Tensors and the Curse of Dimensionality

Quantitative Macroeconomics

Raül Santaeulàlia-Llopis

MOVE-UAB and Barcelona GSE

Fall 2017

1. Multidimensionality: Using Tensors
   - Dealing with more than one dimension
   - Example: A Chebyshev Approximation Algorithm in $\Re^2$
2. Complete Polynomials
3. Smolyak Algorithm

Multidimensionality: Using Tensors

Tensors build multidimensional basis functions by finding the Kronecker product of all unidimensional basis functions.

For example, assume to state variables a safe asset a_t and risky capital k_t. Assume that we use three Chebyshev polynomials for each of these two state variables:

$$\psi_0^k(a_t), \psi_1^k(a_t), \psi_2^k(a_t) \text{ and } \psi_0^k(k_t), \psi_1^k(k_t), \psi_2^k(k_t)$$

Then, the tensor product is given by these nine products:

$$\begin{aligned}
&\psi_0^k(a_t)\psi_0^k(k_t), \psi_0^k(a_t)\psi_1^k(k_t), \psi_0^k(a_t)\psi_2^k(k_t),\\
&\psi_1^k(a_t)\psi_0^k(k_t), \psi_1^k(a_t)\psi_1^k(k_t), \psi_1^k(a_t)\psi_2^k(k_t),\\
&\psi_2^k(a_t)\psi_0^k(k_t), \psi_2^k(a_t)\psi_1^k(k_t), \psi_2^k(a_t)\psi_2^k(k_t)
\end{aligned}$$

Dealing with more than one dimension: Tensors

More formaly, the approximation of a function of n state variables with for example equal domain $[-1, 1]$ for each state, i.e.,$f : [-1, 1]^n \rightarrow \Re$ wit Chebyshev polynomial of degre j is:

$$\widetilde{f}(.|\theta) = \sum_{i_1=0}^{j} ... \sum_{i_n=0}^{j} \theta_{i1,...,i_n} \psi_{i_1}^1(.) * ... * \psi_{i_n}^n(.)$$

where $\psi_{i_\kappa}^\kappa$ is the Chebyshev polynomial of degree i_κ on the state variable κ and θ is the vector of coefficients $\theta_{i1,...,i_n}$.

Note that there is no need to use the same number of Chebyshev polynomials for each state variable. Also, we said Chebyshev but this applies to any polynomial, in fact, each state could use a different type of polynomial.

Two main advantages of tensor basis:

1. It is easy to build.
2. If the one-dimensional basis is orthogonal, then the tensor basis is orthogonal in the product norm.

One big disadvantage: there is exponential growth in the number of coefficients to be computed $\theta_{i_1,\ldots,i_n} : (j+1)^n$. In the example above we had to compute 9 coefficients (2 states and 3 polynomials). If we had 5 states and 3 polynomials then this implies 243 coefficients. If we have 10 polynomials then we need to compute 100,000 coefficients.

A Chebyshev Approximation Algorithm in $\Re^2$

Goal: Given a function $f(x, y)$ defined on $[a, b] \times [c, d]$, find its Chebyshev polynomial approximation $\widetilde{f}(x, y)$.

Chebyshev Approximation Algorithm in $\Re^2$ (Alghtm 6.4 in Judd 1998)

Step 1 Compute $m \geq n+1$ Chebyshev nodes on $[-1,1]$:

$$z_k = -cos\left(\frac{2k-1}{2m}\pi\right),\ k = 1, ..., m$$

Step 2 Adjust the nodes to the $[a,b]$ and $[c,d]$ intervals':

$$x_k = (z_k + 1)\left(\frac{b-a}{2}\right) + a,\ k = 1, ..., m$$

$$y_k = (z_k + 1)\left(\frac{d-c}{2}\right) + c,\ k = 1, ..., m$$

Step 3 Evaluate f at the approximation nodes: $w_{k,l} = f(x_k, y_l)$ for $k = 1, ..., m$ and $l = 1, ..., m$.

