A Sparse Endogenous Grid Method for Quickly Solving Medium-Dimension Economic Models

Mitchell VanVuren (Yale)

April 2024

PRELIMINARY AND INCOMPLETE

Please click here for the latest version

Abstract

Sparse grids are useful for solving large economic models but require knowledge of the value function at carefully chosen gridpoints. Endogenous grid methods avoid numerical optimization but relinquish control over the grid on which updated values are returned. This paper presents a method that resolves this tension, allowing both to be implemented together. The envelope condition provides a guess of the policy function that can be used to control the grid on which the updated value function is known. Although not correct in general, this guess will be correct after convergence is achieved, ensuring that the fixed point of the method solves the desired Bellman equation. The result is a fast and robust method for solving medium-dimension economic models.

For their helpful comments, I thank seminar participants at Yale. Any errors remain my own.
Mitchell VanVuren: Yale Research Initiative on Innovation and Scale (Y-RISE) email: mitchell.vanvuren@yale.edu

1. Introduction

As the use of microdata within economics has grown, the desire to solve economic models with high levels of heterogeneity, in order to match the patterns observed in the data, has also increased. But an additional dimension of heterogeneity almost invariably translates to additional state variables. Consequently, incorporating even one or two additional dimensions results in substantial computational difficulties due to the well-known "curse of dimensionality". Substantial advances in solving these problems have been made, but the largest improvements have focused on how to compute perturbations (i.e. business cycles) from steady-states (e.g. Ahn, Kaplan, Moll, Winberry & Wolf 2018, Auclert, Bardóczy, Rognlie & Straub 2021). In terms of solving steady-states themselves, the endogenous grid method of Carroll (2006) remains the standard approach (see Auclert et al. 2021, as an example), particularly for models with occasionally binding constraints (i.e. borrowing constraints).

Sparse grids are often employed to mitigate the computational cost of additional state variables (e.g. Judd, Maliar, Maliar & Valero 2014, Brumm & Scheidegger 2017) but present an immediate, practical tension — these methods rely on knowledge of the value function at careful chosen points in the state-space. In contrast, the endogenous grid method (as the name suggests) requires relinquishing control over the grid upon which the value function is known in order to avoid numerical optimization. Thus the current state of affairs is one in which economists must choose between avoiding numerical optimization and facing the curse of dimensionality or avoiding the curse and employing costly optimization routines.

Building on the standard endogenous grid approach, this paper presents a method that allows both the analytical solution of the optimization problem (as in the endogenous grid method) and the use of sparse grids, eliminating the need to choose between the two. The key insight is that the traditional endogenous grid approach takes the value of the value function at some (exogenously specified) gridpoint and returns the updated value at some (endogenously determined) gridpoint. If the correspondence between the gridpoint fed into the method and the gridpoint returned by the method were known *a priori*, one could use the endogenous grid method to update the value function at any desired gridpoint by simply plugging in the corresponding gridpoint.

In equilibrium, this correspondence is exactly given by the (inverse of the) policy function. Of course, the policy function is unknown (if it were known, there would be no need to perform an optimization step at all) so this approach appears at first to be a dead end. The second insight of this paper is that the envelope conditions of the problem can be used to construct a guess for the policy function before the optimization step is performed (and at low computational cost). However, this guess is not correct in general and, consequently, feeding the implied gridpoint into the endogenous grid method will not return the desired gridpoint.

The trick, and the major departure of this method from previous methods, is to treat this guess *as if* it were correct and assign the updated value function to the desired gridpoint anyway. The updated guess of the value function, then, is not the solution to the traditional T operator of textbook value function iteration. However, because the envelope condition will correctly return the policy function when the guess of the value function is correct, the fixed point of the T operator remains a fixed point of this new iterative process. In the end, this yields an approach for finding the value function that both avoid numerical optimization (in the same way that the endogenous grid method does) and allows one to precisely choose the gridpoints at which the update values of the value function are known (allowing the use of sparse grids).

A substantial concern with this approach is that erroneously treating the policy function implied by the envelope condition as correct may impede convergence. Further analysis of this issue reveals that, under typical regularity conditions, there are two reasons to suspect that this may not be a major issue. First, the fixed point of this new method — which I refer to as the sparse endogenous grid method (SEGM) — is unique (and, as stated above, the same as the fixed point of the traditional VFI approach), eliminating the concern that the method may find fixed points that do not correspond to the desired solution. Second, the mapping (in function space) implied by the method can be shown to be a local contraction mapping (in the sense that there is a neighborhood, centered at the fixed point, in which the mapping satisfies the definition of a contraction). Thus while convergence is not guaranteed for any initial guess, there are strong reasons to suspect that it will converge.

Applying the method to a simple example supports this intuition and, in fact, yields an even stronger practical result. Comparing otherwise equivalent implementations of a standard endogenous grid approach and the sparse endogenous grid method (i.e. abstracting temporarily from sparse grids and comparing the two on equivalent dense grids with linear interpolation), the SEGM does not require more iterations than the standard method to achieve convergence and, in fact, seems to require slightly fewer. Thus the advantages of the SEGM appear to come "for free" and there is essentially no downside to leveraging this approach.

One major advantage of the sparse endogenous grid method relative to other methods that avoid numerical optimization (e.g. the Euler equation approach of Judd et al. 2014) is that it remains within the value function iteration paradigm and can leverage the various improvements and adaptions to VFI that have been developed. Of these, the most generally useful is the policy function iteration approach developed in Rendahl (2022). After laying out the basic method, I show that these insights can be applied to the SEGM (as long as the chosen sparse interpolation method satisfies some simple properties) to further accelerate computation by reducing the number of iterations required to achieve convergence.

As a simple example, I implement the method using efficient Smolyak interpolation (developed in Judd et al. 2014) for an Aiyagari (1994) model with a high-dimensional income process. With as few as 5 state variables, the SEGM is already faster than a baseline dense grid EGM approach by a factor of 50 (and the advantage becomes overwhelmingly large as the number of state variables increases).

Of course, this improvement comes from the fact that the advantage of employing a sparse grid (the reduction in the number of gridpoints) outweighs the disadvantage (increased cost of performing interpolation). In other words, it largely acts as a measure of the gains from the "sparse" component of the SEGM (and the efficiency of the chosen sparse implementation) without speaking directly to the gains from the "endogenous grid" component (which, ultimately, is the contribution of this paper). It is hard to benchmark the gains from this second component directly, as variety in the implementation of sparse approaches is vast. However, a useful heuristic is that performing interpolation compromises the (often vast) majority of the computational cost in sparse methods. The SEGM requires only a single interpolation per gridpoint thus the advantage over another sparse approach is proportional to the number of interpolations required — for example, relative to a method whose numerical optimization requires 10 interpolations per gridpoint, the SEGM is roughly 10 times faster.

Intended Usage and Related Literature: This paper falls into the large literature aimed at globally solving high-dimension economic models. Most closely related are those that focus on leveraging sparse grids to reduce the number of gridpoints required (Judd et al. 2014, Brumm & Scheidegger 2017). Relative to these approaches, the contribution of the SEGM is to dramatically reduce computation time by removing the need for numerical optimization while remaining within a value (or policy) function iteration framework. This allows the method to be applied to a wide variety of problems (due to a large literature aimed at adapting the traditional EGM to different types of problems) while retaining stable convergence properties.

A second strand of this literature aims at solving economic models by approximating equilibrium conditions directly via neural networks (Fernandez-Villaverde, Nuno, Sorg-Langhans & Vogler 2020, Kahou, Fernández-Villaverde, Perla & Sood 2021, Maliar, Maliar & Winant 2021). Because sparse grids do not entirely avoid the curse of dimensionality (and only mitigate it) while neural networks do, these approaches remain tractable in much high dimension than sparse grid based approaches (i.e. dimensions larger 50). As above, one advantage of the SEGM relative to these approaches is the stability and consistency of convergence that arises from remaining withing a VFI framework. A second advantage that arises from this is that the implementing the SEGM is fairly straightforward conditional on basic knowledge of value function iteration and does not require learning a new paradigm, as is the case for machine-learning-based approaches.

The natural use-case for the SEGM stems from these comparative advantages. It is best suited for practitioners interested in solving complex, medium-dimension economic models and, in particular, estimating these models via procedures that require convergence under a wide variety of parameterizations (i.e. simulated method of moments). For models of this type, the SEGM pushes the frontier in terms of speed while remaining a robust solution method (necessary if the solution of the model is to be imbedded with a simulated method of moments estimation framework).

2. A Sparse Endogenous Grid Method

A canonical Bellman equation with an occasionally binding constraint can be written

$$\begin{aligned} V &= T(V) \\ (TV)(a,y) &= \max_{c,a'} u(c) + \beta \mathbb{E}[V(a',y')|y] \\ s.t.\ \ 0 &= f(c,a,a',y) \\ 0 &\leq g(c,a,a',y) \\ y' &\sim M(y) \end{aligned} \tag{1}$$

where c (consumption) is the control variable, a (assets) is the non-stochastic endogenous state, and y (income) is the stochastic exogenous state following Markov process M.[1] The (unknown *a priori*) value function V is a fixed point of the Bellman operator T, and the textbook solution approach is to show that T is a contraction and thus the contraction mapping theorem ensures that iteratively applying T to any initial guess for the value function will eventually converge to the correct solution.

