Fiscal policy in a simple growth model – A resolution with the "shooting algorithm"

Thomas Bourany

Term paper – Computational Economics – Sciences Po

May 11, 2016

1 Introduction

In this term paper for the course of computational economics, I present a program coded in Julia langage, where I implement a simple growth model with exogenous fiscal policy, with a method called "shooting algorithm"

The model in question is a neoclassical growth-model, which is non-stochastic, with an exogenous government. The government is purchasing exogenously a stream of goods, financing itself with distortive tax (consumption tax, capital tax) and lump-sum tax to equalize its budget constraint. The particularities of this economy are: an inelastic labor supply, a Cobb-Douglas production function, a CRRA utility function. There is no saving but perfect Ricardian equivalence due to capital. The government is exogenous, i.e. it is determined ex-ante and agents take the policy as given.

This program implement a method called "shooting algorithm". This method is detailed in the chapter 11 of the book of Ljungqvist & Sargent, *Recursive macroeconomic theory.* The intuition of this method is simple: the program chooses arbitrarily an initial condition for consumption and deduce the path of consumption and capital, thanks to the equations of the model. Once the path computed, it compares the final value obtained for capital and its steady-state value that would prevail under the "Augmented-Golden Rule". If it is not the case, the program modify, up or down, the initial condition, and with this set of "trail-and-errors", the algorithm compute the definitive path of capital and consumption. The other endogenous variables follows from the capital path. The advantage of this algorithm is that it chooses a sequence of endogenous variables taking into account the exogenous path of government policy. Even though the exogenous variable may vary at each period, this algorithm take into account these shocks. Therefore, unlike in Dynare, this program might implement path of government policies, not only a single shock. Moreover, at first I found this setting relatively easy to understand and flexible compared to other methods (Value Function Iteration). However, the implementation revealed later some issues in terms of convergence and accuracy.

In the first part of this paper I present the model and its main equations. I describe the method and the algorithm coded in a second part. There, I also provide details about the issues I faced concerning the convergence of the algorithm. The third section show the "policy experiments" I implemented. The fourth and final section propose extensions and concluding remarks.

2 Model

There is no uncertainty in this model, and agents take decision with perfect foresight. Household maximizes their utility function over time, consuming a single consumption good.

$$\sum_{t=0}^{\infty} \beta^t u(c_t) \qquad u(c_t) = \frac{c_t^{1-\sigma} - 1}{1-\sigma}$$

The utility – the standard CRRA utility function – is strictly increasing and concave in c_t and β is the discount factor equal to $\frac{1}{1+\rho}$. The constant relative aversion is σ. The agents consume an intermediary good x_t that can also be consumed by government g_t or used as capital k_t, which depreciate at rate δ. The intermediary good is produced following a production function $f(k_t)$ which is a simple Cobb-Douglas, with α the share of capital as input (we suppose the labor supply is inelastic and equal to 1). Therefore:

$$g_t + c_t + x_t \leq f(k_t) \qquad k_{t+1} = (1-\delta)k_t + x_t$$

The equilibrium determines a price system, i.e. $\{q_t, \eta_t, w_t\}_{t=0}^{\infty}$ respectively the pre-tax price of the intermediary good, the rental price of capital (or profit in the renting from households to firms), and the wage. The household and the government are respectively facing the following budget constraints:

$$\sum_{t=0}^{\infty} q_t\{(1+\tau_{ct})c_t + (k_{t+1}-(1-\delta)k_t)\} \leq \sum_{t=0}^{\infty} q_t\{\eta_t k_t - \tau_{kt}(\eta_t-\delta)k_t + (1-\tau_{nt})w_t n_t - \tau_{ht}\}$$
$$\sum_{t=0}^{\infty} q_t g_t \leq \sum_{t=0}^{\infty} q_t\{\tau_{ct}c_t + \tau_{kt}(\eta_t-\delta) + \tau_{nt}w_t n_t + \tau_{ht}\}$$

A competitive equilibrium that solve this model is a price system $\{q_t, \eta_t, w_t\}_{t=0}^{\infty}$, a budget-feasible government policy $\{g_t, \tau_{ct}, \tau_{kt}, \tau_{ht}\}_{t=0}^{\infty}$ and an allocation $\{c_t, k_{t+1}\}_{t=0}^{\infty}$ that solve the set of following equations.

