Efficient Solution and Computation of Models with Occasionally Binding Constraints

Gregor Boehl

University of Bonn

July 6, 2022

Abstract

Structural estimation of macroeconomic models and new HANK-type models with extremely high dimensionality require fast and robust methods to efficiently deal with occasionally binding constraints (OBCs). This paper proposes a novel algorithm that solves for the perfect foresight path of piecewise-linear dynamic models. In terms of computation speed, the method outperforms its competitors by more than three orders of magnitude. I develop a closed-form solution for the full trajectory given the expected duration of the constraint. This allows to quickly iterate and validate guesses on the expected duration until a perfect-foresight equilibrium is found. A toolbox, featuring an efficient implementation, a model parser and various econometric tools, is provided in the Python programming language. Benchmarking results show that for medium-scale models with an occasionally binding interest rate lower bound, more than 150,000 periods can be simulated per second. Even simulating large HANK-type models with almost 1000 endogenous variables requires only 0.2 milliseconds per period.

Keywords: Occasionally Binding Constraints, Effective Lower Bound, Computational Methods
JEL: E63, C63, E58, E32

1 Introduction

Occasionally binding constraints have become an important part of economic modelling, especially since major central banks have seen themselves constrained by the zero lower bound (ZLB) of the nominal interest rate. A binding ZLB constraint poses a major problem for quantitative analysis and the estimation of macroeconomic models: conventional, purely linear solution methods are unable to handle nonlinearities such as the

Email: gboehl@uni-bonn.de. *Address:* Institute for Macroeconomics and Econometrics, University of Bonn, Adenauerallee 24-42, 53113 Bonn, Germany. I am grateful to Flora Budianto, Hans-Martin von Gaudecker, Alexander Meyer-Gohde, Tom Holden, Kenneth Judd, Alexander Richter, Felix Strobel, two anonymous reviewers and participants of the 2018 Stanford MMCI Conference, the 2018 EEA Annual Congress and the 2018 VfS Jahrestagung for discussions and helpful comments on the contents of this paper. Part of the research leading to the results in this paper has received financial support from the Alfred P. Sloan Foundation under the grant agreement G-2016-7176 for the Macroeconomic Model Comparison Initiative (MMCI) at the Institute for Monetary and Financial Stability. I gratefully acknowledge financial support by the DFG under CRC-TR 224 (projects C01 and C05) and under project number 441540692.

ZLB, and existing alternatives tend to be computationally demanding. The urge to investigate macroeconomic questions related to the Great Recession or the Covid-19 crisis in a quantitative-structural framework requires algorithms that are not only accurate, but that are also fast and computationally efficient.

Efficient and fast methods for occasionally binding constraints (OBCs) are particularly important for two applications: first, the Bayesian estimation of macroeconomic models, and, secondly, model with extremely high dimensional state spaces, such as novel heterogeneous agents models. The ZLB in the US was binding from 2008-2015 and from 2020 - early 2022. This calls for methods that are fast enough to explicitly account for endogenously binding ZLB to include the recent data during the estimation process. Further, a growing and promising literature is developing large-scale New Keynesian models that feature heterogeneous agents and a large number of idiosyncratic states (HANK models, see e.g. Reiter, 2009; Kaplan et al., 2018; Bayer et al., 2020). While it is possible to solve linear approximations of these models, researchers naturally wish to also consider aggregate OBCs such as the ZLB in this framework. However, the fact that these models can easily have 1000 dimensions and simply overburdens the currently available methods.

This paper aims to fill this gap, and at the same time seeks to significantly expand the range of possible applications of models with OBCs. Given any dynamic general equilibrium model with one or several OBCs, I develop a closed-form, non-recursive state-space representation for the complete expected trajectory of the endogenous variables as a function of the expected duration at the ZLB (the "ZLB spell duration"). This solution is based on a piecewise-linear approximation of the original model. I then provide the necessary conditions of a perfect foresight equilibrium for a set of ZLB spell durations given the state of the economy. Given these two ingredients, perfect foresight equilibria can be found via a simple iterative scheme.

Computational advantage also depends on the efficient implementation of the algorithm. A reference implementation of the proposed solution method, together with a parser and related econometric tools, is implemented in the Pydsge package. The package is freely available and actively developed on GitHub.[1] The implementation is written in the free and open-source programming language Python and, similar as the Dynare package (Adjemian et al., 2011), presents a complete toolbox which allows to simulate, filter and estimate generic DSGE models.

The massive gain in computational efficiency hence stems from four elements: first, using the closed-form solution together with the equilibrium conditions allows to check for a model equilibrium instantaneously instead of having to simulate the complete anticipated trajectory for a given ZLB spell. This makes the guess-and-verify step extremely fast. Second, I only focus on equilibria with a single spell duration. While this rules out equilibria where the constraint, in expectations, binds more than once, it reduces the search-domain considerably. Third, the closed form solution allows for efficient decomposition of input matrices, which significantly reduces the time to preprocess the matrices. Lastly, the implementation in Python allows for just in time compilation which brings it on the same level with C or Fortran in terms of execution speed.

The method presented here can be seen as complementary to the method of Holden (2016, forthcoming). In particular, for applications where high computation speed is not

[1] See `https://github.com/gboehl/pydsge`.

crucial, their method has a number of advantages. First, for a given initial condition they can provide *all* possible equilibria (up to a given horizon), i.e. all feasible sets of durations of the constraint. Importantly, this may also include equilibria where the constraint in anticipation binds more than once, i.e. agents anticipate that the constraint is binding until period 7 and then again in period 9. In contrast, those equilibria are ruled out by the method suggested here, which may imply limitations on the types of models that can be solved.[2] Additionally, Holden (forthcoming) is also able to provide existence and uniqueness results for models with OBCs. In contrast, whenever the method presented here is unable to find an equilibrium, it remains silent on the reasons why. While these limitations relative to Holden (2016) may be severe for some applications, the speed advantage of my algorithm may be essential for other applications.

Other solution concepts for DSGE models with OBCs exist.[3] Another algorithm that is currently in frequent use is named OccBin, which was introduced in Guerrieri and Iacoviello (2015). The authors propose a recursive representation of the solution given the state of the economy and a set of spell durations. They propose a Newton-like method to iteratively find the set of spell durations. The relatively high computational costs of their approach stems from the fact that for each guess of the spell durations, the Newton-like method requires the complete simulation of the anticipated trajectory. This requires repeated matrix inversions at runtime, which are computationally expensive. While my method shares some features with their algorithm it has a considerable advantage in terms of computation speed and, hence, is also well suited for parameter inference as well as for very high dimensional models.

