Applied Computational Economics
Lab 1
Solving Representative Agent Partial Equilibrium Models

The University of Nottingham

Q1

(a) The state variables are the capital stock and the labour earnings shock. We can then write the Bellman equation as

$$v(k,s) = \max_{c,k'} \ \frac{c^{1-\sigma}}{1-\sigma} + \beta \mathbb{E} v(k',s')$$

where

$$\begin{aligned} c &= (1-\delta+r)k + sw - k' \\ s &\sim G(s|s_-) \end{aligned}$$

where s_- denotes the previous period's value of the labour earnings shock. Notice that I've combined the capital law of motion with the budget constraint.

(b) A quick note about the gridsearch procedure. Let's denote the current guess of the value function in the computational VFI procedure as $v_n(k)$, where note that the s argument is dropped due to the parametric assumptions in this part of the problem. Notice that the code then generates a temporary value function of the following form

$$\tilde{v}(k,k') = \frac{[(1-\delta+r)k + w - k']^{1-\sigma}}{1-\sigma} + \beta v_n(k') \tag{1}$$

where see that the max operator has disappeared. In addition, an extra argument has been added to the left-side of equation 1 for the choice of next period's capital stock. We evaluate 1 for all the possible choices of k' and in the next step of the algorithm, maximise over this variable to yield the value function update $v_{n+1}(k)$ as

$$v_{n+1}(k) = \max_{k'} \ \tilde{v}(k,k').$$

This is the essence of gridsearch — evaluate all possible choices of your control variable, then select the value that gives the largest objective. See figure 1 for the policy and value functions.

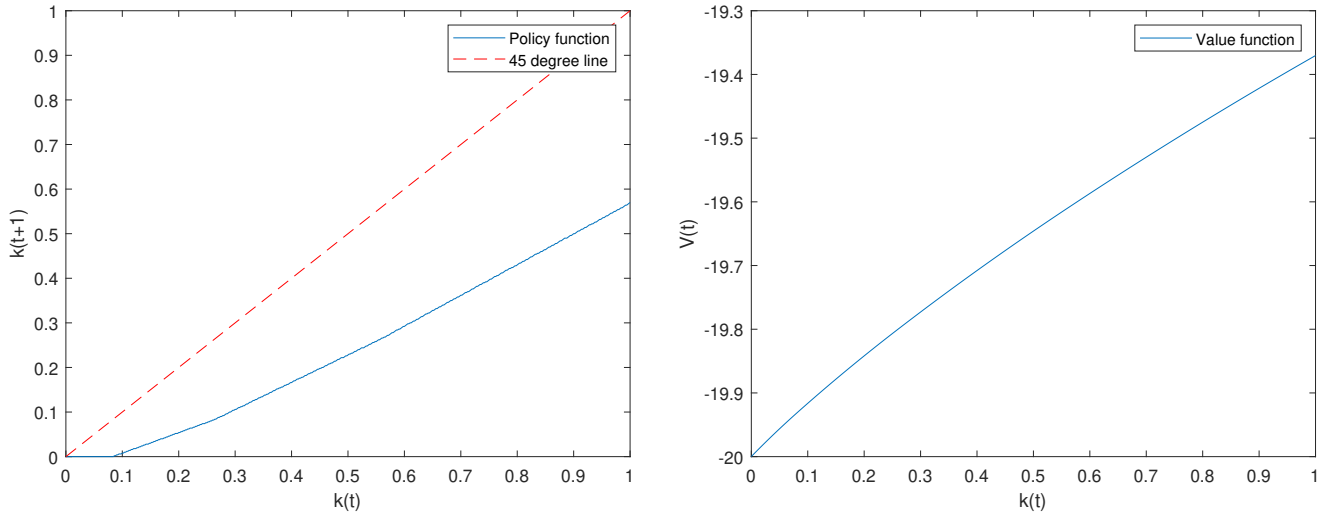

Figure 1: Policy and value functions for part (b)

