

# Numerical Dynamic Programming

Chris Conlon

Grad IO

October 26, 2017

# Reading

- ▶ Rust 1994 "Numerical Dynamic Programming in Economics" (Handbook Chapter)
- ▶ SLP Chapter 9: Stochastic Dynamic Programming (Theory)
- ▶ Judd Chapter 6: Approximation Methods
- ▶ Judd Chapter 7: Numerical Integration
- ▶ Judd Chapter 11: Projection Methods
- ▶ Judd Chapter 12: Numerical Dynamic Programming

# MDP Definition

- ▶ A time index  $t \in \{0, 1, 2, \dots, T\}$ , with  $T \leq \infty$
- ▶ A state space  $S$
- ▶ A decision space  $A$
- ▶ A family of constraint sets  $A_t(s_t) \subseteq A$
- ▶ A family of transition probabilities  $p_{t+1}(\cdot | s_t, a_t)$ , and  $\mathfrak{B}(S) \rightarrow [0, 1]^1$
- ▶ A family of discount  $\beta_t(s_t, a_t) \geq 0$  and a single period utility function  $u_t(s_t, a_t)$  such that the utility functional  $U$  has the additively separable decomposition

$$U(s, a) = \sum_{t=0}^T \left[ \prod_{j=0}^{t-1} \beta_j(s_j, a_j) \right] u_t(s_t, a_t)$$
$$\alpha(s) = \arg \max_{\delta=(a_0, \dots, a_T)} E_a U(s, a)$$

# Characterizations of MDPs

**Finite Horizon** have  $T < \infty$ . Can compute  $a^*$  by backward induction starting in the terminal period  $T$ .

**Infinite Horizon**  $T = \infty$  use a recursive definition of the value function

**Discrete State Space** solve problems up to machine precision. (Good for estimation). How realistic?

**Infinite State Space** most of economics, numerical approximation  $\rightarrow$  errors and interactions of approximation errors.

**Discrete Decision Process**  $D$  takes on a discrete set of values  $\{a_1, \dots, a_J\}$

**Continuous Decision Process (CDP)** These are tricky – there are results about how well they can be approximated by DDPs and they are not overwhelmingly positive. (See Rust 1994).

# Examples of MDPs

- ▶ Entry/Exit decisions of firms
- ▶ R&D Investment
- ▶ Replacement of Durables
- ▶ Consumer Search
- ▶ Advertising?

# Bellman's Equation

It is helpful to consider  $V(s)$  as the solution to the MDP.

$$V(s) = \max_{a \in A(s)} [u(s, a) + \beta \int V(s') p(ds' | s, a)]$$

This is a *functional equation* and  $V$  represents a *fixed point* to the functional equation.

We are interested in existence/uniqueness of a solution to Bellman's Equation:

1.  $A$  and  $S$  are complete metric spaces
2.  $u(s, a)$  is jointly continuous and bounded in  $(s, a)$
3.  $s \rightarrow A(s)$  is a continuous correspondence

# Bellman Operator

Helpful to rewrite Bellmans equation as an operator  $V = \Gamma(V)$   
where  $\Gamma : \mathfrak{B}(S) \rightarrow \mathfrak{B}(S)$  (Measurable, bounded)

The Bellman operator has the contraction mapping property which guarantees a unique fixed point point in  $B$ .

$$\|\Gamma(V) - \Gamma(W)\| \leq \beta \|V - W\| \quad \forall V, W \in B$$

## Blackwell's Theorem

*The stationary, Markovian, infinite horizon policy given by  $\alpha(s)$  (the solution to Bellman's equation) constitutes an optimal decision rule for the infinite horizon MDP.*

# Simple Case

Start by thinking about *Discrete Decision Problems* with a *Discrete State Space*. We can generalize to harder problems later.



# Solution Method: Finite Horizon Problems

- ▶ The optimal decision depends only on the state  $a_t = a(s_t)$  not the whole history because  $p$  is markovian flow utilities are additively separable.
- ▶ See this by thinking about choosing a unique  $a_T(s_t)$  in the ultimate period (unless there are ties) hence randomization can only make the agent worse off.
- ▶ All prior  $d_t$ 's are now deterministic functions of  $s_t$  and so on
- ▶ At time  $t = 0$  the value function  $V_0^T(s_0)$  represents the conditional expectation of utility over all future periods> it follows that  $V_0^T(s) = \max_a E_a\{U(\tilde{s}, \tilde{d}) | s_0 = s\}$ .

