

Online Appendix (User's Guide) for
FiPit: A Simple, Fast Global Method for Solving Models with
Two Endogenous States & Occasionally Binding Constraints

Enrique G. Mendoza
University of Pennsylvania, NBER & PIER

Sergio Villalvazo
University of Pennsylvania

1 Introduction

This Appendix provides a user’s guide for the Matlab codes that implement the *FiPIt* algorithm. It describes how the various steps of the algorithm presented in Section 3 of the paper are undertaken in the computer programs. The programs are distributed in a zip file labeled *MendozaVillalvazoFiPItCode* available online. The main directory of this file has the same name, and it contains two folders named *FiPIt* and *Mfiles*. The main Matlab script is named `mainFiPIt.m` and is located in the *FiPIt* folder. This folder also includes the output files as well as script files used to generate various output components (moments, graphs, etc.). The `mainFiPIt.m` program calls several function scripts that are stored in the *MFiles* folder. Table A1 provides a list of all the files, their location and contents.

Table A1: Files Included in the *MendozaVillalvazoFiPItCode* Directory

<i>Name</i>	<i>Folder Location</i>	<i>Description</i>
<code>mainFiPIt.m</code>	FiPIt/	Main script
<code>script1_Moments.m</code>	FiPIt/	Script to obtain moments in Table 2 and 5.e
<code>script2_PolicyPlot.m</code>	FiPIt/	Script to produce the Figures
<code>script4_TableDiffAmpl.m</code>	FiPIt/	Script to the probability of Sudden Stops
<code>fBiLinearInterpolation.m</code>	MFiles/	Function for bi-linear interpolation
<code>fFiPIt_Cons.m</code>	MFiles/	Function to find consumption rule from the Euler Equation for bonds using FiPIt
<code>fFiPIt_EulerError.m</code>	MFiles/	Function to compute Euler Errors
<code>fFiPIt_MuBond.m</code>	MFiles/	Function to compute Lagrange multiplier of ad-hoc debt limit in the RBC model
<code>fFiPIt_MuHat.m</code>	MFiles/	Function that solves for credit constraint multiplier ratio using a non-linear solver
<code>fFiPIt_PriceK.m</code>	MFiles/	Function to compute price of capital from the capital Euler Equation using FiPIt
<code>fMarkov.m</code>	MFiles/	Function to generate Markov chain simulation

The output of `mainFiPIt.m` is stored in a `.mat` file. To solve the variant of the model in which the credit constraint never binds (denoted the RBC model), set the value of κ high enough so as to ensure that this is the case. Under our calibration, $\kappa > 1$ is sufficient. The `.mat` file with the RBC solution is named `solFiPIt_RBC.mat`. To solve the Sudden Stops (SS) model, set $\kappa < 1$. The `.mat` file with this solution is named `solFiPIt_SS.mat`. The long-run moments of these two models reported in Table 2 and 5.e of the paper are computed using `script1_Moments.m`, choosing to comment in or out either line 14 or 15 to load the corresponding `.mat` file with the RBC or SS solution. Similarly, to produce the policy function plots run `script2_PolicyPlot.m` and to produce the probability of Sudden Stops run `script4_TableDiffAmpl.m`.

2 Recursive Equilibrium Conditions

To implement the *FiPIt* method, we first re-write the equilibrium conditions of the model in recursive form. The model has two endogenous states, b and k , and three exogenous states, using s to denote

the triple of exogenous shocks $s \equiv (A, R, p)$, which includes the shocks to TFP (A), the world interest rate (R) and the price of imported inputs (p). The recursive equilibrium is defined by a set of recursive functions for allocations $[b'(b, k, s), k'(b, k, s), c(b, k, s), L(b, k, s), v(b, k, s)]$, prices $[w(b, k, s), q(b, k, s), d(b, k, s)]$ and multipliers $[\lambda(b, k, s), \mu(b, k, s)]$ that satisfy the following recursive equilibrium conditions:

$$\left(c(b, k, s) - \frac{L(b, k, s)^\omega}{\omega} \right)^{-\sigma} = \lambda(b, k, s)(1 + \tau) \quad (1)$$

$$\alpha A k^\gamma L(b, k, s)^{\alpha-1} v(b, k, s)^\eta = w(b, k, s) \left(1 + \phi(R - 1) + \frac{\mu(b, k, s)}{\lambda(b, k, s)} \phi R \right) \quad (2)$$

$$\eta A k^\gamma L(b, k, s)^\alpha v(b, k, s)^{\eta-1} = p \left(1 + \phi(R - 1) + \frac{\mu(b, k, s)}{\lambda(b, k, s)} \phi R \right) \quad (3)$$

