Lecture Notes

Dynamic Programming in Continuous Time with Adaptive Sparse Grids

Andreas Schaab
Columbia Business School

Allen T. Zhang
Harvard

May 2022

Outline

Part 1: *sparse grids*

- Hat functions and interpolation
- Hierarchical basis functions $\longrightarrow$ *critical for grid adaptation and dynamic programming*
- Regular and adaptive sparse grids

Part 2: *dynamic programming*

- Setting the stage: finite-difference methods on uniform grids
- What goes wrong on non-uniform grids?
- A generalized finite-difference method for sparse grids
- A value function iteration algorithm on adaptive sparse grids
- Use cases and applications

Hat Functions & Interpolation

Overview

- Our object of interest is f (solution of some model)

  f is in some function space $\mathcal{V}$

- We want to represent / approximate $\hat{f} \approx f$ on some grid G

- Intuitively, **sparse grids** are about "picking good grid points"

  We want to formalize questions like :

  - Is G "better" than some other grid?
  - Should we add/remove a grid point?

- What is the "value" of a grid point?

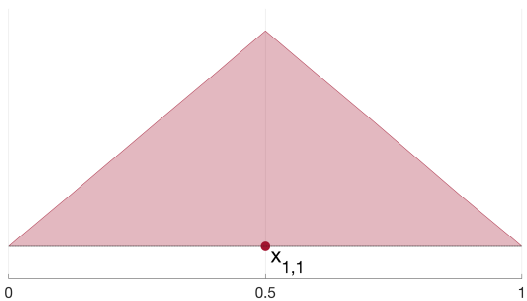
  - How much does it contribute to accuracy of our approximation $\hat{f} \approx f$?
  - $\hat{f}$ on G is a vector $\implies$ how to compare vector $\hat{f}$ and function f?
  - Need to be able to evaluate $\hat{f}$ at points *not on the grid* $\implies$ **interpolation**

- Today: Focus on linear interpolation $\implies$ we need **hat functions**

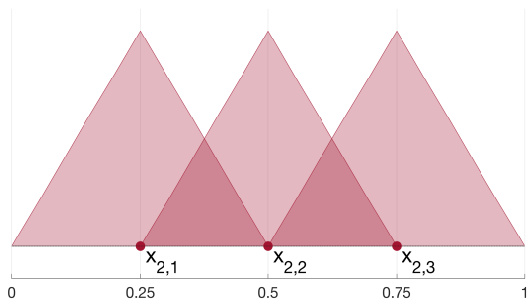
Hat Functions

- With $d = 1$, set domain $\Omega = [0, 1]$
- We call l the **level** and set $h = 2^{-l}$ (called **dyadic** grid structure)
- We define $x_{l,i} = i \cdot 2^{-l}$ as a set of **nodes**, for $i \in I_l$, where $I_l = \{i \in \mathbb{N} \mid 1 \leq i \leq 2^l - 1\}$ is an **index set**
- Hat functions when $d = 1$

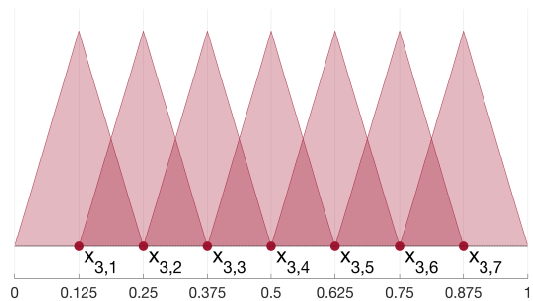
$$\phi_{l,i}(x) = \phi\left(\frac{x - ih_l}{h_l}\right), \qquad \text{where } \phi(x) = \begin{cases} 1 - |x| & \text{for } x \in [-1, 1] \\ 0 & \text{otherwise} \end{cases}$$



(a) $l = 1$ and $I_1 = \{1\}$



(b) $l = 2$ and $I_1 = \{1, 2, 3\}$



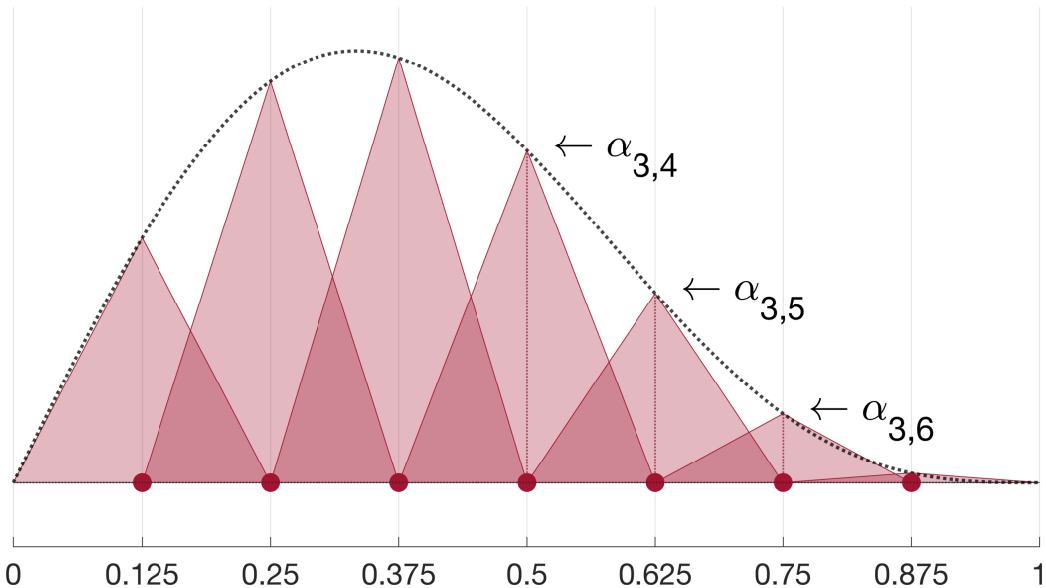
(c) $l = 3$ and $I_1 = \{1, \ldots, 7\}$

Interpolation

- Each node $x_{l,i}$ in grid G associated with a hat function $\phi_{l,i}(x)$
- We can fit function $f(x)$ on grid G via

$$f(x) \approx \sum_{i \in I_l} \alpha_{l,i} \phi_{l,i}(x)$$

  where $\alpha_{l,i} = f(x_{l,i})$ are the **function values**
- We can use $\{\phi_{l,i}(x)\}$ as basis functions $\implies$ **linear interpolation**

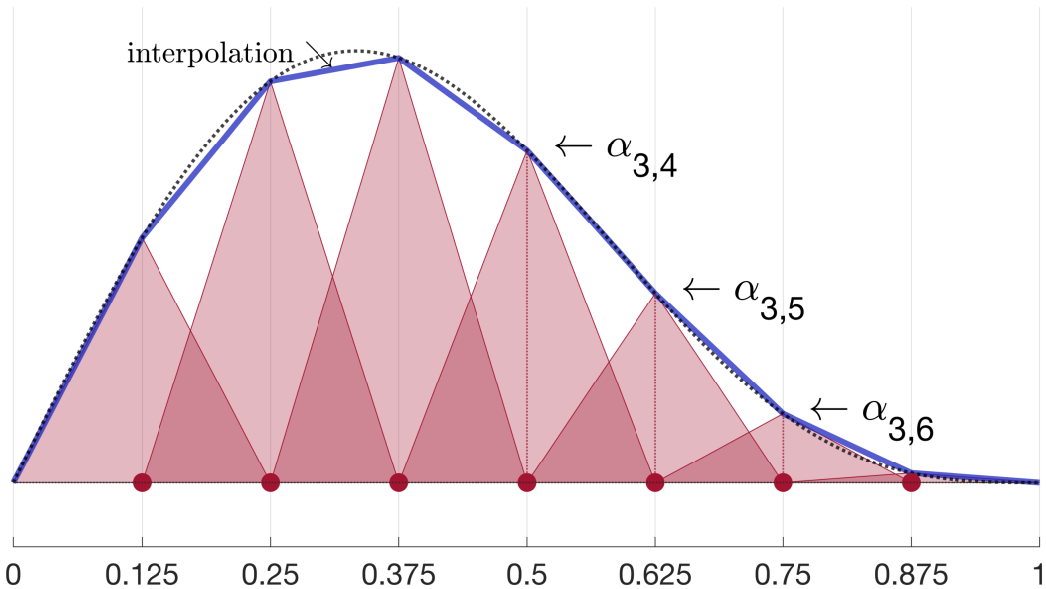


Interpolation

- Each node $x_{l,i}$ in grid G associated with a hat function $\phi_{l,i}(x)$
- We can fit function $f(x)$ on grid G via

$$f(x) \approx \sum_{i \in I_l} \alpha_{l,i} \phi_{l,i}(x)$$

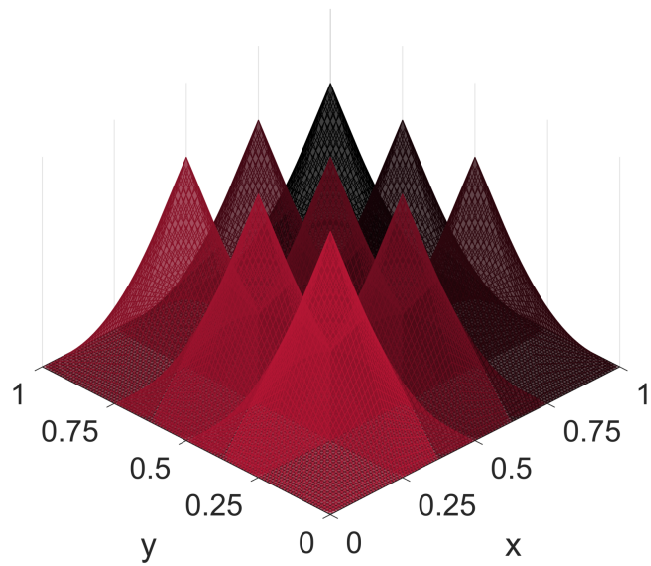
  where $\alpha_{l,i} = f(x_{l,i})$ are the **function values**
- We can use $\{\phi_{l,i}(x)\}$ as basis functions $\implies$ **linear interpolation**



Higher dimensions: $d > 1$

- We can extend everything easily to $d > 1$ by using tensor products
- Use multi-indices $\boldsymbol{l}$, with $\boldsymbol{h} = 2^{-\boldsymbol{l}}$, and $\boldsymbol{i} \implies \boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}} = \{x_{l_1,i_1}, \ldots, x_{l_d,i_d}\}$

$$\phi_{\boldsymbol{l},\boldsymbol{i}}(x) = \prod_{k=1}^{d} \phi_{l_k,i_k}(x_k), \quad x = (x_1, \ldots, x_d)$$

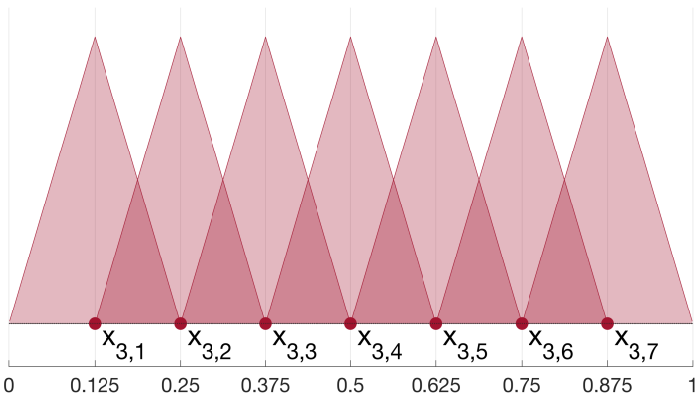


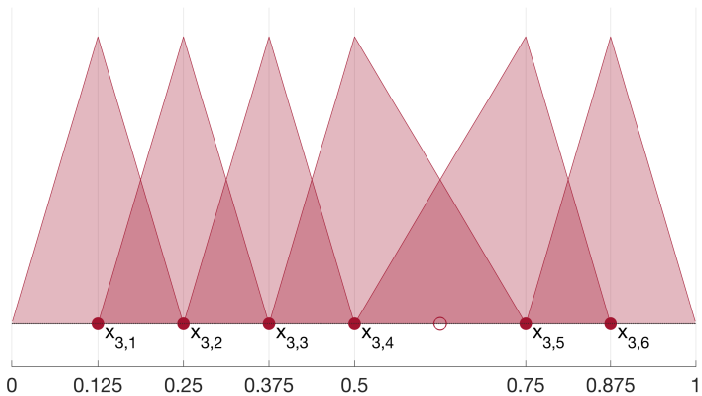
Hierarchical Basis

Hierarchical Basis: Motivation

- Recall our overarching goal:
  - → Which grid points have largest contribution?
  - → How do we systematically add these and remove others?

- There are two problems with the hat function basis introduced above:
  1. Basis functions are not invariant to grid structure
  2. Hard to characterize and compute marginal contribution of a basis function



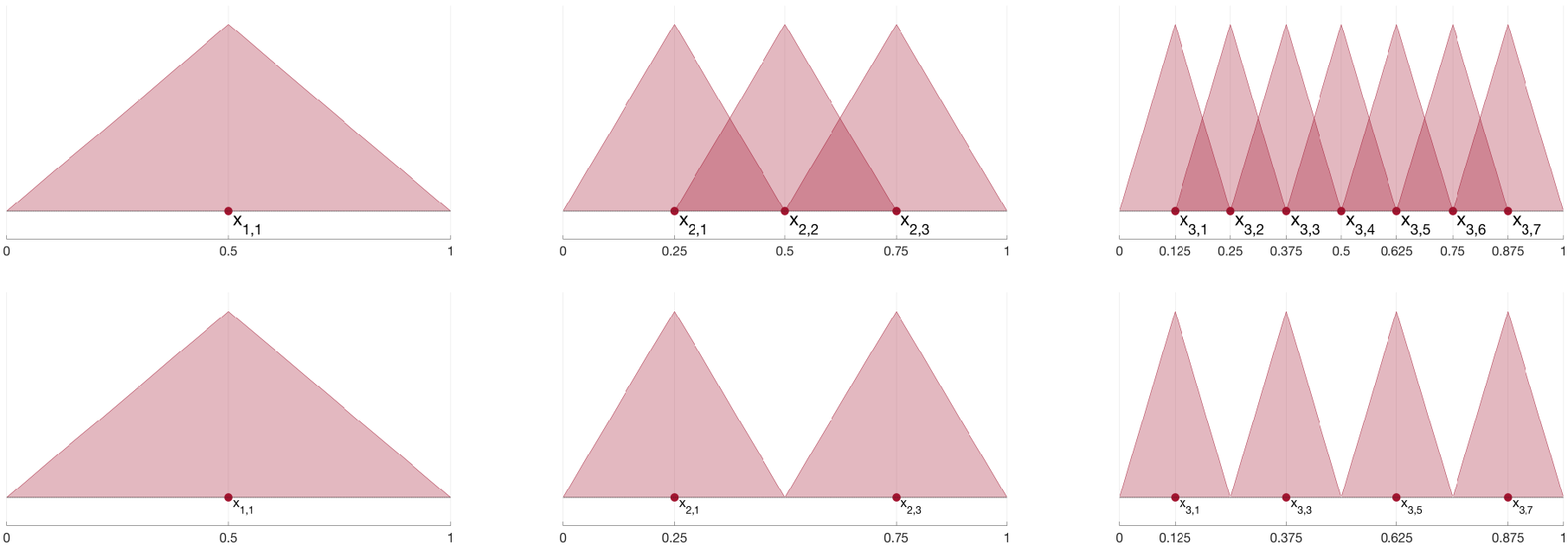


Next: **hierarchical basis functions** have these desirable properties

- Basis functions independent from each other $\Longrightarrow$ invariant as we adapt grid
- Contribution of higher-level "**hierarchical**" basis functions strictly decays

Hierarchical Basis: Definition

- Introduce the **hierarchical basis**: $\left\{\phi_{l,i}(x) : i \in I_l^{\mathrm{H}} \text{ and } x \in \Omega\right\}$
- We define $I_l^{\mathrm{H}} = \{i \in \mathbb{N} \mid 1 \leq i \leq 2^l - 1 \text{ and } i \textbf{ odd}\}$ as the **hierarchical index set**



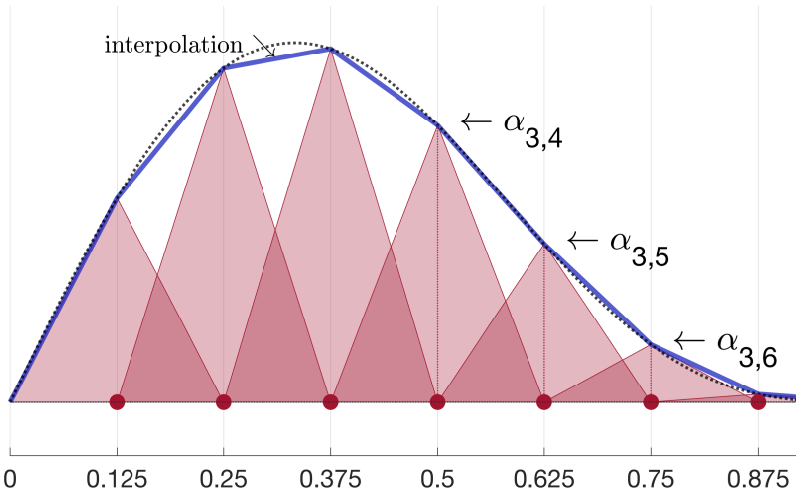
Remarks:

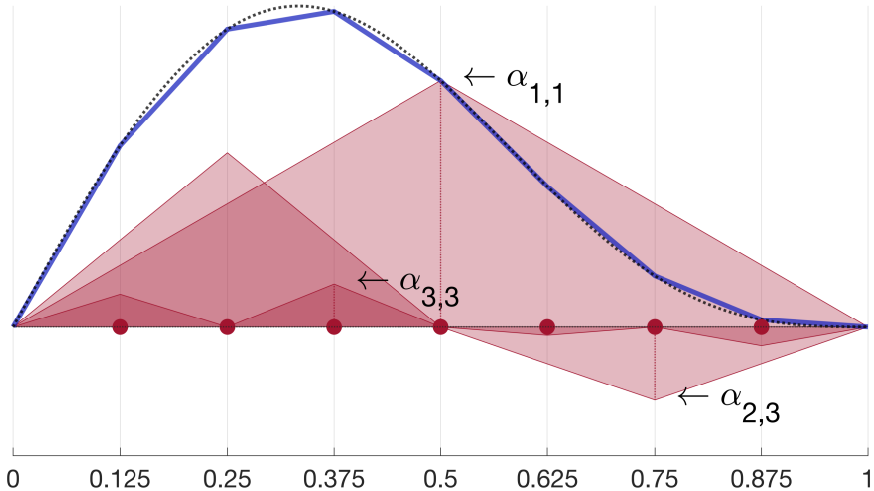
- These are now "disjoint" $\implies$ don't change as we add / remove
- "Hierarchical" because **multi-level**: higher levels don't depend on lower levels
- Definition $d > 1$ as before using tensor product

How to Implement the Hierarchical Basis?

