A two-dimensional working example for the PyMacroFin Python package

Adrien d'Avernas* Damon Petersen† Valentin Schubert*
Quentin Vandeweyer†

This document presents a two-dimensional working example for the PyMacroFin Python toolbox. We derive the model and relate the model expressions to the equations needed to solve the model numerically using the package. In this example, we present a general extension of Brunnermeier and Sannikov (2014) where two agents have Epstein and Zin (1989) utility functions and aggregate volatility is time-varying. The framework can easily be modified to any other general equilibrium framework with n-agents and two state variables.

1 Model Framework

Preferences There are two types of agents: households $h \in H$ and intermediaries $i \in I$. Both agents have stochastic differential utility, as developed by Duffie and Epstein (1992). The utility of agent j over his consumption process c_t^j is defined as

$$U_t^j = \mathbb{E}_t \left(\int_t^\infty f\left(c_s^j, U_s^j\right) ds \right).$$

*Stolkholm School of Economics
†University of Chicago Booth School of Business

The function $f(c, U)$ is a normalized aggregator of consumption and continuation value in each period defined as

$$f(c,U) = \frac{1-\gamma}{1-1/\zeta} U \left[\left(\frac{c}{((1-\gamma)U)^{1/(1-\gamma)}} \right)^{1-1/\zeta} - (\rho + \kappa_L) \right]$$

where ρ is the rate of time preference, γ is the coefficient of relative risk aversion, ζ determines the elasticity of intertemporal substitution, and κ_L is the rate at which agents die. As in Drechsler, Savov, and Schnabl (2018), existing agents die at rate κ_L and new agents are born at the same rate, with a fraction $\overline{\eta}$ of new agents born as intermediaries and $(1-\overline{\eta})$ of new agents born as households. Newly born agents inherit the wealth the passed agents on an equal per capital basis. Each agent chooses its optimal consumption c_t^j, investment ι_t^j, and portfolio weight w_t^j on capital holdings in order to maximize discounted infinite life time expected utilities U_t^j.

At any time, the following budget constraint has to be satisfied:

$$\frac{dn_t^j}{n_t^j} = \left(\left(1 - w_t^j\right) r_t + w_t^j \mu_t^{r,j} - \mathfrak{c}_t^j \right) dt + w_t^j \sigma_t^{q,\sigma} dZ_t^\sigma + w_t^j \left(\sigma_t + \sigma_t^{q,k} \right) dZ^k,$$

where n_t^j is the wealth of agent j, $\mathfrak{c}_t^j = c_t^j / n_t^j$ his consumption rate, and the portfolio weight w_t^j are choice variables. Z_t^σ and Z_t^k are two standard Brownian motions that hit aggregate volatility and capital growth, respectively.

Technology The production technology in the economy is given by:

$$y_t^j = \left(a^j - \iota_t^j\right) k_t^j$$

and

$$\frac{dk_t}{k_t} = \mu_t^k dt + \sigma_t dZ_t^k,$$

where μ_t^k is given by

$$\mu_t^k = \psi_t \Phi(\iota_t^i) + (1 - \psi_t) \Phi(\iota_t^h),$$

$\Phi(\cdot)$ is a concave investment function, and ψ_t is the share of capital in the hands of intermediaries:

$$\psi_t \equiv \frac{w_t^i n_t^i}{w_t^i n_t^i + w_t^h n_t^h}.$$

In this model, we work with the following functional form for the investment function:

$$\Phi(\iota_t^j) = log(1 + \kappa_p \iota_t^j)/\kappa_p - \delta^j$$

The price of a unit of capital is q_t. The volatility of capital returns follows a diffusion:

$$\frac{d\sigma_t}{\sigma_t} = \kappa\big(\overline{\sigma} - \sigma_t\big)dt + \varsigma dZ_t^\sigma.$$

The stochastic law of motion of q_t follows:

$$\frac{dq_t}{q_t} = \mu_t^q dt + \sigma_t^{q,\sigma} dZ_t^\sigma + \sigma_t^{q,k} dZ_t^k.$$

The variables μ_t^q, $\sigma_t^{q,k}$, and $\sigma_t^{q,\sigma}$ are to be determined endogenously. We can use Ito's lemma to write the process of the value of capital:

$$\frac{d(q_t k_t^j)}{q_t k_t^j} = \big(\mu_t^k + \mu_t^q + \sigma_t \sigma_t^{q,k}\big)dt + \sigma_t^{q,\sigma} dZ_t^\sigma + \big(\sigma_t + \sigma_t^{q,k}\big)dZ^k.$$

