Some Issues in 2nd Approximation Using Perturbation

Xiangyang Li*

March 16, 2016

1 A Simple Example: Neoclassical Growth Model

1.1 Theory

Let's briefly cover the topics of Perturbation Methodology. Suppose there is a vector x that represents some state variables which may includes exogenous and predetermined endogenous variables. The FOCs of a model is defined by

$$f(x, y) = 0, \text{for all } x$$

And let's the solution defined by

$$y \equiv g(x)$$

which satisfying the so-called Error function

$$E(x; g) \equiv f(x, g(x)) = 0 \text{ for all } x$$

Finding an analytic form for g is almost impossible except for some simple and special cases. Here approximation is used which require finding an approximated function $\hat{g}$ which is very close to g for all x. But actually, we only

*The Note is still incomplete and preliminary. Typos and Errors may still prevail. This is the first draft and improvements are expected. Any comments and suggestions are welcomed! Please send to ahnulxy#qq.com (Replace # with @ when email). This series notes dedicate to help students and researchers to understand more about Dynare and DSGE models. Special thanks goes to Prof. Christiano for his inspiring lectures and notes.

can not ensure $\hat{g}$ is close to g all the time. Most often, only in a small range. Perturbation typically involves Taylor expansions which based on some fixed points. Here these fixed points are usually the steady states of the variables of the system. This small range usually a small range include the steady states.

But how can we find $\hat{g}$? It is a polynomial function.

1. First, we need calculate the steady states of the model.

2. Then we need to know to what order g is approximated? The higher the approximation order, the higher the accuracy is. But in what order we have to approximate? For problem not involving risk and welfare, first order usually enough. If you want risk to matter in your model and If you want to study the welfare problem, then 2nd approximation is needed. If you want risk premiums to be cyclical, you need at least 3rd order.

3. Finally, we need to find the coefficients of this polynomial function. Now let's see how can we find the coefficients of the polynomial function $\hat{g}$:

Let's take the ith order derivative at the Error function

$$E^{(i)}(x;g) \equiv \frac{\mathrm{d}^i}{\mathrm{d}x^i} E(x;g) = 0 \text{ for all i and all } x$$

then evaluate the result at the $x^\star$:

$$E^{(1)}(x^\star) = \frac{\mathrm{d}}{\mathrm{d}x} f(x, g(x))|_{x=x^\star} = f_1(x^\star, g^\star) + f_2(x^\star, g^\star) g'(x^\star) = 0$$

we have

$$g'(x^\star) = -\frac{f_1(x^\star, g^\star)}{f_2(x^\star, g^\star)}$$

And let's us continue:

$$\begin{aligned} E^{(2)}(x^\star) &= \frac{\mathrm{d}^2}{\mathrm{d}x^2} f(x, g(x))|_{x=x^\star} \\ 0 &= f_{11}(x^\star, g^\star) + 2f_{12}(x^\star, g^\star) g'(x^\star) + f_{22}(x^\star, g^\star) (g'(x^\star))^2 + f_2(x^\star, g^\star) g''(x^\star) \end{aligned}$$

Then we can solve $g''(x^\star)$ linearly using $g'(x^\star)$. Let's continue this process until we find:

$$g'(x^\star), g''(x^\star), \cdots, g^{(n)}(x^\star)$$

Then we have the following Taylor's series approximation (expansion) :

$$g(x) \approx \hat{g}(x)$$

$$\hat{g}(x) = g^{\star} + g'(x^{\star})(x - x^{\star}) + \frac{1}{2}g''(x^{\star})(x - x^{\star})^2 + \cdots + \frac{1}{n!}g^{(n)}(x^{\star})(x - x^{\star})^n$$

This process could be very painful and extremely time-consuming if x is highly dimension-ed vector. Of course, we assume that there are enough differentiability and appropriate invertibility.

Even if x has two entries, it could cost you half a hour or even one hour to manually find out $g''(x^{\star})$ by managing huge numbers of notations, and you could easily run into some mistakes and ends up with some wrong results. Fortunately, there are Matlab codes that will do this for you, such as the codes by Schmitt-Grohé and Uribe(2004).

Though Schmitt-Grohé and Uribe(2004) has already gives us a detailed guidance to find $\hat{g}$ and Matlab codes also presented to help understanding and we also cover this topic in the first series of DSGE video, we are going to give a further explanation and using a very simple example to help you understand the ideas in this classic paper.

1.2 The Model

This is a toy example to show the basic ideas which including functional form characterization of the model solution. The social planner problem

$$\max_{c_t, k_{t+1}} E_0 \sum_{t=0}^{\infty} \beta^t u(c_t), \quad u(c_t) = \frac{c_t^{1-\gamma} - 1}{1 - \gamma}$$

s.t.

$$c_t + \exp(k_{t+1}) \leq f(k_t, a_t), \; t = 0, 1, 2, ..$$

$$f(k_t, a_t) = \exp(\alpha k_t) \exp(a_t) + (1 - \delta) \exp(k_t)$$

wehre $K_t = \exp(k_t)$, $a_t = \rho a_{t-1} + \epsilon_t$, $\epsilon_t \sim$ iid, $E(\epsilon_t) = 0$, $\text{Var}(\epsilon_t) = V_{\epsilon}$[1].

[1]In Schmitt-Grohé and Uribe(2004), exogenous shocks are set as the form $a_t = \rho a_{t-1} + \sigma\epsilon_t$, where $\text{Var}(\epsilon_t)$=1. If we view $\sigma\epsilon_t$ as a overall shock, then σ has an explanation of variance of this overall shock.

