O’ Curse of Dimensionality, Where is Thy Sting?

Prepared for
CEF2006, Cyprus

Kenneth L. Judd
Hoover Institution and NBER
June 21, 2006
Curse of Dimensionality

- Many economic models are high dimensional
  - Dynamic Optimization: Multiple kinds of capital stocks
  - DSGE: Multiple consumers/firms/countries
  - Games: Multiple players and states
  - Bayesian analyses compute high-dimensional integrals
  - Bootstrapping: analyze many $n$-dimensional samples from $n$ data points
  - Simulation of large Markov processes - MCMC, Gibbs sampling, ACE
  - Parameter space searches to find robust conclusions

- Claim: “You can’t solve your model because of the curse of dimensionality.”

- Response I: Analyze silly models
  - Reduce heterogeneity in tastes, abilities, age, etc.
  - Assume no risk
  - Assume common information, beliefs, and learning rules

- Response II: Do bad math

- Response III: Do bad math when analyzing silly models
• The message today: “The curse is not so bad”
  – “Theorems” about the curse are irrelevant for economics
  – There are many underutilized tools from math that can help
  – Sensible modelling choices can avoid curse
  – Mathematicians are currently developing tools to tackle the curse
  – Physicists are working to build computers that can avoid the curse
  – If the Boston Red Sox can beat the “Curse of the Bambino” then economists can beat the “Curse of Dimensionality”
Dynamic Example - Dynamic Programming

- Basic Bellman equation:

\[
V(x) = \max_{u \in D(x)} \pi(u, x) + \beta E\{V(x^+) | x, u\} \equiv (TV)(x). \tag{12.7.1}
\]

- Computational task:
  - Choose a finite-dimensional parameterization (e.g., polynomials, splines, etc.)
    \[
    V(x) \equiv \hat{V}(x; a), \ a \in \mathbb{R}^m \tag{12.7.2}
    \]
  and a finite number of states
  \[
  X = \{x_1, x_2, \cdots, x_n\}, \tag{12.7.3}
  \]
  - Objective: find coefficients \( a \in \mathbb{R}^m \) such that \( \hat{V}(x; a) \) “approximately” satisfies the Bellman equation.
• Value function iteration: For each $x_j$, $(TV)(x_j)$ is defined by

$$v_j = (TV)(x_j) = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u)$$

(12.7.5)

• In practice, we compute the approximation $\hat{T}$

$$v_j = (\hat{TV})(x_j) = (TV)(x_j)$$

– Integration step: for $\omega_j$ and $x_j$ for some numerical quadrature formula

$$E\{V(x^+; a)|x_j, u}\} = \int \hat{V}(g(x_j, u, \varepsilon); a) dF(\varepsilon) = \sum_{\ell} \omega_{\ell} \hat{V}(g(x_j, u, \varepsilon_{\ell}); a)$$

– Maximization step: for $x_i \in X$, evaluate

$$v_i = (T\hat{V})(x_i)$$

– After finding the new $v_j$, we execute a fitting step:

  * Data: $(v_i, x_i), \ i = 1, \cdots, n$
  
  * Objective: find an $a \in R^m$ such that $\hat{V}(x; a)$ best fits the data

– Value function iteration iterates on the coefficient vector $a$. 
• Dimension is important at many stages
  
  – Solving the multidimensional optimization problem: \( u \in \mathbb{R}^m \) implies \( m^2 \) elements in Hessian
  
  – Approximating the \( n \)-dimensional value function \( V(x) \) and choosing the points \( x_j \): cost of simple methods is proportional to \( e^n \) - curse of dimensionality!
  
  – Integrating the conditional expectation with \( q \)-dimensional shocks: many methods have costs proportional to \( e^q \) - curse of dimensionality!

• The good news: For many (if not most) problems in economics
  
  – \( m \)-dimensional Hessians cost only \( m \) to compute
  
  – \( n \)-dimensional approximation is polynomial in \( n \)
  
  – \( q \)-dimensional integrals of \( C^k \) functions can use \( N \) points and converge at rate \( N^{-k} \) independent of dimension!
  
  – Physics offers new ways to defeat the curse of dimensionality.
Dynamic Example - Euler Equation Models

• Euler equation for simple growth model

\[ u'(C(k)) = \beta u'(C(F(k) - C(k)))F'(F(k) - C(k)) \quad (16.4.2) \]

– When \( k \) is \( n \)-dimensional, we need to approximate \( n \)-dimensional functions \( C'(k) \)

– Identifying coefficients in \( C(k) \) approximation is an integration problem

– When we add uncertainty, we get multidimensiona integrals for conditional expectations on right hand side.

– Dynamic equilibrium models have the same computational needs as dynamic programming.

