Working Paper Series

Efficient Computation with Taste Shocks

WP 19-15

Grey Gordon
Federal Reserve Bank of Richmond

FEDERAL RESERVE BANK OF RICHMOND®
Richmond ▪ Baltimore ▪ Charlotte

Efficient computation with taste shocks*

Grey Gordon

Federal Reserve Bank of Richmond

September 11, 2019
First draft: January, 2018.
Working Paper No. 19-15

Abstract

Taste shocks result in nondegenerate choice probabilities, smooth policy functions, continuous demand correspondences, and reduced computational errors. They also cause significant computational cost when the number of choices is large. However, I show that, in many economic models, a numerically equivalent approximation may be obtained extremely efficiently. If the objective function has increasing differences (a condition closely tied to policy function monotonicity) or is concave in a discrete sense, the proposed algorithms are $O(n \log n)$ for n states and n choices—a drastic improvement over the naive algorithm's $O(n^2)$ cost. If both hold, the cost can be further reduced to $O(n)$. Additionally, with increasing differences in two state variables, I propose an algorithm that in some cases is $O(n^2)$ even without concavity (in contrast to the $O(n^3)$ naive algorithm). I illustrate the usefulness of the proposed approach in an incomplete markets economy and a long-term sovereign debt model, the latter requiring taste shocks for convergence. For grid sizes of 500 points, the algorithms are up to 200 times faster than the naive approach.

Keywords: Computation, Monotonicity, Discrete Choice, Taste Shocks, Sovereign Default, Curse of Dimensionality
JEL Codes: C61, C63, E32, F34, F41, F44

*Contact: greygordon@gmail.com. This paper builds on previous work with Shi Qiu, whom I thank. I also thank Amanda Michaud, Gabriel Mihalache, David Wiczer, and participants at LACEA/LAMES (Guayaquil), EEA/ESEM (Manchester), and Midwest Macro (Madison) for helpful comments.

1 Introduction

Taste shocks have been widely used in economics. They give nondegenerate choice probabilities and likelihood functions (Luce, 1959; McFadden, 1974; Rust, 1987); facilitate indirect inference estimation (Bruins, Duffy, Keane, and Smith, 2015); smooth nonconvexities (Iskhakov, Jørgensen, Rust, and Schjerning, 2017); and make computation possible in consumer (Chatterjee, Corbae, Dempsey, and Ríos-Rull, 2015) and sovereign default models (Dvorkin, Sánchez, Sapriza, and Yurdagul, 2018; Gordon and Querron-Quintana, 2018).[1] But for all their advantages, taste shocks come with a significant computational burden. Specifically, to construct the choice probabilities (the "policy function" one usually uses when using taste shocks), one must generally know the value associated with *every* possible choice for *every* possible state. Consequently, taste shocks imply that computational cost grows quickly in the number of states and choices.

In this paper, I show there is a way to reduce this cost in many economic models. Specifically, the cost can be reduced whenever the taste shocks are not very large and the objective function exhibits *increasing differences* and/or concavity in a discrete sense. (While the term increasing differences is uncommon, it is intimately connected with policy function monotonicity and commonly holds, as I will show.) If taste shocks are not very large, some choices will occur with negligible probability, and increasing differences and/or concavity lets one determine the location of those choices *without* having to evaluate them. The algorithms I propose, which build on Gordon and Qiu (2018a), have a cost that grows linearly or almost linearly for small taste shocks. With n states and n choices, the algorithms are $O(n \log n)$ for sufficiently small taste shocks if one has monotonicity or concavity or $O(n)$ if one has both. Compared to the naive case, which has an $O(n^2)$ cost, the improvement is dramatic: a doubling of the points leads to doubles or nearly doubles the improvement over the naive approach. Consequently, the algorithms I propose can be arbitrarily more efficient than the standard approach. With a two-dimensional state space, the proposed algorithm (for a restricted class of problems) is $O(n^2)$ even when only exploiting monotonicity, another drastic improvement over the naive $O(n^3)$ algorithm.

I demonstrate the numeric efficiency of the algorithm for quantitatively relevant grid sizes in two models. The first is a standard incomplete markets (SIM) model developed by Laitner (1979), Bewley (1983), Huggett (1993), Aiyagari (1994), and others. This model exhibits monotonicity in two-state variables and concavity, allowing all the algorithms to be evaluated within one model. I show that for grids of moderate size (500 asset states and choices), the algorithms can be up to 200 times faster than the naive approach. Moreover, to highlight their usefulness, I show taste shocks are useful in this model for reducing the numerical errors arising from a discrete choice space and for making excess demand continuous. Further, I show that the optimal taste shock size—in the

[1] The idea of using taste shocks for convergence in sovereign debt models was developed independently by Dvorkin et al. (2018) and this paper and then followed subsequently by Gordon and Querron-Quintana (2018), Mihalache (2019), Arellano, Bai, and Mihalache (2019), and others.

Additional applications include use in marriage markets (Santos and Weiss, 2016); dynamic migration models (Childers, 2018; Gordon and Guerron-Quintana, 2018); and quantal response equilibria (McKelvey and Palfrey, 1995).

sense of minimizing average Euler equation errors—is decreasing in grid sizes. This suggests the theoretical cost bounds, which require small taste shocks, are the relevant ones.

In the SIM model, taste shocks are useful but not necessary for computing a solution. The second model I use is a long-term sovereign default model that *requires* taste shocks, or something like them, for convergence. The model further serves as a useful benchmark because the policy function is monotone—which implies increasing differences on the graph of the choice correspondence—but increasing differences does *not* hold globally. Adopting a guess-and-verify approach, I show the algorithm works without flaw and is extremely quick.

There are few algorithms for speeding computation with taste shocks. Partly this is because taste shocks require a discrete number of choices, which precludes the use of derivative-based approaches. One clever approach is that of Chiong, Galichon, and Shum (2016), which shows there is a dual representation for discrete choice models. By exploiting it, they can estimate the model they consider five times faster. In contrast, the approach here can be hundreds of times faster. Chatterjee and Eyigungor (2016) use lotteries to solve models having quasigeometric discounting. While the algorithm they use to implement lotteries is efficient, it is also problem specific and does not apply to the long-term debt model considered here. In contrast, taste shocks provide an alternative and fast way of doing lotteries.

The rest of the paper is organized as follows. Section 2 describes taste shocks, shows the usual structure of choice probabilities by means of an example, and derives the properties that will be exploited by the algorithms. Section 3 gives the algorithms for exploiting increasing differences and/or concavity. Section 4 characterizes the algorithms' efficiency theoretically (characterizing the worst-case behavior) and numerically in the SIM and sovereign default applications. Section 5 extends the algorithms to exploiting increasing differences in two state variables, characterizing the performance theoretically and numerically. Section 6 concludes.

2 Taste shock properties

This section establishes key taste shock properties, which will be exploited later by the algorithms.

2.1 The problem and a numerical example

Fix a state $i \in \{1, \dots, n\}$ and let $U(i, i')$ denote the utility associated with choice $i' \in \{1, \dots, n'\}$. Consider the maximization problem

$$\max_{i' \in \{1,\dots,n'\}} U(i, i') + \sigma \epsilon_{i'}. \tag{1}$$

where $\epsilon_{i'}$ is a random variable for each i'. McFadden (1974) pointed out that if the $\{\epsilon_{i'}\}$ are i.i.d. and all distributed Type-I extreme value,[2] then the choice probabilities have a closed form expression[3]

$$\mathbb{P}(i'|i) = \frac{\exp(U(i,i')/\sigma)}{\sum_{j'=1}^{n'} \exp(U(i,j')/\sigma)} \tag{2}$$

From (2), it appears that computing $\mathbb{P}(i'|i)$ requires evaluating U at each (i,i') combination. However, this is not necessarily the case computationally, and an easy way to see this is with a numerical example. Consider a grid of wealth $\mathcal{B} = \{b_1, \ldots, b_n\}$ where we take $n = 100$ and $b_i = i$ (making $\mathcal{B} = \{1, \ldots, 100\}$) for simplicity. Let $U(i,i') = \log(b_i - b_{i'}/2) + \log(b_{i'})$, corresponding to a two-period consumption-savings problem, where first-period (second-period) wealth is b_i ($b_{i'}$). Figure 1 plots contours of the $\log_{10}$ choice probabilities—i.e., $\log_{10}\mathbb{P}(i'|i)$—for $\sigma = 0.01$ and $\sigma = 0.0001$ for this problem.

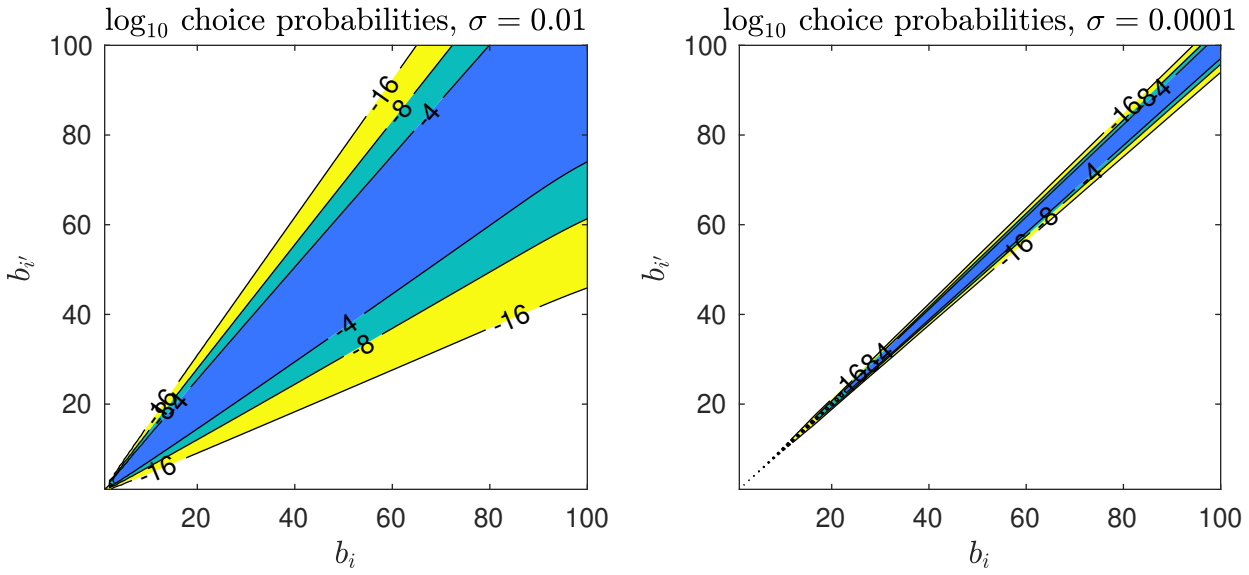

Figure 1: Choice probabilities for different taste shock sizes

The white regions in each graph represent (i,i') combinations where—if one knew where they were—one would not need to evaluate U in order to construct the choice probabilities $\mathbb{P}$. To see this, one must first note that, on a computer, $1+\varepsilon = 1$ if $\varepsilon > 0$ is small enough. The smallest ε for which $1+\varepsilon \neq 1$ is called *machine epsilon*, and, when computation is carried out in *double precision* (the

[2]The Gumbel distribution, or generalized extreme value distribution Type-I, has two parameters, μ and β with a pdf $\beta^{-1}e^{-z+e^{-z}}$ where $z = (x-\mu)/\beta$. This has a mean $\mu + \beta\gamma$ for γ the Euler-Mascheroni constant. Rust (1987) focuses on the special case where $\mu = \gamma$ and $\beta = 1$. The formulas in this paper also assume this special case.

[3]The expected value of the maximum is also given in closed form by the "log-sum" formula, $\mathbb{E}\left[\max_{i' \in \{1,\ldots,n'\}} U(i,i') + \sigma\epsilon_{i'}\right] = \sigma \log\left(\sum_{i'=1}^{n'} \exp(U(i,i')/\sigma)\right)$ (Rust, 1987, p. 1012). These results follow from the maximum of Gumbel-distributed random variables also being Gumbel distributed.

Note that as the number of choices grow, this expectation tends to infinity. To eliminate this behavior, one can make the mean of the Gumbel distribution depend on the number of choices. With a suitably chosen mean, the log-sum formula can be replaced with a log-mean formula. See Gordon and Querron-Quintana (2018) for details.

standard on modern machines), machine epsilon is approximately 10^{-16}.[4] Consequently, the white region characterizes (i, i') combinations where, as far as the computer is concerned, $\mathbb{P}(i'|i)$ is zero in the sense that $1 + \mathbb{P}(i'|i) = 1$. Hence, if one knew where these state/choice combinations were, one could just assign $U(i, i') = \underline{U}$ for a sufficiently negative $\underline{U}$, calculate the choice probabilities $\mathbb{P}(i'|i)$ using (2), and arrive at the same choice probabilities as if one had calculated $U(i, i')$ everywhere.

How can one determine where these low-choice-probability (white) regions are without evaluating U everywhere? There are two ways. First, as evident in the figure, each contour line is monotonically increasing. (As will be shown, this is a consequence of U exhibiting increasing differences.) To see the importance of this, suppose one knew the choice probabilities $\mathbb{P}(\cdot|i = 50)$ and knew that the lowest 10^{-16} contour occurs at $i' = 22$ (like in the left panel of figure 1). Then, for $i > 50$, one knows that $i' < 22$ must have a choice probability less than 10^{-16}. Consequently, one knows the white region, at a minimum, must include all (i, i') having $i > 50$ and $i' < 22$. Consequently, from just this one piece of information (i.e., $\mathbb{P}(\cdot, i = 50)$) one can immediately eliminate 1,050 evaluations, a savings of 10.5% compared to the naive approach.[5] For the smaller taste shock giving rise to the right panel of figure 1, the savings are close to 25%, again just from one piece of information. The algorithms I propose will exploit much more than just this information and produce correspondingly larger gains.