Hence, the return on the physical asset is:

$$dr_t^j = \underbrace{\left(\frac{a^j - \iota_t^j}{q_t} + \mu_t^k + \mu_t^q + \sigma_t \sigma_t^{q,k}\right)}_{\mu_t^{r,j}} dt + \sigma_t^{q,\sigma} dZ_t^\sigma + \big(\sigma_t + \sigma_t^{q,k}\big)dZ^k.$$

2 Solving Equilibrium

Value Function We will guess and verify that the homotheticity of preferences allows us to write the value function for agents of type j as:

$$U\left(n_t^j, \xi_t^j\right) = \frac{\left(n_t^j\right)^{1-\gamma} \xi_t^j}{1-\gamma},$$

where variable ξ_t^j follows

$$\frac{d\xi_t^j}{\xi_t^j} = \mu_t^{\xi,j} dt + \sigma_t^{\xi,\sigma,j} dZ_t^\sigma + \sigma_t^{\xi,k,j} dZ_t^k.$$

We can write the Hamilton Jacobi Bellman (HJB) equation corresponding to the problem of agent j as

$$\begin{aligned}
0 = \max_{\mathfrak{c}_t^j, \iota_t^j, w_t^j} & f\left(\mathfrak{c}_t^j n_t^j, U_t^j\right) \\
& + \left(\left(1-w_t^j\right) r_t + w_t^j \mu_t^{r,j} - \mathfrak{c}_t^j\right) n_t^j U_n(n_t^j, \xi_t^j) + \mu_t^{\xi,j} \xi_t^j U_\xi(n_t^j, \xi_t^j) \\
& + \frac{1}{2}\left[\left(w_t^j \sigma_t^{q,\sigma} n_t^j\right)^2 + \left(w_t^j\left(\sigma_t + \sigma_t^{q,k}\right) n_t^j\right)^2\right] U_{nn}(n_t^j, \xi_t^j) \\
& + \frac{1}{2}\left[\left(\sigma^{\xi,\sigma,j} \xi_t^j\right)^2 + \left(\sigma^{\xi,k,j} \xi_t^j\right)^2\right] U_{\xi\xi}(n_t^j, \xi_t^j) \\
& + \left[w_t^j \sigma_t^{q,\sigma} n_t^j \sigma_t^{\xi,\sigma,j} \xi_t + w_t^j\left(\sigma_t + \sigma_t^{q,k}\right) n_t^j \sigma_t^{\xi,k,j} \xi_t^j\right] U_{n\xi}(n_t^j, \xi_t^j).
\end{aligned}$$

Substituting in the guess, the HJB equation becomes

$$\begin{aligned}
0 = \max_{\mathfrak{c}_t^j, \iota_t^j, w_t^j} & \frac{1}{1-1/\zeta}\left[\frac{\left(\mathfrak{c}_t^j\right)^{1-1/\zeta}}{\left(\xi_t^j\right)^{\frac{1-1/\zeta}{1-\gamma}}} - (\rho + \kappa_L)\right] + (1-w_t^j) r_t + w_t^j \mu_t^{r,j} - \mathfrak{c}_t^j + \frac{\mu^{\xi,j}}{1-\gamma} \\
& - \frac{\gamma}{2}\left(w_t^j \sigma_t^{q,\sigma}\right)^2 - \frac{\gamma}{2}\left(w_t^j \sigma_t + w_t^j \sigma_t^{q,k}\right)^2 + w_t^j \sigma_t^{q,\sigma} \sigma_t^{\xi,\sigma,j} + w_t^j\left(\sigma_t + \sigma_t^{q,k}\right) \sigma_t^{\xi,k,j}.
\end{aligned}$$

Optimality Conditions The first order conditions with respect to $\mathfrak{c}_t^j, \iota_t^j$, and w_t^j are given by

$$\left(\mathfrak{c}_t^j\right)^{-1/\zeta} = \left(\xi_t^j\right)^{\frac{1-1/\zeta}{1-\gamma}},$$

$$1/q_t = \Phi_\iota(\iota_t),$$

$$\mu_t^{r,j} - r_t - \gamma w_t^j \left(\sigma_t^{q,\sigma}\right)^2 - \gamma w_t^j \left(\sigma_t + \sigma_t^{q,k}\right)^2 + \sigma_t^{q,\sigma}\sigma_t^{\xi,\sigma,j} + \left(\sigma_t + \sigma_t^{q,k}\right)\sigma_t^{\xi,k,j} = 0.$$

Plugging in the optimality conditions in the HJB equation gives the following:

$$0 = \frac{1}{1-1/\zeta}\left(\mathfrak{c}_t^j - (\rho + \kappa_L)\right) + r_t - \mathfrak{c}_t^j + \frac{\gamma}{2}\left(w_t^j \sigma_t^{q,\sigma}\right)^2 + \frac{\gamma}{2}\left(w_t^j \sigma_t + w_t^j \sigma_t^{q,k}\right)^2 + \frac{\mu^{\xi,j}}{1-\gamma}.$$