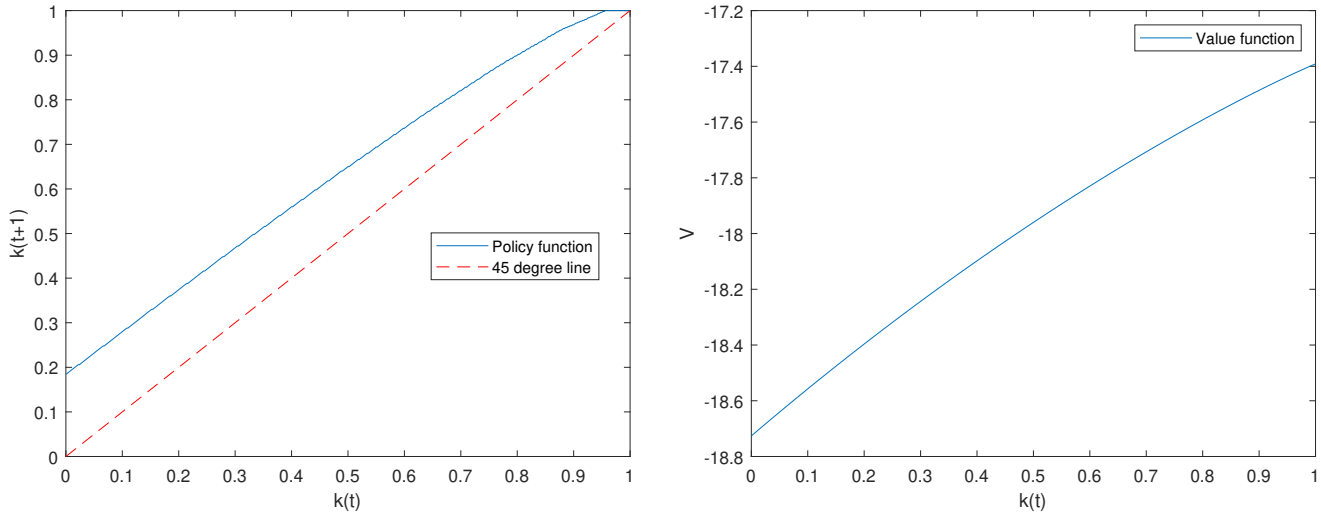

Figure 2: Policy and value functions for part (c)

(c) See figure 2 for the policy and value functions. In the context of part (b), the households were dis-saving for any given level of the capital stock. The opposite is true here: they'll save more than they have at time t since the interest rate is so generous. In fact: you can see from the policy function for (c) that the interest rate is so generous that they actually hit the upper-bound for the capital stock.

