Dynare 中自撰函数求稳态：一个例子

陈普

2022 年 10 月 13 日

1 动态方程

比如我有一个如下动态方程，

$$\begin{aligned}\frac{c_{t+1}}{c_t} =&\beta([\mu(1-\alpha)+1]k_t^{\mu(1-\alpha)}L^{\mu(1-\alpha)}+1-\delta-\gamma)\\ \frac{k_{t+1}}{k_t} =&k_t^{\mu(1-\alpha)}L^{\mu(1-\alpha)}+(1-\delta-\gamma)-\frac{c_t}{k_t}\end{aligned}$$

除了 k_t, c_t 是内生变量外，μ 是外生变量，其他都是参数。这个方程组稳态的解析解并不是那么明显或者好写，反而调用`fsolve`函数数值求解更方便。

2 代码和解读

2.1 Dynare 代码

在这样的思路下，我撰写了 Dynare 代码如下所示。

Listing 1: ai.mod 文件

```
var k c;
varexo mu;

parameters beta, alpha, L, delta, eta;
beta = 0.99;
alpha = 0.36;
L = 0.1;
delta = 0.1;
eta = 0.8;

model;
c(+1)/c = beta*((mu*(1-alpha)+1)*k(-1)^(mu*(1-alpha))*L^(mu*(1-alpha))+1-delta-mu*(1-
    eta));
k/k(-1) = k(-1)^(mu*(1-alpha))*L^(mu*(1-alpha))+1-delta-mu*(1-eta)-c/k(-1);
end;

% 自撰函数
y = findss([0.1,0.5],0.2,beta = beta,alpha = alpha,L = L,delta = delta,eta=eta);

initval;
mu = 0.5;
k = y(1);
```

```
c = y(2);
end;

steady;

endval;
mu = 1;
k = y(1);
c = y(2);
end;
steady;

perfect_foresight_setup(periods=30);
perfect_foresight_solver;

rplot k;
```

上述代码可以存为ai.mod文件，该文件主要做了这样几件事：

- 利用自己写的函数findss来获得 k, c 的稳态，并把它以y(1),y(2)的引用方式传到初值和终值模块中。
- findss可以放在model块和initval块之间，Dynare 一般可以在“块”间直接写 matlab 函数并调用，但这个函数findss要保存为 m 文件，并与ai.mod在同一个目录下。
- 计算了在 $\mu = 0.5$ 变化到 $\mu = 1$ 后，系统如何从旧稳态转移到新稳态，在最后利用rplot绘制了 k 的转移动态图。注意当初值或终值之后没有steady命令，会精确地计算如何从初值抵达终值，有了steady命令，是利用初值和终值作为初始条件计算得到稳态后，再计算两个稳态间的转移。

2.2 findss函数

注意该稳态求解函数findss的结构，主要包括三个部分：**第一参数模块，第二fsolve函数，第三静态方程子函数**。该结构以后撰写时可以参考。

- 先是参数说明模块arguments,...,end，注意这个模块意味着可以使用名称-参数对来匹配函数参数，而不是过去仅仅依赖位置匹配参数。这个功能主要由options结构体完成。
- 把静态方程写成一个子函数kc嵌套在主函数findss下，此时子函数可以直接调用主函数里面的变量。

Listing 2: findss.m 文件

```
function y = findss(x0, mu, options)
        % 参数说明
        arguments
        x0 double
        mu
        options.beta = 0.99
        options.alpha = 0.36
        options.L = 2
        options.delta = 0.9
        options.eta = 0.8
        end

        % 调用fsolve
```

```
14          y = fsolve(@kc,x0);
15
16          % 静态方程
17          function y = kc(x)
18                  y(1) = options.beta * ((mu*(1- options.alpha)+1)*x(1)^(mu*(1-options.
                        alpha))*options.L^(mu*(1-options.alpha)) + 1-options.delta-mu*(1-
                        options.eta)) - 1;
19                  y(2) = x(1)^(mu*(1-options.alpha))*options.L^(mu*(1-options.alpha))+1-
                        options.delta-mu*(1-options.eta)-x(2)/x(1)-1;
20          end
21  end
```