The second way to determine the location of the low-choice-probability region is to note that, for each i, the set connecting two equal contour probabilities is convex. (This is a consequence of concavity, or, more accurately, a version of quasi-concavity where the upper contour sets are convex in a discrete sense.) To see why this is useful, suppose for $i = 50$ one could find the maximum $i' = 50$ of the no taste shock version quickly ($O(\log n')$ evaluations) using Heer and Maußner's (2005) binary concavity algorithm. Then one could move sequentially up (down) from $i' = 50$ until reaching the upper (lower) contour—75 (22) for $\sigma = 0.01$ and 53 (47) for $\sigma = 0.0001$—and then stop; by concavity, any larger (smaller) i' must have even lower probability and therefore be in the white region. Excluding the cost of finding the maximum ($i' = 50$ in this case), which is small, this is extremely efficient because it is evaluating U almost only in the colored region. The algorithms I propose will use this concavity property by itself or jointly (when applicable) with monotonicity.

2.2 Key concepts and formalization of choice probability properties

To formally establish the choice probability properties seen in figure 1, a few definitions are necessary. Define the *log odds ratio* as

$$\mathcal{L}_\sigma(j', i'|i) := \sigma \log\left(\frac{\mathbb{P}(i'|i)}{\mathbb{P}(j'|i)}\right) = U(i, i') - U(i, j'), \tag{3}$$

[4] For single precision, which is the usual case for graphics card computation (such as with CUDA), machine epsilon is close to 10^{-8}.

[5] The number 1,050 is the cardinality of the set $\{(i, i')|i > 50, i' < 22, (i, i') \in \{1, \ldots, 100\}^2\}$, and there are 100^2 combinations of (i, i'), which gives a savings of $1050/100^2 = 10.5\%$.

which uses (2). Denote the optimal value and choices absent taste shocks as

$$U^*(i) = \max_{i' \in \{1,\dots,n'\}} U(i,i') \quad \text{and} \quad G(i) = \arg\max_{i' \in \{1,\dots,n'\}} U(i,i'). \tag{4}$$

Additionally, denote the log odds ratio of i' relative to any choice in $G(i)$ as

$$\mathcal{L}^*_\sigma(i'|i) := \mathcal{L}_\sigma(g(i), i'|i) = \sigma \log\left(\frac{\mathbb{P}(i'|i)}{\mathbb{P}(g(i)|i)}\right) = U(i,i') - U^*(i), \tag{5}$$

where $g(i)$ is any value in $G(i)$. Note $\mathcal{L}^*_\sigma(i'|i) \leq 0$.

As argued above, $\mathbb{P}(i'|i)$, while strictly positive in theory for every choice, is numerically equivalent to zero if it is small enough. To capture this, I will condition the algorithms on an $\varepsilon > 0$ and treat any choice probability $\mathbb{P}(i'|i) < \varepsilon$ as zero. I will say a choice i' is *numerically relevant* at i if $\mathbb{P}(i'|i) \geq \varepsilon$. A necessary condition for i' to be numerically relevant at i is given in the following proposition:

Proposition 1. *For i' to be numerically relevant at i, one must have $\mathcal{L}^*_\sigma(i'|i) \geq \underline{\mathcal{L}}_\sigma := \sigma \log(\varepsilon)$.*

The proofs for this and the other propositions are given in the appendix.

2.2.1 Monotonicity of relative choice probabilities

I will now establish the monotonicity properties of $\mathcal{L}^*$, which hinge on U having increasing differences. By definition, U has (strictly) *increasing differences on S* if and only if $U(i,j') - U(i,i')$ is (strictly) increasing in i for $j' > i'$ whenever $(i,j'),(i,i') \in S$.[6] (Unless explicitly stated otherwise, the set S is $\{1,\dots,n\} \times \{1,\dots,n'\}$.) In the differentiable case, increasing differences requires, essentially, that the cross-derivative $U_{i,i'}$ be nonnegative. However, differentiability is not required, and Gordon and Qiu (2018a,b) collect many sufficient conditions for increasing differences.

The main result the algorithms will exploit is summarized in proposition 2:

Proposition 2. *Fix an (i,i'), let $g(i) \in G(i)$, and suppose $\mathcal{L}^*_\sigma(i'|i) < \underline{\mathcal{L}}_\sigma$. Then, the following hold:*
If $i' < g(i)$, then $\mathcal{L}^*_\sigma(i'|j) < \underline{\mathcal{L}}_\sigma$ for all $j > i$; and
if $i' > g(i)$, then $\mathcal{L}^*_\sigma(i'|j) < \underline{\mathcal{L}}_\sigma$ for all $j < i$.

Hence, if one knows $i' < g(i)$ ($i' > g(i)$) is not numerically relevant at i, then one knows it cannot be numerically relevant at $j > i$ ($j < i$) either. In terms of figure 1, and speaking loosely, it says that the white region below (above) the bottom (top) 10^{-16} contour expands when moving to the right (left).[7]

[6] The order of arguments does not matter, and one could equivalently require $U(j,i') - U(i,i')$ to be increasing in i' for $j > i$. As one might guess, increasing differences is a very general property, which I have simplified to the present context. The concept was introduced by Topkis (1978), and, in general, it is a property of a function of two partially ordered sets (which here are totally ordered). Supermodularity is a closely related property that implies increasing differences (Topkis, 1978, Theorem 3.1).

[7] If this connection is not clear, note the contours describe $\{(i,i')|\mathbb{P}(i'|i) = \varepsilon\}$. If the necessary condition in proposition 1 is "close" to sufficient, then the contours are roughly $\{(i,i')|\mathcal{L}^*_\sigma(i'|i) = \underline{\mathcal{L}}_\sigma\}$.

2.2.2 Concavity of relative choice probabilities

In addition to exhibiting monotonicity properties, $\mathcal{L}_\sigma^*$ can potentially have concavity properties as well, and these hinge on U having concavity properties. Specifically, say a function $f : \{1, \ldots, n\} \times \{1, \ldots, n'\} \to \mathbb{R}$ is *concave* if, for any i and any y, $\{i' \in \{1, \ldots, n'\} | f(i, i') \geq y\}$ is a list of integers with no gaps (e.g., {2,3,4}). Then, we have the following result:

Proposition 3. *If U is concave, then the following hold:*
If $\mathcal{L}_\sigma^*(i'|i) \geq \mathcal{L}_\sigma$ and $\mathcal{L}_\sigma^*(i'+1|i) < \underline{\mathcal{L}}_\sigma$, then $\mathcal{L}_\sigma^*(j'|i) < \underline{\mathcal{L}}_\sigma$ for all $j' > i'$; and
if $\mathcal{L}_\sigma^*(i'|i) \geq \mathcal{L}_\sigma$ and $\mathcal{L}_\sigma^*(i'-1|i) < \underline{\mathcal{L}}_\sigma$, then $\mathcal{L}_\sigma^*(j'|i) < \underline{\mathcal{L}}_\sigma$ for all $j' < i'$.

In other words, if one knows that i' could be numerically relevant but that $i'+1$ ($i'-1$) is not, then all $j' > i'$ ($j' < i'$) must be irrelevant. Because U is concave in the example used in figure 1, the white region above (below) the top (bottom) 10^{-16} contour is convex, as is the colorful region between the two contours.[8] The algorithms will exploit this to avoid evaluating U in the white region.

2.3 The relationship between monotone policies and increasing differences

In the introduction, I claimed increasing differences is closely connected to policy function monotonicity and holds in many economic models. To back up the first claim, I first note that, under mild conditions, increasing differences implies monotonicity.[9] The reason is not difficult to see. Consider $j > i$ and $g(j) \in G(j), g(i) \in G(i)$ (not necessarily with $g(j) \geq g(i)$). Then by optimality

$$U(i, g(j)) - U(i, g(i)) \leq 0 \leq U(j, g(j)) - U(j, g(i)) \tag{6}$$

With increasing differences, this is only possible if $g(j) \geq g(i)$, which gives monotonicity.

Moreover, there is partial converse of this result. Specifically, suppose g is known to be monotone (or in the general case, where g is not unique, that G is *strongly ascending*).[10] Then taking $g(j) \geq g(i)$ and (6) gives that U exhibits increasing differences on $\{(i, i') | i \in \mathcal{I}, i' \in G(i)\}$. That is, monotonicity implies the objective function exhibits increasing differences on the graph of the optimal choice correspondence.

I also claimed increasing differences holds in many economic models. In the weaker sense of having monotone policies (and thereby having increasing differences on a subset of the choice and state space), this is obvious: the real business cycle model (RBC), the SIM model, and the benchmark sovereign default models have monotone policies, along with many others. However, it

[8] The convexity of the colorful region follows as a corollary of proposition 3. Specifically, if $\mathcal{L}_\sigma^*(i'|i) \geq \mathcal{L}_\sigma$ for $i' \in \{a, b\}$, then $\mathcal{L}_\sigma^*(i'|i) \geq \mathcal{L}_\sigma$ for all $i' \in \{a, \ldots, b\}$ (if not, there would be a contradiction of proposition 3).

[9] In particular, the feasible choice correspondence must be ascending (which it is here). In that case, increasing differences implies the optimal choice correspondence is ascending. This ensures a monotone optimal policy exists. Moreover, strictly increasing differences implies the optimal choice correspondence is strongly ascending. This ensures every optimal policy is monotone. See Gordon and Qiu (2018a) for details.

[10] For the general definition, see Topkis (1978); for a simplified one sufficient for the purposes of this paper, see Gordon and Qiu (2018a).

is also true in the stronger sense of U having increasing differences. E.g., in the RBC and SIM models, increasing differences holds globally.[11] While it does not hold globally in the sovereign default model presented here, it will be seen that the monotonicity algorithms deliver the correct result. Hence, even if one cannot prove increasing differences holds globally, a guess-and-verify approach is attractive.

3 Algorithms for exploiting choice probability structures

I now lay out the algorithms: first, the algorithm exploiting monotonicity; second, exploiting concavity; and finally, exploiting both simultaneously. The algorithms' efficiency is explored in Section 4, and exploiting monotonicity in two states is deferred until Section 5.

3.1 Exploiting monotonicity

The algorithm for exploiting the monotonicity of $\mathcal{L}^*$ outlined in proposition 2 is as follows:

Algorithm 1: Binary monotonicity with taste shocks

Parameters: a $\sigma > 0$, an $\varepsilon > 0$ such that $\mathbb{P}(i'|i) < \varepsilon$ will be treated as zero, and $\underline{\mathcal{L}}_\sigma := \sigma \log(\varepsilon)$.
Input: none.
Output: $U(i, i')$ for all (i, i') having $\mathbb{P}(i'|i) \geq \varepsilon$.

1. Let $\underline{i} = 1$ and $\bar{i} := n$
2. Solve for $U(\underline{i}, i')$, and then $\mathcal{L}^*_\sigma(i'|\underline{i})$, at each i'. Find the smallest and largest i' such that $\mathcal{L}^*_\sigma(i'|\underline{i}) \geq \underline{\mathcal{L}}_\sigma$ and save them as $l(\underline{i})$ and $h(\underline{i})$, respectively.
3. Solve for $U(\bar{i}, i')$ for $i' = l(\underline{i}), \ldots, n'$. Then, solve for $\mathcal{L}^*_\sigma(i'|\bar{i})$ for each $i' = l(\underline{i}), \ldots, n'$ assuming that $U(\bar{i}, i') = -\infty$ for $i' = 1, \ldots, l(\underline{i}) - 1$. Find the smallest and largest i' in $\{l(\underline{i}), \ldots, n'\}$ such that $\mathcal{L}^*_\sigma(i'|\bar{i}) \geq \underline{\mathcal{L}}_\sigma$ and save them as $l(\bar{i})$ and $h(\bar{i})$, respectively.
4. Main loop:
   (a) If $\bar{i} < \underline{i} + 1$, STOP. Otherwise, define $m = \lfloor (\underline{i} + \bar{i})/2 \rfloor$.
   (b) Solve for $U(m, i')$ for $i' = l(\underline{i}), \ldots, h(\bar{i})$. Solve for $\mathcal{L}^*_\sigma(i'|m)$ for each $i' = l(\underline{i}), \ldots, h(\bar{i})$ assuming that $U(m, i') = -\infty$ for $i' = 1, \ldots, l(\underline{i}) - 1, h(\bar{i}) + 1, \ldots, n'$. Find the smallest and largest i' in $\{l(\underline{i}), \ldots, h(\bar{i})\}$ such that $\mathcal{L}^*_\sigma(i'|m) \geq \underline{\mathcal{L}}_\sigma$, and save them as $l(m)$ and $h(m)$, respectively.
   (c) Go to step 4(a) twice, once redefining $(\underline{i}, \bar{i}) := (\underline{i}, m)$ and once redefining $(\underline{i}, \bar{i}) := (m, \bar{i})$.

[11] Gordon and Qiu (2018b) explicitly prove this for the RBC model, but the proof for the SIM would be virtually identical.

Note that at step 4 of the algorithm, it is always the case (provided U has increasing differences) that $\mathcal{L}^*_\sigma(i'|m) < \underline{\mathcal{L}}_\sigma$ for all $i' < l(\underline{i})$ or $i' > h(\bar{i})$, which justifies the algorithm's treatment of $U(m, i') = -\infty$ for these choices.

The algorithm as applied to the $U(i, i') = \log(b_i - b_{i'}/2) + \log(b_{i'})$ example is depicted graphically in figure 2. The top panel shows the algorithm when $n = n' = 35$. The blue dots give the computed $l(\cdot)$ and $h(\cdot)$ bounds on numerical relevance. The gray, empty circles show where U ends up being evaluated. For $i = 1$, nothing is known, and so one must—absent an assumption on concavity—evaluate $U(1, \cdot)$ at all i'. This corresponds to step 2. In step 3, the algorithm moves to $i = 35$. There, $U(35, \cdot)$ is evaluated at $\{l(1), \ldots, n\}$, but since $l(1) = 1$ for this example, $U(35, \cdot)$ is evaluated everywhere. In step 4, the algorithm goes to 18, the midpoint of 1 and 35. Because $l(1) = 1$ and $h(35) = 35$, $U(18, \cdot)$ must be evaluated at all i' again. So far, the algorithm has gained nothing.

However, as the divide-and-conquer process continues, the gains grow progressively larger. After $i = 18$, the algorithm goes to $i = 9$ ($\lfloor (1 + 18)/2 \rfloor$) and $i = 26$ ($\lfloor (18 + 35)/2 \rfloor$). At $i = 9$, U must be evaluated at $\{l(1), \ldots, h(18)\} = \{1, \ldots, 21\}$, a 40% improvement over the 35 from the naive approach. Each subsequent step reduces the number of evaluations, and, in the final iterations, the number of evaluations becomes extremely small. For instance, at $i = 4$, $U(4, \cdot)$ is evaluated only at $\{l(3), \ldots, h(5)\} = \{3, 4, 5\}$. This is an order of magnitude fewer evaluations than in the naive approach.

