Recursive, Projection, and Perturbation Methods for Solving DSGE Models*

David Murakami†

10th May 2021

Contents

*These are some brief revision notes on solving DSGE models, and reference texts include but are not limited to: Adda and Cooper (2003), Judd (1998), McCandless (2008), and – of course – Ljungqvist and Sargent (2018). I've also relied on notes from Petr Sedláček and Wouter den Haan.

†St Cross College, University of Oxford. Email: david.murakami@economics.ox.ac.uk

1 Recursive Deterministic Models

Here we will look at recursive methods, starting with a deterministic setting. With recursive methods, one looks for a policy function, a mapping from the initial conditions, given by the past or the present, to a set of decisions about what to do with the variables we can choose during this periods. Because these are normally infinite horizon problems, how we will want to behave in the future matters in determining what we want to do today. Since what one will want to do in the future matters and the whole future time path can be determined, the recursive methods we describe are also known as dynamic programming.

1.1 States and controls

It is helpful to separate the set of variables that we are using into state variables and control variables. In some period t, the state variables are those whose values are already determined, either by our actions in the past or by some other process (such as nature).[1] Normally, for a growth model of the type we have been working with, the capital stock that we inherit from the past must be considered a state variable. One might also think that the technology level in each period is determined by nature and therefore, in any period, the agents living in that economy must take it as a given.

The control variables in period t are those variables whose values individuals explicitly choose in that period with the goal of maximising some objective function. Frequently, a modeller has a choice about which variables will be states and which will be control variables.

1.2 A simple example: Robinson Crusoe model

Consider the simple version of the Robinson Crusoe (RC) model. In that model, Robinson Crusoe wants to maximise:

$$\max \sum_{i=0}^{\infty} \beta^i u(c_{t+i}),$$

subject to the constraints:

$$k_{t+1} = (1-\delta)k_t + i_t,$$

and

$$y_t = f(k_t) = c_t + i_t.$$

[1] We often refer to an equation which describes the transition of a state variable as a "law of motion".

We do some familiar rearranging:

$$\begin{aligned} & i_t = k_{t+1} + (1-\delta)k_t, \\ \implies & c_t = f(k_t) - i_t \\ & \quad = f(k_t) + (1-\delta)k_t - k_{t+1}, \end{aligned}$$

which allows us to write Robinson Crusoe's problem as:

$$\max \sum_{i=0}^{\infty} \beta^i u(f(k_{t+i}) + (1-\delta)k_{t+i} - k_{t+i+1}).$$

In this case, k_{t+1} is the control variable in period t.

Whatever our choice of a control variable, there must be enough budget constraints or market conditions so that the values of the rest of the relevant variables in period t are determined. What may be surprising is that the choice of control variables can matter in how easily we can solve our models. Some choices will simply be more convenient than others.

1.3 The value function

Assume that it is possible to calculate the value of the discounted value of utility that an agent receives when that agent is maximising the infinite horizon objective function subject to the budget constraints. For Robinson Crusoe, this value is clearly a function of the initial per worker capital stock, k_t. As shown above, we can write out a version of this problem where the RC economy us using the capital stock to be carried over to the next period, k_{t+1}, as the control variable. For that example, the value utility is equal to:

$$V(k_t) = \max_{\{k_s\}_{s=t+1}^{\infty}} \sum_{i=0}^{\infty} \beta^i u(f(k_{t+i}) + (1-\delta)k_{t+i} - k_{t+1+i}), \tag{1}$$

where we denote the value of the discounted utility by $V(k_t)$, to stress that it is a function of the value of the initial capital stock, k_t. For any value of k_t, limited to the appropriate domain, the value of the value function, $V(k_t)$, is the discounted value of utility when the maximisation problem has been solved and when k_t was the initial capital stock.

Since $V(k_t)$ is a function, its value can be found for any permitted value of k_t. In particular, the value of the function can be found for the value of k_{t+1} that was chosen in period t. This is possible because the economy is recursive as mentioned above. In period $t+1$, the value of k_{t+1} is given (it's a state variable) and the problem to be solved is simply the maximisation of utility beginning in period

$t+1$:

$$V(k_{t+1}) = \max_{\{k_s\}_{s=t+2}^{\infty}} \sum_{i=0}^{\infty} \beta^i u(f(k_{t+i+1}) + (1-\delta)k_{t+i+1} - k_{t+i+2}), \tag{2}$$

and its value, $V(k_{t+1})$, is a function of the stock of capital per worker at $t+1$.

By separating the period t problem from that of future periods, we can rewrite the value function of (1) as:

$$V(k_t) = \max_{k_{t+1}} \left[u\left(f(k_t) + (1-\delta)k_t - k_{t+1}\right) + \max_{\{k_s\}_{s=t+2}^{\infty}} \sum_{i=1}^{\infty} \beta^i u(f(k_{t+i}) + (1-\delta)k_{t+i} - k_{t+i+1} \right].$$

Adjusting the timing of the second discounted term gives:

$$V(k_t) = \max_{k_{t+1}} \left[u\left(f(k_t) + (1-\delta)k_t - k_{t+1}\right) + \beta \max_{\{k_s\}_{s=t+2}^{\infty}} \sum_{i=0}^{\infty} \beta^i u(f(k_{t+i+1}) + (1-\delta)k_{t+i+1} - k_{t+i+2} \right].$$

The second discounted term is nothing but the value function $V(k_{t+1})$ that we wrote in (2). Making the substitution, the value function in (1) can be written recursively as:

$$V(k_t) = \max_{k_{t+1}} \left[u\left(f(k_t) + (1-\delta)k_t - k_{t+1}\right) + \beta V(k_{t+1}) \right] \tag{3}$$

Equations in this form are known as Bellman equations (Bellman 1957). It presents exactly the same problem as in (1), but written in recursive form. Writing out the problem recursively makes it conceptually simpler. The value of the choice variable, k_{t+1}, is being chosen to maximise an objective function of only a single period. The period is reduced from one of infinite dimensions (picking many future k's) to one of only one dimension.

But there is a problem: both the time t one-period problem, $u(\cdot)$, and the discounted value function evaluated at k_{t+1}, $\beta V(k_{t+1})$, are included. The value of $V(k_{t+1})$ is not yet known. If it were known, then the value of the function $V(k_t)$ would also be known – it is the same function – and solving the maximisation problem at time t would be trivial.

To proceed, we assume that the value function $V(\cdot)$ exists and has a first derivative. We can then proceed with the one-period maximisation problem (3) by taking the derivative with respect to k_{t+1} to yield the FOC:

$$\frac{\partial V(k_t)}{\partial k_{t+1}} = -u'\left(f(k_t) + (1-\delta)k_t - k_{t+1}\right) + \beta V'(k_{t+1}) = 0. \tag{4}$$

Unfortunately, this isn't very helpful. We do not know how or what $V'(\cdot)$ looks like.

Under certain conditions, and this model has been written so that the conditions hold, one can find the derivative of $V(\cdot)$ simply by taking the partial derivative of the value function as written in

equation (3) with respect to k_t. Theorems that provide the sufficient conditions for getting a derivative and that tell us how to find it are called envelope theorems. This partial derivative is:

$$\frac{\partial V(k_t)}{\partial k_t} = u'\left(f(k_t) + (1-\delta)k_t - k_{t+1}\right)\left(f'(k_t) + (1-\delta)\right).$$

Now we can substitute this result into (4) for $V'(k_{t+1})$:

$$\frac{\partial V(k_t)}{\partial k_{t+1}} = -u'\left(f(k_t) + (1-\delta)k_t - k_{t+1}\right) + \beta u'(f(k_t) + (1-\delta)k_t - k_{t+1})(f'(k_t) + (1-\delta)) = 0,$$

which gives us the familiar consumption Euler equation:

$$\frac{u'(c_t)}{u'(c_{t+1})} = \beta(f'(k_{t+1}) + (1-\delta)).$$

In the steady state, where $c_t = c_{t+1} = \bar{c}$, this Euler equation yields:

$$\frac{1}{\beta} - (1-\delta) = f'(\bar{k}).$$

Using recursive methods, we find that for a stationary state, the rental rate on capital is equal to the net interest rate implicit in the discount factor plus the depreciation rate.

1.4 A general version

Let $\mathbf{X}_t$ be a vector of the period t state variables and let $\mathbf{Y}_t$ be a vector of the control variables. In the example above, $\mathbf{X}_t = [k_t]$ and $\mathbf{Y}_t = [k_{t+1}]$.[2] Let $F(\mathbf{X}_t, \mathbf{Y}_t)$ be the time t value of the objective function that is to be maximised. In the example economy, the objective function is the utility function. Given initial values of the state variables, $\mathbf{X}_t$, the problem to be solved at time t is the value function:

$$V(\mathbf{X}_t) = \max_{\{\mathbf{Y}_s\}_{s=t+1}^{\infty}} \sum_{i=0}^{\infty} \beta^i F(\mathbf{X}_{t+i}, \mathbf{Y}_{t+i}),$$

subject to the set of budget constraints given by:

$$\mathbf{X}_{s+1} = G(\mathbf{X}_s, \mathbf{Y}_s),$$

[2]Notice that in this particular case, it turns out that $\mathbf{Y}_t = \mathbf{X}_{t+1}$, that the control at time $t$becomes the state variable at time $t+1$. This is one of the aspects of the specific example that makes it simple to solve and should not be considered the normal relationship between control and state variables.

for $s \geq t$. Using the same recursive argument that we used above, we can write the value function as a Bellman equation:

$$V(\mathbf{X}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_s, \mathbf{Y}_s) + \beta V(\mathbf{X}_{t+1}) \right],$$

subject to the budget constraints:

$$\mathbf{X}_{s+1} = G(\mathbf{X}_s, \mathbf{Y}_s).$$

Or, we can combine the equations to write the problem with a single equation:

$$V(\mathbf{X}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_t, \mathbf{Y}_t) + \beta V(G(\mathbf{X}_t, \mathbf{Y}_t)) \right]. \tag{5}$$

The solution to this problem gives the values of the control variables as a function of the time t state variables:

$$\mathbf{Y}_t = H(\mathbf{X}_t),$$

which we call a policy function, since it describes how the control variables behave as a function of the current state variables. Equation (5) is really a functional equation, since it must hold for every value of $\mathbf{X}_t$ within the permitted domain. Since the policy function optimises the choice of the controls for every permitted value of $\mathbf{X}_t$, it must fulfil the condition:

$$V(\mathbf{X}_t) = F(\mathbf{X}_t, H(\mathbf{X}_t)) + \beta V(G(\mathbf{X}_t, \mathbf{Y}_t)), \tag{6}$$

where maximisation is no longer required because it is implicit in the policy function, $H(\mathbf{X}_t)$.

To find the policy function, $H(\mathbf{X}_t)$, we find the FOCs for the problem in equation (5) with respect to the control variables. The FOCs are:

$$\mathbf{0} = F_Y(\mathbf{X}_t, \mathbf{Y}_t) + \beta V'(G(\mathbf{X}_t, \mathbf{Y}_t)) G_Y(\mathbf{X}_t, \mathbf{Y}_t), \tag{7}$$

where $F_Y(\mathbf{X}_t, \mathbf{Y}_t)$ is the vector of derivatives of the objective function with respect to the control variables, $V'(G(\mathbf{X}_t, \mathbf{Y}_t))$ is the vector of derivatives of the value function with respect to the time $t+1$ state variables, and $G_Y(\mathbf{X}_t, \mathbf{Y}_t)$ is the vector of derivatives of the budget constraints with respect to the control variables. We have the same problem as before: we need to know the derivatives of the value function to be able to solve for the policy function, and the value function is unknown.

But, we can use envelope theorems from Benveniste and Scheinkman (1979) to find an expression for the derivative of the value function. Take the derivative of the value function (5) with respect to

taking time t state variables, $\mathbf{X}_t$, one gets the Benveniste-Scheinkman envelope theorem:[3]

$$V'(\mathbf{X}_t) = F_X(\mathbf{X}_t, \mathbf{Y}_t) + \beta V'(G(\mathbf{X}_t, \mathbf{Y}_t))G_X(\mathbf{X}_t, \mathbf{Y}_t).$$

If, as can frequently be done, the controls have been chosen so that $G_X(\mathbf{X}_t, \mathbf{Y}_t) = \mathbf{0}$, then it is possible to simplify this expression to:

$$V'(\mathbf{X}_t) = F_X(\mathbf{X}_t, \mathbf{Y}_t).$$

The FOCs of (7) can be written as:

$$\begin{aligned}\mathbf{0} &= F_Y(\mathbf{X}_t, \mathbf{Y}_t) + \beta F_X(\mathbf{X}_{t+1}, \mathbf{Y}_{t+1})G_Y(\mathbf{X}_t, \mathbf{Y}_t) \\ &= F_Y(\mathbf{X}_t, \mathbf{Y}_t) + \beta F_X(G(\mathbf{X}_t, \mathbf{Y}_t), \mathbf{Y}_{t+1})G_Y(\mathbf{X}_t, \mathbf{Y}_t).\end{aligned}$$

If the function $F_X(G(\mathbf{X}_t, \mathbf{Y}_t), \mathbf{Y}_{t+1})$ is independent of $\mathbf{Y}_{t+1}$, then this equation can be solved for the implicit function, $\mathbf{Y}_t = H(\mathbf{X}_t)$, which is the required policy function. One can substitute this policy function into equation (6) and solve for the implicit value function $V(\cdot)$. If $F_X(G(\mathbf{X}_t, \mathbf{Y}_t), \mathbf{Y}_{t+1})$ is not independent of $\mathbf{Y}_{t+1}$, then one can solve for the stationary state as we did before, using the condition that $\mathbf{Y}_t = \mathbf{Y}_{t+1} = \bar{\mathbf{Y}}$.

If it is not the case that $G_X(\mathbf{X}_t, \mathbf{Y}_t) = \mathbf{0}$, then an alternative solution method is to find an approximation to the value function numerically. Consider some initial guess for the value function, $V_0(\mathbf{X}_t)$. it doesn't matter very much what this initial guess is, and a convenient one is to assume that it has a constant value of zero. One can then calculate an updated value function, $V_1(\mathbf{X}_t)$, using the formula:

$$V_1(\mathbf{X}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_t, \mathbf{Y}_t) + \beta V_0(G(\mathbf{X}_t, \mathbf{Y}_t))\right],$$

and doing the maximisation numerically over a sufficiently dense set of values from the domain of $\mathbf{X}_t$. This maximisation defines, approximately, the function $V_1(\mathbf{X}_t)$. Using this new function, one can

[3] Suppose we have the following problem:

$$V(x_t) = \max \left[F(x_t, y_t) + \beta V(x_{t+1})\right],$$

subject to:

$$x_{t+1} = G(x_t, y_t).$$

Then the Benveniste-Scheinkman derivative is:

$$V'(x_t) = F_x(x_t, y_t) + \beta V'(G(x_t, y_t)G_x(x_t, y_t).$$

The following four assumptions are sufficient so that the requirements of Benveniste-Scheinkman are met:

1. $x_t \in X$, where X is a convex set with a nonempty interior.
2. $F(\cdot)$ is concave and differentiable.
3. $G(\cdot)$ is concave and differentiable and invertible in y_t.
4. $y_t \in Y$, where Y is a convex set with a nonempty interior.

update again and get a new approximate value function $V_2(\mathbf{X}_t)$ using:

$$V_2(\mathbf{X}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_t, \mathbf{Y}_t) + \beta V_1(G(\mathbf{X}_t, \mathbf{Y}_t)) \right].$$

Repeated application of this process results in a sequence of approximate value functions $\{V_i(\mathbf{X}_t)\}_{i=0}^{\infty}$. Bellman showed that, under a set of conditions that are often met in economic problems, this sequence converges to the value function, $V(\mathbf{X}_t)$.

1.5 Returning to the Robinson Crusoe model

It is useful to write out the example economy showing how each component matches with the general version. For the Robinson Crusoe example we had k_t as the state variable so $\mathbf{X}_t = k_t$, and the capital stock at time $t+1$ was the control variable so $\mathbf{Y}_t = k_{t+1}$.

The objective function is:

$$F(\mathbf{X}_t, \mathbf{Y}_t) = u(f(k_t) + (1-\delta)k_t - k_{t+1}),$$

and the budget constraint is written so that the time $t+1$ state variable is:

$$k_{t+1} = \mathbf{X}_{t+1} = G(\mathbf{X}_t, \mathbf{Y}_t) = \mathbf{Y}_t = k_{t+1}.$$

The FOC for Robinson Crusoe is:

$$\begin{aligned} \mathbf{0} &= F_Y(\mathbf{X}_t, \mathbf{Y}_t) + \beta V'(F(\mathbf{X}_t, \mathbf{Y}_t)) G_Y(\mathbf{X}_t, \mathbf{Y}_t) \\ &= -u'(f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta V'(G(\mathbf{X}_t, \mathbf{Y}_t)) \times 1. \end{aligned} \tag{8}$$

Recall that the Benveniste-Scheinkman envelope theorem gives:

$$V'(\mathbf{X}_t) = F_X(\mathbf{X}_t, \mathbf{Y}_t) + \beta V'(G(\mathbf{X}_t, \mathbf{Y}_t)) G_X(\mathbf{X}_t, \mathbf{Y}_t).$$

For our example, the derivative of the budget constraint with respect to the time t state variable is simply:

$$G_X(\mathbf{X}_t, \mathbf{Y}_t) = \frac{\partial \mathbf{X}_{t+1}}{\partial \mathbf{X}_t} = \mathbf{0},$$

so that the envelope theorem condition can be simplified to:

$$V'(\mathbf{X}_t) = F_X(\mathbf{X}_t, \mathbf{Y}_t) = u'(f(k_t) + (1-\delta)k_t - k_{t+1})(f'(k_t) + (1-\delta)),$$

and the derivative of the value function is defined in terms of functions that we known. We substitute

this into (8) and get:

$$0 = -u'(f(k_t) + (1-\delta) - k_{t+1}) + \beta \left[u'(f(k_{t+1}) + (1-\delta)k_{t+1} - k_{t+2})(f'(k_{t+1}) + (1-\delta))\right].$$

This second-order difference equation can be solved for the steady state, where $k_t = k_{t+1} = k_{t+2} = \bar{k}$, to give:

$$f'(\bar{k}) = \frac{1}{\beta} - (1-\delta) \tag{9}$$

1.5.1 Robinson Crusoe model with a twist

The RC model can be written with different choices for the control variables. The state variable in this version is still time t capital, so $\mathbf{X}_t = k_t$, but one can choose time t consumption to be the time t control variable, $\mathbf{Y}_t = c_t$. So, our objective function is now:

$$F(\mathbf{X}_t, \mathbf{Y}_t) = u(c_t),$$

and the budget constraint is:

$$k_{t+1} = \mathbf{X}_{t+1} = G(\mathbf{X}_t, \mathbf{Y}_t) = f(k_t) + (1-\delta)k_t - c_t.$$

Writing out the model, we have the Bellman equation:

$$V(k_t) = \max_{c_t} \left[u(c_t) + \beta V(f(k_t) + (1-\delta)k_t - c_t)\right],$$

where we have replaced the time $t+1$ state variable, $\mathbf{X}_{t+1} = k_{t+1}$, by the budget constrain the Bellman equation. It should be clear that the problem given above is the exact same economic problem that we solved previously.

This version is somewhat less convenient than the earlier RC model when we try to write out the condition from the envelope theorem. When we take the derivative of the budget constraint with respect to the time t state variable, we get:

$$\frac{\partial G(\mathbf{X}_t, \mathbf{Y}_t)}{\partial \mathbf{X}_t} = f'(k_t) + (1-\delta),$$

and this is generally not equal to zero. If we then write out the envelope theorem condition, we get:

$$\begin{aligned} V'(\mathbf{X}_t) &= F_X(\mathbf{X}_t, \mathbf{Y}_t) + \beta V'(G(\mathbf{X}_t, \mathbf{Y}_t)) G_X(\mathbf{X}_t, \mathbf{Y}_t) \\ &= \beta V'(f(k_t) + (1-\delta)k_t - c_t)(f'(k_t) + (1-\delta)), \end{aligned}$$

and we have the derivative of the value function in terms of the derivative of the value function and

some other terms, which is no improvement.

One of the important tricks of working with the Bellman equation is to write out the objective function and the budget constraints so that one gets a convenient expression of the envelope theorem, that is, so that $G_X(\mathbf{X}_t, \mathbf{Y}_t) = \mathbf{0}$. Doing this usually means putting as much of the model as possible into the objective function and requires keeping the time t state variable out of the budget constraint.