2.1 Key equations of the model

- A law of motion of capital determined by the market clearing condition (11.6.1)

$$k_{t+1} = f(k_t) + (1-\delta)k_t - g_t - c_t \tag{1}$$

- The *Euler equation* (11.6.3)

$$u'(c_t) = u'(c_{t+1})\frac{(1+\tau_{c,t})}{(1+\tau_{c,t+1})}[(1-\tau_{k,t+1})(f'(k_{t+1})-\delta)+1] \tag{2}$$

- The steady state relation, called the *Augmented Golden Rule* (11.6.7)

$$\delta + \frac{\rho}{(1-\tau_k)} = f'(\bar{k}) \tag{3}$$

These three equations are computed in three functions in the Julia-file *FiscalPolicyModel.jl*, respectively *lawofmotion_k()* (line 163) *eulerequation()* (line 175) and *steadystate()* (line 146).

2.2 Other endogenous equilibrium quantities

For a initial condition c_0 and capital path $\{k\}_{t=0}^{\infty}$ respecting the three equations above, we can compute the rest of the endogenous variables, with the following set of 6 equations. First, c_t the consumption (but computed simultaneously by the algorithm as we will seen later), q_t the price of the intermediary goods, η_t the rental price of capital, w_t the wage of the worker, R_t the gross interest-rate and r_t the net interest rate.

$$
\begin{aligned}
c_t &= f(k_t) + (1-\delta)k_t - k_{t+1} - g_t \\
q_t &= \beta^t u'(c_t)/(1+\tau_{c,t}) \\
\eta_t &= f'(k_t) \\
w_t &= f(k_t) + k_t f'(k_t) \\
R_{t,t+1} &= \{(1-\tau_{k,t+1})(f'(k_{t+1}-\delta)+1\} \\
r_{t,t+1} &= R_{t,t+1} - 1
\end{aligned}
$$

This set of equations is computed in *FiscalPolicyModel.jl* in the function *equilibriumquantities()* (line 188)

3 Program in Julia Langage

3.1 Few words on the program

The method implemented for this model is not really advanced from a numerical point of view (no numerical integration, no approximation of functions, etc.). However, I tried to enforce the algorithm as efficiently as possible. I created new types for variables *Exovar*, *Endovar*, collecting the variables cited above (4 exogenous and 7 endogenous variables). I created a new type *Model*, that gathered the functions (utility, production and their derivatives), a dictionary containing parameters and the variables, via the types *Exovar* and *Endovar*. Therefore, if one want to improve the model, for example in adding a new variables (labor supply or productivity), implement a two country model (as I plan to do in the future), or simply change the functional form of the functions and their parameters, then these new features should not take long to code, thank to the new types and the different functions written in separate modules [1]. This method provides a extensive set for "policy experiments", and my aim was to replace "Dynare" software for the implementation of this kind of model.

[1]These points may appear like details, but I tried to go as far as I could in this direction

3.2 The shooting algorithm

From Ljungqvist & Sargent, the method to solve this model is the following :