We introduce a parameter σ before the shock. The FOCs produces the efficiency condition

$$E_t\left(u'\left(c_t\right)-\beta u'\left(c_{t+1}\right)\overbrace{f_K\left(k_{t+1},\rho a_t+\sigma\epsilon_{t+1}\right)}^{\text{period t+1 marginal product of capital}}\right)=0$$

or

$$E_t\left[u'\left(f\left(k_t,a_t\right)-\exp\left(k_{t+1}\right)\right)\right.$$
$$\left.-\beta u'\left(f\left(k_{t+1},\rho a_t+\sigma\epsilon_{t+1}\right)-\exp\left(k_{t+2}\right)\right)f_K\left(k_{t+1},\rho a_t+\sigma\epsilon_{t+1}\right)\right]=0\,(1)$$

where σ is a parameter which index a set of model. For the model we are discussing corresponds to $\sigma=1$. One of the reason why we use this σ parameter is that we want to set it to zero where steady states are calculated.

Let's assume that policy rule

$$k_{t+1}=g\left(k_t,a_t,\sigma\right)$$

and then efficiency condition can be written as

$$E\left(k_t,a_t,\sigma;g\right)=$$
$$E_t\left[u'\left(f\left(k_t,a_t\right)-\exp\left(g\left(k_t,a_t,\sigma\right)\right)\right)\right.$$
$$-\beta u'\left(f\left(g\left(k_t,a_t,\sigma\right),\rho a_t+\sigma\epsilon_{t+1}\right)-\exp\left(g\left(\overbrace{g\left(k_t,a_t,\sigma\right)}^{k_{t+1}},\rho a_t+\sigma\epsilon_{t+1},\sigma\right)\right)\right)$$
$$\left.\times f_K\left(g\left(k_t,a_t,\sigma\right),\rho a_t+\sigma\epsilon_{t+1}\right)\right]=0$$

for all k_t,a_t,σ.

Let's find the steady states of the model. First the capital stock $k^\star$.

$$k^\star=g\left(k^\star,a^\star,0\right)=g\left(k^\star,0,0\right)\overset{\text{def}}{\equiv}g^\star$$

In steady states, log technology has steady state value zero and there is no uncertainty with $\sigma=0$;P

$$k^\star=\log\left(\frac{\alpha\beta}{1-\left(1-\delta\right)\beta}\right)^{\frac{1}{1-\alpha}}$$

The 2nd order Taylor series expansion of the policy function

$$g\left(k_t, a_t, \sigma\right) \approxeq$$
$$\overbrace{k^\star + g_k\left(k_t - k^\star\right) + g_a a_t + g_\sigma \sigma}^{\text{first order terms}}$$
$$\overbrace{+\frac{1}{2}\left(g_{kk}\left(k_t - k^\star\right)^2 + g_{aa}a_t^2 + g_{\sigma\sigma}\sigma^2\right) + g_{ka}\left(k_t - k^\star\right)a_t + g_{a\sigma}a_t\sigma + g_{k\sigma}\left(k_t - k^\star\right)\sigma + \cdots}^{\text{2nd order terms}}$$

PThe certainty equivalence implies that $g_\sigma, g_{ka}, g_{a\sigma}$ all equal to zero all the time. Now we are going to find out what are the rest coefficients in above Taylor series expansion.

In Dynare, the output will be a little bit different from above. Dynare has replaced a_t with $\rho a_{t-1} + \epsilon_t$. Hence, there will be a_{t-1} and ϵ_t terms in the final policy rule.

1.2.1 1st Order

Let's find out what are the coefficients of the linear terms: g_k, g_a. Now taking first order derivative of the error function w.r.t. k_t, a_t, σ and evaluating at $k_t = k^\star, a_t = 0, \sigma = 0$:

$$E_k\left(k_t, a_t, \sigma; g\right) = E_a\left(k_t, a_t, \sigma; g\right) = E_\sigma\left(k_t, a_t, \sigma; g\right) = 0$$

which implies

$$E_k = u'' \times \left(f_k - e^g g_k\right) - \beta u' f_{Kk} g_k - \beta u'' \times \left(f_k g_k - e^g g_k^2\right) f_K = 0$$

$$E_a = -\beta u''\left(f_k g_a + f_a \rho - e^g\left(g_k g_a + g_a \rho\right)\right) f_K + u''\left(f_a - e^g g_a\right) - \beta u'\left(f_{Kk} g_a + f_{Ka}\rho\right) = 0$$

$$\begin{aligned} E_\sigma &= -u'' e^g g_\sigma + \beta u'' f_K\left(f_k - e^g\left(g_k + 1\right)\right) g_\sigma - \beta u' f_{Kk} g_\sigma = 0 \\ &= -\underbrace{\left(u'' e^g - \beta u'' f_K\left(f_k - e^g\left(g_k + 1\right)\right) + \beta u' f_{Kk}\right)}_{\text{source of certainty equivalence}} g_\sigma = 0 \end{aligned}$$

where absence of arguments in $u', u'', g, g_k, g_a, g_\sigma, f_k, f_K, f_{Ka}, f_{Kk}$ reflect that they are evaluated at the steady states of the arguments. It is clearly shown

that g_σ is the root of linear and homogeneous equations (see p761, Schmitt-Grohé and Uribe,2004,JEDC), hence, if unique solution exists, then it must be the case: $g_\sigma = 0$. As we assumed before,

$$f(K_t, A_t) = A_t K_t^\alpha + (1-\delta) K_t$$

or

$$f(k_t, a_t) = \exp(a_t) \exp(\alpha k_t) + (1-\delta) \exp(k_t)$$

Taking derivative w.r.t. K_t,k_t and evaluate at $K = \exp(k^\star)$, $A = \exp(a) = 1$

$$f_K = \alpha K^{\alpha-1} \exp(a) + (1-\delta), K = \exp(k^\star)$$

$$f_k = \alpha \exp(\alpha k^\star + a) + (1-\delta) \exp(k^\star) = f_K e^g$$

$$f_{Kk} = \alpha(\alpha-1) \exp((\alpha-1) k^\star + a)$$

$$f_{KK} = \alpha(\alpha-1) K^{\alpha-2} \exp(a) = f_{Kk} e^{-g}$$

By eq(1) and evaluate at the steady states

$$\beta f_K = 1$$

Then,

$$E_t(k_t, a_t, \sigma; g) = 0$$

can be reduced to

$$\frac{1}{\beta} - \left(1 + \frac{1}{\beta} + \frac{u'}{u''} \frac{f_{Kk}}{e^g f_K}\right) g_k + g_k^2 = 0$$

Hence, there are two solutions for g_k: $0 < g_k < 1, g_k > \frac{1}{\beta}$. Then Stokey-Lucas Theory[2] tell us to pick the small one. In some cases, we could have the two solutions less than unity. At this moment, we could have multiple solutions.

