Solow Growth Model

November 11, 2024

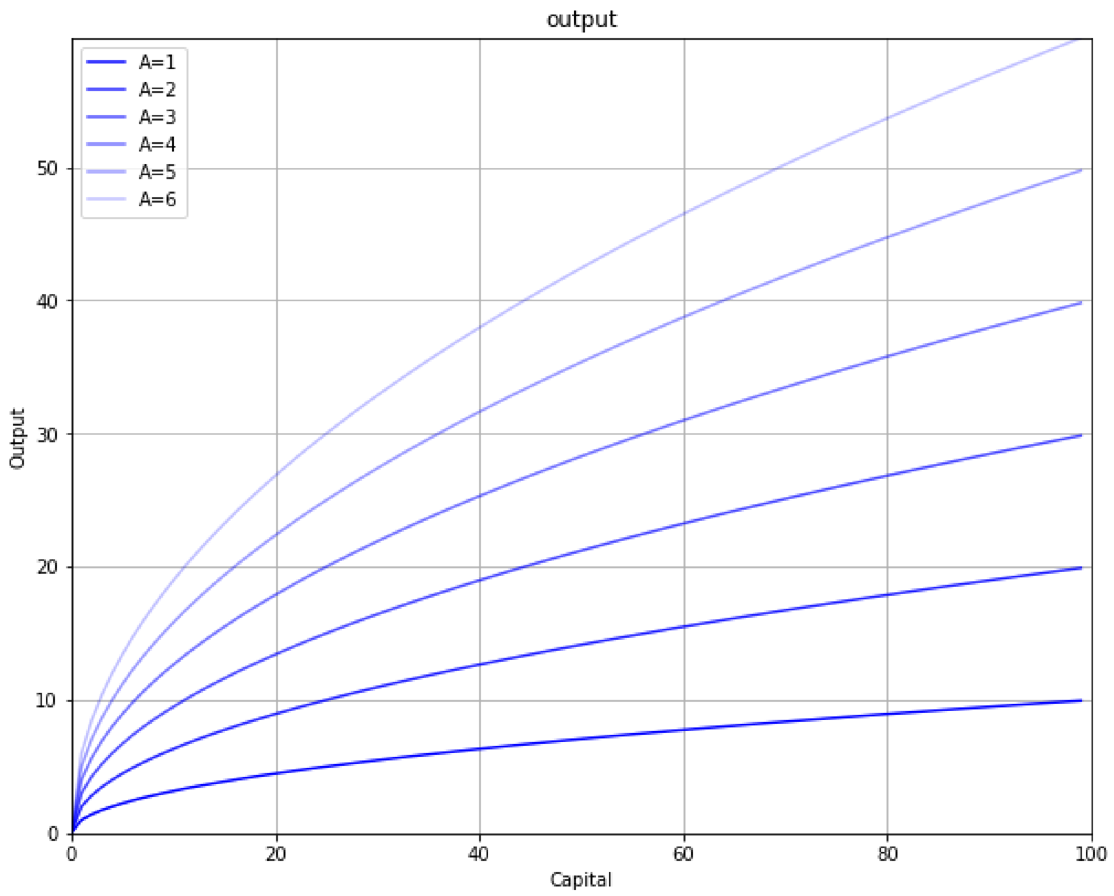
[1]:
```python
from sympy import Symbol
from sympy import latex

A, K, N, alpha = Symbol('A'), Symbol('K'), Symbol('N'), Symbol('alpha')
Y = A * (K)**(alpha) * (N)**(1-alpha)

Yprime = Y.diff(K)
print(Yprime)
latex(Yprime)
```

```
A*K**alpha*N**(1 - alpha)*alpha/K
```

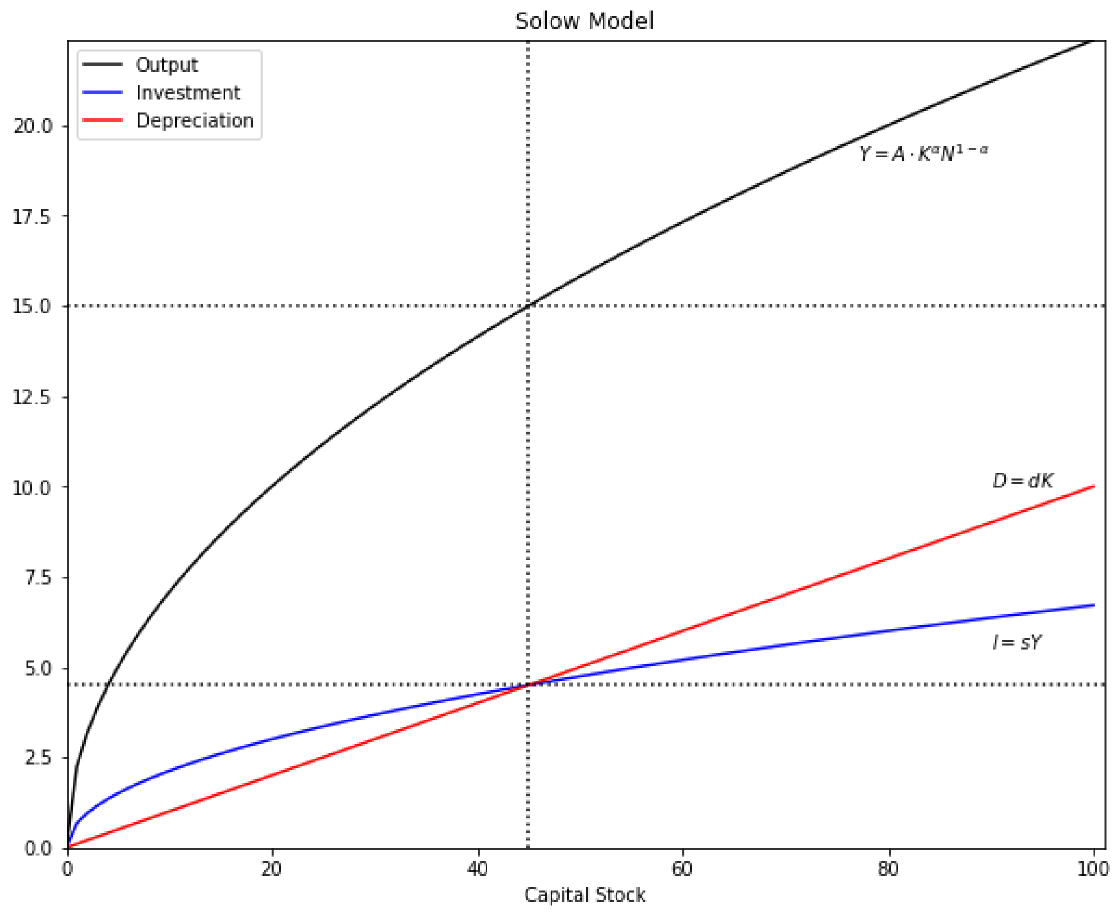
[1]:
```
'\\frac{A K^{\\alpha} N^{1 - \\alpha} \\alpha}{K}'
```