Implementing this approach requires a representation of V that can be stored and manipulated in a computer which almost invariably takes the form of a collection of gridpoints in the statespace $\mathbf{x}$, a collection of real numbers corresponding to values of V at each gridpoint $\mathbf{V}$, and an interpolation scheme to approximate the value of V at points in the statespace not contained in $\mathbf{x}$. The natural approach to implementing the Bellman operator is, for any given set of gridpoints $\mathbf{x}$ and guess for the value function $\mathbf{V}$, to solve the optimization problem in (1) for each $x_i \in \mathbf{x}$, update the corresponding $V_i \in \mathbf{V}$ to arrive at an updated guess $\hat{\mathbf{V}}$, and repeat this process until convergence. Unfortunately, solving the optimization problem in this manner is quite computationally intensive and involves expensive numerical optimization or root-finding techniques.

The endogenous grid method (EGM) of Carroll (2006) exploits the fact that many interpolation schemes (e.g. linear, polynomial, and spline) are fairly agnostic towards precisely which points in the statespace are included in $\mathbf{x}$, providing a degree of freedom that can be used to ease computation. This can be leveraged

[1]For brevity, I restrict to the case of a single control, exogenous state, endogenous state, and constraint; however, everything said applies to the case of multiples as well.

by noting that if $\mathbb{E}[V_a(a', y')|y]$ is known for a particular value of (a', y), the first order condition $\mathbb{E}[V_a(a', y')|y] = \frac{f_a}{f_c}u(c)$ can often be used to find the optimal value of c without numerical optimization (for areas of the state-space in which $g > 0$) and, through the budget constraint, the value of a such that the values of (a', c) are the optimal choices in state (a, y).[2] The EGM can then be implemented by starting with a collection of gridpoints $\mathbf{x}$ and a guess $\mathbf{V}$, choosing a new set of gridpoints $\mathbf{z}$, solving for (a, y, a', c) for each $(a', y) \in \mathbf{z}$ analytically as described, and finally updating *both* the grid $\mathbf{x}$ (using the pairs (a, y)) and the guess $\mathbf{V}$ (using the pairs (c, a')) to obtain the updated value function.

A common refinement of this technique is to note that only the derivative of V matters when determining the value of c using the first order condition, and the level of V itself does not enter anywhere. Thus accuracy can be improved by guessing and updating a guess of $\mathbf{V}_a$ directly, rather than approximating V_a using a guess $\mathbf{V}$. After the tuples (a, y, a', c) are determined, the updating step can then occur via the envelope condition $V_a(a, y) = -\frac{f_a}{f_c}u'(c)$, rather than by applying the T operator. This version of the baseline endogenous grid method (laid out explicitly below) serves as the jumping-off point for the sparse endogenous grid method.

Method (Endogenous Grid Method)**.** *Begin with an endogenous grid $\mathbf{x}$, an initial guess $\mathbf{V}_a$ at each gridpoint in $\mathbf{x}$, an interpolation method $i(z; \mathbf{x}, \mathbf{V}_a)$ and an exogenous grid $\mathbf{z}$. Then...*

1. *Interpolate and compute expectations to obtain $\mathbb{E}[V_a(a', y')|y]$ for each $(a', y) \in \mathbf{z}$*

2. *Use either the first-order condition $\beta\mathbb{E}[V_a(a', y')|y] = u'(c)$ and the budget constraint $f(c, a, a', y) = 0$ (for cases where $g > 0$) or the occasionally binding constraint $g(c, a, a', y) = 0$ and the budget constraint $f(c, a, a', y) = 0$ (for cases where $g = 0$) to find (c, a) for each $(a', y) \in \mathbf{z}$.*[3]

3. *Use the envelope condition $V_a = -\frac{f_a}{f_c}u'(c)$ to obtain an updated guess for the derivative of the value function $\mathbf{V}_a^{up}$ at each (c,a,a′,y).*

4. *Update both the grid $\mathbf{x}$, using (a, y), and the guess $\mathbf{V}_a$, using $\mathbf{V}_a^{up}$.*

Repeat steps 1-4 until some convergence criterion is met (typically between the values of $\mathbb{E}[V_a(a', y')|y]$ computed in step 1 in consecutive iterations, as these are defined on the same

[2]For areas where $g = 0$, a and c can be determined by solving $f(c, a, a', y) = 0, g(c, a, a', y) = 0$.

[3]Here, I skip over the minor detail of determining which case a particular pair (a', y) falls into as it is immaterial to the main point.

grid in all iterations). Then the function $i(z; \mathbf{x}, \mathbf{V}_a)$ approximates (the derivative of) the solution to the Bellman equation (1).

Endogenously choosing the grid $\mathbf{x}$ (as the name of the method implies) eliminates the need for expensive numerical optimization but relinquishes control over the precise set of gridpoints $\mathbf{x}$. This leads to the tension highlighted in the introduction, namely that sparse grids (or other improvements that rely on precise control of $\mathbf{x}$) cannot also be implemented.[4]

Once the problem is viewed this way, a potential solution jumps out. Although the endogenous grid method gives up degrees of freedom in the choice of $\mathbf{x}$ to ease computation, it also introduces new degrees of freedom, namely, the choice of $\mathbf{z}$. For any desired grid $\hat{\mathbf{x}}$ and guess for V_a given by $i(z; \mathbf{x}, \mathbf{V}_a)$, if we knew the choice of $\mathbf{z}$ such that the grid returned in step 4 were $\hat{\mathbf{x}}$, the issue would be resolved; we could use our freedom of choice in $\mathbf{z}$ to ensure that we were always operating on grid $\hat{\mathbf{x}}$.

If our guess $i(z; \mathbf{x}, \mathbf{V}_a)$ is correct (i.e. it approximates the solution to 1), then the choice of $\mathbf{z}$ that leads to the desired grid $\hat{\mathbf{x}}$ could be constructed using the policy function $a'^*(a, y)$ that solves the Bellman operator. In particular, constructing $\mathbf{z}$ as $\{(a'^*(a, y), y) | (a, y) \in \hat{\mathbf{x}}\}$ ensures that $\hat{\mathbf{x}}$ will be returned — the endogenous grid method works by treating a value in $\mathbf{z}$ as the value of the state tomorrow a' and returns a gridpoint that correspond to the state today a in which a' is the optimal choice. Thus the mapping between chosen and return gridpoints is the inverse of the policy function.

Of course, the policy function is not known; if it were, there would be no need to solve for V. The key insight of this paper is that the envelope condition can be used to construct a guess for $a'^*(a, y)$.[5] In particular, the envelope condition implies that

$$c^*(a, y) = u'^{-1}\big(-\frac{f_c V_a(a, y)}{f_a}\big) \tag{2}$$

[4]A second, more minor issue is that basic "dense grid" interpolation methods are only truly agnostic towards gridpoints in the case of a single endogenous state variable. When there are two or more endogenous states (e.g. two-asset models), even basic multi-dimensional interpolation methods are less agnostic often requiring hybrid methods (e.g. Ludwig & Schön 2018), although there are clever workarounds in some models (e.g. Auclert et al. 2021). Thus a method that allows control over $\mathbf{x}$ offers some advantage even in cases where sparse grids are not used.

[5]Another tempting option would be to use the policy function from the last iteration as a guess for the policy function of this iteration. This is an approach often employed in Euler equation iteration. In practice, this approach appears to be inferior to using the envelope condition — the convergence of the resulting algorithm is somewhat finicky and often slow.

which can typically by solved analytically to find the optimal choice of c^* and combined with the budget constraint to find $a'^*(a, y)$.

However, the guess of the policy function implied by (2), call it $\tilde{h}$, is not correct in general. As a result, constructing $\mathbf{z}$ as $\{(\tilde{h}(a, y), y)|(a, y) \in \hat{\mathbf{x}}\}$ does not actually ensure that the desired grid $\hat{\mathbf{x}}$ is returned. The second necessary insight is to notice that treating the guess as correct and assigning the returned values of $\mathbf{V}_a$ to $\hat{\mathbf{x}}$ anyways (i.e. regardless of whether or not $\hat{\mathbf{x}}$ is actually returned) does not change the fact that the solution to (1) is a fixed point of the iterative procedure (as $\tilde{h}$ will be equal to h if the guess is correct).

This, finally, suggests the sparse endogenous grid method.