I provide speed benchmarks of Pydsge for two applications, both featuring an originally linear model that is extended by an endogenously binding ZLB: I first run benchmarks using a standard medium-scale New-Keynesian model as in Smets and Wouters (2007) and then using the linearized HANK model of Bayer et al. (2020). For the medium scale model, my method performs one evaluation of a state vector in less than seven microseconds ($\frac{7}{1000}$ ms) or more than 170,000 evaluations per second. For the HANK-type model with almost 1000 endogenous variables these are still about 0.2 ms per evaluation or 4000 evaluations per second. In comparison, OccBin takes a bit more than 0.005 seconds per evaluation, corresponding to 150 evaluations per second. This implies a speed advantage of the method presented here by a factor of 1500. Since the results obtained from the method presented here is, given uniqueness, identical to that of OccBin, I refer to Guerrieri and Iacoviello (2015) for a comparison with other, fully nonlinear methods such as policy function iteration as used e.g. by Atkinson et al. (2019) or Gust et al. (2017).[4]

The rest of this paper is structured as follows. Section 2 develops the solution method and discusses details regarding the implementation. In Section 3 I provide speed benchmarks. Section 4 concludes.

[2] The benchmarking exercise in Section 3 suggests that for an estimated medium-scale model, ZLB spells where the ZLB binds more than once in expectations do not occur in practice. Yet, this may not be true for all models or for different forms of OBCs.

[3] Early work with nonlinear models also includes e.g. Coenen and Wieland (2003, 2004); Coenen et al. (2004).

[4] The use of fully nonlinear methods has, next to the benefit of an exact transmission of the model equations, the advantage of not ignoring aggregate uncertainty in the agents decision functions.

2 Method

The large speedup of the method provided here comes from levering the closed-form solution for the anticipated trajectory of the constrained variable. This solution is presented in Section 2.2. Section 2.3 then presents the necessary conditions for this anticipated trajectory to be an equilibrium, which can be used in the context of an iterative method. Before that, I will start out by providing some prerequisites.

2.1 Prerequisites

Assume a linear rational expectations model that is subject to $j = 1, 2, \ldots, n_c$ OBCs of the form

$$r_{j,t} = \max\left\{a_j y_{t+1} + b_j y_t + c_j y_{t-1}, \bar{r}_j\right\}. \tag{1}$$

Inspired by Uhlig et al. (1995); Binder and Pesaran (1995); Villemot et al. (2011), the linearized first order conditions[5] of this system can be represented as

$$E_t\left[\mathfrak{A} z_{t+1} + \mathfrak{B} z_t + \mathfrak{C} z_{t-1} + \mathfrak{D}\epsilon_t + \sum_j^{n_c} \mathfrak{h}_j \max\left\{\mathfrak{a}_j z_{t+1} + \mathfrak{b}_j z_t + \mathfrak{c}_j z_{t-1} + \mathfrak{d}_j \epsilon_t, \bar{r}_j\right\}\right] = 0, \tag{2}$$

where z_t is the n-dimensional vector of all model variables and ϵ_t the n_ϵ-dimensional vector of iid. exogenous shocks. $\mathfrak{A}$, $\mathfrak{B}$ and $\mathfrak{C}$ are generic $n \times n$ system matrices whereas, $\mathfrak{D}$ is a $n \times n_\epsilon$ matrix. Equation (2) can elegantly be reduced to

$$E_t\left[A y_{t+1} + B y_t + C y_{t-1} + \sum_j^{n_c} h_j \max\left\{a_j y_{t+1} + b_j y_t + c_j y_{t-1}, \bar{r}_j\right\}\right] = 0, \tag{3}$$

with $y_t = (z_t, \epsilon_{t+1})$, $A = \begin{bmatrix}\mathfrak{A} & 0\\ 0 & 0\end{bmatrix}$, $B = \begin{bmatrix}\mathfrak{B} & 0\\ 0 & I\end{bmatrix}$, $C = \begin{bmatrix}\mathfrak{C} & \mathfrak{D}\\ 0 & 0\end{bmatrix}$ and $a_j = (\mathfrak{a}_j, 0)$, $b_j = (\mathfrak{b}_j, 0)$, $c_j = (\mathfrak{c}_j, \mathfrak{d}_j)$, $h_j = (\mathfrak{h}_j, 0)$ for each constraint j. Each constrained variable $r_{j,t} \forall j \in 1, 2, \cdots, n_c$ is subject to an occasionally binding constraint represented by a set of vectors $\{a_j, b_j, c_j\}$ and a minimum value of $r_{j,t}$ denoted $\bar{r}_j$. $h_j \in \mathbb{R}^n$ contains the coefficients of $r_{j,t}$ in each equation of the linear system.

Denote the system in which all constraints are slack (the *unconstrained system*) as

$$\hat{A} y_{t+1} + \hat{B} y_t + \hat{C} y_{t-1} = 0, \tag{4}$$

[5] See e.g. Schmitt-Grohé and Uribe (2004) for pertubation methods to obtain a linear representation from the nonlinear model.

with

$$\hat{A} = A + \sum_{j}^{n_c} h_j a_j, \tag{5}$$

$$\hat{B} = B + \sum_{j}^{n_c} h_j b_j, \tag{6}$$

$$\hat{C} = C + \sum_{j}^{n_c} h_j c_j, \tag{7}$$

which means, e.g., that the unconstrained matrix $\hat{B}$ for contemporary variables y_t is the matrix B of the system in which all constraints bind plus a correction matrices for each $r_{j,t}$, which contains the endogenous responses of $r_{j,t}$ if constraint j is slack.

Let me borrow the Assumptions 1 and 2 from Holden (2016) and Rendahl (2017) which maintain the existence and uniqueness of a solution for the unconstrained linear system.

Assumption 1. *For any given $y_0 \in \mathbb{R}^n$, Equation (4) has a unique solution, which takes the form $y_t = Fy_{t-1}$ for $t \in \mathbb{N}^+$, where $F = -(\hat{B} + \hat{A}F)^{-1}\hat{C}$, and where all the eigenvalues of F are weakly inside the unit circle.*

Assume further, to imply that all the eigenvalues of F are strictly inside the unit circle, that:

Assumption 2.

$$\det\left(\hat{A} + \hat{B} + \hat{C}\right) \neq 0. \tag{8}$$

Lastly, we are solving the model under *perfect foresight*:

Assumption 3. *Agents expect all shock innovations after time-t to be zero*

This implies that the expectations operator E_t can be omitted in the following, as all uncertainty is resolved after t.

2.2 A closed-form representation conditional on the spell duration

Equation (3) can be cast in the form (see e.g. Klein (2000) or Villemot et al. (2011)):[6]

$$Px_{t+1} = Mx_t + \sum_{j}^{n_c} h_j \max\left\{p_j x_{t+1} + m_j x_t, \bar{r}_j\right\}, \tag{10}$$

with $x_t = \begin{bmatrix} s_{t-1} \\ c_t \end{bmatrix}$, where c_t are the forward looking variables (controls) and s_{t-1} are the states updated by the time-t shocks as above. Each OBC now reads as $r_{j,t} = \max\left\{p_j x_{t+1} + m_j x_t, \bar{r}_j\right\}$.