- Want to approximate $f(x)$ on grid G of level l:

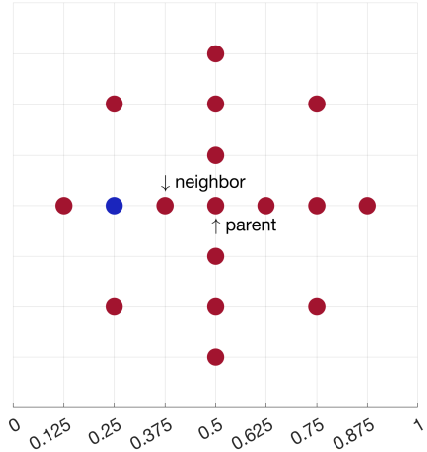
$$f_l(x) = \underbrace{\sum_{i \in I_l} \alpha_{l,i} \phi_{l,i}(x)}_{\text{nodal representation}} = \underbrace{\sum_{k \le l} \sum_{i \in I_k^{\mathrm{H}}} \alpha_{k,i}^{\mathrm{H}} \phi_{k,i}(x)}_{\text{hierarchical representation}}$$





- Both span the same "approximation space" (of piecewise linear interpolants)
- **Q:** $\alpha_{l,i} = f(x_{l,i})$ are simply the function values. $\alpha_{l,i}^{\mathrm{H}}$ are the **hierarchical surpluses**. How do we get one from the other?

Hierarchization



- Introduce some useful notation:
  - $\mathcal{N}_j^{\pm}(\boldsymbol{x}_{l,i})$: right / left neighbor of $\boldsymbol{x}_{l,i}$ in dim j
  - $\mathcal{H}_j^{\pm}(\boldsymbol{x}_{l,i})$: right / left hierarchical parent of $\boldsymbol{x}_{l,i}$ in dim j
- Intuitively, hierarchical surplus $\boldsymbol{\alpha}_{l,i}^{\mathrm{H}}$ answers the question: How much does grid point $x_{l,i}$ contribute *residually* to local function approximation?
- Without grid point $x_{l,i}$, local function approximation is simply linear interpolation:

$$\frac{1}{2}f(\mathcal{H}_1^l(x_{l,i})) + \frac{1}{2}f(\mathcal{H}_1^+(x_{l,i}))$$

- So to get the added contribution of fitting the function at $x_{l,i}$, we take:

$$\boldsymbol{\alpha}_{l,i}^{\mathrm{H}} = H_{x_{l,i}}^1 \circ f(x_{l,i}) = f(x_{l,i}) - \frac{1}{2}f(\mathcal{H}_1^l(x_{l,i})) - \frac{1}{2}f(\mathcal{H}_1^+(x_{l,i}))$$

- We call operation H **hierarchization**: function values $\mapsto$ hierarchical surplus

Why is the Hierarchical Basis so Important?

1. Hierarchical basis is "multi-level"

$\Longrightarrow$ *When adding / removing a grid point, remaining basis functions and hierarchical surpluses unaffected*

2. Hierarchical surpluses decay with their level

- Contribution of hierarchical basis functions decreasing in their level $\boldsymbol{l}$:

$$|\alpha_{l,i}^{\mathrm{H}}| \leq C \cdot 2^{-d} \cdot 2^{-2\,|\boldsymbol{l}|_1}$$

- We can read off a grid point's contribution directly from hierarchical surplus $\alpha_{l,i}^{\mathrm{H}}$

3. Hierarchization operator H (or matrix $\boldsymbol{H}$) will be critical to solve differential equations (dynamic programming) on sparse grids

Sparse Grids

Sparse Grids

Recap: We introduced hierarchical basis and showed

1. basis functions invariant in grid structure
2. hierarchical surpluses decay with level

Motivation for **sparse grids**:

- So what determines the **efficiency** of a grid or value of a grid point?

  Cost: # degrees of freedom / grid points

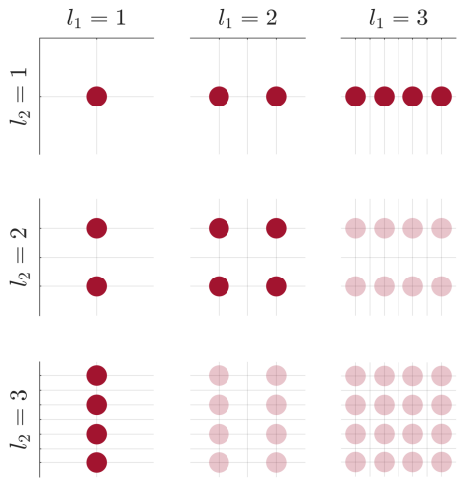
  Benefit: How much does adding a grid point contribute to function approximation?

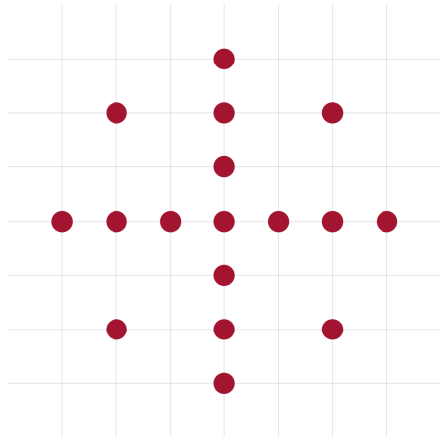
- **Grid points that contribute most are those with largest hierarchical surpluses**
- Applied math literature: sparse grids emerge from optimization problem over active hierarchical spaces / grid points

  max accuracy given # of grid points, or min # grid points given desired accuracy

Regular Sparse Grids

Intuition:





- In $d = 2$, tensor product construction of grid with $\boldsymbol{l} = (l_1, l_2)$
- Uniform grid of level $\boldsymbol{l}$: tensor product of all "1D grids" associated with $\boldsymbol{l}$
- **Regular sparse grid** of level $l^{\max}$: $\sum_d l_d \leq l^{\max} + d - 1$
- Cost (# inner grid points): $|\mathcal{V}_n^{\mathrm{RSG}}| = \mathcal{O}(2^n \cdot n^{d-1})$ **v.s.** $\mathcal{O}(2^{n \cdot d})$

Adaptive Sparse Grids

- So far: regular sparse grids approximate $f \in \mathcal{V}$ with $f_n^{\text{RSG}} \in \mathcal{V}_n^{\text{RSG}}$

$$f_n^{\text{RSG}} = \sum_{|\boldsymbol{l}|_1 \leq n+d-1} f_{\boldsymbol{l}}$$

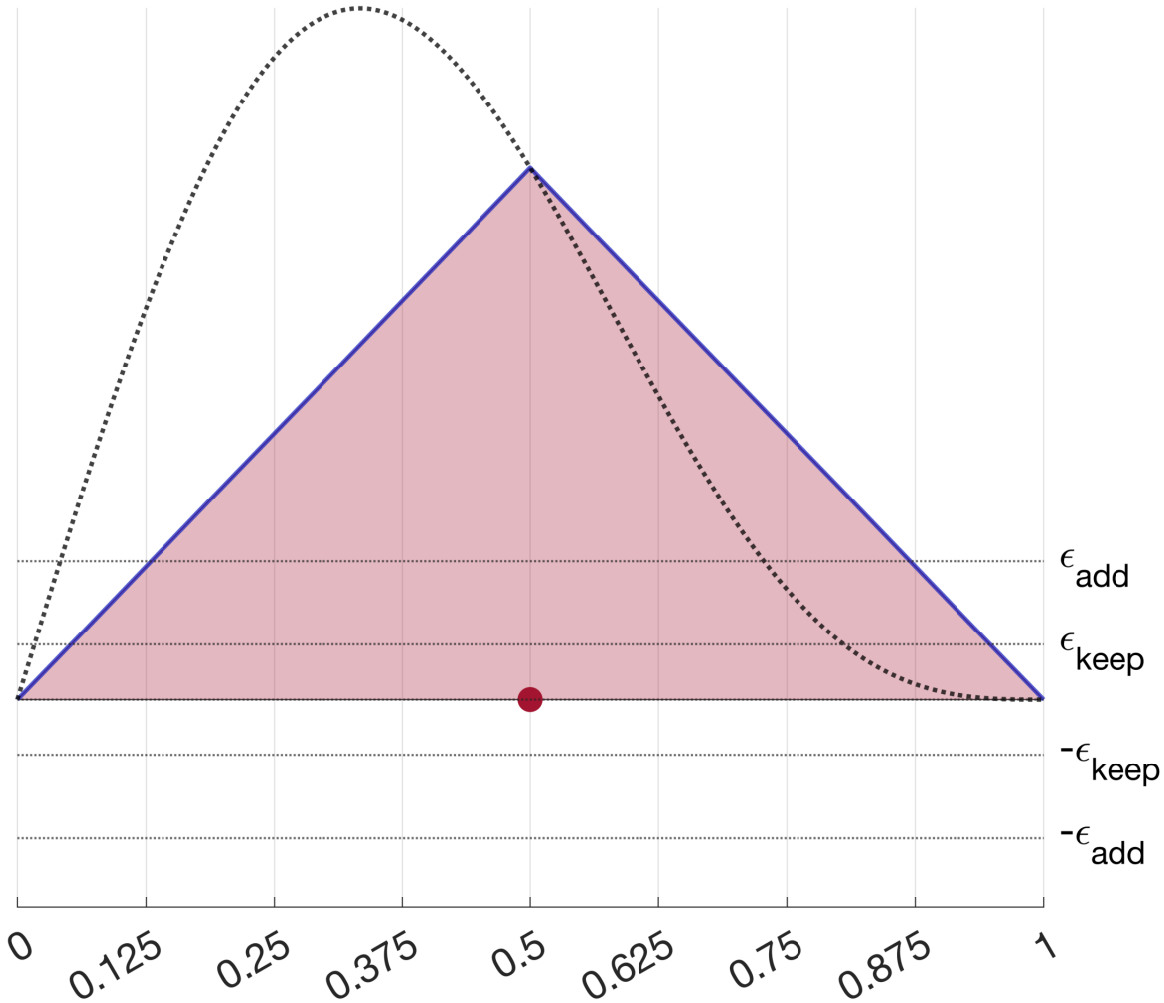
- Now: introduce adaptive grid refinement
- Key idea: hierarchical basis tells us directly about *local residual error*
  $\Longrightarrow$ hierarchical surplus at point $\boldsymbol{x}_{l,i}$ specifies *residual* contribution of basis function at $\boldsymbol{x}_{l,i}$
- Adaptation criterion: compute $\boldsymbol{\alpha}_{l,i}^{\text{H}} = \boldsymbol{H}\boldsymbol{\alpha}_{l,i}$ and

$$\text{add the children of node } \boldsymbol{x}_{l,i} \text{ if: } \quad |\boldsymbol{\alpha}_{l,i}^{\text{H}}| > \epsilon^{\text{add}} \tag{1}$$

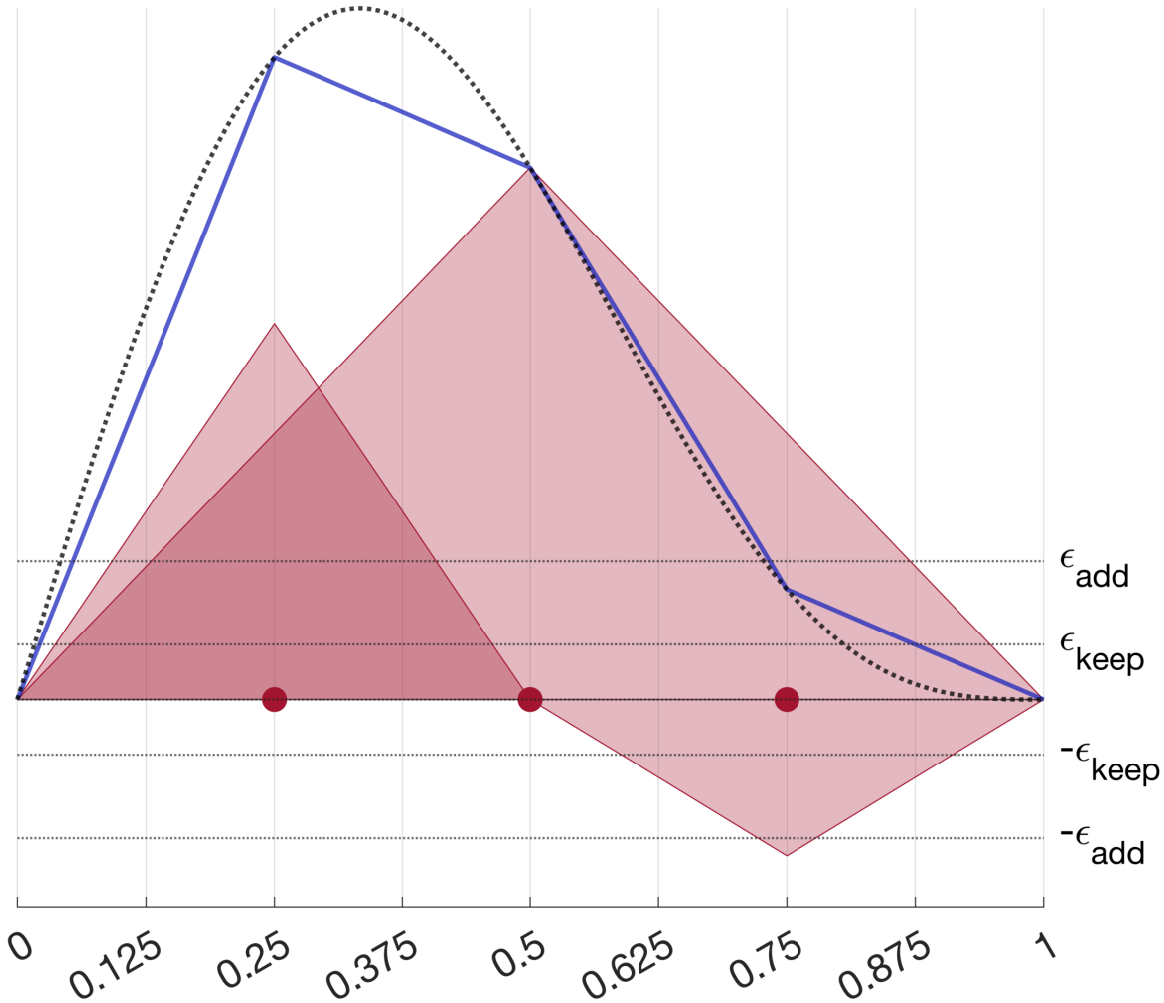
$$\text{keep node } \boldsymbol{x}_{l,i} \text{ if: } \quad |\boldsymbol{\alpha}_{l,i}^{\text{H}}| > \epsilon^{\text{keep}}, \tag{2}$$