1.6 An approximation of the value function

As mentioned before, we can use numerical methods to find an approximation of the value function (and the policy function) for specific economies. Suppose that we have the following production function:

$$f(k_t) = k_t^\theta,$$

for $0 < \theta < 1$, and the utility function is:

$$u(c_t) = \ln c_t.$$

We can write the Bellman equation as:

$$V(k_t) = \max_{k_{t+1}} \left[\ln(k_t^\theta + (1-\delta)k_t - k_{t+1}) + \beta V(k_{t+1}) \right].$$

To use the recursive method of calculating the approximate $V(\cdot)$, we need to choose values for δ, θ, β, and a functional form for $V_0(\cdot)$. Let $\delta = 0.1$, $\theta = 0.36$, and $\beta = 0.98$. The simplest form to choose for the initial guess of the value function is the constant function, $V_0(k_{t+1}) = 0$, for all values of k_{t+1}. Using the equation for the steady state that we found previously, (9), we find that the steady state values[4] for this model are at $k = 0$ and:

$$\begin{aligned} 0.36 \times \bar{k}^{-0.64} &= \frac{1}{0.98} - (1 - 0.1) \\ \implies \bar{k} &= 5.537. \end{aligned}$$

Attached below is some MATLAB code to carry out the iteration procedure:

```
% Value function iteration
global vlast beta delta theta k0 kt
hold off
hold all

%set intial conditions
vlast=zeros(1,100);
```

[4]Notice the plural.

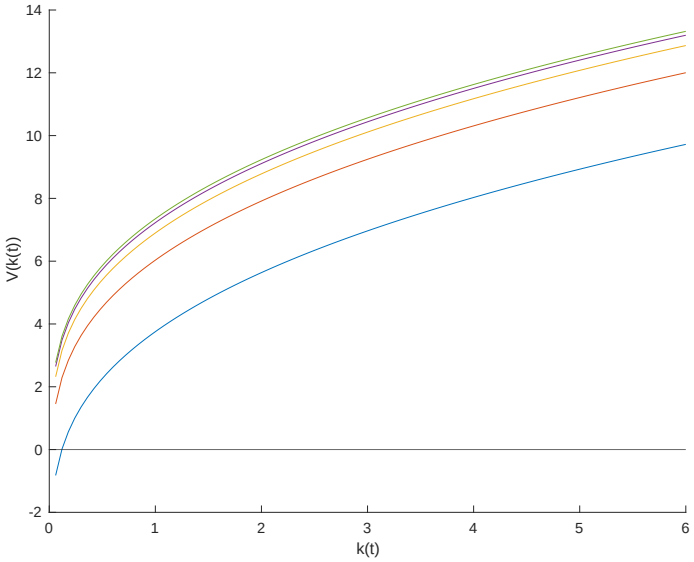
```
k0=0.06:0.06:6
beta=0.98;
delta=0.1;
theta=0.36;
numits=240;

%begin the recursive calculations
for k=1:numits
    for j=1:100
        kt=j*0.06;
        %find the maximum of the value function
        ktp1=fminbnd(@valfun,0.01,6.2)
        v(j)=-valfun(ktp1);
        kt1(j)=ktp1;
    end
    if k/48==round(k/48)
        %plot the steps in finding the value function
        plot(k0,v)
        xlabel('k(t)')
        ylabel('V(k(t))')
        drawnow
    end
    vlast=v;
end
hold off
%plot the final policy function
figure
hold on
plot(k0,kt1)
hline = refline([1 0]);
hline.Color = 'k';
hline.LineStyle = ':';
hline.HandleVisibility = 'off';
xlabel('k(t)')
ylabel('k(t+1)')
```

```
% Subroutine to calculate value function
function val=valfun(k)
global vlast beta delta theta k0 kt
%smooth out the previous value function
g=interp1(k0,vlast,k,'linear');
%calculate consumption with given parameters
kk=kt^theta-k+(1-delta)*kt;
if kk <= 0
    %trick to keep values from going negative
    val=-888-800*abs(kk);
```

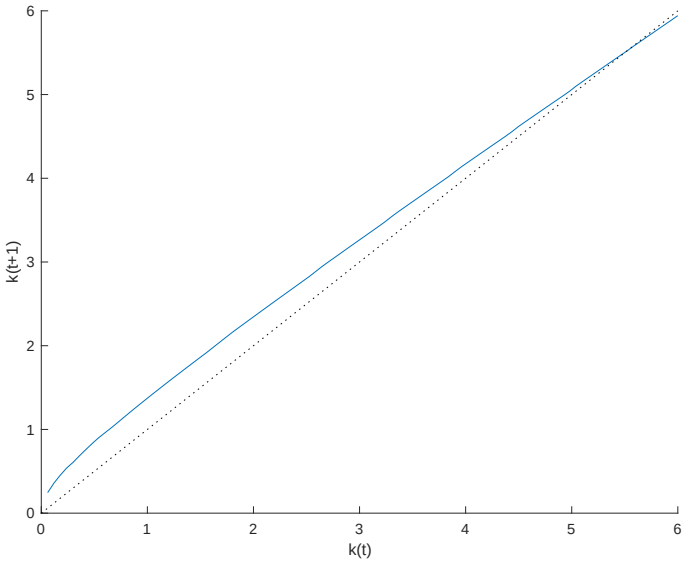
```
else
    %calculate the value of the value function at k
    val=log(kk)+beta*g;
end
%change value to negative since fminbnd" finds minimum
val=-val;
```

Figure 1: Approximating the value function



Source: McCandless (2008)

Figure 2: The policy function after 240 iterations



Source: McCandless (2008)

So what is happening here? The black line in Figure 1 is our initial guess, $V_0(k_t) = 0$, the blue line is the value function after 48 iterations, the orange is the 96th, the yellow is the 144th, the purple is the 192nd, and the green is the 240th. Notice that the steps are gradually getting smaller as the number of iterations increases and the line moves upward.

The policy function for this economy, which finds the optimising value of k_{t+1} for each value of k_t, is generated at the same time as the value functions. The one for our example economy, after 240 iterations, is shown in Figure 2 as the function $k_{t+1} = H(k_t)$. Notice that this function crosses the 45 degree line at the steady state value of 5.537.

We will return to this iterative procedure when we look at stochastic recursive methods.

1.7 An example with variable labour

It might be useful to show that recursive methods can be applied to the RC economy where labour is a variable input. So the utility function is:

$$\sum_{i=0}^{\infty} \beta^i u(c_{t+i}, h_{t+i}),$$

which is maximised subject to the constraints:

$$\begin{aligned} k_{t+1} &= (1-\delta)k_t + i_t, \\ y_t &= f(k_t, h_t) \geq c_t + i_t, \\ h_t &\leq 1. \end{aligned}$$

This problem can be written naturally as the Bellman equation:

$$V(k_t) = \max_{h_t, k_{t+1}} \left[u(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}) + \beta V(k_{t+1}) \right].$$

Here the budget constraint is:

$$k_{t+1} = G(k_{t+1}) = k_{t+1},$$

which implies that the condition $G_X(\mathbf{X}_t, \mathbf{Y}_t) = \mathbf{0}$ is met, so Benveniste-Scheinkman's envelope theorem condition has a simple representation.

There are now two FOCs since there are two controls, h_t and k_{t+1}. These conditions are:

$$\frac{\partial V(k_t)}{\partial h_t} = u_c(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t) f_h(k_t, h_t) + u_h(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t) = 0$$

and

$$\frac{\partial V(k_t)}{\partial k_{t+1}} = -u_c(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t) + \beta V'(k_{t+1}).$$

The envelope condition is:

$$V'(k_t) = u_c(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t)(f_k(k_t, h_t) + (1-\delta)).$$

These conditions result in the equations:

$$\frac{u_h(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t)}{u_c(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t)} = -f_h(k_t, h_t)$$

and

$$\frac{u_c(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t)}{u_c(f(k_{t+1}, h_{t+1}) + (1-\delta)k_{t+1} - k_{t+2}, h_{t+1})} \beta \left[f_k(k_{t+1}, h_{t+1}) + (1-\delta) \right],$$

which are the same conditions we would find if we used variational methods.

This model can also be calculated numerically to find approximations for the value function and for the two policy functions, $k_{t+1} = H^k(k_t)$ and $h_t = H^h(k_t)$. One chooses an initial guess for the value function ($V_0(\cdot) = 0$ os frequently convenient) and repeatedly calculates:

$$V_{j+1}(k_t) = \max_{h_t, k_{t+1}} \left[u(f(k_t, h_t) + (1-\delta)k_t - k_{t+1}, h_t) + \beta V_j(k_{t+1}) \right],$$

for $j = 0, \ldots, j^{\max}$ over a sufficiently dense set of k_t. The sequence of functions, $V_{j+1}(k_t)$, converge to the value function, $V(k_t)$, as $j \to \infty$. As in the case above where labour was fixed, each iteration of this procedure will find the optimising values for k_{t+1} and h_t for each member of the k_t set that was used. This sequence of functions gives approximations for the policy functions that converge to the policy functions $H^k(\cdot)$ and $H^h(\cdot)$ as $j \to j^{\max}$.

Suppose we have: $\delta = 0.1$, $\theta = 0.36$, $\beta = 0.98$, and $A = 0.5$, and that we assume the following utility function:

$$u(c_t, h_t) = \ln c_t + A \ln(1 - h_t),$$

and the production function used is:

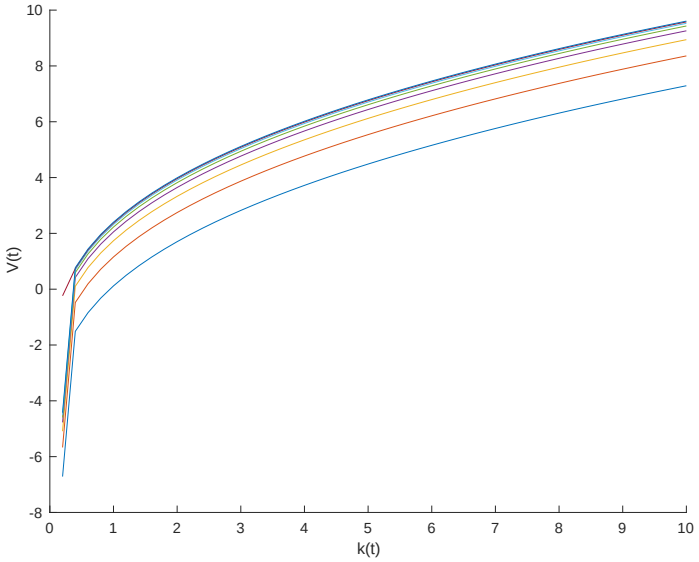
$$f(k_t, h_t) = k_t^{\theta} h_t^{1-\theta}.$$

```
% Value function iteration
global vlast beta delta theta k0 kt A
hold off
hold all
vlast=ones(1,50);
k0=0.2:0.2:10;
kt1=k0;
h=.5*ones(1,50);
xmin=[.21 .01];
xmax=[9.99 .99];
beta=.98;
delta=.1;
theta=.36;
A=.5;
options = optimset('Display','off','LargeScale','off')
numits=240;
for k=1:numits
    for j=1:50
        kt=k0(j);
        z0=[kt,h(j)];
        z=fmincon(@valfun2,z0,[],[],[],[],xmin,xmax,[],options);
        v(j)=-valfun2(z);
        kt1(j)=z(1);
        h(j)=z(2);
    end
    if k/30==round(k/30)
        plot(k0,v)
        xlabel('k(t)')
        ylabel('V(t)')
        drawnow
    end
    vlast=v;
```

```
end
hold off
figure
hold on
plot1=plot(k0,kt1);
plot2=plot(k0,h);
hline = refline([1 0]);
hline.Color = 'k';
hline.LineStyle = ':';
hline.HandleVisibility = 'off';
xlabel('k(t)');
ylabel('k(t+1) and h(t)');
leg1 = "k(t+1)";
leg2 = "h(t)";
legend([plot1,plot2],[leg1,leg2],'Location','northwest');
```

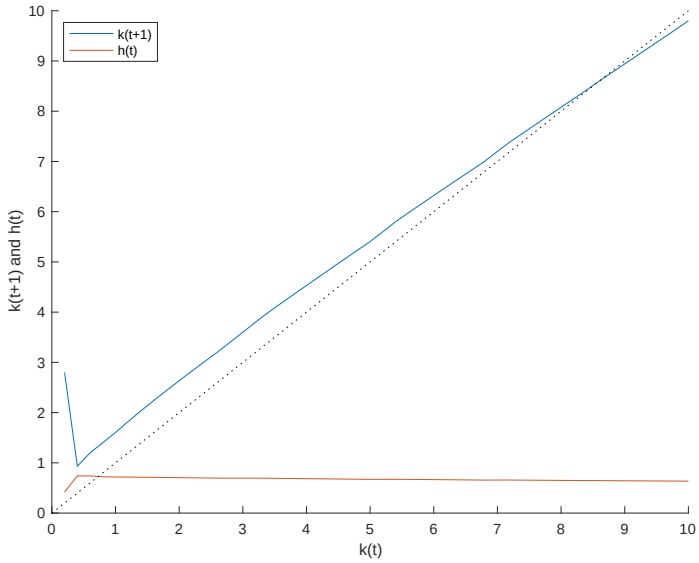
```
% Subroutine to calculate value function
function val=valfun2(x)
global vlast beta delta theta k0 kt A
x;
k=x(1);
h=x(2);
g=interp1(k0,vlast,k,'linear');
kk=1.75*kt^theta*h^(1-theta)-k+(1-delta)*kt;
if kk<=.001
    val=log(.001)+A*log(1-h)+beta*g+(kk-.001);
else
    val=log(kk)+A*log(1-h)+beta*g;
end
val=-val;
```

Figure 3: Approximating the pair of value functions



Source: McCandless (2008)

Figure 4: The two policy functions after 240 iterations



Source: McCandless (2008)

Figure 3 shows the approximate value functions converging upward. The lines shown are $V_m(k_t)$ for iterations number $m = 30, 60, \ldots, 240$. Figure 4 shows the final policy functions for time $t + 1$

capital, $k_{t+1} = H^k(k_t)$, and for time t labour input, $h_t = H^h(k_t)$, along with the 45 degree line so that the value of k_t in the stationary state can be seen. As one might suspect, the amount of labour supplied along an equilibrium path declines as the capital stock increases.

2 Recursive Stochastic Models

We've stuck pretty closely to the material in McCandless (2008)'s introductory material for deterministic dynamic programming. We're going to be moving a bit quicker as we move onto the stochastic case, mostly because this will be the standard across most of the material we will encounter. As such, there are different approaches and notation to cover.

2.1 Probability

A probability space $(\Omega, \mathcal{F}, P)$ is comprised of three elements: 1) Ω, a set that contains all the states of nature that might occur, 2) $\mathcal{F}$, a collection of subsets of Ω, where each subset is called an event, and 3) P, a probability measure over $\mathcal{F}$.

First, consider what this means when Ω is a finite set of possible states of nature. For example, it might contain just two possible values for technology, A_1 and A_2. Then a natural way to define $\mathcal{F}$ is with four elements: the empty set, $\emptyset$, A_1, A_2, and the set $\{A_1, A_2\}$. A probability measure for these four sets is 0 for $\emptyset$, some value $0 \leq p_1 \leq 1$ for A_1, $1 - p_1$ for A_2, and 1 for the set $\{A_1, A_2\}$. This says that either A_1 or A_2 will occur and, for a large enough sample, A_1 will occur with frequency p_1.

For larger finite sets of possible states of nature, the structure is the same, but there are simply more elements to $\mathcal{F}$. If Ω were comprised of three elements, $A_1 = 0.9$, $A_2 = 1.05$, $A_3 = 1.1$, then in addition to the sets given above, $\mathcal{F}$ would include A_3, $\{A_1, A_3\}$, $\{A_2, A_3\}$, and $\{A_1, A_2, A_3\}$. The event $\{A_2, A_3\}$ contains all possible technology levels greater than 1 and occurs with probability $p_2 + p_3$ (when the underlying events are independent and A_1 occurs with probability p_1, A_2 with p_2, and A_3 with p_3, and $p_1 + p_2 + p_3 = 1$.

This may seem obvious and unnecessary vin defining $\mathcal{F}$, the set of subsets of Ω, and then probabilities over this subset. But when the set of possible states of nature is continuous, then the definition is more useful. Consider a growth model where technology, A_t, can take on any value in the set $[0.9, 1.2]$. Suppose that the probability distribution is uniform, so that, in some sense, any value is equally as likely as any other inside the set. In this case, the probability that in some given period t, $A_t = 1.15565$, for example, is zero due to the properties of the uniform distribution. It is in this case that defining subsets of $[0.9, 1.2]$ becomes useful. Imagine that we want to know the probability that technology will have a value in period t between 0.97 and 1.03. Since this is a uniform distribution, this probability can be calculated as $0.06/0.3 = 0.2$.

Although the probability of any one value occurring for A_t is always zero in this example, for any

positive range of values, one can usually find a positive probability. Therefore, by defining probabilities over subsets of the state of nature, the definition encompasses situations with a continuous range of possible states of nature.

2.2 A simple stochastic growth model

Let's return to the case of Robinson Crusoe, but now the production function is:

$$y_t = A^t f(k_t),$$

where we apply the usual assumptions to $f(k_t)$ and A^t can take on two values:

$$A^t = \begin{cases} A_1 & \text{w.p. } p_1, \\ A_2 & \text{w.p. } p_2, \end{cases} \tag{10}$$

and where $A_1 > A_2$.

Capital grows with the following law of motion:

$$k_{t+1} = A^t f(k_t) + (1-\delta)k_t - c_t.$$

At time 0, Robinson Crusoe wants to maximise an expected discounted utility function of the form:

$$\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t u(c_t).$$

Future consumption paths are represented by a kind of tree. Given some initial capital k_0, in period 0 there are two possible technology levels that could occur and two different amounts of production, represented by the ordered pair $\{A_1 f(k_0), A_2 f(k_0)\}$, with probabilities $\{p_1, p_2\}$. Depending on which state occurs in period 0, RC will choose some time 1 capital stocks of $\{k_1^1, k_1^2\}$. In period 1, production will be one of these four possibilities: $\{A_1 f(k_1^1), A_2 f(k_1^1), A_1 f(k_1^2), A_2 f(k_1^2)\}$, with probabilities $\{p_1 p_1, p_1 p_2, p_2 p_1, p_2 p_2\}$.

Suppose that one can write the value of the maximum expected discounted utility given an initial capital stock of k_0, when the time 0 realisation of technology is A_1, as:

$$V(k_0, A_1) = \max_{\{c_t\}_{t=0}^{\infty}} \mathbb{E} \sum_{t=0}^{\infty} \beta^t u(c_t),$$

subject to the budget constraint for $t = 0$:

$$k_1 = A_1 f(k_0) + (1-\delta)k_0 - c_0,$$

and those for $t \geq 1$:

$$k_{t+1} = A^t f(k_t) + (1-\delta)k_t - c_t,$$

and the independent realisations of A^t given by (10). One could write a similar setup by replacing A_1 with A_2.

