Numerical Methods-Lecture IX: Quadrature (and Markov Chains)

(See Judd Chapter 7, Stokey Lucas Prescott Chapter 11)

Trevor Gallen

Motivation

- $E(x) = \int_{-\infty}^{\infty} xf(x)$ for x with PDF $f(x)$
- Most Bellman Problems look something like:

$$V(x) = \max_{y} \{\phi(x, y) + \beta E(V(x))\}$$

- One option is to force our probability space to be easy to use
- Another is to integrate properly (at least, within some tolerance)

Easy to use probability spaces

- Markov chains are wonderfully simple
- Chapter 11 of RMED
- Allow your states to evolve stochastically, within bounds
- Summarize probability of transition as a matrix

Markov Chains

- We have some state (income, say) that consists of a finite number of elements:

$$S = \{s_1, s_2, ..., s_n\}$$

- We define the probability of transitioning from state i to state j by probability π_{ij} in row i, column j:

$$\Pi = \begin{bmatrix} \pi_{1\to1} & \pi_{1\to2} & \cdots & \pi_{1\to n} \\ \pi_{2\to1} & \pi_{2\to2} & \cdots & \pi_{2\to n} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n\to1} & \pi_{n\to2} & \cdots & \pi_{n\to n} \end{bmatrix}$$

Properties of Markov Chains

$$\Pi = \begin{bmatrix} \pi_{1\to 1} & \pi_{1\to 2} & \cdots & \pi_{1\to n} \\ \pi_{2\to 1} & \pi_{2\to 2} & \cdots & \pi_{2\to n} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n\to 1} & \pi_{n\to 2} & \cdots & \pi_{n\to n} \end{bmatrix}$$

- All rows must add up to 1!
- If your position is summarized by row vector V, then the probability you'll be in each state next period is given by:

$$V_{t+1} = V_t \Pi$$

- This is true for multiple iterations of π:

$$V_{t+1} = V_t \Pi^2$$

- It's easy to find the invariant distribution (if it exists) as:

$$\lim_{n\to\infty} V_t \Pi^n$$

Markov Chain: Death Example

$$\Pi = \begin{bmatrix} 0 & 0.9 & 0 & 0 & 0 & 0 & 0 & 0 & 0.1 \\ 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 & 0.7 \\ 0 & 0 & 0 & 0 & 0.96 & 0 & 0 & 0 & 0.04 \\ 0 & 0 & 0 & 0 & 0 & 0.6 & 0 & 0 & 0.6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0.7 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.7 & 0.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- Where I interpret entries as $\{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, Death\}$
- Interpret

Markov Chain: Death Example

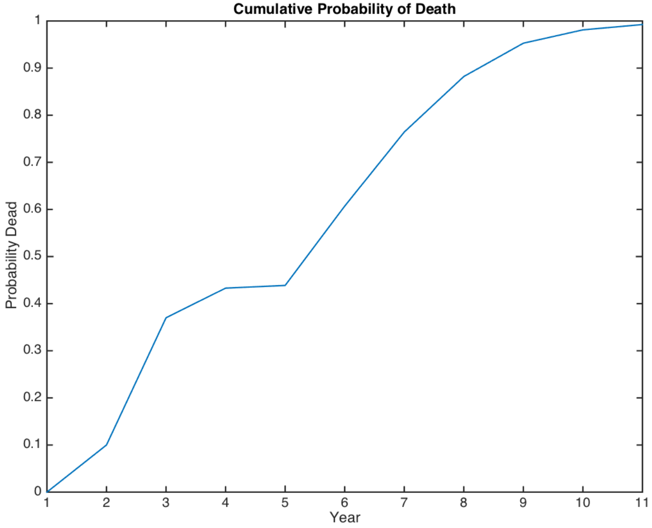
- Starting at

$$V_0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Apply Π repeatedly:

$$\begin{aligned}
V_1 &= [\ 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ] \\
V_2 &= [\ 0 \quad 0.9 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.1\ ] \\
V_3 &= [\ 0 \quad 0 \quad 0.63 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.37\ ] \\
V_4 &= [\ 0 \quad 0 \quad 0 \quad 0.57 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.43\ ] \\
V_5 &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0.56 \quad 0 \quad 0 \quad 0 \quad 0.44\ ] \\
V_6 &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.39 \quad 0 \quad 0 \quad 0.61\ ] \\
V_7 &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.24 \quad 0 \quad 0.76\ ] \\
V_8 &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.12 \quad 0.88\ ] \\
V_9 &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.05 \quad 0.95\ ] \\
V_{10} &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.02 \quad 0.98\ ] \\
V_{11} &= [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0.01 \quad 0.99\ ]
\end{aligned}$$