$$\lambda(b, k, s) = R\beta E[\lambda(b', k', s'), \mu(b', k', s')] + \mu(b, k, s) \quad (4)$$

$$\lambda(b, k, s) = \frac{1}{q(b, k, s)} \beta E \left[\lambda(b', k', s') (d(b', k', s'), k'(b', k', s'), s') \right. \\ \left. + q'(b', k', s', k'(b', k', s'), s') \right] + \mu(b, k, s) \kappa \quad (5)$$

$$d(b, k, s) = \gamma A k^{\gamma-1} L(b, k, s)^\alpha v(b, k, s)^\eta - \delta + \frac{a (k'(b, k, s) - k)^2}{2 k^2} \quad (6)$$

$$q(b, k, s) = 1 + a \left(\frac{k'(b, k, s) - k}{k} \right) \quad (7)$$

$$w(b, k, s) = L(b, k, s)^{\omega-1} (1 + \tau) \quad (8)$$

$$c(b, k, s)(1 + \tau) + k'(b, k, s) - (1 - \delta)k + \frac{a (k'(b, k, s) - k)^2}{2 k} = A k^\gamma L(b, k, s)^\alpha v(b, k, s)^\eta \\ - p v(b, k, s) - \phi(R - 1)(L(b, k, s)^\omega (1 + \tau) + p v(b, k, s)) - R^{-1} b'(b, k, s) + b \quad (9)$$

3 Contents of the mainFiPIt.m program

The mainFiPIt.m file is divided into 5 cells, each one including comments describing how the contents of each cell relate to each of the seven algorithm steps described in subsection 3.2 of the paper. The itemized step numbers labeled in bold typeface below match the step numbers in the paper description, with the line in the Matlab code in which the step is executed indicated in parenthesis.

Cell 1. Parameterization & State Space: Sets the model’s parameter values, creates the discrete grids of bonds and capital, defines the Markov processes of shocks, and sets the values of program parameters that define the method to solve for capital price, the convergence criteria, the maximum number of iterations and the updating coefficients for decision rule conjectures between one iteration and the next. The endogenous states are foreign bonds b and domestic capital k . The exogenous states are included in s , which denotes a triple of shocks $s \equiv (A, R, p)$ that includes TFP (A), the world interest rate (R) and the price of imported inputs (p). The realization set for shock triples $s \in \mathbf{S}$ comes from the discretization of the stochastic processes of the shocks, which is typically done using Tauchen’s quadrature method. Here, we take \mathbf{S} and the associated Markov transition probability matrix from Mendoza (2010), where each shock has two realizations and hence \mathbf{S} has eight triples. For the endogenous states, the algorithm defines grids with a total of $nBondGrid$ nodes for bonds and $nCapitalGrid$ nodes for capital. The state space has $nBondGrid \times nCapitalGrid \times 8$ elements and is defined by all $(b, k, s) \in \mathbf{B} \otimes \mathbf{K} \otimes \mathbf{S}$. The conditional statement starting in Line 82 adjusts the bonds grid when the SS model is being solved to make sure the collateral constraint binds before the lower bound of the grid. The recursive equilibrium is defined by a set of recursive functions for allocations $[b'(b, k, s), k'(b, k, s), c(b, k, s), L(b, k, s), v(b, k, s)]$, prices $[w(b, k, s), q(b, k, s), d(b, k, s)]$ and the multipliers $[\lambda(b, k, s), \mu(b, k, s)]$. The model and program parameters are listed in Table A2.

Table A2: Parameter Values

<i>Calibrated parameters</i>		
σ	coefficient of relative risk aversion	2.0
ω	labor elasticity coefficient	1.8461
β	discount factor	0.92
a	capital adjustment costs coefficient	2.75
ϕ	fraction of input costs requiring working capital	0.2579
δ	depreciation rate	0.088
α	labor share in gross output	0.59
η	imported inputs share in gross output	0.10
γ	capital share in gross output	0.31
τ	tax on consumption	0.17
A	average TFP	6.982
<i>Algorithm parameters</i>		
ρ^b	Updating weight for bonds decision rule	1.00
ρ^μ	Updating weight for multiplier ratio	1.00
ρ^q	Updating weight for price of capital	0.30
ε^f	Function convergence criterion	10e-4

Cell 2. Initial Conjectures, Array Definitions & Non-linear Solver Options: This cell defines the initial conjectures for the equilibrium recursive functions. Following the notation in the paper, at any iteration j the initial conjectured functions are denoted $\hat{q}_j(b, k, s)$ for the price of capital, $\hat{b}'_j(b, k, s)$ for the decision rule for bonds, and $\hat{\mu}_j(b, k, s) \equiv \mu_j(b, k, s)/\lambda_j(b, k, s)$ for the multiplier ratio. This cell also initializes the arrays for other variables and constructs a function that sets the optimization options for the non-linear solver used later in the program to solve for allocations when the credit constraint binds.

