MAT244 Assignment : Mankiw, Romer, and Weil

August 10, 2018

# 1.

## (a)

First, let $\frac{\mathrm{d}k}{\mathrm{d}t} = 0$. Then $k' = s_k k^\alpha h^\beta - (\delta + n)k = 0$, so $k(s_k k^{\alpha-1} h^\beta - (\delta + n)) = 0$. The critical points are at $k = 0$, or $s_k k^{\alpha-1} h^\beta = \delta + n$. Similarly, if we let $\frac{\mathrm{d}h}{\mathrm{d}t} = 0$, we get $h = 0$ or $s_h k^\alpha h^{\beta-1} = \delta + n$. Letting $\delta + n = s_k k^{\alpha-1} h^\beta = s_h k^\alpha h^{\beta-1}$ makes us to conclude $s_k k^{-1} \cancel{k^\alpha h^\beta} = s_h \cancel{k^\alpha h^\beta} h^{-1} \Rightarrow \frac{s_k}{k} = \frac{s_h}{h} \Rightarrow s_k h = s_h k \Rightarrow \frac{s_k h}{s_h k} = 1$. This implies:

$$
\begin{aligned}
s_k k^{\alpha-1} h^\beta = \delta + n = (\delta + n) \times \frac{s_k h}{s_h k} &= (\delta + n) \times \frac{(s_k)^\beta h^\beta}{(s_h)^\beta k^\beta} \\
\Rightarrow s_k k^{\alpha-1} \cancel{h^\beta} &= (\delta + n)(s_k)^\beta \cancel{h^\beta} (s_h)^{-\beta} k^{-\beta} \\
\Rightarrow \quad s_k k^{\alpha-1} &= (\delta + n)(s_k)^\beta (s_h)^{-\beta} k^{-\beta}
\end{aligned}
$$

Move all the terms from the right hand side to the left except $(s_h)^{-\beta}$ and obtain:

$$
\begin{aligned}
s_k k^{\alpha-1} (\delta + n)^{-1} (s_k)^{-\beta} k^\beta &= (s_h)^{-\beta} \\
\Rightarrow (s_k)^{-1} k^{1-\alpha} (\delta + n)(s_k)^\beta k^{-\beta} &= (s_h)^\beta \\
\Rightarrow \qquad (s_k)^{\beta-1} k^{1-\alpha-\beta} (\delta + n) &= (s_h)^\beta \\
\Rightarrow \qquad (\delta + n) k^{1-\alpha-\beta} &= (s_k)^{1-\beta} (s_h)^\beta \\
\Rightarrow \qquad k^{1-\alpha-\beta} &= \frac{(s_k)^{1-\beta} (s_h)^\beta}{\delta + n} \\
\Rightarrow \qquad k &= \left( \frac{(s_k)^{1-\beta} (s_h)^\beta}{\delta + n} \right)^{\frac{1}{1-\alpha-\beta}}
\end{aligned}
$$

Similarly, $h = \left( \frac{(s_k)^\alpha (s_h)^{1-\alpha}}{\delta + n} \right)^{\frac{1}{1-\alpha-\beta}}$. Therefore, the critical points $(k, h)$ are at $(0, 0)$, and

$$
(k_1, h_1) := \left( \left( \frac{(s_k)^{1-\beta} (s_h)^\beta}{\delta + n} \right)^{\frac{1}{1-\alpha-\beta}}, \left( \frac{(s_k)^\alpha (s_h)^{1-\alpha}}{\delta + n} \right)^{\frac{1}{1-\alpha-\beta}} \right)
$$

## (b)

Let $J_F$ be the Jacobian matrix of $F(k,h) = (k', h')$. The Taylor expansion gives us $F(k,h)$ $\approx F(k_0,h_0) + J_F(k_0,h_0)\begin{bmatrix} k-k_0 \\ h-h_0 \end{bmatrix}$, where $(k_0,h_0)$ is some arbitrary fixed point. Since $J_F$ is equal to $\begin{bmatrix} \frac{\partial k'}{\partial k} & \frac{\partial k'}{\partial h} \\ \frac{\partial h'}{\partial k} & \frac{\partial h'}{\partial h} \end{bmatrix}$, we compute the partial derivatives:

$$\begin{aligned}
\frac{\partial k'}{\partial k} &= \frac{\partial}{\partial k}\Big(s_k k^\alpha h^\beta - (\delta+n)k\Big) = h^\beta\Big(\alpha s_k k^{\alpha-1} + k^\alpha (s_k)'\Big) - (\delta+n) \\
&= \alpha s_k k^{\alpha-1} h^\beta + (s_k)' k^\alpha h^\beta - (\delta+n) \\
\frac{\partial k'}{\partial h} &= \frac{\partial}{\partial h}\Big(s_k k^\alpha h^\beta - (\delta+n)k\Big) = \beta s_k k^\alpha h^{\beta-1} \\
\frac{\partial h'}{\partial k} &= \frac{\partial}{\partial k}\Big(s_h k^\alpha h^\beta - (\delta+n)h\Big) = \alpha s_h k^{\alpha-1} h^\beta \\
\frac{\partial h'}{\partial h} &= \frac{\partial}{\partial h}\Big(s_h k^\alpha h^\beta - (\delta+n)h\Big) = k^\alpha\Big(\beta s_h h^{\beta-1} + h^\beta (s_h)'\Big) - (\delta+n) \\
&= \beta s_h k^\alpha h^{\beta-1} + (s_h)' k^\alpha h^\beta - (\delta+n)
\end{aligned}$$

Assuming $s_k$ and $s_h$ as constants, we get these partial derivatives and $J_F$:

$$\begin{aligned}
&\frac{\partial k'}{\partial k} = \alpha k^{-1}(s_k k^\alpha h^\beta) - (\delta+n), \quad && \frac{\partial k'}{\partial h} = \beta h^{-1}(s_k k^\alpha h^\beta) \\
&\frac{\partial h'}{\partial k} = \alpha k^{-1}(s_h k^\alpha h^\beta), && \frac{\partial h'}{\partial h} = \beta h^{-1}(s_h k^\alpha h^\beta) - (\delta+n)
\end{aligned}$$

$$\Rightarrow J_F(k_0,h_0) = \begin{bmatrix} \alpha(k_0)^{-1}[s_k(k_0)^\alpha(h_0)^\beta] - (\delta+n) & \beta(h_0)^{-1}[s_k(k_0)^\alpha(h_0)^\beta] \\ \alpha(k_0)^{-1}[s_h(k_0)^\alpha(h_0)^\beta] & \beta(h_0)^{-1}[s_h(k_0)^\alpha(h_0)^\beta] - (\delta+n) \end{bmatrix}_{2\times 2}$$