• Same computational tasks are present for dynamic games, which have both value functions and policy functions to compute.
Evaluating Derivatives Efficiently

- Common belief: “Newton’s method is impractical for large problems”
  - Analytic derivatives are slow

<table>
<thead>
<tr>
<th>Analytic Derivatives</th>
<th>(+, -, *, ÷)</th>
<th>Power</th>
<th>Total fleet</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>function</td>
<td>(u = (x^\sigma + y^\sigma + z^\sigma)\rho)</td>
<td>2</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>gradient</td>
<td>(u_x = \sigma \rho x^{\sigma-1} (x^\sigma + y^\sigma + z^\sigma)^{\rho-1})</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>(u_y = \sigma \rho y^{\sigma-1} (x^\sigma + y^\sigma + z^\sigma)^{\rho-1})</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>(u_z = \sigma \rho z^{\sigma-1} (x^\sigma + y^\sigma + z^\sigma)^{\rho-1})</td>
<td>4</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>grad. total:</td>
<td>12</td>
<td>9</td>
<td>15</td>
<td>36</td>
</tr>
<tr>
<td>Hessian</td>
<td></td>
<td></td>
<td></td>
<td>(\geq 400)</td>
</tr>
</tbody>
</table>
Finite differences are slow

Finite Difference Derivatives

<table>
<thead>
<tr>
<th>Function</th>
<th>$u = (x^\sigma + y^\sigma + z^\sigma)^\rho$</th>
<th>+,-</th>
<th>*,-,÷</th>
<th>Power</th>
<th>Total flops</th>
<th>Total time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gradient</td>
<td>$u_x = (u(x + \Delta, y, z) - u) / \Delta$</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>$u_y = (u(x, y + \Delta, z) - u) / \Delta$</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>$u_z = (u(x, y, z + \Delta) - u) / \Delta$</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>24</td>
</tr>
<tr>
<td>Grad total:</td>
<td></td>
<td>9</td>
<td>3</td>
<td>12</td>
<td>24</td>
<td>72</td>
</tr>
<tr>
<td>Hessian</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( \geq 150 )</td>
</tr>
</tbody>
</table>
• Automatic Differentiation to the rescue! Reduce redundant computations

<table>
<thead>
<tr>
<th>function</th>
<th>( x_1 = x^\sigma , \ y_1 = y^\sigma , \ z_1 = z^\sigma )</th>
<th>( A = x_1 + y_1 + z_1 )</th>
<th>( u = A^\rho )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0  3  3  15</td>
<td>2  2  2</td>
<td>1  1  5</td>
</tr>
<tr>
<td></td>
<td>2  0  4  6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>gradient</td>
<td>( x_2 = x_1/x ), ( y_2 = y_1/y ), ( z_2 = z_1/z ), ( A_1 = \rho \sigma u/A )</td>
<td>( u_x = x_2 \ A_1 )</td>
<td>( u_y = y_2 \ A_1 )</td>
</tr>
<tr>
<td></td>
<td>3  3  3</td>
<td>1  1  1</td>
<td>1  1  1</td>
</tr>
<tr>
<td></td>
<td>9  9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>15 31</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Two kinds of gains
  * Fewer operations
  * Less use of expensive operations: power (~10 adds), exponential (~5 adds),
• Insights are old
  – Many, including Leigh Tesfatsion, recognized these ideas by mid 1980’s.
  – Software development was slow. Tesfatsion was an early contributor.

• Theorem: (Griewank) For an \( n \)-dimensional function \( f \):
  – \( \text{Cost (Jacobian)} < 5 \text{ Cost (} f \text{)} \)
  – \( \text{Cost (Hessian)} < 5 n \text{ Cost (} f \text{)} \)

• Comments:
  – This is worst-case analysis
  – This ignores any savings from avoiding costly operations.
• Current applications and implications

  – Use Newton methods (and discard DFP, BFGS, and BHHH)
    * AD is now incorporated into much software; AMPL and GAMS (but not empirical packages!)
    * L-B-J versus Fair-Taylor: already exploits sparseness, could exploit AD
    * Solve stochastic dynamic games - Pakes-Maguire
      · Most use Gauss–Seidel methods (e.g., mimic best reply dynamics)
      · Ferris-Judd-Schmedders: 10,000 states, 40,000 unknowns with binding constraints; done in 5 seconds on a laptop
  – Perturbation methods
    * Perturbation methods are gaining in popularity: Judd, Guu, Gaspar, Anderson, Juillard, Collard, Kim², Jesus Fernandez-Villaverde, Juan Rubio.
    * Some have incorporated AD ideas into their code: Anderson-Levin-Swanson
    * Laplace expansions in statistics.
Function Approximation

- Linear polynomial methods:

\[ f(x, y, z, \ldots) = \sum_{i=1}^{m} a_i \phi_i (x, y, z, \ldots), \ \phi_i \ \text{multivariate polynomials} \]

- Choices for \( \phi \) are tensor versus complete:

