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.)

    $$V(x) \doteq \hat{V}(x;a),\ \ a \in R^m \tag{12.7.2}$$

    and a finite number of states

    $$X = \{x_1, x_2, \cdots, x_n\}, \tag{12.7.3}$$

  - Objective: find coefficients $a \in R^m$ such that $\hat{V}(x;a)$ “approximately” satisfies the Bellman equation.

- Value function iteration: For each x_j, $(TV)(x_j)$ is defined by

$$v_j = (TV)(x_j) = \max_{u \in D(x_j)} \pi(u, x_j) + \beta \int \hat{V}(x^+; a) dF(x^+ | x_j, u) \tag{12.7.5}$$

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

$$v_j = (\hat{T}V)(x_j) \doteq (TV)(x_j)$$

  – Integration step: for ω_j and x_j for some numerical quadrature formula

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

  – 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: $u \in R^m$ 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

$$u'(C(k)) = \beta u'(C(F(k) - C(k)))F'(F(k) - C(k)) \tag{16.4.2}$$

  - When k is n-dimensional, we need to approximate n-dimensional functions $C\left(k\right)$
  - Identifying coefficients in $C\left(k\right)$ 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

Analytic Derivatives

		+,-	*,÷	Power	Total flops	Total time
function	$u = (x^\sigma + y^\sigma + z^\sigma)^\rho$	2	0	4	6	22
gradient	$u_x = \sigma\rho x^{\sigma-1}(x^\sigma + y^\sigma + z^\sigma)^{\rho-1}$	4	3	5		32
	$u_y = \sigma\rho y^{\sigma-1}(x^\sigma + y^\sigma + z^\sigma)^{\rho-1}$	4	3	5		32
	$u_z = \sigma\rho z^{\sigma-1}(x^\sigma + y^\sigma + z^\sigma)^{\rho-1}$	4	3	5		32
grad. total:		12	9	15	36	114
Hessian						$\geq$400

– Finite differences are slow

Finite Difference Derivatives

		+,-	*,÷	Power	Total flops	Total time
function	$u = (x^\sigma + y^\sigma + z^\sigma)^\rho$	2	0	4	6	22
gradient	$u_x = (u(x+\Delta, y, z) - u)/\Delta$	3	1	4		24
	$u_y = (u(x, y+\Delta, z) - u)/\Delta$	3	1	4		24
	$u_z = (u(x, y, z+\Delta) - u)/\Delta$	3	1	4		24
grad total:		9	3	12	24	72
Hessian						$\geq$150

- Automatic Differentiation to the rescue! Reduce redundant computations

		+,-	*,÷	a^b	Total flops	Appx. clock time
function	$x1 = x^\sigma,\ y1 = y^\sigma,\ z1 = z^\sigma$		0	3	3	15
	$A = x1 + y1 + z1$	2			2	2
	$u = A^\rho$			1	1	5
		2	0	4	6	22
gradient	$x2 = x1/x,\ y2 = y1/y,\ z2 = z1/z,$		3		3	3
	$A1 = \rho\ \sigma\ u/A$		3		3	3
	$u_x = x2\ A1$		1		1	1
	$u_y = y2\ A1$		1		1	1
	$u_z = z2\ A1$		1		1	1
grad. cost					9	9
					15	31

  - 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:
  - Cost (Jacobian) $<$ 5 Cost (f)
  - Cost (Hessian) $<$ 5 n Cost (f)
- 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, Kim^2, 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:

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

  – Choices for ϕ are tensor versus complete:

	degree 1 in each variable	degree 2 in each variable
one D	$1, x$	$1, x, x^2$
2D tensor	$\{1,x\}\otimes\{1,y\}$	$\left\{1,x,x^2\right\}\otimes\left\{1,y,y^2\right\}$
product	$=\{1,x,y,xy\}$	$=\left\{1,x,x^2,y,y^2,xy,\right.$
3D tensor	$\{1,x\}\otimes\{1,y\}\otimes\{1,z\}$	$\left.x^2y,xy^2,x^2y^2\right\}$
product	$=\{1,x,y,z,xy,xz,yz,xyz\}$	
2D complete	$1, x, y$	$1,x,x^2,y,y^2,xy$
3D complete	$1, x, y, z$	$1,x,y,z,xy,xz,yz,x^2,y^2,z^2,$

