Problem Set 5

Due on Tuesday, 18, h 13:00. Mail box F230.

Optimization (10 points)

This is a warm-up exercise. When doing GMM or SMM on the computer, you often have to minimize an objective criterion such as

$$J_{TN}(b) = [M_T(x) - M_N(y(b))]' W [M_T(x) - M_N(y(b))]$$

where b is the vector of parameters to be estimated. Unfortunately, nothing ensures that the objective criterion $J_{TN}(b)$ is nicely behaved. In particular, the criterion $J_{TN}(b)$ could have several local minima and in that case the minimization routine (e.g. `fminbnd` or `fminsearch` in Matlab) may stop before the true global minimum is found.
The aim of this exercise is to familiarize you with *global* minimization routines such as the simulated annealing that search over the entire parameter space and thus don't suffer from the shortcomings of *local* routines such as `fminbnd` or `fminsearch`.
Suppose you want to find the global minimum of the very irregular function $f : [-10, 10] \to \mathbb{R}$ depicted in Figure (1). The function shown in the figure is coded in the file `crazy.m`.

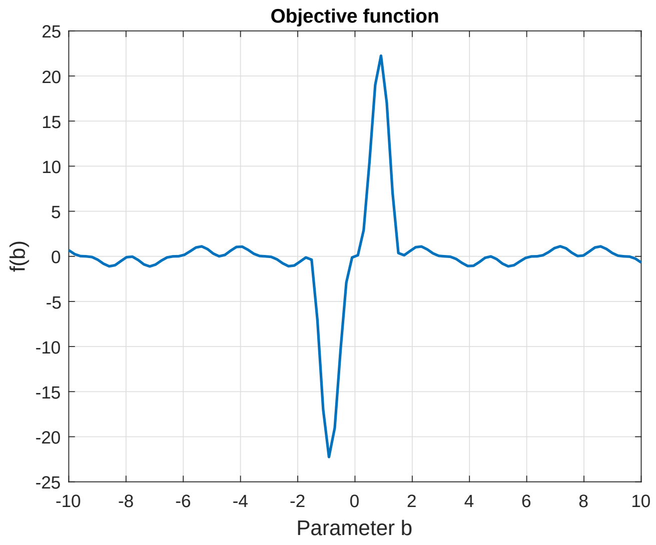

Figure 1: "Crazy" function to be minimized. There are several local extrema, one global minimum and one global maximum.

1. Find the (approximate) global minimum searching over a discrete grid for $b \in [-10, 10]$. What are $b_{\min}$ and $f(b_{\min})$?

2. Try to find the global minimum using `fminsearch`. Choose as initial condition $b_0 = -6$. What is the minimum found by `fminsearch`? Is it the global minimum?
3. Choose as initial condition $b_0 = 4$. What is the minimum found by `fminsearch`? Is it the global minimum?
4. Can you find the global minimum with `fminsearch` by setting the initial condition b_0 very close to the global minimum?
5. Redo the minimization exercise using the simulated annealing. You can rely on the files `example` and `simulan` uploaded on ILIAS. Set $b_0 = -6$ as initial condition. Does the algorithm find the true global minimum?

Suggestion: To report your findings in a clear way, please make a table following this template:

Initial condition	$b_{\min}$ with gridsearch	$b_{\min}$ with fminsearch	$b_{\min}$ with SA
$b_0 = -6$			
$b_0 = 4$			
$b_0 = \dots$			

SMM: Stochastic Growth Model (20 points)

This problem set introduces you to estimating a subset of the deep parameters of a structural model via the simulated method of moments. For simplicity we focus on the stochastic growth model (with *inelastic* labor supply). The planner's problem can be stated as:

$$\max_{C_t, K_{t+1}} E_0 \left\{ \sum_{t=0}^{\infty} \beta^t \log(C_t) \right\}$$

subject to

$$C_t + K_{t+1} = Z_t K_t^{\theta} + (1-\delta) K_t$$

and

$$Z_t = (1-\rho) + \rho Z_{t-1} + \varepsilon_t$$

where ε_t is an i.i.d. random shock drawn from $N(0, \sigma^2)$. Output in the model is defined as $Y_t = Z_t K_t^{\theta}$. In order to keep the estimation exercise simple, we will assume that a subset of the deep structural parameters are set to their calibrated values: $\theta = 0.36$, $\delta = 0.015$,and $\beta = 0.9921$.