Increasing the number of points to 250, as is done in the bottom panel of figure 2, shows the algorithm wastes very few evaluations. In particular, everywhere in between the blue lines must be evaluated to construct the choice probabilities, since every choice in that region is numerically relevant. The gray area outside the blue lines are inefficient in the sense that knowledge of $U(i, i')$ there is not necessary for constructing the choice probabilities, but evidently these wasted evaluations make up only a small percentage of the overall space.

3.2 Exploiting concavity

Using the concavity property established in proposition 3, algorithm 2 identifies the set of relevant i' values for a given i.

Algorithm 2: Binary concavity with taste shocks

Parameters: a $\sigma > 0$, an $\varepsilon > 0$ such that $\mathbb{P}(i'|i) < \varepsilon$ will be treated as zero, and $\underline{\mathcal{L}}_\sigma := \sigma \log(\varepsilon)$.
Input: an $i \in \{1, \ldots, n\}$ and a, b such that $\mathcal{L}^*_\sigma(i'|i) < \underline{\mathcal{L}}_\sigma$ for all $i' < a$ and $i' > b$.

Note that this implies $G(i) = \arg\max_{i' \in \{a, \ldots, b\}} U(i, i')$, because $i' \in G(i)$ has $\mathcal{L}^*_\sigma(i'|i) = 0 \geq \underline{\mathcal{L}}_\sigma$.
Output: $U(i, i')$ for all i' having $\mathbb{P}(i'|i) \geq \varepsilon$, and $l(i), h(i)$ such that $\mathcal{L}^*_\sigma(i'|i) \geq \underline{\mathcal{L}}_\sigma$ if and only if $i' \in \{l(i), \ldots, h(i)\}$.

1. Solve for any element $g(i) \in \arg\max_{i' \in \{a, \ldots, b\}} U(i, i') = G(i)$ using Heer and Maußner's (2005) binary concavity algorithm as described in Gordon and Qiu (2018a).

   Note: $U^*(i) = U(i, g(i))$, so knowing $U(i, i')$ gives $\mathcal{L}^*(i'|i) = U(i, i') - U^*(i)$.

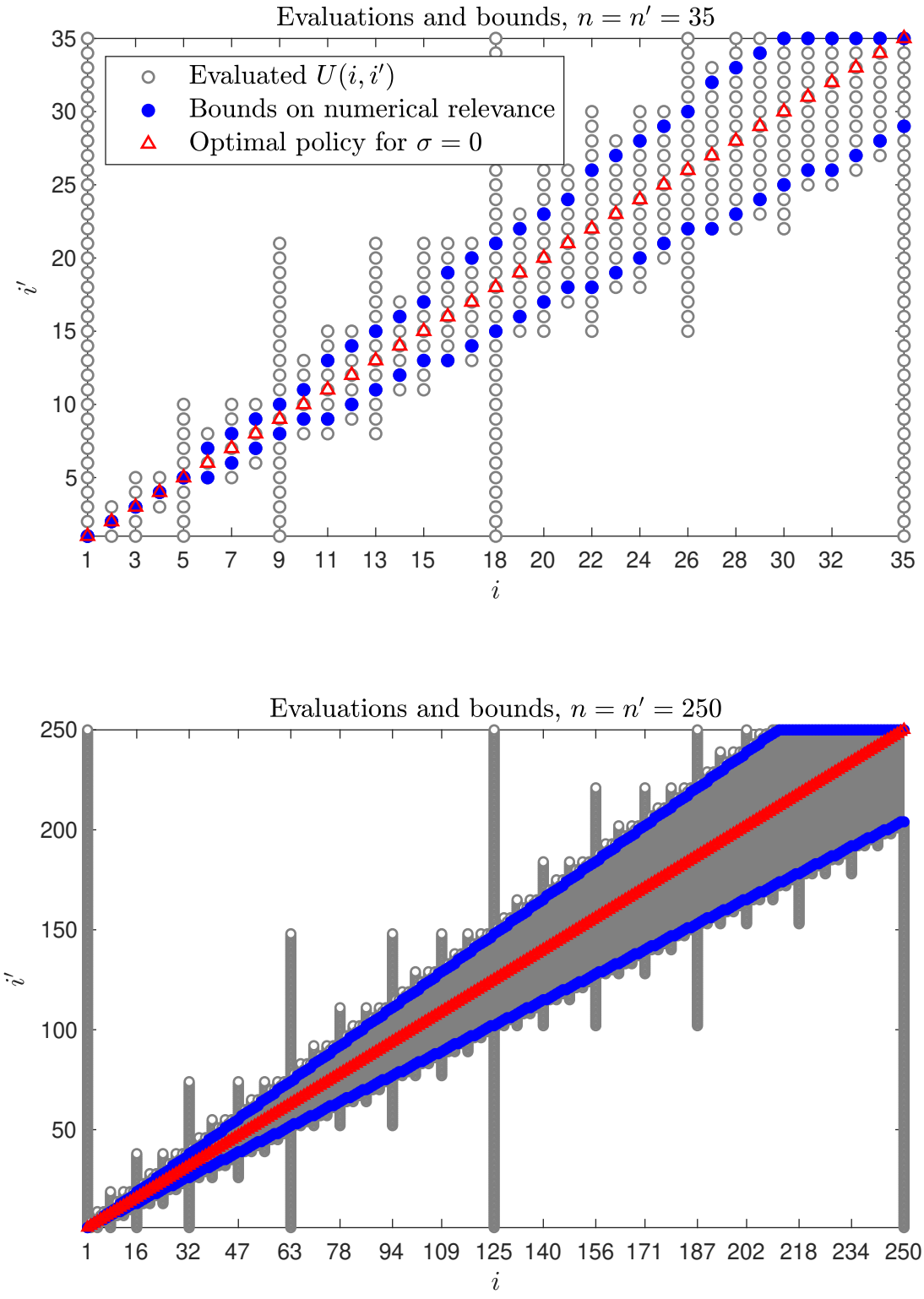

Figure 2: Illustration of algorithm 1 evaluations for two grid sizes

2. Define $i' = g(i)$

   (a) If $i' = a$, STOP. Store i' as $l(i)$.

   (b) Evaluate $U(i, i'-1)$. If $\mathcal{L}^*_\sigma(i'-1|i) < \underline{\mathcal{L}}_\sigma$, store i' as $l(i)$ and STOP.

   (c) Decrement i' by 1 and go to step 2(a).

3. Define $i' = g(i)$

   (a) If $i' = b$, STOP. Store i' as $h(i)$.

   (b) Evaluate $U(i, i'+1)$. If $\mathcal{L}^*_\sigma(i'+1|i) < \underline{\mathcal{L}}_\sigma$, store i' as $h(i)$ and STOP.

   (c) Increment i' by 1 and go to step 3(a).

For an illustration of how algorithm 2 works, consider figure 3. In particular, focus on the middle column of dots in the top panel, which corresponds to algorithm 2 inputs $(i, a, b) = (18, 1, 35)$. Heer and Maußner's (2005) algorithm locates the optimal policy by comparing two adjacent values in the middle of the $\{a, \ldots, b\}$ range, which in this case are 18 and 19. Since $U(18, 18) \geq U(18, 19)$, the algorithm eliminates the range $\{19, \ldots, 35\}$ because there must be a maximum in $\{1, \ldots, 18\}$.[12] The algorithm then compares $U(18, i')$ at $i' = 9$ and $i' = 10$, which are adjacent points in the middle of $\{1, \ldots, 18\}$, and compares them. Since $g(18) = 18$, this binary up-or-down step proceeds up, evaluating $i' = 14, 15$; then $17, 18$; and stops. Having located $g(18) = 18$, step 2 then moves sequentially through $i' = 19, 20, \ldots$ and stops when it reaches the not numerically relevant $i' = 22$. Then, step 3 moves sequentially through $i' = 17, 16, \ldots$ and stops when it reaches $i' = 13$. In this case, only four evaluations are wasted—$U(18, \cdot)$ is evaluated eleven times and it is necessary to evaluate it seven times—, which represents a savings of 69% $(1 - 11/35)$ over the naive algorithm. For $n = n' = 250$, as in the bottom panel, again only four evaluations are wasted, and the savings, consequently, are even larger.

3.3 Exploiting monotonicity and concavity

I now combine algorithms 1 and 2 to simultaneously exploit monotonicity and concavity. As one may have already guessed from examining figure 3, doing so will be extremely efficient.

Algorithm 3: Binary monotonicity and binary concavity with taste shocks

Parameters: a $\sigma > 0$, a $\varepsilon > 0$ such that $\mathbb{P}(i'|i) < \varepsilon$ will be treated as zero, and $\underline{\mathcal{L}}_\sigma := \sigma \log(\varepsilon)$.

Input: none.

Output: $U(i, i')$ for all (i, i') having $\mathbb{P}(i'|i) \geq \varepsilon$.

[12] There is a subtlety here related to feasibility. The canonical problem in Gordon and Qiu (2018a,b) is (1) with $\sigma = 0$, which assumes that every choice is feasible. However, they prove that, under mild conditions, a more general problem with nonfeasible choices can be mapped into it. The mapping consists of replacing $U(i, i')$ at non-feasible (i, i') pairs with a sufficiently negative number (and if a state has no feasible choice at all, replacing it with $\mathbf{1}[i' = 1]$), which produces indifference when both choices are not feasible. They prove the algorithms will deliver correct solutions if the choice set is increasing in i and, if using concavity, has the form $\{1, \ldots, \bar{n}(i)\}$ for some $\bar{n}(\cdot)$ (see Section D and especially proposition 6 in Gordon and Qiu, 2018b). Binary concavity works there, in part, because at points of indifference (which could correspond to two non-feasible choices), the algorithm eliminates the higher range, thereby moving towards feasible choices (if they exist).

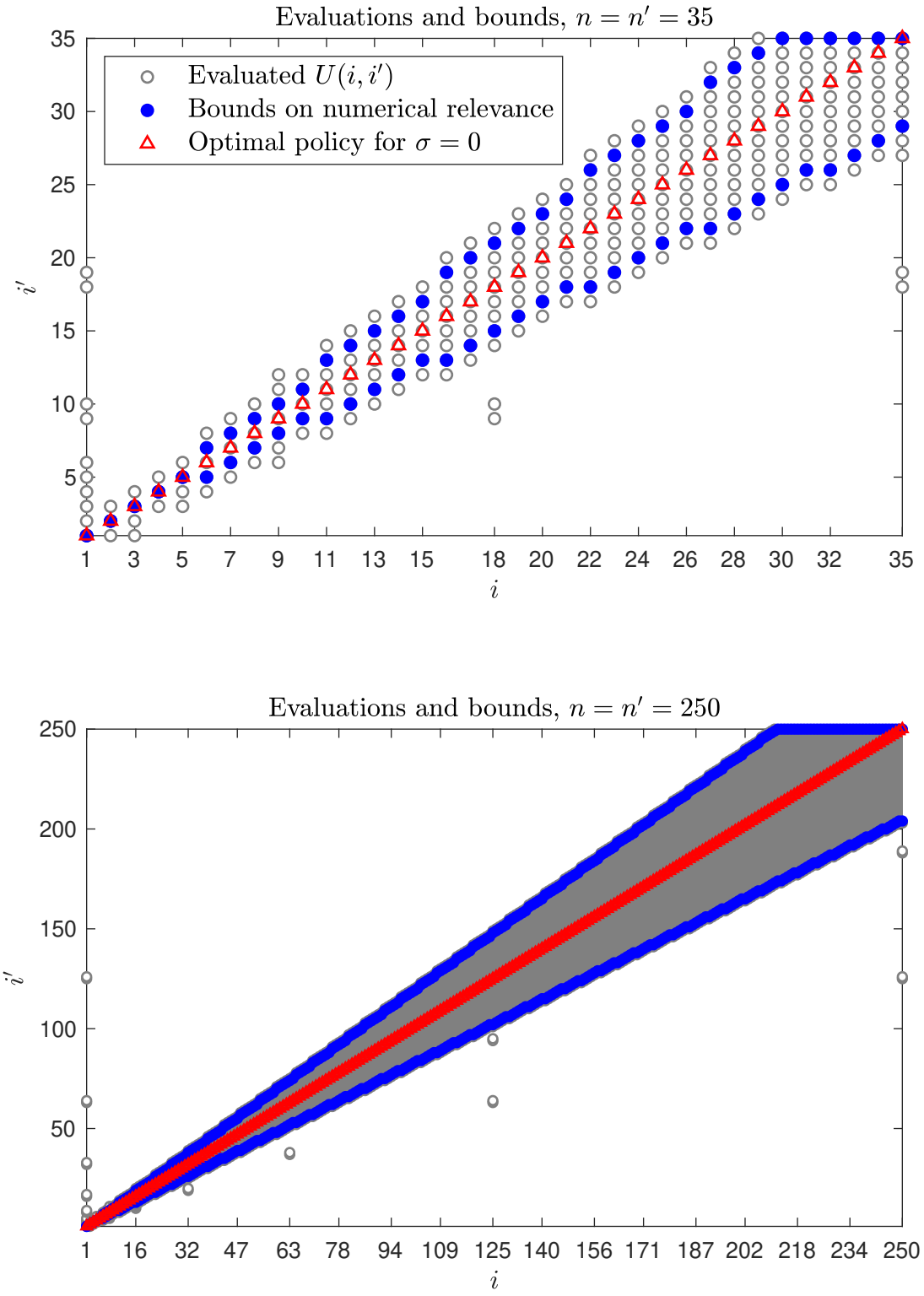

Figure 3: Illustration of algorithm 3 evaluations for two grid sizes

1. Let $\underline{i} = 1$ and $\bar{i} := n$
2. Use algorithm 2 with $(i, a, b) = (\underline{i}, 1, n')$ to solve for $l(\underline{i}), h(\underline{i})$ and $U(\underline{i}, i')$ at numerically relevant i'.
3. Use algorithm 2 with $(i, a, b) = (\bar{i}, l(\underline{i}), n')$ to solve for $l(\bar{i}), h(\bar{i})$ and $U(\bar{i}, i')$ at numerically relevant i'.
4. Main loop:
   (a) If $\bar{i} < \underline{i} + 1$, STOP. Otherwise, define $m = \lfloor (\underline{i} + \bar{i})/2 \rfloor$.
   (b) Use algorithm 2 with $(i, a, b) = (m, l(\underline{i}), h(\bar{i}))$ to solve for $l(m), h(m)$, and $U(m, i')$ at numerically relevant i'.
   (c) Go to step 4(a) twice, once redefining $(\underline{i}, \bar{i}) := (\underline{i}, m)$ and once redefining $(\underline{i}, \bar{i}) := (m, \bar{i})$.