Step 1. (Line 106) Sets the first-iteration recursive function conjectures to $\hat{b}'_0(b, k, s) = b$, $\hat{q}_0(b, k, s) = 1$ and $\hat{\mu}_0(b, k, s) = 0$ for all $(b, k, s) \in \mathbf{B} \otimes \mathbf{K} \otimes \mathbf{S}$. The instructions after those initialize the arrays for other variables, the first-iteration value of the convergence metric for the recursive functions ($nMaxDif$) and the iterations counter ($nIter$), and they also define the function $pSolverOpt$ to set the options for Matlab's *fsolve* non-linear solver used later in the code when solving for allocations in states in which $\tilde{\mu} > 0$.

Cell 3. Main Loop Executing Iterations on Equilibrium Recursive Functions: The *While* loop starting in line 143 executes the successive iterations on the equilibrium recursive functions for bonds, price of capital and multiplier ratio. The current iteration number (j) is stored in the integer $nIter$, and the value of the convergence metric attained in iteration $nIter$ is stored in $nMaxDif$.

Step 2. (Line 146) Generates decision rules for capital, investment, factor allocations, gross

output and consumption in iteration j implied by the conjectures $\hat{q}_j(b, k, s), \hat{b}'_j(b, k, s), \hat{\mu}_j(b, k, s)$:

$$\begin{aligned}
k'_j(b, k, s) &= \frac{k}{a} [\hat{q}_j(b, k, s) - 1 + a] \\
\tilde{i}_j(b, k, s) &= (k'_j(b, k, s) - k) \left[1 + \frac{a}{2} \left(\frac{k'_j(b, k, s) - k}{k} \right) \right] - \delta k \\
v_j(b, k, s) &= \left\{ \frac{Ak^\gamma \eta^{\frac{\omega-\alpha}{\omega}} \frac{\alpha}{1+\tau} \frac{\alpha}{\omega}}{p^{\frac{\omega-\alpha}{\omega}} [1 + \phi(R-1) + \hat{\mu}_j(b, k, s)\phi R]} \right\}^{\frac{\omega}{\omega(1-\eta)-\alpha}} \\
L_j(b, k, s) &= \left\{ \frac{\alpha}{\eta(1+\tau)} p v_j(b, k, s) \right\}^{\frac{1}{\omega}} \\
y_j(b, k, s) &= Ak^\gamma L_j(b, k, s)^\alpha v_j(b, k, s)^\eta \\
(1+\tau)c_j(b, k, s) &= y_j(b, k, s) - p v_j(b, k, s) - \phi(R-1) [(1+\tau)L_j(b, k, s)^\omega + p v_j(b, k, s)] \\
&\quad - \tilde{i}_j(b, k, s) - \frac{\hat{b}'_j(b, k, s)}{R} + b
\end{aligned}$$

The code uses here the same set of expressions for the RBC and SS solutions. For the latter, the values of factor allocations, gross output and consumption vary with $\tilde{\mu}(\cdot)$, whereas in the RBC solution they do not because $\tilde{\mu}(\cdot) = 0$ always. Note also that since $\tilde{\mu}(\cdot)$ is always set to zero in the first iteration, the first-iteration results of this step are identical when solving either the RBC or SS models. When solving the RBC model, $\tilde{\mu}(\cdot)$ remains zero in all iterations, but when solving the SS model, $\tilde{\mu}(\cdot) > 0$ in states in which the credit constraint binds.

Step 3.1 (Line 166) Assume the collateral constraint does not bind. Solve for new decision rules (indexed $j+1$) for labor, intermediate goods and output. Since the constraint is assumed to be non-binding, these decision rules are the same in RBC and SS solutions:

$$\begin{aligned}
v_{j+1}(b, k, s) &= \left\{ \frac{Ak^\gamma \eta^{\frac{\omega-\alpha}{\omega}} \frac{\alpha}{1+\tau} \frac{\alpha}{\omega}}{p^{\frac{\omega-\alpha}{\omega}} [1 + \phi(R-1)]} \right\}^{\frac{\omega}{\omega(1-\eta)-\alpha}} \\
L_{j+1}(b, k, s) &= \left\{ \frac{\alpha}{\eta(1+\tau)} p v_{j+1}(b, k, s) \right\}^{\frac{1}{\omega}} \\
y_{j+1}(b, k, s) &= Ak^\gamma L_{j+1}(b, k, s)^\alpha v_{j+1}(b, k, s)^\eta
\end{aligned}$$

