COMPUTATIONAL TOOLS FOR THE CAPITAL TAX EXERCISE

The file `ktax.R` contains some R code that could be helpful with the problem set or with further explorations of the model. For the problem actually required, where you are to use the zero-tax steady state that can be computed by hand, you only need to use `g0g1`, giving it `ktaxsys` as an argument. You might find `ktaxsys` and `ktaxss` useful to check that you've computed the steady state correctly, though the computation of the steady state is pretty straightforward. If you want to find other steady states, `ktaxss` would be useful as input to `csolve`, which could search for other steady states. It could then be interesting to see if the dynamics are very different in the neighborhood of other steady states.

`ktaxsys`: This is an expression vector containing the equations of the model as expressions that should be zero in equilibrium. Note that an `eval` of `ktaxsys` itself evaluates one element of the vector after another, returning the final expression's value only. You have to evaluate a particular element explicitly (`eval(ktaxsys[iq]`) or else use `sapply` as is done in the code farther down in the file.

`ktaxss`: This is a function that takes as arguments `ktaxsys` and R lists (with variable names) of values for the proposed steady state, the parameters, and the exogenous variable values in steady state. For our problem, for example, you would set `parlist <- list(alph=.3,bet=.95,delt=.93)` and `xog <- list(a=1,g=0)`. Because the function is set up to be usable as an argument to `csolve`, it requires that its argument `vbar` be not a list but a column vector with `dimnames(vbar)[[1]]=c("cc","k",...)`. The other arguments should be lists or vectors with a `names` attribute.

`g0g1`: This function takes a steady-state variable value list and uses it to compute all the matrices (except the constant vector, which in our case just a 9x1 vector of zeros) needed as input to `gensys`.

For the zero-tax steady state that you can compute by hand, you need only use `g0g1` and `gensys` to get the usual solved system, from which you can get impulse responses as described in the problem description.

An R version of `gensys` is available on my website. It needs to have `qz.R, qzdiv.R`, and `qzswitch.R` available, and those programs are also available at the same site. Note that the R version of `qz.R` uses the `lapack` fortran routine `zgges`. On a Linux computer, the location given in the program for `lapack` is likely to be correct, but it may not be. Most Linux computers have the `lapack` library installed, so you should be able to find it and type in the correct location in the first line of `qz.R` if there are any initial problems.

On a Windows computer I'm not at all sure that it will be possible to invoke `lapack`. The library may be installed on the departmental computers (probably as a `.dll` rather than a `.so` file), in which case you may be able to run `gensys.R` from Windows R by just changing the file location at the start of `qz.R`. Otherwise, you can cut and paste the matrices produced by `g0g1.R` into `gensys.m` in Matlab.

It is possible in principle to do the whole calculation in Matlab. Analytic derivatives with Matlab will require using Matlab's symbolic package. Though I have solved problems like this using Matlab symbolic differentiation, so I know it can be done, I don't have a prepackaged program like `g0g1` written for Matlab. Matlab symbolics are quite different from R's, so the R code will not translate into Matlab code.

Nearly all of you have used either R or Matlab earlier this year in econometrics, and Martin gave some discussion of `gensys` in precept. Nonetheless part (d) of the problem set could be challenging. I will be away Friday, but will check email Friday and on the weekend. Office hours are 2-4 on Thursday as usual.