Step 4 Compute Chebyshev coefficients, θ_{ij} for $i, j, = 0, ..., n$, as

$$\theta_{ij} = \frac{\sum_{k=1}^{m}\sum_{l=1}^{m} w_{k,l}\psi_i(z_k)\psi_j(z_l)}{\left(\sum_{k=1}^{m}\psi_i(z_k)\right)^2\left(\sum_{l=1}^{m}\psi_j(z_l)\right)^2}$$

to arrive to the approximation:

$$\widetilde{f}(x,y) = \sum_{i=0}^{n}\sum_{j=0}^{n}\theta_{ij}\psi_i\left(2\frac{x-a}{b-a} - 1\right)\psi_j\left(2\frac{y-c}{d-c} - 1\right)$$

A Chebyshev example in $\Re^2$: A CES production function

[Graph]

Steps to Reduce the Curse of Dimensionality

1. Complete Polynomials
2. Smolyak Algorithm

Complete Polynomials

We can turn to bases that with multiple dimenions grow only polynomally (as opposed to exponentially).

Recall the Taylor's theorem for many dimensions implies the following approximation:

$$\widetilde{f} \approx f(x^0) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(x^0)(x_i - x_i^0) + ... + \frac{1}{k!} \sum_{i_1=1}^{n} ... \sum_{i_k=1}^{n} \frac{\partial^k f}{\partial x_{i_1}, ..., \partial x_{i_k}}(x^0)(x_{i_1} - x_{i_1}^0)...(x_{i_k} - x_{i_k}^0)$$

Focusing in the terms of the kth-degree expansion:

- For $k = 1$, Taylor's theorem for n dimensions used the linear functions $P_1^n \equiv \{1, x_1, ..., x_n\}$
- For $k = 2$, Taylor-s theorem used $P_2^n = P_1^n U \equiv \{x_1^2, ..., x_n^2, x_1x_2, x_1x_3, ..., x_{n-1}x_n\}$

That is, P_2^n contains some product terms, but not all; for example, $x_1x_2x_3$ is not in P_2^n.

In general, the kth-degree expansion of n dimensional functions uses functions in

$$P_k^n \equiv \left\{ x_1^{i_1}, ..., x_n^{i_n} | \sum_{l=1}^{n} i_l \leq k, 0 \leq i_1, ..., i_n \right\}$$

The set P_k^n is called the complete set of polynomials of total degree k in n variables.

Using complete sets of polynomials we can construct bases for multivariate approximation that are often more efficient than tensor products.

For example, if $\Psi_k \equiv \{1, x, ..., x^k\}$ is the collection of kth-degree basis faunctions, then its n-dimensional tensor product contains many more elements than P_k.

Table: Size of alternative bases (Table 6.6 in Judd 1998)

Degree k	P_k^n	Tensor product Ψ_k^n
2	$1 + n + \frac{n(n+1)}{2}$	3^n
3	$1 + n + \frac{n(n+1)}{2} + n^2 + \frac{n(n-1)(n-2)}{6}$	4^n

Smolyak Algorithm

An interesting way to can handle the curse of dimensionality better than other methods is Smolyak's algorithm. You need to check Krueger and Kubler (2004) and Malin et al. (2011) for a comprehensive exposition.

Goal: Approximate a function on n state variables, $f : [-1, 1]^n \to \Re$.

The idea: Find a grid of points $\mathbb{G}(q,n) \in [-1,1]^n$ with $q > n$ and an approximant $\widetilde{f}(x|\theta,q,n) : [-1,1]^n \to \Re$ indexed by some coefficients θ such that:

- at the points $x_i \in \mathbb{G}(q,n)$, the unknown function $f(.)$ and the approximant are equal:

$$f(x_i) = \widetilde{f}(x_i|\theta,q,n)$$

- and, at the points $x_i \notin \mathbb{G}(q,n)$, the approximant is close to the unkown function.

The second bit makes this more demanding than collocation. That is, at the points $x_i \in \mathbb{G}(q,n)$ the operator $D(f) = 0$. In addition, at other points $x_i \notin \mathbb{G}(q,n)$ the residual function will be close to zero.

The integer q indexes the size of the grid and, with it, the precision of the approximation.

The challenge is to judiciously select grid points $\mathbb{G}(q, n)$ in such a way that the number of coefficients θ does not explode with n.

Smolyaks algorithm is (almost) optimal for that task within the set of polynomial approximations (Barthelmann et al., 2000). Also, the method is universal, that is, almost optimal for many different function spaces.

Smolyak's algorithm

We search for a grid of points $\mathbb{G}(q, n)$ and a function $\widetilde{f}(x|\theta, q, n)$ in several steps:

Step 1 **Transform the domain of the state variables**

For any state variable $\widetilde{x}_l$, $l = 1, ..., n$ that has a domain $[a, b]$, we transform it linearly into the space $[-1, 1]$,

$$x_l = 2\frac{\widetilde{x}_l - a}{b - 1} - 1$$

Step 2 **Setting the order of the polynomial**

Define $m_1 = 1$ and $m_i = 2^{i-1} + 1$ for $i = 2, ...$ The order of the polynmomial is $m_i - 1$.