Market Clearing Conditions We start by providing the definition of such an equilibrium in the state variables $\{\eta_t, \sigma_t\}$, where η_t is defined as the share of wealth in the hands of the intermediaries:

$$\eta_t = \frac{n_t^i}{n_t^h + n_t^i} = \frac{n_t^i}{q_t k_t}.$$

Then, we can use the market clearing condition for consumption to find q_t. Market clearing for consumption dictates that consumption from both types of agents equals

the surplus from the production technology:

$$\begin{aligned}
\mathfrak{c}_t^i n_t^i + \mathfrak{c}_t^h n_t^h &= (a^i - \iota_t^i)k_t^i + (a^h - \iota_t^h)k_t^h \\
\frac{\mathfrak{c}_t^i n_t^i}{q_t k_t} + \frac{\mathfrak{c}_t^h n_t^h}{q_t k_t} &= \frac{(a^i - \iota_t^i)k_t^i}{q_t k_t} + \frac{(a^h - \iota_t^h)k_t^h}{q_t k_t} \\
\mathfrak{c}_t^i \eta_t + \mathfrak{c}_t^h (1-\eta_t) &= \frac{(a^i - \iota_t^i)n_t^i k_t^i}{n_t^i q_t k_t} + \frac{(a^h - \iota_t^h)n_t^h k_t^h}{n_t^h q_t k_t} \\
\mathfrak{c}_t^i \eta_t + \mathfrak{c}_t^h (1-\eta_t) &= \frac{(a^i - \iota_t^i)\eta_t k_t^i}{n_t^i} + \frac{(a^h - \iota_t^h)(1-\eta_t)k_t^h}{n_t^h} \\
\mathfrak{c}_t^i \eta_t + \mathfrak{c}_t^h (1-\eta_t) &= (a^i - \iota_t^i)\eta_t w_t^i/q_t + (a^h - \iota_t^h)(1-\eta_t)w_t^h/q_t
\end{aligned}$$

Now note that

$$\psi_t = \frac{w_t^i n_t^i}{w_t^i n_t^i + w_t^h n_t^h} = w_t^i \eta_t,$$

So finally, we have

$$\left(\mathfrak{c}_t^i \eta_t + \mathfrak{c}_t^h (1-\eta_t)\right) q_t = \psi_t (a^i - \iota_t^i) + (1-\psi_t)(a^h - \iota_t^h)$$

The market clearing condition for capital allows us to identify r_t:

$$\begin{aligned}
k_t^i + k_t^h &= k_t \\
\frac{k_t^i}{k_t} + \frac{k_t^h}{k_t} &= 1 \\
\frac{k_t^i n_t^i q_t}{n_t^i q_t k_t} + \frac{k_t^h n_t^h q_t}{n_t^h q_t k_t} &= 1 \\
w_t^i \eta_t + w_t^h (1-\eta_t) &= 1.
\end{aligned}$$

Laws of Motion Using our definition of η_t and Ito's lemma, we can derive the law of motion of η_t before accounting for agent lifecycle turnover as:

$$\begin{aligned}\frac{d\eta_t}{\eta_t} =& \Bigg(r_t + w_t^i(\mu_t^{r,j} - r_t) - \mathfrak{c}_t^i - \mu_t^k - \mu_t^q + (\sigma_t)^2 + (\sigma_t^{q,k})^2 + (\sigma_t^{q,\sigma})^2 \\ &+ \sigma_t \sigma_t^{q,k} - w_t^i(\sigma_t^{q,\sigma})^2 - w_t^i(\sigma_t + \sigma_t^{q,k})^2 \Bigg) dt \\ &+ \big(w_t^i - 1\big)\sigma_t^{q,\sigma} dZ_t^\sigma + \big(w_t^i - 1\big)\big(\sigma_t + \sigma_t^{q,k}\big) dZ^k \\ =& \underline{\mu_t^\eta} dt + \underline{\sigma_t^{\eta,\sigma}} dZ_t^\sigma + \underline{\sigma_t^{\eta,k}} dZ_t^k\end{aligned}$$

