# Solution Algorithm

*Monetary Policy, Segmentation, and the Term Structure*

Rohan Kekre[1] Moritz Lenel[2] Federico Mainardi[3]

This document describes the numerical solution method used in the paper, providing more detail on the algorithm outlined in appendix C. The referenced equations can be found in the same appendix. The key idea of the numerical solution is to use the Feynman-Kac formula (39) to solve for equilibrium bond prices. We use Monte Carlo simulations to solve for the expected value under the risk neutral measure Q.

## Additional definitions

In the following, we define the expected excess returns on the arbitrageurs' portfolio as

$$\hat{\omega}_t \equiv \int_0^\infty \left(\alpha(\tau)\log\left(P_t^{(\tau)}\right) + \theta_t(\tau)\right)\left(\omega_t^{(\tau)} - r_t\right) d\tau. \tag{N1}$$

The drift of arbitrageurs' wealth in (43) can then be rewritten as

$$\omega_t = \xi\left(\bar{W} - W_t\right) dt + W_t r_t dt + \hat{\omega}_t dt.$$

Furthermore, define the correction to the drifts of $r$, $\beta$ and $W$ under $Q$ as

$$\hat{\mu}_{r,t} \equiv \frac{\gamma}{W_t}\sigma_r \eta_{r,t} \tag{N2}$$

$$\hat{\mu}_{\beta,t} \equiv \frac{\gamma}{W_t}\sigma_\beta \eta_{\beta,t} \tag{N3}$$

$$\hat{\mu}_{W,t} \equiv \frac{\gamma}{W_t}\left(\eta_{r,t}^2 + \eta_{\beta,t}^2\right). \tag{N4}$$

so that, for example, the drift of $r$ under $Q$, as defined in (36), can be written as

$$\mu_{r,t} = \kappa_r(\bar{r} - r_t) - \hat{\mu}_{r,t}.$$

[1]Chicago Booth and NBER. Email: rohan.kekre@chicagobooth.edu.
[2]Princeton and NBER. Email: lenel@princeton.edu.
[3]Chicago Booth. Email: fmainard@chicagobooth.edu.

## Numerical algorithm

**State space, time steps, and expectations** The model is solved over a three-dimensional tensor grid in the state variables $r$, $\beta$ and $W$. The goal of the solution algorithm is to compute equilibrium bond prices at each point of the grid. All relevant variables are stored as values at each grid point. Off grid, we compute values using cubic spline interpolation. For the exogenous state variables, $r$ and $\beta$, we allow for linear extrapolation above and below the grid's upper and lower bounds.

In the time dimension, we simulate the model in discrete time steps $dt$ and store bond prices at each grid point at lower frequency $d\tau$. Concretely, we pick $d\tau$ to be one month, and $dt$ to be $d\tau/6$.

In order to compute expected bond returns over the next time step $dt$, we form expectations over future values as weighted sums over Gauss-Hermite quadrature nodes for the two shocks, $\epsilon_{r,t+dt}$ and $\epsilon_{\beta,t+dt}$, that drive the discretized processes of $r$ and $\beta$. On a given grid point, each quadrature node is associated with a transition to a new state in the next time step. Values of bond prices at each of the quadrature nodes are calculated using cubic splines, as the corresponding states will generically lie off the state grid.

**Numerical simulation** When simulating the model, the processes of the exogenous state variables, $r$ and $\beta$, are approximated using the Euler-Maruyama method, so that

$$r_{t+dt} = r_t + \kappa_r \left(\bar{r} - r_t\right) dt + \sigma_r \sqrt{dt}\epsilon_{r,t+dt}, \tag{N5}$$

$$\beta_{t+dt} = \beta_t + \kappa_\beta \left(\bar{\beta} - \beta_t\right) dt + \sigma_\beta \sqrt{dt}\epsilon_{\beta,t+dt}, \tag{N6}$$