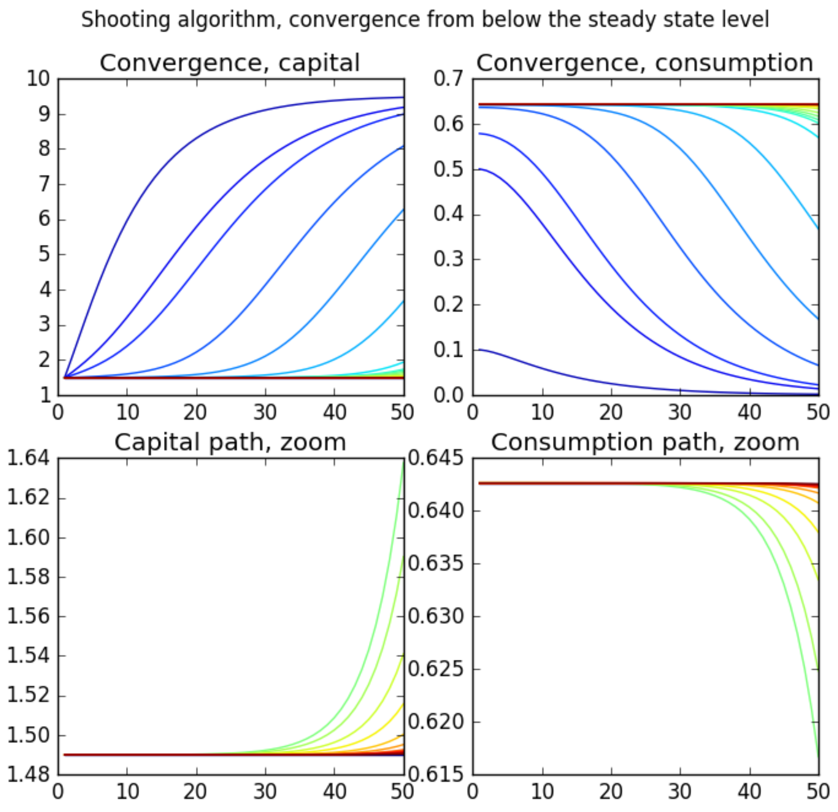
- Solve (11.6.4) for the terminal steady-state $\bar{k}$ that is associated with the permanent policy vector $z = (g, \tau_c, \tau_k)$ i.e., find the solution of (11.6.7), i.e. the *Augmented Golden Rule Equation* 3
- Select a large time index $S >> T$ and guess an initial consumption rate c_0 . Compute $u'(c_0)$ and solve (11.6.1), i.e. *the law of motion of capital* 1 for k_1 .
- For $t = 0$, use (11.6.3) i.e. *Euler equation* 2, to solve for $u'(c_{t+1})$. Then invert u' and compute c_{t+1} . Use (11.6.1) to compute k_{t+2}.
- Iterate on step 3 to compute candidate values $\hat{k_t}$, for $t = 1, ..., S$
- Compute $k_S - \bar{k}$.
- If $k_S > \bar{k}$, raise c_0 and compute a new $\hat{k_t}$, $t = 1, ..., S$
- If $k_S < \bar{k}$, lower c_0
- In this way, search for a value of c_0 that makes $k_S \approx \bar{k}$
- Compute the lump-sum taxes that satisfies the government budget constraint at equality.

3.3 Implementation of the algorithm, convergence and accuracy

The description given above suggests a "trial-and-errors" strategy: choose the right initial condition c_0 targetting the "shoot" of the path of capital $\{k\}_{t=0}^{S}$ close to the steady-state value $\bar{k}$. Therefore, the first intuition encourages the implementation of a *while*-loop, in order to find the right c_0. Following the description above, "while" $k_S > \bar{k}$, raise c_0, and conversely. However, this method is not really accurate and highly inefficient. For example, with a while-loop, the algorithm needs more than 200'000 replications of the model (i.e. the full path of $\{k\}_{t=0}^{S}$ of say a 50-periods model) to get a level k_S close to the true $\bar{k}$ at 10^{-4}.This issue arises because the criteria of the loop – the level $\bar{k}$ – is distinct from the instrument c_0. Furthermore, the behavior of the path $\{k\}_{t=0}^{S}$ is highly unpredictable when c_0 is far from the efficient level (as we will see in the following graphs). We can even say that the function I implemented in this manner was not a "contraction mapping" (in c_0 and k_S) because if you choose a c_0 just above the efficient threshold, the path of capital can largely deviate from $\bar{k}$.

