_{t+1}\right) + 1 - \delta\right)k_{t+1}\right]$$

along with the resource constraint

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

all we did was multiply the Euler equation by k_{t+1} on each side, and note that k_{t+1} can be passed through the expectation since it is known at time t. Guess that the policy function is

$$k_{t+1} = G(k_t, a_t) \approx \Psi(k_t, a_t; b)$$

for some vector of coefficients b. Note that this approach approximates the decision rule and not the conditional expectation, which will turn out to be a lot more accurate and convenient; also note that Ψ need not be a polynomial, it can be any scattered-grid interpolating function for which we can easily compute the relevant coefficients. In high dimensions only polynomials are really feasible, though, which is why we are discussing them exclusively.

1. Make an initial guess $b^{(0)}$. Draw a sequence of productivity shocks $\{a_t\}_{t=1}^T$ and set (k_0, a_0).

2. At step p, use $b^{(p)}$ to simulate the model for T periods:

$$\begin{aligned} k_{t+1} &= \Psi\left(k_t, a_t; b^{(p)}\right) \\ c_t &= \exp(a_t) f(k_t) + (1-\delta) k_t - k_{t+1}. \end{aligned}$$

   For more complicated models we may need to solve a system of nonlinear equations at this step; in big models it is critical to make this part fast and accurate, so doing it analytically is preferred if possible.

3. Define

$$y_t = \sum_{j=1}^{J} \omega_{t,j} \left(\beta \frac{Du(c_{t+1,j})}{Du(c_t)} \left(1 - \delta + \exp(a_{t+1,j}) Df(k_{t+1})\right) k_{t+1} \right)$$

   where

$$\begin{aligned} a_{t+1,j} &= \rho a_t + \epsilon_{t+1,j} \\ k_{t+2,j} &= \Psi\left(\Psi\left(k_t, a_t; b^{(p)}\right), a_{t+1,j}; b^{(p)}\right) \\ c_{t+1,j} &= \exp(a_{t+1,j}) f(k_{t+1}) + (1-\delta) k_{t+1} - k_{t+2,j} \end{aligned}$$

   with Gauss-Hermite nodes for ϵ.

4. Find $\widehat{b}$ that minimizes the errors in a regression equation of the form

$$y_t = \Psi\left(k_t, a_t; b\right) + \varepsilon_t.$$

5. Check for convergence in the capital stock series and update b if necessary:

$$b^{(p+1)} = \left(1 - \xi\right) b^{(p)} + \xi\widehat{b}.$$

6. Check for accuracy using either a deterministic sequence of points or a new random sequence of points.

The regression step is where we need to be careful, because the simulated data is highly collinear in general. We can detect multicollinearity in a set of points by computing the conditioning number of the matrix $X^T X$. The procedure we use is designed to deal with the potential for ill-conditioning in the projection matrix, should it appear.

First, we normalize the variables to zero mean and unit standard deviation:

$$\widehat{y}_t = \frac{y_t - \mu_y}{\sigma_y}$$
$$\widehat{x}_{i,t} = \frac{x_{i,t} - \mu_{x_i}}{\sigma_{x_i}}.$$

We then estimate the regression without a constant term to obtain $\widetilde{b}_i$ and find the non-normalized coefficients and intercept by

$$\widehat{b}_i = \frac{\sigma_y}{\sigma_{x_i}}\widetilde{b}_i$$

for $i \in \{1, ..., n\}$ and

$$\widehat{b}_0 = \mu_y - \sum_{i=1}^{n} \widetilde{b}_i \mu_{x_i}.$$

Now we construct the principal components. To begin, we compute the Singular Value Decomposition of X:

$$X = USV^T$$

where U $(T \times n)$ and V $(n \times n)$ are unitary matrices ($U^H U = I$ and $V^H V = I$) and S $(n \times n)$ is a diagonal matrix with elements

$$s_1 \geq s_2 \geq ... \geq s_n \geq 0$$

known as singular values. The singular values of X are related to the singular values of X^TX by

$$s_i = \sqrt{\lambda_i}.$$

Note that this relationship implies the OLS estimator can be written

$$\begin{aligned}\widehat{b} &= \left(X^TX\right)^{-1}\left(X^Ty\right) \\ &= VS^{-1}U^Ty.\end{aligned}$$

In general S is better conditioned than X^TX and so the matrix will be easier to accurately invert; if $s_{ii} = 0$ then we set that element of S^{-1} to zero, otherwise we set it to s_{ii}^{-1}. Singular values identify the directions in the data with the most variation, and therefore can be used to eliminate (linear combinations of) variables that do not vary independently (data reduction).

Next, make a linear transformation of X:

$$Z = XV.$$

The vectors $\{z_1, ..., z_n\}$ are called the principal components of X and are mutually orthogonal with

$$z_i^T z_i = s_i.$$

The sample mean of each component is zero and the variance is

$$\frac{s_i^2}{T}.$$

We can make the problem better conditioned by discarding some low-variance components. Let κ denote the largest condition number for X that we find acceptable. We drop any component for which

$$\frac{s_1}{s_n} \geq \kappa.$$

Define the matrix Z^r as the Z matrix with only the first r components retained; this matrix by definition has conditioning number no bigger than κ. Running the regression using Z^r yields

$$y = Z^r\vartheta^r + \varepsilon$$

using OLS (since it is now well-behaved). We then recover the coefficients of interest by

$$\widehat{b}\,(\kappa) = V^r\widehat{\vartheta^r}.$$

An alternative to the principal component method is to use a truncated SVD. Let

$$X^r = U^r S^r \left(V^r\right)^T$$

be the SVD on the first r components ordered by singular value. That is, U^r and V^r are the first r columns of U and V and S^r is a diagonal matrix of the largest r singular values of X. Then we run the regression

$$y = U^r S^r \vartheta^r + \varepsilon$$

and recover

$$\begin{aligned}\widehat{b}\left(r\right) &= V^r \widehat{\vartheta^r} \\ &= V^r \left(S^r\right)^{-1} \left(U^r\right)^T y.\end{aligned}$$

The two methods are related by

$$Z^r = X^r V^r.$$

In my experience neither one is clearly better than the other.

For the integration step we could use Monte Carlo integration, but that turns to be worse than Gauss-Hermite quadrature, even one-point quadrature where we use only the conditional mean ($\epsilon_{t+1,1} = 0$) and weight 1. Since we primarily need simulation methods for large problems, we need low-cost integration methods like monomial methods and one-point quadrature to make these methods feasible. Monte Carlo methods converge slowly and therefore should only be used as a last resort.

11.1 Adaptive Grid and Clustering Methods

PEA explicitly constructs the solution on the simulated values of the state, which is very wasteful in one sense but conserves effort in another. It is wasteful because the realizations of the capital stock are highly autocorrelated with low conditional variance, meaning they tend to be close together and bunched up around the simulation mean. Thus, if decision rules are smooth, a lot of the points play no role in determining their values; the points are just too close together. On the other hand, the points are located only where the model lives, so that we do not waste effort computing solutions in parts of the state space the model never visits.

We can get the best of both worlds by setting our grid iteratively – solve the model, find the ergodic set, put points in that set only, and iterate to convergence (Maliar and Maliar 2016). The first step is easy – use a low accuracy method to construct an initial ergodic set (say, by simulating the perturbation solution). We then fix a projection grid on the ergodic set; since we don't need all the points in the simulation, we use a "clustering algorithm" to isolate groups of points that will be "represented" by a single point (either the center or the mean of the cluster). There are several options for this algorithm which are detailed in the next few subsections. We then do a standard projection on this grid, use this solution to generate a new grid, and repeat until the ergodic moments converge (in general, we cannot guarantee that we get the same points iteration by iteration, so the discrete representation of the ergodic set itself likely will not converge).

To keep the calculations managable, we will use a subsample of the ergodic set, called the **essentially ergodic set**. To find an essentially ergodic set, which is just an ergodic set with the least likely points dropped, we take a simulation and sample only every k periods to generate a sample $(x_1, ..., x_n)$ that has approximately been purged of serial correlation. We then use the Gaussian kernel estimator

$$\widehat{g}(x) = \frac{1}{n(2\pi)^{\frac{d}{2}} n^{-\frac{1}{d+4}}} \sum_{i=1}^{n} \exp\left(-\frac{D(x, x_i)}{2n^{-\frac{1}{d+4}}}\right)$$

on the normalized principal component matrix $\widetilde{Z}$ defined by

$$\begin{aligned}
\widetilde{X}_j &= \frac{X_j - \mu_j}{\sigma_j} \\
\widetilde{X} &= (x_1, ..., x_n) = USV^T \\
Z &= \widetilde{X}V \\
\widetilde{Z}_j &= \frac{Z_i}{\omega_j} \\
D(x, x_i) &= \|z - z_i\|_2
\end{aligned}$$

to estimate the density of point x. If $\widehat{g}(x) < \eta$ for some specified parameter η we discard x. Now we have an ergodic set that contains only points with sufficient likelihood of being visited; we can choose η such that the number of points dropped equals some fraction of the sample (say, 5 percent) or we can choose it to eliminate points with some fixed maximum probability of being visited.

11.1.1 Ward's Clustering Algorithm

The first clustering algorithm we will discuss is called Ward's clustering. For all cases, we will normalize variables to the same unit using principal component transformation. Specifically, let $X \in \mathcal{R}^{N \times L}$ be a data matrix, normalized to zero mean and unit variance. Then we can use the spectral decomposition

$$X^T X = V \Lambda V^T$$

where $\Lambda \in \mathcal{R}^{L \times L}$ is a diagonal matrix of eigenvalues and $V \in \mathcal{R}^{L \times L}$ is an orthogonal eigenvector matrix; assume that Λ is ordered by increasing diagonal elements which is the usual way algorithms construct Λ and V anyway. Let

$$Z = XV;$$

the elements of $Z \in \mathcal{R}^{N \times L}$ $(z^1, ..., z^L)$ are the principal (orthogonal) components of X. This transformation in effect rotates the ellipse that defines the ergodic set of X to align its axes with the standard basis vectors; now normalize Z to unit variance again and we get a hypersphere. As we have already noted, the advantage is that the ratio of the volume of a hypersphere to that of a hypercube that encloses the hypersphere goes to zero as the number of dimensions goes to infinity, and does so very rapidly. Note – it is much faster to construct the principal components using the Singular Value Decomposition

$$X^T X = U \Sigma W^H$$

and then computing

$$Z = U\Sigma.$$

1. Define a cluster set. In order, merge the closest components of the Z data together and create a hierarchical tree of clusters (starting with L clusters and continuing down to 1). Then, choose the number of clusters desired. Let

$$SSE_A = \sum_{i \in A} \sum_{l=1}^{L} \left(x_i^l - \frac{1}{h_A} \sum_{i \in A} x_i^l \right)^2$$

be the sum of squared errors (variance) of cluster A, where h_A is the number of points in A. The distance between two sets of clusters is the change in the SSE obtained by proposing a

merger of A and B:

$$d(A, B) = SSE_{AB} - (SSE_A + SSE_B),$$

where SSE_{AB} is the sum of squared errors for the merged cluster $A \cup B$.

2. The center of each cluster forms the grid for the projection algorithm; centers of clusters are computed by taking the mean of all elements in the cluster.

Here is a simple example of Ward's clustering.

	Variable	
Observation	x_i^1	x_i^2
1	1	0.5
2	2	3
3	0.5	0.5
4	3	1.6
5	3	1

We have to calculate the SSE for each possible merger; since each element is just one observation,

they have zero variance to start. Thus, we have (for example)

$$
\begin{aligned}
d(1,2) &= SSE_{1,2} \\
&= \sum_{i=1}^{2}\sum_{l=1}^{2}\left(x_i^l - \frac{1}{2}\sum_{i=1}^{2}x_i^l\right)^2 \\
&= \left[\left(x_1^1 - \frac{1}{2}\left(x_1^1 + x_2^1\right)\right)^2 + \left(x_1^2 - \frac{1}{2}\left(x_1^2 + x_2^2\right)\right)^2\right] + \left[\left(x_2^1 - \frac{1}{2}\left(x_1^1 + x_2^1\right)\right)^2 + \left(x_2^2 - \frac{1}{2}\left(x_1^2 + x_2^2\right)\right)^2\right] \\
&= \left(1 - \frac{1}{2}(1+2)\right)^2 + \left(0.5 - \frac{1}{2}(0.5+3)\right)^2 + \left(2 - \frac{1}{2}(1+2)\right)^2 + \left(3 - \frac{1}{2}(0.5+3)\right)^2 \\
&= 3.625 \\
d(1,3) &= \left(1 - \frac{1}{2}(1+0.5)\right)^2 + \left(0.5 - \frac{1}{2}(0.5+0.5)\right)^2 + \left(0.5 - \frac{1}{2}(0.5+0.5)\right)^2 + \left(0.5 - \frac{1}{2}(0.5+0.5)\right)^2 \\
&= 0.0625 \\
d(1,4) &= \left(1 - \frac{1}{2}(1+3)\right)^2 + \left(0.5 - \frac{1}{2}(0.5+1.6)\right)^2 + \left(3 - \frac{1}{2}(1+3)\right)^2 + \left(1.6 - \frac{1}{2}(0.5+1.6)\right)^2 \\
&= 2.605 \\
d(1,5) &= \left(1 - \frac{1}{2}(1+3)\right)^2 + \left(0.5 - \frac{1}{2}(0.5+1)\right)^2 + \left(3 - \frac{1}{2}(1+3)\right)^2 + \left(1 - \frac{1}{2}(0.5+1)\right)^2 \\
&= 2.125
\end{aligned}
$$

and so on. Once we assemble all of these points, we merge the two closest according to our distance measure to form a new cluster (in this case 1 and 3 are merged), and repeat until we have only one cluster; we then select from the tree of clusters the "row" corresponding to the number of clusters we want. Essentially, Ward's algorithm trades off the increase in the "average within-cluster variance" created by expanding the sets against the reduction in the "across-cluster variance" created by reducing the number of clusters; some implementations allow you to specify the ratio of these changes that triggers the end of the process, endogenously determining the number of clusters.

11.1.2 k-means Clustering

An alternative to Ward's clustering is the k-means clustering method, which seeks to partition the n observations into $k \leq n$ sets to minimize the within-cluster variance:

$$\arg\min_{S} \left\{ \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \right\} = \arg\min_{S} \left\{ \sum_{i=1}^{k} |S_i| \, var\left(S_i\right) \right\}.$$

The standard approach to constructing the clusters is called Lloyd's algorithm. Given a set of initial cluster means $\{\mu_k\}$, Lloyd's algorithm alternates between two steps:

1. Assignment: Assign each observation to the cluster with the nearest mean;

2. Update: Update the means according to

$$\mu_k^{i+1} = \frac{1}{\left|S_k^i\right|} \sum_{x_j \in S_k^n} x_j.$$

The algorithm terminates when the assignments stop changing. Unfortunately, the k-means problem is NP-hard, and this algorithm is not guaranteed to converge. Initialization is frequently done either by the Forgy method (k randomly-chosen points), the Random Partition method (k randomly-chosen clusters), or the k-means++ algorithm that picks a centroid at random, then picks the next one using an inverse-weighted distribution based on $\|x - \mu_i\|^2$; that is, if μ_i is the most-recently chosen initial centroid, then we pick from a distribution that weights the remaining data points according to the square of their distance, maximizing the likelihood of picking a distant point. The last choice is likely the most efficient. Then we run Lloyd's algorithm starting from those initial points; efficient versions of Lloyd's algorithm avoid the costly repeated distance calculations as the clusters start to settle down.

Alternatives are k-medians clustering (medians instead of means) and k-medoid clustering, where a **medoid** is defined as

$$x_{medoid} = \arg\min_{y} \left\{ \sum_{i=1}^{n} d\left(x_i, y\right) \right\};$$

the medoid of a data set is the point that minimizes the sum of distances to all other points. Note that, unlike a mean or median, a medoid is required to be one of the data points. With the large samples we can use to construct our grids, this restriction is not likely to be an issue; the more important problem is that computing the medoid is costly.

11.1.3 ε-Distinguishable Sets

An alternative to clustering is to use ε-distinguishable sets (Maliar and Maliar 2016).

Definition 116 *Let (X, D) be a bounded metric space. A set P^{ε} consisting of points $(x_1^{\varepsilon}, ..., x_m^{\varepsilon}) \in X$ is called ε-**distinguishable** if $D\left(x_i^{\varepsilon}, x_j^{\varepsilon}\right) > \varepsilon$ for all $i \neq j$.*

We will use points in an ε-distinguishable set to "cover" the space with as little overlap as possible. To construct an EDS, we let P^{ε} begin as the empty set. Then we pick $x_i \in P$ and compute $D(x_i, x_j)\ \forall j$ in P; a good choice for the initial point is the one closest to the ergodic mean. We discard all x_j for which $D(x_i, x_j) < \varepsilon$. We then remove x_i from P and place it in P^{ε}. We then continue until all points have been removed from P. In the end we will have a good set of points if the discrepancy from a uniform distribution is small.

Definition 117 *Let P be a set of points $(x_1, ..., x_n) \in X \subset \mathcal{R}^d$ and let $\mathcal{J}$ be a family of Lebesgue-measurable subsets of X. The **discrepancy of** P **under** J is given by*

$$D_n(P; \mathcal{J}) = \sup_{J \in \mathcal{J}} \left\{ \frac{C(P; J)}{n} - \lambda(J) \right\}$$

where $C(P; J)$ counts the number of points from P in J and $\lambda(J)$ is a Lebesgue measure on J.

Definition 118 *The **star discrepancy** $D_n^*(P; \mathcal{J})$ is defined as the discrepancy of P over the family J generated by the intersection of all subintervals of $\mathcal{R}^d$ of the form $\prod_{i=1}^{d} [-\infty, v_i)$ where $v_i > 0$.*

We call a sequence S **low-discrepancy** if

$$D_n^*(S; \mathcal{J}) = O\left(n^{-1} (\log(n))^d\right);$$

that is, if the star discrepancy goes to zero asymptotically at rate at least $n^{-1} (\log(n))^d$. The star discrepancy of a sequence randomly drawn from a uniform distribution on $[0, 1]^d$ converges to zero almost everywhere at rate

$$\log(\log(n))^{\frac{1}{2}} (2n)^{-\frac{1}{2}}.$$

We have seen discrepancy before, in the context of quasi-Monte Carlo integration; here we are using it for essentially the same purpose – to find points that best cover an underlying space.

One could also use a Halton or Sobol' sequence restricted to the ergodic set; if the set has a weird shape this restriction might require us to use rejection sampling methods to determine whether the points are "in" or "out", which would seem to be more costly than it is worth.

Constructing the density can be expensive, since it involves pairwise comparisons of all points in the ergodic sample. If we reverse the order of operations – that is, we construct the EDS first and then estimate the density and remove low probability points – the number of operations drops considerably.

12 Dynamic Games and Approximating Sets

The quasi-geometric discounting problem we solved earlier using perturbation actually has many non-Markovian equilibria. To refresh, remember that an agent with these preferences makes decisions described by the equilibrium of the dynamic game

$$V_0(k) = \max_{k'} \left\{ u\left(f(k) - k'\right) + \delta\beta V\left(k'\right) \right\}$$

where

$$\begin{aligned} V(k) &= u\left(f(k) - g(k)\right) + \delta V\left(g(k)\right) \\ g(k) &= \operatorname*{argmax}_{k'} \left\{ u\left(f(k) - k'\right) + \delta\beta V\left(k'\right) \right\}. \end{aligned}$$

Above we found continuously-differentiable equilibria ($g(k)$ can be approximated as a Taylor expansion in (k, β) or as a projection onto the space of polynomials), but as noted there are many other equilibria (provided the state space is bounded; see Chatterjee and Eyigungor 2019). Bernheim, Ray, and Yeltekin (2015) study a related problem and discuss how to compute the entire set of payoffs that can be supported in an equilibrium; one can then recover the strategies afterwards, in particular the non-Markovian ones that are supported by trigger strategies. Since their problem is a little different, we'll state it clearly here.

The preferences of the household are given by

$$u(c_0) + \beta \sum_{t=1}^{\infty} \delta^t u(c_t)$$

and the budget set is

$$c_t = A_t + y + b$$
$$A_{t+1} = \min\left\{\overline{A}, \alpha\left(A_t + y - c_t\right)\right\}$$

This problem implies that assets must lie on the interval $\left[-\alpha b, \overline{A}\right]$. We will look for the state-contingent set of payoffs $\mathcal{W}(A)$ that a household can get in equilibrium; this set, once we introduce a public randomization device (like a sunspot), is a convex set (thus, an interval).

1. Define grid for assets $\mathcal{A}$ on $\left[-\alpha b, \overline{A}\right]$;

2. Guess utility bounds $\left[\overline{w}(A), \underline{w}(A)\right]$ for each $A \in \mathcal{A}$;

3. Let
$$\mathbf{A}(A_j) = \left\{A_i \in \mathcal{A} | c(A_i, A_j) = A_j + y - \frac{A_i}{\alpha} \in [0, A_j + y + b]\right\};$$
for each $A_i \in \mathbf{A}(A_j)$ compute
$$\widetilde{P}(A_i, A_j) = (1+\beta)\, u(c(A_i, A_j)) + \delta \underline{w}(A_i)$$
and for each $A_j \in \mathcal{A}$ compute
$$P(A_j) = \max_{A_i \in \mathbf{A}(A_j)} \left\{\widetilde{P}(A_i, A_j)\right\};$$

4. Compute
$$\overline{\Phi}(\overline{w}, \underline{w})(A_j) = \max_{A_i \in \mathbf{A}(A_j)} \left\{u(c(A_i, A_j)) + \delta \overline{w}(A_i)\right\}$$
$$(1+\beta)\, u(c(A_i, A_j)) + \delta \overline{w}(A_i) \geq P(A_j)$$
and
$$\underline{\Phi}(\overline{w}, \underline{w})(A_j) = \min_{A_i \in \mathbf{A}(A_j)} \left\{u(c(A_i, A_j)) + \delta \max\left\{\underline{w}(A_i), \frac{1}{\delta}\left(P(A_j) - (1+\beta)\, u(c(A_i, A_j))\right)\right\}\right\};$$

5. Set
$$\begin{bmatrix} \overline{w}(A_j) \\ \underline{w}(A_j) \end{bmatrix} = \begin{bmatrix} \overline{\Phi}(\overline{w}, \underline{w})(A_j) \\ \underline{\Phi}(\overline{w}, \underline{w})(A_j) \end{bmatrix}$$
and repeat steps 3-5 until convergence.

Here, since the set is one-dimensional, we can approximate it by finding the end points; the above algorithm iterates on these points until the set of payoffs converges. The optimization method used here is "brute force" grid search over a discrete state and control space, because the constraint set is not particularly well-behaved across states A_ℓ; it can fail to be lower hemicontinuous, meaning it can experience "explosions" where the set gets discontinuously larger.

In higher dimensions, we can approximate the set of payoffs using hyperplanes.

Definition 119 *Let $L^{n-1} \subset \mathcal{R}^n$ be a linear $n-1$-dimensional subspace of $\mathcal{R}^n$. The set*

$$H(p, \alpha) = \{x \in \mathcal{R}^n : px = \alpha\}$$

*with $p \neq 0$ is called a **hyperplane**.*

The idea of approximating a convex set is that, at any point in the boundary, we can invoke the Hahn-Banach theorem to obtain a supporting hyperplane.

Definition 120 *A hyperplane $H(p, \alpha)$ **separates** two sets A and B if $px \leq \alpha \leq py$ for all $x \in A$ and $y \in B$. If one inequality holds with equality the hyperplane **supports** that set.*

Proposition 121 *(Hahn-Banach Theorem) $C \subset \mathcal{R}^n$ is nonempty, closed, and convex and $z \notin C$. There exists $y \in C$ such that*

$$pz < py = \alpha = \inf_{c \in C} \{pc\};$$

that is, there exists a hyperplane supporting C and separating C from any convex set X that contains z but does not intersect C.

Obviously we cannot do this construction at every point in the boundary, so we use ideas from projection to construct a sample of hyperplanes. The intersection of the "left side" of these hyperplanes (that is, the points x where $px \leq \alpha$) is an "outer approximation" to the set; we call it an outer approximation because it contains some points that are not in the set due to approximation error between supporting points. However, we know for sure that anything outside the set is not an equilibrium payoff. We can also generate an "inner approximation" just using the convex hull of the approximation points; due to nonlinearity some points in the

set will lie outside the convex hull, but we know for sure that anything in the convex hull is an equilibrium payoff. The convex hull of a set is given by

$$co\left(C\right)=\left\{b\in\mathcal{R}^{n}:b=\sum_{i=1}^{n}\lambda_{i}y_{i},\sum_{i=1}^{n}\lambda_{i}=1,\lambda\geq0\right\};$$

the convex hull is the smallest convex set that contains C. The difference between the outer and inner approximations gives us the approximation error; as the number of points gets large the approximation error converges to zero.

For future notational ease, let $g\left(a,x\right)$ denote the state transition function, Π_{i} be the current payoff to player i, and the incentive compatibility constraint (that defines a subgame perfect equilibrium) as

$$IC\left(i,a,\overline{a},w:x\right)=\left(1-\delta\right)\Pi_{i}\left(a_{i},a_{-i},x\right)+\delta w_{i}-\left(1-\delta\right)\Pi_{i}\left(\overline{a},a_{-i},x\right)-\delta\mu_{i,g(\overline{a},a_{-i},x)}$$

for any alternative feasible action $\overline{a}$ and the resulting alternative continuation $\mu_{i,g(\overline{a},a_{-i},x)}$.

We iterate on these two sets as follows.

1. Outer Monotone Approximation

   (a) Inputs:

      i. Subgradient points $R_{x}=\left\{r_{x}^{1},...,r_{x}^{m_{x}}\right\}$ where x is the current state;

      ii. Boundary points $Z_{x}=\left\{z_{x}^{1},...,z_{x}^{m_{x}}\right\}$;

      iii. Define the correspondence

$$W_{x}=\bigcap\nolimits_{j=1}^{m_{x}}\left\{b\in\mathcal{R}^{n}:r_{x}^{j}\left(b-z_{x}^{j}\right)\leq0\right\}.$$

   (b) Search subgradient points $S_{x}=\left\{s_{x}^{1},...,s_{x}^{m_{x}}\right\}$ used to update subgradient points;

   (c) For each x and each $s\in S_{x}$ solve

      i. For each action profile a compute

$$c\left(a,s,x\right)=\max_{w}\left\{s\left[\left(1-\delta\right)\Pi\left(a,x\right)+\delta w\right]:w\in W_{g(a,x)},IC\left(i,a,\overline{a},w\right)\geq0\ \forall\overline{a}\in A_{i}\right\}$$

$$w^{*}\left(a,s,x\right)=\operatorname*{argmax}_{w}\left\{s\left[\left(1-\delta\right)\Pi\left(a,x\right)+\delta w\right]:w\in W_{g(a,x)},IC\left(i,a,\overline{a},w\right)\geq0\ \forall\overline{a}\in A_{i}\right\}$$

$$v\left(a,s,x\right)=\left(1-\delta\right)\Pi\left(a,x\right)+\delta w^{*}\left(a,s,x\right);$$

if no feasible point exists, then

$$c(a,s,x) = -\infty$$
$$w^* = \varnothing$$
$$v(a,s,x) = -\infty.$$

ii. Choose action profile

$$a^*(s,x) \in \underset{a}{\operatorname{argmax}} \{c(a,s,x)\}$$
$$v^*(s,x) = v(a^*(s,x),s,x).$$

(d) Update R and Z

i. $R_x^+ = S_x$;

ii. $Z_x^+ = \{v^*(s,x) : s \in S_x\}$;

iii. Define outer approximation:

$$W_x^+ = \bigcap\nolimits_{j=1}^{l_x} \left\{b \in \mathcal{R}^n : r_x^j b \leq r_x^j z_x^j, r_x \in R_x^+, z_x \in Z_x^+\right\}.$$

iv. Check convergence:

$$\left|\left|W_x^+ - W_x\right|\right| = \max_x \left\{\max_{j=1,...,l_x} \left\{\left|r_x^{j,+} z_x^{j,+} - r_x^j z_x^j\right|\right\}\right\} < \epsilon.$$

2. Inner Monotone Approximation

(a) Inputs:

i. Set of points $Z_x = \left\{z_x^1, ..., z_x^{m_x}\right\}$;

ii. Define the correspondence

$$W_x = co(Z_x);$$

(b) Search subgradient points $S_x = \left\{s_x^1, ..., s_x^{m_x}\right\}$;

(c) Update Z. At each x and each $s \in S_x$, solve

i. For each action profile a compute

$$c(a,s,x) = \max_w \left\{s\left[(1-\delta)\Pi(a,x) + \delta w\right] : w \in co\left(Z_{g(a,x)}\right), IC(i,a,\overline{a},w) \geq 0 \ \forall \overline{a} \in A_i\right\}$$
$$w^*(a,s,x) = \underset{w}{\operatorname{argmax}} \left\{s\left[(1-\delta)\Pi(a,x) + \delta w\right] : w \in co\left(Z_{g(a,x)}\right), IC(i,a,\overline{a},w) \geq 0 \ \forall \overline{a} \in A_i\right\}$$
$$v(a,s,x) = (1-\delta)\Pi(a,x) + \delta w^*(a,s,x);$$

if no feasible point exists, then

$$c(a,s,x) = -\infty$$
$$w^* = \varnothing$$
$$v(a,s,x) = -\infty.$$

ii. Find set to be used for computing equilibrium strategies:

$$\Omega_x^a = \{v(a,s,x) : v(a,s,x) > -\infty, s \in S_x\};$$

iii. Choose action profile

$$a^*(s,x) \in \underset{a}{\operatorname{argmax}} \{c(a,s,x)\}$$
$$v^*(s,x) = v(a^*(s,x),s,x).$$

(d) Define inner approximation

$$Z_x^+ = Z_x \cup \{v^*(s,x) : s \in S\}$$

and check convergence:

$$\left|\left|W_x^+ - W_x\right|\right| = \max\left\{\max_{z\in Z_x}\min_{w\in W_x^+}\{||z-w||\}, \max_{z\in Z_x^+}\min_{w\in W_x}\|z-w\|\right\} < \epsilon.$$

3. Find equilibrium strategies:

(a) Inputs:

i. $v_x \in co(Z_x)$;

(b) Choose equilibrium actions and continuation values that support v_x:

i. Check if v_x is pure-strategy implementable. If $v_x \in co(\Omega_x^a)$ for some a, then

$$w(a,x) = \delta^{-1}(v_x - (1-\delta)\Pi(a,x)) \in co\left(Z_{g(a,x)}\right)$$

and therefore a is a pure strategy action profile that supports v_x. If multiple pure strategy actions exist, pick one.

ii. If there is no a such that $v_x \in co\left(\Omega_x^a\right)$, then pick λ such that

$$v_x = \sum\nolimits_a \lambda_a v_x^a$$

with

$$\lambda \geq 0$$
$$\sum\nolimits_a \lambda_a = 1.$$

Yeltekin, Cai, and Judd (2016) prove these iterations converge to a **self-generating correspondence**:

$$W \subseteq B^*\left(W\right),$$

where

$$B^*\left(W\right)_x = \bigcup\nolimits_{(a,w)} \left\{\left(1-\delta\right)\Pi\left(a,x\right) + \delta w\right\}$$
$$w \in W_{g(a,x)}$$
$$IC\left(i,a,\overline{a},w:x\right) \geq 0.$$

A self-generating correspondence is a fixed point in the space of correspondences; since the operator B^* is monotone (in the set inclusion ordering, meaning if $W \subseteq W'$ then $B^*\left(W\right) \subseteq B^*\left(W'\right)$) and preserves compactness, one can show that the equilibrium value correspondence is the maximal fixed point of B^* and can be obtained by iterations starting from some large-enough initial set W_0.

This approach can be used to solve for all the sustainable equilibria of optimal policy games as well, where the government plays a game against a continuum of private agents. It can also be used to solve for non-optimal equilibria in models where continuous Markov equilibria may not exist. To see an example, I follow Feng, Miao, Peralta-Alva, and Santos (2011) to illustrate a method that works even for non-convex sets. The implicit randomization method used above does not make sense in a competitive economy, so this approach does not convexify things but rather uses discretization methods to describe non-convex sets.

Take the growth model with Euler equation

$$Du\left(F\left(k_t,1\right)+\left(1-\delta\right)k_t-k_{t+1}\right) = \beta Du\left(F\left(k_{t+1},1\right)+\left(1-\delta\right)k_{t+1}-k_{t+2}\right)\left(D_1F\left(k_{t+1},1\right)+1-\delta\right).$$

Under some specifications of F (such as those with nonconvexities), a continuous function $g(k_t)$ may not exist that solves this equation:

$$Du\left(F\left(k_t,1\right)+\left(1-\delta\right)k_t-g\left(k_t\right)\right)=\beta Du\left(F\left(g\left(k_t\right),1\right)+\left(1-\delta\right)g\left(k_t\right)-g\left(g\left(k_t\right)\right)\right)\left(D_1F\left(g\left(k_t\right),1\right)+1-\delta\right).$$

Define

$$m_t\equiv Du\left(c_t\right)\left(D_1F\left(k_t,1\right)+1-\delta\right).$$

Let $K\times M$ be the state space for (k,m). Let $V:K\to M$ be set-valued and sufficiently large to contain all equilibria. For each k, let $m\in B\left(V\right)\left(k\right)$ if there exists (c,k_+,m_+) such that

$$\begin{aligned}c+k_+&=F\left(k,1\right)+\left(1-\delta\right)k\\Du\left(c\right)&=\beta m_+\\k&>0\\c&>0\\m_+&\in V\left(k_+\right).\end{aligned}$$

That is, given k, m lies in an interval

$$V\left(k\right)=\left[\underline{m},\overline{m}\right]\left(k\right).$$

To implement this procedure, we start by picking $\left[\underline{m}_0,\overline{m}^0\right]\left(k\right)$. For each k check, for some $m_+\in\left[\underline{m}_0,\overline{m}^0\right]\left(k_+\right)$, if

$$\begin{aligned}c&=\left(Du\right)^{-1}\left(\beta m_+\right)\\k_+&=F\left(k,1\right)+\left(1-\delta\right)k-c>0;\end{aligned}$$

this check can be done "brute force" using discretization or by using an SQP method to hunt for a feasible point of the set

$$B\left(k_+,m_+\right)=\left\{\left(k_+,m_+\right):m_+\in\left[\underline{m}_0,\overline{m}^0\right]\left(k_+\right),k_+\in\left(0,F\left(k,1\right)+\left(1-\delta\right)k-\left(Du\right)^{-1}\left(\beta m_+\right)\right)\right\}.$$

If some (k_+,m_+) exists, then

$$m=Du\left(c\right)\left(D_1F\left(k,1\right)+1-\delta\right)$$

is contained in $\left[\underline{m}_1, \overline{m}^1\right](k)$; the updated interval takes the maximum and minimum over the new points and repeats the process until the endpoints converge. We will need to interpolate the functions that define the lower and upper bounds, $\underline{m}_0(k_+)$ and $\overline{m}^0(k_+)$, since they are defined only at a fixed set of points and k_+ will generally not be one of them; since they are one-dimensional, we can use a pchip spline.

The brute force method is to discretize (k_+, m_+) and systematically check whether a given point (k_+, m_+) satisfies

$$\begin{aligned} m_+ &\in \left[\underline{m}_0, \overline{m}^0\right](k_+) \\ k_+ &\in (0, F(k,1) + (1-\delta)k) \end{aligned}$$

and

$$F(k,1) + (1-\delta)k - (Du)^{-1}(\beta m_+) \in (0, F(k,1) + (1-\delta)k).$$

If the grids for k and k_+ are not the same, we interpolate the boundary functions as noted above.