Expected utility is a function of two state variables: capital inherited and the realised technology level. As shown previously, this expression can be written recursively as:

$$V(k_0, A^0) = \max_{c_0} \left[u(c_0) + \beta \mathbb{E}_0 V(k_1, A^1) \right],$$

subject to:

$$k_1 = A^0 f(k_0) + (1-\delta)k_0 - c_0.$$

There is a subtle change in how the value function is written. It is now written as a function of the time 0 realisation of the technology shock. As this function is written, k_0 and A^0 are the state variables and c_0 is the control variable. The second part of the value function is written with the expectations term because given a choice for c_0 (and through the budget constraint of k_1), it will have a value of $V(k_1, A_1)$ with probability p_1, and a value of $V(k_1, A_2)$ with probably p_2. For any particular choice of $\hat{k}_1$ of the time 1 capital stock, the expectations expression is equal to:

$$\mathbb{E}_0 V(\hat{k}_1, A^1) = p_1 V(\hat{k}_1, A_1) + p_2 V(\hat{k}_1, A_2).$$

For any initial time period t, the problem can be written as:

$$V(k_t, A^t) = \max_{c_t} \left[u(c_t) + \beta \mathbb{E}_t V(k_{t+1}, A^{t+1}) \right],$$

subject to:

$$k_{t+1} = A^t f(k_t) + (1-\delta)k_t - c_t.$$

Or, by making k_{t+1} as the control/choice variable, we can write the problem as:

$$V(k_t, A^t) = \max_{k_{t+1}} \left[u(A^t f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta \mathbb{E}_t V(k_{t+1}, A^{t+1}) \right], \tag{11}$$

and the budget constraint (using the definition in the previous section) is:

$$k_{t+1} = G(\mathbf{X}_t, \mathbf{Y}_t) = k_{t+1}.$$

The solution to a stochastic recursive problem finds a function that gives the values of the control variables that maximises the value function over the domain of the state variables. Since the state variables include both the results of previous choices of control variables and the results of nature's

choices of the value for the stochastic state variables, we call the solution function a plan and write it as:

$$k_{t+1} = H(k_t, A^t).$$

The plan gives the optimising choice of the control variables in every period as a function of the regular state variables and of the states of nature. A plan fulfils the condition that:

$$V(k_t, A^t) = u(A^t f(k_t) + (1-\delta)k_t - H(k_t, A^t)) + \beta \mathbb{E}_t V(H(k_t, A^t), A^{t+1}).$$

2.3 A general version

Using the notation in the previous section, we can write the value function as:

$$V(\mathbf{X}_t, \mathbf{Z}_t) = \max_{\{\mathbf{Y}_s\}_{s=t}^{\infty}} \mathbb{E}_t \sum_{s=t}^{\infty} \beta^{s-t} F(\mathbf{X}_s, \mathbf{Y}_s, \mathbf{Z}_s),$$

subject to the budget constraints given by:

$$\mathbf{X}_{s+1} = G(\mathbf{X}_s, \mathbf{Y}_s, \mathbf{Z}_s),$$

for $s \geq t$, where $\mathbf{X}_t$ is the set of regular state variables, $\mathbf{Z}_t$ is the set of state variables determined by nature (stochastic state variables), and $\mathbf{Y}_t$ are the control variables. As before, $F(\cdot)$ is the objective function and $G(\cdot)$ are the budget constraints. This problem can be written recursively as a Bellman equation of the form:

$$V(\mathbf{X}_t, \mathbf{Z}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_s, \mathbf{Y}_s, \mathbf{Z}_s) + \beta \mathbb{E}_t V(\mathbf{X}_{t+1}, \mathbf{Z}_{t+1}) \right], \tag{12}$$

subject to:

$$\mathbf{X}_{t+1} = G(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t).$$

The solution is a plan of the form:

$$\mathbf{Y}_t = H(\mathbf{X}_t, \mathbf{Z}_t),$$

where

$$V(\mathbf{X}_t, \mathbf{Z}_t) = F(\mathbf{X}_t, H(\mathbf{X}_t, \mathbf{Z}_t), \mathbf{Z}_t) + \beta \mathbb{E}_t V(G(\mathbf{X}_t, H(\mathbf{X}_t, \mathbf{Z}_t), \mathbf{Z}_t), \mathbf{Z}_{t+1}),$$

holds for all values of the state variables (including the stochastic state variables).

The FOCs for the problem in equation (12), and its budget constraints are:

$$\mathbf{0} = F_Y(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) + \beta \mathbb{E}_t \left[V_X G(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t), \mathbf{Z}_{t+1}) G_X(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) \right].$$

When one is able to choose the controls so that $G_X(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) = \mathbf{0}$, the above equation is:

$$V_X(\mathbf{X}_t, \mathbf{Z}_t) = F_X(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t),$$

and the FOCs give the consumption Euler equation (in stochastic form):

$$\mathbf{0} = F_Y(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) + \beta \mathbb{E}_t \left[F_X(G(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t), \mathbf{Y}_{t+1}, \mathbf{Z}_{t+1}) G_Y(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) \right].$$

2.3.1 The problem of dimensionality

Up to this point, the discussion of the general version has said nothing about the dimension of the stochastic variable, $\mathbf{Z}_t$. In the example economy previously, the stochastic variable had only two possible realisations in any period. That the stochastic shock could take on so few values makes the exposition simple. In theory, there is no necessity that the dimension be small, and it is quite possible to describe a model in which the realisation of the stochastic variable comes from a continuous distribution. In practice, the dimension of the state space and the variables in it do matter.

Logically, it should be possible to follow the same steps as the deterministic case and begin with an initial guess for the value function, $V_0(\mathbf{X}_t, \mathbf{Z}_t)$, and iterate on the equation:

$$V_{j+1}(\mathbf{X}_t, \mathbf{Z}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) + \beta \mathbb{E}_t V_j(G(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t), \mathbf{Z}_{t+1}) \right],$$

to find approximations of the value function and the policy functions that converge on the actual value function and plans. Indeed it is possible if the dimensions of $\mathbf{X}_t$ and $\mathbf{Z}_t$ are not too large. The reason for this is that the dimensions of the calculations for the stochastic case are the dimensions of $\mathbf{X}_t$ multiplied by the dimensions of $\mathbf{Z}_t$. If one is using numerical techniques to calculate iterations of the value function, the number of points to be found in each iteration can be come quite burdensome.

2.4 The value function for the simple economy

Let's write equation (11) as a pair of Bellman equations, one for each of the two possible time t realisations of A^t, as:

$$V(k_t, A_1) = \max_{k_{t+1}} \left\{ u(A_1 f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta \left[p_1 V(k_{t+1}, A_1) + p_2 V(k_{t+1}, A_2) \right] \right\},$$

and

$$V(k_t, A_2) = \max_{k_{t+1}} \left\{ u(A_2 f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta \left[p_1 V(k_{t+1}, A_1) + p_2 V(k_{t+1}, A_2) \right] \right\},$$

where we've written out the expected values in terms of their respective probabilistic outcomes.

The iteration process requires choosing starting functions for both $V_0(k_t, A_1)$ and $V_0(k_t, A_2)$. Given these initial functions, the functions from the first iteration, $V_1(k_t, A_1)$ and $V_1(k_t, A_2)$, are found by simultaneously calculating:

$$V_1(k_t, A_1) = \max_{k_{t+1}} \{u(A_1 f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta [p_1 V_0(k_{t+1}, A_1) + p_2 V_0(k_{t+1}, A_2)]\},$$

and

$$V_1(k_t, A_2) = \max_{k_{t+1}} \left\{u(A_1 f(k_{t)} + (1-\delta)k_t - k_{t+1}) + \beta [p_1 V_0(k_{t+1}, A_1) + p_2 V_0(k_{t+1}, A_2)]\right\},$$

over the discrete subset of values of k_t. To find the results of the next iterations, $V_2(k_t, A_1)$ and $V_2(k_t, A_2)$, we calculate:

$$V_2(k_t, A_1) = \max_{k_{t+1}} \{u(A_1 f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta [p_1 V_1(k_{t+1}, A_1) + p_2 V_1(k_{t+1}, A_2)]\},$$

and

$$V_2(k_t, A_2) = \max_{k_{t+1}} \{u(A_2 f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta [p_1 V_1(k_{t+1}, A_1) + p_2 V_1(k_{t+1}, A_2)]\}.$$

Repeated iterations result in a sequence of pairs of functions $\{V_j(k_t, A_1), V_j(k_t, A_2)\}_{j=0}^{\infty}$ that converge to the desired pair of value functions, $\{V(k_t, A_1), V(k_t, A_2)\}$.

2.4.1 Calculating the value functions

Use our parameters from before: $\delta = 0.1$, $\beta = 0.98$, $\theta = 0.36$, the production function is $f(k_t) = k_t^{\theta}$, and the utility function is $u(c_t) = \ln c_t$. Let $A_1 = 1.75$ with $p_1 = 0.8$ and $A_2 = 0.75$ with $p_2 = 0.2$. We choose initial guesses as $V_0(k_t, A_1) = 20$ and $V_0(k_t, A_2) = 20$. The first iteration round results in calculations for $V_1(k_t, A^t)$ of:

$$V_1(k_t, A_1 = 1.75) = \max_{k_{t+1}} \ \ln(1.75 k_t^{0.36} + 0.9 k_t - k_{t+1}) + 0.98 \times 20,$$

and

$$V_1(k_t, A_2 = 0.75) = \max_{k_{t+1}} \ \ln(0.75 k_t^{0.36} + 0.9 k_t - k_{t+1}) + 0.98 \times 20.$$

The second round $V_2(k_t, A^t)$ functions are found maximising:

$$V_2(k_t, 1.75) = \max_{k_{t+1}} \left\{\ln(1.75 k_t^{0.36} + 0.9 k_t - k_{t+1}) + 0.98 [0.8 V_1(k_{t+1}, 1.75) + 0.2 V_1(k_{t+1}, 0.75)]\right\},$$

and

$$V_2(k_t, 0.75) = \max_{k_{t+1}} \left\{ \ln(0.75k_t^{0.36} + 0.9k_t - k_{t+1}) + 0.98 \left[0.8V_1(k_{t+1}, 1.75) + 0.2V_2(k_{t+1}, 0.75) \right] \right\}.$$

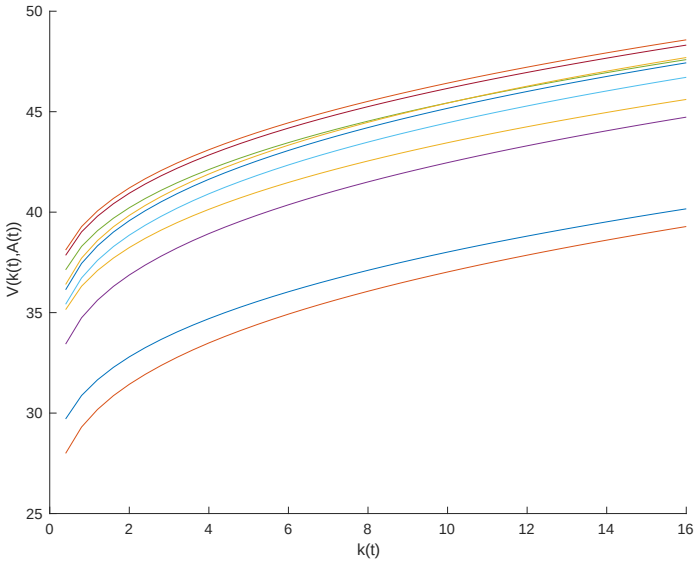
Continued iterations result in the value functions shown in Figure 5. In this figure, the two curves are shown every 50 iterations (the last is at iteration 250). The pair of policy functions that we have after 250 iterations is shown in Figure 6, where the lower one is for $A^t = 0.75$.

```
% Value function iteration
global vlast1 vlast2 beta delta theta k0 kt At p1 p2
hold off
hold all
vlast1=20*ones(1,40);
vlast2=vlast1;
k0=0.4:0.4:16;
kt11=k0;
kt12=k0;
beta=.98;
delta=.1;
theta=.36;
A1=1.75;
p1=.8;
p2=1-p1;
A2=.75;
numits=250;
for k=1:numits
    for j=1:40
        kt=k0(j);
        At=A1;
        z=fminbnd(@valfunsto,.41,15.99);
        v1(j)=-valfunsto(z);
        kt11(j)=z;
        At=A2;
        z=fminbnd(@valfunsto,.41,15.99);
        v2(j)=-valfunsto(z);
        kt12(j)=z;
    end
    if k/50==round(k/50)
        plot(k0,v1,k0,v2)
        xlabel('k(t)')
        ylabel('V(k(t),A(t))')
        drawnow
    end
    vlast1=v1;
    vlast2=v2;
end
```

```
hold off
figure;
hold on
plot1=plot(k0,kt11);
plot2=plot(k0,kt12);
hline = refline([1 0]);
hline.Color = 'k';
hline.LineStyle = ':';
hline.HandleVisibility = 'off'
xlabel('k(t)');
ylabel('k(t+1)');
leg1 = "H(k(t),1.75)";
leg2 = "H(k(t),0.75)";
legend([plot1,plot2],[leg1,leg2],'Location','northwest');
```

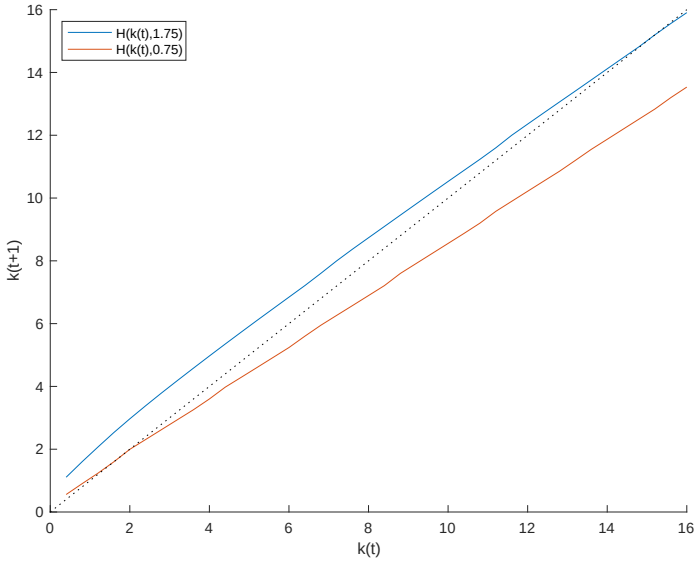
```
% Subroutine to calculate value function
function val=valfunsto(x)
global vlast1 vlast2 beta delta theta k0 kt At p1 p2
k=x;
g1=interp1(k0,vlast1,k,'linear');
g2=interp1(k0,vlast2,k,'linear');
kk=At*kt^theta-k+(1-delta)*kt;
if kk<=.001
    val=log(.001)+beta*(p1*g1+p2*g2)+200*(kk-.001);
else
    val=log(kk)+beta*(p1*g1+p2*g2);
end
val=-val;
```

Figure 5: Approximating the pair of value functions



Source: McCandless (2008)

Figure 6: The two policy functions after 240 iterations



Source: McCandless (2008)

2.5 Markov chains

We can use Markov chains for a richer random process. In a Markov chain, the probabilities for the realisations of the states of nature in period t are a function of the realisation that occurred in period $t-1$ and only in period $t-1$. To use recursive methods, we want a Markov chain that is time invariant – the probabilities depend on the previous period realisation and not the actual period that we're currently in.

There are three elements to a Markov chain. The first is the set of realisations for the state of nature; in the example we have been using, it is the set of values that our A^t variable can take on. This set has a fixed, finite dimension, n, and $\{A^t\} = \{A_1, A_2, \ldots, A_n\}$. The dimension n and the values A_i are the same in every period. In the example above, $n = 2$ and $\{A_t\} = \{A_1 = 1.75, A_2 = 0.75\}$.

The second element of a Markov chain is a matrix of transition probabilities, P, where element $p_{i,j}$ is the probability that state j will occur when the state of nature in the previous period was state i. For example, we could have the following transition matrix:

$$P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix} = \begin{bmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{bmatrix},$$

for A^t. These values are chosen so that the shock exhibits some kind of persistence.

The third element of a Markov chain is the initial state of nature, from period $t-1$, that determines the row of the P matrix where we begin (or the probability distribution of the initial period's stochastic variable if this is different from a row of P). With the initial distribution given, the probability of any future outcome can be calculated.

The probabilities given by the matrix P are condition probabilities in the sense that once the economy is in the state of nature A_1, the top row of the P matrix describes the probabilities for the state of nature in the next period. Once the economy is in the state of nature A_2, the second of the P matrix describes the probabilities of ending up in the state of nature A_1 or A_2 in the next period. These are conditional probabilities since they tell how the economy will proceed once it is in a particular state. We may also be interested in the unconditional probabilities of the occurrence of the states of nature in this economy. These are the probabilities that the economy will be in state of nature A_1 or A_2 when we know nothing about the previous period. If we observe the economy long enough, the unconditional probabilities will tell us how often we observe state of nature A_1 and A_2.

Unique invariant unconditional probability distributions exist if every element of the P matrix is positive ($p_{ij} > 0$). Suppose that p_0 is the initial probability distribution (the one for period 1). The unconditional probability distribution for period 2 is p_0P, given the initial distribution. Multiplying p_0 by the transition matrix gives the probability distribution for period 2. In period 3, the distribution is $p_0P^2 = p_0PP$. In any period $n+1$, the probability distribution for the states of nature is p_0P^n. The claim is that as $n \to \infty$, $p_0P^n \to P^\infty$ independently of p_0.

Using the 2×2 matrix P from above, the first elements of this sequence are:

$$P = \begin{bmatrix} 0.9 & 0.1 \\ 0.4 & 0.6 \end{bmatrix},$$
$$P^2 = \begin{bmatrix} 0.85 & 0.15 \\ 0.60 & 0.40 \end{bmatrix},$$
$$P^3 = \begin{bmatrix} 0.825 & 0.175 \\ 0.70 & 0.30 \end{bmatrix},$$

and in the limit:

$$P^\infty = \begin{bmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \end{bmatrix}.$$

Notice a special characteristic of the P^∞ matrix: all the rows of the matrix are the same. This is because each row gives the unique invariant unconditional probability distribution. Now, let's suppose we have the following initial probability distribution:

$$p_0 = \begin{bmatrix} 0.36 & 0.64 \end{bmatrix}.$$

If we multiply this vector by the matrix P^∞ we get:

$$\begin{aligned} p_0 P^\infty &= \begin{bmatrix} 0.36 & 0.64 \end{bmatrix} \begin{bmatrix} 0.8 & 0.2 \\ 0.8 & 0.2 \end{bmatrix} \\ &= \begin{bmatrix} 0.36 \times 0.8 + 0.64 \times 0.8 & 0.36 \times 0.2 + 0.64 + 0.2 \end{bmatrix} \\ &= \begin{bmatrix} 0.8 & 0.2 \end{bmatrix}. \end{aligned}$$

No matter your initial probability distribution, with a Markov chain you can find the unique invariant unconditional probability distribution for a long enough time path.

We can write the value function for an economy with a Markov chain as:

$$V_{j+1}(\mathbf{X}_t, \mathbf{Z}_t) = \max_{\mathbf{Y}_t} \left[F(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t) + \beta \mathbb{E}_t V_j(G(\mathbf{X}_t, \mathbf{Y}_t, \mathbf{Z}_t), \mathbf{Z}_{t+1}) | \mathbf{Z}_t \right].$$

For the growth economy, modified by using the Markov chain, the value functions are:

$$V(k_t, A_1) = \max_{k_{t+1}} \left\{ u(A_1 f(k_t) + (1-\delta)k_t - k_{t+1}) + \beta \left[p_{11} V(k_{t+1}, A_1) + p_{12} V(k_{t+1}, A_2) \right] \right\},$$

and

$$V(k_t, A_2) = \max_{k_{t+1}} \{u(A_2 f(k_t) + (1-\delta)k_t - k_{t+1})\beta \left[p_{21} V(k_{t+1}, A_1) + p_{22} V(k_{t+1}, A_2)\right]\}.$$

As before, for specific economies, the value function is found by iterating, given some initial choice of the function, $V_0(k_t, A^t)$. For the example economy we have been using, but with the matrix P used in place of the constant probabilities, the $j+1$-th iteration of the value function is:

$$V_{j+1}(k_t, 1.75) = \max_{k_{t+1}} \left\{\ln(1.75k_t^{0.36} + 0.9k_t - k_{t+1}) + \beta \left[0.9V_j(k_{t+1}, 1.75) + 0.1V_j(k_{t+1}, 0.75)\right]\right\}$$

and

$$V_{j+1}(k_t, 0.75) = \max_{k_{t+1}} \left\{\ln(0.75k_t^{0.36} + 0.9k_t - k_{t+1}) + \beta \left[0.4V_j(k_{t+1}, 1.75) + 0.6V_j(k_{t+1}, 0.75)\right]\right\}.$$

As before, these value functions converge to a $V(k_t, A^t)$ and contingent plans. The main catch with Markov chains is that while they help with improving shock persistence in a simulated time path for a state variable, they do not explain why the persistence exists. The time series that result from these kinds of models can be made to display persistence, but this persistence is not explained in economic terms (see Ljungqvist and Sargent (2018) for further discussion on Markov chains).

3 Projection Methods

We now switch away from value function iteration to projection methods, introducing a slight change in notation. These methods compute directly the policy function without calculating the value functions. They use the first-order conditions (Euler equation) to back out the policy rules. Let's consider a simple example and suppose that x is some exogenous variable and that the following equation implicitly defines y:

$$g(x, y) = 0, \ \forall x \in X.$$

Let the solution be defined by the policy rule:

$$y = h(x),$$

which satisfies the following error function condition:

$$R(x, h) \equiv g(x, h(x)) = 0, \ \forall x \in X.$$

As we know, finding the policy rule, h, is a big problem outside of special cases – there are an

infinite number of unknowns (i.e., one value of h for each possible x) in an infinite number of equations (i.e., one equation for each possible x). There are broadly two approaches to this problem: projection and perturbation, each with their own different sub-versions and variations.

