Endogenous Gridpoints in Multiple Dimensions:
Interpolation on Non-Linear Grids

By

Matthew N. White

WORKING PAPER SERIES
The views expressed in the Working Paper Series are those of the author(s) and do not necessarily reflect those of the Department of Economics or of the University of Delaware. Working Papers have not undergone any formal review and approval and are circulated for discussion purposes only and should not be quoted without permission. Your comments and suggestions are welcome and should be directed to the corresponding author. Copyright belongs to the author(s).
Endogenous Gridpoints in Multiple Dimensions: Interpolation on Non-Linear Grids

Matthew N. White†

December 9, 2014

Abstract

In dynamic optimization problems with multiple continuous state variables and multiple continuous controls, the method of endogenous gridpoints (ENDG) generates an irregular collection of gridpoints for which standard interpolation techniques do not apply, while alternative interpolation methods are extremely slow. This paper presents an interpolation technique that allows ENDG to be used in multi-dimensional problems in an intuitive and computationally efficient way. The method translates irregular grid sectors onto the unit square (unit cube, unit hypercube, etc) and then applies standard linear interpolation. This method’s superiority to traditional solution approaches, in terms of speed and accuracy, is demonstrated on a benchmark model. At commonly used grid densities, the method of endogenous gridpoints with non-linear grid interpolation is 7.7 times faster than the traditional solution method, with slightly greater accuracy. This computational acceleration erodes only very slowly as grid density increases, unlike with alternative interpolation methods.

JEL Classification: C61, C63, E21

Keywords: Dynamic models, numerical solution, endogenous gridpoint method, non-linear grid interpolation, endogenous human capital

† Univ of Delaware, Dept of Economics (315A Purnell Hall, Newark DE 19711); mnwecon@udel.edu
1 Introduction

The method of endogenous gridpoints as originally developed by Carroll (2006) greatly speeds up solution to dynamic optimization problems with a single continuous control variable and a one-dimensional state space. While the method was subsequently extended several times (see discussion below), the inclusion of multiple endogenous state dimensions with multiple choice variables has proved problematic. As the policy functions are non-linear, the method of endogenous gridpoints produces an irregular array of pre-decision state space points, rather than an ordinary rectilinear grid that can be used with standard interpolation methods to generate an approximation to the value and policy functions spanning the state space. Existing methods for interpolating on an arbitrary collection of points are computationally burdensome and are difficult to extend beyond two dimensions. This paper presents an alternate method for interpolating on an irregular array of points that corrects for these deficiencies by exploiting known relationships among the points, allowing the method of endogenous gridpoints to be efficiently used in a much larger class of dynamic problems.

Consider a generic dynamic optimization problem in which an agent must decide, given his current state, what action to take among a continuous set of choices; that action will yield both a present flow of utility and affect the distribution of states he might arrive at next period. Canonically, this might be an individual deciding how much to consume today and how much to reserve as savings, subject to uncertainty in his future income. The traditional solution method for such a problem is to choose a set of gridpoints in the state space, solve for the optimal action at each of these gridpoints, and then generate an approximation to the true policy function by interpolating between the gridpoints. Finding the optimal action at one state space point might be done by explicit maximization of expected utility or by searching for a solution to the first order condition(s). In either case, the traditional solution method requires checking multiple guesses of the action to evaluate their optimality—because the action affects both current utility and the future state (i.e. future expected utility), the optimal decision at a given state space point cannot be solved explicitly. Examining each of these guesses involves numeric integration of the possible future states the agent might arrive in next period after taking the hypothetical action; this integral requires many evaluations of the subsequent period’s policy or value function, themselves interpolations generated earlier. Across many state space points and a large number of time periods, repeatedly evaluating numeric integrals to find the optimal action is extremely computationally burdensome.\(^1\)

The method of endogenous gridpoints significantly cuts down on the number of nu-

\(^1\)With heterogeneous agents, a general equilibrium framework, or in an estimation setting, the problem must be solved several thousand times, if not more.
meric integrations executed in a dynamic optimization problem: rather than taking multiple
guesses of the optimal action at each gridpoint, the method of endogenous gridpoints suc-
ceeds on its first guess, requiring only a single integral. It accomplishes this by inverting the
maximization problem: rather than asking what the agent optimally should do now if he
begins the period in state $x$, the method of endogenous gridpoints asks what an agent ending
the period in state $z$ must have just done to find himself there, assuming he acted optimally
to arrive at $z$. That is, it fixes a grid of “post-decision states”, calculates the expectation
of future marginal value conditional on ending the period in this way, and then solves the
first order condition for optimal behavior – the action whose marginal utility matches the
expected marginal value of the future in some way. Subject to conditions on the nature of
risk and state transitions in the problem, we can find the “pre-decision state” $x$ from which
that optimal action was taken to arrive at post-decision state $z$. The pre-decision state space
points are found endogenously as part of the solution of the model, rather than exogenously
fixed in advance. In this way, interpolated approximations to the policy and value functions
can be constructed without repeated guesses, minimizing the numeric integrals needed to
solve the problem.

While the original method of endogenous gridpoints (ENDG) described the solution to a
problem with a single dimensional state space and single continuous choice variable, it has
since been extended multiple times: Barillas and Fernández-Villaverde (2007) describe how
to accommodate multiple choice variables, Fella (2014) allows non-concave value functions,
Hintermaier and Koeniger (2010) adjust the method for occasionally binding constraints,
while Iskhakov, Rust, and Schjerning (2012) develop a method that allows one continuous
and one discrete choice variable. However, the problem of extending ENDG into models with
multiple continuous state and choice variables has proven problematic. As the interpolation
gridpoints are found by inverting (almost surely) non-linear policy functions from a grid of
post-decision states, the resulting collection of pre-decision state points is not an ordinary
mesh, so standard interpolation technique using a rectilinear grid (see section 3.1) does
not apply and an alternative must be found. Very recently, Ludwig and Schön (2014)
have shown that Delaunay interpolation can be used to interpolate the irregular array of
gridpoints generated by a problem with two continuous, endogenous state dimensions and
two choice variables. The computational gains from ENDG, however, are partly eroded by
the additional work necessary to construct the Delaunay triangulation in each period. This
is particularly a problem when the number of gridpoints is large, as the construction step
is $O(N \log N)$ and thus computational time grows more than linearly. As $N$ becomes very
large, ENDG is actually slower than the traditional “exogenous gridpoints” approach, with
a long triangulation construction step overwhelming faster behavioral search.
The interpolation method presented in this paper was chosen specifically for working with the irregular array of gridpoints generated by ENDG so as to minimize computational burden. While they do not form a rectilinear grid, the endogenous pre-decision gridpoints are ordered identically to the exogenous post-decision gridpoints that generated them (subject to regularity conditions met by most models). That is, the econometrician knows the location of each pre-decision gridpoint relative to the others, as it is preserved from the relationships of the post-decision gridpoints. The pre-decision gridpoints thus subdivide a (two dimensional) state space into a grid of irregular quadrilateral sectors, each of which can be continuously mapped to the unit square to allow standard bilinear interpolation.

This method achieves the speed gains of ENDG from obviating iterative searches for optimal behavior (and their accompanying numeric integrals) while avoiding the long construction or evaluation procedures used in other interpolation methods for irregular arrays (see appendix A). Moreover, this method extends to three or more dimensions with only slight modifications and no change in the order of computational difficulty or the storage needed (see appendix B).

The rest of the paper proceeds as follows. Section 2 specifies a benchmark problem in a two-dimensional state space, along with its solution using the traditional method (EXOG) and ENDG. Section 3 presents a method for efficiently interpolating on the irregular or non-linear grids generated by the method of endogenous gridpoints. Section 4 compares the solution times and accuracy of EXOG using linear interpolation on a standard rectilinear grid versus ENDG using interpolation on non-linear grids, and section 5 concludes.

2 Benchmark Model

The benchmark model concerns an individual who must allocate his resources each period among consumption, investment in health (or human) capital, and savings. The individual derives a flow of utility from consumption, while both his income and survival probability depend on the level of health capital; unspent resources can be saved for use in future periods. For ease of reference, the model is a slight extension of the one used by Ludwig and Schöhn (2014), adding additional risk to interperiod transitions.3

2 In \( K \geq 3 \) dimensional state spaces, the sectors are defined by \( 2^K \) gridpoints, but are not irregular polyhedrons. Appendix B discusses modifications to the interpolation method in three or more dimensions.

3 The addition of risk both generalizes the model and eliminates the need for the constrained optimization routine used by Ludwig and Schöhn, originally presented in Hintermaier and Koeniger (2010). This maintains focus on non-linear interpolation rather than the complications of dual search routines.


