Appendix: Solution Method

The Matlab code named "Model.m" provides the algorithm for solving the model. The code is divided in six sections.

Section 1. Parameter Values sets the parameters values shown in table 1. We use 100 points in the grid for bonds and three states for y^T shocks, three states for news shocks and two states for interest rates shocks. The convergence tolerance level for the solution of decision rules, as defined in Section 3 below, is set to $\epsilon = 1e^{-5}$.

Section 2. Construction of Markov Chain discretizes y^T shocks using Tauchen and Hussey's method. The time-series properties of the y^T process that the method targets are estimates obtained by Bianchi (2011)using data for Argentina, and the corresponding moments are reported in table 1. We incorporate news shocks according to the formulae in the Section 2.3 of the paper. Then we add global liquidity shocks to construct the entire transition matrix, assuming y^T shocks and global liquidity shocks are independent.

Section 3. Decentralized Equilibrium solves the decentralized equilibrium using the time-iteration method. Intuitively, this algorithm solves the model by backward recursive-substitution of the model's optimality conditions written in recursive form. In particular, the algorithm solves for the recursive functions $c^T(b,z)$,$P^N(b,z)$ and $B(b,z)$ that satisfy these four conditions:

$$P^N(b,z) = \frac{1-\omega}{\omega}\left(\frac{c^T(b,z)}{y^N}\right)^{1+\eta} \tag{24}$$

$$u_T(c^T(b,z),y^N) \geq \beta R(z) E_z[u_T(c^T(B(b,z),z'),y^N)] \tag{25}$$

$$B(b,z) \geq -\kappa R(z)(P^N(b,z)y^N + y^T(z)) \tag{26}$$

$$c^T(b,z) + q(z)B(b,z) = b + y^T(z) \tag{27}$$

where z is a triple (y^T, q, s) that includes the realizations of the exogenous shocks to y^T, the news signal s, and q (recall that $q = \frac{1}{R}$).

Start the algorithm at an initial point defined by setting $K = 1$ and define conjectures for the equilibrium functions at this point, denoted $c^T_K(b,z)$, $p^N_K(b,z)$ and $B_K(b,z)$. Then proceed with the following steps:

1. Set $B_{K+1}(b,z) = -\kappa R(z)(P^N_K(b,z)y^N + y^T(z))$, and calculate $c^T_{K+1}(b,z)$ using equation 27, which yields $c^T_{K+1}(b,z) = b + y^T(z) - (1/R(z))B_{K+1}(b,z)$

2. Compute
$$U \equiv u_T(c^T_{K+1}(b,z),y^N) - \beta R(z) E_z\left[u_T(c^T_K(B_K(b,z),z'),y^N)\right] \tag{28}$$

3. If $U > 0$, the collateral constraint binds and then equation 24 implies that the equilibrium

price must be given by $p^N_{K+1}(b,z) = \frac{1-\omega}{\omega}\left(\frac{c^T_{K+1}(b,z)}{y^N}\right)^{1+\eta}$

4. If $U \leq 0$, the collateral constraint does not bind. Discard the values of $B_{K+1}(b,z)$ and $C^T_{K+1}(b,z)$ set in Step 1, and solve for $c^T_{K+1}(b,z)$ as the recursive function that satisfies the Euler equation 25 with equality using the fsolve root-finding routine. We then compute $P^N_{K+1}(b,z)$ using again equation 24 and $B_{K+1}(b,z)$ using equation 27

5. The above steps will in general produce a new set of functions $c^T_{K+1}(b,z)$, $p^N_{K+1}(b,z)$ and $B_{K+1}(b,z)$ that will differ from the conjectures $c^T_K(b,z)$, $p^N_K(b,z)$ and $B_K(b,z)$. We thus check the convergence criterion $\sup|x_{K+1} - x_K| \leq \epsilon$ for $x = B, c^T, p^N$. If the criterion fails, the conjectures are replaced with the solutions $c^T_{K+1}(b,z)$, $p^N_{K+1}(b,z)$ and $B_{K+1}(b,z)$ and the procedure returns to step 1 using these new conjectures. If the convergence criterion $\sup|x_{K+1} - x_K| \leq \epsilon$ holds, the recursive functions are a solution to the decentralized competitive equilibrium in recursive form.

Section 4. Social Planner solves the social planer's problem. The algorithm is also a time-iteration code similar to that of decentralized equilibrium. The difference is that u_T in equation 25 becomes

$$u^{SP}_T = u_T + \mu^{SP}\psi \tag{29}$$

where $\mu^{SP} \geq 0$, with strict inequality if the collateral constraint 26 binds, and ψ is the externality term given by $\kappa(\eta+1)\left(\frac{1-\omega}{\omega}\right)\left(\frac{c^T}{y^N}\right)^{\eta}$.

Section 5. Welfare Calculation takes the optimal policy functions we derived from section 3 and 4 of the Matlab code, and iterates until convergence to get value functions of the private agent and social planer. We then calculate the welfare gain as in Bianchi (2011).

Section 6. Optimal Tax calculates optimal macro-prudential tax according to equation 23, and set tax rate to zero when the borrowing constraint binds.

The Matlab code named "Simulation.m" simulate our model. The code is divided in three sections.

Section 1. Simulation simulates our model for 201,000 periods. The first 1,000 periods are discarded to eliminate initial condition dependence. The initial bond position is set as mid point of the bond grid for both DE and SP economies.

Section 2. Event Analysis identifies sudden stop events, and find the surrounding three periods before and after the event. Crisis is defined as current account goes beyond two standard deviation and collateral constraint binds in the decentralized economy. The crisis moments are obtained by taking average across all crisis episodes.

Section 3. Crisis Breakdown analyzes crises effects preceded by different news (see figure 5) and by different liquidity regimes (see figure 6). The definition of the breakdowns can be found in section 4.3.

The Matlab code "figures.m" generates figures 2, 3, 4,5, 6, 8 and 9 presented in the paper, and display table 2 in the command window.