This way the order of the polynomials follows the sequence:

$$\begin{aligned} m_1 - 1 &= 0, \\ m_2 - 1 &= 2, \\ m_3 - 1 &= 4, \\ m_4 - 1 &= 8, \\ m_5 - 1 &= 16, \\ m_6 - 1 &= 32 \end{aligned}$$

and so on...

Smolyak's algorithm (continued)

Step 3 **Build the sets of Chebyshev Extrema (Gauss-Lobotto Nodes or Clenshaw-Curtis points)**

We build sets:

$$\mathcal{G}^i = \{\zeta_1^i, ..., \zeta_m^i\} \subset [-1, 1]$$

that contain the extrema of the Chebyshev polynomials,

$$\zeta_j^i = -cos\left(\frac{j-1}{m_i - 1}\pi\right) \quad j = 1, ..., m_i$$

with initial $\mathcal{G}^1 = \{0\}$.

For instance, the first three sets are given by:

$\mathcal{G}^1 = \{0\}$, where $i = 1, m_1 = 1$, & polynomial order is $m_i - 1 = 0$

$\mathcal{G}^2 = \{-1, 0, 1\}$, where $i = 2, m_2 = 3$, & polynomial order is $m_i - 1 = 2$

$\mathcal{G}^3 = \left\{-1, -cos\left(\frac{1}{4}\pi\right), 0, -cos\left(\frac{3}{4}\pi\right), 1\right\}$, where $i = 3, m_3 = 5$, & poly order $m_i - 1 = 4$

By imposing $m_i = 2^{i-1} + 1$ in the construction of $\mathcal{G}$, we generate sequentially nested sets: $\mathcal{G}^i \subset \mathcal{G}^{i+1}\ \forall i = 1, 2, ...$ This nestedness is crucial for the success of the algorithm.

Smolyak's algorithm (continued)

Step 4 **Build a sparse grid**

For any integer q bigger than he number of state variables n, $q > n$, we define a sparse grid as the union of the Cartesian products:

$$\mathbb{G}(q,n) = \bigcup_{q-n+1 \leq |\mathbf{i}| \leq q} \left(\mathcal{G}^{i_1} \times ... \times \mathcal{G}^{i_n} \right)$$

where $|\mathbf{i}| = \sum_{l=1}^{n} i_l$

To illustrate how this sparse grid works:

- Imagine that we are dealing with a model with two continuous state variables. If we pick $q = 2 + 1 = 3$, we have the sparse grid:

$$\begin{aligned}\mathbb{G}(3,2) &= \bigcup_{2\leq|\mathbf{i}|\leq 3}\left(\mathcal{G}^{i_1}\times\mathcal{G}^{i_2}\right)\\ &= \left(\mathcal{G}^1\times\mathcal{G}^1\right)\cup\left(\mathcal{G}^1\times\mathcal{G}^2\right)\cup\left(\mathcal{G}^2\times\mathcal{G}^2\right)\\ &= \{(-1,0),(0,1),(0,0),(0,-1),(1,0)\}\end{aligned}$$