[2]:
```python
# Production function for different values of Hicks-neutral technological␣
↪progress

import numpy as np
import matplotlib.pyplot as plt


K_size = 100
A = 1
N = K_size/2
alpha = 0.50

k = np.arange(K_size)


def output(k, A):
    y = A * (k)**(alpha)
    return y

y  = output(k, A)
y2 = output(k, A+1)
y3 = output(k, A+2)
y4 = output(k, A+3)
y5 = output(k, A+4)
```

```
y6 = output(k, A+5)


y_max = np.max(y6)
v = [0, K_size, 0, y_max]
fig, ax = plt.subplots(figsize=(10, 8))
ax.set(title="output", xlabel="Capital", ylabel="Output")
ax.grid()
ax.plot(k, y,  "b-", alpha=1.00, label="A=1")
ax.plot(k, y2, "b-", alpha=0.85, label="A=2")
ax.plot(k, y3, "b-", alpha=0.70, label="A=3")
ax.plot(k, y4, "b-", alpha=0.55, label="A=4")
ax.plot(k, y5, "b-", alpha=0.40, label="A=5")
ax.plot(k, y6, "b-", alpha=0.25, label="A=6")
ax.legend()
plt.axis(v)
plt.show()
```



[3]:
```
# Behavior of output, investment and depreciation rate.

import numpy as np
import matplotlib.pyplot as plt



K_size = 101
A = 1
N = 5
alpha = 0.50
s = 0.30
d = 0.10

K = np.arange(K_size)


def output(K):
    Y = A * (K)**(alpha) * (N)**(1-alpha)
    return Y

Y = output(K)
D = d*K
I = s*Y

Kstar = ((s*A*(N)**(1-alpha))/d)**(1/(1-alpha))
Ystar = A  *(Kstar**alpha)*((N)**(1-alpha))
Istar = s*Ystar
Cstar = Ystar - Istar
Dstar = d*Kstar


y_max = np.max(Y)
v = [0, K_size, 0, y_max]

fig, ax = plt.subplots(figsize=(10, 8))
ax.plot(K, Y, "k", ls = '-', label="Output")
ax.plot(K, I, "b", ls = '-', label="Investment")
ax.plot(K, D, "r", ls = '-', label="Depreciation")
ax.set(title="Solow Model", xlabel="Capital Stock")
plt.text(77, 19,  r'$Y = A \cdot K^{\alpha} N^{1-\alpha}$')
plt.text(90, 10,  r'$D = dK$')
plt.text(90, 5.5, r'$I = sY$')
plt.legend(loc=2)
plt.axvline(x = Kstar, ls = ":", color = 'k')
plt.axhline(y = Istar, ls = ":", color = 'k')
plt.axhline(y = Ystar, ls = ":", color = 'k')
plt.axis(v)
plt.show()
```



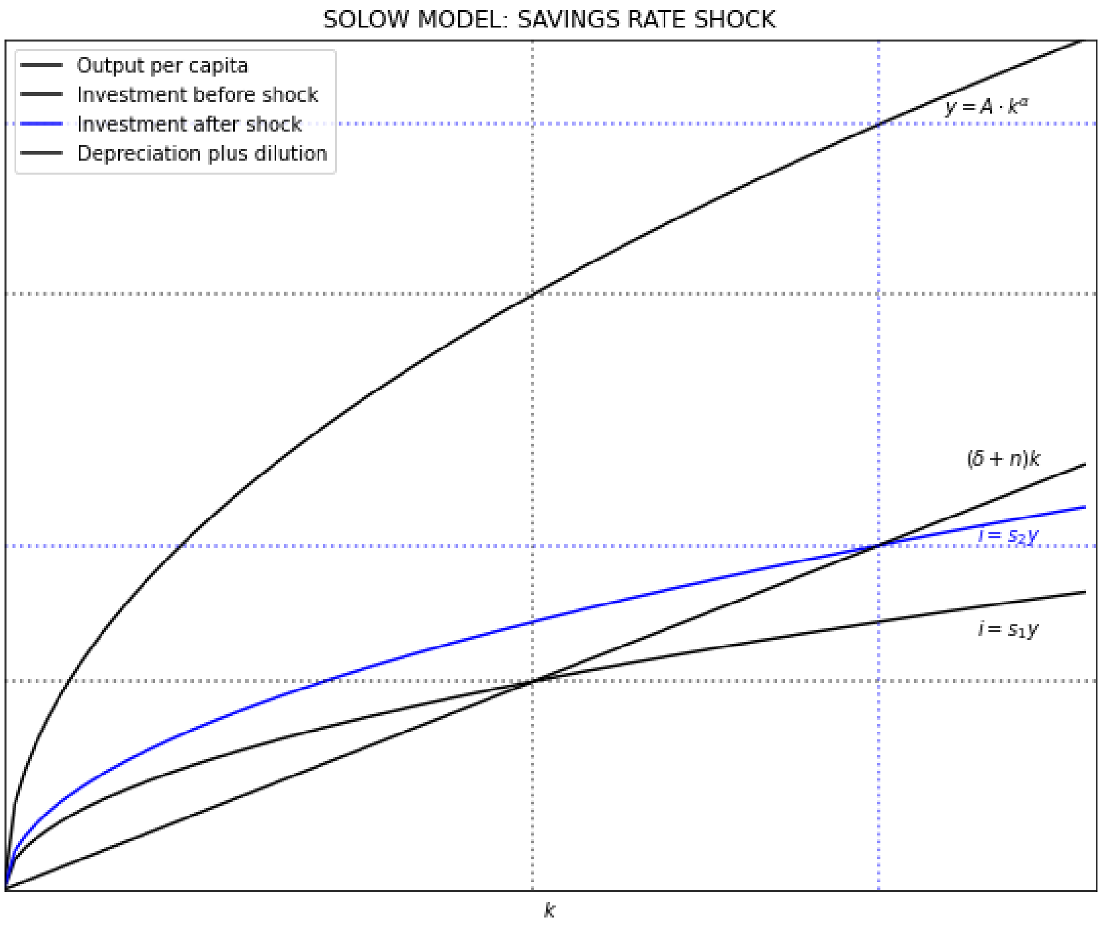
[4]:
```
# Replications of Solow model as in Romer textbook

import numpy as np
import matplotlib.pyplot as plt

K_size = 101
A = 1
alpha = 0.50
delta = 0.03
s1 = 0.35
s2 = 0.45
n  = 0.02

k  = np.arange(K_size)
```