So $F(k_0,h_0) = \begin{bmatrix} s_k(k_0)^\alpha(h_0)^\beta - (\delta+n)(k_0) \\ s_h(k_0)^\alpha(h_0)^\beta - (\delta+n)(h_0) \end{bmatrix}_{2\times 1}$, and $J_F(k_0,h_0)\begin{bmatrix} k-k_0 \\ h-h_0 \end{bmatrix}$ should be:

$$\begin{bmatrix} s_k(k_0)^\alpha(h_0)^\beta\Big[\alpha(k_0)^{-1}(k-k_0) + \beta(h_0)^{-1}(h-h_0)\Big] - (\delta+n)(k-k_0) \\ s_h(k_0)^\alpha(h_0)^\beta\Big[\alpha(k_0)^{-1}(k-k_0) + \beta(h_0)^{-1}(h-h_0)\Big] - (\delta+n)(h-h_0) \end{bmatrix}_{2\times 1}$$

Define $A_{k_0} := s_k(k_0)^\alpha(h_0)^\beta$, and $A_{h_0} := s_h(k_0)^\alpha(h_0)^\beta$. Then $F(k_0,h_0) + J_F(k_0,h_0)\begin{bmatrix} k-k_0 \\ h-h_0 \end{bmatrix}$

$$= \begin{bmatrix} \alpha(k_0)^{-1}A_{k_0} - (\delta+n) & \beta(h_0)^{-1}A_{k_0} \\ \alpha(k_0)^{-1}A_{h_0} & \beta(h_0)^{-1}A_{h_0} - (\delta+n) \end{bmatrix}\begin{bmatrix} k \\ h \end{bmatrix} + \begin{bmatrix} A_{k_0}(1-\alpha-\beta) \\ A_{h_0}(1-\alpha-\beta) \end{bmatrix}$$

## (c)

If $(k_0, h_0)$ is the critical point of $F$, then $F(k_0, h_0) = (k', h') = 0$. So the linearization around the critical point gives $(k', h') = J_F(k_0, h_0) \begin{bmatrix} k - k_0 \\ h - h_0 \end{bmatrix}$. We know that $J_F(k_0, h_0)$ is:

$$\begin{bmatrix} \alpha(k_0)^{-1} A_{k_0} - (\delta + n) & \beta(h_0)^{-1} A_{k_0} \\ \alpha(k_0)^{-1} A_{h_0} & \beta(h_0)^{-1} A_{h_0} - (\delta + n) \end{bmatrix}_{2 \times 2}$$

Define $S := \alpha(k_0)^{-1} A_{k_0} + \beta(h_0)^{-1} A_{h_0}$. Then the eigenvalues are $(i = 1, 2)$:

$$\lambda_i = \frac{S - 2(\delta + n)}{2} \pm \frac{1}{2} \sqrt{\Big(S - 2(\delta + n)\Big)^2 - 4\Big((\delta + n)^2 - S(\delta + n)\Big)}$$

Notice that terms in the square root simplifies to:

$$S^2 - \cancel{4S(\delta + n)} + \cancel{4(\delta + n)^2} - \cancel{4(\delta + n)^2} + \cancel{4S(\delta + n)} = S^2$$

That is, $\lambda_i = \frac{S - 2(\delta + n)}{2} \pm \frac{S}{2}$ since $S > 0$. $S$ has to be strictly positive because:

1. "$(k_0)^{\alpha}$" and "$(h_0)^{\beta}$" terms don't make sense when $k_0$ and $h_0$ are negative, and
2. $S$ contains "$(k_0)^{-1}$" and "$(h_0)^{-1}$" terms, so $k_0$ and $h_0$ cannot be zero;
3. we assume $s_k, s_h, \alpha$, and $\beta$ are all positive.

Thus, $\lambda_1 = \frac{\cancel{S} - \cancel{2}(\delta + n) - \cancel{S}}{\cancel{2}} = -(\delta + n) < 0$ (regardless of $k_0$ and $h_0$), and $\lambda_2 = \frac{S - 2(\delta + n) + S}{2}$ $= S - (\delta + n)$. We observe that eigenvalues are distinct real ($\because S \neq 0$), and one of them ($\lambda_1$) is always negative.

**Behaviour near** $(0, 0)$

As $(k_0, h_0) \to (0^+, 0^+)$, we have:

$$\begin{aligned} S &= \alpha(k_0)^{-1} A_{k_0} + \beta(h_0)^{-1} A_{h_0} \\ &= \alpha s_k (k_0)^{\alpha - 1} (h_0)^{\beta} + \beta s_h (k_0)^{\alpha} (h_0)^{\beta - 1} \\ &= (k_0)^{\alpha} (h_0)^{\beta} \Big( \alpha s_k (k_0)^{-1} + \beta s_h (h_0)^{-1} \Big) \to \infty \end{aligned}$$

since $\alpha$ and $\beta$ are both less than 1 which makes "$(k_0)^{-1}$" and "$(h_0)^{-1}$" dominating terms of $S$. This explains why linearization *at* $(0, 0)$ is not possible: the linearization of $F$ goes to $(\infty, \infty)$ as we try to linearize at the point closer to $(k_0, h_0) \approx (0^+, 0^+)$. So we get $\lambda_1 < 0$ and $\lambda_2 > 0$ near $(0, 0)$; it is the saddle point of $F$.

**Behaviour near** $(k_1, h_1)$

At the critical point $(k_1, h_1)$ shown in (a), we know that $s_k(k_1)^{\alpha-1}(h_1)^{\beta} = s_h(k_1)^{\alpha}(h_1)^{\beta-1}$ $= \delta + n$. That is, $A_{k_1} = s_k(k_1)^{\alpha}(h_1)^{\beta} = k_1(\delta+n)$, and $A_{h_1} = s_h(k_1)^{\alpha}(h_1)^{\beta} = h_1(\delta+n)$. Thus,

$$J_F(k_1, h_1) = \begin{bmatrix} (\alpha-1)(\delta+n) & \beta(h_1)^{-1}k_1(\delta+n) \\ \alpha(k_1)^{-1}h_1(\delta+n) & (\beta-1)(\delta+n) \end{bmatrix}_{2\times 2}$$

Recall from (a) that at this critical point, $s_k h_1 = s_h k_1$. So $\frac{k_1}{h_1} = \frac{s_k}{s_h}$, and $\frac{h_1}{k_1} = \frac{s_h}{s_k}$. Hence,

$$J_F(k_1, h_1) = \begin{bmatrix} (\alpha-1)(\delta+n) & \frac{\beta s_k(\delta+n)}{s_h} \\ \frac{\alpha s_h(\delta+n)}{s_k} & (\beta-1)(\delta+n) \end{bmatrix}_{2\times 2}$$