4 Algorithm efficiency

This section first establishes theoretical efficiency bounds, and then examines the empirical performance in two common models.

4.1 Theoretical worst-case bounds

For all the algorithms, the worst-case behavior can be very bad for two reasons. First, if $U(i, \cdot)$ is a constant, then $\mathcal{L}^*_\sigma(i'|i) = 0$ for all i, i'. Consequently, $l(i) = 1$ and $h(i) = n'$ for all i and U must be evaluated everywhere. Second, if σ is large enough, every choice will be chosen with virtually equal probability, which again requires evaluating U everywhere. However, if σ is small and $U(i, \cdot)$ has a unique maximizer for each i, then one can obtain the following theoretical cost bounds as stated in propositions 4, 5, and 6.

Proposition 4. *Consider algorithm 1. Suppose $U(i, \cdot)$ has a unique maximizer for each i. Then for any $\varepsilon > 0$ and any (n, n'), there is a sufficiently small $\sigma(n, n') > 0$ such that U is evaluated at most $n' \log_2(n) + 3n' + 2n$ times and, fixing $n' = n$, the algorithm is $O(n \log_2 n)$ with a hidden constant of 1.*

Proposition 5. *Consider algorithm 2. Suppose $U(i, \cdot)$ has a unique maximizer for each i. Then for any $\varepsilon > 0$ and any (n, n'), there is a sufficiently small $\sigma(n, n') > 0$ such that U is evaluated at most $2n \log_2(n') + 3n$ times if $n' \geq 3$ and, fixing $n' = n$, the algorithm is $O(n \log_2 n)$ with a hidden constant of 2.*

Proposition 6. *Consider algorithm 3. Suppose $U(i, \cdot)$ has a unique maximizer for each i. Then for any $\varepsilon > 0$ and any (n, n'), there is a sufficiently small $\sigma(n, n') > 0$ such that U is evaluated fewer than $8n + 8n' + 2\log_2(n')$ times and, fixing $n' = n$, the algorithm is $O(n)$ with a hidden constant of 16.*

4.2 Numerical example #1: Bewley-Huggett-Aiyagari

The previous subsection established very efficient worst-case bounds for the case of small taste shocks. I now explore the algorithm's efficiency properties numerically for moderately sized taste shocks, quantifying the trade-off between taste shock size and speed.

4.2.1 Model description

The incomplete market model I consider is from Aiyagari (1994) with a zero borrowing limit. The household problem can be written

$$\begin{aligned} V(a,e) &= \max_{a' \in \mathcal{A}} u(c) + \beta \mathbb{E}_{e'|e} V(a',e') \\ \text{s.t. } & c + a' = we + (1+r)a \end{aligned} . \tag{7}$$

where $\mathcal{A}$ is a the asset choice and state space. The equilibrium prices w and r are determined by optimality conditions of a competitive firm and given by $r = F_K(K,N) - \delta$ and $w = F_N(K,N)$. Aggregate labor and capital are given by $N = \int e d\mu$ and $K = \int a d\mu$, respectively, where μ is the invariant distribution of households.

To map this into a discrete problem with taste shocks, take $\mathcal{A} = \{a_1, \ldots, a_n\}$ and assume the support of e is finite also. Then one can write

$$W(i,e) = \mathbb{E}_\epsilon \left[\max_{i' \in \{1,\ldots,n\}} \underbrace{u(we + (1+r)a_i - a_{i'}) + \beta \mathbb{E}_{e'|e} W(i',e')}_{=:U(i,i';e)} + \sigma \epsilon_{i'} \right] .$$

When a choice i' is not feasible for some e, Gordon and Qiu (2018a) prove the algorithms will still work provided a large negative number is assigned to $U(i,i';e)$ there.[13]

4.2.2 The trade-off between speed and taste shock size

Figure 4 shows the algorithm performance for commonly used grid sizes at differing levels of taste shock sizes. The performance is measured in two ways. The primary measure is evaluation counts, which is used in the left panels. This is a programming-free and system-independent way to measure performance. The secondary way is run times, which is used in the right panels. This metric depends on programming and the processing system used.

First consider the top left panel. This gives the evaluation count speedup—i.e., the ratio of n^2 (the naive algorithm's evaluation counts) to the proposed algorithm's evaluation count—for using monotonicity and concavity together (algorithm 3). Proposition 6 shows that this speedup at worst grows linearly when taste shocks are sufficiently small. And as seen in the blue line, where taste shocks are very small, the speedup in fact grows linearly. Consequently, algorithm 3 is around 140

[13] In the case that *all* the i' are not feasible, one should assign a large negative number to each choice but an even more negative number to $i' = 1$.

times faster than the naive algorithm by the time $n = n' = 500$. As the size of taste shocks grow, which increases size of the numerically relevant region, performance necessarily worsens and speedup no longer grows linearly. However, as will be shown later, the optimal taste shock size—in the sense of minimizing Euler equation errors—goes to zero as the number of gridpoints grows. Consequently, if one is using taste shocks for this reason, one can in fact expect very good performance as one progresses upward, from the orange (circled), to green (dotted), to red (dashed), and to blue (solid) lines.

For algorithms 1 and 2, which exploit only monotonicity (middle left panel) and concavity (bottom left panel), respectively, the speedup is almost linear for small taste shocks. This is guaranteed by the theory, as the speedup must grow linearly up to a log factor. Again, larger taste shocks diminish performance.

None of the algorithms attain linear speedup when measuring speedups in run times (in the right panels). The reason, ironically, is that all three of the algorithms are quite efficient. Consequently, the time spent on maximization of the objective function, which is where the algorithms help, becomes small in comparison to the other necessary parts of the solution (such as computing expectations). Nevertheless, one can still expect an order of magnitude gain for the moderate grid sizes considered here.

4.2.3 Optimal taste shock levels

The previous subsection showed the performance of the algorithms is much greater for small taste shocks. I now show that (1) there is an optimal, in the sense of minimizing Euler equation errors, taste shock size; and (2) that this value tends to zero as grid sizes increase. Consequently, the algorithms' performance for smaller taste shocks is the most relevant metric, and, as seen, the performance is very good.

First, consider how taste shocks change the optimal policy as illustrated in figure 5. Absent taste shocks, consumption (in the top left panel) exhibits a saw-tooth pattern that is familiar to anyone who has worked with these models. Using taste shocks, as in the top right panel, smooths out "the" consumption policy—i.e., the expected consumption associated with a given asset and income level (integrating out the taste shock values), $\mathbb{E}_\epsilon c(a, e, \epsilon)$.

The effect on the Euler equation errors can be large. At medium levels of asset holdings, the errors drop by as much as two orders of magnitude. However, at the upper bound and lower bound of the grid, there is not necessarily any improvement. The reason is that the consumption policy is being treated as $\mathbb{E}_\epsilon c(a, e, \epsilon)$, and so the calculation does not account for the points where the borrowing constraint (or saving constraint) are binding for only some ϵ. This can be seen for the high-earnings policy in which the borrowing constraint is not binding and the Euler error is small.

Figure 6 shows a clear trade-off between shock size and the average Euler errors. At small values of taste shocks, the comparatively large errors are driven by the discrete grid. At large taste shock sizes, the errors are instead driven by "mistakes" coming from agents purposely choosing positions that have comparatively low utility $U(i, i')$ but large taste value $\epsilon_{i'}$. The optimal value for this

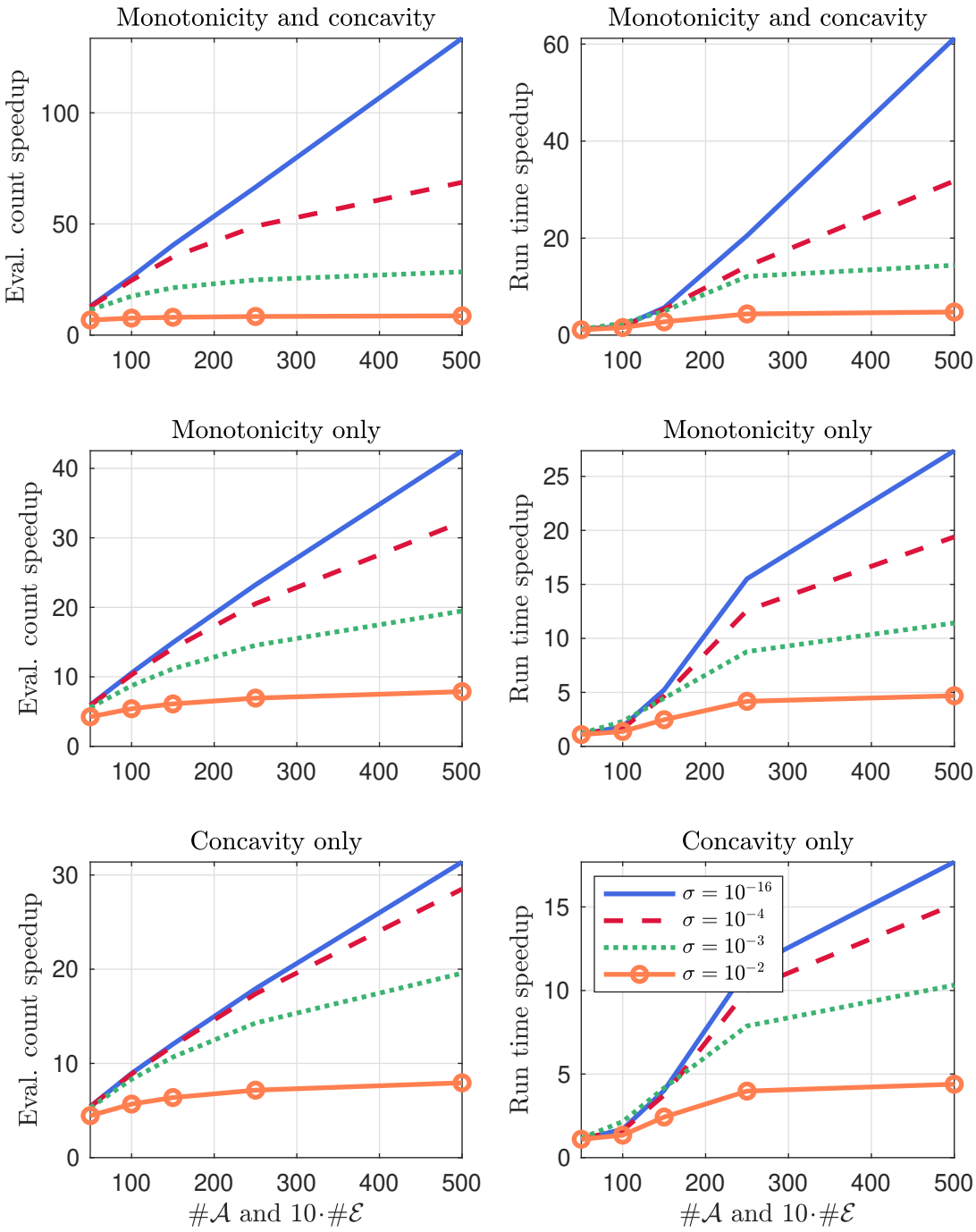

Figure 4: Algorithm speedup by taste shock magnitude and grid size

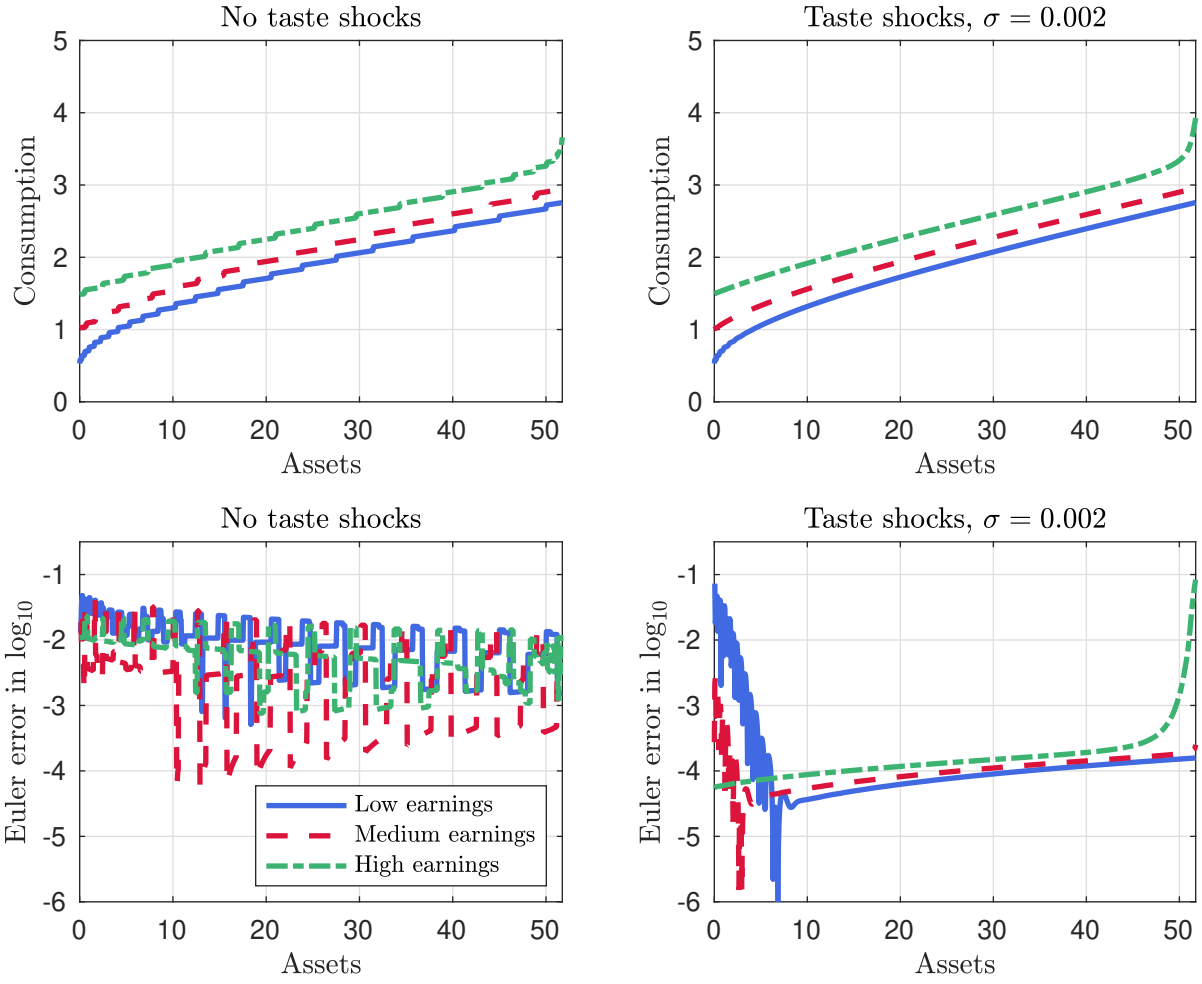

Figure 5: Policy and Euler error comparison

calibration and grid size is around $10^{-2.6} \approx 0.002$, which is the value used in figure 5.