Method (Sparse Endogenous Grid Method)**.** *Begin with an exogenous grid $\hat{\mathbf{x}}$, an initial guess $\mathbf{V}_a$ at each gridpoint in $\hat{\mathbf{x}}$, and an interpolation method $i(z; \hat{\mathbf{x}}, \mathbf{V}_a)$. Then...*

1. *Evaluate $\mathbf{V}_a$ for each $(a, y) \in \hat{\mathbf{x}}$ and construct a guess for the policy function $\tilde{h}$ using the envelope condition (2) and the budget constraint.*
2. *Construct $\mathbf{z}$ as $\{(\tilde{h}(a, y), y)|(a, y) \in \hat{\mathbf{x}}\}$ (i.e. for each $(a, y) \in \hat{\mathbf{x}}$, the corresponding z is $(\tilde{h}(a, y), y)$). Note the correspondence between x and z.*
3. *Interpolate and compute expectations to obtain $\mathbb{E}[V_a(a', y')|y]$ for each $(a', y) \in \mathbf{z}$.*
4. *Use either the first-order condition $\beta\mathbb{E}[V_a(a', y')|y] = u'(c)$ and the budget constraint $f(c, a, a', y) = 0$ (for cases where $g > 0$) or the occasionally binding constraint $g(c, a, a', y) = 0$ and the budget constraint $f(c, a, a', y) = 0$ (for cases where $g = 0$) to find c for each $(a', y) \in \mathbf{z}$.*
5. *Use the envelope condition $V_a = -\frac{f_a}{f_c}u'(c)$ to obtain an updated guess for the derivative of the value function $\mathbf{V}_a^{up}$ at each (c, a', y).*
6. *Do not update the grid (i.e. it remains precisely $\hat{\mathbf{x}}$). Update the guess $\mathbf{V}_a$, assigning each value of V_a computed in step 5 to gridpoints in $\hat{\mathbf{x}}$ using the correspondence noted in step 2.*

Repeat steps 1-6 until some convergence criterion between iterations of $\mathbf{V_a}$ is met. Then the function $i(z; \hat{\mathbf{x}}, \mathbf{V}_a)$ approximates (the derivative of) the solution to the Bellman equation (1).

Steps 2 and 6 represent the key innovations of the sparse endogenous grid method. Step 2 uses the envelope condition to construct $\mathbf{z}$ based on the desired

$\hat{\mathbf{x}}$, and step 6 forgoes assign the updated values $\mathbf{V}_{\text{up}}$ to $\hat{\mathbf{x}}$ via the correspondence noted in step 2, rather than the value of a computed via the budget constraint.

At this point, the purpose of these modifications to the baseline EGM should be clear. The fact that the guess $\mathbf{V}_a$ is always defined on the grid $\hat{\mathbf{x}}$, which is chosen exogeneously at the beginning of the method, allows for dramatically greater flexibility in the choice of interpolation method i, as i is no longer required to be "grid-agnostic". The most obvious use-case for this flexibility, as the name of the method suggests, is to allow the use of sparse grid interpolation methods. But there are potential advantages even in situations not well-suited to the use of sparse grids (e.g. the speed of interpolation can often be improved when the grid is known, even for methods that could be grid-agnostic).

2.1. Convergence

Because incorrectness in the guess V_a generates incorrectness in the policy function implied by the envelope condition, an initial concern is that this incorrectness may be "self-sustaining" and generate fixed points that do not characterize the solution to the Bellman equation (1), impeding convergence. This concern can be addressed by defining the S **operator induced by the sparse endogenous grid method** as

$$\begin{aligned}(SV_a)(a,y) &= -\frac{f_a}{f_c}u'(c) \\ s.t.\;\; a' &= \tilde{h}(a,y,V_a(a,y)) \text{ if } 0 < \tilde{h}(a,y,V_a(a,y)) \\ u'(c) &= \beta\mathbb{E}[V_a(a',y')|y] \text{ if } 0 < \tilde{h}(a,y,V_a(a,y)) \\ 0 &= a' \text{ otherwise} \\ 0 &= f(c,a,a',y) \text{ otherwise}\end{aligned} \tag{3}$$

for $\tilde{h}$ defined as

$$\tilde{h}(a,y,V_a) = x \text{ where } x \text{ solves } 0 = f\left(u'^{-1}\big(-\frac{f_cV_a}{f_a}\big),a,x,y\right) \tag{4}$$

Mimicking the traditional T operator, SV_a corresponds to the function that would be returned after a single iteration of the method with initial guess V_a, assuming that the interpolation method i interpolates functions perfectly.

Once the S operator is written down formally, it is clear that any solution to (1) must have a derivative that is a fixed point of (3) as this follows directly from the envelope condition. More interestingly, in the case of concave solution V and utility function u (so that the first-order conditions are sufficient to characterize the solution) the converse is also true — any fixed point of (3) must be the derivative of a solution to (1).[6] Consequently, the uniqueness of the fixed point of S follows directly from the uniqueness of the solution to the canonical Bellman equation (1), eliminating the concern that there may be fixed points that do not correspond to the desired solution.

A second concern is that even if this incorrectness does not generate additional fixed points, it may generate cycles or other instabilities, and iteratively applying S to some initial guess may not converge to the fixed point. This concern turns out to be much more difficult to address than the first; unlike T, S is not a contraction in general so the Contraction Mapping Theorem does not apply.

It can be shown, however, that S is a contraction "locally" (under some mild assumptions that are satisfied by most models). The contraction is local in the sense that $\sup_{a,y} |(SW_{a,1})(a,y) - (SW_{a,2})(a,y)| \leq -\frac{f_a}{f_c}\beta \sup_a |W_{a,1}(a,y) - W_{a,2}(a,y)|$ is only guaranteed if $W_{a,1}$ and $W_{a,2}$ are sufficiently close to the fixed point V. Although not a guarantee of convergence, this does strongly suggest that the method will converge for good initial guesses.[7] Indeed, this appears be to the case in practice; my experience is that the method has converged smoothly and without issue, even for very precise ($\approx 10^{-14}$) tolerances in the maximum log-difference between iterations.

2.2. Implementation Details

The sparse endogenous grid method laid out above provides a way to update a guess of the (derivative of the) value function at a set of exogenously-specific gridpoints $\hat{\mathrm{x}}$ while avoiding numerical optimization, allowing the use of sparse grids. With numerical optimization eliminated, there remain two costly computational steps. The first is that each iteration of the SEGM requires evaluation of the interpolation function i per gridpoint. For basic interpolation methods (i.e. linear), the cost of evaluating an interpolant is small; however, sparse interpolation methods are often substantially slower. Second, expectations must be computed

[6] The easiest way to see this is to note that plugging the envelope condition and budget constraint into the first order equation arising from (1) yields the functional equation $V_a(a,y) = -\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)))]$ which must also be satisfied by any fixed point of (3).

[7] The formalization and proof of this statement are fairly unenlightening and are relegated to Appendix B.

once for each gridpoint per iteration (which, naively, would require many more interpolation calls).

The cost of these steps is largely determined by the properties of the interpolating function i and, consequently, the choice of i is a substantial determinant of the overall speed of the method. I opt to use the efficient Smolyak polynomials of Judd, Maliar, Maliar & Valero (2014) as they are (relatively) quick to evaluate and allow some optimizations that dramatically improve performance. Most importantly, the structure of efficient Smolyak polynomials is such that for any set of interpolants $\mathbf{b}$, the vector of interpolated values $i(\mathbf{b};\hat{\mathbf{x}},\mathbf{V}_a)$ takes the form

$$i(\mathbf{b};\hat{\mathbf{x}},\mathbf{V}_a)=\mathbf{A}(\mathbf{b},\hat{\mathbf{x}})\mathbf{V_a} \tag{5}$$

where $\mathbf{A}$ is an M-by-N matrix, M is the number of interpolants in $\mathbf{b}$, and N is the number of gridpoints in $\mathbf{x}$.

This property essentially says that the interpolated value at any (fixed) interpolant can be expressed as a linear combination of the function values $\mathbf{V}_a$ that depends only on the grids $\mathbf{b}$ and $\hat{\mathbf{x}}$ and not on the function values themselves. [8] As a result of this, many costly computations can be pre-computed and have their cost reduced to the cost of a simple matrix multiplication (at least within the iterative loop). This property can also be used to accelerate iteration, but I defer discussion of this until Section 3.

Interpolant Evaluation: As their name suggests, it is fairly quick to perform interpolation (i.e. compute $\mathbf{A}(\mathbf{b},\hat{\mathbf{x}})$) via efficient Smolyak polynomials — indeed, that is their purpose. To further reduce computation time, however we can leverage the fact that the SEGM only requires interpolation along endogenous state variables. That is, $\mathbf{z}=\{(\tilde{h}(a,y),y)|(a,y)\in\hat{\mathbf{x}}\}$ implies that $\mathbf{z}$ and $\hat{\mathbf{x}}$ align exactly in their values of y (the exogenous state) and only differ in their values for a (the endogenous state). Because $\hat{\mathbf{x}}$ is specified exogenously, this means that the values of the exogenous state variables in $\mathbf{z}$ at which i is evaluated do not change from iteration to iteration. In essence, if the point $(q,y_1,y_2,\ldots,y_k)$ is an interpolant needed for one iteration (for some q), then the point $(r,y_1,y_2,\ldots,y_k)$ will be an interpolant needed for the next iteration (for some potentially different r).