Steps 3.2 & 3.3. (Line 181) Solve for the $j+1$ consumption and bonds decision rules using the bonds' Euler Equation and the resource constraint. For each (b, k, s) in the state space, consumption is solved for using the `fFiPIt_Cons.m` function located in the *Mfiles* folder. This function finds the new consumption rule by solving “directly” from the Euler equation, as

explained in Step 3.2 of the algorithm description in the paper:

$$c_{j+1}(b, k, s) = \left\{ \beta RE \left[\left(c_j(\hat{b}'_j(b, k, s), k'_j(b, k, s), s') - \frac{L_j(\hat{b}'_j(b, k, s), k'_j(b, k, s), s')^\omega}{\omega} \right)^{-\sigma} \right] + \frac{L_{j+1}(b, k, s)^\omega}{\omega} \right\}^{-\frac{1}{\sigma}}$$

fFiPIt_Cons.m calls the function fBiLinearInterpolation.m, also in the *Mfiles* folder, in order to find the values of $c_j(\hat{b}'_j(b, k, s), k'_j(b, k, s), s')$, $L_j(\hat{b}'_j(b, k, s), k'_j(b, k, s), s')$, which are determined using bi-linear interpolation because $\hat{b}'_j(b, k, s)$ and $k'_j(b, k, s)$ are not on the nodes of the bonds and capital grids in general. Once $c_{j+1}(b, k, s)$ is determined, the new bonds policy function $b'_{j+1}(b, k, s)$ is solved for using the resource constraint, and the implied leverage ratio is computed (i.e. the value of $\frac{-q_t^b b_{t+1} + \phi R_t (w_t L_t + p_t v_t)}{\kappa q_t k_{t+1}}$).

Step 3.4. (Line 192) Check if collateral constraint binds using the new decision ($j+1$ -indexed) decision rules:

$$\frac{b'_{j+1}(b, k, s)}{R} - \phi R [(1 + \tau)L_{j+1}(b, k, s)^\omega + p v_{j+1}(b, k, s)] + \kappa \hat{q}_j(b, k, s) k'_j(b, k, s) \geq 0$$

Line 202 evaluates if there are (b, k, s) states for which the new bonds decision rule is below the lower bound of the bonds grid. In these cases, the lower bound is a binding ad-hoc debt limit. The bonds decision rule is re-set equal to this debt limit, the consumption decision rule is re-set to the value implied by the resource constraint, and we also compute the associated Lagrange multiplier for the binding ad-hoc debt limit.

Step 4. (Line 217) This step is only executed when solving the SS model and only for states (b, k, s) in which the constraint was found to be binding in Step 3.4, because these are the only states in which the decision rules depend on $\tilde{\mu}$. This step solves for $\tilde{\mu}_{j+1}(b, k, s)$ by applying Matlab's *fsolve* root finder to a function formed using the fFiPIt_MuHat.m script located in the *Mfiles* folder. fFiPIt_MuHat.m forms equation (38) in the paper. It uses the j -indexed functions for consumption and labor to form the expected value in the right-hand-side of eq. (38), which requires the same bi-linear interpolation method used to solve for c_{j+1} in step 3.2. The solver uses the optimization options set in *pSolverOpt* as defined in Cell 1 and returns the value of $\tilde{\mu}_{j+1}(b, k, s)$. The solver uses these options: `optimoptions('fsolve','Display','off','TolFun',1e-18)`. A small tolerance convergence criterion is needed in order to attain convergence of the

recursive functions and small Euler errors. We use as initial condition (*vInitX*) the current iteration's initial conjecture $\tilde{\mu}_j(b, k, s)$. After $\tilde{\mu}_{j+1}(b, k, s)$ is solved for, we compute the associated $j+1$ values of the decision rules using eqs. (33)-(36) in the paper. Keep in mind that there are many variations of occasionally binding constraints for which the constrained allocations and the multiplier of the binding constraint can be solved separately, in which case there is no need to use a non-linear solver in this step. Two cases explored in the paper are one in which working capital is removed from the collateral constraint and one in which the credit constraint is set to a constant value instead of the value of collateral (see p. 15 and p. 24 of the paper). This makes the *FiPIt* algorithm significantly faster.

Step 5. (Line 250) This step is just a comment noting that at this point in the code we have solved the new ($j + 1$ -indexed) optimal decision rules for all (b, k, s) in the state space conditional on the conjectured $\hat{q}_j(b, k, s)$ function.