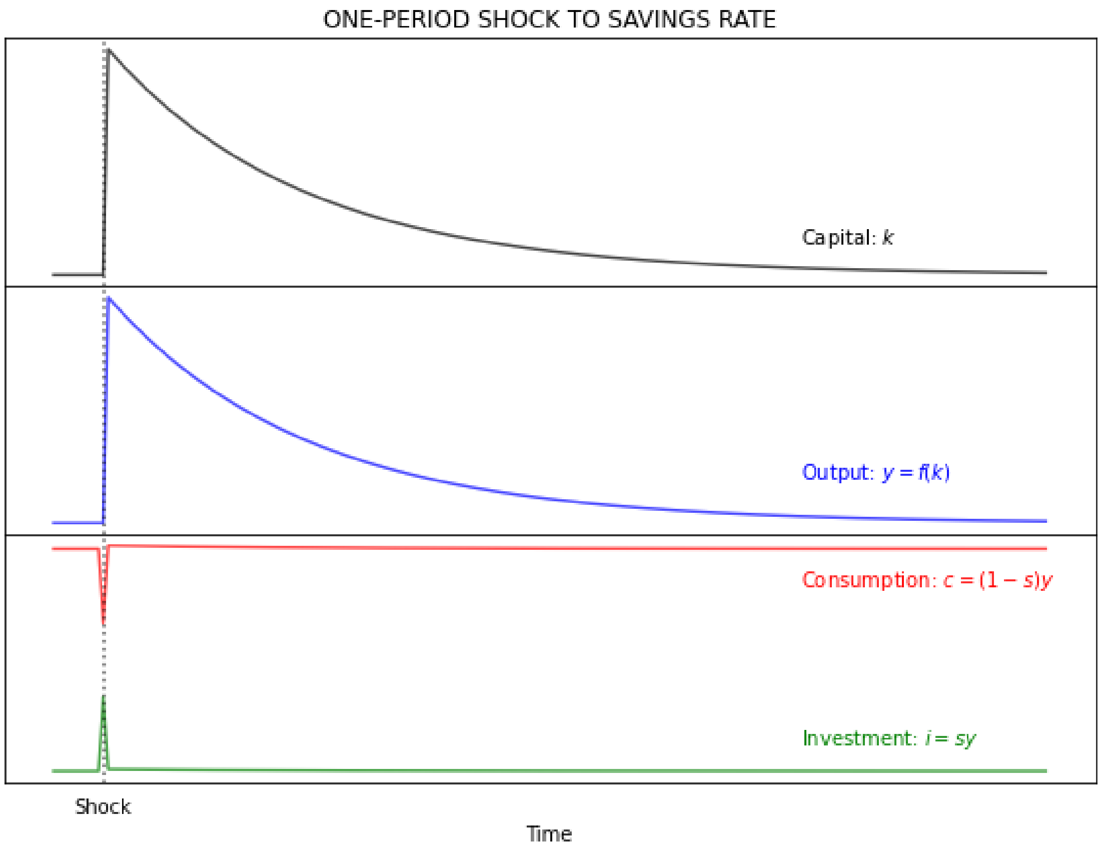
```python
def output(k):
    y = A * (k)**(alpha)
    return y

y  = output(k)
d  = delta*k
i1 = s1*y
i2 = s2*y
d_and_i = (delta + n)*k


k_star1 = (s1/(n+delta)*A)**(1/(1-alpha))
k_star2 = (s2/(n+delta)*A)**(1/(1-alpha))
y_star1 = A*(k_star1**alpha)
y_star2 = A*(k_star2**alpha)
i_star1 = s1*y_star1
i_star2 = s2*y_star2
c_star1 = y_star1 - i_star1
c_star2 = y_star2 - i_star2
d_star1 = delta*k_star1
d_star2 = delta*k_star2


y_max = np.max(y)
v = [0, K_size, 0, y_max]
fig, ax = plt.subplots(figsize=(10, 8))
ax.set(title="SOLOW MODEL: SAVINGS RATE SHOCK", xlabel=r'$k$')
ax.plot(k, y      , "k", ls = '-', label="Output per capita")
ax.plot(k, i1     , "k", ls = '-', label="Investment before shock")
ax.plot(k, i2     , "b", ls = '-', label="Investment after shock")
ax.plot(k, d_and_i, "k", ls = '-', label="Depreciation plus dilution")
plt.text(87, 9.1, r'$y = A \cdot k^{\alpha}$', color = 'k')
plt.text(89, 5.0, r'$(\delta + n)k$'         , color = 'k')
plt.text(90, 3.0, r'$i = s_{1}y$'            , color = 'k')
plt.text(90, 4.1, r'$i = s_{2}y$'            , color = "b")
plt.legend(loc=2)
plt.axvline(x = k_star1, ls = ":", color = 'k', alpha = 0.6)
plt.axhline(y = i_star1, ls = ":", color = 'k', alpha = 0.6)
plt.axhline(y = y_star1, ls = ":", color = 'k', alpha = 0.6)
plt.axvline(x = k_star2, ls = ":", color = 'b', alpha = 0.6)
plt.axhline(y = i_star2, ls = ":", color = 'b', alpha = 0.6)
plt.axhline(y = y_star2, ls = ":", color = 'b', alpha = 0.6)
ax.yaxis.set_major_locator(plt.NullLocator())
ax.xaxis.set_major_locator(plt.NullLocator())
plt.axis(v)
plt.show()
```