It turns out that efficient Smolyak polynomials are constructed in such a way

[8]Here, it is worth noting that this is not a property unique to Smolyak interpolation. For example, the adaptive sparse grids of Brumm & Scheidegger (2017) also exhibit this property.

that changing a single input in this way does not require recomputing the entire matrix A from scratch. In other words, knowledge of the row vector $\mathbf{A}((q_1, \ldots, q_n, y_1, \ldots, y_n), \hat{\mathbf{x}})$ can be used to determine $\mathbf{A}((r_1, \ldots, r_n, y_1, \ldots, y_n), \hat{\mathbf{x}})$ at lower computational cost. The full description of why this is true and precisely how it is implemented is beyond the scope of this paper, but combining this with the fact above that only the endogenous-state-components of the interpolants change from iteration to iteration substantially increases the speed of the algorithm. Because the reduction in computational time is larger when number of fixed variables is large, this approach offers the largest acceleration in models where the number of endogenous states is smaller relative to the number of exogenous (to the household) states, which describes the vast majority of models.[9]

Expectations: We can further leverage the linear structure of (5) to accelerate the computation of expectations. The standard approach to approximating expectations at a point z in the state-space is to use some quadrature rule, consisting of nodes $\mathbf{q}(z)$ and weights $\mathbf{w}$ so that $\mathbb{E}V_a(z) \approx \mathbf{w}'V_a(\mathbf{q}(z))$. Of course since the value of V_a is only know at certain gridpoints, $V_a(\mathbf{q}(z))$ must be interpolated as $\mathbf{A}(\mathbf{q}(z), \hat{\mathbf{x}})\mathbf{V}_a$ combining these two equations and stacking $\mathbb{E}V_a(z_i)$ into a k-by-1 vector corresponding to the expectations at k desired points yields

$$\mathbb{E}\mathbf{V_a}(\mathbf{z}) \approx \begin{bmatrix} \mathbf{w}'\mathbf{A}(\mathbf{q}(z_1), \hat{\mathbf{x}}) \\ \vdots \\ \mathbf{w}'\mathbf{A}(\mathbf{q}(z_k), \hat{\mathbf{x}}) \end{bmatrix} \mathbf{V}_a \equiv \mathbf{E}(\mathbf{z}, \hat{\mathbf{x}})\mathbf{V}_a \tag{6}$$

Because the grid on which $\mathbf{V}_a$ is known, $\hat{\mathbf{x}}$, is exogenously fixed and chosen ahead of time, the matrix $\mathbf{E}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$ can be computed ahead of time (i.e. before iteration and/or estimation begin), reducing the cost of computing expectations to a simple N-by-N matrix multiplication where N is the number of gridpoints in $\hat{\mathbf{x}}$.

$$\mathbb{E}\mathbf{V}_a \approx \mathbf{E}(\hat{\mathbf{x}}, \hat{\mathbf{x}})\mathbf{V}_a \tag{7}$$

The choice of quadrature rule is relatively unimportant, although well-chosen

[9] This approach was directly inspired by the Efficiently Updated Neural Networks of the Stockfish chess engine (The Stockfish developers (see AUTHORS file) n.d.). Although the structure of Smolyak polynomials is different enough from neural networks that the gains in efficiency are of a substantially smaller scale, the core intuition of leveraging previous computations to accelerate the computation at a value that is only slightly different than a previous value remains the same.

quadrature rules can improve accuracy of the approximation and reduce the computation burden of the initial calculation of $\mathbf{E}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$. In the example below, all the innovations to the exogenous state variables follow normal distributions. Thus it is natural to use a Gauss-Hermite quadrature designed to integrate with respect to these shocks. In particular, I use the sparse Gauss-Hermite quadrature rules implemented in the Julia package `SparseGrids.jl`.

3. Howard-Rendahl Acceleration

Other methods of iteration, such as Euler equation iteration or envelope condition iteration, can also be modified to use sparse grids (although none do so quite as naturally as the sparse endogenous grid method in the presence of borrowing constraints). The main advantage of the SEGM is that, unlike these methods, it continues to iterate backwards through the Bellman equation and remains conceptually very close to textbook value function iteration. As a result, it can be combined with a variety of already-developed techniques aimed at modifying VFI to solve atypical problems such as non-concave value functions (Fella 2014) or discrete choices (Iskhakov, Jørgensen, Rust & Schjerning 2017).

Of these, the largest (and most general) benefit comes from Howard acceleration, specifically the variant developed in Rendahl (2022) which I refer to as Howard-Rendahl (HR) acceleration. Convergence of (the derivative of) the value function can often take many iterations, particularly for models with high discount factors, leading to long computation times even when each individual iteration is relatively quick to perform. This slowness is a result of the fact that the updated value function $u(c^*) + \beta\mathbb{E}[V^{\text{pre}}(a'^*, y')|y]$ puts substantial weight (roughly proportion β) on the previous guess of the value function V^{pre} and only a small portion on the new policy function $u(c^*)$. In the SEGM, this is issue is made worse by the fact that a'^* is eliminated from the updating equation and instead replaced by $\tilde{h}(a; y, V_a)$ which increases dependence on the previous guess.

Conceptually, HR-acceleration mitigates this and increases the weight placed on the policy function by applying the new policy function c^* repeated so that the updated value function is given by $\mathbb{E}_0 \sum_{t=0}^{T} \beta^t u(c^*) + \beta^{T+1}\mathbb{E}[V^{\text{pre}}(a'^*, y')|y]$ where larger values of T correspond to more weight placed on c^*. Taking the limit as $T \to \infty$ and making a recursive substitution yields

$$V^{\text{new}}(a, y) = u(c^*(a, y)) + \beta\mathbb{E}[V^{\text{new}}(a'^*(a, y), y')|y] \tag{8}$$

In essence, the value function is now updated by assuming that an agent acts according to policies c^*, a^* forever, rather than a single period (as in traditional VFI), and updating the value function accordingly.

The first step in implementing this approach in the SEGM is to replace equation (8) with an equation in terms of V_a^{new}. This can be done by differentiating (8) with respect to a[10] (This is a key insight taken from Rendahl 2022) which yields

$$\begin{aligned} V_a^{\text{new}}(a,y) &= u'(c^*)c_a^* + \beta\mathbb{E}[V_a^{\text{new}}(a'^*(a,y),y')|y]a_a'^* && (9) \\ &= u'(c^*)c_a^* + \beta\big(\frac{f_a + f_c c_a^*}{-f_{a'}}\big)\mathbb{E}[V_a^{\text{new}}(a'^*(a,y),y')|y] && (10) \end{aligned}$$

where the second line follows from the first via the budget constraint.

This yields an updating equation for V_a^{new} directly but is, in general, a non-linear functional equation and thus difficult to solve. The key is that if the interpolation method chosen satisfies the property described in (5) (i.e. the value of any interpolant can be expressed as a linear combination of the function values at the gridpoints), we can use this updating equation to write a system of linear equations that update the values $\mathbf{V}_a$ for the gridpoints $\hat{\mathbf{x}}$. Making these substitutions, as well as substituting $\tilde{\mathbf{h}}$ (the vector of values for $\tilde{h}$ at each point in $\hat{\mathbf{x}}$) for a'^* yields

$$\mathbf{V}_a^{\text{new}} = u'(\mathbf{c})\mathbf{dc} + \beta\big(\frac{\mathbf{f}_a + \mathbf{f}_c\mathbf{dc}}{-\mathbf{f}_{a'}}\big)\mathbf{A}(\tilde{\mathbf{h}},\hat{\mathbf{x}})\mathbf{E}(\hat{\mathbf{x}},\hat{\mathbf{x}})\mathbf{V}_a^{\text{new}} \tag{11}$$

$$\Rightarrow \big(I - \beta\big(\frac{\mathbf{f}_a + \mathbf{f}_c\mathbf{dc}}{-\mathbf{f}_{a'}}\big)\mathbf{A}(\tilde{\mathbf{h}},\hat{\mathbf{x}})\mathbf{E}(\hat{\mathbf{x}},\hat{\mathbf{x}})\big)\mathbf{V}_a^{\text{new}} = u'(\mathbf{c})\mathbf{dc} \tag{12}$$

where $\mathbf{dc}$ is a matrix with diagonal entries corresponding to the derivative of $c^*(a)$ at each point in $\hat{\mathbf{x}}$ and zero elsewhere. The matrices $\mathbf{f}_a$, $\mathbf{f}_c$, and $\mathbf{f}_{a'}$ are the same but for partial derivatives of the budget constraint.

Equation (12) is the HR updating equation. Given a (vector of values of a) policy function $\mathbf{c}$, this equation can be used to obtain an updated guess of $\mathbf{V}_a$. This amounts to solving an N-by-N system of linear equations where N is the number of gridpoints in $\hat{\mathbf{x}}$. In general, this is a fairly expensive operation (relative to simple operations such as matrix-vector multiplication), but the reduction in the

[10]Here I skip over the technical detail that the policy functions are not necessarily everywhere differentiable. While true in a technical sense, Rendahl (2022) provides good discussion into why this is not a practical issue. The essence is that the envelope condition can be used to show that any errors arising from treating these functions as differentiable everywhere disappear once convergence is achieved.

number of iterations dominates the resulting increase in time-per-iteration, often by an overwhelming amount. Additionally, there are ways to approximate the solution to (12) that avoid directly solving the system, which can often be used to accelerate computation substantially (discussed below).

3.1. More Implementation Details

As discussed above, solving the linear system in (12) for $\mathbf{V}_a^{\text{new}}$ directly is somewhat slow as typical approaches involve finding the LU-factorization of the LHS matrix. To accelerate this, we can leverage both the fact that we have a previous guess for the value function $\mathbf{V}_a$ and the approximate sparsity of $\mathbf{A}(\tilde{\mathbf{h}}, \hat{\mathbf{x}})\mathbf{E}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$.