To obtain a feasible point, denote the nonlinear constraints by

$$g(x) \leq 0;$$

here, we eliminate c and consider the constraints

$$\begin{aligned} g_1(k_+, m_+) &= (Du)^{-1}(\beta m_+) + k_+ - F(k,1) - (1-\delta)k \\ g_2(k_+, m_+) &= \underline{m}_0(k_+) - m_+ \\ g_3(k_+, m_+) &= m_+ - \overline{m}^0(k_+) \end{aligned}$$

and the bounds

$$\begin{bmatrix} \epsilon \\ \underline{m}_0(\epsilon) \end{bmatrix} \leq \begin{bmatrix} k_+ \\ m_+ \end{bmatrix} \leq \begin{bmatrix} F(k,1) + (1-\delta)k - \epsilon \\ \overline{m}^0(F(k,1) + (1-\delta)k - \epsilon) \end{bmatrix}.$$

Given an initial guess at a feasible point $\left(k_+^0, m_+^0\right)$, we first solve the problem

$$\min_v \{\langle v, v \rangle\}$$

subject to

$$\begin{bmatrix} \epsilon \\ \underline{m}_0(\epsilon) \end{bmatrix} \leq \begin{bmatrix} k_+ \\ m_+ \end{bmatrix} + v \leq \begin{bmatrix} F(k,1) + (1-\delta)k - \epsilon \\ \overline{m}^0(F(k,1) + (1-\delta)k - \epsilon) \end{bmatrix}.$$

Then starting from $\left(k_+^0, m_+^0\right) + v$, we iterate on

$$\min_{(k_+,m_+)} \max_{j\in\{1,2,3\}} \left\{g_j\left(k_+, m_+\right)\right\}$$

subject to

$$\begin{bmatrix} \epsilon \\ \underline{m}_0\left(\epsilon\right) \end{bmatrix} \leq \begin{bmatrix} k_+ \\ m_+ \end{bmatrix} + v \leq \begin{bmatrix} F\left(k,1\right) + \left(1-\delta\right)k - \epsilon \\ \overline{m}^0\left(F\left(k,1\right) + \left(1-\delta\right)k - \epsilon\right) \end{bmatrix}.$$

using SQP until $\max_{j\in\{1,2,3\}} \left\{g_j\left(k_+, m_+\right)\right\} \leq 0$ is obtained. If this process converges to a point with $\max_{j\in\{1,2,3\}} \left\{g_j\left(k_+, m_+\right)\right\} > 0$, there are no feasible points for that k. Obviously for this case the SQP method is overkill, but for more complicated problems it may be more useful; we can also permit linear equality and inequality constraints.

A less trivial example is from Peralta-Alva and Santos (2012), which features a regressive capital income tax whose rate depends on aggregate, not individual, capital. The structural equations are

$$\begin{aligned} \frac{1}{c_t} &= \frac{0.95}{c_{t+1}}\left(1-\tau\left(k_{t+1}\right)\right)\frac{1}{3}k_{t+1}^{-\frac{2}{3}} \\ k_{t+1} &= k_t^{\frac{1}{3}} - c_t \end{aligned}$$

where

$$\tau\left(k\right) = \begin{cases} 0.1 & \text{if } k < 0.160002 \\ 0.05 - 10\left(k - 0.165002\right) & \text{if } 0.160002 \leq k \leq 0.170002 \\ 0.0 & \text{if } k > 0.170002 \end{cases}.$$

In this case, no continuous equilibrium function exists. There are three steady states, two saddle-stable ones surrounding an unstable one with complex roots, as the equation

$$1 = \frac{0.95}{3}\left(1-\tau\left(k^*\right)\right)\left(k^*\right)^{-\frac{2}{3}}$$

has the solutions

$$k^* \in \left\{0.15215, 0.165, 0.17820\right\}$$

with associated consumption values

$$c^* \in \left\{0.38171, 0.38348, 0.38453\right\}.$$

Peralta-Alva and Santos (2012) show that a piecewise linear projection can have low Euler equation errors, but the implied dynamics are not similar to the true solution and the function implies

additional false steady states. The lesson is really that you need to know your problem has a continuous solution if you are going to use a solution method that imposes continuity.

As before, we define

$$m_t = \frac{1}{3c_t}\left(1-\tau\left(k_t\right)\right)k_t^{-\frac{2}{3}}$$

and initialize our search with an interval $\left[\underline{m}_0, \overline{m}^0\right](k)$. Then we look for (k_+, m_+) such that

$$c = \frac{1}{\beta m_+}$$
$$k_+ = k^{\frac{1}{3}} - c$$

such that

$$m_+ \in \left[\underline{m}_0, \overline{m}^0\right](k_+)$$
$$k_+ > 0;$$

if such a point exists, then

$$m = \frac{1}{3c}\left(1-\tau\left(k\right)\right)k^{-\frac{2}{3}} \in \left[\underline{m}_1, \overline{m}^1\right](k).$$

13 Finite State Space Dynamic Programming

Assume that the state space is finite: k,$k' \in \{k_1, ..., k_n\}$. The value function then takes on only finitely-many values:

$$v\left(k_i\right) = \max_{k' \in \{k_1,...,k_n\}} \left\{u\left(f\left(k_i\right) + \left(1-\delta\right)k_i - k'\right) + \beta v\left(k'\right)\right\}.$$

Straightforward grid search can be implemented to solve the maximization. For each i, evaluate the RHS at each grid point for k':

$$\widehat{v}\left(k_i, k_j\right) = u\left(f\left(k_i\right) + \left(1-\delta\right)k_i - k_j\right) + \beta v\left(k_j\right).$$

Choose the j that attains the maximum value of $\widehat{v}$ and then update the value function guess:

$$v^{n+1}\left(k_i\right) = u\left(f\left(k_i\right) + \left(1-\delta\right)k_i - k_{j^*}\right) + \beta v\left(k_{j^*}\right).$$

It is trivial to prove that the RHS has a solution (finite number of possibilities means that the constraint set is compact and the objective function is trivially continuous) and that the above procedure defines a contraction map with a linear convergence rate equal to β.

Since the above approach will typically be quite slow, we will exploit some methods to speed it up. The first thing to do is to precompute the utility function at all grid points:

$$u\left(f\left(k_i\right)+\left(1-\delta\right)k_i-k_j\right)$$

does not depend on the value function, and so needs to be computed only once.

Next, since we know that the policy function is nondecreasing, we need only search over a limited range of k'. Assume that $j^*\left(k_{i-1}\right)$ is the policy choice at grid point $i-1$. Then we can begin our search for grid point i at this value, rather than k_1, since it cannot be optimal to choose less capital when endowed with more (although the planner may choose the same j^* at i as at $i-1$). Second, we know that the true objective on the RHS is concave (as opposed to the discrete approximation), so that we need only search until the objective begins to decline; that is, if

$$\widehat{v}\left(k_i,k_j\right)>\widehat{v}\left(k_i,k_{j+1}\right),$$

j is the optimal choice (provided that you guessed a discrete set of points for v^0 that actually are concave). Exploiting monotonicity and concavity (when possible), Gordon and Qiu (2018) showed that you can do the search more efficiently by reordering the states. That is, follow this algorithm:

1. Find $k'\left(k_1\right)$ and $k'\left(k_n\right)$; set $\underline{i}=1$ and $\overline{i}=n$.
2. Find the optimal choice for $i\in\left\{\underline{i},...,\overline{i}\right\}$ by
   (a) If $\overline{i}=\underline{i}+1$, stop;
   (b) For $m=\left\lfloor\frac{\underline{i}+\overline{i}}{2}\right\rfloor$ compute $k'\left(k_m\right)$ by searching over $\left\{k_{\underline{i}},...,k_{\overline{i}}\right\}$ ($\lfloor x\rfloor$ is the floor function, equal to the largest integer smaller than x);
   (c) Goto step 2 twice, computing $k'\left(k_i\right)$ for $i\in\left\{\underline{i},...,m\right\}$ and $i\in\left\{m,...,\overline{i}\right\}$.

To give a sense of how much this approach improves performance, consider $n=20$; the standard monotonicity improvement leads to an average search size (computed over i) of 10.6 points for k', and if $n=100$ this average search size is 51.8, which is roughly a 50 percent improvement over not exploiting monotonicity at all. Gordon and Qiu's method yields average search sizes of 7.0 and 9.5, which are considerably smaller (particularly for the larger value of n).

A fourth acceleration tool is called Howard's improvement algorithm. Assume we have computed the entire set of optimal choices for our current guess v^n:

$$k'(k) = \left\{k_{j^*(1)}, ..., k_{j^*(n)}\right\}.$$

Normally one would update the value function once and then recompute the optimal choices using the updated value function. Howard's improvement algorithm says to update many times without recomputing the decision rules. That is, do the following update:

$$\widetilde{v}^0(k_i) = u\left(f(k_i) + (1-\delta)k_i - k_{j^*(i)}\right) + \beta v^n\left(k_{j^*(i)}\right)$$
$$\widetilde{v}^{n+1}(k_i) = u\left(f(k_i) + (1-\delta)k_i - k_{j^*(i)}\right) + \beta \widetilde{v}^n\left(k_{j^*(i)}\right)$$

and so on. This procedure is equivalent to using Newton's method on the Bellman equation and therefore has quadratic convergence near the solution, but may diverge (if the state space is discrete, as it is here, then it converges globally at a quadratic rate); call the converged value $\widetilde{v}^*(k_i)$. Then set

$$v^{n+1}(k_i) = \widetilde{v}^*(k_i).$$

We save ourselves the task of computing decision rules that are ultimately simply discarded, since they depend on the wrong value functions. Each value iteration now takes more time but we do about two orders of magnitude fewer iterations, so total computing time declines a lot. Note that we can also exploit the fact that the j^*-greedy value function $v_{j^*}(k)$ (defined as the value function that satisfies the Bellman recursion for policy function $j^*(k)$) satisfies the linear equation

$$v_{j^*}(k_i) = u_{j^*}\left(f(k_i) + (1-\delta)k_i - k_{j^*(i)}\right) + \beta v_{j^*}\left(k_{j^*(i)}\right)$$

which can easily be solved using matrix algebra. For high numbers of grid points in the state space, the iterative approach is better.

Fifth, we can implement a Gauss-Seidel sweep through the state space (pure value iteration solves the system of equations using Gauss-Jacobi sweeping). In this procedure, we use the updated value function $v^{n+1}(k_i)$ as the continuation value for the values of i that we have already solved; that is, instead of

$$v^{n+1}(k_i) = \max_{k' \in \{k_1, ..., k_n\}} \left\{u\left(f(k_i) + (1-\delta)k_i - k'\right) + \beta v^n\left(k'\right)\right\}$$

we have

$$v^{n+1}(k_i) = \max_{k' \in \{k_1,\ldots,k_n\}} \left\{ u\left(f(k_i) + (1-\delta)k_i - k'\right) + \beta \mathbf{1}\left(k' \leq k_i\right) v^{n+1}\left(k'\right) + \beta \mathbf{1}\left(k' > k_i\right) v^{n}\left(k'\right) \right\}$$

where $\mathbf{1}(A)$ is the indicator function for set A. It can be shown that both Howard's improvement and Gauss-Seidel sweeping generate contraction mappings that converge faster than pure value iteration.

Sixth, we can also exploit the McQueen-Porteus error bounds to speed up convergence. At any iteration n, one can easily show that

$$v^{n+1}(k) + \underline{b}_n \leq v(k) \leq v^{n+1}(k) + \overline{b}_n$$

where the bounds satisfy

$$\underline{b}_n = \frac{\beta}{1-\beta} \min_k \left\{ v^{n+1}(k) - v^n(k) \right\}$$
$$\overline{b}_n = \frac{\beta}{1-\beta} \max_k \left\{ v^{n+1}(k) - v^n(k) \right\}.$$

Because $\left\| v^{n+1} - v^n \right\|_\infty \to 0$ uniformly, the bounds get tighter as $n \to \infty$. We can exploit these bounds by updating to

$$v^{n+1}(k) = v^{n+1}(k) + \frac{\underline{b}_n + \overline{b}_n}{2}.$$

Bertsekas (1987) shows that this procedure converges faster than standard value function iteration. In a stochastic model, standard VFI converges at a rate proportional to the dominant eigenvalue of the state transition equation, which is always 1, so multiplying by β gives us a rate of convergence β. MQP iterations converge at a rate proportional to the subdominant eigenvalue, which will typically be related to the persistence of the shock; if the shock is not persistent, then the MQP bounds will greatly accelerate convergence, and they are essentially free. Note that the b terms are not norms, they are actual differences and can be negative.

Note that these acceleration methods can be combined – to solve the maximization problem we can use the Gordon-Qiu speedup combined with Gauss-Seidel sweeping, then explot the MacQueen-Porteus bounds and Howard's improvement during the updating step. These methods work best if we start from a function v^0 uniformly above or below the true function v; for many problems it is easy to come up with such functions.

For a stochastic problem

$$v\left(k_i, z_j\right) = \max_{k' \in \{k_1, ..., k_n\}} \left\{ u\left(\exp\left(z_j\right) f\left(k_i\right) + \left(1-\delta\right) k_i - k'\right) + \beta E_{z'}\left[v\left(k', z'\right)\right]\right\}$$

we face an additional cost in that we need to compute the expectation. We know how to do it already – turn it into a finite sum either by quadrature or by replacing the continuous z process with a Markov chain, which means we are solving a problem that takes the form

$$v\left(k_i, z_j\right) = \max_{k' \in \{k_1, ..., k_n\}} \left\{ u\left(\exp\left(z_j\right) f\left(k_i\right) + \left(1-\delta\right) k_i - k'\right) + \beta \sum_{l=1}^{n} \omega_l v\left(k', z_l\right)\right\}$$

where the weights and nodes differ depending on the choice we make. For the Markov chain case, we have

$$\omega_l = \pi\left(z' = z_l | z = z_j\right)$$

and the nodes z_l are just the states in the Markov chain. For the quadrature case it is a little different, because the integration points are not grid points for z_j; instead we use linear interpolation to obtain

$$v\left(k', z' = \rho z + \sqrt{2}\sigma x_l\right) = \left(1 - A\right) v\left(k', z_i\right) + A v\left(k', z_{i+1}\right)$$

$$A = \frac{\rho z + \sqrt{2}\sigma x_l - z_i}{z_{i+1} - z_i}$$

$$\rho z + \sqrt{2}\sigma x_l \in \left[z_i, z_{i+1}\right].$$

The weights are then a combination of the integration weights and the linear interpolation weights.

Note that we do not need that the "state space" $k \in \{k_1, ..., k_n\}$ and the choice space $k' \in \{k'_1, ..., k'_m\}$ are the same; we can interpolate the value function at different points, so we can use $m > n$; we can also have a different grid for each k_i, so that $m < n$ is also feasible. The "grid search" is simply the method for computing the maximum and should not be used as a synonym for "discrete state dynamic programming", although that is unfortunately quite common.

To evaluate the Euler equation error we need to be faithful to the discreteness in the decision rules, which makes evaluating the capital investment function tomorrow a little more difficult. Probably the most faithful way to do it is a "nearest neighbor" spline, which is a step function:

$$g\left(k\right) = g\left(k_i\right)$$

$$k_i = \underset{j \in \{1, ..., n\}}{\operatorname{argmin}} \left\{\left|k - k_j\right|\right\}.$$

Typically errors are quite large and they do not depend much on the grid size until it gets very dense, which leads to very long run times. An alternative is to compute a continuous approximation (see below) and compare the policy functions; of course, then one might naturally ask why bother computing a discrete version. For some problems, particularly nonconvex ones, the discrete version may be the only practical option. But if you can avoid it, you should; see below for some methods for converting nonconvex problems into convex ones.

13.1 Evolutionary Programming

Evolutionary programming solves the Bellman equation using a process similar to a genetic algorithm; it attempts to learn the value function through experimentation. Take the discrete dynamic program

$$V\left(k_i, z_j\right) = \max_{k'} \left\{ \log\left(\exp\left(z_j\right) k_i^{\alpha} + \left(1-\delta\right) k_i - k'\right) + \beta \sum_{k=1}^{n} \omega_k V\left(k', \rho z_j + \sqrt{2}\sigma x_k\right) \right\}.$$

Make a guess at the RHS V^0 as usual.

For each $n \in \{1, 2, ..., m\}$ guess a set of solutions

$$k' = K_n\left(k_i, z_j\right).$$

Then compute

$$V_n\left(k_i, z_j\right) = \log\left(\exp\left(z_j\right) k_i^{\alpha} + \left(1-\delta\right) k_i - K_n\left(k_i, k_j\right)\right) + \beta \sum_{k=1}^{n} \omega_k V^0\left(K_n\left(k_i, k_j\right), \rho z_j + \sqrt{2}\sigma x_k\right)$$

and

$$\overline{V}_n = \frac{1}{N_k N_z} \sum_{k_i} \sum_{z_j} V_n\left(k_i, z_j\right).$$

Now sort these values in decreasing order:

$$\overline{V}_1 > \overline{V}_2 > ... > \overline{V}_m.$$

Elements $n \in \left\{\frac{m}{2}, ..., m\right\}$ will be replaced:

$$K_n\left(k_i, z_j\right) = k^p$$

$$p = \max\left\{\min\left\{q + INT\left(x\right), N_k\right\}, 1\right\}.$$

q is the index of the capital grid point associated with $K_{n-\frac{m}{2}}(k_i, z_j)$, INT takes the integer part of a real number, and x is a standard normal random variable with variance σ^2. Then

$$V^1(k_i, z_j) = \max_{n \in \{1,\ldots,m\}} \{V_n(k_i, z_j)\}$$

and replace $K_{m/2}(k_i, z_j)$ with $K_{n^*}(k_i, z_j)$. Then reduce σ and repeat. As σ goes to zero, the value function will converge. Gomme (1994) shows that there is a significant increase in speed compared to standard dynamic programming.

13.2 Forward Dynamic Programming

Related to evolutionary programming is forward dynamic programming, which is used to solve problems like the blood inventory problem in which the number of states can be quite large (there are eight different types of blood, each of which has an expiration date of roughly three weeks after collection, blood is often set aside for particular patients in advance and also frequently not ultimately needed by those patients, there are random factors that drive demand for blood of each type and the supply of donors of each type, and a given blood bank usually exists on a network of other blood banks that can trade with each other; you can see how that might become rather unmanageable). We'll do something simpler, the growth model, to illustrate how it works (following Hull 2015). The Bellman equation is

$$V(k, z) = \max_{k'} \left\{ u\left(f(k, z) - k'\right) + \beta \sum \pi\left(z'|z\right) V\left(k', z'\right) \right\}.$$

First we guess an initial matrix $V^0(k, z)$ and select an initial state (k_0, z_0). Now solve

$$k_0' = \underset{k' \in (0, f(k_0, z_0))}{\operatorname{argmax}} \left\{ u\left(f(k_0, z_0) - k'\right) + \beta \sum \pi\left(z'|z_0\right) V^0\left(k', z'\right) \right\}$$

and compute

$$v(k_0, z_0) = u\left(f(k_0, z_0) - k_0'\right) + \beta \sum \pi\left(z'|z_0\right) V^0\left(k_0', z'\right)$$

to update the value function only in state (k_0, z_0):

$$V^1(k, z) = \begin{cases} V^0(k, z) & \text{if } (k, z) \neq (k_0, z_0) \\ (1-\alpha) V^0(k_0, z_0) + \alpha v(k_0, z_0) & \text{if } (k, z) = (k_0, z_0) \end{cases}.$$

Then draw z_1 using $\pi(z_1|z_0)$ and set $k_1 = k_0'$.

$$V(k, z_{-1}) = \sum_z \pi(z|z_{-1}) \max_{k'} \left\{ u\left(f(k, z) - k'\right) + \beta V\left(k', z\right) \right\}.$$

Repeat as needed. The method works better if we reduce the updating parameter α over time, in order to converge more quickly as we learn more about the function; one possible scheme is presented below.

The full algorithm is here:

1. Guess initial value function $V^0(k,z)$;

2. Choose simulation length T and cutoff $\overline{T} < T$, and generate a shock realization sequence $\{z_t\}_{t=1}^{T}$;

3. In period $t \in \{1,...,T\}$ solve

$$\widehat{k}'(k_t,z_t) = \underset{k'}{\operatorname{argmax}} \left\{ u\left(f(k_t,z_t) - k'\right) + \beta V^0\left(k',z_t\right)\right\}$$

and set

$$\widetilde{V}^0(k_t,z_t) = u\left(f(k_t,z_t) - \widehat{k}'(k_t,z_t)\right) + \beta V^0\left(\widehat{k}'(k_t,z_t),z_t\right);$$

4. Compute recursively from period T

$$\widehat{V}(k_t,z_{t-1}) = u\left(f(k_t,z_t) - \widehat{k}'(k_t,z_t)\right) + \beta \widetilde{V}^0\left(\widehat{k}'(k_t,z_t),z_t\right);$$

5. Set for $t \in \left\{1,...,\overline{T}\right\}$

$$\widetilde{V}^1(k_t,z_{t-1}) = \widehat{V}(k_t,z_{t-1});$$

6. Update by setting

$$V^1(k_t,z_{t-1}) = (1-\alpha_0(k_t,z_{t-1}))V^0(k_t,z_{t-1}) + \alpha_0(k_t,z_{t-1})\widetilde{V}^1(k_t,z_{t-1});$$

7. Update the state using $\widehat{k}'(k_t,z_t)$;

8. If

$$\left\|V^1 - V^0\right\|_\infty > \epsilon,$$

return to Step 3 and repeat using

$$\alpha_n(k_t,z_{t-1}) = \alpha_0\left(\frac{1+\frac{b}{\alpha_0}\frac{Q_{i,j}}{\tau}}{1+\frac{b}{\alpha_0}\frac{Q_{i,j}}{\tau}+\frac{Q_{i,j}^2}{\tau}}\right)$$

at iteration n, where Q_{ij} is the frequency of state (k_i, z_j) and (α_0, b, τ) are parameters; this formula is a modification of Darken and Moody's (1992) 'Search-then-Converge' algorithm that updates more aggressively states that are visited more frequently.

In effect, we "learn" the value function by passing through the states many times and locally improving it each time; if we transit through all the states sufficiently many times, we eventually get to the true value function. For large problems it is common to approximate the value function using the sum of linear functions in each state (which is not a piecewise linear spline, which would include cross-terms, but rather a linear combination of lines) in order to keep the maximization step feasible.

Note the difference between this approach and standard dynamic programming; rather than working backward from a terminal point (that is possibly infinitely-far away), we are working forward and exploiting ergodicity; this approach will not work with a finite horizon particularly well (you would need to treat calender time explicitly as another state, which would substantially increase the size of the required simulation).

The key to forward dynamic programming is to write the Bellman equation in terms of "post-decision states". To be specific, consider the dynamic program

$$V(s_t, x_t) = \max_{c_t} \left\{ u(c_t) + \beta E_{x_{t+1}} \left[V(s_{t+1}, x_{t+1}) | s_t, x_t \right] \right\},$$

where s_t is an endogenous state and x_t is an exogenous state (both could be vectors of course), with the constraints

$$\begin{aligned} s_{t+1} &= T(c_t, s_t, x_t) \\ c_t &\in \Gamma(s_t, x_t) \\ x_{t+1} &= f(x_t, \varepsilon_{t+1}). \end{aligned}$$

It isn't hard to see that we could rewrite this equation as

$$v(s_t, x_{t-1}) = E_{x_t} \left[\max_{c_t} \{ u(c_t) + \beta v(s_{t+1}, x_t) \} \middle| s_t, x_{t-1} \right],$$

where c_t is contingent on the realized x_t. Approximate Dynamic Programming using Post-Decision States (ADP-POST) has the following steps:

1. Initialize $v^0(s,x)$ and state (s_0,x_0);
2. Choose a simulation length T and generate a sample $\{x_t\}_{t=1}^T$;
3. For each period $t \in \{1,...,T\}$:
   (a) Choose c_t to maximize the expression inside the expectation for the realized x_t; denote this maximized value by
   $$\widetilde{v}^1(s_t,x_{t-1}) = u(c_t^*) + \beta v^0(T(c_t^*,s_t,x_t),x_t);$$
   (b) Compute expectation:
   $$v^1(s_t,x_{t-1}) = (1-\alpha_0)v^0(s_t,x_{t-1}) + \alpha_0\widetilde{v}^1(s_t,x_{t-1});$$
   (c) Update state to $(T(c_t^*,s_t,x_t),x_t)$ and repeat;
4. Reset $v^0 = v^1$ and repeat.

14 Continuous State Dynamic Programming

Finite-state methods are slow because the optimization problem is not continuous and the constraint set is not convex, so the problem is not a concave program, and approximation errors are large unless the grid is dense. We know that concave programming problems are much easier to solve and the algorithms are fast, so how can we get one? By using functional approximation tools to represent the value function – which we know is continuous under very mild conditions – as a continuous function (because shape preservation is important, we will use splines here). Let's start with a grid for k and a guess $v^0(k)$ for the value function at those points. Then, at each k, we need to solve the constrained maximization problem

$$v^1(k) = \max_{k',c}\left\{u(c) + \beta v^0\left(k'\right)\right\}$$

subject to

$$c \geq 0$$
$$k' \geq 0$$
$$c + k' \leq f(k) + (1-\delta)k.$$

We know v^0 at some points (those in the grid for k) but not at others; however, if we construct a piecewise cubic Hermite polynomial through our points $\left(k, v^0\right)$ we can evaluate it at any arbitrary capital stock k' we select. Now the problem is continuous, and if our guess for v^0 is "increasing and concave" (meaning it is a sequence of points whose values increase in k at a decreasing rate) our problem is also concave. We can then use an SQP solver to maximize the RHS at each k, which delivers a new vector of values $\left(k, v^1\right)$. If $\left|\left|v^1 - v^0\right|\right|_\infty < \epsilon$ we are done, otherwise replace v^0 with v^1 and repeat. It is helpful to retain the policy function

$$k' = g^0\left(k\right)$$

to generate the initial guesses for the solver at each new step; as the value function converges the guesses will be more and more accurate and computation will actually speed up.

We can still use Howard's improvement to speed up convergence, provided we are not using the envelope condition to obtain the derivative of the value function. In fact, we can use Howard's improvement exclusively once the policy function has converged; as noted already, the constant in the value function is the slow part, so the policy functions will typically converge much faster and, at that point, there is no need to continue the optimization. However, for some problems you may not want to do this step to convergence, but instead some prespecified number of times, because as noted the procedure is equivalent to Newton's method and therefore may diverge. Instead, use Howard's improvement (i) only once the value function has "settled down" a bit and (ii) only for a fixed number of iterations (Howard's improvement is only guaranteed to converge if the iterations are monotone, so that our guess is either everywhere above or everywhere below the fixed point). We can also still use the MacQueen-Porteus error bounds; use them before Howard's improvement.

With stochastic elements, we can approximate the conditional expectation function to speed up the program. Consider the Bellman equation

$$v^1\left(k, z\right) = \max_{k', c} \left\{u\left(c\right) + \beta E\left[v^0\left(k', z'\right)|z\right]\right\}$$

subject to

$$c \geq 0$$

$$k' \geq 0$$

$$c + k' \leq \exp(z) f(k) + (1 - \delta) k$$

along with some process for z. We can change this problem into

$$v^1(k, z) = \max_{k', c} \left\{ u(c) + \beta \widehat{v}\left(k', z\right) \right\}$$

where

$$\widehat{v}\left(k', z\right) = E\left[v^0\left(k', z'\right) | z \right].$$

If we put a spline through $\widehat{v}$ instead of v we can save a lot of calculations. Note however that this trick does not work if the state evolves "stochastically", meaning that the value of the state tomorrow differs with z'. For example, we could have written the problem as

$$v^1(y, z) = \max_{k', c} \left\{ u(c) + \beta E\left[v^0\left(\exp\left(z'\right) \left(k'\right)^{\alpha} + (1 - \delta) k' \right) \right] \right\}$$

subject to

$$c \geq 0$$

$$k' \geq 0$$

$$c + k' \leq y.$$

Now the value of total resources y' changes with z', so we cannot compute the expectation in advance. There are circumstances in which this change is helpful, such as portfolio problems where total wealth may be sufficient. For example, consider this dynamic portfolio problem between a risky and a riskless asset:

$$v(s, b, r) = \max_{s, b, c} \left\{ u(c) + \beta E\left[v\left(s', b', r'\right) \right] \right\}$$

subject to

$$c \geq 0$$

$$s \geq \underline{s}$$

$$b \geq \underline{b}$$

$$c + s + b \leq rs + Rb.$$

Rewriting the problem using total wealth

$$a = rs + Rb$$

yields

$$v(a, r) = \max_{s,b,c} \left\{ u(c) + \beta E\left[v\left(a', r'\right)\right]\right\}$$

subject to

$$\begin{aligned} c &\geq 0 \\ s &\geq \underline{s} \\ b &\geq \underline{b} \\ c + s + b &\leq a \\ a' &\leq r's + Rb. \end{aligned}$$

One state variable is better than two because we can more easily preserve shape, which makes the portfolio problem a lot easier.

Note that it is straighforward to add constraints to this problem. For example, suppose we want a nonnegativity of gross investment constraint

$$k' \geq (1 - \delta) k.$$

We just add this constraint into the problem. Note that two-dimensional problems can be difficult due to the problems associated with preserving shape, particularly if there are occasionally-binding constraints.

If the problem is continuous but not concave, we will need to use global optimization methods to find a solution. For example, suppose that the planner can only adjust the capital stock by paying a fixed cost:

$$v^1(k) = \max\left\{ \max_{k'} \left\{ u\left(f(k) + (1-\delta)k - k' - \eta\right) + \beta v^0\left(k'\right)\right\}, u\left(f(k) - \delta k\right) + \beta v^0(k)\right\}.$$

In general, even if v^0 is concave v^1 will not be, unless the planner either always or never adjusts. Note that preserving shape becomes really important with nonconvexities – around the kinks cubic splines will tend to wiggle a lot, so they should be avoided. Note that we can rewrite this

problem as

$$v^1(k) = \max_{d\in[0,1],k'} \left\{ d\left[u\left(f(k) + (1-\delta)k - k' - \eta\right) + \beta v^0\left(k'\right)\right] + (1-d)\left[u\left(f(k) - \delta k\right) + \beta v^0(k)\right]\right\}$$

which now has a convex constraint set, but v is still not concave because the decision rule is not continuous (it is only upper hemicontinuous). That really does not help us much, as we may encounter local maxima; it is actually common that the local minima are very similar in "height", implying that small errors could lead us to make the large mistakes in the optimal actions.

The dynamic program has structure that we can exploit. For example, if you know that the adjustment/no adjustment regions are intervals, then they are defined by a unique set of thresholds that make the agent indifferent:

$$\begin{aligned} v_a(k_i) &= v_{na}(k_i) \\ v_a(k_i) &= u\left(f(k_i) + (1-\delta)k_i - g(k_i) - \eta\right) + \beta v\left(g(k_i)\right) \\ v_{na}(k_i) &= u\left(f(k_i) - \delta k_i\right) + \beta v(k_i). \end{aligned}$$

For the fixed cost of adjustment problem, there are generically three such thresholds. At low enough k, the agent cannot afford to adjust ($\eta > f(k) + (1-\delta)k$), implying that no adjustment is optimal. At the steady state, obviously no adjustment is also optimal; by continuity, near enough the steady state the same holds. Finally, far enough from the steady state (in either direction) the gain from adjustment is sufficiently large, due to the Inada conditions on consumption and the fact that

$$c \to 0$$

as k approaches values that satisfy

$$\begin{aligned} \eta &= f(k) + (1-\delta)k \\ f(k) &= \delta k; \end{aligned}$$

also, for any k above the level that satisfies the second equation, no adjustment is not feasible. Thus, there exist three thresholds, $0 < k_1 < k_2 < k^* < k_3$, such that

$$\begin{aligned} \text{adjust if } k &\in [k_1, k_2] \cup [k_3, \infty) \\ \text{no adjust if } k &\in (0, k_1] \cup [k_2, k_3]. \end{aligned}$$

While we still need to compute the choice of k' given adjustment using global methods, the computation of the thresholds are just nonlinear equations in k whose solutions lie between known points.

There are two ways we can "convexify" the problem. First, suppose we assume the fixed cost is additive to utility:

$$v^1(k) = \max\left\{\max_{k'}\left\{u\left(f(k)+(1-\delta)k-k'\right)-\eta+\beta v^0\left(k'\right)\right\}, u\left(f(k)-\delta k\right)+\beta v^0(k)\right\}.$$

If we assume that η is random and distributed as a Type-I extreme value (Gumbel) with parameter σ and the choice of k' is made after η is known, then this problem can be solved in nearly closed-form. The probability of adjusting k is

$$p(k) = \frac{1}{1+\exp\left(\sigma\left(v_{na}(k)-v_a(k)\right)\right)} \in (0,1)$$

where

$$\begin{aligned} v_a(k) &= \max_{k'}\left\{u\left(f(k)+(1-\delta)k-k'\right)+\beta v^0\left(k'\right)\right\} \\ v_{na}(k) &= u\left(f(k)-\delta k\right)+\beta v^0(k) \end{aligned}$$

and

$$v^1(k) = \frac{\gamma_E}{\sigma} + \frac{1}{\sigma}\left(\exp\left(\sigma v_a(k)\right)+\exp\left(\sigma v_{na}(k)\right)\right)$$

is the updated value function. This problem is concave. The p function implies that adjustment is more likely if the gap between adjusting and not adjusting is larger, but it always occurs with positive probability. As $\sigma \to \infty$, the problem converges to the discrete choice case, and as $\sigma \to 0$ we get a 50-50 value for $p(k)$ independent of k. Note: we could have left η as a resource cost and introduced a preference shock as well; η would simply reduce the chance of adjusting.

Another alternative is to convexify the problem using a lottery. Suppose households with wealth k can form a lottery – winners receive $k_2 > k$, losers receive $k_1 < k$, and the lottery is resource neutral (the winners are compensated by the losers). Since all such households are identical, they would pick the same winning probability π; by the law of large numbers, this probability must satisfy

$$\pi k_2 + (1-\pi)k_1 = k.$$

An interior solution to the equilibrium lottery satisfies three equalities

$$Dv_1(k_1) = Dv_2(k_2),$$

$$v_2(k_2) = v_1(k_1) + Dv_1(k_1)(k_2 - k_1),$$

and

$$\pi = \frac{k - k_1}{k_2 - k_1};$$

the points (k_1, k_2) define a linear segment that smoothly connects $v_1(k_1)$ and $v_2(k_2)$ (these equations are just the Kuhn-Tucker conditions with the multipliers set to zero). The resulting value function is

$$v(k) = \begin{cases} v_1(k) & \text{if } k \leq k_1 \\ v_1(k_1) + Dv_1(k_1)(k - k_1) & \text{if } k \in (k_1, k_2) \\ v_2(k) & \text{if } k \geq k_2 \end{cases},$$

which is concave (but not strictly concave). If $p \in \{0, 1\}$, then the derivative conditions do not hold, but the solution is obviously trivial:

$$p = 0$$

$$k_2 = k$$

or

$$p = 1$$

$$k_1 = k$$

with the other variable undetermined. A sufficient condition for a corner is that $v_1 > v_2$ (or $v_1 < v_2$) at all capital stocks.

This problem is very difficult to solve, although quite easy to visualize – we are looking for the line with the smallest intercept and lowest slope that lies above two concave functions. That is, the function is an upper support function for both v_1 and v_2. Unfortunately, this problem does not satisfy constraint qualification – as noted already, if $p = 0$, then the choice of k_1 is undetermined, and similarly $p = 1$ leaves k_2 undetermined.