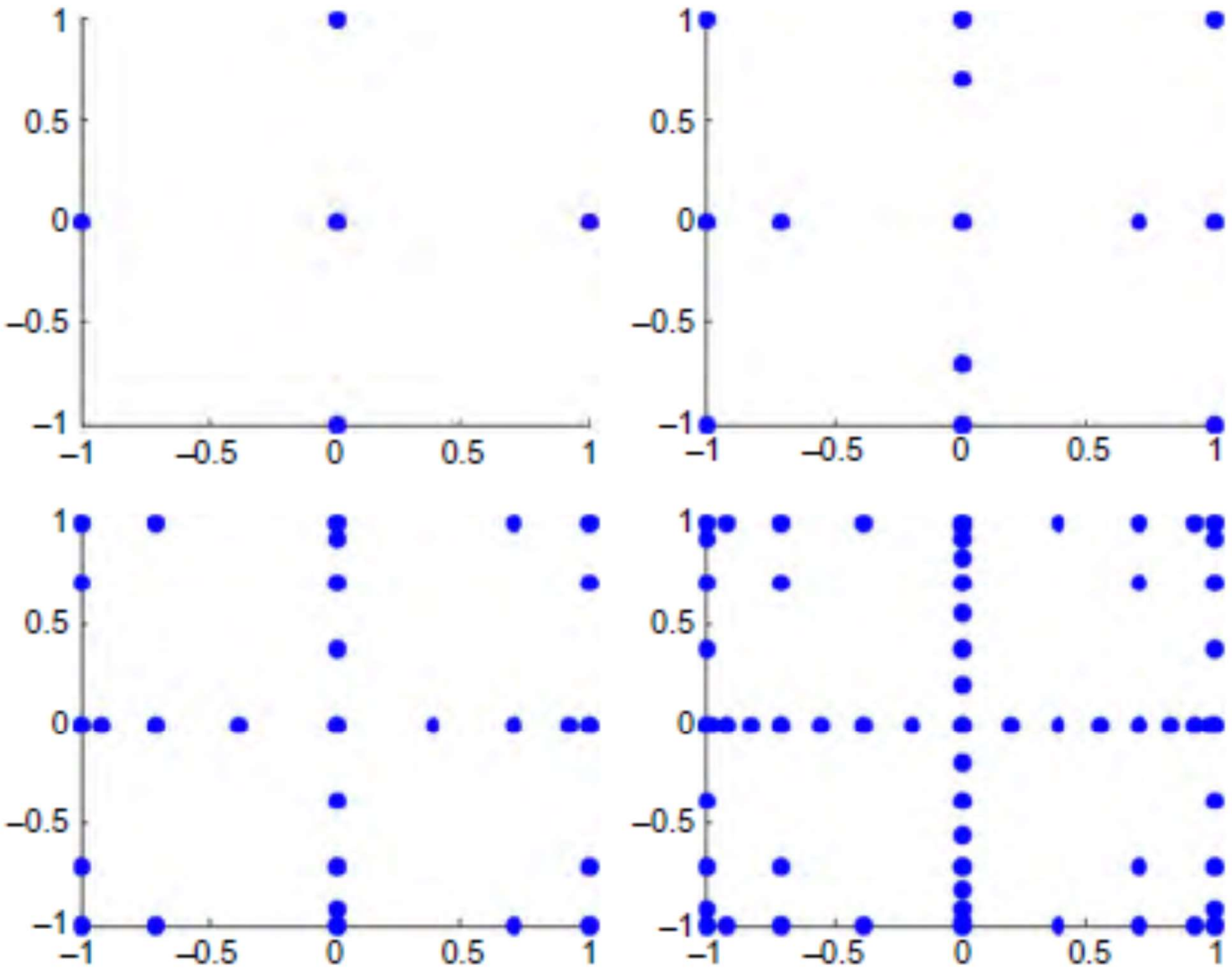
  This is the top left panel in the next slide.
- Imagine we pick $q = 2 + 2 = 4$, we have the sparse grid:

$$\begin{aligned}\mathbb{G}(4,2) &= \bigcup_{3\leq|\mathbf{i}|\leq 4}\left(\mathcal{G}^{i_1}\times\mathcal{G}^{i_2}\right)\\ &= \left(\mathcal{G}^1\times\mathcal{G}^2\right)\cup\left(\mathcal{G}^1\times\mathcal{G}^3\right)\cup\left(\mathcal{G}^2\times\mathcal{G}^2\right)\cup\left(\mathcal{G}^3\times\mathcal{G}^1\right)\\ &= \left\{\begin{array}{c}(-1,1),\ (-1,0),\ (-1,-1),\ \left(-cos\left(\frac{1}{4}\pi\right),0\right),\\ (0,1),\ \left(0,-cos\left(\frac{3}{4}\pi\right)\right),(0,0),\left(0,-cos\left(\frac{1}{4}\pi\right)\right),\\ (0,-1),\ \left(-cos\left(\frac{3}{4}\pi\right),0\right),(1,1),(1,0),(1,-1)\end{array}\right\}\end{aligned}$$

  This is the top right panel in the next slide. Note that the sparse grids have a hierarchial structure, where $\mathbb{G}(3,2)\in\mathbb{G}(4,2)$ or, more generally, $\mathbb{G}(q,n)\in\mathbb{G}(q+1,n)$.