– 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

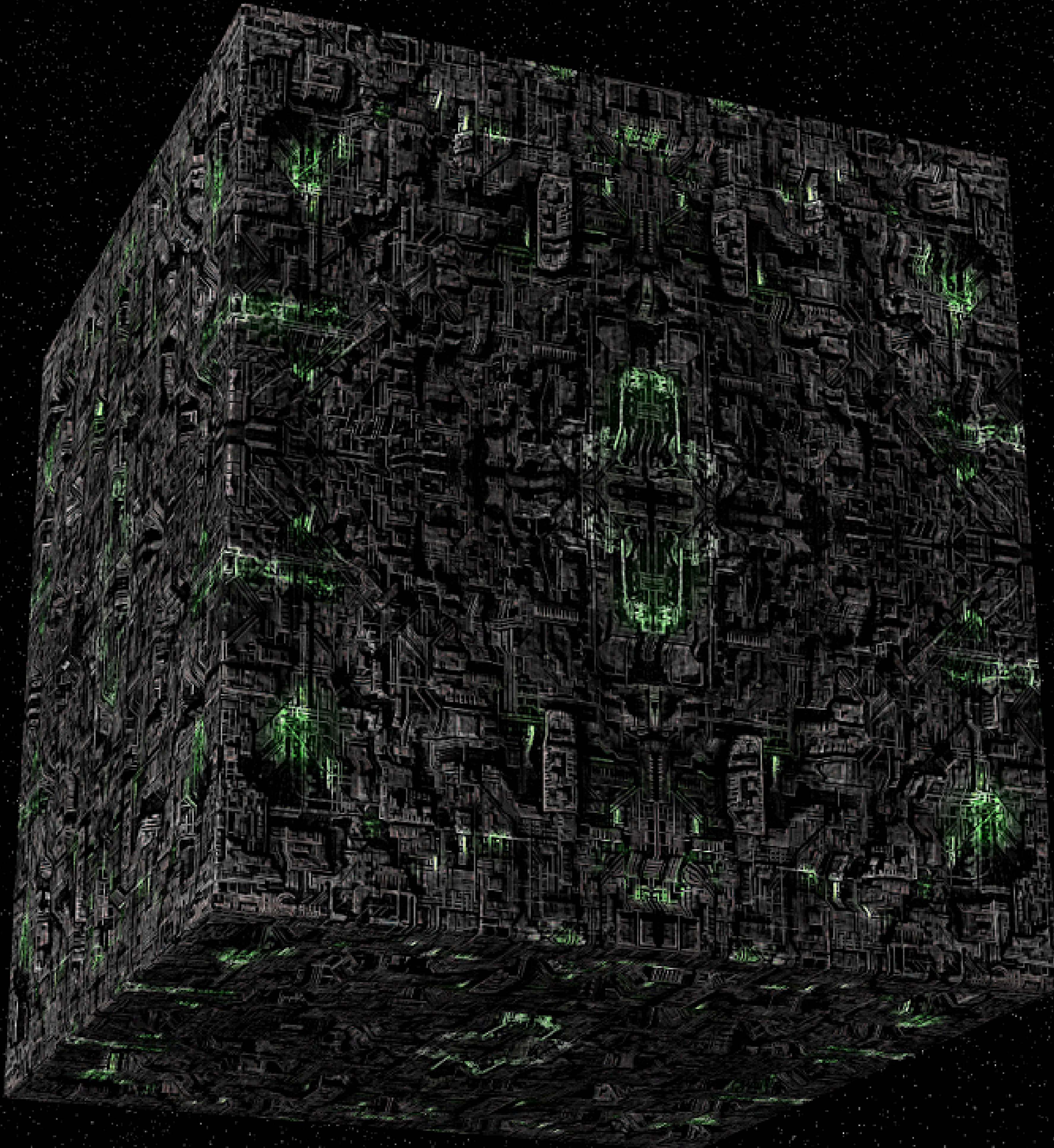
degree k	Number terms in complete poly	Number terms in tensor product
2	$\approx \frac{1}{2}n^2$	3^n
3	$\approx \frac{1}{6}n^3$	4^n