This difference when one chooses an initial condition $\hat{c_0}$ below or above the "efficient" c_0 and the inefficiency of the while-loop in this context requires a different solution. A simple method – seen in class in different contexts – is to draw a grid of points (here a "grid" of $\hat{c_0}$ equally) and compute for these values the path $\{k\}_{t=0}^{S}$. Once the 1000 value of k_S computed, we can select the $\hat{c_0}$ minimizing the distance $\bar{k} - k_S$. To improve even more the accuracy, we take the value $\hat{c_0} = c_1$ as a lower bound for a new grid of 1000 values of c_1. Again, you can choose the best value $\hat{c_1}$ for the optimal path of capital, and draw a grid c_2 etc. I implemented this method with 4 grids, each one at a smaller level: 1e-3, 1e-6, 1e-9 and 1e-12. Therefore, with "only" 4000 replications of the model, we can choose c_0 in order to obtain a very small gap $\bar{k} - k_S$. In most of our experiments, the difference is smaller than 1e-7. Finally, the algorithm is rather faster, as it take less than 20 second to compute the optimal with this accuracy.

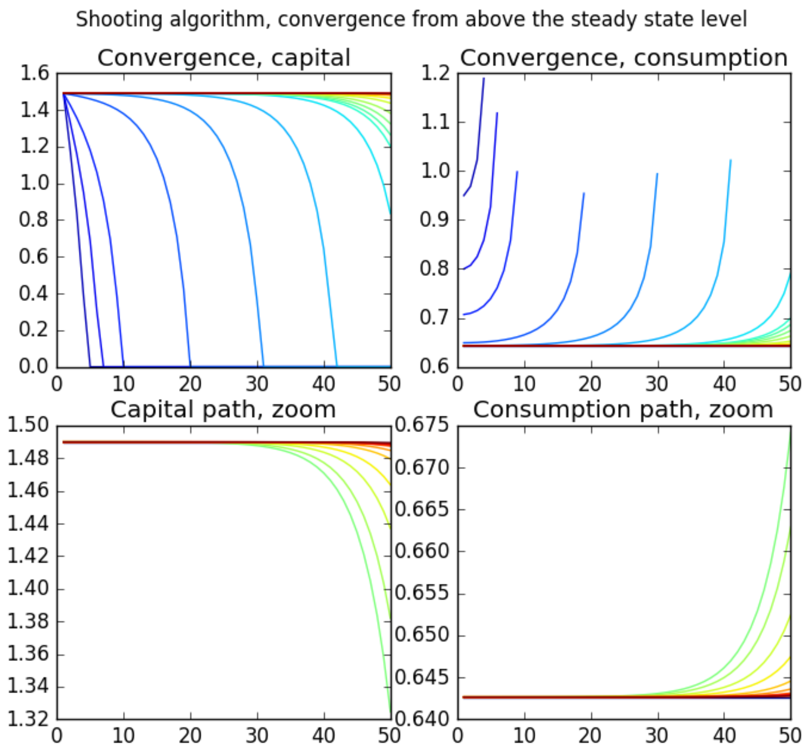
Figure 1: Convergence of the algorithm, starting from a initial condition for c_0 below the steady-state level.



In my code, the function *shootingalgorithm()* (line 288) implement this method with the 4 grids, and this function call another function *loop_ c_ k()* (line 241/265) that compute the path of capital and consumption for an initial value c_0 and the steady state value $\bar{k}$

As seen in figure 1 and figure 2, the path of consumption and capital can be very different when one starts from a "too-low" or "too-high" value of the initial c_0. As shown in figure 1, if c_0 is too small, the consumption goes very fast to zero and the agents over-accumulate capital stocks. Therefore, the difference $\bar{k} - k_S$ might be great. The hotter-color line is when we increase $\hat{c_0}$. We can also observe a deviation toward the end (this trend shows up whatever the level of the zoom). Therefore, even for very accurate range of initial condition (the green, yellow and orange lines, below 1e-4 of difference between $\hat{c_0}$ and the efficient c_0), the difference of the variables increases exponentially after S period.

Figure 2: Convergence of the algorithm, starting from a initial condition for c_0 below the steady-state level.