2.1 Statement of Model

The individual is a lifetime expected utility maximizer who will die with certainty at the end of period $t = T$ (with $T = \infty$ possible), and who discounts future utility at a rate of $\beta$. The individual gets a utility flow from a CRRA utility function $u(c_t)$ with coefficient $\rho$:

$$u(c) = \frac{c^{1-\rho}}{1-\rho}, \quad u'(c) = c^{-\rho}. \tag{1}$$

In order to ensure that the value function is strictly positive, we must restrict $\rho < 1$. The individual begins each period $t$ with bank balances $b_t$ and health/human capital $h_t$, and receives income $I_t = w_th_t$. The wage rate $w_t$ is randomly drawn each period: with probability $\varnothing$ the individual is unemployed and gets $w_t = 0$, else it is log normal: $w_t \sim N(\bar{w} / (1-\varnothing), \sigma^2_w)$.\footnote{The possibility of unemployment ensures that the individual will never choose to consume all resources, which could result in having an infinite marginal utility in the next period. To simplify the model, requiring strictly positive saving eliminates the need for an additional constrained optimization routine.}

For convenience of the solution methods, define total money resources as $m_t = b_t + I_t$. The individual must choose his levels of consumption $c_t$ and investment in his health capital $\eta_t$ so that $m_t - c_t - \eta_t \geq 0$. Investment generates additional health capital according to capital production function $f(\eta_t)$ defined by:

$$f(\eta) = (\gamma/\nu)\eta^\nu, \quad f'(\eta) = \gamma\eta^{\nu-1}. \tag{2}$$

To ensure that produced health is never negative, we require $\nu < 1$.

Any unspent resources accumulate at gross interest factor $R$, so that next period’s resources are $m_{t+1} = R(m_t - c_t - \eta_t) + w_{t+1}h_{t+1}$. Health capital depreciates from one period to the next at depreciation rate $\delta_{t+1} \sim U[\bar{\delta} - \sigma_{\delta}, \bar{\delta} + \sigma_{\delta}]$, and so next period’s capital is $h_{t+1} = (h_t + f(\eta_t))(1 - \delta_{t+1})$. The individual thus faces both permanent (via $\delta$) and transitory (via $w$) income risk. Moreover, the individual faces a mortality risk based on his health capital. The probability that the individual survives into the period is given by:

$$s(h_{t+1}) = 1 - \frac{\phi}{1 + h_{t+1}}, \quad s'(h_{t+1}) = \frac{\phi}{(1 + h_{t+1})^2}. \tag{3}$$

If the individual does not survive, he receives no more income, nor can he consume and derive utility; death is modeled by setting $m_{t+1} = h_{t+1} = 0$ (and implicitly $V_{t+1}(0,0) = 0$).

In a companion paper, White (2015) presents a theoretical framework for dynamic optimization problems in continuous state spaces, with conditions under which ENDG can be
used to solve a model. In the terminology presented there, the control space $Y_t = \mathbb{R}^2_+$, with
$y_t = (c_t, \eta_t)$. The state space is $X_t = \mathbb{R}^2_+$, with typical element $x_t = (m_t, h_t)$. The constraint
on action or control is the budget restriction: $\Gamma_t(m_t, h_t) = \{(c, \eta) \in \mathbb{R}^2_+ | m_t - c - \eta \geq 0\}$.
There are three shocks or random variables that occur during interperiod transitions: $w_{t+1}$, $\delta_{t+1}$, and the (implicit) mortality shock $\theta_{t+1} \sim U[0, 1]$; thus the “transition error” is $\epsilon_{t+1} = (w_{t+1}, \delta_{t+1}, \theta_{t+1})$. The transition function can be written as:

$$
x_{t+1} = \Delta(x_t, y_t, \epsilon_{t+1}) = \begin{cases} 
(0, 0) & \text{if } \theta_{t+1} > s((h_t + f(\eta_t))(1 - \delta_{t+1})) \\
((m_t - c_t - \eta_t)R + w_{t+1}h_{t+1}, (h_t + f(\eta_t))(1 - \delta_{t+1})) & \text{else.}
\end{cases}
$$

(4)

2.2 Solving the Model with EXOG

The above model can be solved using the traditional method of exogenous gridpoints (EXOG),
employing Newton’s method to seek optimal consumption and investment at each point in
a pre-determined grid of $(m_t, h_t)$ gridpoints.

Defining $F(\cdot)$ as the joint distribution of the wage and decay rates, the individual’s
problem in any non-terminal\(^6\) period $t$ can be written in Bellman form as:

$$
V_t(m_t, h_t) = \max_{c_t, \eta_t} u(c_t) + \beta \int s(h_{t+1})V_{t+1}(m_{t+1}, h_{t+1})dF(w, \delta)
$$

(5)

s.t. $m_t - c_t - \eta_t \geq 0, \quad m_{t+1} = (m_t - c_t - \eta_t)R + wh_{t+1}, \quad h_{t+1} = (h_t + f(\eta_t))(1 - \delta)$.

This has two first order conditions, with respect to each of the controls $c_t$ and $\eta_t$:

$$
u'(c_t) = \beta R \int s(h_{t+1})V_{t+1}^m(m_{t+1}, h_{t+1})dF(w, \delta),
$$

(6)

$$
f'(\eta_t) \int (1 - \delta) \left[s'(\cdot)V_{t+1}(\cdot) + s(\cdot)\left(wV_{t+1}^m(\cdot) + V_{t+1}^h(\cdot)\right)\right]dF(w, \delta) = R \int s(\cdot)V_{t+1}^m(\cdot)dF(w, \delta).
$$

Whenever the arguments of the value function or its derivatives are suppressed, they would be
$(m_{t+1}, h_{t+1})$; likewise, suppressed argument of the survival probability function is $h_{t+1}$. The
first order conditions say that the marginal utility of consumption must equal the discounted
expected marginal value of wealth in the subsequent period; and that the marginal benefit
of a dollar spent on health investment— respectively through increased survival probability,

\(^6\)In a finite horizon with an absolute terminal period $T$, the solution in that period is trivial. With no future to look forward to, the individual should consume all resources, $c_T = m_T$, and invest nothing in his health, $\eta_T = 0$. The value function in the terminal period is thus simply $V_T(m, h) = u(m)$. The terminal period marginal value functions are $V_T^m(m, h) = u'(m)$ and $V_T^h(m, h) = 0$. 

6
higher future wages, and higher future health capital—must equal the marginal cost in forsaken future wealth.

To solve the problem in period $t$, the standard exogenous gridpoints method has the econometrician specify a mesh of money-health pairs and then search for the corresponding consumption-investment values that satisfy the first order conditions, testing multiple guesses and computing a numeric integral each time. For simplicity (and comparability to the results of Ludwig and Schön), Newton’s method will be used to iteratively converge to optimal behavior.\footnote{The Jacobian of the first order conditions is computed numerically. The search terminates when successive iterations differ by less than $10^{-6}m_t$.}

In each of the two state space dimensions, specify a finite number of gridpoints $J_m \subset \mathbb{R}^+$ and $J_h \subset \mathbb{R}^+$; define a mesh or rectilinear grid as $J = J_m \times J_h$. For each $(m^j_t, h^j_t)$ gridpoint in the mesh, we search for the unique solution $(c^j_t, \eta^j_t)$ to:

$$
\begin{bmatrix}
\beta R \int s(h_{t+1})V^m_{t+1}(m_{t+1}, h_{t+1})dF(w, \delta) - u'(c^j_t) \\
R \int s(h_{t+1})V^m_{t+1}(m_{t+1}, h_{t+1})dF(w, \delta) - f'(\eta^j_t)
\end{bmatrix}
= \begin{bmatrix}
0 \\
0
\end{bmatrix}
$$

After finding $(c^j_t, \eta^j_t)$ for each gridpoint, the policy functions for consumption and investment (respectively $\hat{c}_t(m_t, h_t)$ and $\hat{\eta}_t(m_t, h_t)$) are then generated by interpolating over the $(c^j_t, \eta^j_t)$ gridpoints with the standard bilinear method (see section 3.1). The value function for period $t$ can likewise be generated by computing its level at each of the gridpoints $V_t(m^j_t, h^j_t)$ using (5) and then applying standard bilinear interpolation.

The first order conditions (6) require knowing not just the level of the value function in period $t + 1$, but also its derivatives with respect to money resources $V^m_{t+1}(\cdot)$ and health capital $V^h_{t+1}(\cdot)$. In a non-terminal period, the marginal value function at each gridpoint can be calculated using the envelope conditions of the maximization problem:

$$
V^m_t(m^j_t, h^j_t) = u'(c^j_t), \quad V^h_t(m^j_t, h^j_t) = u'(c^j_t)/f'(\eta^j_t).
$$

As before, these can each be bilinearly interpolated to generate the marginal functions used in the previous period.\footnote{Just as the value function can be “de-curved” before interpolation by applying the inverse utility function, both marginal value functions can be transformed by the inverse marginal utility function.} The solution process is then conducted recursively backwards from $t = T - 1$ until the initial period $t = 0$ is reached. In an infinite horizon variant (where $T = \infty$), the solution process can be recurved until the optimal actions at each gridpoint converge within some specified tolerance.
2.3 Solving the Model with ENDG