<table>
<thead>
<tr>
<th>One D</th>
<th>degree 1 in each variable</th>
<th>degree 2 in each variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, x</td>
<td>1, x, x^2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2D tensor</th>
<th>( {1, x} \otimes {1, y} )</th>
<th>( {1, x, x^2} \otimes {1, y, y^2} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>product</td>
<td>( {1, x, y, xy} )</td>
<td>( {1, x, x^2, y, y^2, xy, x^2y, xy^2, x^2y^2} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3D tensor</th>
<th>( {1, x} \otimes {1, y} \otimes {1, z} )</th>
<th>( {1, x, y, z, xy, xz, yz, xyz} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>product</td>
<td>( {1, x, y, z, xy, xz, yz, xyz} )</td>
<td>( {1, x, y, z, xy, xz, yz, x^2, y^2, z^2} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>2D complete</th>
<th>1, x, y</th>
<th>1, x, x^2, y, y^2, xy</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D complete</td>
<td>1, x, y, z</td>
<td>1, x, y, z, xy, xz, yz, x^2, y^2, z^2, \ldots</td>
</tr>
</tbody>
</table>
Proper notion of “degree” in multivariate context is sum of powers

$$\text{degree} \left( x^i y^j z^k \right) = i + j + k$$

Complete polynomials like

$$\sum_{i+j+k \leq m} a_{ijk} x^i y^j z^k$$

have far fewer terms than tensor products like

$$\sum_{i=0}^{m} \sum_{j=0}^{m} \sum_{k=0}^{m} a_{ij} x^i y^j z^k$$

with ratio being about $d!$ in $d$-dimensional case.

Complete polynomials are better in terms of approximation power per term

degree $k$ Number terms in complete poly Number terms in tensor product
2 \quad \approx \frac{1}{3} n^2 \quad 3^n
3 \quad \approx \frac{1}{6} n^3 \quad 4^n

• Splines

– One dimension is easy

\[ \sum_{i=0}^{m} a_i B_i(x) \]

– Tensor approach is bad; no “complete” approach since no \( B_i \) covers all \( x, y \):

\[ \sum_{i=0}^{m} \sum_{j=0}^{m} a_{ij} B_i(x) B_j(y) \]

– Radial basis functions to the rescue:

* Functional form uses arbitrary, scattered points \( p_i \) in

\[ \sum_{i=1}^{N} a_i \phi(\|x - p_i\|) \]

* \( \phi \) choices include

\[ e^{-r^2}, \frac{1}{\sqrt{1 + r^2}}, \frac{1}{1 + r^2}, \sqrt{1 + r^2}, ... \]

* New results show that these can be excellent approximations. Need to figure out best choices for \( p_i \) points.

* Recent work shows that RBFs can be very effective on PDEs similar to ones from economics.
Defining the Domain

- Choosing the domain of our problem (e.g., states in a DP or dynamic GE model) is important
  - Want to include values for state that are part of the solution
  - Choosing too large a domain will create unnecessary computational burdens.

- More choices with higher dimensions
  - One dimension: Domain is interval; just need to know max and min
  - Two dimensions: More choices - square/rectangle, sphere/ellipse, simplex, etc.
  - Three dimensions: More choices - cube, sphere, ellipsoid, cylinder, simplex, etc.

DeathStarBattle

The Place to Kill Stuff

www.deathstarbattle.com
• Cube versus Sphere

  – Spheres are much more compact:
    * In cube of unit length at edge, length of longest diagonal is \( n^{1/2} \)
    * Ratio of sphere to cube volume is
      \[
      \frac{\pi^{n/2}}{(n/2)!}, \text{ n even}
      \]
      \[
      \frac{2^{n/2+1}\pi^{n/2}}{1\cdot3\cdot5\cdot\ldots\cdot n}, \text{ n odd}
      \]
    * Smaller volume reduces costs of approximation; allows one to exploit periodicity
    * Smaller volume reduces cost of integration
  – If solution has a central tendency, then it rarely visits vertices
an edge corresponds to a scheme of size 1. As an example, the scheme $00\ast$ represents the edge going from corner $(0,0,0)$ to the corner $(0,0,1)$, or the scheme $\ast\ast0$ the lower face of a cube.

The count of schemes of size $k$ equals the number of possibilities for selecting $k$ wildcard positions out of $n$, multiplied with the number of possibilities to assign 0 or 1 to the remaining $n-k$ positions, i.e. $2^{n-k}$. Therefrom it follows for the count $N_k^n$ of $k$-dimensional hypercubes bordering the $n$-dimensional hypercube:

$$N_k^n = 2^{n-k}\binom{n}{k}. \quad (1)$$

Equation (1) was used to derive the values in table 1. The number of hypersurfaces, to which a corner belongs, can be found in a similar manner. It has to be counted, how many schemes of size $n-1$ are realized by a given bitstring. This are just $n$ schemes, one for each bit position. The remaining $2n-n=n$ hypersurfaces of the hypercube are disjoint to that corner, and each connecting line from the corner to an inner point of one of those disjoint hypersurfaces completely lies within the hypercube.