Previous Guess: When the number of gridpoints is large, it is often much faster to approximate the solution to linear systems such as (12) rather than solving them directly. One such approximation exploits the power series $(\mathbf{I} - \mathbf{B})^{-1} = \mathbf{I} + \mathbf{B} + \mathbf{B}^2 + \ldots$ and computes an approximate solution using the first T terms of the RHS of this equation. Although this approach is suboptimal in general and dominated by algorithms such as the Generalized Minimal Residual Method (GMRES), it is convenient here as it allows us to leverage the fact that we possess a previous guess for the value function that we expect to be reasonably close to the desired solution $\mathbf{V}_a^{\text{new}}$.

Taking the power series formula, making a recursive substitution after the first T terms, and using the approximation $\mathbf{V}_a^{\text{new}} \approx \mathbf{V}_a^{\text{old}}$ on the RHS yields

$$\mathbf{V}_a^{\text{new}} \approx u'(\mathbf{c})\mathbf{dc} + \mathbf{B}u'(\mathbf{c})\mathbf{dc} + \ldots + \mathbf{B}^T u'(\mathbf{c})\mathbf{dc} + \mathbf{B}^{T+1}\mathbf{V}_a^{\text{old}} \tag{13}$$

where $\mathbf{B}$ is a placeholder for $\beta\left(\frac{\mathbf{f}_a + \mathbf{f}_c \mathbf{dc}}{-\mathbf{f}_{a'}}\right)\mathbf{A}(\tilde{\mathbf{h}}, \hat{\mathbf{x}})\mathbf{E}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$. Computing this approximation recursively requires only matrix-by-vector multiplication and entirely sidesteps the need for costly factorization or even matrix-by-matrix multiplication.

An alternative view of this approximation formula is that the LHS is the updated guess of the value function that would be achieved if the HR-updating rule were derived using a finite updating horizon $T < \infty$ by applying equation (11) to $\mathbf{V}_a^{\text{old}}$, repeated T times. While this gives us confidence that this approximation will not prevent convergence, it does suggest that choosing a value for T this is too low may serve to undo the benefits of HR acceleration as weight is shifted away from the policy function and back towards the previous guess of the value function. In practice, however, even values of T as small as 15 appear to achieve a close to identical ($+/-$ one iteration) improvement in performance. Thus this "finite-horizon" approach that avoids inverting a linear system is often superior.[11]

Approximate Sparsity: In the case of efficient Smolyak polynomials, the matrices $\mathbf{A}(\tilde{\mathbf{h}}, \hat{\mathbf{x}})$ and $\mathbf{E}(\hat{\mathbf{x}}, \hat{\mathbf{x}})$ are not sparse in general. In fact, the opposite is true and these matrices are entirely dense (i.e. possess no zero values) by construction. When the dimension of the problem is high, however, they do contain many entries that are very close to zero and thus are "approximately sparse". This can be exploited to accelerate either the inversion in (12) or the multiplication in (13) at the cost of some accuracy.

The simplest approach is is to "sparsify" these matrices by replacing all of their elements with absolute value less than some threshold by zero. Then the inversion (if applying 12) or multiplication (if applying 13) can proceed with this new sparse matrices. This approach merits some caution, however, as sparse grid algorithms only represent speed gains relative to dense grid algorithms if the matrices used achieve a sufficient level of sparsity which is often not true in the case of Smolyak polynomials, even for medium dimension problems.[12]

3.2. Comparisons to Other Methods

How does this variant of the SEGM compare to other similar methods for solving Bellman equations? Continuous-time methods (Ahn et al. 2018) and the discrete-time Howard method developed in Rendahl (2022) both obtain updated guesses for the value function (or its derivative) by solving a similar system of linear equations, making them the natural comparisons. Relative to these methods, the SEGM is qualitatively different in two ways.

First, both other methods approximate the value function via dense grid linear interpolation. A substantial advantage of this approach is that it leads to sparse linear systems (i.e. those with many zeros). This structure can be exploited to substantially reduce the computational expense of solving the linear system in order to update the value function guess (although this advantage begins to shrink in more complicated models that start to reduce the sparsity of the system such as

[11]I refer to this as "finite-horizon HR-acceleration" due to this intuition, but it is worth noting that this is equivalent to the "modified policy function iteration" of Phelan & Eslami (2022).

[12]This represents the major weakness of Smolyak approaches vs linear basis function approaches, as the latter generate substantially sparser matrices and, consequently, the updating step can be performed much faster. Still, in the intended applications of this paper (problems with state-spaces spanning 5-10 dimensions), the benefits of the Smolyak approach in the other steps (i.e. the speed at which $\mathbf{A}(\tilde{\mathbf{h}}, \hat{\mathbf{x}})$ itself can be computed) outweigh this cost. The development of a sparse interpolation algorithm in which $\mathbf{A}$ is both quick-to-compute and highly sparse would be a massive benefit to the method overall.

multiple discrete states). In contrast, the SEGM requires solving systems that are (in general) non-sparse.

The advantage of the SEGM, of course, is that the value function must be updated at much fewer gridpoints and, as a result, the system of equations to be solved is much smaller than the other methods, even if it is non-sparse. Thus the extent to which the SEGM reduces computation time depends on the extent to which this reduction in the number of gridpoints outweighs the cost of solving the non-sparse system. Fortunately, the reduction is gridpoints is quite large even for modest numbers of states ensuring that the SEGM represents substantial reductions in computation time (even ignoring concerns about memory). Judd et al. (2014) and Brumm & Scheidegger (2017) provide a more in-depth explanation, but as a simple example, a sparse Smolyak grid approximation for a function of 3 variables at depth 6 (65 nodes along the densest axes) requires on the order of 10^{-3} times fewer gridpoints that the equivalent (65 node) dense grid, and this falls dramatically at the number of state variables increases (10^{-6} for 5, 10^{-9} for 7).

4. A Simple Example

To demonstrate the method and provide some basic benchmarks that give a sense for the overall runtime of the method as well as the relative runtime of the various steps, I use a simple model in the style of Aiyagari (1994). The only change is that I enrich the income process, which in the baseline model follows a basic AR(1) structure, to introduce additional state variables.

The simplest way to do this is to add additional processes so that income depends on the sum of multiple stochastic variables. That is, an individuals period income y_t is given by

$$\begin{aligned} y_t &= e^{z_{1,t}+\ldots+z_{N,t}} \\ \mathbf{z}_{t+1} &= \mathbf{P}\mathbf{z}_t + \epsilon_t \\ \epsilon_t &\sim N(\mathbf{0}, \mathbf{\Sigma}) \end{aligned} \tag{14}$$

where $\mathbf{z}_t$ is the vector $(z_{1,t}, \ldots, z_{N,t})$ and $\mathbf{P}$ is a diagonal matrix of persistence parameters.

For the purpose of this paper, this extension does not have much economic meaning. Increasing N to be larger than unity serves as a straightforward way to purely increase the dimension of the state-space, and thus the benefits of leveraging sparse grids, without introducing any additional complications that would cloud comparisons (either between methods at a given dimension or between different dimensionalities for a given method). It is not hard to imagine, however, the economic relevance of richer, higher-dimension income process in general. For example, the income process could be one of the many estimated in Guvenen, Karahan, Ozkan & Song (2021).

Other than this change, to rest of the model remains standard. Households received CRRA utility from consumption (satisfying the Inada conditions and ensuring that we do not need to worry non-convexity or non-monotonicity), save in a risk free asset a that pays an exogenous rate of return r,[13] and face a standard budget constraint. The Bellman equation of the household is then given by

$$\begin{aligned} V(a, z_1, \ldots, z_N) = \max_{c,a'} & \frac{c^{1-\sigma}}{1-\sigma} + \beta \mathbb{E}[V(a', z_1', \ldots, z_N') | z_1, \ldots, z_N] \\ s.t.\ \ c + a' &= (1+r)a + y \\ y &= e^{z_1 + \ldots + z_N} \\ a' &\geq 0 \\ \mathbf{z}' & \text{ is distributed as in (14)} \end{aligned} \tag{15}$$

The parameters of the problem are taken to be typical and are fully described in Appendix Table A.1. The exact values of the parameters are unimportant (as long as $\frac{1}{1+r} < \beta$ to ensure that the value function is finite and well-defined). The only thing worth noting is that the various elements of $\mathbf{z}$ are assumed to be identical and independent. That is, the matrices $\mathbf{P}$ and $\mathbf{\Sigma}$ are diagonal, and their elements along the diagonal do not vary. Again, this is uninteresting but merely serves to allow the dimension of the problem to increase with N.

[13]Because r is taken to be exogenous, this is a so-called partial equilibrium problem and does not focus on computing an equilibrium interest rate. Structuring the problem this way means that the only computational task is the solution of the household's problem which is natural as this is the focus of the SEGM. Although the SEGM approach, in particular the HR equations in Section 3 that leverage sparse interpolation, does suggest some applications towards the computation of steady-state distributions (and thus applications towards general equilibrium calculations), it is unclear at the time of writing whether these approaches should be included in an additional section of this paper or are complex enough to warrant their own paper. Thus I opt to sidestep the issue (for now) and focus only on solving the household problem. This is further justified by the fact that approximating the steady-state distribution via Monte Carlo simulation often dominates direct solution of the distribution via policy functions and transition probabilities in high-dimensional problems as the cost of the former does not depend on the number of state variables.