[6]The fundamental idea is that

$$Ay_{t+1} + By_t + Cy_{t-1} = 0 \quad \Leftrightarrow \quad \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} y_{t+1} \\ y_t \end{bmatrix} + \begin{bmatrix} 0 & C \\ -I & 0 \end{bmatrix} \begin{bmatrix} y_t \\ y_{t-1} \end{bmatrix} = 0. \tag{9}$$

Analog to equations (5)-(7), denote the system in which *all* constraints are slack (the *unconstrained system*) as

$$\hat{P}x_{t+1} = \hat{M}x_t, \tag{11}$$

with

$$\hat{P} = \left(P + \sum_j^{n_c} h_j p_j\right) \qquad \text{and} \qquad \hat{M} = \left(M + \sum_j^{n_c} h_j m_j\right), \tag{12}$$

which are again the system matrices of the constraint system plus the sum of correction matrices for the endogenous responses.

To fix ideas, let us focus on the case with only one constraint, the latter being given by (h_0, p_0, m_0). The generalization to many occasionally binding constraints is then straightforward and left to the reader. Assume further that P and $\hat{P}$ are invertible. The more general case where this is not satisfied is presented in Appendix A.

Under this assumption, System (10) can be rewritten as

$$x_{t+1} = \begin{cases} \hat{G}x_t & \text{if } \; p_0 x_{t+1} + m_0 x_t - \bar{r} \geq 0 \\ \\ Gx_t + q_0 \bar{r} & \text{if } \; p_0 x_{t+1} + m_0 x_t - \bar{r} < 0, \end{cases} \tag{13}$$

with $\hat{G} = \hat{P}^{-1}\hat{M}$, $G = P^{-1}M$ and $q_0 = P^{-1}h_0$. If the constraint is slack the upper case holds with $x_{t+1} = \hat{G}x_t$, otherwise the lower case holds.

To simplify notation, let me first assume that if the constraint binds, it always binds already in the current period t (hence there is *no transition* to the constraint). Let k_t be the expected duration of the spell to the constraint in period t. Denote a rational expectations solution to (10) given k_t and the state variables s_{t-1} as the function λ such that

$$c_t = \lambda(k_t, s_{t-1}). \tag{14}$$

In slight abuse of notation, I will occasionally use k and $\lambda(k)$ as shorthand where the states s_{t-1} and the time-t subscripts are understood.

For the unconstrained system $\hat{G}$, the controls c_t can be found using familiar methods like the QZ-decomposition as suggested by Klein (2000). Denote this (linear) solution for c_t by the matrix Ω:[7]

$$c_t = \Omega s_{t-1} \quad \text{if } \; p_0 x_{t+1} + m_0 x_t > \bar{r} \tag{15}$$

For $\Psi = \begin{bmatrix} -\Omega & I \end{bmatrix}$, Equation (15) implies that

$$\Psi \begin{bmatrix} s_{t+k} \\ c_{t+k+1} \end{bmatrix} = 0 \quad \text{if } \; p_0 x_{t+k+1} + m_0 x_{t+k} \geq \bar{r}, \tag{16}$$

i.e. this must hold for every future period $t + k$ in which the system is expected to be unconstrained.

[7] Note that Ω and F are closely related: while F is the solution in variables space (mapping y_{t-1} to y_t), Ω is the solution of controls in terms of states (mapping s_{t-1} to c_t). This means that Ω can be recovered from F and vice versa.

Now assume that the constraint binds at time t and will continue to do so until period $t+k$. Iterating the bottom part of System (13) k periods forward yields

$$\begin{bmatrix} s_{t+k-1} \\ c_{t+k} \end{bmatrix} = G^k x_t + (I-G)^{-1}(I-G^k)q_0\bar{r}, \tag{17}$$

where G^k denotes the matrix power and $(I-G)^{-1}(I-G^k) = \sum_{i=0}^{k-1} G^i$ comes from the transformation for a geometric series of matrices. Finally, we can combine Equations (16) and (17) to find λ, i.e. a solution of the controls c_t in terms of the state variables s_{t-1} given k:

$$\lambda(k, s_{t-1}) = \left(c_t : \Psi G^k \begin{bmatrix} s_{t-1} \\ c_t \end{bmatrix} = -\Psi(I-G)^{-1}(I-G^k)q_0\bar{r} \right). \tag{18}$$

Since q_0 is a vector of constants, the whole RHS of Equation (18) is known and solving for c_t is straightforward. In other words, Equation (18) transmits the impulse of the initial conditions forwards through the states. At the terminal date $t+k$, the controls are implied by the standard infinite horizon solution. These states can then be backwards-propagated through time to find the *current* sets of controls.

Let us now relax the assumption that a shock triggers the constraint to hold immediately in time t. This case is in particular relevant for models that feature persistent *endogenous* state variables. Take Equation (18) as the starting point and allow for a number of periods l in the unconstrained system $\hat{G}$ *until* the system is at the constraint. We can proceed in a similar manner as above to obtain

$$\lambda(l, k, s_{t-1}) = \left(c_t : \Psi G^k \hat{G}^l \begin{bmatrix} s_{t-1} \\ c_t \end{bmatrix} = -\Psi(I-G)^{-1}(I-G^k)q_0\bar{r} \right). \tag{19}$$

Let $[x_{t+j}|(l,k)]$ denote the perfect foresight solution of variable vector x_{t+j} in period $t+j$, at time t conditional on assuming spell durations of (l_t, k_t). Using Equation (19) and Equation (17) augmented by $\hat{G}^l$, we can express $[x_{t+j}|(l_t,k_t)]$, as a function Λ with

$$\begin{aligned} [x_{t+j}|(l_t,k_t)] = \Lambda_j(l_t, k_t, s_{t-1}) =& G^{\max\{j-l_t,0\}} \hat{G}^{\min\{l_t,j\}} \begin{bmatrix} \lambda(l_t,k_t,s_{t-1}) \\ s_{t-1} \end{bmatrix} \\ &+ (I-G)^{-1}\left(I - G^{\max\{j-l_t,0\}}\right) q_0\bar{r}. \end{aligned} \tag{20}$$

Note that $\Lambda_1(0,0,s_{t-1})$ is the generic solution to the unconstrained system with $\lambda(0,0,s_{t-1}) = \Omega s_{t-1}$. This is the key equation of the algorithm. It states that for any guess on (l,k), the expected value in any period $t+j$ of all endogenous variables, x_{t+j}, can be readily obtained.