All other entries in table 1 are obvious.

The volume of a $n$-dimensional unit hypercube is 1. For $n \to \infty$, the volume of a hypercube with $l > 1$ goes to infinity, while for $l < 1$ it goes to 0. Also, the length of the diagonal of a unit hypercube ($\sqrt{n}$) goes to infinity, the hypercube becomes more and more extended. According to [4], the hypercube can be imagined as a highly anisotropical body, more ressembling a spherical “hedgehog” than a convex body. The inner ball-like part with radius $1/2$ is covered with a large number ($2^n$) of “spikes” of length $\sqrt{n}/2$ (going to infinity for large $n$) (see figure 4). "The surfaces of cubes are so horribly jagged that they might even be thought of as being almost fractal." ([4], p. 42).

![Figure 4: "Spiking Hypercube"](image_url)

Finally, a short computation will show the change of relative volumina within the hypercube, when problem dimension increases. On the main diagonal of a hypercube, a random point $P$ is selected with coordinates $p_1 = p_2 = \ldots = p_n = p$ and $p > 1/2$. This way, two subcubes of the hypercube are defined, with one including the point
<table>
<thead>
<tr>
<th>$n$</th>
<th>$V_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.00000</td>
</tr>
<tr>
<td>2</td>
<td>3.14159</td>
</tr>
<tr>
<td>3</td>
<td>4.18879</td>
</tr>
<tr>
<td>4</td>
<td>4.93480</td>
</tr>
<tr>
<td>5</td>
<td>5.26379</td>
</tr>
<tr>
<td>6</td>
<td>5.16771</td>
</tr>
<tr>
<td>7</td>
<td>4.72477</td>
</tr>
<tr>
<td>8</td>
<td>4.05871</td>
</tr>
<tr>
<td>9</td>
<td>3.29851</td>
</tr>
<tr>
<td>10</td>
<td>2.55016</td>
</tr>
</tbody>
</table>

Table 2: Volume of the unit cube for the dimensions 1 to 10.

As it was the case for the hypersimplex, the volume of a hypersphere goes to 0, independent from the size of its radius. Table 2 gives the volumes of some unit cubes for $n = 1, \ldots, 10$. The hypersphere for $n = 5$ has the biggest volume, but this depends on $R$. Especially, for $R = 1/\sqrt{2}$, the hypersphere attains its maximum in “our” 3-dimensional world.

As an example consequence for search spaces, the ratio of volumes of unit hypersphere and embedding hypercube (with a sidelength of 2) will be considered (see figure 5). Despite of the fact that the volume of the hypercube goes to infinity, and the hypersphere touches all faces of the hypercube (i.e. at $2n$ points), the volume of the embedded hypersphere goes to 0! Moreover, for $n > 10$ we could neglect this volume part within the hypercube for all practical computations. This is an important distinction between search methods, which explores the hypercube, and search methods, which explores the hypersphere. The last ones will not “see” very much from their world.

Also, it has to be noted that the term “high dimension” may refer to values of $n$ as small as 10 or so.
Integration - Gaussian-style formulas

- Integration formulas for one dimension look like

\[
\int_{-1}^{1} f(x) \, dx = \sum_{i=1}^{n} \omega_i f(x_i)
\]

\[
\int_{-\infty}^{\infty} f(x) e^{-x^2} \, dx = \sum_{i=1}^{n} \omega_i f(x_i)
\]

- \(n\) point formula
- \(2n\) parameters (points and weights)
- uses \(n\) bits of information
- exactly integrates all polynomials of degree \(2n - 1\).

- Simple approach for higher dimensions:
  - Take product of one-dimensional methods:

\[
\sum_{i_1=1}^{m} \cdots \sum_{i_d=1}^{m} \omega_{i_1}^1 \omega_{i_2}^2 \cdots \omega_{i_d}^d f(x_{i_1}^1, x_{i_2}^2, \cdots, x_{i_d}^d)
\]

- Curse of dimensionality - number of points used is exponential in dimension \(d\)
There are other approaches

- Do not have to use simple Cartesian grids
- Consider \( X = \{(x, y, z) | x, y, z \in \{-1, 1\}\} \). The rule
  \[
  \frac{4}{3} (f(1, 0, 0) + f(-1, 0, 0) + f(0, 1, 0) + f(0, -1, 0) + f(0, 0, 1) + f(0, 0, -1))
  \]
  uses 6 points and exactly integrates all degree 3 polynomials
  \[
  \{1, x, y, z, x^2, y^2, z^2, xyz, xy^2, x^2y, xz^2, x^2z, yz^2, y^2z\}
  \]
  over \([-1, 1]^3\)
- More generally, in dimension \( d \) you can use \( 2d \) points and exactly integrate all degree 3 polynomials over \([-1, 1]^d\) with
  \[
  \int_{[-1,1]^d} f = \omega \sum_{i=1}^{d} (f(u e^i) + f(-u e^i)),
  \]
  where
  \[
  e^i \text{ is } +1 \text{ or } -1 \text{ in dimension } i
  \]
  \[
  u = \left( \frac{d}{3} \right)^{1/2}, \quad \omega = \frac{2^{d-1}}{d}
  \]
- In general, there are nongrid sets of points that can be used.
• New research direction I: Find rules that are good for many polynomials

  - Choose points $z_i$ and weights $\omega_i, \ i = 1,..,m$, to create a quadrature rule, $Q(f; z, \omega)$, to minimize errors.