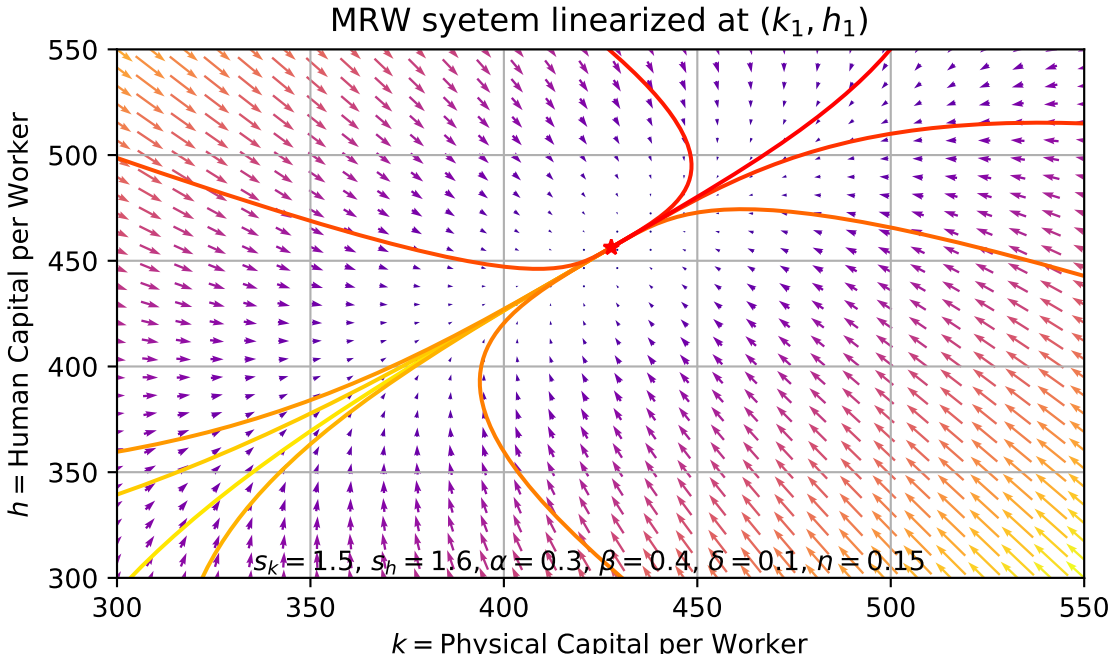
And therefore the linearization of $(k', h')$ at the critical point $(k_1, h_1)$ is:

$$\begin{aligned} J_F(k_1, h_1)\begin{bmatrix} k-k_1 \\ h-h_1 \end{bmatrix} &= \begin{bmatrix} (\alpha-1)(\delta+n) & \frac{\beta s_k(\delta+n)}{s_h} \\ \frac{\alpha s_h(\delta+n)}{s_k} & (\beta-1)(\delta+n) \end{bmatrix}\begin{bmatrix} k-k_1 \\ h-h_1 \end{bmatrix} \\ &= (\delta+n)\begin{bmatrix} (\alpha-1) & \frac{\beta s_k}{s_h} \\ \frac{\alpha s_h}{s_k} & (\beta-1) \end{bmatrix}\begin{bmatrix} k-k_1 \\ h-h_1 \end{bmatrix} \\ &= (\delta+n)\Bigg(\begin{bmatrix} (\alpha-1) & \frac{\beta s_k}{s_h} \\ \frac{\alpha s_h}{s_k} & (\beta-1) \end{bmatrix}\begin{bmatrix} k \\ h \end{bmatrix} + \begin{bmatrix} (1-\alpha)(k_1) + (-\frac{\beta s_k}{s_h})(h_1) \\ (-\frac{\alpha s_h}{s_k})(k_1) + (1-\beta)(h_1) \end{bmatrix}\Bigg) \end{aligned}$$

Eigenvalues of $J_F(k_1, h_1)$ show that $(k_1, h_1)$ is the stable node because:

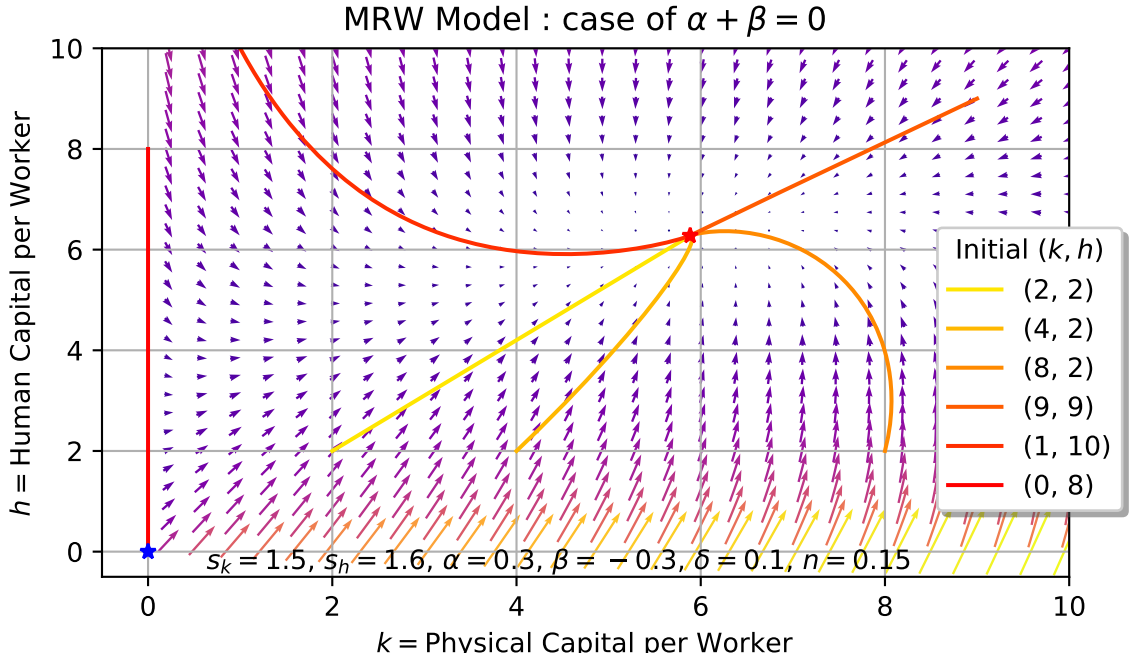
$$\begin{aligned} \lambda_1 &= -(\delta+n) < 0, \text{ and} \\ \lambda_2 &= S - (\delta+n) \\ &= \alpha\cancel{(k_1)^{-1}k_1}(\delta+n) + \beta\cancel{(h_1)^{-1}h_1}(\delta+n) - (\delta+n) \\ &= (\alpha+\beta-1)(\delta+n) < 0 \ \ \because \alpha+\beta \in (0,1) \end{aligned}$$

