Introduction to Numerical Integration

Biostatistics 615/815
Lecture 21

Last Series of Lectures

- Numerical Optimization
- Progressively sophisticated techniques
  - Optimization in a single dimension
  - Optimization along multiple dimensions
  - Stochastic optimization strategies
- Last Lecture: Gibbs Sampler

Today: Numerical Integration

- Strategies for numerical integration
- Simple strategies with equally spaced abscissas
- Gaussian quadrature methods
- Introduction to Monte-Carlo Integration

The Problem

- Evaluate:

$$I = \int_a^b f(x)dx$$

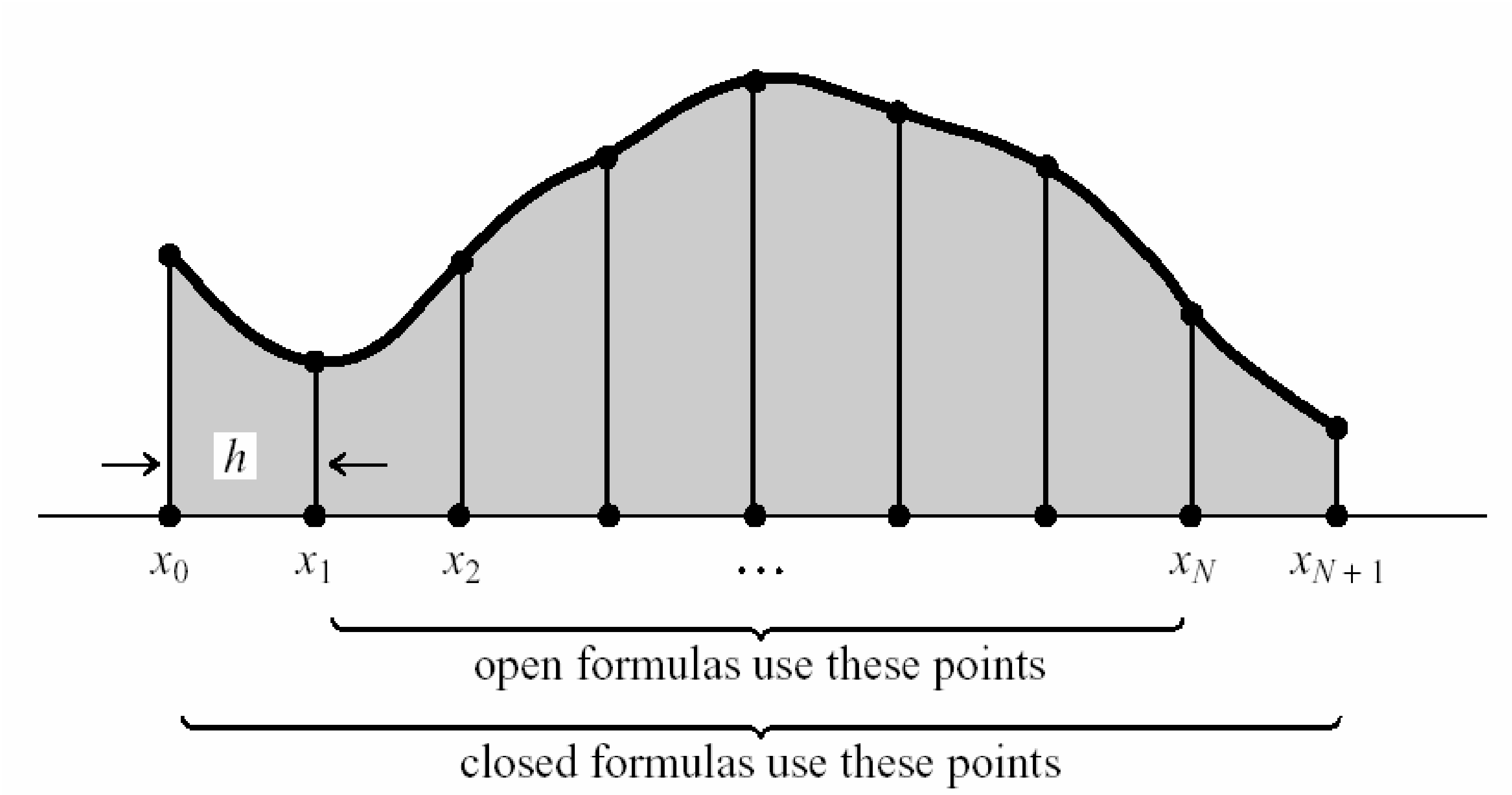
- When no analytical solution is readily available

- Many applications in statistics
  - Analysis of censored data,
  - Evaluation of cumulative distributions, etc.