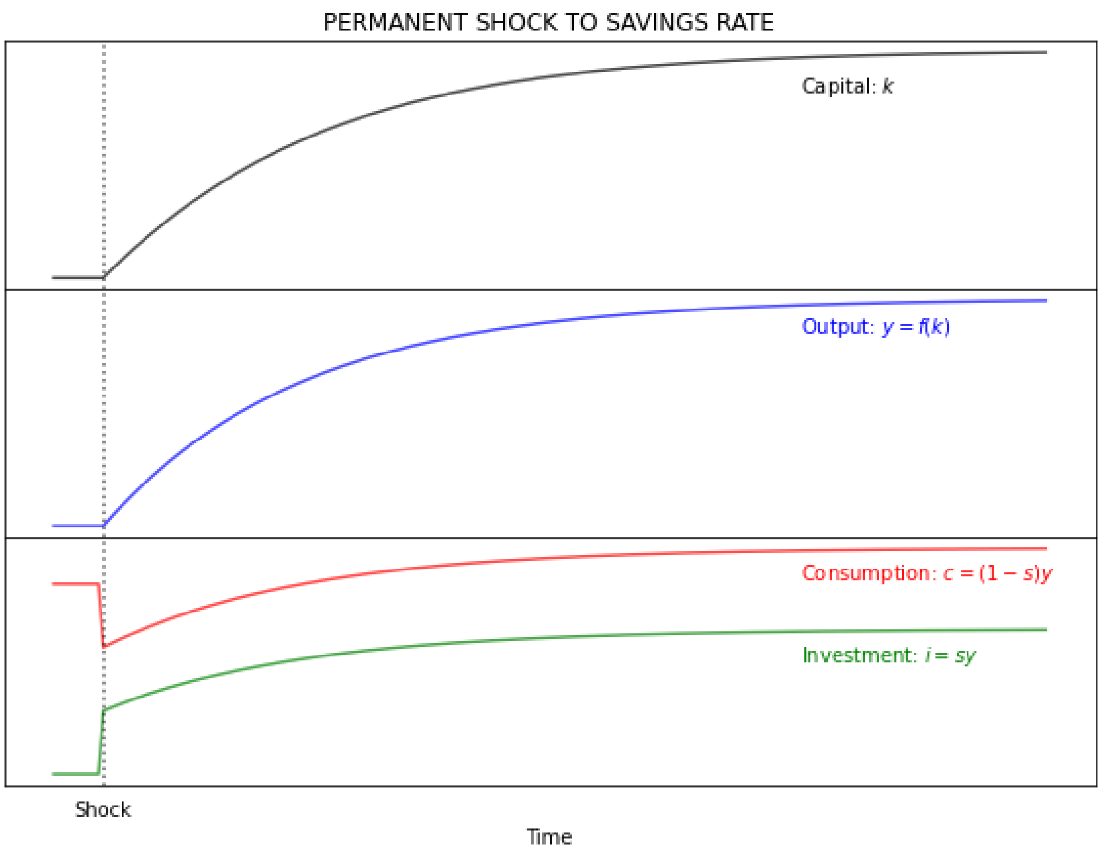
```
T = 200
t_shock = 10
time = np.arange(T)
s = np.zeros(T)
y = np.zeros(T)
k = np.zeros(T)
i = np.zeros(T)
c = np.zeros(T)

y[0] = y_star1
k[0] = k_star1
i[0] = i_star1
c[0] = c_star1

s = np.zeros(T)
s[0:T] = s1
s[t_shock] = s2

for j in range(1, T):
    k[j] = k[j-1] + i[j-1] - (n + delta)*k[j-1]
    y[j] = A*k[j]**alpha
    i[j] = s[j]*y[j]
    c[j] = y[j] - i[j]


ticks = [""]*T
ticks[t_shock] = 'Shock'

fig, (ax1, ax2, ax3) = plt.subplots(3, 1, sharex=True, figsize=(10, 7))
fig.subplots_adjust(hspace=0)
ax1.set(title="ONE-PERIOD SHOCK TO SAVINGS RATE")
ax1.plot(time, k, "k-", alpha = 0.7)
ax1.axvline(x = t_shock, color="k", ls = ':', alpha = 0.6)
ax1.yaxis.set_major_locator(plt.NullLocator())
ax1.text(150, 49.1, 'Capital: '+r'$k$')

ax2.plot(time, y, "b-", alpha = 0.7)
ax2.axvline(x = t_shock, color="k", ls = ':', alpha = 0.6)
ax2.yaxis.set_major_locator(plt.NullLocator())
ax2.text(150, 7.01, 'Output: '+ r'$y=f(k)$', color = "b")

ax3.plot(time, i, "g-", alpha = 0.7)
ax3.plot(time, c, "r-", alpha = 0.7)
ax3.axvline(x = t_shock, color="k", ls = ':', alpha = 0.6)
ax3.yaxis.set_major_locator(plt.NullLocator())
```

```
ax3.xaxis.set_major_locator(plt.NullLocator())
ax3.text(150, 4.2, 'Consumption: '+r'$c = (1-s)y$', color = "r")
ax3.text(150, 2.7, 'Investment: '+r'$i = sy$'     , color = "g")
plt.xticks(time, ticks)
plt.xlabel('Time')
plt.tick_params(axis='both', which='both', bottom=False, top=False,
                labelbottom=True, left=False, right=False, labelleft=True)



time = np.arange(T)
s = np.zeros(T)
y = np.zeros(T)
k = np.zeros(T)
i = np.zeros(T)
c = np.zeros(T)

y[0] = y_star1
k[0] = k_star1
i[0] = i_star1
c[0] = c_star1

s = np.zeros(T)
s[0:t_shock] = s1
s[t_shock:T] = s2

for j in range(1, T):
    k[j] = k[j-1] + i[j-1] - (n + delta)*k[j-1]
    y[j] = A*k[j]**alpha
    i[j] = s[j]*y[j]
    c[j] = y[j] - i[j]


fig, (ax1, ax2, ax3) = plt.subplots(3, 1, sharex=True, figsize=(10, 7))
fig.subplots_adjust(hspace=0)
ax1.set(title="PERMANENT SHOCK TO SAVINGS RATE")
ax1.plot(time, k, "k-", alpha = 0.7)
ax1.axvline(x = t_shock, color="k", ls = ':', alpha = 0.6)
ax1.yaxis.set_major_locator(plt.NullLocator())
ax1.text(150, 75.1, 'Capital: '+r'$k$')

ax2.plot(time, y, "b-", alpha = 0.7)
ax2.axvline(x = t_shock, color="k", ls = ':', alpha = 0.6)
ax2.yaxis.set_major_locator(plt.NullLocator())
ax2.text(150, 8.7, 'Output: '+ r'$y=f(k)$', color = "b")

ax3.plot(time, i, "g-", alpha = 0.7)
```