A trick that can help with the computational cost is "straightening out" the value function. Suppose the utility function is isoelastic:

$$u(c) = \frac{c^{1-\sigma}}{1-\sigma}.$$

Preferences are represented by the function

$$U\left(\{c_t\}_{t=0}^{\infty}\right) = \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma},$$

which leads to the usual Bellman equation

$$\begin{aligned} V(k) &= \max_{k',c} \left\{ \frac{c^{1-\sigma}}{1-\sigma} + \beta V\left(k'\right) \right\} \\ c + k' &\leq k^{\alpha} + (1-\delta)\, k. \end{aligned}$$

But preferences are also represented by

$$\mathcal{U}\left(\{c_t\}_{t=0}^{\infty}\right) = \left(\sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\sigma}}{1-\sigma} \right)^{\frac{1}{1-\sigma}},$$

which leads to the ordinally equivalent

$$\begin{aligned} \mathcal{V}(k) &= \max_{k',c} \left\{ \frac{c^{1-\sigma}}{1-\sigma} + \beta \mathcal{V}\left(k'\right)^{1-\sigma} \right\}^{\frac{1}{1-\sigma}} \\ c + k' &\leq k^{\alpha} + (1-\delta)\, k. \end{aligned}$$

This representation, which is a special case of preferences we will see later called Epstein-Zin preferences, is "more linear" in k and therefore easier to interpolate. Furthermore, it ranks all consumption paths equivalently to the standard formulation:

$$U\left(\{c_t\}_{t=0}^{\infty}\right) \geq U\left(\{\widehat{c}_t\}_{t=0}^{\infty}\right) \text{ iff } \mathcal{U}\left(\{c_t\}_{t=0}^{\infty}\right) \geq \mathcal{U}\left(\{\widehat{c}_t\}_{t=0}^{\infty}\right).$$

In stochastic settings, this approach still works:

$$\begin{aligned} \mathcal{V}(k,z) &= \max_{k',c} \left\{ \frac{c^{1-\sigma}}{1-\sigma} + \beta \sum \pi\left(z'|z\right) \mathcal{V}\left(k',z'\right)^{1-\sigma} \right\}^{\frac{1}{1-\sigma}} \\ c + k' &\leq \exp(z)\, k^{\alpha} + (1-\delta)\, k \end{aligned}$$

is ordinally equivalent to

$$V(k,z) = \max_{k',c} \left\{ \frac{c^{1-\sigma}}{1-\sigma} + \beta \sum \pi\left(z'|z\right) V\left(k',z'\right) \right\}.$$

The reason these changes "work" is that $\mathcal{U}$ is just a monotone transformation of U, and therefore represents the same preferences. But that does not make them equivalent for numerical work. Be aware that for many values of σ the function $\mathcal{V}(k')^{1-\sigma}$ will not permit a non-positive argument, so we have to replace it with $-(-\mathcal{V}(k'))^{1-\sigma}$.

Another trick that is useful for environments like Matlab is to eliminate the looping over the current state. Note that solving

$$v^{n+1}(k) = \max_{k'} \left\{ \log\left(k^{\alpha} + (1-\delta)k - k'\right) + \beta v^{n}\left(k'\right) \right\}$$

for each $k \in \{k_1, ..., k_m\}$ generates the same decision rule as solving

$$V = \max_{\{k'_1,...,k'_m\}} \left\{ \sum_{i=1}^{m} \left(\log\left(k_i^{\alpha} + (1-\delta)k_i - k'_i\right) + \beta v^{n}\left(k'_i\right) \right) \right\}.$$

We can then obtain the updated value function by

$$v^{n+1}(k_i) = \log\left(k_i^{\alpha} + (1-\delta)k_i - g(k_i)\right) + \beta v^{n}\left(g(k_i)\right).$$

The gradient and Hessian are easy to derive and can be supplied to your solver:

$$\nabla V = \begin{bmatrix} -\frac{1}{k_1^{\alpha} + (1-\delta)k_1 - k'_1} + \beta D v^{n}(k'_1) \\ \vdots \\ -\frac{1}{k_m^{\alpha} + (1-\delta)k_m - k'_m} + \beta D v^{n}(k'_m) \end{bmatrix}$$

$$\nabla^2 V = diag\left(\begin{bmatrix} -\left(\frac{1}{k_1^{\alpha} + (1-\delta)k_1 - k'_1}\right)^2 + \beta D^2 v^{n}(k'_1) \\ \vdots \\ -\left(\frac{1}{k_m^{\alpha} + (1-\delta)k_m - k'_m}\right)^2 + \beta D^2 v^{n}(k'_m) \end{bmatrix} \right).$$

The curse of dimensionality is not that severe, even for large m, because the problems are concave and "disconnected" (which is why the Hessian is diagonal).

14.1 Preserving Contractions on the Computer

We now explore the extent to which the continuous state approximations preserve contraction mappings, a necessary part of the proof of existence and convergence to the value function. As an example, we will show that the linear interpolation case satisfies the conditions for contractiveness and also provide an error bound on the distance between the computed object and the true unknown value function. This discussion comes from Stachurski (2008).

Define two operators:

$$Tw(x) = \sup_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \int w(x') Q(x,a;x') dx' \right\}$$

and

$$T_\pi w(x) = r(x, \pi(x)) + \beta \int w(x') Q(x, \pi(x); x') dx'.$$

We already know that these operators are contractions under mild conditions – boundedness of r and $\beta < 1$ are sufficient, along with continuity of (r, Q, Γ), as we saw in the introductory chapters. If $\pi(x)$ is the optimal policy function from the first operator

$$\pi(x) = \arg\max_{a \in \Gamma(x)} \left\{ r(x,a) + \beta \int w(x') Q(x,a;x') dx' \right\},$$

then both contraction mappings converge to the same fixed point.

We now define a mapping L called an approximation operator; that is, L takes functions w and maps them into some family F, such as those defined on a finite number of points or parameterized by splines. We only consider L maps that satisfy the projection property:

$$L \circ L = L.$$

Thus, if w is itself an element of the family F then $Lw = w$; two examples of the projection property are using linear regression to compute a Chebyshev approximation to w and using a spline to interpolate w. The operator L is further restricted to be **nonexpansive**:

$$||Lv - Lw||_\infty \leq ||v - w||_\infty$$

for any functions w and v; note that nonexpansiveness is weaker than contractiveness since it permits $\beta = 1$, so that it only requires that points not get further apart. One example of a nonexpansive operator is the nearest neighbor spline:

$$Lv(x) = v(x_i)$$
$$i = \operatorname*{argmin}_j ||x - x_j||;$$

as we noted already, a nearest neighbor spline defines a step function. It is clear that

$$||Lw - Lv||_\infty \leq \sup_{1 \leq i \leq k} \{|w(x_i) - v(x_i)|\}$$

so that L is nonexpansive. Iterating on $\widehat{T} = L \circ T$ produces a form of discrete value iteration, where $\widehat{v}^n = \widehat{T}^n v$ satisfies

$$\begin{aligned} T\widehat{v}^n(x_j) &= \sup_{a \in \Gamma(x_j)} \left\{ r(x_j, a) + \beta \int \widehat{v}^n(x') Q(x_j, a; x') dx' \right\} \\ &= \sup_{a \in \Gamma(x_j)} \left\{ r(x_j, a) + \beta \sum_{i=1}^{k} \widehat{v}^n(x_i) Q(x_j, a; B_i) \right\} \end{aligned}$$

where B_i is the subset of the space where $\widehat{v}_n = \widehat{v}_n(x_i)$. Then define

$$LT\widehat{v}_n = \widehat{T}^{n+1} v$$

and proceed to the next step. $\widehat{T}$ is a contraction mapping and will satisfy the error bounds given below.

Piecewise linear interpolation also defines a nonexpansive approximation operator and therefore satisfies the contraction mapping properties. Let $\{x_i\}_{i=1}^k$ be a finite subset of the state space with the property that its convexification is the state space itself (this construction is feasible if the space is compact). Define the set of vertices

$$\{\zeta_i\}_{i=1}^{d+1} \subset \{x_i\}_{i=1}^k$$

for each possible simplex Δ in a given triangularization of S, the state space. Any point $x \in \Delta$ is then representable using the barycentric coordinates

$$x = \sum_{i=1}^{d+1} \lambda(x, i) \zeta_i$$

where

$$\begin{aligned} \lambda(x, i) &\geq 0 \\ \sum_{i=1}^{d+1} \lambda(x, i) &= 1 \end{aligned}$$

are the weights on the vertices. Define

$$Lv(x) = \sum_{i=1}^{d+1} \lambda(x, i) v(\zeta_i),$$

the piecewise linear interpolation operator over simplices. It is straightforward to show that

$$|Lw(x) - Lv(x)| = \left| \sum_{i=1}^{d+1} \lambda(x, i) (w(\zeta_i) - v(\zeta_i)) \right| \leq \sup_{1 \leq i \leq d+1} \{|w(\zeta_i) - v(\zeta_i)|\} \leq ||w - v||_\infty .$$

Thus L is nonexpansive. The key in this proof is that the weights $\lambda(x,i)$ are all nonnegative.

One can also show that any shape-preserving spline operator is nonexpansive, so all the error bounds presented here will apply to those cases as well; as noted already, the key property is that the weighting scheme must produce only nonnegative weights. Cubic splines may not be nonexpansive, however, since they could involve negative weights which then would invalidate the second step of the proof (a cubic function is monotone increasing if and only if it has only positive coefficients, and the smoothness requirements may not permit that for a given set of nodes). The same problem could arise with global polynomial approximations. As a result, in some cases Bellman iterations could be explosive (some examples are given in Pál and Stachurski 2013 with Chebyshev polynomials).

The error bound we are after is of the form

$$\varepsilon(\widehat{\pi}) = v^*(x) - v_{\widehat{\pi}}(x)$$

where $\widehat{\pi}$ is the approximate policy function and $v_{\widehat{\pi}}$ is the fixed point of the operator defined as using the approximate policy function forever. This error has two components. The first is the deviation of $\widehat{v}$ from v^*, the result of imperfect approximation by L. The second is the deviation of $\widehat{T}^n v$ from $\widehat{v}$, the result of truncating convergence. The error bound is

$$v^*(x) - v_\pi(x) \leq \frac{2}{(1-\beta)^2}\left(\beta\epsilon + \|v^* - Lv^*\|_\infty\right)$$

where ϵ is the tolerance for convergence in the iterations. For many types of approximation schemes we have results on the term $||v^* - Lv^*||_\infty$ so that we can evaluate the RHS for given L. Standard contraction mapping arguments establish that

$$\|v^* - v_\pi\| \leq \left\|v^* - \widehat{T}^{n+1}v\right\|_\infty + \frac{1+\beta}{1-\beta}\left\|\widehat{T}^{n+1}v - v^*\right\|_\infty$$

which implies

$$(1-\beta)\|v^* - v_\pi\| \leq 2\left\|\widehat{T}^{n+1}v - v^*\right\|_\infty.$$

To obtain the error bound, assume that the convergence occurs in $n+1$ steps, so that

$$v^*(x) - v_\pi(x) \leq \frac{2}{1-\beta}\left\|\widehat{T}^{n+1}v - v^*\right\|_\infty.$$

The triangle inequality then gives us

$$v^*(x) - v_\pi(x) \leq \frac{2}{(1-\beta)^2}\left(\beta\left\|\widehat{T}^{n+1}v - \widehat{T}^n v\right\|_\infty + ||v^* - Lv^*||_\infty\right).$$

Since convergence requires

$$\left|\left|\widehat{T}^{n+1}v - \widehat{T}^{n}v\right|\right|_{\infty} \leq \epsilon$$

the error bound is established.

Thus, if we accurately compute the maximum, shape-preserving methods will retain the contraction property and ensure that we converge to the unique fixed point of $\widehat{T}$; we have shown that this fixed point is close to the fixed point of T.

15 Euler Equation Methods

Dynamic programming is slow – it converges linearly at rate $1-\beta$. It turns out that much of this slowness can be attributed to the "constant" that determines the level of the value function (it is easy to see this in the LQ regulator problem by watching the P matrix converge; the $(1,1)$ element corresponding to the constant converges much slower than the rest of the entries). If our problem is differentiable, it may be valuable to use the first-order conditions instead, where the constant in the value function disappears. To see how, let's take the FOC of the problem above:

$$\begin{aligned} Du\left(c\right) - \lambda &= 0 \\ \beta Dv^{0}\left(k'\right) - \lambda &= 0 \\ f\left(k\right) + \left(1-\delta\right)k - c - k' &= 0. \end{aligned}$$

We can use the envelope theorem to obtain

$$Dv\left(k\right) = Du\left(c\right)$$

and then put everything together into the Euler equation

$$Du\left(f\left(k\right) + \left(1-\delta\right)k - k'\right) - \beta Du\left(f\left(k'\right) + \left(1-\delta\right)k' - k''\right)\left(Df\left(k'\right) + 1 - \delta\right) = 0.$$

We are looking for a time invariant function $k' = g\left(k\right)$ that solves this equation at each k and has the property that $k'' = g\left(k'\right)$. Since the value function v does not appear, only its derivative, the constant term would not be part of the iterations (and as a result they converge faster).

Now guess $k'' = g^{0}\left(k'\right)$; this guess amounts to a forecast about what investment choice will be made tomorrow, conditional on whatever capital stock is delivered then. The Euler equation

takes that guess and identifies the optimal k':

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-k'\right)-\beta Du\left(f\left(k'\right)+\left(1-\delta\right)k'-g^{0}\left(k'\right)\right)\left(Df\left(k'\right)+1-\delta\right)=0.$$

Using a spline we can evaluate our guess g^0 at any k', so we can solve this equation for k' (provided we have the right shape – monotonically-increasing g^0 – and preserve it there will be a unique interior solution). Then we can check whether $k'=g^1\left(k\right)$ equals g^0, and if not we update and continue. As with dynamic programming, the search for the solution speeds up as we iterate if we use the previous policy function as an initial guess for the solver (that does not work with some bracket-based methods because they terminate only if the bracket is sufficiently small, but would work with safe Newton-Raphson since it terminates based on the function value or we can adapt the solver to also terminate if the function is sufficiently close to zero independent of the bracket size).

We can handle any occasionally-binding constraints using the trick of converting the KT conditions into equations. Note also that we can solve economies which are not Pareto efficient using this method; there is nothing special about the functional equations being the FONC of some planning problem. To do the non-negativity constraint case from above, define the multiplier to be

$$\mu=\max\left\{\lambda,0\right\}^{2}$$

so that we have the pair of functional equations

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-k'\right)-\max\left\{\lambda,0\right\}^{2}-\beta Du\left(f\left(k'\right)+\left(1-\delta\right)k'-g^{0}\left(k'\right)\right)\left(Df\left(k'\right)+1-\delta\right)+\beta\left(1-\delta\right)\max$$

$$\max\left\{-\lambda,0\right\}^{2}-\left(k$$

Alternatively use the Fischer-Baumeister function

$$Du\left(f\left(k\right)+\left(1-\delta\right)k-k'\right)-\mu-\beta Du\left(f\left(k'\right)+\left(1-\delta\right)k'-g^{0}\left(k'\right)\right)\left(Df\left(k'\right)+1-\delta\right)+\beta\left(1-\delta\right)h^{0}\left(k'\right)=0$$

$$\mu+\left(k'-\left(1-\delta\right)k\right)-\sqrt{\mu^{2}+\left(k'-\left(1-\delta\right)k\right)^{2}}=0.$$

With stochastic elements, we can again reduce the cost of computation by approximating the entire function under the integral with a spline. That is, guess $c'=g^0\left(k',z'\right)$ and then compute for each $\left(k',z\right)$ point

$$M\left(k',z\right)=\sum_{i=1}^{n}\omega_{i}Du\left(g^{0}\left(k',z_{i}\right)\right)\left(\exp\left(z_{i}\right)Df\left(k'\right)+1-\delta\right).$$

We can use a spline to approximate this entire function, which can generate significant cost savings.

Another trick is to "precompute" parts of the integral – if we are using piecewise polynomial functions, then the integral inolves the sum of moments of the innovation and can be done once; this method was developed by Judd, Maliar, and Maliar (2015). For example, consider the expectation of a quadratic in k' and z', where

$$z' = \rho z + \sigma\epsilon$$

with $\epsilon \sim N(0,1)$. We have

$$\begin{aligned} &E\left[a_0 + a_1 k' + a_2 z' + a_3 \left(k'\right)^2 + a_4 k' z' + a_5 \left(z'\right)^2 |z\right] \\ &= a_0 + a_1 k' + a_3 \left(k'\right)^2 + \left(a_2 + a_4 k'\right) E\left[z'|z\right] + a_5 E\left[\left(z'\right)^2 |z\right] \end{aligned}$$

where

$$\begin{aligned} E\left[z'|z\right] &= \rho z \\ E\left[\left(z'\right)^2 |z\right] &= E\left[\left(z' - E\left[z'|z\right]\right)^2\right] + \left(E\left[z'|z\right]\right)^2 \\ &= \sigma^2 + \rho^2 z^2. \end{aligned}$$

Therefore,

$$E\left[a_0 + a_1 k' + a_2 z' + a_3 \left(k'\right)^2 + a_4 k' z' + a_5 \left(z'\right)^2 |z\right] = a_0 + a_1 k' + a_3 \left(k'\right)^2 + \left(a_2 + a_4 k'\right)\rho z + a_5\left(\rho^2 z^2 + \sigma^2\right).$$

Higher-order moments are equally straightforward:

$$\begin{aligned} E\left[\left(z'\right)^3 |z\right] &= E\left[\left(z' - E\left[z'|z\right]\right)^3 |z\right] + 3E\left[\left(z'\right)^2 |z\right] E\left[z'|z\right] - 2\left(E\left[z'|z\right]\right)^3 \\ &= 0 + 3\left(\rho^2 z^2 + \sigma^2\right)\rho z - 2\rho^3 z^3 \\ &= \rho^3 z^3 + 3\sigma^2 \rho z \\ E\left[\left(z'\right)^4 |z\right] &= \rho^4 z^4 + 6\sigma^2 \rho^2 z^2 + 3\sigma^4 \end{aligned}$$

and so on. This approach will also work for alternative distributions for ϵ, provided only that it possesses a sufficient number of moments; in fact, we don't actually need to specify the distribution for this approach, only the moments.

For some models, the law of motion for the state cannot be solved explicitly for s'; for example, a trivial example is a quadratic adjustment cost for capital

$$k' = (1-\delta)\,k + i - \frac{\phi}{2}\left(\frac{k'-k}{k}\right)^2 k.$$

This case is trivial because there are other ways to write down the problem that alleviate the issue, but for some models that will not be possible. Cao, Luo, and Nie (2020) address this problem by adding extra equations that include the future endogenous states s' as part of the system to be solved. To take an explicit example, consider the risk-sharing problem of Heaton and Lucas (1996), where two households try to share idiosyncratic income risk by trading equities and bonds. The optimality conditions for each household are

$$\begin{aligned}
1 &= \beta E_t\left[\left(\frac{c^i_{t+1}}{c^i_t}\right)^{-\gamma}\left(y^a_{t+1}\right)^{1-\gamma}\frac{p_{t+1}+d_{t+1}}{p_t}\right] + \max\left\{\mu^{i,s}_t, 0\right\}^2 \\
1 &= \beta E_t\left[\left(\frac{c^i_{t+1}}{c^i_t}\right)^{-\gamma}\left(y^a_{t+1}\right)^{1-\gamma}\frac{1}{q_t}\right] + \max\left\{\mu^{i,b}_t, 0\right\}^2 \\
c^i_t + p_t s^i_{t+1} + q_t b^i_{t+1} &= (p_t + d_t)\,s^i_t + b^i_t \\
\max\left\{-\mu^{i,s}_t, 0\right\}^2 &= s^i_{t+1} \\
\max\left\{-\mu^{i,b}_t, 0\right\}^2 &= b^i_{t+1} - B
\end{aligned}$$

plus the market clearing conditions

$$\begin{aligned}
s^1_t + s^2_t &= 1 \\
b^1_t + b^2_t &= 1.
\end{aligned}$$

Rather than keep track of two state variables (such as the stock and bond holdings of one agent), we need only keep track of relative wealth of person 1

$$\omega^1_t = \frac{(p_t + d_t)\,s^1_t + b^1_t}{p_t + d_t}.$$

Unfortunately, the law of motion for relative wealth involves future endogenous variables, which are themselves functions of future relative wealth (Bacchetta, van Wincoop, and Young 2022 have the same problem, in much higher dimensions, but we circumvent it by assuming the policy functions are locally, but not globally, linear):

$$\omega^1_{t+1} = \frac{\left(p_{t+1}\left(z_{t+1}, \omega^1_{t+1}\right) + d_{t+1}\right)s^1_{t+1} + b^1_{t+1}}{p_{t+1}\left(z_{t+1}, \omega^1_{t+1}\right) + d_{t+1}}.$$

Suppose we use N points to compute the expectations (quadrature or discretization); then we can add N equations that determine the values for ω_{t+1}^1:

$$\omega_{t+1}^{1,n} = \frac{\left(p_{t+1}\left(z_{t+1}^n, \omega_{t+1}^1\right) + d_{t+1}^n\right) s_{t+1}^1 + b_{t+1}^1}{p_{t+1}\left(z_{t+1}^n, \omega_{t+1}^1\right) + d_{t+1}^n}.$$

We then get a multifunction that maps the current state $\left(z, \omega^1\right)$ into the "distribution" of future endogenous states $\omega^{1,n}$. If N is large, this approach is not practical so it is better to use $\left(s_t^1, b_t^1\right)$ as a state space, as those variables evolve deterministically.

15.1 Sunspots and Projection Methods

We can use projection methods to study the global dynamics of models with sunspots. The New Keynesian model with a zero lower bound is described by the equations

$$\begin{aligned}
\frac{C_t^{-\sigma}}{R_t} &= \beta E_t\left[\frac{C_{t+1}^{-\sigma}}{\Pi_{t+1}}\right] \\
R_t &= \max\left\{1, \frac{\Pi^*}{\beta}\left(\frac{\Pi_t}{\Pi^*}\right)^{\phi_\pi}\right\} \\
C_t &= \left(1 - \frac{\psi}{2}\left(\Pi_t - \Pi^*\right)^2\right)\exp\left(a_t\right) L_t \\
C_t^{-\sigma} w_t &= 1 \\
w_t &= \left(1 - \lambda_t\right)\exp\left(a_t\right) \\
0 &= \left(1 - \psi\left(\Pi_t - \Pi^*\right)\Pi_t - \lambda_t\gamma\right)\exp\left(a_t\right) L_t C_t^{-\sigma} + \beta E_t\left[C_{t+1}^{-\sigma}\left(\Pi_{t+1} - \Pi^*\right)\Pi_{t+1}\exp\left(a_{t+1}\right) L_{t+1}\right].
\end{aligned}$$

In general, this model has two steady states, one with $R = \frac{\Pi^*}{\beta}$ and one with $R = 1$. Around the first steady state, the model is locally determinate if $\phi_\pi > 1$, which results in the unique state variable a_t. Around the second steady state, or if $\phi_\pi < 1$, the model is locally indeterminate and therefore has sunspot equilibria. Sunspot equilibria are not functions of just a_t, but rather require an expanded state space with additional predetermined variables and extrinsic shocks. Furthermore, as shown in Benhabib, Schmitt-Grohé, and Uribe (2001), some locally unstable dynamics around the determinate steady state cannot be ruled out as equilibria because they converge to $R = 1$ instead of exploding.

To solve for sunspot equilibria, define the new variable

$$S_t = C_t^{-\sigma}\left(\Pi_t - \Pi^*\right)\Pi_t \exp\left(a_t\right) L_t;$$

then we can solve for

$$\{C_t, R_t, \Pi_t, L_t, w_t, \lambda_t, S_t\}$$

using the structural equations

$$\begin{aligned}
\frac{C_t^{-\sigma}}{R_t} &= \beta E_t \left[\frac{C_{t+1}^{-\sigma}}{\Pi_{t+1}}\right] \\
R_t &= \max\left\{1, \frac{\Pi^*}{\beta}\left(\frac{\Pi_t}{\Pi^*}\right)^{\phi_\pi}\right\} \\
C_t &= \left(1 - \frac{\psi}{2}\left(\Pi_t - \Pi^*\right)^2\right)\exp\left(a_t\right) L_t \\
C_t^{-\sigma} w_t &= 1 \\
w_t &= \left(1 - \lambda_t\right)\exp\left(a_t\right) \\
C_t^{-\sigma}\left(\Pi_t - \Pi^*\right)\Pi_t \exp\left(a_t\right) L_t &= S_{t-1}\exp\left(\nu_t\right) \\
0 &= \left(1 - \psi\left(\Pi_t - \Pi^*\right)\Pi_t - \lambda_t\gamma\right)\exp\left(a_t\right) L_t C_t^{-\sigma} + \beta S_t.
\end{aligned}$$

The states are now $\{S_{t-1}, a_t, \nu_t\}$. Let

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

and approximate expectations with Gauss-Hermite quadrature:

$$\begin{aligned}
\frac{C_t^{-\sigma}}{R_t} &= \beta \sum_{i,j=1}^{n} \omega_{i,j}\left[\frac{C_{t+1,i,j}^{-\sigma}}{\Pi_{t+1,i,j}}\right] \\
R_t &= \max\left\{1, \frac{\Pi^*}{\beta}\left(\frac{\Pi_t}{\Pi^*}\right)^{\phi_\pi}\right\} \\
C_t &= \left(1 - \frac{\psi}{2}\left(\Pi_t - \Pi^*\right)^2\right)\exp\left(a_t\right) L_t \\
w_t &= C_t^{\sigma} \\
\lambda_t &= 1 - \frac{w_t}{\exp\left(a_t\right)} \\
S_{t-1}\exp\left(\nu_t\right) &= C_t^{-\sigma}\left(\Pi_t - \Pi^*\right)\Pi_t \exp\left(a_t\right) L_t \\
0 &= \left(1 - \psi\left(\Pi_t - \Pi^*\right)\Pi_t - \lambda_t\gamma\right)\exp\left(a_t\right) L_t C_t^{-\sigma} + \beta S_t.
\end{aligned}$$

16 Endogenous Grid Method

The state space of a dynamic model is not unique, so we can sometimes change the states in a way that simplifies the computation. One particularly clever method is to use a state variable that allows us to analytically solve the first-order condition, once we put a grid on the *future* state instead of the current; Carroll (2006) was the first to suggest this approach and gave it the name "endogenous grid method" or EGM.

Rewrite the growth model in terms of the state variable total output

$$y = k^{\alpha} + (1-\delta)\, k,$$

so we get

$$v(k) = \max_{k'} \left\{ \frac{(y-k')^{1-\sigma} - 1}{1-\sigma} + \beta v\left(\left(k'\right)^{\alpha} + (1-\delta)\, k'\right) \right\}.$$

Fix grids for y and k' and guess $v^0(y)$. The first-order condition is

$$\left(y - k'\right)^{-\sigma} = \beta D v^0\left(\left(k'\right)^{\alpha} + (1-\delta)\, k'\right)\left(\alpha\left(k'\right)^{\alpha-1} + 1 - \delta\right),$$

which we evaluate using a spline. Solving for $y(k')$ yields

$$\begin{aligned} c\left(k'\right) &= \left(\beta D v^0\left(\left(k'\right)^{\alpha} + (1-\delta)\, k'\right)\left(\alpha\left(k'\right)^{\alpha-1} + 1 - \delta\right)\right)^{-\frac{1}{\sigma}} \\ y\left(k'\right) &= c\left(k'\right) + k'. \end{aligned}$$

The value function is then

$$V\left(k'\right) = \frac{1}{1-\sigma}\left(y\left(k'\right) - k'\right)^{1-\sigma} + \beta v^0\left(\left(k'\right)^{\alpha} + (1-\delta)\, k'\right).$$

We now have a pair of vectors $\left(y\left(k'\right), V\left(k'\right)\right)$, but we need $v^1(y)$. We therefore interpolate V using the grid for k' to the grid for y. That process is harder to describe than to do – you just construct a spline through the nodes $y(k')$ and the function values $V(k')$, then evaluate this spline at the grid of points for y. That is, think of stacking the points and values

$$\left(\begin{bmatrix} y\left(k_1'\right) \\ \vdots \\ y\left(k_n'\right) \end{bmatrix}, \begin{bmatrix} V\left(k_1'\right) \\ \vdots \\ V\left(k_n'\right) \end{bmatrix}\right).$$

If we ignore the k' argument, this is a vector of nodes and associated function values, which we can use to construct a spline interpolant. We then compute the value of that spline interpolant at the vector of nodes $\{y_1, ..., y_n\}$.

Note that EGM also works with Euler equations, which should be obvious since the Euler equation is derived using the derivative of the value function. The Euler equation is

$$c^{-\sigma} = \beta \left(c'\right)^{-\sigma} \left(\alpha \left(k'\right)^{\alpha-1} + 1 - \delta\right).$$

We guess $c' = g^0\left(k'\right)$, and then note that

$$\begin{aligned} c\left(k'\right) &= \left(\beta \left(\alpha \left(k'\right)^{\alpha-1} + 1 - \delta\right)\right)^{-\frac{1}{\sigma}} g^0\left(k'\right) \\ y\left(k'\right) &= c\left(k'\right) + k' \end{aligned}$$

as before.

EGM can also be done with stochastic TFP:

$$\left(y - k'\right)^{-\sigma} = \beta E\left[Dv^0\left(\exp\left(z'\right)\left(k'\right)^{\alpha} + \left(1-\delta\right)k', z'\right)\left(\exp\left(z'\right)\alpha\left(k'\right)^{\alpha-1} + 1 - \delta\right)\middle| z\right].$$

Solving for $y\left(k'\right)$ yields

$$\begin{aligned} c\left(k', z\right) &= \left(\beta E\left[Dv^0\left(\exp\left(z'\right)\left(k'\right)^{\alpha} + \left(1-\delta\right)k', z'\right)\left(\exp\left(z'\right)\alpha\left(k'\right)^{\alpha-1} + 1 - \delta\right)\middle| z\right]\right)^{-\frac{1}{\sigma}} \\ y\left(k', z\right) &= c\left(k', z\right) + k'. \end{aligned}$$

The value function is then

$$V\left(k', z\right) = \frac{\left(y\left(k'\right) - k'\right)^{1-\sigma} - 1}{1-\sigma} + \beta E\left[v^0\left(\exp\left(z'\right)\left(k'\right)^{\alpha} + \left(1-\delta\right)k'\right)\middle| z\right].$$

Our pair of vectors is now $\left(y\left(k', z\right), V\left(k', z\right)\right)$, which means that the grid for y will vary with z.

EGM can easily handle borrowing constraints. Take a buffer-stock saving model described by

$$v\left(m, e\right) = \max_{c,k} \left\{\frac{c^{1-\sigma} - 1}{1-\sigma} + \beta E_{e'}\left[v\left(m', e'\right)\right]\right\}$$

subject to

$$\begin{aligned} m &\geq c + k \\ m' &\leq Rk + e' \\ k &\geq 0. \end{aligned}$$

The Euler equation is

$$(m-k)^{-\sigma} = \beta R E_{e'}\left[\left(c'\right)^{-\sigma}\right] + \mu.$$

Now guess $c' = g^0(m')$, and then set a grid for k. If $k > 0$ we know $\mu = 0$, so we can solve for

$$\begin{aligned} m(k) &= k + \left(\beta R E_{e'}\left[\left(g^0\left(Rk + e'\right)\right)^{-\sigma}\right]\right)^{-\frac{1}{\sigma}} \\ c(k) &= m(k) - k \end{aligned}$$

and then interpolate from $(m(k), c(k))$ to the grid for m to get $g^1(m)$. Instead, if $k = 0$, we can find the highest value of m such that the constraint binds (which is where $\mu = 0$ and $k = 0$ both hold), and we can then set $g^1(m) = m$ for any values of m below that value and choose μ to satisfy the Euler equation at $k = 0$:

$$\mu = m^{-\sigma} - \beta R E_{e'}\left[\left(g^0\left(e'\right)\right)^{-\sigma}\right].$$

We can also solve versions of the model with elastic labor. Suppose the utility function takes the form

$$u(c,h) = \frac{c^{1-\sigma}}{1-\sigma} - \frac{h^{1+\theta}}{1+\theta}.$$

The first-order conditions are

$$\begin{aligned} c_t^{-\sigma} &= \beta(1+r-\delta) E_t\left[c\left(k_{t+1}, \epsilon'\right)^{-\sigma}\right] \\ c_t^{-\sigma} w\epsilon &= \psi h_t^{\frac{1}{\theta}} \\ c_t + k_{t+1} &= m_t + w\epsilon h_t \end{aligned}$$

which, given k_{t+1}, can be solved to obtain

$$\begin{aligned} c_t(k_{t+1}) &= \left(\beta(1+r-\delta) E_t\left[c\left(k_{t+1}, \epsilon'\right)^{-\sigma}\right]\right)^{-\frac{1}{\sigma}} \\ h_t(k_{t+1}) &= \left(\frac{c_t(k_{t+1})^{-\sigma} w\epsilon}{\psi}\right)^{\theta} \\ m_t(k_{t+1}) &= c_t(k_{t+1}) + k_{t+1} - w\epsilon h_t(k_{t+1}). \end{aligned}$$

Alternatively, suppose the utility function takes the form

$$u(c,h) = \frac{\left(c(1-h)^{\mu}\right)^{1-\sigma}}{1-\sigma};$$

now there is the complication that h cannot be negative. The first-order conditions are

$$\begin{aligned} c_t^{-\sigma}\left(1-h_t\right)^{\mu(1-\sigma)} &= \beta\left(1+r-\delta\right)E_t\left[c\left(k_{t+1},\epsilon'\right)^{-\sigma}\left(1-h\left(k_{t+1},\epsilon'\right)\right)^{\mu(1-\sigma)}\right] \\ \mu c_t^{1-\sigma}\left(1-h_t\right)^{\mu(1-\sigma)-1} &= c_t^{-\sigma}\left(1-h_t\right)^{\mu(1-\sigma)}w\epsilon \\ c_t+k_{t+1} &= m_t+w\epsilon h_t \end{aligned}$$

which imply

$$1-h_t=\frac{\mu}{we}c_t.$$

Solving the system yields

$$\begin{aligned} c_t\left(k_{t+1}\right) &= \left(\frac{\beta\left(1+r-\delta\right)}{\left(\frac{\mu}{w\epsilon}\right)^{\mu(1-\sigma)}}E_t\left[c\left(k_{t+1},\epsilon'\right)^{-\sigma}\left(1-h\left(k_{t+1},\epsilon'\right)\right)^{\mu(1-\sigma)}\right]\right)^{\frac{1}{\mu(1-\sigma)-\sigma}} \\ h_t\left(k_{t+1}\right) &= 1-\frac{\mu}{ew}c_t\left(k_{t+1}\right) \\ m_t\left(k_{t+1}\right) &= k_{t+1}+c_t\left(k_{t+1}\right)-weh_t\left(k_{t+1}\right); \end{aligned}$$

if $h\left(k'\right)<0$ then

$$\begin{aligned} h_t\left(k_{t+1}\right) &= 0 \\ c_t\left(k_{t+1}\right) &= \left(\beta\left(1+r-\delta\right)E_t\left[c\left(k_{t+1},\epsilon'\right)^{-\sigma}\left(1-h\left(k_{t+1},\epsilon'\right)\right)^{\mu(1-\sigma)}\right]\right)^{-\frac{1}{\sigma}} \\ m_t\left(k_{t+1}\right) &= k_{t+1}+c_t\left(k_{t+1}\right). \end{aligned}$$

With other utility functions, we may need to still use a nonlinear solver, which eliminates the benefits from EGM (some papers still use it, but that is wasteful since, at least for concave problems, the number of choice variables is largely irrelevant).