To solve for optimal behavior in a non-terminal period using the method of endogenous gridpoints, we first define the “post-decision state space” as combinations of retained assets $a_t = m_t - c_t - \eta_t$ and post-investment health $H_t = h_t + f(\eta_t)$. A post-decision state represents the individual’s situation after actions have been taken, but before the risky transition into the next period. ENDG has the econometrician specify a mesh of gridpoints in the post-decision state space, with typical element $(a_j, H_j)$. With this transformation, consider the slightly rearranged first order conditions for optimal behavior, based on (6) and the transition equations of (5):

$$u'(c_t) = \beta R \int s(h_{t+1})V_{t+1}^{m-1}(m_{t+1}, h_{t+1})dF(w, \delta),$$

$$f'(\eta_t) = \frac{R \int s(h_{t+1})V_{t+1}^{m-1}(m_{t+1}, h_{t+1})dF(w, \delta)}{(1-\delta) \int (1-s(w)V_{t+1}(\cdot) + s(\cdot)(wV_{t+1}(\cdot) + V_{t+1}(\cdot)))dF(w, \delta)},$$

s.t. $m_{t+1} = a_t R + wh_{t+1}, \quad h_{t+1} = H_t(1-\delta).$

(9)

The RHS of both first order conditions are pinned down for each post-decision gridpoint, no longer depending on the control variables. For each gridpoint, we can then find the action that must have just been taken in order for ending the period at $(a_j, H_j)$ to have been optimal; that is, after computing the RHS, (9) can be uniquely solved for $(c_j, \eta_j)$. With optimal behavior known, the pre-decision state from which these actions were taken can be found by inverting the definitions of the post-decision state:

$$m_j = a_j + c_j + \eta_j,$$

$$h_j = H_j - f(\eta_j),$$

(10)

These are the endogenous gridpoints that give ENDG its name: rather than solving for behavior at preselected, exogenous gridpoints, we instead fix a grid of end-of-period states (thus locking in the distribution of future states) and then find the unique action that was taken to reach the post-decision state, assuming optimality. The iterative consideration of successive guesses of control variables (each requiring numeric integrals over future outcomes) is thus avoided, greatly speeding up solution of the model.

As with traditional EXOG, the value and marginal value functions can be generated by interpolating these quantities at the endogenous gridpoints, using (5) and (8). The behavioral or policy functions $\hat{c}_t(m, h)$ and $\hat{h}_t(m, h)$ can similarly be constructed using the gridpoints and the optimal actions just solved for. The process is then recurred backwards to the previous period until reaching $t = 0$ (or convergence in an infinite horizon variant).

The same ENDG solution can be achieved using the theoretical framework of White...
As before, the inputs suppressed due to space constraints represent the future state solution is presented in the following section.

When the level of health for a gridpoint was calculated as

\[ h_t = \Xi_t(x_t, \eta_t) = (m_t - c_t - \eta_t, h_t + f(\eta_t)), \]

the non-linear investment and production functions; likewise for money resources, which now curve through the pre-decision state set because of the non-linear policy functions. This demonstrates the key difficulty of using the method of endogenous gridpoints when there is more than one endogenous state variable: ordinary linear interpolation requires a rectilinear grid, so an alternative interpolation technique must be found.\(^9\) An efficient and elegant solution is presented in the following section.

\(^9\)Appendix A discusses the flaws of other techniques for interpolating on an irregular set of gridpoints.
3 Interpolation on Non-Linear Grids

This section begins with a brief review of linear interpolation on an ordinary rectilinear grid, for comparison and reference to interpolation on non-linear grids presented in sections 3.2 and 3.3.

3.1 Interpolation on Rectilinear Grids

The state space of the generic problem is $X \subset \mathbb{R}^K$, with typical element $x$. Define a *mesh* $J$ on $X$ as a set of points generated as the cross product of $K$ ordered finite subsets of $\mathbb{R}$, such that $J \subset X$; that is, $J = J_1 \times \cdots \times J_K$.

First, consider the simple one dimensional case. To linearly interpolate a function $f : X \to \mathbb{R}$ over a set of points with known function values, we literally “connect the dots” with line segments between adjacent $(x, f(x))$ points. If $x^A$ and $x^B$ are two consecutive gridpoints in a one-dimensional mesh and $x \in [x^A, x^B]$, then the linear interpolation of $f(\cdot)$ at $x$ is:

$$\alpha = \frac{x - x^A}{x^B - x^A}, \quad f(x) \approx (f(x^B) - f(x^A))\alpha + f(x^A) = (1 - \alpha)f(y^A) + \alpha f(y^B).$$ \hspace{1cm} (14)

This effectively maps the gridpoint interval $x \in [x^A, x^B]$ into the unit interval $\alpha \in [0,1]$. Evaluation of the interpolation at a particular point requires finding the appropriate grid interval (by linear search if the number of gridpoints is small or bisecting search otherwise).

The extension of linear interpolation to two dimensions uses the points of the mesh to partition the state space $X$ into rectangular sectors (see Figure 1). Just as an interval of $\mathbb{R}$ can be continuously mapped into the unit interval, any rectangular subset of $\mathbb{R}^2$ can be translated into the unit square. Defining $x_k$ as the $k$th index of $x$, the bilinearly interpolated approximation to a function $f : \mathbb{R}^2 \to \mathbb{R}$ on the rectangle defined by lower-left vertex $x^A$ and upper-right vertex $x^B$ is:

$$\alpha = \frac{x_1 - x^A_1}{x^B_1 - x^A_1}, \quad \beta = \frac{x_2 - x^A_2}{x^B_2 - x^A_2},$$ \hspace{1cm} (15)

$$f(x) \approx (1 - \alpha)(1 - \beta)f(x^A_1, x^A_2) + \alpha(1 - \beta)f(x^B_1, x^A_2) + (1 - \alpha)\beta f(x^A_1, x^B_2) + \alpha\beta f(x^B_1, x^B_2).$$

Bilinear interpolation approximates the true function by applying linear weights to the known function values at the four vertices. This method can easily be extended to three (or more) dimensions, with each state in a sector mapped to $(\alpha, \beta, \gamma) \in [0,1]^3$.

Linear interpolation on an ordinary rectilinear grid is extremely simple and fast in any number of dimensions: once the sector for a point $x$ has been identified, both the translation
to relative coordinates and evaluation of the interpolation based on these coordinates are trivial steps. Moreover, finding the sector that contains a point is very straightforward, as the $K$ dimensions can be treated as independent bisecting searches; with $N$ values in each grid dimension, the sector can be identified in $O(K \log N)$ time.

The key insight of the interpolation method presented below is that the irregular array of gridpoints generated by ENDG also divides the state space into sectors with $2^K$ vertices (e.g. the four corners of a quadrilateral in 2D, or eight vertices in 3D), and that each of these sectors can be continuously mapped into the unit box (i.e. unit square, unit cube, etc). That is, any point in the state space is an element of a single “irregular sector” and corresponds to a unique value in $[0,1]^K$, so that linear interpolation can be applied. The following subsections address the additional complications that arise with an irregular two dimensional grid in identifying the correct sector for a state space point and calculating the relative coordinates $(\alpha, \beta)$ within that sector.\(^{10}\)

### 3.2 Identifying the Sector on a Non-Linear Grid

Consider the problem of locating the sector in which some point $x \in X \subset \mathbb{R}^2$ resides. With an ordinary rectilinear grid, we can separately identify the sector indices by comparing $x_1$ to the elements of $J_1$ and $x_2$ to the elements of $J_2$; sectors in the same row share vertical bounds, and sectors in the same column share horizontal bounds. This is not the case with an irregular grid generated by the method of endogenous gridpoints: the bounds of the sectors do not align within “rows” or “columns” in a non-linear grid, so we cannot simply perform a bisecting search as with an ordinary mesh. However, we can use a very similar technique to that in Ludwig and Schön for Delaunay interpolation (what they refer to as a “visibility walk”). Starting from some initial guess, we search by “walking” from sector to sector, checking whether $x$ is in the current guess and updating the sector based on which of the boundaries it violates.

Figure 3 shows the sector “walking” search process. At the initial sector guess, the point to be located violates the upper and left boundaries, so the guess is updated to the one up and to the left; at the second guess, only the left boundary is violated, so the sector is updated one to the left. In the third guess, the point is within all four boundaries and the sector has been located, so we can proceed to calculating $\alpha$ and $\beta$ for this point within that sector. At least some points to be evaluated in the interpolated function will fall outside of the convex hull of the gridpoints and thus not inside the bounds of any sector. In this case, the function approximation can be extrapolated using the sector from which the search

\(^{10}\)Appendix B discusses adjustments to these methods when working in three dimensional (or higher) state spaces, while appendix C discusses additional concerns.
process exited the convex hull. So long as the highest state point in each dimension is sufficiently large that the function is approximately linear at such high values, this linear extrapolation will be sufficiently accurate.

To evaluate whether the point $x$ violates the boundary defined by $x^A$ and $x^B$, where $x^B$ is counterclockwise from $x^A$, we evaluate:

$$(x_2^B - x_2^A)x_1 - (x_1^B - x_1^A)x_2 \leq x_1^Ax_2^B - x_2^Ax_1^B.$$ (16)

If this inequality is violated, then $x$ is on the outside of this sector boundary. Referencing the sector in Figure 4, the bounding conditions defined by $x^A$ to $x^B$ (“bottom”), $x^B$ to $x^D$ (“right”), $x^D$ to $x^C$ (“top”), and $x^C$ to $x^A$ (“left”) must each be checked to ensure that $x$ is inside the sector, and the indices of the next sector guess are incremented by $-\pm$ (or zero) depending on the boundary violations.