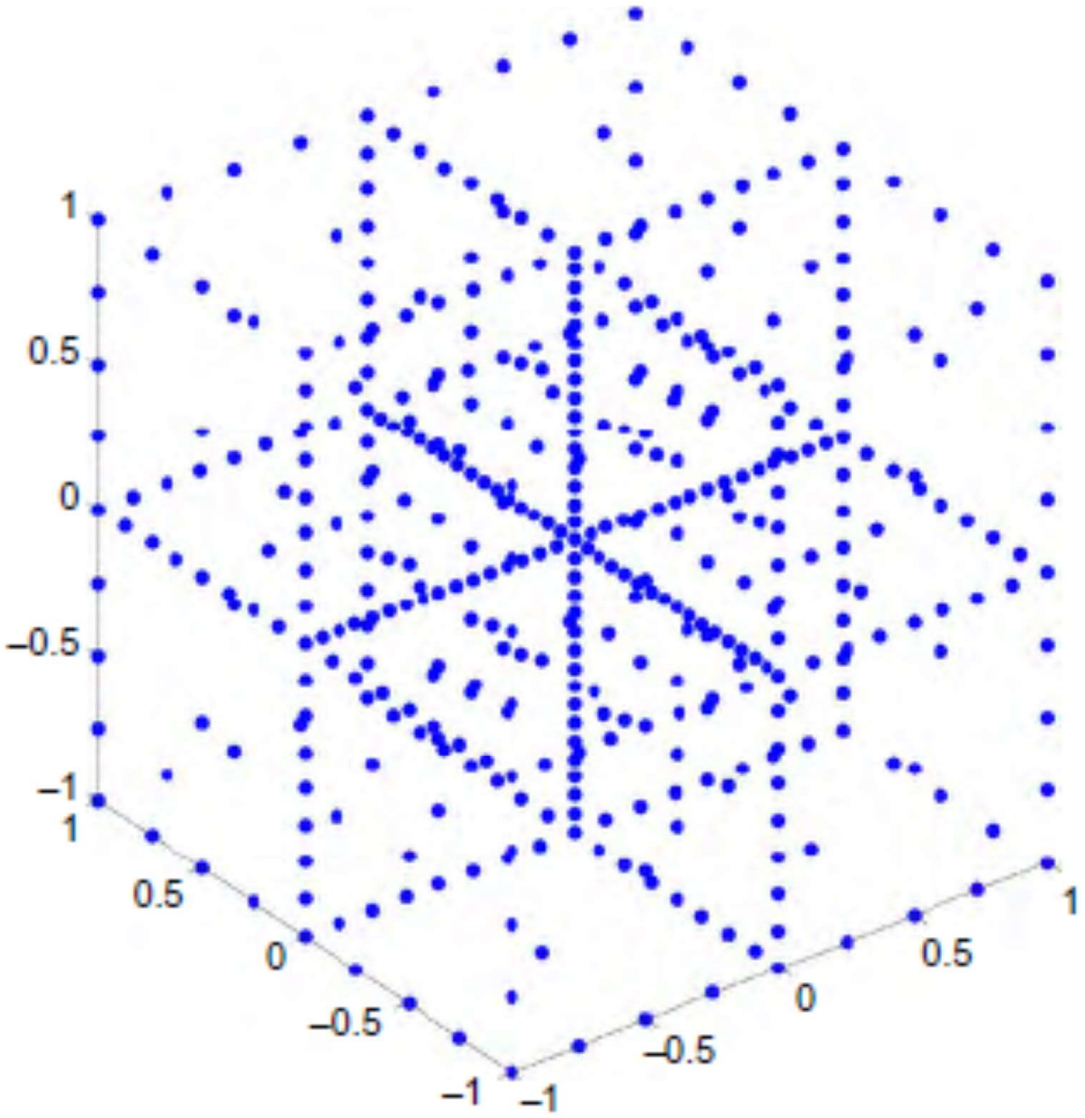
  Following the same strategy, we can build $\mathbb{G}(5,2)$, plotted in the bottom left panel and $\mathbb{G}(6,2)$ plotted in the bottom right panel. In two slides we plot a grid for a problem with 3 state vairables $\mathbb{G}(5,3)$.

Example: Four sparse grids:



[Figure from Fernandez-Villaverde, Rubio-Ramirez and Schorfheide]

Example: A sparse grid with 3 state variables ($\mathbb{G}(5,3)$)



[Figure from Fernandez-Villaverde, Rubio-Ramirez and Schorfheide]

The sparse grid has two important properties:

- First, the grid points cluster around the corners of the domain of the Chebyshev polynomials and the central cross.
- Second, the number of points in a sparse grid when $q = n + 2$ is given by $1 + 4n + 2n(n-1)$. The cardinality of this grid grows polynomially on n^2. Similar formulae hold for other $q \geq n$. For example, the cardinality of the grid grows polynomially on n^3 when $q = n + 3$. In fact, the computational burden of the method notably increases as we keep n fixed and a rise q. Experience suggests that $q = n + 2$ and $q = n + 3$ are usually enough to deliver the desired accuracy in DSGE models.