2.3 Solving for the spell durations (l,k)

Again, let us first consider the simpler case in which we assume that any shock that causes the constraint to bind, it will cause it to bind immediately in time t (the *non-transitory* case). The following proposition summarizes the conditions for (x_t, s_{t-1}, k) to be a rational expectations equilibrium:

Proposition 1 (non-transitory equilibrium)**.** *Assuming non-transition, a number of expected periods k^* at the constraint is part of a rational expectations equilibrium for the initial state s_{t-1} iff*

i)

$$p_0\Lambda_{k+1}(0,k^*,s_{t-1}) + m_0\Lambda_k(0,k^*,s_{t-1}) \geq \bar{r} > p_0\Lambda_{k+1}(0,k,s_{t-1}) + m_0\Lambda_k(0,k,s_{t-1}) \tag{21}$$

for all $k \in \{0,\dots,k^*-1\}$, hence if in expectations the system is constrained for exactly k^* periods, and

ii)

$$p_0\Lambda_{k+1}(0,k^*,s_{t-1}) + m_0\Lambda_k(0,k^*,s_{t-1}) \geq \bar{r} \tag{22}$$

for all $k > k^*$, i.e. the system will not return to the constraint after k^*.

Proof. The proof follows from the definition of k and r_t. First acknowledge that $r_{t+j} = p_0 x_{t+j+1} + m_0 x_{t+j}$ and hence the expected value of r in period $t+j$ conditional on a guess k^* is given by

$$[r_{t+j}|k^*] = p_0[x_{t+j+1}|k^*] + m_0[x_{t+j}|k^*]. \tag{23}$$

Further, through (20) we have that $[x_{t+j}|k^*] = \Lambda_j(0,k^*,s_{t-1})$. For k^* to be an equilibrium it is required that the constraint binds for exactly k^* periods. This would be violated if $[r_{t+j}|k^*] > \bar{r}$ for any $j < k^*$. Similarly, k^* cannot be an equilibrium if $[r_{t+j}|k^*] < \bar{r}$ for any $j > k^*$. ■

Proposition 1 states that k^* is only an equilibrium if the trajectory for assuming $k = k^*$ is at the constraint for exactly k^* periods, and any $k < k^*$ is not an equilibrium because it is inconsistent ($\bar{r} > p_0[x_{t+k+1}|k] + m_0[x_{t+k}|k]$ is inconsistent because r_t as implied by k must just be larger than $\bar{r}$). Let us proceed to the practically relevant case where agents may expect the constraint to remain slack for some transition time before binding for k periods. Using Equation (20), proposition 2 summarizes the respective equilibrium conditions.

Proposition 2 (transitory equilibrium)**.** *A pair (l^*,k^*) is part of a rational expectations equilibrium iff*

i)

$$p_0\Lambda_{j+1}(l^*,k^*,s_{t-1}) + m_0\Lambda_j(l^*,k^*,s_{t-1}) \geq \bar{r} \tag{24}$$

for all $j < l^* \wedge j \geq k^* + l^*$, and

ii)

$$p_0\Lambda_{j+1}(l^*,k^*,s_{t-1}) + m_0\Lambda_j(l^*,k^*,s_{t-1}) < \bar{r} \tag{25}$$

for all $j \in \{l^*,\dots,k^*+l^*-1\}$.

Proof. The proof is analogue to the proof of proposition 1. Additionally, for l^* to be an equilibrium it is required that the constraint is slack for exactly l^* periods, before becoming binding in period l^*. This requires that

$$[r_{t+j}|(l^*,k^*)] > \bar{r} \qquad \forall\, 0 \leq j < l^* \tag{26}$$

since otherwise l^* cannot be an equilibrium.

In other words, (l, k) are part of an equilibrium, if in expectations, the constraint starts binding exactly in period $t + l$ and ends to bind exactly in period $t + l + k$. ■

In simple words, proposition 2 states that for (l^*, k^*) to be an equilibrium, it must be that neither before l^* nor after $l^* + k^*$ (the presumed exit) the economy is at the constraint. Additionally it is required that in the period from l^* to $l^* + k^*$, the constraint is binding.

Unfortunately there is no closed form solution for (l^*, k^*) given initial conditions s_{t-1}. A set of (l^*, k^*) that satisfies proposition 2 must be found using an iterative scheme. As this problem requires an iterative scheme on an integer domain, a theoretical assessment is at least difficult because most theoretical work on similar algorithms deals with real valued functions. Given those limits, insights regarding the existence and uniqueness of such solutions are provided by Holden (forthcoming). Note that an integer based Newton-like method as employed by Guerrieri and Iacoviello (2015) is not efficient because it requires the evaluation of the complete anticipated trajectory of x_t and furthermore can not draw on the nice convergence properties of the regular real-valued Newton method. This renders their method impractical once OBC problems are not well behaved. Note that for a given s_{t-1}, an equilibrium satisfying conditions (24) and (25) will generally be the same as in Holden (forthcoming), if it is unique.

The crucial advantage of the formulation provided above is the closed form expression of $[r_{t+j}|(l, k)]$. Given (j, l, k) and using the function Λ in (20), the mapping $[r_{t+j}|(l, k)] : s_{t-1} \to \mathbb{R}$ is linear in s_{t-1} and consists of a pair $(z_1(j, l, k), z_2(j, l, k))$, which is a row vector and a scalar. To further increase efficiently when checking the conditions in proposition 2, these pairs can be pre-processed and stored. Under the assumption that there is only a single ZLB spell, it is sufficient to check that r_{t+j} satisfies Proposition 2 only right before and right after a presumed system change. To cover all potentially relevant cases from proposition 2 then only requires the coefficients in $[r_{t+j}|(l, k)]$ for $j \in \{0, l-1, l, l+k-1, l+k\}$, leaving us to only pre-calculate a number of $\frac{k_{\max}(l_{\max}-1)}{2}5$ pairs.