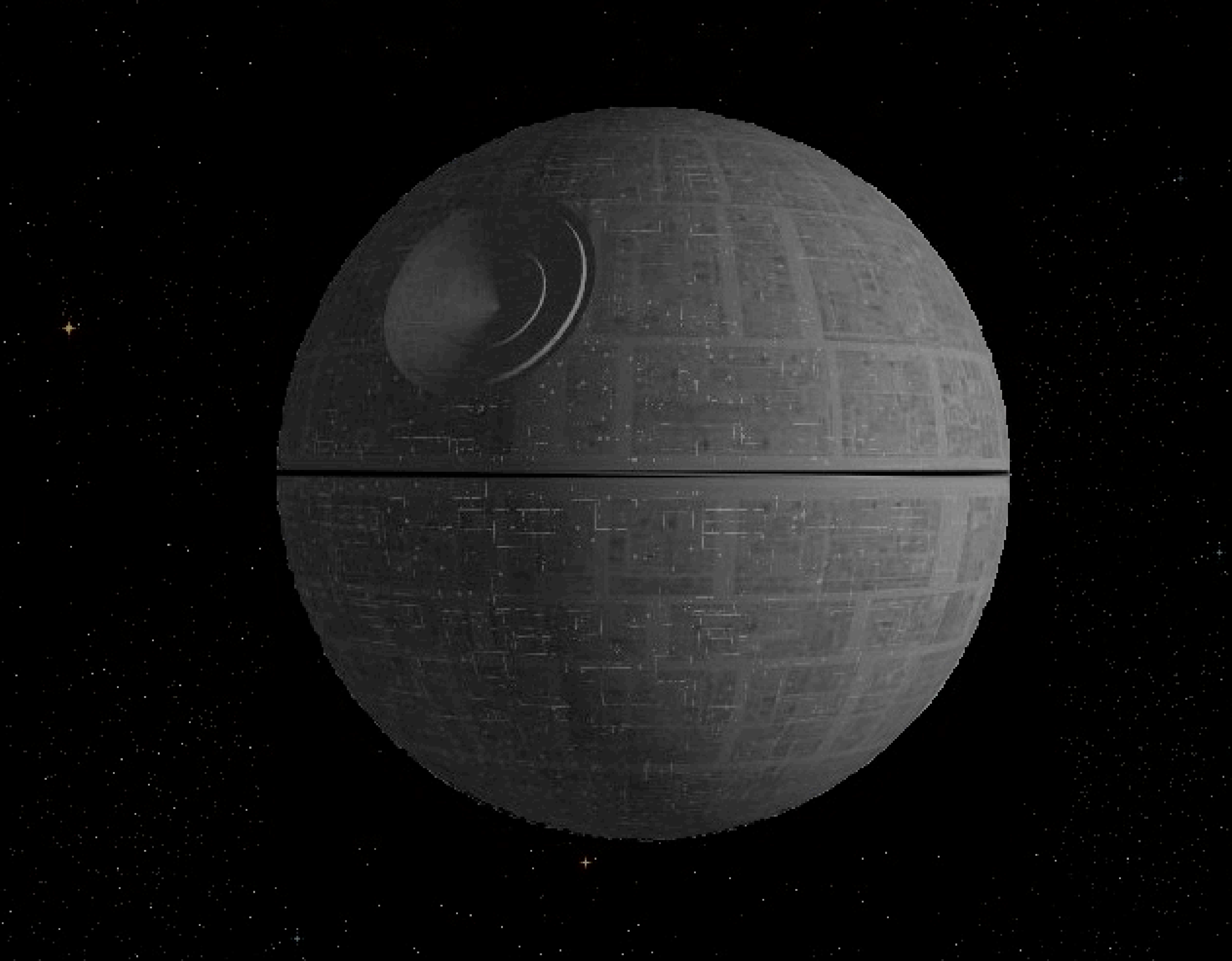
– See Gaspar-Judd (1997), Kubler-Krueger (2003).

- 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
$$\sum_{i=1}^{N} a_i \phi(\|x - p_i\|)$$
    - ϕ choices include
$$e^{-r^2}, \frac{1}{\sqrt{1+r^2}}, \frac{1}{1+r^2}, \sqrt{1+r^2}, \ldots$$
    - 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.
- Judd (1992), Gaspar-Judd (1997) made mechanical choice of hypercubes.