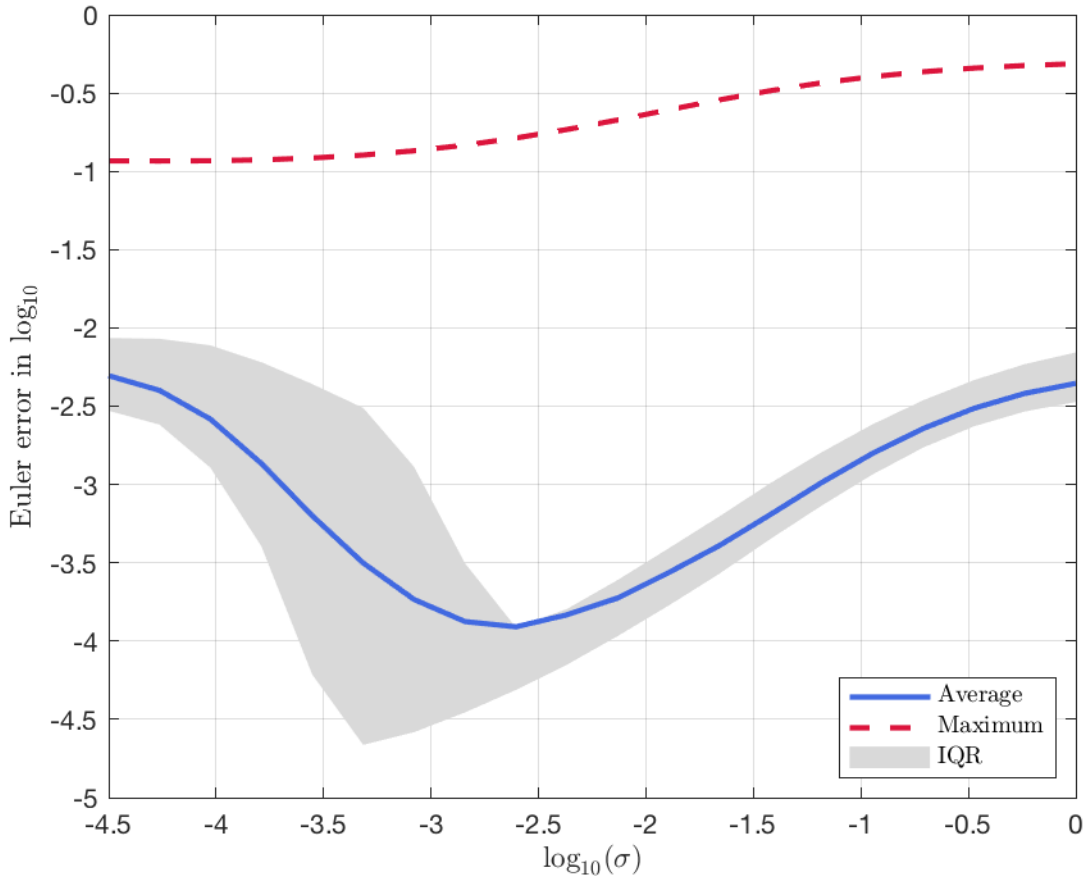

Figure 6: Taste shocks' impact on Euler errors

Clearly, as grid sizes increase, the first type of error falls. The second type of error, however, does not tend to change. The import of this is that the optimal taste shock size—in the sense of minimizing average Euler equation errors—is diminishing in grid size. This is revealed in figure 7, which plots average Euler equation errors for differing grid and taste shock sizes. For grids of 100, $\sigma = 0.1$ is optimal. This decreases to $\sigma = 0.01$ for grid sizes of 250. For grids of 1,000, the optimal size falls by another order of magnitude to $\sigma = 0.001$.

For the grid sizes here, the difference between an optimal taste shock size and an arbitrary size is one to two orders of magnitude. Additionally, a 250-point grid with an optimally chosen σ is far better (with average errors around $4 \approx 10^{-.6}$ times smaller) than a 1,000-point grid with no taste shocks. Consequently, taste shocks are a cost-effective way—when using the proposed algorithms—of reducing computational error. Moreover, the fact that the optimal σ is decreasing in grid size implies that the algorithms' speedups grow linearly or almost linearly when using optimal taste shock magnitudes.

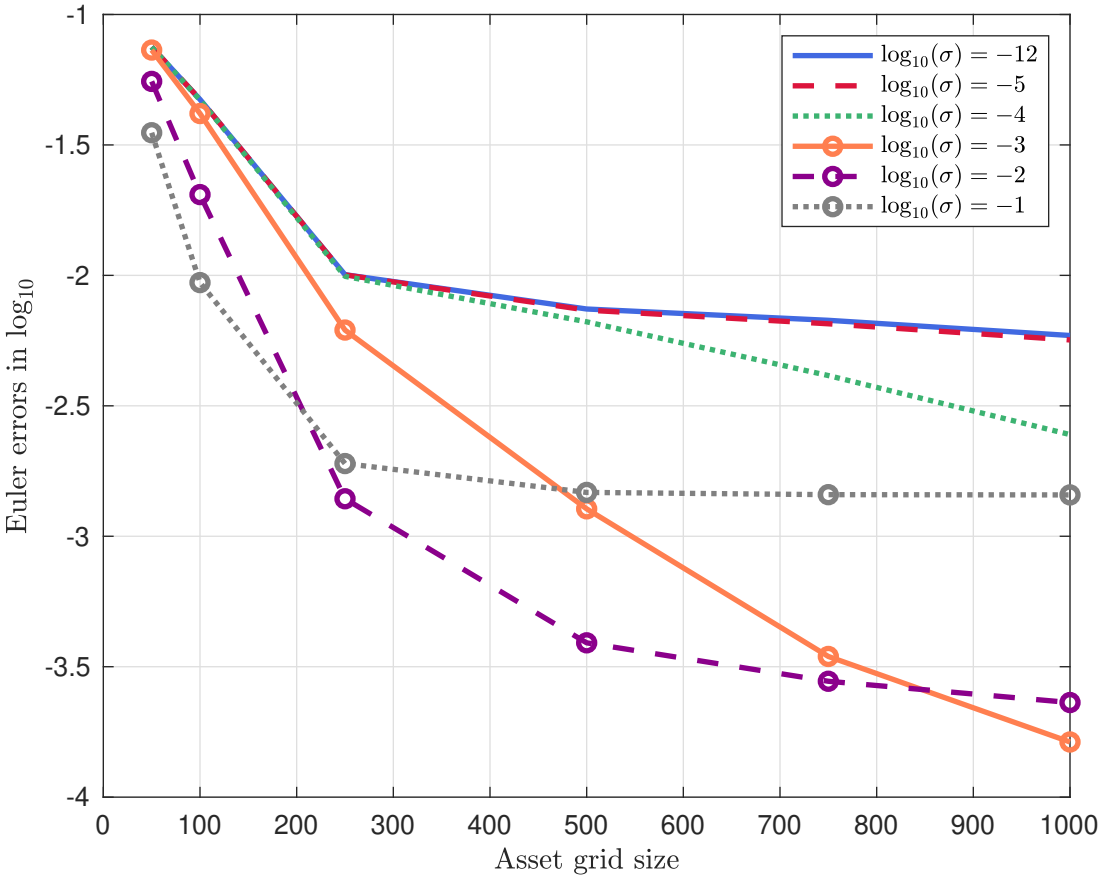

Figure 7: Euler errors by taste shock magnitude and grid size

4.2.4 Two more advantages of taste shocks: market clearing and calibration

We also briefly point out two more advantages of having preference shocks. First, provided the utility function U moves continuously in the underlying object, the policy functions (averaged across choice probabilities) do as well. This has many benefits, one of which can be seen in figure 8, a reproduction of the classic market clearing figure in Aiyagari (1994). In the top panels, it seems that as the interest rate r increases, the implied aggregate asset holdings labeled $Ea(r)$ increases continuously. However, upon further inspection in the bottom panels, aggregate assets actually move discontinuously for $\sigma = 0$. For some parameterizations, this is more of an issue than for others. Here, $K(r)$, firms' demand for capital, falls close to $Ea(r)$ for $r \approx 0.014$. However, the model rules out the possibility that capital would be between 7.7 and 8, which could be an issue in calibration. In contrast, for $\sigma > 0.01$, demand moves continuously.

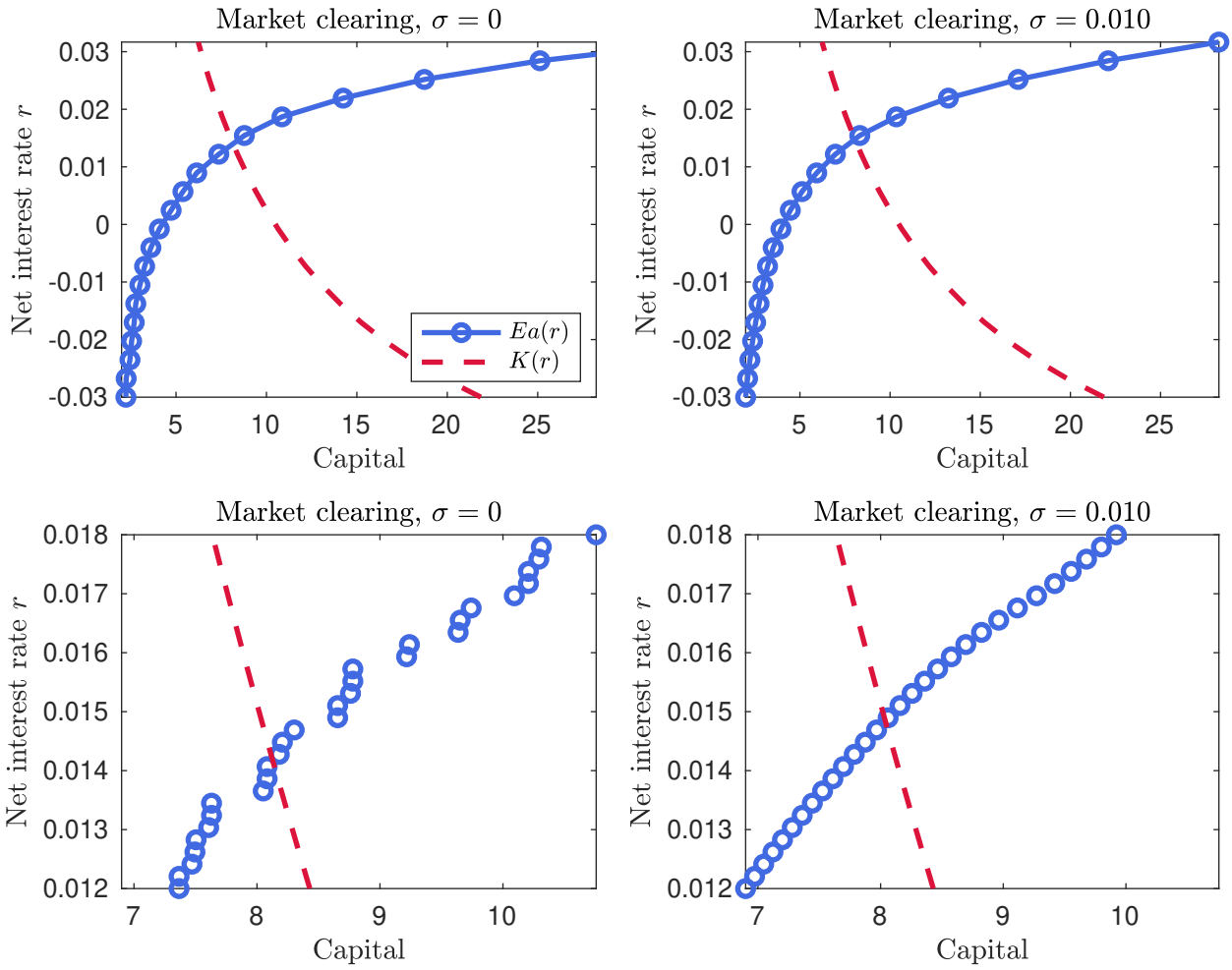

Figure 8: Market clearing in Aiyagari (1994)

A second advantage is that taste shocks aid in calibration and estimation, a point made, e.g., in Bruins et al. (2015). Consider, for simplicity, trying to choose the depreciation rate δ, to match some target interest rate. Increasing δ lowers firm capital demand, shifting $K(r)$ down in a parallel

fashion.[14] In the bottom left panel, one can see this shift down in $K(r)$ (the red dashed line) would at first lower the equilibrium interest rate r^* and equilibrium capital K^* in a fairly smooth fashion. But, after reaching the gap in the supply of capital that occurs between $K \in [7.7, 8]$, an equilibrium does not even approximately exist.[15] Consequently, whatever "equilibrium" moment the code returned for r^* or K^* would be arbitrary and not necessarily vary smoothly in δ. In contrast, taste shocks typically make the equilibrium moments vary smoothly in δ and other parameters, facilitating moment-based estimation techniques.