Note that this process assumes that the endogenous gridpoints that compose the irregular grid are ordered as expected (given their ordering in the fixed grid of post-decision states), allowing the sectors to be easily defined to be known with no additional computations. This assumption is not a strong one and should be met by nearly every model of interest. A violation of the assumption would imply, for example, that beginning a period with a higher amount of resources is associated with lower end of period assets (requiring a marginal propensity to spend from wealth greater than one), or that beginning a period with less health will result in greater health at the end of the period (so that greater health investment overcomes the lower initial health), when the optimal behavioral function is obeyed. Both of these possibilities (as well as versions with other control and state variables) seem absurd and unlikely to arise in practice. Relying on known relationships among the gridpoints is an improvement over Delaunay triangulation, which requires a long construction step to determine the relationships among the points and build the correct triangulation. Similarly, it avoids inverting an $N \times N$ matrix to calculate the weighting scheme for polyharmonic spline interpolation and related techniques.

### 3.3 Calculating Relative Coordinates Within a Sector

Once the sector containing the point $x$ has been identified, we must then calculate the relative coordinates $(\alpha, \beta)$ of the point within the sector so that bilinear interpolation can be applied. To visualize this mapping, consider Figure 4, showing how the $\alpha$ and $\beta$ dimensions are defined

---

11Ideally, the visibility walk should be prevented from exiting the gridpoints unless it is the only valid move, i.e. all boundaries violated at the current guess would make the next guess out of bounds.

12Should the “weak monotonicity” condition be violated and the relative position of gridpoints cannot be known ex ante, then Delaunay interpolation is a viable alternative for constructing the sector map.
on an irregular quadrilateral. The four vertices form an irregular convex quadrilateral, where the \( \alpha \) dimension is defined along the “bottom” \( (x^A \text{ to } x^B) \) and “top” \( (x^C \text{ to } x^D) \) edges and the \( \beta \) dimension along the “left” \( (x^D \text{ to } x^A) \) and “right” \( (x^B \text{ to } x^C) \) edges. The red iso-\( \alpha \) and blue iso-\( \beta \) lines are reproduced on the unit square to demonstrate the mapping from the irregular quadrilateral. To identify points where \( \alpha = 0.4 \), for example, we would go 40\% of the way from \( x^A \) to \( x^B \), and 40\% of the way from \( x^C \) to \( x^D \), and draw a line segment between those points; this is the fifth red line counting from the left. Likewise, to find points corresponding to \( \beta = 0.7 \), we go 70\% of the way from \( x^A \) to \( x^C \), and 70\% of the way from \( x^B \) to \( x^D \), and draw a line segment between those points; this is the eighth red line from the bottom. The \( (\alpha, \beta) = (0.4, 0.7) \) point for this sector is thus defined by the unique intersection of the fifth red line and eighth blue line.

Every point in the convex hull of the four vertices has a unique \( (\alpha, \beta) \) identifier, just as with a traditional rectangular sector generated from an ordinary mesh. Thus the title of this section (and the paper itself): we will use bilinear interpolation on a non-linear grid. Each grid sector is non-linearly warped, so that a linear path through the sector almost always corresponds to a non-linear path in \( (\alpha, \beta) \) space. Translating from \( (\alpha, \beta) \) to the state space \( X \) is very easy, but we are interested in the slightly more difficult inverse process.

In a two dimensional state space, \( (\alpha, \beta) \) can be found algebraically from \( (x_1, x_2) \). The translation from the unit square to the irregular quadrilateral sector is described by two bilinear equations, one for each dimension:

\[
x_1 = (1 - \alpha)(1 - \beta)x_1^A + \alpha(1 - \beta)x_1^B + (1 - \alpha)\beta x_1^C + \alpha \beta x_1^D, \tag{17}
\]

\[
x_2 = (1 - \alpha)(1 - \beta)x_2^A + \alpha(1 - \beta)x_2^B + (1 - \alpha)\beta x_2^C + \alpha \beta x_2^D.
\]

These two equations can be rearranged and relabeled as:

\[
x_1 = x_1^A + (x_1^B - x_1^A)\alpha + (x_1^C - x_1^A)\beta + (x_1^A - x_1^B - x_1^C + x_1^D)\alpha \beta \equiv a + b\alpha + c\beta + d\alpha \beta, \tag{18}
\]

\[
x_2 = x_2^A + (x_2^B - x_2^A)\alpha + (x_2^C - x_2^A)\beta + (x_2^A - x_2^B - x_2^C + x_2^D)\alpha \beta \equiv e + f\alpha + g\beta + h\alpha \beta.
\]

These expressions can likewise be manipulated to yield an algebraic solution for \( \alpha \) and \( \beta \):

\[
\mu = \frac{hb - df}{dg - hc}, \quad \tau = \frac{h(a - x_1) - d(e - x_2)}{dg - hc},
\]

\[
\zeta = a - x_1 + c\tau, \quad \eta = b + c\mu + d\tau, \quad \theta = d\tau.
\]

\[
\alpha = (-\eta \pm \sqrt{\eta^2 - 4\theta\zeta})/(2\theta), \quad \beta = \mu\alpha + \tau. \tag{19}
\]
Once the correct sector for a point has been identified and the \((\alpha, \beta)\) values have been calculated, the interpolated function can be approximated at that point using simple bilinear interpolation, as in (15).

The quadratic form of the algebraic solution means that two \((\alpha, \beta)\) pairs seemingly solve the system in (17); however, only one of these pairs is the true solution, as the other will be well outside the unit square. For any given sector, either the “plus” or “minus” solution (from the ±) will be correct for every point in that sector, but there does not seem to be a way to know which one ex ante. Instead, we simply check the “polarity” of the solution by calculating \((\alpha, \beta)\) for a point known to be inside the sector—say, the midpoint between \(x_A\) and \(x_D\)—using the “plus” solution. If this yields an \((\alpha, \beta)\) inside the unit square, then the “plus” solution should be used throughout this sector; otherwise the “minus” solution should be used. This check must be done only once per sector, just after the endogenous gridpoints are found.

As with the sector identification process in section 3.2, the validity of the relative coordinates depends on the solution to the model being “weakly monotone”, so that increasing a pre-decision state never causes the resulting post-decision state to decrease (as with an MPC greater than 1). Moreover, the existence of these coordinates also requires that every sector adheres to a weak form of convexity: no vertex of a sector may be inside the convex hull of all other vertices of that sector. In a two dimensional state space, this condition aligns exactly with each sector being a convex quadrilateral, but is weaker than convexity in higher dimensional models, where the sectors are almost surely not convex. Violations of this condition have not yet been found, but could theoretically occur if the behavioral functions rapidly flip from being convex to concave (or vice versa) and the grid is insufficiently dense; as above, Delaunay interpolation is a viable alternative if “weak convexity” is not met.

4 Speed & Accuracy of Non-Linear Grid Interpolation

This section presents a comparison of the speed and accuracy of ENDG with non-linear grid interpolation versus traditional EXOG with ordinary bilinear interpolation, using the model of section 2 as the benchmark.

4.1 Calibration and Methods

For ease of reference, all parameters are calibrated identically to those in Ludwig & Schön; these are provided in Table 1. As my model incorporates additional uncertainty in the wage
and health capital depreciation rates (including the possibility of unemployment), additional parameters governing these distributions are also presented.  

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>Number of non-terminal periods</td>
<td>99</td>
</tr>
<tr>
<td>( \rho )</td>
<td>Coefficient of relative risk aversion</td>
<td>0.5</td>
</tr>
<tr>
<td>( \nu )</td>
<td>Curvature of health production function</td>
<td>0.35</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Magnitude of health production function</td>
<td>1</td>
</tr>
<tr>
<td>( \phi )</td>
<td>Maximum death probability ((as \ h_{t+1} \to 0))</td>
<td>0.5</td>
</tr>
<tr>
<td>( \beta )</td>
<td>Intertemporal discount factor</td>
<td>0.9615</td>
</tr>
<tr>
<td>( \bar{w} )</td>
<td>Average wage rate</td>
<td>0.1</td>
</tr>
<tr>
<td>( \delta )</td>
<td>Average capital depreciation rate</td>
<td>0.05</td>
</tr>
<tr>
<td>( R )</td>
<td>Gross interest factor</td>
<td>1.05</td>
</tr>
<tr>
<td>( \Upsilon )</td>
<td>Unemployment rate</td>
<td>0.07</td>
</tr>
<tr>
<td>( \sigma_w )</td>
<td>Standard deviation of wage shocks</td>
<td>0.1</td>
</tr>
<tr>
<td>( \sigma_\delta )</td>
<td>Width of depreciation risk band</td>
<td>0.05</td>
</tr>
</tbody>
</table>

The model is solved at a variety of grid sizes (in \( m_t \times h_t \) for EXOG and \( a_t \times H_t \) for ENDG), with the grid values in each dimension on roughly \([0, 300]\). Specified with double log spacing, but similar results are reached with triple log spacing. 

Experience reveals that this is a sufficient approximation. Using a finer shock grid or more complex numeric integration technique would only make the timing results below even more striking.