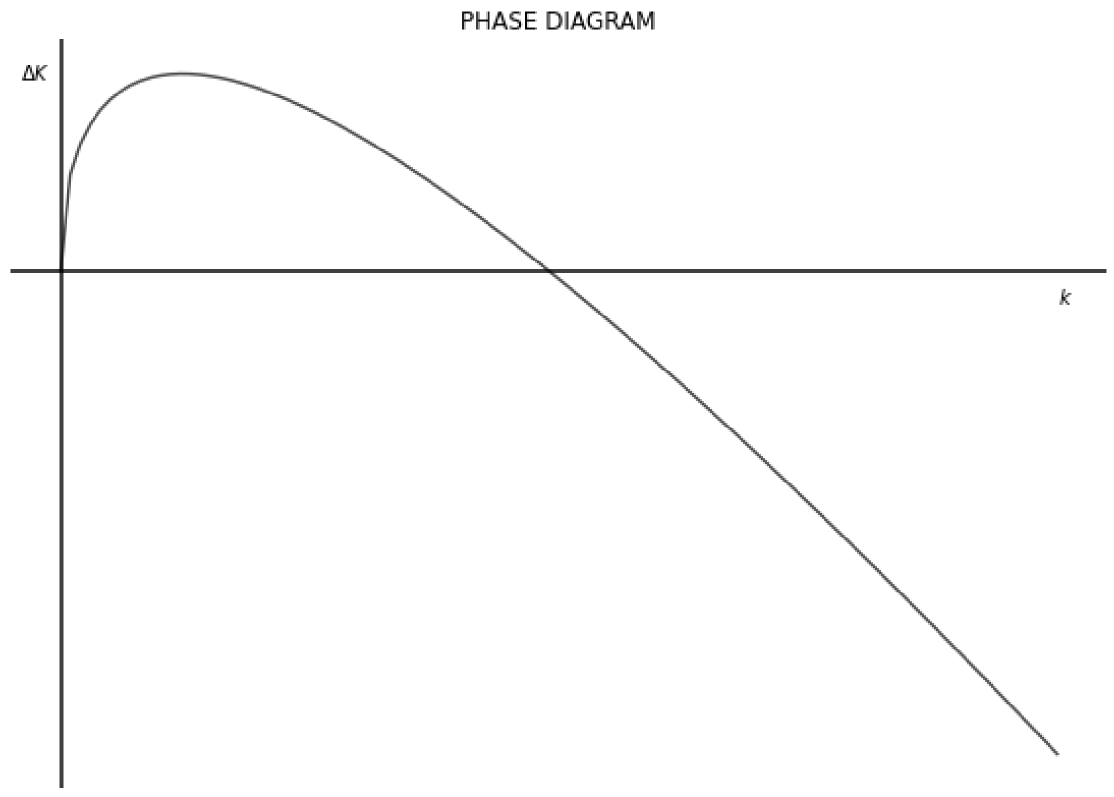
```
ax3.plot(time, c, "r-", alpha = 0.7)
ax3.axvline(x = t_shock, color="k", ls = ':', alpha = 0.6)
ax3.yaxis.set_major_locator(plt.NullLocator())
ax3.xaxis.set_major_locator(plt.NullLocator())
ax3.text(150, 4.6, 'Consumption: '+r'$c = (1-s)y$', color = "r")
ax3.text(150, 3.7, 'Investment: '+r'$i = sy$'      , color = "g")
plt.xticks(time, ticks)
plt.xlabel('Time')
plt.tick_params(axis='both', which='both', bottom=False, top=False,
                labelbottom=True, left=False, right=False, labelleft=True)
```




ONE-PERIOD SHOCK TO SAVINGS RATE
Capital: k
Output: $y = f(k)$
Consumption: $c = (1 - s)y$
Investment: $i = sy$
Shock
Time