Accounting for redistribution of wealth due to agent death and birth at rate κ_L we arrive at the following:

$$\begin{aligned}d\eta_t &= \kappa_L(\overline{\eta} - \eta_t)dt + \eta_t(1-\eta_t)\left[\underline{\mu_t^\eta} dt + \underline{\sigma_t^{\eta,\sigma}} dZ_t^\sigma + \underline{\sigma_t^{\eta,k}} dZ_t^k\right] \\ \frac{d\eta_t}{\eta_t} &= \frac{\kappa_L}{\eta_t}(\overline{\eta} - \eta_t)\, dt + (1-\eta_t)\left[\underline{\mu_t^\eta} dt + \underline{\sigma_t^{\eta,\sigma}} dZ_t^\sigma + \underline{\sigma_t^{\eta,k}} dZ_t^k\right]\end{aligned}$$

which leads to the following values for the drift and volatilities for the law of motion of η_t as the following:

$$\begin{aligned}\mu_t^\eta &= \kappa_L(\overline{\eta} - \eta_t) + \eta_t(1-\eta_t)\underline{\mu_t^\eta} \\ \sigma_t^{\eta,\sigma} &= \eta_t(1-\eta_t)\underline{\sigma_t^{\eta,\sigma}} \\ \sigma_t^{\eta,k} &= \eta_t(1-\eta_t)\underline{\sigma_t^{\eta,k}}\end{aligned}$$

By applying Ito's lemma, we can find $\sigma_t^{q,\sigma}$, $\sigma_t^{q,k}$, $\sigma_t^{\xi,\sigma,j}$, $\sigma_t^{\xi,k,j}$, and μ_t^q from:

$$
\begin{aligned}
q(\sigma_t, \eta_t)\sigma_t^{q,\sigma} &= q_\sigma(\sigma_t, \eta_t)\varsigma\sigma_t + q_\eta(\sigma_t, \eta_t)\sigma_t^{\eta,\sigma}\eta_t, \\
q(\sigma_t, \eta_t)\sigma_t^{q,k} &= q_\eta(\sigma_t, \eta_t)\sigma_t^{\eta,k}\eta_t, \\
\xi^j(\sigma_t, \eta_t)\sigma_t^{\xi,\sigma,j} &= \xi_\sigma^j(\sigma_t, \eta_t)\varsigma\sigma_t + \xi_\eta^j(\sigma_t, \eta_t)\sigma_t^{\eta,\sigma}\eta_t, \\
\xi^j(\sigma_t, \eta_t)\sigma_t^{\xi,k,j} &= \xi_\eta^j(\sigma_t, \eta_t)\sigma_t^{\eta,k}\eta_t, \\
q(\sigma_t, \eta_t)\mu_t^q &= q_\sigma(\sigma_t, \eta_t)\mu_t^\sigma\sigma_t + q_\eta(\sigma_t, \eta_t)\mu_t^\eta\eta_t + \frac{1}{2}q_{\sigma\sigma}(\sigma_t, \eta_t)\left(\varsigma\sigma_t\right)^2 \\
&\quad + \frac{1}{2}q_{\eta\eta}(\sigma_t, \eta_t)\left[\left(\sigma_t^{\eta,\sigma}\eta_t\right)^2 + \left(\sigma_k^{\eta,k}\eta_t\right)^2\right] \\
&\quad + q_{\sigma\eta}(\sigma_t, \eta_t)\varsigma\sigma_t\sigma_t^{\eta,\sigma}\eta_t.
\end{aligned}
$$

3 Numerical Solution with PyMacroFin

In this section we link the algebra from Sections 1 and 2 to the code used for numerical solution. The full code for this model is shown in Appendix A. Note that this section is not a replacement for the detailed object documentation contained with the package documentation[1]. To begin, a model object should be defined and given a name as follows:

```
from model import macro_model
m = macro_model(name='BruSan')
```

Parameters The model parameters are reported in Table 1. The parameters are specified and given values using the following command:

`m.params.add_parameter('parameter_name', parameter_value)}.`

Variables Model specific variables are shown in Table 2. Endogenous variables are defined with the method `m.set_endog()`, state variables are defined with the method `m.set_state()`, and value variables are defined with the method `m.set_value()`. Intermediate variables are defined with `m.equation()` commands.

[1] https://adriendavernas.com/pymacrofin/index.html

Table 1: Model parameters

Parameter	Definition
γ^j	relative risk aversion
ζ^j	intertemporal elasticity of substitution
ρ	discount rate
κ_L	death rate (and birth rate) of agents
a^j	productivity
κ_p	investment costs
κ	drift of volatility
$\bar{\sigma}$	average volatility
ς	loading of volatility process

Table 2: Variables

Variables	Definition
Endogenous	q_t, ψ_t, μ_t^{η}, $\sigma_t^{q,k}$, $\sigma_t^{q,\sigma}$
Secondary	w_t^i, w_t^h, $\mathbf{c}_t^i$, $\mathbf{c}_t^h$, ι_t^i, ι_t^h, $\mu_t^{r,i}$, $\mu_t^{r,h}$, μ_t^k, μ_t^q, $\mu_t^{n,i}$, $\mu_t^{n,h}$, μ_t^{σ}, σ_t, $\sigma_t^{\eta,\sigma}$, $\sigma_t^{\eta,k}$, $\sigma_t^{n,i,k}$, $\sigma_t^{n,h,k}$, $\sigma_t^{n,i,\sigma}$, $\sigma_t^{n,h,\sigma}$, $\sigma_t^{\xi,i,k}$, $\sigma_t^{\xi,h,k}$, $\sigma_t^{\xi,i,\sigma}$, $\sigma_t^{\xi,h,k}$