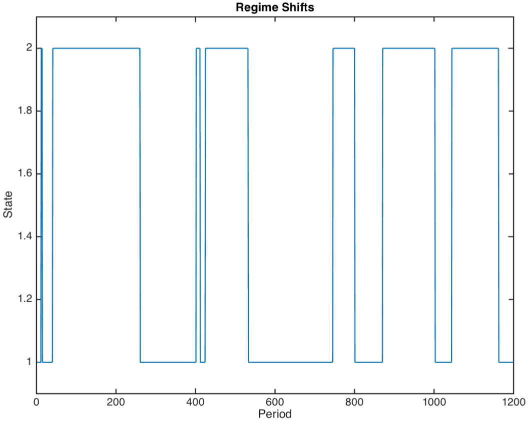
Markov Chain: Death Example



Regime Shifts

- We could also model persistent "regime shifts"

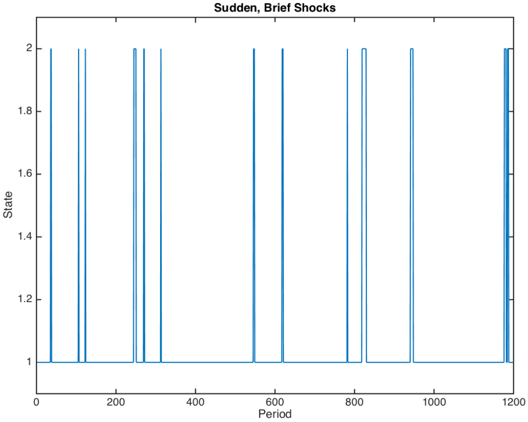
$$\Pi = \begin{bmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{bmatrix}$$



Sudden Brief Shocks

- We could also model sudden and brief shocks

$$\Pi = \begin{bmatrix} 0.99 & 0.01 \\ 0.3 & 0.7 \end{bmatrix}$$



Period of Uncertainty

- We could model a "time of uncertainty"

$$\Pi = \begin{bmatrix} 0.99 & 0.01 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.01 & 0.99 \end{bmatrix}$$

- You could imagine a model of investment under uncertainty
- When in state 1 or 3, you know the deal
- If in state 2, wait to find out what state you're in next period

Cyclical Behavior

- Some stochastic behavior is cyclical

$$\Pi = \begin{bmatrix} 0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 & 0.5 \\ 0.5 & 0 & 0 & 0 & 0.5 \end{bmatrix}$$

- The real minimum wage doesn't look dissimilar to this (if you assign states properly)

Markov Chains: Summary

- Very flexible, incredibly easy to use, program, simulate
- Easy to estimate using maximum likelihood
- Good for learning problems or any regime shifting problems
- Require discrete states
- Easy to "integrate" and get probabilities
- Don't leave home without them

How to approximate time series with Markov chains

- Can get closed-form solutions using "Tauchen's method"
- Choose transition matrix and points to match CDFs
- But a good opportunity to show minimization techniques
- Let's try to match $y_t = (1-\rho)\overline{y} + \rho y_{t-1} + \epsilon$, with $\epsilon \sim \mathcal{N}(0,1)$ to a 5x5 Markov process

How to approximate time series with Markov chains

- So:

$$P = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \theta_4 & \theta_5 \\ \theta_6 & \theta_7 & \theta_8 & \theta_9 & \theta_{10} \\ \theta_{11} & \theta_{12} & \theta_{13} & \theta_{14} & \theta_{15} \\ \theta_{16} & \theta_{17} & \theta_{18} & \theta_{19} & \theta_{20} \\ \theta_{21} & \theta_{22} & \theta_{23} & \theta_{24} & \theta_{25} \end{bmatrix} \quad s = \begin{bmatrix} \theta_2 6 \\ \theta_{27} \\ \theta_{28} \\ \theta_{29} \\ \theta_{30} \end{bmatrix}$$