The Challenge

- Evaluate $f(x)$ as few times as possible
- Select appropriate set of abscissas
- Select appropriate set of weights

The Basic Approach



Notation

- Consider a series of abscissas
  - $x_0, x_1, x_2, \ldots, x_n, x_{n+1}$
- Let these be a constant step size *h* apart
  - $x_i = x_0 + i\,h$
- Further define:
  - $f_i = f(x_i)$

Two Point Trapezoidal Rule

$$\int_{x_1}^{x_2} f(x)dx \approx h\left[\frac{1}{2} f_1 + \frac{1}{2} f_2\right]$$

- Exact for polynomials up to degree 1
  - For example, $f(x) = 2x + 1$
- Error proportional to h^3 and $f^{(2)}$

Three Point Simpson's Rule

$$\int_{x_1}^{x_3} f(x)dx \approx h\left[\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{1}{3}f_3\right]$$

- Exact for polynomials up to degree 3 (not 2!)
  - Due to some symmetries in derivation
- Error proportional to h^5 and $f^{(4)}$

Four Point Rule

$$\int_{x_1}^{x_4} f(x)dx \approx h\left[\frac{3}{8}f_1 + \frac{9}{8}f_2 + \frac{9}{8}f_3 + \frac{3}{8}f_4\right]$$

- Exact for polynomials up to degree 3
  - No lucky symmetries this time…
- Error proportional to h^5 and $f^{(4)}$
- Additional formulas exist for higher orders …

Extended Rules

- Combine simple rules along consecutive intervals
- Two and three point rules allow for adaptive integration
  - Gradually add points and check accuracy…

Extended Trapezoidal Rule

$$\int_{x_1}^{x_n} f(x)dx = h\left[\frac{1}{2}f_1 + f_2 + f_3 + \ldots + \frac{1}{2}f_n\right]$$

- Results from application of trapezoidal rule to consecutive intervals …

Simple C Implementation

```c
// Integrates function f(x) between a and b
// by evaluating it at the edges of the interval
// and at n interior points
double integrate2(double a, double b, double (*f)(double x), int n)
   {
   double h = (b – a) / (n + 1), sum;
   int i;

   sum = 0.5 * ((*f)(a) + (*f)(b));

   for (int i = 1; i <= n; i++)
        sum += (*f)(a + i * h);

   return sum * h;
   }
```

Extended Simpson Rule

$$\int_{x_1}^{x_n} f(x)dx = h\left[\frac{1}{3}f_1 + \frac{4}{3}f_2 + \frac{2}{3}f_3 + \frac{4}{3}f_4 \ldots + \frac{1}{3}f_n\right]$$

- Results from application of Simpson's rule to consecutive intervals …
  - Note alternating 2/3 and 4/3 weights …