# Solution Method: Value Function Iteration

- ▶ Well known, basic algorithm of dynamic programming
- ▶ Based on our backward induction method of solving finite-horizon problems.
- ▶ We have tight convergence properties and bounds on errors
- ▶ Well suited for parallelization
- ▶ It is linearly convergent (slow depends on modulus of contraction mapping).

# Value Function Iteration: Finite Horizon

Begin with the Bellman operator:

$$\Gamma(V^t)(s) = \max_{a \in A(s)} \left[ u(s, a) + \beta \int V^t(s') p(ds'|s, a) \right]$$

Specify  $V^T$  and apply the Bellman operator:

$$\Gamma(V^{t-1})(s) = \max_{a \in A(s)} \left[ u(s, a) + \beta \int V^T(s') p(ds'|s, a) \right]$$

Iterate to first period:

$$\Gamma(V^1)(s) = \max_{a \in A(s)} \left[ u(s, a) + \beta \int V^2(s') p(ds'|s, a) \right]$$

# Value Function Iteration: Infinite Horizon

Begin with the Bellman operator:

$$\Gamma(V^t)(s) = \max_{a \in A(s)} \left[ u(s, a) + \beta \int V^t(s') p(ds' | s, a) \right]$$

Specify  $V^0$  and apply the Bellman operator:

$$V^1(s) = \max_{a \in A(s)} \left[ u(s, a) + \beta \int V^0(s') p(ds' | s, a) \right]$$

Iterate until convergence  $\sup_s \|\Gamma(V^t)(s) - V^t(s)\| \leq \epsilon$

$$V^T(s) = \max_{a \in A(s)} \left[ u(s, a) + \beta \int V^{T-1}(s') p(ds' | s, a) \right]$$

## Value Function Iteration: Bounds

- ▶ Suppose we set  $V_0 = 0$  then the value function iteration approach is just like solving the finite horizon problem by backward induction.
- ▶ The CMT guarantees consistency at a geometric rate or *linear* convergence with modulus  $\beta$
- ▶ We can derive an expression for the number of steps we need to get an  $\epsilon$ -approximation.

$$T(\epsilon, \beta) = \frac{1}{|\log(\beta)|} \log \left( \frac{1}{(1 - \beta)\epsilon} \right)$$

# Initial Value in Finite Horizon

- ▶ Economics provides natural choices (sometimes)
- ▶ Example: value of not being in the market is zero
- ▶ There are subtle issues: What is the value of dying? Bequests? OLG?

# Initial Guesses in Infinite Horizon

- ▶ Theorems tell us we will converge from any initial guess
- ▶ Doesn't mean we should choose bad guesses!
- ▶ Try the steady state.
- ▶ Reduce dimensions:  $K/L$  ratio in Solow model

# Policy Iteration (Howard 1960)

An alternative to value function iteration is policy function iteration.

- ▶ Make a guess for an initial policy, call it  $a(s) = \arg \max_a U(a, s)$  that maps each state into an action
- ▶ Assume the guess is stationary compute the implied  $V(a, s)$
- ▶ Improvement Step: improve on policy  $a_0$  by solving

$$a' = \arg \max_a U(a, s) + \beta \sum_{s'} V(a, s') p(s'|s, a)$$

- ▶ Helpful to define  $p(\tilde{s}'|s)$  as transition probability under optimal choice  $a(s)$ .
- ▶ Determine if  $\|a' - a\| < \epsilon$ . If yes then we have found the optimal policy  $a^*$  otherwise we need to go back to step 2.



# Policy Iteration (Howard 1960)

Policy Iteration is even easier if choices AND states are discrete.

- ▶ For Markov transition matrix  $\sum_j P_{ij} = 1$ , we want  $\pi P = \pi$
- ▶  $\lim_{t \rightarrow \infty} P^t = \pi$  where the  $j$ th element of  $\pi$  represents the long run probability of state  $j$ .
- ▶ We want the eigenvalue for which  $\lambda = 1$ .