  for some $0 < \epsilon^{\text{keep}} < \epsilon^{\text{add}}$

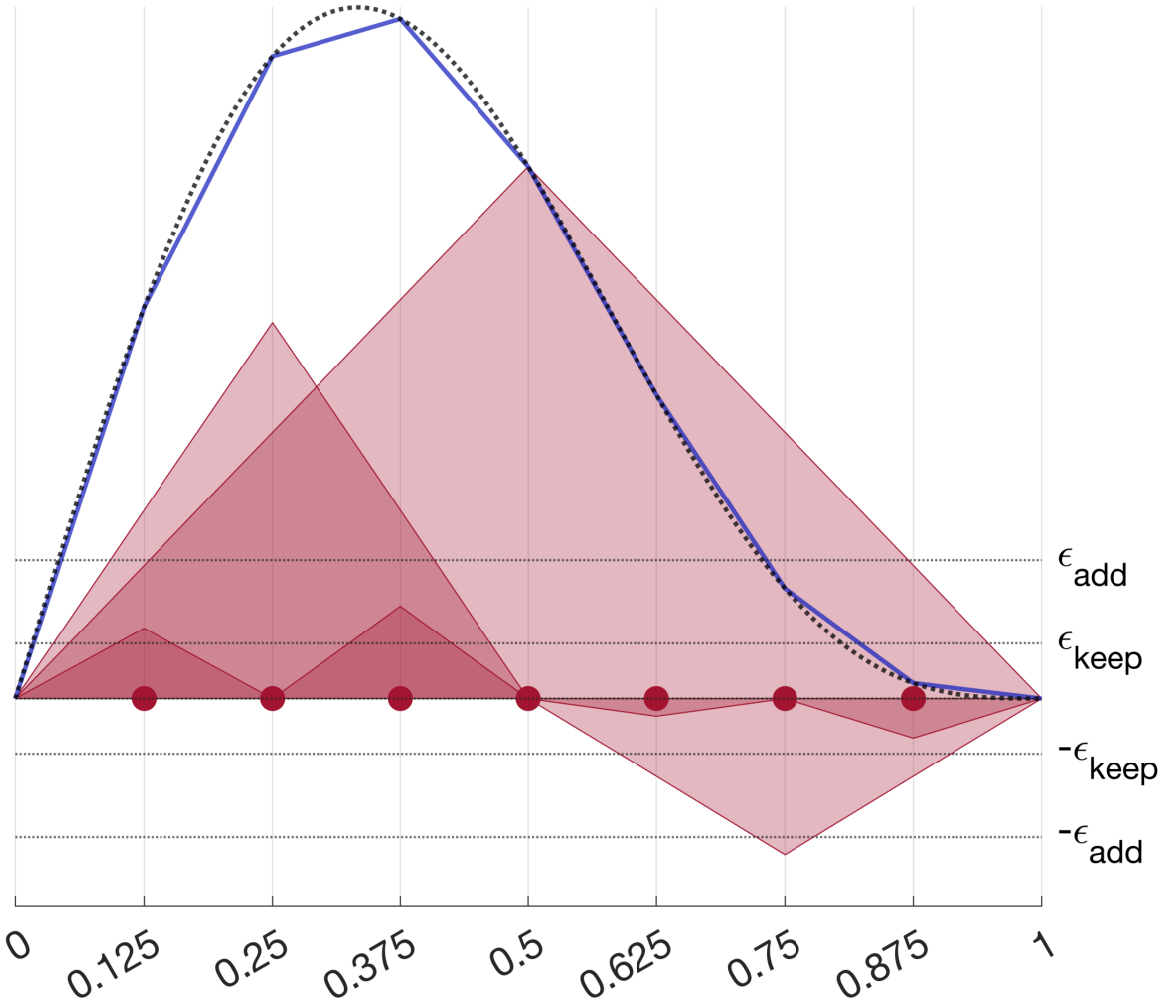
Adaptive Sparse Grids



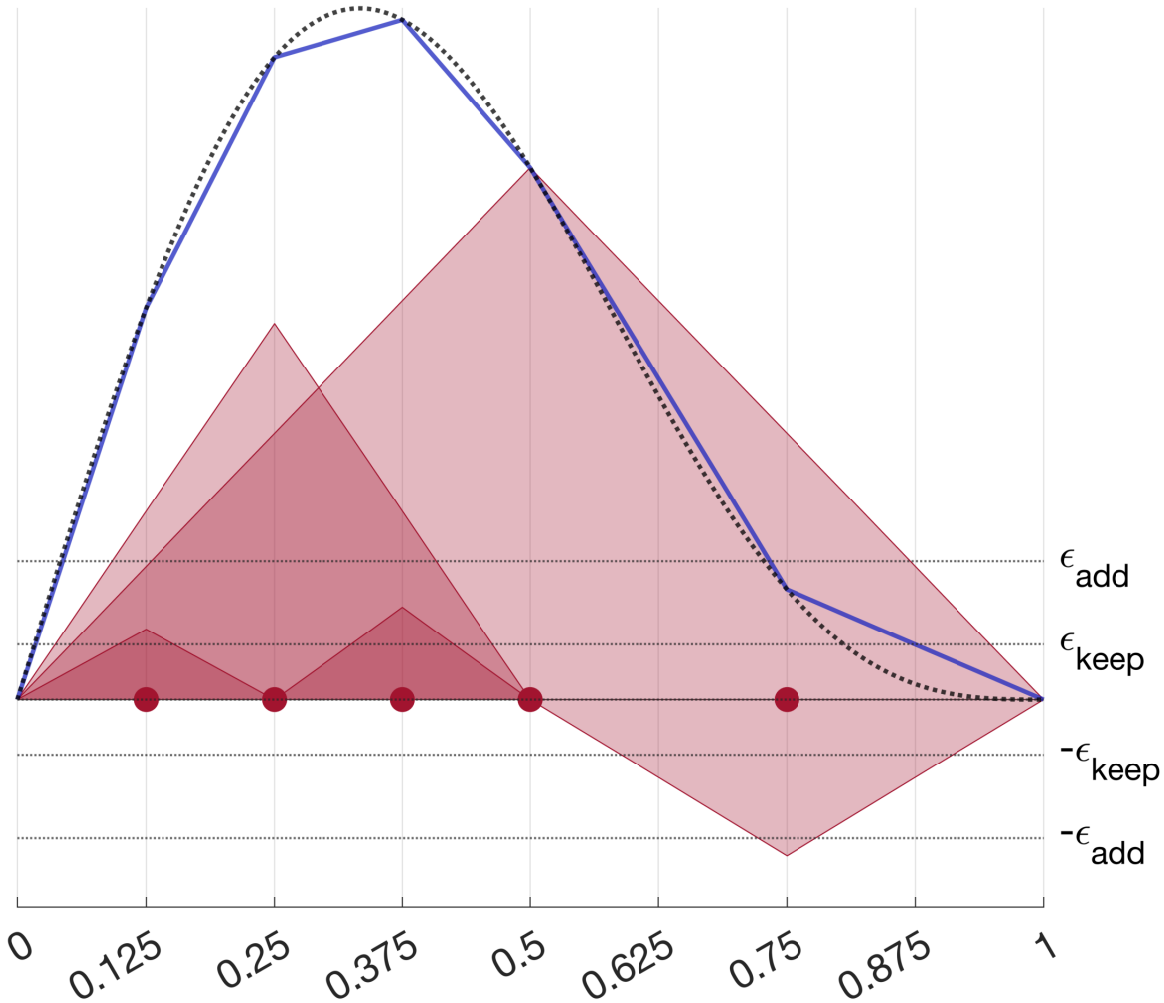
Adaptive Sparse Grids



Adaptive Sparse Grids



Adaptive Sparse Grids



Dynamic Programming:
Uniform Grids

The Neoclassical Stochastic Growth Model

- Time is continuous, $t \in [0, \infty)$
- Representative household operators production technology: $F(k_t, z_t)$
- State of the economy (k_t, z_t) evolves as:

$$dk_t = (F(k_t, z_t) - c_t - \delta k_t)dt \tag{3}$$
$$dz_t = -\theta z_t dt + \sigma dB_t \tag{4}$$

- Household preferences:

$$V(k_0, z_0) = \max_{\{c_t\}_{t \geq 0}} \mathbb{E}_0 \int_0^\infty e^{-\rho t} u(c_t) dt \tag{5}$$

- Given initial state (k_0, z_0), our objective is to solve program (5) subject to (3) and (4) for value function $V(k, z)$ over *state space*

$$\mathcal{X} = \left\{ (k, z) : k \in [0, \bar{k}] \text{ and } z \in [\underline{z}, \bar{z}] \right\}$$

The Neoclassical Stochastic Growth Model

- Easy to show: value function in (5) solves Hamilton-Jacobi-Bellman equation

$$\rho V(k,z) = u(c(k,z)) + \Big(F(k,z) - c(k,z) - \delta k\Big) V_k(k,z) - \theta z V_z(k,z) + \frac{\sigma^2}{2} V_{zz}(k,z) \quad (6)$$

  for all $(k,z) \in \mathcal{X}$ in the interior of $\bar{\mathcal{X}}$, using shorthand $V_k(k,z) = \frac{\partial}{\partial k} V(k,z)$

- Solving HJB (6) directly requires resolving $c(\cdot)$ from FOC $u'(c(k,z)) = V_k(k,z)$
  $\implies$ *results in a highly non-linear PDE*

- We transform non-linear PDE (6) into sequence of linear equations in $\{V^n\}$ using **semi-implicit time-marching scheme** *LeVeque (2007), Achdou et al. (2022)*

$$\frac{V^{n+1} - V^n}{\Delta t} + \rho V^{n+1} = u\Big(c^n\Big) + \Big(F - c^n - \delta k\Big) V_k^{n+1} - \theta z V_z^{n+1} + \frac{\sigma^2}{2} V_{zz}^{n+1} \quad (7)$$

- Equation (7) linear in V^{n+1} using: $c^n(k,z) = (u')^{-1} V_k^n(k,z)$

- Can prove that (7) is a contraction mapping that converges to V

Discretization on Uniform Grids

- Let G be a uniform grid on $\mathcal{X}$ with grid points $x_{i,j} = (k_i, z_j)$, with mesh size $h = k_{i+1} - k_i = z_{j+1} - z_j$ for all i, j
- We look for grid function $\boldsymbol{V}^n$ on G, a $J \times 1$ vector, s.t. $V^n_{i,j} \approx V^n(k_i, z_j)$
- Discretization of (7) on grid G can be written

$$\frac{1}{\Delta t}(\boldsymbol{V}^{n+1} - \boldsymbol{V}^n) + \rho \boldsymbol{V}^{n+1} = u\Big(\boldsymbol{c}^n\Big) + \Big(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\Big)\boldsymbol{V}^{n+1}_k - \theta \boldsymbol{z}\boldsymbol{V}^{n+1}_z + \frac{\sigma^2}{2}\boldsymbol{V}^{n+1}_{zz}$$

- Derivatives are linear maps, so $\boldsymbol{V}^{n+1}_k = \boldsymbol{D}_k \boldsymbol{V}^{n+1}$ for some matrix $\boldsymbol{D}_k$
- Introduce discretized generator: $\boldsymbol{A}^n = (\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k})\boldsymbol{D}_k - \theta \boldsymbol{z}\boldsymbol{D}_z + \frac{\sigma^2}{2}\boldsymbol{D}_{zz}$

$$\Big(\rho + \frac{1}{\Delta t} - \boldsymbol{A}^n\Big)\boldsymbol{V}^{n+1} = u\Big(\boldsymbol{c}^n\Big) + \frac{1}{\Delta t}\boldsymbol{V}^n$$

Q: Which derivative matrices $\boldsymbol{D}_k$, $\boldsymbol{D}_z$, and $\boldsymbol{D}_{zz}$ yield the "correct" solution?

Finite Differences

- Derivative $V'(x)$ can be numerically approximated with $\frac{V(x+h)-V(x)}{h}$ for small h
- Recall: $\mathcal{N}_j^{\pm}(x)$ is the right/left neighbor of x
- We define the **standard finite-difference operators** as

$$D_j^+ V(x) = \frac{V(\mathcal{N}_j^+(x)) - V(x)}{|\mathcal{N}_j^+(x) - x|} = \frac{V(..., x_j + h, ...) - V(x)}{h}$$

$$D_j^- V(x) = \frac{V(x) - V(\mathcal{N}_j^-(x))}{|x - \mathcal{N}_j^-(x)|} = \frac{V(x) - V(\ldots, x_j - h, \ldots)}{h}$$

$$D_{jj} V(x) = \frac{V(\mathcal{N}_j^+(x)) - 2V(x) + V(\mathcal{N}_j^-(x))}{|\mathcal{N}_j^+(x) - x| \cdot |x - \mathcal{N}_j^-(x)|} = \frac{V(\ldots, x_j + h, \ldots) - 2V(x) + V(\ldots, x_j - h, \ldots)}{h^2}$$

- We use bold $\boldsymbol{D}_j^{\pm}$ to denote the matrix that discretizes the operator on the grid

Important Concepts in Finite-Difference Methods

For illustration: $u''(x) = f(x)$ on $x \in [0,1] \implies \frac{1}{h^2}(\boldsymbol{u}_{j-1} - 2\boldsymbol{u}_j + \boldsymbol{u}_{j+1}) = f(\boldsymbol{x}_j)$

Local Truncation Error: Replace $\boldsymbol{u}_j$ by $u(\boldsymbol{x}_j)$, i.e.,

$$\boldsymbol{\tau}_j = \frac{1}{h^2}(u(\boldsymbol{x}_{j-1}) - 2u(\boldsymbol{x}_j) + u(\boldsymbol{x}_{j+1})) - f(\boldsymbol{x}_j)$$

Consistency: A finite-difference scheme is consistent with the underlying differential equation if $\|\boldsymbol{\tau}\| \to 0$ as the grid becomes finer

Intuition: discretization "makes sense" (locally) for underlying differential equation

Stability: A finite-difference scheme $\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}$ is stable if, for all sufficiently fine grids, $\boldsymbol{A}^{-1}$ exists and and there is a grid-invariant constant C such that $\|\boldsymbol{A}^{-1}\| \leq C$

Intuition: local errors don't propagate "too much"

Convergence: A finite-difference scheme is convergent if the global error $\to 0$ as the grid becomes finer

Monotonicity: Roughly, $\boldsymbol{A}\boldsymbol{u} - \boldsymbol{f}$ at node i is decreasing in $\boldsymbol{u}_j$ for $j \neq i$

Standard FD scheme for (7) on uniform grids satisfies: **1.** consistency, **2.** stability, and **3.** monotonicity $\implies$ therefore converges

Dynamic Programming:
Adaptive Sparse Grids

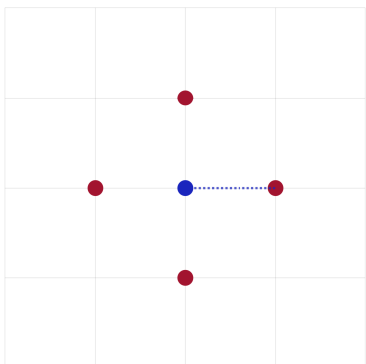
What Goes Wrong on Sparse Grids?

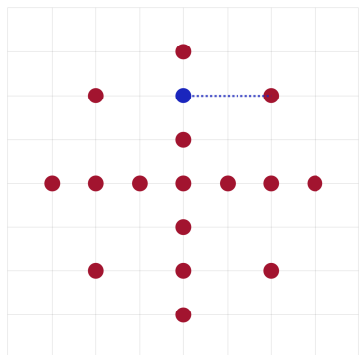
Proposition. Let G be a regular sparse grid of level $\boldsymbol{l}$ and $\boldsymbol{D}_k^{\pm}$, $\boldsymbol{D}_z^{\pm}$ and $\boldsymbol{D}_{zz}$ the standard FD matrices. Construct $\{\boldsymbol{V}^n\}$ using (7).

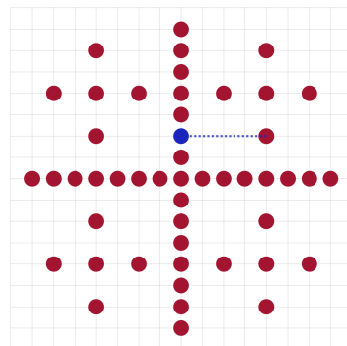
(i) In iteration n, there exist at least $d \cdot 2^{|\boldsymbol{l}|_\infty - 1}$ points with discretization error (LTE) of order $\mathcal{O}(1)$.

(ii) The discretization is not consistent.

(iii) $\{\boldsymbol{V}^n\}$ does not converge.







Generalized Sparse FD Operators

- Recall hierarchization operation (get hierarchical surplus from function values):

$$H^j_{x_{l,i}} \circ V(x_{l,i}) = V(x_{l,i}) - \frac{1}{2} V(\mathcal{H}^-_j(x_{l,i})) - \frac{1}{2} V(\mathcal{H}^+_j(x_{l,i})) \tag{8}$$

- Important notation:
  - H^j performs hierarchization at all nodes in dim j (associated matrix: $\boldsymbol{H}^j$)
  - H_j performs hierarchization in all dims *but* j (associated matrix: $\boldsymbol{H}_j$)
  - $E_j = H_j^{-1}$ performs dehierarchization in all dims *but* j (associated matrix: $\boldsymbol{E}_j$)

Definition. We define the generalized sparse finite-difference operators / matrices via

$$\boldsymbol{D}_j^{\pm,S} = \boldsymbol{E}_j \, \boldsymbol{D}_j^{\pm} \, \boldsymbol{H}_j \tag{9}$$

$$\boldsymbol{D}_{jj}^{S} = \boldsymbol{E}_j \, \boldsymbol{D}_{jj} \, \boldsymbol{H}_j \tag{10}$$

1. Hierarchize in all dimensions but j
2. Apply desired standard finite-difference stencil in dimension j
3. Dehierarchize in all dimensions but j

A Consistent FD Method on Sparse Grids

Proposition.