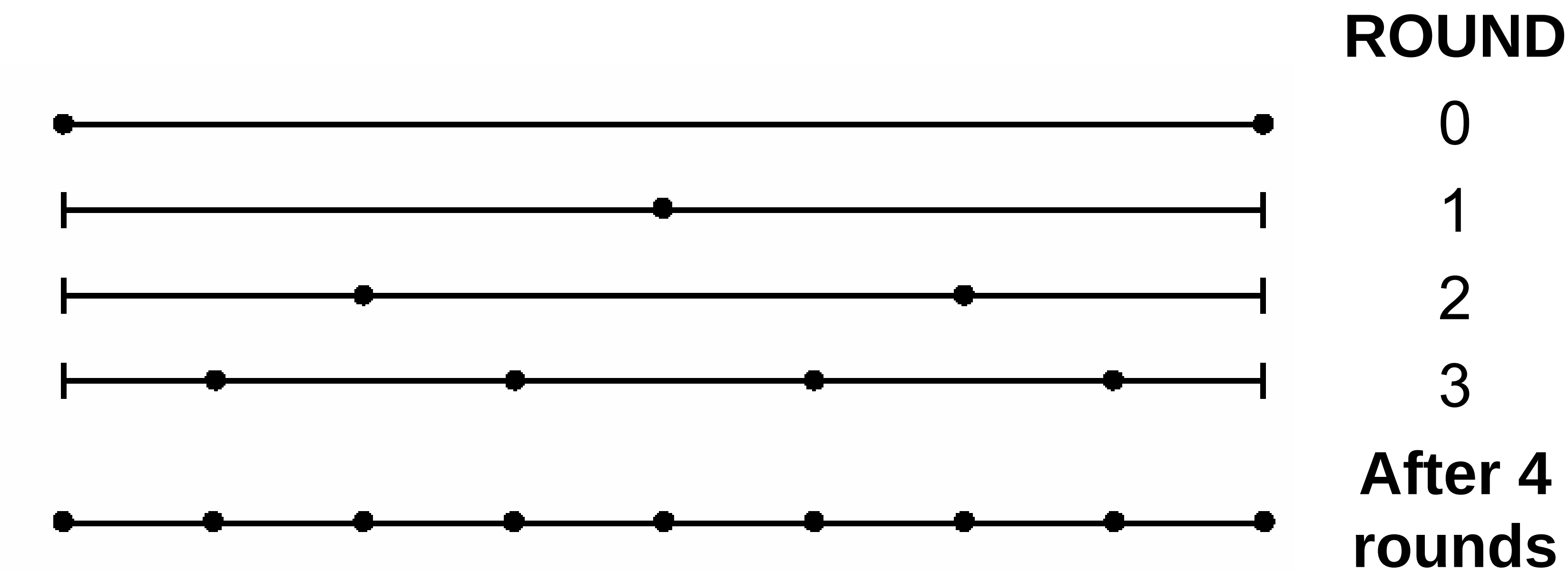
Simple C Implementation

```
double integrate3(double a, double b, double (*f)(double x), int n)
   {
   double h, sum; int i;

   if (n % 2 == 0) n++;        // n must be odd
   h = (b – a) / (n + 1);

   sum = (*f)(a) + (*f)(b) + 4.0 * (*f)(a + h);

   for (int i = 2; i <= n; i += 2)
        sum += 2.0 * (*f)(a + i*h) + 4.0 * (*f)(a + (i + 1)*h);

   return sum * h / 3.0;
   }
```

Problem ...

- Knowing the required number of points before hand may not be practical…
- Is there a simple way to "add more points" ?

Gradually Adding Points



Can you derive formula for updating integral if points are added in this manner?
How would you check if a desired accuracy has been reached?

Simple C Implementation

```
double update_integral(double a, double b,
                       double (*f)(double x),
                       double previous, int round)
   {
   double h, sum;
   int i, n = 1 << (round – 1);

   if (round == 0)
        return 0.5 * ((*f)(a) + (*f)(b)) * (b – a);

   sum = previous * n / (b - a);
   h = (b - a) / (2 * n);
   for (int i = 1; i < 2 * n; i += 2)
        sum += (*f)(a + i*h);

   return sum * h;
   }
```

Simple C Implementation

```
#define ZEPS      1e-10

double integral(double a, double b, double (*f)(double x),
                double eps)
  {
  double old = update_integral(a, b, f, 0.0, 0), result;
  int round = 1;

  while (1)
     {
     result = update_integral(a, b, f, old, round++);
     if ( fabs(result-old) < eps*(fabs(result)+fabs(old))+ZEPS)
        return result;
     old = result;
     }
  }
```

Simpson's Extended Rule ...

- Define T_N and T_{2N} to be trapezoidal rule results with N and $2N$ points, respectively

- Then the application of Simpson's rule gives:

$$S = \frac{4}{3}T_{2N} - \frac{1}{3}T_N$$

Simple C Implementation

```
double simpson(double a, double b, double (*f)(double x),
               double eps)
   {
   double old = update_integral(a, b, f, 0.0, 0), result;
   double sold = old, sresult;
   int round = 1;

   while (1)
      {
      result = update_integral(a, b, f, old, round++);
      sresult = (4.0 * result - old) / 3.0;
      if (fabs(sresult-sold)<eps*(fabs(sresult)+fabs(sold))+ZEPS)
         return sresult;
      old = result; sold = sresult;
      }
   }
```