An optimal iterative scheme can be hand-tailored to the problem. For the purpose of the estimation of large-scale DSGE models, in which the constraint is the zero lower bound on nominal interest rates, Boehl et al. (forthcoming); Boehl and Lieberknecht (2021); Boehl and Strobel (2022a) and Boehl and Strobel (2022b) use the following iterative scheme, where `LAM(j,l,k,s)` represents the function $\Lambda_j(l, k, s_{t-1})$ and `s` is the current state vector s_{t-1}:

```
l, k = 0, 0
for l from 0 to l_max:
    if b LAM(l, l, 0, s) - r_bar < 0:
        # constraint expected to bind: interrupt loop and continue by finding k
        break
    if l is l_max - 1:
        # return that l=0, k=0 is an equilibrium since the economy is predicted
        # to not touch the constraint until at least l_max
        return (0, 0)
...
```

Hence, if the constraint is not reached within `l_max` periods ahead in the future, exit and assume that the constraint will not be binding. Otherwise (`break`-statement) assume $k > 0$ and iterate over l and k until the equilibrium conditions in (21), (24) and (25) are satisfied. Figure 1 continues the scheme for the case that the above evaluated that $k > 0$. This scheme is very efficient for the specific problem because for more than 50% of the data points used the ZLB is not binding and the method will already exit in the first loop.[8]

```
...
for l from 0 to l_max:
    for k from 1 to k_max:
        if l > 0:
            # only execute these if necessary
            if b LAM(0, l, k, s) - r_bar < 0:
                # constraint binding in t=0 -> violates l > 0
                skip to next item in loop
            if b LAM(l-1, l, k, s) - r_bar < 0:
                # constraint binding in t=l-1 -> violates that is is
                # supposed to start binding in l
                skip to next item in loop
        if b LAM(k+l, l, k, s) - r_bar < 0:
            # constraint binding in t=k+l -> violates that is is supposed to
            # stop binding in k+l-1
            skip to next item in loop
        if b LAM(l, l, k, s) - r_bar > 0:
            # constraint not binding in t=l -> violates that is is supposed to
            # start binding then
            skip to next item in loop
        if b LAM(k+l-1, l, k, s) - r_bar > 0:
            # constraint not binding in t=k+l-1 -> violates that is is supposed to
            # be still binding until then
            skip to next item in loop
        # if none of the above was true, the current guess
        # of (l,k) must be an equilibrium
        return (l, k)
# if the loop went though without finding an equilibrium, throw a warning
# and/or set error flag
flag = True
warn('No equilibrium found!11')
```

Figure 1: Pseudo code for an iterative loop to solve models with a ZLB.

If the loop does not exit, hence if presumably $k^* > 0$, then for data points after 2008 it is predominantly the case that the ZLB already is binding. In this case $l^* = 0$ and only k^* is to be determined. As, according to the Primary Dealer Survey, most market

[8] For example, Boehl et al. (forthcoming) use US data from 1998 to 2020, which contains the ZLB period from 2008Q4 to 2015Q4.

participants expected the ZLB to be binding for about eight quarters, the procedure will on average need 8 guesses until an equilibrium is found. If at all, l^* will only be positive in 2008Q3 when the economy is not yet at the ZLB, but the shocks originating from the Subprime Mortgage Crisis have triggered the ZLB to be expected to bind in the very near future.

While the above procedure is set-up to work most efficiently in the context of estimating DSGE models with the ZLB, it is generic and applicable to any sort of constraint.[9] The resulting transition function is linear for the region where the constraint does not bind and (increasingly) nonlinear when it binds. Note that this algorithm includes an active assumption on equilibrium selection: if for a s_{t-1} several sets of (l^*, k^*) exist that satisfy proposition 2, then implicitly, the set with the lowest l^* is chosen.

3 Benchmarking Compuation Speed

Speed is the key aspect of the design of the algorithm introduced in the previous section. This section presents benchmarks. For this purpose I consider the medium-scale model of Smets and Wouters (2007) and the Heterogeneous Agent New Keynesian (HANK) model of Bayer et al. (2020), both extended by the lower bound on the nominal interest rate as an occasionally binding constraint. The parameter values in use are documented in Appendix B. The parameter values for the medium scale model is obtained from estimating the model to US data from 1998 to 2020 using the presented method together with the nonlinear filtering methodology of Boehl and Strobel (2022a). In the data, the ZLB binds from 2008 to 2015.[10] The HANK model is estimated to US data from 1954 to 2019, but without explicitly accounting for an endogenously binding ZLB during the estimation.[11] While the first example is chosen because it is the workhouse model of modern monetary economics which is widely used by central banks around the world, the second model is chosen because of its high dimensionality with a total of 993 endogenous variables.

I benchmark Pydsge against OccBin (c.f. Guerrieri and Iacoviello, 2015) which is implemented in Dynare (see Adjemian et al., 2011) and used frequently in applied work. I use the implementation from Cuba-Borda et al. (2019) which goes beyond the current standard implementation in Dynare. In particular, by caching intermediate results it avoids solving the model repeatedly, and processing the system matrices for every new initial condition. Notably, these steps together were accountable for about 98% of the computation time of the original implementation of Guerrieri and Iacoviello (2015).[12] Benchmarking against the implementation of Cuba-Borda et al. (2019) hence biases the comparison of the two methods favorably for OccBin. For the benchmarks of OccBin,

[9] Note that the ordering of checks in Figure 1 is chosen such that the number of checks before exiting is minimized.

[10] The inclusion of the ZLB in the estimation procedure, apart from its obvious appeal for economic analysis, has the further advantage that regions of the parameter space in which ZLB solutions are irregular are not included in the posterior distribution as they necessarily exhibit a lower likelihood.

[11] The system matrices were kindly provided by the authors.

[12] When, instead, benchmarking against the original implementation, the method presented here even performs more than 10,000 times faster.

	Pydsge			Occbin		
	mean	std	% of draws	mean	std	% of draws
$k = 0$	6.03e-06	2.55e-04	71.19%	4.29e-03	8.52e-04	71.19%
$k \in (1, 5)$	6.83e-06	5.92e-06	9.76%	8.28e-03	1.80e-03	9.76%
$k \in (6, 10)$	6.40e-06	3.66e-06	10.35%	1.16e-02	2.90e-03	10.35%
$k \in (11, 15)$	5.82e-06	1.60e-06	7.45%	1.42e-02	2.90e-03	7.45%
$k \in (16, 20)$	5.90e-06	8.55e-07	1.25%	1.66e-02	3.37e-03	1.25%
$k > 20$	6.57e-06	3.06e-07	0.01%	1.79e-02	2.83e-03	0.01%
$l > 0 \vert k > 0$	1.55e-05	8.36e-06	2.96%	1.08e-02	3.33e-03	2.96%
No solution found	–	–	0.00%	2.40e-01	3.27e-01	0.00%
Total	6.13e-06	2.15e-04	100.00%	6.33e-03	4.09e-03	100.00%

Table 1: Speed benchmark of Pydsge and OccBin for the medium-scale model of (Boehl and Strobel, 2020) (57 variables).
Note: The table shows the execution times per state vector in seconds. "No solution found" collects draws for which no solution was found (two draws in total for OccBin). "Preprocessing" times include the time spend for solving the linear model using Klein (2000) and preprocessing all system matrices. Preprocessing is not benchmarked for OccBin.

Matlab version R2019a is used with Dynare 4.6.3.[13] I do not benchmark the HANK model as the Dynare preprocessor is incapable of handling matrices of this size.

