Documentation for replication of model simulations for "Learning and Money Adoption" (2022) by Michael Choi and Fan Liang

```
**********************************************************************
[1] REPLICATION - BASELINE MODEL LEARNING ABOUT STORAGE COST
**********************************************************************

[1.1] FILES AND REQUIREMENTS

Requirements:
      Anaconda Jupyter

Files:
      Baseline.ipynb

**********************************************************************
[1.2] FIGURE 3: ASSET PRICES

Parameters for the left panel:
      given_beta = 0.9894,
      given_alpha_b = 0.1,
      given_alpha_c = 0.65,
      given_c_ub = 0.15,
      given_c_lb = 0.0001,
      given_s = 0.54,
      given_eta = 0.7,
      given_chi = 0.07,
      given_B = 0.0098,
      given_i = 0.01,
      eta_grid = [0.2,0.7,0.8,0.99]

Parameters for the right panel:
      given_beta = 0.99,
      given_alpha_b = 0.1,
      given_alpha_c = 0.65,
      given_c_ub = 0.19,
      given_c_lb = 0.0001,
      given_s = 0.54,
      given_eta = 0.7,
      given_chi = 0.07,
      given_B = 0.0098,
      given_i = 0.01,
      eta_grid = [0.4,0.7,0.8,0.99]

**********************************************************************
[1.3] FIGURE 4: WELFARE COMPARISON

Parameters for the left panel:
      given_beta = 0.9894,
      given_alpha_b = 0.1,
      given_alpha_c = 0.65,
      given_c_ub = 0.15,
      given_c_lb = 0.0001,
      given_s = 0.54,
```

```
      given_chi = 0.07,
      given_B = 0.0098,
      given_i = 0.01,
      eta_grid = [0.2,0.7,0.8,0.99]

Parameters for the right panel:
      given_beta = 0.99,
      given_alpha_b = 0.1,
      given_alpha_c = 0.65,
      given_c_ub = 0.19,
      given_c_lb = 0.0001,
      given_s = 0.54,
      given_chi = 0.07,
      given_B = 0.0098,
      given_i = 0.01,
      eta_grid = [0.4,0.7,0.8,0.99]

*****************************************************************
[2] REPLICATION - ONLINE APPENDIX LEARNING ABOUT USABILITY
*****************************************************************

[2.1] FILES AND REQUIREMENTS

Requirements:
      Anaconda Jupyter

Files:
      Online_Appendix_Learning_Usability.ipynb

*****************************************************************

[2.2] FIGURE 6: WELFARE COMPARISON

Parameters for the left panel:
      given_beta = 0.996,
      given_alpha_hat = 0.005,
      given_alpha_u = 0.99,
      given_alpha_l = 0.01,
      given_c = 0.025,
      given_s = 0.58,
      given_chi = 0.07,
      given_B = 0.97
      eta_grid = [0.2,0.6,0.8,0.99]

Parameters for the right panel:
      given_beta = 0.996,
      given_alpha_hat = 0.0005,
      given_alpha_u = 0.77,
      given_alpha_l = 0.01,
      given_c = 0.025,
      given_s = 0.58,
      given_chi = 0.07,
      given_B = 0.97
      eta_grid = [0.5,0.6,0.8,0.99]
```

****
[3] REPLICATION - ONLINE APPENDIX GENERAL PREFERENCE
****

[3.1] FILES AND REQUIREMENTS

Requirements:
Anaconda Jupyter

Files:
Online_Appendix_General_Preference.ipynb

****

[3.2] FIGURE 8: EXAMPLE OF MULTIPLE EQUILIBRIA

Parameters:
beta=0.99
alpha=0.05
rho=0.5
gamma_h=10
gamma_l=-10
s_h=0.55
eta=0.01
zeta=0.01

Define utility function as:
```
def utility(q):
    return 2*q**0.5
```

****

[3.3] FIGURE 9 RIGHT PANEL: ASSET PRICES AS GAMMA^L AND GAMMA^H SCALED BY A FACTOR KAPPA

Parameters:
beta=0.99
alpha=0.02
rho=0.5
gamma_h=10
gamma_l=-10
s_h=0.55
eta=0.99
zeta=0.007

Define utility function as:
```
def utility(q):
    return 2*q**0.5
```

****

[3.4] FIGURE 9 LEFT PANEL: EXAMPLE OF NON-MONOTONE LAMBDA

Parameters of the left panel:

```
      given_beta=0.958,
      given_alpha=0.005,
      given_rho=1,
      given_gamma_h=1.444262,
      given_gamma_l= -1.236,
      given_s_h=0.6,
      given_s_l=0.4,
      given_zeta=0.01,

Define utility function of the left panel as:
        C = 120
        def utility(q):
            return 1-np.exp(-C*q)

        def utility_prime(q):
            return C*np.exp(-C*q)

        def utility_doubleprime(q):
            return -C*C*np.exp(-C*q)

*****************************************************************

[3.5] FIGURE 10: NUMERICAL EXAMPLES OF AN INCREASE IN eta

Parameters of the left panel:
      given_beta=0.99,
      given_alpha=0.01,
      given_rho=0.5,
      given_gamma_h=10,
      given_gamma_l= -10,
      given_s_h=0.55,
      given_s_l=0.45,
      given_zeta=0.005,

      eta_grid = [0.1,0.99]

Define utility function of the left panel as:
         def utility(q):
             return 2*q**0.5
         def utility_prime(q):
             return q**(-0.5)
         def utility_doubleprime(q):
             return -0.5*q**(-1.5)

Parameters of the right panel:
      given_beta=0.96,
      given_alpha=0.3,
      given_rho=1,
      given_gamma_h=13,
      given_gamma_l= -12,
      given_s_h=0.68,
      given_s_l=0.32,
      given_zeta=0.01,

      eta_grid = [0.15,0.95]
```