(i) On regular sparse grids, $\boldsymbol{D}_j^{\pm,S}$ and $\boldsymbol{D}_{jj}^{S}$ lead to consistent discretizations of the first- and second-order partial derivative operators.

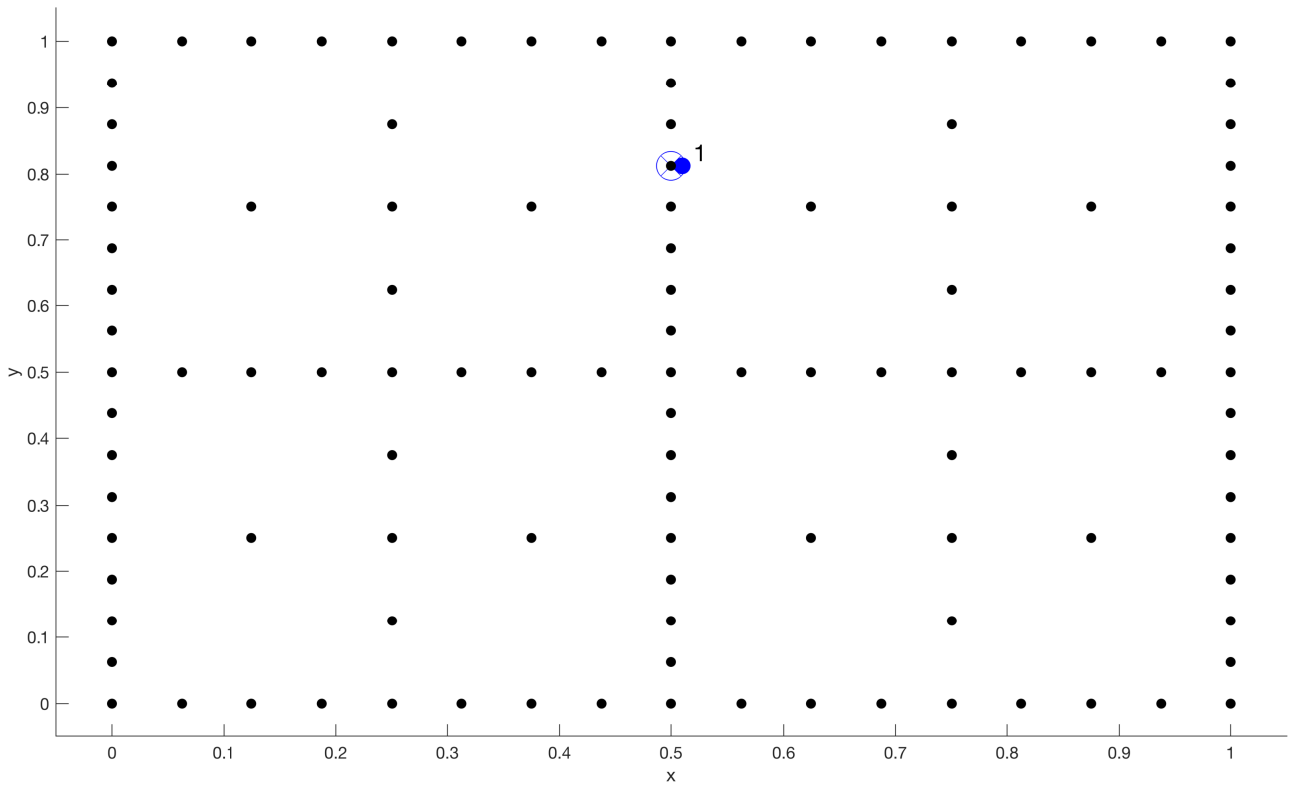
(ii) The generalized scheme

$$\frac{\boldsymbol{V}^{n+1}-\boldsymbol{V}^{n}}{\Delta t}+\rho\boldsymbol{V}^{n+1}=u\left(\boldsymbol{c}^{n}\right)+\left(\boldsymbol{F}-\boldsymbol{c}^{n}-\delta\boldsymbol{k}\right)\boldsymbol{D}_k^{S}\boldsymbol{V}^{n+1}-\theta\boldsymbol{z}\boldsymbol{D}_z^{S}\boldsymbol{V}^{n+1}+\frac{\sigma^2}{2}\boldsymbol{D}_{zz}^{S}\boldsymbol{V}^{n+1} \tag{11}$$

is a consistent discretization of the HJB equation (7).

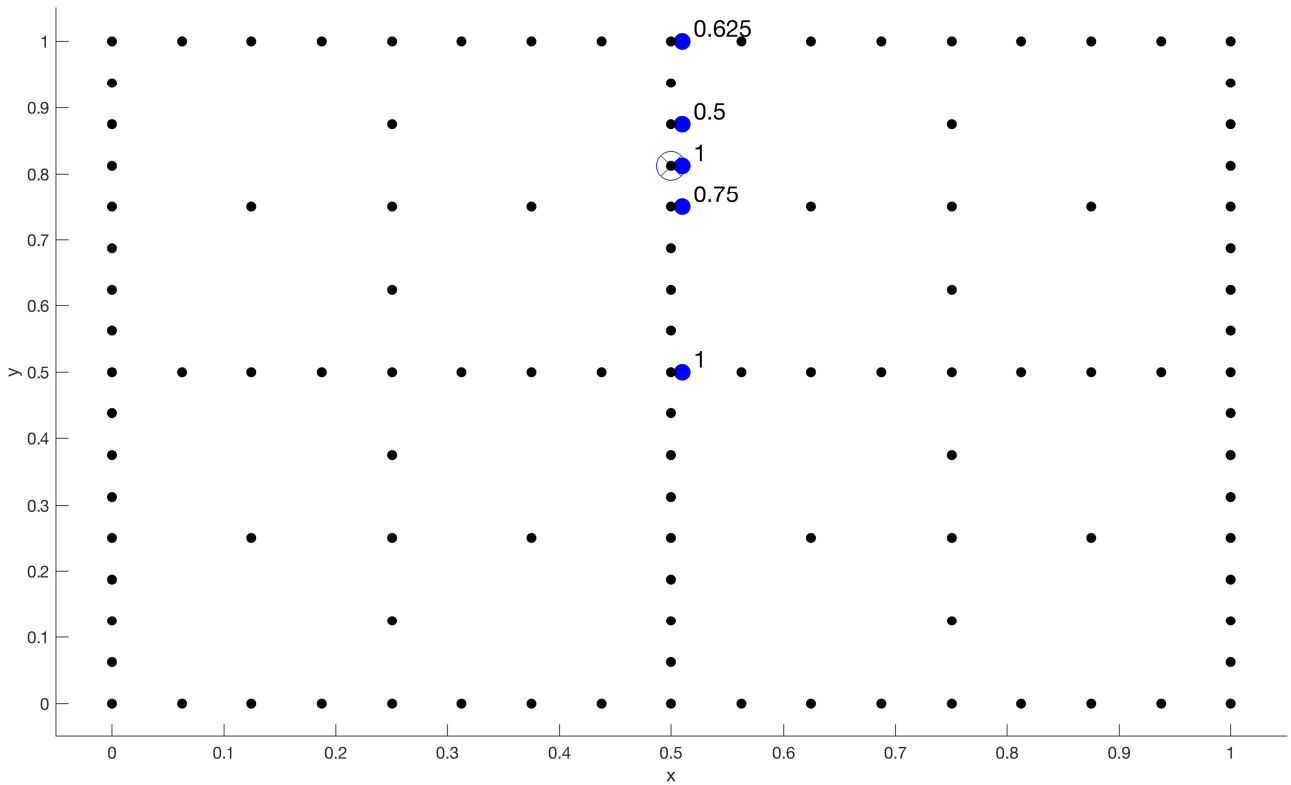
Intuition 1: Pulling Information from Off-Dimensions

$$D_x^{+,S} = E^y \; D_x^+ \; H^y$$



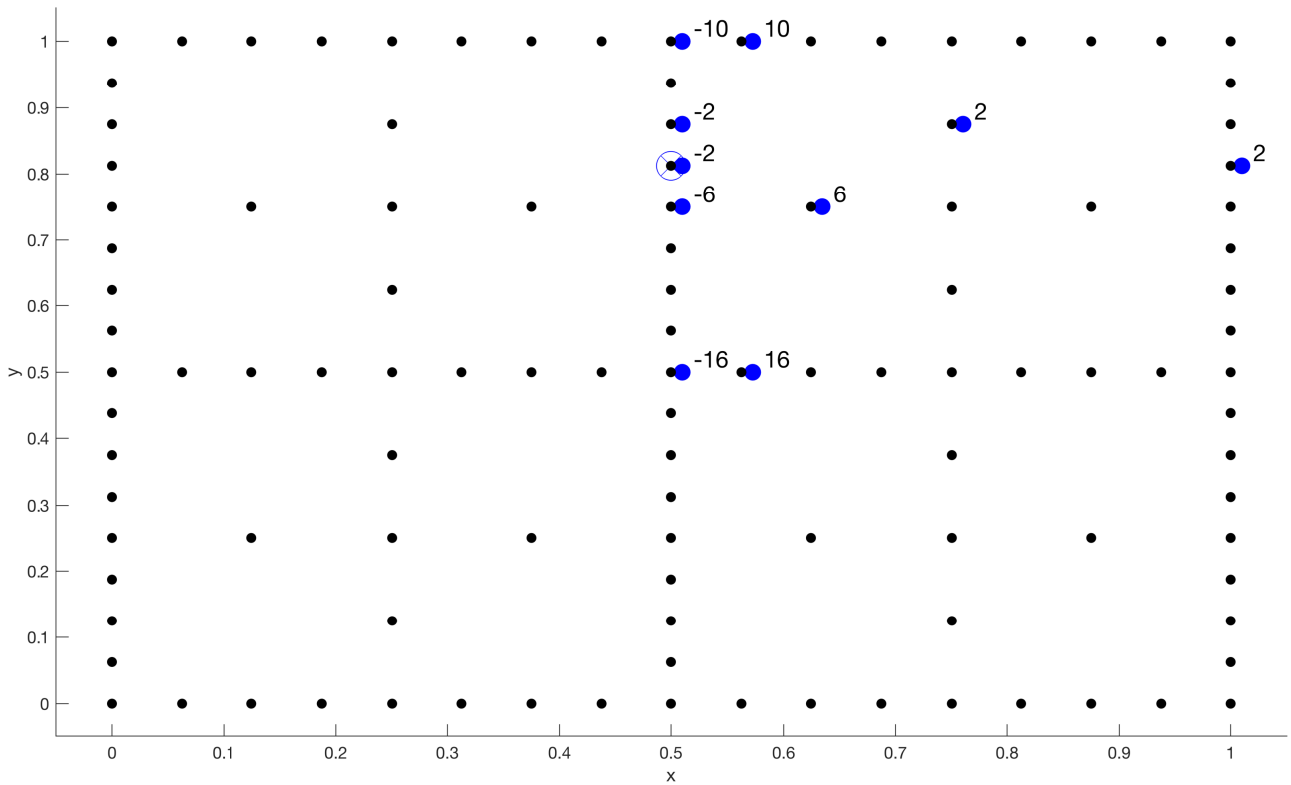
Intuition 1: Pulling Information from Off-Dimensions

$$D_x^{+,S} = E^y \; D_x^+ \; H^y$$



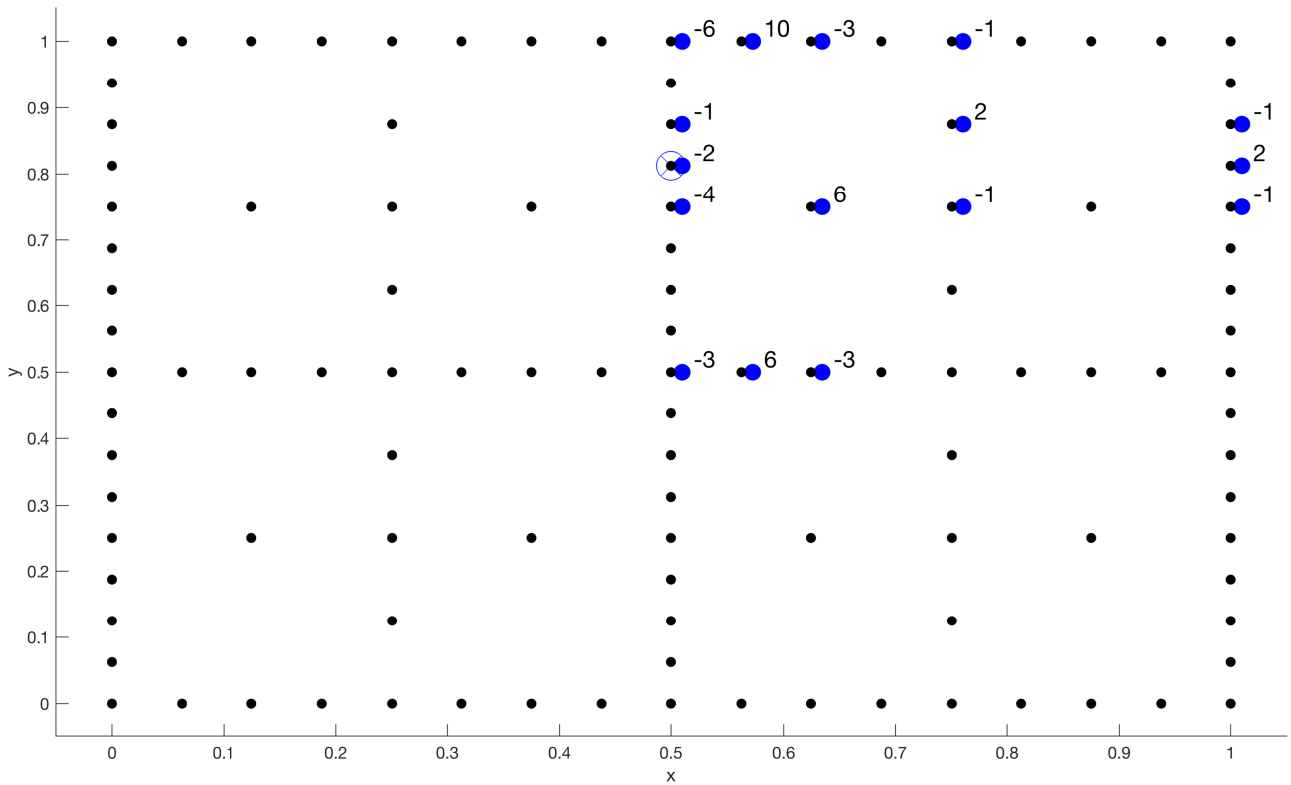
Intuition 1: Pulling Information from Off-Dimensions

$$D_x^{+,S} = E^y \; D_x^+ \; H^y$$



Intuition 1: Pulling Information from Off-Dimensions

$$D_x^{+,S} = E^y \; D_x^+ \; H^y$$



Intuition 2: Interpolation

- By definition, $\partial_{x_j} V(x) = \lim_{h \to 0} \frac{V(\ldots, x_j + h, \ldots) - V(x)}{h}$, so **any** consistent discretization of $D_j \approx \partial_{x_j}$ must satisfy

$$D_j V = \frac{V(\ldots, x_j + h, \ldots) - V(x)}{h} + \mathcal{O}(h)$$

- Intuitively: We want to interpolate V onto the point $(x_1, \ldots, x_j + h, \ldots, x_d)$ and then implement the usual stencil.

  Notice: the interpolation error itself is $\mathcal{O}(h)$!