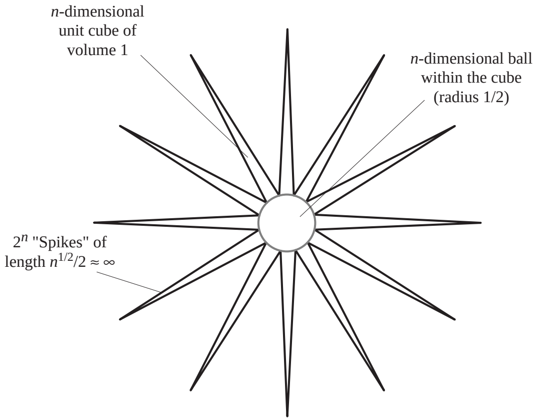


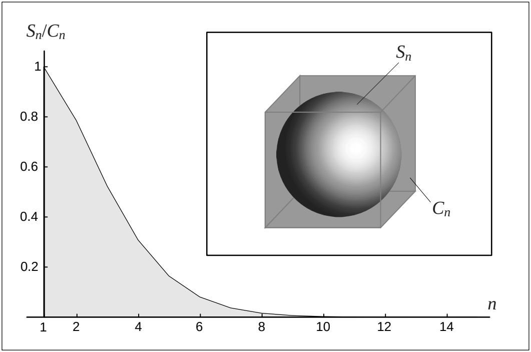


- Cube versus Sphere
  - Spheres are much more compact:
    - In cube of unit length at edge, length of longest diagonal is $n^{1/2}$
    - Ratio of sphere to cube volume is

$$\begin{array}{l} \frac{\pi^{n/2}}{(n/2)!}, \text{ n even} \\ \frac{2^{n/2+1}\pi^{n/2}}{1\cdot 3\cdot 5\cdot \ldots \cdot n}, \text{ n odd} \end{array}$$

    - 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


n-dimensional unit cube of volume 1
n-dimensional ball within the cube (radius 1/2)
2^n "Spikes" of length $n^{1/2}/2 \approx \infty$


S_n/C_n
1
0.8
0.6
0.4
0.2
1
2
4
6
8
10
12
14
n
S_n
C_n

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

    $$\frac{4}{3}\left(f(1,0,0) + f(-1,0,0) + f(0,1,0) + f(0,-1,0) + f(0,0,1) + f(0,0,-1)\right)$$

    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

    $$\int_{[-1,1]^d} f \doteq \omega \sum_{i=1}^{d} \left(f(ue^i) + f(-ue^i)\right),$$

    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 ω_i, $i = 1, .., m$, to create a quadrature rule, $Q(f; z, \omega)$, to minimize errors.
    - The literature is for one-dimensional problems:

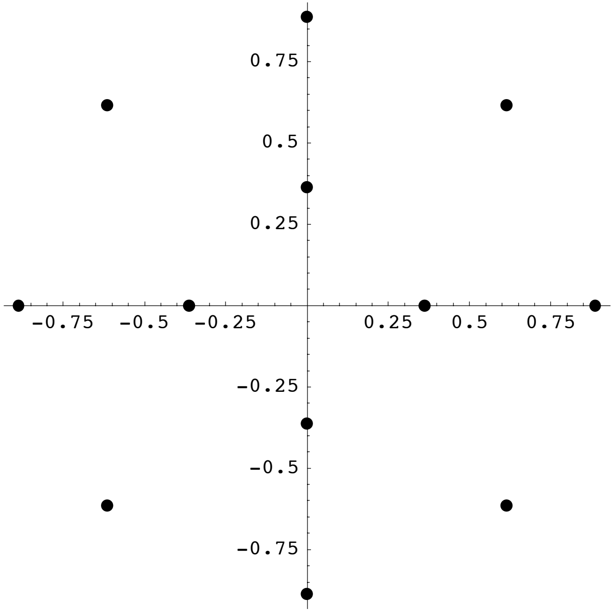
$$\min_{z,\omega} \sum_{i=0}^{\infty} \left(Q\left(x^i; z, \omega\right) - \int x^i \, dx \right)^2$$

    - 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

$$\begin{aligned} &0.2227\ (f[-0.8871, 0] + f[0, -0.8871] + f[0, 0.8871] + f[0.8871, 0]) \\ +&0.2735\ (f[-0.6149, 0.6149] + f[-0.6149, -0.6149] \\ &+f[0.6149, -0.6149] + f[0.6149, 0.61496]) \\ +&1.0744f[-0.3628, 0] + 1.0744f[0, 0.3628] \\ &+1.0744f[0, -0.3628] + 1.0744f[0.3628, 0] \end{aligned}$$

    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.