The original $F$ tells us that $F(0, h_0) = (0, -(\delta+n)h_0)$, $F(k_0, 0) = (-(\delta+n)k_0, 0)$, and $F(0,0) = (0,0)$, i.e. if either $k_0$ or $h_0$ of the initial value point $(k_0, h_0)$ is 0 (but not both), then trajectories converge to $(0,0)$ at a constant rate of $\delta + n$, and stays at $(0,0)$ if the initial value is the origin. If neither $k_0$ nor $h_0$ is 0, then trajectories converge to $(k_1, h_1)$. The plot below (MRW system linearized at $(k_1, h_1)$) coincides with our analysis on the stable node, and the first plot of 1.(e) shows some trajectories and a direction field of the original $F(k, h) = (k', h')$. Notice that in the first plot of 1.(e), the origin is indeed the saddle point, and $(k_1, h_1)$ is the stable node of $F$.

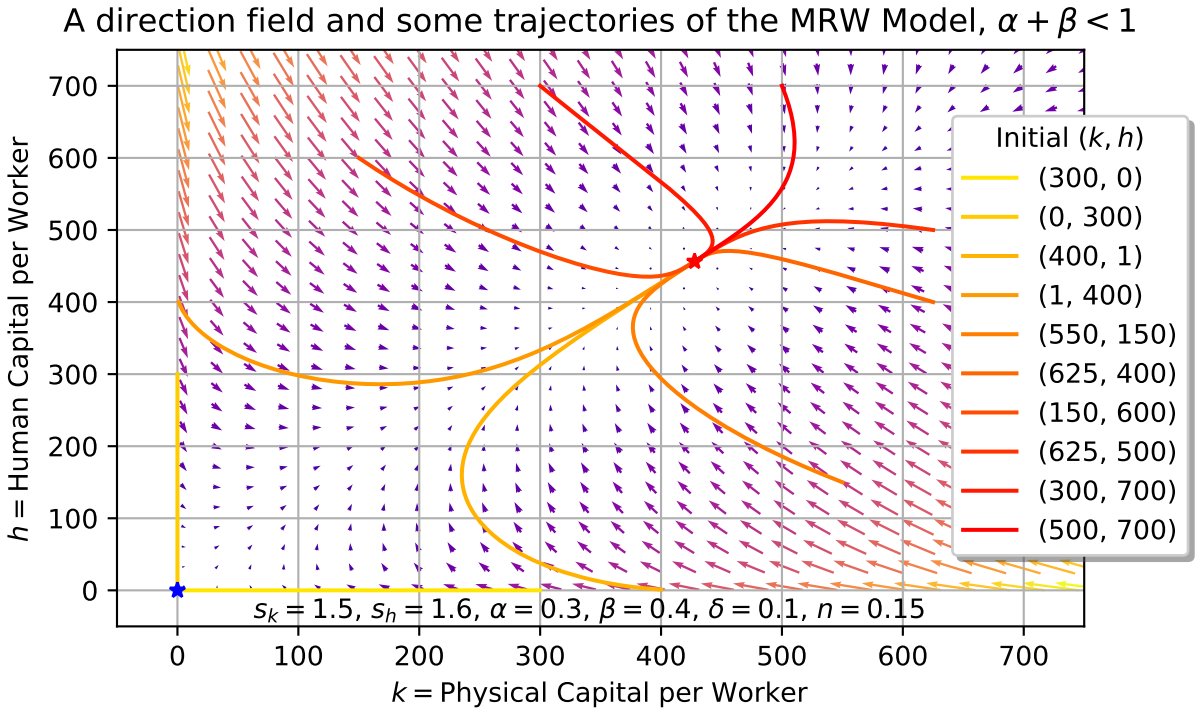


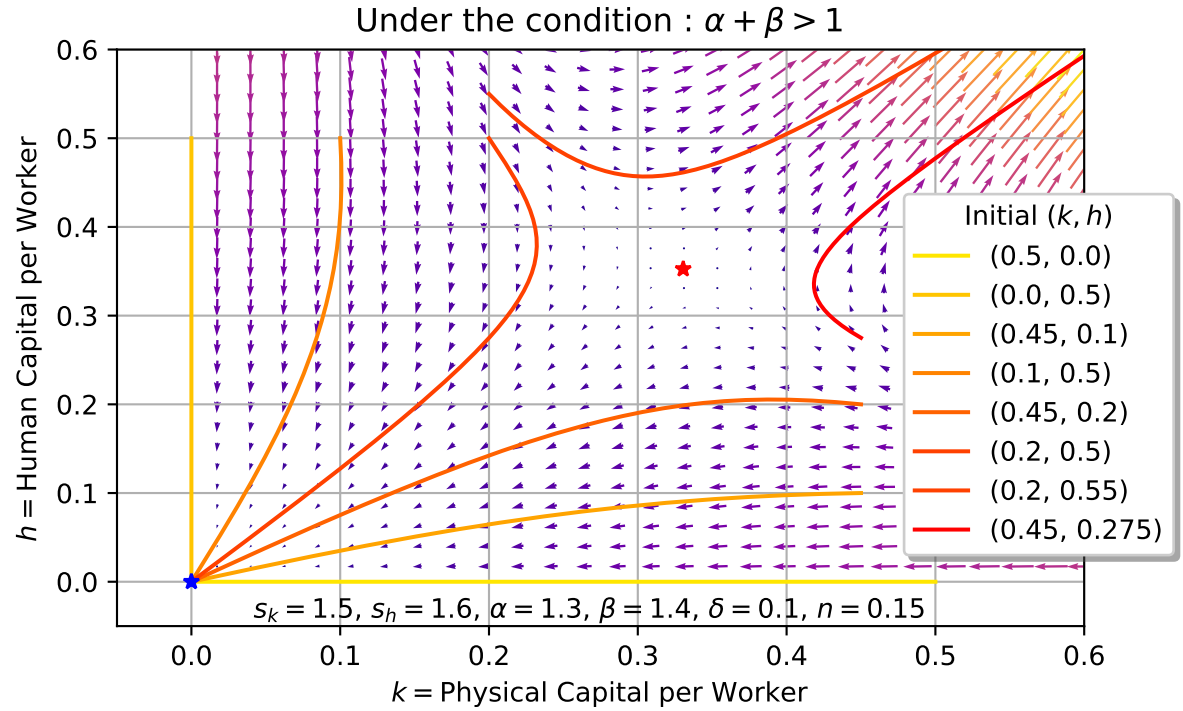

## (d)

For every combination of $\alpha$ and $\beta$ where we have $\alpha + \beta < 1$, our original stable node stays stable. If $\alpha + \beta = 0$, we get repeated eigenvalues $\lambda = -(\delta + n)$; $(k_1, h_1)$ should still be stable. For combinations that yield $\alpha + \beta > 1$, our original stable node becomes saddle since $\lambda_2 > 0$ in those cases. If we have a combination where $\alpha + \beta = 1$, then $(k_1, h_1)$ is not even defined.