Proposition. Define the operator that interpolates onto $\boldsymbol{x_{l,i}} + h\boldsymbol{e}_j$ via

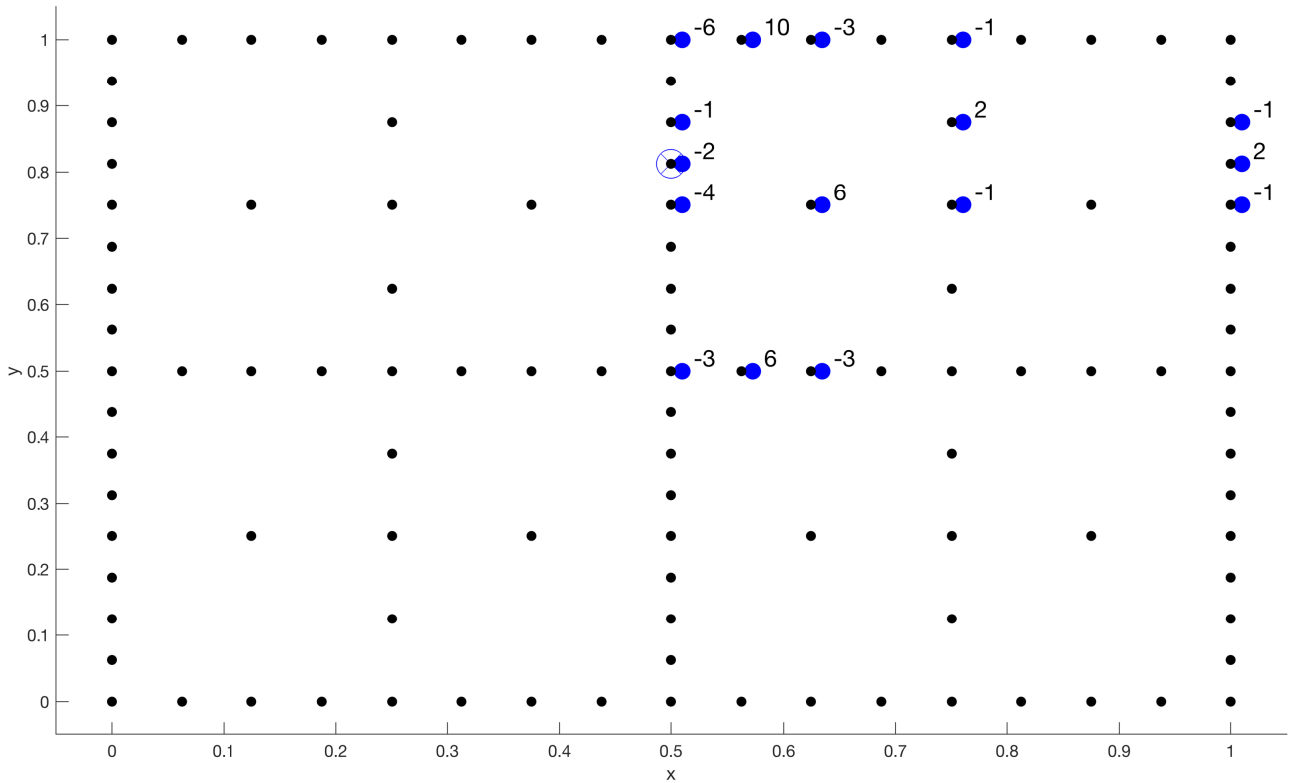
$$I^+_{j,h} \circ V(x) = V(x_1, \ldots, x_j + h, \ldots, x_d).$$

The generalized sparse finite-difference operators implement an interpolation stencil given by

$$\boldsymbol{E}_j \, \boldsymbol{D}^+_j \, \boldsymbol{H}_j = \frac{\boldsymbol{I}^+_{j,h} - \boldsymbol{I}}{h} = \frac{\boldsymbol{B}^+_{x,j}\boldsymbol{H} - \boldsymbol{I}}{h}$$

For illustration: figures › and proof sketch ›

Intuition 3: Mechanics of Taylor Expansions



- Let (x, y) be grid point with cross and consider the "problem point" on the right boundary

$$f(x+8h, y) \approx f(x,y) + f_x(x,y) \cdot 8h$$
$$f(x+8h, y+h) \approx f(x,y) + f_x(x,y) \cdot 8h + f_y(x,y) \cdot h$$
$$f(x+8h, y-h) \approx f(x,y) + f_x(x,y) \cdot 8h - f_y(x,y) \cdot h$$

- Combine with stencil: "problem point" goes away to $\mathcal{O}(h)$

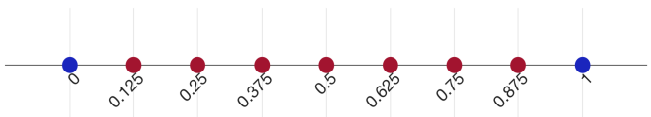
$$2 \cdot f(x+8h, y) - 1 \cdot f(x+8h, y+h) - 1 \cdot f(x+8h, y-h) \approx 0$$

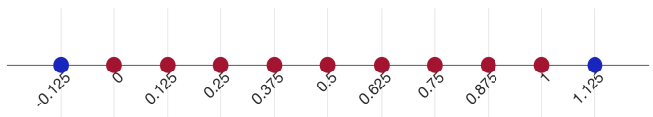
Boundary Conditions

- Consider boundary conditions (BCs) of the form

$$\mathcal{B}(x, V, V_x, V_{xx}) = 0\,, \quad \text{for } x \in \partial\bar{\mathcal{X}} \tag{12}$$

- In applied math, two broad approaches to implement BCs:
  1. Differential equation holds in interior, i.e., on x_{ij} for $i \in \{2, \ldots, I-1\}$ and $j \in \{2, \ldots, J-1\}$. BCs (12) used to substitute out for $V(\partial\bar{\mathcal{X}})$ in linear system.
  2. Differential equation holds on entire grid. Introduce hypothetical exterior ghost nodes $i \in \{0, I+1\}$ and $j \in \{0, J+1\}$. Think of BCs (12) as holding *just outside* boundary. Leads to linear system of $I \times J$ equations in $I \times J$ unknowns.





- Example: von-Neumann boundary condition $\partial_k V(k, z) = \kappa$ at $k = \underline{k}$.
  - Using approach 2, consider $(\boldsymbol{D}_k^- \boldsymbol{V})_{1j} = \frac{V_{1j} - V_{0j}}{h} \implies$ how to handle V_{0j}?
  - Use BC to substitute $V_{0j} = h\kappa + V_{1j}$ everywhere in linear system

- More generally: discretizing BC at x_{1j} yields $\mathcal{B}(x_{1j}, V_{1j}, V_{0j}) = 0$
  $\implies$ due to linearity, we can always invert and solve for V_{0j}

Boundary Conditions

- This leads to BC–adjusted FD scheme: $\boldsymbol{D}_j \boldsymbol{V} = (\boldsymbol{D}_j^{\text{int}} + \boldsymbol{D}_j^{\text{bound}})\boldsymbol{V} + \text{const}_j$
- We develop general implementation for following BCs:
  - von-Neumann: $\partial_x V(x) = \kappa$
    Examples: borrowing constraints, ...
  - Reflecting: $\partial_x V(x) = 0$
    Examples: "default" BC, exogenous earnings / productivity risk, ...
  - Dirichlet: $V(x) = \kappa$
    Examples: life-cycle, initial / terminal conditions in transition dynamics, ...
  - "Asset pricing": $\partial_{xx} V(x) = \kappa(x)\partial_x V(x)$
    Examples: portfolio choice, ...
- In each case, the idea is to solve for the exterior ghost node in terms of interior nodes, and then modify the FD matrix accordingly (*fully automated*)

Proposition.

(i) The scheme with boundary-modified standard FD matrices yields a consistent discretization of the HJB equation (7) on uniform grids.

(ii) The scheme with boundary-modified generalized sparse FD matrices yields a consistent discretization of the HJB equation (7) on regular sparse grids.

Value Function Iteration on Adaptive Sparse Grids

- So far: recursively construct $\{\boldsymbol{V}^n\} \to \bar{\boldsymbol{V}}$ on a given (regular) sparse grid G
- Now: introduce adaptive grid refinement as iterative outer step
  - Iteration k associated with grid G^k and solution $\{\boldsymbol{V}^{k,n}\} \to \bar{\boldsymbol{V}}^k$
  - Construct G^{k+1} to add grid points where approximation error $\|V - \boldsymbol{V}^k\|$ is large
- On ASG, neighbors no longer equidistant (as on RSG) $\implies$ use stencil for non-equidistant grids
- Can show:
  - Suppose grid adaptation never removes parents
  - $\boldsymbol{E}_j \boldsymbol{D}_j \boldsymbol{H}_j$ still implements "correct" interpolation stencil ($\to$ unaffected by adaptation)
  - Any adaptation scheme where interpolation error is $\mathcal{O}(h^{\text{asg}})$ is consistent in h^{asg}

Value Function Iteration on Adaptive Sparse Grids

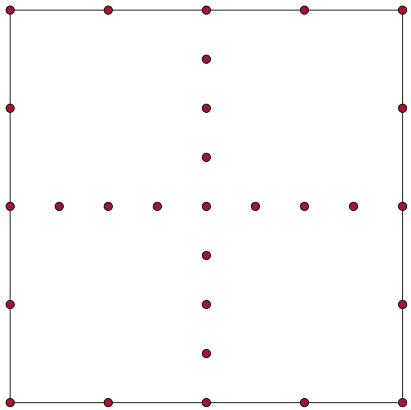
Algorithm Value Function Iteration on Adaptive Sparse Grids

Choose an index $\boldsymbol{l}^0$ and construct associated regular sparse grid G^0
Construct $\boldsymbol{H}^0$ and sparse FD operators on G^0 ▷ Equations (8), (9), and (10)
Initialize guess for the value function $\boldsymbol{V}^0$ on G^0
for $k \geq 0$ **do**
    for $n \geq 0$ **do**
        Compute $\boldsymbol{A}^{k,n}$ using sparse FD operators
        Solve linear system for $V^{k,n+1}$ ▷ Equation (11)
        Continue if $|V^{k,n+1} - V^{k,n}| > \epsilon^{\text{VFI}}$
    end
    Compute hierarchical surplus $\boldsymbol{\alpha}^{H,k} = \boldsymbol{H}^k \bar{\boldsymbol{V}}^k$
    If adaptation criterion met, adapt grid G^{k+1} ▷ Equations (1) and (2)
    Reconstruct $\boldsymbol{H}^{k+1}$ and sparse FD operators on G^{k+1} ▷ Equations (8), (9), and (10)
end

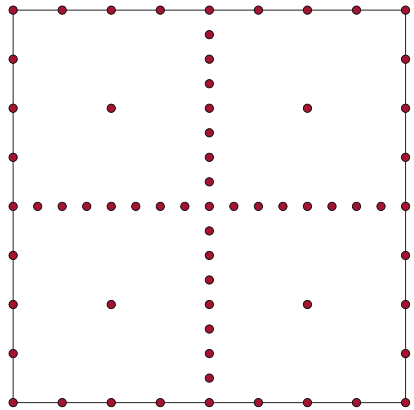
Really important: $\boldsymbol{D}^{\text{S},k}$ must be updated in k but **not** in n loop

Towards More Efficient Grid Structures

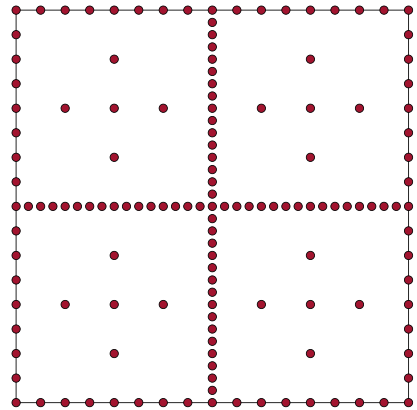
Efficient grid (l=3, n=29)



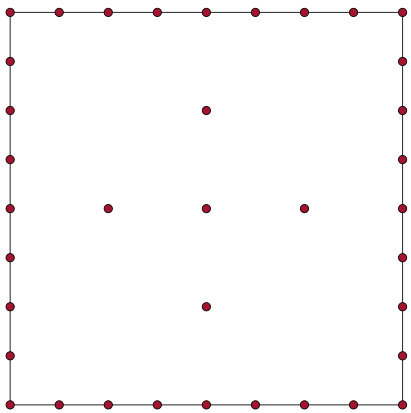
Efficient grid (l=4, n=65)



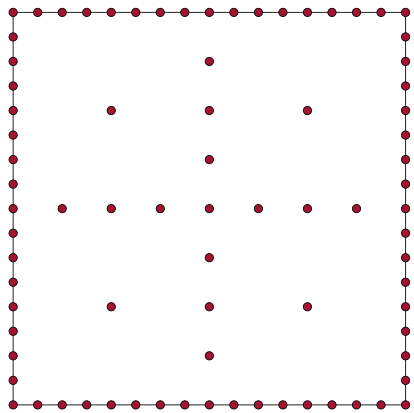
Efficient grid (l=5, n=145)



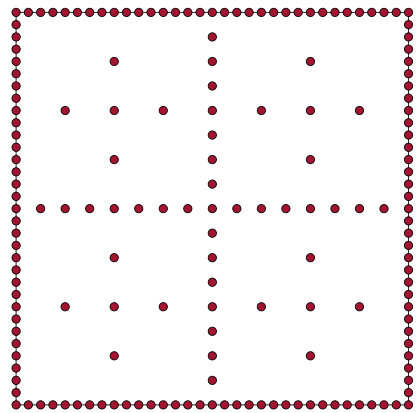
Standard grid (l=3, n=37)



Standard grid (l=4, n=81)



Standard grid (l=5, n=177)



Towards More Efficient Grid Structures

- Since we now miss boundary points, many grid points have no neighbors at all
- Again leverage interpolation idea and define modified sparse FD operators as

$$\tilde{\boldsymbol{D}}_j^{\pm,\mathrm{S}} = \boldsymbol{E}_j \, \boldsymbol{D}_j \, \boldsymbol{H}_j \, \tilde{\boldsymbol{B}}^{\mathrm{bound}}$$

$$\tilde{\boldsymbol{D}}_{jj}^{\mathrm{S}} = \boldsymbol{E}_j \, \boldsymbol{D}_{jj} \, \boldsymbol{H}_j \, \tilde{\boldsymbol{B}}^{\mathrm{bound}}$$

- What does this buy us?
  1. Projection operators used one dimension at a time (derivatives operate in 1 dimension)
     $\Longrightarrow$ constructing $\tilde{\boldsymbol{B}}^{\mathrm{bound}}$ for all dims (or constructing the associated grid) very expensive
  2. $\tilde{\boldsymbol{B}}^{\mathrm{bound}}$ only shows up when constructing FD matrices
     $\Longrightarrow$ never have to evaluate function at implied points (very expensive in VFI as matrices grow)

Proposition.

(i) On modified regular sparse grids, $\tilde{\boldsymbol{D}}_j^{\pm,\mathrm{S}}$ and $\tilde{\boldsymbol{D}}_{jj}^{\mathrm{S}}$ lead to consistent discretizations of the first- and second-order partial derivative operators.

(ii) The generalized scheme

$$\frac{\boldsymbol{V}^{n+1} - \boldsymbol{V}^n}{\Delta t} + \rho \boldsymbol{V}^{n+1} = u\left(\boldsymbol{c}^n\right) + \left(\boldsymbol{F} - \boldsymbol{c}^n - \delta \boldsymbol{k}\right) \tilde{\boldsymbol{D}}_k^{\mathrm{S}} \boldsymbol{V}^{n+1} - \theta \boldsymbol{z} \tilde{\boldsymbol{D}}_z^{\mathrm{S}} \boldsymbol{V}^{n+1} + \frac{\sigma^2}{2} \tilde{\boldsymbol{D}}_{zz}^{\mathrm{S}} \boldsymbol{V}^{n+1}$$

is a consistent discretization of the HJB equation (7).