0.75
0.5
0.25
-0.25
-0.5
-0.75
-0.75
-0.5
-0.25
0.25
0.5
0.75

- 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

$$\begin{aligned}\int_{-1}^{1}\int_{-1}^{1} f(x,y)\,dx\ dy = & \sum_{i=1}^{n} \omega_{i,0} f(x_i, y_i) \\ & + \sum_{i=1}^{n} \left(\omega_{i,x} f_x(x_i, y_i) + \omega_{i,y} f_y(x_i, y_i)\right) \\ & + \sum_{i=1}^{n} \left(\omega_{i,xx} f_{xx}(x_i, y_i) + \omega_{i,xy} f_{xy}(x_i, y_i) + \omega_{i,yy} f_{yy}(x_i, y_i)\right)\end{aligned}$$

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

$$\int_0^1 f(x)\,dx \equiv \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{3}$$

- There are alternatives called quasi-Monte Carlo methods. Two simple examples in R^d are

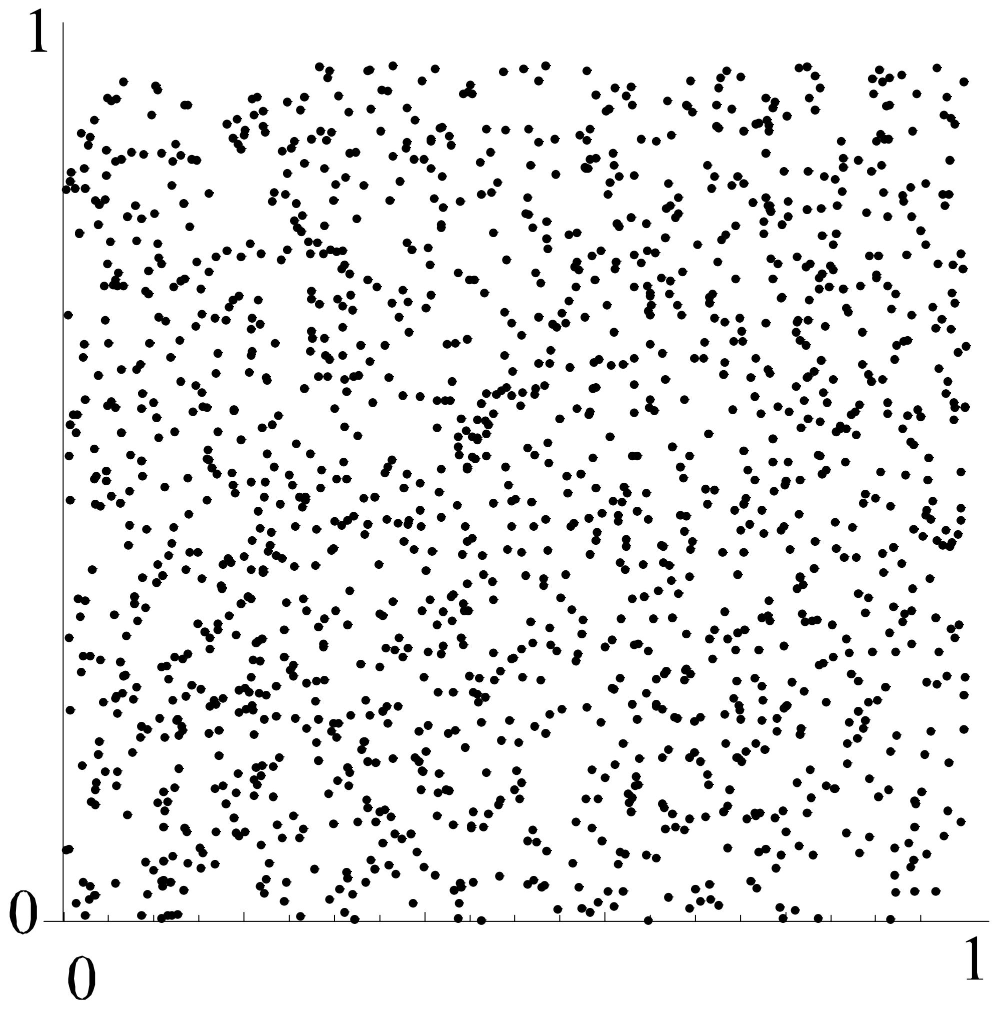
$$\begin{array}{ll} \text{Weyl:} & x^n = \left(n\,p_1^{1/2}, \cdots, n\,p_d^{1/2}\right) \bmod 1 \\ \text{Niederreiter:} & x^n = \left(n\,2^{1/(d+1)}, \cdots, n\,2^{d/(d+1)}\right) \bmod 1 \end{array}$$