Then we solve g_a conditional on g_k and $E_a = 0$:

$$g_a = \frac{u'' f_a - \beta u' f_{Ka} \rho - \beta u'' f_a \rho f_K}{u'' e^g + \beta u' f_{Kk} + \beta u'' (f_k - (\rho + g_k) e^g) f_K}$$

[2]Stokey, N. L. and J. Robert E. Lucas, et al. (1989). Recursive Methods in Economic Dynamic, Harvard University Press.

1.2.2 2nd Order

Now we have found g_k, g_a, g_σ. If we want to find 2nd solutions, we need to find g_{kk}, g_{ka}, g_{aa} conditional on

$$E_{kk}\left(k_t, a_t, \sigma; g\right) = E_{ka}\left(k_t, a_t, \sigma; g\right) = E_{aa}\left(k_t, a_t, \sigma; g\right) = 0$$

for all k_t, a_t, σ. We are not going to detail on this point because it will cost too much time and energy. The notations involve in 2nd approximation will simply unendurable when using pencil and paper to derive. We will left this for Dynare.

In what follows, we will manually code in Matlab to find out g_k, g_a. And then use Dynare to find both the 1st order and 2nd order coefficients. And we will compare the two results.

1.2.3 Parameterization

In order to get further understanding on the solutions, let's calibrate the structural parameters in the model which are taken from Prescott(1986).

$$\beta = 0.99, \gamma = 2(20), \alpha = 0.36, \delta = 0.02, \rho = 0.95, V_\epsilon = 0.01^2$$

Using above analysis and conditional on above calibration, we can program in Matlab to find out g_k, g_a and then we compare this two values to what found in Dynare. Here is the Matlab codes[3]:

```
%2015-10-14@Beijing
%written by Xiangyang  Li

beta = 0.99;
gamma = 2; %20
alpha = 0.36;
delta =0.02;
rho =0.95;
Veps = 0.01^2;

Kbar = (alpha*beta/(1-(1-delta)*beta))^(1/(1-alpha));
kbar = log(Kbar);
```

[3]See first_order_coefficients_calcu.m

```
%find g_k, reads as g sub k;
ca = 1;  % coefficient of quad. equation for gk;
as = 0 ; %steady state of technology
fbar = exp(alpha*kbar+as)+(1-delta)*exp(kbar);
cbar = fbar - exp(kbar);

%the 1st and 2nd derivatives of utility
uprime = cbar^(-gamma);
udprime = -gamma*cbar^(-1-gamma);

fK = alpha*Kbar^(alpha-1)*exp(as)+(1-delta);
fk = Kbar*fK;
fKk= alpha*(alpha-1)*exp((alpha-1)*kbar+as);
cb = -(1+1/beta + uprime/udprime*fKk/fk);
cc = 1/beta;

gk1 = (-cb + sqrt(cb^2 - 4*ca*cc))/2/ca;
gk2 = (-cb - sqrt(cb^2 - 4*ca*cc))/2/ca;
if gk1 >1
    gk = gk2;
else
    gk =gk1;
end

%find g_a conditional on g_k and E_a = 0;
fa = fbar - (1-delta)*exp(kbar);
fKa = fK - (1-delta);
ga1 = udprime*fa-beta*uprime*fKa*rho-beta*udprime*fa*rho*fK;
ga2 = udprime*exp(kbar)+beta*uprime*fKk+beta*udprime...
          *(fk - exp(kbar)*(rho+gk))*fK;
ga = ga1/ga2;

%display
gk
ga
```

The code will produces

$$g_k = 0.9801, g_a = 0.0631$$

by setting $\sigma = 1$ which correspond to the original model we are interested in.

1.2.4 Dynare Result

Let's attach the Dynare code here first[4]:

```
//Written by Xiangyang@BJ,2015-10
//the log-level model of the CGG;
var c k a f;
varexo eps_a;

parameters gamma alpha delta beta rho Veps;
parameters cs ks as fs;

beta = 0.99;
gamma = 2; //20
alpha = 0.36;
delta =0.02;
rho =0.95;
Veps = 0.01^2;
ks = (alpha*beta/(1-(1-delta)*beta))^(1/(1-alpha));
as = 0;
fs = ks^alpha+(1-delta)*ks;
cs  = fs - ks;

model;
//(1) resource constraint
exp(c) + exp(k) = exp(f);

//(2) the production technology
exp(f) = exp(a)*exp(alpha*k(-1)) + (1-delta)*exp(k(-1));

//(3) the Euler equation
```

[4]See cgg_level.mod in the code directory.

```
beta*exp(-gamma*c(+1))*(alpha*exp((alpha-1)*k)
*exp(a(+1))+(1-delta))=exp(-gamma*c);

//(4) the technology motion
a = rho*a(-1) + eps_a;
end;

initval;
a = as;
f = log(fs);
k = log(ks);
c = log(cs);
end;

shocks;
var eps_a = Veps;
end;

//solve and simulate the model, set qz_zero_threshold=1e-15 if gamma=20
stoch_simul(order =2,nograph,qz_zero_threshold=1e-15);
```