(e)


A direction field and some trajectories of the MRW Model, $\alpha + \beta < 1$
$h$ = Human Capital per Worker
$k$ = Physical Capital per Worker
$s_k = 1.5$, $s_h = 1.6$, $\alpha = 0.3$, $\beta = 0.4$, $\delta = 0.1$, $n = 0.15$
Initial $(k, h)$
(300, 0)
(0, 300)
(400, 1)
(1, 400)
(550, 150)
(625, 400)
(150, 600)
(625, 500)
(300, 700)
(500, 700)


Under the condition : $\alpha + \beta > 1$
$h$ = Human Capital per Worker
$k$ = Physical Capital per Worker
$s_k = 1.5$, $s_h = 1.6$, $\alpha = 1.3$, $\beta = 1.4$, $\delta = 0.1$, $n = 0.15$
Initial $(k, h)$
(0.5, 0.0)
(0.0, 0.5)
(0.45, 0.1)
(0.1, 0.5)
(0.45, 0.2)
(0.2, 0.5)
(0.2, 0.55)
(0.45, 0.275)

# 2.

## (a)

Note that $Y_t = K_t^{\alpha} L_t^{1-\alpha} = (k(t))^{\alpha} L_t$ since $k := \frac{K}{L}$. Thus,

$$\begin{aligned}
\frac{\mathrm{d}k}{\mathrm{d}t} &= \frac{L_t K_t' - K_t L_t'}{L_t^2} \\
&= \lim_{\Delta t \to 0} \frac{L_t \times \frac{K_{t+\Delta t} - K_t}{\Delta t} - K_t \times \frac{L_{t+\Delta+t} - L_t}{\Delta t}}{L_t^2} \\
&= \frac{\cancel{L_t}\Big(s_k Y_t - \delta K_t\Big) - K_t\Big(n\cancel{L_t}\Big)}{L_t^{\cancel{2}}} \\
&= \frac{s_k Y_t - \delta K_t}{L_t} - \frac{n K_t}{L_t} \\
&= s_k \times \frac{k_t^{\alpha} \cancel{L_t}}{\cancel{L_t}} - (\delta + n) k(t) \\
&= \quad \therefore s_k k^{\alpha} - (\delta + n) k
\end{aligned}$$

## (b)

The equilibrium point is the critical point of $f(k) = k'$. If we let $k' = 0$, we get $s_k k^{\alpha} - (\delta + n)k = 0 \;\Rightarrow\; k = 0$ or $s_k k^{\alpha - 1} = \delta + n$. That is, either $k = 0$ or $k = (\frac{s_k}{\delta + n})^{1/(1-\alpha)}$.

The Taylor expansion of $f$ gives us $f(k) \approx f(k_0) + f'(k_0)(k - k_0)$, where $k_0$ is some arbitrary fixed point. Assuming $s_k$ as a constant, we get $f'(k) = \frac{\partial k'}{\partial k} = \alpha s_k k^{\alpha - 1} - (\delta + n)$. So the linearization of $f$ at the critical point $k_0$ should be $f(k) = 0 + (\alpha s_k k_0^{\alpha - 1} - (\delta + n))(k - k_0)$. Notice that linearization is not possible at $k_0 = 0$ since $f' \to \infty$ as $k \to 0^+$. We should linearize at $k_0 = k_1 = (\frac{s_k}{\delta + n})^{1/(1-\alpha)}$. At $k_1$, we have $s_k k_1^{\alpha - 1} = \delta + n$, so $f(k) = f'(k_1)(k - k_1)$ $= \Big(\alpha(\delta + n) - (\delta + n)\Big)(k - k_1) = (\alpha - 1)(\delta + n)(k - k_1)$.

Therefore, the rate of convergence is $|(\alpha - 1)(\delta + n)| = |(1 - \alpha)(\delta + n)|$.

## (c)

Clearly $-(\delta + n) < (\alpha + \beta - 1)(\delta + n)$ since $\alpha + \beta - 1 \in (-1, 0)$, and $(\delta + n) > 0$. So $(\alpha + \beta - 1)(\delta + n)$ is the largest eigenvalue of the MRW model around the equilibrium point. It is also clear that $|(\alpha + \beta - 1)(\delta + n)| = |(1 - \alpha - \beta)(\delta + n)| < |(1 - \alpha)(\delta + n)|$ $\because |(1 - \alpha - \beta)| < |(1 - \alpha)|$.

## (d)

MRW system is derived from a Cobb-Douglas production function:

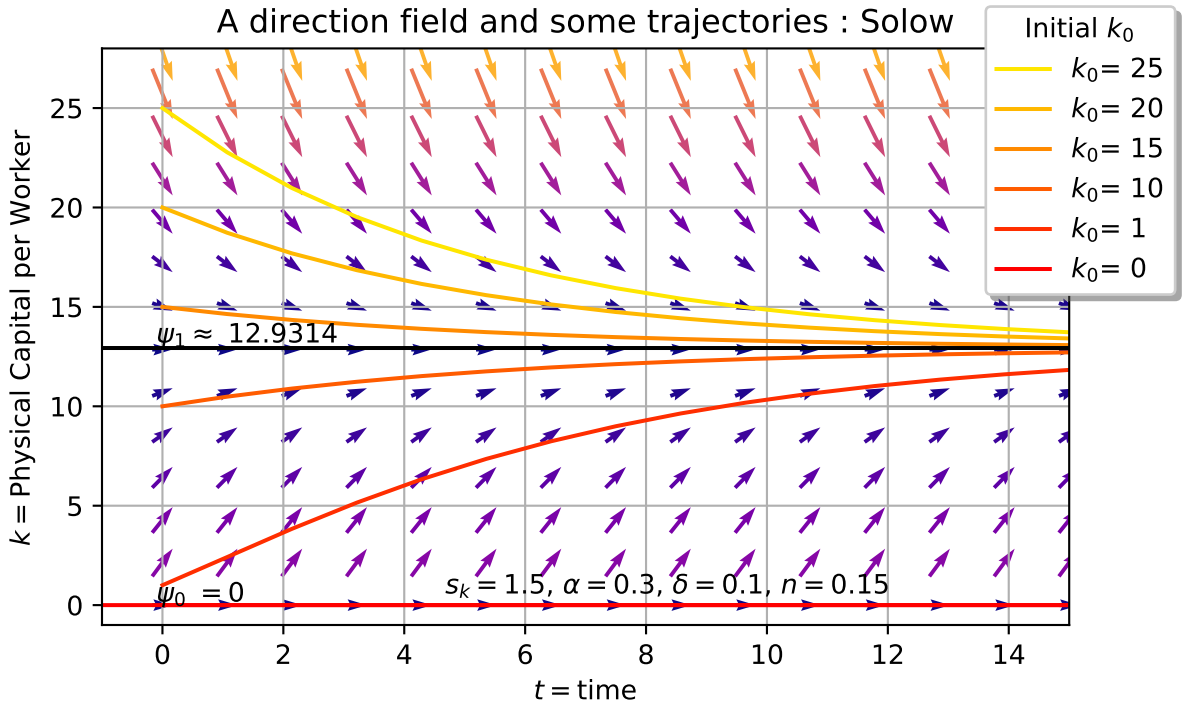
$$Y_t = K_t^{\alpha} H_t^{\beta} L_t^{1-\alpha-\beta} \Rightarrow \ \log(Y_t) = \alpha \log(K_t) + \beta \log(H_t) + (1-\alpha-\beta)\log(L_t)$$