- Choose parameters to minimize simulation time series difference compared to actual time series difference

How to approximate time series with Markov chains

- Choose θ to minimize the sum of squared differences for:
  - Mean (mean of values)
  - Unconditional variance (range of values)
  - One-step variance (ρ and shock variance)
  - Two-step variance (ρ, given shock variance)
  - Skewness (force a rough symmetry)
  - Kurtosis (endpoints)

How to approximate time series with Markov chains

- See TauchenLike.m

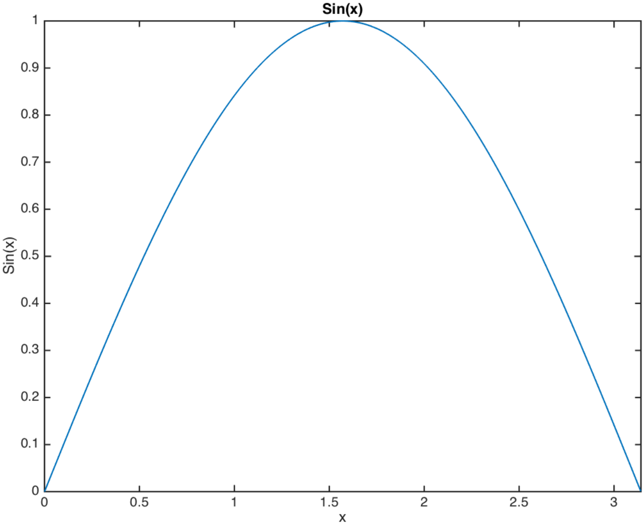
Quadrature

- Frequently, one wants to use continuous probability distributions
- It turns out there are a bunch of rules that get us very accurate integrals from a finite sampling of points
- Theoretically, you all basically learned one method ~ 5 years ago...

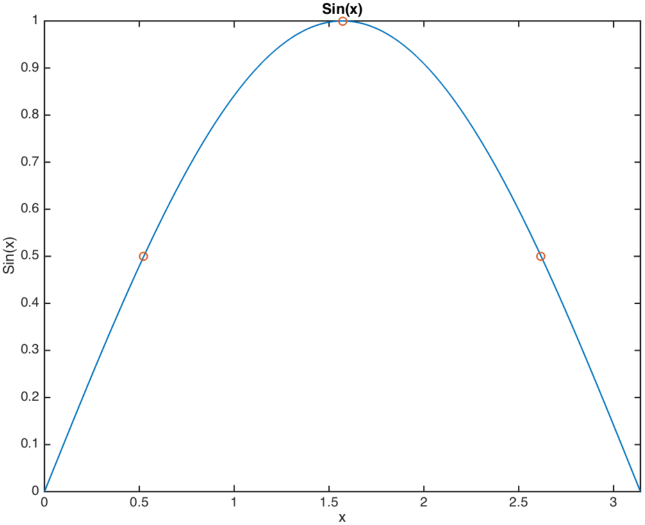
Quadrature: Basic Idea

- Let's say we want to integrate $\sin(x)$ over a uniform distribution from $(0, 2\pi)$
- Take 3 points: $\frac{\pi}{3}$, π, and $\frac{5}{3}\pi$, assign the surrounding $\frac{\pi}{3}$ on both sides to them.