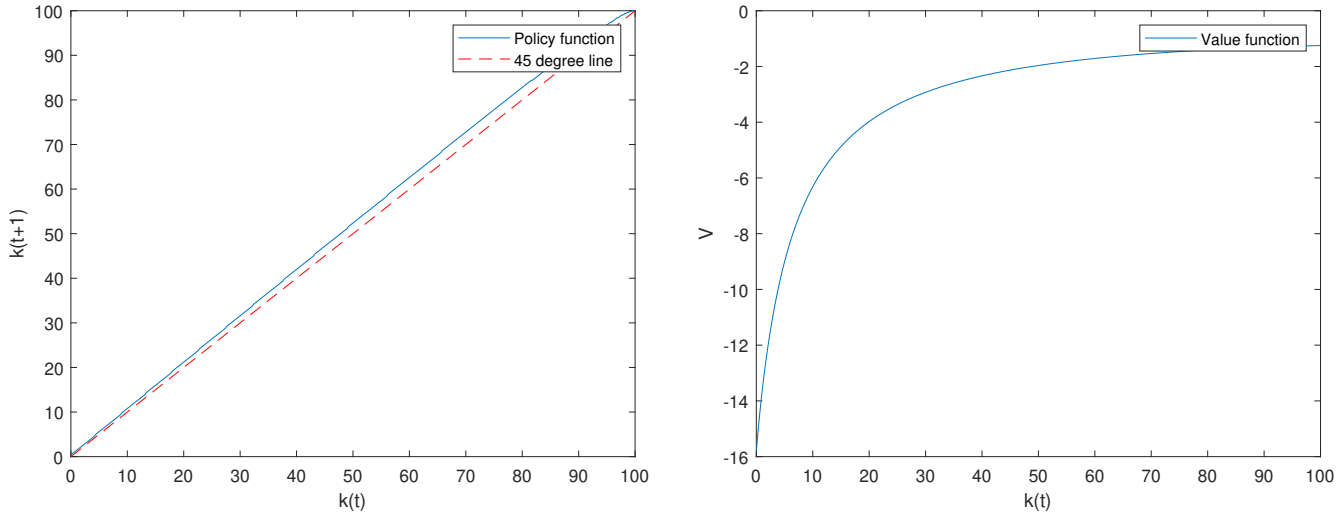

Figure 3: Policy and value functions for part (d)

(d) See figure 3. The household still saves more than their current capital stock always! All because the return is still super-high! What's going on?

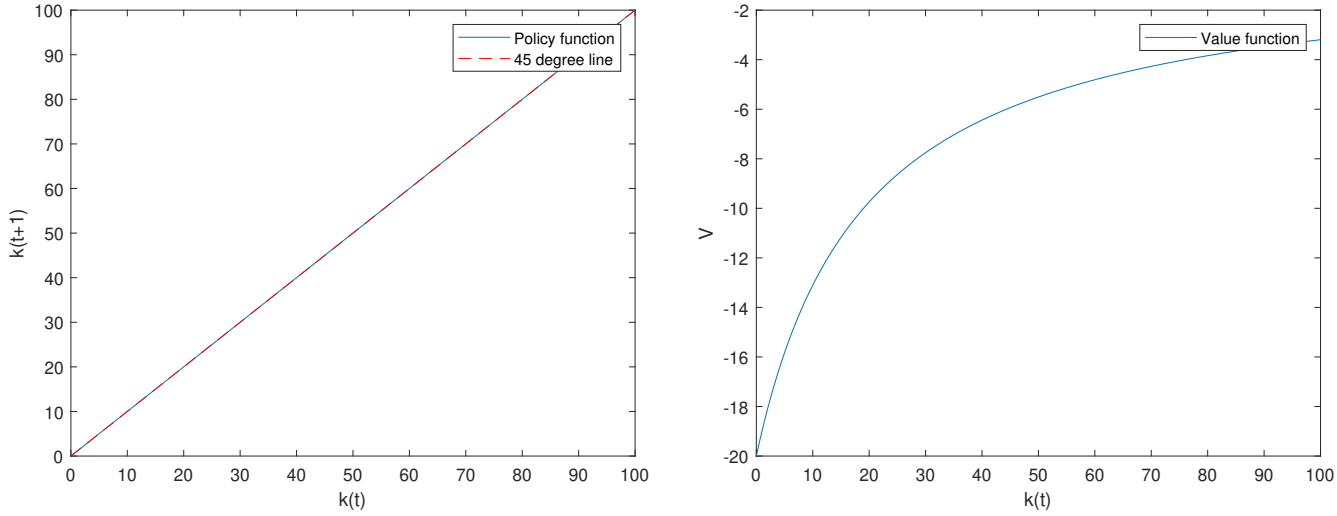

Figure 4: Policy and value functions for part (e)

(e) See figure 4. The interest rate and wage are always constant in this problem since it's in partial equilibrium. Since these prices aren't a function of the household's capital stock, we

can never find a unique steady state capital stock that sits above zero, (i.e. no point that crosses the 45 degree line above zero). In the case of part (b), the unique steady state would be zero capital, in contrast with parts (c) and (d) where the capital stock explodes due to the high interest rate. There is one more case that quickly deserves some attention though. You can find the Euler equation for the problem to be

$$c_t^{-\sigma} = \beta c_{t+1}^{-\sigma}[r + 1 - \delta],$$

where a steady state exists where $c_t = c_{t+1}$, giving a relationship between the parameters of $1 = \beta[r + 1 - \delta]$. The policy and value functions for this particular case are plotted in figure 4, (where δ and β are as in Q1 with r set such that the steady state holds). Every capital stock value gives a steady state. Although these solutions are all valid, they really motivate the need to think about some concept of general equilibrium, such that the household's savings decisions are reflected in asset returns, allowing us to find an interior steady state. We'll tackle this in the next lecture and exercise set.