Iterative Methods for Linear Systems

- In Matlab, `mldivide` ("backslash") is very powerful to solve linear systems *exactly*
- For large linear systems ($> 10,000$), **iterative methods** become useful
- In Matlab, `gmres` works well
- Open question: efficient pre-conditioners for macro dynamic programming problems

Use Cases and Applications

Use Cases for Sparse Grids in Economics

- Adaptive sparse grids are powerful in many economic settings:
  1. dynamic models with high-dimensional state spaces
  2. occasionally-binding constraints
  3. non-linearities, non-convexities, and kinks
  4. life-cycle and OLG models
  5. free boundary problems
  6. discrete-choice models
  7. dynamic oligopoly

- SparseEcon repository at: `https://github.com/schaab-lab/SparseEcon`

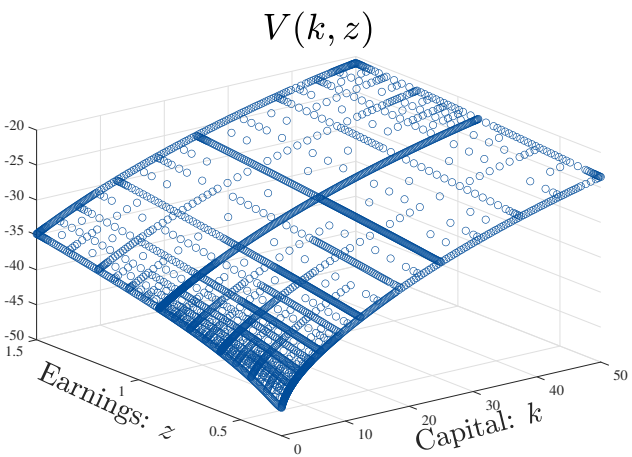
Occasionally-binding constraints

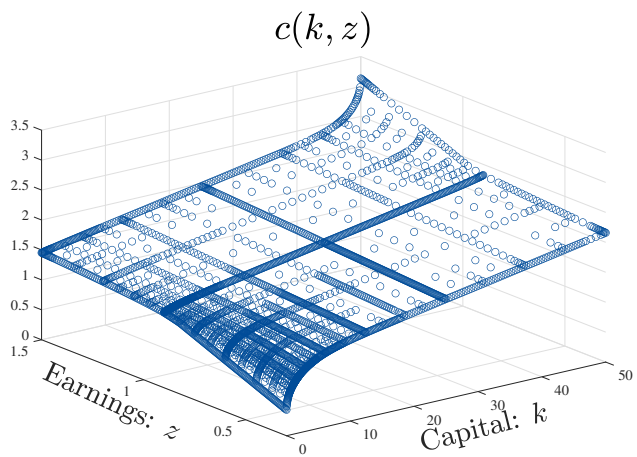
- Consider Aiyagari (1994) model with occasionally-binding borrowing constraint
- Households consume (c_t), accumulate capital (k_t), and face earnings risk (z_t):

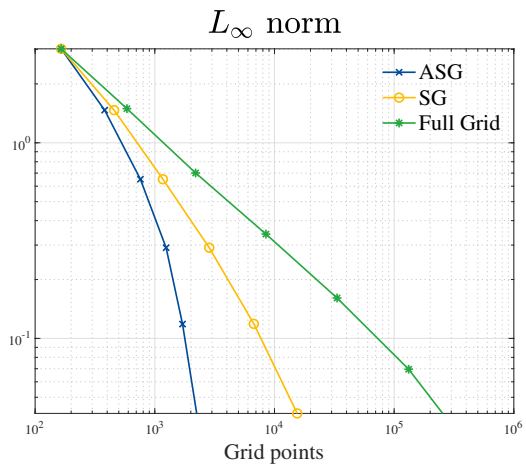
$$dk_t = rk_t + e^{z_t} - c_t$$
$$dz_t = -\theta z_t dt + \sigma dB_t$$

- Borrowing constraint: $k_t \geq \underline{k}$
- Value function solves the HJB equation

$$\rho V(k,z) = u(c(k,z)) + \Big(rk + e^z - c(k,z)\Big)V_k(k,z) - \theta z V_z(k,z) + \frac{\sigma^2}{2}V_{zz}(k,z)$$







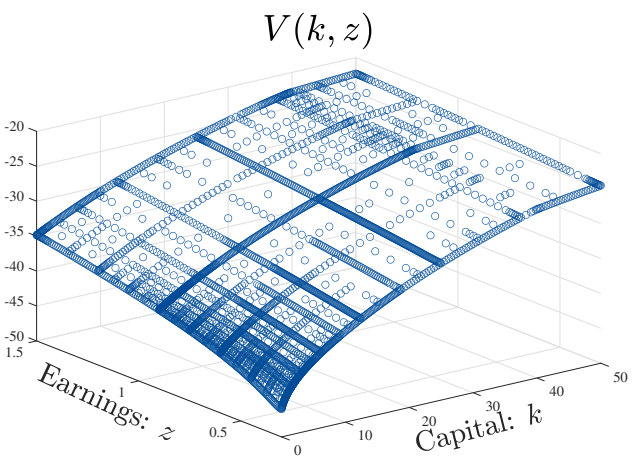
Non-Linearities, Non-Convexities, and Kinks

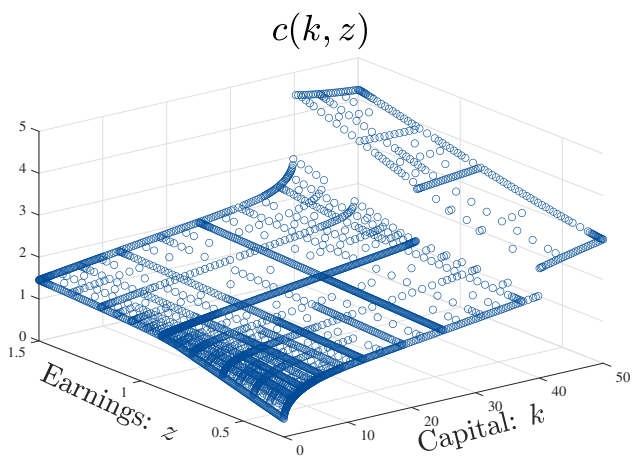
- Consider same Aiyagari (1994) model with capital tax

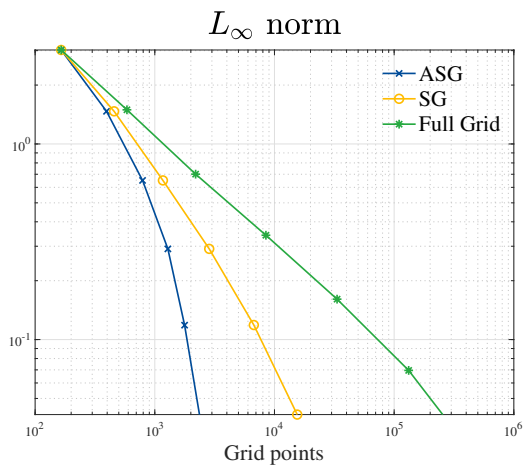
$$dk_t = \Big(1 - T(k_t)\Big) r k_t + e^{z_t} - c_t$$

- Households pay constant tax rate on financial income once wealth exceeds k^*

$$T(k_t) = \begin{cases} 0 & \text{if } k \leq k^* \\ \kappa & \text{if } k > k^* \end{cases}$$

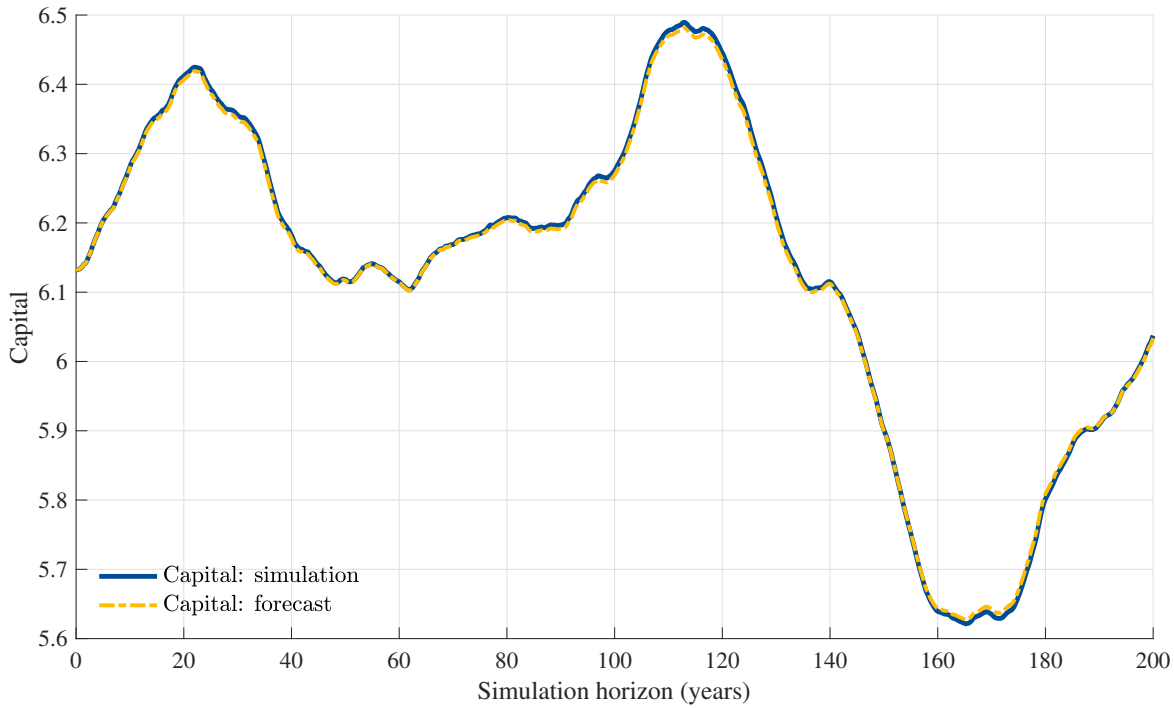






Breaking Curse of Dimensionality

- Consider Krusell-Smith (1998) model: Aiyagari + aggregate TFP shocks Z_t
- State space becomes $(k, z, Z, g(\cdot))$, where $g(k, z)$ is distribution (∞-dimensional)
- Parametrize $g_t(k, z) \approx \hat{g}_t(k, z) = F(\alpha_t)(k, z)$ with 22 Chebyshev polynomials
- Solve dynamic programming problem on 25-dimensional state space (k, z, Z, α)



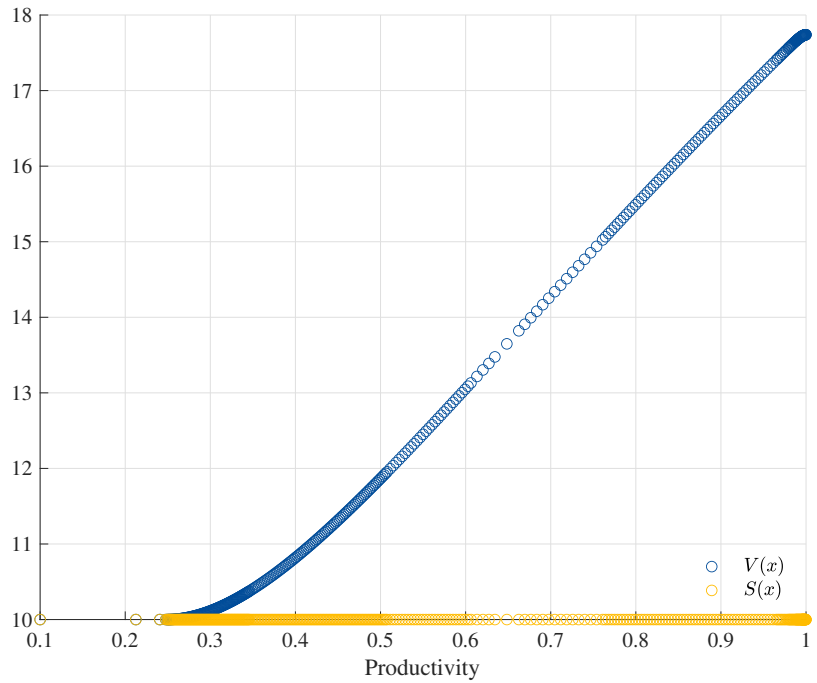
Free Boundary Problems

- Firm value is

$$V(x) = \max_{\tau} \mathbb{E}_0 \int_0^{\tau} e^{-\rho t} u(x_t) dt + e^{-\rho \tau} S(x_\tau), \quad \text{where } dx_t = \mu(x_t)dt + \sigma(x_t)dW_t$$

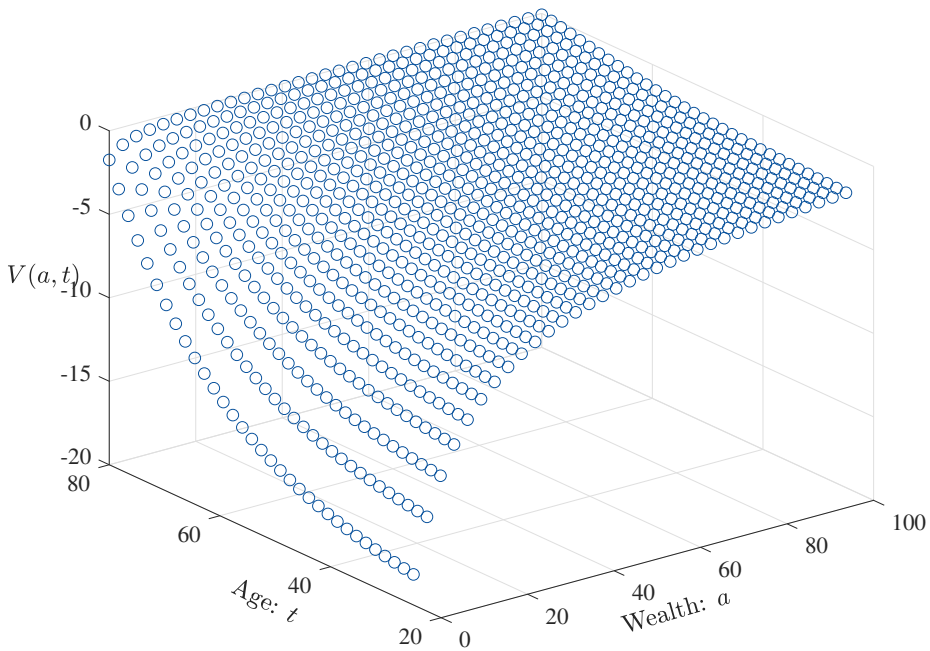
- τ is optimal time of exit or plant shut-down
- Value function solves HJB variational inequality

$$\rho V(x) = \max \left\{ u(x) + \mu(x) V_x(x) + \frac{\sigma(x)^2}{2} V_{xx}(x),\ \rho S(x) \right\}$$

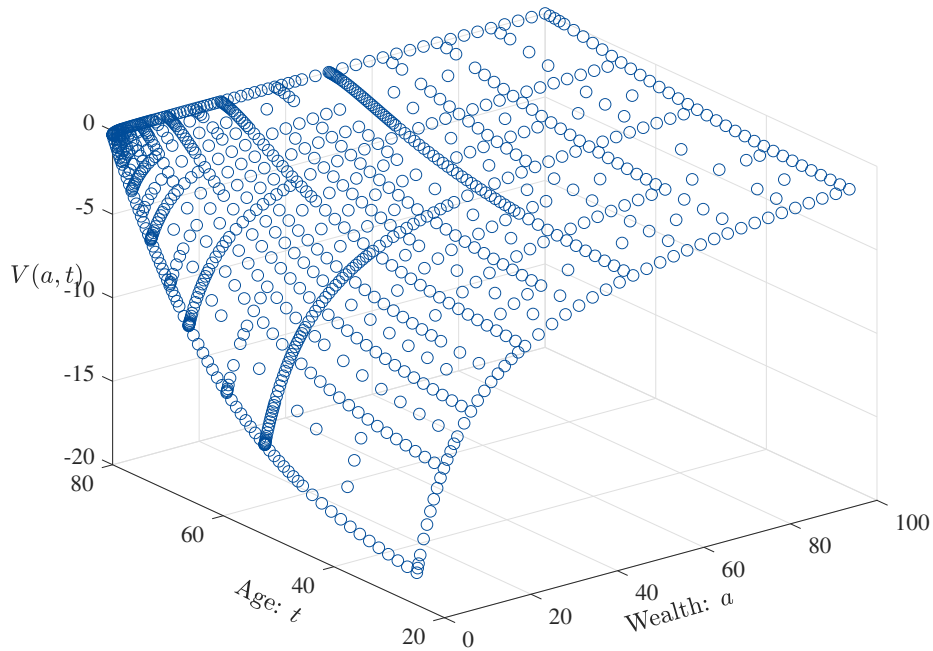


Life-Cycle / OLG

- Again Aiyagari (1994) model but with life-cycle, age $t \in [25, 80]$
- Treat age just like usual state variable: $dt = 1 \cdot dt + 0 \cdot dB$
- Dirichlet boundary condition: $V(a, z, 80) = \epsilon_1 u(\epsilon_2 + a)$



(a) Uniform Grid



(b) Fully Adapted Sparse Grid

Thanks for having me!

Introduction

- Sparse grids can alleviate the curse of dimensionality
  - Powerful to solve high-dimensional dynamic economic models
  - Also powerful in other contexts: occasionally binding constraints, free boundaries, kinks and non-convexities, ...
- Increasingly popular in economics:
  - Earlier work: smolyak grids (basis functions with global support)
    Krueger-Kubler (2004, 2006), Judd et al. (2014), Fernández-Villaverde et al. (2015), ...
  - More recent: adaptive sparse grids (basis functions with local support) for discrete-time models
    Brumm-Scheidegger (2017), Usui (2019), Cao et al. (2020)
  - In continuous time, dynamic programming $\implies$ differential equations (subtle on sparse grids)
    Garcke-Ruttscheidt (2019), Schaab (2020), Schaab-Zhang (2022)
- Builds on active literature in applied math since 1990s

  Zenger (1991), Griebel (1991), Bungartz (1992), Schiekofer (1998), Griebel (1998), Bungartz-Griebel (2004), Garcke (2006), Plueger (2010), ...

Today

1. Intro to (adaptive) sparse grids *based on* *Bungartz-Griebel (2004), Brumm-Scheidegger (2017), ...*
2. Use them for continuous-time dynamic programming *based on* *Schaab-Zhang (2022)*

Overview

- Intuitively, **sparse grids** are about "picking good grid points"
- But to formalize this, we have to use some function space arguments
- Object of interest is f (solution of some model) in some **function space** $\mathcal{V}$
  We will focus on space of functions with bounded mixed derivatives up to order 2
- Function space $\mathcal{V}$ is ∞-dimensional. Why? What does this mean?
  - Function $f : [0,1] \to \mathbb{R}$ like a vector with one component for each real number
  - Can we use finitely many function values to describe the function completely? No!
  - Can we use finitely many "basis functions" to describe the function completely? No!

Overview

Q: What is most efficient finite-dimensional approximation of $\mathcal{V}$ (and $f \in \mathcal{V}$)?