The nestedness of the sets of the Gauss-Lobotto nodes plays a central role in controlling the cardinality of $\mathbb{G}(q,n)$. In comparison, the number of points inthe rectangular grid is 5^n, an integer that grows exponentially on n. If $n=2$, this would correspond (in the top right panel in the previous figure) to having all possible tensors of

$$\left\{-1,-cos\left(\frac{1}{4}\pi\right),0,-cos\left(\frac{3}{4}\pi\right),1\right\}$$

and itself

$$\left\{-1,-cos\left(\frac{1}{4}\pi\right),0,-cos\left(\frac{3}{4}\pi\right),1\right\}$$

covering the whole of $[-1,1]^2$ square. Instead of keeping these 25 points, Smolyak's algorithm eliminates 12 of them and only keeps 13.

To illustrate how dramatic is the difference between polynomial and exponential growth, the following table shows the cardinality of both grids as we move from 2 state variables to 12

Table: Size of the grid for $q = n + 2$

n	$\mathbb{G}(q, n)$	5^n
2	13	25
3	25	125
4	41	625
5	61	3,125
12	313	244,140,625

Smolyak's algorithm (continued)

Step 5 **Building Tensor Products**

We use Chebyshev polynomials $\psi_i(x_i) = T_{i-1}(x_i)$ to build the tensor-product multivariate polynomial:

$$p^{|\mathbf{i}|}(x|\theta) = \sum_{l_1=1}^{m_{i_1}} ... \sum_{l_n=1}^{m_{i_n}} \theta_{l_1,...,l_n} \psi_{l_1}(x_1), ..., \psi_{l_n}(x_n)$$

where $|\mathbf{i}| = \sum_{l=1}^{n} i_l$, $x_i \in [-1, 1]$, $x = \{x_1, ..., x_n\}$ and θ stacks all the coefficients $\theta_{l_1,...,l_n}$.

There is nothing special about the use of Chebyshev polynomials as the basis functions $\psi_j(x)$ and we could rely, if required, on other basis functions. For instance, one can implement a finite element method with the Smolyak algorithm by partitioning Ω into elements and defining local basis functions.

Smolyak's algorithm (continued)

Step 6 **Building the Interpolating Function in n Dimensions**

The Smolyak function that interpolates on $\mathbb{G}(q, n)$ is:

$$f(x|\theta, q, n) = \sum_{\max(n, q-n+1) \leq |\mathbf{i}| \leq q} (-1)^{q-|\mathbf{i}|} \begin{pmatrix} n-1 \\ q-|\mathbf{i}| \end{pmatrix} p^{|\mathbf{i}|}(x|\theta)$$

which is nothing more than the weighted sum of the tensors.

Smolyak's algorithm (continued)

Step 7 **Solve for the polynomial coefficients**

We plug $f(x|\theta, q, n)$ into the operator $D(f)$ for all $x_i \in \mathbb{G}(q, n)$. At this point the operator needs to be exactly zero:

$$D(f(x|\theta, q, n)) = 0$$

and we solve for the unkown coefficients in θ.

Note that the system of equatoins

$$D(f(x|\theta, q, n)) = 0, \; x_i \in \mathbb{G}(q, n)$$

is simply a system of nonlinear equations in θ. Krueger and Kubler (2004) and Malin et al. (2011) suggest a time-iteration method that starts, as an initial guess, from the first-order perturbation of the model. This choice is, nevertheless, not essential to the method.

Recently, Judd, Mailar, Mailar and Valero (2014) have proposed improvements of Smolyaks algorithm.

- First, the authors first present a more efficient implementation of Smolyaks algorithm that uses disjoint-set generators that are equivalent to the sets G^i.
- Second, the authors use a Lagrange interpolation scheme.
- Third, the authors build an anisotropic grid, which allows having a different number of grid points and basis functions for different state variables. This may be important to capture the fact that, often, it is harder to approximate the decision rules of agents along some dimensions than along others.
- Finally, the authors argue that it is much more efficient to employ a derivativefree fixed-point iteration method instead of the time-iteration scheme proposed by Krueger and Kubler (2004) and Malin et al. (2011).