The main idea of projection is to find a parametric function, $\hat{h}(x;\psi)$, where ψ is a vector of parameters chosen so that it imitates the property of the exact solution, i.e., $R(x;h) = 0$, $\forall x \in X$, as well as possible. So we choose values for ψ so that

$$\hat{R}(x;\psi) = g(x, \hat{h}(x;\psi)) \approx 0, \ \forall x \in X.$$

The method is defined by the meaning of 'close to zero', by the parametric function, $\hat{h}(x;\psi)$, that is used, and spectral functions. Spectral functions are functions, such as $\hat{h}(x;\psi)$, in which each parameter in ψ influences $\hat{h}(x;\psi)$, $\forall x \in X$. For example:

$$\hat{h}(x;\psi) = \sum_{i=0}^{n} \psi_i G_i(x), \ \psi = \begin{bmatrix} \psi_1 \\ \vdots \\ \psi_n \end{bmatrix},$$

where $G_i(x) = T_i(\varphi(x))$, and $T_i(z) : [-1,1] \to [-1,1]$ is an i-th order Chebyshev polynomial, and $\varphi : X \to [-1,1]$. What does this all mean? Before proceeding, it's worth talking about approximating functions and numerical integration.

3.1 Function approximation

If you have a finite set of data points and wish to determine the underlying functional form, you will need to resort of function approximation. For example, suppose one knows the next period's value of the interest rate, R_{t+1}, is some function of the current interest rate, but one doesn't know what the functional form is. That is, one has the empirical relationship:

$$R_{t+1} = f(R_t) + \epsilon_{t+1},$$

and the question is then whether with enough data one can discover the functional form of f.

The need for approximating functional forms also arises if one could in principle calculate the function value for any given set of arguments but that it is very expensive to do so. For example, the function value may be the outcome of many complex calculations and it may take a lot of computing time to calculate one function value. With an approximating functional form, one could obtain (approximate) function values much quicker. Again, the goal would be to come up with an approximating functional form using a finite set of data points. There is a big difference with the problem that the econometrician faces, however, because the econometrician cannot choose their data points. We will

see that the freedom to choose the location of the arguments makes it much easier to come up with accurate approximations. Finally, the theory of function approximation is very useful if one is trying to solve for a function that is implicitly defined by a system of functional equations.

3.1.1 Polynomial approximations

We will mostly look at polynomial approximations, i.e.:

$$y_t = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n,$$

where x is a scalar. Later, we will discuss functions with multiple arguments. We will see that there are actually many different types of polynomials by choosing different basis functions, $T_i(x)$. That is, a more general way to write polynomials is:

$$y_t = a_0 + a_1 T_1(x) + a_2 T_2(x) + \cdots + a_n T_n(x).$$

For example, $T_i(x)$ could be $(\ln x))^i$. Moreover, one can take transformations. For example, if one knows that the function is always positive, one could use this information by letting

$$y_t = \exp\left\{a_0 + a_1 T_1(x) + a_2 T_2(x) + \cdots + a_n T_n(x)\right\}.$$

How good are polynomial approximations? Weierstrass' theorem[5] tells us that a continuous real-valued function defined on a bounded interval on the real line can be approximated arbitrarily well using the sup norm if the order of the polynomial goes to infinite. Even functions that have a discontinuity can be approximated arbitrarily well if one uses another norm then the sup norm.

So how to find these approximations? We will discuss four different ways to come up with a polynomial approximation. the procedures differ in whether they use only local information (Taylor expansion) or whether they use information about derivatives or not:

Taylor expansion: If one has the function value and n derivatives at one point, x_0, then one can calculate a polynomial approximation using the Taylor expansion:

$$f(x) \approx f(x_0) + (x - x_0) \left.\frac{\partial f(x)}{\partial x}\right|_{x=x_0} + \cdots + \frac{(x - x_0)^n}{n!} \left.\frac{\partial^n f(x)}{\partial x^n}\right|_{x=x_0}.$$

Note that the RHS is indeed a polynomial.

Projection: More common though is that one has $n+1$ function values, $f_0, f_1, \ldots, f_n$, at $n+1$ arguments, $x_0, x_1, \ldots, x_n$. There are two procedures to find the coefficients of the polynomial. The

[5] Let f be a continuous real-valued function defined on the real interval $[a, b]$. For every $\epsilon > 0$, $\exists$ a polynomial p such that for all $x \in [a, b]$ we have $|f(x) - p(x)| < \epsilon$. In other words: there exists a polynomial that approximates any continuous function arbitrarily well.

first is to run a regression. If you do it right you get an R^2 equal to 1.

Lagrange interpolation: The approximating polynomial is also given by:

$$f(x) \approx f_0 L_0(x) + \cdots + f_n L_n(x), \tag{13}$$

where:

$$L_i(x) = \frac{(x - x_0) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_0)}{(x_i - x_0) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)}.$$

$L_i(x)$ is a polynomial so that the RHS of (13) is a polynomial too. So we only have to show that this polynomial gives an exact fit at the $n+1$ points. This is easy if one realises that:

$$L_i(x) = \begin{cases} 1 & \text{if } x = x_i, \\ 0 & \text{if } x \in \{x_0, \ldots, x_n\} \backslash \{x_i\}. \end{cases}$$

Consequently, one gets:

$$f(x_i) = f_i.$$

It is unlikely that you will actually find it useful to write the polynomial like this, but we will see that this way of writing a polynomial is useful for numerical integration.

Hermite interpolation: The last two procedures only used function values at a set of grid points. Now suppose that in addition to $n+1$ function values one also has numerical values for the derivatives at the nodes, i.e., $f_0^{'}, f_1^{'}, \ldots, f_n^{'}$. With these $2(n+1)$ pieces of information one calculate a $(2n+1)$-th order polynomial.

$$f(x) \approx \sum_{i=0}^{n} f_i H_i(x) + \sum_{i=0}^{n} f_i^{'} \tilde{H}_i(x),$$

where:

$$H_i = (1 - 2L_i^{'}(x_i)(x - x_i))L_i(x)^2,$$
$$\tilde{H}_i = (x - x_i)L_i(x)^2.$$

Note that:

$$H_i(x) = H_i^{'}(x) = \begin{cases} 1 & \text{if } x = x_i, \\ 0 & \text{if } x \in \{x_0, \ldots, x_n\} \backslash \{x_i\}. \end{cases}$$

The approximation gets the function values right at all nodes because the $\tilde{H}_i(x_j)$ terms are all zero and the $H_i(x_j)$ terms are 1 at $x_j = x_i$ and zero otherwise. The approximation gets the derivatives right because the $H_i(x_j)$ are all zero and the $\tilde{H}_i(x_j)$ select with all its zero/one structure the appropriate derivative.

3.1.2 Orthogonal polynomials

From the discussion above, it became clear that finding the coefficients of the polynomial is like a projection on the space spanned by the basis function ,i.e., like a regression. In any introductory econometrics course, one learns about the problem of multicollinearity and the advantage of having uncorrelated explanatory variables. The same is true in function approximation. Moreover, in numerical problems it is important to have good initial conditions. The problem with the basis functions of regular polynomials (i.e., $1, x, x^2$, and etc.) is that these terms are often highly correlated unless one has a lot of variation in the argument. Adding additional polynomial terms could thus very well mean that the coefficients of all polynomial terms change substantially even if the extra terms have little additional explanatory power.

Orthogonal polynomials are such that the basis functions are by construction orthogonal with respect to a certain measure. That is, the basis functions of all orthogonal polynomials satisfy:[6]

$$\int_a^b T_i(x)T_j(x)w(x)dx = 0, \ \forall i, j \ \ni i \neq j,$$

for some weighting function $w(x)$. Popular orthogonal polynomials are Chebyshev polynomials and the reason they are popular will become clear below. Chebyshev polynomials are defined on the interval $[a, b] = [-1, 1]$ and the weighting function is given by:

$$w(x) = \frac{1}{\sqrt{(1-x^2)}}.$$

The basis functions of the Chebyshev polynomials are given by:

$$\begin{aligned} T_0^c(x) &= 1, \\ T_1^c(x) &= x, \\ &\vdots \\ T_{i+1}^c(x) &= 2xT_i^c(x) - T_{i-1}^c(x), \ i > 1. \end{aligned}$$

Note that if one builds a polynomial with Chebyshev basis functions, i.e.:

$$f(x) = \sum_{j=0}^{n} a_j T_j^c(x),$$

[6] A note on terminology: $\ni$ means "contains as an element".

then one also has a standard polynomial of the form:

$$b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n,$$

where the b_j's are functions of the a_j's. The same is true for other orthogonal polynomials. So the reason why we use orthogonal polynomials is that it is easier to find the coefficients, for example, because if one adds higher order terms, good initial conditions are the coefficients one found with the lower-order approximation. Chebyshev polynomials are defined on a particular interval but note that a continuous functions on a compact interval can always be transformed so that is defined in this range.

3.1.3 Chebyshev nodes

We now defined a concept of which the importance will become clear in the remainder of this section. Chebyshev nodes are the x-values at which a basis function is equal to zero. For example:

$$T_2^c(x) = 2x^2 - 1,$$

and the corresponding roots are equal to $-\sqrt{\frac{1}{2}}$ and $\sqrt{\frac{1}{2}}$. Similarly:

$$T_3^c(x) = 4x^3 - 3x,$$

and the roots are equal to $-\sqrt{\frac{3}{4}}$, 0, and $\sqrt{\frac{3}{4}}$. If one wants to construct n Chebyshev nodes, one thus takes the n-th Chebyshev basis function and finds the roots that set it equal to zero.

3.1.4 Uniform convergence

Weierstrass' theorem implies that there are polynomial approximations that converge uniformly towards the true function, because convergence in the sup norm implies uniform convergence. To find this sequence, however, one must be smart in choosing the points that one uses in the approximation. If one uses observed data points one doesn't have this degree of freedom, but in many numerical problems one does. The flexibility to choose the approximation points will turn out to be a great benefit in many numerical problems.

It turns out that by fitting the polynomial at the Chebyshev nodes guarantees uniform convergence. A famous function to document how terrible not having uniform convergence can be is:

$$f(x) = \frac{1}{1 + 25x^2},$$

defined on $[-1, 1]$. As an exercise you should compare the following two strategies to find the coefficients of the approximating polynomial. The first strategy finds the coefficients of the approximating

polynomial using $n+1$ equidistant points and its function values. The second strategy uses Chebyshev nodes. The polynomials that one obtains with equidistant points only converge point wise and as n increases one sees bigger and bigger oscillations.

For a formal discussion one should read Judd (1998). But some intuition of why Chebyshev nodes are so powerful can be obtained by thinking of the formula for standard errors in a standard regression problem. If $\mathbf{X}$ is the matrix with all the observations of the explanatory variables and σ^2 is the error variance, then the standard error is given by $\sigma^2(\mathbf{X}^\top\mathbf{X})^{-1}$. That is, the further apart the $\mathbf{X}$ values the smaller the standard error. Chebyshev nodes are more spread towards the boundaries than equidistant points and, thus, we obtain a more accurate approximation using polynomials fitted at Chebyshev nodes.

3.1.5 Other types of basis functions

Orthogonal polynomials can be written as ordinary polynomials. They differ from ordinary polynomials by having different basis functions, but an n-th order Chebyshev polynomial can be written as n-th order regular polynomial. Nevertheless one has quite a bit of flexibility with polynomial approximations. For example, instead of approximating $f(x)$ one can approximate $f(\exp\tilde{x}) = f(\exp(\ln x))$. Or if one knows that the function value is always positive, one can approximate $\ln f(x)$.

Of course, one could consider alternatives using polynomial basis functions. An alternative is to use neural nets. The idea behind neural nets is very similar to using polynomial approximations but they use different basis functions to build up the approximating function. In particular let $\mathbf{X}$ be a vector with function arguments and let $f : \mathbb{R}^n \to \mathbb{R}$. Then the leural net approximations is given by:

$$f(\mathbf{X}) \approx \sum_{j=1}^{J} \gamma_j g(\mathbf{w}_j^\top \mathbf{X} + b_j),$$

where $\mathbf{w}_j \in \mathbb{R}^n$, γ_j, $b_j \in \mathbb{R}$, and $g : \mathbb{R} \to \mathbb{R}$ is a scalar squashing function – a function with function values in the unit interval. Neural net approximations are not very popular in macroeconomics. The reason is that neural net approximations need quite a few parameters (layers) to approximate low-order polynomials and many series in economics are well approximated with polynomials. Neural nets have been more successful in explain time series with more chaotic behaviour.

3.2 Splines

The approximation procedures discussed above all had in common that for each possible argument at which one would like to evaluate the function, there is one identical polynomial. The idea about splines is to split up the domain into different regions and to use a different polynomial for each region. This would be a good strategy if, the function can only be approximated well with a polynomial of a very high order over the entire domain, but can be approximated well with a sequence of low-order

polynomials for different parts of the domain. The inputs to construct a spline are again $n+1$ function values at $n+1$ nodes.

Splines can still be expressed as a linear combination of basis functions which is the same for each possible argument, but this is not a polynomial. The basis functions are zero over most of the domain. Thus splines take a much more local approach and a change in a function value far away from x_i is much less likely to affect the approximation for $f(x_i)$ when using splines than when using polynomial approximations.

Piecewise linear: The easiest spline to consider a piecewise linear interpolation. That is for $x \in [x_i, x_{i+1}]$:

$$f(x) \approx \left(1 - \frac{x - x_i}{x_{i+1} - x_i}\right) f_i + \left(\frac{x - x_i}{x_{i+1} - x_i}\right) f_{i+1}.$$

n-th order spline: Piecewise linear splines are in general not differentiable at the nodes and this could be a disadvantage. But it is easy to deal with this by fitting a low-order polynomial on each segment and choose the coefficients such that it fits the function values at the nodes and the function is smooth at the nodes. Consider what needs to be done to implement a cubic spline. A cubic spline uses:

$$f(x) \approx a_i + b_i x + c_i x^2 + d_i x^3, \ x \in [x_{i-1}, x_i].$$

Since we have n segments, we have n separate cubic approximations and thus $4n$ coefficients. What are the conditions that we have to pin down these coefficients?

We have $2 + 2(n-1)$ conditions to ensure that the function values correspond to given function values at the nodes. For the two endpoints, x_0 and x_{n+1}, we only have one cubic that has to fit it correctly. But for the intermediate nodes we need that the cubic approximations of both adjacent segments give the correct answer. For example, we need that:

$$\begin{aligned} f_1 &= a_1 + b_1 x_1 + c_1 x_1^2 + d_1 x_1^3, \text{ and} \\ f_1 &= a_2 + b_2 x_1 + x_2 x_1^2 + d_2 x_1^3. \end{aligned}$$

To ensure differentiability at the intermediate nodes we need:

$$b_i + 2c_i x_i + 3d_i x_i^2 = b_{i+1} + 2c_{i+1} x_i + 3d_{i+1} x_i^2, \ x_i \in \{x_1, \ldots, x_{n-1}\},$$

which gives us $n-1$ conditions.

With a cubic spline one can also ensure that second derivatives are equal:

$$2c_i + 6d_i x_i = 2c_{i+1} + 6d_{i+1} x_i, \ x_i \in \{x_1, \ldots, x_{n-1}\}.$$

We now have $2 + 4(n-1) = 4n - 2$ conditions to find $4n$ unknowns. So we need two additional

conditions. For example, one can set the derivatives at the end points equal to zero. There are several algorithms to find the coefficients of spline approximations.

Coefficients of the spline: What are the coefficients of the spline? There are two answers. The first answer is to say that the coefficients that determine the functional form are the $n+1$ coefficients combinations for x_i, f_i. The other answer is to say that they are all the coefficients of the separate polynomials. Both answers are, of course, correct. But for latter applications it is more convenient to think of the coefficients of the splines as the (x_i, f_i) pairs and when the nodes don't change, then the coefficients are just the $n+1$ function values, i.e., the f_i's. Now note that these values may not directly reveal the function value at an arbitrary x, but given that we do use a particular spline, these function values pin down the spline and thus the approximating function value at x. More importantly, if finding the approximating function is part of a bigger numerical project, then the goal is to find the $n+1$ function values at the nodes. These completely pin down the solution.

3.3 Shape-preserving approximations

Polynomial approximations oscillate around the true function value. Moreover, these oscillations could be such that the approximating function does not inherit important properties of the approximating function, such as monotonicity or concavity. Can you approximate a function and preserve such properties? That is, suppose that the $n+1$ function values satisfy properties such as being positive, monotonic, and concave. Can you come up with an approximation that also has these properties?

If one uses one polynomial for the complete domain then this is more likely to happen if one uses a low-order polynomial. But since the fit in terms of distance may be worse for the low-order polynomial one could face a trade-off between accuracy and desired shape.

Actually, there is one approximation we discussed that automatically preserves monotonicity and concavity and that is the piece-wise linear approximation. That is, if the function $f(x)$ is monotonic and concave, then the $n+1$ function values will of course inherit these properties and so will the interpolated values. Schumacher's algorithm finds second order splines that preserve (if present) monotonicity and concavity/convexity.

3.4 Multivariate polynomials

3.4.1 Polynomial approximations

Extending polynomial approximations to multivariate problems is very easy. Just like one easily formulates multivariate polynomials using standard basis functions, one can also formulate multivariate polynomials using other basis functions. For example, consider a function that has x and y as argu-

ments. Then the n-th order complete polynomial is given buy:

$$\sum_{i+j\leq n} T_i(x)T_j(y).$$

The n-th order tensor product polynomial is given by:

$$\sum_{i\leq n, j\leq n} T_i(x)T_j(y).$$

3.4.2 Splines

Linear interpolation is easy for multivariate problems. Here we give the formulas for the interpolation of a function that depends on x and y and one has the four function values, f_{xy}, for the (x, y) combinations equal to (a, c), (b, c), (b, d), and (a, d). In this rectangle the interpolated value is given by:

$$f(x,y) = \begin{array}{c} \frac{1}{4}\left(2-2\frac{x-a}{b-a}\right)\left(2-2\frac{y-c}{d-c}\right) f_{ac} \\ +\frac{1}{4}\left(2\frac{x-a}{b-a}\right)\left(2-2\frac{y-c}{d-c}\right) f_{bc} \\ +\frac{1}{4}\left(2\frac{x-a}{b-a}\right)\left(2\frac{y-c}{d-c}\right) f_{bd} \\ +\frac{1}{4}\left(2-2\frac{x-a}{b-a}\right)\left(2\frac{y-c}{d-c}\right) f_{ad} \end{array}.$$

Since the RHS has the cross product of x and y, this is a first order tensor but not a first order complete polynomial. At the boundaries, i.e., the walls of the box, the function is linear, which means that the function automatically coincides with the interpolated values of the box right next to it.

Extending splines to multivariate problems is tricky. To see why think about what one has to ensure to construct a two-dimensional spline. The equivalent of the segment for the univariate case is now a box with the floor representing the space of the two arguments. Now focus on the fitted values on one of the walls. These function values and the corresponding derivatives have to be equal to those on the wall of the box next to it. So instead of ensuring equality at a finite set of nodes one now has to ensure equality at many points even in the two dimensional case.

3.4.3 What type of approximation to use

Before one uses any approximation method one should ask oneself what one knows about the function. For example, it is possible that there are special cases for which one knows the functional form. Also, sometimes one knows that the functional form is more likely to be simple in the logs than in the original arguments. This would be the case if one expects the elasticity of f with respect to x to be fairly constant (as opposed to $\frac{\partial f}{\partial x}$).

The big question one faces is whether one should use one (possibly high-order) polynomial for the entire domain or several (possibly lower-order) polynomials for separate segments. The latter doesn't

mean one has to use splines with many segments. Using prior knowledge one can also split the domain in a small number of separate regions, for example, a region where a borrowing constraint is present and one where it is not binding. For each of the regions one then fits a separate approximating function. Note that for the case of borrowing constraints one typically wouldn't want the function at the node that connects the two regions to be differentiable. So one really could simply fit two polynomials on the two regions.

3.5 Numerical integration

Numerical evaluation of a definite integral is a frequent problem encountered in economic modelling. If a firm pays a continuous stream of dividends, $d(t)$, and the interest rate is r, then the present value of the dividends equals:

$$\int_0^\infty \exp(-rt)d(t)dt.$$

If the random variable X is distributed $N(0,1)$, then the expectation of $f(X)$ is:

$$(2\pi)^{-\frac{1}{2}} \int_{-\infty}^{\infty} f(x) \exp\left(-\frac{x^2}{2}\right) dx.$$

In Bayesian statistics, if one's prior belief over the parameter space, Θ, has density $f(\theta)$, if the data are X, and if $g(X|\theta)$ is the density of X conditional on θ, then the posterior mean belief is:

$$\frac{\int \theta g(X|\theta) f(\theta) d\theta}{\int g(X|\theta) f(\theta) d\theta}.$$

Not only do integrals arise naturally in formulations of economic and econometric problems, but we will often introduce them as part of our numerical procedures. For example, computing the coefficients of orthogonal polynomial approximations.