- Sparse grids will be the answer to this question
- Deep link between grids and approximation (function) spaces

Our strategy will proceed as follows:

1. Split the function space $\mathcal{V}$ into components
   Intuition: think of $\mathcal{V}$ as span of basis functions, then "component spaces" are span of subsets
2. Figure out how much each component space contributes to the representation of f
   Intuition: which basis functions are the most "valuable"?
3. Approximation space = finite combination of components that contribute most
   Intuition: allow only finite subset of basis functions that span $\mathcal{V}$ to be active
4. Show that combining component spaces is equivalent to picking grid points
   Intuition: will associate each grid point with hat basis function centered on grid point

Sparse grids: finite combination of grid points / component spaces that contribute most to $\mathcal{V}$

Function Spaces

- Our object of interest f is in some function space $\mathcal{V}$
- Our basis of hat functions implicitly induces an **approximation space** of piecewise linear functions:

$$\mathcal{V}_l = \text{span}\left\{\phi_{l,i}(x) \mid i \in I_l \text{ and } x \in \Omega\right\}$$

- For $l < \infty$, $\mathcal{V}_l \subset \mathcal{V}$ means that approximating f in $\mathcal{V}_l$ incurs error
  $\implies$ While $\mathcal{V}$ is ∞-dimensional, $\mathcal{V}_l$ is finite-dimensional
- There are 2 valid ways to think about **sparse grids**:
  1. picking good grid points
  2. constructing an efficient finite-dimensional approximation space for $\mathcal{V}$

Key observation: grids $\leftrightarrow$ finite-dimensional approximation spaces

$\rightarrow$ Whenever we put $f(x)$ on any grid, we solve for an interpolant in some approximation space

$\rightarrow$ We can associate any grid with the approximation space it induces (given some basis functions)

Uniform Grids

- A **uniform grid** of level $\boldsymbol{l}$ is associated with the approximation space

$$v \in \mathcal{V}_{\boldsymbol{l}}^{\mathrm{UG}} = \bigoplus_{|\boldsymbol{k}| \leq \boldsymbol{l}} \mathcal{W}_{\boldsymbol{k}} \quad \Longrightarrow \quad v(x) = \sum_{\boldsymbol{i} \in I_{\boldsymbol{l}}} \alpha_{\boldsymbol{l},\boldsymbol{i}} \phi_{\boldsymbol{l},\boldsymbol{i}}(x)\,, \quad \text{where } \alpha_{\boldsymbol{l},\boldsymbol{i}} = v(\boldsymbol{x}_{\boldsymbol{l},\boldsymbol{i}})$$

  $\Longrightarrow$ usual space of piecewise d-linear functions on grid with equidistant mesh size $\boldsymbol{h} = 2^{-\boldsymbol{l}}$

- Suppose $\boldsymbol{l} = (l, \ldots, l)$ and $h = 2^{-l}$ (same mesh size in each dimension)

- **Cost**: how large is $\mathcal{V}_{\boldsymbol{l}}^{\mathrm{UG}}$?

$$|\mathcal{V}_{\boldsymbol{l}}^{\mathrm{UG}}| = (2^l - 1)^d = \mathcal{O}(2^{d \cdot l}) = \mathcal{O}(h^{-d})$$

- **Benefit**: how accurate are approximations in $\mathcal{V}_{\boldsymbol{l}}^{\mathrm{UG}}$?

$$||f - f_{\boldsymbol{l}}^{\mathrm{UG}}||_\infty \leq C \cdot \frac{d}{6^d} \cdot 2^{-2l} = \mathcal{O}(h^2)$$

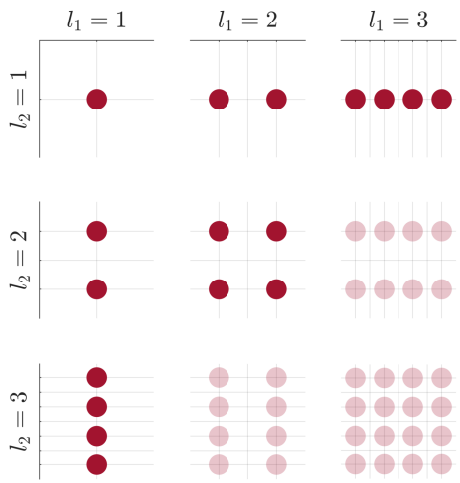
- **Curse of dimensionality**: # points to achieve accuracy $\mathcal{O}(h^2)$ grows exponentially

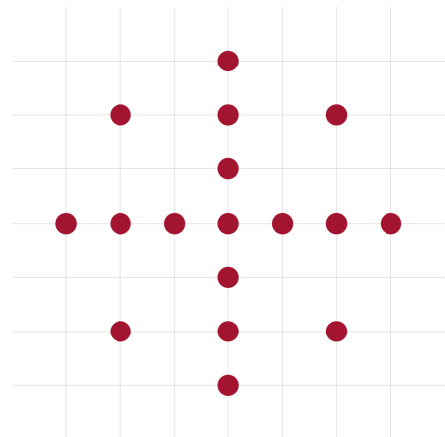
Regular Sparse Grids

Formal optimization problem:

- Associate approximation space (grid) $\mathcal{U}$ with index $\boldsymbol{I} \subset \mathbb{N}^d$ via $\mathcal{U} = \bigoplus_{\boldsymbol{l} \in \boldsymbol{I}} \mathcal{W}_{\boldsymbol{l}}$
- Given some norm: $\max_{|\mathcal{U}| \leq c} \|f - f_{\mathcal{U}}\|$, where $f_{\mathcal{U}} = \sum_{\boldsymbol{l} \in \boldsymbol{I}} f_{\boldsymbol{l}}$ for $f_{\boldsymbol{l}} \in \mathcal{W}_{\boldsymbol{l}}$
  - Again: Selecting a particular $\mathcal{W}_{\boldsymbol{l}}$ due to its importance $\leftrightarrow$ selecting associated grid points
  - Regular sparse grids emerge from an *a priori optimization*
- Under L^2 norm: $\mathcal{V}_n^{\mathrm{RSG}} = \bigoplus_{|\boldsymbol{l}|_1 \leq n+d-1} \mathcal{W}_{\boldsymbol{l}}$

Intuition:





- Cost (# inner grid points): $|\mathcal{V}_n^{\mathrm{RSG}}| = \mathcal{O}(2^n \cdot n^{d-1})$ **v.s.** $\mathcal{O}(2^{n \cdot d})$

Regular Sparse Grids 1

- More broadly, we can construct finite-dimensional approximation spaces $\mathcal{U}$ for $\mathcal{V}$ via a *subspace selection* $\boldsymbol{I} \subset \mathbb{N}^d$ and

$$\mathcal{U} = \bigoplus_{\boldsymbol{l} \in \boldsymbol{I}} \mathcal{W}_{\boldsymbol{l}}, \quad \text{with } f_{\mathcal{U}} = \sum_{\boldsymbol{l} \in \boldsymbol{I}} f_{\boldsymbol{l}}, \ f_{\boldsymbol{l}} \in \mathcal{W}_{\boldsymbol{l}}$$

- Our goal is to find the most efficient approximation space $\mathcal{U}$ in terms of some norm

$$||f - f_{\mathcal{U}}|| = \left|\left| \sum_{\boldsymbol{l}} f_{\boldsymbol{l}} - \sum_{\boldsymbol{l} \in \boldsymbol{I}} f_{\boldsymbol{l}} \right|\right| \leq \sum_{\boldsymbol{l} \notin \boldsymbol{I}} ||f_{\boldsymbol{l}}||$$

- That is: We look for some optimal finite index $\boldsymbol{I} \subset \mathbb{N}^d$ and consider the approximation space $\mathcal{U} = \bigoplus_{\boldsymbol{l} \in \boldsymbol{I}} \mathcal{W}_{\boldsymbol{l}}$
  Selecting a particular $\mathcal{W}_{\boldsymbol{l}}$ due to its importance $\leftrightarrow$ selecting associated grid point

- Important: Regular sparse grids emerge from an *a priori optimization* (for all functions in some space) while adaptive sparse grids emerge from problem-dependent optimization

Regular Sparse Grids 2

- Define local cost $c(\boldsymbol{l}) = |\mathcal{W}_{\boldsymbol{l}}| = 2^{|\boldsymbol{l}-\mathbf{1}|_1}$. Define local benefit $b(\boldsymbol{l}) = \gamma\beta(\boldsymbol{l})$ where $\beta(\boldsymbol{l})$ is upper bound for $||f_{\boldsymbol{l}}||^2$. Grid optimization: max (global) benefit subject to (global) cost = given amount of work.

- Different norms lead to differently structured **regular sparse grids**
  L^2 and L^∞ norm yield:

$$\mathcal{V}_n^{\mathrm{RSG}} = \bigoplus_{|\boldsymbol{l}|_1 \leq n+d-1} \mathcal{W}_{\boldsymbol{l}}$$

  $\Longrightarrow$ "standard sparse grids" introduced by Zenger (1991) and Bungartz (1992)

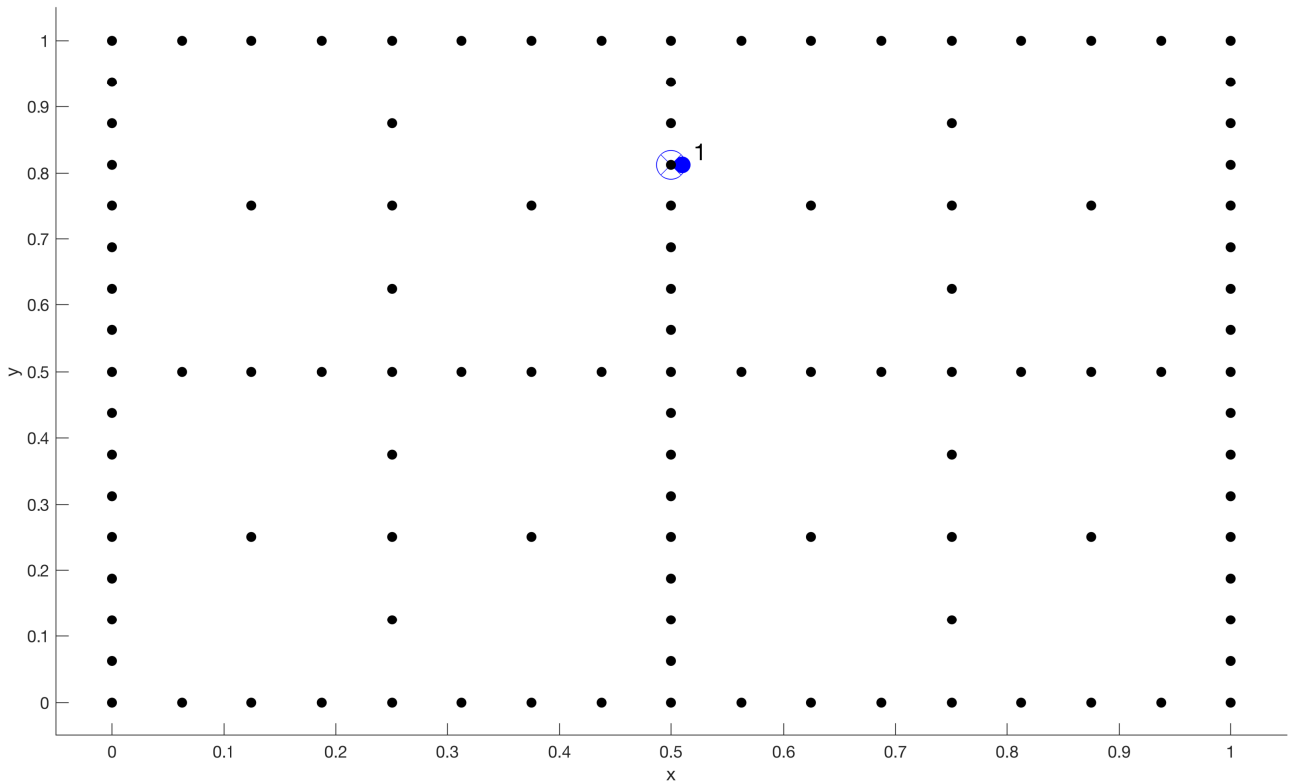
- The cost (# degrees of freedom or inner grid points) of $\mathcal{V}_n^{\mathrm{RSG}}$ is

$$|\mathcal{V}_n^{\mathrm{RSG}}| = \mathcal{O}\left(\frac{1}{h_n}\left|\log_2 h_n\right|^{d-1}\right) \leftrightarrow \mathcal{O}(2^n \cdot n^{d-1}) \quad \textbf{v.s.} \quad \mathcal{O}(2^{n\cdot d})$$

Figure: The standard figure that illustrates the construction of a regular sparse grid