where $\epsilon_{r,t+dt}$ and $\epsilon_{\beta,t+dt}$ are independently distributed as standard Normal random variables. We similarly discretize the process of the endogenous state variable, $W$, in (22) as

$$\begin{aligned} W_{t+dt} = {} & W_t + \left(1 - e^{-\xi dt}\right)\left(\bar{W} - W_t\right) \\ & + e^{-\xi dt}\left(W_t r_t dt + \hat{\omega}_t dt + \eta_{r,t}\sqrt{dt}\epsilon_{r,t+dt} + \eta_{\beta,t}\sqrt{dt}\epsilon_{\beta,t+dt}\right). \end{aligned} \tag{N7}$$

When simulating under measure $Q$, we use (N2), (N3) and (N4) to correct the drifts of $r$, $\beta$ and $W$ as described in equations (36), (37) and (38), so that

$$
\begin{aligned}
r_{t+dt} &= r_t + \left[\kappa_r\left(\bar{r} - r_t\right) - \hat{\mu}_{r,t}\right] dt + \sigma_r \sqrt{dt}\epsilon_{r,t+dt}, && \text{(N8)} \\
\beta_{t+dt} &= \beta_t + \left[\kappa_\beta\left(\bar{\beta} - \beta_t\right) - \hat{\mu}_{\beta,t}\right] dt + \sigma_\beta \sqrt{dt}\epsilon_{\beta,t+dt}, && \text{(N9)} \\
W_{t+dt} &= W_t + \left(1 - e^{-\xi dt}\right)\left(\bar{W} - W_t\right) \\
&\quad + e^{-\xi dt}\left[\left(W_t r_t + \hat{\omega}_t - \hat{\mu}_{W,t}\right) dt + \eta_{r,t}\sqrt{dt}\epsilon_{r,t+dt} + \eta_{\beta,t}\sqrt{dt}\epsilon_{\beta,t+dt}\right]. && \text{(N10)}
\end{aligned}
$$

When simulating, we take as given a function $\mathcal{I}(r, \beta, W)$ that returns values for $\hat{\mu}_{r,t}$, $\hat{\mu}_{\beta,t}$, $\hat{\mu}_{W,t}$, $\hat{\omega}_t$, $\eta_{r,t}$, $\eta_{\beta,t}$ taking $r_t$, $\beta_t$, and $W_t$ as inputs. Concretely, this function is constructed at the beginning of each iteration as a cubic-spline interpolant over the current guess of those variables at each grid point.

**Overview over solution steps** At each grid point, we solve for the expectation in (39) for each maturity $\tau$ by simulating $N$ different paths for the three state variables, holding fixed the function $\mathcal{I}(r, \beta, W)$. In other words, we replace the integral in (39) by a Monte Carlo sum:

$$
P^{(\tau)}(r, \beta, W) = E_t^Q\left[e^{-\int_0^\tau r_{t+s} ds} | (r, \beta, W)\right] \approx \frac{1}{N}\sum_{n=1}^{N} e^{-\sum_{i=1}^{\tau/dt} r_{n,i} dt}. \qquad \text{(N11)}
$$

For each path and at each time step, we interpolate over the state space using $\mathcal{I}(r, \beta, W)$ to obtain current values of $(\hat{\mu}_r, \hat{\mu}_\beta, \hat{\mu}_W, \hat{\omega}, \eta_r, \eta_\beta)_{(r,\beta,W)}$. These values are used to inform the dynamics of the state variables over the next time step. The $N$ paths of $r_t$ then enter (N11).

**Initialization** The algorithm starts from initial guesses for drift corrections under measure $Q$, $\hat{\mu}_r$, $\hat{\mu}_\beta$ and $\hat{\mu}_W$, expected returns on arbitrageurs' portfolio, $\hat{\omega}$, and wealth diffusions, $\eta_r$ and $\eta_\beta$. These initial guesses are sufficient to simulate forward the dynamics of the aggregate states under $Q$, which we use to compute bond prices for each maturity $\tau$ and at each grid point. The $N$ series of exogenous shocks used along the simulations are held fixed across iterations and we use antithetic sampling across paths for variance reduction.