With a single state variable, as long as the relationship between y and V is monotonic we can do any kind of interpolation. With more than one state variable, we will not get a rectangular grid in general, so we need to use a scattered data interpolation method (Delaunay triangulation for example). Here is an example, taken from White (2018). Guess $V\left(m,h\right)$ on a grid for

(m,h). Set up the grid for (a,z). Then

$$\begin{aligned} V(m,h) &= \max_{c,n} \left\{ \frac{c^{1-\sigma}-1}{1-\sigma} + \beta E\left[\left(1-\frac{\phi}{1+h'}\right) V\left(m',h'\right)\right]\right\} \\ m &= c+n+a \\ m' &= aR+w'h' \\ h' &= \left(1-\delta'\right) z \end{aligned}$$

The first-order conditions can be solved to obtain

$$\begin{aligned} h'(a,z) &= \left(1-\delta'\right) z \\ m'(a,z) &= aR+w'h' \\ c(a,z) &= \left(\beta RE\left[D_m V\left(m'(a,z),h'(a,z)\right)\right]\right)^{-\frac{1}{\sigma}} \end{aligned}$$

and

$$n(a,z) = \left(\frac{RE\left[\left(1-\frac{\phi}{1+h'(a,z)}\right) D_m V\left(m'(a,z),h'(a,z)\right)\right]}{E\left[(1-\delta')\left(\begin{array}{c} \frac{\phi}{(1+h'(a,z))^2} V\left(m'(a,z),h'(a,z)\right)+ \\ \left(1-\frac{\phi}{1+h'(a,z)}\right)\left(w' D_m V\left(m'(a,z),h'(a,z)\right)+D_h V\left(m'(a,z),h'(a,z)\right)\right) \end{array} \right)\right]} \right)^{\frac{1}{\nu-1}}.$$

Now we invert to get the initial states:

$$\begin{aligned} m(a,z) &= c(a,z)+n(a,z)+a \\ h(a,z) &= z-\frac{\gamma}{\nu} n(a,z)^{\nu}. \end{aligned}$$

The intermediate value function is

$$v(a,z) = \frac{c(a,z)^{1-\sigma}}{1-\sigma} + \beta E\left[\left(1-\frac{\phi}{1+h'(a,z)}\right) V\left(m'(a,z),h'(a,z)\right)\right];$$

now we can use Delaunay tesselation to interpolate $(m(a,z),h(a,z),v(a,z))$ to the grid for (m,h). There are some complications associated with the boundaries that have to be addressed, so beware that using this method may be more complicated than it seems. Furthermore, since the nodes change each iteration, we have to reconstruct the triangles repeatedly, which slows down this method. White (2018) suggests using a "warped" version of linear splines which is significantly faster but more complicated and likely problem-specific.

There are some recent papers that extend EGM to non-convex environments where Euler equations are only necessary. The key is the envelope theorem by Clausen and Strub (2020) that shows for many nonconvex problems the points of nondifferentiability are never optimal (so called "bad kinks"), so we can basically ignore them and use the Euler equation as long as we check which solution is optimal (using the Bellman equation). For example, suppose we have a model with linear costs of adjusting a stock of durable goods:

$$\begin{aligned} v\left(p_t, n_t, m_t\right) &= \max\left\{v^{keep}\left(p_t, n_t, m_t\right), v^{adj}\left(p_t, x_t\right)\right\} \\ x_t &= m_t + (1-\tau)\, n_t \end{aligned}$$

where

$$\begin{aligned} v^{keep}\left(p_t, n_t, m_t\right) &= \max_{c_t}\left\{u\left(c_t, n_t\right) + E\left[v\left(p_{t+1}, n_{t+1}, m_{t+1}\right)\right]\right\} \\ a_t &= m_t - c_t \\ m_{t+1} &= Ra_t + p_{t+1} \\ n_{t+1} &= (1-\delta)\, n_t \\ a_t &\geq 0 \\ v^{adj}\left(p_t, x_t\right) &= \max_{c_t, d_t}\left\{u\left(c_t, d_t\right) + \beta E\left[v\left(p_{t+1}, n_{t+1}, m_{t+1}\right)\right]\right\} \\ a_t &= x_t - c_t - d_t \\ m_{t+1} &= Ra_t + p_{t+1} \\ n_{t+1} &= (1-\delta)\, d_t \\ a_t &\geq 0. \end{aligned}$$

We can rewrite the problems as

$$\begin{aligned} v^{keep}\left(p_t, n_t, m_t\right) &= \max_{c_t}\left\{u\left(c_t, n_t\right) + w\left(p_t, d_t, a_t\right)\right\} \\ c_t + a_t &\leq m_t \\ d_t &= n_t \end{aligned}$$

where

$$w\left(p_t, d_t, a_t\right) = \beta E\left[v\left(p_{t+1}, n_{t+1}, m_{t+1}\right)\right]$$
$$n_{t+1} = \left(1-\delta\right) d_t$$
$$m_{t+1} = R\left(m_t - c_t\right) + p_{t+1}$$

and

$$v^{adj}\left(p_t, x_t\right) = \max_{d_t}\left\{v^{keep}\left(p_t, d_t, m_t\right)\right\}$$
$$m_t = x_t - d_t$$
$$0 \leq d_t \leq m_t.$$

Assume

$$u\left(c, d\right) = \frac{\left(c^{\alpha}\left(d+\underline{d}\right)^{1-\alpha}\right)^{1-\sigma}}{1-\sigma}.$$

The Euler equation is therefore

$$\alpha c_t^{\alpha(1-\rho)-1}\left(d_t+\underline{d}\right)^{(1-\alpha)(1-\rho)} = q_t$$
$$q_t = \beta R E\left[\alpha c_{t+1}^{\alpha(1-\rho)-1}\left(d_{t+1}+\underline{d}\right)^{(1-\alpha)(1-\rho)}\right]$$

and using EGM we get

$$c_t\left(p_t, d_t, a_t\right) = \left(\frac{q_t}{\alpha\left(d_t+\underline{d}\right)^{(1-\alpha)(1-\rho)}}\right)^{\frac{1}{\alpha(1-\rho)-1}}$$
$$m_t\left(p_t, d_t, a_t\right) = a_t + c_t\left(p_t, d_t, a_t\right).$$

To eliminate non-optimal choices, Druedahl (2018) suggests the following upper envelope method. The given inputs are the current income state p_t, the stock of durables d_t, a grid for the cash-on-hand variable $G_m = \{m_j\}_{j=1}^{n_m}$ with $m_1 = 0$, and a grid for the end-of-period asset holdings $G_a = \left\{a^i\right\}_{i=1}^{n_a}$ with $a^1 = 0$. Then the algorithm is

1. Loop $j \in \{1, ..., n_m\}$
2. $v_j = -\infty$
3. Loop $i \in \{1, ..., n_a\}$

4. $w^i = w\left(p, d, a^i\right)$

5. $c^i = c\left(p, d, a^i\right)$

6. $m^i = m\left(p, d, a^i\right)$

   (a) Loop $j \in \{1, ..., n_m\}$

   (b) if $m^i \leq m^1$ set $c_j = m_j$ and $v_j = u\left(c_j, d\right) + w^1$

7. Loop $i \in \{1, ..., n_a - 1\}$

   (a) Loop $j \in \{1, ..., n_m\}$

      i. if $m_j \in \left[m^i, m^{i+1}\right]$ set $c_j^i = c^i + \frac{c^{i+1} - c^i}{m^{i+1} - m^i}\left(m_j - m^i\right)$ and $v_j^i = u\left(c_j^i, d\right) + \beta\left(w^i + \frac{w^{i+1} - w^i}{a^{i+1} - a^i}\left(m_j - c_j^i -\right.\right.$

      ii. if $v_j^i > v^j$ set $c_j = c_j^i$ and $v^j = v_j^i$

8. Return list $\{v_1, ..., v_{n_m}\}$ and $\{c_1, ..., c_{n_m}\}$

This "nested-EGM" is significantly faster than alternatives, and handles efficiently the fact that the Euler equation is necessary but not sufficient for optimality (it may have multiple roots).

Related to EGM is ECM, or the Envelope Condition Method, which solves the model "forward" using the envelope theorem (developed by Maliar and Maliar 2014). Here are the steps. Guess v^0 and compute Dv^0; then use the envelope condition to obtain

$$c\left(k\right) = \left(\frac{Dv^0\left(k\right)}{\alpha k^{\alpha - 1} + 1 - \delta}\right)^{-\frac{1}{\sigma}}.$$

Now use the resource constraint to obtain

$$k'\left(k\right) = k^\alpha + \left(1 - \delta\right)k - c\left(k\right)$$

and update the value function

$$v^1\left(k\right) = \frac{c\left(k\right)^{1-\sigma}}{1 - \sigma} + \beta v^0\left(k'\left(k\right)\right).$$

Note that, if we are not interested in the value function per se, we could proceed by simply guessing the derivative directly Dv^0 and iterating on

$$Dv^{n+1}\left(k\right) = \beta\left(1 - \delta + \alpha k^{\alpha - 1}\right) Dv^n\left(\left(k'\right)^n\left(k\right)\right)$$

where

$$c^n(k) = \left(\frac{Dv^n(k)}{\alpha k^{\alpha-1} + 1 - \delta}\right)^{-\frac{1}{\sigma}}$$
$$\left(k'\right)^n(k) = k^\alpha + (1-\delta)k - c^n(k).$$

While both of these approaches generate significant time savings, they are quite limited – they apply only to models with very specific structure. White (2018) derives some conditions needed to apply the EGM with multiple states.

17 Continuous Time Optimal Control

Optimal control theory deals with the problem of optimizing the path of a continuous time dynamic system relative to some performance metric. An optimal control problem specifies the objective, the constraints, and the relationship between the state of the system and the controls applied to it. In continuous time the state evolution equation is written

$$\dot{x}(t) = f(x(t), u(t), t)$$

where $x(t)$ is the state vector, $u(t)$ is the control vector, and $\dot{x}(t)$ is the current change in the state vector (the time derivative). We will assume f is continuously differentiable. The constraints are

$$u(t) \in \Omega(t)$$

where $\Omega(t)$ is a subset of the space that $u(t)$ lives in. Time is assumed to flow over the (possibly infinite) interval $[0, T]$.

The objective function is

$$J = \int_0^T F(x(t), u(t), t)\, dt + S(x(T), T),$$

where $F(x, u, t)$ is the instantaneous return and $S(x, T)$ is the **salvage function**. Both are assumed to be continuously differentiable.

The control problem is then

$$\max_{u(t)\in\Omega(t)} \left\{ \int_0^T F(x(t), u(t), t)\, dt + S(x(T), T) \right\}$$

subject to

$$\dot{x}(t) = f(x(t), u(t), t).$$

We will assume the initial state is known:

$$x(0) = x_0;$$

as with difference equations, a first-order differential equation requires a boundary condition to uniquely identify the optimal path.

The solution to an optimal control problem will satisfy the Principle of Optimality, in which an optimal control remains optimal as time goes along. To prove this intuitively, we begin by deriving the Hamilton-Jacobi-Bellman equation, which parallels the discrete time Bellman equation. Denote the value function by $V(x,t)$. Between times t and $t+\delta t$ the value function changes from $V(x(t),t)$ to $V(x(t+\delta t),t+\delta t)$, or $V(x+\delta x, t+\delta t)$. The value function over this interval consists of two parts – the incremental change in J from t to $t+\delta t$ and the value function at $t+\delta t$; as in discrete time, the latter term captures the returns from that point forward. Thus, the optimal control maximizes

$$V(x,t) = \max_{\substack{u(\tau)\in\Omega(\tau)\\ \tau\in[t,t+\delta t]}} \left\{ \int_t^{t+\delta t} F(x(\tau), u(\tau), \tau)\, d\tau + V(x(t+\delta t), t+\delta t) \right\}.$$

δt is small.

Since F is continuous, the integral will be approximately $F(x,u,t)\,\delta t$ so that we have

$$V(x,t) = \max_{u(t)\in\Omega(t)} \{F(x(t), u(t), t)\,\delta t + V(x(t+\delta t), t+\delta t)\} + o(\delta t).$$

$o(\delta t)$ captures the neglected higher-order terms and satisfies

$$\lim_{\delta t\to 0} \left\{ \frac{o(\delta t)}{\delta t} \right\} = 0.$$

Assuming that V is continuously differentiable we can use a Taylor expansion to write

$$V(x(t+\delta t), t+\delta t) = V(x,t) + (D_x V(x,t)\,\dot{x} + D_t V(x,t))\,\delta t + o(\delta t).$$

Using the expression for $\dot{x}$ and replacing the resulting expression in the value function equation we can obtain

$$V(x,t) = \max_{u(t)\in\Omega(t)} \{F(x(t), u(t), t)\,\delta t + V(x(t), t) + D_x V(x(t), t)\, f(x(t), u(t), t)\,\delta t + D_t V(x(t), t)\,\delta t\} + o(\delta t)$$

V thus cancels from both sides, and we then divide by δt:

$$0 = \max_{u(t)\in\Omega(t)} \left\{F\left(x\left(t\right),u\left(t\right),t\right) + D_x V\left(x\left(t\right),t\right) f\left(x\left(t\right),u\left(t\right),t\right) + D_t V\left(x\left(t\right),t\right)\right\} + \frac{o\left(\delta t\right)}{\delta t}.$$

Letting $\delta t \to 0$ yields

$$0 = \max_{u(t)\in\Omega(t)} \left\{F\left(x\left(t\right),u\left(t\right),t\right) + D_x V\left(x\left(t\right),t\right) f\left(x\left(t\right),u\left(t\right),t\right) + D_t V\left(x\left(t\right),t\right)\right\}.$$

The boundary condition that must be satisfied is

$$V\left(x\left(T\right),T\right) = S\left(x\left(T\right),T\right),$$

so that value at the end of the horizon is equal to the salvage function.

The derivative terms are the marginal contributions of the states to the objective function. The marginal return vector is denoted by an adjoint (co-state) vector $\lambda\left(t\right)$ defined as

$$\lambda\left(t\right) \equiv D_x V\left(x\left(t\right),t\right).$$

This quantity is interpreted as the per unit change in the objective for small changes in the state. Defining the Hamiltonian function as

$$H\left(x,u,V_x,t\right) = F\left(x,u,t\right) + D_x V\left(x,t\right) f\left(x,u,t\right)$$

or

$$H\left(x,u,\lambda,t\right) = F + \lambda f,$$

we can write the HJB equation

$$0 = \max_{u\in\Omega(t)} \left\{H + D_t V\right\}.$$

Since $D_t V$ does not depend on u, it can be removed from the maximization.

We now need the adjoint equation, which describes how the λ vector evolves over time. Let

$$x\left(t\right) = x^*\left(t\right) + \delta x\left(t\right)$$

denote small perturbations around the optimal path $x^*\left(t\right)$. At a fixed instant t the HJB equation requires

$$H\left(x^*\left(t\right),u^*\left(t\right),D_x V\left(x^*\left(t\right),t\right),t\right) + D_t V\left(x^*\left(t\right),t\right) \geq H\left(x\left(t\right),u^*\left(t\right),D_x V\left(x\left(t\right),t\right),t\right) + D_t V\left(x\left(t\right),t\right).$$

The LHS is zero, since it attains the LHS of the HJB equation. The RHS is only zero if $u^*(t)$ is also optimal for the path $x(t)$; since these paths are not equal, in general that condition will be false. The RHS attains its maximum therefore at $x^*(t)$ and is not constrained; therefore, the derivative must vanish:

$$\frac{\partial}{\partial x}H\left(x(t), u^*(t), \frac{\partial}{\partial x}V(x(t), t), t\right) + \frac{\partial^2}{\partial x \partial t}(x(t), t) = 0.$$

From the definition of the Hamiltonian we then obtain

$$\frac{\partial}{\partial x}F + \frac{\partial}{\partial x}V\frac{\partial}{\partial x}f + f'\frac{\partial^2}{\partial x^2}V + \frac{\partial^2}{\partial x \partial t}V = \frac{\partial}{\partial x}F + \frac{\partial}{\partial x}V\frac{\partial}{\partial x}f + \left(\frac{\partial^2}{\partial x^2}Vf\right)' + \frac{\partial}{\partial x \partial t}V = 0.$$

Next, we take the time derivative of $V_x(x,t)$ to obtain

$$\frac{d\left(\frac{\partial}{\partial x}V\right)}{dt} = \left(\frac{\partial^2}{\partial x^2}V\dot{x}\right)' + \frac{\partial}{\partial x \partial t}V = \left(\frac{\partial^2}{\partial x^2}Vf\right)' + \frac{\partial}{\partial x \partial t}V.$$

The equation above implies

$$\frac{dV_x}{dt} = -\frac{\partial}{\partial x}F - \frac{\partial}{\partial x}V\frac{\partial}{\partial x}f.$$

Using the definition of λ one obtains the adjoint equation

$$\dot{\lambda}(t) = -\frac{\partial}{\partial x}F - \lambda\frac{\partial}{\partial x}f$$

or

$$\dot{\lambda}(t) = -\frac{\partial}{\partial x}H.$$

Finally, we have a boundary condition on the adjoint as well:

$$\lambda(T) = \left.\frac{\partial S(x,T)}{\partial x}\right|_{x=x(T)} = \frac{\partial}{\partial x}S(x(T), T).$$

The state equation can now be written as

$$\dot{x}(t) = f = \frac{\partial}{\partial \lambda}H.$$

Collecting our functions together we find that an optimal control path for the system must obey the equations

$$\begin{aligned}
\dot{x}(t) &= \frac{\partial}{\partial \lambda}H \\
\dot{\lambda}(t) &= -\frac{\partial}{\partial x}H \\
x(0) &= x_0 \\
\lambda(T) &= \frac{\partial}{\partial x}S(x(T), T),
\end{aligned}$$

which is a system of first-order differential equations with two boundary conditions. The optimal control is obtained from the requirement that

$$H\left(x^{*}\left(t\right),u^{*}\left(t\right),\lambda\left(t\right),t\right)\geq H\left(x^{*}\left(t\right),u\left(t\right),\lambda\left(t\right),t\right)$$

$\forall u\left(t\right)\in\Omega\left(t\right)$ and $\forall t\in\left[0,T\right]$.

Let's do some simple examples. Consider the problem

$$\max\left\{J=\int_{0}^{1}-xdt\right\}$$

subject to the state equation

$$\dot{x}\left(t\right)=u\left(t\right)$$
$$x\left(0\right)=1$$

and the control constraint

$$\Omega\left(t\right)=\left[-1,1\right].$$

Essentially, this is a problem of minimizing the signed area under the curve $x\left(t\right)$. First we form the Hamiltonian

$$H=-x+\lambda u.$$

Since the objective is linear, the solution will be of the 'bang-bang' variety:

$$u^{*}\left(t\right)=\begin{cases}1 & \text{if }\lambda>0\\ \left[0,1\right] & \text{if }\lambda=0\\ -1 & \text{if }\lambda<0\end{cases};$$

this is often written

$$u^{*}\left(t\right)=\text{bang}\left[-1,1;\lambda\right],$$

which means if $\lambda<0$ the value is -1 and if $\lambda>0$ the value is 1. The function to the right of the semicolon is called the switching function, and it determines when to go from one value to the next. To find the path for λ we write the adjoint equation

$$\dot{\lambda}\left(t\right)=-\frac{\partial}{\partial x}H$$
$$=1$$

and append the boundary condition

$$\lambda(1) = 0.$$

It is easy to solve this to obtain

$$\lambda(t) = t - 1.$$

Clearly, $\lambda(t) < 0 \ \forall t \in [0,1]$, so that we can evaluate the control as

$$u^*(t) = -1.$$

The path of the state is then

$$\dot{x}(t) = -1$$

with boundary

$$x(0) = 1.$$

The solution is

$$x(t) = 1 - t.$$

Another example is

$$\max\left\{J = \int_0^1 -\frac{1}{2}x^2 dt\right\}$$

subject to

$$\begin{aligned}\dot{x}(t) &= u(t)\\ x(0) &= 1\\ u(t) &\in [-1,1].\end{aligned}$$

The Hamiltonian is

$$H = -\frac{1}{2}x^2 + \lambda u$$

which is still linear in u. The solution is again

$$u^*(t) = \text{bang}[-1,1;\lambda].$$

The adjoint equation is

$$\dot{\lambda}(t) = -H_x = x$$

with boundary condition

$$\lambda(1) = 0.$$

Assume that the optimal solution involves $\lambda \leq 0$ over the entire interval. If so, $u = -1$ so that

$$x(t) = 1 - t.$$

In the adjoint equation we now have

$$\dot{\lambda}(t) = 1 - t.$$

Integrating both sides yields

$$\int_t^1 \dot{\lambda}(\tau)\, d\tau = \int_t^1 (1 - \tau)\, d\tau$$

or

$$\lambda(1) - \lambda(t) = \left(\tau - \frac{1}{2}\tau^2\right)\Big|_t^1.$$

Using the boundary condition we obtain

$$\lambda(t) = -\frac{1}{2}t^2 + t - \frac{1}{2}.$$

Since $\lambda(t)$ is in fact nonpositive over the entire interval our assumption is verified.

The final example is more involved, as it will not have a bang-bang optimal control. The problem is

$$\max\left\{J = \int_0^2 \left(2x - 3u - u^2\right) dt\right\}$$

subject to

$$\begin{aligned} \dot{x} &= x + u \\ x(0) &= 5 \\ u &\in [0, 2]. \end{aligned}$$

The Hamiltonian is

$$\begin{aligned} H &= \left(2x - 3u - u^2\right) + \lambda(x + u) \\ &= (2 + \lambda)\, x - \left(u^2 + 3u - \lambda u\right). \end{aligned}$$

The optimal policy satisfies

$$H_u = -2u - 3 + \lambda = 0$$

or

$$u^{*}(t)=\frac{\lambda(t)-3}{2},$$

as long as this remains within the interval $[0,2]$. The adjoint equation is

$$\dot{\lambda}(t)=-2-\lambda$$
$$\lambda(2)=0.$$

This can be rewritten as

$$\dot{\lambda}+\lambda=-2$$
$$\lambda(2)=0.$$

Using an integration factor this differential equation has the solution

$$\lambda(t)=2\left(e^{2-t}-1\right).$$

To see this, note that the solution to the homogeneous equation

$$\dot{\lambda}+\lambda=0$$

can be obtained by multplying by the integration factor e^{t}:

$$e^{t}\dot{\lambda}+e^{t}\lambda=\frac{d}{dt}\left(\lambda e^{t}\right).$$

Setting the LHS to zero implies that

$$\frac{d}{dt}\left(\lambda e^{t}\right)=0.$$

Integrating then yields

$$\lambda e^{t}=C$$

or

$$\lambda=Ce^{-t}.$$

To get the particular solution, we then integrate the differential equation to obtain

$$\begin{aligned} e^t\dot{\lambda} + e^t\lambda &= -2e^t \\ \lambda(t)\, e^t|_t^2 &= -2\int e^t dt \\ \lambda(2)\, e^2 - \lambda(t)\, e^t &= -2e^t|_t^2 \\ \lambda(t)\, e^t &= 2\left(e^2 - e^t\right) \\ \lambda(t) &= 2\left(e^{2-t} - 1\right). \end{aligned}$$

Using the terminal condition $\lambda(T) = 0$ yields

$$\lambda(t) = -2e^t$$

The optimal control is therefore

$$u^*(t) = \begin{cases} 2 & \text{if } t < 2 - \log(4.5) \\ e^{2-t} - 2.5 & \text{if } t \in [2 - \log(4.5), 2 - \log(2.5)] \\ 0 & \text{if } t > 2 - \log(2.5) \end{cases}.$$

We next extend our approach to permit general constraints on the controls:

$$g(x, u, t) \geq 0.$$

We assume for now that u must enter g, and that g is continuously differentiable. Terminal constraints take the form

$$\begin{aligned} a(x(T), T) &\geq 0 \\ b(x(T), T) &= 0. \end{aligned}$$

Define the Hamiltonian as before:

$$H(x, u, \lambda, t) = F(x, u, t) + \lambda f(x, u, t)$$

and now also define the Lagrangian

$$L(x, u, \lambda, \mu, t) = H(x, u, \lambda, t) + \mu g(x, u, t)$$

where μ is a vector of multipliers. The adjoint vector λ now satisfies the differential equation

$$\dot{\lambda}(t) = -\frac{\partial}{\partial x} L(x, u, \lambda, \mu, t)$$

with the boundary conditions

$$\begin{aligned} \lambda(T) &= \frac{\partial}{\partial x} S(x(T), T) + \alpha \frac{\partial}{\partial x} a(x(T), T) + \beta \frac{\partial}{\partial x} b(x(T), T) \\ \alpha &\geq 0 \\ \alpha a(x(T), T) &= 0. \end{aligned}$$

The optimal control now must satisfy the following conditions:

$$\begin{aligned} \dot{x}^* &= f(x^*, u^*, t) \\ x^*(0) &= x_0 \\ a(x^*(T), T) &\geq 0 \\ b(x^*(T), T) &= 0 \\ \dot{\lambda} &= -\frac{\partial}{\partial x} L(x^*, u^*, \lambda, \mu^*, t) \\ \lambda(T) &= \frac{\partial}{\partial x} S(x^*(T), T) + \alpha \frac{\partial}{\partial x} a(x^*(T), T) + \beta b \frac{\partial}{\partial x}(x^*(T), T) \\ \alpha &\geq 0 \\ \alpha a(x^*(T), T) &= 0 \\ H(x^*(t), u^*(t), \lambda(t), t) &\geq H(x^*(t), u, \lambda(t), t) \\ g(x^*(t), u, t) &\geq 0 \\ \frac{\partial L}{\partial u}(u = u^*) &= 0 \\ \mu(t) &\geq 0 \\ \mu(t)\, g(x^*, u^*, t) &= 0. \end{aligned}$$

If we are allowed to choose the terminal time T, it must satisfy the additional "smooth pasting" condition

$$H(x^*(T^*), u^*(T^*), \lambda(T^*), T^*) + \frac{\partial}{\partial T} S(x^*(T^*), T^*) = 0.$$

To illustrate how to solve optimal control problems with constraints, we'll do the example

$$\max_u \left\{ J = \int_0^1 u dt \right\}$$

subject to

$$\begin{aligned} \dot{x} &= u \\ x(0) &= 1 \\ u &\geq 0 \\ x - u &\geq 0. \end{aligned}$$

The Hamiltonian is

$$H = u + \lambda u$$

which implies the optimal control

$$u = \text{bang}\left[0, x; 1 + \lambda\right].$$

The Lagrangian is

$$\begin{aligned} L &= H + \mu_1 u + \mu_2 (x - u) \\ &= \mu_2 x + (1 + \lambda + \mu_1 - \mu_2) u. \end{aligned}$$

The adjoint equation is therefore

$$\begin{aligned} \dot{\lambda} &= -\mu_2 \\ \lambda(1) &= 0. \end{aligned}$$

The optimal control must obey

$$\frac{\partial}{\partial u} L = 1 + \lambda + \mu_1 - \mu_2 = 0$$

and we also have the complementary slackness conditions

$$\begin{aligned} \mu_1 &\geq 0 \\ \mu_1 u &= 0 \\ \mu_2 &\geq 0 \\ \mu_2 (x - u) &= 0. \end{aligned}$$

Assume that the solution is

$$u^*(t) = x(t).$$

The implied path of the state is

$$x(t) = e^t > 0;$$

from here we get

$$u^* = e^t > 0$$

and therefore $\mu_1 = 0$. We therefore have

$$\mu_2 = 1 + \lambda,$$

since the constraint $x - u \geq 0$ is binding. Inserting this into the adjoint equation yields

$$\dot{\lambda} = -1 - \lambda$$

so that we can obtain the solution

$$\lambda(t) = e^{1-t} - 1.$$

Since this solution implies that the switching function in the bang function is always positive, we have that $u^* = x$ satisfies this condition as well.

Optimal controls may not be continuous if the constraint $g(x, u, t)$ does not depend on u. One can easily see that from the examples above, where the constraints took the form of simple bounds. When the objective was linear the solution was 'bang-bang' which is not continuous. In such cases the problem does not satisfy any constraint qualification condition. But this situation can occur even if the objective is nonlinear (for example, if it is piecewise linear).

The above cases were ones where we could solve the resulting differential equations by hand; naturally, as one gets more nonlinearity that will become rather difficult, if not impossible, so we will now turn to using the computer again.

17.0.1 Solving Ordinary Differential Equations

Suppose our goal is to solve the equation

$$y' = f(x, y)$$

for the path of y, where

$$y(x_0) = y_0$$

is known. The existence of a solution is guaranteed by the Cauchy-Peano theorem.

Theorem 122 *(Cauchy-Peano Theorem) Let* $(x_0, y_0) \in \mathcal{R} \times X$ *be given. Suppose* $f : Q \subset \mathcal{R} \times X \to X$ *is continuous and bounded on some region*

$$Q = \{(x, y) : |x - x_0| \leq a, |y - y_0| \leq b\}$$

with $(a, b) > 0$*. Then there exists* $\delta > 0$ *and a continuous function* $\phi : [x_0 - \delta, x_0 + \delta] \to X$ *such that* $y = \phi(x)$ *is a solution to* $y' = f(x, y)$*.*

Note that the theorem does not guarantee uniqueness; to get uniqueness we need uniform Lipschitz continuity in y.

Definition 123 *A function* $f : X \to Y$ *is* ***Lipschitz continuous*** *if* $d_Y(f(x_1), f(x_2)) \leq K(x_1, x_2) d_X(x_1, x_2)$ *for some finite* $K(x_1, x_2) \geq 0$ *holds at each* (x_1, x_2)*. A function is* ***uniformly Lipschitz continuous*** *if* K *can be chosen independently of* (x_1, x_2)*. If* $K \leq 1$ f *is a nonexpansive (or short) mapping, and if* $K < 1$ f *is a strict contraction.*

Theorem 124 *(Picard-Lindelöf Theorem) Let* $(x_0, y_0) \in \mathcal{R} \times X$ *be given. Suppose* $f : Q \subset \mathcal{R} \times X \to X$ *is uniformly Lipschitz continuous in* y *and continuous in* x *on some region*

$$Q = \{(x, y) : |x - x_0| \leq a, |y - y_0| \leq b\}$$

with $(a, b) > 0$*. Then there exists* $\delta > 0$ *and a continuous function* $\phi : [x_0 - \delta, x_0 + \delta] \to X$ *such that* $y = \phi(x)$ *is the unique solution to* $y' = f(x, y)$*.*

A more general existence theorem handles the case where the RHS is not continuous in x, although it requires us to think of a solution in a different sense.

Definition 125 *A function* $y = \phi(x)$ *is a* ***solution to*** $y' = f(x, y)$ ***in the extended sense*** *if (i)* $\phi(x)$ *is absolutely continuous (implying that the derivative exists almost everywhere), (ii)* $\phi(x)$ *satisfies the differential equation almost everywhere, and (iii)* $\phi(x_0)$ *satisfies the initial condition.*

Theorem 126 *(Carathéodory's Theorem) Let* $(x_0, y_0) \in \mathcal{R} \times X$ *be given. Suppose* $f : Q \subset \mathcal{R} \times X \to X$ *is continuous in* y *for fixed* x*, continuous in* x *for fixed* y *on the region*

$$Q = \{(x, y) : |x - x_0| \leq a, |y - y_0| \leq b\}$$

with $(a,b) > 0$*, and there exists a Lebesgue-integrable function* $m : [x_0 - a, x_0 + a] \to [0,\infty)$ *such that* $|f(y,x)| \leq m(x)\ \forall (x,y)$*. Then there exists* $\delta > 0$ *and a continuous function* $\phi : [x_0 - \delta, x_0 + \delta] \to X$ *such that* $y = \phi(x)$ *is a solution in the extended sense to* $y' = f(x,y)$*. If in addition there exists a Lebesgue-integrable function* $k : [x_0 - a, x_0 + a] \to [0,\infty)$ *such that* $|f(x,y_1) - f(x,y_2)| \leq k(x)|y_1 - y_2|\ \forall (x,y_1,y_2)$*, then* ϕ *is unique.*

The simplest method for solving a differential equation uses finite-differences to approximate the derivatives. Finite-difference methods specify a grid for x; for now, assume this grid is evenly spaced so that

$$x_i = x_0 + ih.$$

The intent is to find Y_i that approximates $y(x_i)$ at each grid point i. In essence we will construct a difference equation

$$Y_{i+1} = F(Y_i, Y_{i-1}, ..., x_{i+1}, x_i, ...)$$

and solve it sequentially, using the initial condition to pin down Y_0.

The first finite-difference method is Euler's method:

$$Y_{i+1} = Y_i + hf(x_i, Y_i).$$

This method can be motivated by a Taylor expansion argument, in which we keep only the linear term. As the step size goes to 0, Euler's method approaches $y(x)$. Even better is the implicit Euler's method, which is given by

$$Y_{i+1} = Y_i + hf(x_i, Y_{i+1}).$$

To obtain Y_{i+1} one must now solve a nonlinear equation at each grid point i, but the approximation is much better for large h. Convergence for this method is linear.

A second improvement on Euler's method is to use a better integration rule, the trapezoidal rule, which yields

$$Y_{i+1} = Y_i + \frac{h}{2}\left(f(x_i, Y_i) + f(x_{i+1}, Y_{i+1})\right).$$

Convergence here is quadratic.

These implicit methods are special cases of the family of Runge-Kutta methods. RK2 is given by

$$Y_{i+1} = Y_i + \frac{h}{2}\left(f(x_i, Y_i) + f(x_{i+1}, Y_i + hf(x_i, Y_i))\right);$$

the convergence here is quadratic. Even better is RK4, which has fourth-order convergence:

$$Y_{i+1} = Y_i + \frac{h}{6}\left(z_1 + z_2 + z_3 + z_4\right)$$

where

$$\begin{aligned}
z_1 &= f\left(x_i, Y_i\right) \\
z_2 &= f\left(x_i + \frac{1}{2}h, Y_i + \frac{1}{2}hz_1\right) \\
z_3 &= f\left(x_i + \frac{1}{2}h, Y_i + \frac{1}{2}hz_2\right) \\
z_4 &= f\left(x_i + h, Y_i + hz_3\right).
\end{aligned}$$

More generally, we can write an implicit RK method in the form

$$Y_{i+1} = Y_i + h\sum_{j=1}^{s} b_j z_j$$

where

$$\begin{aligned}
z_1 &= f\left(x_i, Y_i\right) \\
z_2 &= f\left(x_i + c_2 h, Y_i + h a_{21} z_1\right) \\
z_3 &= f\left(x_i + c_3 h, Y_i + h\left(a_{31} z_1 + a_{32} z_2\right)\right) \\
&\vdots \\
z_s &= f\left(x_i + c_s h, Y_i + h\sum_{k=1}^{s} a_{sk} z_k\right).
\end{aligned}$$

One specifies s, the number of "stages", and the coefficients (a, c).

Runge-Kutta methods can be easily understood using a Butcher tableau:

$$\begin{array}{c|cccc}
c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\
c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\
\vdots & \vdots & & \ddots & \\
c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\
\hline
 & b_1 & b_2 & \cdots & b_s \\
 & b_1^* & b_2^* & \cdots & b_s^*
\end{array}$$

where consistency requires that

$$\sum_{i=1}^{s} b_i = \sum_{i=1}^{s} b_i^* = 1.$$

Consistency means that the local truncation error (the error made from the current step, assuming all the previous steps were exactly correct) goes to zero faster than h as h goes to zero. The global error is the integral over the local truncation errors.