The exercise is to match the autocorrelation and standard deviation of real output data (i.e. ρ_y and σ_y in an AR1 of Hodrick-Prescott filtered output data) to the same moments simulated from the model in order to determine the "unobservable" ρ and σ parameters of the technology shock process $\{z_t\}$. Call the data moment vector $M_T^d = (\rho_y, \sigma_y)$. Call the parameter vector $\psi = (\rho, \sigma)$. Call the simulated model moments $M_{TN}^m(\psi) = (\rho_y(\psi), \sigma_y(\psi))$.

1. Assume that $T = 200$ and the sample data moments are $M_T^d = (\rho_y, \sigma_y) = (0.8426, 0.0190)$.

2. Implement the SMM estimation method following these steps[1]:

   (a) For a given parameter vector ψ, calculate the saving decision rule as a function of the state variables and parameter vector using the Blanchard-Kahn method as in Problem Set 2 (or using Dynare if you prefer). That is, solve for $k_{t+1} = g(k_t, z_t; \psi) = \varkappa_k k_t + \varkappa_z z_t$. The decision rules for the other variables of interest, like Y_t, follow easily.

   (b) Using a random number generator, simulate N repetitions of T period samples for the technology shocks (we will call the simulated data z_t^n). Thus we simulate a total of $T \cdot N$ realizations. Save your draw $\{\varepsilon_t^n\}_{t=1,n=1}^{T\ \ N}$. Given the law of motion for technology shocks and the decision rules from step (2a), use the simulated $z_t^n(\psi)$ process to simulate N sample paths of output. Analogous to the data moments, construct the model moments $M_{TN}^n(\psi) = (\sigma_y^n(\psi), \rho_y^n(\psi))$. Finally, take the average of $(\sigma_y^n(\psi), \rho_y^n(\psi))$ across the $n = 1, .., N$ repetitions.

   (c) Solve
   $$\widehat{\psi}_{TN} = \arg\min_{\psi} \left[M_T^d - M_{TN}^m(\psi)\right]' \widehat{W}_T \left[M_T^d - M_{TN}^m(\psi)\right]$$
   where for simplicity we set $\widehat{W}_T = I$ (this will still yield consistent, just not efficient estimates of the true parameter vector). Note that the criterion $\widehat{\psi}_{TN}$ is calculated in the function `smmJ`. In the files uploaded on ILIAS, the minimization routine is `fminsearch`, which is OK since the exercise is very simple and the objective criterion is well behaved. You can try to use `simulan` if you want to practice further.

Note on Step 2(a). The simulated moments (autocorrelation and standard deviation of simulated output) will come from the decision rules for technology, capital and output that you can obtain either with (i) Blanchard-Kahn method or with (ii) Dynare. If you go for option (i), you have to log-linearize the equilibrium equations and then use the function `fun_blanchard_kahn2` that I uploaded on ILIAS. If you go for option (ii), you can skip the log-linearization/matrix form step and write in the mod file the equilibrium equations in their original, non-linear form. To make your life easier, I wrote down in the Appendix the equations that you need in order to solve the model.

Matlab files. You can use as a template the files `mainSMM`, `smmJ` and `smmgenerate`. The parts that need to be filled in by you are denoted with TBC (i.e. to be completed). If you do not plan to use Dynare, you can use also `fun_blanchard_kahn2` to get the decision rules.