Step 6. (Line 252) Compute the new pricing function. This step is coded so as to allow the user to choose one of the two alternatives to compute the pricing function described in Steps 6.1 and 6.2 of the paper. The former uses fixed-point iteration, the latter finds q as the forward solution of the capital Euler equation. The fixed-point iteration (forward) solution is chosen by setting `pFixPointPriceK == 1` (`pFixPointPriceK == 0`) in the algorithm parameters of Cell 1. In both cases, we solve for $q_{j+1}(b, k, s)$ using the `fFiPIt_PriceK.m` script located in the *Mfiles* folder. This script solves the following equation, which is eq. (41) in the paper (we use (\cdot) to denote (b, k, s) so as to shorten the notation):

$$q_{j+1}(b, k, s) = \frac{\beta E_t \left[\left(c_{j+1} \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right) - \frac{L_{j+1}(b'_{j+1}(\cdot), k'_j(\cdot), s')^\omega}{\omega} \right)^{-\sigma} \left[d' \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right) + \hat{q}_j \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right) \right] \right]}{\left(c_{j+1}(\cdot) - \frac{L_{j+1}(\cdot)^\omega}{\omega} \right)^{-\sigma} (1 - \kappa \tilde{\mu}_{j+1}(\cdot))}$$

where

$$d' \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right) = \gamma A' k'_j(\cdot)^{\gamma-1} L_{j+1} \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right)^\alpha v_{j+1} \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right)^\eta - \delta + \frac{a \left(k'_j \left(b'_{j+1}(\cdot), k'_j(\cdot), s' \right) - k'_j(\cdot) \right)^2}{k'_j(\cdot)^2}$$

When solving by fixed-point iteration, the above Euler equation solves directly for $q_{j+1}(\cdot)$, since all the terms in the right-hand-side of the expression are known at this point in the code. The equation is solved once and the solution passed on as the new pricing function. Note that in forming the conditional expectation, we use j -indexed conjectures of the price of capital and

the capital decision rule (since their $j + 1$ values are not known), but the rest of the relevant recursive functions are indexed $j + 1$ (since they have been solved for in the previous steps of the algorithm). As before, bi-linear interpolation is used to determine the values of all the functions that have $(b'_{j+1}(\cdot), k'_j(\cdot))$ as arguments (the $t + 1$ variables in the conditional expectation of the Euler equation), since those functions are only known at grid nodes.¹ When solving by forward solution, `fFiPIt_PriceK.m` is used repeatedly to iterate on the above capital Euler equation until $q_{j+1}(\cdot)$ and $\hat{q}_j(\cdot)$ converge, but keeping all the other functions unchanged. For these iterations, the iteration counter is the integer $nIterInner$, and the value of the convergence metric at iteration $nIterInner$ is denoted $nMaxDifInner$. The convergence criterion is the value assigned to the parameter $nTolInner$ in Cell 1.

Step 6.1. (Line 271) If $pFixPointPriceK = 1$, then the first solution for $q_{j+1}(b, k, s)$ generated for each (b, k, s) using `fFiPIt_PriceK.m` is retained as the new pricing function.

Step 6.2. (Line 274) If $pFixPointPriceK = 0$ (which is executed by the *else* instruction when $pFixPointPriceK = 1$ is not valid), then `fFiPIt_PriceK.m` is used to generate the new values of $q_{j+1}(b, k, s)$ for each (b, k, s) as we iterate to convergence on the capital pricing function.

Step 7. (Line 281) Check convergence and update conjectures. The convergence criterion is given by $nMaxDif \leq \varepsilon^f$, where $nMaxDif$ is the following convergence metric:

$$nMaxDif = \max \left\{ |q_{j+1}(b, k, s) - \hat{q}_j(b, k, s)|, |b'_{j+1}(b, k, s) - \hat{b}'_j(b, k, s)|, |\tilde{\mu}_{j+1}(b, k, s) - \hat{\mu}_j(b, k, s)| \right\}$$

$\forall (b, k, s) \in \mathbf{B} \otimes \mathbf{K} \otimes \mathbf{S}$. The value of ε^f is defined by setting the program parameter $nTol$ in Cell 1. If convergence is attained, the recursive equilibrium has been solved and the results are stored in either the `solFiPIt_SS.mat` file for the SS model or the `solFiPIt_RBC.mat` file for the RBC model. If convergence is not attained, then generate new conjectures as follows:

$$\hat{x}_{j+1}(b, k, s) = (1 - \rho^x) \hat{x}_j(b, k, s) + \rho^x x_{j+1}(b, k, s)$$

for $x = [q, b, \tilde{\mu}]$ and some $0 \leq \rho^x$. $\hat{x}_j(b, k, s)$ in the right-hand-side of this expression represents the initial conjectures used in the current iteration, while $\hat{x}_{+1}j(b, k, s)$ in the left-hand-side

¹For evaluating dividends, we found that the algorithm performs better if we interpolate the functions that enter in the definition of dividends individually and then generate the value of dividends, instead of first defining dividends and then interpolating the dividends function.

denotes the new conjectures for the next iteration. Use $0 < \rho^x < 1$ ($\rho^x > 1$) for the particular function $x(\cdot)$ that is not converging (converging too slowly). The values of the ρ^x coefficients are set with the parameters $nUpdateGuessB$ and $nUpdateGuessPK$ in Cell 1.² Return to Step 2 (Line 146) using the new conjectures for the next iteration.