Using the truncation error from a Taylor expansion, we can obtain some restrictions on these coefficients. For example, suppose we want an order $p = 2$ method with two stages; then we must have

$$\begin{aligned} b_1 + b_2 &= 1 \\ b_2 c_2 &= \frac{1}{2} \\ b_2 a_{21} &= \frac{1}{2}. \end{aligned}$$

A frequent condition that is imposed is

$$\sum_{j=1}^{i-1} a_{ij} = c_i$$

$\forall i \in \{2, ..., s\}$, but this condition is neither necessary nor sufficient for consistency. For the trapezoid method (also called the Crank-Nicolson method) with $p = 2$, the Butcher tableau is

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & \frac{1}{2} & \frac{1}{2} \\ \hline & \frac{1}{2} & \frac{1}{2} \\ & 1 & 0 \end{array}.$$

In general, to obtain an order p method we require $s \geq p$ for $p < 5$ and $s \geq p + 1$ for $p \geq 5$; whether these bounds are tight is unknown. For example, while the bound says that we need at least $s = 9$ for $p = 8$, the best method currently known has $s = 11$. Adaptive RK methods use both an order p and an order $p - 1$ method to get an estimate of the error, and then adjust h; for example, the lower order step would be

$$Y_{i+1}^* = Y_i^* + h \sum_{i=1}^{s} b_i^* z_i$$

with the same z_i as the higher-order step. The error estimate is then

$$e_{i+1} = Y_{i+1} - Y_{i+1}^* = h \sum_{i=1}^{s} \left(b_i - b_i^*\right) z_i.$$

If the error is too large, we repeat with a smaller h; if the error is too small, we repeat with a larger h.

RK methods discard information from previous steps each time; that is, the step at i ignores all the information gained from the step at $i-1$. Linear multistep methods attempt to exploit that information to get higher order performance. For example, the Adams-Moulton method with $s = 2$ is

$$Y_{i+1} = Y_i + \frac{h}{2} \left(f\left(x_{i+1}, Y_{i+1}\right) + f\left(x_i, Y_i\right)\right).$$

The general form is

$$\sum_{j=0}^{s} a_j Y_{i+j} = h \sum_{j=0}^{s} b_j f\left(x_{i+j}, Y_{i+j}\right)$$

with $a_s = 1$. The (a, b) coefficients determine the order; the order bound is $s+1$. Consistency requires

$$\sum_{k=0}^{s-1} a_k = -1$$
$$\sum_{k=0}^{s} b_k = s + \sum_{k=0}^{s-1} k a_k.$$

For higher-order ODEs, the leapfrog method is more accurate. Consider an ODE of the form

$$x'' = A\left(x\right)$$

which has a companion form

$$x' = v$$
$$v' = A\left(x\right).$$

The key to the leapfrog method is it updates x and v at interleaved time steps, staggered such that they "leapfrog" each other, yielding a second-order accurate method. The method updates

as

$$\begin{aligned} a_i &= A\left(x_i\right) \\ x_{i+1} &= x_i + v_i \Delta t + \frac{1}{2} a_i \Delta t^2 \\ v_{i+1} &= v_i + \frac{1}{2}\left(a_i + a_{i+1}\right) \Delta t. \end{aligned}$$

If the time steps are not uniform, this algorithm is not stable, so an alternative formulation is used:

$$\begin{aligned} v_{i+1/2} &= v_i + a_i \frac{\Delta t}{2} \\ x_{i+1} &= x_i + v_{i+1/2} \Delta t \\ v_{i+1} &= v_{i+1/2} + a_{i+1} \frac{\Delta t}{2}. \end{aligned}$$

Using Yoshida coefficients, the leapfrog method becomes fourth-order:

$$\begin{aligned} x_i^1 &= x_i + c_1 v_i \Delta t \\ v_i^1 &= v_i + d_1 a\left(x_i^1\right) \Delta t \\ x_i^2 &= x_i^1 + c_2 v_i^1 \Delta t \\ v_i^2 &= v_i^1 + d_2 a\left(x_i^2\right) \Delta t \\ x_i^3 &= x_i^2 + c_3 v_i^2 \Delta t \\ v_i^3 &= v_i^2 + d_3 a\left(x_i^3\right) \Delta t \\ x_{i+1} &= x_i^3 + c_4 v_i^3 \Delta t \\ v_{t+1} &= v_i^3 \end{aligned}$$

with

$$\begin{aligned} w_0 &= -\frac{\sqrt[3]{2}}{2 - \sqrt[3]{2}} \\ w_1 &= \frac{1}{2 - \sqrt[3]{2}} \\ c_1 = c_4 &= \frac{w_1}{2} \\ c_2 = c_3 &= \frac{w_0 + w_1}{2} \\ d_1 = d_3 &= w_1 \\ d_2 &= w_0. \end{aligned}$$

Note that the total step with an iteration is Δt, so the coefficients sum to 1, and some intermediate steps are "backward in time" (c_2 and c_3 are negative).

Most economic problems are not IVPs, however, they are BVPs (boundary value problems). They often have a terminal condition as well, generally on the choice variable y. That is, we usually are trying to solve problems of the form

$$\begin{aligned}
\dot{x} &= f(x, y, t) \\
\dot{y} &= g(x, y, t) \\
x(0) &= x^0 \\
y(T) &= y^T,
\end{aligned}$$

where T may be infinite. We will discuss BVP solution methods below.

Note that some differential equation systems are "stiff", which means that we cannot accurately integrate it without using a very small h; strangely, stiffness is a term that everyone seems to know what it means, but there is no formal definition. The stiffness ratio is given by the ratio of the absolute value of the real part of the largest eigenvalue of the Jacobian to the smallest:

$$\frac{|\mathrm{Re}(\lambda_{\max})|}{|\mathrm{Re}(\lambda_{\min})|}$$

where

$$|\mathrm{Re}(\lambda_{\max})| \geq |\mathrm{Re}(\lambda_t)| \geq |\mathrm{Re}(\lambda_{\min})|;$$

the stiffness ratio is clearly closely related to the conditioning number of a matrix. One useful rule of thumb is that a system is stiff if all eigenvalues have negative real parts and the stiffness ratio is large. What "large" means exactly is not clear.

A useful property for any solution method confronted with a stiff system is A-stability. Consider the simple test system

$$y' = ky$$

with k a complex number and $y(0) = 1$, which has solution

$$y(t) = \exp(kt);$$

this solution converges to 0 as $t \to \infty$ if $\mathrm{Re}(k) < 0$. If the numerical solution also converges to zero for a fixed step size, we call the method A-stable. Explicit RK and multistep methods

are never A-stable; implicit RK methods can be A-stable (for example, the trapezoidal method), whereas implicit multistep methods are only A-stable if their order is no more than 2 (the so-called second Dahlquist barrier).

17.1 Functional Differential Equations

For some economic models, in particular those with gestation lags, the resulting equations are more complicated and fall into the category of functional differential equations. The simplest is called a **retarded functional differential equation**:

$$\dot{x}(t) = f(t, x(t), x(t-\tau)).$$

Note that the past explicitly plays a role in the current evolution of the state; one example would be time-to-build technologies for capital (Kydland and Prescott 1982). A specific example is a **delay differential equation** with a single delay:

$$\dot{x}(t) = f(x(t), x(t-\tau)).$$

Typically one solves these equations using the **method of steps**. Suppose we have the initial condition $\phi : [-\tau, 0] \to \mathcal{R}^n$. The solution on the interval $[0, \tau]$ is given by $\psi(t)$, which solves the inhomogeneous IVP

$$\dot{\psi}(t) = f(\psi(t), \phi(t-\tau))$$

with initial condition $\psi(0) = 0$; this equation is solved using a standard ODE solver. We then advance to the next interval $[\tau, 2\tau]$, using our solution as the inhomogeneous part (that is, it replaces ϕ).

For example, suppose

$$f(x(t), x(t-\tau)) = \alpha x(t-\tau)$$

and $\phi(t) = 1$. The IVP is easy to solve using integration:

$$\begin{aligned} x(t) &= x(0) + \int_{s=0}^{t} \frac{d}{dt} x(s)\, ds = 1 + a \int_{s=0}^{t} \phi(s)\, ds \\ &= 1 + at \end{aligned}$$

since $x(0) = \phi(0) = 1$. On the interval $[\tau, 2\tau]$, we integrate again and impose the initial condition $x(\tau)$:

$$\begin{aligned} x(t) &= x(\tau) + \int_{s=\tau}^{t} \frac{d}{dt} x(s)\, ds = a\tau + 1 + a\int_{s=\tau}^{t} (a(s-\tau)+1)\, ds \\ &= 1 + a\tau + \int_{s=0}^{t-\tau} (as+1)\, ds \\ &= 1 + a\tau + a(t-\tau)\left(\frac{a(t-\tau)}{2} + 1\right). \end{aligned}$$

Stability analysis of delay differential equations is substantially more complicated. Consider the (linearized) multiple DDE

$$\dot{x}(t) = A_0 x(t) + A_1 x(t-\tau_1) + ... + A_m x(t-\tau_m).$$

The eigenvalues solve the equation

$$\det\left(-\lambda I + A_0 + A_1 e^{-\tau_1 \lambda} + ... + A_m e^{-\tau_m \lambda}\right) = 0.$$

There are infinitely-many roots λ to this equation, but we know some sufficiency conditions for stability. First, if every root has negative real part, then the system is stable. Another, more complicated, condition, involves the logarithmic matrix norm

$$\mu(W) = \lim_{\Delta \to 0} \left\{ \frac{\|I + \Delta W\| - 1}{\Delta} \right\}.$$

Then if

$$\mu(A_0) + \sum_{k=1}^{m} \|A_k\| < 0,$$

the system is stable. We do not know what happens if the condition is not satisfied; the system may be stable or unstable.

The following time-to-build model requires us to solve a DDE:

$$\max_{c(t)} \left\{ \int_0^\infty \frac{c(t)^{1-\sigma}}{1-\sigma} \exp(-\rho t)\, dt \right\}$$

subject to

$$\begin{aligned} \dot{k}(t) &= Ak(t-d)^\alpha - \delta k(t-d) - c(t) \\ k(t) &= k_0(t) \ \ \forall t \in [-d, 0]. \end{aligned}$$

Capital purchased at instant t will not be useful for production until instant $t+d$, where d is a fixed delay. Note that this structure is a special case of Kydland and Prescott (1982), who assume that investment is paid for in installments with a fixed distribution of weights $(\phi_i)_{i=1}^{I}$; here, the entirement investment is purchased at t, so $\phi_1 = 1$ and $\phi_{i>1} = 0$. We could accommodate the distributed lag structure by allowing for multiple delays.

The Euler equations are

$$\begin{aligned} c(t)^{-\sigma} \exp(-\rho t) &= \lambda(t) \\ -\lambda(t+d)\left(\alpha A k(t)^{\alpha-1} - \delta\right) &= \dot{\lambda}(t) \end{aligned}$$

which can be combined into

$$\frac{\dot{c}(t)}{c(t)} = \frac{1}{\sigma}\left(\left(\alpha A k(t)^{\alpha-1} - \delta\right)\left(\frac{c(t)}{c(t+d)}\right)^{\sigma} \exp(-\rho d) - \rho\right).$$

The steady state is

$$\begin{aligned} k^* &= \left(\frac{\alpha A}{\rho \exp(\rho d) + \delta}\right)^{\frac{1}{1-\alpha}} \\ c^* &= A(k^*)^{\alpha} - \delta k^*. \end{aligned}$$

To solve this model, first pick an initial consumption prediction function $\widetilde{c}_0(t+d)$ and time span $[0,T]$ where $T = S \times d$. Choose a bracket for initial consumption $[c_l, c_h]$, set counter to $l = 0$, and fix stopping criteria $\varepsilon^e > 0$ and $\varepsilon^s > 0$.

1. Given $\widetilde{c}_l(t)$, use a shooting algorithm to compute the implied path $(c(t), k(t))$:

   (a) Set $c_0 = \frac{c_l + c_h}{2}$

   (b) Given c_0 and $\widetilde{c}_l(t)$, solve dynamic system using method of steps:

      i. Set $k(t) = k_0(t)$ for $t \in [-d, 0]$ and $i = 0$

      ii. Given $k_i(t)$, $t \in [(i-1)d, id]$, $\widetilde{c}_l(t)$, and c_0 solve

$$\begin{aligned} \dot{k}(t) &= A k_i(t-d)^{\alpha} - c(t) - \delta k_i(t-d) \\ \dot{c}(t) &= \frac{c(t)}{\sigma}\left(\exp(-\rho d)\left(\alpha A k(t)^{\alpha-1} - \delta\right)\left(\frac{c(t)}{\widetilde{c}_l(t+d)}\right)^{\sigma} - \rho\right) \end{aligned}$$

   and set $k_i(t) = k(t)$ for $t \in [id, (i+1)d]$.

iii. Set $i = i + 1$ and repeat (i)-(iii) until $i = S$

(c) If $|c(t) - c^*| < \varepsilon^s$ for $t \in ((S-1)d, Sd)$ stop, else

i. if $c(t) > c^*$ for $t \in ((S-1)d, Sd)$, set $c_H = c_0$ and return to step 1

ii. if $c(t) < c^*$ for $t \in ((S-1)d, Sd)$, set $c_L = c_0$ and return to step 1

2. Update $\widetilde{c}$:

(a) Use a functional approximator $c(t)$ to obtain a new $\widetilde{c}_1(t)$, increment l, and return to step 1 until $\|c(t) - \widetilde{c}(t)\| < \varepsilon^e$. One option is a Chebyshev polynomial

$$\widetilde{c}_l(t+d) = \exp\left(\sum_{i=0}^{n} \theta_{i,l} T_i(\psi(t+d))\right)$$

where

$$\psi(t) = 2\frac{t}{Sd} - 1$$

adapts the time interval to $[-1, 1]$. The update is then

$$\theta_{l+1} = \lambda\theta_l + (1-\lambda)\widehat{\theta}_l$$

where

$$\widehat{\theta}_l = \arg\min_{\theta} \left\{ \left\| \log(c(t)) - \sum_{i=0}^{n} \theta_i T_i(\psi(t+d)) \right\|^2 \right\}.$$

Another option is to simply use the entire path

$$\widetilde{c}_{l+1}(t) = \lambda\widetilde{c}_l(t) + (1-\lambda)c_l(t).$$

A third option is to fit a spline to the path.

A related class of problems involve capital with "one-hoss shay" depreciation – the capital stock does not depreciate for a fixed time period, then completely depreciates, and capital stocks of different ages are perfect substitutes. The capital stock at instant t is

$$k(t) = \int_{t-\tau}^{t} \int_{s}^{s+\tau} i(k(\upsilon))\, d\upsilon ds,$$

the sum of investments over the lifetime of the capital stock (note that the investment decision depends on the capital stock at each instant). These problems are called mixed-type functional

differential equations, because they mix delay and advance differential equations together; they also arise in continuous-time overlapping generations models. Related models of vintage capital, in which the distribution of capital ages is relevant because they depreciate at different rates, also give rise to mixed-type FDEs.

17.2 Stochastic Calculus

Adding stochastic elements to optimal control problems leads us into the field of stochastic calculus. There are a number of advantages to writing models in continuous time – they are often nonstochastic and sparse (which lessens the curse of dimensionality), first-order conditions hold even with constraints and discrete choices, and effectively everything is normal and quadratic (there is a theorem that if process is continuous in time and has iid increments it is normal, and normal random variables are quadratic since their behavior is described by the first two moments).

Consider the problem

$$v\left(x\left(0\right)\right)=\max_{x(t),y(t)}\left\{E\left[\int_{0}^{\infty}e^{-\rho t}f\left(x\left(t\right),y\left(t\right)\right)dt\right]\right\}$$

subject to

$$\begin{aligned}dx\left(t\right)&=g\left(x\left(t\right),y\left(t\right)\right)dt+\sigma\left(x\left(t\right)\right)dW\left(t\right)\\x\left(0\right)&=x_{0}.\end{aligned}$$

The evolution of x is called a **controlled diffusion**, and $dW\left(t\right)$ is a **Brownian motion** (also called a Wiener process); Brownian motions are time-continuous with stationary, independent normal increments.

Definition 127 *A **Wiener process** W_t is a stochastic process such that (i) $W_0=0$; (ii) W has independent increments ($\forall t>0$, $W_{t+u}-W_t$, $u\geq 0$, are independent of past W_s $\forall s\leq t$); (iii) W has Gaussian increments ($W_{t+u}-W_t$ is normally distributed with mean 0 and variance u; (iv) W is continuous in t.*

Another way to describe a Wiener process is the following, which can be useful for understanding their behavior.

Definition 128 *(Lévy) A Wiener process W_t is an almost surely continuous martingale with $W_0 = 0$ and quadratic variation ($W^2 - t$ is also a martingale).*

Donsker's theorem shows that the Wiener process is the limit of a random walk. Let $\{\xi_t\}$ be iid variables with mean 0 and unit variance. For each n, define the stochastic process

$$W_n(t) = \frac{1}{\sqrt{n}} \sum_{1 \leq k \leq \lfloor nt \rfloor} \xi_k$$

for $t \in [0,1]$. For large n, $W_n(t) - W_n(s)$ is distributed $N(0, t-s)$ by the central limit theorem.

Wiener processes have mean 0:

$$E[W_t] = 0.$$

Their variance scales with t:

$$Var[W_t] = t;$$

Wiener processes are therefore said to display quadratic variation in time. The correlation between W_t and W_s ($s \leq t$) is

$$Corr(W_s, W_t) = \sqrt{\frac{s}{t}}.$$

Thus,

$$W_s = W_t + \sqrt{s-t}Z$$

for a standard normal Z. A peculiar property of Wiener processes is that they have unbounded variation on every interval. They are also scale-invariant (fractal).

If we allow that the increments are non-Gaussian (but still stationary and independent), we get a **Lévy process**; one can decompose any Lévy process into a drift, a Brownian motion, and a jump process (the sum of independent Poisson random variables). The moments of a Lévy process $\mu_n(t) = E[X_t^n]$ satisfy the recursion

$$\mu_n(t+s) = \sum_{k=0}^{n} \binom{n}{k} \mu_k(t) \mu_{n-k}(s),$$

provided they are finite. Many Lévy processes do not have finite moments, however, and only the Wiener process has continuous time paths (the jumps are zero).

Our goal is derive the analogue to the Bellman equation for continuous time. Take a small interval of time Δt:

$$v(x(0)) \approx \max_{x,y} \left\{ f(x(0), y(0)) \Delta t + \frac{1}{1+\rho\Delta t} E[v(x(0+\Delta t))] \right\}.$$

Multiply by $1+\rho\Delta t$, divide by Δt, and rearrange:

$$\rho v\left(x\left(0\right)\right) \approx \max_{x,y}\left\{ f\left(x\left(0\right),y\left(0\right)\right)\left(1+\rho\Delta t\right)+\frac{1}{\Delta t}E\left[\Delta v\right]\right\}.$$

Take the limit as $\Delta t \to 0$:

$$\rho v\left(x\left(0\right)\right) = \max_{x,y}\left\{ f\left(x\left(0\right),y\left(0\right)\right)+\frac{1}{dt}E\left[dv\right]\right\}.$$

To compute the expectation we need to use Itô's lemma.

Lemma 129 *(Itô's Lemma) Suppose X_t follows an Itô drift-diffusion process*

$$dX_t = \mu_t dt + \sigma_t dB_t$$

where B_t is a Weiner process, μ_t is the drift process, and σ_t is the diffusion process. Then for any twice-differentiable function $f\left(t,x\right)$, we have

$$df\left(t,X_t\right) = \left(\frac{\partial f}{\partial t}+\mu_t\frac{\partial f}{\partial x}+\frac{\sigma_t^2}{2}\frac{\partial^2 f}{\partial x^2}\right)dt+\sigma_t\frac{\partial f}{\partial x}dB_t.$$

That is, $f\left(t,x\right)$ is itself an Itô drift-diffusion process with drift $\frac{\partial f}{\partial t}+\mu_t\frac{\partial f}{\partial x}+\frac{\sigma_t^2}{2}\frac{\partial^2 f}{\partial x^2}$ and diffusion $\sigma_t\frac{\partial f}{\partial x}$.

Itô's lemma can be informally proven by taking a Taylor expansion, dropping the dt^2 and $dtdB_t$ terms, and replacing dB_t^2 with dt since the standard deviation of a Weiner process is linear in time. Itô's lemma can be used to derive the Black-Scholes pricing equation. To do so, suppose that the stock price follows a geometric Brownian motion

$$\frac{dS_t}{S_t} = \mu dt + \sigma dB_t.$$

Let the option value by $f\left(t,S_t\right)$; then by Itô's lemma we have

$$df\left(t,S_t\right) = \left(\frac{\partial f}{\partial t}+\frac{1}{2}\left(S_t\sigma\right)^2\frac{\partial^2 f}{\partial S^2}\right)dt+\frac{\partial f}{\partial S}dS_t.$$

Suppose that the value of cash grows at instantaneous rate r and the agent follows a strategy of holding $\frac{\partial f}{\partial S}$ units of stock at any instant. The value of the portfolio is then given by

$$dV_t = r\left(V_t-\frac{\partial f}{\partial S}S_t\right)dt+\frac{\partial f}{\partial S}dS_t;$$

if $V_t = f(t, S_t)$ then we replicate the option. Combining the equations gives us Black-Scholes:

$$\frac{\partial f}{\partial t} + \frac{\sigma^2 S_t^2}{2}\frac{\partial^2 f}{\partial S^2} + rS_t\frac{\partial f}{\partial S} - rf = 0.$$

In the controlled diffusion case, Itô's lemma implies

$$\frac{1}{dt}E[dv] = gv' + \frac{1}{2}\sigma^2 v'';$$

inserting this expression into the dynamic functional equation we get the Hamilton-Jacobi-Bellman (HJB) equation

$$\rho v(x) = \max_{x,y}\left\{f(x,y) + g(x,y)\,Dv(x) + \frac{1}{2}\sigma^2(x)\,D^2 v(x)\right\}.$$

Note that this equation is not stochastic. However, the deterministic case is not equivalent:

$$\rho v(x) = \max_{x,y}\left\{f(x,y) + g(x,y)\,Dv(x)\right\};$$

the extra term involves the diffusion process of the Brownian motion. Concentrating the decision variable out we get the Feynman-Kac formula:

$$\rho v(x) = f(x, y(x)) + g(x, y(x))\,Dv(x) + \frac{1}{2}\sigma^2(x)\,D^2 v(x).$$

Given $y(x)$, this equation defines the value function $v(x)$.

The foc for the HJB equation is

$$D_y f(x,y) + D_y g(x,y)\,Dv(x) = 0$$

and the envelope condition is

$$(\rho - D_x g(x,y))\,Dv(x) - D_x f(x,y) = g(x,y)\,D^2 v(x) + \frac{1}{2}\sigma^2(x)\,D^3 v(x) + \sigma(x)\,D\sigma(x)\,D^2 v(x).$$

These two equations determine the functions $(v, y)(x)$.

Suppose instead there is a Poisson process with two states (z_1, z_2) and intensities λ_i; that is, if we are in state i at this instant then with arrival rate λ_i we "switch" to state j. Then the HJB equation is

$$\rho v_i(x) = \max_{x,y}\left\{f(x,y,z_i) + g(x,y,z_i)\,Dv_i(x) + \lambda_i(v_j(x) - v_i(x))\right\}.$$

If we have both a Poisson process and a diffusion, the HJB equation is

$$\rho v_i(x) = \max_{x,y} \left\{ f(x,y,z_i) + g(x,y,z_i) Dv_i(x) + \frac{1}{2}\sigma^2(x) D^2 v_i(x) + \lambda_i (v_j(x) - v_i(x)) \right\},$$

which is a simple example of a jump diffusion process with constant jump size. A more general jump diffusion process has random jump sizes:

$$dx(t) = g(x(t), y(t))\, dt + \sigma(x(t))\, dW(t) + \lambda(x(t)) \int_a^b \eta \phi(\eta)\, d\eta;$$

in this case the HJB equation becomes

$$\rho v(x) = \max_{x,y} \left\{ f(x,y) + g(x,y) Dv_i(x) + \frac{1}{2}\sigma^2(x) D^2 v(x) + \lambda(x) \left(\int_a^b v(x+\eta)\phi(\eta)\, d\eta - v(x) \right) \right\}.$$

We would need a procedure for computing the integral; I suspect that some kind of quadrature would work here.

Let's do a simple example: the stochastic growth model. The continuous-time planning problem is

$$\max_{\{c(t),k(t)\}} \left\{ E_0 \left[\int_0^\infty e^{-\rho t} u(c(t))\, dt \right] : \dot{k}(t) = f(z(t), k(t)) - \delta k(t) - c(t), dz(t) = -\lambda z(t)\, dt + \sigma z(t)\, dW(t) \right\}.$$

The HJB equation is

$$\rho v(k,z) = \max_c \left\{ u(c) + \frac{\partial}{\partial k} v(k,z) (f(k,z) - \delta k - c) - \lambda z \frac{\partial}{\partial z} v(k,z) + \frac{1}{2} (\sigma z)^2 \frac{\partial^2}{\partial z^2} v(k,z) \right\}.$$

The invariant distribution over states g satisfies

$$\begin{aligned} 0 &= -\frac{\partial}{\partial k} \left((f(z,k) - \delta k - c(k,z)) g(k,z) \right) - \lambda \frac{\partial}{\partial z} g(k,z) + \frac{\sigma^2}{2} \frac{\partial^2}{\partial z^2} g(k,z) \\ 1 &= \int_0^\infty \int_{-\infty}^\infty g(k,z)\, dk dz; \end{aligned}$$

the interpretation of g is the fraction of time that the model "spends" in state (k,z), technically which needs to be evaluated as a limit over shrinking Borel sets over the (k,z) space.

Another example is the partial equilibrium Bewley model. Suppose we have income as a two-state Poisson arrival process. The household problem is

$$\begin{aligned} \rho v_1(a) &= \max_c \{ u(c) + Dv_1(a)(z_1 + ra - c) + \lambda_1 (v_2(a) - v_1(a)) \} \\ \rho v_2(a) &= \max_c \{ u(c) + Dv_2(a)(z_2 + ra - c) + \lambda_2 (v_1(a) - v_2(a)) \} \end{aligned}$$

and the invariant distribution g satisfies

$$\begin{aligned}
0 &= -\frac{d}{da}\left(s_1\left(a\right)g_1\left(a\right)\right) - \lambda_1 g_1\left(a\right) + \lambda_2 g_2\left(a\right) \\
0 &= -\frac{d}{da}\left(s_2\left(a\right)g_2\left(a\right)\right) - \lambda_2 g_2\left(a\right) + \lambda_1 g_1\left(a\right) \\
1 &= \int_{\underline{a}}^{\infty} g_1\left(a\right)da + \int_{\underline{a}}^{\infty} g_2\left(a\right)da \\
0 &= \int_{\underline{a}}^{\infty} a g_1\left(a\right)da + \int_{\underline{a}}^{\infty} a g_2\left(a\right)da.
\end{aligned}$$

Define the savings functions as

$$\begin{aligned}
s_1\left(a\right) &= z_1 + ra - c \\
s_2\left(a\right) &= z_2 + ra - c
\end{aligned}$$

and note that the first-order conditions are

$$c_j\left(a\right) = \left(Du\right)^{-1}\left(v_j\left(a\right)\right).$$

One nice example of the value of continuous time can be found in Maxted (2020), in which households have quasi-geometric discounting and "instantaneous gratification"; here, the usual time rate of preference is ρ, but the first-order condition is solved using an extra discount $\beta \neq 1$:

$$\begin{aligned}
\rho v_j\left(a\right) &= u\left(c\right) + Dv_j\left(a\right)\left(y_j + ra - c\right) + \lambda_j\left(v_{-j}\left(a\right) - v_j\right)\left(a\right) \\
Du\left(c\right) &= \beta Dv_j\left(a\right).
\end{aligned}$$

We will see that one can change the agent into a "normal one" with a different discount factor; in discrete time, we have already seen the complications induced by deviating away from geometric discounting, so being able to solve an equivalent but geometric model is extremely convenient.

17.3 Solving the HJB Equation

We are going to use a "finite-difference" method to solve the HJB equation – for concreteness we will do the Bewley example with Poisson arrivals first. Suppose there are I nodes in the asset space; we can approximate the derivatives (forward and backward) as

$$\begin{aligned}
\left(Dv\right)_{i,j,F} &= Dv_j\left(a_i\right) = \frac{v_{i+1,j} - v_{i,j}}{\Delta a} \\
\left(Dv\right)_{i,j,B} &= Dv_j\left(a_i\right) = \frac{v_{i-1,j} - v_{i,j}}{\Delta a}
\end{aligned}$$

and note that the borrowing constraint $a \geq \underline{a}$ imposes a "state constraint" that takes the form

$$Dv_j(\underline{a}) \geq Du(z_j + r\underline{a}).$$

This constraint prevents a from falling at $\underline{a}$. Note that this constraint looks backward from the discrete time version – it says that the marginal value of wealth exceeds the marginal utility of consumption, rather than the opposite from a discrete time model where

$$\beta E[Dv(\underline{a})] \leq Du(z_j + r\underline{a})$$

would hold. The difference arises from the fact that the a in the continuous time model is the same on both sides of the inequality, whereas in discrete time the LHS value is a', not a; that is, it is the value of assets some discrete amount of time into the future.

There are two options for constructing the solution, an explicit and an implicit method. These are the same options that give rise to Newton vs Runge-Kutta ODE solvers.

17.3.1 Explicit method

Make an initial guess $v_j^0(a)$. Then update that guess as

$$\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta} + \rho v_{i,j}^n = u\left(c_{i,j}^n\right) + \left(Dv_{i,j}^n\right)\left(z_j + ra_i - c_{i,j}^n\right) + \lambda_j\left(v_{i,-j}^n - v_{i,j}^n\right)$$
$$c_{i,j}^n = (Du)^{-1}\left(Dv_{i,j}^n\right)$$

for fixed step size Δ. Repeat to convergence. The explicit method will generally only work for small time steps Δ but does not require us to solve any equations. Nevertheless, it is generally dominated by the implicit method, which works for any time step Δ but may require us to solve equations; fortunately, for many problems the equations are linear.

17.3.2 Implicit method

Make an initial guess $v_j^0(a)$. Then update that guess as

$$\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta} + \rho v_{i,j}^{n+1} = u\left(c_{i,j}^n\right) + \left(Dv_{i,j}^{n+1}\right)\left(z_j + ra_i - c_{i,j}^n\right) + \lambda_j\left(v_{i,-j}^{n+1} - v_{i,j}^{n+1}\right)$$
$$c_{i,j}^n = (Du)^{-1}\left(Dv_{i,j}^n\right).$$

To ensur stability for any Δ we will use an upwind scheme, in which the derivative is approximated "in the direction" of the drift of the state a; that is, if a is rising (s is positive) we use Dv_F, if a is falling (s is negative) we use Dv_B, and if a is constant ($s = 0$) we set the derivative using the value of constant consumption. Why use such a complicated procedure instead of simply approximating the derivative using forward, backward, or central finite differences? Because we want to satisfy the Barles-Souganides condition. Define

$$S\left(\Delta a, a_i, v_i; v_{i-1}, v_{i+1}\right)$$

as the finite difference approximation of the HJB equation and

$$G\left(v\left(a\right), Dv\left(a\right), D^2v\left(a\right)\right)$$

as the true HJB equation. Barles-Souganides show that a numerical scheme converges uniformly to the unique solution to the HJB equation if it is (i) monotone (S is non-increasing in both v_{i-1} and v_{i+1}); (ii) consistent ($S \to G$ as $\Delta a \to 0$ and $a_i \to a$); and (iii) stable ($\forall \Delta a > 0$ it has a solution v_i that is uniformly bounded independent of Δa). It is not hard to see that using the backward difference if the state is rising leads to a failure of monotonicity, as does the inverse, and a central difference approach would therefore always fail. Note that the convergence is not about the algorithm converging to the fixed point given an approximation scheme, but about the approximation converging to the true function as Δa goes to zero. That $v^n \to v$ is a consequence of the Contraction Mapping Theorem and occurs no matter the approximation scheme (in principle).

To derive the solution to the finite difference approximation, start with the upwind approximation:

$$\begin{aligned}\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta} + \rho v_{i,j}^{n+1} = u\left(c_{i,j}^n\right) + \left(Dv_{i,j,F}^{n+1}\right)\max\left\{z_j + ra_i - c_{i,j,F}^n, 0\right\} + \\ \left(Dv_{i,j,B}^{n+1}\right)\min\left\{z_j + ra_i - c_{i,j,B}^n, 0\right\} + \lambda_j\left(v_{i,-j}^{n+1} - v_{i,j}^{n+1}\right) \\ c_{i,j}^n = \left(Du\right)^{-1}\left(Dv_{i,j}^n\right).\end{aligned}$$

The derivative is therefore approximated as

$$\begin{aligned}\left(Dv_{i,j}^n\right) &= \left(Dv_{i,j,F}^n\right)1\left(s_{i,j,F} > 0\right) + \left(Dv_{i,j,B}^n\right)1\left(s_{i,j,B} < 0\right) + \left(D\overline{v}_{i,j}^n\right)1\left(s_{i,j,F} \leq 0 \leq s_{i,j,B}\right) \\ \left(D\overline{v}_{i,j}^n\right) &= Du\left(z_j + ra_i\right).\end{aligned}$$

Collecting terms we obtain

$$\frac{v_{i,j}^{n+1}-v_{i,j}^{n}}{\Delta}+\rho v_{i,j}^{n+1}=u\left(c_{i,j}^{n}\right)+v_{i-1,j}^{n+1}x_{i,j}+v_{i,j}^{n+1}y_{i,j}+v_{i+1,j}^{n+1}z_{i,j}+v_{i,-j}^{n+1}\lambda_{j}$$

$$x_{i,j}=-\frac{\min\left\{s_{i,j,B}^{n},0\right\}}{\Delta a}$$

$$y_{i,j}=-\frac{\max\left\{s_{i,j,F}^{n},0\right\}}{\Delta a}+\frac{\min\left\{s_{i,j,B},0\right\}}{\Delta a}$$

$$z_{i,j}=\frac{\max\left\{s_{i,j,F}^{n},0\right\}}{\Delta a}$$

and at the boundaries we have

$$x_{1,j}=z_{I,j}=0.$$

This equation can be written compactly and solved:

$$\frac{1}{\Delta}\left(v^{n+1}-v^{n}\right)+\rho v^{n+1}=u^{n}+\left(A^{n}+C\right)v^{n+1}$$

$$v^{n+1}=\left(\left(\frac{1}{\Delta}+\rho\right)I-A^{n}-C\right)^{-1}\left(u^{n}+\frac{1}{\Delta}v^{n}\right).$$

Now just repeat to convergence. The implicit scheme works independently of the value of Δ, even in the limit as $\Delta\to\infty$:

$$\rho v^{n+1}=u^{n}+\left(A^{n}+C\right)v^{n+1}$$

$$v^{n+1}=\left(\rho I-A^{n}-C\right)^{-1}u^{n}.$$

It is generally better not to use $\Delta=\infty$ however, because numerical errors may lead to instability; $\Delta=1000$ is an acceptably fast choice in the growth model and requires only a few iterations to converge. The tradeoff is that we have to solve a linear system of equations, which could be difficult if the number of points is large. Fortunately, A and C are sparse, and therefore we can use a large number of grid points without significantly affecting computational time. For a

simple case $I = 4$ we get

$$A^n = \begin{bmatrix} y_{1,1} & z_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 \\ x_{2,1} & y_{2,1} & z_{2,1} & 0 & 0 & 0 & 0 & 0 \\ 0 & x_{3,1} & y_{3,1} & z_{3,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & x_{4,1} & y_{4,1} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_{1,2} & z_{1,2} & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{2,2} & y_{2,2} & z_{2,2} & 0 \\ 0 & 0 & 0 & 0 & 0 & x_{3,2} & y_{3,2} & z_{3,2} \\ 0 & 0 & 0 & 0 & 0 & 0 & x_{4,2} & y_{4,2} \end{bmatrix}$$

$$C = \begin{bmatrix} -\lambda_1 & 0 & 0 & 0 & \lambda_1 & 0 & 0 & 0 \\ 0 & -\lambda_1 & 0 & 0 & 0 & \lambda_1 & 0 & 0 \\ 0 & 0 & -\lambda_1 & 0 & 0 & 0 & \lambda_1 & 0 \\ 0 & 0 & 0 & -\lambda_1 & 0 & 0 & 0 & \lambda_1 \\ \lambda_2 & 0 & 0 & 0 & -\lambda_2 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & -\lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 & 0 & 0 & -\lambda_2 & 0 \\ 0 & 0 & 0 & \lambda_2 & 0 & 0 & 0 & -\lambda_2 \end{bmatrix}.$$

Note that most entries are zero and the matrices are banded; in continuous time the process can only go to adjacent values of a, which generates the sparseness (it cannot go to non-adjacent values of a because it must be continuous in time and therefore "pass through" the adjacent values first, and we are implicitly assuming the time steps are small enough that it cannot get all the way through the next interval).