The two state variables are

$$\eta_t = \mathsf{e} \tag{1}$$

$$\sigma_t = \mathsf{z} \tag{2}$$

and are defined in the code as follows:

```
m.set_state(['e','z'])
```

The wealth multipliers are

$$\xi_t^i = \texttt{vi} \tag{3}$$

$$\xi_t^h = \texttt{vh} \tag{4}$$

defined in the code as

```
m.set_value(['vi','vh'],init=[0.04,0.04],latex=[r'$\xi^i$',r'$\xi^h$'])
```

The endogenous variables, listed in Table 2, are defined in the code by the following command:

```
m.set_endog(['q','psi','mue','sigqk','sigqs'],init=[1,0.95,0,0,0],
            latex=[r'$q$',r'$\psi$',r'$\mu^{\eta}$',
                   r'$\sigma^{q,k}$',r'$\sigma^{q,\sigma}$'])
```

Leverage was defined as

$$w_t^i = \frac{\psi_t}{\eta_t} \tag{5}$$

which is an intermediate variable in the code, written as follows:

```
m.equation("wi = psi/e")
```

The following intermediate variables are defined similarly using the `m.equation()` method.

$$w_t^h = \frac{1-\psi_t}{1-\eta_t} \tag{6}$$

Consumption-to-wealth ratio is given by the first order condition:

$$\mathfrak{c}_t^i = (\xi_t^i)^{\frac{1-\zeta^i}{1-\gamma^i}} \tag{7}$$

$$\mathfrak{c}_t^h = (\xi_t^h)^{\frac{1-\zeta^h}{1-\gamma^h}} \tag{8}$$

The investment ratio is also given by its first order condition:

$$\iota_t^i = \frac{q_t - 1}{\kappa_p} \tag{9}$$

$$\iota_t^h = \frac{q_t - 1}{\kappa_p} \tag{10}$$

The functional form for Φ^j was assumed to be:

$$\Phi_t^i = log(1 + \kappa_p \iota_t^i)/\kappa_p - \delta^i \tag{11}$$

$$\Phi_t^h = log(1 + \kappa_p \iota_t^h)/\kappa_p - \delta^h \tag{12}$$

The drift of the state variable σ_t was assumed to be

$$\mu_t^\sigma = \kappa(\overline{\sigma} - \sigma_t) \tag{13}$$

Using Ito's lemma and the process for k_t^i and k_t^h, we get the drift of aggregate capital:

$$\mu_t^k = \psi_t \Phi_t^i + (1 - \psi_t)\Phi_t^h \tag{14}$$

From the law of motion of wealth, we have the following loadings:

$$\sigma_t^{n,i,\sigma} = w_t^i \sigma_t^{q,\sigma} \tag{15}$$

$$\sigma_t^{n,h,\sigma} = w_t^h \sigma_t^{q,\sigma} \tag{16}$$

$$\sigma_t^{n,i,k} = w_t^i(\sigma_t + \sigma_t^{q,k}) \tag{17}$$

$$\sigma_t^{n,h,k} = w_t^h(\sigma_t + \sigma_t^{q,k}) \tag{18}$$

Similarly for η_t:

$$\underline{\sigma_t^{\eta,\sigma}} = (w_t^i - 1)\sigma_t^{q,\sigma} \tag{19}$$

$$\underline{\sigma_t^{\eta,k}} = (w_t^i - 1)(\sigma_t + \sigma_t^{q,k}) \tag{20}$$

$$\sigma_t^{\eta,\sigma} = \eta_t(1-\eta_t)\underline{\sigma_t^{\eta,\sigma}} \tag{21}$$

$$\sigma_t^{\eta,k} = \eta_t(1-\eta_t)\underline{\sigma_t^{\eta,k}} \tag{22}$$

The law of motion for ξ_t^j was derived using Ito's lemma and yielded[2]

$$\sigma_t^{\xi,i,k} = \frac{\xi_\eta^i}{\xi_t^i}\sigma_t^{\eta,k}\eta_t \tag{23}$$

$$\sigma_t^{\xi,h,k} = \frac{\xi_\eta^h}{\xi_t^h}\sigma_t^{\eta,k}\eta_t \tag{24}$$

$$\sigma_t^{\xi,i,\sigma} = \frac{\xi_\eta^i}{\xi_t^i}\sigma_t^{\eta,\sigma}\eta_t + \frac{\xi_\sigma^i}{\xi_t^i}\varsigma\sigma_t \tag{25}$$

$$\sigma_t^{\xi,h,\sigma} = \frac{\xi_\eta^h}{\xi_t^h}\sigma_t^{\eta,\sigma}\eta_t + \frac{\xi_\sigma^h}{\xi_t^h}\varsigma\sigma_t \tag{26}$$