4.1. Solution Times

With the example model specified, we can now evaluate the performance of the SEGM, and the various tricks/implementation optimizations laid out in this paper, under different levels of dimensionality.

To that end, I solve the model using multiple approaches. As a baseline, I solve the model using the endogenous grid method and dense grid linear interpolation. I then resolve the model introducing the components of the SEGM one at a time in order to get a sense of the benefits (and costs) of each. This correspond to solving the model using the SEGM approach implemented with dense grids and linear interpolation, then moving to the SEGM implemented via efficient Smolyak grids, and, finally, the SEGM with HR acceleration. After this, I then fiddle with the optimizations described at the end of Section 3 (finite-horizon HR acceleration and approximate sparsity) to see how much more performance can be eked out.

Across all of these solution approaches, I keep the other aspects of the computational problem as similar as possible. The Smolyak grid is chosen to have depth 7 along the dimension of assets and depth 5 in the remaining dimensions, leveraging the anisotropic flexibility of these grids. The closest equivalent dense grid is one with 129 ($2^7 + 1$) gridpoints in the asset dimension and 33 ($2^5 + 1$) gridpoints in the remaining dimensions,[14] as this is the number of gridpoints that the Smolyak grid maintains along its "densest" axes. Relative to the dense grid, the Smolyak grid is slightly rescaled (to align with Chevyshev polynomials) and drops points deemed unimportant by the method (of course this potentially carries a drop in accuracy, which must be assessed later). Other optimizations, such as the pre-computation of expectations via sparse Gauss-Hermite quadrature (so that integration can be performed via a simple matrix-vector multiplication) and pre-allocation of memory are also made equivalent where possible.

Table 1 displays the time required to compute the solution (up to a tolerance of 10^{-6}) for all five methods (rows) and the cases of $N \in \{2, 3, 4\}$ (columns). Column (1) corresponds to the case where $N = 2$ — the income process is two-dimensional which, when combined with the asset, means there are three state variables in total and the dimension of the problem is only slightly larger than a textbook model. The baseline EGM is able to solve this problem in 4.7 seconds. Shifting to the SEGM with the dense grid increases this to 10.7 seconds (the reasons for this increase will

[14]These levels of approximation are somewhat low but are necessary to make the dense grid approach tractable in higher dimensions for the purpose of comparison.

Table 1: Time-to-Solve using Various Approaches

	(1)	(2)	(3)
	$N = 2$	$N = 3$	$N = 4$
Method:			
Baseline EGM	4.7 s	2.9 m	316 m
SEGM w/ dense grid	10.7 s	7.6 m	1146 m
Baseline SEGM	10.4 s	1.7 m	10 m
SEGM w/ HR acceleration	7.6 s	†	†
SEGM w/ finite-horizon HR	3.7 s	0.8 m	6.5 m
Precomputation:			
Sparse grids	4.0 s	1.1 m	14 m
Dense grids	796 s	††	††
Ratio of sparse-to-dense gridpoints	$\approx 10^{-2}$	$\approx 10^{-3}$	$\approx 10^{-4}$

Note: This table displays the total computation time for the various methods discussed throughout the paper denoted in either seconds (column 1) or minutes (column 2 and column 3). Computations are conducted on an AMD Ryzen 5 5600x processor using Julia via a single-threaded implementation. All approaches iterate until a unit-free (i.e. log-difference) error of less than 10^{-6} is achieved between iterations. †: These computations fail to achieve convergence. See text for further discussion. ††: Pre-computing expectations for dense grids in a manner equivalent to that employed for the sparse grids quickly becomes prohibitively expensive as N increases. Thus for the dense-grid implementations in these columns, expectations are ignored.

be further discussed below).

The main result of column (1) is seen by comparing the dense-grid SEGM to the baseline SEGM. Here, despite the sparse grid possessing a dramatically lower number of gridpoints (on the order of 10^{-2}, switching to the sparse grid barely reduces computation time (from 10.7 to 10.4 seconds). This is a result of the fact that performing sparse interpolation is a dramatically more expensive operation than performing linear interpolation (which is a simple, highly-optimizable operation in this setting). In three dimensions, the reduction in gridpoints is not large enough to overcome this effect — the two channels are roughly of the same magnitude and thus lead to no net change in computation time. Implementing HR-acceleration can further improve the computation time of the SEGM, but the same could be said of the dense-grid approaches, so this point is moot. Overall (and consistent with intuition), the gains from sparse grids in low-dimension problems are small or even negative.

This conclusion, however, changes rapidly as the dimension of the problem increases. In column (2), corresponding to 4 state variables, the baseline implementation of the SEGM is now substantially faster than the baseline EGM (1.7 minutes vs 2.9 minutes) and can be further improved to 0.8 minutes via HR-acceleration (although only in the finite-horizon implementation which will be discussed later). Although the relative increase in computation time required for sparse interpolation does not change much from the $N = 2$ case,[15] the relative number of gridpoints required falls to 10^{-3}, shifting things in favor of the SEGM.

The advantage becomes overwhelming even at $N = 4$ which most would consider a "medium-dimension" problem. Here the baseline dense-grid EGM requires roughly 5 hours (316 minutes) to achieve a solution while the SEGM can acheive this in 10 minutes or 6.5 minutes if HR-acceleration is applied. The overall speed-up is roughly 50x (316/6.5) indicating that the SEGM performs dramatically better in these problems than the naive method. Of course, it may be more reasonable in higher dimensions to instead compare the SEGM to a sparse grid method that utilitizes numerical optimization (i.e. compare on the "EGM" component of the method rather than the "S" component). Even at $N = 4$, performing sparse-grid interpolation makes up the majority of the computational cost. Thus the gain relative to a method utilizing numerical optimization can be well-approximated by comparing the number of interpolations required per gridpoint. The SEGM requires only one interpolation so relative to a method that requires 10 (e.g. a Newton's-method-based approach), we expect the SEGM to be roughly 10 times faster.

Discussion of HR acceleration: The second major lesson, beyond the speed of the SEGM, from Table 1 is that the finite-horizon implementation of HR-acceleration dominates the infinite-horizon approach the relies on matrix inversion both in speed and stability. The fact that the finite-horizon approach is faster echoes the results of Santos & Rust (2004) as well as Phelan & Eslami (2022) and arises from the fact that the matrix-vector multiplication requires computations on the order of n^2 (for an n-by$n$ matrix) while matrix inversion (more precisely: matrix factor-

[15]Roughly speaking. The interpolation time does increase somewhat substantially with the number of dimensions. But in relative terms this is dwarfed by the magnitude of the change in the relative number of gridpoints.

ization) requires n^3. These values can change with sparsity, but unless the sparsity satisfies particular conditions, the ordering itself does not change.

More surprising is the fact that the infinite-horizon HR approach does not converge at all for $N \in \{3, 4\}$ despite the fact that the baseline SEGM converges without issue. While it is difficult to determine the precise reason for this, the core reason for this appears to be the fact that the infinite-horizon approach "over-fits" to errors in the policy function. These errors can arise from interpolation error in $\mathbb{E}[V_a(\tilde{h}, y')|y]$ or interpolation error in the consumption function (which is used to approximate the derivative of $c^*(a)$ via finite-differences). By applying maximal weight to the policy function (which is impacted by these errors) and no weight to the previous guess of the value function (which is not), the infinite-horizon approach fails to purge these errors over the course of multiple iterations.

This interpretation of the convergence failure is supported by two numerical experiments. First, the convergence of the infinite-horizon approach seems to depend heavily on the depth of the Smolyak grid used. If the grid is made too shallow, convergence will fail even at low dimensions ($N = 2$), supporting the interpretation that interpolation error is to blame. Second, the finite-horizon approach will eventually fail to converge as well if the horizon is set too high, lending support to the fact that the high weight placed on the policy function by the infinite-horizon approach is to blame. Still, this discussion is largely academic as the finite-horizon approach, which does not appear to suffer from these issues at useful horizons lengths, dominates the infinite-horizon approach anyways.

Number of Iterations for Various Approaches: Table 2 decomposes the run-time of each approach into the number of iterations required to achieve convergence and the time required to complete a single iteration so that (# of iterations) $*$ (Time per iteration) $\approx$ Total run-time. There are two main lessons to take from this table.

The first is that the key step of the EGM, using the envelope condition to construct a guess for the true policy function that is then treated as correct, does not appear to impede convergence at all. Comparing the number of iterations required for convergence between the baseline EGM and the SEGM implemented with dense grids, the number of iterations required for convergence are more or less identical between the two (and even somewhat lower for the SEGM). This is somewhat surprising, as one may expect the additional error introduced by this approximation to lead to additional iterations. In practice, however, this appears not to be a concern and the this aspect of the SEGM can be implemented essentially "for free", which may be useful even in situations where sparse grids are not optimal (i.e. situations with two endogenous variables where a traditional EGM approach would lead to a highly irregular grid and require a complex interpolation approach).

The baseline SEGM, however, does require substantially more iterations than either of the other (non-HR-approaches), though not too many more. The fact that these extra iterations are not required when using the SEGM with a dense grid suggests that this is a function of sparse grid interpolation. Intuitively, as the dimension of the problem increases and the number of points dropped by the sparse grid (relative to the dense grid) increases, the accuracy of the interpolation of the continuation value decreases. Overcoming this error requires additional iterations.