3.6 Quadrature techniques

Suppose we want to calculate:

$$I = \int_a^b f(x)dx,$$

where $f(x)$ is a scalar function. This could be a difficult problem, e.g., because the functional form is nasty or because we do not even have a functional form, but only a set of function values.

Quadrature techniques are numerical integration techniques for which the formula of the numerical integral can be written as:

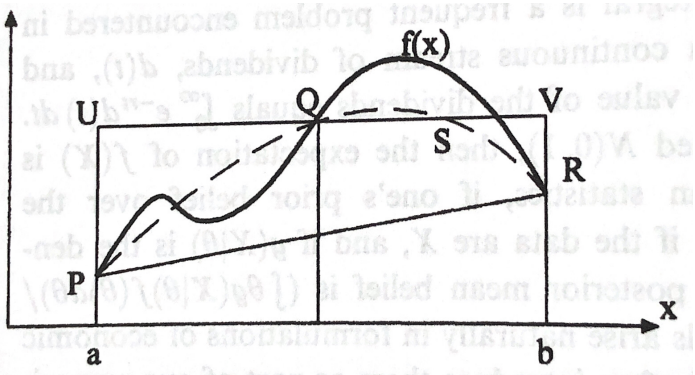
$$I = \int_a^b f(x)dx \approx \sum_{i=1}^{n} \omega_i f_i,$$

where f_i is the function value of f at node x, and ω_i is a weight. We will discuss two types of quadrature techniques. The first is Newton-Cotes. Newton-Cotes is not very careful about choosing the location of the nodes, but is clever about choosing the weights. The second is Gaussian Quadrature. This procedure is clever about choosing the weights as well as the nodes. To implement quadrature methods you can forget the details of the derivation. All you have to remember is how to construct what kind of weights and this is relatively easy.

3.6.1 Newton-Cotes formulas

The Newton-Cotes quadrature formulas use ideas from piecewise-polynomial approximation theory. They evaluate f at a finite number of points, use this information to construct a piecewise-polynomial approximation of f, and then integrate this approximation of f to approximate $\int_D f(x)dx$. This section will present various Newton-Cotes formulas and their error properties.

Figure 7: Newton-Cotes rules



Source: Judd (1998)

Consider the graph in Figure 7. Suppose that f is the solid curve through the points P, Q, and R. The integral $\int_a^b f(x)dx$ is the area under the function f and above the horizontal axis. Three approximations are immediately apparent. The box $aUQVb$ approximates f with a constant function equaling f at Q, which is the midpoint of $[a, b]$. The trapezoid $aPRb$ approximates f with a straight line through points P and R. The area under the broken curve $PQSR$ approximates f with a parabola through P, Q, and R. These approximations are based on one, two, and three evaluations of f, respectively, which are used to compute interpolating polynomials of degree one, two, and three. This approach yields Newton-Cotes quadrature formulas.

Midpoint rule: The simplest quadrature formula is the midpoint rule:

$$\int_a^b f(x)dx = (b-a)f\left(\frac{a+b}{2}\right) + \frac{(b-a)^3}{24}f''(\xi), \tag{14}$$

for some $\xi \in [a, b]$. We will express many integration formulas in the fashion of (14), where the first terms comprise the integration rule and the last term is the error of the integration rule. Hence the midpoint rule is the first term on the RHS of (14), and the second term is its error term. Equation (14) is provided by applying Taylor's theorem and the intermediate value theorem. The midpoint rule is the simplest example of an open rule, which is a rule that does not use the end points.

This approximation is too coarse to be of value generally. Instead, we break the interval $[a, b]$ into smaller intervals, approximate the integral over each of the smaller intervals, and add those approximations. The result is a composite rule. Let $n \geq 1$ be the number of intervals, $h = (b-a)/n$, and $x_j = a + (j - \frac{1}{2})h$, $j = 1, 2, \ldots, n$. Then the composite midpoint rule derives the equation:

$$\int_a^b f(x)dx = h\sum_{j=1}^{n} f(x_j) + \frac{h^2(b-a)}{24} f''(\xi), \tag{15}$$

for some $\xi = [a, b]$. Notice that the error is proportional to h^2; doubling the number of quadrature nodes will have the step size h and reduce the error by about 75 percent. Therefore the composite midpoint rule converges quadratically for $f \in C^2$.[7]

Trapezoid rule: The trapezoid rule is based on the linear approximation of f using only the value of f at the endpoints of $[a, b]$, The trapezoid rule is:

$$\int_a^b f(x)dx = \frac{b-a}{2}\left[f(a) - f(b)\right] - \frac{(b-a)^3}{12} f''(\xi), \tag{16}$$

for some $\xi \in [a, b]$. The trapezoid rule is the simplest example of a closed rule, which is a rule that uses the end points. Let $h = (b-a)/n$, and $x_i = a + ih$, and let f_j denote $f(x_j)$; the composite trapezoid rule is:

$$\int_a^b f(x)dx = \frac{h}{2}\left[f_0 + 2f_1 + \cdots + 2f_{n-1} + f_n\right] - \frac{h^2(b-a)}{12} f''(\xi), \tag{17}$$

for some $\xi \in [a, b]$.

Simpson's rule: Piecewise linear approximation of f in the composite trapezoid rule is unnecessarily coarse if f is smooth. An alternative is to use a piecewise quadratic approximation of f which uses the value of f at a, b, and the midpoint, $\frac{1}{2}(a + b)$. The result is Simpson's rule over the interval $[a, b]$. Simpson's rule is:

$$\int_a^b f(x)dx = \left(\frac{b-a}{6}\right)\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] - \frac{(b-a)^5}{2880} f^{(4)}(\xi), \tag{18}$$

[7] C^0: Curves are continuous
C^1: First derivatives are continuous
C^2: First and second derivatives are continuous
C^n: First through n-th derivatives are continuous

for some $\xi \in [a, b]$.

We next construct the corresponding $(n+1)$-point composite rule over $[a, b]$. Let $n \geq 2$ be an even number of intervals; then $h = (b-a)/n$, $x_j = a + jh$, $j = 0, \ldots, n$, and the composite Simpson's rule is:

$$S_n(f) = \frac{h}{3}\left[f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 4f_{n-1} + f_n\right] - \frac{h^4(b-a)}{180} f^{(4)}(\xi), \tag{19}$$

for some $\xi \in [a, b]$. The composite Simpson's rule essentially takes three consecutive x_j nodes, uses the interpolating quadratic function to approximate f, and integrates the interpolating quadratic to approximate the integral over that interval.

Notice that by using a locally quadratic approximation to f we have an error order h^4, whereas the locally linear approximation yields the trapezoidal rule which has error of order h^2. As with any local approximation of smooth functions, higher order approximations yield asymptotically smaller errors.

3.6.2 Gaussian formulas

Newton-Cotes formulas use a collection of low-order polynomial approximations on small intervals to derive piecewise polynomial approximations to f. Gaussian quadrature instead builds on the orthogonal polynomial approach to functional approximation. All Newton-Cotes rules are of the form:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i), \tag{20}$$

for some quadrature nodes $x_i \in [a, b]$ and quadrature weights ω_i. The key feature of Newton-Cotes formulas is that the x_i points are chosen arbitrarily, usually being the uniformly spaced nodes on $[a, b]$, and the ω_i wights are chosen so that if f is locally a low degree polynomial then the approximation (20) will be correct.

In contrast, Gaussian formulas are constructed by efficient choices of both the nodes and weights. In general, the Gaussian approach is to find points $\{x_i : i = 1, \ldots, n\}$ and weights $\{\omega_i : i = 1, \ldots, n\}$ so as to make the approximation (20) of $\int f$ a "good" one.

In order to accomplish this, we must define what we mean by a "good" quadrature formula. The criterion we use is exact integration for a finite-dimensional collection of functions. More specifically, we choose the weights and nodes so that the approximation is exactly correct when f is a low-order polynomial. The remarkable feature of Gaussian quadrature is that it accomplishes this for spaces of degree $2n-1$ polynomials using only n nodes and n weights.

Furthermore, Gaussian quadrature is more general than (20). For any fixed nonnegative weighting function $w(x)$, Gaussian quadrature creates approximations of the form:

$$\int_a^b f(x)w(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i), \tag{21}$$

for some nodes $x_i \in [a,b]$ and positive weights ω_i, and the approximation (21) is exact whenever f is a degree $2n-1$ polynomial. Specifically, given a nonnegative function $w(x)$ and the $2n$-dimensional family $\mathscr{F}_{2n-1}$ of degree $2n-1$ polynomials, we can find n points $\{x_i\}_{i=1}^n \subset [a,b]$, and n nonnegative weights $\{\omega_i\}_{i=1}^n$ such that:

$$\int_a^b f(x)w(x)dx = \sum_{i=1}^n \omega_i f(x_i),\ \forall f \in \mathscr{F}_{2n-1},$$

based on the following theorem.

Theorem (Judd (1998), p.258): Suppose that $\{\varphi_i(x)\}_{k=0}^\infty$ is an orthogonal family of polynomials with respect to $w(x)$ on $[a,b]$. Furthermore define q_k so that $\varphi_k(x) = q_k x^k + \cdots$. Let $x_i, i = 1,\ldots,n$ be the n zeros of $\varphi_n(x)$. Then, $a < x_i < x_2 < \cdots < x_n < b$, and if $f \in C^{(2n)}[a,b]$, then

$$\int_a^b w(x)f(x)dx = \sum_{i=1}^n \omega_i f(x_i) + \frac{f^{(2n)}(\xi)}{q_n^2(2n)!},$$

for some $\xi \in [a,b]$ and where:

$$\omega_i = -\frac{q_{n+1}/q_n}{\varphi_n'(x_i)\varphi_{n+1}(x_i)} > 0.$$

Furthermore, the formula $\sum_{i=1}^n \omega_i f(x_i)$ is the unique Gaussian integration formula on n nodes that exactly integrates $\int_a^b f(x)w(x)dx$ for all polynomials in $\mathscr{F}_{2n-1}$.

We develop a Gaussian quadrature scheme over any interval $[a,b]$ using any weighting function. A key substantive result of the above theorem is that Gaussian quadrature uses the zeros of the orthogonal polynomials and that they lie in the interval $[a,b]$. Furthermore, the ω_i weights are always positive, avoiding the precision problems of high-order Newton-Cotes formulas. The formulas in theorem tell us how to compute the necessary nodes and weights.

Gauss-Chebyshev quadrature: Integrals of the form:

$$\int_1^1 f(x)(1-x^2)^{-\frac{1}{2}}dx,$$

have the weighting function $(1-x^2)^{-\frac{1}{2}}$, which is the weighting function defining Chebyshev polynomials. To evaluate such integrals, we use the Gauss-Chebyshev quadrature formula:

$$\int_{-1}^1 f(x)(1-x^2)^{-\frac{1}{2}}dx = \frac{\pi}{n}\sum_{i=1}^n f(x_i) + \frac{\pi}{2^{2n-1}}\frac{f^{(2n)}(\xi)}{(2n)!},\ \xi \in [-1,1], \tag{22}$$

where the quadrature nodes are:

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right),\ i = 1, \ldots, n.$$

The Gauss-Chebyshev rule is particularly easy because of the constant weight, π/n, for each node and the easy formula for the quadrature nodes. We will see below that integrals of this form arise naturally when solving various functional equations.

Generally, most researchers do not compute integrals of the form in (22); instead, they compute integrals of the form $\int_a^b f(x)dx$ where the range of integration is $[a, b]$ rather than $[-1, 1]$, and where the weight function, $(1-x^2)^{-\frac{1}{2}}$, is missing in the integrand. To apply Gauss-Chebyshev quadrature, we use the linear change of variables,

$$x = \frac{2(y-a)}{b-a} - 1,$$

to convert the range of integration to $[-1, 1]$, and multiply the integrand by $(1-x^2)^{-\frac{1}{2}}/(1-x^2)^{-\frac{1}{2}}$. This identity implies that:

$$\int_a^b f(y)dy = \frac{b-a}{2}\int_{-1}^{1} f\left(\frac{(x+1)(b-a)}{2} + a\right)\frac{(1-x^2)^{\frac{1}{2}}}{(1-x^2)^{\frac{1}{2}}}dx.$$

We then use the Gauss-Chebyshev quadrature to evaluate the RHS, producing the approximation:

$$\int_a^b f(y)dy \approx \frac{\pi(b-a)}{2n}\sum_{i=1}^{n} f\left(\frac{(x_i+1)(b-a)}{2} + a\right)(1-x_i^2)^{\frac{1}{2}},$$

where the x_i are the Gauss-Chebyshev quadrature nodes over $[-1, 1]$.

Gauss-Legendre quadrature: Integrals over $[-1, 1]$ could use the trivial weighting function, $w(x) = 1$, resulting in the Gauss-Legendre quadrature formula:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n}\omega_i f(x_i) + \frac{2^{2n+1}(n!)^4}{(2n+1)!(2n)!}\frac{f^{(2n)}(\xi)}{(2n)!},\ \xi \in [-1, 1]. \tag{23}$$

A linear change of variables is necessary to apply Gauss-Legendre quadrature to general integrals. In general:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=1}^{n}\omega_i f\left(\frac{(x_i+1)(b-a)}{2} + a\right), \tag{24}$$

where the ω_i and x_i are the Gauss-Legendre quadrature weights and nodes over $[-1, 1]$.

The Gauss-Legendre formula is typical of the rapid convergence of Gaussian quadrature schemes. Applying Stirling's formula, $n! \approx \exp(-n-1)n^{n+(1/2)}\sqrt{2\pi n}$, to the error term in (23), we find that

the error is bounded above by $\pi 4^{-n} M$, where

$$M = \sup_m \left[\max_{-1 \leq x \leq 1} \frac{f^{(m)}(x)}{m!} \right].$$

For many functions, such as analytic functions, M is finite. This bound shows that if M is finite, the convergence of Gauss-Legendre quadrature is of exponential order as the number of quadrature nodes goes to infinity. Since Newton-Cotes formulas are only polynomial in convergence, Gauss-Legendre quadrature is much better when f is C^∞ and its derivatives are tame. The same considerations show that the Gauss-Chebyshev error is also proportional to $4^{-n}M$.

Gauss-Legendre integration can be used to compute discounted sums over finite horizons. For example, suppose that consumption at time t equals $c(t) = 1 + t/5 - 7(t/50)^2$,where $0 \leq t \leq 50$. The discounted utility is $\int_0^{50} \exp(-\rho t) u(c(t)) dt$, where $u(c)$ is the utility function and ρ is the pure rate of time preference. Let $\rho = 0.05$ and $u(c) = c^{1+\gamma}/(1+\gamma)$. We can approximate the discounted utility with (24).

Gauss-Hermite quadrature:

Gauss-Hermite quadrature arises naturally because normal random variables are used often in economic problems. To evaluate $\int_{-\infty}^{\infty} f(x) \exp(-x^2) dx$ using n points, the Gauss-Hermite quadrature rule uses the weights, ω_i, and nodes, x_i, $i = 1, \ldots, n$, and is defined by:

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

Gauss-Hermite quadrature will be used in connection with normal random variables. In particular, if Y is distributed $N(\mu, \sigma^2)$, then:

$$\mathbb{E}[f(Y)] = (2\pi\sigma^2)^{-\frac{1}{2}} \int_{\infty}^{\infty} f(y) \exp\left\{ \frac{-(y-\mu)^2}{2\sigma^2} \right\} dy.$$

However, one must remember that to use Gauss-Hermite quadrature to compute such expectations, it is necessary to use the linear change of variables, $x = (y - \mu)/\sqrt{2}\sigma$, and use the identity:

$$\int_{-\infty}^{\infty} f(y) \exp\left\{ -\frac{-(y-\mu)^2}{2\sigma^2} \right\} dy = \int_{-\infty}^{\infty} f(\sqrt{2}\sigma x + \mu) \exp(-x^2) \sqrt{2}\sigma dx.$$

Hence, the general Gauss-Hermite quadrature rule for expectations of functions of a normal random

variable is:

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

where the ω_i and x_i are the Gauss-Hermite quadrature weights and nodes over $[-\infty, \infty]$.

Examples of Gauss-Hermite quadrature to compute expectations arise naturally in portfolio theory. For example, suppose that an investor holds one bond which will be worth 1 in the future and equity whose value is Z, where $Z \sim N(\mu, \sigma^2)$. If he consumes his portfolio at a future date and his future utility is $u(c)$, then his expected utility is:

$$U = (2\pi\sigma^2)^{-\frac{1}{2}} \int_{-\infty}^{\infty} u(1 + \exp(z)) \exp\left\{-\frac{(z-\mu)^2}{2\sigma^2}\right\} dz,$$

and the certainty equivalent is $u^{-1}(U)$.

Gauss-Laguerre quadrature: Exponential discounted sums are often used in economic problems. To approximate integrals of the form $I = \int_0^{\infty} f(x) \exp(-x) dx$, we use Gauss-Laguerre quadrature. In this case $w(x) = \exp(-x)$, and the appropriate weights, ω_i, and nodes, x_i, $i = 1, \ldots, n$, to use in (21) are listed in Table 1 for various choices of n.

Table 1: Gauss-Laguerre quadrature

N	x_i	ω_i	N	x_i	ω_i
2	0.5857864376	0.8535533905	7	0.1930436765	0.4093189517
	0.3414213562(1)	0.1464466094		0.1026664895(1)	0.4218312778
3	0.4157745567	0.7110930099		0.2567876744(1)	0.1471263486
	0.2294280360(1)	0.2785177335		0.4900353084(1)	0.2063351446(−1)
	0.6289945082(1)	0.1038925650(−1)		0.8182153444(1)	0.1074010143(−2)
4	0.3225476896	0.6031541043		0.1273418029(2)	0.1586546434(−4)
	0.1745761101(1)	0.3574186924		0.1939572786(2)	0.3170315478(−7)
	0.4536620296(1)	0.3888790851(−1)	10	0.1377934705	0.3084411157
	0.9395070912(1)	0.5392947055(−3)		0.7294545495	0.4011199291
5	0.2635603197	0.5217556105		0.1808342901(1)	0.2180682876
	0.1413403059(1)	0.3986668110		0.3401433697(1)	0.6208745609(−1)
	0.3596425771(1)	0.7594244968(−1)		0.5552496140(1)	0.9501516975(−2)
	0.7085810005(1)	0.3611758679(−2)		0.8330152746(1)	0.7530083885(−3)
	0.1264080084(2)	0.2336997238(−4)		0.1184378583(2)	0.2825923349(−4)
				0.1627925783(2)	0.4249313984(−6)
				0.2199658581(2)	0.1839564823(−8)
				0.2992069701(2)	0.9911827219(−12)

Source: Judd (1998)
Note: $a(k)$ means $a \times 10^k$. An (x_i, ω) entry for N means that x is a quadrature node in the N-point formula, weight ω.

The Gauss-Laguerre formulas are defined by:

$$\int_0^\infty f(x)\exp(-x)dx = \sum_{i=1}^{n}\omega_i f(x_i) + (n!)^2\frac{f^{(2n)}(\xi)}{(2n)!},\ \xi\in[0,\infty].$$

To compute the more general integral, $\int_a^\infty f(y)\exp(-ry)dy$, we must use the linear change of variables $x = r(y-a)$, implying that:

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

where ω_i and x_i are the Gauss-Laguerre quadrature weights and nodes over $[0,\infty)$, respectively.

Gauss-Laguerre quadrature can be used to compute the present value of infinitely long streams of utility or profits. For example, suppose that a monopolist faces a demand curve $D(p) = p^{-\eta}$, $\eta > 1$, and has unit cost of $m(t)$ at time t. If unit costs change over time, say $m(t) = a + b\exp(-\lambda t)$, and the interest rate is r, then discounted profits equal:

$$\eta\left(\frac{\eta-1}{\eta}\right)^{\eta-1}\int_0^\infty \exp(-rt)m(t)^{1-\eta}dt. \tag{25}$$

Table 2 displays the errors of several rules in computing (25) with $a = 2$, $b = -1$, and $\eta = 0.8$.

The errors in Table 2 follow an expected pattern. Since we are using the Laguerre formula, the critical piece of the integrand is $m(t)^{1-\eta}$. Gauss-Laguerre integration implicitly assumes that $m(t)^{1-\eta}$ is a polynomial. When $\lambda = 0.05$, $m(t)$ is nearly constant, but when $\lambda = 0.2$, $m(t)^{1-\eta}$ is not constant and becomes less polynomial-like. Therefore it is not surprising that the errors are much larger when $\lambda = 0.2$.