Now updating the value function is easy for  $k$ th iterate of PI

$$\begin{aligned} V^k(s) &= Eu(a^k(s), s) + \beta \tilde{P}^k V^k(s) \\ \Rightarrow V^k(s) &= [1 - \beta \tilde{P}^k]^{-1} Eu(a^k(s), s) \end{aligned}$$

- ▶ Very fast when  $\beta > 0.95$  and  $s$  is relatively small. (Rust says 500 more like 5000).
- ▶ Inverting a large matrix is tricky

## More Comments

- ▶ In the DDP case convergence of PI easy to verify:  $a^{t+1} = a^t$ .
- ▶ It is helpful to exploit the monotonicity of policy or value function: (S-s Rules, bus replacement, etc.)
- ▶ Exploit Concavity of value and/or policy functions (decreasing return to R&D investment, etc.)

# Linear Programming

When everything is discrete (DDP, w/ discrete state space) we can write the dual instead

$$\min_{a, V} \sum_{s \in S} V(s) \text{ s.t.}$$

$$V(s_i) \geq Eu(a, s_i, \epsilon_t) + \beta \sum_j \tilde{p}_{ij}(a, s_i, ) V_j \quad \forall s_i$$

This problem is now linear in  $V$  and can be solved with linear programming techniques (may be large since we need to find  $a, V$ ).

# Collocation Method (Judd 1992)

These methods are specifically designed for problems with continuous state space.

- ▶ Use a polynomial representation of the value function  $\tilde{V}(s, c)$  with coefficients  $c$  that approximate the true value function
- ▶ Successively approximate until  $\|c^k - c\| < \epsilon$

$$\tilde{V}(s, c) = \sum_{i=1} c_i \phi_i(x)$$

$$\sum_{j=1} c_j \phi_j(s_i) - \max_z E \left[ u(s_i, z) + \beta \sum_{i=1}^m \tilde{p}_{il}(s) \sum_{j=1}^n c_j \phi_j(s_i) \right] = 0$$

- ▶ Nonlinear system of equations (at each grid point  $s_i$ ).
- ▶ Tricky because it involves a  $\max$
- ▶ Very fast (Christiano and Fisher)

# Discretization

Many problems we are interested in have continuous states or continuous decisions.

- ▶ In the case where we have a continuous state space, we need to discretize it into a grid
- ▶ How do we do that?
- ▶ Dealing with the curse of dimensionality.
- ▶ Do we let future states outside the grid?

# New Approximation Problem

Exact Problem

$$V(s) = \max_{a \in A(s)} \left[ (1 - \beta)u(s, a) + \beta \int V(s')p(ds'|s, a) \right]$$

Approximation to the problem:

$$\hat{V}(s) = \max_{a \in \hat{A}(s)} \left[ (1 - \beta)u(s, a) + \beta \sum_{k=1}^N \hat{V}(s')p_N(s'_k|s, a) \right]$$

# How to Approximate on Grids

- I. Huge literature on numerical analysis on how to efficiently generate grids
- II. Two main issues:
  - A. How to select grid points  $s_k$
  - B. How to approximate transition matrix  $p$  with  $p_N$ .
- III. Answer to second issue follows from answer to first problem
- IV. We can combine strategies to generate grids

# Uniform Grid

- ▶ Decide how many grid points
- ▶ Distribute them uniformly on the state space
- ▶ What if the state space is unbounded?
- ▶ Advantages and disadvantages (bounded errors)



# Non-Uniform Grid

- ▶ Economic theory or error analysis to evaluate where to accumulate points
- ▶ Standard argument: close to curvature of value function: nobody replaces bus at 10 miles.
- ▶ Problem: this is a heuristic argument
- ▶ Self-confirming equilibria in computations (Citation)

# Quadrature Grids

- ▶ Tauchen and Hussey (Econometrica, 1991)
- ▶ Motivation quadrature points in integrals

$$\int f(s)p(s)ds \approx \sum_{k=1}^N f(s_k)w_k$$

- ▶ Gaussian quadrature: choose nodes and weights so that the equation holds exactly for all polynomials of degree less than or equal to  $2N - 1$ .