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

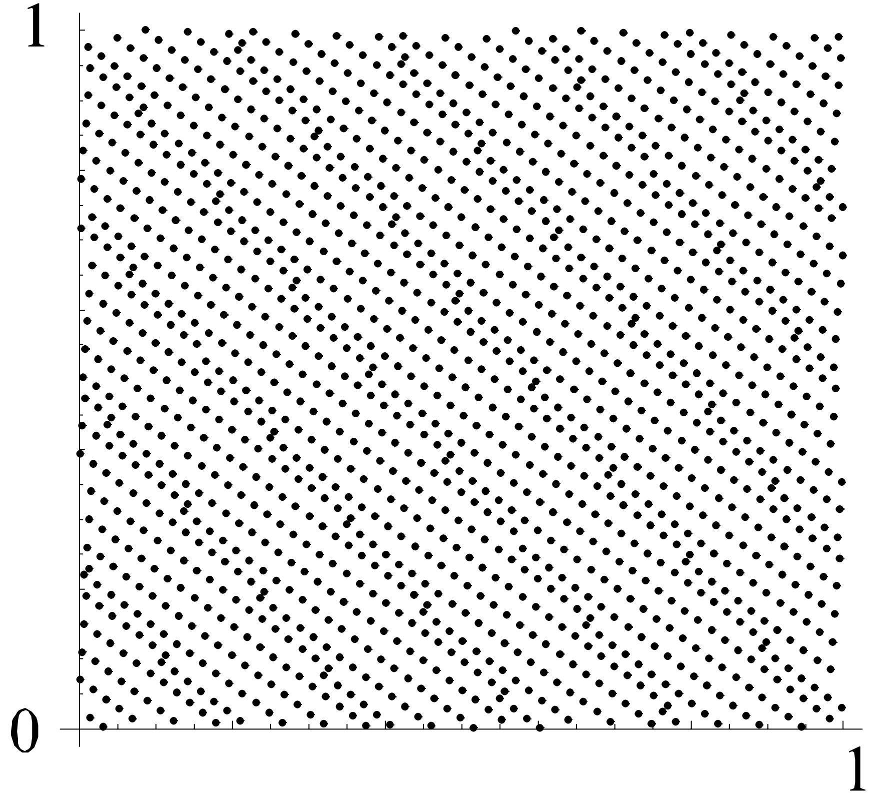
- The debate is which deterministic formulas we should use, not deterministic versus random.

  – MC sequences are designed to look like iid sequences; coincidentally, they do well at integration

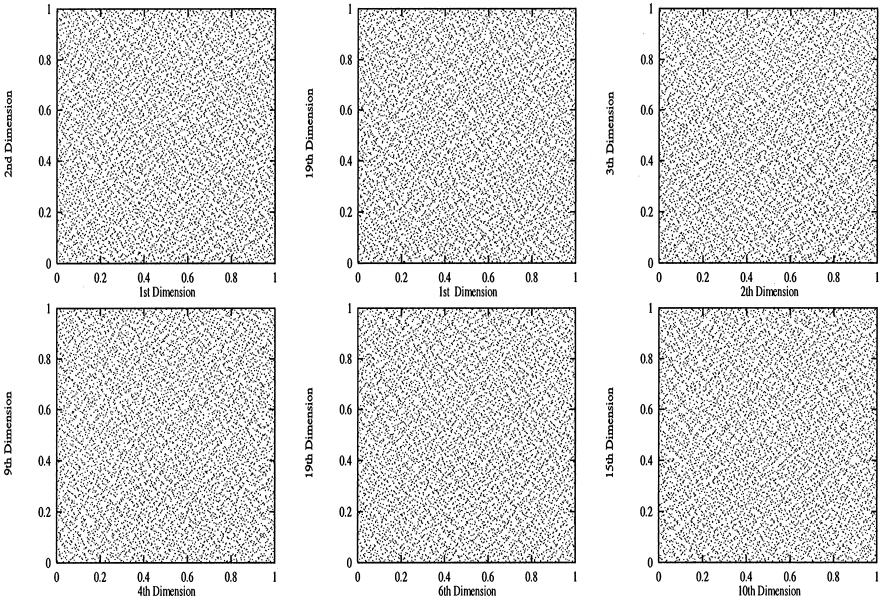
  – qMC sequences are designed to be uniformly distributed and to do well at integration



1500 Points generated by LCM



First 1500 Weyl points


2nd Dimension
1st Dimension
19th Dimension
1st Dimension
3th Dimension
2th Dimension
9th Dimension
4th Dimension
19th Dimension
6th Dimension
15th Dimension
10th Dimension

