Chapter 9: Numerical Differentiation

Numerical Differentiation

Formulation of equations for physical problems often involve derivatives (rate-of-change quantities, such as velocity and acceleration). Numerical solution of such problems involves numerical evaluation of the derivatives.

One method for numerically evaluating derivatives is to use Finite DIfferences:

From the definition of a first derivative

$$\frac{dy}{dx} = y'(x) \equiv \lim_{\Delta x \to 0} \frac{(yx + \Delta x) - y(x)}{\Delta x}$$

we can take a finite approximation as

$$y'(x) \approx \frac{y(x + \Delta x) - y(x)}{\Delta x}$$

which is called Forward DIfference Approximation.

Similarly, we could use the
Backward Difference Approximation:

$$y'(x) \approx \frac{y(x) - y(x - \Delta x)}{\Delta x}$$

And, in either case, we can reduce a roundoff errors in the computations by choosing Δx to be a small value that can be exactly represented in binary (for example not 0.1).

Also, in addition to an *x* difference that can be exactly represented in binary, we can get a somewhat better approximation to the first derivative with a Central Finite Difference:

$$y'(x) \approx \frac{y(x + \Delta x) - y(x - \Delta x)}{2\Delta x}$$

Example:

Given an expression for the function *f(x, y)* in the equation:

$$\frac{dy}{dx} = f(x, y)$$

we can numerically approximate *y* values over the range of *x*, with the difference equation:

$$y(x + \Delta x) - y(x) = \Delta x f(x, y)$$

Assuming we know the initial value y_0 and we subdivide the *x* range from x_0 to xn into equal intervals Δx, we can solve for each successive *y* value as

$$y_k = y_{k-1} + \Delta x f(x_{k-1}, y_{k-1}), \qquad k = 1, 2, \ldots, n$$

In general, such problems can be solved with nonuniform *x* subintervals and with known initial or boundary conditons.

Using similar methods, we can evaluate equations involving partial derivatives, and we can set up finite difference approximations for higher order derivatives.

For instance, for second derivatives, we can use the approximation

$$y'(x) \approx \frac{y(x + \Delta x) - 2y(x - \Delta x)}{\Delta x^2}$$

Derivative Functions in Mathematica

Numerical evaluation of different equations accomplished with

NDSolve[eqns, y, {x, ymin, xmax]} - numerically solves eqns (which include initial or boundary equations) for y with independent variable x in the range *xmin* to *xmax.*

NDSolve[eqns, {y1, y2, . . .}, {x, xmin, xmax}] - solves the set of *eqn*s for values y1, y2, etc.

Example:

```
NDSolve[{y'[x]==Exp[x],y[0]==1},y,{x,0,5}]
{{y -> InterpolatingFunction[{0., 5.}, <>]}}
```

Solution then stored as an interpolation table, and we retrieve values with statements such as

```
y[1.0' /. %

{2.71829}
```

Also, we can use statements such as

```
Plot[Evaluate[y[x] /. %n], {x,xmin, xmax}]
```

to plot the numeric solution.

Symbolic difference performed with

> D[f(x), x] - evaluates first derivative of *f(x)* with respect to *x*. (Can also be applied to f(x1, x2, . . .).)
>
> D[f(x), {x, n}] - evaluates nth derivative of *f(x)* with respect to *x*.

(Also can use D function to obtain partial derivatives.)

Examples:

```
D[Exp[x],x]
```
E^x

```
D[x^3, x]
```
$3\ x^2$

```
D[x^3, {x, 2}]
```
{second derivative

6^x

```
D[x^2 y, x]
```
{partial derivative

```
2 x y
```

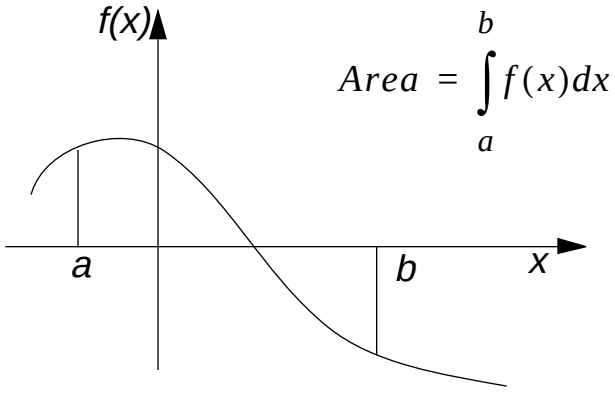
Chapter 9: Numerical Integration

Numerical Integration

Most integrals arising from solutions of problems in engineering and science cannot be represented in "closed form"

- they must be evaluated numerically.