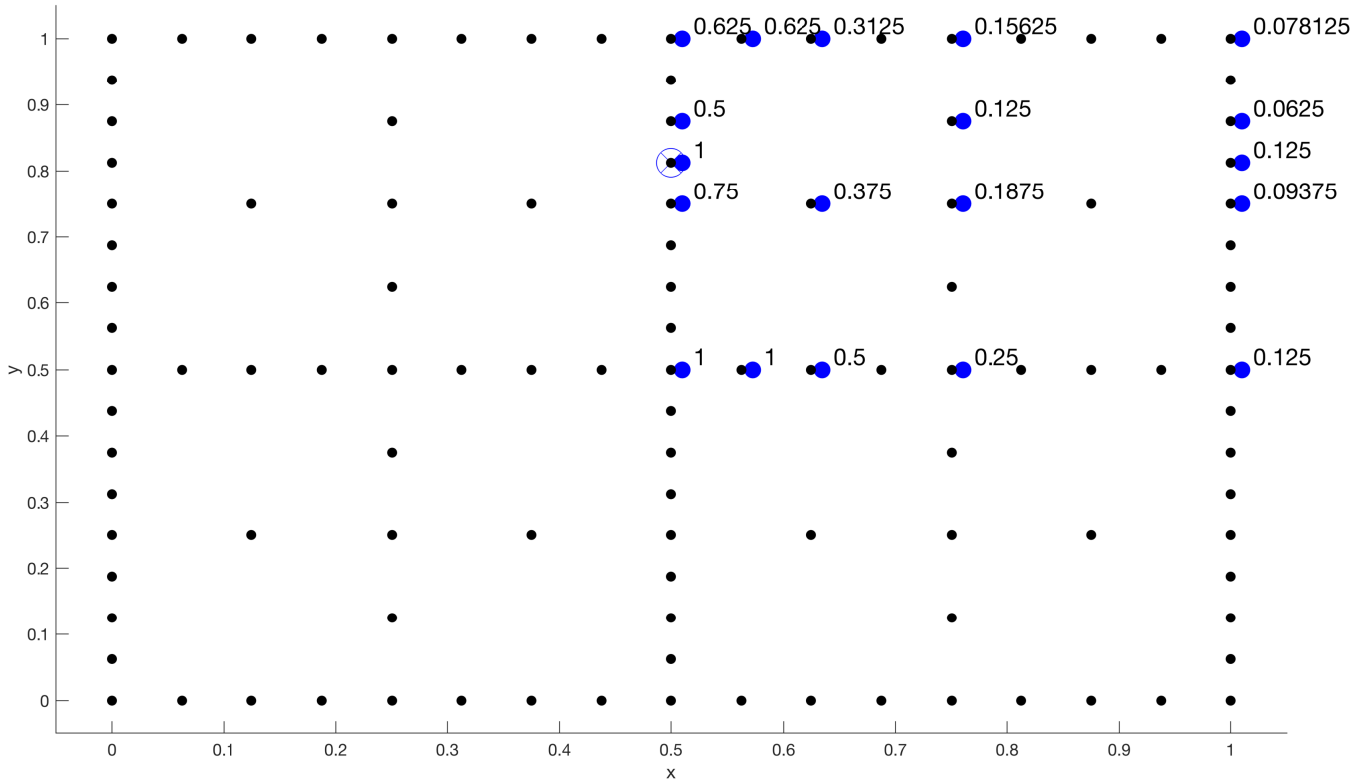
Interpolation: Stencil Figures

$$E^y \; D_x^+ \; H^y = \frac{B_{x,h}^+ H^x H^y - I}{h}$$



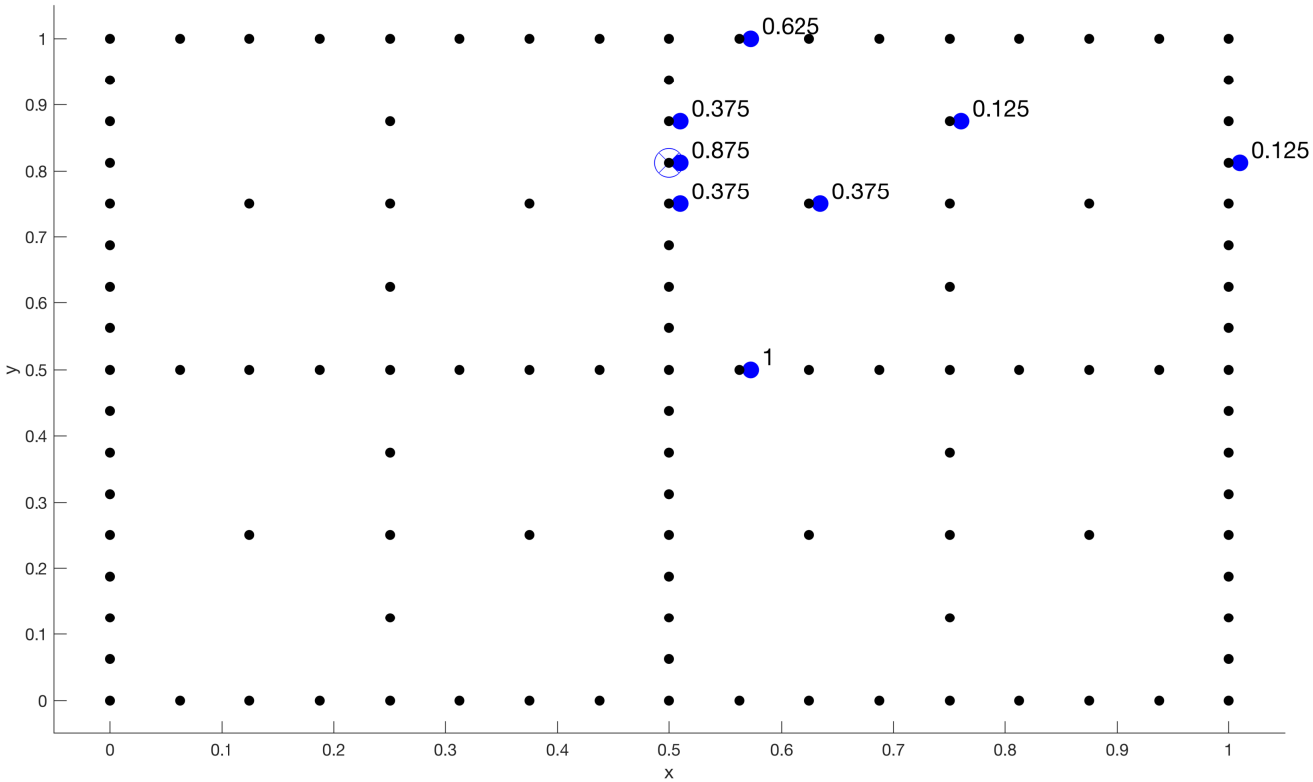
Interpolation: Stencil Figures

$$E^y \; D_x^+ \; H^y = \frac{B_{x,h}^+ H^x H^y - I}{h}$$



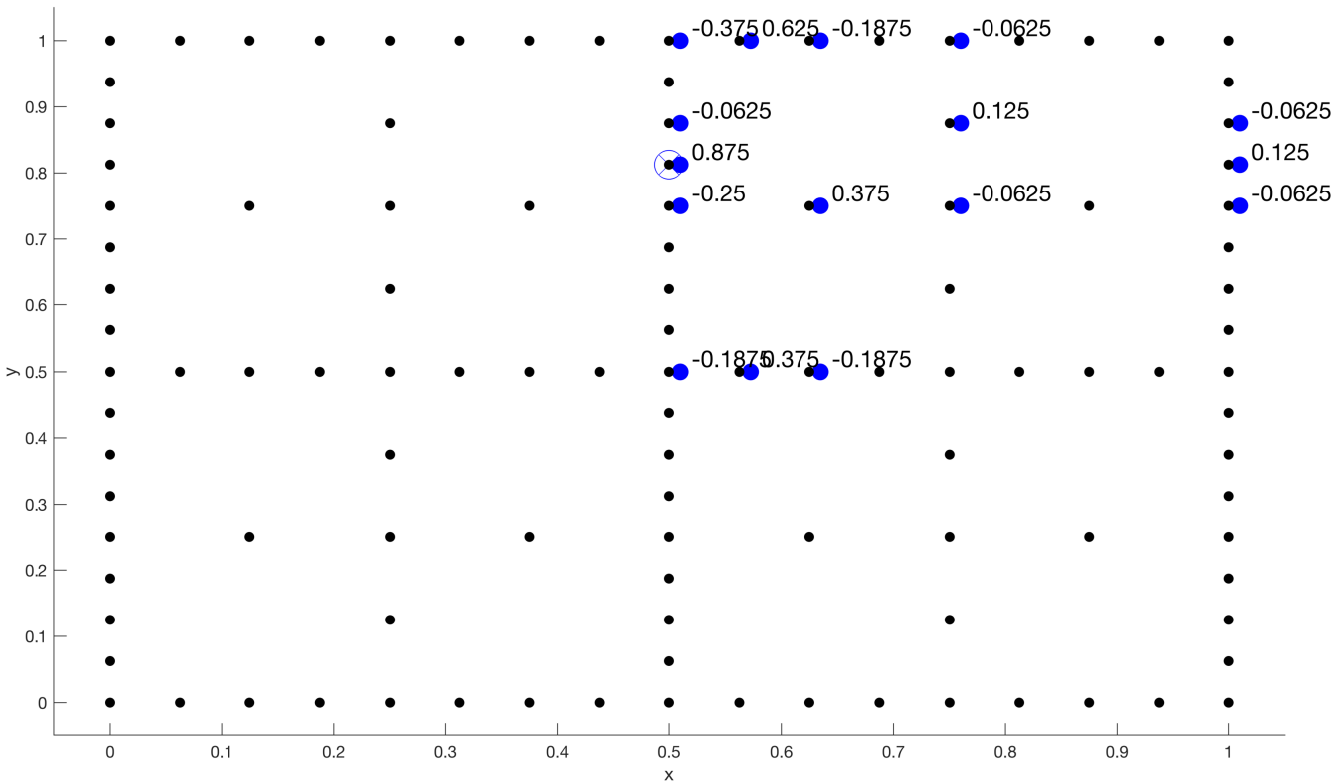
Interpolation: Stencil Figures

$$E^y \; D_x^+ \; H^y = \frac{B_{x,h}^+ H^x H^y - I}{h}$$



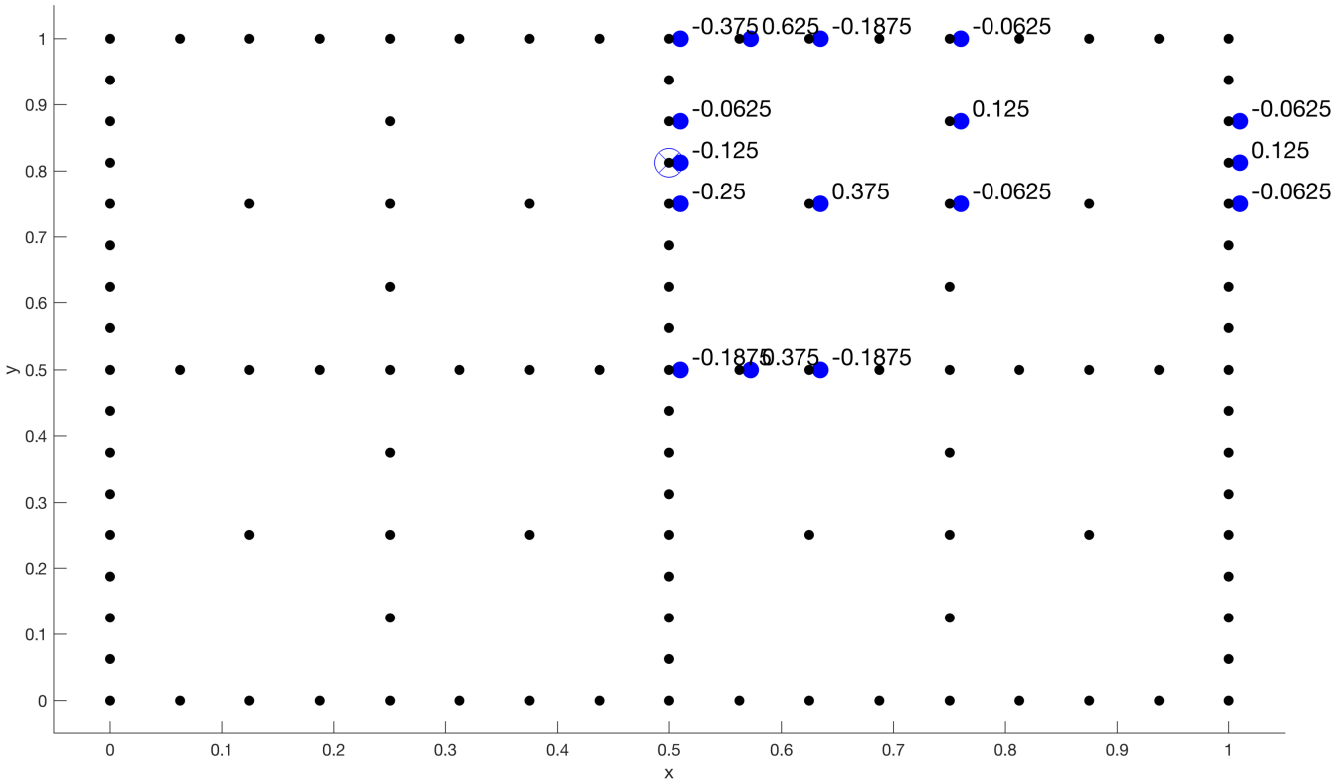
Interpolation: Stencil Figures

$$E^y \; D_x^+ \; H^y = \frac{B_{x,h}^+ H^x \, H^y - I}{h}$$



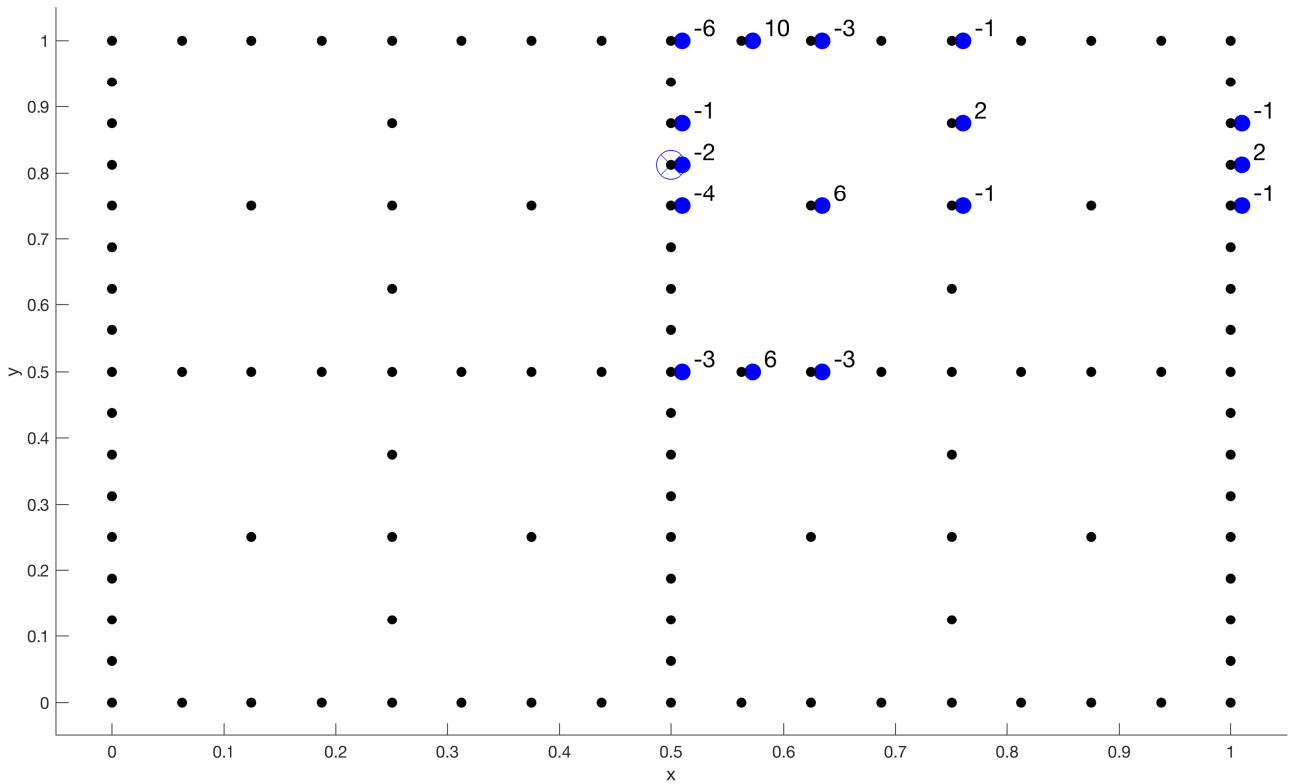
Interpolation: Stencil Figures

$$E^y \; D_x^+ \; H^y = \frac{B_{x,h}^+ H^x H^y - I}{h}$$



Interpolation: Stencil Figures

$$E^y \; D_x^+ \; H^y = \frac{B_{x,h}^+ H^x H^y - I}{h}$$



Interpolation: Proof Sketch (1)

- Consider grid G with J grid points $(\boldsymbol{x}, \boldsymbol{y})$, smallest mesh size h and local distance to "right neighbor" h^{loc}
- Let $\boldsymbol{B}_x^+$ be matrix that evaluates all J hierarchical BFs $\tilde{\phi}_j(\cdot)$ at $(\boldsymbol{x} + h, \boldsymbol{y})$:

$$\boldsymbol{B}_x^+ = \begin{pmatrix} \dots & \tilde{\phi}_j(\boldsymbol{x} + h, \boldsymbol{y}) & \dots \end{pmatrix}$$

- By tensor product construction of $\tilde{\phi}_j(\cdot)$:

$$\boldsymbol{B}_x^+ = \begin{pmatrix} \dots & \tilde{\phi}_j^y(\boldsymbol{y}) \cdot \tilde{\phi}_j^x(\boldsymbol{x} + h) & \dots \end{pmatrix}$$

- $\boldsymbol{x} + h$ may not be on grid! Construct its value using grid points:

$$\begin{aligned} \boldsymbol{B}_x^+ &= \begin{pmatrix} \dots & \tilde{\phi}_j^y(\boldsymbol{y}) \cdot \tilde{\phi}_j^x\left(\frac{h}{h^{\text{loc}}}(\boldsymbol{x} + h^{\text{loc}}) + \frac{h^{\text{loc}} - h}{h^{\text{loc}}} \boldsymbol{x} \right) & \dots \end{pmatrix} \\ &= \begin{pmatrix} \dots & \tilde{\phi}_j^y(\boldsymbol{y}) \cdot \tilde{\phi}_j^x\left(\boldsymbol{x} + h \frac{(\boldsymbol{x} + h^{\text{loc}}) - \boldsymbol{x}}{h^{\text{loc}}} \right) & \dots \end{pmatrix} \end{aligned}$$

Interpolation: Proof Sketch (2)

- Every $\tilde{\phi}_j$ is locally linear over the open interval $x \in (x_i, x_i + h^{\text{loc}})$ for all i:

$$\tilde{\phi}_j^x\left(x + h\frac{(x + h^{\text{loc}}) - x}{h^{\text{loc}}}\right) = \tilde{\phi}_j^x(x) + h\frac{\tilde{\phi}_j^x(x + h^{\text{loc}}) - \tilde{\phi}_j^x(x)}{h^{\text{loc}}}$$

- Rearranging:

$$\boldsymbol{B}_x^+ = \begin{pmatrix} \dots & \tilde{\phi}_j^y(y) \cdot \tilde{\phi}_j^x(x) + h\tilde{\phi}_j^y(y) \cdot \left(\frac{\tilde{\phi}_j^x(x+h^{\text{loc}}) - \tilde{\phi}_j^x(x)}{h^{\text{loc}}}\right) & \dots \end{pmatrix}$$

- Resolving tensor product *roughly* via:

$$\tilde{\phi}_j^y(y) \cdot \tilde{\phi}_j^x(x) \to \boldsymbol{E}^y\, \boldsymbol{E}^x$$

$$\tilde{\phi}_j^y(y) \cdot \left(\frac{\tilde{\phi}_j^x(x + h^{\text{loc}}) - \tilde{\phi}_j^x(x)}{h^{\text{loc}}}\right) \to \boldsymbol{E}^y\, \boldsymbol{D}_x^+\, \boldsymbol{E}^x$$

- We obtain:

$$\boldsymbol{B}_x^+ = \boldsymbol{E}^y\, \boldsymbol{E}^x + h\boldsymbol{E}^y\, \boldsymbol{D}_x^+\, \boldsymbol{E}^x$$

Interpolation: Proof Sketch (3)

- Now to get interpolation, by definition we have

$$\boldsymbol{I}_x^+ = \boldsymbol{B}_x^+ \, \boldsymbol{H}^x \, \boldsymbol{H}^y$$

- Plugging in, we get

$$\boldsymbol{I}_x^+ = (\boldsymbol{E}^y \, \boldsymbol{E}^x + h\boldsymbol{E}^y \, \boldsymbol{D}_x^+ \, \boldsymbol{E}^x) \, \boldsymbol{H}^x \, \boldsymbol{H}^y = \boldsymbol{I} + h \, \boldsymbol{E}^y \, \boldsymbol{D}_x^+ \, \boldsymbol{H}^y$$

- Finally:

$$\frac{\boldsymbol{I}_x^+ - \boldsymbol{I}}{h} = \boldsymbol{E}^y \, \boldsymbol{D}_x^+ \, \boldsymbol{H}^y$$