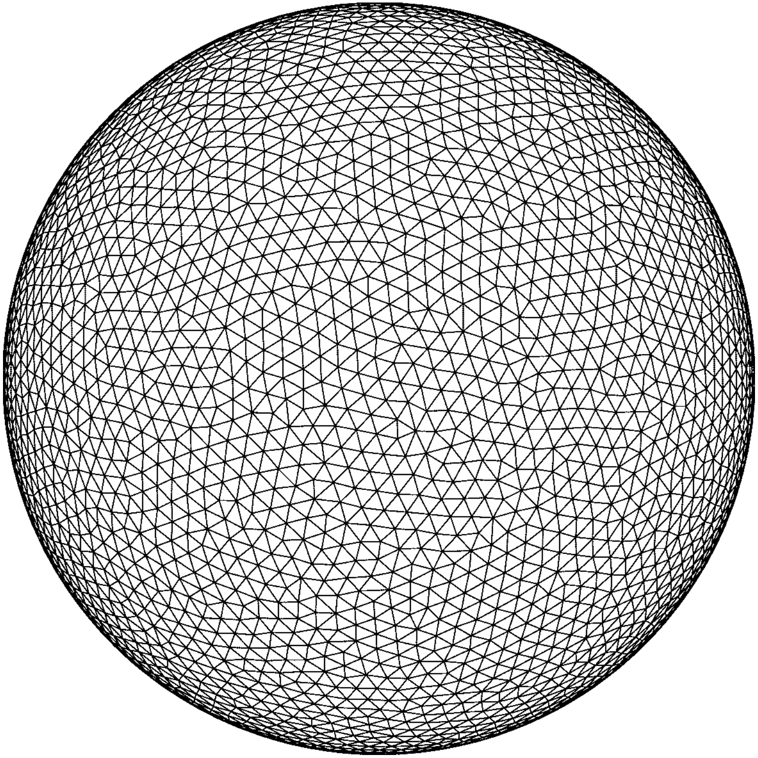


Figure 1. Computed extremal system for $n = 64$, $d_n = 4225$.

- 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 *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 $(x \sim N(0,1))$

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

    see Sandor-Andras (*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: $V\left(F\left(x,U\left(x\right)\right)\right)$ - double composition
  - Continuous-time: $V'\left(x\right)F\left(x,U\left(x\right)\right)$ - single composition plus multiplication and gradient
  - Stochastic discrete time: $E\left\{V\left(F\left(x_t,U\left(x_t\right)\right),\theta_{t+1}\right)|\theta_t\right\}$ - double composition plus multidimensional integral
  - Stochastic continuous time: $V'\left(x\right)F\left(x,U\left(x\right)\right)+\sigma^2V''\left(x\right)$ - 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
  $$\begin{aligned} V(k) &= \max_{I} u(c) + \beta V(k+I) \\ c &= \Sigma_i f(k_i) - \Sigma_i g^i\left(I^i(k)\right) \end{aligned}$$
  - Equations defining $V(k)$ and $I(k)$:
  $$\begin{aligned} V(k) &= u(c) + \beta V(k + I(k)) \\ 0 &= -u'(c)\left(1 + \alpha I^i(k)\right) + V_i(k + I(k)) \end{aligned}$$
  - 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 $2q$ independent
  - Solutions depend on N; take $N \rightarrow \infty$ 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,