    * The literature is for one-dimensional problems:

      $$\min_{z, \omega} \sum_{i=0}^{\infty} \left( Q(x^i; z, \omega) - \int x^i \, dx \right)^2$$

    * A few mathematicians do this - Gismalla, Cohen, Minka

    * This is not done often since “you can’t publish the results”.
– I created one for a two-D sphere:

* We need new formulas if we switch to spheres
* Choose 12 points (24 coordinates and 12 weights) to minimize sum of squared errors of formula applied to \( x^i y^j, \ i, j \leq 20 \).
* Use unconstrained optimization software; use many restarts to avoid local solution
* Result was

\[
0.2227 \left( f[-0.8871, 0] + f[0, -0.8871] + f[0, 0.8871] + f[0.8871, 0] \right) \\
+ 0.2735 \left( f[-0.6149, 0.6149] + f[-0.6149, -0.6149] + f[0.6149, -0.6149] + f[0.6149, 0.6149] \right) \\
+ 1.0744 f[-0.3628, 0] + 1.0744 f[0, 0.3628] \\
+ 1.0744 f[0, -0.3628] + 1.0744 f[0.3628, 0]
\]

with relativized errors of \( 10^{-5} \) on average and \( 10^{-4} \) at worst on degree 20 polynomials
* Result had interesting symmetry - 3 groups of 4 points lying on 3 circles - which gives indication as to what symmetries I should try in higher dimensions.
– General strategy: Look for formulas with small numbers of points to find desirable patterns for point sets, then assume those patterns when searching for bigger formulas.

– General principal: Use your time to come up with ideas, and use the computer to do the tedious work.

  * Idea here: use formulas that integrate an important set of polynomials.
  * Tedious work here: searching for optimal rule that satisfies the criterion.
• New research direction II: Use more information

  – Gauss-Turan methods use derivatives

\[
\int_{-1}^{1} f(x) \, dx = \sum_{i=1}^{n} \omega_{i,0} f(x_i) + \sum_{i=1}^{n} \omega_{i,1} f'(x_i) + \sum_{i=1}^{n} \omega_{i,2} f''(x_i)
\]

  * n-point formula has 4n parameters, and uses 3n bits of information to integrate first 4n polynomials

  * In one dimension, the cost of \( f \) and first two derivatives is about same as three \( f \)’s, so no gain in one dimension.
However, Gauss-Turan has potential for high-dimensional integrals (Judd, 2006)

* The formula

\[
\int_{-1}^{1} \int_{-1}^{1} f(x, y) \, dx \, dy = \sum_{i=1}^{n} \omega_{i,0} f(x_i, y_i) \\
+ \sum_{i=1}^{n} (\omega_{i,x} f_x(x_i, y_i) + \omega_{i,y} f_y(x_i, y_i)) \\
+ \sum_{i=1}^{n} (\omega_{i,xx} f_{xx}(x_i, y_i) + \omega_{i,xy} f_{xy}(x_i, y_i) + \omega_{i,yy} f_{yy}(x_i, y_i))
\]

uses \( 6n \) bits of information at \( n \) points - one \( f \) evaluation and five derivatives - has \( 7n \) parameters and can integrate first \( 7n \) polynomials

* Using automatic differentiation, multidimensional Gauss-Turan will beat regular quadrature rules that use only \( f \) values
Integration - General Sampling Methods

- Sampling methods (including MC) use sequence $x_i$ and computes $N$-point approximation
  \[
  \int_0^1 f(x) \, dx \equiv \frac{1}{N} \sum_{i=1}^{N} f(x_i)
  \]  

- There are alternatives called quasi-Monte Carlo methods. Two simple examples in $R^d$ are
  
  Weyl: 
  \[
  x^n = \left( np_1^{1/2}, \cdots, np_d^{1/2} \right) \mod 1
  \]
  
  Niederreiter: 
  \[
  x^n = \left( n 2^{1/(d+1)}, \cdots, n 2^{d/(d+1)} \right) \mod 1
  \]

- Convergence for integrals using Weyl or Neiderreiter is $N^{-1}$.

- The debate is which deterministic formulas we should use, not deterministic versus random.
  
  - MC sequences are designed to look like iid sequences; coincidentally, they do well at integration
  
  - qMC sequences are designed to be uniformly distributed and to do well at integration
First 1500 Weyl points
Fig. 9. Orthogonal projection of a randomized (0,3,19)-net.