Table 2: Errors in computing (25) with Gauss-Laguerre rule

	$r = 0.05$ $\lambda = 0.05$	$r = 0.10$ $\lambda = 0.05$	$r = 0.05$ $\lambda = 0.20$
Truth:	49.7472	20.3923	74.4005
Nodes: 4	3(−1)	4(−2)	6(0)
5	7(−3)	7(−4)	3(0)
10	3(−3)	6(−5)	2(−1)
15	6(−5)	3(−7)	6(−2)
20	3(−6)	8(−9)	1(−2)

Source: Judd (1998)

3.7 Monte Carlo Integration

One more technique to look at: The idea behind Monte Carlo integration is simple. Consider a random variable x with CDF $F(x)$. Then one can approximate the integral of the function $h(x)$ with:

$$\int_a^b h(x)dF(x) \approx \frac{\sum_{t=1}^T h(x_t)}{T},$$

where $\{x_t\}_{t=1}^T$ is a series drawn from a random number generator corresponding to the distribution of x. Although very simple there is one important disadvantage: it is not very accurate. Above we saw that we can get an accurate answer with just a few quadrature nodes for a large class of functions. Monte Carlo is subject to sampling variation and this only disappears at root n. Suppose we calculate the mean of random variable with a uniform distribution on the unit interval. With $T = 100$ the standard error is 0.029 which is 5.8% of the true mean. Even with $T = 1000$ we have a standard error that is 1.8% of the true mean.

If one doesn't have a CDF then one can use a uniform distribution. That is,

$$\int_a^b h(x)dx = (b-a)\int_a^b h(x)f^{ab}(x)dx,$$

where f^{ab} is the density of a random variable with a uniform distribution over $[a, b]$, that is, $f^{ab} = (b-a)^{-1}$. Thus , one could approximate the integral with:

$$\int_a^b h(x)dx \approx (b-a)\frac{\sum_{t=1}^T h(x_t)}{T},$$

where x_t is generate using a random number generator for a variable that is uniform on $[a, b]$.

Typically one doesn't have access to true random numbers and one only has access to a computer program that generates them. Therefore, these procedures are referred to as pseudo random numbers. The computer program generates data that are (if it is a good program) indistinguishable from a true series of random numbers. But the function that generates the series is deterministic (and chaotic) so that one should be careful in using theorems for true random numbers to think about things like rates of convergence.

3.8 Projection: a worked example

With all that out of the way, let's work through an actual example. Suppose we have the following model which is characterised by the consumption Euler equation, resource constraint, and law of

motion for productivity:

$$\begin{aligned} c_t^{-\gamma} &= \mathbb{E}_t \left[\beta c_{t+1}^{-\gamma} \alpha Z_{t+1} k_{t+1}^{\alpha-1} \right], \\ c_t + k_{t+1} &= Z_t k_t^{\alpha}, \\ \ln Z_{t+1} &= \rho \ln Z_t + \epsilon_{t+1}, \end{aligned}$$

where $\epsilon_{t+1} \sim N(0, \sigma^2)$ and k_0 and Z_0 are given. We do not know the policy rules for this economy so we will use function approximation and numerical integration to find them. Let us use the following notation to define the true rational expectations solution:

$$\begin{aligned} c_t &= c(k_t, Z_t), \\ k_{t+1} &= k(k_t, Z_t), \end{aligned}$$

where we will approximate $c(k_t, Z_t)$ with the polynomial $P_n(k_t, Z_t; \psi_n)$.

What about $k(k_t, Z_t)$? Well, once we have c_t, we can use the budget constraint to back out for k_{t+1}. Bare in mind, we could approximate $c_t$and k_t and then use their approximations to find k_{t+1}. But this is cumbersome. It's far easier to just find c_t first and then find k_{t+1}. So we are solving for the policy rule using function approximations, and let us first define the Euler equation error terms as:

$$e(k_t, Z_t) = -c_t^{-\gamma} + \mathbb{E}_t \left[\beta c_{t+1} \alpha Z_{t+1} k_{t+1}^{\alpha-1} \right],$$

where we will substitute out c_t with $P_n(k_t, Z_t; \psi_n)$.

But we run into our first issue: we only have one optimality condition (from the Euler equation) and we have more than one coefficient for our polynomial approximation (N_n elements of ψ_n). For example, we this is a second-order polynomial then we have six coefficients to pin down with just the one Euler equation. What should we do? Define M grid points for $\{k_t, Z_t\}$ (it could be something like $M = 100$), so let's rewrite things using grids for k and Z:

$$\begin{aligned} e(k_t, Z_t) &= -c_t^{-\gamma} + \mathbb{E}_t \left[\beta c_{t+1} \alpha Z_{t+1} k_{t+1}^{\alpha-1} \right] \\ \Leftrightarrow e(k_i, Z_i) &= -c_i^{-\gamma} + \mathbb{E}_t \left[\beta c_i^{'} \alpha Z_i^{'} (k_i^{'})^{\alpha-1} \right]. \end{aligned}$$

Then, replace c_i with $P_n(k_i, Z_i; \psi_n)$:

$$e(k_i, Z_i) = P_n(k_i, Z_i; \psi_i)^{-\gamma} - \beta \alpha \mathbb{E}_t \left[P_n(k_i^{'}, Z_i^{'}; \psi_n)^{-\gamma} Z_i^{'} (k_i^{'})^{\alpha-1} \right],$$

and then make the appropriate substitutions for $k_i^{'}$ and $Z_i^{'}$, using the AR(1) process (and remember

it's written in logs):

$$= P_n(k_i, Z_i; \psi_i)^{-\gamma} - \beta\alpha\mathbb{E}_t \left\{ \begin{array}{c} P_n \left(Z_i, k_i^{\alpha} - P_n(k_i, Z_i; \psi_n), \exp\left\{\rho \ln Z_i + \epsilon'\right\}; \psi_n' \right)^{-\gamma} \\ \times \exp\left\{\rho \ln Z_i + \epsilon'\right\} \left[Z_i k_i - P_n\left(k_i, Z_i; \psi_n\right)\right]^{\alpha-1} \end{array} \right\}.$$

Lastly, account for the expectations for ϵ' and do the numerical integration (and, do not forget the Jacobian term, $\frac{1}{\sqrt{\pi}}$!):

$$= P_n(k_i, Z_i; \psi_i)^{-\gamma} - \beta\alpha \sum_{j=1}^{J} \omega_j^{\text{GH}} \left\{ \begin{array}{c} P_n \left(Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n), \exp\left\{\rho \ln Z_i + \sqrt{2}\sigma\zeta_j^{\text{GH}}\right\}; \psi_n \right)^{-\gamma} \\ \times \exp\left\{\rho \ln Z_i + \sqrt{2}\sigma\zeta_j^{\text{GH}}\right\} \left[Z_i k_i - P_n\left(k_i, Z_i; \psi_n\right)\right]^{\alpha-1} \end{array} \right\} \frac{1}{\sqrt{\pi}},$$

where ω_i^{GH} and ζ_i^{GH} are the Gauss-Hermite weights and nodes. This expression can be taken to the computer, which will then solve for the coefficients of P_n so that $e(k_i, Z_i; \psi_n) = 0$ is approximately satisfied. Adda and Cooper (2003) has a simple and concise explanation for ways to go about this, but broadly there are two solvers and minimisation routines we can use: collocation and Galerkin.

Collocation: Used when $M = N_n$ and to obtain ψ_n at which $e(k_i, Z_i; \psi_n) = 0$ for each grid point. But the choice of the grid points is important, as you may run into the problem of interpolation.

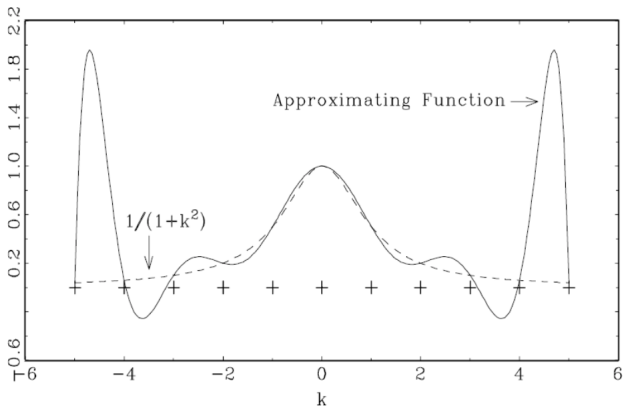
Consider the following example. You get to evaluate a function a set of grid points that you select, and you must guess the shape of the function between the grid points. Consider the 'Runge' function:

$$f(k) = \frac{1}{1+k^2}, \ k \in [-5, 5].$$

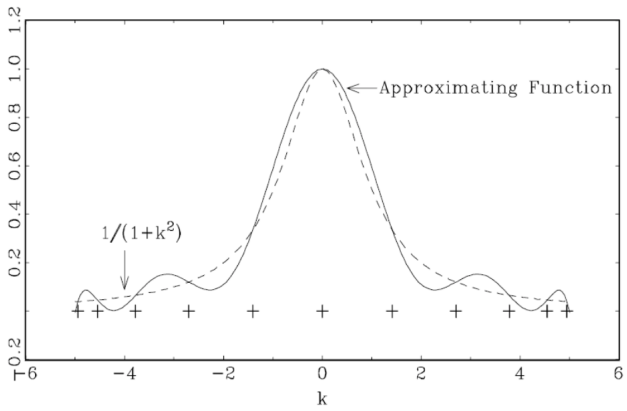
Look what happens when you select 11 equidistant grid points and interpolate by fitting a 10th order polynomial (Figure 8a). As you increase the number of grid points on a fixed interval grid, oscillations in tails grow more and more volatile. One can use the Chebyshev approximation theorem to distribute more points in the tails (by selecting zeros of the Chebyshev polynomial) and get convergence in sup norm (Figure 8b).

Figure 8

(a) Function approximations with fixed interval grid



(b) Function approximations with Chebyshev Zeros



Source: Christiano and Fisher (2000)

Galerkin: Used when $M > N_n$ where a minimisation routine is used to obtain ψ_n:

$$\min_{\psi_n} e(k_i, Z_i; \psi_n).$$

4 Perturbation

Here we show how perturbation techniques can be used to obtain first and higher-order Taylor expansions of the true rational expectations policy function about the steady state. If this sounds familiar, then it should – perturbation is what Dynare uses. We will also discuss the paper of Schmitt-Grohé and Uribe (2004) that makes clear in which way uncertainty affects the policy rules obtained with

perturbation solutions.

We also make clear what the difference is between the first-order approximation obtained with the perturbation procedure and the first-order approximation obtained with what Benigno and Woodford (2012) refer to as the naive LQ procedure. This is the linear solution one obtains using a quadratic approximation of the objective function and a linear approximation of the constraints. T his LQ procedure does not generate, in general, the first-order Taylor expansion of the true rational expectations solution. The reason is that the constraints are only approximated with first-order approximations. The rational expectations solution is itself based on first-order conditions and so the correct first-order Taylor expansion of the true policy rule includes second-order aspects of the objective function as well as the constraints. Moreover, one cannot use a second-order approximation of the constraints because the solution would no longer be linear and the whole convenience of the LQ framework disappears.

This result implies that it is better to get rid of the constraints by substituting out variables. This is not always possible. Benigno and Woodford (2012) show that one can incorporate second-order properties of the constraints into the Lagrangian and still have a standard LQ problem. Using a simple example, we show why this procedure also results in a first-order Taylor expansion of the true solution.

4.1 Case without uncertainty

Consider the standard growth model, where labour is exogenously set to unity and initial capital, k_1, is given:

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

subject to:

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

The Euler equation is given by:

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

When we substitute out consumption using the budget constraint we get:

$$(k_t^{\alpha} + (1-\delta)k_t - k_{t+1})^{-\gamma} = \beta \left(k_{t+1}^{\alpha} + (1-\delta)k_{t+1} - k_{t+2} \right)^{-\gamma} \left(\alpha k_{t+1}^{\alpha-1} + 1 - \delta \right),$$

which is a second-order difference equation in k_t. We are looking for a recursive solution of the form:

$$k_{t+1} = h(k_t).$$

More generally, we are looking for a solution to equations like:

$$f(x'', x', x) = 0, \tag{26}$$

of the form:

$$x' = h(x).$$

To simplify the notation we let (for now) x be a scalar. Define $F(x)$ as:

$$F(x) \equiv f(h(h(x)), h(x), x).$$

Since $h(x)$ is a solution to equation (26), we know that:

$$F(x) = 0.$$

Let $\bar{x}$ be a fixed point of $h(x)$. Thus,

$$\bar{x} = h(\bar{x}).$$

The Taylor expansion of the solution, $h(x)$, about $\bar{x}$ is given by:

$$\begin{aligned} h(x) &\approx h(\bar{x}) + (x - \bar{x})h'(\bar{x}) + \frac{(x - \bar{x})^2}{2}h''(\bar{x}) + \cdots \\ &= \bar{x} + \bar{h}_1(x - \bar{x}) + \bar{h}_2\frac{(x - \bar{x})^2}{2} + \cdots . \end{aligned}$$

So the goal is to find $\bar{x}$, $\bar{h}_1$, $\bar{h}_2$, and so on.

Clearly, $\bar{x}$ has to satisfy:

$$f(\bar{x}, \bar{x}, \bar{x}) = 0.$$

Finding $\bar{x}$ can be a non-trivial problem is f is a nasty non-linear function, but a good equation solver combined with some decent initial conditions should do the trick. The key insight of the perturbation procedure is to solve for the coefficients $\bar{h}_i$ not simultaneously, but sequentially. So let's start.

4.1.1 Finding the coefficient for the linear term, $\bar{h}_1$

For what follows below, it is important to understand that the functional form of f and numerical values of parameter values that appear in f are known. Since:

$$F(x) = 0, \ \forall x,$$

we know that:

$$F'(x) = 0, \ \forall x.$$

The derivative of F is given by:

$$F'(x) = \frac{\partial f}{\partial x''}\frac{\partial h(x')}{\partial x'}\frac{\partial h(x)}{\partial x} + \frac{\partial f}{\partial x'}\frac{\partial h(x)}{\partial x} + \frac{\partial f}{\partial x}. \tag{27}$$

Let:

$$\begin{aligned}
\left.\frac{\partial f(x'', x', x)}{\partial x''}\right|_{x''=x'=x=\bar{x}} &= \bar{f}_1, \\
\left.\frac{\partial f(x'', x', x)}{\partial x'}\right|_{x''=x'=x=\bar{x}} &= \bar{f}_2, \\
\left.\frac{\partial f(x'', x', x)}{\partial x}\right|_{x''=x'=x=\bar{x}} &= \bar{f}_3.
\end{aligned}$$

Also, note that:

$$\left.\frac{\partial h(x)}{\partial x}\right|_{x=\bar{x}} = \left(\bar{h}_1 + \bar{h}_2(x-\bar{x}) + \cdots\right)\big|_{x=\bar{x}} = \bar{h}_1.$$

Using this in equation (27) we get:

$$F'(\bar{x}) = \bar{f}_1\bar{h}_1^2 + \bar{f}_2\bar{h}_1 + \bar{f}_3 = 0.$$

Note that there are no approximations in obtaining this equation. That is, the first-order term of the Taylor expansion of the true policy function is exactly pinned down by this equation. Solving this quadratic equation for $\bar{h}_1$ corresponds to the standard problem of obtaining a solution from the linearised first-order conditions. See, for example, the notes of Uhlig (1998). The concavity of the utility function and the production function implies that the one solution corresponds to an explosive time path and that the other root corresponds to the unique non-explosive solution of the system.

4.1.2 Finding the coefficient for the second-order term, $\bar{h}_2$

Given the solution for $\bar{h}_1$, it is actually relative simple to get the second-order term. Let's calculate $F''(x)$ by differentiating the expression for $F'(x)$ in equation (27):

$$F''(x) = \left\{ \begin{array}{c}
\left(\frac{\partial^2 f}{\partial (x'')^2}\frac{\partial h(x')}{\partial x'}\frac{\partial h(x)}{\partial x} + \frac{\partial^2 f}{\partial x''\partial x'}\frac{\partial h(x)}{\partial x} + \frac{\partial^2 f}{\partial x''\partial x}\right)\left(\frac{\partial h(x')}{\partial x'}\frac{\partial h(x)}{\partial x}\right) \\
+\frac{\partial f}{\partial x''}\left(\frac{\partial h(x')}{\partial x'}\frac{\partial^2 h(x)}{\partial x^2} + \frac{\partial^2 h(x')}{\partial (x')^2}\frac{\partial h(x)}{\partial x}\frac{\partial h(x)}{\partial x}\right) \\
+\left(\frac{\partial^2 f}{\partial x'\partial x''}\frac{\partial h(x')}{\partial x'}\frac{\partial h(x)}{\partial x}\frac{\partial^2 f}{\partial (x')^2}\frac{\partial h(x)}{\partial x} + \frac{\partial^2 f}{\partial x'\partial x}\right)\frac{\partial h(x)}{\partial x} \\
+\frac{\partial f}{\partial x'}\frac{\partial^2 h(x)}{\partial x^2} \\
+\left(\frac{\partial^2 f}{\partial x\partial x''}\frac{\partial h(x')}{\partial x'}\frac{\partial h(x)}{\partial x} + \frac{\partial^2 f}{\partial x\partial x'}\frac{\partial h(x)}{\partial x} + \frac{\partial^2 f}{\partial x^2}\right)
\end{array} \right\}. \tag{28}$$

This looks absolutely horrendous, but we can make it look less intimidating using subscripts to indicate second-order derivatives evaluated about the steady state. For example:

$$\left.\frac{\partial f(x'',x',x)}{\partial x''\partial x}\right|_{x''=x'=x=\bar{x}} = \bar{f}_{13},$$

and also:

$$\left.\frac{\partial^2 h(x)}{\partial x^2}\right|_{x=\bar{x}} = \left(\bar{h}_2 + \bar{h}_3(x-\bar{x}) + \cdots\right)\Big|_{x=\bar{x}} = \bar{h}_2.$$

So, we get:

$$F''(x) = \left\{\begin{array}{c} \left(\bar{f}_{11}\bar{h}_1^2 + \bar{f}_{12}\bar{h}_1 + \bar{f}_{13}\right)\bar{h}_1^2 \\ +\bar{f}_1\left(\bar{h}_1\bar{h}_2 + \bar{h}_2\bar{h}_1^2\right) \\ +\left(\bar{f}_{21}\bar{h}_1^2 + \bar{f}_{22}\bar{h}_1 + \bar{f}_{23}\right)\bar{h}_1 \\ +\bar{f}_2\bar{h}_2 \\ +\left(\bar{f}_{32}\bar{h}_1^2 + \bar{f}_{32}\bar{h}_1 + \bar{f}_{33}\right) \end{array}\right\} = 0.$$

We already solved for $\bar{h}_1$, and this equation is linear in the only unknown, $\bar{h}_2$. so this is an easy equation to solve. Obtaining higher-order terms can be done by repeating this procedure – and all higher-order terms can be solved from a linear system.