Q2

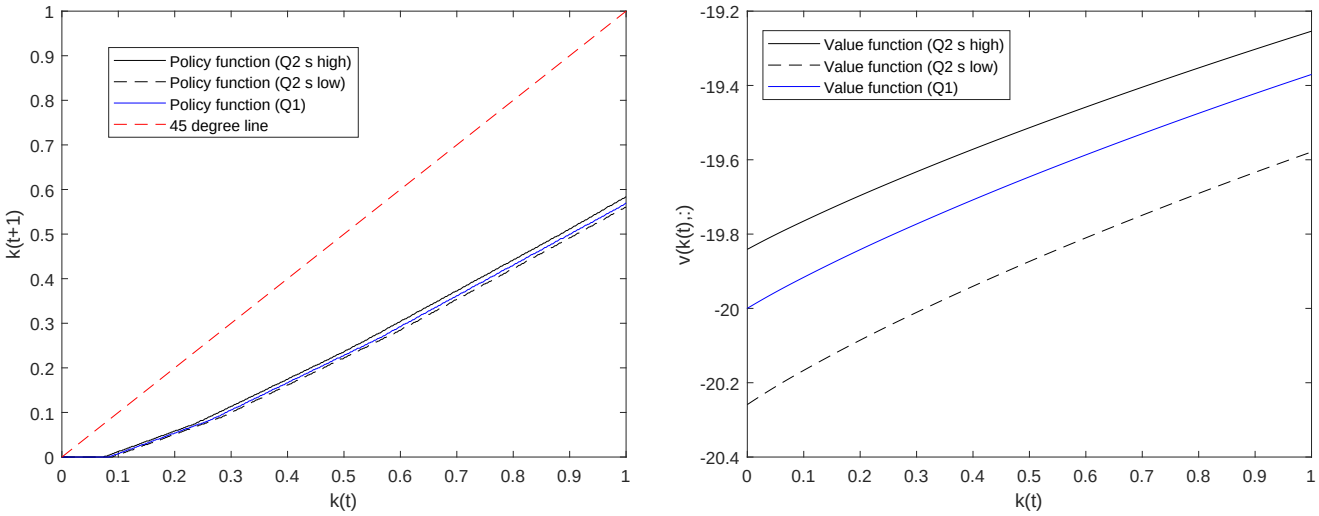

Figure 5: Policy and value functions for part Q2

Figure 5 shows the policy and value functions for Q2 (for the two different values of s) and contrasts with that for the deterministic model of Q1. The two functions for the stochastic version of the model sit either side of that from Q1. A household with more (less) labour earnings chooses to invest more (less) uniformly for their current capital state.

Let's think a bit more about the expectation part inside the code. Take a simpler example. Imagine that we have some function $f(z)$ for some variable $z \in \mathbb{R}$. Say that z is a random variable that

follows a Markov process $z \sim G(z'|z)$. Assume that $z \in \{z_L, z_H\}$. The expectation of $f(z')$ is then given by

$$\mathbb{E}_{z'|z}[f(z')] = G(z_L|z)f(z_L) + G(z_H|z)f(z_H).$$

The analogue of how I'm constructing this in the code is as follows. Define a variable M and follow the three steps:

1. Set $M = 0$ to initialise,
2. Compute 1^{st} update $M = M + G(z_L|z)f(z_L)$,
3. Compute 2^{nd} update $M = M + G(z_H|z)f(z_H)$,

where $M = \mathbb{E}_{z'|z}[f(z')]$ after the execution of the final step. It's easy to then extend to a for-loop like we have in the code, (which is important to do if you have a large state space for the stochastic variable). Notice also — it's absolutely critical to set $M = 0$ before proceeding to the cumulative sums. Otherwise, if the computation is embedded inside some other loop as it is in our code, the M variable will just explode.

