# Numerical Integration and Monte Carlo Integration

- Elementary schemes for 1D integrals with no singularities
- More sophisticated methods exist (use "canned" routines)
- Multi-dimensional integrals "dimension-by-dimension"
- Monte Carlo (stochastic) integration for high-D integrals

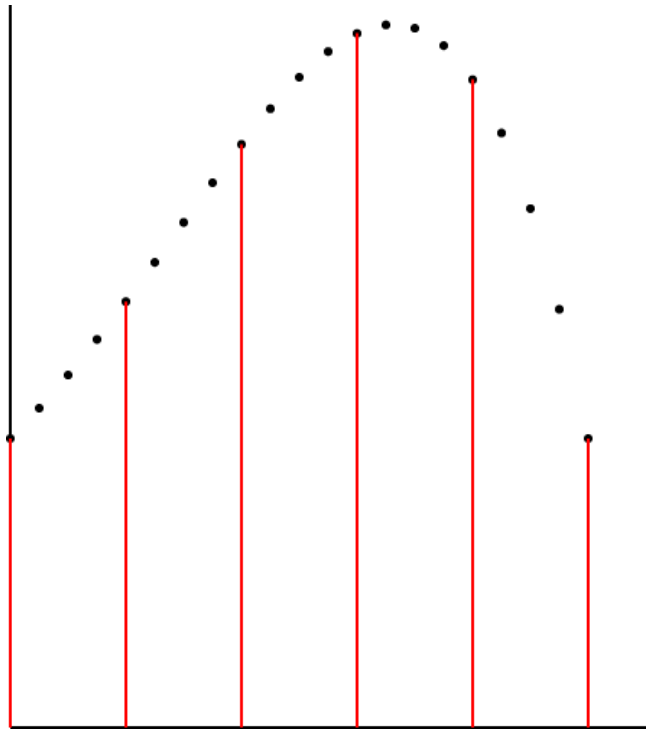
Useful resource for numerical-analysis tools:

- http://gams.nist.gov (GAMS subroutine repository)
- Numerical Recipes. Free material versions available on-line; www.nr.com

# Numerical integration; 1D methods

- Assuming that the integrand has no singularities

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



- Discretize:

$$[a, b] \longrightarrow x_0, x_1, \ldots, x_n$$

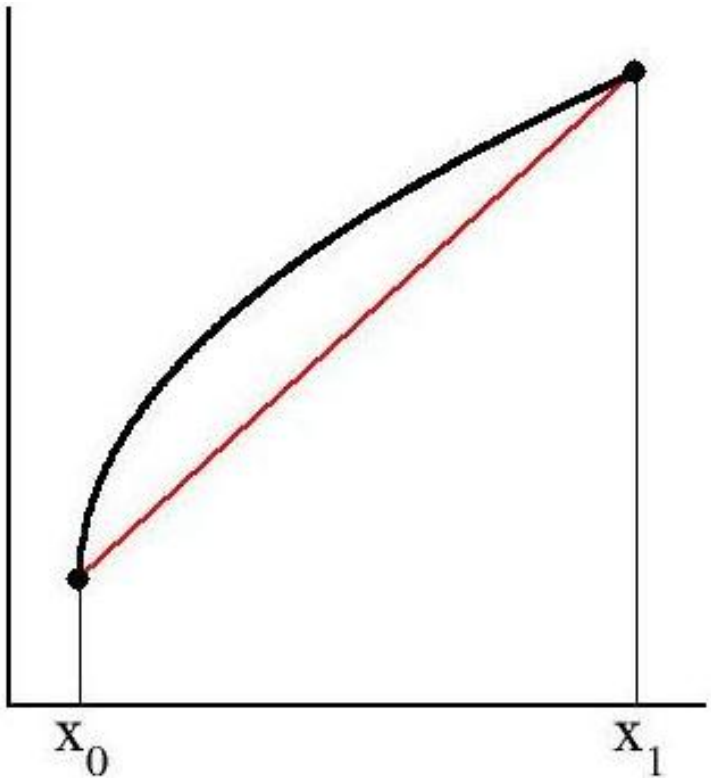
$$h = x_{i+1} - x_i$$

- Fit N-th order polynomial to N+1 points; integrate pieces
- Error typically $O(h^{N+1})$