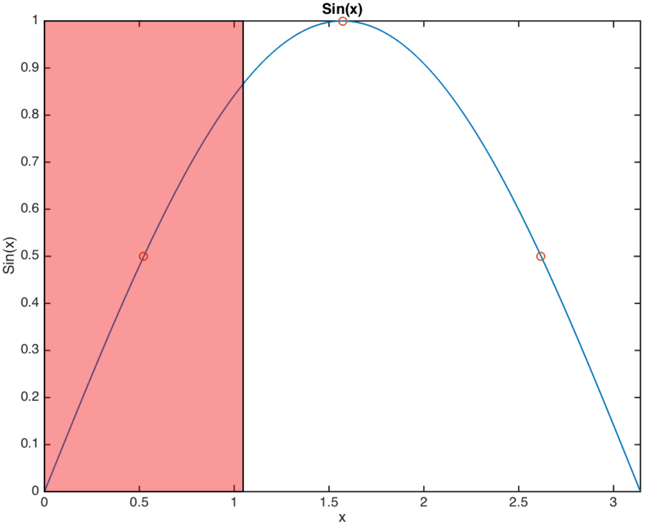
Quadrature: Basic Idea



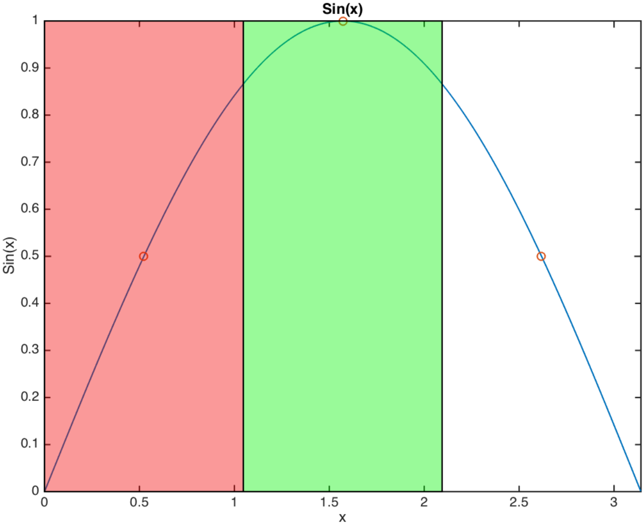
Quadrature: Basic Idea



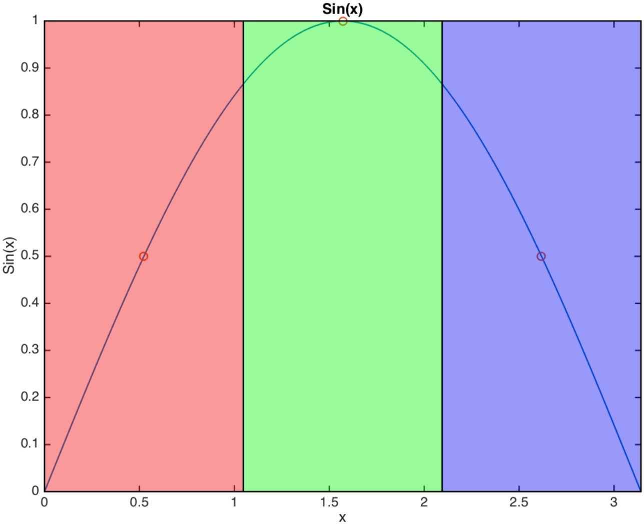
Quadrature: Basic Idea



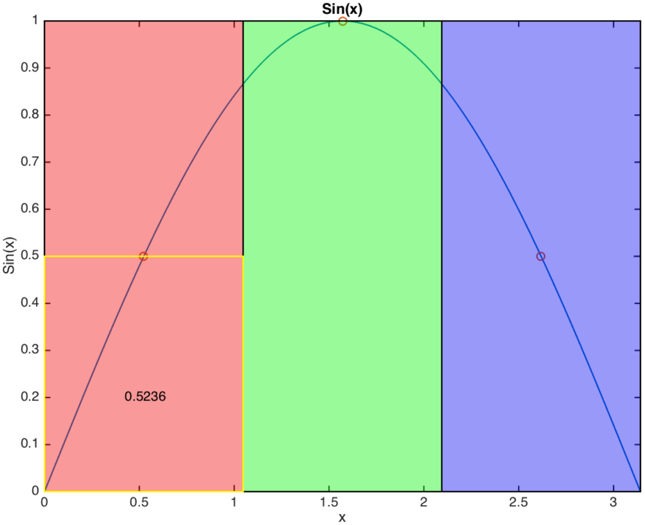
Quadrature: Basic Idea



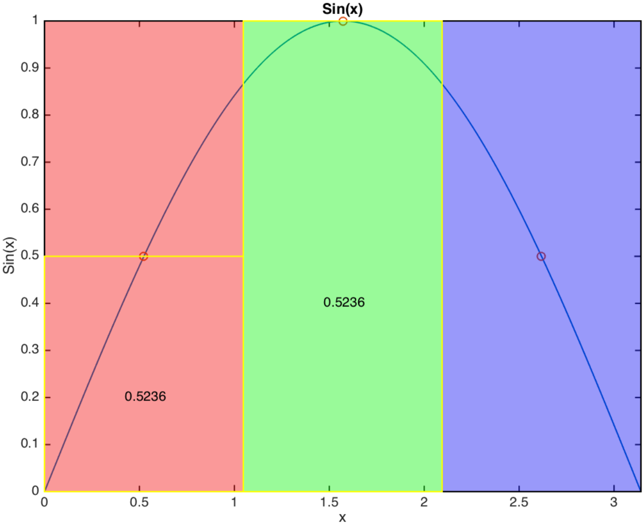
Quadrature: Basic Idea



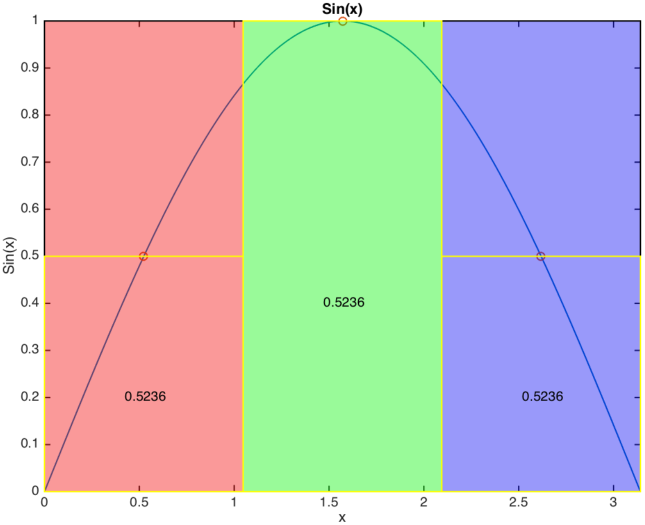
Quadrature: Basic Idea



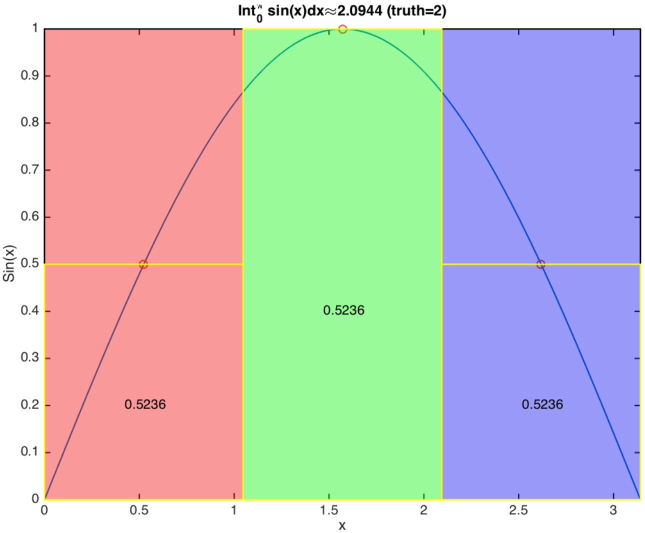
Quadrature: Basic Idea



Quadrature: Basic Idea



Quadrature: Basic Idea



We can do better

- Two choices: where points are, and weights of points
- We chose equal weights, equidistant points
- Simpson's rule takes better weights:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

- Break up into six points, heavily weight the middle
- This results from a quadratic approximation
- Simpson's rule is a special case of Newton-Coates

Newton-Coates

- If we're given $f(x)$ at a series of equidistant points, but control the weights, how well can we do?
- Trapezoid rule is as good as you'll do with 2 equidistant points
- Simpson's rule is as good as you'll do with 3 equidistant points.
- Can look up arcane rules for higher-degree approximations
- Note that if you interpolate and integrate yourself, vulnerable to Runge's phenomeon
- Picking your own points and weights opens up a whole new ballgame

Quadrature

- Equidistant methods can go off the rails
- Two common types of quadrature
  - Clenshaw-Curtis/Fejer Quadrature: use Chebyshev points (roots, extrema)
  - There are actually three types, depending on whether you use Chebyshev roots or extrema
    - Chebyshev roots (no extrema): Fejer's first rule (we'll do this)
    - Chebyshev extrema (not including endpoints): Fejer's second rule
    - Chebyshev extrema (including endpoints): Clenshaw Curtis
  - Gaussian: find optimal points
- For a long time, Clenshaw-Curtis got a short shrift
- "Half as efficient"
- Trefethen (2008) suggests can be roughly just as good, given bounds

