Analytical

September 24, 2021

1 Analytical Example (fixed)

James Yu, 24 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]:
$$-\frac{K_1x\left(8470329472543.0K_2 + 1.26875\cdot 10^{32}\right)}{5.25625\cdot 10^{31}K_1 + 6.25\cdot 10^{28}K_2 + 1.5625\cdot 10^{32}}$$

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

[5]:

$$-\frac{70.0K_2x}{841.0K_1 + K_2 + 2500.0}$$

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]:

$$\frac{0.1318688K_1^2\left(6.67612175175803\cdot 10^{-20}K_2+1\right)^2}{(0.3364K_1+0.0004K_2+1)^2}+\frac{0.392K_1\left(7.55503050992827\cdot 10^{-21}K_1^2K_2+2.36118324143482\cdot\right.}{}$$

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]:

$$\frac{0.0001568K_2^2}{(0.3364K_1+0.0004K_2+1)^2}+\frac{0.392K_2\left(7.55503050992827\cdot 10^{-21}K_1^2K_2+2.36118324143482\cdot 10^{-17}K_1^2+8.983389\right.}{(0.3364K_1 -}$$

1

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

[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 = 1.29314983914354 , K_2 = 1.21944559943071
K_1 = 1.35300682323108 , K_2 = 1.23208755014799
K_1 = 1.36423671239342 , K_2 = 1.22805052305727
K_1 = 1.36631033335162 , K_2 = 1.22612820128246
K_1 = 1.36669229576187 , K_2 = 1.22555862291467
K_1 = 1.36676265709895 , K_2 = 1.22541418704317
K_1 = 1.36677562496290 , K_2 = 1.22538031953137
K_1 = 1.36677801640053 , K_2 = 1.22537274513599
K_1 = 1.36677845767724 , K_2 = 1.22537110398241
K_1 = 1.36677853915208 , K_2 = 1.22537075637493
K_1 = 1.36677855420410 , K_2 = 1.22537068399437
```

```
K_1 = 1.36677855698651 , K_2 = 1.22537066912158
K_1 = 1.36677855750115 , K_2 = 1.22537066609776
K_1 = 1.36677855759639 , K_2 = 1.22537066548827
K_1 = 1.36677855761403 , K_2 = 1.22537066536631
K_1 = 1.36677855761730 , K_2 = 1.22537066534205
K_1 = 1.36677855761790 , K_2 = 1.22537066533725
K_1 = 1.36677855761802 , K_2 = 1.22537066533630
K_1 = 1.36677855761804 , K_2 = 1.22537066533612
K_1 = 1.36677855761804 , K_2 = 1.22537066533608
K_1 = 1.36677855761804 , K_2 = 1.22537066533607
K_1 = 1.36677855761804 , K_2 = 1.22537066533607
```

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

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

```
-3.04004273977039
```

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

```
-0.0939833700734357
```