Here is the result from Dynare:

```
POLICY AND TRANSITION FUNCTIONS
                        c                k                a                f
Constant         1.122302         3.877427                0         3.939087
(correction)     -0.000189         0.000012                0                0
k(-1)            0.470841         0.980149                0         0.949688
a(-1)            0.306741         0.059920         0.950000         0.074682
eps_a             0.322886         0.063074         1.000000         0.078613
k(-1),k(-1)       0.014439         0.007102                0         0.014834
a(-1),k(-1)      -0.094121        -0.033335                0        -0.044039
a(-1),a(-1)       0.039994         0.030399                0         0.032685
eps_a,eps_a       0.044314         0.033683                0         0.036217
k(-1),eps_a      -0.099074        -0.035089                0        -0.046357
a(-1),eps_a       0.084197         0.063997                0         0.068811
```

By looking at the 2nd column on k_t, we could see that the coefficient on k_{t-1} is the same as the one we calculate: $g_k = 0.9801$. As we mentioned before, Dynare has replaced a_t with $\rho a_{t-1} + \epsilon_t$. From the output, we could write the policy function as:

$$\begin{aligned}\hat{g}\left(k_t, a_{t-1}, \epsilon_t, \sigma\right) &= \overbrace{k^\star}^{3.8774} + \overbrace{g_k}^{0.9801(0.996)}\left(k_t - k^\star\right) + \overbrace{\rho g_a}^{0.0599()} a_{t-1} + \overbrace{g_\sigma}^{0} \sigma + \overbrace{g_a}^{0.0631()} \sigma\epsilon_t \\ &\frac{1}{2}\left(\overbrace{g_{kk}}^{0.0142(0.00017)}\left(k_t - k^\star\right)^2 + \overbrace{\rho^2 g_{aa}}^{0.0608()} a_{t-1}^2 + \overbrace{g_{\sigma\sigma}}^{0.000024()} \sigma^2 + \overbrace{g_{aa}}^{0.0674()} \sigma^2\epsilon_t^2\right) \\ &+ \overbrace{\rho g_{ka}}^{-0.033335()}\left(k_t - k^\star\right) a_{t-1} + \overbrace{\rho g_{a\sigma}}^{0} a_{t-1}\sigma + \overbrace{g_{k\sigma}}^{0}\left(k_t - k^\star\right)\sigma \\ &+ \overbrace{g_{ka}}^{-0.0351()}\left(k_t - k^\star\right)\sigma\epsilon_t + \overbrace{\rho g_{aa}}^{0.0640()} a_{t-1}\sigma\epsilon_t\end{aligned}$$

If we collect terms and substitute out $\rho a_{t-1} + \sigma\epsilon_t$ with a_t, then we could have:

$$\begin{aligned}\hat{g}\left(k_t, a_t, \sigma\right) &= \overbrace{k^\star}^{3.8774} + \overbrace{g_k}^{0.9801()}\left(k_t - k^\star\right) + \overbrace{g_a}^{0.0631(0.07)} a_t + \overbrace{g_\sigma}^{0}\sigma \\ &+\frac{1}{2}\left(\overbrace{g_{kk}}^{0.0142()}\left(k_t - k^\star\right)^2 + \overbrace{g_{aa}}^{0.0674(0.079)} a_t^2 + \overbrace{g_{\sigma\sigma}}^{0.000024(0.00068)} \sigma^2\right) \\ &+ \overbrace{g_{ka}}^{-0.0351(-0.028)}\left(k_t - k^\star\right) a_t + \overbrace{g_{a\sigma}}^{0} a_t\sigma + \overbrace{g_{k\sigma}}^{0}\left(k_t - k^\star\right)\sigma\end{aligned} \tag{2}$$

This will let's see that Dynare has produced the same result $g_a = 0.0631$ as we have done manually. Note that the value in parenthesis refer to the case when $\gamma = 20$ while the benchmark value is 2. I intentionally left most of them blank and you can fill it out to help you understand more[5].

Special note to $g_{\sigma\sigma}$ which is stored in oo_.dr.ghs2. In this case, $g_{\sigma\sigma}$ is the 2nd entry in oo_.dr.ghs2 since the capital stock is the No.2 in the DR order.

Once you have the 2nd approximation, you can do simulation, calculate the IRF and so on.

[5]Actually, I have already filled it for you. You could easily get the results by simply change the value of the parameter $\gamma = 20$ in the mod file.

2 Shift effect of the variance of future shocks: An illustration

In the above analysis, we have a 2nd approximation to the policy rule, let repeat the result here again

$$\begin{aligned}\hat{g}\left(k_t, a_t, \sigma\right) &= \overbrace{k^\star}^{3.8774} + \overbrace{g_k}^{0.9801}\left(k_t - k^\star\right) + \overbrace{g_a}^{0.0631} a_t + \overbrace{g_\sigma}^{0}\sigma \\ &+ \frac{1}{2}\left(\overbrace{g_{kk}}^{0.0142}\left(k_t - k^\star\right)^2 + \overbrace{g_{aa}}^{0.0674} a_t^2 + \overbrace{g_{\sigma\sigma}}^{0.000024}\sigma^2\right) \\ &+ \overbrace{g_{ka}}^{-0.0351}\left(k_t - k^\star\right) a_t + \overbrace{g_{a\sigma}}^{0} a_t\sigma + \overbrace{g_{k\sigma}}^{0}\left(k_t - k^\star\right)\sigma\end{aligned}$$

In the 2nd approximation method, Dynare Reference Manual gives

$$y_t = 0.5\Delta^2 + Ey_{t-1}^h + Fu_t + 0.5G\left(y_{t-1}^h \otimes y_{t-1}^h\right) + 0.5H\left(u_t \otimes u_t\right) + I\left(y_{t-1}^h \otimes u_t\right)$$

where Δ^2 is the shift effect of the variance of future shocks which is stored in *oo_.dr.ghs2*. Matrix G is stored in *oo_.dr.ghxx*. H is stored in *oo_.dr.ghuu*. I is stored in *oo_.dr.ghxu*. Matrix rows order corresponds to DR order.

