Analytical

September 24, 2021

1 Analytical Example

James Yu, 23 September 2021

[1]:
```
from sympy import *

a = 0.7
b_1 = 0.29
b_2 = 0.01
R = 0.2
delta = 0.8

K_1, K_2, x, r_1, r_2 = symbols("K_1 K_2 x r_1 r_2")
```

First, the first-order conditions for each strategic agent:

[2]:
```
rt1 = -(a*b_1*K_1*x - b_1*b_2*K_1*r_2) / (b_1*b_1*K_1 + R/delta)
rt1
```

[2]:

$$\frac{0.0029K_1r_2 - 0.203K_1x}{0.0841K_1 + 0.25}$$

[3]:
```
rt2 = -(a*b_2*K_2*x - b_1*b_2*K_2*r_1) / (b_2*b_2*K_2 + R/delta)
rt2
```

[3]:

$$\frac{0.0029K_2r_1 - 0.007K_2x}{0.0001K_2 + 0.25}$$

We want to solve for r_1 and r_2, so we can substitute the equations into each other to get explicit forms:

[4]:
```
also_rt1 = simplify(rt1.subs(r_2, rt2))
Lx1 = solve(r_1 - also_rt1, r_1)[0]
Lx1
```

[4]:

$$-\frac{203.0K_1x\left(K_2 + 1250.0\right)}{105125.0K_1 + 125.0K_2 + 312500.0}$$

[5]:
```
also_rt2 = simplify(rt2.subs(r_1, rt1))
Lx2 = solve(r_2 - also_rt2, r_2)[0]
Lx2
```

[5]:

$$-\frac{7.0K_2x\,(841.0K_1+1250.0)}{105125.0K_1+125.0K_2+312500.0}$$

This yields the expressions $r_1 = L_1x$ and $r_2 = L_2x$. Notice that x is linearly related to each expresion now.

[6]:

```
L_1 = Lx1 / x
L_2 = Lx2 / x
Ksub1_1 = simplify(1 + R*L_1*L_1 + delta * K_1 * (a + b_1*L_1 + b_2*L_2)**2)
Ksub1_1
```

[6]:

$$\frac{0.1318688K_1^2\,(0.0008K_2+1)^2}{(0.3364K_1+0.0004K_2+1)^2}+\frac{0.392K_1\,(0.00053824K_1K_2-1)^2}{(0.3364K_1+0.0004K_2+1)^2}+1$$

The above is the recursive formulation for K_t^1 given K_{t-1}^1, substituting the optimal strategies. We can do the same for agent 2:

[7]:

```
Ksub1_2 = simplify(1 + R*L_2*L_2 + delta * K_2 * (a + b_1*L_1 + b_2*L_2)**2)
Ksub1_2
```

[7]:

$$\frac{0.0001568K_2^2\,(0.6728K_1+1)^2}{(0.3364K_1+0.0004K_2+1)^2}+\frac{0.392K_2\,(0.00053824K_1K_2-1)^2}{(0.3364K_1+0.0004K_2+1)^2}+1$$

Now simply iterate both of these from $K_l = 1$ upwards to get the steady-state metrices.

[8]:

```
K1unit = 1
K2unit = 1
while True:
    K1_prime = Ksub1_1.subs(K_1, K1unit).subs(K_2, K2unit)
    K2_prime = Ksub1_2.subs(K_1, K1unit).subs(K_2, K2unit)
    if K1_prime == K1unit and K2_prime == K2unit:
        break
    print("K_1 =", K1unit, ", K_2 =", K2unit)
    K1unit = K1_prime
    K2unit = K2_prime
```

```
K_1 = 1 , K_2 = 1
K_1 = 1.29303188274632 , K_2 = 1.21936731360347
K_1 = 1.35277591957769 , K_2 = 1.23197442084119
K_1 = 1.36396979601910 , K_2 = 1.22794268864224
K_1 = 1.36603519717924 , K_2 = 1.22602609151453
K_1 = 1.36641557711119 , K_2 = 1.22545877906690
K_1 = 1.36648567406287 , K_2 = 1.22531502286461
K_1 = 1.36649860540057 , K_2 = 1.22528133200540
K_1 = 1.36650099366283 , K_2 = 1.22527379946421
K_1 = 1.36650143524911 , K_2 = 1.22527216756993
K_1 = 1.36650151698981 , K_2 = 1.22527182189806
K_1 = 1.36650153213737 , K_2 = 1.22527174990166
K_1 = 1.36650153494746 , K_2 = 1.22527173510117
K_1 = 1.36650153546932 , K_2 = 1.22527173209013
```

```
K_1 = 1.36650153556634 , K_2 = 1.22527173148273
K_1 = 1.36650153558440 , K_2 = 1.22527173136106
K_1 = 1.36650153558776 , K_2 = 1.22527173133684
K_1 = 1.36650153558839 , K_2 = 1.22527173133204
K_1 = 1.36650153558850 , K_2 = 1.22527173133109
K_1 = 1.36650153558853 , K_2 = 1.22527173133090
K_1 = 1.36650153558853 , K_2 = 1.22527173133087
K_1 = 1.36650153558853 , K_2 = 1.22527173133086
K_1 = 1.36650153558853 , K_2 = 1.22527173133086
K_1 = 1.36650153558853 , K_2 = 1.22527173133086
```

From here, we can compute the steady-state message.

[9]:
```
print(L_1.subs(K_1, K1unit).subs(K_2, K2unit) * 4)
```

```
-3.04260012703272
```

[10]:
```
print(L_2.subs(K_1, K1unit).subs(K_2, K2unit) * 4)
```

```
-0.180386963025061
```

Clearly, this is not the same as in the chart of the other notebook.