For each exercise I draw 100,000 state vectors from a multivariate normal distribution with zero mean and covariance $\tau\Sigma$, where Σ is the ergodic distribution of states as implied by the linear model, and $\tau \geq 1$ is a scalar. For the model of Smets and Wouters (2007) I set $\tau = 5$ to inflate the covariance such that a notable number of draws with $l > 0$ or $k > 20$ are generated.[14] For the same reason, τ is set equal 2 for HANK. Each sample is passed trough the nonlinear transition function and grouped according to its calculated ZLB duration. I set `l_max`=8 and `k_max`=30, which is enough to ensure that for all samples a valid equilibrium is found.[15] If within this range no ZLB equilibrium is found, the sample counts as "No solution found". Note that there are many reasons why an equilibrium may not be found. Apart from the actual non-existence an equilibrium, potential reasons may be, e.g., that the ZLB may bind more than once in expectations, or the reversals documented by Carlstrom et al. (2015). For OccBin, `nperiods` is also set to 30. Although this causes a very small number of samples to not converge, it decreases computation time again favorably for Occbin.[16]

For the method presented in Section 2, the overall computation times are 178,249 draws per second (0.561 seconds for 100,000 draws in total) for the model of Smets and

[13] Comparing a Python package with a Matlab package involves the problem that, given equivalently efficient implementation, Python packages are likely to perform better due to the superiority of the language design. This alone, however, is not sufficient to explain the significant speed advantage documented here.

[14] For $\tau = 1$ the ZLB will never bind longer than 15 periods in expectations.

[15] Under empirically relevant calibrations, shocks for which the number of expected periods *until* the ZLB binds is larger than 3 are extremely rare. Even reverse engineering such a calibration is challenging because $l > 3$ implies extremely slow and persistent endogenous transmission of shocks.

[16] Only for 2 of all 100 draws no solution is found. This number can be squeezed down to zero by setting `nperiods` to 100.

	mean	std	% of samples
$k = 0$	2.010e-04	3.638e-05	66.89%
$k \in (1, 5)$	2.611e-04	2.160e-05	4.50%
$k \in (6, 10)$	2.638e-04	2.592e-05	1.24%
$k \in (11, 15)$	2.731e-04	1.842e-05	0.25%
$k \in (16, 20)$	2.557e-04	2.544e-05	8.41%
$k > 20$	2.712e-04	2.182e-05	2.60%
$l > 0 \vert k > 0$	2.887e-04	3.961e-05	0.01%
No solution found	3.090e-04	3.989e-05	16.12%
Total	2.285e-04	5.448e-05	100.00%
Preprocessing times (once per parameter draw):			
Solving linear model	6.123	0.204	
Preprocessing OBC	44.354	0.711	

Table 2: Speed benchmark of the method from Section 2 for the HANK model of (Bayer et al., 2020) with 993 variables.
Note: The table shows the execution times per state vector in seconds. "No solution found" collects draws for which no solution was found for a maximum of `l_max` and `k_max` periods ahead.

Wouters (2007) and 4,376 draws per second (22.63 seconds for 100,000 draws in total) for the HANK model. OccBin performed 157.97 draws per second for the medium-scale model, corresponding to a computation time of 10 minutes 55 seconds for 100,000 draws.

The left column of Table 1 presents the results of the benchmarking exercise for the medium scale model using Pydsge. In about 70% of the samples the ZLB is not binding. For these cases the calculation takes the least time because the algorithm only has to ensure that the ZLB is not binding within the first `l_max` periods. Calculation time increases with expected ZLB durations, which was expected given that more guesses are needed. The incremental increase in computation time for higher k^* is, however, rather marginal. The left column of the table shows speed benchmarks for OccBin. The exercise suggests that Pydsge performs between 1000 times ($k = 0$) and more than 2500 times ($k > 20$) faster than OccBin. Note that the percentages of draws in each bin is equal for both methods. This confirms that, conditional on uniqueness, indeed both methods find the same solution. Further, while for Pydsge a draw with an higher k^* does not seem to bear significantly higher computational costs, computation times for OccBin increase notably with k^*. However, the Newton-like method of Occbin seems to require relatively less time to find solutions for draws with $l > 0$, while such draws are relatively more expensive for Pydsge.

Table 2 finally presents benchmarking results for the HANK model. Compared with Table 1, this documents that the increase in the number of variables reflects less than one-to-one in calculation times. Recall that the first phase of the algorithm – finding (l^*, k^*) – only requires dot multiplications. In vectorized code such calculations generally scale disproportionate relative to the size of the vectors due to the relative reduction of computational fixed costs. The actual execution of $\lambda(l^*, k^*)$ and $\nu(l^*, k^*)$,[17] ignoring the

[17] See Appendix A.

additive component, has a maximal complexity of $O(n^3)$ and is likely to be faster. Since not negligible for the HANK model, the times spent on the linear solution and spend on preprocessing are given for completeness.

The HANK model exhibits a more complicated structure than the medium scale model, which reflects in a row of findings. Similar as the medium scale model, HANK features a large number of draws where the ZLB does not bind at all. Differently however, HANK does not feature many draws with durations of intermediary lenghts of say, between one and 15 periods. These only count for 5% of all draws while, in contrast, more than 8% of all draws cause the ZLB to bind for more than 15 periods. Further, while for the medium-scale model, Pydsge finds a solution for *all* draws, in HANK for slightly more than 16% of draws no solution is found. This finding is independent of the maximum durations `l_max` and `k_max`.[18] As stated above, this may have several causes, e.g. that the model runs into indeterminancy issues for these draws. By construction, Pydsge is not well suited to study the reasons behind this finding and the method of Holden (forthcoming) is potentially better equipped to examine the dynamics of these draws in detail.[19]

4 Conclusion

This paper presents a fast and efficient solution method for piecewise-linear macroeconomic models with occasionally binding constraints. The method is implemented in the freely available Pydsge package. I transform the linearized equilibrium conditions into an extended reduced-form system that depends only on the initial states and the expected number of periods at the constraint. This allows for a very efficient computation of the solution via guess-and-verify loops. Speed benchmarks confirm extremely fast computation times even for large-scale models with heterogeneous agents and almost 1000 variables. I document that the method performs more than 1500 times faster than OccBin (Guerrieri and Iacoviello, 2015), its closest competitor. The work in Boehl et al. (forthcoming) and Boehl and Strobel (2022a,b) confirms that the proposed method, when combined with a nonlinear filter for likelihood inference, enables the estimation of large-scale dynamic models while fully accounting for an endogenously binding zero lower bound on nominal interest rates during the estimation procedure.

References

Adjemian, S., Bastani, H., Juillard, M., Karamé, F., Maih, J., Mihoubi, F., Perendia, G., Pfeifer, J., Ratto, M., Villemot, S., 2011. Dynare: Reference Manual Version 4. Dynare Working Papers 1. CEPREMAP.