At each grid size, two versions of the model are solved using EXOG and ENDG. First, I solve a version of the model with the wage and depreciation shocks turned off \((\sigma_w = \sigma_\delta = 0, \text{ but with unemployment})\), allowing nearly direct comparison with the results of Ludwig and Schöen using Delaunay interpolation. I then solve the full model with wage and depreciation shocks, approximating each distribution with a seven point discretization for a total of \((7 + 1) \cdot 8 = 56\) discrete shocks. This second version more fully illuminates the benefits of ENDG with non-linear grid interpolation, as nearly all models of interest will incorporate interperiod transition risk. The solution time is recorded for each grid size, ignoring the fixed time cost of defining the problem.

\[15\] For any gridsize, there is approximately 0.3-0.4 seconds of setup time to define grids and load the OpenCL kernel, after which the model can be solved an arbitrary number of times at different parameters, as in an estimation. The times presented below thus represent “marginal time cost” per parameter set.

\[13\] I solve only a finite horizon version of the problem, as no additional insight would be provided by the infinite horizon variant in this case.

\[14\] Specified with double log spacing, but similar results are reached with triple log spacing.

\[15\] Experience reveals that this is a sufficient approximation. Using a finer shock grid or more complex numeric integration technique would only make the timing results below even more striking.

\[16\] For any gridsize, there is approximately 0.3-0.4 seconds of setup time to define grids and load the OpenCL kernel, after which the model can be solved an arbitrary number of times at different parameters, as in an estimation. The times presented below thus represent “marginal time cost” per parameter set.
The model is solved on a desktop computer with a consumer grade processor (Intel Core i7-3770). The solution code (available upon request) is written primarily in OpenCL, with Matlab used to define the parameters and pass arrays to the solver. Most solution methods are single threaded, using only one core on the CPU and thus a fraction of its power. OpenCL is a relatively new language that allows heterogeneous computing: the source code can be compiled and run on graphics processing units (GPUs) as well as standard CPU computing; it is specifically designed for parallel processing to take advantage of the hardware on a GPU. This allows multiple parameter sets to be solved simultaneously through a multi-threaded process; in economic settings, this might represent preference or other exogenous heterogeneity across agents, an increasingly standard feature of serious models.

To demonstrate the computational advantage of OpenCL (or the more popular CUDA\textsuperscript{17}), I present both single-threaded and multi-threaded timings for each solution method.

To evaluate the accuracy of the solution at each grid size and solution method, I calculate the normalized Euler errors (in the style of Judd (1992) or Santos (2000)) for consumption and investment in each time period for 100 simulated individuals,\textsuperscript{18} who are initialized with \((m_0, h_0)\) states evenly spaced on \([10, 100] \times [50, 100]\). By comparing the action actually taken in period \(t\) to the optimal action implied by the first order and envelope conditions, based on the distribution of subsequent actions in period \(t + 1\), the errors for consumption and investment can be calculated as:

\[
e_{1,t} = c_t - \left( \beta R \int s(h_{t+1})^\rho c_{t+1}^\rho dF(w, \delta) \right)^{-1/\rho},
\]

\[
e_{2,t} = \eta_t - \frac{R \int s(h_{t+1})^\rho c_{t+1}^\rho dF(w, \delta)}{\gamma \int (1 - \delta) \left( s'(h_{t+1})V_{t+1}(m_{t+1}, h_{t+1}) + s(h_{t+1})c_{t+1}^\rho (w + \frac{\eta_{t+1} - \gamma}{\nu}) \right) dF(w, \delta)} \right)^{1/(\nu-1)}.
\]

These errors are then normalized and converted into “digits of accuracy” by:

\[
\tilde{e}_{1,t} = -\log_{10}(|e_{1,t}/c_t|), \quad \tilde{e}_{2,t} = -\log_{10}(|e_{2,t}/\eta_t|).
\]

A normalized Euler error of \(\tilde{e}_{1,t} = 4\) indicates that the individual makes a $1 error for every $10,000 of consumption. Averaged across all individuals and over all non-terminal simulated periods, this yields average digits of accuracy for the behavioral functions; I also report the

\textsuperscript{17}While CUDA has existed longer and is more commonly used, the primary advantage of OpenCL is the wider array of devices on which it runs: CUDA is restricted to nVidia GPUs and Intel CPUs, while OpenCL runs on these devices as well as CPUs and GPUs by AMD. The latter group is significantly less expensive than the former so that even the most budget constrained economist can take advantage of heterogeneous computing and massively parallel processing.

\textsuperscript{18}Simulated individuals are assumed to survive from period to period.
average of the 0.1% worst Euler errors as a measure of the lower tail of accuracy. So as not to conflate error from numeric integrals (i.e. discrete approximations to the shock distributions) with error from the solution method itself, I report these accuracy measures only for the version of the model with no wage or depreciation shocks, but with unemployment.

4.2 Results

Table 2 presents timing results for the version of the model with wage and depreciation rate shocks turned off (but with unemployment) for a variety of grid sizes. For any plausible number of gridpoints, the method of endogenous gridpoints with non-linear grid interpolation is 4.3 to 5.4 times faster than the traditional exogenous gridpoints method. This is a marked improvement over the speedup achieved by Ludwig and Schöen for nearly the same model using ENDG with Delaunay triangulation; they report a speedup of about 2.4 times for relatively sparse grids (25 × 25 to 50 × 50), but these gains dwindle to less than a 10% reduction in solution time for a very dense 300 × 300 state grid.

Table 2: Timing of Solution Methods, No Shocks (seconds, speed relative to EXOG)

| Number of grid points | Single threaded | | Multi threaded |
|-----------------------|-----------------|-----------------|
|                       | EXOG            | ENDG            | EXOG            | ENDG            |
| 25 × 25               | (0.445, 1×)     | (0.083, 5.4×)   | (0.084, 1×)     | (0.019, 4.4×)   |
| 50 × 50               | (1.803, 1×)     | (0.336, 5.4×)   | (0.330, 1×)     | (0.066, 5.0×)   |
| 100 × 100             | (7.537, 1×)     | (1.418, 5.3×)   | (1.338, 1×)     | (0.258, 5.2×)   |
| 150 × 150             | (16.90, 1×)     | (3.417, 4.9×)   | (3.012, 1×)     | (0.623, 4.8×)   |
| 200 × 200             | (30.08, 1×)     | (6.480, 4.6×)   | (5.316, 1×)     | (1.203, 4.4×)   |
| 250 × 250             | (47.45, 1×)     | (10.73, 4.4×)   | (8.473, 1×)     | (1.960, 4.3×)   |
| 300 × 300             | (70.29, 1×)     | (16.26, 4.3×)   | (12.38, 1×)     | (2.976, 4.2×)   |

The key difference between the performance of the interpolation techniques is that Delaunay triangulation requires a lengthy, serial construction step that non-linear grid interpolation does not. With $N$ gridpoints, triangulation construction time is $O(N \log N)$, explaining the decreasing benefit of ENDG relative to EXOG in their paper. This erosion barely occurs with non-linear grid interpolation, shown graphically in Figure 5: solution time with EXOG is linear in the number of gridpoints and almost linear with ENDG. The computational benefits of the method of endogenous gridpoints in multiple dimensions are thus not limited to small grid sizes, but are roughly constant in the size of the problem. This is particularly useful for models with three or four endogenous dimensions, where even relatively sparse grids multiply to very large numbers of gridpoints.
A skeptical (or, less generously, suspicious) reader might question the stark difference in absolute timing between the two non-standard interpolation techniques. That is, does ENDG with non-linear grid interpolation only achieve such a relative speed gain because the traditional EXOG method was poorly coded or somehow “half-assed”? The answer is no. Note that the timings for EXOG in Table 2 are just less than twice those of Ludwig and Schöhn for each corresponding grid size; this is precisely the result that should be achieved. The “no shocks” version of the model is identical to theirs but for the possibility of unemployment, so that two possible (non-dead) future states must be considered for each guess of behavior at each gridpoint, compared to only one future state in their version. The amount of computation necessary is thus roughly twice as large, and the solution time correspondingly twice as long. Even using slightly different optimization methods and possibly different hardware, our traditional EXOG solutions yield congruent timings.