Rather than using independent replications, an alternate approach to estimate \( \hat{\sigma}^2 \) (or \( \hat{\sigma}^2 \)) is to rely on a single large scrambled \((0,m,s)\)-net. Recall that a \((0,m,s)\)-net is a point set with \(b^m\) points \(\{A_n, n = 1, \ldots, b^m\} \). Suppose the \((0,m,s)\)-net is divided equally into \(b\) subsets of points as follows:

\[
\begin{align*}
A_1, \ldots, A_{b^{m-1}}, & \quad A_{b^{m-1} + 1}, \ldots, A_{2b^{m-1}}, \quad \ldots, \quad A_{(b-1)b^{m-1} + 1}, \ldots, A_{b^m}.
\end{align*}
\]

so that each subset (or netlet) becomes a \((0,m-1,s)\)-net with \(b^{m-1}\) points. Hence, a single scrambled \((0,m,s)\)-net can be interpreted as a \((0,m-1,s)\)-net.
where from (2.1)

\[ w_j = \int_{\mathbb{S}^r} \ell_j(x) \, ds(x). \]

The practical computation of the weights proceeds differently. The reproducing kernel basis functions \( g_j(x) = G_n(x_j, x) \), where \( G_n(x, y) \) is the reproducing kernel for \( \mathbb{P}_n(S^r) \) [19], satisfy

\[ \int_{\mathbb{S}^r} g_j(x) \, ds(x) = 1 \quad \text{for } j = 1, \ldots, d_n, \]

so the condition that the cubature rule (1.1) is exact for all polynomials in \( \mathbb{P}_n \) can be written as

\[ Gw = e, \quad (3.1) \]

where \( w \) is the vector of cubature weights, \( e \) is the vector of 1’s in \( \mathbb{R}^{d_n} \) and \( G \) is the matrix defined in (2.4). In particular the cubature rule is exact for the constant polynomial \( 1 \in \mathbb{P}_n \), so \( \sum_{j=1}^{d_n} w_j = |S^r| \). Hence the average weight is \( w_{\text{avg}} = |S^r|/d_n = 1/\lambda_{\text{avg}} \), or \( w_{\text{avg}} = 4\pi/d_n \) for \( r = 2 \). The computed minimum and maximum values of the ratio of the cubature weights to the average weight for the extremal systems of section 2 are
- Portfolio example
  - More relevant for dynamic stochastic general equilibrium than option pricing examples
  - Assumptions
    * $n$ assets (iid Uniform), $n = 10, 15, 20, 25, 30$
    * random portfolio with total variance fixed
    * $u(c) = -e^{-c}$
  - Goal: compute expected utility, a $n$ - dimensional integral

$$
\int_{[-1,1]^n} u \left( 1 + \sum_{i=1}^{n} \theta_i z_i \right) dz
$$
- Methods
  * degree 8 Taylor series (exploiting some AD methods): high fixed cost of computing general formula but application is practically instantaneous
  * Monte Carlo, Weyl, one randomly shifted Weyl
  * $10^5$ points

- Results:
  * Taylor series: high fixed cost to get general formula but application is practically instantaneous - shows value of AD and symbolic method for integrals
  * Taylor and Weyl were less than two standard errors from MC: MC could not reject the others
  * Weyl was 10-100 times better than MC
  * Performance at $n = 30$ was same as $n = 10$. 
• Practical facts
  – qMC has been used for many high-dimension (e.g., 360) problems in option pricing problems
  – pMC asymptotics kick in early; qMC asymptotics take longer for the qMC sequences we know
  – Therefore, pMC methods have \textit{finite sample advantages}, not asymptotic advantages.
  – “quasi-MC” is bad name since qMC methods have no logical connection to probability theory
• News:
  
  – New sequences: randomized \((t - m - s)\) sequences have \(N^{-3/2}\) convergence - Owen  
  
  – New methods are now producing good qMC rules.  
  
  – qMC outperforms MC by factor of 10 in mixed logit discrete-choice models and simulated maximum likelihood multinormal models - Train, Bhat  
  
  – qMC successes for computing Normal probabilities \((x \sim N(0, 1))\)

\[
\Pr [Ax \leq b], \quad x \sim N(0_n, I_{n \times n});
\]

see Sandor-Andras (\textit{J. Econometrics}, 2004); beats GHK by 10-100 for dimension< 10; by more than 2 for dimension 10-50.  

– qMC can do MCMC and Gibbs, and be faster than MC by 10-100 on ordinary problems

  * “Sampling Strategies for MCMC” (November, 2005) Tribble and Owen.  
  * “A quasi-Monte Carlo Metropolis Algorithm\(^1\),” Owen and Tribble, 2005.  

• Conjecture: There is a lot of “low hanging fruit” available for econometricians working on applying qMC to econometrics problems.