4.3 Numerical example #2: Sovereign default

While taste shocks are very useful in the SIM model above, for decades the profession has made due without them because they are, strictly speaking, not necessary. A class of models where taste shocks, or something like them, are necessary is sovereign default models with long-term bonds (which were first considered by Hatchondo and Martinez, 2009, and Chatterjee and Eyigungor, 2012). I will now show that, in that class of models, convergence fails absent taste shocks (but can hold with them). I will then characterize the algorithm speedup, which turns out to be very similar to the speedup in the SIM model.

4.3.1 Model description

The model has sovereign with a total stock of debt $-b$ and output y. If the sovereign decides to default on its debt, the country moves to autarky with output reduced by $\phi(y)$. The country returns from autarky with probability ξ, in which case all its outstanding debt has gone away. When not defaulting and not in autarky, the sovereign has to pay back $(\lambda + (1-\lambda)z)(-b)$, reflecting that λ fraction of the debt matures and, for the fraction not maturing, a coupon z must be paid. Additionally, the sovereign issues $-b' + (1-\lambda)b$ units of debt at price $q(b', y)$—the price depends only on the next period total stock of debt and current output because they are sufficient statistics for determining repayment rates.

The sovereign's problem may be written

$$V(b, y) = \max_{d \in \{0,1\}} (1-d)V^r(b, y) + dV^d(y)$$

where the value of repaying is

$$V^r(b, y) = \max_{b' \in \mathcal{B}} u(c) + \beta \mathbb{E}_{y'|y} V(b', y')$$
$$\text{s.t. } c = y - q(b', y)(b' - (1-\lambda)b) + (\lambda + (1-\lambda)z)b$$

[14] In particular, $K(r)$ is given implicitly by $r = F_K(K(r), N) - \delta$.

[15] Of course, an equilibrium *does* exist, but constructing it involves using demand correspondences and exploiting points of indifference, not the simple policy functions here.

and the value of default is

$$V^d(y) = u(y - \phi(y)) + \mathbb{E}_{y'|y}\left[(1-\xi)V^d(y') + \xi V(0, y')\right]$$

Let the optimal bond policy for the V^r problem be denoted $a(b, y)$.

The price schedule q is a solution to $q = T \circ q$ where

$$(T \circ q)(b', y) = (1 + r^*)^{-1}\mathbb{E}_{y'|y}(1 - d(b', y'))\left(\lambda + (1-\lambda)(z + q(a(b', y'), y')\right), \tag{8}$$

with r^* an exogenous risk-free rate. The price schedule reveals a fundamental problem with using a discrete grid $\mathcal{B}$ in this model: very small changes to q can cause discrete changes in a, which then cause discrete changes in $T \circ q$ that inhibit or prevent convergence. The literature has found ways to get around this issue. E.g., Hatchondo and Martinez (2009) use continuous choice spaces with splines and seem to not have convergence troubles, and Chatterjee and Eyigungor (2012) incorporate a continuously distributed i.i.d. shock to facilitate convergence. Taste shocks are, however, a simpler option that accomplishes the same task.

Taste shocks can be added to either the default decision, the debt issuance decision, or both. For expositional simplicity, I add them only to the debt issuance decision, which will be sufficient to generate convergence.[16] Taking $\mathcal{B} = \{b_1, \ldots, b_n\}$, I define

$$V^r(i, y) = \mathbb{E}_\epsilon\left[\max_{i' \in \{1,\ldots,n\}} u(c) + \beta\mathbb{E}_{y'|y}V(i', y') + \sigma\epsilon_{i'}\right]$$
$$\text{s.t. } c = y - q(b_{i'}, y)(b_{i'} - (1-\lambda)b_i) + (\lambda + (1-\lambda)z)b_i$$

(while also changing V and V^d to use indices instead of bond values). The price schedule update then becomes

$$(T \circ q)(i', y) = (1 + r^*)^{-1}\mathbb{E}_{y'|y}\left[(1 - d(i', y'))\left(\lambda + (1-\lambda)(z + \sum_{i''}\mathbb{P}(i''|i', y')q(i'', y')\right)\right]. \tag{9}$$

The advantage of using taste shocks is that $\mathbb{P}(i''|i', y')$ moves continuously in the q guess. Consequently, $T \circ q$ changes little when q changes little, except when the default decision d changes. In contrast, in (8), small changes in q can result in discontinuous jumps in a, which then create large jumps in $T \circ q$ as Chatterjee and Eyigungor (2012) discuss.[17]

[16] A previous version of the paper, available by request, combined the repayment and default decision into one choice, which can be considerably faster.

Another issue here is that, as the number of choices grow, V^r will tend to infinity because the support of the taste shocks is infinite. As discussed in Gordon and Querron-Quintana (2018), a good way to correct this is to make the mean of the taste shocks dependent on the grid size. This converts the "log-sum formula" from Rust (1987) into a "log-mean" formula, which eliminates this systematic growth in V^r.

[17] Why is it important to smooth jumps in a more than jumps in d? Consider the case of V^r and V^d being close to their fixed points with V^r not almost identical to V^d. Then, small changes in q will *not* change the default decision but may change the policy a in a substantial way, thereby affecting $T \circ q$.

4.3.2 Convergence only with preference shocks

Figure 9 shows the lack of convergence for $\sigma = 0$, even after 2,500 iterations with a relaxation parameter above 0.95 and with 300 income states.[18] In contrast with $\sigma = 10^{-4}$, convergence happens without any relaxation. For the intermediate case of $\sigma = 10^{-6}$, convergence does not occur without relaxation, but as soon as the relaxation parameter begins increasing (which happens at the 500th iteration, as seen in the bottom panel), the sup norms for V and q trend downward.

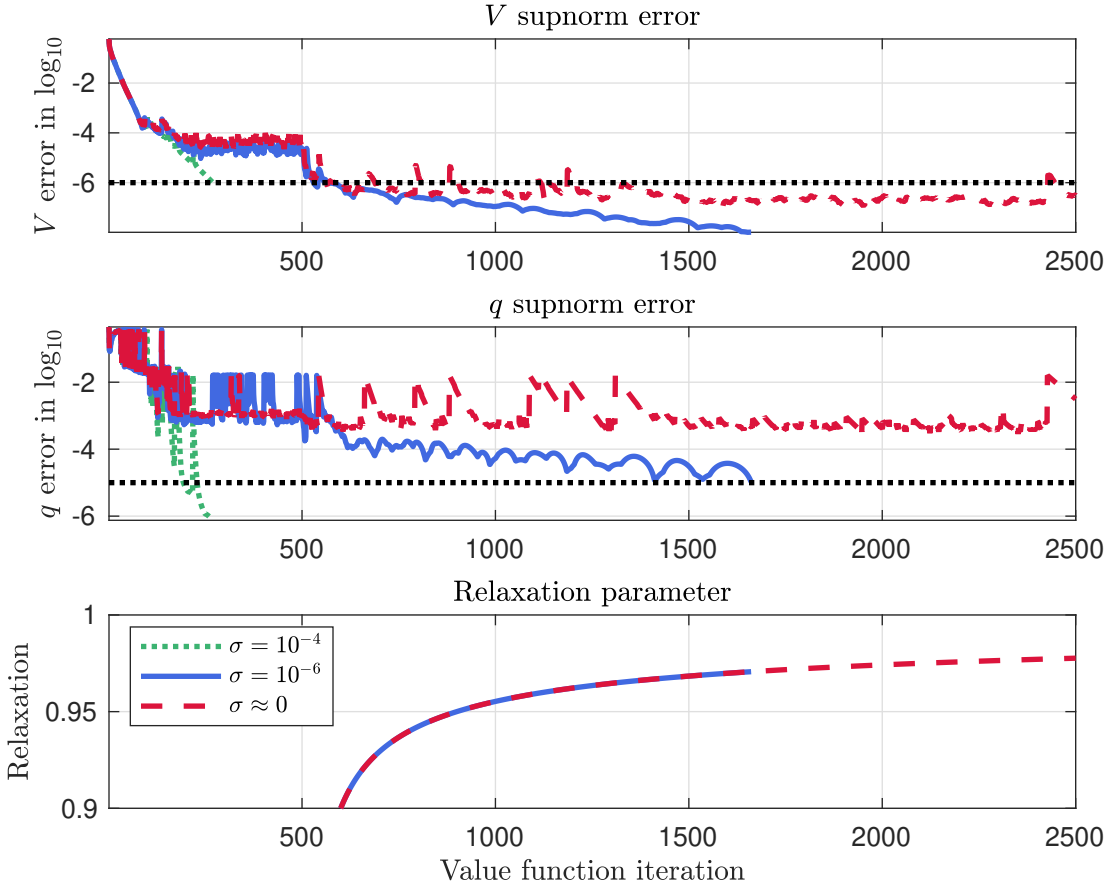

Figure 9: Convergence and lack of it with long-term debt

4.3.3 Monotone policies, but not necessarily increasing differences

Using results from Gordon and Qiu (2018b), increasing differences of the objective function will hold globally if $c(b, b') := y - q(b', y)(b' - (1-\lambda)b) + (\lambda + (1-\lambda)z)b$ is increasing in b, decreasing in b', and if c has increasing differences. All of these hold except for possibly c decreasing in b'. With short-term debt $\lambda = 1$, c may not be decreasing in b' because q is non-monotone; with long-term debt $\lambda < 1$, the situation is even worse because of debt dilution (see Hatchondo, Martinez, and

[18] The relaxation parameter is the constant θ used when updating q in the fixed point iteration algorithm. Specifically, at iteration t, one has a guess q_t and computes an update $T \circ q_t$. The guess at iteration $t+1$ is then $q_{t+1} = \theta q_t + (1-\theta)(T \circ q_t)$. See Judd (1998) for more details.

Sosa-Padilla, 2014, 2016 for details). However, one can also show, using results from Gordon and Qiu (2018a), that the policy function absent taste shocks is nevertheless monotone, which implies increasing differences holds on the graph of the optimal choice correspondence.

Can the monotonicity algorithms be used without a proof of increasing differences? Yes. Analogously to figure 2, figure 10 plots choice probability contours for an intermediate level of output computed without any monotonicity assumptions. Both the double precision cutoff of $\mathbb{P}(i'|i) \geq 10^{-16}$ and the single precision cutoff of $\mathbb{P}(i'|i) \geq 10^{-8}$ are monotone. This means the monotonicity algorithm applied to either of these cutoff levels will work correctly. However, this behavior is not guaranteed, so one should use caution.

A good way to be cautious is to use a guess-and-verify approach, computing the optimal policy assuming the algorithms work and lastly checking, using the naive algorithm, whether the computed V and q functions constitute an equilibrium. Note that as the final verification step is only one iteration out of several hundred or thousands, its overall cost is small.

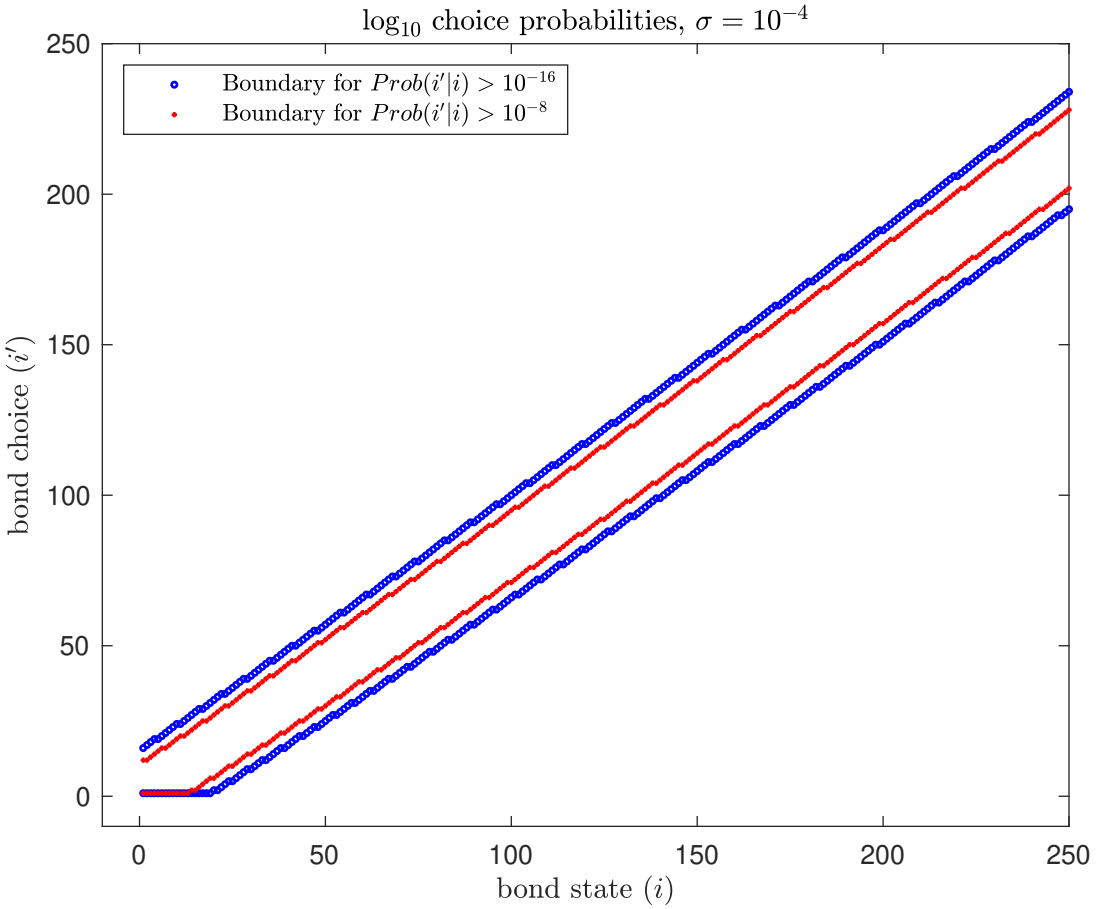

Figure 10: Choice probabilities

4.3.4 Empirical speedup

As can be seen in figure 11, the algorithm's speedups are essentially identical to those in the incomplete markets model. In particular, the speedups grow almost linearly as measured by evaluation counts provided σ is close to zero. However, convergence cannot be obtained for very small σ. The best performance—while still obtaining convergence—occurs for σ between 10^{-5} and 10^{-6}. There, the algorithm is twelve to twenty-four times more efficient than the naive algorithm, a large improvement.