The same was done for

$$\begin{aligned}\mu_t^q =& \frac{q_\sigma}{q_t}\mu_t^\sigma\sigma_t + \frac{q_\eta}{q_t}\mu_t^\eta\eta_t + \frac{1}{2}\frac{q_{\sigma\sigma}}{q_t}\left(\varsigma\sigma_t\right)^2 \\ &+ \frac{1}{2}\frac{q_{\eta\eta}}{q_t}\left[\left(\sigma_t^{\eta,\sigma}\eta_t\right)^2 + \left(\sigma_t^{\eta,k}\eta_t\right)^2\right] \\ &+ \frac{q_{\sigma\eta}}{q_t}\varsigma\sigma_t\sigma_t^{\eta,\sigma}\eta_t\end{aligned} \tag{27}$$

[2]Note that ς is denoted as `sigs` in the code.

Finally, the drift of r_t and n_t^j using Ito's lemma:

$$\mu_t^{r,i} = (a^i - \iota_t^i)/q_t + \Phi^i + \mu_t^q + \sigma_t \sigma_t^{q,k} \tag{28}$$

$$\mu_t^{r,h} = (a^h - \iota_t^h)/q_t + \Phi^h + \mu_t^q + \sigma_t \sigma_t^{q,k} \tag{29}$$

$$r_t = \mu_t^{r,i} - \gamma^i w_t^i((\sigma_t^{q,\sigma})^2 + (\sigma_t + \sigma_t^{q,k})^2) + \sigma_t^{q,\sigma}\sigma_t^{\xi,i,\sigma} + (\sigma_t + \sigma_t^{q,k})\sigma_t^{\xi,i,k} \tag{30}$$

$$\mu_t^{n,i} = r_t + w_t^i(\mu_t^{r,i} - r_t) - \mathfrak{c}_t^i \tag{31}$$

$$\mu_t^{n,h} = r_t + w_t^i(\mu_t^{r,h} - r_t) - \mathfrak{c}_t^h \tag{32}$$

It is important to note that an intermediate variable cannot be used in another equation before it is defined.

Equilibrium To solve for the values of endogenous variable values at equilibrium, a system of equations is defined. The number of equations must be equal to the number of endogenous variables and each equation must be equal to zero at equilibrium. The first endogenous variable we define in the code is the drift of the state variable η_t. It was derived using Ito's lemma. The equation is written down as:

$$\begin{aligned} 0 = \frac{\kappa_L}{\eta_t}(\overline{\eta} - \eta_t) + (1 - \eta_t)(\mu_t^{n,i} - \mu_t^q - \mu_t^k - \sigma_t \sigma_t^{q,k} + (\sigma_t^{q,k} + \sigma_t)^2 \\ + (\sigma_t^{q,\sigma})^2 - w^i(\sigma_t^{q,\sigma})^2 - w_t^i(\sigma_t^{q,k} + \sigma_t)^2) - \mu_t^\eta \end{aligned} \tag{33}$$

This is defined in the code using the `m.endog_equation()` method as follows:

```
m.endog_equation("kappa_l/e*(ebar-e)+(1-e)*(muni - muk - muq\
                  - sigma*sigqk + (sigqk+sigma)**2 + sigqs**2 \
                  - wi*sigqs**2 - wi*(sigqk+sigma)**2) - mue")
```

The other equations in the endogenous system are similarly defined using the same syntax. The price of capital q_t comes from the market clearing condition of consumption:

$$0 = (\mathfrak{c}_t^i \eta_t + \mathfrak{c}_t^h(1 - \eta_t))q_t - (a^i - \iota_t^i)\eta_t w_t^i - (a^h - \iota_t^h)(1 - \eta_t)w_t^h \tag{34}$$

ψ_t solves the difference between the first order condition for w_t^i and w_t^h:

$$\begin{aligned} 0 =& \mu_t^{r,i} - \mu_t^{r,h} \\ &+ \gamma^h w_t^h((\sigma_t^{q,\sigma})^2 + (\sigma_t + \sigma_t^{q,k})^2) \\ &- \gamma^i w_t^i((\sigma_t^{q,\sigma})^2 + (\sigma_t + \sigma_t^{q,k})^2) \\ &+ \sigma_t^{q,\sigma}\sigma_t^{\xi,i,\sigma} + (\sigma_t + \sigma_t^{q,k})\sigma_t^{\xi,i,k} \\ &- \sigma_t^{q,\sigma}\sigma_t^{\xi,h,\sigma} - (\sigma_t + \sigma_t^{q,k})\sigma_t^{\xi,h,k} \end{aligned} \tag{35}$$

Finally, using Ito's lemma, we have:

$$0 = (\varsigma q_\sigma \sigma_t + \sigma_t^{\eta,\sigma} q_\eta \eta) - \sigma_t^{q,\sigma} q_t \tag{36}$$

$$0 = \sigma_t^{\eta,k} q_\eta \eta - \sigma_t^{q,k} q_t \tag{37}$$

Hamilton-Jacobi-Bellman The variables associated with the HJB equation must also be specified to inform the package how to iterate backward through time to find the steady-state equilibrium solution to the system[3].