\(^1\)Should be called Metropolis - Rosenbluth - Rosenbluth - Teller - Teller method.
Methodology: Pure Math versus Experimental Math

- MC methods as practiced is very useful and sound but not supported by usual mathematical theorems. Real proof is
  - Suppose $f(x) = \sum_{i=0}^{\infty} a_i x^i$ on $[0, 1]$ and $\sum_{i=K}^{\infty} a_i x^i$ is negligible for some
  - Suppose computations show that a sequence $X_i$ properly computes $\int x^i dx$ at rate $N^{-1/2}$ for each $i < K$.
  - Then, MC will compute $\int f(x)$ at rate $N^{-1/2}$

- This is experimental mathematics NOT probability theory!

- Experimental math:
  - Test out conjecture on many cases to explore validity
  - Combine computational results with pure math to arrive at conclusions with known range of validity
  - Computational results may inspire theorems, such as Neiderreiter analysis of LCM.
Problem is not with using MC, but with understanding logical underpinnings.

- MC in practice is not based on probability theory
- It is inspired by probability theory, but theorems do not apply
- This inspiration led to search for pMC sequences which, by testing, were found to do a good job on some problems

Why are these logical points important?

- All agree that Monte Carlo is a very important and useful tool.
- Recognition of the true foundation for MC will encourage us to develop other methods based on a similarly disciplined combination of analysis and computational experimentation.
Modelling Suggestion I: Use Continuous Time

- We pay a high price when we choose discrete-time formulations

- Dynamic programming: “next period’s value”
  - Discrete-time: $V(F(x, U(x)))$ - double composition
  - Continuous-time: $V'(x)F(x, U(x))$ - single composition plus multiplication and gradient
  - Stochastic discrete time: $E\{V(F(x_t, U(x_t)), \theta_{t+1}) | \theta_t\}$ - double composition plus multidimensional integral
  - Stochastic continuous time: $V'(x)F(x, U(x)) + \sigma^2 V''(x)$ - single composition, multiplication, gradient, and Hessian
  - Composition of unknown functions ($V$ and $U$) is far costlier than derivatives for both perturbation and projection methods

- Stochastic games: Doraszelski-Judd show that continuous-time games are orders of magnitude faster than discrete-time games.
Modelling Suggestion II: Use Finite-Dimensional States

- Many economists have problems with infinite-dimensional states, such as the distribution of income.

- Alternative approach: There is only a finite number of people.

- Example: suppose you have dynamic programming problem with $N$ factories, each with DRTS, with adjustment costs for investment.

  - Bellman equation
    \[
    V(k) = \max_I u(c) + \beta V(k + I)
    \]
    \[c = \sum_i f(k_i) - \sum_i g^i(I^i(k))\]

  - Equations defining $V(k)$ and $I(k)$:
    \[
    V(k) = u(c) + \beta V(k + I(k))
    \]
    \[0 = -u'(c)(1 + \alpha I^i(k)) + V_i(k + I(k))\]

  - Idea: Use perturbation method to compute Taylor series for $V(k)$ and the $I^i(k)$.
Problems:

* $V_i$ is a vector of length $N$; $V_{ij}$ is a matrix with $N^2$ elements;

* $I^i_k$ is a list of $N$ functions; $I^i_j(k)$ is an $N \times N$ matrix; $I^i_{jm}(k)$ is $N \times N \times N$ tensor, etc.

* If $N = 10^9$, that is a lot of unknowns

Solution: Exploit symmetry at steady state

* $V_i = V_j, \forall i, j$

* $V_{ii} = V_{11}, \forall i; V_{ij} = V_{12}, \forall i \neq j$

* $V_{iii} = V_{111}, \forall i; V_{iij} = V_{112}, V_{ijj} = V_{122}, \forall i \neq j; V_{ijm} = V_{123}, \forall i \neq j \neq m \neq i$

* Similarly for $I^i$ functions

High-order Taylor series are feasible

* The number of unknowns when computing $q$’th derivative is $2^q$ independent

* Solutions depend on $N$; take $N \to \infty$ to find infinite population solution

* Risk - idiosyncratic and aggregate - can be added with little extra computational cost.

Similar to Gaspar-Judd (1997) and Krusell-Smith (1997) uses of symmetry, but far more efficient
Modelling Suggestion III: Get Rid of Kinks

• General observation: the more smoothness, the better for computation.

• Economists love to put in discontinuities.

• Hubbard (1986) example (why not pick on a Republican big shot business school dean?)
  – Wanted to examine tax policy implications of borrowing constraints.
  – Assumed one could not borrow against future wages; equivalent to

        \[ r(W) = \begin{cases} 
        r, & W > 0 \\
        \infty, & W < 0 
        \end{cases} \]

    or, equivalently,

        \[ u(c, W) = \begin{cases} 
        u(c), & W > 0 \\
        -\infty, & W < 0 
        \end{cases} \]
– Is this economically reasonable? Borrowing is not infinitely painful
  * First, go to parents.
  * Second, run up credit card debt.
  * In general, there is a set of sources of credit, with rising interest rates
  * Empirical fact: people do have debt!
– Results were interesting, but hampered by computational inefficiencies
  * Used a slow method to deal with endogenous age at which one is constrained.
  * Taxing capital income may be a good idea if it reduces wage taxation and liquidity constraint of borrowing constrained people.