$$u(c, W) = \begin{cases} u(c), W > 0 \\ -\infty, W < 0 \end{cases}$$

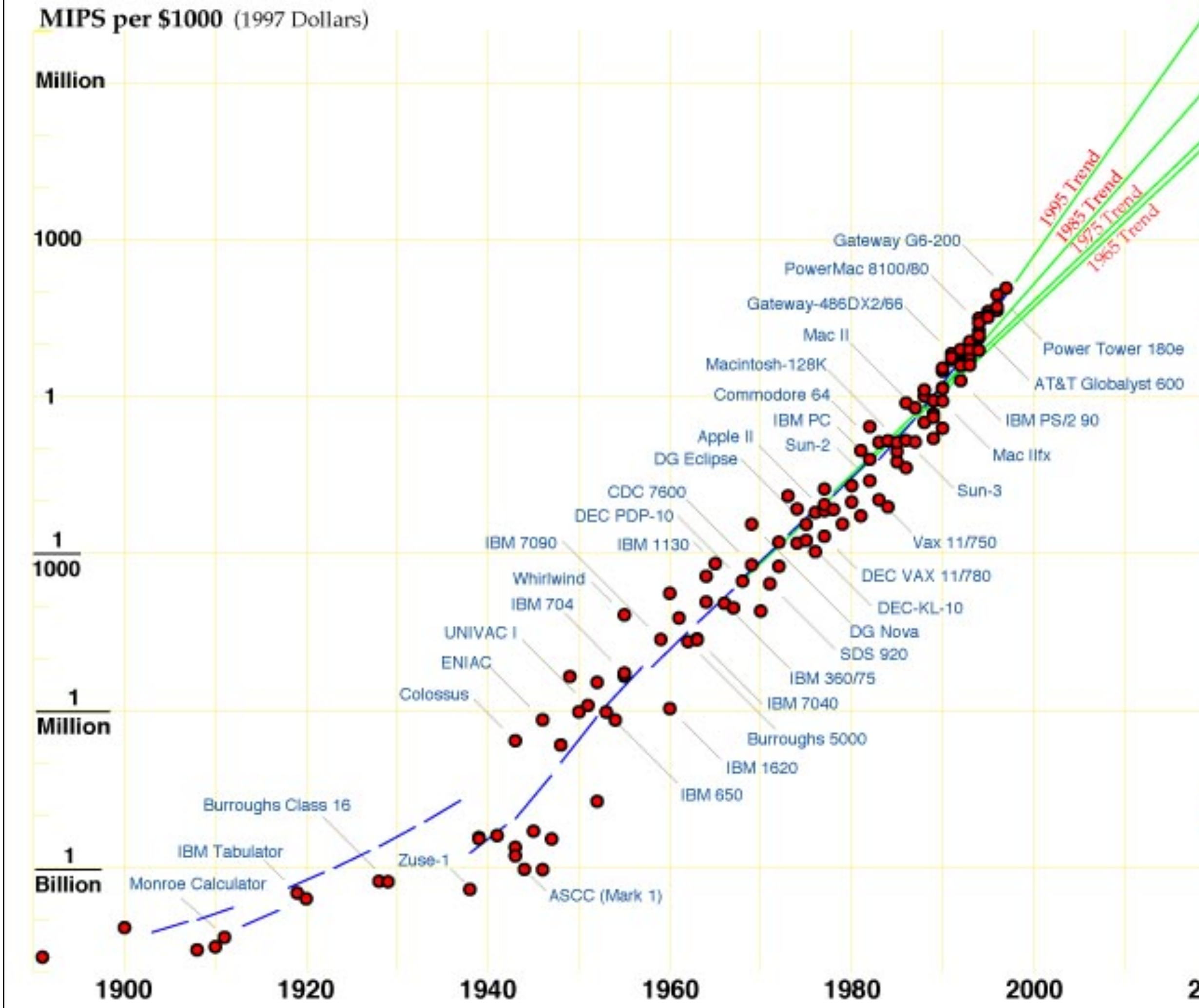
- Is this economically reasonable? Borrowing is not infinitely painful
  - First, go to parents.
  - Second, run up credit card debt.
  - In general, there is a set of sources of credit, with rising interest rates
  - Empirical fact: people do have debt!
- Results were interesting, but hampered by computational 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

- Citation: R.Glenn Hubbard and Kenneth L. Judd, “Liquidity Constraints, Fiscal Policy, and Consumption,” *Brookings Papers on Economic Activity*, 1986: 1.

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.


MIPS per $1000 (1997 Dollars)
Million
1000
1
1/1000
1/Million
1/Billion
1900
1920
1940
1960
1980
2000
1995 Trend
1985 Trend
1975 Trend
1965 Trend
Gateway G6-200
PowerMac 8100/80
Gateway-486DX2/66
Mac II
Macintosh-128K
Commodore 64
IBM PC
Apple II
Sun-2
DG Eclipse
CDC 7600
DEC PDP-10
IBM 7090
IBM 1130
Whirlwind
IBM 704
UNIVAC I
ENIAC
Colossus
Burroughs Class 16
IBM Tabulator
Monroe Calculator
Zuse-1
ASCC (Mark 1)
Power Tower 180e
AT&T Globalyst 600
IBM PS/2 90
Mac IIfx
Sun-3
Vax 11/750
DEC VAX 11/780
DEC-KL-10
DG Nova
SDS 920
IBM 360/75
IBM 7040
Burroughs 5000
IBM 1620
IBM 650

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 Integrationand 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.,

$$\int_{[0,1]^n} g\left(x\right) dF\left(x\right) = \int_{[0,1]} \dots \int_{[0,1]} \int_{[0,1]} g\left(x\right) dF_1\left(x_1\right) dF_2\left(x_2\right) \dots dF_n\left(x_n\right)$$

  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 $f(i)$, $i = 1, .., n$
  - 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 dimensionlity.
  - "Path Integration on a Quantum Computer," Traub and Wozniakowski (2001).
    - Path integration on a quantum computer is tractable.
    - Path integration on a quantum computer can be solved roughly ε^{-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.