Christiano,
Econ 416
Fall, 2014

Notes on Second Order Perturbation and Symbolic Algebra

This handout has two parts. The first describes basic aspects of symbolic algebra in MATLAB needed to solve the neoclassical model by perturbation. The second section provides a series of hints about how to actually compute the second order perturbation.[1]

1 Symbolic Algebra

You need to differentiate between two types of objects: actual numbers and symbolic numbers. An actual number is what it sounds like, it's a variable that takes on a numerical value. For example, suppose b is an actual number, then to do anything with it you must assign it a specific value, such as $b = 7$. A symbolic variable is what we informally refer to as a *variable*, an abstract object that can take on numerical value in some specific range. For example, we often use x to represent a variable that is an element of the real line and we do manipulations on it which do not depend on the precise value taken on by x. Thus, in the case of the expression, $f(x) = x^2$, we compute the derivative, $f'(x)$, and find that it is equal to $2x$ without having to take a stand on what specific value x takes on. So, x is a symbolic variable.

When you work with a variable in MATLAB, the assumption is that it is what I called an actual number in the previous paragraph. But, if you want the variable to be a symbolic variable, then you have to declare it as such using a syms command. Suppose, for example, x and y are symbolic variables. Then, you declare this in MATLAB using the following expression:

$$syms\ x\ y$$

(No need for a ';' after this expression.) A symbolic variable can also be defined implicitly. Suppose you've defined x and y as symbolic and you've set $a = 0.4309890234$ and $b = 7$. That is, a and b are normal variables (you'll see the reason for the weird value for a in a moment). Then, consider the following expression:

$$f = a * x + b * y; \tag{1}$$

[1] This note assumes familiarity with the discussion of perturbation in the class handout, http://faculty.wcas.northwestern.edu/~lchrist/course/Korea_2012/lecture_on_solving_rev.pdf. For a recent critical analysis of perturbation as a solution method, see den Haan and de Wind, 2009, "How well-behaved are higher-order perturbation solutions," DNB working paper, no. 240, December, and den Haan and de Wind, 2021, "Nonlinear and Stable Perturbation-Based Approximations," available at http://www.wouterdenhaan.com/papers/MS7284_main_text.pdf.

Because the variables on the right of the equality are symbolic variables, then f is (implicitly) defined as a symbolic variable in this expression. In addition, the entire expression is referred to as a *symbolic expression*.

Type the syms statement, the values for a and b and the expression, (1), at the command line in MATLAB. Then, type f at the command line to see what MATLAB thinks f is. You will see that it is the algebraic expression, (1). The reason for the somewhat odd value for a is that you'll be acquainted with an interesting feature of symbolic algebra in MATLAB. Real numbers are stored as rational numbers, so that a is actually represented as 970501002592507/2251799813685248. You'll see that when you type f at the MATLAB prompt.

The perturbation method is all about differentiation. To differentiate a symbolic expression with respect to a symbolic variable, say x, simply use the command, $diff\,(f,x)$. Try this with the example above and note that $diff\,(f,x)$ is 7, as it should be. Try it again with $f = a + b * y^2$, and you'll see that the answer is the symbolic expression, $14 * y$.

Another thing we want to do when we apply the perturbation method is to evaluate a derivative at a specific value for some or all of the symbolic variables. To do this, you can use the subs command. Thus, to evaluate the derivative of f with respect to y at $y = 7$, first compute the derivative, $df = diff(f,y)$. Then, execute $subs\,(df,[y],[2])$.

When we apply the perturbation method in dynamic models we find ourselves convolving functions. Thus, consider consumption, $c = f - k'$, where, say, $f = \exp(z)\,k^{\alpha}$.[2] Here, α is a variable and z, k, k' are symbolic variables. Then, the equation for c implicitly defines c as a symbolic variable. The expression for c is a symbolic expression that expresses c as a function of z, k, k'.

Although it might be convenient for debugging purposes to have the easily recognized expression, $c = f - k'$ in our code, c may in fact be a function of k, z only. This would happen if, for example, we think of k' as being a function of k and z, say, $k' = 6 * k + 7 * k * z$. In this case, c is actually only a function of k and z alone:

$$c = f - G,$$

where, say, $G = 6 * k + 7 * k * z$. Given $c = f - k'$, we can express c as a function of k and z alone by using the sub command. After typing $c = f - k'$, you type the symbolic expression defining G. Then, you write $c = subs\,(c,[k'],[G])$. The object on the right of the equality replaces the symbolic variable, k', in the symbolic expression for c with the symbolic expression, G. The resulting symbolic expression that involves only k and z becomes the new symbolic expression for c when the equality is evaluated.