Hence, we could come to the conclusion that $\Delta^2 = g_{\sigma\sigma}$. In this example, $\Delta^2 = 0.000024$ which is very small. If there is uncertainty, i.e., $\sigma \neq 0$, there will be a shift effect of risk. That is to say that people will counter risk by increasing capital stock for the next period.

In practice, $g_{\sigma\sigma}$ is very small comparing to other coefficients in the approximation and we usually ignore it for simplicity.

3 Naïve Simulation And Its Limitations

Once we have the approximation solution, we can do simulations. Simulation is useful since artificial data simulated from the solution can be used to compute various moments which can be compared with analogous statistics in data. This comparison will be helpful when you evaluate the fitness of the model to the data.

And the computation of impulse responses also requires simulations when the solution is nonlinear especially in 2nd approximation.

Let's see how simulation carries out in Dynare.

3.1 In Theory

In theory, if we have initial states, for example, k_0, a_0, ϵ_1 are given[6], then at time 1,

$$k_1 = g\left(k_0, a_0, \epsilon_1, \sigma\right)$$

Recursively, at time 2,

$$k_2 = g\left(k_1, a_1, \epsilon_2, \sigma\right) = g\left(\overbrace{g\left(k_0, a_0, \epsilon_1, \sigma\right)}^{k_2}, \overbrace{\rho a_0 + \sigma\epsilon_1}^{a_1}, \epsilon_2, \sigma\right)$$

at time 3,

$$k_3 = g\left(k_2, a_2, \epsilon_3, \sigma\right) = g\left(\overbrace{g\left(g\left(k_0, a_0, \epsilon_1\right), \rho a_0 + \epsilon_1, \epsilon_2, \sigma\right)}^{k_2}, \overbrace{\rho^2 a_0 + \rho\sigma\epsilon_1 + \sigma\epsilon_2}^{a_2}, \epsilon_3, \sigma\right)$$

This process can continue to any periods you want. And in this way, you simulate your model by drawing shocks and calculate new states k_{t+1} using policy function. And then controls, like c_t, could be obtained using this new states. Everything seems to be wonderful and perfect. In literature, it is the so-called naïve simulation.

If you do not have a functional form for g, then everything are just illusions. That is to say that this recursion can not carry out at all.

3.2 In Practice

In practice, approximation solution is used. Here we study the case where a 2nd order approximation solution applied since we do not have exact policy rule g. Just consider a 2nd order approximation of the mapping from $k_t, a_t, \epsilon_{t+1}, \sigma$ to k_{t+2}:

$$k_{t+2} = g\left(g\left(k_t, a_t, \epsilon_{t+1}, \sigma\right), \rho a_t + \sigma\epsilon_{t+1}, \epsilon_{t+2}, \sigma\right)$$

If we use the 2nd order approximation solution to recursively compute the new states, then problems arise. Let's have a close examination.In period

[6]The shocks are actually drawn at beginning of each period and hence known during the whole period. For example, ϵ_1 is drawn at the beginning of time 1 and known at whole period.

$t+1$, y_{t+1} can be written as

$$\begin{aligned} y_{t+1} &= g_k y_t + g_a a_t \\ &\quad + \frac{1}{2}\left(g_{kk} \overbrace{y_t^2}^{\text{problematic term in simulation}} + g_{aa} a_t^2 + g_{\sigma\sigma}\sigma^2 \right) \\ &\quad + g_{ka} y_t a_t + g_{a\sigma} a_t \sigma + g_{k\sigma} y_t \sigma \end{aligned} \tag{3}$$

$$\begin{aligned} y_{t+2} &= g_k y_{t+1} + g_a a_{t+1} \\ &\quad + \frac{1}{2}\left(g_{kk} \overbrace{y_{t+1}^2}^{\text{problematic term in simulation}} + g_{aa} a_{t+1}^2 + g_{\sigma\sigma}\sigma^2 \right) \\ &\quad + g_{ka} y_{t+1} a_{t+1} + g_{a\sigma} a_{t+1} \sigma + g_{k\sigma} y_{t+1} \sigma \end{aligned} \tag{4}$$

where $y_t = k_t - k^\star$. The square term in above expression will cause serious stationary problem in simulation. You could see this in next section.

Substitute out y_{t+1} in y_{t+2} using simplified y_{t+1}:

$$\begin{aligned} y_{t+2} &= g_k^2 y_t + \left(g_k g_a + \rho g_a\right) a_t + g_a \sigma \epsilon_{t+1} \\ &\quad + \frac{1}{2}\left(\left(g_k g_{kk} + g_{kk} g_k^2\right) y_t^2 + \left(\rho^2 g_{aa} + g_{kk} g_a^2 + g_k g_{aa} + \cdots\right) a_t^2 + \cdots \right) \\ &\quad + \frac{1}{4} g_{kk}^3 y_t^4 + \cdots \end{aligned}$$

This iteration process will be very painful and could have serious stationary problem. And as you can see that it is no longer 2nd order approximation since there are higher order terms appear.