Atkinson, T., Richter, A.W., Throckmorton, N.A., 2019. The zero lower bound and estimation accuracy. Journal of Monetary Economics .

Bayer, C., Born, B., Luetticke, R., 2020. Shocks, Frictions, and Inequality in US Business Cycles. CEPR Discussion Papers 14364.

[18] E.g., setting `l_max`= 30 and `k_max`= 50 will lead to a rejection rate of the same order.

[19] Note that the HANK model of BBL features rule-based government spending/taxation. These become necessary to stabilize government debt, which determines the supply of liquid assets in the economy. One reason for unconventional ZLB dynamics may be that the interaction between government rules, liquid asset supply and the nominal rate create complicated dynamics.

Binder, M., Pesaran, M.H., 1995. Multivariate rational expectations models and macroeconometric modelling: A review and some new results. Handbook of applied econometrics 1, 139–187.
Boehl, G., Goy, G., Strobel, F., forthcoming. A structural investigation of quantitative easing. Review of Economics and Statistics .
Boehl, G., Lieberknecht, P., 2021. The hockey stick Phillips curve and the zero lower bound. Technical Report. IMFS Working Paper Series.
Boehl, G., Strobel, F., 2020. US business cycle dynamics at the zero lower bound. Bundesbank Discussion Papers 65/2020. Deutsche Bundesbank.
Boehl, G., Strobel, F., 2022a. Estimation of DSGE Models with the Effective Lower Bound. CRC 224 Discussion Papers. University of Bonn and University of Mannheim, Germany.
Boehl, G., Strobel, F., 2022b. The Empirical Performance of Financial Frictions Since 2008. CRC 224 Discussion Papers. University of Bonn and University of Mannheim, Germany.
Carlstrom, C.T., Fuerst, T.S., Paustian, M., 2015. Inflation and output in new keynesian models with a transient interest rate peg. Journal of Monetary Economics 76, 230–243.
Coenen, G., Orphanides, A., Wieland, V., 2004. Price stability and monetary policy effectiveness when nominal interest rates are bounded at zero. Advances in Macroeconomics 4, 1187–1187. doi:10.2202/1534-6013.1187.
Coenen, G., Wieland, V., 2003. The zero-interest-rate bound and the role of the exchange rate for monetary policy in japan. Journal of Monetary Economics 50, 1071–1101.
Coenen, G., Wieland, V.W., 2004. Exchange-rate policy and the zero bound on nominal interest rates. American Economic Review 94, 80–84.
Cuba-Borda, P., Guerrieri, L., Iacoviello, M., Zhong, M., 2019. Likelihood evaluation of models with occasionally binding constraints. Journal of Applied Econometrics 34, 1073–1085.
Guerrieri, L., Iacoviello, M., 2015. Occbin: A toolkit for solving dynamic models with occasionally binding constraints easily. Journal of Monetary Economics 70, 22–38.
Gust, C., Herbst, E., López-Salido, D., Smith, M.E., 2017. The empirical implications of the interest-rate lower bound. American Economic Review 107, 1971–2006.
Holden, T.D., 2016. Computation of solutions to dynamic models with occasionally binding constraints. Technical Report.
Holden, T.D., forthcoming. Existence and uniqueness of solutions to dynamic models with occasionally binding constraints. Technical Report.
Kaplan, G., Moll, B., Violante, G.L., 2018. Monetary policy according to HANK. NBER Working Papers 3. National Bureau of Economic Research, Inc. URL: https://ideas.repec.org/p/nbr/nberwo/21897.html.
Klein, P., 2000. Using the generalized schur form to solve a multivariate linear rational expectations model. Journal of economic dynamics and control 24, 1405–1423.
Reiter, M., 2009. Solving heterogeneous-agent models by projection and perturbation. Journal of Economic Dynamics and Control 33, 649–665.
Rendahl, P., 2017. Linear time iteration. Technical Report. IHS Economics Series.
Schmitt-Grohé, S., Uribe, M., 2004. Solving dynamic general equilibrium models using a second-order approximation to the policy function. Journal of economic dynamics and control 28, 755–775.
Smets, F., Wouters, R., 2007. Shocks and frictions in us business cycles: A bayesian dsge approach. American Economic Review 97, 586–606.
Uhlig, H., et al., 1995. A toolkit for analyzing nonlinear dynamic stochastic models easily .
Villemot, S., et al., 2011. Solving rational expectations models at first order: what Dynare does. Technical Report. Citeseer.

Appendix A Preprocessing and the case of singular P or $\hat{P}$

In this appendix relax the assumption that P and $\hat{P}$ are invertible. Use the QL decomposition on $M = QL$ and on $\hat{M} = \hat{Q}\hat{L}$. Let n_c be the number of control variables. Premultiplication of (10) by Q ($\hat{Q}$, respectively), and premultiplication of the n_c lower

rows by the inverse of the lower-right $n_c \times n_c$ submatrix of L (or $\hat{L}$) leads to:[20]

$$\begin{bmatrix} \hat{P}_{11} & \hat{P}_{12} \\ \hat{P}_{21} & \hat{P}_{22} \end{bmatrix} \begin{bmatrix} s_t \\ c_{t+1} \end{bmatrix} = \begin{bmatrix} \hat{M}_{11} & 0 \\ \hat{M}_{21} & I \end{bmatrix} \begin{bmatrix} s_{t-1} \\ c_t \end{bmatrix} \qquad \forall\ p_0 x_{t+1} + m_0 x_t - \bar{r} \geq 0, \quad \text{(A.1)}$$

$$\begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} s_t \\ c_{t+1} \end{bmatrix} = \begin{bmatrix} M_{11} & 0 \\ M_{21} & I \end{bmatrix} \begin{bmatrix} s_{t-1} \\ c_t \end{bmatrix} + \begin{bmatrix} h_{0,s} \\ h_{0,c} \end{bmatrix} \bar{r} \quad \forall\ p_0 x_{t+1} + m_0 x_t - \bar{r} < 0. \quad \text{(A.2)}$$

Additionally to the function λ introduced in Equation (14), define a function ν by

$$s_t = \nu(l, k, s_{t-1}), \tag{A.3}$$

and note that

$$\Lambda_0(l, k, s_{t-1}) = \begin{bmatrix} \nu(l, k, s_{t-1}) \\ \lambda(l, k, s_{t-1}) \end{bmatrix}. \tag{A.4}$$

Again, both functions are linear in s_{t-1} given (l, k) and take the form

$$\lambda(l, k, s_{t-1}) = \tilde{\lambda}(l, k) s_{t-1} + \bar{\lambda}(l, k)\bar{r}, \tag{A.5}$$

$$\nu(l, k, s_{t-1}) = \tilde{\nu}(l, k) s_{t-1} + \bar{\nu}(l, k)\bar{r}, \tag{A.6}$$

where $\tilde{\lambda}(0, 0)$ and $\tilde{\nu}(0, 0)$ are found using any solution routine for linear systems, and $\bar{\lambda}(0, 0) = \bar{\nu}(0, 0) = \overrightarrow{0}$.