Not surprisingly, if you do a lot of substitutions like the ones above, you may become unsure about what symbolic variables enter a particular symbolic expression. To find out, you can execute the command, $symvar\,(c)$, and it

[2] The object, k', is not a valid variable name in MATLAB because ' is interpreted as the transposition operator. In practice, you would have to express k' in a different way, for example, as kp.

will list all the symbolic variables in the expression for c. Thus, immediately after executing $c = f - k'$, the output for $symvar\,(c)$ is k, z, k'. After executing $c = subs\,(c, [k'], [G])$, the output for $symvar\,(c)$ is k, z.

There is one more type of calculation that must be done to do perturbations. You have to find the value of a symbolic variable that sets an equation to zero. For example, in class we have discussed the derivative of an error function, R, with respect to k, evaluated at the steady state values of variables. We found that this object, R_k, is a function of only one unknown, g_k. (We obtain R by combining a number of symbolic expressions. We then obtain R_k by applying the diff operator to R and evaluating the result in steady state (two applications of the *subs* command).)

We seek a value of g_k that sets $R_k = 0$. This solution may be found in MATLAB in two steps. First, execute $sym2poly(R_k)$. This MATLAB function returns the coefficients in the polynomial expression of R_k in terms of g_k. The coefficient on the highest power of g_k appears first, the second highest power second, and so on. In the case of g_k in the lecture we saw that the polynomial is second order. (In the case of all other coefficients in the Taylor series expansion of the policy rule the relevant derivative of R is a *linear* function of the unknown.) Second, apply the MATLAB command, *root*, to the result, to find the values of g_k that set $R_k = 0$. In the case of g_k we showed that there are two values, and you have to pick the smaller of the two. (In the case of the other coefficients, there is exactly one value (conditional on the value of g_k).) In practice, the two steps just described can be put into one command as follows, $root\,(sym2poly(R_k))$.

2 Second Order Perturbation for the Neoclassical Model

We now turn to the problem of applying the perturbation method to solving the neoclassical growth model, using the symbolic algebra tools described above. The error function is as follows:

$$R\,(k, a, \varepsilon, \sigma; g) = E_{\varepsilon'}\left[u_c\,(c) - \beta u_c\,(c')\,f'_k\right], \tag{2}$$

where

$$\begin{aligned}
c &= f\,(k, a, \varepsilon) - k' \\
c' &= f\,(k', \rho a + \varepsilon, \sigma\varepsilon') - k'' \\
f\,(k, a, \varepsilon) &= \exp\,(\rho a + \varepsilon)\,k^{\alpha} + (1-\delta)\,k, \\
f'_k &= \alpha \exp\,(\rho\,[\rho a + \varepsilon] + \sigma\varepsilon')\,(k')^{\alpha-1} + (1-\delta), \\
u\,(c) &= \frac{c^{1-\gamma} - 1}{1-\gamma},\ u_c\,(c) = c^{-\gamma},
\end{aligned}$$

where ‘′’ indicates the value of a variable in the next period. Also, a denotes the value of technology *in the previous period* and the current value of technology is

$\rho a + \varepsilon$. So, the state of the system is k, a, ε.[3] What I mean by the subscript, ε', on the E operator is that the expectation is taken over the values of the random variable, ε', i.e., the value the shock that will be realized in the next period.

It is worth dwelling for a moment on the asymmetric treatment of the current and next period's shock. The current period shock enters as ε and the next period's shock enters as $\sigma\varepsilon'$. The actual problem of interest is characterized by symmetry, and has $\sigma = 1$. We allow for asymmetry because it gives us an important technical advantage for approximating g. The case, $\sigma = 0$, corresponds to the scenario in which ε' does not enter the problem, so that all the variables in the problem are known, i.e., the problem is deterministic. In this special situation, there is a value of the arguments of g (which now correspond to k, a, ε, *and* σ) for which we know the value of g. This is crucial for us to be able to apply the perturbation method. For example, suppose we did not introduce σ and in effect adopted the specification, $\sigma = 1$. In this case g is a function only of k, a, ε, and there is no value of these variables for which we know the value of g. We would not be able to apply the perturbation method. Once we are done using the perturbation method to obtain an approximation to g we set $\sigma = 1$, the economically interesting case.