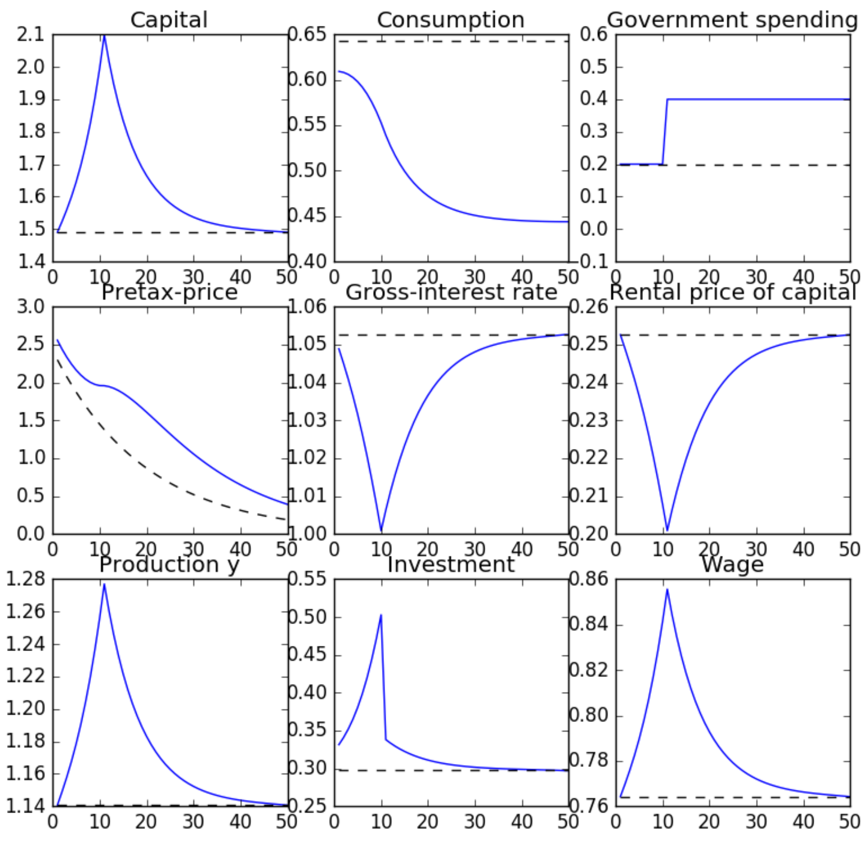
In figure 2, a contrario, I computed the capital and consumption paths, starting from $\hat{c_0}$ above the threshold. There, too-high c_t lowers the marginal utility of consumption, increases the marginal productivity and increases (exponentially) the value of consumption solving the Euler equation. At some point, there is no capital anymore to produce and the model stop (or continue at level zero). We observe the same trend, even if we increase the precision.

The function drawing these graphs are coded line respectively line 35 (figure 1) and line 103 (figure 2). The function *graph1()* and *graph2()* can draw the convergence process of the algorithm for every model of the same kind, including those with "policy shocks". Two examples of these graphs are shown in the folder on the github repository (*graph3_ shock1_ convergence*).

4 Policy experiment

In this part, I compute basic policy experiments, to understand the mechanisms involved. First, an increase in public spending, second a raise in consumption tax, third a raise in capital tax, fourth a non-permanent pulse in spending. These shocks are the most standard policy experiment.

Figure 3: Increase in g from 0.2 to 0.4 at time $t = 10$.



The results are shown in the Impulse-Response function, where I plot successively the capital path, the consumption, the relevant policy variable (the one which is shocked: spending, tax, etc.), and 6 others endogenous variables (including two computed ex-post that I found relevant: the production and investment). The dashed-line correspond to the steady-state value without shock.