# N=1

$$f(x_0+\delta) = a + b\delta, \qquad 0 \le \delta \le h$$

$$f(x_0) = f_0, \quad f(x_0+h) = f_1$$




Solving this gives:

$$a = f_0, \qquad b = (f_1 - f_0)/h$$

Integrating;

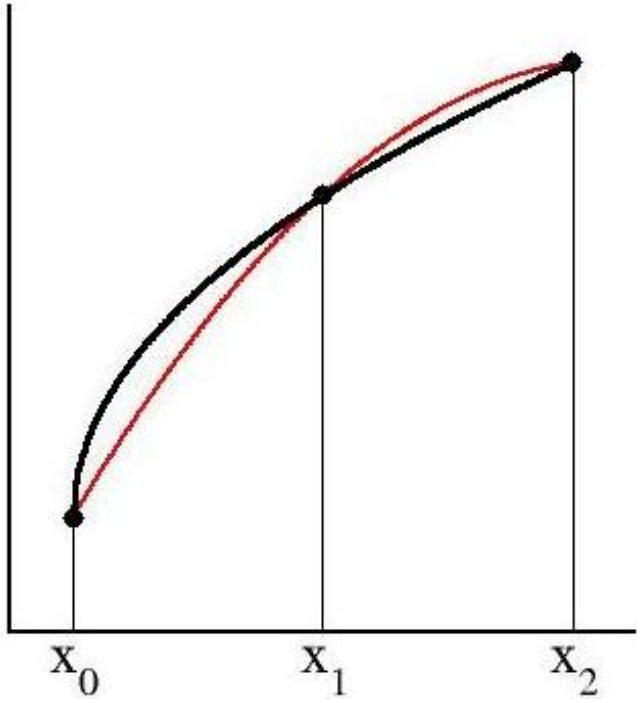
$$\int_{x_0}^{x_1} f(x)dx = h(a + bh/2) = \frac{h}{2}[f_0 + f_1] + O(h^3)$$

This is called the trapezoidal rule

# N=2

$$f(x_0+\delta) = a + b\delta + c\delta^2$$

$$0 \le \delta \le 2h$$




$$f(x_0) = f_0, \quad f(x_0+h) = f_1, \quad f(x_0+2h) = f_2$$

$$f_0 = a \quad f_1 = a + bh + ch^2 \quad f_2 = a + 2bh + 4ch^2$$

Solving for a,b,c and integrating --> Simpson's rule

$$\int_{x_0}^{x_1} f(x)dx = \frac{h}{3}[f_0 + 4f_1 + f_2] + O(h^5),$$

# Extended trapezoidal and Simpson's rules

$$\int_{x_0}^{x_N} f(x)dx = h(\frac{1}{2}f_0 + f_1 + \ldots + f_{N-1} + \frac{1}{2}f_N) + O(h^2),$$

$$\int_{x_0}^{x_N} f(x)dx = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + \ldots + 2f_{N-2} + 4f_{N-1} + f_N) + O(h^4).$$

Extended open formulas (boundaries excluded):

$$\int_{x_0}^{x_N} f(x)dx = h(f_{1/2} + f_{3/2}+, \ldots + f_{N-3/2} + f_{N-1/2}) + O(h^2)$$

$$\int_{x_0}^{x_N} f(x)dx = h(\frac{3}{2}f_1 + f_2 + f_3 + \ldots + f_{N-2} + \frac{3}{2}f_{N-1}) + O(h^2)$$

$$\int_{x_0}^{x_N} dx = h(\frac{23}{12}f_1 + \frac{7}{12}f_2 + f_3 + f_4 + \ldots + f_{N-3} + \frac{7}{12}f_{N-2} + \frac{23}{12}f_{N-1}) + O(h^3)$$

# Integrable singularities

- The open formulas discussed can be used
- Error scaling with h will be worse
- Some singularities can be transformed away
- Asymptotic divergence can some times be subtracted (analytically solvable, leaving non-singular integral)
- More sophisticated methods exist for difficult cases

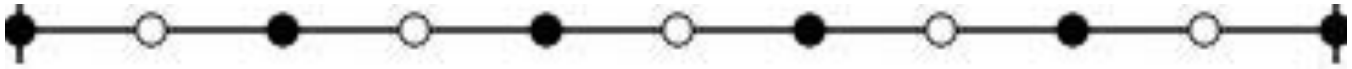
Simpson's rule is adequate for many "proper" integrals