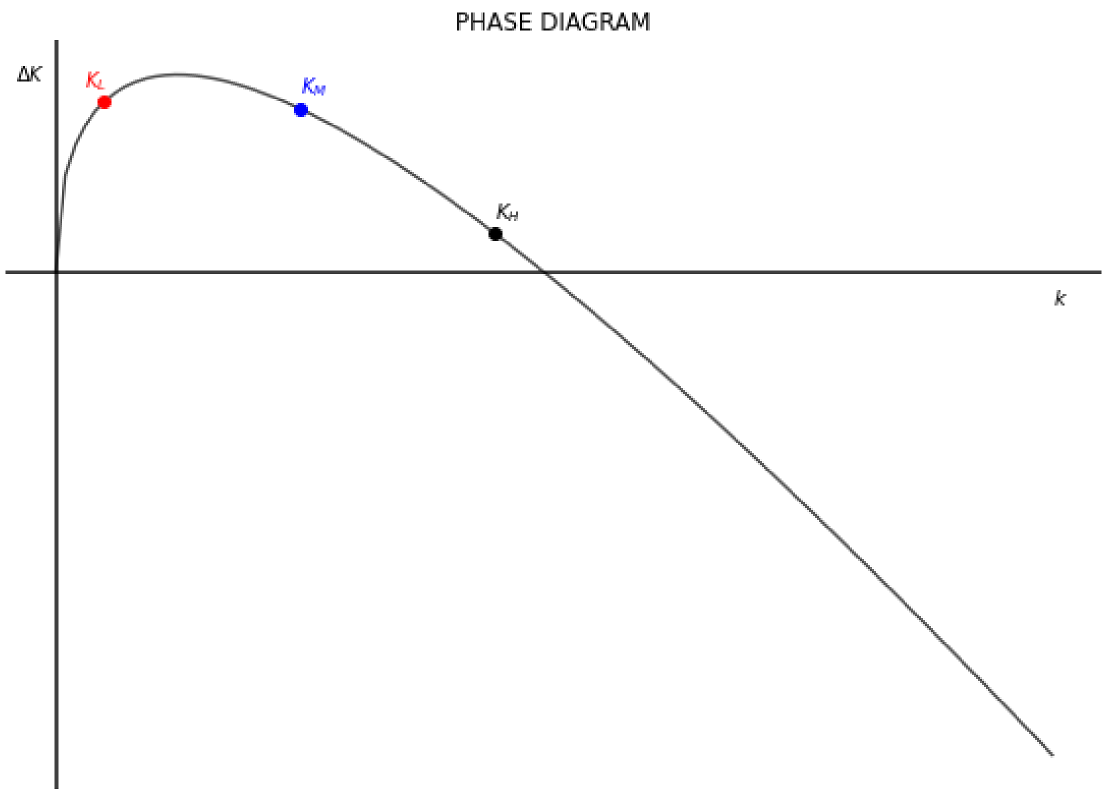


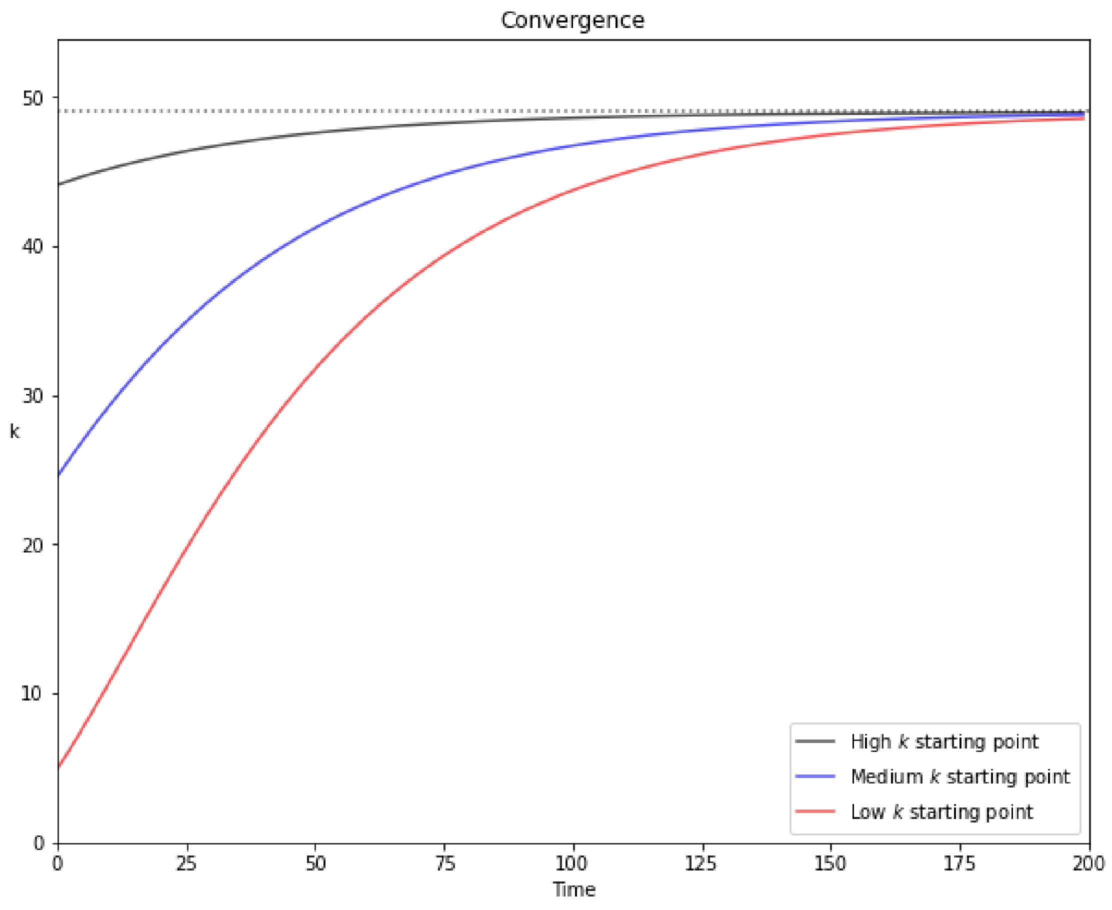
[8]:
```
import numpy as np
import matplotlib.pyplot as plt


K_size = 101
A = 1
alpha = 0.50
delta = 0.03
s = 0.35
n = 0.02
# Arrays
k = np.arange(K_size)
y = np.zeros(K_size)
d = np.zeros(K_size)
i = np.zeros(K_size)
c = np.zeros(K_size)
d_and_i = np.zeros(K_size)

def motion(k):
    DeltaK = s * A*(k)**(alpha) - (n + delta)*k
    return DeltaK

DeltaK = motion(k)



fig, ax1 = plt.subplots(figsize=(10, 7))
ax1.set(title="PHASE DIAGRAM")
ax1.plot(k, DeltaK, "k-", alpha = 0.7)
ax1.yaxis.set_major_locator(plt.NullLocator())
ax1.xaxis.set_major_locator(plt.NullLocator())
ax1.axvline(x=0, color = 'k')
ax1.axhline(y=0, color = 'k')
plt.box(False)
plt.text(100, -0.1, r'$k$')
plt.text( -4,  0.6, r'$\Delta K$')
plt.show()
```