Fejer's Rule

- Fejer's rule works with the interpolation we've been doing
- Integrate with Chebyshev polynomials on Chebyshev nodes
- This is pretty convienient if we were already interpolating using Chebyshev polynomials
- We can use a fast Fourier transform and trigonometric definitions rather than recursive (shortcuts)

Chebyshev

- Use chebfull to get "fits" (c_k) and integration weights (w_k)
- Interpolation formula

$$f(x) \approx \sum_{k=1}^{n} c_k cos(k \cdot arcos(x))$$

- Integration formula

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} w_k f(z_k)$$

- See my chebfull.m and QuadratureExample.m for an example.

Gaussian Quadrature

- We want: $\int_a^b f(x)$
- As with interpolation, we state the problem as:

$$\int_{-1}^{1} f(x)dx = \sum_{i=1}^{n} w_i f(x_i)$$

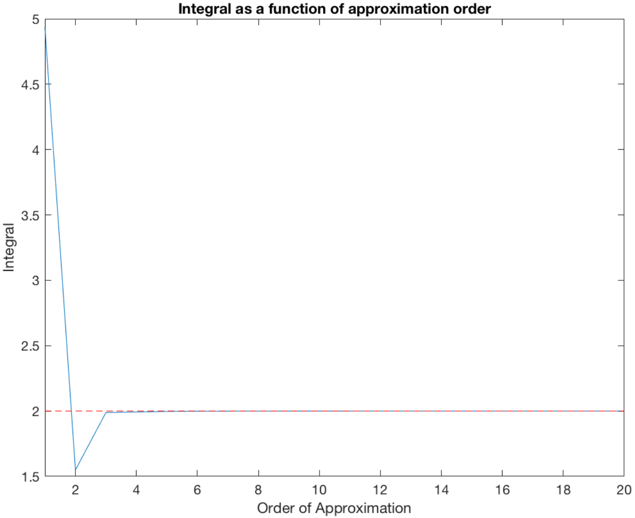
- Our choice of points and weights are determined by our polynomials
- Several
  - Gauss-Legendre (Legendre roots and polynomials)
  - Chebyshev-Gauss (Chebyshev points and polynomials)
  - Gauss-Hermite (Hermite roots and polynomials)
  - Gauss-Jacobi (Jacobi roots and polynomials)
- Most of these are only technically difficult, not difficult to implement once you get the formula

Example: $\sin(x)$ with Chebyshev-Gauss

- Formula:

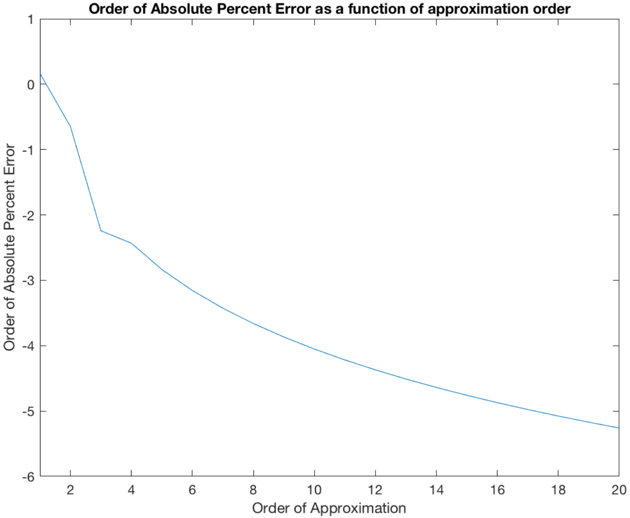
$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}} dx \approx \sum_{i=1}^{n} \frac{\pi}{n} f\left(cos\left(\frac{2i-1}{2n}\pi\right)\right) \sqrt{1-\left(cos\left(\frac{2i-1}{2n}\pi\right)^2\right)}$$

EXAMPLE: $\sin(x)$ FROM 0 TO PI



Note: See Quadrature.m for details.