Cell 4. Compute Euler Equation Errors.

The solution of the recursive equilibrium is completed when the program exits Cell 3. The next two cells generate two important objects based on the model solution. First, Cell 4 computes the errors of the Euler equations of bonds and capital using the `fFiPIt_EulerError.m` function located in the *Mfiles* folder. Then Cell 5 computes the ergodic distribution of bonds, capital and shocks. To compute the Euler errors, the Euler equations are evaluated at the equilibrium solutions rather than used to solve for the equilibrium. Hence, `fFiPIt_EulerError.m` uses the equilibrium functions (the last solutions generated by the functions that converged according to the tolerance criterion) in all the relevant terms of the Euler equations.

Cell 5. Compute the Ergodic Distribution.

We compute the ergodic distribution of (b, k, s) by iterating to convergence on the law of motion of the conditional transition probabilities from (b, k, s) (denoted $M_j(b, k, s)$) to (b', k', s') (denoted $M_{j+1}(b', k', s')$) $\forall (b, k, s), (b', k', s') \in \mathbf{B} \otimes \mathbf{K} \otimes \mathbf{S}$. The initial guess (called *mErgDistGuess* in line 404) is a uniform distribution. The law of motion is formed using the decision rules for capital and bonds and the exogenous Markov process of the shocks. Since we have solved for approximately continuous decision rules using bi-linear interpolation, we use a standard modification of this law of motion adjusted for the fact that decision rules do not yield values on the nodes of the bonds and capital grids in general. For every (b, k, s) we find $b_L \leq b'(b, k, s) \leq b_U$ and $k_L \leq k'(b, k, s) \leq k_U$, where b_L, b_U, k_L, k_U are the grid points closest to $b'(\cdot)$ and $k'(\cdot)$. Then we iterate on the conditional distributions as follows:

$$M_{j+1}(b_L, k_L, s') = \sum_s Pr[s'|s] M_j(b, k, s) \left(\frac{b_U - b'(b, k, s)}{b_U - b_L} \right) \left(\frac{k_U - k'(b, k, s)}{k_U - k_L} \right)$$

$$M_{j+1}(b_L, k_U, s') = \sum_s Pr[s'|s] M_j(b, k, s) \left(\frac{b_U - b'(b, k, s)}{b_U - b_L} \right) \left(\frac{k'(b, k, s) - k_L}{k_U - k_L} \right)$$

²We set $\rho^B = \rho^{\bar{\mu}} = 1$ and $\rho^q = 0.3$ because this produced the best convergence performance, but this can change with other parameterizations or in other applications of the algorithm.

$$M_{j+1}(b_U, k_L, s') = \sum_s Pr[s'|s] M_j(b, k, s) \left(\frac{b'(b, k, s) - b_L}{b_U - b_L} \right) \left(\frac{k_U - k'(b, k, s)}{k_U - k_L} \right)$$

$$M_{j+1}(b_U, k_U, s') = \sum_s Pr[s'|s] M_j(b, k, s) \left(\frac{b'(b, k, s) - b_L}{b_U - b_L} \right) \left(\frac{k'(b, k, s) - k_L}{k_U - k_L} \right)$$

The convergence criterion is $\max |M_{j+1}(b, k, s) - M_j(b, k, s)| < \epsilon_{Dist} \quad \forall (b, k, s) \in \mathbf{B} \otimes \mathbf{K} \otimes \mathbf{S}$, with the value of ϵ_{Dist} set by the parameter $nTolDist$ in Cell 1.