For $k > 0$ we can then express these functions recursively as

$$\nu(l, k, s_{t-1}) = \begin{cases} \left(P_{11} + P_{12}\tilde{\lambda}(l-1, k)\right)^{-1} \left(N_{11} s_{t-1} - P_{12}\bar{\lambda}(l-1, k)\right) & \text{if } l > 0, \\ \left(\hat{P}_{11} + \hat{P}_{12}\tilde{\lambda}(0, k-1)\right)^{-1} \left(\hat{N}_{11} s_{t-1} + h_{0,s} - \hat{P}_{12}\bar{\lambda}(0, k-1)\right) & \text{if } l = 0, \end{cases} \tag{A.7}$$

$$\lambda(l, k, s_{t-1}) = \begin{cases} \left(P_{21} + P_{22}\tilde{\lambda}(l-1, k)\right) \nu(l, k, s_{t-1}) - M_{21} s_{t-1} + P_{22}\bar{\lambda}(l-1, k) & \text{if } l > 0, \\ \left(\hat{P}_{21} + \hat{P}_{22}\tilde{\lambda}(0, k-1)\right) \nu(l, k, s_{t-1}) - \hat{M}_{21} s_{t-1} + \hat{P}_{22}\bar{\lambda}(0, k-1) - h_{0,c} & \text{if } l = 0. \end{cases} \tag{A.8}$$

As these expressions involve an inversion of a matrix of the same dimensionality of the state space, it is efficient to preprocess all functions within a reasonable range `l_max` and `k_max` and store the result for later use. In the same run, $p_0[x_{j+1}|(l, k)] + m_0 x_j|(l, k) = p_0\Lambda_{j+1}(l, k, s_{t-1}) + m_0\Lambda_{j+1}(l, k, s_{t-1})$ can be pre-processed and stored for efficient checking of the conditions in proposition 2. This is a $(1 \times n)$ vector and a scalar for each combination of (l, k, s) under consideration. Checking each condition in proposition 2 then only requires a dot-vector multiplication and a scalar addition.

[20] The fact that the lower-right submatrices of L and $\hat{L}$ are nonsingular follows simply from the fact that controls are defined on their future values.

Appendix B Parameter values for the models

The medium scale model is presented in detail in Boehl and Strobel (2022a). Table B.3 presents the mode of the estimation, which is used for the benchmarking exercise in the main body. The notation in use is exactly the same as in Boehl and Strobel (2022a). For reference, the parameters presented are obtained from estimating the model to US data ranging from 1998 to 2019. This is a much shorter sample as in Boehl and Strobel (2022a), that zooms in on the ZLB episode from 2009 to 2015. Additionally I use a slightly different data treatment.

The HANK model is put forward in Bayer et al. (2020). Table B.4 presents the parameter values used for the benchmarking exercise in the main body. The values are are the mode of the estimation provided from their current version on Github. For further details, see their original paper.

Parameter		Value
σ_c	preferences: intertemporal substitution	0.916
σ_l	preferences: labor	2.627
β_{tpr}	preferences: discount factor	0.109
h	preferences: habit formation	0.842
S''	capital adjustment costs	4.851
ι_p	price inertia	0.149
ι_w	wage inertia	0.385
α	capital share	0.185
ζ_p	Calvo parameter: prices	0.779
ζ_w	Calvo parameter: wages	0.752
Φ_p	production fixed costs	1.459
ψ	capital utilization adjustment costs	0.796
ϕ_π	monetary policy: inflation	1.306
ϕ_y	monetary policy: output gap	0.240
ϕ_{dy}	monetary policy: output gap growth	0.155
ρ	monetary policy: smoothing	0.793
ρ_g	persistence: government spending	0.838
ρ_u	persistence: risk premium	0.855
ρ_z	persistence: technology	0.995
ρ_r	persistence: monetary policy shocks	0.863
ρ_p	persistence: price markups	0.845
ρ_w	persistence: wage markups	0.226
ρ_i	persistence: investment efficiency	0.635
μ_p	moving average: price markups	0.723
μ_w	moving average: wage markups	0.448
ρ_{gz}	correlation gov. spending and technology	0.647
σ_g	standard deviation: government spending	0.171
σ_u	standard deviation: risk premium	0.677
σ_z	standard deviation: technology	0.360
σ_r	standard deviation: monetary policy	0.080
σ_p	standard deviation: price markups	0.101
σ_w	standard deviation: wage markups	0.610
σ_i	standard deviation: investment efficiency	0.487
$\overline{\gamma}$	trend growth rate	0.293
$\overline{l}$	steady state labor	0.849
$\overline{\pi}$	steady state inflation	0.605

Table B.3: Parameters for the medium scale model.

Parameter		Value
δ_s	depreciation rate increase	0.702
ϕ	capital adjustment costs	0.263
κ	price adjustment costs	0.139
κ_w	Wage adjustment costs	0.115
θ_π	monetary policy: inflation	2.590
θ_y	monetary policy: output	0.241
γ_B	reaction of deficit to debt	0.017
γ_π	reaction of deficit to inflation	-0.221
γ_Y	reaction of deficit to output	-0.234
ρ_τ	persistence: tax level	0.961
$\gamma_{B\tau}$	reaction of tax level to debt	0.187
$\gamma_{Y\tau}$	reaction of tax level to output	0.143
ρ_A	persistence: bond-spread shock	0.968
ρ_Z	persistence: technology	0.999
ρ_Ψ	persistence: investment	0.919
ρ_μ	persistence: price markup	0.867
$\rho_{\mu w}$	persistence: wage markup	0.923
ρ_s	persistence: idiosyncratic income risk	0.748
ρ_D	persistence: structural deficit	0.935
ρ_P	persistence: tax progressivity rule	0.969
ρ_R	monetary policy: smoothing	0.716
σ_A	standard deviation: bond-spread shock	0.001
σ_Z	standard deviation: technology	0.006
σ_Ψ	standard deviation: investment	0.030
σ_μ	standard deviation: price markup	0.017
$\sigma_{\mu w}$	standard deviation: wage markup	0.059
σ_s	standard deviation: idiosyncratic income risk	0.734
σ_D	standard deviation: structural deficit	0.004
σ_P	standard deviation: tax progressivity rule	0.0347
σ_R	standard deviation: monetary policy	0.003

Table B.4: Parameters for the HANK model.