Example: $\sin(x)$

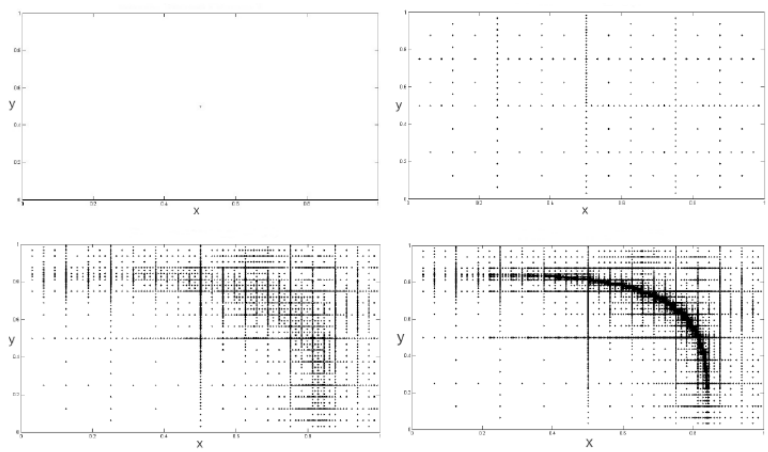


Note: See Quadrature.m for details.

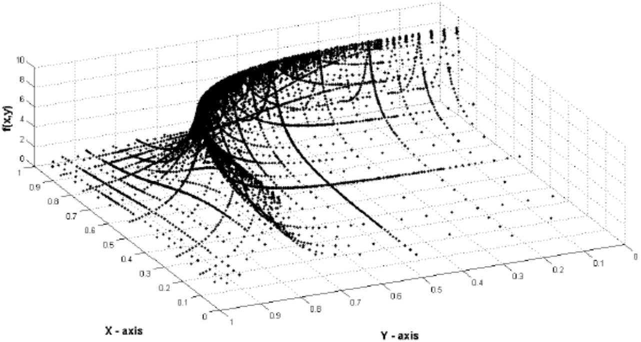
Matlab

- There's a new guy in town
- Don't exert thousands of (particularly valuable) man-hours picking the best polynomials and best points to maximize computational efficiency and running horse races
- Nested sets of points to narrow things down is one option
- Or, zoom in on trouble spots, make a more fine approximation there (Adaptive quadrature)
- Adaptive sparse grid interpolation
- Matlab: Sparse Grid Interpolation Toolbox by Andreas Klimke

Brumm & Scheidegger (2015)



Brumm & Scheidegger (2015)



Judd, Maliar, Maliar, and Valero

- Taking every point in grid is inefficient
- Good quick easy toolbox to interpolate and integrate functions on hypercubes
- Sparse grids can allow you to up your dimensions dramatically
- What does the grid look like?

Judd, Maliar, Maliar, and Valero

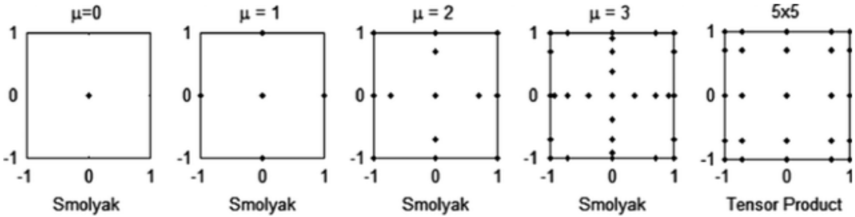


Fig. 1. Smolyak grids versus a tensor-product grid.

Table 1
Number of grid points: tensor-product grid with 5 points in each dimension versus Smolyak grids.

d	Tensor-product grid with 5^d points	Smolyak grid		
		$\mu=1$	$\mu=2$	$\mu=3$
1	5	3	5	9
2	25	5	13	29
10	9,765,625	21	221	1581
20	95, 367, 431, 640, 625	41	841	11,561

Later in the Quarter: Monte Carlo Methods

- Analytical integration isn't always possible
- Numerical quadrature frequently focuses on and has good properties in a few dimensions
- Monte Carlo integration has become popular
- We'll talk about this a little later, but the idea is simple
- "Randomly"* walk around the space under the integral and sample points
- Your "sample" will be representative of the "population" so you can add it up, average it, etc.
- This can be hundreds of thousands of times less efficient, but it's also easy