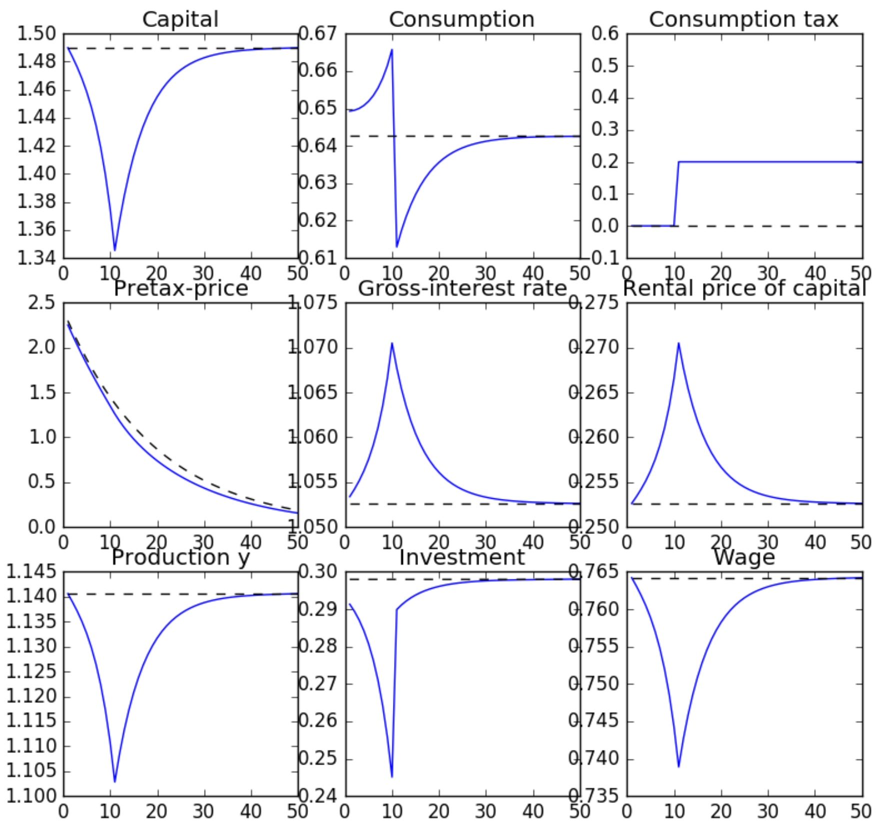
The common feature of all these graphs, as one could expected, is that the policy shocks, foreseen, are anticipated by the variations of capital and consumption.

4.1 A *once-and-for-all* increase in government spending

See previous page.

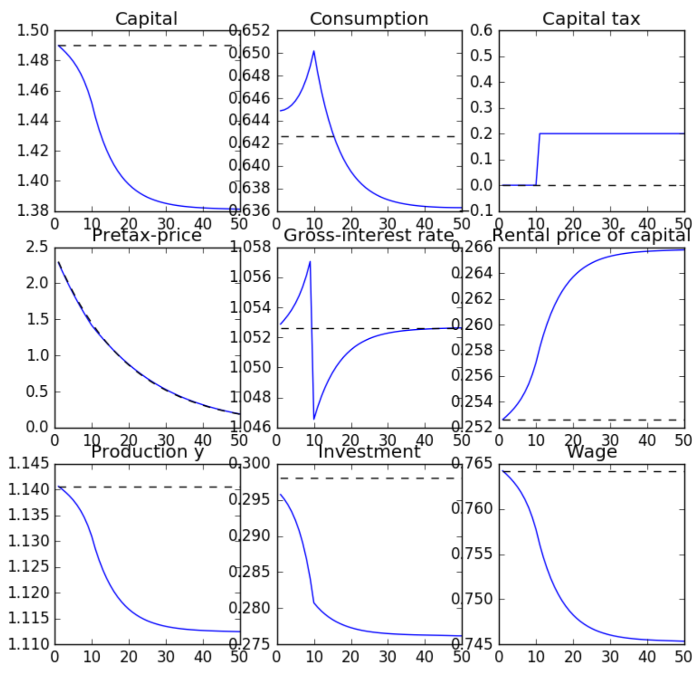
4.2 A *once-and-for-all* increase in consumption tax

Figure 4: Increase in τ_c from 0 to 0.2 at time $t = 10$.