17.4 Computing the Distribution

To get the distribution g, start with the evolution equation and adding up constraint

$$\begin{aligned} 0 &= -\left[s_{i,j} g_{i,j}\right]' - \lambda_j g_{i,j} + \lambda_{-j} g_{i,j} \\ 1 &= \sum_{i=1}^{I} g_{i,1} + \sum_{i=1}^{I} g_{i,2}. \end{aligned}$$

The upwind scheme for approximating g' is

$$-\frac{\max\left\{s_{i,j,F}^{n},0\right\}g_{i,j}-\max\left\{s_{i-1,j,F}^{n},0\right\}g_{i-1,j}}{\Delta a}-\frac{\min\left\{s_{i+1,j,B}^{n},0\right\}g_{i+1,j}-\min\left\{s_{i,j,B}^{n},0\right\}g_{i,j}}{\Delta a}-g_{i,j}\lambda_{j}+g_{i,-j}\lambda_{-j}=0$$

which can be written as

$$A^{T}g=0.$$

Note that A is the same matrix that we got from solving the HJB equation (it describes the evolution of a). g is therefore an eigenvector of A associated with a zero eigenvalue. To solve, fix

$$g_{i,j}=0.1$$

for some arbitrary (i,j) by replacing the (i,j) element of the zero vector with 0.1; if we don't do this normalization A will not be invertible. Solve the system for $\widetilde{g}$ and renormalize

$$g_{i,j}=\frac{\widetilde{g}_{i,j}}{\sum_{i=1}^{I}\widetilde{g}_{i,1}+\sum_{i=1}^{I}\widetilde{g}_{i,2}};$$

this last step imposes the adding up condition on g. Note that you need to be sure that you normalize an entry (i,j) that will have positive measure.

If we want to impose market clearing, we can approximate total savings as

$$S\left(r\right)=\sum_{i=1}^{I}a_{i}g_{i,1}+\sum_{i=1}^{I}a_{i}g_{i,2}$$

and adjust r until $S\left(r\right)=0$ or $MPK\left(S\left(r\right)\right)-r=0$, depending on how we interpret a (bonds or capital).

If we want transitions, then we solve a time-dependent problem. Suppose the transition takes T "periods" and we discretize each period into n "steps"; the total number of time steps is then $N=Tn$ and each step is of size $dt=\frac{N}{T}=n$. The HJB equation at time t is

$$\rho v_{t,1}\left(a\right)=\max_{c}\left\{u\left(c\right)+v_{t,1}'\left(a\right)\left(z_{1}+ra-c\right)+\lambda_{1}\left(v_{t,2}\left(a\right)-v_{t,1}\left(a\right)\right)\right\}$$

$$\rho v_{t,2}\left(a\right)=\max_{c}\left\{u\left(c\right)+v_{t,2}'\left(a\right)\left(z_{2}+ra-c\right)+\lambda_{2}\left(v_{t,1}\left(a\right)-v_{t,2}\left(a\right)\right)\right\}$$

and the the distribution evolves according to

$$dg_{t+1,1}\left(a\right)=-\frac{d}{da}\left(s_{t,1}\left(a\right)g_{t,1}\left(a\right)\right)-\lambda_{1}g_{t,1}\left(a\right)+\lambda_{2}g_{t,2}\left(a\right)$$

$$dg_{t+1,2}\left(a\right)=-\frac{d}{da}\left(s_{t,2}\left(a\right)g_{t,2}\left(a\right)\right)-\lambda_{2}g_{t,2}\left(a\right)+\lambda_{1}g_{t,1}\left(a\right).$$

17.4.1 Alternative Processes

Generalizing the process for z to a diffusion

$$dz_t = \mu(z_t)\,dt + \sigma(z_t)\,dW_t$$

where W_t is a Brownian motion (Wiener process) so that

$$W_s - W_t \sim N(0, t-s).$$

The equilibrium is described by

$$\begin{aligned}
\rho v(a) &= \max_c \left\{ u(c) + \partial_a v(a,z)(z + ra - c) + \mu(z)\partial_z v(a,z) + \frac{\sigma^2(z)}{2}\partial_{zz} v(a,z) \right\} \\
0 &= -\partial_a\left(s(a,z)\,g(a,z)\right) - \partial_z\left(\mu(z)\,g(a,z)\right) + \frac{1}{2}\partial_{zz}\left(\sigma^2(z)\,g(a,z)\right) \\
1 &= \int_0^\infty \int_{\underline{a}}^\infty g(a,z)\,da\,dz \\
0 &= \int_0^\infty \int_{\underline{a}}^\infty a\,g(a,z)\,da\,dz
\end{aligned}$$

along with boundary conditions (z is reflected at $\underline{z}$ and $\overline{z}$ to keep it bounded):

$$\begin{aligned}
0 &= \partial_z v(a,\underline{z}) = \partial_z v(a,\overline{z}) \\
\partial_a v(\underline{a},z) &\geq u'(z + r\underline{a}).
\end{aligned}$$

Implicit scheme, upwind method:

$$\begin{aligned}
\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta} + \rho v_{i,j}^{n+1} &= u\left(c_{i,j}^n\right) + \partial_a v_{i,j}^{n+1}\left(z_j + ra_i - c_{i,j}^n\right) + \mu_j \partial_z v_{i,j}^{n+1} + \frac{\sigma_j^2}{2}\partial_{zz} v_{i,j}^{n+1} \\
\frac{v_{i,j}^{n+1} - v_{i,j}^n}{\Delta} + \rho v_{i,j}^{n+1} &= u\left(c_{i,j}^n\right) + \frac{v_{i+1,j}^{n+1} - v_{i,j}^{n+1}}{da}\left(z_j + ra_i - c_{i,j}^n\right)_+ + \\
&\quad \frac{v_{i,j}^{n+1} - v_{i-1,j}^{n+1}}{da}\left(z_j + ra_i - c_{i,j}^n\right)_- + \mu_j^+ \frac{v_{i,j+1}^{n+1} - v_{i,j}^{n+1}}{dz} + \\
&\quad \mu_j^- \frac{v_{i,j}^{n+1} - v_{i,j-1}^{n+1}}{dz} + \frac{\sigma_j^2}{2}\frac{v_{i,j+1}^{n+1} - 2v_{i,j}^{n+1} + v_{i,j-1}^{n+1}}{(dz)^2}
\end{aligned}$$

Collecting terms we can get

$$\frac{v_{i,j}^{n+1}-v_{i,j}^{n}}{\Delta}+\rho v_{i,j}^{n+1}=u\left(c_{i,j}^{n}\right)+v_{i-1,j}^{n+1}x_{i,j}+v_{i,j}^{n+1}\left(y_{i,j}+\upsilon_{j}\right)+v_{i+1,j}^{n+1}z_{i,j}+v_{i,j-1}^{n+1}\chi_{j}+v_{i,j+1}^{n+1}\zeta_{j}$$

$$\begin{aligned}
x_{i,j} &= -\frac{\min\left\{s_{i,j,B}^{n},0\right\}}{\Delta a}\\
y_{i,j} &= -\frac{\max\left\{s_{i,j,F}^{n},0\right\}}{\Delta a}+\frac{\min\left\{s_{i,j,B}^{n},0\right\}}{\Delta a}\\
z_{i,j} &= \frac{\max\left\{s_{i,j,F}^{n},0\right\}}{\Delta a}\\
\chi_{j} &= -\frac{\min\left\{\mu_{j},0\right\}}{\Delta z}+\frac{\sigma_{j}^{2}}{2\left(\Delta z\right)^{2}}\\
\upsilon_{j} &= \frac{\min\left\{\mu_{j},0\right\}}{\Delta z}-\frac{\max\left\{\mu_{j},0\right\}}{\Delta z}-\frac{\sigma_{j}^{2}}{\left(\Delta z\right)^{2}}\\
\zeta_{j} &= \frac{\max\left\{\mu_{j},0\right\}}{\Delta z}+\frac{\sigma_{j}^{2}}{2\left(\Delta z\right)^{2}}
\end{aligned}$$

and at the boundaries

$$\begin{aligned}
\frac{v_{i,1}^{n+1}-v_{i,1}^{n}}{\Delta}+\rho v_{i,1}^{n+1} &= u\left(c_{i,1}^{n}\right)+v_{i-1,1}^{n+1}x_{i,1}+v_{i,1}^{n+1}\left(y_{i,1}+\upsilon_{1}+\chi_{1}\right)+v_{i+1,1}^{n+1}z_{i,1}+v_{i,2}^{n+1}\zeta_{1}\\
\frac{v_{i,J}^{n+1}-v_{i,J}^{n}}{\Delta}+\rho v_{i,J}^{n+1} &= u\left(c_{i,J}^{n}\right)+v_{i-1,J}^{n+1}x_{i,J}+v_{i,J}^{n+1}\left(y_{i,J}+\upsilon_{J}+\zeta_{J}\right)+v_{i+1,J}^{n+1}z_{i,J}+v_{i,J-1}^{n+1}\chi_{j}.
\end{aligned}$$

can be rewritten in the form

$$\frac{1}{\Delta}\left(v^{n+1}-v^{n}\right)+\rho v^{n+1}=u^{n}+\left(A^{n}+C\right)v^{n+1}$$

where A^{n} and C again have a banded structure. Solving yields

$$v^{n+1}=\left(\left(\frac{1}{\Delta}+\rho\right)I-A^{n}-C\right)^{-1}\left(u^{n}+\frac{1}{\Delta}v^{n}\right).$$

For the simple case $I = 4$ and $J = 3$ we get

$$A = \begin{bmatrix}
y_{1,1} & z_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x_{2,1} & y_{2,1} & z_{2,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & x_{3,1} & y_{3,1} & z_{3,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & x_{4,1} & y_{4,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & y_{1,2} & z_{1,2} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & x_{2,2} & y_{2,2} & z_{2,2} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & x_{3,2} & y_{3,2} & z_{3,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & x_{4,2} & y_{4,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & y_{1,3} & z_{1,3} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{2,3} & y_{2,3} & z_{2,3} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{3,3} & y_{3,3} & z_{3,3} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_{4,3} & y_{4,3}
\end{bmatrix}$$

$$C = \begin{bmatrix}
\upsilon_1 + \chi_1 & 0 & 0 & 0 & \zeta_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \upsilon_1 + \chi_1 & 0 & 0 & 0 & \zeta_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \upsilon_1 + \chi_1 & 0 & 0 & 0 & \zeta_1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \upsilon_1 + \chi_1 & 0 & 0 & 0 & \zeta_1 & 0 & 0 & 0 & 0 \\
\chi_2 & 0 & 0 & 0 & \upsilon_2 & 0 & 0 & 0 & \zeta_2 & 0 & 0 & 0 \\
0 & \chi_2 & 0 & 0 & 0 & \upsilon_2 & 0 & 0 & 0 & \zeta_2 & 0 & 0 \\
0 & 0 & \chi_2 & 0 & 0 & 0 & \upsilon_2 & 0 & 0 & 0 & \zeta_2 & 0 \\
0 & 0 & 0 & \chi_2 & 0 & 0 & 0 & \upsilon_2 & 0 & 0 & 0 & \zeta_2 \\
0 & 0 & 0 & 0 & \chi_3 & 0 & 0 & 0 & \upsilon_3 + \zeta_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \chi_3 & 0 & 0 & 0 & \upsilon_3 + \zeta_3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \chi_3 & 0 & 0 & 0 & \upsilon_3 + \zeta_3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \chi_3 & 0 & 0 & 0 & \upsilon_3 + \zeta_3
\end{bmatrix}.$$

Note the pattern – A is block tridiagonal and C has three diagonals displaced by $I-1$ rows/columns.

Both Poisson and diffusion:

$$\rho v_1(a) = \max_c \left\{ \begin{array}{c} u(c) + \partial_a v_1(a,z)(zx_1 + ra - c) + \mu(z)\partial_z v_1(a,z) + \\ \frac{\sigma^2(z)}{2}\partial_{zz} v_1(a,z) + \lambda_1(v_2(a) - v_1(a)) \end{array} \right\}$$

$$\rho v_2(a) = \max_c \left\{ \begin{array}{c} u(c) + \partial_a v_2(a,z)(zx_2 + ra - c) + \mu(z)\partial_z v_2(a,z) + \\ \frac{\sigma^2(z)}{2}\partial_{zz} v_2(a,z) + \lambda_2(v_1(a) - v_2(a)) \end{array} \right\}$$

$$0 = -\partial_a(s_1(a,z)g_1(a,z)) - \partial_z(\mu(z)g_1(a,z)) + \frac{1}{2}\partial_{zz}\left(\sigma^2(z)g_1(a,z)\right) - \lambda_1 g_1(a) + \lambda_2 g_2(a)$$

$$0 = -\partial_a(s_2(a,z)g_1(a,z)) - \partial_z(\mu(z)g_2(a,z)) + \frac{1}{2}\partial_{zz}\left(\sigma^2(z)g_2(a,z)\right) - \lambda_2 g_2(a) + \lambda_1 g_1(a)$$

$$1 = \int_0^\infty \int_{\underline{a}}^\infty g_1(a,z)\,dadz + \int_0^\infty \int_{\underline{a}}^\infty g_2(a,z)\,dadz$$

$$0 = \int_0^\infty \int_{\underline{a}}^\infty a g_1(a,z)\,dadz + \int_0^\infty \int_{\underline{a}}^\infty a g_2(a,z)\,dadz$$

The HJB equation can be approximated as

$$\begin{aligned} \frac{v_{i,j,k}^{n+1} - v_{i,j,k}^n}{\Delta} + \rho v_{i,j,k}^{n+1} = u\left(c_{i,j,k}^n\right) + \partial_a v_{i,j,k}^{n+1}\left(z_j x_k + ra_i - c_{i,j}^n\right) + \\ \mu_j \partial_z v_{i,j,k}^{n+1} + \frac{\sigma_j^2}{2}\partial_{zz} v_{i,j,k}^{n+1} + \lambda_k\left(v_{i,j,-k}^{n+1} - v_{i,j,k}^{n+1}\right) \\ c_{i,j,k}^n = \left(u'\right)^{-1}\left(\left(v_{i,j,k}^n\right)'\right). \end{aligned}$$

An Ornstein-Uhlenbeck process is a diffusion with state-varying mean and/or variance

$$dz = \theta(\overline{z} - z)\,dt + \sigma dW$$

If $\theta > 0$ this process is stationary (it is similar to an AR(1)); note that the drift changes with z, so the upwind scheme has to adapt to this drift when taking derivatives with respect to z. A "Moll" process is an Ornstein-Uhlenbeck process that remains in a bounded interval (here $[0,1]$):

$$dz = \theta(\overline{z} - z)\,dt + \sigma z(1-z)\,dW.$$

There is also a geometric Brownian motion (lognormal process)

$$dz = \mu z dt + \sigma z dW,$$

which has no stationary distribution. Finally, the Feller square root process (Cox-Ingersoll-Ross process)

$$dz = \theta(\overline{z} - z)\,dt + \sigma\sqrt{z}dW$$

which converges to a gamma distribution.

The instantaneous gratification model in Maxted (2020) does not satisfy monotonicity if $\beta < 1$. He develops a trick that recasts the IG agent as a regular one with a particular utility function, designed to equalize their continuation values. Suppose that

$$u(c) = \frac{c^{1-\gamma}}{1-\gamma}$$

and define

$$\psi = \frac{\gamma - (1-\beta)}{\gamma} \in (0, 1].$$

Let

$$\widehat{u}^{+}(\widehat{c}) = \frac{\psi}{\beta} u\left(\frac{\widehat{c}}{\psi}\right) - \frac{1-\psi}{\beta} < u(\widehat{c})$$

and define

$$\widehat{u}(\widehat{c}) = \begin{cases} \widehat{u}^{+}(\widehat{c}) & \text{if } a > \underline{a} \\ \widehat{u}^{+}(\widehat{c}) & \text{if } a = \underline{a} \text{ and } \widehat{c} < \psi(y_j + r\underline{a}) \\ -\infty & \text{if } a = \underline{a} \text{ and } \widehat{c} \in (\psi(y_j + r\underline{a}), y_j + r\underline{a}) \\ u(\widehat{c}) & \text{otherwise} \end{cases}.$$

Then the value function for this 'hat' agent solves the standard HJB equation

$$\rho \widehat{v}_j(a) = \max_{\widehat{c}} \left\{ \widehat{u}(\widehat{c}) + \widehat{v}_j'(a)(y_j + ra - \widehat{c}) + \lambda_j(\widehat{v}_{-j}(a) - \widehat{v}_j(a)) \right\}$$

and one can prove that $\widehat{v}_j(a) = v_j(a)$ and

$$c_j(a) = \frac{1}{\psi} \widehat{c}_j(a)$$

if $a > \underline{a}$ or $a = \underline{a}$ and $\widehat{c}_j(a) \leq \psi(y_j + r\underline{a})$ and

$$c_j(a) = \widehat{c}_j(a)$$

if $a = \underline{a}$ and $\widehat{c}_j(a) > \psi(y_j + r\underline{a})$. Thus, we can compute the standard problem and convert it to an IG agent with minimal effort. Note that the discontinuity ensures that $\widehat{c} \in (\psi(y_j + r\underline{a}), y_j + r\underline{a})$ is never chosen by the standard agent.

17.5 Shooting Methods

Shooting methods solve boundary value problems; they can also be applied in discrete time, but we'll describe them here.

17.5.1 Forward Shooting

Take the growth model system

$$\dot{K} = Y(K) - \delta K - C$$
$$\dot{C} = \sigma C(Y_K(K) - \rho - \delta).$$

The steady state is

$$C^* = Y(K^*) - \delta K^*$$
$$Y_K(K^*) = \rho + \delta.$$

To solve this problem, we make a guess $C^0(0)$ for the post-jump initial value of consumption and solve the dynamic system forward for T periods. Consider the Euclidean distance function

$$D(C(0);T) = \sqrt{(C(T) - C^*)^2 + (K(T) - K^*)^2}.$$

The optimal solution sets $D(C(0);T) = 0$; while we could simply have chosen to solve the single equation

$$C(T) - C^* = 0,$$

a small inaccuracy here might well imply a large deviation in $K(T) - K^*$ (and vice versa) so we want to make sure both are close to zero, necessitating a minimization and the potential for a tradeoff between hitting C^* and hitting K^*.

As noted by Hebert and Stamp (2004), one can often exploit properties of the solution to discard paths quickly. For example, suppose we have an initial capital stock below the steady state; since transition paths in the growth model are monotonic, any path that implies a decline in capital at any point can be discarded without bothering to check the distance measure.

Forward shooting is generally numerically unstable; every path except the optimal one is attracted to the unstable manifold and therefore typically these paths diverge as you move forward in time, meaning small errors at the beginning get amplified. Thus, at T the path will typically be very far from the steady state, unless your guess is extremely accurate; this problem is worse in multiple dimensions where bracketing is unavailable and it is easy for variables to violate feasibility conditions like nonnegativity for bad initial guesses, forcing the use of ill-behaved penalty functions.

17.5.2 Smart Forward Shooting

Atolia and Buffie (2011) alters the objective function of the shooting algorithm to make it more robust. AB modify the distance function to

$$\widehat{D}\left(C\left(0\right);t_{\min}\right)=\sqrt{\left(C\left(t_{\min}\right)-C^{*}\right)^{2}+\left(K\left(t_{\min}\right)-K^{*}\right)^{2}}$$

where $t_{\min}\left(C\left(0\right)\right)=\arg\min_{t}\left\{D\left(C\left(0\right);t\right)\right\}$ is the date where the path starting from $C\left(0\right)$ is closest to the steady state; it may be that $t_{\min}<<T$, which cuts off the divergent behavior at the end of the path that often defeats forward shooting. However, the min function is ill-behaved around 0, so minimizing it requires some care. A bracketing method in one dimension can easily handle this problem; in higher dimensions, finding the minimum is difficult. For example, consider the economy

$$\begin{aligned}\frac{\tau\alpha_{m}+\sigma\alpha_{c}}{\sigma\tau}\frac{\dot{C}}{C} & =r-\rho+\frac{\tau-\sigma}{\sigma\tau}\alpha_{m}\frac{z+rb-vY\left(K\right)-\pi m}{m}\\ \dot{m} & =z+rb-vY\left(K\right)-\pi m\\ \dot{K} & =Y\left(K\right)-\delta K-C.\end{aligned}$$

Jumping to the stable manifold requires

$$\left[\begin{array}{c}C\left(0\right)-C^{*}\\ m\left(0\right)-m^{*}\end{array}\right]=\left[\begin{array}{c}x_{11}\\ x_{21}\end{array}\right]h_{1},$$

where λ_{1} is the stable eigenvalue, $\left[\begin{array}{c}x_{11}\\ x_{21}\end{array}\right]$ is the associated stable eigenvector of the linearized system at C^{*}, and $h_{1}=K\left(0\right)-K^{*}$; the other values satisfy $h_{2}=h_{3}=0$. If the guessed jumps are not correct, we instead have

$$\left[\begin{array}{c}C_{g}\left(0\right)-C^{*}\\ m_{g}\left(0\right)-m^{*}\end{array}\right]=\left[\begin{array}{ccc}x_{11} & x_{12} & x_{13}\\ x_{21} & x_{22} & x_{23}\end{array}\right]\left[\begin{array}{c}h_{1}\\ h_{2}\\ h_{3}\end{array}\right];$$

using the equation defining the true jump, we can obtain

$$\left[\begin{array}{c}C_{g}\left(0\right)-C^{*}\\ m_{g}\left(0\right)-m^{*}\end{array}\right]=\left[\begin{array}{cc}x_{12} & x_{13}\\ x_{22} & x_{23}\end{array}\right]\left[\begin{array}{c}h_{2}\\ h_{3}\end{array}\right]$$

or

$$\begin{bmatrix} h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} n_{12} & n_{13} \\ n_{22} & n_{23} \end{bmatrix} \begin{bmatrix} C_g(0) - C^* \\ m_g(0) - m^* \end{bmatrix}.$$

Since λ_3 is the largest positive (therefore unstable) eigenvalue, it disappears from the system if we follow the path

$$\frac{m_g(0) - m(0)}{C_g(0) - C(0)}|_{h_3=0} = -\frac{n_{21}}{n_{22}}.$$

Atolia and Buffie term this path the Rosenbrock Road; the solution $(C^*(0), m^*(0))$ lies at a point along this path that depends on the value of $K(0)$. The name comes from the classic Rosenbrock function

$$f(x, y) = (1 - x)^2 + 100\left(y - x^2\right)^2,$$

which is notoriously ill-behaved (you may also see it called the banana function). The global minimum is $(1, 1)$, but it is located in a parabolic-shaped flat and narrow valley; it is easy to find your way into the valley, but moving along it to the minimum is challenging because it is so flat and nonlinear (DFO methods seem to work better than quasi-Newton ones for this function).

The minimum distance function for a saddle path delivers a narrow steep valley located inside a very flat plane because all of the non-optimal paths are attracted to the same path (the unstable manifold) and therefore tend to have very similar minimum distances; only an extremely accurate guess will get inside the narrow valley and get closer to the steady state.

Atolia and Buffie (2011) develop a simple polytope method based on circles to find the minimum. We start with a guess $(C_g(0), m_g(0))$. We then sample points from a circle centered on that point, solve the economy forward starting from each of these points, and obtain the minimum distance value from this set. Now use this minimum as the center of a new circle and repeat. If there is no local improvement available the routine shrinks the current circle. In higher dimensions, one first uses a sphere to isolate the best meridien, then uses circle search in that direction. Due to the nested nature of the algorithm, it suffers heavily from the curse of dimensionality.

One might ask why we need to use a specialized routine, given that we have many good nonlinear minimizers; the reason is that the minimum distance function is quite ill-behaved. I have already mentioned the narrow flat valley that defines the Rosenbrock Road, which is both difficult to find and difficult to find the bottom once you're there; furthermore, the rest of the function is very flat, so it is difficult to even know which direction to proceed given an initial

guess. In addition, the minimum distance map may not be differentiable at the Rosenbrock Road; it frequently has a cusp at $D = 0$. A function with a cusp poses a difficult challenge for any minimizer based on derivatives (or even finite approximations to derivatives), since the solvers check that the derivative is close to zero and it will definitely not be. A standard kink (where the derivative is finite on both sides) is relatively easy to handle, since it can be smoothed away (differentiable functions are dense in the space of continuous functions, so there is a "nearby" one we can use). Combining the cusp with the steep valley and flat behavior elsewhere leads to a very difficult minimization for standard methods to handle. Sophisticated global minimizers may be able to find the right value, but they are slow.

17.5.3 Reverse Shooting

Reverse shooting takes time backwards; rather than guessing the value of the initial jump variable, we guess values just before we reach the steady state, then solve backward and compare the initial states to the implied one from the solution. Rather than guess just one variable (the initial consumption value $C(0)$), now we need to guess both $(C(T), K(T))$ for some T; the assumption is that these points lie on an ϵ-ring around the steady state. Then we unroll the system backward through time; instead of solving for Y_{n+1} given Y_n, as in the standard RK solver, we solve for Y_n given Y_{n+1}, then continue backward until time 0, and adjust the guesses so that the initial state $K(0)$ is close to the initial condition.

Reverse shooting exploits a feature of the dynamic system called a **separatrix** – the stable forward and stable backward curves (the stable and unstable manifolds) divide the plane into four disjoint areas and these curves block all orbits; we sometimes therefore refer to these curves as the stable and unstable separatrices. That is, no orbit crosses the stable or unstable manifold. Note: these four areas are not the same as the ones defined by the loci of points that keep the variables constant; as we have seen in class, the dynamic system definitely can cross those boundaries, and any orbit other than the stable manifold will be attracted toward the unstable manifold.

17.5.4 Smart Reverse Shooting

Atolia and Buffie (2010) improves on reverse shooting. Take a system of the form

$$\dot{z} = f(z, x, w)$$
$$\dot{x} = g(z, x, w)$$
$$0 = h(z, x, w).$$

x are the controls (jumps), z are the states, and w are other endogenous variables not directly under the control of an individual agent (like prices). Suppose z is two-dimensional; then we can draw ϵ-rings around the steady state and the initial condition, then follow paths in reverse time from different points on the steady state circle and find where they intersect the initial point circle, using the minimum distance mapping to sort through them. Suppose that, from the initial sampling, the closest approach is from p_j; then the true solution lies on the arc between p_{j-1} and p_{j+1}. Thus, if we take q and r to be the midpoints of the arcs $p_{j-1}p_j$ and p_jp_{j+1}, then the new minimum distance arc starts at one of the points (q, p_j, r). The only arcs of interest are now $p_{j-1}p_j$, qr, and p_jp_{j+1}; we can continue this "circle bisection" until we get an arbitrarily-good approximation. We can use the maximal eigenvalue of the linearized system around the steady state to reduce the size of the initial arc.

Complex eigenvalues lead to oscillating paths and therefore the path may intersect the ϵ-ring multiple times; however, there is no dominant eigenvalue ray, so trajectories do not gravitate toward a single point, leading to a much-better behaved minimization (albeit one with multiple global minima).

18 Filtering

The state space representation of a process with unobserved states x_t, exogenous observed states u_t, and observed variables z_t is given by

$$x_t = f(x_{t-1}, w_t)$$
$$z_t = h(x_t, v_t).$$

The first equation is the called the state transition and is subject to random structural shocks w_t; the second equation is called the observation equation and is subject to measurement errors

v_t. The goal of **filtering** is to recover "estimates" of the state $\{x_t\}$ using the structure of the system and the observations $\{z_t\}$. The general form is rather formidable, but fortunately there are special cases that are easy to handle; we will work our way up from the easiest special case, linear-Gaussian, to the full nonlinear-nonGaussian case.

Suppose that we have a data sample of length T on $\{z_t\}_{t=1}^T$; note that elements of x may be in z, as some states may be directly observed, and v_t may not have as many elements as z_t (some observations may be without noise).

18.1 Inversion Filter

Suppose there is no measurement error ($v_t = 0 \forall t$) and the dimensions of w_t and z_t are equal. Then we can construct the likelihood using the inversion filter. First, note that

$$\begin{aligned} z_t &= h\left(f\left(x_{t-1}, w_t\right)\right) \\ &\equiv g\left(x_{t-1}, w_t\right). \end{aligned}$$

We can solve for w_t by inverting g along the second dimension:

$$w_t = \widetilde{g}\left(z_t, x_{t-1}\right).$$

If w_t is normal, we can use a simple change of variables to get the likelihood:

$$\log\left(P\left(z_{1:T}\right)\right) = -\frac{T}{2}\log\left(|\Sigma|\right) - \frac{1}{2}\sum_{t=1}^{T}\widetilde{g}\left(z_t, x_{t-1}\right)^T \Sigma^{-1}\widetilde{g}\left(z_t, x_{t-1}\right) + \sum_{t=1}^{T}\log\left|\frac{\partial\widetilde{g}\left(z_t, x_{t-1}\right)}{\partial y_t}\right|.$$

18.2 Kalman Filter

Suppose (f, h) are linear and (w, v) are Gaussian; therefore, x and z are also Gaussian. Then the state space representation is

$$\begin{aligned} x_t &= F_t x_{t-1} + w_t \\ z_t &= H_t x_t + v_t \end{aligned}$$

with variance-covariance matrices Q_t and R_t, respectively. The Kalman filter is the optimal way to estimate the unobserved state vector x_t:

$$\begin{aligned}
\widehat{x}_{t|t-1} &= F_t \widehat{x}_{t-1|t-1} \\
P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + Q_t \\
y_t &= z_t - H_t \widehat{x}_{t|t-1} \\
S_t &= H_t P_{t|t-1} H_t^T + R_t \\
K_t &= P_{t|t-1} H_t^T S_t^{-1} \\
\widehat{x}_{t|t} &= \widehat{x}_{t|t-1} + K_t y_t \\
P_{t|t} &= (I - K_t H_t) P_{t|t-1} \\
y_{t|t} &= z_t - H_t \widehat{x}_{t|t}.
\end{aligned}$$

The variable $\widehat{x}_{t|t}$ is the conditional mean of the distribution of x_t given information up to and including time t (that is, all past estimates and all current and past observations) and $P_{t|t}$ is the variance-covariance matrix of that distribution. Since everything is Gaussian, those two statistics are all we need; it is common to refer to $\widehat{x}_{t|t}$ as the estimated state and $P_{t|t}$ as the uncertainty about the state estimate.

The process is as follows. First, given a state estimate from $t-1$, $\widehat{x}_{t-1|t-1}$, we forecast the value of $\widehat{x}_{t|t-1}$, which is the estimate of x_t given information up to $t-1$, and the uncertainty $P_{t|t-1}$. We then calculate the forecast error in the observation equation y_t, which is the difference between the observation z_t and what we would expect given our estimated state $\widehat{x}_{t|t-1}$. We then "process" that error using the Kalman gain matrix K_t, which describes the optimal weight given to each component of the signal z_t for each unknown state. Given the gain, we can use our new information to construct a new estimate of the state, $\widehat{x}_{t|t}$, and a new uncertainty measure $P_{t|t}$ and the process repeats itself. The initial state $\widehat{x}_{0|0}$ must be imposed; it is common to either use the steady state value or, if the model is being estimated, to treat the initial state as a parameter.

The first thing to notice is that the P matrices do not depend on the signals $\{z_t\}_{t=1}^T$; they only depend on the initial value $P_{0|0}$ and the length of the sample T. Thus, they can be processed "offline". Second, if the process is time-invariant, then we can compute a stationary filter by

allowing $P_{t|t} \to P^*$ and using the stationary Kalman filter process

$$\begin{aligned}
\widehat{x}_{t|t-1} &= F\widehat{x}_{t-1|t-1} \\
y_t &= z_t - H\widehat{x}_{t|t-1} \\
S_t &= HPH^T + R \\
K &= PH^T S^{-1} \\
\widehat{x}_{t|t} &= \widehat{x}_{t|t-1} + Ky_t \\
y_{t|t} &= z_t - H\widehat{x}_{t|t}
\end{aligned}$$

where

$$P = F\left(P - PH^T\left(HPH^T + R\right)^{-1} HP\right)F^T + Q;$$

iterating on this equation (make a positive-definite guess for P on the right-hand-side, compute the left-hand-side, and repeat) will converge provided that all the eigenvalues of F are bounded below 1 in modulus. The assumption built into using P is that the economy has experienced an infinitely-long past that is not observed by the econometrician, but that the econometrician knows about.

In some circumstances the Kalman filter is equivalent to a learning mechanism called recursive least squares (Sargent 1999). Consider the state space model in which the state follows a random walk

$$\begin{aligned}
\xi_t &= \xi_{t-1} + w_t, E\left[w_t w_t'\right] = R_{1t} \\
z_t &= \xi_t'\varphi_t + e_t, E\left[e_t e_t'\right] = R_{2t} \\
P_t &= E\left[\left(\xi_t - \widehat{\xi}_t\right)\left(\xi_t - \widehat{\xi}_t\right)'\right].
\end{aligned}$$

The Kalman filter recursion is

$$\begin{aligned}
\widehat{\xi}_t &= \widehat{\xi}_{t-1} + L_t\left(z_t - \varphi_t'\widehat{\xi}_{t-1}\right) \\
L_t &= \frac{P_{t-1}\varphi_t}{R_{2t} + \varphi_t' P_{t-1}\varphi_t} \\
P_t &= P_{t-1} - \frac{P_{t-1}\varphi_t\varphi_t' P_{t-1}}{R_{2t} + \varphi_t' P_{t-1}\varphi_t} + R_{1t}.
\end{aligned}$$

Now define the recursive least squares problem

$$\begin{aligned} V_T(\xi) &= \sum_{t=1}^{T} \kappa_{T,t}\alpha_t \left(z_t - \phi_t'\widehat{\xi}_{t-1}\right)^2 \\ \widehat{\xi}_t &= \widehat{\xi}_{t-1} + \alpha_t g_t R_t^{-1}\phi_t \left(z_t - \phi_t'\widehat{\xi}_{t-1}\right) \\ R_t &= R_{t-1} + g_t\left(\alpha_t\phi_t\phi_t' - R_{t-1}\right) \\ g_t &= \left(\sum_{s=1}^{t}\kappa_{t,s}\right)^{-1}; \end{aligned}$$

the last term is the gain, κ terms are weights equal to $\kappa_{T,t} = \lambda^{T-t}$ and the α_t terms are positive.

The Kalman filter is equivalent to RLS under the restrictions

$$\begin{aligned} R_{1t} &= \left(\frac{1}{\lambda} - 1\right) P_{t-1} \\ R_{2t} &= \frac{\lambda}{\alpha_t} \\ g_t &= \left(\frac{1-\lambda^t}{1-\lambda}\right)^{-1} \end{aligned}$$

with constant forgetting factor $\lambda \leq 1$. That is, we can view the Kalman filter as a learning mechanism. Note that if $\lambda = 1$, we get $g_t = \frac{1}{t}$, which is the fully-optimal learning gain rate.