# Stochastic Grids

- ▶ Randomly chosen grids
- ▶ Rust (1995): Breaks the Curse of Dimensionality – you don't need exponentially more points as the dimension increases for a fixed level of accuracy
- ▶ Downside: accuracy is low to start with
- ▶ How do we generate random numbers?

# Interpolation

- ▶ Discretization means we need to interpolate the function at intermediate values for CDPs
- ▶ Simple: Linear interpolation
- ▶ Problem: In more than one dimension, linear interpolation does not preserve concavity
- ▶ Cubic Splines?
- ▶ Shape-preserving splines: Schumacher (1983).

# Multigrid Algorithms

- ▶ Old tradition in numerical literature
- ▶ Idea: Solve a problem in coarser grid and use it as a guess for a more refined solution (and Iterate)
- ▶ Examples: Differential Equations, Projection Methods, Dynamic Programming (Chow and Tsisiklis, 1991).

# Applying the Algorithm

After deciding initialization and discretization we need to implement each step

$$V^T(s) = \max_{a \in \hat{A}(s)} \left[ u(s, a) + \beta \sum_{k=1}^N V^{T-1}(s'_k) p_N(s'_k | s, a) \right]$$

Two numerical operations

1. Maximization
2. Integration

# Maximization

Maximization is the costly step of the value function iteration  
(Especially for CDPs)

- ▶ Brute force: check all possible choices in the grid
- ▶ Sensibly: using quasi-Newton algorithm

## Brute Force

- ▶ Only alternative: discrete choices, constraints, non-differentiabilities
- ▶ Tricks: Previous solution, Exploit Monotonicity, Concavity of  $V(\cdot), A(\cdot)$ .

## Newton's Method

- ▶ Fast but we need to compute derivatives

# Accelerator

For many algorithms maximization is the most expensive part

- ▶ Often we update  $V(\cdot)$  and choice that are not optimal.
- ▶ Trick: Don't always apply the `max` operator at every iteration.
- ▶ How do we choose the best timing of `max` operator.



# Convergence Assessment

For many algorithms maximization is the most expensive part

- ▶ How do we assess convergence?
- ▶ By the contraction mapping property:

$$\|V - V^k\|_{\infty} \leq \frac{1}{1 - \beta} \|V^{k+1} - V^k\|_{\infty}$$

- ▶ Relates the error in VFI to Euler Equation error
- ▶ Can use this to refine the grid

# Integration Methods

- ▶ Exact
- ▶ Approximate and integrate that: Taylor's Rule, Laplace's Method.
- ▶ Quadrature
- ▶ Monte Carlo simulation

# Approximation Methods (Judd Ch 6)

Objective: Given a complicated  $f(x)$  at some points, can we construct a simpler approximation  $g(x)$ ?

- ▶ What data should be produced and used?
- ▶ What family of “simpler” functions? What kind of approximation
- ▶ Like regression but not “natural” data.

# Interpolation Methods (Judd Ch 6)

Find  $g(x)$  from  $n$ -dimensional family of functions to fit  $n$  data points exactly.

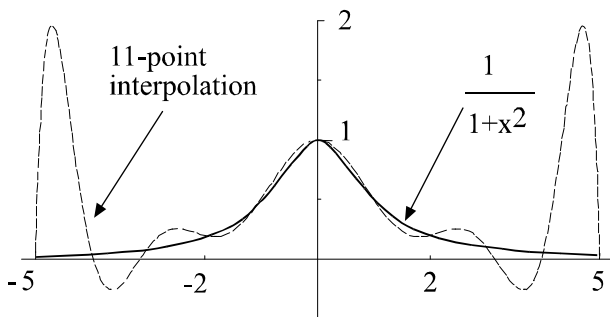
## Lagrange Polynomial Interpolation

- ▶ Data  $(x_i, y_i)$ ,  $i = 1, \dots, n$
- ▶ Objective, find an  $n - 1$  degree polynomial  $p_n(x)$  which agrees with the data  $y_i = f(x_i)$
- ▶ Result: If  $x_i$  are distinct there is a unique interpolating polynomial
- ▶ Does  $p_n(x)$  converge to  $f(x)$  as we use more points? *No!*