Table 3: Timing of Solution Methods, With Shocks (seconds, speed relative to EXOG)

<table>
<thead>
<tr>
<th>Number of grid points</th>
<th>EXOG</th>
<th>ENDG</th>
<th>Single threaded</th>
<th>EXOG</th>
<th>ENDG</th>
<th>Multi threaded</th>
</tr>
</thead>
<tbody>
<tr>
<td>25 × 25</td>
<td>(9.336, 1×)</td>
<td>(1.216, 7.7×)</td>
<td>(1.765, 1×)</td>
<td>(0.248, 7.1×)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50 × 50</td>
<td>(38.24, 1×)</td>
<td>(4.893, 7.8×)</td>
<td>(7.099, 1×)</td>
<td>(0.994, 7.1×)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100 × 100</td>
<td>(156.2, 1×)</td>
<td>(20.27, 7.7×)</td>
<td>(28.90, 1×)</td>
<td>(3.761, 7.7×)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>150 × 150</td>
<td>(364.8, 1×)</td>
<td>(47.11, 7.7×)</td>
<td>(65.70, 1×)</td>
<td>(8.723, 7.5×)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>200 × 200</td>
<td>(650.6, 1×)</td>
<td>(86.88, 7.5×)</td>
<td>(118.1, 1×)</td>
<td>(15.91, 7.4×)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>250 × 250</td>
<td>(1023, 1×)</td>
<td>(140.4, 7.3×)</td>
<td>(183.9, 1×)</td>
<td>(25.92, 7.1×)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300 × 300</td>
<td>(1475, 1×)</td>
<td>(210.1, 7.0×)</td>
<td>(266.8, 1×)</td>
<td>(38.29, 7.0×)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In the version of the model with non-trivial interperiod risk (through wage and health capital depreciation shocks), the computational acceleration of ENDG relative to EXOG is even more dramatic, as presented in Table 3. With risk, ENDG is 7.0 to 7.8 times faster than traditional EXOG, a greater improvement than in the “no shock” model (shown graphically in Figure 6). The additional speed gain with risk can be explained by visibility walks that visit fewer sectors on average. That is, the sector for one discrete shock is fairly likely to be the same as the discrete shock previously considered (or at most one step away); this very good initial sector guess leads to a very short search in most cases. In the model without risk, however, the preceding correct sector corresponds to an entirely different end-of-period state half of the time and is thus a worse initial guess. In this way, solving the model with shocks takes only 13-15 times longer than without shocks, even though 28 times (56/2) more future states are visited.\footnote{Ludwig and Schöhn do not present a version with interperiod risk, but presumably Delaunay triangulation would show an even greater speedup difference (compared to no risk) because the lengthy construction step is a smaller portion of total computation time.}
The right hand panel of Tables 2 and 3 present alternative timings for both versions of the model when the multi-threading capabilities of OpenCL are used. The single threaded timings shown in the left hand panels use only one core on an eight-core CPU, whereas the multi-threaded version uses all eight cores to solve eight parameter sets simultaneously, as if the model included preference heterogeneity among individuals; the timings presented here are per parameter set. While this could theoretically result in an eight-fold speed gain over single threading, the acceleration is actually only by a factor of five to six; this could be due to memory limitations as multiple threads try to access the same information. Both with and without wage and depreciation shocks, the relative speed up of ENDG over EXOG is very similar in the multi-threaded version at each gridsize. Even greater computational gains are achieved by solving the model on a GPU rather than a CPU,\textsuperscript{20} so that multiple gridpoints are solved simultaneously within each parameter set: solution times are about 10-12 times faster on a mid-grade consumer GPU (Radeon R9 280X) than in the multi-threaded times shown here.\textsuperscript{21} This is an additional benefit for non-linear grid interpolation over Delaunay triangulation: the serial construction step is ill-suited for the massively parallel structure of GPU architecture, limiting the possible speed gains.

<table>
<thead>
<tr>
<th>Number of grid points</th>
<th>Overall average ((c, \eta))</th>
<th>Average of 0.1% worst ((c, \eta))</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXOG</td>
<td>ENDG</td>
</tr>
<tr>
<td>25 \times 25</td>
<td>(3.48, 2.45)</td>
<td>(3.87, 2.79)</td>
</tr>
<tr>
<td>50 \times 50</td>
<td>(4.07, 3.11)</td>
<td>(4.26, 3.27)</td>
</tr>
<tr>
<td>100 \times 100</td>
<td>(4.65, 3.65)</td>
<td>(4.90, 3.87)</td>
</tr>
<tr>
<td>150 \times 150</td>
<td>(5.00, 3.97)</td>
<td>(5.17, 4.18)</td>
</tr>
<tr>
<td>200 \times 200</td>
<td>(5.21, 4.18)</td>
<td>(5.41, 4.39)</td>
</tr>
<tr>
<td>250 \times 250</td>
<td>(5.36, 4.35)</td>
<td>(5.55, 4.57)</td>
</tr>
<tr>
<td>300 \times 300</td>
<td>(5.50, 4.48)</td>
<td>(5.66, 4.69)</td>
</tr>
</tbody>
</table>

The accuracy of EXOG and ENDG at a variety of grid sizes is presented in Table 4. The numbers in the left hand panel should be interpreted as the average digits of accuracy experienced along the simulated paths described in section 4.1, for consumption and investment respectively. For any number of gridpoints, the method of endogenous gridpoints is slightly more accurate in each control variable, and this accuracy increases as expected as the grid of states becomes finer in each dimension. The same pattern holds in the right hand panel, presenting a “worst case scenario” of accuracy through the lower tail of normalized Euler errors. Further validating the comparison between methods, the accuracy measures

\textsuperscript{20} Operationally, this requires changing one integer, indicating that OpenCL should use a different device.

\textsuperscript{21} Versions of the timing tables using a GPU are available upon request.
are nearly identical to those of Ludwig and Schön (though their implementation of EXOG does not suffer from the slight accuracy loss shown here). The tradeoff between computational speed and accuracy of solution is represented graphically in Figures 7 and 8, plotting average normalized Euler error against log solution time. The average slope of these lines reveals that attaining an extra digit of accuracy costs about ten times more computation time (slope of 2.3 in log terms) for EXOG and sixteen times more for ENDG (twelve vs fifteen times longer in the model with shocks). Between being slightly more accurate and considerably faster for any given number of gridpoints, the advantage of ENDG over EXOG can be alternatively phrased purely in terms of accuracy: for any given computational time, ENDG yields 0.8 to 1.2 more digits of accuracy than EXOG.

5 Conclusion

This paper presents an intuitive technique for interpolating on an irregular collection of gridpoints in two or more dimensions, subject to knowing the ordering of these gridpoints; such collections are generated as the beginning-of-period states calculated when using the method of endogenous gridpoints to solve problems with multiple endogenous state variables and multiple controls. This interpolation technique maps irregular quadrilaterals into the unit square, acting as an intuitive extension of bilinear interpolation on an ordinary rectilinear grid. Unlike other methods for interpolating on an irregular collection of points, the method presented here involves neither a lengthy construction step nor evaluation step.

The method’s computational efficiency when used with the method of endogenous gridpoints is demonstrated on a benchmark two dimensional model (with interperiod risk), yielding a speedup of about 7.7 times relative to the traditional solution method. Critically, this computational acceleration erodes very slowly as the density of gridpoints increases, so that additional accuracy can be achieved while maintaining the speed advantages of the method of endogenous gridpoints. Moreover, the method extends to three or more dimensions with only minor modifications, translating irregular, non-polyhedral shapes into the unit cube.

Beyond the requirements for a model to be solvable by the method of endogenous gridpoints (discussed in a theoretical paper written as this work’s companion), the only additional condition for non-linear grid interpolation to be used is likely to be met by the vast majority of models. A violation of this condition would require a marginal propensity to spend from wealth greater than one or similarly non-intuitive results. Given the dramatic computational improvement and the fairly broad range of models for which the methods presented here are applicable, the technique represents a significant contribution to efforts to make complex dynamic optimization models solvable and able to be estimated in reasonable time.
References


Figure 1: An ordinary rectilinear grid formed by exogenous gridpoints; identifying the rectangular sector for any point in the space is easy

Figure 2: A non-linear grid formed by endogenous gridpoints in period $T - 1$; identifying the quadrilateral sector requires more work
Figure 3: Identifying the sector for point $x$ by visibility walk: guess #1 violates top and left boundaries (red lines), guess #2 violates left boundary (blue line), guess #3 is correct.

Figure 4: Identifying relative coordinates $(\alpha, \beta)$ in a non-linear grid sector; red lines are iso-$\alpha$; blue lines are iso-$\beta$; irregular quadrilateral continuously maps to unit square.
Figure 5: Solution time of model by number of gridpoints, EXOG with ordinary bilinear interpolation vs ENDG with non-linear grid interpolation; only unemployment shocks

Figure 6: Solution time of model by number of gridpoints, EXOG with ordinary bilinear interpolation vs ENDG with non-linear grid interpolation; with wage and depreciation shocks
Figure 7: Digits of accuracy achieved by EXOG and ENDG vs solution time; unemployment shocks only

Figure 8: Digits of accuracy achieved by EXOG and ENDG vs solution time; with wage and depreciation shocks (accuracy based on no shocks model)
Figure 9: Identifying relative coordinates using Newton’s method: want to find $(\alpha, \beta)$ for red point. Begin with initial guess of $(\alpha_0, \beta_0) = (0.5, 0.5)$ at blue point.

Figure 10: Assume entire sector is similar to iso-$\alpha$ and iso-$\beta$ lines through initial guess, generating blue grid. Solve location of red point on blue grid, generating $(\alpha_1, \beta_1)$; project onto original black grid to get green point (fairly close to target red point).
Figure 11: Now assume entire sector is similar to iso-\(\alpha\) and iso-\(\beta\) lines through updated guess, generating green grid. Solve location of red point on green grid, generating \((\alpha_2, \beta_2)\); project onto original black grid to get magenta point (covers up target red point).

Figure 12: Extrapolation with non-linear grids. Function can be reasonably approximated at points somewhat outside the grid, but problems arise at far distant points.
Appendices

A Alternative Interpolation Methods

There do exist other methods for interpolating on an irregular collection of gridpoints, but all of these techniques suffer from one or more deficiencies: they are computationally burdensome, have a bottleneck in construction, potentially introduce non-concavities, and/or do not translate well to higher dimensions. This section will briefly describe some of the existing interpolation methods and how they are imperfect.