18.3 Extended Kalman Filter

Now suppose that h and f are no longer linear, but they are differentiable. The extended Kalman filter replaces the state-space representation with a linear approximation around the current estimate $\widehat{x}_{t-1|t-1}$ when updating the distribution:

$$\begin{aligned} \widehat{x}_{t|t-1} &= f\left(\widehat{x}_{t-1|t-1}\right) \\ P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + Q_t \\ y_t &= z_t - h\left(\widehat{x}_{t|t-1}\right) \\ S_t &= H_t P_{t|t-1} H_t^T + R_t \\ K_t &= P_{t|t-1} H_t^T S_t^{-1} \\ \widehat{x}_{t|t} &= \widehat{x}_{t|t-1} + K_t y_t \\ P_{t|t} &= \left(I - K_t H_t\right) P_{t|t-1}, \end{aligned}$$

where

$$F_t = \left.\frac{\partial f}{\partial x}\right|_{\widehat{x}_{t-1|t-1}}$$
$$H_t = \left.\frac{\partial h}{\partial x}\right|_{\widehat{x}_{t|t-1}}$$

are the partial derivative terms. That is, we linearize the variance-covariance evolution, effectively treating the posterior as if it were still normal despite the nonlinear evolution of the states and the nonlinear observation equation. Because the true process is not Gaussian, the extended Kalman filter can diverge (P may not converge as the number of observations increases).

If the measurement error is not additive, we need to adapt the EKF slightly:

$$\begin{aligned}
\widehat{x}_{t|t-1} &= f\left(\widehat{x}_{t-1|t-1}\right) \\
P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + L_{t-1} Q_t L_{t-1}^T \\
y_t &= z_t - h\left(\widehat{x}_{t|t-1}\right) \\
S_t &= H_t P_{t|t-1} H_t^T + M_t R_t M_t^T \\
K_t &= P_{t|t-1} H_t^T S_t^{-1} \\
\widehat{x}_{t|t} &= \widehat{x}_{t|t-1} + K_t y_t \\
P_{t|t} &= \left(I - K_t H_t\right) P_{t|t-1},
\end{aligned}$$

where

$$L_{t-1} = \left.\frac{\partial f}{\partial x}\right|_{\widehat{x}_{t-1|t-1}}$$
$$M_t = \left.\frac{\partial h}{\partial v}\right|_{\widehat{x}_{t|t-1},0}.$$

Note that we linearize h around the zero mean of the measurement error process.

18.4 Unscented Kalman Filter

The unscented Kalman filter uses quadrature to approximate the prior and posterior distributions. For a random vector $x = (x_1, ..., x_l)$, the σ-points are a set of vectors $\{(s_{01}, ..., s_{0l}), ..., (s_{N1}, ..., s_{Nl})\}$

and an associated set of first-order weights that satisfy

$$\sum_{j=0}^{N} w_j^1 = 1$$
$$E\left[x_i\right] = \sum_{j=0}^{N} w_j^1 s_{j,i}$$

and second-order weights that satisfy

$$\sum_{j=0}^{N} w_j^2 = 1$$
$$E\left[x_i x_k\right] = \sum_{j=0}^{N} w_j^2 s_{j,i} s_{j,k}.$$

One common choice for the σ-points is

$$\begin{aligned}
s_{0,t} &= \widehat{x}_{t-1|t-1} \\
w_{0,t}^1 &= \frac{\alpha^2\kappa - l}{\alpha^2\kappa} \\
w_{0,t}^2 &= w_{0,t}^1 + 1 - \alpha^2 + \beta \\
s_{j,t} &= \widehat{x}_{t-1|t-1} + \alpha\sqrt{\kappa}A_{j,t} \\
s_{l+j,t} &= \widehat{x}_{t-1|t-1} - \alpha\sqrt{\kappa}A_{j,t} \\
w_{j,t}^1 &= w_{j,t}^2 = \frac{1}{2\alpha^2\kappa}
\end{aligned}$$

where (α, κ) control the range of the σ-points, β is a parameter that depends on the distribution of x, and $A_{j,t}$ is the jth column of the Cholesky decomposition of $P_{t-1|t-1}$:

$$P_{t-1|t-1} = AA^T.$$

If x is Gaussian, then $\beta = 2$ is optimal, and is often a good approximation anyway. Common recommendations are $\alpha = 10^{-3}$ and $\kappa = 1$, although sometimes larger values of α are used if the

distribution is very disperse or the model is highly nonlinear. The filter is then

$$
\begin{aligned}
x_{j,t} &= f\left(s_j\right) \\
\widehat{x}_{t|t-1} &= \sum_{j=0}^{2l} w_{j,t}^1 x_{j,t} \\
P_{t|t-1} &= \sum_{j=0}^{2l} w_{j,t}^2 \left(x_{j,t} - \widehat{x}_{t|t-1}\right)\left(x_{j,t} - \widehat{x}_{t|t-1}\right)^T + Q_t \\
z_t &= h\left(s_t\right) \\
\widehat{z}_t &= \sum_{j=0}^{2l} w_{j,t}^1 z_t \\
S_t &= \sum_{j=0}^{2l} w_{j,t}^2 \left(z_{j,t} - \widehat{z}_{j,t}\right)\left(z_{j,t} - \widehat{z}_{j,t}\right)^T + R_t \\
C_t &= \sum_{j=0}^{2l} w_{j,t}^2 \left(s_{j,t} - \widehat{x}_{t|t-1}\right)\left(z_{j,t} - \widehat{z}_{j,t}\right)^T \\
K_t &= C_t S_t^{-1} \\
\widehat{x}_{t|t} &= \widehat{x}_{t|t-1} + K_t\left(z_t - \widehat{z}_t\right) \\
P_{t|t} &= P_{t|t-1} - K_t S_t K_t^T.
\end{aligned}
$$

In general the quadrature nodes (the σ-points) change over time, as do the weights.

18.5 Particle (Sequential Monte Carlo) Filter

The general state transition equation and observation equation

$$
\begin{aligned}
x_t &= h\left(x_{t-1}, w_t\right) \\
z_t &= g\left(x_t, \nu_t\right)
\end{aligned}
$$

where ϵ_t are innovations to structural shocks and ν_t are innovations to measurement error with density functions $f_w\left(w\right)$ and $f_\nu\left(\nu\right)$. The particle filter is based on Monte Carlo integration.

1. Draw N samples from known distribution of w, denoted $w_{t|t-1,i}$, using density $f_w\left(w\right)$;

2. The prediction step: generate N values of $x_{t|t-1,i}$ using

$$
x_{t|t-1,i} = h\left(x_{t-1,i}, w_{t|t-1,i}\right);
$$

if $t = 1$ then must obtain $x_{0,i}$ by draws from some distribution

3. Use observed z_t to obtain N values of $\nu_{t|t-1,i}$ by solving

$$z_t - g\left(x_{t|t-1,i}, \nu_{t|t-1,i}\right) = 0;$$

if the observer equation is linear in ν this becomes the simpler expression

$$\nu_{t|t-1,i} = z_t - g\left(x_{t|t-1,i}\right),$$

otherwise it must be solved numerically (which can pose a problem if the g function is not monotonic in the error, such as if it has been approximated by a polynomial);

4. Compute

$$p\left(\nu_{t|t-1,i}\right) = f_\nu\left(\nu_{t|t-1,i}\right)$$

using the known distribution of ν; note that to estimate a model with the particle filter, we must have measurement errors in each observation equation or this distribution becomes degenerate, whereas these errors are not necessary in the variants of the Kalman filter;

5. Compute the resampling weights

$$q_{t,i} = \frac{p\left(\nu_{t|t-1,i}\right)}{\sum_{i=1}^{N} p\left(\nu_{t|t-1,i}\right)};$$

the resampling step ensures that we do not face a sample depletion problem, in which only a few particles continue to contribute to the filtering process as we move through the sample;

6. Resample, with replacement, N values of $x_{t|t-1,i}$ using weights $q_{t,i}$, denoting this new sample $x_{t,i}$; this sample is an equally-weighted sample from the posterior;

7. Return to Step (a) until end of sample.

Since the particle filter integrates using Monte Carlo methods, it is both very flexible (in principle it can handle highly-nonlinear functions and any distribution with finite moments) and rather slow (the convergence rate is $\sqrt{N}$). To avoid "chattering" (failure of stochastic equicontinuity), you should use the same set of particles if you are evaluating the filter many times (as in the MCMC methods that are discussed next).

18.6 Monte Carlo Markov Chains

Filtering can be used to estimate dynamic macro models. The likelihood function for the model, given the observable data Z, is

$$L(Z;\theta) = \prod_{t=1}^{T} \Pr(Z|\theta)$$

for a collection of structural parameters θ. This function is factored and the independent probabilities are computed using a filter:

$$L(\theta;Z) = \prod_{t=1}^{T} \Pr(z_t|z_1,...,z_{t-1},\theta);$$

this calculation is normally done using the log-likelihood to avoid scaling problems:

$$l(\theta;Z) = \sum_{t=1}^{T} \log\left(\Pr(z_t|x_1,...,z_{t-1},\theta)\right).$$

The calculation of the likelihood is then a filtering problem, which we have already discussed. For example, suppose the system is linear and Gaussian, so that the Kalman filter can be used. Then

$$l(\theta;Z) = -\sum_{t=1}^{T}\left(\frac{d}{2}\log(2\pi) - \frac{1}{2}\log\left|\Omega_{t|t-1}\right| + \frac{1}{2}y_t^T\Omega_{t|t-1}y_t\right)$$

where d is the dimension of z_t and

$$\Omega_{t|t-1} = HP_{t|t-1}H^T + R.$$

To compute the ML estimate, we employ a Metropolis-Hastings algorithm and a random walk Markov Chain. Begin with a parameter vector θ_1. Then randomly draw a new parameter vector $\widetilde{\theta}$ from distribution with mean θ_1 (one typically uses a normal with some variance-covariance matrix Σ); this distribution is called the **proposal density**. Form the likelihood ratio

$$\lambda = \frac{L\left(\widetilde{\theta};Z\right)}{L(\theta_1;Z)} = \exp\left(l\left(\widetilde{\theta};Z\right) - l(\theta_1;Z)\right).$$

If $\lambda \geq 1$, then $\theta_2 = \widetilde{\theta}$; if $\lambda \leq 1$, then $\theta_2 = \widetilde{\theta}$ with probability λ and $\theta_2 = \theta_1$ with probability $1-\lambda$. The frequency with which $\theta_2 = \theta_1$ is called the **rejection rate**; if the rejection rate is too low, then we may get stuck around a local mode (after all, we are simply doing global optimization, so if we have multiple modes we need to avoid a bad one), whereas if it is too high the chain will

take forever to converge. A good rule of thumb is to choose Σ to get a rejection rate of $20-30$ percent. After constructing a long sample, we take the mode of the sample as our ML estimate of θ and we can produce confidence intervals. Note: we could simply maximize the likelihood using any other global maximizer, but Metropolis-Hastings has some advantages if we take a Bayesian perspective.

Definition 130 *A* γ***-confidence interval*** *is a pair of random functions* $A(Z)$ *and* $B(Z)$ *such that,* $\forall \theta \in \Theta$*,*

$$\Pr(A(Z) < \theta < B(Z)) \geq \gamma.$$

θ is viewed as fixed and the endpoints are random numbers. If instead we are taking a Bayesian perspective (which we should), then the likelihood ratio incorporates the prior $\mathcal{P}(\theta)$:

$$\lambda = \frac{L\left(\widetilde{\theta}; Z\right)}{L(\theta_1; Z)} \frac{\mathcal{P}\left(\widetilde{\theta}\right)}{\mathcal{P}(\theta_1)}.$$

The sample we get from the MCMC procedure can now be viewed as a sample from the posterior, which would in general be intractable, because we do not have a conjugate pair (no matter the prior, the nonlinearity of the likelihood in terms of θ will mess that all up). It is common in Bayesian models to start the chain at the posterior mode, which is found by directly maximizing the posterior (which can be done using the Metropolis-Hastings algorithm). We then produce credible intervals, which are the Bayesian analogues to confidence intervals.

Definition 131 *A* γ***-credible interval*** *is a pair of endpoints* $A(z)$ *and* $B(z)$ *such that*

$$\Pr(A(z) < \theta < B(z) \,|Z = z) \geq \gamma.$$

Credible intervals are defined as conditioned on the realization of the data z; now θ is viewed as random. Defining these ideas in higher dimensions is straightforward. To get confidence intervals, we need the distribution of the MLE estimator $\widehat{\theta}$, which we can obtain easily for multivariate normals. For credible intervals, we simply need to compute the fraction of periods that the chain spends within some distance of the mode, then find the appropriate distance to get that number equal to γ; we can do so by solving the nonlinear equation

$$\frac{1}{T}\sum_{t=1}^{T} \mathbf{1}\left(\left\|\theta_t - \overline{\theta}\right\|^2 \leq d\right) = \gamma,$$

where d is the target distance and $\mathbf{1}$ is an indicator function. Because T is finite, this equation is not continuous, but we can linearly interpolate if necessary.

18.7 Dynamic Programming with Incomplete Information

We can use filtering techniques in models in which agents do not observe everything they might want to see; there will be an issue with the state space not generally being finite, however. Following Porapakkarm and Young (2009), consider a variant of the Krusell-Smith model but with idiosyncratic shocks to both capital and labor income. Assume also that only prices are observed. We can write the dynamic program as

$$v\left(k,\varepsilon_{t|t},\eta_{t|t},K_{t|t},z_{t|t}\right)=\max_{k'\geq 0,c\geq 0}\left\{\log\left(c\right)+\beta E\left[v\left(k',\varepsilon_{t+1|t+1},\eta_{t+1|t+1},K_{t+1|t+1},z_{t+1|t+1}\right)\right]\right\}$$

subject to the budget constraint

$$c+k'\leq\left(1+r-\delta\right)k+w$$

and the state-space equations for the hidden states

$$\begin{bmatrix}\varepsilon'\\ \eta'\\ K'\\ z'\end{bmatrix}=\begin{bmatrix}0\\ 0\\ a_0\\ 0\end{bmatrix}+\begin{bmatrix}\rho_\epsilon & \rho_{\epsilon\eta} & 0 & 0\\ \rho_{\epsilon\eta} & \rho_\eta & 0 & 0\\ 0 & 0 & a_1 & a_2\\ 0 & 0 & 0 & \rho_z\end{bmatrix}\begin{bmatrix}\varepsilon\\ \eta\\ K\\ z\end{bmatrix}+\begin{bmatrix}\zeta_{t+1}\\ \nu_{t+1}\\ 0\\ e_{t+1}\end{bmatrix}$$

and the observation equations

$$\begin{bmatrix}\log\left(r'\right)\\ \log\left(w'\right)\end{bmatrix}=\begin{bmatrix}\log\left(\alpha\right)+\left(1-\alpha\right)\log\left(N\right)\\ \log\left(1-\alpha\right)-\alpha\log\left(N\right)\end{bmatrix}+\begin{bmatrix}0 & 1 & \alpha-1 & 1\\ 1 & 0 & \alpha & 1\end{bmatrix}\begin{bmatrix}\varepsilon\\ \eta\\ K\\ z\end{bmatrix}.$$

Using these equations, we can derive the law of motion for the estimated states and the uncertainty of these states, and use cubature to compute the expectations; note the assumption that the law of motion for aggregate capital is log-linear, consistent with the equilibrium under complete information.

There is a problem with this approach, however, the state space is not finite (Townsend 1983, Sargent 1991). To see why, we need to define a "higher-order expectation". Define individual i's first-order expectation of x_t as

$$\widehat{x}_{t,i}^{(1)} = E_{i,t}\left[x_t\right].$$

If we average this expression over i we get

$$\widehat{x}_{t}^{(1)} = \int E_{i,t}\left[x_t\right] di,$$

the average first-order expectation. We can then ask an agent to predict the average expectation:

$$\widehat{x}_{t,i}^{(2)} = E_{i,t}\left[\widehat{x}_{t}^{(1)}\right],$$

which is the second-order expectation. We can then average again:

$$\widehat{x}_{t}^{(2)} = \int E_{i,t}\left[\widehat{x}_{t}^{(1)}\right] di.$$

The recursive formula for the average expectation of order k is

$$\begin{aligned}\widehat{x}_{t}^{(k)} &= \int E_{i,t}\left[\widehat{x}_{t}^{(k-1)}\right] di \\ &= \int E_{i,t}\left[\int E_{i,t}\left[\widehat{x}_{t}^{(k-1)}\right] di\right] di.\end{aligned}$$

Because $E_{i,t}$ depends on i, we cannot use the law of iterated expectations to reduce this expression; if $i \neq j$, then

$$E_{i,t}\left[E_{j,t}\left[y\right]\right] \neq E_{i,t}\left[y\right]$$

because neither information set is finer than the other.

The following example from Nimark (2017) is useful for understanding how the infinite state space arises. Suppose we are trying to solve for p_t in the following equation:

$$p_t = \beta \int E\left[p_{t+1}|\Omega_{t,j}\right] dj - \left(\theta_t + \varepsilon_t\right)$$

where

$$\theta_t = \rho\theta_{t-1} + u_t$$

and both (ε_t, u_t) are normal. The signal vector is

$$s_{t,j} = \begin{bmatrix} z_{t,j} \\ p_t \end{bmatrix}$$

where the noisy signal z_t satisfies

$$z_{t,j} = \theta_t + \eta_{t,j}$$

with η also being normal.

Consider starting with a naive conjecture that

$$p_t = -\theta_t - \varepsilon_t;$$

that is, agents believe other expectations wash out in aggregation. The Kalman filter update then implies

$$\widehat{\theta}^{(1)}_{t,j} = \rho\widehat{\theta}^{(1)}_{t-1,j} + K_0\left(s_{t,j} - D_0\rho\widehat{\theta}^{(1)}_{t-1,j}\right)$$
$$D_0 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Stacking the true state and the expectation state we get

$$\begin{bmatrix} \theta_t \\ \widehat{\theta}^{(1)}_t \end{bmatrix} = \begin{bmatrix} \rho & 0 \\ K_0D_0\rho & (I - K_0D_0)\,\rho \end{bmatrix} \begin{bmatrix} \theta_{t-1} \\ \widehat{\theta}^{(1)}_{t-1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ K_0D_0 & K_0 \end{bmatrix} \begin{bmatrix} u_t \\ \varepsilon_t \end{bmatrix}$$

or

$$\theta^{0:1}_t = M_1\theta^{0:1}_{t-1} + N_1 w_t.$$

The average expectation of p_{t+1} is then

$$\int E\left[p^0_{t+1}|\Omega^0_{t,j}\right] dj = -\rho\widehat{\theta}^{(1)}_t.$$

Substituting into the pricing equation we get

$$p^1_t = -\begin{bmatrix} 1 & \beta\rho \end{bmatrix}\theta^{0:1}_t - \varepsilon_t.$$

The new measurement equation is

$$s_{t,j} = D_1\theta^{0:1}_t + e_1\eta_{t,j} + e_2\varepsilon_t$$
$$D_1 = \begin{bmatrix} 1 & 0 \\ -1 & -\beta\rho \end{bmatrix}.$$

But now agents need to formulate an expectation of other agents' expectations about the average (the average expectation of the average expectation), which in turn becomes part of the state

space and then requires a higher order expectation. Makarov and Rytchkov (2012) prove that there does not exist a finite-dimensional state space that can solve this particular equation.

Nimark (2017) suggests a truncation approach to deal with this problem – simply "chop off" the recursion at some order N. Iterations satisfy the bound

$$\left\|p^k - p^*\right\| \leq \frac{\beta^k}{1-\beta}\left\|p^1 - p^0\right\|$$

which in turn implies, $\forall \delta > 0$, there exists $N \geq 1$ such that

$$\left\|p^N - p^*\right\| < \delta.$$

An alternative approach is to solve the model in the frequency domain – see below.

18.7.1 Porapakkarm and Young (2009)

First I show how to derive the price process for R_t^i and w_t^i used by an FI agent. Given competitive factor markets one can solve for z_t and $\log K_t$ by equating factor prices to marginal products:

$$\begin{aligned}
\log\left(R_t^i\right) - \log\alpha - (1-\alpha)\log\left(\overline{N}\right) &= z_t + (\alpha-1)\log\left(K_t\right) + \eta_t^i \\
\log\left(w_t^i\right) - \log\left(1-\alpha\right) + \alpha\log\left(\overline{N}\right) &= z_t + \alpha\log\left(K_t\right) + \varepsilon_t^i,
\end{aligned}$$

which implies

$$\begin{aligned}
z_t = {} & \alpha\left(\log\left(R_t^i\right) - \log\alpha - (1-\alpha)\log\left(\overline{N}\right)\right) + \\
& (1-\alpha)\left(\log\left(w_t^i\right) - \log\left(1-\alpha\right) + \alpha\log\left(\overline{N}\right)\right) - (1-\alpha)\,\varepsilon_t^i - \alpha\eta_t^i \\
\log\left(K_t\right) = {} & \left(\log\left(w_t^i\right) - \log\left(1-\alpha\right) + \alpha\log\left(\overline{N}\right)\right) - \\
& \left(\log\left(R_t^i\right) - \log\alpha - (1-\alpha)\log\left(\overline{N}\right)\right) - \varepsilon_t^i + \eta_t^i.
\end{aligned}$$

Now we substitute the laws of motion for aggregate capital, the technology shock, and the idiosyncratic shock processes into the pricing equations in period $t+1$, obtaining

$$\begin{aligned}
\log\left(R_{t+1}^i\right) - \log\alpha - (1-\alpha)\log\left(\overline{N}\right) &= \rho_z z_t + e_{t+1} + (\alpha-1)\left(a_0 + a_1 z_t + a_2\log\left(K_t\right)\right) + \mu_\eta + \rho_\eta\eta_t^i + \zeta_{t+1}^i \\
\log\left(w_{t+1}^i\right) - \log\left(1-\alpha\right) + \alpha\log\left(\overline{N}\right) &= \rho_z z_t + e_{t+1} + \alpha\left(a_0 + a_1 z + a_2\log\left(K_t\right)\right) + \mu_\varepsilon + \rho_\varepsilon\varepsilon_t^i + \nu_{t+1}^i.
\end{aligned}$$

Substituting z_t and $\log K_t$ out, we get the forecast equations

$$\begin{aligned}
\begin{bmatrix} \log\left(R_{t+1}^i\right) \\ \log\left(w_{t+1}^i\right) \end{bmatrix} &\sim N\left(\begin{bmatrix} E_t\left[\log\left(R_{t+1}^i\right)\right] \\ E_t\left[\log\left(w_{t+1}^i\right)\right] \end{bmatrix}, \mathbf{V}_2\right) \\
E_t\left[\log\left(R_{t+1}^i\right)\right] &= A_0 + A_1 \log\left(R_t^i\right) + A_2 \log\left(w_t^i\right) - A_2\varepsilon_t^i + \left(\rho_\eta - A_1\right)\eta_t^i \\
E_t\left[\log\left(w_{t+1}^i\right)\right] &= A_3 + A_4 \log\left(R_t^i\right) + A_5 \log\left(w_t^i\right) + \left(\rho_\varepsilon - A_5\right)\varepsilon_t^i - A_4\eta_t^i \\
\mathbf{V}_2 &= \begin{bmatrix} \sigma_e^2 + \sigma_\zeta^2 + 2\rho_{e\zeta}\sigma_e\sigma_\zeta & \sigma_e^2 + \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta + \rho_{e\nu}\sigma_e\sigma_\nu + \rho_{e\zeta}\sigma_e\sigma_\zeta \\ \sigma_e^2 + \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta + \rho_{e\nu}\sigma_e\sigma_\nu + \rho_{e\zeta}\sigma_e\sigma_\zeta & \sigma_e^2 + \sigma_\nu^2 + 2\rho_{e\nu}\sigma_e\sigma_\nu \end{bmatrix}
\end{aligned} \tag{150}$$

where

$$\begin{aligned}
A_0 &= \left(\alpha - 1\right)a_0 + \left(1 - A_1\right)\log\left(\alpha\right) - A_2\log\left(1-\alpha\right) + \left(\left(1 - A_1\right)\left(1-\alpha\right) + A_2\alpha\right)\log\left(\overline{N}\right) + \mu_\eta \\
A_1 &= \rho_z\alpha + \left(1-\alpha\right)\left(a_2 - a_1\alpha\right) \\
A_2 &= \left(1-\alpha\right)\left(\rho_z - \left(1-\alpha\right)a_1 - a_2\right) \\
A_3 &= \alpha a_0 - A_4\log\left(\alpha\right) + \left(1 - A_5\right)\log\left(1-\alpha\right) + \left(\alpha\left(A_5 - 1\right) - \left(1-\alpha\right)A_4\right)\log\left(\overline{N}\right) + \mu_\varepsilon \\
A_4 &= \left(\rho_z + \alpha a_1 - a_2\right)\alpha \\
A_5 &= \left(1-\alpha\right)\rho_z + \alpha\left(a_1\left(1-\alpha\right) + a_2\right).
\end{aligned}$$

The error in the forecasts of future prices is the innovation in the technology shock and the idiosyncratic shocks.

A detailed derivation is now given. For returns, we have

$$\begin{aligned}
\log\left(R_{t+1}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right) &= \rho_{z}z_{t}+e_{t+1}+\left(\alpha-1\right)\left(a_{0}+a_{1}z_{t}+a_{2}\log\left(K_{t}\right)\right)+\mu_{\eta}+\rho_{\eta}\eta_{t}^{i}+\zeta_{t+1}^{i}\\
&= \left(\alpha-1\right)a_{0}+\left(\left(\alpha-1\right)a_{1}+\rho_{z}\right)z_{t}+\left(\alpha-1\right)a_{2}\log\left(K_{t}\right)+\\
&\mu_{\eta}+\rho_{\eta}\eta_{t}^{i}+e_{t+1}+\zeta_{t+1}^{i}\\
&= \left(\alpha-1\right)a_{0}+\\
&\left(\left(\alpha-1\right)a_{1}+\rho_{z}\right)\alpha\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)+\\
&\left(\left(\alpha-1\right)a_{1}+\rho_{z}\right)\left(1-\alpha\right)\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)-\alpha\log\left(\overline{N}\right)\right)-\\
&\left(\left(\alpha-1\right)a_{1}+\rho_{z}\right)\left(1-\alpha\right)\varepsilon_{t}^{i}-\left(\left(\alpha-1\right)a_{1}+\rho_{z}\right)\alpha\eta_{t}^{i}+\\
&\left(\alpha-1\right)a_{2}\varepsilon_{t}^{i}+\left(\alpha-1\right)a_{2}\eta_{t}^{i}+\mu_{\eta}+\rho_{\eta}\eta_{t}^{i}+e_{t+1}+\zeta_{t+1}^{i}\\
&= \left(\alpha-1\right)a_{0}+\\
&\left(\rho_{z}\alpha+\left(1-\alpha\right)\left(a_{2}-a_{1}\alpha\right)\right)\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)+\\
&\left(1-\alpha\right)\left(\rho_{z}-\left(1-\alpha\right)a_{1}-a_{2}\right)\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right)\right)-\\
&\left(1-\alpha\right)\left(\rho_{z}-\left(1-\alpha\right)a_{1}-a_{2}\right)\varepsilon_{t}^{i}+\\
&\left(\rho_{\eta}-\alpha\rho_{z}-\left(1-\alpha\right)\left(a_{2}-\alpha a_{1}\right)\right)\eta_{t}^{i}+\mu_{\eta}+e_{t+1}+\zeta_{t+1}^{i}.
\end{aligned}$$

For wages, we have

$$\begin{aligned}
\log\left(w_{t+1}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right) &= \rho_{z}z_{t}+e_{t+1}+\alpha\left(a_{0}+a_{1}z+a_{2}\log\left(K_{t}\right)\right)+\mu_{\varepsilon}+\rho_{\varepsilon}\varepsilon_{t}^{i}+\nu_{t+1}^{i}\\
&= \alpha a_{0}+\left(\rho_{z}+\alpha a_{1}\right)z_{t}+\alpha a_{2}\log\left(K_{t}\right)+\mu_{\varepsilon}+\rho_{\varepsilon}\varepsilon_{t}^{i}+e_{t+1}+\nu_{t+1}^{i}\\
&= \alpha a_{0}+\left(\rho_{z}+\alpha a_{1}\right)\alpha\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)+\\
&\left(\rho_{z}+\alpha a_{1}\right)\left(1-\alpha\right)\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right)\right)-\\
&\left(\rho_{z}+\alpha a_{1}\right)\left(1-\alpha\right)\varepsilon_{t}^{i}-\left(\rho_{z}+\alpha a_{1}\right)\alpha\eta_{t}^{i}+\\
&\alpha a_{2}\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right)\right)-\\
&\alpha a_{2}\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)-\\
&\alpha a_{2}\varepsilon_{t}^{i}+\alpha a_{2}\eta_{t}^{i}+\mu_{\varepsilon}+\rho_{\varepsilon}\varepsilon_{t}^{i}+e_{t+1}+\nu_{t+1}^{i}\\
&= \alpha a_{0}+\left(\rho_{z}+\alpha a_{1}-a_{2}\right)\alpha\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)+\\
&\left(\rho_{z}\left(1-\alpha\right)+\alpha\left(a_{1}\left(1-\alpha\right)+a_{2}\right)\right)\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right)\right)+\\
&\left(\rho_{\varepsilon}-\rho_{z}\left(1-\alpha\right)-\alpha\left(a_{1}\left(1-\alpha\right)+a_{2}\right)\right)\varepsilon_{t}^{i}-\\
&\left(\rho_{z}+\alpha a_{1}-a_{2}\right)\eta_{t}^{i}+\mu_{\varepsilon}+e_{t+1}+\nu_{t+1}^{i}.
\end{aligned}$$

Collecting terms we can write these equations as

$$\begin{aligned}
\log\left(R_{t+1}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right) &= \left(\alpha-1\right)a_{0}+\mu_{\eta}+\\
&A_{1}\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)+\\
&A_{2}\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right)\right)-\\
&A_{2}\varepsilon_{t}^{i}+\left(\rho_{\eta}-A_{1}\right)\eta_{t}^{i}+e_{t+1}+\zeta_{t+1}^{i}\\
\log\left(w_{t+1}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right) &= \alpha a_{0}+\mu_{\varepsilon}+\\
&A_{4}\left(\log\left(R_{t}^{i}\right)-\log\left(\alpha\right)-\left(1-\alpha\right)\log\left(\overline{N}\right)\right)+\\
&A_{5}\left(\log\left(w_{t}^{i}\right)-\log\left(1-\alpha\right)+\alpha\log\left(\overline{N}\right)\right)+\\
&\left(\rho_{\varepsilon}-A_{5}\right)\varepsilon_{t}^{i}-A_{4}\eta_{t}^{i}+e_{t+1}+\nu_{t+1}^{i}.
\end{aligned}$$

The equations (150) are produced by collecting the constants.

I now present the filtering problem for the PI agent. Using matrix notation, write the state-

space system (**??**) as

$$\begin{aligned}\mathbf{Y}_{t+1} &= \mathcal{C} + \mathcal{D}\mathbf{Y}_t + {}_{t+1} \\ \mathbf{X}_t &= \mathcal{E} + \mathcal{F}\mathbf{Y}_t \\ {}_{t+1} &\sim N\left(0, \right).\end{aligned}$$

Since the observed variables are linear combinations of state variables, we can reduce the number of states. Define

$$\mathcal{H} = \begin{bmatrix} & \mathcal{F} & & \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and premultiply to obtain

$$\mathcal{H}\mathbf{Y}_{t+1} = \mathcal{H}\mathcal{C} + \mathcal{H}\mathcal{D}\mathcal{H}^{-1}\mathcal{H}\mathbf{Y}_t + \mathcal{H}_{t+1}.$$

The first two rows of $\mathcal{H}\mathbf{Y}_t$ are the rows of $\mathbf{X}_t - \mathcal{E}$ and therefore observable in the current period. The unobserved state variables are now only ε_t^i and η_t^i, the idiosyncratic shocks. Rewrite the transformed system as

$$\begin{aligned}{}^{i}_{t+1} &= \mathbf{G}_1 + \mathbf{G}_2 {}^{i}_{t} + {}^{i}_{t+1} \\ \mathbf{Z}^i_{t+1} &= \mathbf{B}_1 + \mathbf{B}_2\mathbf{Z}^i_t + \mathbf{B}_3 {}^{i}_{t} + {}^{i}_{t+1}\end{aligned} \tag{151}$$

where

$$
\begin{aligned}
{}^{i}_{t} &= \begin{bmatrix} \varepsilon_t^i \\ \eta_t^i \end{bmatrix} \\
{}^{i}_{t+1} &= \begin{bmatrix} \nu_{t+1}^i \\ \zeta_{t+1}^i \end{bmatrix} \\
\mathbf{G}_1 &= \begin{bmatrix} \mu_\varepsilon \\ \mu_\eta \end{bmatrix} \\
\mathbf{G}_2 &= \begin{bmatrix} \rho_\varepsilon & 0 \\ 0 & \rho_\eta \end{bmatrix} \\
\mathbf{Z}_t^i &= \begin{bmatrix} \log\left(R_t^i\right) - \log\left(\alpha\right) - \left(1-\alpha\right)\log\left(\overline{N}\right) \\ \log\left(w_t^i\right) - \log\left(1-\alpha\right) + \alpha\log\left(\overline{N}\right) \end{bmatrix} \\
{}^{i}_{t+1} &= \begin{bmatrix} e_{t+1} + \zeta_{t+1}^i \\ e_{t+1} + \nu_{t+1}^i \end{bmatrix} \\
\mathbf{B}_1 &= \begin{bmatrix} \left(\alpha-1\right)a_0 + \mu_\eta \\ \alpha a_0 + \mu_\varepsilon \end{bmatrix} \\
\mathbf{B}_2 &= \begin{bmatrix} A_1 & A_2 \\ A_4 & A_5 \end{bmatrix} \\
\mathbf{B}_3 &= \begin{bmatrix} -A_2 & \rho_\eta - A_1 \\ \rho_\varepsilon - A_5 & -A_4 \end{bmatrix}
\end{aligned}
$$

with $\{A_i\}_0^5$ the same terms as in Appendix A. The covariance between ${}^{i}_{t+1}$ and ${}^{i}_{t+1}$ is given by the matrix

$$
= \begin{bmatrix} \mathbf{V}_1 & \mathbf{V}_3^T \\ \mathbf{V}_3 & \mathbf{V}_2 \end{bmatrix},
$$

where

$$\mathbf{V}_1 = \begin{bmatrix} \sigma_\nu^2 & \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta \\ \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta & \sigma_\zeta^2 \end{bmatrix}$$

$$\mathbf{V}_2 = \begin{bmatrix} \sigma_e^2 + \sigma_\zeta^2 + 2\rho_{e\zeta}\sigma_e\sigma_\zeta & \sigma_e^2 + \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta + \rho_{e\nu}\sigma_e\sigma_\nu + \rho_{e\zeta}\sigma_e\sigma_\zeta \\ \sigma_e^2 + \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta + \rho_{e\nu}\sigma_e\sigma_\nu + \rho_{e\zeta}\sigma_e\sigma_\zeta & \sigma_e^2 + \sigma_\nu^2 + 2\rho_{e\nu}\sigma_e\sigma_\nu \end{bmatrix}$$

$$\mathbf{V}_3 = \begin{bmatrix} \rho_{e\nu}\sigma_e\sigma_\nu + \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta & \sigma_\nu^2 + \rho_{e\nu}\sigma_e\sigma_\nu \\ \sigma_\zeta^2 + \rho_{e\zeta}\sigma_e\sigma_\zeta & \rho_{e\zeta}\sigma_e\sigma_\zeta + \rho_{\nu\zeta}\sigma_\nu\sigma_\zeta \end{bmatrix}.$$

Assume that the prior belief over ${}^{i}_{t}$ in period t, ${}^{i}_{t|t}$, is independent of $\left\{{}^{i}_{s}, {}^{i}_{s}\right\}_{s>t}$ and normal:

$${}^{i}_{t|t} \sim N\left({}^{i}_{t|t}, \mathbf{P}_{t|t}\right).$$

Conditioning on period t information, we have

$$\begin{bmatrix} {}^{i}_{t+1|t} \\ \mathbf{Z}^i_{t+1|t} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{G}_1 + \mathbf{G}_2 {}^{i}_{t|t} \\ \overline{\mathbf{Z}}^i_{t+1|t} \end{bmatrix}, \begin{bmatrix} \mathbf{G}_2\mathbf{P}_{t|t}\mathbf{G}_2^T + \mathbf{V}_1 & \mathbf{G}_2\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_3 \\ \mathbf{B}_3\mathbf{P}_{t|t}\mathbf{G}_2^T + \mathbf{V}_3^T & \mathbf{B}_3\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_2 \end{bmatrix}\right)$$

$$\mathbf{Z}^i_{t+1|t} = \mathbf{B}_1 + \mathbf{B}_2\mathbf{Z}^i_t + \mathbf{B}_3 {}^{i}_{t|t}.$$

After observing $\mathbf{Z}^i_{t+1}$ in period $t+1$, the updated value for ${}^{i}_{t+1|t+1}$ will obey

$${}^{i}_{t+1|t+1} \sim N\left({}^{i}_{t+1|t+1}, \mathbf{P}_{t+1|t+1}\right)$$

$${}^{i}_{t+1|t+1} = \mathbf{G}_1 + \mathbf{G}_2 {}^{i}_{t|t} + \left(\mathbf{G}_2\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_3\right)\left(\mathbf{B}_3\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_2\right)^{-1}\left(\mathbf{Z}^i_{t+1} - \mathbf{B}_1 - \mathbf{B}_2\mathbf{Z}^i_t - \mathbf{B}_3 {}^{i}_{t|t}\right)$$

$$\mathbf{P}_{t+1|t+1} = \left(\mathbf{G}_2\mathbf{P}_{t|t}\mathbf{G}_2^T + \mathbf{V}_1\right) - \left(\mathbf{G}_2\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_3\right)\left(\mathbf{B}_3\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_2\right)^{-1}\left(\mathbf{B}_3\mathbf{P}_{t|t}\mathbf{G}_2^T + \mathbf{V}_3^T\right).$$

From (151), conditioned on period t we can write out

$$\begin{aligned}\log\left(R^i_{t+1}\right) - \log(\alpha) - (1-\alpha)\log\left(\overline{N}\right) = {} & (\alpha-1)a_0 + \mu_\eta + A_1\left(\log\left(R^i_t\right) - \log(\alpha) - (1-\alpha)\log\left(\overline{N}\right)\right) + \\ & A_2\left(\log\left(w^i_t\right) - \log(1-\alpha) + \alpha\log\left(\overline{N}\right)\right) - \\ & A_2\epsilon^i_{t|t} + (\rho_\eta - A_1)\eta^i_{t|t} + e_{t+1} + \zeta^i_{t+1}\end{aligned}$$

$$\begin{aligned}\log\left(w^i_{t+1}\right) - \log(1-\alpha) + \alpha\log\left(\overline{N}\right) = {} & \alpha a_0 + \mu_\epsilon + A_4\left(\log\left(R^i_t\right) - \log(\alpha) - (1-\alpha)\log\left(\overline{N}\right)\right) + \\ & A_5\left(\log\left(w^i_t\right) - \log(1-\alpha) + \alpha\log\left(\overline{N}\right)\right) + \\ & (\rho_\epsilon - A_5)\epsilon^i_{t|t} - A_4\eta^i_{t|t} + e_{t+1} + \nu^i_{t+1}.\end{aligned}$$

The forecast rules are obtained after collecting the constant terms.