Simple Application

- Integrate standard normal density
  - Between 0.0 and 1.0
  - Correct result is 0.341345
- With $\varepsilon=10^{-5}$, I got the following results:
  - Trapezoidal rule, 7 rounds, 129 evaluations, 0.341344
  - Simpson's rule, 4 rounds, 17 evaluations, 0.341355
- In this case, higher order approximation was more efficient

Notes on Classical Methods

- These methods are most intuitive
- Two major applications:
  - Functions that are not smooth
  - Function can be pre-calculated along a grid
- Exact solutions for polynomials of degree n typically require n or $n\text{-}1$ evaluations

Classical Methods

- Function evaluated at equally spaced points
- Choice of weights for combining results determines order of approximation

Quadrature Methods

- Select locations of function evaluations and weights simultaneously
  - Abscissas correspond to zeros of particular classes of orthogonal polynomials
- Achieves higher order approximations faster

Gaussian Quadrature

$$\int_a^b f(x)dx \approx \sum_{j=1}^{N} w_j f(x_j)$$

- The original idea is due to Gauss (1814)
  - Described a strategy for choosing appropriate weights and abscissas
- Weights and abscissas can be chosen to provide exact results for polynomials of degree $2N - 1$ or integrable functions of the form $W(x) * polynomial(2N - 1)$

Intuition Behind Idea

- Evaluating function at any two points, we can derive exact solution for polynomials of degree 1.
  - E.g. The trapezoidal rule does this.
- But a single well chosen point can achieve the same result…
  - Which point?

Some Example Abscissas

N	$Abscissas$	$Weights$	$MaxDegree$
2	$\begin{pmatrix} -\sqrt{1/3} \\ +\sqrt{1/3} \end{pmatrix}$	$\begin{pmatrix} 1.0 \\ 1.0 \end{pmatrix}$	3
3	$\begin{pmatrix} -0.77459667 \\ 0.0 \\ +0.77459667 \end{pmatrix}$	$\begin{pmatrix} 0.5555555 \\ 0.8888889 \\ 0.5555555 \end{pmatrix}$	5
4	$\begin{pmatrix} -0.86113631 \\ -0.33998104 \\ +0.33998104 \\ +0.86113631 \end{pmatrix}$	$\begin{pmatrix} 0.34785485 \\ 0.65214515 \\ 0.65214515 \\ 0.34785485 \end{pmatrix}$	7

C Code

```
double gauss3(double a, double b, double (*f)(double x))
   {
   double abscissas[] = {-0.77459667, 0.0, 0.77459667 };
   double weights[] = {0.55555555, 0.88888889, 0.55555555};
   double midpoint = 0.5 * (a + b);
   double h = 0.5 * (b - a);
   double sum = 0.0;

   for (int i = 0; i < 3; i++)
      sum += weights[i] * (*f)(midpoint + abscissas[i] * h);

   return sum * h;
   }
```

Comparison

- Integrate standard normal density
  - Between 0.0 and 1.0
  - Correct result is 0.341345

- With 2, 3 and 4 function evaluations I got:
  - Using trapezoidal rule
    - 0.320457, 0.336261, 0.339096
  - Using quadrature
    - 0.341221, 0.341346, 0.341345
  - Using Simpson's rule (3 evaluations)
    - **********, 0.341529, ***********

Multi-Dimensional Integrals

$$\int_{x=a}^{x=b}\int_{y=c}^{y=d} f(x,y)\,dx\,dy = \int_{x=a}^{x=b} g(x)\,dx$$

$$g(x) = \int_{c}^{d} f(x,y)\,dy$$

- Simplest strategy is to evaluate as a series of one dimensional integrals
  - Exponential increase in function evaluations

Monte-Carlo Methods

- Evaluate and average function at random points
- Adaptive methods focus on areas where integrand is most significant
  - Crucial for multiple dimensions

Monte-Carlo Importance Sampling

- Assume N evaluations are available
- Evaluate function at kN random points
- Divide region of integration into high and low variance regions
  - Allocate remaining $(1-k)N$ points so that most are used in high variance region

Today:

- Numerical integration
- Classical strategies, with equally spaced abscissas
- Discussion of quadrature methods and Monte-Carlo methods

Recommended Reading

- Numerical Recipes
  - Chapters 4.0 – 4.2 for Classical Methods
  - Chapter 4.5 for Gaussian Quadrature
  - Chapter 7.8 for Monte-Carlo methods
- Available online at:
  - http://www.nr.com

Happy Thanksgiving!