[1]Please refer also to Lecture 7, pages 24-27, for additional details.

Appendix

Equilibrium equations: nonlinear. The equilibrium equations are

$$\frac{1}{C_t} = \beta E_t \left\{ \frac{1}{C_{t+1}} \left[\theta Z_{t+1} K_{t+1}^{\theta-1} + 1 - \delta \right] \right\}$$

$$C_t + K_{t+1} = Z_t K_t^{\theta} + (1-\delta) K_t$$

and

$$Z_t = (1-\rho) + \rho Z_{t-1} + \varepsilon_t$$

Steady-state values. The steady states are

$$\overline{K} = \left[\frac{\theta}{\frac{1}{\beta} - (1-\delta)} \right]^{\frac{1}{1-\theta}}$$

$$\begin{aligned} \overline{C} &= \overline{ZK}^{\theta} - \delta \overline{K}, \\ \overline{Y} &= \overline{ZK}^{\theta} \end{aligned}$$

and

$$\overline{Z} = 1.$$

Equilibrium equations: loglinear. Let $x_t \equiv \log X_t - \log \overline{X}$ denote the (approximate) percentage deviation around the steady-state. The equations are:

$$k_{t+1} = \left(\theta \frac{\overline{Y}}{\overline{K}} + 1 - \delta \right) k_t + \frac{\overline{Y}}{\overline{K}} z_t - \frac{\overline{C}}{\overline{K}} c_t$$

$$-E_t c_{t+1} + (1 - \beta(1-\delta)) E_t z_{t+1} - (1-\theta)(1-\beta(1-\delta)) k_{t+1} = -c_t$$

and

$$E_t z_{t+1} = \rho z_t.$$

If you use Dynare you can stop here and enter in your mod file either the nonlinear equations or the loglinear equations.

Matrix form. Write down the equations as $AE_t x_{t+1} = Bx_t + Cz_t$ where $x_t = [k_t, c_t]'$ is the vector with "dynamic" variables. The state variables are $s_t = [k_t, z_t]'$ and the control variable is $u_t = [c_t]$. The matrix form is therefore

$$\underbrace{\begin{bmatrix} ?? & ?? \\ ?? & ?? \end{bmatrix}}_{A} \begin{bmatrix} k_{t+1} \\ E_t c_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} ?? & ?? \\ ?? & ?? \end{bmatrix}}_{B} \begin{bmatrix} k_t \\ c_t \end{bmatrix} + \underbrace{\begin{bmatrix} ?? \\ ?? \end{bmatrix}}_{C} z_t.$$

Once you write down the matrices A, B, C in Matlab (and specifically in `smmgenerate.m`), you can solve the model by calling the function `[PI,W,U]=fun_blanchard_kahn2(A,B,C,rho)`. As usual, the solution will be:

$$\begin{bmatrix} k_{t+1} \\ z_{t+1} \end{bmatrix} = \underbrace{\begin{bmatrix} \varkappa_k & \varkappa_z \\ 0 & \rho \end{bmatrix}}_{\Pi} \begin{bmatrix} k_t \\ z_t \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix}}_{W} \varepsilon_{t+1},$$

$$\widetilde{c}_t = \underbrace{\begin{bmatrix} ?? & ?? \end{bmatrix}}_{U} \begin{bmatrix} \widetilde{k}_t \\ \widetilde{a}_t \end{bmatrix}.$$

Note that the solution for consumption is not relevant for this problem set, since your goal is to simulate time series for output and output depends only on TFP and capital. Once you have the coefficients $\varkappa_k, \varkappa_z$ and ρ you can simulate an artificial time series for TFP, capital and finally model output as:

$$Y_t = Z_t K_t^{\theta-1}$$

and

$$Z_t = z_t + 1$$

(since $z_t = \frac{Z_t - \overline{Z}}{\overline{Z}}$ but $\overline{Z} = 1$). See file `smmgenerate.m` for further clarifications.