Similarly, the Solow model is from $\log(Y_t) = \alpha \log(K_t) + (1-\alpha)\log(L_t)$. We observe that both models are designed to have parameters $(\alpha, \beta, 1-\alpha, \text{ and } 1-\alpha-\beta)$ summing up to one, i.e. both models assume their predictors $(K_t, H_t, L_t)$ perfectly explain the production $(Y_t)$, and have no interaction effects (mutually independent).

Fix $\alpha > 0$. Then since $\beta > 0$, we notice that the MRW model takes away some proportion originally explained by the population growth $(L_t)$ in the Solow model, and claims that that much of proportion (namely $\beta$) of $\log(Y_t)$ is actually attributable to the human capital $(H_t)$. Since the MRW model has more (log-transformed) predictors to explain the (log-transformed) response, and takes each parameter sum up to one, it is clear that the effects of each predictor decrease as more predictors are introduced in the model. That is, the rate of convergence of the MRW model is slower because the accumulation of human capital offsets the impact of the population growth.

## (e)

The MRW model is already shown in the first plot of 1.(e).



# Appendix

Below are codes used to generate plots in 1.(c), 1.(d), 1.(e), and 2.(e), in this order. We used `Python` to draw direction fields and trajectories:

## Code used in 1.(c)

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate
# Ignore warnings when dividing by zero
np.seterr(divide='ignore', invalid='ignore')
# Define parameters
s_k, s_h = 1.5, 1.6
alpha, beta = 0.3, 0.4
delta, n = 0.1, 0.15
# Define the non-trivial equilibrium point
EQ = ([((((s_k)**(1-beta))*((s_h)**(beta)))/(delta+n))**(1/(1-alpha-beta)),
      ((((s_k)**(alpha))*((s_h)**(1-alpha)))/(delta+n))**(1/(1-alpha-beta))
      ])

k_1, h_1 = EQ[0], EQ[1]
k_0, h_0 = k_1, h_1
A_k_0 = s_k * (k_0 ** alpha) * (h_0 ** beta)
A_h_0 = s_h * (k_0 ** alpha) * (h_0 ** beta)
# Define our system
def linearized_mrw_model(P, t = 0):
    k, h = P[0], P[1]
    return ([(alpha * (k_0 ** (-1)) * A_k_0 - (delta + n)) * k +\
             (beta * (h_0 ** (-1)) * A_k_0) * h +\
             A_k_0 * (1 - alpha - beta),
             (alpha * (k_0 ** (-1)) * A_h_0) * k +\
             (beta * (h_0 ** (-1)) * A_h_0 - (delta + n)) * h +\
             A_h_0 * (1 - alpha - beta)
            ])
# Set range of t and plot limits
f = plt.figure()
t = np.linspace(0, 150, 1000)
xlim = 300
ylim = 550
# Set initial values to start
P0 = np.array([(300, 0), (0, 300), (400, 1), (1, 400),
               (550, 150), (625, 400), (150, 600),
```