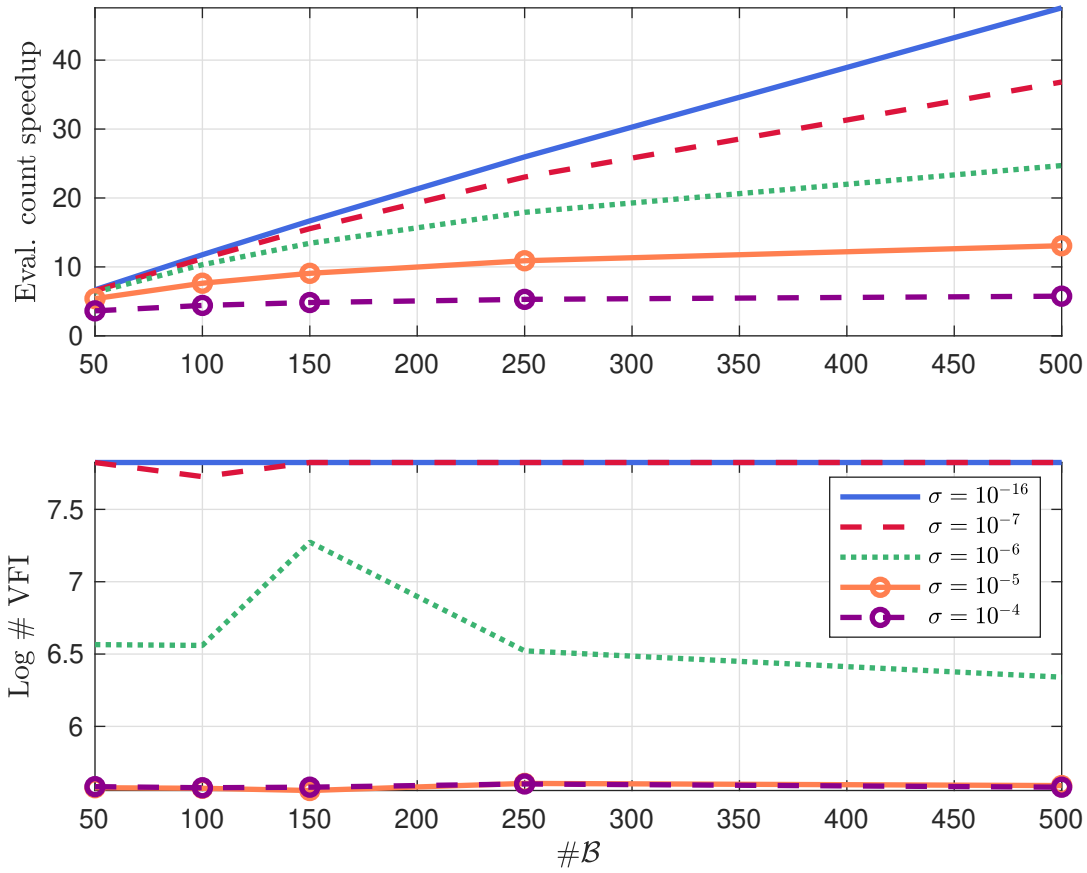

Figure 11: Default model speedups and iterations to convergence

5 Exploiting monotonicity in two states

So far, I have focused on the one-dimensional case. But in many cases there can be monotonicity in more than one state variable, such as in the incomplete markets model where the savings policy is monotone bonds and earnings. I now show how the ideas from the one-state-variable algorithms can be extended to having two state variables.

5.1 The algorithm

Consider a problem with a two-dimensional state space $(i_1, i_2) \in \{1, \ldots, n_1\} \times \{1, \ldots, n_2\}$. Defining everything analogously to section 2, e.g.,

$$\max_{i' \in \{1,\ldots,n'\}} U(i_1, i_2, i') + \sigma \epsilon_{i'} \tag{10}$$

and

$$\mathbb{P}(i'|i_1, i_2) = \frac{\exp(U(i_1, i_2, i')/\sigma)}{\sum_{j'=1}^{N} \exp(U(i_1, i_2, j')/\sigma)}, \tag{11}$$

all the preceding results for the one-dimensional case go through holding the other state fixed. In particular, propositions 1 and 3 are unchanged (except replacing i with i_1, i_2), while the monotonicity result in proposition 2 becomes

$$\mathcal{L}_\sigma(j', i'|i_1, i_2) \text{ is } \begin{cases} \text{decreasing in } i_1 \text{ and } i_2 \text{ if } j' > i' \\ \text{increasing in } i_1 \text{ and } i_2 \text{ if } j' < i' \end{cases}. \tag{12}$$

The below algorithm exploits monotonicity (and, if applicable, concavity) in an efficient way.

Algorithm 4: Binary monotonicity in two states with taste shocks

Parameters: a $\sigma > 0$, an $\varepsilon > 0$ such that $\mathbb{P}(i'|i_1, i_2) < \varepsilon$ will be treated as zero, and $\underline{\mathcal{L}}_\sigma := \sigma \log(\varepsilon)$.
Input: none.
Output: $U(i_1, i_2, i')$ for all (i_1, i_2, i') having $\mathbb{P}(i'|i_1, i_2) \geq \varepsilon$.

1. Solve for $U(\cdot, 1, i')$ using algorithm 1 or 3. Save the l function from it as $l(\cdot, 1)$.
2. Solve for $U(\cdot, n_2, i')$ using algorithm 1 or 3 while only checking $i' \geq l(\cdot, 1)$. Save the h function from it as $h(\cdot, n_2)$. Set $(\underline{i}_2, \bar{i}_2) := (1, n_2)$.
3. Main loop:
   (a) If $\bar{i}_2 < \underline{i}_2 + 1$, STOP. Otherwise, define $m_2 = \lfloor (\underline{i}_2 + \bar{i}_2)/2 \rfloor$.
   (b) Solve for $U(\cdot, m_2, i')$ using algorithm 1 or 3 but restricting the search space additionally to $l(\cdot, \underline{i}_2)$ and $h(\cdot, \bar{i}_2)$. Save the l and h coming from the one-state algorithm as $l(\cdot, m_2)$ and $h(\cdot, m_2)$, respectively.
   (c) Go to step 3(a) twice, once redefining $(\underline{i}_2, \bar{i}_2) := (\underline{i}_2, m_2)$ and once redefining $(\underline{i}_2, \bar{i}_2) := (m_2, \bar{i}_2)$.

5.2 Theoretical cost bounds

The theoretical cost bounds in the two-state case are more difficult to establish. The following proposition shows, for a subset of problems, that the algorithm performs very efficiently.

Proposition 7. *Consider algorithm 4 for exploiting monotonicity only (i.e., using algorithm 1 instead of 3). Suppose that, for each i_1, i_2, $U(i_1, i_2, \cdot)$ has a unique maximizer $g(i_1, i_2)$—monotone*

in both arguments—and that $n_1, n_2, n' \geq 4$. *Let* $\lambda \in (0, 1]$ *be such that for every* $j \in \{2, \ldots, n_2 - 1\}$, *one has* $g(n_1, j) - g(1, j) + 1 \leq \lambda(g(n_1, j + 1) - g(1, j - 1) + 1)$.

Then for any $\varepsilon > 0$ *and any* (n_1, n_2, n'), *there is a* $\sigma(n_1, n_2, n') > 0$ *such that* U *is evaluated fewer than* $(1+\lambda^{-1}) \log_2(n_1) n' n_2^{\kappa} + 3(1+\lambda^{-1}) n' n_2^{\kappa} + 4 n_1 n_2 + 2 n' \log_2(n_1) + 6 n'$ *times where* $\kappa = \log_2(1+\lambda)$.

Moreover, for $n_1 = n' =: n$ *and* $n_1/n_2 = \rho$ *with* ρ *a constant, the cost is* $O(n_1 n_2)$ *with a hidden constant of 4 if* $(g(n_1, j) - g(1, j) + 1)/(g(n_1, j + 1) - g(1, j - 1) + 1)$ *is bounded away from 1 for large* n.

To see the large potential benefit of the algorithm, note that in the special case of $n_1 = n_2 = n' =: n$, the $O(n_1 n_2)$ behavior is $O(n^2)$, which grows increasingly efficient compared to the naive algorithm's n^3 cost. I now turn to how the algorithm fairs in the incomplete markets model.

5.3 Empirical speedup

Figure 12 reveals the evaluation counts per state $\sigma \approx 0$ (in the top panel) and $\sigma = 10^{-3}$ (in the bottom panel). As suggested (though not guaranteed for all U) by proposition 7, for $\sigma \approx 0$ the two-state monotonicity algorithm's evaluations counts fall and seem to level off at slightly more than 3.0 by the time $\#\mathcal{A} = 500$ and $\#\mathcal{E} = 50$ are reached. This is around a 167-fold improvement on brute force and is obtained without an assumption on concavity. When also exploiting concavity, the evaluation counts fall to around 2.4, a 208-fold improvement on brute force. As it is always necessary to have at least an evaluation count of 1 (which is required simply to evaluate U at a known optimal policy), the algorithm is extremely efficient in this case.

With larger σ, the evaluation counts continue to rise in grid sizes, resulting in less dramatic speedups. For instance, at the largest grid size, the two-state algorithms and monotonicity with concavity all have evaluation counts of around seventeen. While this is not the dramatic 200 times better than brute force, it is still twenty-nine times better.

6 Conclusion

Taste shocks have a large number of advantages, but they also imply a large computational cost when using naive algorithms. By using the proposed algorithms, this cost can be drastically reduced if the objective function exhibits increasing differences or concavity. This behavior was proven theoretically for very small taste shocks and demonstrated empirically for larger taste shocks. Consequently, the algorithms proposed in this paper should enable the computation of many new and challenging models.

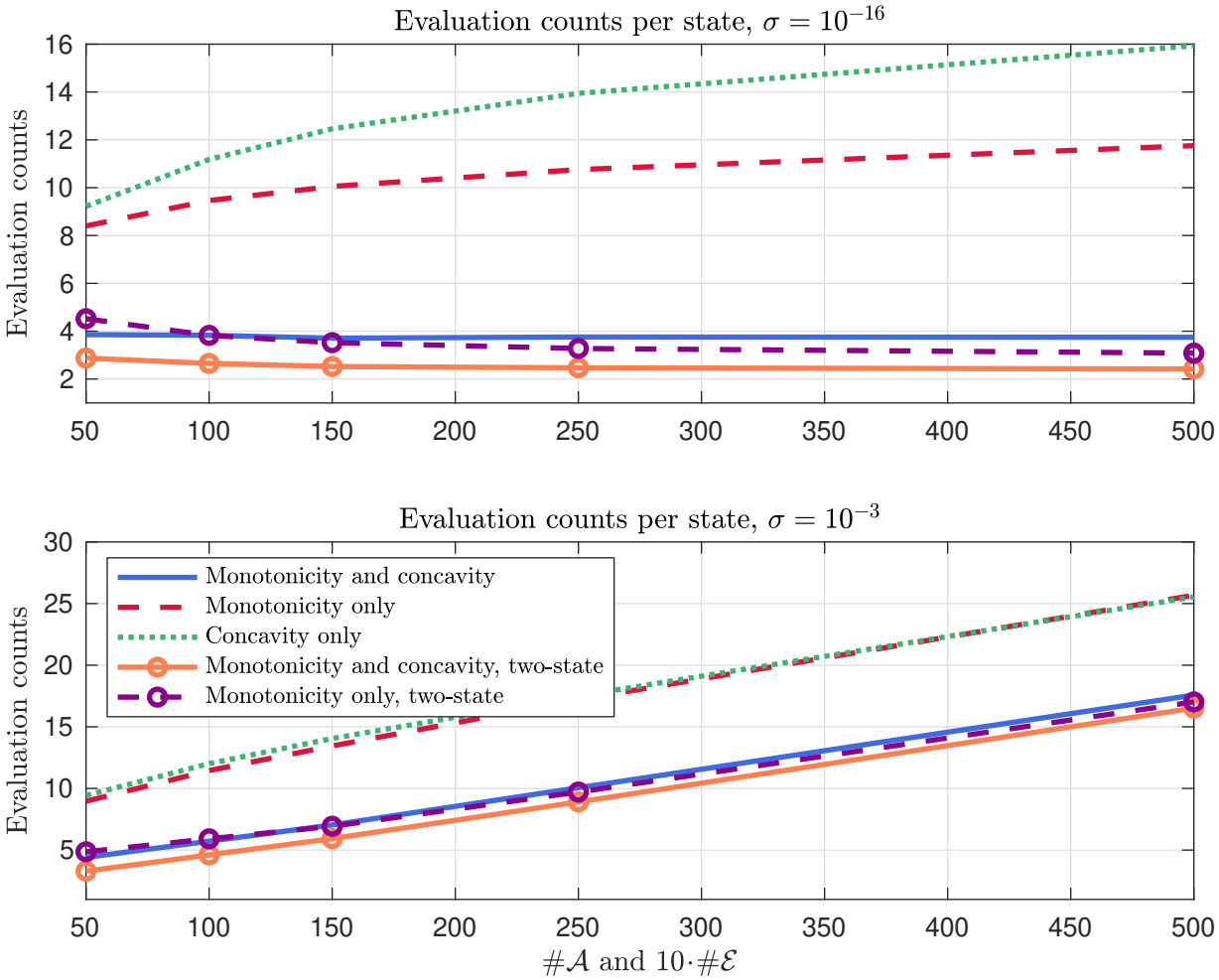

Figure 12: Evaluations per state

References

S. R. Aiyagari. Uninsured idiosyncratic risk and aggregate savings. *Quarterly Journal of Economics*, 109(3):659–684, 1994.

C. Arellano, Y. Bai, and G. Mihalache. Monetary policy and sovereign risk in emerging economies (NK-Default). Working Paper 19-02, Stony Brook University, 2019.