There is another way around. Just to take 2nd order approximation of the following equation and evaluate at $k_t = k^\star, a_t = 0, \sigma = 0$. And this will also lead the same result as above[7].

$$k_{t+2} = g\left(g\left(k_t, a_t, \epsilon_{t+1}, \sigma\right), \rho a_t + \sigma \epsilon_{t+1}, \epsilon_{t+2}, \sigma\right)$$

[7]The two ways are basically the same thing.

$$
\begin{aligned}
\partial k_t: \quad & g_k\left(g\left(k_t, a_t, \epsilon_{t+1}, \sigma\right), \rho a_t + \sigma\epsilon_{t+1}, \epsilon_{t+2}, \sigma\right) g_k\left(k_t, a_t, \epsilon_{t+1}, \sigma\right) = g_k^2 \\
\partial a_t: \quad & g_k\left(g\left(k_t, a_t, \epsilon_{t+1}, \sigma\right), \rho a_t + \sigma\epsilon_{t+1}, \epsilon_{t+2}, \sigma\right) g_a\left(k_t, a_t, \epsilon_{t+1}, \sigma\right) \\
& + \rho g_a\left(g\left(k_t, a_t, \epsilon_{t+1}, \sigma\right), \rho a_t + \sigma\epsilon_{t+1}, \epsilon_{t+2}, \sigma\right) = g_k g_a + \rho g_a \\
\partial \epsilon_{t+1}: \quad & \sigma g_a\left(g\left(k_t, a_t, \epsilon_{t+1}, \sigma\right), \rho a_t + \sigma\epsilon_{t+1}, \epsilon_{t+2}, \sigma\right) = g_a \sigma \\
\ldots &
\end{aligned}
$$

We are not going to detail on it. You can do it by yourself for exercise.

This iteration process will also be painful and could have stationary problem too. Let's use an example to illustrate the potential problem in naive simulation in the next section so that we can introduce pruning.

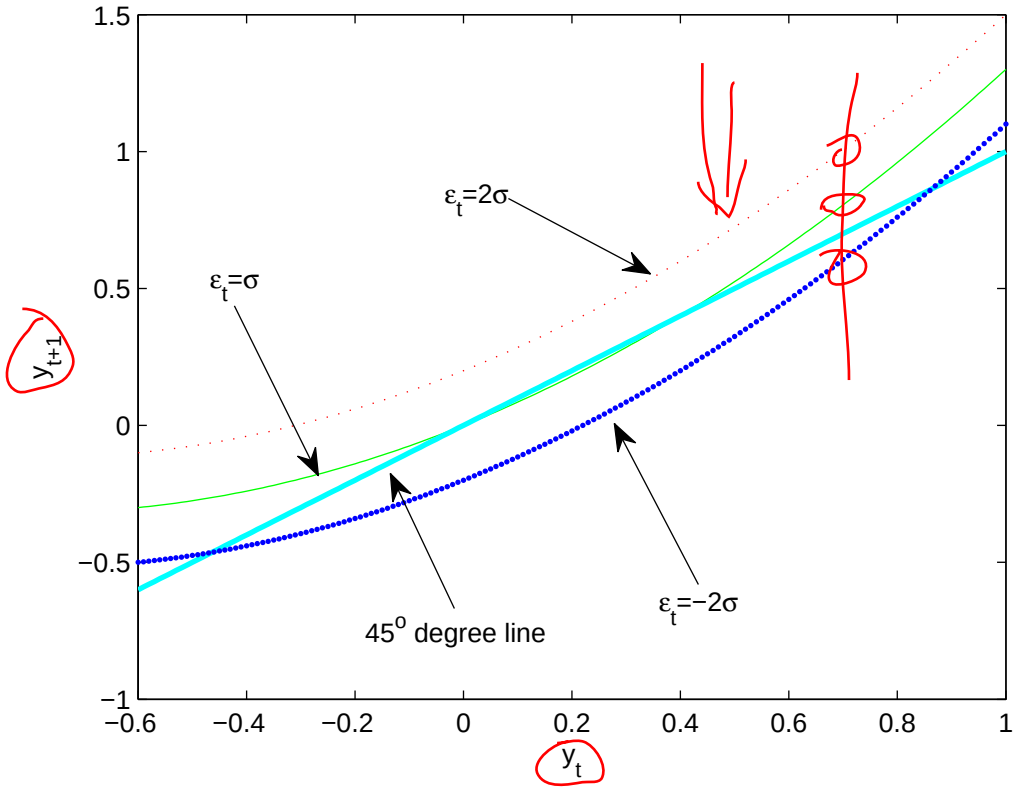
4 Pruning

4.1 An Example

Let's study a simplified version of 2nd order approximation to neoclassical model solution:

$$y_t = \rho y_{t-1} + \alpha y_{t-1}^2 + \epsilon_t, \ \rho = 0.8, \alpha = 0.5, \sigma = 0.10$$

Figure 1: A Simple Example of Pruning and Simulation



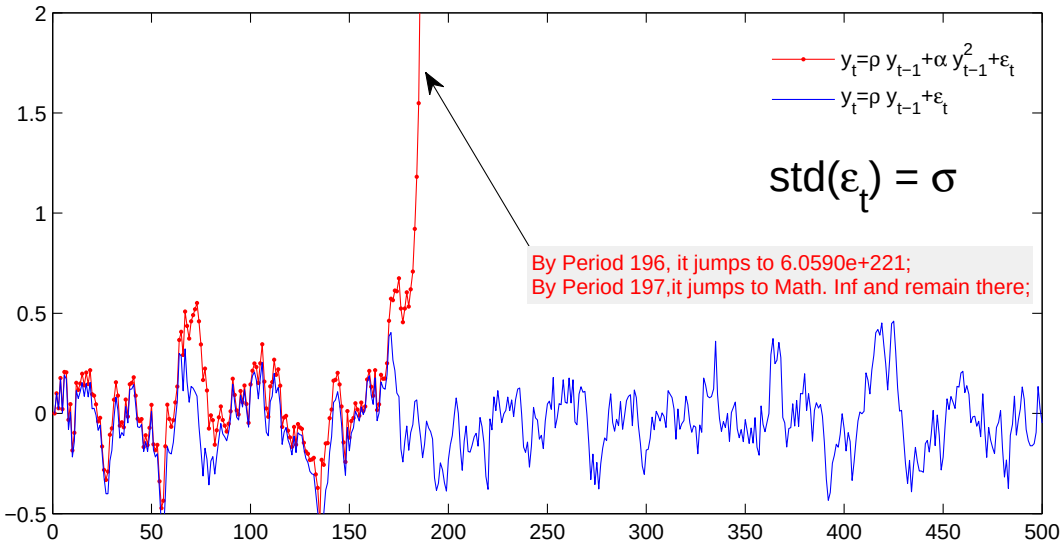
Note: The process will explode after several positive shocks and will explode fast since its convexity.