• General point: kinks and discontinuities create problems but there are few problems where nonsmooth functions are necessary
The Future: Computing Speed

- We need more speed to do the necessary heavy lifting - searches for good methods, symbolic manipulation, experimental mathematics - implicit in the ideas mentioned above.

- More speed is coming.
Vacuum tubes were replaced by transistors, and transistors by integrated circuits, whose components became ever smaller and more numerous. During the 1980s microcomputers reached the consumer market, and the industry became more diverse and competitive. Powerful, inexpensive computer workstations replaced the drafting boards of circuit and computer designers, and an increasing number of design steps were automated. The time to bring a new generation of computer to market shrank from two years at the beginning of the 1980s to less than nine months. The computer and communication industries grew into the largest on earth.

Computers doubled in capacity every two years after the war, a pace that became an industry given: companies that wished to grow sought to exceed it, companies that failed to keep up lost business. In the 1980s the doubling time contracted to 18 months, and computer performance in the late 1990s seems to be doubling every 12 months.

---

**Evolution of Computer Power/Cost**

**Brain Power Equivalent per $1000 of Computer**

Faster than Exponential Growth in Computing Power. The number of MIPS in $1000 of computer from 1900 to the present. Steady improvements in mechanical and electromechanical calculators before World War II had increased the speed of calculation a thousandfold over manual methods from 1900 to 1940. The pace quickened with the appearance of electronic computers during the war, and 1940 to 1980 saw a millionfold increase. The pace has been even quicker since then, a pace which would make humanlike robots possible before the middle of the next century. The vertical scale is logarithmic, the major divisions represent thousandfold increases in computer performance. Exponential growth would show as a straight line, the upward curve indicates faster than exponential growth, or, equivalently, an accelerating rate of innovation. The reduced spread of the data in the 1990s is probably the result of intensified competition: underperforming machines are more rapidly squeezed out. The numerical data for this power curve are presented in the appendix.
The Future: Griebel-Wozniakowski Theorem

• Question: Are there good rules out there to defeat the curse of dimensionality? We want assurances before we begin this search.

• Answer: Yes, if we formulate problem in reasonable spaces.

• "On the Optimal Convergence Rate of Universal and Non-Universal Algorithms for Multivariate Integration and Approximation" by Griebel and Wozniakowski

  – Consider functions that belong to reproducing kernel Hilbert spaces.
    * Without loss of generality it is enough to consider linear algorithms.
    * The best algorithms for approximation and integration that works for all RKHS displays a curse of dimensionality.
    * For any given RKHS, the optimal rate of convergence is at least 1/2 for multivariate integration 1/4 for multivariate approximation.
If the kernel is a product of univariate kernels, i.e.,
\[
\int_{[0,1]^n} g(x) \, dF(x) = \int_{[0,1]} \cdots \int_{[0,1]} g(x) \, dF_1(x_1) \, dF_2(x_2) \cdots dF_n(x_n)
\]
then the optimal algorithm converges at the same rate as the slowest optimal algorithm across the univariate kernels. Hence, the optimal rate of convergence of universal algorithms for product kernels does not depend in dimension!

- Proof is nonconstructive, but tells us that computer searches are not necessarily futile.

- Economics problems are generally integrals of smooth functions over products of smooth univariate kernels as long as we stay away from kinks.
The Future: Quantum Computing

• New technology may also break curse of dimensionality

• Quantum computer example

  – Load quantum computer with a function $f$ and a number $n$.
  – ZAP it and it becomes $n$ computers (more precisely, the quantum state of the computer will be a superposition of the $n$ possible states) where computer $i$ computes $f(i)$, $i = 1, \ldots, n$
  – ZAP it $n^{-1/2}$ times
  – Take a random draw among the $n$ computers before they collapse back to one, but sample is now biased so that you get $\max f(i)$ with probability $1 - n^{-1}$!
• Quantum complexity theory

  – Examines possible efficiency of quantum computer algorithms.
  – There are examples of where quantum breaks curse of dimensionality.
    ∗ Path integration on a quantum computer is tractable.
    ∗ Path integration on a quantum computer can be solved roughly $\varepsilon^{-1}$ times faster than on a classical computer using randomization, and
    ∗ exponentially faster than on a classical computer with a worst case assurance.
    ∗ The number of quantum queries is the square root of the number of function values needed on a classical computer using randomization.

• In general, integration is faster on a quantum computer than a classical computer - Brassard-Hoya-Mosca-Tapp.
Conclusion

• If you formulate models in the right way, and If you use best available math, then you can avoid the curse of dimensionality.

• New developments are making that easier to do.

• The path is clear, but there is a lot of work to do to build the road.