As outlined in the documentation and solution methods paper[3], we solve a partial differential equation of the following form:

$$r(X)F(X,t) = u(X) + \sum_{i=1}^{m} \mu_i(X)\frac{\partial F(X,t)}{\partial x_i} + \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{\sigma_i(X)\sigma_j(X)}{2}\frac{\partial^2 F(X,t)}{\partial x_i \partial x_j} + \frac{\partial F(X,t)}{\partial t}$$

where x_i are state variables, $F(X,t)$ are value variables, and X is a vector of all variables. The `m.hjb_equation()` method is used to inform the model object of the values to be used in this partial differential equation. The values of $u(X)$ are provided as follows:

```
m.hjb_equation('u','vi',0)
m.hjb_equation('u','vh',0)
```

The values of $r(X)$ are provided as follows:

[3]See the solution method paper for details on the solution methods employed in the package: https://adriendavernas.com/papers/solutionmethod.pdf

```
m.hjb_equation('r','vi',"-1*(1-gammai)*(1/(1-1/zetai)*(ci-(rhoi+kappa_l))\
                +r-ci+gammai/2*(wi*(sigqs)**2 +wi*(sigqk+sigma)**2))")
m.hjb_equation('r','vh',"-1*(1-gammah)*(1/(1-1/zetah)*(ch-(rhoh+kappa_l))\
                +r-ch+gammah/2*(wh*(sigqs)**2 +wh*(sigqk+sigma)**2))")
```

The values for state variable drifts and volatilities are similarly defined.

```
m.hjb_equation('mu','e','mue*e')
m.hjb_equation('mu','z','muz*z')
m.hjb_equation('sig','e',"(siges*e)**2 + (sigek*e)**2")
m.hjb_equation('sig','z',"(sigz*z)**2")
m.hjb_equation('sig','cross',"siges*e*sigz*z")
```

Once again, all variables referenced in these expressions were defined previously when defining intermediate, value, state, and endogenous variables.

References

M. K. Brunnermeier and Y. Sannikov. A macroeconomic model with a financial sector. *The American Economic Review*, 104(2):379–421, 2014.

I. Drechsler, A. Savov, and P. Schnabl. A model of monetary policy and risk premia. *The Journal of Finance*, LXXIII(1):317–373, 1 February 2018 2018.

D. Duffie and L. Epstein. Stochastic differential utility. *Econometrica*, pages 353–394, 1992.

L. Epstein and S. Zin. Substitution, risk aversion, and the temporal behavior of consumption and asset returns: A theoretical framework. *Econometrica*, pages 937–969, 1989.

Appendix A: Code for Solution with PyMacroFin

This appendix contains the full code to run the model described in this document. For details on syntax and reasoning behind the structure of the model definition,

please see the remainder of the documentation[4].

```
from model import macro_model
import numpy as np
import pandas as pd
import time
import utilities as util

def define_model():
    m = macro_model(name='BruSan')

    m.set_endog(['q','psi','mue','sigqk','sigqs'],init=[1,0.95,0,0,0],
                latex=[r'$q$',r'$\psi$',r'$\mu^{\eta}$',
                       r'$\sigma^{q,k}$',r'$\sigma^{q,\sigma}$'])
    m.prices = ['q']
    m.set_state(['e','z'])
    m.set_value(['vi','vh'],init=[0.04,0.04],latex=[r'$\xi^i$',r'$\xi^h$'])

    m.params.add_parameter('gammai',2)
    m.params.add_parameter('gammah',3)
    m.params.add_parameter('ai',.1)
    m.params.add_parameter('ah',.1)
    m.params.add_parameter('rhoi',.04)
    m.params.add_parameter('rhoh',.04)
    m.params.add_parameter('sigz',.01)
    m.params.add_parameter('sigbar',.5)
    m.params.add_parameter('deltai',.04)
    m.params.add_parameter('deltah',.04)
    m.params.add_parameter('kappa_p',2)
    m.params.add_parameter('kappa_z',5)
    m.params.add_parameter('zetai',1.15)
    m.params.add_parameter('zetah',1.15)
    m.params.add_parameter('kappa_l',.9)
    m.params.add_parameter('ebar',0.5)

    m.equation("sigma = z")
```