As a comparison, we will plot another time series as follows. For small and large shocks, we will separately plot.

$$y_t = \rho y_{t-1} + \epsilon_t$$

As we can clearly see from figure 2, if there are square term of lags, then the series could easily goes to infinity for a relatively large shock. For small shocks, it seems there are no problems arise. If we have square terms, but we still want to do simulations. Then What should we do if we face large shocks? At this time, Pruning technology come into play. There are possible steps for simulation under this circumstances:

Figure 2: Simulation with large shocks



1. Draw a sequence of shock from given distribution: $\epsilon_1, \epsilon_2, \cdots, \epsilon_N$; usually, the shocks are white noises (normal random variables).

2. We need to get rid of the square term y_{t-1}^2 in $y_t = \rho y_{t-1} + \alpha y_{t-1}^2 + \epsilon_t$. If we simply remove this term, this could lead to huge bias in simulation. Hence we should replace ('prune') y_{t-1} with something not far away from it. The natural choice is to replace y_{t-1} with[8]

$$\tilde{y}_{t-1} = \rho \tilde{y}_{t-2} + \epsilon_{t-1}.$$

$\tilde{y}_t$ is stationary by construction, hence

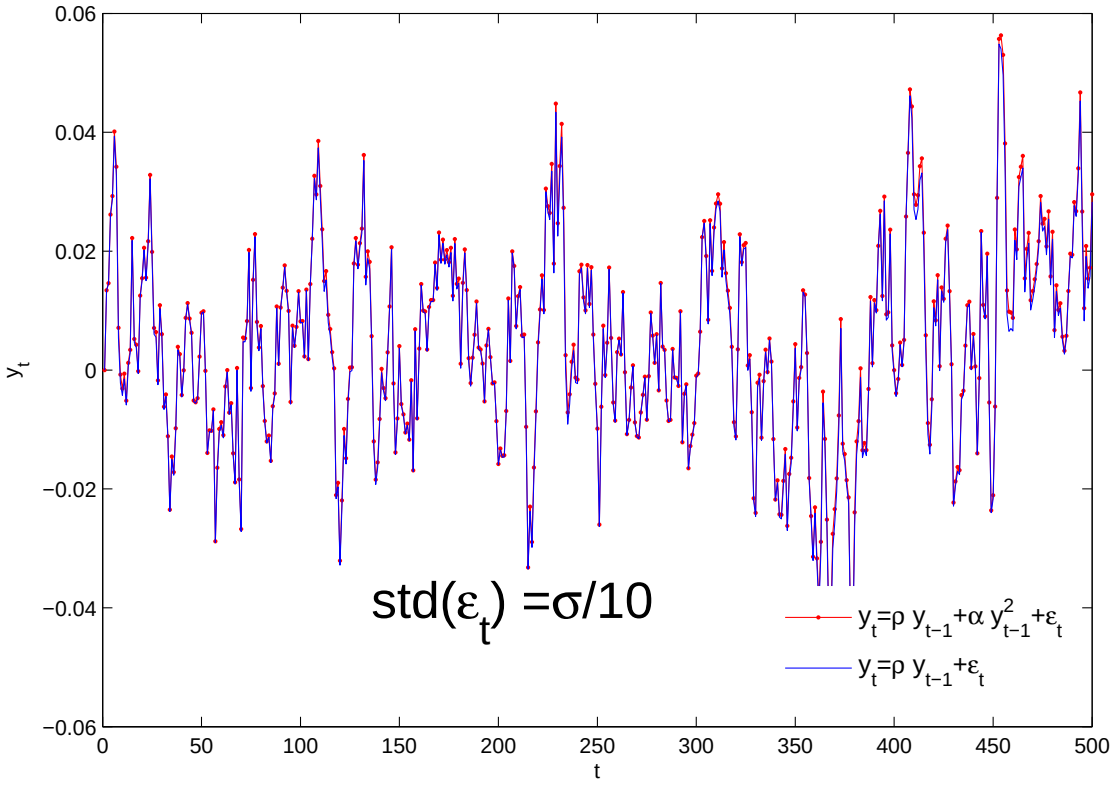
$$y_t = \rho y_{t-1} + \alpha \tilde{y}_{t-1}^2 + \epsilon_t$$

must be stationary too. This y_t is so-called 'pruned' solution for original equation $y_t = \rho y_{t-1} + \alpha y_{t-1}^2 + \epsilon_t$ since you have pruned the unwanted part of the equation to make the simulation could be successfully carried out. But there are problems. This simulation is no longer the original one

[8] where the real value for y_{t-1} should follow the process $\tilde{y}_{t-1} = \rho \tilde{y}_{t-2} + \alpha \tilde{y}_{t-2}^2 + \epsilon_{t-1}$. Here we throw away ('prune') the squre terms.

since you have remove the exact square term and use an approximation term to replace it. And inevitably, there are some differences between simulated data using pruning technology and real data from original series. Hence pruning technology should be used with care.

Figure 3: Simulation with small shocks



The two lines are virtually the same for small shocks.

4.2 What is Pruning and Why we need it?

The original meaning of the 'prune' is that to weed out unwanted or unnecessary things. Here we use pruning technology to help us remove non-stationary problem.

As you can see in last section, problems arise if there are square terms in the equation when we do simulation. For a 2nd approximation solution to policy function, similar problems arise if we want to do simulation. So we use pruning technology. We have a 2nd order approximation solution here,

$$y_{t+1} = g_k y_t + g_a a_t + \frac{1}{2}\left(g_{kk} y_t^2 + g_{aa} a_t^2 + g_{\sigma\sigma}\sigma^2\right) + g_{ka} y_t a_t$$

Then we prune y_t^2 and use $\tilde{y}_t^2$ to replace it.

$$\tilde{y}_{t+1} = g_k \tilde{y}_t + g_a a_t$$

$$a_t = \rho a_{t-1} + \epsilon_t$$

You are effectively pruning high order terms in iteration which just like a gardener who removes diseased or unwanted parts of plants in his garden.