The policy rule we're after is expressed as follows:

$$k' = g\left(k, a, \varepsilon, \sigma\right),$$

where the state is k, a, ε. We seek the first order Taylor series expansion of the above object, about *non-stochastic steady state*, $(k, a, \varepsilon, \sigma) = (k^*, 0, 0, 0)$:

$$\begin{aligned} k' &= k^* + g_k\left(k - k^*\right) + g_a a + g_\varepsilon \varepsilon + g_\sigma \sigma \\ &\quad + \frac{1}{2}\left[g_{kk}\left(k - k^*\right)^2 + g_{aa}a^2 + g_{\varepsilon\varepsilon}\varepsilon^2 + g_{\sigma\sigma}\sigma^2\right] \\ &\quad + g_{ka}ka + g_{k\varepsilon}k\varepsilon + g_{k\sigma}k\sigma + g_{a\varepsilon}a\varepsilon + g_{a\sigma}a\sigma + g_{\varepsilon\sigma}\varepsilon\sigma. \end{aligned} \tag{3}$$

Here, g_x is the derivative of g with respect to x, evaluated in steady state and g_{xy} is the cross derivative of g with respect to x and y, also evaluated in steady state. You should define the objects, $g_x, g_{x,y}$ as symbolic variables, for $x, y = a, \varepsilon, \sigma$. The values of these objects that we seek are the ones that solve $R_x = R_{x,y} = 0$ for $x, y = a, \varepsilon, \sigma$. In class we discussed the results that $g_\sigma = g_{k\sigma} = g_{a\sigma} = g_{\varepsilon\sigma} = 0$. You can check whether these results are in fact true by not imposing them when you do your computations.

There is one issue that has not been addressed yet, which is how to evaluate the expectation in the definition of R in (2). Among other things, this discussion will reveal exactly how the variance of ε' enters the computations. The object in

[3] I have deviated in two ways from the position taken in the class handout: (i) there the state of the system was k and the current value of the technnology shock, which was denoted by a;and (ii) there, k, was interpreted as the log of the technology shock. The change in (i) makes no mathematical difference, though it puts our approach in line with the convention used in the computational package, Dynare. It would be interesting to explore which of the two options, interpreting k as the log or the level of capital, produces a better approximation. This can be accomplished by studying the error function in each case.

square brackets in (2) is a function of $k, a, \varepsilon, \sigma$ and $\sigma\varepsilon'$, after convolving various functions and taking into account that the next period's shock enters as $\sigma\varepsilon'$. Specifically, we can write (2) as follows:

$$E_{\varepsilon'} h\left(x, \sigma, \sigma\varepsilon'\right), \tag{4}$$

where

$$\begin{aligned} & h\left(x, \sigma, \sigma\varepsilon'\right) \\ \equiv \quad & u_c\left(\overbrace{f\left(k, a, \varepsilon\right) - g\left(k, a, \varepsilon, \sigma\right)}^{c}\right) - u_c\left(\overbrace{f\left(g\left(k, a, \varepsilon, \sigma\right), \rho a + \varepsilon, \sigma\varepsilon'\right) - g\left(g\left(k, a, \varepsilon, \sigma\right), \rho a + \varepsilon, \sigma\varepsilon', \sigma\right)}^{c'}\right) \\ & \times \left[\overbrace{\alpha \exp\left(\rho\left[\rho a + \varepsilon\right] + \sigma\varepsilon'\right)\left[g\left(k, a, \varepsilon, \sigma\right)\right]^{\alpha-1} + 1 - \delta}^{f_k'}\right], \end{aligned} \tag{5}$$

where $x \equiv (k, a, \varepsilon)$. From (2), we see that $R\left(k, a, \varepsilon, \sigma; g\right) = E_{\varepsilon'} h\left(x, \sigma, \sigma\varepsilon'\right)$. Note that when we differentiate the object in (4) with respect to k, a, ε and evaluate the resulting derivative in steady state, ε' disappears because it is multiplied by σ, which is zero in steady state.[4] So, evaluating the expectation operator is trivial in these cases. But, things are different in the case of differentiation with respect to σ.