[4] https://adriendavernas.com/pymacrofin/index.html

```
m.equation("wi = psi/e")
m.equation("wh = (1-psi)/(1-e)")
m.equation("ci = vi**((1-zetai)/(1-gammai))")
m.equation("ch = vh**((1-zetah)/(1-gammah))")
m.equation("iotai = (q-1)/kappa_p")
m.equation("iotah = (q-1)/kappa_p")
m.equation("phii = log(1+kappa_p*iotai)/kappa_p-deltai")
m.equation("phih = log(1+kappa_p*iotah)/kappa_p-deltah")
m.equation("muz = kappa_z*(sigbar-sigma)")
m.equation("muk = psi*phii+(1-psi)*phih")
m.equation("signis = wi*sigqs")
m.equation("signhs = wh*sigqs")
m.equation("signik = wi*(sigqk+sigma)")
m.equation("signhk = wh*(sigqk+sigma)")
m.equation("siges = e*(1-e)*(signis -sigqs)")
m.equation("sigek = e*(1-e)*(signik - (sigqk+sigma))")
m.equation("sigxik = d(vi,e)/vi*sigek*e")
m.equation("sigxhk = d(vh,e)/vh*sigek*e")
m.equation("sigxis = d(vi,e)/vi*siges*e + d(vi,z)/vi*sigz*z")
m.equation("sigxhs = d(vh,e)/vh*siges*e + d(vh,z)/vh*sigz*z")
m.equation("muq = d(q,e)/q*mue*e + d(q,z)/q*muz*z + \
            1/2*d(q,e,e)/q*((siges*e)**2 + (sigek*e)**2) + \
            1/2*d(q,z,z)/q*(sigz*z)**2 + d(q,e,z)*siges*e*sigz*z")
m.equation("muri = (ai-iotai)/q + phii + muq + sigma*sigqk")
m.equation("murh = (ah-iotah)/q + phih + muq + sigma*sigqk")
m.equation("r = muri - gammai*wi*((sigqs**2)+(sigma+sigqk)**2) + \
            sigqs*sigxis + (sigqk+sigma)*sigxik")
m.equation("muni = r + wi*(muri-r)-ci")
m.equation("munh = r + wh*(murh-r)-ch")

m.endog_equation("kappa_l/e*(ebar-e)+(1-e)*(muni - muk - muq\
                  - sigma*sigqk + (sigqk+sigma)**2 + sigqs**2 \
                  - wi*sigqs**2 - wi*(sigqk+sigma)**2) - mue")
m.endog_equation("(ci*e+ch*(1-e))*q - psi*(ai-iotai) - (1-psi)*(ah-iotah)")
m.endog_equation("muri - murh + gammah*wh*((sigqs**2)+(sigqk+sigma)**2) - \
                  gammai*wi*((sigqs)**2+(sigqk+sigma)**2) + sigqs*sigxis + \
                  (sigqk+sigma)*sigxik - sigqs*sigxhs - (sigqk+sigma)*sigxhk")
```

```
    m.endog_equation("(sigz*z*d(q,z) + siges*e*d(q,e))-sigqs*q")
    m.endog_equation("sigek*e*d(q,e) - sigqk*q")

    m.hjb_equation('mu','e','mue*e')
    m.hjb_equation('mu','z','muz*z')
    m.hjb_equation('sig','e',"(siges*e)**2 + (sigek*e)**2")
    m.hjb_equation('sig','z',"(sigz*z)**2")
    m.hjb_equation('sig','cross',"siges*e*sigz*z")
    m.hjb_equation('u','vi',0)
    m.hjb_equation('u','vh',0)
    m.hjb_equation('r','vi',"-1*(1-gammai)*(1/(1-1/zetai)*(ci-(rhoi+kappa_l))\
                    +r-ci+gammai/2*(wi*(sigqs)**2 +wi*(sigqk+sigma)**2))")
    m.hjb_equation('r','vh',"-1*(1-gammah)*(1/(1-1/zetah)*(ch-(rhoh+kappa_l))\
                    +r-ch+gammah/2*(wh*(sigqs)**2 +wh*(sigqk+sigma)**2))")

    m.options.loop = False
    m.options.outer_plot = True
    m.options.n0 = 50
    m.options.n1 = 50
    m.options.start0 = 0.05
    m.options.start1 = 0.05
    m.options.end0 = 0.95
    m.options.end1 = 0.95
    m.options.inner_solver = 'newton-raphson'
    m.options.parallel = True

    return m

if __name__=='__main__':
    tic = time.time()
    m = define_model()
    util.deploy_dash(m)
    m.run()
    toc = time.time()
    print('elapsed time: {}'.format(toc-tic))
```