In Dynare, you can set options pruning in stoch_simul or estimation command to instruct Dynare to do a pruning simulation for you. If you have large shock and you do not set this option on, you could get some strange results.

5 Spurious Steady states

Naive simulation will not only cause stationarity problems, but also cause spurious steady states problems too. The 2nd order approximation to the policy function has a second steady state which is redundant in neoclassical growth model. We call this redundant steady state spurious steady state.

Let's consider a 2nd order approximation solution here

$$y_{t+1} = g_k y_t + g_a a_t + \frac{1}{2}\left(g_{kk} y_t^2 + g_{aa} a_t^2 + g_{\sigma\sigma}\sigma^2\right) + g_{ka} y_t a_t$$

Setting $a_t = 0$ and ignoring shift effect of future shocks since it is very small comparing to other coefficients in the approximation, then the solution becomes

$$y_{t+1} = g_k y_t + \frac{1}{2} g_{kk} y_t^2$$

This equation has two steady states. One is the desired one $y_t = 0$.

In the neoclassical growth model, let's focus on the capital stock k_t. In steady state, one solution is $k_t = k^*$. Another one is

$$y_t = \frac{2(1-g_k)}{g_{kk}} = \frac{2(1-0.98)}{0.014} = 2.86$$

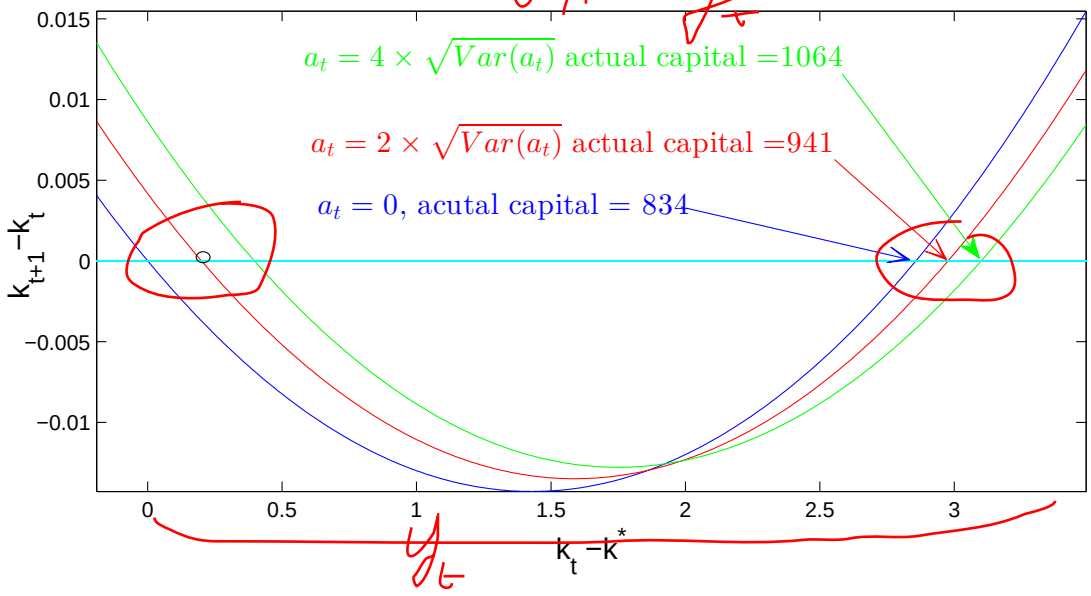
using the example in section 1.2. This means that

$$k_t = k^\star + 2.86 \approx 3.87 + 2.86 = 6.73$$

Then the level capital stock is $K = \exp(6.73) = 837.15$(after rounding) using the definition that $k - k^* = \log K/K^*$. This value is too large in this neoclassical model may not worth worrying about. But it could be close to real steady states in large DSGE models. Hence, it may cause some confusions and unexpected consequences.

Let's plot $k_t - k^*$ against $k_{t+1} - k_t$ under different technology shocks. Using Eq.(2) and do not ignore the shift effect of future shocks, then we have:

Figure 4: Spurious Steady States Under different Technology Shocks



Here it the Matlab code to reproduce Figure 4:

```
%% spurious steady state, 2015-10-27@BJ
%Written by Xiangyang Li.
%2nd approximation under gamma=2
ks = 3.8774;
kt = -0.2: 0.01:3.5; %deviation from steady states;
yt = zeros(length(kt),1)';
xt = zeros(length(kt),1)';
yt(1) = kt(1);
gk = 0.98;
ga = 0.063;
gkk = 0.014;
gaa= 0.067;
gss = 0.000024;
gka = -0.035;
Vara = 0.0320^2; %var(a) = Veps/(1-rho^2); rho = 0.95; Veps =0.01^2;
at = [0,2*sqrt(Vara),4*sqrt(Vara)];
for jj=1:length(at)
    for ii=2:length(kt)
        yt(ii)= gk*kt(ii-1)+ga*at(jj)+...
        0.5*(gkk*kt(ii-1)^2+gaa*at(jj)^2+gss)+gka*kt(ii-1)*at(jj);
        xt(ii)=yt(ii) - kt(ii-1);
    end
    switch jj
        case 1
            plot(kt(2:end-1),xt(3:end));
        case 2
           plot(kt(2:end-1),xt(3:end),'r');
        case 3
            plot(kt(2:end-1),xt(3:end),'g');
    end
    hold on;
end
plot(kt(2:end-1),zeros(length(kt(2:end-1))),'cyan');
hold off;
axis tight;
```