4 Auxiliary Notes

- Interpolation:** Bi-linear interpolation can be done using the “interp2” Matlab function, but we found that programming the interpolation directly improved the performance of the code. We determine first the interpolation nodes, and then apply the standard bi-linear interpolation rule. The scripts that implement the functions interpolations determine the relevant interpolation nodes and then perform the bi-linear interpolation. To determine the interpolation nodes, for each (b, k, s) , create first vectors with the differences $h^b(b, k, s) = \hat{b}'_j(b, k, s) - b$ and $h^k(b, k, s) = k'_j(b, k, s) - k$, then find the location of the smallest positive difference and smallest negative difference (i.e. the difference closest to zero from below) in these vectors. For example, for the interpolation nodes over the b dimension (b_n, b_{n+1}) , find the locations of $\text{argmin} h^b(b, k, s)$ for $h^b(b, k, s) \geq 0$ and $\text{argmax} h^b(b, k, s)$ for $h^b(b, k, s) \leq 0$. b_n is the location of the argmin and b_{n+1} is the location of the argmax. Once the interpolation nodes are found, the interpolation is executed by calling the `fBiLinearInterpolation.m` function located in the `Mfiles` folder. The scripts also make these adjustments when the interpolated functions return decision rule values outside the state space: Use extrapolation if $k'_j(b, k, s)$ returns a value below (above) the first node k^1 (last node $k^{NCapitalGrid}$) and also if $\hat{b}'_j(b, k, s)$ returns a value above the last node $b^{NBondGrid}$, but for $\hat{b}'_j(b, k, s) < b^1$ evaluate the functions at b^1 , because the lower bound on bonds represents an ad-hoc debt limit used for calibration.
- Parallelization:** There are several loops that run faster in parallel, using *parfor* instead of *for*. This can be done with all loops that do not need to run sequentially. The outmost loop controlling the iterations of the policy and pricing functions needs to be executed sequentially, but several others can be parallelized. *Parfor* can be used in Step 2, 3, 4, and 6. For the *FiPit* variant of Step 6 a sequential sum is needed to attain convergence. We included comments in the code indicating specific loops where *parfor* was used. Using *parfor* requires

Matlab’s Parallel Computing Toolbox. Note also that setting the number of workers to the largest feasible (i.e. the number of processors) does not necessarily minimize execution time, particularly in machines with several processors. In various computers with more than 16 processors, we found that using 6 or 7 workers produced the fastest execution times.

- **Invalid allocations:** Rule out allocations with non-positive arguments in the utility function. These are cases such that, at any iteration and for a given triple (b, k, s) the conjectured functions (indexed by j) or the unconstrained or constrained new decision rules (indexed by $j + 1$) yield $c - L^\omega/\omega \leq 0$. In these cases, the solution of consumption when the constraint does not bind and/or of the multiplier $\tilde{\mu}$ when it binds cannot be obtained because they involve the fractional exponent $1/\sigma$ (for $\sigma > 1$), which requires a positive base. Note that this requirement is stricter than feasibility, because it is not just that the allocations are technologically feasible, they also need to avoid hitting the Inada condition of the CRRA utility function. In the *mainFiPit.m* program, this causes an error that stops execution at the point in which the first attempt to solve for a state of nature with $c - L^\omega/\omega \leq 0$ is encountered. As explained in the paper (see p. 13 and p. 15), however, the *FiPit* algorithm has the advantage that starting from initial conjectures $\hat{b}'_0(b, k, s)$, $\hat{q}_0(b, k, s)$, $\hat{\mu}_0(b, k, s)$ such that the implied labor and consumption decision rules satisfy $c_0(\cdot) - \frac{L_0(\cdot)^\omega}{\omega} > 0$, implies that $c_j(\cdot) - \frac{L_j(\cdot)^\omega}{\omega} > 0$ for any iteration $j > 0$. For the baseline calibration and all six variations we solved for, the initial conditions $\hat{b}'_0(b, k, s) = b$, $\hat{q}_0(b, k, s) = 1$ and $\hat{\mu}_0(b, k, s) = 0$ satisfied this condition.

5 Sketch of Other *FiPit* Applications

We provide here a brief sketch of four examples:

1. Mendoza [1995]: This is an RBC small open economy with incomplete markets and three sectors, quadratic capital adjustment costs given by $(\phi/2)(k_{t+1} - k_t)^2$, and a maximum debt limit as the only occasionally binding constraint. The model has endogenous discounting, but consider a variant with a standard constant discount factor. The adjustment costs formulation does not satisfy the Hayashi conditions required for the average and marginal Tobin’s Q to be the same, but for implementing *FiPit* define a quasi capital pricing function given by $q_t \equiv 1 + \phi(k_{t+1} - k_t)$, so that given a conjecture of this pricing

function we can obtain an implied capital decision rule. Start with this pricing conjecture and a conjectured bonds decision rule. The model's equilibrium conditions, the implied capital decision rule, and the bonds decision rule can be used so that the resource constraint for tradable goods yields an implied decision rule for tradables consumption. *FiPIt* can then be used on the Euler equation for bonds to solve for a new tradables consumption decision rule, and the resource constraint yields a new bonds decision rule. *FiPIt* can then be applied to the Euler equation for capital to solve for a new q function.