Therefore, we obtain the following system of equations that describes the evolution of the prices and beliefs in the PI economy:

$$\begin{bmatrix} \log\left(R^i_{t+1}\right) \\ \log\left(w^i_{t+1}\right) \end{bmatrix} \sim N\left(\begin{bmatrix} E_t\left[\log\left(R^i_{t+1}\right)\right] \\ E_t\left[\log\left(w^i_{t+1}\right)\right] \end{bmatrix}, \mathbf{B}_3\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_2\right) \tag{152}$$

$$E_t\left[\log\left(R^i_{t+1}\right)\right] = A_0 + A_1\log\left(R^i_t\right) + A_2\log\left(w^i_t\right) - A_2\overline{\varepsilon}^i_{t|t} + \left(\rho_\eta - A_1\right)\overline{\eta}^i_{t|t}$$

$$E_t\left[\log\left(w^i_{t+1}\right)\right] = A_3 + A_4\log\left(R^i_t\right) + A_5\log\left(w^i_t\right) + \left(\rho_\varepsilon - A_5\right)\overline{\varepsilon}^i_{t|t} - A_4\overline{\eta}^i_{t|t}$$

$$\begin{bmatrix} \overline{\varepsilon}^i_{t+1|t+1} \\ \overline{\eta}^i_{t+1|t+1} \end{bmatrix} = \begin{bmatrix} \rho_\varepsilon\overline{\varepsilon}^i_{t|t} \\ \rho_\eta\overline{\eta}^i_{t|t} \end{bmatrix} + \mathbf{G}_3\left(\begin{bmatrix} \log\left(R^i_{t+1}\right) \\ \log\left(w^i_{t+1}\right) \end{bmatrix} - \begin{bmatrix} E_t\left[\log\left(R^i_{t+1}\right)\right] \\ E_t\left[\log\left(w^i_{t+1}\right)\right] \end{bmatrix}\right). \tag{153}$$

where $\mathbf{G}_3 = \left(\mathbf{G}_2\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_3\right)\left(\mathbf{B}_3\mathbf{P}_{t|t}\mathbf{B}_3^T + \mathbf{V}_2\right)^{-1}$. Letting the process for $\mathbf{P}_{t|t}$ converge to a constant yields the laws of motion in the main body of the paper. Since the Kalman filter recursion depends on endogenous variables its convergence is not ensured (see Baxter, Graham, and Wright 2007).

The algorithm for solving the FI agent's problem is modified from Krusell and Smith (1998) and Young (2010). The objective of the algorithm is to obtain the coefficients in the law of motion for aggregate capital (**??**). We divide the algorithm into three main parts. In summary, the first part is to solve for the value functions $\left(V^{FI}, V^{PI}\right)$ over a finite grid of $\left(k^i, \varepsilon^i, \eta^i, R^i, w^i\right)$, given a law of motion (**??**). The second part is to solve for the policy functions k' over a much finer grid of $\left(k^i, \varepsilon^i, \eta^i, R^i, w^i\right)$ using V^{FI} and V^{PI} from the first part. The third part is to simulate the time series of $\{K_t, MPK_t, MPN_t, z_t\}_{t=1}^T$ using the policy function from the second part and update the law of motion (**??**). This procedure is iterated from the first part using the updated law of motion until the coefficients (a_0, a_1, a_2) converge. The following subsections explain the algorithm in detail. For the PI agents simply substitute beliefs for actual idiosyncratic shock values and change the laws of motion as appropriate.

18.7.2 Part 1: Solving for $V^{FI}\left(k^i, \varepsilon^i, \eta^i, \log\left(R^i\right), \log\left(w^i\right)\right)$ and $V^{PI}\left(k^i, \varepsilon^i, \eta^i, \log\left(R^i\right), \log\left(w^i\right)\right)$

1. Discretize the space of $\left\{k^i, \varepsilon^i, \eta^i, \log\left(R^i\right), \log\left(w^i\right)\right\}$ and denote this grid $\{\mathbf{k1}, \varepsilon\mathbf{1}, \eta\mathbf{1}, \mathbf{R1}, \mathbf{w1}\}$.

2. Guess $\{a_j\}_{j=0}^2$ in (**??**). Then compute $\{A_j\}_{j=0}^5$ and the parameters for belief dynamics of ε^i and η^i in the PI economy.

3. Guess the initial value functions V_0^{FI} and V_0^{PI} on the discretized grids of $\{\mathbf{k1}, \varepsilon\mathbf{1}, \eta\mathbf{1}, \mathbf{R1}, \mathbf{w1}\}$.

4. Given the above initial guess, solve the FI and PI agents' problems to get the policy functions for k' and use them to get V_1^{FI} and V_1^{PI}. Iterate until V^{FI} and V^{PI} converge.

18.7.3 Part 2: Update law of motion

1. Simulate a long time series of $\{z_t\}_{t=1}^T$.

2. Assign an initial distribution of N households whose state variables are $\left\{k_1^i, \varepsilon_1^i, \eta_1^i\right\}$ for the FI economy and $\left\{k_1^i, \varepsilon_1^i, \eta_1^i, \varepsilon_{1|1}^i, \eta_{1|1}^i\right\}$ for the PI economy. For the PI economy we set $\varepsilon_1^i = \varepsilon_{1|1}^i$ and $\eta_1^i = \eta_{1|1}^i$.

3. Given the distribution in period 1 and z_1, we can compute $\left\{\log\left(R_1^i\right), \log\left(w_1^i\right)\right\}$ for each households.

4. Simulate next period distribution of realized ε_2^i and η_2^i using (**??**) and (**??**). Then use the policy function k' from the second part to get the next period distribution $\left\{k_2^i, \varepsilon_2^i, \eta_2^i\right\}$. Since the state variables do not generally lie on the grid, we use linear interpolation to evaluate k'. For the PI economy, use z_2 to compute $\log\left(MPK_2\right)$, $\log\left(MPN_2\right)$, and $\left\{\log\left(R_2^i\right), \log\left(w_2^i\right)\right\}$. Then use (153) to update beliefs $\left\{\varepsilon_{2|2}^i, \eta_{2|2}^i\right\}$.

5. Repeat from step 3 for $\{z_t\}_{t=3}^T$.

6. Drop the first B simulation periods and use OLS on the equilibrium time series of $\{K_t, z_t\}_{t=B+1}^T$ to get a new value for $\{a_j\}_{j=0}^2$.

7. Update these coefficients using the updating rule: $x_{update} = \lambda x_{new} + (1-\lambda) x_{old}$ and repeat from step 3 in part 1 until all the coefficients $\{a_j\}_{j=0}^2$ converge.

18.8 Analytic Frequency Policy Iteration

Han, Tan, and Wu (2021) consider the linear dynamic system

$$\sum_{k=0}^{l} A_k y_{t-k} + \sum_{k=0}^{h} B_k E_t y_{t+k} = 0$$

where

$$y_t = \begin{bmatrix} x_t \\ \overline{x}_t \\ s_t \\ w_t \end{bmatrix}$$

$$A_k = \begin{bmatrix} A_k^x & A_k^{\overline{x}} & A_k^s & A_k^w \end{bmatrix}$$

$$B_k = \begin{bmatrix} B_k^x & B_k^{\overline{x}} & B_k^s & B_k^w \end{bmatrix}$$

and

$$w_t = \sum_{k=1}^{p_w} C_k^w w_{t-k} + \sum_{k=0}^{q_w} D_k^w \epsilon_{t-k}$$

is an exogenous ARMA process of shocks and

$$s_t = \sum_{k=1}^{p_s} C_k^s s_{t-k} + \sum_{k=0}^{q_s} D_k^s \epsilon_{t-k}$$

are signals. $\overline{x}_t$ is the cross-sectional average of x_t.

Information set

$$\Omega_t = s^t \vee x_e^t$$

where

$$s^t = \{s_t, s_{t-1}, ...\}$$

$$x_e^t = \{x_{e,t}, x_{e,t-1}, ...\}$$

$$x_{e,t} \subset x_t$$

are the history of exogenous and endogenous signals.

Supposing only x_t appears, then

$$A_0 x_t = -\sum_{k=1}^{l} A_k x_{t-k} - \sum_{k=0}^{h} B_k E_t x_{t+k}$$

and the solution takes the form

$$x_t = \Gamma^x(L)\,\epsilon_t$$
$$\Gamma^x(L) = \sum_{k=1}^{\infty} \Gamma_k^x L^k$$

and the signal takes the form

$$s_t = C^s(L)^{-1} D^s(L)\,\epsilon_t$$
$$\equiv \Gamma^s(L)\,\epsilon_t$$
$$C^s(L) = I - \sum_{k=1}^{p_s} C_k^s L^k$$
$$D^s(L) = \sum_{k=0}^{q_s} D_k^s L^k.$$

The spectrum is

$$S_x(\omega) = \frac{1}{2\pi}\Gamma^x\left(e^{-i\omega}\right)\Sigma_\epsilon\Gamma^x\left(e^{i\omega}\right)^T.$$

Signals:

$$\Omega_t = \begin{bmatrix} x_{e,t} \\ s_t \end{bmatrix} = \begin{bmatrix} \Gamma^{x,e}(L) \\ \Gamma^s(L) \end{bmatrix}\epsilon_t = \begin{bmatrix} \Gamma^{x,e}(L) \\ C^s(L)^{-1} D^s(L) \end{bmatrix}\epsilon_t = \Gamma^\Omega(L)\,\epsilon_t.$$

Using the Weiner-Hopf forecasting formula, we get

$$E\left[x_{t+k}|\Omega^t\right] = \left[L^{-k}\Gamma^x(L)\,\Sigma_\epsilon\Gamma^\Omega\left(L^{-1}\right)^T\left(\widetilde{\Gamma}\left(L^{-1}\right)^T\right)^{-1}\right]_+ \Sigma_u^{-1}\widetilde{\Gamma}^\Omega(L)^{-1}\,\Omega_t$$
$$= F_k^s\left(\Gamma^x(L)\right)\Omega_t$$
$$= F_k^s\left(\Gamma^x(L)\right)\Gamma^\Omega(L)\,\epsilon_t,$$

where $\widetilde{\Gamma}(z)$ is a spectral factorization of the spectrum and $[\cdot]_+$ is the annihilator operator for negative values. Therefore, the policy functions $\Gamma^x(z)$ solve

$$A_0\Gamma^x(z) = -\sum_{k=1}^{l} A_k z^k \Gamma^x(z) - \sum_{k=0}^{h} B_k F_k^s\left(\Gamma^x(z)\right)\Gamma^\Omega(z).$$

Suppose that

$$x_t \approx \sum_{k=1}^{p_x} C_k^x x_{t-k} + \sum_{k=0}^{q_x} D_k^x \epsilon_{t-k},$$

which is a rational function in the frequency domain

$$\Gamma^x(z) \approx C^x(z)^{-1} D^x(z)$$
$$C^x(z) = I - \sum_{k=1}^{p_x} C_k^x z^k$$
$$D^x(z) = \sum_{k=0}^{q_x} D_k^x z^k.$$

Any VARMA process that is covariance stationary can be approximated arbitrarily well by VARMA processes, in the mean-square sense.

Definition 132 *A function f is **rational** if it can be written as*

$$f(x) = \frac{P(x)}{Q(x)}$$

*where P and Q are polynomials and Q is not the zero function. f is a **proper rational function** if the order of P is not greater than the order of Q.*

Theorem 133 *Define the set of $n_x \times n_\epsilon$ matrices of proper rational analytic functions that correspond to VARMA$(k, k-1)$ processes as $\mathbf{R}_k$, with typical element*

$$R_k^{m,n} = c^{m,n} \frac{\prod_{j=1}^{k-1} \left(1 - b_j^{m,n} z\right)}{\prod_{i=1}^{k} \left(1 - a_i^{m,n} z\right)}$$

with (a, b, c) being complex coefficients and $|a_i^{m,n}| < 1$. Then $\cup_{k \in \mathcal{N}} \mathbf{R}_k$ is dense in $\mathbf{H}^2_{n_x \times n_\epsilon}(\mathcal{D})$, the Hardy space of analytic functions on the unit disk.

Set up a grid $\{z_j\} \in (-1, 1)$, the real line inside the complex open unit disk.

Guess initial policy functions $\{\Gamma^x(z_j)\}_{j=1}^N$. Obtain the matrices (C, D) from

$$C^x(z_j)^{-1} D^x(z_j) = \Gamma^x(z_j)$$

using projection in the frequency domain. Now compute

$$\left\{ z_j^k \Gamma^x(z_j) \right\}_{k=1}^l$$
$$\left\{ F_k^s\left(\Gamma^x(z_j)\right) \Gamma^\Omega(z_j) \right\}_{k=0}^h.$$

Now we can update Γ^x using

$$\widehat{\Gamma}^x(z_j) = -A_0^\dagger \sum_{k=1}^{l} A_k z_j^k \Gamma^x(z_j) - A_0^\dagger \sum_{k=0}^{h} B_k F_k^s\left(\Gamma^x(z_j)\right)\Gamma^\Omega(z_j),$$

where $A_0^\dagger$ is the Moore-Penrose inverse of A_0:

$$A_0^\dagger = \left(A^T A\right)^{-1} A^T.$$

Next, one can show that solving the model on the real part of the open unit disk is equivalent to solving it on the entire open unit disk.

Theorem 134 *Let $\Gamma^x(z)$ be any analytic function in the Hardy space $\mathbf{H}^2_{n_x\times n_\epsilon}(\mathcal{D})$ such that there exists $\Psi(z) \in \mathbf{H}^2_{n_x\times n_\epsilon}(\mathcal{D})$ with the property that $\Gamma^x(z) = \Psi(z)\ \forall z \in (-1,1)$. Then $\Gamma^x = \Psi$ on the entire open unit disk.*

The next obstacle is the computation of the annihilation operator $[\cdot]_+$ in F^s. Here, we can employ an inverse discrete Fourier transform.

Theorem 135 *Define the function*

$$\Theta(z) \equiv z^{-k}\Gamma^x(z)\,\Sigma_\epsilon\Gamma^\Omega\left(z^{-1}\right)^T\left(\widetilde{\Gamma}^\Omega\left(z^{-1}\right)^T\right)^{-1}.$$

The Laurent expansion is

$$\Theta(z) = \sum_{k=-\infty}^{\infty}\Theta_k z^k$$

and the annihilation is

$$[\Theta(z)]_+ = \sum_{k=0}^{\infty}\Theta_k z^k.$$

Suppose $\Gamma^x(z)$ is rational and the spectral factorization $\widetilde{\Gamma}^\Omega(z)$ is invertible on the closed unit disk. Then the coefficient matrices $\{\Theta_k\}_{k=0}^\infty$ in the annihilation of $\Theta(z)$ are given by the inverse discrete Fourier transform around the unit circle $\mathcal{T}$:

$$\begin{aligned}\Theta_k &= \frac{1}{2\pi i}\oint_{\mathcal{T}}\Theta(z)\,z^{-k}\frac{dz}{z}\\ &\approx \frac{1}{N}\sum_{n=0}^{N-1}\Theta\left(\exp\left(-i\frac{2\pi n}{N}\right)\right)\exp\left(i\frac{2\pi n}{N}k\right), k\in\{0,1,...,N/2-1\}\end{aligned}$$

for some N such that, given $\epsilon > 0$, we have

$$\Theta_N(z) = \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} \Theta_k z^k$$

$$\|\Theta_N(z) - \Theta(z)\|_2 < \epsilon.$$

The projection step involves solving the equation system

$$\begin{bmatrix} z_1\Gamma^x(z_1)^T & \cdots & z_1^{p_x}\Gamma^x(z_1)^T & I & z_1 I & \cdots & z_1^{q_x} I \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ z_N\Gamma^x(z_N)^T & \cdots & z_N^{p_x}\Gamma^x(z_N)^T & I & z_N I & \cdots & z_N^{q_x} I \end{bmatrix} \begin{bmatrix} C_1^x \\ \vdots \\ C_{p_x}^x \\ D_0^x \\ \vdots \\ D_{q_x}^x \end{bmatrix} = \begin{bmatrix} \Gamma^x(z_1)^T \\ \vdots \\ \Gamma^x(z_N)^T \end{bmatrix}$$

$$\Pi_0 \Phi = \Pi_1.$$

Unfortunately, in general the order (p_x, q_x) is not known in advance and in fact can vary across iterations. Using a very loose order and allowing for zeros adds unnecessary cost, so we will look for the lowest possible order.

First, start with a loose conjecture $(\overline{p}, \overline{q})$ and identify the lowest possible orders $\left(\underline{p}, \underline{q}\right)$: set $l = 0$ and $r = \overline{p}$ and fix $\overline{q}$. If $l \geq r$, then $\underline{p} = r$ and terminate. Else, set

$$m = \left\lfloor \frac{l + r}{2} \right\rfloor$$

and check

$$\left\| \left(I - U_0 U_0^T\right) U_1 \right\| < \epsilon$$

where

$$\Pi_0 = U_0 S_0 V_0^T$$

$$\Pi_1 = U_1 S_1 V_1^T$$

are singular value decompositions, then set $r = m$, else set $l = m + 1$ and repeat. Now do the same for $\underline{q}$ and set

$$p_x = \underline{p}$$

$$q_x = \underline{q}.$$

That is, use the lowest orders consistent with fitting the "data" to some fixed tolerance. Now compute

$$\Phi = V_0 S_0^{-1} U_0^T \Pi_1.$$

The spectral factorization can be obtained by the Kalman filter. Write the state space representation of the information set as

$$\Omega_t = \sum_{k=1}^{r} C_k \Omega_{t-k} + \sum_{k=0}^{r-1} D_k \epsilon_{t-k}$$

where $r = \max\{p, q+1\}$, $C_k = 0$ if $k > p$, and $D_k = 0$ if $k > q$. Then the latent states satisfy

$$\begin{bmatrix} \xi_{1,t} \\ \xi_{2,t} \\ \vdots \\ \xi_{r-1,t} \\ \xi_{r,t} \end{bmatrix} = \begin{bmatrix} C_1 & I & 0 & \cdots & 0 \\ C_2 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ C_{r-1} & 0 & 0 & \cdots & I \\ C_r & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \xi_{1,t-1} \\ \xi_{2,t-1} \\ \vdots \\ \xi_{r-1,t-1} \\ \xi_{r,t-1} \end{bmatrix} + \begin{bmatrix} D_{1,t} \\ D_{2,t} \\ \vdots \\ D_{r-1,t} \\ D_{r,t} \end{bmatrix} \epsilon_t$$

$$\xi_t = F \xi_{t-1} + v_t$$

where

$$\begin{aligned} \Omega_t &= \begin{bmatrix} I & 0 & \cdots & 0 \end{bmatrix} \xi_t \\ &= H \xi_t \\ v_t &\sim N(0, \Sigma_v) \\ \Sigma_v &= \begin{bmatrix} D_{1,t} \\ D_{2,t} \\ \vdots \\ D_{r-1,t} \\ D_{r,t} \end{bmatrix} \Sigma_\epsilon \begin{bmatrix} D_{1,t} \\ D_{2,t} \\ \vdots \\ D_{r-1,t} \\ D_{r,t} \end{bmatrix}^T. \end{aligned}$$

The Wold representation is given by

$$\begin{aligned} \Omega_t &= \widetilde{\Gamma}^{\Omega}(L) u_t \\ &= \left(I + H(1 - FL)^{-1} KL \right) u_t \end{aligned}$$

where

$$\Sigma_u = HPH^T$$

and (K, P) satisfy the stationary Kalman filter equations

$$\begin{aligned} P &= FPF^T - FPH^T \left(HPH^T\right)^{-1} HPF^T + \Sigma_v \\ K &= FPH^T \left(HPH^T\right)^{-1}. \end{aligned}$$

19 Conclusion

If you made it this far, you probably deserve a medal of some sort; hopefully, that medal is a successful career applying the tools discussed here.

References

[1] Achdou, Yves, Jiequn Han, Jean-Michel Lasry, Pierre-Louis Lions, and Benjamin Moll (2020), "Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach," forthcoming, *Review of Economic Studies*, manuscript available at https://benjaminmoll.com/wp-content/uploads/2019/07/HACT.pdf.

[2] Ahn, SeHyoun (2019), "Computing the Distribution: Adaptive Finite Volume Methods for Economic Models with Heterogeneous Agents," manuscript available at https://sehyoun.com/static/2019_06_12_Ahn_adaptive_finite_volume.pdf.

[3] Arnoud, Antoine, Fatih Guvenen, and Tatjana Kleineberg (2019), "Benchmarking Global Optimizers," manuscript available at https://fguvenendotcom.files.wordpress.com/2019/09/agk2019-september-nber-submit.pdf

[4] Atolia, Manoj and Edward Buffie (2009a), "Smart Forward Shooting," *Computational Economics* **33(1)**, pp. 1-30.

[5] Atolia, Manoj and Edward Buffie (2009b), "Reverse Shooting Made Easy: Automating the Search for the Global Nonlinear Saddle Path," *Computational Economics* **34(3)**, pp. 273-308.

[6] Atolia, Manoj and Edward Buffie (2011), "Solving the Unit Root Problem in Models with an Exogenous World Market Interest Rate," *Macroeconomic Dynamics* **15(5)**, pp. 681-712.

[7] Bacchetta, Phillipe, Eric van Wincoop, and Eric R. Young (2020), "Infrequent Random Portfolio Decisions in an Open Economy Model," forthcoming *Review of Economic Studies*, manuscript available at www.people.virginia.edu/~ey2d.

[8] Berndt, Ernst R., Bronwyn Hall, Robert E. Hall, and Jerry Hausman (1974), "Estimation and Inference in Nonlinear Structural Models, " *Annals of Economic and Social Measurement* **3(4)**, pp. 653-665.

[9] Bewley, Truman (undated), "Interest Bearing Money and the Equilibrium Stock of Capital," manuscript.

[10] Boyd, John H. (1990), "Recursive Utility and the Ramsey Problem, " *Journal of Economic Theory* **50**, pp. 326-345.

[11] Brumm, Johannes and Michael Grill (2014), "Computing Equilibria in Dynamic Models with Occasionally-Binding Constraints," *Journal of Economic Dynamics and Control* **38(1)**, pp. 142-160.

[12] Carroll, Christopher D. (2006), "The Method of Endogenous Gridpoints for Solving Dynamic Stochastic Optimization Problems," *Economics Letters* **91(3)**, pp. 312-320.

[13] Chang, Yongsung and Sunill Kim (2007), "Heterogeneity and Aggregation: Implications for Labor-Market Fluctuations," *American Economic Review* **97**, pp. 1939-1956.

[14] Cheng, Jian and Marek J. Druzdzel (2000), "Computational Investigation of Low-Discrepancy Sequences in Simulation Algorithms for Bayesian Networks," *Uncertainty in Artificial Intelligence Proceedings*, pp. 72-81.

[15] den Haan, Wouter J. and Joris de Wind (2012), "Nonlinear and Stable Perturbation-Based Approximations," manuscript available at www.wouterdenhaan.com/papers.

[16] den Haan, Wouter J., Michal L. Kobielarz, and Pontus Rendahl (2016), "Exact-Present Solution with Consistent-Future Approximation: A Gridless Algorithm to Solve Stochastic Dynamic Models," manuscript available at www.wouterdenhaan.com/papers.

[17] Farmer, Leland E. and Alexis Akira Toda (2017), "Discretizing Nonlinear, Non-Gaussian Stochastic Processes with Exact Conditional Moments," *Quantitative Economics* **8**, pp. 651-683.

[18] Fernández-Villaverde, Jesús and Juan F. Rubio-Ramírez (2006), "Solving DSGE Models with Perturbation Methods and a Change of Variables," *Journal of Economic Dynamics and Control* **30(12)**, pp. 2509-2531.

[19] Flodén, Martin (2008), "A Note on the Accuracy of Markov-Chain Approximations to Highly Persistent AR(1) Processes," *Economics Letters* **99**, pp. 516-520.

[20] Foerster, Andrew, Juan F. Rubio-Ramírez, Daniel F. Waggoner, and Tao Zha (2016), "Perturbation Methods for Markov-Switching Dynamic Stochastic General Equilibrium Models,"*Quantitative Economics* **7**, pp. 637-669.

[21] Garcia, C.B. and Willard I. Zangwill (1981), *Pathways to Solutions, Fixed Points, and Equilbria*, Prentice-Hall.

[22] Gordon, Grey and Shi Qiu (2018), "A Divide and Conquer Algorithm for Exploiting Policy Function Monotonicity," *Quantitative Economics* **9(2)**, pp. 521-540.

[23] Gospodinov, Nikolay, and Damba Lkhagvasuren (2014), "A Moment-Matching Method for Approximating Vector Autoregressive Processes by Finite-State Markov Chains," *Journal of Applied Econometrics* **29(5)**, pp. 845-859.

[24] Han, Zhao, Fei Tan, and Jieran Wu (2020), "Analytic Policy Function Iteration," manuscript available at https://sites.google.com/site/jieranwu/research.

[25] Hull, Isaiah (2015), "Approximate Dynamic Programming with Post-Decision States as a Solution Method for Dynamic Economic Models," *Journal of Economic Dynamics and Control* **55**, pp. 57-70.

[26] Judd, Kenneth L. (1992), "Projection Methods for Solving Aggregate Growth Models," *Journal of Economic Theory* **58**, pp. 410-452.

[27] Judd, Kenneth L. (2002), "The Parametric Path Method: An Alternative to Fair-Taylor and L-B-J for Solving Perfect Foresight Models," *Journal of Economic Dynamics and Control* **26**, pp. 1557-1583.

[28] Judd, Kenneth L. and Yongyang Cai (2013), "Shape-Preserving Dynamic Programming," *Mathematical Methods of Operations Research* **77**, pp. 407-421.

[29] Judd, Kenneth L. and Sy-Ming Guu (1997), "Asymptotic Methods for Aggregate Growth Models," *Journal of Economic Dynamics and Control* **6(1)**, pp. 1025-1042.

[30] Judd, Kenneth L. and Andrew Solnick (1994), "Numerical Dynamic Programming with Shape-Preserving Splines" manuscript available at https://web.stanford.edu/~judd/papers/dpshape.pdf.

[31] King, Robert G., Charles I. Plosser, and Sergio T. Rebelo (2002), "Production, Growth, and Business Cycles: Technical Appendix," *Computational Economics* **20(1)**, pp. 87-116.

[32] Klein, Paul (2000), "Using the Generalized Schur Form to Solve a Multivariate Linear Rational Expectations Model," *Journal of Economic Dynamics and Control* **24(10)**, pp. 1405-1423.

[33] Krusell, Per, Burhanettin Kuruşçu, and Anthony A. Smith, Jr. (2001), "Equilibrium Welfare and Government Policy with Quasi-Geometric Discounting," *Journal of Economic Theory* **105(1)**, pp. 42-72.

[34] Krusell, Per and Anthony A. Smith, Jr. (1998), "Income and Wealth Heterogeneity in the Macroeconomy," *Journal of Political Economy* **106(6)**, pp. 867-896

[35] Kopecky, Karen A. and Richard Suen (2010), "Finite State Markov-Chain Approximations to Highly-Persistent Processes," *Review of Economic Dynamics* **13(3)**, pp. 701-714.

[36] Kydland, Finn E. (1975), "Noncooperative and Dominant Player Solutions in Discrete Dynamic Games," *International Economic Review* **16**, pp. 321-335.

[37] Kydland, Finn E. and Edward C. Prescott (1977), "Rules Rather Than Discretion: The Inconsistency of Optimal Plans," *Journal of Political Economy* **85**, pp. 473-491.

[38] Kydland, Finn E. and Edward C. Prescott (1980), "Dynamic Optimal Taxation, Rational Expectations and Optimal Control," *Journal of Economic Dynamics and Control* **2**, pp. 79-91.

[39] Kydland, Finn E. and Edward C. Prescott (1982), "Time-to-Build and Aggregate Fluctuations," *Econometrica* **50**, pp. 1345-1370.

[40] Lantoine, Gregory, Ryan P. Russell, and Thierry Dargent (2012), "Using Multicomplex Numbers for Automatic Computation of High-Order Derivatives," *ACM Transactions on Mathematical Software* **38(3)**, Article 16.

[41] Lee, Jae Won and Woong Yong Park (2021), "Solving Reduced-Form Linear Rational Expectations Models," manuscript.

[42] Lee, Jae Won and Woong Yong Park (2021), "System Reduction of Dynamic Stochastic General Equilibrium Models Solved by Gensys," *Economics Letters* **199**.

[43] Levintal, Oren (2017), "Fifth-Order Perturbation Solutions to DSGE Models," *Journal of Economic Dynamics and Control* **80(1)**, pp. 1-16.

[44] Levintal, Oren (2018), "Taylor Projection: A New Solution Method for Dynamic General Equilibrium Models," *International Economic Review* **59**, pp. 1345-1373.

[45] McGrattan, Ellen R. (1996), "Solving the Stochastic Growth Model with a Finite Element Method," *Journal of Economic Dynamics and Control* **20(1)**, pp. 19-42.

[46] Miao, Jianjun, Jieran Wu, and Eric R. Young (2020), "Multivariate Rational Inattention" forthcoming *Econometrica*, manuscript available at https://sites.google.com/site/jieranwu/research.

[47] Möller, H.M. (1979), "Lower Bounds for the Number of Nodes in Cubature Formulas," *Numerische Integration* **45**, pp. 221-230.

[48] Moro, Boris (1995), "The Full Monte," *Risk* **8(2)**, pp. 57-58.

[49] Odell, P.L. and A.H. Feiveson (1966), "A Numerical Procedure to Generate a Sample Covariance Matrix," *Journal of the American Statistical Association* **61(313)**, pp. 199-203.

[50] Pál, Jenő and John Stachurski (2013), "Fitted Value Function Iteration with Probability One Contractions," *Journal of Economic Dynamics and Control* **37**, pp. 251-264.

[51] Porapakkarm, Ponpoje and Eric R. Young (2009), "Information Heterogeneity in the Macroeconomy," manuscript.

[52] Rendahl, Pontus (2015), "Inequality Constraints and Euler Equation-based Solution Methods," *Economic Journal* **125(585)**, pp. 1110-1135.

[53] Rouwenhorst, K. Geert (1995), "Asset Pricing Implications of Business Cycle Models," in Cooley, Thomas F. (ed.), *Frontiers of Business Cycle Research*, Princeton University Press, pp. 294-330.

[54] Santos, Manuel S. and Adrian Peralta-Alva (2005), "Accuracy of Simulations for Dynamic Models," *Econometrica* **73**, pp. 1939-1975.

[55] Schmitt-Grohé, Stephanie and Martín Uribe (2004), "Solving Dynamic General Equilibrium Models using a Second-Order Approximation to the Policy Function," *Journal of Economic Dynamics and Control* **28(4)**, pp. 755-775.

[56] Stachurski, John (2008), "Continuous State Dynamic Programming via Nonexpansive Approximation," *Computational Economics* **31(2)**, pp. 141-160.

[57] Stokey, Nancy L. and Robert E. Lucas, Jr. with Edward C. Prescott (1989), *Recursive Methods for Economic Dynamics*, Harvard University Press.

[58] Tauchen, George (1986), "Finite-State Markov Approximations to Univariate and Vector Autoregressions," *Economics Letters* **20**, pp. 177-181.

[59] Tauchen, George and R. Hussey (1991), "Quadrature-Based Methods for Obtaining Approximate Solutions to Nonlinear Asset Pricing Models," *Econometrica* **59**, pp. 371-396.

[60] White, Matthew N. (2015), "The Method of Endogenous Gridpoints in Theory and Practice," *Journal of Economic Dynamics and Control* **60(1)**, pp. 26-41.

[61] Young, Eric R. (2007), "Generalized Quasi-Geometric Discounting," *Economics Letters* **96**, pp. 343-350.

[62] Young, Eric R. (2010), "Solving the Incomplete Markets Model with Aggregate Uncertainty Using the Krusell-Smith Algorithm and Non-Stochastic Simulations," *Journal of Economic Dynamics and Control* **34(1)**, pp. 36-41.

[63] Young, Eric R. (2012), "Robust Policymaking in the Face of Sudden Stops,"*Journal of Monetary Economics* **59(5)**, pp. 512-527.