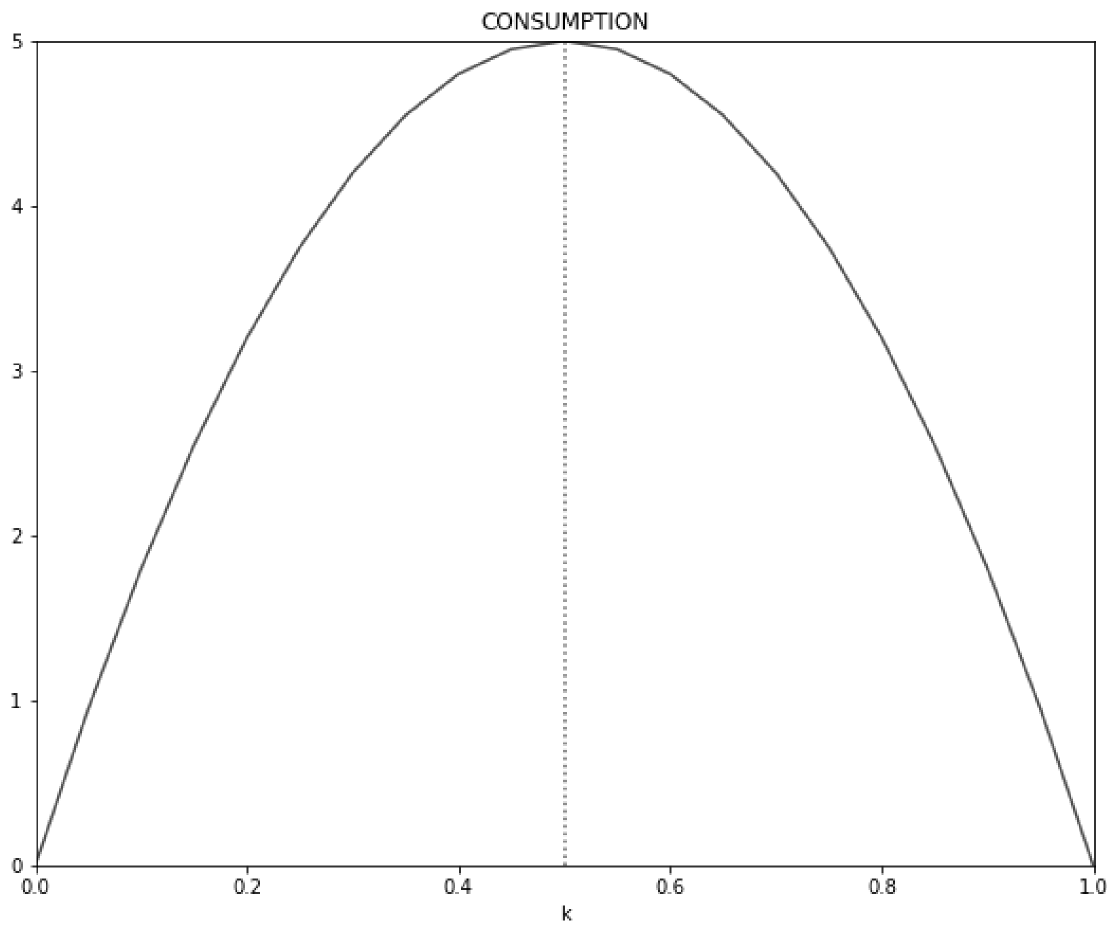
[6]:
```
import numpy as np
import matplotlib.pyplot as plt
```