4.2 Is perturbation just a local procedure?

To better understand the formal ideas behind perturbation techniques you should check Judd (1998). But the basic idea is the Implicit Function Theorem:

Theorem: Let $H(x,y) : \mathbb{R}^{n+m} \to \mathbb{R}^m$ be a continuously differentiable function and let $\mathbb{R}^{n+m}$ have coordinates (x,y). Fix a point $(\bar{x},\bar{y})$ with $H(\bar{x},\bar{y}) = 0$. If the Jacobian matrix $\mathcal{J}_{H,y}(\bar{x},\bar{y})$ is invertible, then there exists an open U of $\mathbb{R}^n$ containing $\bar{x}$ such that there exists a unique and continuously differentiable function $h : U \to \mathbb{R}^m$ such that:

$$h(\bar{x}) = \bar{y},$$

and

$$H(x, g(x)) = 0, \ \forall x \in U.$$

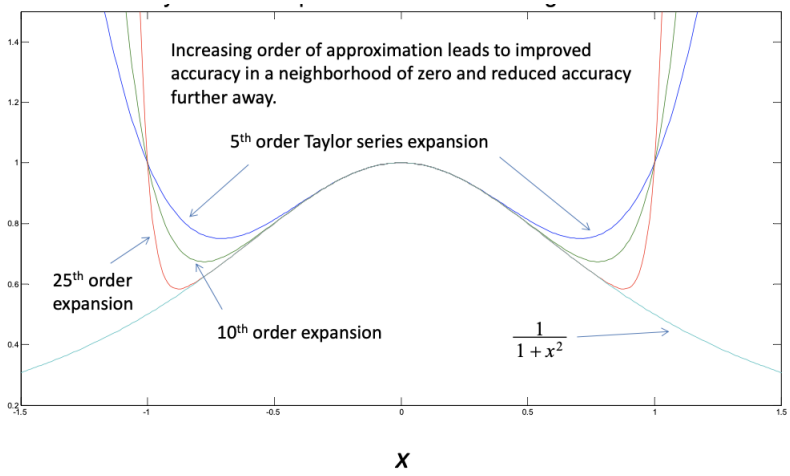
Moreover, the partial derivatives of h in U are given by the matrix product:

$$\frac{\partial h(x)}{\partial x_j} = -\left[\mathcal{J}_{H,y}(x,h(x))\right]^{-1}\left[\frac{\partial H(x,h(x))}{\partial x_j}\right].$$

You may think that perturbation procedures can only provide local approximations and that these techniques are not very good in evaluating the policy functions at values of the state variables that

are not close to the steady state. It is important to realise though that for a smooth and sufficiently differentiable function $d(z)$, one can approximate $d(z^*)$ well using a Taylor expansion about $\bar{z}$ even though one only uses information about d at a point that is far away from z^* (namely $\bar{z}$). The reason is that for a regular function, the functional form of d at z^* is also present in the derivatives of d at $\bar{z}$. For example, suppose that $d(z)$ is a 10th-order polynomial. The value of d at $\bar{z}$ together with the 10 derivatives at $\bar{z}$completely pin down the function. The 10th-order Taylor expansion, thus, would give a perfect approximation for any value z no matter how far away from $\bar{z}$. The story, of course, breaks down if there are non-differentiable components – See Figure 9 for how Taylor approximations struggle to deal with the Runge function.

Figure 9: Taylor series expansions about 0 of the Runge function

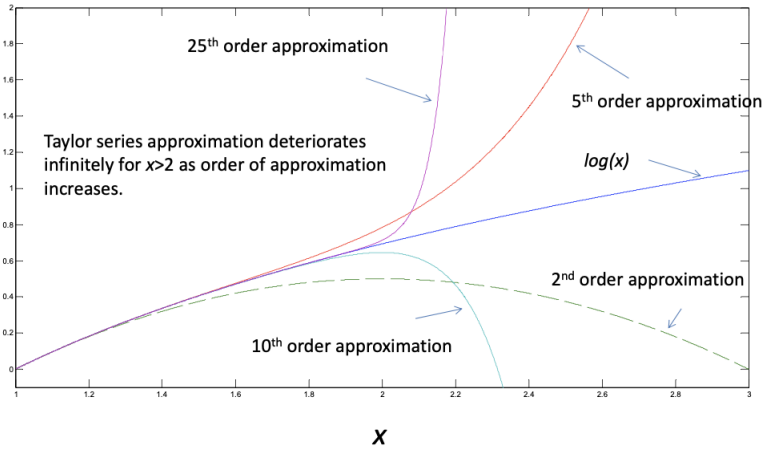


Another example of Taylor expansions struggling: the log function, which is common throughout economics. Surprisingly, Taylor series expansion does not provide a great global approximation. We approximate $\log x$ by its k-th order Taylor series approximation about the point $x = a$:

$$\log x = \log a + \sum_{i=1}^{k} (-1)^{i+1} \frac{1}{i} \left(\frac{x-a}{a} \right)^i .$$

This expression diverges as $N \to \infty$ for x such that $|\frac{x-a}{a}| \geq 1$.

Figure 10: Taylor series expansion of the log function about $x = 1$



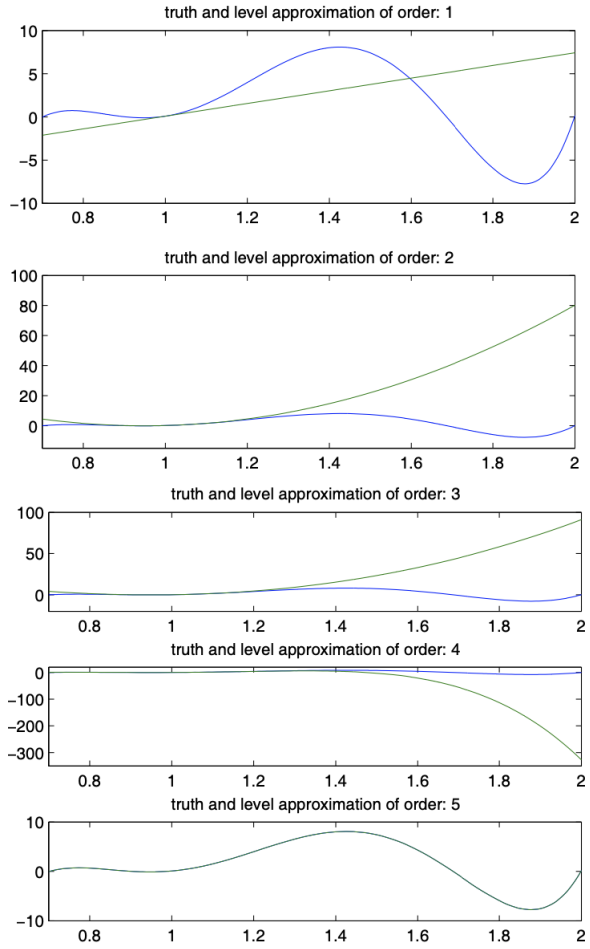
Also, in practice the question is whether low-order perturbation methods are accurate and how they compare with low-order approximations obtained from global numerical solution procedures. For example, suppose that the true policy rule is given by $d(z) = z^{10}$ and $\bar{z} = 0$, then anything below a 10th order perturbation would result in a flat policy function, whereas the truth is not flat.

The following numerical example documents this. It also points out, however, that convergence towards the truth as the order of the polynomial increases can display very strange patterns. The function considered is a fifth-order polynomial equal to:

$$f(x) = -690.59 + 3202.4x - 5739.45x^2 + 4954.2x^3 - 2053.6x^4 + 327.10x^5,$$

defined on the interval $[0.7, 2]$. The five panels of Figure 11 plot the true function and the Taylor approximations about $x = 1$ from the first-order to the fifth-order. This function shows wild oscillations, but the fifth-order Taylor expansion is identical to the truth. Interestingly, of the other approximations, the first-order is the best and the fourth-order is the worst. Note that the scale of the vertical axis is very different in each of the five panels.

Figure 11: Level approximations for the polynomial

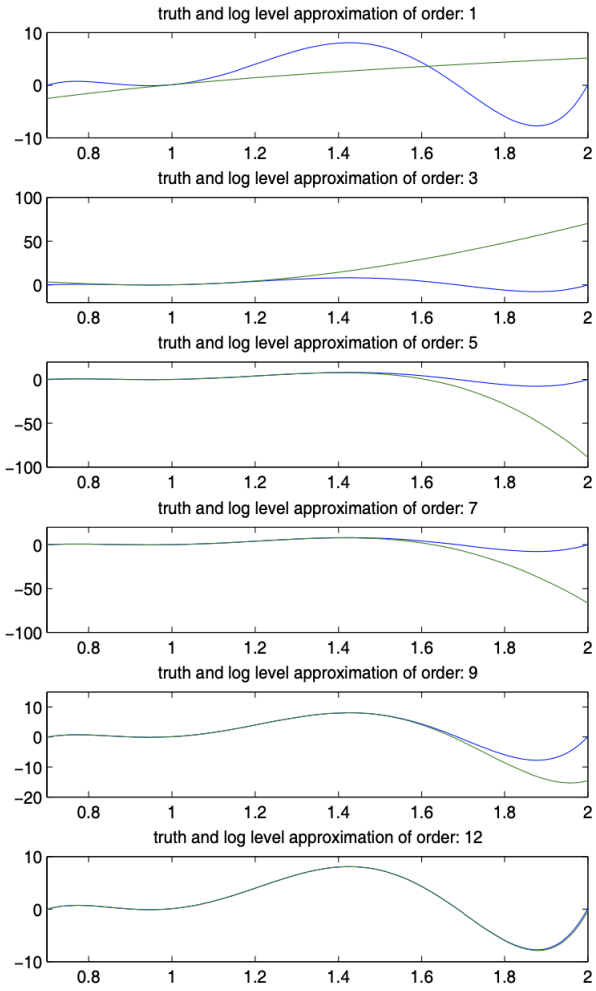


This raises the question, what would convergence look like if one would use a different type of polynomial. For example, suppose we think of $f(x)$ as a function of $z = \log x$. Thus:

$$f(x) = -690.59 + 3202.4 \exp(z) - 5739.45 \exp(2z) + 4954.2 \exp(3z) - 2053.6 \exp(4z) + 327.10 \exp(5z).$$

The six panels of Figure 12 plot the Taylor expansions about $z = 0$ of order 1, 3, 5, 7, 9, and 12. Again, convergence displays an odd pattern with the approximation actually first getting worse if one goes beyond first-order and only around the 9th-order approximation does the approximation start to resemble the truth and converges monotonically towards it. But note that even for the 7th-order approximation the deviation from the truth is huge.

Figure 12: Log level approximations for the polynomial



4.3 The case with uncertainty

Consider the standard growth model with uncertainty:

$$\max_{\{c_t,k_{t+1}\}_{t=0}^{\infty}} \mathbb{E}_t \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\gamma}-1}{1-\gamma},$$

subject to:

$$c_t + k_{t+1} = \exp(Z_t)k_t^{\alpha} + (1-\delta)k_t, \tag{29}$$

where Z_t is a stochastic productivity shock and the initial capital stock k_0 is given. A typical law of motion for Z_t is given by:

$$Z_t = \rho Z_{t-1} + \sigma \epsilon_t, \tag{30}$$

where σ controls the amount of uncertainty.

The consumption Euler equation is given by:

$$c_t^{-\gamma} = \beta \mathbb{E}_t \left[c_{t+1}^{-\gamma} (\alpha \exp(Z_t) k_{t+1}^{\alpha-1} + 1 - \delta) \right]. \tag{31}$$

The budget constraint, law of motion for productivity, and consumption Euler equation give a system of three equations in three unknowns. Such a system can be written as:

$$\mathbb{E} f(x, x', y, y') = 0,$$

where x is an $n_x \times 1$ vector endogenous and exogenous state variables and y is an $n_y \times 1$ vector of endogenous choice/control variables. When applied to the stochastic growth model, we would have that $y = c$ and $x = [k, Z]$.

When solving the model with perturbation techniques, we write the problem such that the amount of uncertainty is controlled by one scalar parameter such as in σ. Even if there are multiple stochastic driving processes one can still use one parameter to scale the amount of uncertainty. If $\sigma = 0$ then there is no uncertainty. Again, the goal is to find the policy functions. To apply the perturbation technique under uncertainty, we follow the following two steps.

4.3.1 Step one: Solution as a function of σ

The first step is to make explicit that σ enters the system of equations and that the policy functions depend on the amount of uncertainty. For the standard growth model we have:

$$\mathbb{E} f([k, Z], [k', \rho Z + \sigma \epsilon'], y, y') = 0.$$

We are trying to solve for functions of the form:

$$y = g(x, \sigma),$$

and

$$x' = h(x, \sigma) + \sigma \eta \epsilon',$$

where ϵ' is an $n_\epsilon \times 1$ vector and η is an $n_x \times \eta_\epsilon$ matrix. The policy functions depend on all the structural parameter values, not just σ. But we will see below why we make explicit that it depends on the amount of uncertainty. For example, for the standard growth model presented, these equations

would be:

$$c = c(k, Z, \sigma),$$

and

$$\begin{bmatrix} k' \\ Z' \end{bmatrix} = \begin{bmatrix} k'(k, Z, \sigma) \\ \rho Z \end{bmatrix} + \sigma \begin{bmatrix} 0 \\ 1 \end{bmatrix} \epsilon'.$$

4.3.2 Step two: Perturb about $\bar{y}$ and $\bar{x}$ and $\sigma = 0$

The second key step of the perturbation procedure is to take a Taylor expansion of the true solution to the system about the steady state values of the variables and around $\sigma = 0$. That is, one starts at the steady state but then allows uncertainty to increase.

A disadvantage of perturbation techniques is that the notation is a bit tedious. Below we will see the notation that the literature has used and see how to do perturbation under uncertainty. Do not be concerned about the difficult notation. Below, we will go back to the case of the standard model and redo the analysis.

Let $\bar{x} = h(\bar{x}, \sigma)$ and $\bar{y} = g(\bar{x}, \sigma)$, giving:

$$f(\bar{x}, \bar{x}, \bar{y}, \bar{y}) = 0.$$

Define the following:

$$\begin{aligned} F(x, \sigma) &= \mathbb{E}_t f[x, x', y, y'] = 0 \\ &= \mathbb{E}_t f[x, h(x\sigma) + \sigma\eta\epsilon', g(x, \sigma), g(x', \sigma)] \\ &= \mathbb{E}_t f[x, h(x, \sigma) + \sigma\eta\epsilon', g(x, \sigma), g(h(x, \sigma) + \sigma\eta\epsilon', \sigma)]. \end{aligned}$$

This is a system of $n = n_x + n_y$ equations in n unknowns. Since we are also perturbing σ(about 0), the Taylor expansions of the true policy functions are given by:

$$g(x, \sigma) = g(\bar{x}, 0) + g_x(\bar{x}, 0)(x - \bar{x}) + g_\sigma(\bar{x}, 0)\sigma + \cdots,$$

and

$$h(x, \sigma) = h(\bar{x}, 0) + h_x(\bar{x}, 0)(x - \bar{x}) + h_\sigma(\bar{x}, 0)\sigma + \cdots.$$

Let:

$$\begin{aligned} &\bar{g}_x = g_x(\bar{x}, 0),\ \bar{g}_\sigma = g_\sigma(\bar{x}, 0),\ \text{and} \\ &\bar{h}_x = h_x(\bar{x}, 0),\ \bar{h}_\sigma = h_\sigma(\bar{x}, 0). \end{aligned}$$

The goal is to find the $n_y \times n_x$ matrix $\bar{g}_x$, the $n_y \times 1$ vector $\bar{g}_\sigma$, the $n_x \times n_x$ matrix $\bar{h}_x$, and the $n_x \times 1$

vector $\bar{h}_\sigma$. The total number of unknowns is thus:

$$(n_x + n_y)(n_x + 1) = n(n_x + 1).$$

We solve for these unknowns by imposing:

$$F_x(\bar{x}, 0) = 0,$$

which gives us $n \times n_x$ equations and:

$$F_\sigma(\bar{x}, 0) = 0,$$

which gives us n equations.

To help with the exposition, we introduce some notation. In particular, we denote the derivative of the i-th element of f with respect to the k-th element of z, for $z \in \{x, y\}$, with:

$$\frac{\partial f^i}{\partial z^k} = [f_z]^i_k.$$

To understand the notation consider the functions $v = v(x)$ and $w = w(x)$, which map $\mathbb{R}^{n_1}$ into $\mathbb{R}^{n_1}$ and a function $D(v, w)$ which maps $\mathbb{R}^{2n_1}$ into $\mathbb{R}^{n_2}$. Now consider the function $D(v(x), w(x))$. The derivative of the i-th element of D with respect to the j-th element of x is equal to:

$$\frac{\partial D^i(v(x), w(x))}{\partial x^j} = \sum_{k_v=1}^{n_1} \frac{\partial f^i}{\partial v^{k_v}} \frac{\partial v^{k_v}}{\partial x^j} + \sum_{k_w=1}^{n_1} \frac{\partial f^i}{\partial w^{k_w}} \frac{\partial w^{k_w}}{\partial x^j}.$$

We will denote this by:

$$\frac{\partial D^i(v(x), w(x))}{\partial x^j} = [f_v]^i_{k_v} [v_x]^{k_v}_j + [f_w]^i_{k_w} [w_x]^{k_w}_j.$$

That is, the index k showing up as a subscript and superscript in adjacent terms indicates the summation. Moreover, the subscript of k indicates over how many elements the summation is. That is, k_v implies summing from $k_v = 1$ up to the number of elements of v.

We can use the same convenient notation if there are no derivatives involved. If η is an $n_x \times n_\epsilon$ matrix and ϵ is an $n_\epsilon \times 1$ vector then:

$$\sum_{k_\epsilon=1}^{n_\epsilon} \eta_{i,k} \epsilon_k = [\eta]^i_{k_\epsilon} [\epsilon']^{k_\epsilon},$$

where $\eta_{i,k}$ is the (i, k) element of η and ϵ_k is the k-th element of ϵ.

Using this notation to calculate the $n \times n_x$ derivatives of F with respect to x, we get:

$$[F_x(\bar{x},0)]^i_j = [\bar{f}_x]^i_j + [\bar{f}_{x'}]^i_{k_h}[\bar{h}_x]^{k_h}_j + [\bar{f}_y]^i_{k_g}[\bar{g}_x]^{k_g}_j + [\bar{f}_{y'}]^i_{k_g}[\bar{g}_x]^{k_g}_{k_h}[\bar{h}_x]^{k_h}_j = 0.$$

An upper bar over the function f indicates that the function is evaluated at the steady state values of x and y and at $\sigma = 0$. Note that the $\bar{h}_x$ and $\bar{g}_x$ terms are multiplied. This is, thus, a second-order system of equations in the $n_x \times n_x$ values of $\bar{h}_x$ and the $n_y \times n_x$ values of $\bar{g}_x$. Although the notation is new, this part of the perturbation routine is identical to what has been done for many years by linearising the first-order conditions. But the perturbation procedure adds the h_σ and g_σ coefficients. Those are solved from:

$$[F_\sigma(\bar{x},0)]^i_j = \left\{ \begin{array}{c} \mathbb{E}_t\left[[\bar{f}_{x'}]^i_{k_h}[\bar{h}_\sigma]^{k_h} + [\bar{f}_{x'}]^i_{k_h}[\eta]^{k_h}_{k_\epsilon}[\epsilon']^{k_\epsilon}\right] \\ +\mathbb{E}_t\left[[\bar{f}_y]^i_{k_g}[\bar{g}_{\sigma]^{k_g}}\right] \\ +\mathbb{E}_t\left[[\bar{f}_{y'}]^i_{k_g}[\bar{g}_x]^{k_g}_{k_h}[\bar{h}_\sigma]^{k_h} + [\bar{f}_{y'}]^i_{k_g}[\bar{g}_x]^{k_g}_{k_h}[\eta]^{k_h}_{k_\epsilon}[\epsilon']^{k_\epsilon}\right] \\ \times\mathbb{E}_t\left[[\bar{f}_{y'}]^i_{k_g}[\bar{g}_\sigma]^{k_g}\right] \end{array} \right\} = 0.$$

This gives us n equations to solve for $\bar{g}_\sigma$ and $\bar{h}_\sigma$. Below we will show that these coefficients are zero so that first-order perturbation will imply the same answer as "old fashioned" linearisation of the first order conditions.

4.4 How does uncertainty matter?

The two important contributions of the perturbation procedure are that it allows for higher-order approximation and that it is explicit about the role of uncertainty. The question arises how important uncertainty is and whether it matters in lower-order approximations. Below we will discuss the very important results from Schmitt-Grohé and Uribe (2004). These results are actually quite straightforward to derive using the notation developed above, but the notation also hides a bit of what is actually going on. So let's go back to the standard growth model and work out the equations in this simpler setup.

In the standard growth model we would have:

$$\begin{aligned} F(x,\sigma) &= \mathbb{E}_t f(k, Z, k', Z', c, c') \\ &= \mathbb{E}_t f\left(k, Z, \underbrace{h(k,Z,\sigma)}_{k'}, \underbrace{\rho Z + \sigma\epsilon'}_{Z'}, \underbrace{g(k,Z,\sigma)}_{c}, \underbrace{g(h(k,Z,\sigma), \rho Z + \sigma\epsilon', \sigma)}_{c'}\right), \end{aligned}$$

where f represents the budget constraint, Euler equation, and the law of motion for Z. It thus maps $\mathbb{R}^6$ into a 3×1 vector.