2. Ludwig and Schön [2018]: This is a model of optimal human capital accumulation h with a no-borrowing constraint on an asset a that pays an exogenous interest rate R (i.e. a small open economy). Human capital depreciates at rate δ and is produced with a concave function of human capital investment $f(i)$. Agents have CRRA period utility and an exogenous probability of survival given by an increasing, concave function $s(h)$. To solve using *FiPIt*, start with a conjectured decision rule for assets $\hat{A}(a, h)$ and a conjecture for the shadow relative price of human capital investment $\hat{\mu}(a, h)$ where $\mu \equiv \mu/\lambda$ and λ and μ are the multipliers on the resource constraint and law of motion of human capital accumulation respectively. Given these conjectures, the model's equilibrium conditions yield implied decision rules for human capital, consumption and investment in human capital. Then *FiPIt* can be applied to the Euler equation on assets to solve for a new consumption decision rule and using the result in the resource constraint yields a new decision rule for assets $A(a, h)$. If $A(a, h) < 0$, re-define the decision rule as $A(a, h) = 0$, set the associated consumption to the amount supported by the resource constraint, and compute the ratio ψ/λ (where ψ is the multiplier on the no-borrowing constraint). Finally, rewrite the Euler equation for human capital in terms of the ratio μ/λ and apply *FiPIt* to solve for a new decision rule for $\tilde{\mu}$. There is no need to use a root-finder in this case.
3. Mendoza and Smith [2006]: This is a stochastic model of a small open economy in which agents trade in world bond markets and in a market where equity on the economy's capital can be bought by foreign investors, who face a quadratic cost of purchasing equity. A productivity shock affects equity returns. There is a credit constraint imposing a limit on the ratio of debt to the market value of the equity holdings of domestic agents, and a short-selling limit on the equity position. Given conjectures of the decision rule for bonds and the equity pricing function, the optimality condition of foreign investors

and market clearing conditions yield an implied decision rule for equity holdings (the quadratic adjustment cost plays a role similar to the capital adjustment cost in the SS model we solved earlier). Given these, the resource constraint of the small open economy yields a decision rule for consumption. Assuming the credit constraints do not bind, *FiPIt* can then be applied to the bonds Euler equation to solve for a new consumption decision rule, and the resource constraint yields a new bonds decision rule. If the latter yields a value that violates the credit constraint, the constraint is imposed with equality to obtain new values for the bonds decision rule and consumption, and for the ratio of the multiplier of the borrowing constraint. Finally, *FiPIt* is applied to the Euler equation for equity holdings to obtain a new equity pricing function.

4. Huggett [1993]: This is one of the canonical heterogeneous agents models in which a continuum of agents trade non-state-contingent debt facing idiosyncratic Markov income shocks and a maximum debt limit. The optimization problem solved by an individual agent, who takes an exogenously-determined value of the interest rate as given, is identical to that of the small open endowment economy studied in Section 2, which has only one endogenous state variable. Start with a conjectured decision rule for bonds, use the resource constraint to obtain the implied decision rule for consumption. Then apply *FiPIt* to solve for a new consumption decision rule, and use the resource constraint to obtain a new bonds decision rule. If the latter violates the maximum debt limit, redefine to bonds decision rule to match the debt limit and set the associated consumption decision rule to the amount supported by the resource constraint. Iterate to convergence on the bonds decision rule and then use the decision rules and Markov process of income shocks to compute the ergodic distribution of bonds and income (i.e. the wealth and income distribution across agents). The difference with the small open economy is that now the interest rate is also part of the solution. The ergodic distribution is used to compute the aggregate demand for bonds (i.e. the mean of asset demand across agents), which must be equal zero at equilibrium in order to clear the bond market. If it yields exceeds demand (supply), the interest rate is reduced (increased) until the market-clearing condition holds up to a convergence criterion. Again *FiPIt* does not require a root finder.

References

Mark Huggett. The risk-free rate in heterogeneous-agent incomplete-insurance economies. *Journal of economic Dynamics and Control*, 17(5-6):953–969, 1993.

Alexander Ludwig and Matthias Schön. Endogenous grids in higher dimensions: Delaunay interpolation and hybrid methods. *Computational Economics*, 51(3):463–492, 2018.

Enrique G Mendoza. The terms of trade, the real exchange rate, and economic fluctuations. *International Economic Review*, pages 101–137, 1995.

Enrique G Mendoza and Katherine A Smith. Quantitative implications of a debt-deflation theory of sudden stops and asset prices. *Journal of International Economics*, 70(1):82–114, 2006.