# Interpolation Methods (Judd Ch 6)

Figure: (from Judd)

$$f(x) = \frac{1}{1+x^2}$$
$$x_i = -5, -4, \dots, 3, 4, 5$$



# Interpolation Methods (Judd Ch 6)

## Hermite Polynomial Interpolation

- ▶ Data  $(x_i, y_i, y'_i)$ ,  $i = 1, \dots, n$
- ▶ Objective, find an  $2n - 1$  degree polynomial  $p_n(x)$  which agrees with the data  $y_i = p(x_i)$  AND  $y'_i = p'(x_i)$
- ▶ Result: If  $x_i$  are distinct there is a unique interpolating polynomial

## Least Squares Approx

- ▶ Data the function  $f(x)$
- ▶ Objective: find a function  $g(x)$  in class  $G$  that best approximates  $f(x)$  ie:

$$g = \arg \min_{g \in G} \|f - g\|^2$$

# Orthogonal Polynomials

## General Case

- ▶ Space: polynomials over domain  $D$
- ▶ Weighting function:  $w(x) > 0$  (positive everywhere)
- ▶ Inner product  $\langle f, g \rangle = \int_D f(x)g(x)w(x)dx$
- ▶ Polynomials are orthogonal wrt to  $w(x)$  IFF

$$\langle \phi_i, \phi_j \rangle = 0, \quad i \neq j$$

- ▶ Can compute orthogonal polynomials using recurrence formulas

$$\begin{aligned}\phi_0(x) &= 1 \\ \phi_1(x) &= x \\ \phi_{k+1}(x) &= (a_{k+1}x + b_k)\phi_k(x) + c_{k+1}\phi_{k-1}(x)\end{aligned}$$

# Chebyshev Polynomials

- ▶ Can compute orthogonal polynomials using recurrence formulas
- ▶  $[a, b] = [-1, 1]$  and  $w(x) = (1 - x^2)^{-1/2}$
- ▶  $T_n(x) = \cos(n \cos^{-1} x)$
- ▶ Recursive Definition

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$$

## General Intervals

- ▶ Most problems aren't on the  $[-1, 1]$  interval so we need a COV

$$y = -1 + 2\frac{x-a}{b-a}$$

- ▶ Polynomials  $\phi_i^*(x) \equiv \phi_i(-1 + 2\frac{x-a}{b-a})$  are orthogonal over  $x \in [a, b]$  with respect to the weight  $w^*(x) \equiv (-1 + 2\frac{x-a}{b-a})$  iff the  $\phi_i(y)$  are orthogonal over  $y \in [-1, 1]$  wrt  $w(y)$ .



# Chebyshev Approximation Algorithm

1. Compute the  $m \geq n + 1$  Chebyshev nodes on  $[-1, 1]$

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

2. Adjust the nodes to  $[a, b]$  interval

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

3. Evaluate  $f$  at the nodes  $w_k = f(x_k)$  for  $k = 1, \dots, m$
4. Compute the coefficients  $a_i$  to get the approximation  $p(x)$

$$a_i = \frac{\sum_{k=1}^m w_k T_i(z_k)}{\sum_{k=1}^m T_i(z_k)^2}$$
$$p(x) = \sum_{i=0}^n a_i T_i\left(2\frac{x-a}{b-a} - 1\right)$$

# Minimax Approximation

- ▶ Data:  $(x_i, y_i)$ ,  $i = 1, \dots, n$
- ▶ Objective:  $L^\infty$  fit

$$\min_{\beta \in R^m} \max_i \|y_i - f(x_i; \beta)\|$$

- ▶ Difficult to do (minimax problems are non-convex)
- ▶ Chebyshev Approximation satisfies this property, for  $C^2, C^3$  functions but doesn't get  $f'(x)$  right!

## Theorem

*Suppose  $f : [-1, 1] \rightarrow R$  is  $C^k$  for some  $k \geq 1$ , and let  $I_n$  be the degree  $n$  polynomial interpolation of  $f$  based at the zeroes of  $T_{n+1}(x)$  then*

$$\|f - I_n\|_\infty \leq \left( \frac{2}{\pi} \log(n+1) + 1 \right) \frac{(n-k)!}{n!} \left( \frac{\pi}{2} \right)^k \left( \frac{b-a}{2} \right)^k \|f^{(k)}\|_\infty$$

# Splines

Splines are piecewise interpolating functions

## Definition

A function  $s(x)$  on  $[a, b]$  is a spline of order  $n$  IFF

1.  $s$  is  $C^{n-2}$  on  $[a, b]$  and
2. there is a grid of points (nodes)  $a = x_0 < x_1 < \dots < x_m = b$  such that  $s(x)$  is a polynomial of degree  $n - 1$  on each subinterval  $[x_i, x_{i+1}]$ ,  $i = 0, \dots, m - 1$

Second order plane is piecewise linear.