Note that σ enters h in (5) in two places. The first is indicated by the second argument in h, which is there to capture the impact of σ on the last argument in g. The second way σ enters is by multiplying the next period's shock, i.e., $\sigma\varepsilon'$. That operates through the third argument of next period's g function and also via the level and derivative of the production function in the next period. These two ways in which σ enters (4) can be seen explicitly in (5). Consider first the derivative of (4) with respect to σ, evaluated in steady state, $x = x^*, \sigma = 0$:

$$\begin{aligned} \frac{d}{d\sigma} E_{\varepsilon'} h\left(x, \sigma, \sigma\varepsilon'\right)|_{\sigma=0, x=x^*} &= E_{\varepsilon'} h_2\left(x, \sigma, \sigma\varepsilon'\right)|_{\sigma=0, x=x^*} + E_{\varepsilon'}\left[h_3\left(x, \sigma, \sigma\varepsilon'\right)|_{\sigma=0, x=x^*}\right]\varepsilon' \\ &= h_2\left(x^*, 0, 0\right), \end{aligned}$$

making use of the fact, $E_{\varepsilon'}\varepsilon' = 0$.

Now consider the second derivative with respect to σ :

$$\begin{aligned} \frac{d}{d\sigma d\sigma} E_{\varepsilon'} h\left(x, \sigma, \sigma\varepsilon'\right)|_{\sigma=0, x=x^*} &= \frac{d}{d\sigma}\left[\frac{d}{d\sigma} E_{\varepsilon'} h\left(x, \sigma, \sigma\varepsilon'\right)\right]|_{\sigma=0, x=x^*} \\ &= \frac{d}{d\sigma} E_{\varepsilon'}\left[h_2\left(x, \sigma, \sigma\varepsilon'\right) + h_3\left(x, \sigma, \sigma\varepsilon'\right)\varepsilon'\right]|_{\sigma=0, x=x^*} \\ &= E_{\varepsilon'}\left[h_{22}\left(x, \sigma, \sigma\varepsilon'\right) + 2h_{23}\left(x, \sigma, \sigma\varepsilon'\right)\varepsilon' + h_{33}\left(x, \sigma, \sigma\varepsilon'\right)\left(\varepsilon'\right)^2\right]|_{\sigma=0, x=x^*} \\ &= h_{22}\left(x^*, 0, 0\right) + 2h_{23}\left(x^*, 0, 0\right) E_{\varepsilon'}\varepsilon' + h_{33}\left(x^*, 0, 0\right) E_{\varepsilon'}\left(\varepsilon'\right)^2 \\ &= h_{22}\left(x^*, 0, 0\right) + h_{33}\left(x^*, 0, 0\right) E_{\varepsilon'}\left(\varepsilon'\right)^2, \end{aligned}$$

[4] When we differentiate the object in (4), we pass the differentiation operator through the expectation operator.

so that we require $h_{22}(x^*, 0, 0)$ and $h_{33}(x^*, 0, 0)$. We obtain $h_{22}(x^*, 0, 0)$ simply by evaluating the second derivative,

$$h_{22}(x, \sigma, \sigma\varepsilon') + 2h_{23}(x, \sigma, \sigma\varepsilon')\varepsilon' + h_{33}(x, \sigma, \sigma\varepsilon')(\varepsilon')^2, \tag{6}$$

at $x = x^*, \sigma = 0$ and $\varepsilon' = 0$. We obtain $h_{22}(x^*, 0, 0) + h_{33}(x^*, 0, 0)$ by taking the average of two versions of (6). In one version, $x = x^*, \sigma = 0$ and $\varepsilon' = 1$ and in the other, $x = x^*, \sigma = 0$ and $\varepsilon' = -1$. With this information we can compute

$$R_{\sigma\sigma}(k^*, 0, 0, 0; g).$$

In this object, $g_{\sigma\sigma}$ (conditional on previously computed coefficients in g, for example, the terms in the linear expansion) is the only unknown.

Our analysis shows that to evaluate the expectation in (2) only requires the variance of ε' when applying second order perturbation. We do not need to know the whole distribution of ε'.