The second takeaway is that the cost, in terms of the number of additional iterations required, of switching infinite-horizon HR acceleration to finite-horizon is small. I opt to use a horizon of 13 (that is, I choose T in equation (13) to be 13) so that the updated value function corresponds to the value of following the current guess of the policy function for 13 periods plus the continuation value from the previous iteration. This value is chosen to (roughly) minimizes the run-time of the approach. Even with this short horizon, the finite-horizon approach requires only 5 more iterations than the infinite-horizon approach (about a 25 percent increase) when $N = 2$. Making comparisons in higher dimensions is impossible as the infinite-horizon approach fails to converge.

4.2. Accuracy

[To be written]

5. Conclusion

This paper presents a method aimed at harmonizing endogenous grid approaches to solving economic models (which avoid numerical optimization) with sparse grid approaches (which mitigate the curse of dimensionality). The key tension between these two approaches is that the former leverages flexibility in the precise gridpoints at which the value function is computing while the latter relies on knowing the value function at a carefully chosen set of gridpoints. This tension is resolved by using envelop conditions to construct a guess of the policy function. If this guess were correct, one could use this guess to precise control the set of gridpoints returned by an endogenous grid approach. The key insight is that while this guess is not correct in general, it will be correct after convergence is achieved and treating it as correct (allowing one to leverage this precise control) does not change the fixed point of the iterative procedure.

The resulting method, the Sparse Endogenous Grid Methods (SEGM), is best suited to quickly solving medium-dimension economic models. Comparing the run-time of the approach to a standard endogenous grid approach suggests that there is little advantage gained when dimensionality is low (3 dimensions or less) as the reduction in the number of gridpoints is outweighed by the increased computational cost of sparse grid interpolation (relative to linear interpolation). In higher dimensions, the advantage of the SEGM (relative to dense grid approaches) becomes overwhelming. This advantage can be furthered increased by leveraging the fact that the SEGM naturally lends itself to acceleration via policy-function iteration. Applying this, the SEGM is about 50 times faster than a dense endogenous grid approach for a 5 dimensional problem. In higher dimensions, the gains are even larger.

Beyond allowing the implementation of sparse grids, the method has a natural application to models with multiple endogenous choice variables (i.e. multi-asset models such as HANK models). Even in low dimensions, the presence of multiple choice variables leads to complications for traditional endogenous grid methods due to grid-misalignment, requiring hybrid methods (e.g. Ludwig & Schön 2018) or re-gridding approaches (e.g. Druedahl & Jørgensen 2017). The SEGM avoid these issues completely by providing the value of the value function at precise gridpoints, even if a dense-grid interpolation method is chosen.

The largest computational bottlenecks of the SEGM are the cost of performing interpolation and the cost of the matrix-vector multiplications performed during the finite-horizon policy function updating step. Relative to the implementation presented in this paper, there is room for substantial improvement on these fronts. In particular, an interpolation approach based on linear basis functions, rather than Smolyak polynomials, has the potential to dramatically decrease the computational cost of the updating step (as the vector of interpolated values is a highly sparse linear combination of the gridded function values). Developing (or, more precisely, translating to Julia) and efficient approach to performing sparse linear basis interpolation is somewhat beyond the scope of this paper is left to future work.

Table 2: Time-to-Solve Decomposed into Iterations and Time-per-Iteration

		(1)	(2)	(3)
		$N = 2$	$N = 3$	$N = 4$
Baseline EGM	Total time:	4.7 s	2.9 m	316 m
	Iterations:	267	296	324
	Time-per-it:	18 ms	0.6 s	59 s
SEGM w/ dense grid	Total time:	10.7 s	7.6 m	1146 m
	Iterations:	261	289	317
	Time-per-it:	41 ms	1.6 s	217 s
Baseline SEGM	Total time:	10.4 s	1.7 m	10 m
	Iterations:	292	396	408
	Time-per-it:	36 ms	0.3 s	1.5 s
SEGM w/ HR acceleration	Total time:	7.6 s		
	Iterations:	16		
	Time-per-it:	475 ms		
SEGM w/ finite-horizon HR	Total time:	3.7 s	0.8 m	6.5 m
	Iterations:	21	27	48
	Time-per-it:	176 ms	1.8 s	8.1 s

Note: This table decomposes the timing results of Table 1 into iterations and time-per-iteration in terms of milliseconds (column 1) or seconds (column 2 and column 3). Included in "Total time" is the additional step of memory pre-allocation before iterations begin; however, the contribution of this step is tiny and is thus ignored.

References

Ahn, S., Kaplan, G., Moll, B., Winberry, T. & Wolf, C. (2018), 'When inequality matters for macro and macro matters for inequality', *NBER macroeconomics annual* **32**(1), 1–75.

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

Auclert, A., Bardóczy, B., Rognlie, M. & Straub, L. (2021), 'Using the sequence-space jacobian to solve and estimate heterogeneous-agent models', *Econometrica* **89**(5), 2375–2408.

Brumm, J. & Scheidegger, S. (2017), 'Using adaptive sparse grids to solve high-dimensional dynamic models', *Econometrica* **85**(5), 1575–1612.

Carroll, C. D. (2006), 'The method of endogenous gridpoints for solving dynamic stochastic optimization problems', *Economics letters* **91**(3), 312–320.

Druedahl, J. & Jørgensen, T. H. (2017), 'A general endogenous grid method for multi-dimensional models with non-convexities and constraints', *Journal of Economic Dynamics and Control* **74**, 87–107.

Fella, G. (2014), 'A generalized endogenous grid method for non-smooth and non-concave problems', *Review of Economic Dynamics* **17**(2), 329–344.

Fernandez-Villaverde, J., Nuno, G., Sorg-Langhans, G. & Vogler, M. (2020), 'Solving high-dimensional dynamic programming problems using deep learning', *Unpublished working paper* .

Guvenen, F., Karahan, F., Ozkan, S. & Song, J. (2021), 'What do data on millions of us workers reveal about lifecycle earnings dynamics?', *Econometrica* **89**(5), 2303–2339.

Iskhakov, F., Jørgensen, T. H., Rust, J. & Schjerning, B. (2017), 'The endogenous grid method for discrete-continuous dynamic choice models with (or without) taste shocks', *Quantitative Economics* **8**(2), 317–365.

Judd, K. L., Maliar, L., Maliar, S. & Valero, R. (2014), 'Smolyak method for solving dynamic economic models: Lagrange interpolation, anisotropic grid and adaptive domain', *Journal of Economic Dynamics and Control* **44**, 92–123.

Kahou, M. E., Fernández-Villaverde, J., Perla, J. & Sood, A. (2021), Exploiting symmetry in high-dimensional dynamic programming, Technical report, National Bureau of Economic Research.

Ludwig, A. & Schön, M. (2018), 'Endogenous grids in higher dimensions: Delaunay interpolation and hybrid methods', *Computational Economics* **51**, 463–492.

Maliar, L., Maliar, S. & Winant, P. (2021), 'Deep learning for solving dynamic economic models.', *Journal of Monetary Economics* **122**, 76–101.

Phelan, T. & Eslami, K. (2022), 'Applications of markov chain approximation methods to optimal control problems in economics', *Journal of Economic Dynamics and Control* **143**, 104437.

Rendahl, P. (2022), 'Continuous vs. discrete time: Some computational insights', *Journal of Economic Dynamics and Control* **144**, 104522.

Santos, M. S. & Rust, J. (2004), 'Convergence properties of policy iteration', *SIAM Journal on Control and Optimization* **42**(6), 2094–2115.

The Stockfish developers (see AUTHORS file) (n.d.), 'Stockfish'.
URL: *https://github.com/official-stockfish/Stockfish*

Appendix

A. Additional Tables and Figure

Table A.1: Parameters of the Simple Model

Parameter	Value	Description
β	0.97	Discount Factor
r	0.01	Rate of Return
σ	2	CRRA Parameter
$\mathbf{P}$	$0.5\mathbf{I}$	Persistence of **z**'s
$\mathbf{\Sigma}$	$0.1\mathbf{I}$	Variance of innovations in **z**'s

Note: This table displays the parameters of the simple model used to perform the exercises in Section 4.

B. Proof of Convergence

It is helpful to formally define the S' operator that will be shown to be a contraction. To simplify notation and exposition, here I specialize to the case where the occasionally binding constraint takes the form $a' \geq 0$, but the extension to more general constraints (and the assumptions about those constraints required for convergence to hold) is straightforward.

Definition 1. *Let X be the space of all continuous, bounded, convex functions $(a, y) \to \mathbb{R}$ weakly decreasing in a and $|\cdot|$ be the sup-norm so that $(X, |\cdot|)$ is a metric space. Then $S' : X \to X$ is the operator defined by*

$$\begin{aligned}(S'V_a)(a,y) &= -\frac{f_a}{f_c}u'(c) \qquad (16)\\ s.t.\ \ 0 &= f(c,a,a',y)\\ a' &= \tilde{h}(a,y,V_a(a,y)) \text{ if } 0 < \tilde{h}(a,y,V_a(a,y))\\ u'(c) &= \beta\mathbb{E}[V_a(a',y')|y] \text{ if } 0 < \tilde{h}(a,y,V_a(a,y))\\ 0 &= a' \text{ otherwise}\end{aligned}$$

*We say that S' is a **local contraction** if there exists some $\epsilon > 0$ such that for all W, V*

satisfying

$$W(a,y) = V^*(a,y) + \epsilon f_W(a,y)$$
$$V(a,y) = V^*(a,y) + \epsilon f_V(a,y)$$

for some Lipschitz-continuous f_W, f_V with Lipschitz constant K (assumed to be equal WLOG), we have that $|S'W - S'V| \leq \delta |W - V|$ for some $\delta \in (0,1)$.