Define utility function of the right panel as:

```
D = 1
def utility(q):
    return 2*np.log(q+D)

def utility_prime(q):
    return 2/(q+D)

def utility_doubleprime(q):
    return -2/(q+D)**2
```

****

[3.6] FIGURE 5: THREE EXAMPLES OF b_ji

Parameters:

```
given_beta=0.958,
given_alpha=0.005,
given_rho=1,
given_gamma_h=1.444262,
given_gamma_l= -1.236,
given_s_h=0.6,
given_s_l=0.4,
given_zeta=0.01,

eta_grid = [0.1,0.9]
```

Define utility function as:

```
C = 120
def utility(q):
    return 1-np.exp(-C*q)

def utility_prime(q):
    return C*np.exp(-C*q)

def utility_doubleprime(q):
    return -C*C*np.exp(-C*q)
```

Change j = 4, 9, and 15 to simulate pi_j = 0.1. 0.46, and 0.91 respectively.

****

[3.7] FIGURE 11: TWO EXAMPLES OF A CHANGE IN eta

Parameters for the blue lines:

```
given_beta=0.9,
given_alpha=0.6,
given_rho=1,
given_gamma_h=14,
given_gamma_l= -6,
given_s_h=0.6,
given_s_l=0.4,
given_zeta=0.01,
```

```
      eta_grid = [0.01,0.9]

Utility function for the blue lines:
          A_u = 2
          B_u = 1.9
          lambda_u = 0.6
          a_u = 2.2
          b_u = 2
          theta_u = 10
          def utility(q):
              return lambda_u*B_u*q +(1-lambda_u)*A_u*(theta_u -
np.exp(-b_u*q**a_u))

          def utility_prime(q):
              return lambda_u*B_u + (1-lambda_u)*A_u*np.exp(-
b_u*q**a_u)*b_u*a_u*q**(a_u-1)

          def utility_doubleprime(q):
              return (-b_u*a_u*(q**(a_u-1))*(1-lambda_u)*A_u*np.exp(-
b_u*q**a_u)*b_u*a_u*q**(a_u-1)
                      + (1-lambda_u)*A_u*np.exp(-
b_u*q**a_u)*b_u*a_u*(a_u-1)*q**(a_u-2) )

Parameters for the red lines:
      given_beta=0.9,
      given_alpha=0.6,
      given_rho=1,
      given_gamma_h=9,
      given_gamma_l= -6,
      given_s_h=0.6,
      given_s_l=0.4,
      given_zeta=0.01,

      eta_grid = [0.01,0.9]

Utility function for the red lines:
          A_u = 1.18
          rho_u = 0.01
          def utility(q):
              if q == 0:
                  return A_u*0.001**(1-rho_u)/(1-rho_u)
              else:
                  return A_u*q**(1-rho_u)/(1-rho_u)
          def utility_prime(q):
              if q == 0:
                  return A_u*0.001**(-rho_u)
              else:
                  return A_u*q**(-rho_u)
          def utility_doubleprime(q):
              if q == 0:
                  return A_u*(-rho_u)*0.001**(-rho_u-1)
              else:
                  return A_u*(-rho_u)*q**(-rho_u-1)
```

```
**********************************************************************

[3.8] FIGURE 12: ASSET PRICES IN A DUAL-ASSET ECONOMY

Parameters for the left panel:
      beta=0.99
      alpha=0.05
      rho=0.5
      gamma_h = 10
      gamma_l = -10
      gamma_s=0.0001
      sigma=0.5
      s_h=0.55
      eta=0.5
      zeta=0.01

Define utility function of the left panel:
      def utility(q):
            return 2*q**0.5

Parameters for the right panel:
      beta=0.99
      alpha=0.05
      rho=0.5
      gamma_h = 10
      gamma_l = -10
      gamma_s=0.0001
      sigma=0.5
      s_h=0.55
      eta=0.5
      zeta=0.0007

      kappa= 0.009

Define utility function of the right panel:
      def utility(q):
            return 2*q**0.5

**********************************************************************

[3.9] FIGURE 13: NON-MONOTONE tau AND WELFARE Omega

Parameters:
      beta=0.99
      alpha=0.05
      rho=0.5
      gamma_h = 10
      gamma_l = -10
      gamma_s=0.0001
      sigma=0.5
      s_h=0.55
      eta=0.5
      zeta=0.0007

Define utility function:
```

```
      def utility(q):
            return 2*q**0.5
```

****

[3.10] FIGURE 14: BELIEFS OF REGULAR BUYERS AND HODLERS
AND FIGURE 15: ASSET PRICES AND HOLDING WITH HODLERS

Parameters:

```
      beta=0.99
      alpha=0.01
      rho=0.5
      gamma_h = 10
      gamma_l = -10
      sigma=0.5
      s_h=0.55
      eta=0.5
      zeta=0.01
      x=12
```

Define utility function as:

```
      def utility(q):
            return 2*q**0.5
```

Additional parameter used in the right panel:

```
      kappa_grid = [0.001,0.5]
```