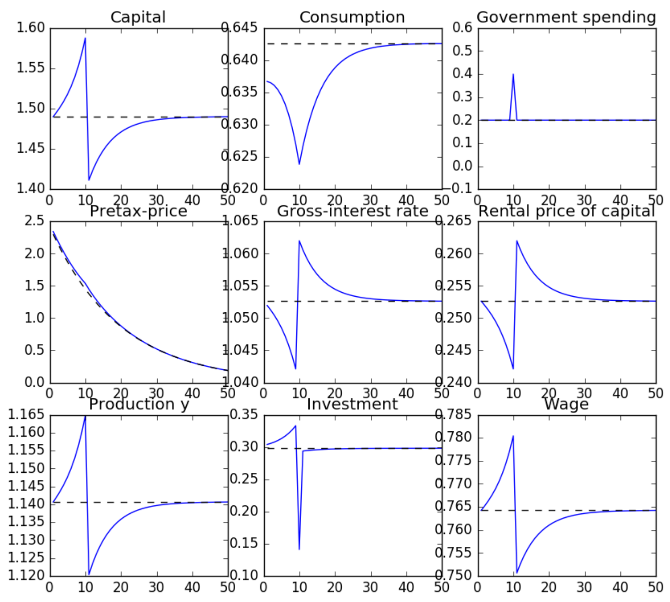
4.3 A *once-and-for-all* increase in capital tax

Figure 5: Increase in τ_k from 0 to 0.2 at time $t = 10$.



4.4 A *one-time pulse* increase in government spending

Figure 6: Temporary increase in g from 0.2 to 0.4 at time $t = 10$.

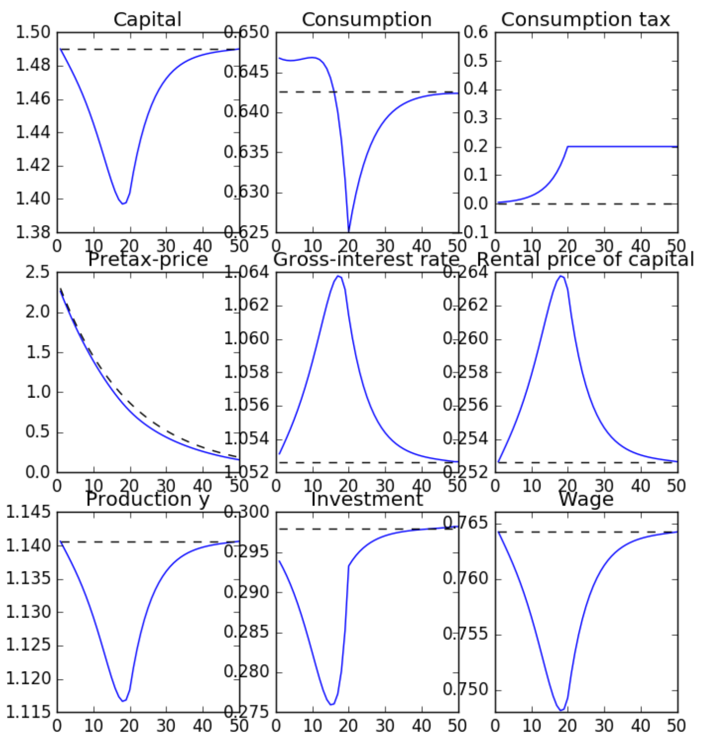


4.5 Other kind of policy experiments

These Impulse response are identical to the IRF computed thanks to Dynare. However, the advantage with this setting, is the ability to draw sequence of shocks from the set of exogenous variables. Therefore, we are not restricted by a "one-time" shock, and the natural transition dynamics of the endogenous variables but we can draw a full sequence of policy shocks. The algorithm will take into account this policy path in the computation of the optimal path for capital (and in the choice of c_0). There are plenty of application to be done. I show few of them in the repository on Github. Here, I only reference the two most striking example, concerning the consumption tax.

If government increases once-and-for-all the consumption tax from 0 to 0.2 at time t=10 (cf. the Figure 4), the consumption response shows first an exponential increase before tax, then a drop at time t=10 and then again an increase until steady state. What I show in the two following graphs is that the response might be very different if one increases the tax progressively, first with a *convex* path and second with a *concave* path.

Figure 7: Increase in $\tau_{c,t}$ from 0 to 0.2 progressively, with a convexe path.