We usually use cubic splines.

# Splines

## Cubic Splines

- ▶ Lagrange data set  $(x_i, y_i)$  for  $i = 0, \dots, n$ .
- ▶ Nodes: the  $x_i$  are the nodes of the spline
- ▶ Functional form  $s(x) = a_i + b_i x + c_i x^2 + d_i x^3$  on  $[x_{i-1}, x_i]$
- ▶ Unknowns  $4n$  unknown coefficients
- ▶  $2n$  interpolation and continuity conditions:

$$y_i = a_i + b_i x_i + c_i x_i^2 + d_i x_i^3 \quad i = 1, \dots, n$$

$$y_i = a_{i+1} + b_{i+1} x_i + c_{i+1} x_i^2 + d_{i+1} x_i^3 \quad i = 0, \dots, n-1$$

- ▶  $2n - 2$  conditions from  $C^2$  at the interior for  $i = 1, \dots, n-1$

$$b_i + 2c_i x_i + 3d_i x_i^2 = b_{i+1} + 2c_{i+1} x_i + 3d_{i+1} x_i^2$$

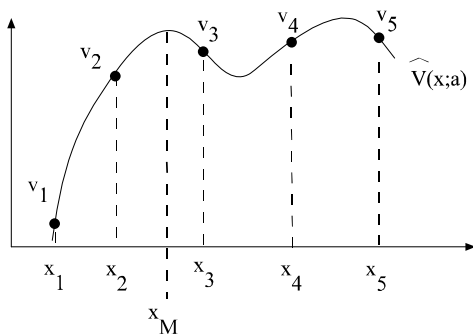
$$2c_i + 6d_i x_i = 2c_{i+1} + 6d_{i+1} x_i$$

# Side Conditions

We have  $4n - 2$  linear equations and  $4n$  unknowns we need two side conditions to identify the system

- ▶ Natural spline:  $s''(x_0) = s''(x_n) = 0$  minimizes the total curvature  $\int_{x_0}^{x_n} s''(x)^2 dx$
- ▶ Hermite spline:  $s'(x_0) = y'_0$  and  $s'(x_n) = y'_n$  (with extra data)
- ▶ Secant Hermite:  $s'(x_0) = \frac{s(x_1) - s(x_0)}{x_1 - x_0}$  ,  $s'(x_n) = \frac{s(x_n) - s(x_{n-1})}{x_n - x_{n-1}}$
- ▶ Solvers are built in to packages like MATLAB (check documentation for which method).

# Shape Issues



- ▶ Concave (monotone) data may lead to non concave (non monotone) approximations
- ▶ Shape problems stabilize VFI.

# Schumaker Procedure (Shape Preserving Splines)

1. Take level (and maybe slope) data at nodes  $x_i$
2. Add intermediate nodes  $z_i^+ \in [x_i, x_{i+1}]$
3. Run quadratic spline with nodes at the  $x$  and  $z$  nodes which interpolate data and preserves shape
4. Schumaker formulas tell you how to choose the  $z$  and spline coefficient
5. Detail in Judd and in companion paper (Judd and Solnick)

# Numerical Integration

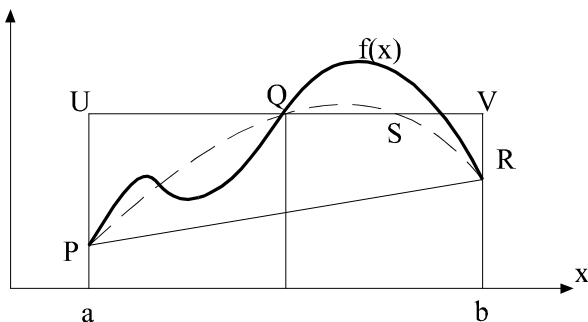
We are interested in lots of problems that require computing difficult integrals (e.g.: evaluating expectations )

1. Midpoint/Trapezoid Rules
2. Simpson's Rule
3. Gaussian Rules
4. Higher-Dimensional Rules



# Quadrature Rules

Basic idea of quadrature is to approximate complicated functions with something easier to integrate, and then integrate that exactly.