Most saliently, Ludwig and Schön (2014) use Delaunay triangulation to interpolate the value and behavioral functions in a paper currently under revision at the JEDC. The Delaunay triangulation among a set of points is the triangular graph such that no point is inside the circumference of any triangle of the graph. It maximizes the minimum angle of the triangles in the graph, avoiding “skinny” triangles. The interpolation is evaluated at a particular point in the domain by first locating the unique triangle that contains the point, and then approximating the function by using triangular weights based on the distance from the point to each vertex. The authors demonstrate that a two dimensional dynamic optimization problem can be solved about three times faster using the method of endogenous gridpoints with Delaunay triangulation than with traditional solution and interpolation methods, while also matching the accuracy level for any number of gridpoints used.

However, Delaunay interpolation is an imperfect method for these purposes. In each time period, the triangulation must be constructed point-by-point in a serial fashion. While there is a method that is guaranteed to eventually yield the unique Delaunay triangulation, this process involves making mistakes along the way, requiring adjustments to previously placed edges. With \( N \) gridpoints, construction of the triangulation is an \( O(N \log(N)) \) process, and thus the computational burden grows more than linearly as the grid of states becomes finer (to increase accuracy). Moreover, the construction must proceed serially, adding one vertex at a time, with no parallelization possible. Advances in computer architecture now focus on parallel processing— as engineers reach physical limits of circuit size and thus processing speed— and so Delaunay triangulation cannot take full advantage of computational resources. Constructing the triangulation becomes so burdensome at larger numbers of gridpoints that the method is dominated by a hybrid technique presented in the same paper, mixing exogenous and endogenous gridpoint approaches. Indeed, at extremely fine grids this method would be slower than the traditional solution approach, as the gains from faster behavioral optimization are more than offset by the burden of constructing the triangulation. Further, the memory storage needs of Delaunay triangulation grow exponentially in the number of
dimensions, as each gridpoint has more associated edges.

While Delaunay triangulation is an imperfect technique for dynamic optimization problems, Ludwig and Schönh did not err in their choice: all other available methods are significantly less suitable. Voronoi (or natural neighbor) interpolation also requires constructing the Delaunay triangulation, and needs additional manipulation to make the Voronoi tessellation (the dual graph of the Delaunay triangulation) and a more difficult evaluation of the interpolated function—essentially adding an additional point to the tessellation before computing weights. Moreover, Voronoi interpolation has no way of evaluating an “extrapolated” domain point— one that is outside the convex hull of the gridpoints. Other interpolation methods have similarly burdensome evaluation, as they use information about every gridpoints rather than only nearby points. For example, the function can be approximated by weighting the function values of all gridpoints by the inverse distance from the domain point to be evaluated; powers of the inverse distance can also be used, with the method approaching nearest neighbor interpolation as the power becomes very large. While inverse distance weighting requires no construction step, a single evaluation of the interpolation is $O(N)$ with $N$ gridpoints, a computational burden.

Interpolation methods using polyharmonic splines (including thin plate spline interpolation) likewise use data from all gridpoints for every function evaluation, weighting according to so-called radial basis functions. Even worse, additional parameters of the interpolation must be calculated, requiring the inversion of a matrix larger than $N \times N$, with computational complexity of at least $O(N^{2.373})$. Related, polyharmonic spline methods (as well as Kriging) can potentially introduce non-concavities into the value function, potentially yielding behavioral solutions that are non-monotone. These numeric instabilities tend to grow and eventually “explode” as the problem is iterated back to earlier periods, often collapsing into complex values being assigned to behavior. In short, other interpolation methods are appropriate in a static setting for approximating a function based on values at a limited set of gridpoints, but are unsuitable for use when the approximated function is to be used in a dynamic optimization.

B  Higher Dimensional Interpolation on Non-Linear Grids

In models with three or more continuous and endogenous state dimensions, both the sector search and calculation of relative coordinates must be somewhat modified from the two dimensional algorithms. While these methods are slower than their basic 2D counterparts, the additional computational burden is significantly less than the lengthy construction step of Delaunay or Voronoi interpolation in higher dimensions (nor does the required memory
to store the interpolation grow exponentially with the number of dimensions) or the $O(N)$ evaluation procedure of inverse distance weighting and related methods. Note that if a state dimension is not endogenous—its transition from period to period is entirely exogenous, rather than depending on the individual’s action—then non-linear grid interpolation need not (and should not) be used in that dimension; ordinary linear interpolation can be used instead.

In two dimensions, we can evaluate whether a point $x$ is in a sector by explicitly evaluating the four linear bounding conditions, as in (16). With a three dimensional state space, the six “sides” of an irregular grid sector are not defined by planes, but rather have some curvature to them. That is, the four vertices of a “side” of a sector almost surely do not align on a single plane and thus we cannot use a 3D extension of the 2D boundary evaluation method. Instead, we first check whether it is plausible for $x$ to be in the sector based on the “bounding box”: the smallest rectangular prism superset of the sector. Defining $\chi$ as the set of $2^K$ sector vertices, $x$ must satisfy:

$$\left(\min_{i} x^i_k \in \chi\right) \leq x_k \leq \left(\max_{i} x^i_k \in \chi\right) \text{ for } k = 1, \cdots, K.$$  \hspace{1cm} (22)

If any of the $2K$ inequalities are violated, then $x$ cannot be in the current sector; the indices of the next guess can be incremented based on these violations, as in section 3.3. If all of the inequalities are satisfied, then we know that $x$ is either in the current sector or fairly close to it. We then calculate the relative coordinates of $x$ in the current sector using the procedure in the paragraphs below, under the assumption that it is in this sector. If that process returns valid coordinates (each in $[0, 1]$) then the sector has been correctly identified and the search terminates. If invalid coordinates are returned, then the sector guess can be updated based on the violations; e.g. if $\beta > 1$, then the next sector guess should be incremented one to the “right”. Barring certain bad grid orientations (discussed in appendix C), it is rare that more than two sectors will have the relative coordinates of $x$ calculated.

Calculating the relative coordinates of point $x$ within a sector can be achieved algebraically in a two dimensional setting through judicious use of the quadratic formula; this method cannot be used in three dimensions or higher.\footnote{Though Cardano and Ferrari (1545) devised the cubic and quartic formulas, a three dimensional model would require successive application of the cubic and quadratic formulas, yielding six potential solutions, with at most one being valid. In four dimensions, one would need to successively apply the quartic, cubic, and quadratic formulas, yielding twenty-four potential solutions. Given the considerable arithmetic involved in those formulas, and then successive checking for validity, it is both easier to program and faster to compute the Newton search described here.} Instead, we can use Newton’s method to find these coordinates. In three dimensions, the mapping from the unit cube into
an irregular sector\textsuperscript{23} consists of a triple trilinear system:

\[
x_k = (1 - \alpha)(1 - \beta)(1 - \gamma)x_k^A + \alpha(1 - \beta)(1 - \gamma)x_k^B + (1 - \alpha)\beta(1 - \gamma)x_k^C + \alpha\beta(1 - \gamma)x_k^D \\
+ (1 - \alpha)(1 - \beta)\gamma x_k^E + \alpha(1 - \beta)\gamma x_k^F + (1 - \alpha)\beta\gamma x_k^G + \alpha\beta\gamma x_k^H \quad \text{for } k = 1, 2, 3.
\]

(23)

Each partial derivative of this system is itself a bilinear equation of the other relative coordinates. For example:

\[
\frac{\partial x_k}{\partial \alpha} = (1 - \beta)(1 - \gamma)(x_k^B - x_k^A) + \beta(1 - \gamma)(x_k^D - x_k^C) + (1 - \beta)\gamma(x_k^F - x_k^E) + \beta\gamma(x_k^H - x_k^G).
\]

(24)

Starting from an initial guess of \((\alpha_0, \beta_0, \gamma_0) = (0.5, 0.5, 0.5)\) and defining \(x^i\) as the point yielded when \((\alpha_i, \beta_i, \gamma_i)\) are used in (23), we can then iteratively update the guess of relative coordinates until the process converges to a given tolerance by doing:

\[
\begin{bmatrix}
\alpha_{i+1} \\
\beta_{i+1} \\
\gamma_{i+1}
\end{bmatrix} =
\begin{bmatrix}
\alpha_i \\
\beta_i \\
\gamma_i
\end{bmatrix} +
\begin{bmatrix}
\frac{\partial x_1}{\partial \alpha} & \frac{\partial x_1}{\partial \beta} & \frac{\partial x_1}{\partial \gamma} \\
\frac{\partial x_2}{\partial \alpha} & \frac{\partial x_2}{\partial \beta} & \frac{\partial x_2}{\partial \gamma} \\
\frac{\partial x_3}{\partial \alpha} & \frac{\partial x_3}{\partial \beta} & \frac{\partial x_3}{\partial \gamma}
\end{bmatrix}^{-1}
\begin{bmatrix}
x_1 - x_1^i \\
x_2 - x_2^i \\
x_3 - x_3^i
\end{bmatrix}.
\]

(25)

Because Newton’s method converges quadratically, the function implied by (23) is roughly linear, and the initial guess is reasonably close to the true relative coordinates, this procedure converges to double precision (approximately \(10^{-16}\)) in fewer than six iterations on average.