Q3

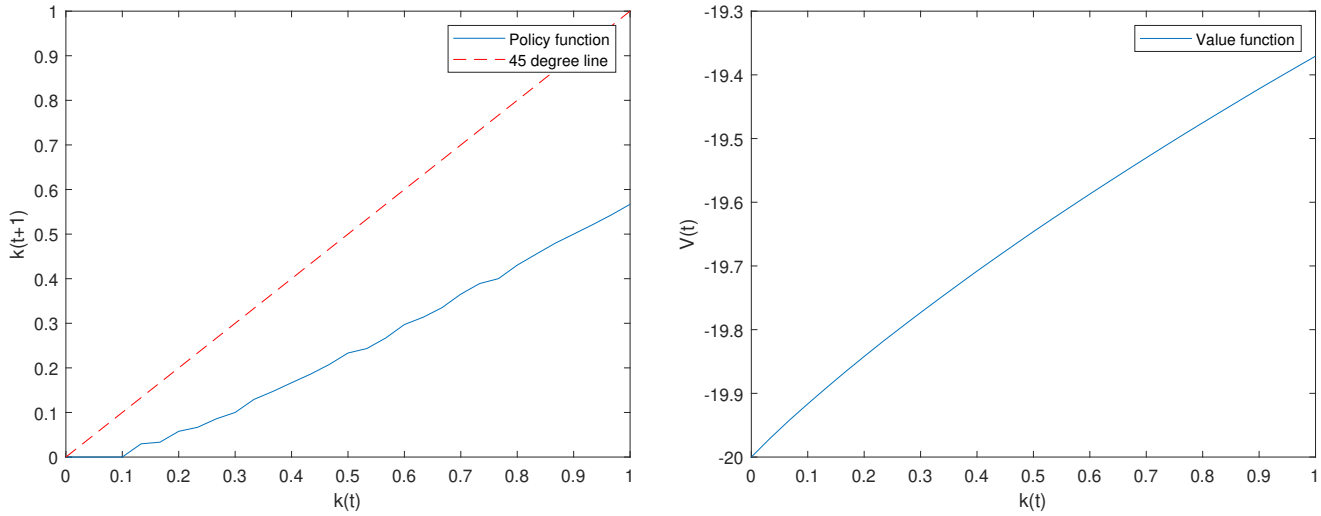

Figure 6: Policy and value functions for Q3 with 31 gridpoints

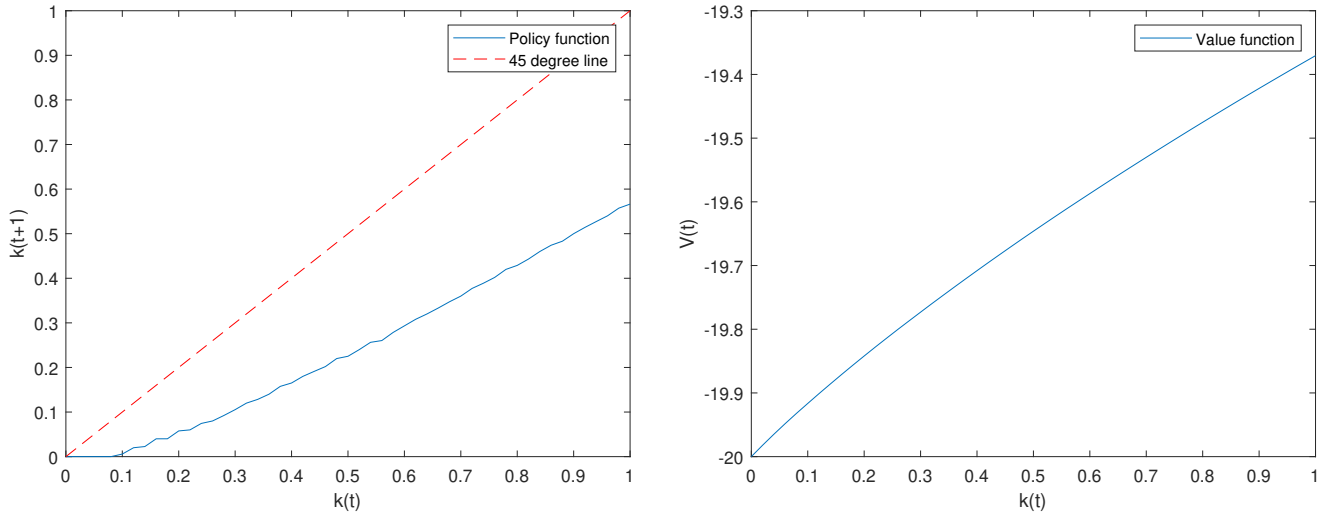

Figure 7: Policy and value functions for Q3 with 51 gridpoints

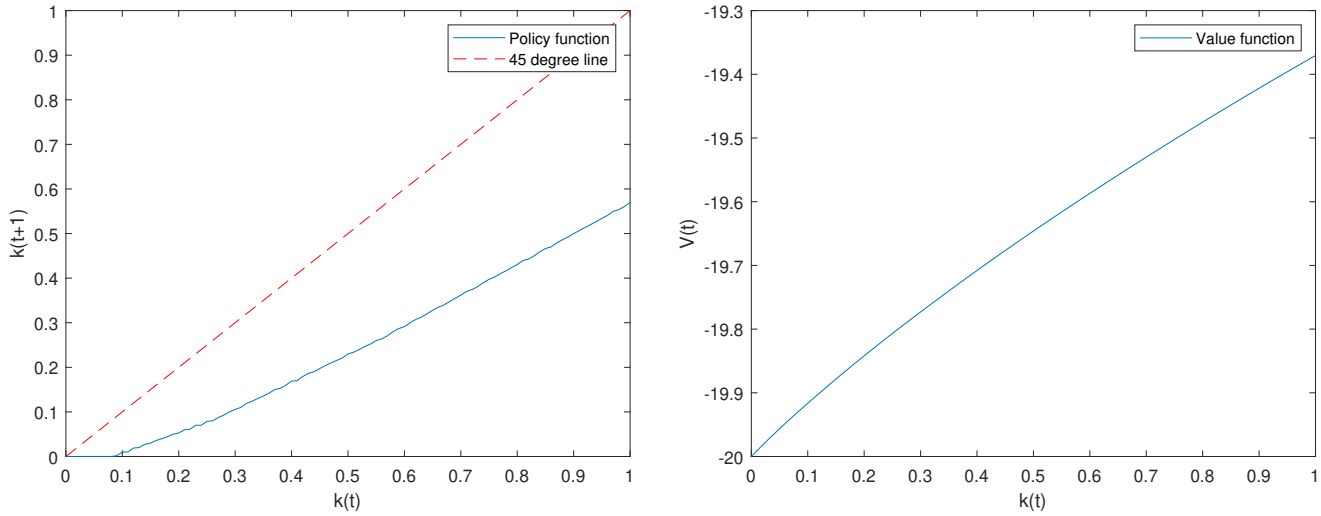

Figure 8: Policy and value functions for Q3 with 101 gridpoints

Figures 6, 7 and 8 present the policy and value functions for 31, 51 and 101 capital gridpoints respectively. The results are as you would expect: the accuracy of the policy functions improves with the number of gridpoints. The computation times also increase quite a bit. On my desktop, the times are 6.00, 8.81 and 14.88 seconds for each of the three respective runs. How do they compare with the results of the gridsearch from Q1? The policy function in figure 8 comes pretty

close, which is neat since it uses 5 times fewer gridpoints as Q1.