4.4.1 Uncertainty and the first-order perturbation

To obtain the coefficients $\bar{g}_\sigma$ and $\bar{h}_\sigma$ (which are scalars in this case), we use:

$$F_\sigma(\bar{x}, 0) = 0,$$

so:

$$F_\sigma(x, 0) = \mathbb{E}_t \begin{bmatrix} f_{k'}(s)h_\sigma(k, Z, \sigma) + f_{Z'}(s)\epsilon' + f_c(s)g_\sigma(k, Z, \sigma) \\ +f_{c'}(s)\left\{g_k(k', Z', \sigma)h_\sigma(k, Z, \sigma) + g_Z(k', Z', \sigma)\epsilon' + g_\sigma(k', Z', \sigma)\right\} \end{bmatrix}, \tag{32}$$

where s denotes the arguments of f, that is, $s = [k, Z, k', Z', c, c']$. Evaluating this expression at $\bar{x} = 0$ and calculating the expectation gives:

$$F_\sigma(\bar{x}, 0) = \left(\bar{f}_{k'} + \bar{f}_{c'}\bar{g}_k\right)\bar{h}_\sigma + \left(\bar{f}_c + \bar{f}_{c'}\right)\bar{g}_\sigma = 0.$$

Note that this system gives us three equations, since f consists of three elements, in the two unknowns $\bar{g}_\sigma$ and $\bar{h}_\sigma$. But note that the equation corresponding to the law of motion of productivity gives $0 = 0$, so we are left with two equations in two unknowns.[8] In particular, if we let f^{bc} denote the element of f corresponding to the budget constraint and f^{eu} the element of f corresponding to the Euler equation we get:

$$\begin{bmatrix} \bar{f}^{bc}_{k'} + \bar{f}^{bc}_{c'}\bar{g}_k & \bar{f}^{bc}_c + \bar{f}^{bc}_{c'} \\ \bar{f}^{eu}_{k'} + \bar{f}^{eu}_{c'}\bar{g}_k & \bar{f}^{eu}_c + \bar{f}^{eu}_{c'} \end{bmatrix} \begin{bmatrix} \bar{h}_\sigma \\ \bar{g}_\sigma \end{bmatrix} = 0$$
$$\implies \bar{g}_\sigma = \bar{h}_\sigma = 0,$$

and if there is no singularity in the system then this would be unique solution. this, of course, corresponds to the certainty equivalence that one also obtains if linear policy rules are obtained using the LQ procedure. Although we only show this result in a simple model, Schmitt-Grohé and Uribe (2004) prove that this result holds in more general frameworks.

4.4.2 Uncertainty and second-order perturbation

The second-order Taylor expansions of the policy functions are:

$$h(k, Z, \sigma) = \bar{k} + \bar{h}_k(k - \bar{k}) + \bar{h}_Z(Z - \bar{Z}) + \bar{h}_\sigma\sigma$$
$$+ \frac{1}{2}\begin{Bmatrix} \bar{h}_{kk}(k - \bar{k})^2 + 2\bar{h}_{kZ}(k - \bar{k})(Z - \bar{Z}) + 2\bar{h}_{k\sigma}(k - \bar{k})\sigma \\ +\bar{h}_{ZZ}(Z - \bar{Z})^2 + 2\bar{h}_{Z\sigma}(Z - \bar{Z})\sigma + \bar{h}_{\sigma\sigma}\sigma^2 \end{Bmatrix},$$

[8] $\bar{g}_k$ is known. It is solved from $F_x(\bar{x}, 0) = 0$.

and

$$g(k,Z,\sigma) = \bar{c} + \bar{g}_k(k-\bar{k}) + \bar{g}_Z(Z-\bar{Z}) + \bar{g}_\sigma\sigma + \frac{1}{2}\begin{Bmatrix} \bar{g}_{kk}(k-\bar{k})^2 + 2\bar{g}_{kZ}(k-\bar{k})(Z-\bar{Z}) + 2\bar{g}_{k\sigma}(k-\bar{k})\sigma \\ \bar{g}_{ZZ}(Z-\bar{Z})^2 + 2\bar{g}_{Z\sigma}(Z-\bar{Z})\sigma + \bar{g}_{\sigma\sigma}\sigma^2 \end{Bmatrix}.$$

From the discussion above we know that $\bar{g}_\sigma$ and $\bar{h}_\sigma$ are equal to zero. We will now show that $\bar{g}_{k\sigma}$, $\bar{g}_{Z\sigma}$, $\bar{h}_{k\sigma}$, and $\bar{h}_{Z\sigma}$ are equal to zero too. That is, the value of σ only shows up in the constant of the policy rules. Consider for example, $\bar{g}_{k\sigma}$ and $\bar{h}_{k\sigma}$. These are solved from:

$$F_{k\sigma}(\bar{x},0) = 0.$$

We get the derivative by differentiating the expression in (32). This gives:

$$F_{\sigma k}(x,\sigma) = \mathbb{E}_t \begin{Bmatrix} (f_{k'k} + f_{k'k'}h_k + f_{k'c}g_k + f_{k'c'}g_k h_k)h_\sigma + f_{k'}h_{\sigma k} \\ +(f_{Z'k} + f_{Z'k'}h_k + f_{Z'c}g_k + f_{Z'c'}g_k h_k)\epsilon' \\ +(f_{ck} + f_{ck'}h_k + f_{cc}g_k + f_{cc'}g_k h_k)g_\sigma + f_c g_{\sigma k} \\ +(f_{c'k} + f_{c'k'}h_k + f_{c'c}g_k + f_{c'c'}g_k h_k)(g_k h_\sigma + g_Z\epsilon' + g_\sigma) \\ +f_{c'}(g_{kk}h_k h_\sigma + g_k h_{\sigma k})f_{c'}g_{Zk}h_k\epsilon' + f_{c'}g_{\sigma k}h_k \end{Bmatrix}.$$

Note that the arguments of the functions are suppressed. Here we stop at the second-order. That is, we are not going to differentiate further and suppressing arguments doesn't matter. But if you do want to go on and obtain higher-order terms, it is better not to do so. That is, the above notation doesn't make clear whether g_k stands for $g_k(k,Z,\sigma)$ or $g_k(k',Z',\sigma) = g_k(h(k,Z,\sigma),Z',\sigma)$ and of course this difference is important if you differentiate.

Evaluating the last expression about the steady state we get:

$$\begin{aligned} F_{\sigma k}(\bar{x},0) &= \bar{f}_{k'}\bar{h}_{\sigma k} + \bar{f}_c\bar{g}_{\sigma k} + \bar{f}_{c'}(\bar{g}_k\bar{h}_{\sigma k}) + \bar{f}_{c'}\bar{g}_{\sigma k}\bar{h}_k \\ \Leftrightarrow 0 &= (\bar{f}_{k'} + \bar{f}_{c'}\bar{g}_k)\bar{h}_{\sigma k} + (\bar{f}_c + \bar{f}_{c'})\bar{g}_{\sigma k}\bar{h}_k. \end{aligned}$$

Again, we have two independent equations (together with $0=0$) in two unknowns $\bar{g}_{\sigma k}$ and $\bar{h}_{\sigma k}$. The solution is $\bar{g}_{\sigma k} = \bar{h}_{\sigma k} = 0$ and unless there is a singularity, this is the unique solution.

So only the constants $\bar{g}_{\sigma\sigma}$ and $\bar{h}_{\sigma\sigma}$ are affected by the amount of uncertainty. Do not underestimate the importance of this. A difference value for the constant term in a policy rule implies that the system will operate in a different part of the state space. For example, the higher coefficients can capture a precautionary savings motive that induces agents to have on average higher capital stocks. Now if the second-order terms of capital are not equal to zero then being in a different part of the state space also implies that the response of the system depends on where you are. Consequently, different values for

$\bar{g}_{\sigma\sigma}$ and $\bar{h}_{\sigma\sigma}$ can indirectly also affect how sensitive the economy is to shocks.

4.5 Why not all ways to LQ approximations are correct

A linear quadratic dynamic programming problem has a quadratic objective and linear constraints. This problem the optimal linear regulator problem, is studied extensively and can be solved without numerical error. Suppose that one has a problem which the problem is not quadratic and the constraints not linear. One might be tempted to take quadratic approximation of the objective function and a linear approximation to the constraints and then solve the optimal linear regulator problem. This turns out not to be the right way to implement the LQ procedure in the sense that the linear solution does not in general correspond to the first-order Taylor expansion of the true solution. We will document this using a simple example. In particular, we will first give the set of equations that determine the first-order solution of the perturbation procedure, which by construction gives the first-order Taylor expansion of the true policy function. This makes clear that the correct first-order Taylor expansion also depends on second-order terms of the constraint. Benigno and Woodford (2012) refer to this way to take LQ approximations as the "naive LQ approximation". Next we use the example to discuss a modified LQ procedure that is correct.

WE focus on the following model:

$$\max_{x-y} \min_{\lambda} \; f(x,y,a) + \lambda(b - g(x,y,a)),$$

subject to:

$$\lambda \geq 0.$$

Here f and g are scalar functions and x, y, and a are scalars as well. The simple structure will be helpful in clarifying the points made but the arguments are much more general. The first-order conditions can be written as:

$$\begin{aligned} f_x(x,y,a) - \lambda g_x(x,y,a) &= 0, \\ f_y(x,y,a) - \lambda g_y(x,y,a) &= 0, \\ g(x,y,a) &= b. \end{aligned}$$

The solution to this system of equations is:

$$\begin{aligned} x &= h^x(a), \\ y &= h^y(a), \\ \lambda &= h^\lambda(a). \end{aligned}$$

Using this, we can write the first-order conditions as:

$$\begin{aligned} f_x(h^x(a), h^y(a), a) - \lambda(a) g_x(h^x(a), h^y(a), a) &= 0, \\ f_y(h^x(a), h^y(a), a) - \lambda(a) g_y(h^x(a), h^y(a), a) &= 0, \\ g(h^x(a), h^y(a), a) &= b. \end{aligned}$$

The Taylor expansions of the policy functions are given by:

$$\begin{aligned} h^x(a) &= \bar{h}^x + \bar{h}^x_a(a - \bar{a}) + \bar{h}^x_{aa}(a - \bar{a})^2 + \cdots, \\ h^y(a) &= \bar{h}^u + \bar{h}^y_a(a - \bar{a}) + \bar{h}^y_{aa}(a - \bar{a})^2 + \cdots, \\ h^\lambda(a) &= \bar{h}^\lambda + \bar{h}^\lambda_a(a - \bar{a}) + \bar{h}^\lambda_{aa}(a - \bar{a})^2 + \cdots. \end{aligned}$$

The first-order derivatives evaluated about $a = \bar{a}$ are $\bar{h}^x_a$, $\bar{h}^y_a$, and $\bar{h}^\lambda_a$. Differentiating the first-order conditions and evaluating them about $a = \bar{a}$ gives:

$$\begin{aligned} \bar{f}_{xx}\bar{h}^x_a + \bar{f}_{xy}\bar{h}^y_a + \bar{f}_{xa} - \bar{\lambda}(\bar{g}_{xx}\bar{h}^x_a + \bar{g}_{xy}\bar{h}^y_a + \bar{g}_{xa}) - \bar{g}_x\bar{h}^\lambda_a &= 0, \\ \bar{f}_{yx}\bar{h}^x_{xa} + \bar{f}_{yy}\bar{h}^y_a + \bar{f}_{ya} - \bar{\lambda}(\bar{g}_{yx}\bar{h}^x_a + \bar{g}_{yy}\bar{h}^y_a + \bar{g}_{ya}) - \bar{g}_y\bar{h}^\lambda_a &= 0, \\ \bar{g}_x\bar{h}^x_a + \bar{g}_y\bar{h}^y_a + \bar{g}_a &= 0. \end{aligned}$$

This can be written as:

$$\begin{bmatrix} \bar{f}_{xx} - \bar{\lambda}\bar{g}_{xx} & \bar{f}_{xy} - \bar{\lambda}\bar{g}_{xy} & -\bar{g}_x \\ \bar{f}_{yx} - \bar{\lambda}\bar{g}_{yx} & \bar{f}_{yy} - \bar{\lambda}\bar{g}_{yy} & -\bar{g}_y \\ -\bar{g}_y & -\bar{g}_y & 0 \end{bmatrix} \begin{bmatrix} \bar{h}^x_a \\ \bar{h}^y_a \\ \bar{h}^\lambda_a \end{bmatrix} = \begin{bmatrix} -\bar{f}_{xa} + \bar{\lambda}\bar{g}_{xa} \\ -\bar{f}_{ya} + \lambda\bar{g}_{ya} \\ -\bar{g}_a \end{bmatrix}. \tag{33}$$

With this system we can solve for the first-order perturbation terms. Note that they depend on the second-order properties of the constraints ($\bar{g}_{xx}$, $\bar{g}_{xy}$, and etc.). These would not show up when one linearises the constraints.

4.6 Correct LQ procedure of Benigno and Woodford (2012)

There are different ways in which one can deal with the problem encountered above. One could try to get rid of the constraints (or make them linear) by substituting out or redefining variables. This is not always possible. Here we discuss a general LQ procedure for which the linear solution does correspond to the first-order Taylor expansion. That is, it corresponds with the solution procedure implied by equation (33). The standard formulation of the LQ approximation can be written as:

$$\max_{x,y} \min_{\lambda} \; \bar{f}_x\tilde{x} + \bar{f}_y\tilde{y} + \bar{f}_a\tilde{a} + \frac{1}{2}\tilde{\mathbf{\Lambda}}^\top \bar{\mathbf{\Phi}}\tilde{\mathbf{\Lambda}} - \lambda(\bar{g}_x\tilde{x} + \bar{g}_y\tilde{y} - \bar{g}_a\tilde{a}), \tag{34}$$

where $\tilde{z} = z - \bar{z}$ and:

$$\tilde{\mathbf{\Lambda}} = \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{a} \end{bmatrix}, \ \tilde{\mathbf{\Phi}} = \begin{bmatrix} \bar{f}_{xx} & \bar{f}_{xy} & \bar{f}_{xa} \\ \bar{f}_{yx} & \bar{f}_{yy} & \bar{f}_{ya} \\ \bar{f}_{ax} & \bar{f}_{ay} & \bar{f}_{aa} \end{bmatrix}.$$

Now consider the second-order approximation of the constraint:

$$0 \approx \bar{g}_x \tilde{x} + \bar{g}_y \tilde{y} + \bar{g}_a \tilde{a} + \frac{1}{2} \tilde{\mathbf{\Lambda}}^\top \tilde{\mathbf{\Gamma}} \tilde{\mathbf{\Lambda}},$$

where:

$$\tilde{\mathbf{\Gamma}} = \begin{bmatrix} \bar{g}_{xx} & \bar{g}_{xy} & \bar{g}_{xa} \\ \bar{g}_{yx} & \bar{g}_{yy} & \bar{g}_{ya} \\ \bar{g}_{ax} & \bar{g}_{ay} & \bar{g}_{aa} \end{bmatrix}.$$

Next we multiply both sides of this expression by $\bar{\lambda}$ and use the first-order conditions at the steady state values, i.e., about $a = \bar{a}$. This gives:

$$\bar{\lambda} \bar{g}_x \tilde{x} + \bar{\lambda} \bar{g}_y \tilde{y} + \bar{\lambda} \bar{g}_a \tilde{a} + \frac{\bar{\lambda}}{2} \tilde{\mathbf{\Lambda}}^\top \tilde{\mathbf{\Gamma}} \tilde{\mathbf{\Lambda}}, \tag{35}$$

and subtracting this expression from the Lagrangian (34) (and ignoring constant terms) gives:

$$\max_{x,y} \min_{\lambda} \ \frac{1}{2} \tilde{\mathbf{\Lambda}}^\top \left(\tilde{\mathbf{\Phi}} - \bar{\lambda} \tilde{\mathbf{\Gamma}} \right) \tilde{\mathbf{\Lambda}} + \lambda \left(b - \bar{g} - \bar{g}_x \tilde{x} - \bar{g}_y \tilde{y} - \bar{g}_a \tilde{a} \right). \tag{36}$$

By entering second-order properties of the constraint we have at least some chance of getting the correct first-order Taylor expansion. Also note that the first-order terms have disappeared from the objective function. This is already a convenient property. The linear solution that comes out of this is going to be at best the correct first-order Taylor expansion. Consequently, it is in general wrong in the second-order terms. But if you substitute such a policy function into the linear terms of the objective function then the objective function also has second-order mistakes. But the objective function is supposed to be the correct second-order approximation.

Anyway, we are really interested in knowing whether this leads to the correct first-order Taylor expansion. To see this, calculate the first-order conditions of this problem. They are given by:

$$\begin{bmatrix} \bar{f}_{xx} - \bar{\lambda} \bar{g}_{xx} & \bar{f}_{xy} - \bar{\lambda} \bar{g}_{xy} & -\bar{g}_x \\ \bar{f}_{yx} - \bar{\lambda} \bar{g}_{yx} & \bar{f}_{yy} - \bar{\lambda} \bar{g}_{yy} & -\bar{g}_y \\ -\bar{g}_x & -\bar{g}_y & 0 \end{bmatrix} \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \lambda \end{bmatrix} = \begin{bmatrix} -\bar{f}_{xa} + \bar{\lambda} \bar{g}_{xa} \\ -\bar{f}_{ya} + \bar{\lambda} \bar{g}_{ya} \\ -\bar{g}_a \end{bmatrix} \tilde{a}. \tag{37}$$

But this system corresponds exactly to (33).

4.6.1 Modification of the objective function and change in Lagrangian multiplier

The procedure outlined above boils down to subtracting the second-order formulation of the constraint given in (35) from the original objective function given in (34). Thus, whenever the constraints hold, this does not entail any change in the objective function since one would simply be adding zero to the objective function, This is not true when the constraints are not satisfied, so we are indeed changing the objective function. Note that the modification of the objective function results in a zero value of the Lagrange multiplier when $a = 0$ even though in the original problem $\bar{\lambda} > 0$. Note that by construction, the constraint does still hold with equality when $a = 0$. If one is interested in getting the right value of λ that is close to that of the original problem, one can replace λ by $(\lambda - \bar{\lambda})$ in (36).

This shift in the objective function does have an effect on the Lagrange multiplier. In fact, (37) makes clear that when $\tilde{a} = 0$, that the value of the Lagrange multiplier is equal to zero. In fact, (37) makes clear that when $\tilde{a} = 0$, that the value of the Lagrange multiplier is equal to zero.[9] But we have also seen that this shift does not have an effect on the first-order term that relates changes in $\tilde{a}$ to changes in λ. An easy way to "undo" the effect of the shift of the objective function on the Lagrange multiplier is to use the following modified LQ specification:

$$\max_{x,y} \min_{\lambda} \ \frac{1}{2}\tilde{\mathbf{\Lambda}}^{\top}\left(\tilde{\mathbf{\Phi}} - \bar{\lambda}\tilde{\mathbf{\Gamma}}\right)\tilde{\mathbf{\Lambda}} + \tilde{\lambda}\left(b - \bar{g} - \bar{g}_x\tilde{x} - \bar{g}_y\tilde{y} - \bar{g}_a\tilde{a}\right). \tag{38}$$

That is, we replace λ by $\tilde{\lambda} = \lambda - \bar{\lambda}$. Note that this is not a replacement of the Lagrange multiplier term by its first-order approximation. This would give $\lambda = \tilde{\lambda} + \bar{\lambda}$. Obviously, replacing λ by $\tilde{\lambda}$ is just a change in notation and the value for $\tilde{\lambda}$ obtained with (38) we will now get that $\tilde{\lambda} = 0$ when $\tilde{a} = 0$. But this means that if we use (38) we get $\lambda = \bar{\lambda}$ when $\tilde{a} = 0$, which is the desired outcome.

[9] Note that although the Lagrange multiplier is zero, the constraint is still satisfied at $\tilde{a} = 0$, it is just not binding.

References

Adda, J. and Cooper, R. (2003), *Dynamic Economics* (MIT Press).

Bellman, R. (1957), *Dynamic Programming* (Princeton University Press).

Benigno, P. and Woodford, M. (2012), "Linear-Quadratic Approximation of Optimal Policy Problems", *Journal of Economic Theory*, 147/1: 1–42.

Benveniste, L. M. and Scheinkman, J. A. (1979), "On the Differentiability of the Value Function in Dynamic Models of Economics", *Econometrica*, 47: 727–32.

Christiano, L. J. and Fisher, J. D. (2000), "Algorithms for Solving Dynamic Models with Occasionally Binding Constraints", *Journal of Economic Dynamics and Control*, 24/8: 1179–232.

Judd, K. (1998), *Numerical Methods in Economics* (MIT Press).

Ljungqvist, L. and Sargent, T. J. (2018), *Recursive Macroeconomic Theory* (4th Edition, MIT Press).

McCandless, G. (2008), *ABCs of RBCs* (Harvard University Press).

Schmitt-Grohé, S. and Uribe, M. (2004), "Solving Dynamic General Equilibrium Models using a Second-Order Approximation to the Policy Function", *Journal of Economic Dynamics and Control*, 28/4: 755–75.

Uhlig, H. (1998), "A Toolkit for Analyzing Nonlinear Dynamic Stochastic Models Easily".