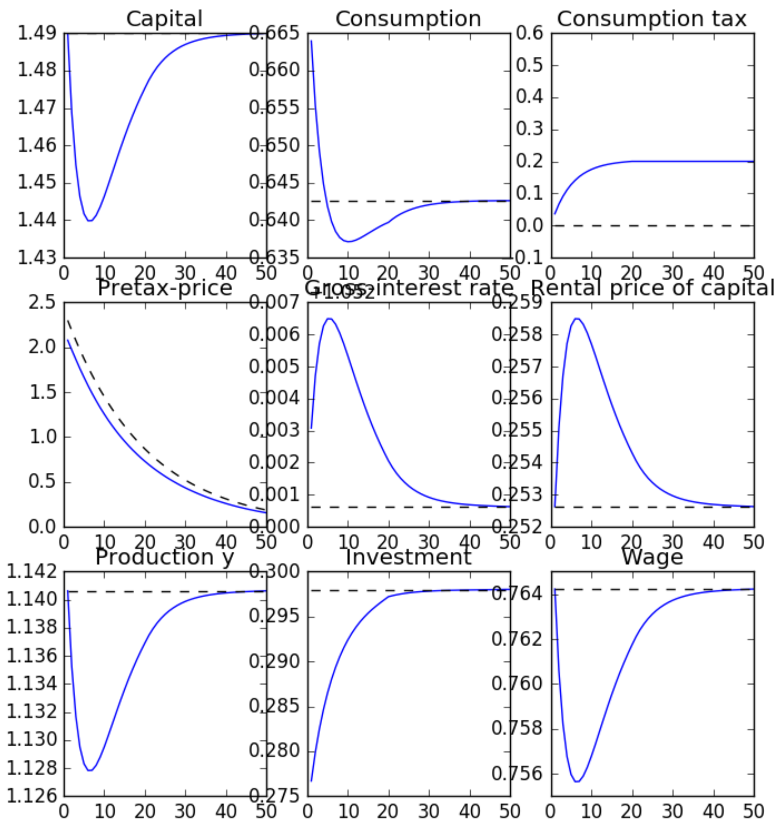


The first – convex – path shows a stable level for 10 periods, and then a drop in consumption around t=20 similar to the Figure 4. This fall in consumption happens when the discounted value of the taxes is too high for the households and that reduces sharply their consumption.

On the contrary, the second path, with a concave increase do not show a sharp drop in consumption but rather a great increase at the beginning of the period, reducing until below the steady-state level, to increase progressively until the steady state. The key feature here is that there is no sharp drop in consumption. More, when we look more closely to the capital path, the decrease in capital is greater in the convex path of consumption tax than in the concave path. These differences lie in the form of the Euler equation, showing the ratio of consumption tax today against tax tomorrow.

From a consumption smoothing and production efficiency point of view, the concave increase of consumption tax is therefore optimal. This conclusion has been easily highlighted in the context of this model with the shooting algorithm.

Figure 8: Increase in $\tau_{c,t}$ from 0 to 0.2 progressively, with a concave path.



5 Proposition of extension and concluding remarks

My aim for future research is to extend this model in different directions. I would like to add productivity and this should be easily implementable as there is only one variable to add in the production. Second, we should integrate the reaction of labor supply (and therefore the labor tax), and integrate it to the set of "key-equations" of the model, especially if the utility is non-separable. Finally, the third possible extension would be to add a second identical country, along with equations like "world market clearing condition". This would allow to observe the spillovers of shocks in one country (especially policy shocks) into the second country.

These three directions can be implemented with the same "shooting algorithm". Concerning this method and as a conclusion, this algorithm is rather accurate. If we want to increase its accuracy, adding a 5th grid in the algorithm at level 10^{-15} for the consumption initial condition c_0 would be easy and improve its accuracy even further. However, the current level of precision is enough to display the policy shocks and the reactions of endogenous variables for stream of policy variables. .

At last, I would like to implement a RBC model, with stochastic shocks (and not a fully deterministic model), and take the fiscal policy not exogenous but endogenous, in the "optimal taxation" framework. This would need different and more advanced methods in computational economics.