In this setting, the Newton search for relative coordinates has an intuitive graphical interpretation, shown in two dimensions in Figures 9-11. At the current guess \((\alpha_i, \beta_i)\), the updating step assumes that all iso-\(\alpha\) lines are parallel to the iso-\(\alpha\) line through \(x^i\), and likewise for iso-\(\beta\), creating a “pseudo-sector”. To update the guess, we find the \((\alpha, \beta)\) for \(x\) in the pseudo-sector: the relative position of \(x\) when we ignore the non-linearity of the sector and assume that the local isoquants represent the entire quadrilateral. In Figure 9, we want to find the relative coordinates of the red point when the sector’s true isoquants are defined by the black grid. Starting from the initial guess of the blue point at \((\alpha_0, \beta_0) = (0.5, 0.5)\), we temporarily assume that the sector’s isoquants all have the same slope as at this point to generate the overlaid blue grid (shown in Figure 10). We then calculate that the red point falls at approximately \((\alpha_1, \beta_1) = (0.21, 0.13)\) on the blue grid. Using this new guess in (17) (the true sector mapping, black grid), we find \(x^i\) at the green point—fairly close to the target red point in a single iteration. Figure 11 shows how this updating step would be repeated from \((\alpha_1, \beta_1)\), generating the green grid; after only two iterations, \(||x^i - x|| \approx 0.005\).

\textsuperscript{23}With points \(x^A, x^B, x^C, x^D\) defining the “low side” in the same orientation as in Figure 4, and points \(x^E, x^F, x^G, x^H\) their respective “high side” counterparts.
Both the sector search method and the Newton search for relative coordinates apply in four (or more) dimensional state spaces in the same way as in 3D, though the corresponding partial derivatives in (24) require more arithmetic operations, and the inversion of the Jacobian in (25) is less trivial. Note that finding the maximal and minimal vertices in each dimension for a sector (as in (22)) need only be performed once, when the irregular grid is created. The indices of the extrema for each sector can be stored, requiring only \(2(K^2 - K)\) bits per sector; even in a model with six endogenous dimensions, this information can be stored in a long integer. In contrast, the storage needs of Delaunay interpolation grows exponentially in the number of dimensions.

C Other Considerations

Beyond the basic mechanics of interpolation on non-linear grids, there are additional considerations that affect the speed and/or accuracy of the method, which will be discussed in this appendix. Firstly, the sector identification process is susceptible to bad gridpoint orientations, as alluded to in appendix B, which can result in unnecessarily long searches. If the gridpoints are significantly more dense in one dimension than another (so that the sector map looks like a stack of spaghetti), then the search process will not be able to efficiently move through the state space, taking repeated small steps in only one dimension. Conversely, if the grid is insufficiently dense in a region where the behavioral functions are highly curved, then the search process can lead to overshooting the correct sector due to sharp changes in the boundary gradient across adjacent sectors. Related, the roughly linear approximations to the behavioral and value functions are less accurate when the grid is insufficiently dense in regions where these functions are highly curved. In practice, care must be taken to calibrate the density of the grid of post-decision states in each dimension so as to balance the speed of the search process versus the accuracy of the resulting approximation (see section 4). Double log and triple log spacing of gridpoints is fairly common (so that the grids are more dense at lower state values, where curvature tends to be greatest), but the number of gridpoints in each dimension should be hand calibrated for each model.

In certain gridpoint orientations, it is possible for the sector search processes described in section 3.2 and appendix B to become trapped in perpetual loops, cycling among the same sectors but never landing on the correct one. The number of steps taken must thus be capped; the correct sector for the point \(x\) can then be found by brute force (checking every sector) or simply approximated by the last sector visited before termination (sacrificing some accuracy to avoid the exhaustive search). The grid orientations that generate these infinite
loops do not seem to occur with models where the sector map is generated by the method of endogenous gridpoints, but has occurred in tests of randomly generated sector maps.

Assuming the gridpoints are not so badly aligned as to induce infinite loops in the sector search process, the worst case number of sectors that will be visited during the visibility walk is approximately \( \max_{k \in \{1, \ldots, K\}} N_k \), the largest grid dimension. The correct sector for a point \( x \) can be identified in many fewer guesses if the initial guess is fairly close to the truth. When using the method of endogenous gridpoints in a model with risk, the interpolated function must be evaluated at several (or even many) nearby points to numerically integrate out the risk; these “clustered” points should be evaluated consecutively, with the true sector for the preceding point used as the initial guess for the following point. The initial guess for the first point evaluated in each cluster can be set as the index of the endogenous gridpoint currently being solved for; whether solving a finite or infinite horizon model, the grids for successive periods begin to converge after a few iterations. Using both of these techniques usually results in only a few sectors visited in the visibility walk, and often just one.

While non-linear grid interpolation can be used to approximate a function at points outside the convex hull of the gridpoints (as described in section 3.2), this extrapolation should only be used for points that are reasonably close to the interpolated region. Consider the two dimensional case in Figure 12: the blue point is only somewhat outside the closest sector, with relative coordinates of approximately \((\alpha, \beta) = (0.5, 1.5)\), while the red point is very far away and has relative coordinates \((0.2, 4)\). Even if the true function is fairly linear in the extrapolated region, the interpolated value for the red point will be inaccurate: it is so far away from the sector that the “left” and “right” boundaries have switched positions. That is, it is interpreted as having a low value of \(\alpha\) even though it is closer to the upper right vertex than upper left vertex of the closest sector. This flaw will not arise when solving a model using the method of endogenous gridpoints unless the distribution of shocks is very wide (or a very fine discrete approximation to an unbounded shock distribution is used), but can occasionally occur when using the solved model in simulations. However, even potentially flawed extrapolation is an improvement over other interpolation techniques for irregular collections of gridpoints, which either have ill defined extrapolations (Voronoi and Delaunay) or tend to “explode” outside the convex hull.

As with linear interpolation on an ordinary rectilinear grid, the approximation generated by non-linear grid interpolation will be most accurate when the function is roughly linear. While selecting the grid to be more dense at lower state values (with high function curvature) can improve accuracy, this issue can also be addressed by adjusting the function itself to be more linear. In particular, the value function of a dynamic optimization problem can be
“de-curved” by compounding it with the inverse utility function:\(^{24}\) \(W_t(x) \equiv u^{-1}(V_t(x))\); the value of \(W_t(x)\) at each gridpoint are stored for the interpolated approximation rather than \(V_t(x)\). As the degree of curvature of the value function is often similar to the curvature of the single period utility function, \(W_t(x)\) is often a relatively linear function that can be well approximated with non-linear grid interpolation.

D Derivations

The Bellman form of the individual’s problem in (5) is a very slight jump forward, incorporating part of the solution of the problem, albeit a trivial one. Recalling that \(\theta\) is the suppressed or implicit shock that determines survival, the “basic” Bellman equation here is:

\[
V_t(m_t, h_t) = \max_{c_t, \eta_t} u(c_t) + \beta \int V_{t+1}(m_{t+1}, h_{t+1})d\tilde{F}(w, \delta, \theta) \tag{26}
\]

s.t. \(m_t - c_t - \eta_t \geq 0, m_{t+1} = (m_t - c_t - \eta_t)R + wh_{t+1}, h_{t+1} = (h_t + f(\eta_t))(1 - \delta)\).

The RHS of this equation depends discontinuously on \(\theta\), and can be rewritten as:

\[
V_t(m_t, h_t) = \max_{c_t, \eta_t} u(c_t) + \beta \left( \int s(h_{t+1})V_{t+1}(m_{t+1}, h_{t+1})dF(w, \delta) + \int (1 - s(h_{t+1}))V_{t+1}(0, 0)dF(w, \delta) \right). \tag{27}
\]

Death is modeled as setting money and health capital to zero, so that the individual neither generates income or has bank balances to consume, so that his only available choice is to consume zero; his continuation value is \(V_{t+1}(0, 0) = u(0) \cdot (T - t) = 0\). The second term inside the parentheses is thus simply zero, yielding the form of the problem given in (5).\(^{25}\)

The first derivatives of the value function in (8) are reached by taking the derivative of the Bellman equation in (5) while applying the “envelope principle”. For money, this yields:

\[
V^{m}_t(m_t, h_t) = \beta R \int s(h_{t+1})V^{m}_{t+1}(m_{t+1}, h_{t+1})dF(w, \delta) = u'(\hat{c}_t(m_t, h_t)). \tag{27}
\]

Marginal value with respect to health can be found by using (9):

\[
V^{h}_t(m_t, h_t) = \beta \int (1 - \delta) \left[ s'(\cdot)V_{t+1}(\cdot) + s(\cdot) (wV^{m}_{t+1}(\cdot) + V^{h}_{t+1}(\cdot)) \right]dF(w, \delta) = \frac{u'(\hat{c}_t(m_t, h_t))}{f'(\hat{\eta}_t(m_t, h_t))}. \tag{28}
\]

\(^{24}\) In models with state dependent utility or multiple inputs to \(u(\cdot)\), the choice of “inverse utility function” is more complex but analogues do exist in many models.

\(^{25}\) While this transformation is obvious, it is included here to reinforce the point that discrete “exit” shocks are incorporated in the general framework, even if they are not expressed that way.