```
                (625, 500), (300, 700), (500, 700)])
vcolors = plt.cm.autumn_r(np.linspace(0.1, 1, len(P0)))
# Calculate trajectories
for i in range(len(P0)):
    P = integrate.odeint(linearized_mrw_model, P0[i], t)
    plt.plot(P[:,0], P[:,1],
             color = vcolors[i])
# Determine the number of arrows in the direction field
nb_points = 30
# Set limits of x and y that the plot shows
x = np.linspace(xlim, ylim, nb_points)
y = np.linspace(xlim, ylim, nb_points)
# Create meshgrid
X1 , Y1 = np.meshgrid(x, y)
# Calculate growth rate at each grid point
DX1, DY1 = linearized_mrw_model([X1, Y1])
# Direction at each grid point is the hypotenuse of the direction of k
# and the direction of h
M = (np.hypot(DX1, DY1))
# Give a title to the plot
plt.title('MRW syetem linearized at ' + '$(k_1, h_1)$')
# Use the quiver function to plot the direction field using
# DX1 and DY1 for directions, and M for speed
Q = plt.quiver(X1, Y1, DX1, DY1, M, pivot = 'mid', cmap = plt.cm.plasma)
plt.xlabel('$k =$' + 'Physical Capital per Worker')
plt.ylabel('$h =$' + 'Human Capital per Worker')
plt.grid()
plt.xlim(xlim, ylim)
plt.ylim(xlim, ylim)
# Plot critical (i.e. equilibrium) points
# plt.plot(0, 0, "b*")
plt.plot(EQ[0], EQ[1], "r*")
# Write down the caption
plt.figtext(.775, .12, "$s_k = {0}$, ".format(s_k) +\
            "$s_h = {0}$, ".format(s_h) +\
            "$\\alpha = {0}$, ".format(alpha) +\
            "$\\beta = {0}$, ".format(beta) +\
            "$\\delta = {0}$, ".format(delta) +\
            "$n = {0}$".format(n),
            horizontalalignment = "right", fontsize = 10)
# Plot what we've calculated
plt.show()
```

## Code used in 1.(d)

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate
# Ignore warnings when dividing by zero
np.seterr(divide='ignore', invalid='ignore')
# Define parameters
s_k, s_h = 1.5, 1.6
alpha,beta = 0.3, -0.3
delta, n = 0.1, 0.15
# Define our system
def mrw_model(P, t = 0):
    k, h = P[0], P[1]
    return ([s_k * (k ** alpha) * (h ** beta) - (delta + n) * k,
             s_h * (k ** alpha) * (h ** beta) - (delta + n) * h])
# Define the non-trivial equilibrium point
EQ = ([((((s_k)**(1-beta))*((s_h)**(beta)))/(delta+n))**(1/(1-alpha-beta)),
      ((((s_k)**(alpha))*((s_h)**(1-alpha)))/(delta+n))**(1/(1-alpha-beta))
      ])
# Set range of t
f = plt.figure()
t = np.linspace(0, 150, 1000)
# Set initial values to start
P0 = np.array([(2, 2), (4, 2), (8, 2), (9, 9), (1, 10), (0, 8)])
vcolors = plt.cm.autumn_r(np.linspace(0.1, 1, len(P0)))
# Calculate trajectories
for i in range(len(P0)):
    P = integrate.odeint(mrw_model, P0[i], t)
    plt.plot(P[:,0], P[:,1],
             color = vcolors[i],
             label = '({0}, {1})'.format(P0[i, 0], P0[i, 1]))
# Determine the number of arrows in the direction field
nb_points = 30
# Set limits of x and y that the plot shows
x = np.linspace(-.5, 10, nb_points)
y = np.linspace(-.5, 10, nb_points)
# Create meshgrid
X1 , Y1 = np.meshgrid(x, y)
# Calculate growth rate at each grid point
DX1, DY1 = mrw_model([X1, Y1])
# Direction at each grid point is the hypotenuse of the direction of k
# and the direction of h
M = (np.hypot(DX1, DY1))
```

```
# Give a title to the plot
plt.title('MRW Model : case of ' + '$\\alpha + \\beta = 0$')
# Use the quiver function to plot the direction field using
# DX1 and DY1 for directions, and M for speed
Q = plt.quiver(X1, Y1, DX1, DY1, M, pivot = 'mid', cmap = plt.cm.plasma)
plt.xlabel('$k =$' + 'Physical Capital per Worker')
plt.ylabel('$h =$' + 'Human Capital per Worker')
plt.legend(loc = 'lower left', bbox_to_anchor = (.85, .1),
           ncol = 1, fancybox = True, shadow = True,
           title = "Initial $(k, h)$")
plt.grid()
plt.xlim(-.5, 10)
plt.ylim(-.5, 10)
# Plot critical (i.e. equilibrium) points
plt.plot(0, 0, "b*")
plt.plot(EQ[0], EQ[1], "r*")
# Write down the caption
plt.figtext(.775, .12, "$s_k = {0}$, ".format(s_k) +\
            "$s_h = {0}$, ".format(s_h) +\
            "$\\alpha = {0}$, ".format(alpha) +\
            "$\\beta = {0}$, ".format(beta) +\
            "$\\delta = {0}$, ".format(delta) +\
            "$n = {0}$".format(n),
            horizontalalignment = "right", fontsize = 10)
# Plot what we've calculated
plt.show()
```

## Code used in 1.(e)

For the case $\alpha + \beta < 1$:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate
# Ignore warnings when dividing by zero
np.seterr(divide='ignore', invalid='ignore')
# Define parameters
s_k, s_h = 1.5, 1.6
alpha, beta = 0.3, 0.4
delta, n = 0.1, 0.15
# Define our system
def mrw_model(P, t = 0):
    return ([s_k * (P[0] ** alpha) * (P[1] ** beta) - (delta + n) * P[0],
```

```
            s_h * (P[0] ** alpha) * (P[1] ** beta) - (delta + n) * P[1]])
# Define the non-trivial equilibrium point
EQ = ([((((s_k)**(1-beta))*((s_h)**(beta)))/(delta+n))**(1/(1-alpha-beta)),
      ((((s_k)**(alpha))*((s_h)**(1-alpha)))/(delta+n))**(1/(1-alpha-beta))
      ])
# Set range of t
f = plt.figure()
t = np.linspace(0, 150, 1000)
# Set initial values to start
P0 = np.array([(300, 0), (0, 300), (400, 1), (1, 400),
               (550, 150), (625, 400), (150, 600),
               (625, 500), (300, 700), (500, 700)])
vcolors = plt.cm.autumn_r(np.linspace(0.1, 1, len(P0)))
# Calculate trajectories
for i in range(len(P0)):
    P = integrate.odeint(mrw_model, P0[i], t)
    plt.plot(P[:,0], P[:,1],
             color = vcolors[i],
             label = '({0}, {1})'.format(P0[i, 0], P0[i, 1]))
# Determine the number of arrows in the direction field
nb_points = 30
# Set limits of x and y that the plot shows
x = np.linspace(-50, 750, nb_points)
y = np.linspace(-50, 750, nb_points)
# Create meshgrid
X1 , Y1 = np.meshgrid(x, y)
# Calculate growth rate at each grid point
DX1, DY1 = mrw_model([X1, Y1])
# Direction at each grid point is the hypotenuse of the direction of k
# and the direction of h
M = (np.hypot(DX1, DY1))
# Give a title to the plot
plt.title('A direction field and some trajectories of the MRW Model, ' +\
          '$\\alpha + \\beta < 1$')
# Use the quiver function to plot the direction field using
# DX1 and DY1 for directions, and M for speed
Q = plt.quiver(X1, Y1, DX1, DY1, M, pivot = 'mid', cmap = plt.cm.plasma)
plt.xlabel('$k =$' + 'Physical Capital per Worker')
plt.ylabel('$h =$' + 'Human Capital per Worker')
plt.legend(loc = 'lower left', bbox_to_anchor = (.85, .1),
           ncol = 1, fancybox = True, shadow = True,
           title = "Initial $(k, h)$")
plt.grid()
plt.xlim(-50, 750)
```