- ▶ Constant  $f(x)$  at midpoint of  $[a, b]$   $aUQVb$  for box.
- ▶ Linear: Trapezoid  $aPRb$
- ▶ Parabola through  $f(x)$  at  $a, b$  and  $\frac{a+b}{2}$  for  $aPQRb$

# Simpsons Rule (Newton-Cotes)

Piecewise Quadratic Approximation at some  $\xi \in [a, b]$

$$\int_a^b f(x)dx \approx \left(\frac{b-a}{6}\right) \left[ f(a) + 4f\left(\frac{a+b}{2}\right) + f(b) \right] - \frac{(b-a)^5}{2880} f^{(4)}(\xi)$$

With approximation error

$$\frac{1}{90} \left(\frac{b-a}{2}\right)^5 |f^{(4)}(\xi)|$$

Works well when quadratic approximation is good  $f^{(4)}$  is small or interval is small.

# Gaussian Quadrature

Formulas of the form

$$\int_a^b f(x)dx \approx \sum_{i=1}^n w_i f(x_i)$$

for some quadrature nodes  $x_i \in [a, b]$  and weights  $w_i$ .

- ▶ Let  $\mathcal{F}_k$  be the space of degree  $k$  polynomials
- ▶ Quadrature formulas are exact of degree  $k$  if it correctly integrates each function in  $\mathcal{F}_k$
- ▶ Gaussian quadrature formulas use  $n$  points and are exact of degree  $2n - 1$ .

Approximation Error

$$(f, g) = \int_a^b w(x)f(x)dx - \sum_{i=1}^n w_i f(x_i) = \frac{f^{(2n)}(\xi)}{(2n)!}(p_n, p_n)$$

# Gaussian Quadrature

**Legendre** Domain:  $[-1, 1]$ ,  $w(x) = 1$

**Chebyshev** Domain:  $[-1, 1]$ ,  $w(x) = \frac{1}{\sqrt{1-x^2}}$

**Laguerre** Domain:  $[0, \infty]$ ,  $w(x) = \exp[-x]$  (useful for present value)

**Hermite** Domain:  $[-\infty, \infty]$ ,  $w(x) = \exp[-x^2]$  (useful for normal)

Helpful if function is  $C^\infty$  or analytic.

## Gauss Hermite

Let  $Y \sim N(\mu, \sigma^2)$  and apply COV  $x = (y - \mu)/\sqrt{2}\sigma$

$$\begin{aligned} E[f(Y)] &= (2\pi\sigma^2)^{-\frac{1}{2}} \int_{-\infty}^{\infty} f(y) \exp\left[-\frac{(y - \mu)^2}{2\sigma^2}\right] dy \\ \int_{-\infty}^{\infty} f(y) \exp\left[-\frac{(y - \mu)^2}{2\sigma^2}\right] dy &= \int_{-\infty}^{\infty} f(\sqrt{2}\sigma x + \mu) e^{-x^2} \sqrt{2}\sigma dx \end{aligned}$$

Gives the quadrature formula using Gauss Hermite  $(x_i, w_i)$ .

$$E[f(Y)] = \frac{1}{\sqrt{\pi}} \sum_{i=1}^n w_i f(\sqrt{2}\sigma x_i + \mu)$$

# Higher Dimensional Integration

- ▶ In higher dimension we can use product rules of 1-D integrals.
- ▶ This grows exponentially in dimension  $D$  (Curse of Dimensionality)
- ▶ Monte Carlo is not cursed but slow to converge  $\frac{1}{\sqrt{n}}$  vs  $\frac{1}{2n!} f^{(2n)}$
- ▶ Some monomial rules (Judd), (Skrainka and Judd) aren't cursed
- ▶ Sparse Grids show how to combine 1-D rules more efficiently ([www.sparse-grids.de](http://www.sparse-grids.de))