Model-Solving Exercise Extension Answers

The solution, in terms of polynomial coefficients, is about (-1.0444, 0.4473, -0.0395) or (-1.0344, 0.4383, -0.0379). These are two distinct solutions to the normal equations that I obtained with one draw on the technology shocks. When plotted, the implied rules for C are nearly indistinguishable by eye except for W's above the range observed in simulations. A second draw of a 50x100 matrix of technology shocks produced a similar pair of solutions:

(-1.0561 0.4531 -0.0402) and

(-1.0445 0.4423 -0.0383) .

The fact that there are two nearby solutions reflects the lack of smoothness in the objective function. If one starts from far from these solutions, even a couple of percentage points away, the equation solver can easily get stuck and require many random searches to get unstuck. With my own routine, which has only one level of loop (rather than nested loops) each evaluation is fast, so that it was not too onerous to let the routine take 50 or 60 unsuccessful random steps before it hit on a good path. My own code and a corrected version of code submitted by one of the groups working on the problem both gave exactly the same answers when given the same starting point and the same matrix of random shocks. Adding a third order term to the polynomial seems simply to make $C(W)$ closer to linear, without changing the position of the function by much.

There are two m-files, `iterexg.m` and `glerkg.m` , posted on the web page for this problem. The former solves the problem as the problem statement requested, using dynamic simulations to generate the discrepancy vectors to be set to zero. The latter instead uses a fixed grid of state variable values and Monte Carlo integration of the Euler equation at each value of the state to generate the discrepancy function, while using the same functional form for the policy rule and the same type of matrix of random shocks. This latter version is a form of the Galerkin method approach advocated by Judd (though he usually objects to Monte Carlo integration). Comparing these programs should give you insight into the similarities and differences of the "Marcet method" and the Galerkin methods.

The Galerkin method code converges, in my experience, quite easily if the range of the state variable W is kept to, say, (1.0,7.0). As one can see from the results for the dynamic simulation method, this covers the relevant range starting from K=.5. However, if the range of W is extended to lower values, convergence problems become common. With the range (0.1,7.0), convergence is very difficult and the Euler equation errors have a very large mean at the first few W values. This is a symptom of there being problems with our functional form when we try to include these low W values.