Optional Homework 06

ARE 254 Fall 2019

In class, we discussed the L&L Example 6.4.1 growth model with and without an irreversible investment constraint: $f(k) \geq c$.

$$\max_{c(t)} \int_0^T e^{-\rho t} u(c(t)) dt \qquad st \qquad \dot{k} = f(k) - c - mk$$

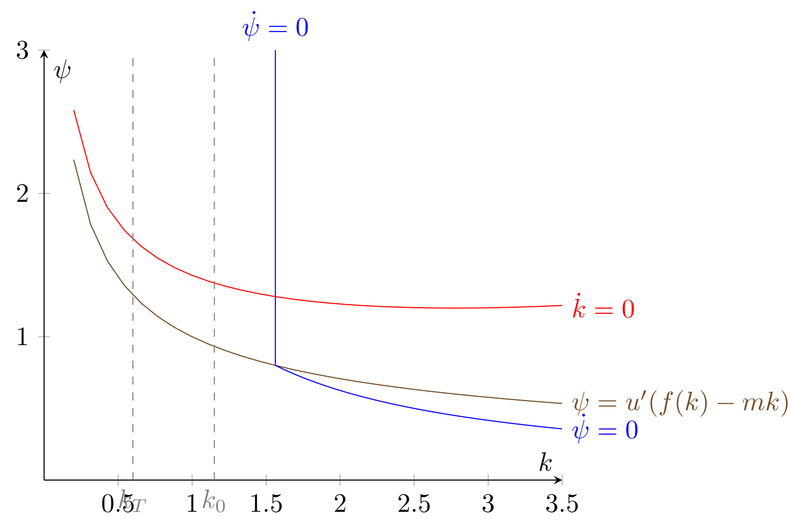
Boundary conditions k_0, k_T, T

Assume $u(c) = \log c$ and $f(k) = k^\alpha$

1. Write down the optimality conditions for the problem with the irreversibile investment constraint. Derive the ODEs that characterize the optimal path
2. Plot phase-plane diagrams in k, ψ (state / co-state) space for the model with and without irreversibility constraints. Indicate steady-states and nullclines. Assume $\alpha = 0.5, \rho = 0.1, m = 0.3$. You might find the tikz/pgfplots code from the lecture notes helpful (see raw markdown.)
3. Numerically solve for the optimal path given $k_0 = 1.15$ and $k_T = 1.15$ in both the constrained and unconstrained cases. Compare the optimal paths depending on whether the constraint binds. Try to find a short value for T such that $c(0)$ increases when you impose the constraint. Now find a long value for T such that $c(0)$ decreases.

   For the constrained case, you'll need to divide your problem into two parts – an unconstrained part for $t \in [0, \hat{t}]$ and a constrained part for $t \in [\hat{t}, T]$. My suggestion is that you characterize the constrained part analytically and compute the value of entering the constrained region—$V(\hat{k}, \hat{t})$. Then make this a scrap value for your unconstrained part. You'll need to find the right TVC for a free $\hat{t}$ and free $\hat{k}$. You should be able to solve the unconstrained part with one of the 3 methods discussed in class (shooting or finite-element collocation with `BoundaryValueDiffEq.jl` or spectral collocation M&F/Judd style).

   Alternatively, you can try using the callback capability of `BoundaryValueDiffEq.jl` to switch the ODEs at the point $\psi(t) = u'(f(k(t)))$. See docs. I'm also posting some draft code from my own research that tries implementing this. Note that it's not elegant, and I make no guarantees that it'll work...



Note

I know it takes a long time for packages to load and precompile when you first start a Jupyter Notebook. You can ease the pain here by creating a Julia *package* that includes a `Project.toml` and `Manifest.toml` file. If you put your Jupyter Notebook in this directory, it will know about all the Packages in the `.toml` files... and, if you `resolve` the package, it will quit precompiling so much. See the `Pkg.jl` docs.

To do this

1. In the REPL, switch to `shell` mode by typing `;` and change to a base directory, `cd [basedir]`
2. In the REPL, switch to `package` mode by typing `]` and generate a new package `generate MyGeniusHwk06`
3. In the REPL, cd into this directory `; cd MyGeniusHwk06`
4. Activate the package in the current directory `] activate .`
5. Add packages to your new package, for example, `] add Plots`

Now create a new Jupyter notebook in this directory. This Jupyter notebook should be aware of all the packages in your project `MyGeniusHwk06`. You can now manage the packages directly from your notebook.

To stop repetitive re/precompiling messages, resolve your project environment before getting going. `] resolve` Do this either in the `MyGeniusHwk06` environment from the REPL, or from your Jupyter noteoobk.

Note: if you need to install packages for someone else's environment, you can *activate the environment and run* `] instantiate`.