Materials 12d - Two general solution methods for the learning model, consolidated

–

See Notes 31 Dec 2019

Laura Gáti

January 3, 2020

1 Model equations and goal

$$x_t = -\sigma i_t + \hat{\mathbb{E}}_t \sum_{T=t}^{\infty} \beta^{T-t}\big((1-\beta)x_{T+1} - \sigma(\beta i_{T+1} - \pi_{T+1}) + \sigma r_T^n\big) \tag{1}$$

$$\pi_t = \kappa x_t + \hat{\mathbb{E}}_t \sum_{T=t}^{\infty} (\alpha\beta)^{T-t}\big(\kappa\alpha\beta x_{T+1} + (1-\alpha)\beta\pi_{T+1} + u_T\big) \tag{2}$$

$$i_t = \psi_\pi \pi_t + \psi_x x_t + \rho i_{t-1} + \bar{i}_t \tag{3}$$

Goal: obtain endogenous stuff as a function of expectations and states:

$$z_t = \begin{bmatrix} \pi_t \\ x_t \\ i_t \end{bmatrix} = A_a f_a + A_b f_b + A_s s_t \tag{4}$$

where I already have expectations f_a, f_b and the state vector can vary by model, but in this default case with $\rho = 0$ it is

$$s_t = \begin{bmatrix} r_t^n \\ \bar{i}_t \\ u_t \end{bmatrix} \tag{5}$$

That is, we want the matrices A_a, A_b and A_s.

2 Method 1 - M-N method (old method)

$$\underbrace{\begin{bmatrix} \sigma\psi_\pi & 1+\sigma\psi_x \\ 1 & -\kappa \end{bmatrix}}_{\equiv M} \begin{bmatrix} \pi_t \\ x_t \end{bmatrix} = \underbrace{\begin{bmatrix} c_{x,b} f_b + c_{x,s} s_t \\ c_{\pi,a} f_a + c_{\pi,s} s_t \end{bmatrix}}_{\equiv N} \tag{6}$$

where M and the c are model-specific. In the baseline model they are given by

$$c_{x,b} = \begin{bmatrix} \sigma(1-\beta\psi_\pi), & 1-\beta-\sigma\beta\psi_x, & 0 \end{bmatrix} \tag{7}$$

$$c_{x,s} = -\sigma \begin{bmatrix} -1 & 1 & 0 & \rho \end{bmatrix} (I_{nx} - \beta h_x)^{-1} \tag{8}$$

$$c_{\pi,a} = \begin{bmatrix} (1-\alpha)\beta, & \kappa\alpha\beta, & 0 \end{bmatrix} \tag{9}$$

$$c_{\pi,s} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} (I_{nx} - \alpha\beta h_x)^{-1} \tag{10}$$

$$c_{i,s} = \begin{bmatrix} 0 & 1 & 0 & \rho \end{bmatrix} \tag{11}$$

In Mathematica (`materials12d.nb`), I write the N matrix as:

$$\begin{bmatrix} \begin{bmatrix} \sigma(1-\beta\psi_\pi), & 1-\beta-\sigma\beta\psi_x, & 0 \end{bmatrix} f_b + d_1 \\ \begin{bmatrix} (1-\alpha)\beta, & \kappa\alpha\beta, & 0 \end{bmatrix} f_a + d_2 \end{bmatrix} \tag{12}$$

Then Mathematica solves for x and π as

$$\begin{bmatrix} \pi_t^* \\ x_t^* \end{bmatrix} = M^{-1} N \tag{13}$$

Then the solution for the interest rate will just be

$$i_t = \psi_\pi \pi_t^* + \psi_x x_t^* + \underbrace{c_{is} s_t}_{\equiv d_6} \tag{14}$$

The last step is to gather the matrices $g_{i,j}$, the coefficients of i on j, $i = x, \pi, i, j = f_a, f_b, s$. Mathematica will output these g-matrices and stack them appropriately to give the A-matrices:

$$\underbrace{A_a}_{ny \times ny} = \begin{pmatrix} g_{\pi,a} \\ g_{x,a} \\ g_{i,a} \end{pmatrix} \quad \underbrace{A_b}_{ny \times ny} = \begin{pmatrix} g_{\pi,b} \\ g_{x,b} \\ g_{i,b} \end{pmatrix} \quad \underbrace{A_s}_{ny \times nx} = \begin{pmatrix} g_{\pi,s} \\ g_{x,s} \\ g_{i,s} \end{pmatrix} \tag{15}$$

Now you can be copy this directly into Matlab (for the default model `matrices_A2.m`), specifying $d1, d2, d6$ in Matlab.

3 Method 2 - P-Q method (new method)

Instead of subbing out the interest rate in the original equations, write the system as:

$$\underbrace{\begin{bmatrix} 0 & 1 & \sigma \\ 1 & -\kappa & 0 \\ -\psi_\pi & -\psi_x & 1 \end{bmatrix}}_{\equiv P} \begin{bmatrix} \pi_t \\ x_t \\ i_t \end{bmatrix} = \underbrace{\begin{bmatrix} c_{x,b} f_b + c_{x,s} s_t \\ c_{\pi,a} f_a + c_{\pi,s} s_t \\ c_{i,s} s_t \end{bmatrix}}_{\equiv Q} \tag{16}$$

Then (13) is replaced by

$$\begin{bmatrix} \pi_t^* \\ x_t^* \\ i_t^* \end{bmatrix} = P^{-1} Q \tag{17}$$

where you also have to impose the relation

$$f_b(3) = \psi_\pi f_b(1) + \psi_x f_b(2) + \frac{1}{\beta}\{\begin{bmatrix} -1 & 1 & 0 \end{bmatrix} (I_{nx} - \beta h_x)^{-1} s_t - \begin{bmatrix} -1 & 1 & 0 \end{bmatrix} s_t\} \tag{18}$$

so that Mathematica recognizes that the interest rate expectations are just a function of those of π and x. Then the rest works mechanically the same way as in the M-N method.

The problem with the PQ-method is that since there are a bunch of steps Mathematica has a hard time with (I haven't even figured out how to stop Mathematica from multiplying out the matrices!), the PQ method becomes less robust than the MN method, because in the latter you're more in control of how expectations are treated.