T. Bewley. A difficulty with the optimum quantity of money. *Econometrica*, 51(5):145–1504, 1983.

M. Bruins, J. A. Duffy, M. P. Keane, and A. A. Smith, Jr. Generalized indirect inference for discrete choice models. Mimeo, July 2015.

S. Chatterjee and B. Eyigungor. Maturity, indebtedness, and default risk. *American Economic Review*, 102(6):2674–2699, 2012.

S. Chatterjee and B. Eyigungor. Continuous Markov equilibria with quasi-geometric discounting. *Journal of Economic Theory*, 163:467–494, 2016.

S. Chatterjee, D. Corbae, K. Dempsey, and J.-V. Ríos-Rull. A theory of credit scoring and competitive pricing of default risk. Mimeo, August 2015.

D. Childers. Solution of rational expectations models with function valued states. Mimeo, 2018.

K. X. Chiong, A. Galichon, and M. Shum. Duality in dynamic discrete-choice models. *Quantitative Economics*, 1:83–115, 2016.

M. Dvorkin, J. M. Sánchez, H. Sapriza, and E. Yurdagul. Sovereign debt restructurings: A dynamic discrete choice approach. Working paper, Federal Reserve Bank of St. Louis, 2018.

G. Gordon and P. A. Guerron-Quintana. On regional migration, borrowing, and default. Mimeo, 2018.

G. Gordon and S. Qiu. A divide and conquer algorithm for exploiting policy function monotonicity. *Quantitative Economics*, 9(2):521–540, 2018a.

G. Gordon and S. Qiu. Supplement to "A divide and conquer algorithm for exploiting policy function monotonicity". Mimeo, 2018b.

G. Gordon and P. A. Querron-Quintana. A quantitative theory of hard and soft sovereign defaults. Mimeo, 2018.

J. C. Hatchondo and L. Martinez. Long-duration bonds and sovereign defaults. *Journal of International Economics*, 79(1):117–125, 2009.

J. C. Hatchondo, L. Martinez, and C. Sosa-Padilla. Voluntary sovereign debt exchanges. *Journal of Monetary Economics*, 61:32–50, 2014.

J. C. Hatchondo, L. Martinez, and C. Sosa-Padilla. Debt dilution and sovereign default risk. *Journal of Political Economy*, 124(5):1383–1422, 2016.

B. Heer and A. Maußner. *Dynamic General Equilibrium Modeling: Computational Methods and Applications*. Springer, Berlin, Germany, 2005.

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

F. Iskhakov, T. H. Jørgensen, J. Rust, and B. Schjerning. The endogenous grid method for discrete-continuous dynamic choice models with (or without) taste shocks. *Quantitative Economics*, 8(2): 317–365, 2017.

K. L. Judd. *Numerical Methods in Economics*. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1998.

J. Laitner. Household bequest behavior and the national distribution of wealth. *The Review of Economic Studies*, 46(3):467–483, 1979.

R. D. Luce. *Individual Choice Behavior: A Theoretical Analysis*. Wiley, New York, 1959.

D. McFadden. Conditional logit analysis of qualitative choice behavior. In P. Zarembka, editor, *Frontiers in Econometrics*, chapter 4. Academic Press, 1974.

R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10:6–38, 1995.

G. Mihalache. Sovereign default resolution through maturity extension. Mimeo, 2019.

J. Rust. Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher. *Econometrica*, 55(5):999–1033, 1987.

C. Santos and D. Weiss. "Why not settle down already?" A quantitative analysis of the delay in marriage. *International Economic Review*, 57(2):425–452, 2016.

G. Tauchen. Finite state Markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, 20(2):177–181, 1986.

D. M. Topkis. Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321, 1978.

A Calibration [Not for publication]

For the SIM model, I follow the calibration in Aiyagari (1994) fairly closely, using constant relative risk aversion (CRRA) of 3, a discount factor of $\beta = 0.96$, depreciation of $\delta = 0.08$, and a capital

share of $\alpha = 0.36$. The earnings process follows an AR(1) with a persistence parameter of 0.9 and innovation standard deviation of 0.4. The asset grid is linearly spaced from zero to $\delta^{1/(\alpha-1)}$, where the maximum is essentially that used by Aiyagari. I discretize the income process using Tauchen's (1986) method with a varying number of points (depending on whether the one- or two-state algorithm is being used) and a linearly spaced grid covering three unconditional standard deviations.

For the sovereign default model, I follow closely the calibration in Chatterjee and Eyigungor (2012). The default cost $\phi(y) = \max(0, y\phi_0 + y^2\phi_1)$ with $(\phi_0, \phi_1) = (-0.18819, 0.24558)$. The discount factor is $\beta = 0.95402$, and flow utility has CRRA of 2. The income process is an AR(1) with a persistence parameter of 0.9485 and innovation standard deviation of 0.27092. The maturity rate $\lambda = 0.05$, the coupon rate is $z = .03$, and the probability of escaping autarky is $\xi = 0.0385$. I discretize the income process using Tauchen's method with 300 points and a linearly spaced grid covering three unconditional standard deviations. (A large number of discretization points smooths out jumps in $T \circ q$, allowing a smaller taste-shock size to be used while still obtaining convergence.) The bond grid is linearly spaced from -0.7 to 0.

B Proofs [Not for publication]

Proof of proposition 1. Since i' is numerically relevant at i, $\mathbb{P}(i'|i) \geq \varepsilon$. By definition, $\mathcal{L}^*_\sigma(i'|i) = \mathcal{L}_\sigma(g(i), i'|i) = \sigma \log\left(\frac{\mathbb{P}(i'|i)}{\mathbb{P}(g(i)|i)}\right)$. Since $\mathbb{P}(i'|i) \geq \varepsilon$, $\sigma \log\left(\frac{\mathbb{P}(i'|i)}{\mathbb{P}(g(i)|i)}\right) \geq \sigma \log\left(\frac{\varepsilon}{\mathbb{P}(g(i)|i)}\right)$. Lastly, since $\mathbb{P}(g(i)|i) \leq 1$, $\sigma \log\left(\frac{\varepsilon}{\mathbb{P}(g(i)|i)}\right) \geq \sigma \log(\varepsilon) =: \underline{\mathcal{L}}_\sigma$. Hence, $\mathbb{P}(i'|i) \geq \varepsilon$ implies $\mathcal{L}^*_\sigma(i'|i) \geq \underline{\mathcal{L}}_\sigma$. □

Proof of proposition 2. Recall $\mathcal{L}_\sigma(j', i'|i) = U(i, i') - U(i, j')$. Increasing differences implies $U(i, i') - U(i, j') < U(j, j') - U(j, j')$ if $i' > j'$ and $j > i$. Consequently, $\mathcal{L}_\sigma(j', i'|\cdot)$ is increasing if $i' > j'$. Similarly, increasing differences implies $U(i, i') - U(i, j') > U(j, j') - U(j, j')$ if $j' > i'$ and $j > i$. Hence, $\mathcal{L}_\sigma(j', i'|\cdot)$ is decreasing if $j' > i'$. Summarizing,

$$\mathcal{L}_\sigma(j', i'|\cdot) \text{ is } \begin{cases} \text{decreasing if } j' > i' \\ \text{increasing if } j' < i' \end{cases}. \tag{13}$$

Now suppose $\mathcal{L}^*_\sigma(i'|i) = \mathcal{L}_\sigma(g(i), i'|i) < \underline{\mathcal{L}}_\sigma$, $i' < g(i)$ and $j > i$. Then $\mathcal{L}_\sigma(g(i), i'|\cdot)$ is decreasing. Since $j > i$, one has $\mathcal{L}_\sigma(g(i), i'|i) \leq \mathcal{L}_\sigma(g(i), i'|j) < \underline{\mathcal{L}}_\sigma$ as well. Therefore,

$$\begin{aligned} \underline{\mathcal{L}}_\sigma &> \mathcal{L}_\sigma(g(i), i'|j) \\ &= U(j, i') - U(j, g(i)) \\ &\geq U(j, i') - U(j, g(j)) \\ &= \mathcal{L}_\sigma(g(j), i'|j) \\ &= \mathcal{L}^*_\sigma(i'|j), \end{aligned}$$

where the second inequality follows from the optimality of $g(j)$.

Symmetrically, suppose $\mathcal{L}^*_\sigma(i'|i) < \underline{\mathcal{L}}_\sigma$, $i' > g(i)$, and $j < i$. Then $\mathcal{L}_\sigma(g(i), i'|\cdot) < \underline{\mathcal{L}}_\sigma$ is increasing. Since $j < i$, one has $\mathcal{L}_\sigma(g(i), i'|j) \leq \mathcal{L}_\sigma(g(i), i'|i) < \underline{\mathcal{L}}_\sigma$ as well. Therefore,

$$\begin{aligned}\underline{\mathcal{L}}_\sigma &> \mathcal{L}_\sigma(g(i), i'|j)\\ &= U(j, i') - U(j, g(i))\\ &\geq U(j, i') - U(j, g(j))\\ &= \mathcal{L}^*_\sigma(i'|j),\end{aligned}$$

where the second inequality follows from the optimality of $g(j)$. □

Proof of proposition 3. Since U is concave, $\mathcal{L}^*_\sigma(i'|i) = U(i, i') - U(i, g(i))$ is concave (thinking of $\mathcal{L}^*$ as a function taking $\{1, \ldots, n\} \times \{1, \ldots, n'\}$ to $\mathbb{R}$, which is denoted $\mathcal{L}^*_\sigma(i'|i)$ rather than $\mathcal{L}^*_\sigma(i, i')$).

To prove $\mathcal{L}^*_\sigma(i'|i) \geq \mathcal{L}_\sigma$ and $\mathcal{L}^*_\sigma(i'+1|i) < \underline{\mathcal{L}}_\sigma$ implies $\mathcal{L}^*_\sigma(j'|i) < \underline{\mathcal{L}}_\sigma$ for all $j' > i'$, suppose not. Then there is some $j' > i'$ having $\mathcal{L}^*_\sigma(j'|i) \geq \underline{\mathcal{L}}_\sigma$ (and clearly $j' > i'+1$). Consequently, the upper contour set $\{\tilde{i}'|\mathcal{L}^*_\sigma(\tilde{i}'|i) \geq \underline{\mathcal{L}}_\sigma\}$ has i' and $j' > i'$ in it but not $i'+1$. Therefore, $\mathcal{L}^*_\sigma$ is not concave, which is a contradiction. The fact that $\mathcal{L}^*_\sigma(i'|i) \geq \mathcal{L}_\sigma$ and $\mathcal{L}^*_\sigma(i'-1|i) < \underline{\mathcal{L}}_\sigma$ implies $\mathcal{L}^*_\sigma(j'|i) < \underline{\mathcal{L}}_\sigma$ for all $j' < i'$ can be proved in the same way. □

Proof of proposition 4. Note that $\mathcal{L}^*_\sigma(i'|i) = U(i, i') - U(i, g(i))$ is invariant to σ. Additionally, $\mathcal{L}^*_\sigma(i'|i) \leq 0$ with equality if and only if $i' = g(i)$. The cutoff $\underline{\mathcal{L}}_\sigma = \sigma \log(\varepsilon) < 0$ can be made arbitrarily close to zero by decreasing σ. Consequently, for a small enough σ, the condition $\mathcal{L}^*_\sigma(i'|i) \geq \underline{\mathcal{L}}_\sigma$ is satisfied if and only if $(i', i) = (g(i), i)$. In this case, $l(i)$ and $h(i)$ equal $g(i)$.

For l and h equal to g, step 4 evaluates $U(m, i')$ only at $\{g(\underline{i}), \ldots, g(\bar{i})\}$. This is the same as in Step 3 of Gordon and Qiu (2018a). Similarly, step 2 (3) evaluate $U(1, i')$ ($U(\overline{n}, i')$) only at $\{1, \ldots, n'\}$ ($\{g(1), \ldots, n'\}$). This is the same as in step 1 (2) of Gordon and Qiu (2018a). Since the same divide and conquer step is used, the total number of evaluations of U does not exceed $(n'-1)\log_2(n-1) + 3n' + 2n - 4$ (proposition 1 in Gordon and Qiu, 2018a). Simplifying this expression by making it slightly looser gives the desired result. □

Proof of proposition 5. As in the proof of proposition 4, we note that for small enough σ one has $l(i)$ and $h(i)$ equal to $g(i)$. In this case, the algorithm reduces to binary concavity as stated in Gordon and Qiu (2018a) except for steps 3 and 4 of algorithm 2. For small enough preference shocks, each of these requires 1 additional evaluation—specifically $U(i, g(i)-1)$ and $U(i, g(i)+1)$, respectively—, which results in $2n$ extra evaluations overall. Gordon and Qiu (2018a) show binary concavity for a fixed i requires at most $2\lceil\log_2(n')\rceil - 1$ evaluations if $n' \geq 3$ and n evaluations for $n' \leq 2$. Consequently, for $n' \geq 3$ there are at most $n \cdot (2 + 2\lceil\log_2(n')\rceil - 1)$ evaluations for proposition 5 if $n' \geq 3$. Simplifying, there are fewer than $2n\log_2(n') + 3n$ evaluations required. □

Proof of proposition 6. As in the proof of proposition 4, we note that for small enough σ one has $l(i)$ and $h(i)$ equal to $g(i)$. In this case, the algorithm reduces to binary monotonicity with binary concavity as stated in Gordon and Qiu (2018a) except for steps 3 and 4 of algorithm 2. For small

enough preference shocks, each of these requires 1 additional evaluation—specifically $U(i, g(i) - 1)$ and $U(i, g(i) + 1)$, respectively—, which results in $2n$ extra evaluations overall. Consequently, the $6n + 8n' + 2\log_2(n' - 1) - 15$ bound in proposition 1 of Gordon and Qiu (2018a) becomes $8n + 8n' + 2\log_2(n' - 1) - 15$. Simplifying slightly by making the bounds looser gives the desired result. □

Proof of proposition 7. As in the proof of proposition 4, we note that for small enough σ, one has l and h equal g. Consequently, when using algorithm 1 at each step, the algorithm reduces to the two-state binary monotonicity algorithm in Gordon and Qiu (2018a). Proposition 2 of Gordon and Qiu (2018a) then gives the desired upper bound and $O(n_1 n_2)$ behavior. □