```
plt.ylim(-50, 750)
# Plot critical (i.e. equilibrium) points
plt.plot(0, 0, "b*")
plt.plot(EQ[0], EQ[1], "r*")
# Write down the caption
plt.figtext(.775, .12, "$s_k = {0}$, ".format(s_k) +\
            "$s_h = {0}$, ".format(s_h) +\
            "$\\alpha = {0}$, ".format(alpha) +\
            "$\\beta = {0}$, ".format(beta) +\
            "$\\delta = {0}$, ".format(delta) +\
            "$n = {0}$".format(n),
            horizontalalignment = "right", fontsize = 10)
# Plot what we've calculated
plt.show()
```

For the case $\alpha + \beta > 1$:

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate
# Ignore warnings when dividing by zero
np.seterr(divide='ignore', invalid='ignore')
# Define parameters
s_k, s_h = 1.5, 1.6
alpha, beta = 1.3, 1.4
delta, n = 0.1, 0.15
# Set plot limits
xmin, xmax = -.05, .6
ymin, ymax = -.05, .6
# Determine the number of arrows in the direction field
nb_points = 30
# Define our system
def mrw_model(P, t = 0):
    return ([s_k * (P[0] ** alpha) * (P[1] ** beta) - (delta + n) * P[0],
             s_h * (P[0] ** alpha) * (P[1] ** beta) - (delta + n) * P[1]])
# Define the non-trivial equilibrium point
EQ = ([((((s_k)**(1-beta))*((s_h)**(beta)))/(delta+n))**(1/(1-alpha-beta)),
      ((((s_k)**(alpha))*((s_h)**(1-alpha)))/(delta+n))**(1/(1-alpha-beta))
      ])
# Set range of t
f = plt.figure()
t = np.linspace(0, 150, 1000)
# Set initial values to start
P0 = np.array([(.5, 0), (0, .5), (.45, .1), (.1, .5),
               (.45, .2), (.2, .5)])
```

```
vcolors = plt.cm.autumn_r(np.linspace(.1, .7428571, len(P0)))
# Calculate trajectories
for i in range(len(P0)):
    P = integrate.odeint(mrw_model, P0[i], t)
    plt.plot(P[:,0], P[:,1],
             color = vcolors[i],
             label = '({0}, {1})'.format(P0[i, 0], P0[i, 1]))
# Interesting trajectories
t_star = np.linspace(0, 7, 1000)
P02 = np.array([(.2, .55), (.45, .275)])
vcolors2 = plt.cm.autumn_r(np.linspace(0.7428571, 1, len(P02)))
for i in range(len(P02)):
    P2 = integrate.odeint(mrw_model, P02[i], t_star)
    plt.plot(P2[:,0], P2[:,1],
             color = vcolors2[i],
             label = '({0}, {1})'.format(P02[i, 0], P02[i, 1]))
# Set limits of x and y that the plot shows
x = np.linspace(xmin, xmax, nb_points)
y = np.linspace(ymin, ymax, nb_points)
# Create meshgrid
X1 , Y1 = np.meshgrid(x, y)
# Calculate growth rate at each grid point
DX1, DY1 = mrw_model([X1, Y1])
# Direction at each grid point is the hypotenuse of the direction of k
# and the direction of h
M = (np.hypot(DX1, DY1))
# Give a title to the plot
plt.title('Under the condition : $\\alpha + \\beta > 1$')
# Use the quiver function to plot the direction field using
# DX1 and DY1 for directions, and M for speed
Q = plt.quiver(X1, Y1, DX1, DY1, M, pivot = 'mid', cmap = plt.cm.plasma)
plt.xlabel('$k =$' + 'Physical Capital per Worker')
plt.ylabel('$h =$' + 'Human Capital per Worker')
plt.legend(loc = 'lower left', bbox_to_anchor = (.8, .1),
           ncol = 1, fancybox = True, shadow = True,
           title = "Initial $(k, h)$")
plt.grid()
plt.xlim(xmin, xmax)
plt.ylim(ymin, ymax)
# Plot critical (i.e. equilibrium) points
plt.plot(0, 0, "b*")
plt.plot(EQ[0], EQ[1], "r*")
# Write down the caption
plt.figtext(.775, .12, "$s_k = {0}$, ".format(s_k) +\
```

```
            "$s_h = {0}$, ".format(s_h) +\
            "$\\alpha = {0}$, ".format(alpha) +\
            "$\\beta = {0}$, ".format(beta) +\
            "$\\delta = {0}$, ".format(delta) +\
            "$n = {0}$".format(n),
            horizontalalignment = "right", fontsize = 10)
# Plot what we've calculated
plt.show()
```

## Code used in 2.(e)

```
import numpy as np
from matplotlib import pyplot as plt
from scipy import integrate
# Ignore warnings when dividing by zero
np.seterr(divide='ignore', invalid='ignore')
# Define parameters
s_k, alpha, delta, n = 1.5, 0.3, 0.1, 0.15
def solow_model(k, t):
    dkdt = s_k * (k ** alpha) - (delta + n) * k
    return dkdt
# Equilibrium line that is not x-axis
k_1 = ((s_k)/(delta + n)) ** (1 / (1 - alpha))
# Initial values and colors
k_0 = np.array([25, 20, 15, 10, 1, 0])
vcolors = plt.cm.autumn_r(np.linspace(0.1, 1, len(k_0)))
# Set range of t
nb_points = 15
t = np.linspace(0, 15, nb_points)
# Determine the number of arrows in the direction field
k = np.linspace(0, 30, nb_points)
# Create meshgrid
t , k = np.meshgrid(t, k)
# Calculate the slope at a point
dk = solow_model(k, t)
# Create an array of 1's that has the same length as dk
dt = np.ones(np.size(dk))
# Size of arrows
M = np.hypot(1, dk)
# Use quiver function to draw direction field
Q = plt.quiver(t, k, dt, dk, M, pivot = 'mid', cmap = plt.cm.plasma)
# Redefine t
```

```
t = np.linspace(0, 15, nb_points)
# Solve ODEs
for i in range(len(k_0)):
    k = integrate.odeint(solow_model, k_0[i], t)
    plt.plot(t, k,
             color = vcolors[i],
             label = '$k_0$' + '= {0}'.format(k_0[i]))
# Plot
# plt.plot(t, k)
plt.title("A direction field and some trajectories : Solow")
plt.xlabel('$t =$' + 'time')
plt.ylabel('$k =$' + 'Physical Capital per Worker')
plt.xlim(-1, 15)
plt.ylim(-1, max(k_0) + 3)
plt.legend(loc = 'lower left', bbox_to_anchor = (.9, .55),
           ncol = 1, fancybox = True, shadow = True,
           title = "Initial $k_0$")
plt.grid(True)
plt.figtext(.4, .15, "$s_k = {0}$, ".format(s_k) +\
            "$\\alpha = {0}$, ".format(alpha) +\
            "$\\delta = {0}$, ".format(delta) +\
            "$n = {0}$".format(n),
            horizontalalignment = "left", fontsize = 10)
plt.axhline(y = 0, color = 'red', linestyle = '-')
plt.text(-.1, 0.1, "$\\psi_0$ $= 0$")
plt.axhline(y = k_1, color = 'black', linestyle = '-')
plt.text(-.1, k_1 + 0.2, "$\\psi_1 \\approx$ {0}".format(round(k_1, 4)))
plt.show()
```

## Code reference

From the blog post written by `Antonia Hadjimichael`, and from `Stack Overflow`