**Computational steps** In each iteration, the subroutine `calcP` in `solution.jl` performs the following steps:

1. **Interpolation** At the beginning of each iteration, we use the current guess of $(\hat{\mu}_r, \hat{\mu}_\beta, \hat{\mu}_W, \hat{\omega}, \eta_r, \eta_\beta)$ at each grid point to calculate an updated interpolant $\mathcal{I}(r, \beta, W)$.

2. **Forward simulation.** For each grid point, we simulate $N$ paths of the state variables $r$, $\beta$ and $W$ under $Q$. Given states $(r_{n,i}, \beta_{n,i}, W_{n,i})$ at time step $i$ of path $n$, we first compute drift corrections $\hat{\mu}_r$, $\hat{\mu}_\beta$ and $\hat{\mu}_W$, expected returns on arbitrageurs' portfolio, $\hat{\omega}$, and wealth diffusions, $\eta_r$ and $\eta_\beta$, using $\mathcal{I}(r, \beta, W)$. Given values of $(\hat{\mu}_r, \hat{\mu}_\beta, \hat{\mu}_W, \hat{\omega}, \eta_r, \eta_\beta)$ at the current state, we derive next period states $(r_{n,i+1}, \beta_{n,i+1}, W_{n,i+1})$ using equations (N8), (N9) and (N10). We repeat our simulation technique until we reach time step $\bar{\tau}/dt$, where $\bar{\tau}$ denotes the maximum bond maturity that we consider. Along the way, we record the cumulative sum $\sum_i r_{n,i}$ as an input for (N11) (since we store bond prices at lower frequency $d\tau$, we do not store the complete cumulative sum vector, but only the elements that are necessary to derive bond prices at frequency $d\tau$ and their expectations with $dt$ shorter maturity at the next time step).

3. **Bond prices.** Given the cumulative sum of $r$ along each path $n$, we use equation (N11) to compute bond prices for all maturities at frequency $d\tau$ and for each initial grid point. For each maturity $\tau$, we interpolate bond prices across grid points and use these interpolation coefficients to derive derivatives of prices with respect to the state variables $r$, $\beta$ and $W$.

4. **Expected returns.** For each bond at maturity $\tau$, we also derive a corresponding set of prices for maturity $\tau - dt$. We proceed to derive the instantaneous expected bond returns $\omega^{(\tau)}(r, \beta, W)$ at each grid point. To do so, we use price interpolation coefficients to construct bond prices for all maturities $\tau - dt$ at each future quadrature node. We then construct instantaneous expected returns as the log of the weighted sum of future bond prices minus the log bond price at the initial grid point.

5. **Drifts, portfolio returns and diffusion parameters.** Given bond prices, derivatives of bond prices with respect to the state variables and instantaneous expected returns, we proceed to calculate new values of $(\hat{\mu}_r, \hat{\mu}_\beta, \hat{\mu}_W, \hat{\omega}, \eta_r, \eta_\beta)$.

We first update $\hat{\omega}$ using equation (N1). We also update $\eta_r$ and $\eta_\beta$ using equations (44) and (45). Finally, given bond prices, derivatives of bond prices, instantaneous expected returns and the updated values of $\eta_r$ and $\eta_\beta$, we can update $\hat{\mu}_r$, $\hat{\mu}_\beta$ and $\hat{\mu}_W$ using equations (N2), (N3) and (N4) respectively. We use a weighted average of the previous guesses and the updated values as a new initial guess for the next iteration to ensure stable convergence of the algorithm.

These steps are repeated until the differences between the previous guesses and updated values are sufficiently small. Once the algorithm has converged, we use the resulting bond prices as well as the estimates for $\eta_r$ and $\eta_\beta$ to simulate the model under the physical measure to derive simulated moments and impulse responses. The key function for that purpose is `calcSeries` in `results.jl`.