If faster convergence needed -- Romberg integration

# Romberg integration

Idea: Use trapezoidal rule with different h

- Error scales as $h^2$ (+ higher-order terms)
- **Only even powers**, i.e., polynomial in $h^2$
- Extrapolate to h=0 using interpolating polynomial



Simplest case; n=1, use $h=h_0$ and $h=h_0/2$

$$I_0 = I_\infty + \epsilon h_0^2, \quad I_1 = I_\infty + (\epsilon/4) h_0^2$$

Extrapolate to zero leading error (solving for $I_\infty$):

$$I_\infty = \frac{4}{3} I_1 - \frac{1}{3} I_0$$

Equivalent to Simpson's rule!

Trick: Doubling number of points ($h_{k+1} = h_k/2$)

$$h_k = h_0/2^k, \quad k = 0, \ldots, n$$

- old points can be used; factor-2 time-savings

$$I_{k+1} = I_k/2 + h(f_1 + f_3 + \ldots + f_{N-3} + f_{N-1})$$

# High-order Romberg integration

Need polynomial interpolating between n approximants;

$$I_0, I_1, I_2, \ldots, I_n$$

Lagrange's formula for n points ($x_i$,$y_i$):

$$P(x) = \sum_{i=0}^{n} y_i \prod_{k \neq i} \frac{x - x_k}{x_i - x_k}$$

Error is polynomial in $x=h^2$, $h=h_0/2^k$; evaluate polynomial at x=0

$$I_\infty = \sum_{i=0}^{n} I_i \prod_{k \neq i} \frac{-h_0^2 2^{-2k}}{h_0^2(2^{-2i} - 2^{-2k})} = (-1)^n \sum_{i=0}^{n} I_i \prod_{k \neq i} \frac{1}{2^{2(k-i)} - 1}.$$

Remaining error is $O(h^{2(n+1)})$ Very rapid convergence!

# Program implementation of Romberg's method

Trapezoidal rule with point-doubling

- integrates from x0 to xn; old and new result in `trap`
- start with n+1 points (n segments)

```
subroutine trapezoidal(x0,xn,n,trap,step)

integer :: i,n,step
real(8) :: h,h2,x0,xn,x0h,trap,func

h=(xn-x0)/dble(n*2**step)
if (step == 0) then
  trap=0.5d0*(func(x0)+func(xn))
  do i=1,n-1
    trap=trap+func(x0+h*dble(i))
  enddo
else
  h2=2.d0*h
  x0h=x0+h
  trap=trap/h2              ! trap contains old (n-1) result
  do i=0,n*2**(step-1)-1
     trap=trap+func(x0h+h2*dble(i))
  enddo
endif
trap=trap*h
```

## Extrapolating trapezoidal approximants 0,..,n

```
function polext(n,y)

integer :: j,k,n
real(8) :: y(0:n),p,polext

polext=0.d0
do k=0,n
   p=1.d0
   do j=0,n
      if (j /= k) p=p/(2.d0**(2*(j-k))-1.d0)
   enddo
   polext=polext+p*y(k)*(-1)**n
enddo
```

## Loop in main program

```
do i=0,steps-1
   call trapezoidal(x0,xn,n,t,i)
   trap(i)=t
   if (i /= 0) print*,trap(i),polext(i,trap)
enddo
```

# Multi-dimensional Integration

$$I = \int_{a_n}^{b_n} dx_n \cdots \int_{a_2}^{b_2} dx_2 \int_{a_n}^{b_n} dx_1 f(x_1, x_2, \ldots, x_n),$$

Can be carried out "dimension-by-dimension" using 1D method

2D example with non-rectangular boundaries:

$$I = \int_{a_y}^{b_y} dy \int_{a_x(y)}^{b_x(y)} dx f(x, y)$$

Can be written as 1D y-integral of function that is an x-integral

$$I = \int_{a_y}^{b_y} dy F(y), \qquad F(y) = \int_{a_x(y)}^{b_x(y)} dx f(x, y)$$

Very time-consuming for D > 3. Scaling: $M_1^D$ M1=time for 1D