The simplest approach to showing the local contractivity of S' is to apply Blackwell's sufficient conditions to W, V that satisfying the definition of locality.

Monotonicity: Suppose that $W_a(a,y) \geq V_a(a,y)$ for all (a,y). We wish to show that $(S'W_a)(a,y) \geq (S'V_a)(a,y)$ for all (a,y) and proceed by cases.

Main Case: Both $W_a(a,y)$ and $V_a(a,y)$ yield $0 < a'$ when applying S'. We wish to show that $-\frac{f_a}{f_c}\beta\mathbb{E}[W_a(\tilde{h}(a,y,W_a(a,y)),y')|y] \geq -\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)),y')|y]$. To save on notation define constants and suppress dependence on y so that

$$-\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)),y')|y] \equiv R\beta\mathbb{E}V(a,\tilde{h}(a,V(a))) \tag{17}$$

where R is a shorthand for $-\frac{f_a}{f_c}$. In a typical ABH model, this would be the total return on assets ($1+r$) divided by the price of consumption (1).

Consider the function$f(\lambda)$ defined by

$$\lambda R\beta\mathbb{E}W(a,\tilde{h}(a,\lambda W_a(a)+(1-\lambda)V(a)))+(1-\lambda)R\beta\mathbb{E}V(a,\tilde{h}(a,\lambda W(a)+(1-\lambda)V(a))) \tag{18}$$

Our approach will be to show that $f'(x) > 0 \forall x \in (0,1)$, and then the fundamental theorem of calculus will guaranteed that $f(1) - f(0) > 0$ which is what we desire to show. Differentiating wrt λ and grouping terms yields (note here that WLOG I divide the entire expression by $R\beta$ to simplify as we only care about the sign of the expression)

$$\begin{aligned}\big(\mathbb{E}W(\cdot) - \mathbb{E}V(\cdot)\big) + \big(W(a) - V(a)\big)\big(\lambda\mathbb{E}W'(\cdot)\tilde{h}_2(a,\lambda W(a) + (1-\lambda)V(a))\big)+ \\ +\big((1-\lambda)\mathbb{E}V'(\cdot)\tilde{h}_2(a,\lambda W(a)+(1-\lambda)V(a))\big)\end{aligned} \tag{19}$$

Note that the first-order and envelope conditions of the problem imply that $V^*(a) = R\beta\mathbb{E}V^*(\cdot)$. This would be useful to leverage to further group terms, but it is not true in general for W, V. Instead, we leverage the fact that the assumption that W, V are close to V^* lets us write

$$W(a) = R\beta \mathbb{E} W(\cdot) + g(\epsilon, K) \tag{20}$$

for some g satisfying $\lim_{\epsilon\to 0} g(\epsilon, K) = 0, \forall K$ (and the same for V).The statement is offered here without proof and is straightforward (though tedious) to verify under the regularity conditions laid out in the problem statement. Making this substitution, grouping terms, and absorbing some unimportant terms in g (as well as combining g's) yields

$$\big(\mathbb{E}W(\cdot) - \mathbb{E}V(\cdot)\big)\big(1 + R\beta \mathbb{E}U(\cdot)\tilde{h}_2(a, U(a)))\big) + g(\epsilon, K) \tag{21}$$

where $U = \lambda W + (1 - \lambda)V$.

Our next step is to consider the definition of the policy function c^* (note that I have written the value function in terms of an integral, to keep consistent with the notation used so far)

$$\int_0^a V^*(x)dx = u(c^*(a)) + \beta \mathbb{E} \int_0^a V^*(\tilde{h}(a, V^*(a))) \tag{22}$$

where the fact that V^* is the solution to the bellman equation ensures that $\tilde{h}(a, V^*(a))$ corresponds to the savings policy function. Differentiating this equation with respect to a and making some substitutions via envelope and first-order conditions yields

$$\beta \mathbb{E} V^*(\tilde{h}(a, V^*(a)))\tilde{h}_2(a, V^*(a)) = \big(1 - \frac{c'(a)}{R} - \frac{1}{R}\big) \geq -\frac{1}{R} \tag{23}$$

where the inequality follows from the fact that monotonicity of the savings function ensures that $c'(a) \leq R$ everywhere.

As before, we can leverage the fact that W, V are close to V^* to show that equation (21) can be rewritten as

$$\big(\mathbb{E}W(\cdot) - \mathbb{E}V(\cdot)\big)\big(1 + R\beta \mathbb{E}V^*(\cdot)\tilde{h}_2(a, V^*(a)))\big) + \hat{g}(\epsilon, K) \tag{24}$$

where the difference between V^* and U has been folded into $\hat{g}$.

The term $\big(\mathbb{E}W(\cdot) - \mathbb{E}V(\cdot)\big)$ is positive by assumption. Leveraging equation (23) shows that the term $\big(1 + R\beta \mathbb{E}V^*(\cdot)\tilde{h}_2(a, V^*(a)))\big)$ is also strictly positive as $\beta < 1$. Thus there is some $\epsilon > 0$ for which (24) is strictily positive, completing the proof of (local) monotonicity.

Remaining Cases: The remaining cases are (fortunately) much more straightforward.

- Case 2: Both $W_a(a,y)$ and $V_a(a,y)$ yield $0 = a'$ when applying S'. Then we have
$$(S'W_a)(a,y) = -\frac{f_a}{f_c}u'(c^*) = (S'V_a)(a,y)$$
where c^* is the solution to $0 = f(c,a,0,y)$ given (a,y).

- Case 3: $W_a(a,y)$ yields $0 < a'$ and $V_a(a,y)$ yields $0 = a'$ when applying S'. Then we have
$$\begin{aligned}(S'W_a)(a,y) &= -\frac{f_a}{f_c}\beta\mathbb{E}[W_a(a',y')|y] \\ &= -\frac{f_a}{f_c}u'(u'^{-1}(\beta\mathbb{E}[W_a(a',y')|y]) \\ &\geq -\frac{f_a}{f_c}u'(c^*) \\ &= (S'V_a)(a,y)\end{aligned}$$
where the inequality arises from the fact that $\frac{f_c}{f_{a'}} > 0 \Rightarrow c^* \geq u'^{-1}(\beta\mathbb{E}[V_a(a',y')|y])$ as c^* satisfies $f(c,a,0,y)$ while $u'^{-1}(\beta\mathbb{E}[W_a(a',y')|y])$ satisfies $f(c,a,a',y)$ for some $a' > 0$.

The potential fourth case where $W_a(a,y)$ yields $0 = a'$ and $V_a(a,y)$ yields $0 < a'$ is not possible due to the fact that $\tilde{h}$ is decreasing in its third argument. Thus all the cases are exhausted and we have shown monotonicity.

Discounting: We wish to show that there is some $\gamma \in (0,1)$ such that $\big(S'(V_a + k)\big)(a,y) \leq (S'V_a)(a,y) + \gamma k$ for all $\big(V_a, k, (a,y)\big)$. In our case, γ will be $\big(-\frac{f_a}{f_c}\beta\big)$ (which, recall, falls between 0 and 1 by assumption). As with monotonicity, we proceed by cases.

- Case 1: Both $(V_a + k)(a,y)$ and $V_a(a,y)$ yield $0 < a'$ when applying S'. Then we have

$$\begin{aligned}\big(S'(V_a+k)\big)(a,y) &= -\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)),y')+k|y] \\ &= -\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)),y')|y]+\big(-\frac{f_a}{f_c}\beta\big)k \\ &= (S'V_a)(a,y)+\big(-\frac{f_a}{f_c}\beta\big)k\end{aligned}$$

- Case 2: Both $(V_a+k)(a,y)$ and $V_a(a,y)$ yield $0=a'$ when applying S'. Then we have

$$\big(S'(V_a+k)\big)(a,y) = -\frac{f_a}{f_c}u'(c^*) = (S'V_a)(a,y) \leq (S'V_a)(a,y)+\big(-\frac{f_a}{f_c}\beta\big)k$$

- Case 3: $(V_a+k)(a,y)$ yields $0<a'$ and $V_a(a,y)$ yields $0=a'$ when applying S'. Then we have

$$\begin{aligned}\big(S'(V_a+k)\big)(a,y) &= -\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)+k),y')+k|y] \\ &= -\frac{f_a}{f_c}\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)+k),y')|y]+\big(-\frac{f_a}{f_c}\beta\big)k \\ &= -\frac{f_a}{f_c}u'(u'^{-1}(\beta\mathbb{E}[V_a(\tilde{h}(a,y,V_a(a,y)),y')|y]))+\big(-\frac{f_a}{f_c}\beta\big)k \\ &\leq -\frac{f_a}{f_c}u'(c^*)+\big(-\frac{f_a}{f_c}\beta\big)k \\ &= (S'V_a)(a,y)+\big(-\frac{f_a}{f_c}\beta\big)k\end{aligned}$$