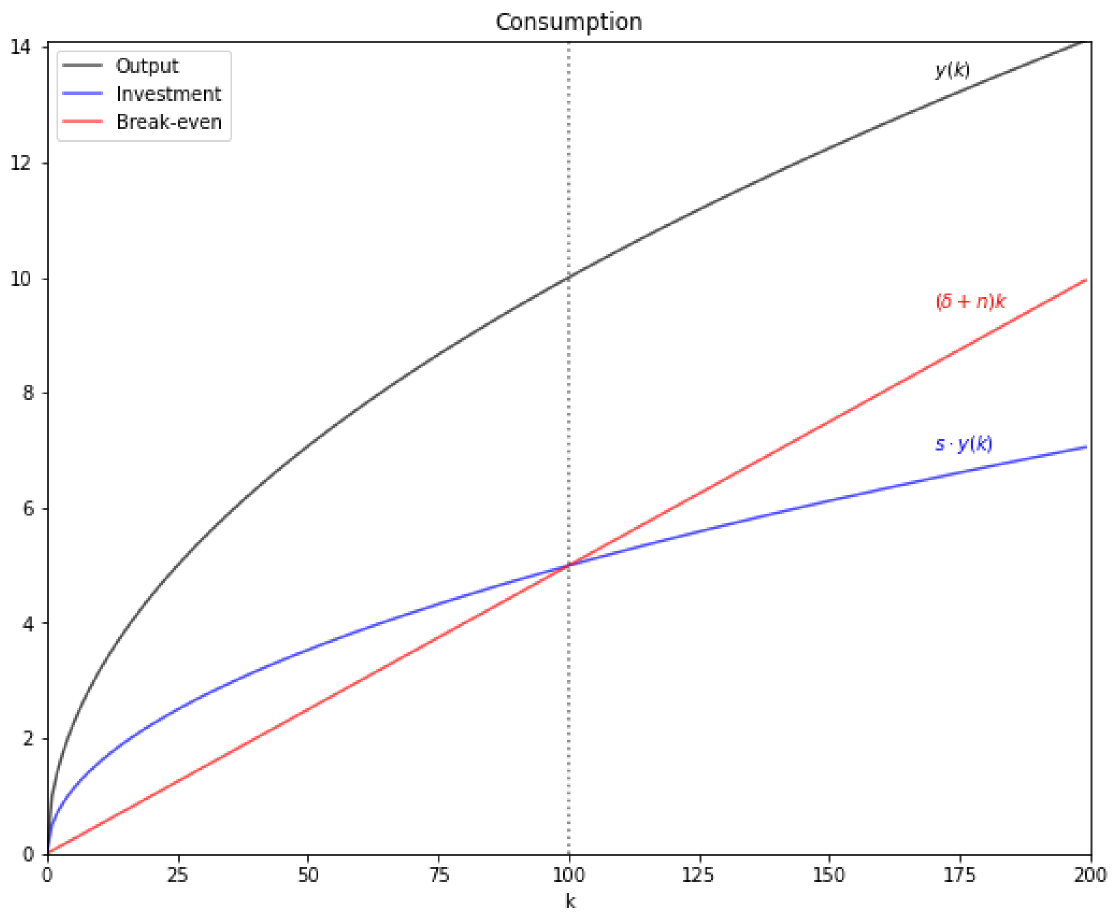
```
K_size = 101
A = 1
alpha = 0.50
delta = 0.03
s = 0.35
n = 0.02

k = np.arange(K_size)


T = 200
time = np.arange(T)
k1 = np.zeros(T)
k2 = np.zeros(T)
k3 = np.zeros(T)

k_star = (s/(n+delta)*A)**(1/(1-alpha))
k1[0] = k_star * 0.9
k2[0] = k_star * 0.5
k3[0] = k_star * 0.1

def output(k):
    y = A * (k)**(alpha)
    return y

def motion(k):
    DeltaK = s * A*(k)**(alpha) - (n + delta)*k
    return DeltaK

DeltaK = motion(k)

Kdelta1 = motion(k1[0])
Kdelta2 = motion(k2[0])
Kdelta3 = motion(k3[0])

fig, ax1 = plt.subplots(figsize=(10, 7))
ax1.set(title="PHASE DIAGRAM")
ax1.plot(k, DeltaK, "k-", alpha = 0.7)
ax1.yaxis.set_major_locator(plt.NullLocator())
ax1.xaxis.set_major_locator(plt.NullLocator())
ax1.axvline(x=0, color = 'k')
ax1.axhline(y=0, color = 'k')
plt.box(False)
plt.text(100, -0.1, r'$k$')
plt.text( -4,  0.6, r'$\Delta K$')
plt.plot(k1[0], Kdelta1, 'ko')
plt.plot(k2[0], Kdelta2, 'bo')
plt.plot(k3[0], Kdelta3, 'ro')
plt.text(k1[0]  , Kdelta1+0.05, r'$K_{H}$', color = 'k')
plt.text(k2[0]  , Kdelta2+0.05, r'$K_{M}$', color = 'b')
plt.text(k3[0]-2, Kdelta3+0.05, r'$K_{L}$', color = 'r')


for j in range(1, T):
    k1[j] = k1[j-1] + s*output(k1[j-1]) - (delta + n)*k1[j-1]
    k2[j] = k2[j-1] + s*output(k2[j-1]) - (delta + n)*k2[j-1]
    k3[j] = k3[j-1] + s*output(k3[j-1]) - (delta + n)*k3[j-1]

v = [0, T, 0, k_star*1.1]
fig, ax = plt.subplots(figsize=(10, 8))
ax.plot(time, k1, "k-", label="High $k$ starting point"   , alpha = 0.7)
ax.plot(time, k2, "b-", label="Medium $k$ starting point", alpha = 0.7)
ax.plot(time, k3, "r-", label="Low $k$ starting point"    , alpha = 0.7)
ax.set(title="Convergence", xlabel=r'$k$')
plt.legend(loc=4)
plt.axhline(y = k_star, ls = ":", color = 'k', alpha = 0.6)
plt.axis(v)
plt.xlabel('Time')
plt.ylabel('k', rotation = 0)
plt.show()
```





[7]:
```
import numpy as np
import matplotlib.pyplot as plt


K_size = 200
A = 1
alpha = 0.50
delta = 0.03
s = 0.35
n = 0.02
# Arrays
k = np.arange(K_size)


def output(k):
    y = A * (k)**(alpha)
    return y

k_gold = ((alpha*A)/(delta+n))**(1/(1-alpha))
y_gold = output(k_gold)
s_gold = ((delta+n)*k_gold)/y_gold
i_gold = s_gold * y_gold
c_gold = y_gold - i_gold
d_gold = delta*k_gold


step = 0.05
size = int(1/step)+1
savings = np.arange(0, 1.01, step)

k_s = (savings / (n+delta)*A)**(1/(1-alpha))
y_s = output(k_s)
c_s = (1-savings)*y_s

v = [0, 1, 0, c_gold]
fig, ax = plt.subplots(figsize=(10, 8))
ax.plot(savings, c_s, "k-", label="Output", alpha = 0.7)
ax.set(title="CONSUMPTION", xlabel=r'$k$')
plt.axvline(x=s_gold, ls=":", color='k', alpha=0.6)
plt.xlabel('k')
plt.axis(v)
plt.show()


y = output(k)
i = s_gold * y
c = (1 - s_gold)*y
d_and_i = (delta + n)*k

v = [0, K_size, 0, y[K_size-1]]
fig, ax = plt.subplots(figsize=(10, 8))
ax.plot(k, y      , "k-", label="Output"     , alpha = 0.7)
ax.plot(k, i      , "b-", label="Investment", alpha = 0.7)
ax.plot(k, d_and_i, "r-", label="Break-even", alpha = 0.7)
ax.set(title="Consumption", xlabel=r'$k$')
plt.legend(loc=2)
plt.axvline(x = k_gold, ls = ":", color = 'k', alpha = 0.6)
plt.xlabel('k')
plt.text(170, 13.5, r'$y(k)$'           , color = 'k')
plt.text(170,  9.5, r'$(\delta + n)k$', color = 'r')
plt.text(170,  7.0, r'$s \cdot y(k)$' , color = 'b')
plt.axis(v)
plt.show()
```


CONSUMPTION
5
4
3
2
1
0
0.0
0.2
0.4
0.6
0.8
1.0
k


Consumption
Output
Investment
Break-even
$y(k)$
$(\delta + n)k$
$s \cdot y(k)$
k
0
2
4
6
8
10
12
14
0
25
50
75
100
125
150
175
200