- For function of a single variable, we seek an approximation to the area "under" the curve:



- For function of several variables, we seek approximation to the size of the "volume" for the specified integration region.

We will consider only algorithms for functions of a single variable: *f(x)*.

Numerical evaluation procedures for a function of a single variable *f(x)* involve summation of small subareas, using simple functions to approximate *f(x)*.

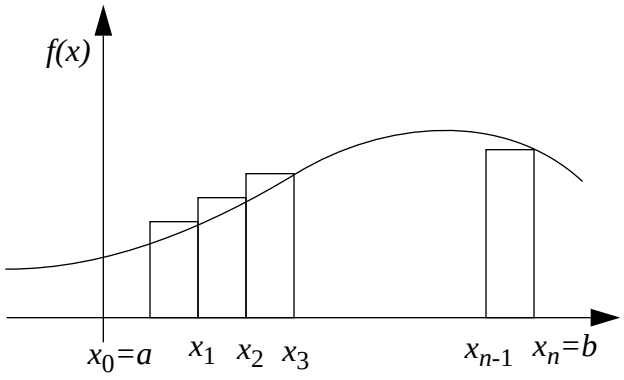
For example, by definition of integral:

$$\int_a^b f(x)dx = \lim_{a \to \infty} \sum_{k=1}^{n} f(x_k)\Delta_k$$

$$\text{with } \Delta x_k = x_k - x_{k-1}$$

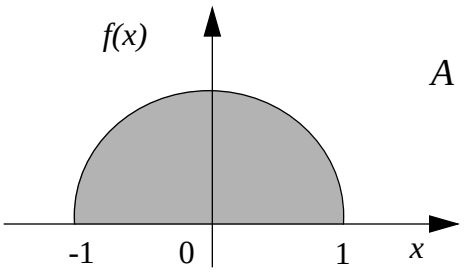
We can approximate integral with a finite number of subintervals using a constant value for *f(x)* in each subinterval:

$$f(x) = f(x_1) \qquad x_{k-1} \le x \le x_k, \qquad k = 1, 2, \ldots, n$$



In this case, we approximate integral by adding the areas of a set of approximating rectangles.

Example: $f(x) = \sqrt{1 - x^2}, \qquad a = -1, b = 1$



$$A = \int_{-1}^{1} g(x)dx$$

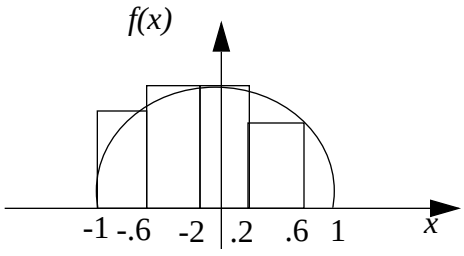
Selecting equal width intervals with with $n = 5$, we have

$$\Delta x_k = \frac{b-a}{n} = \frac{2}{5} = 0.4 \qquad \forall k$$

The area is then approximated as

$$A \approx \sum_{k=1}^{n} f(x_k)\Delta x_k$$

with $x_k = x_{k-1} + \Delta x_k$ and $x_0 = -1$.



Then

$$A \approx \Delta x \sum_{k=1}^{n} f(x_k) \approx 0.4(0.8 + 0.98 + 0.98 + 0.8 + 0.0)$$

$$\approx 1.42$$

Actual size of area: $A \approx \frac{\pi r^2}{2} \approx 1.57$

We can improve numerical estimate of integral by

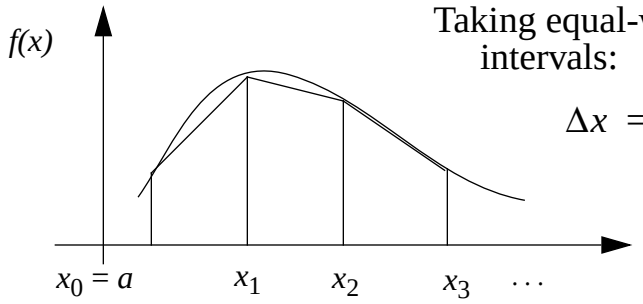
- Increasing number of intervals.
- Using a more accurate approximation of f(x) in each interval.

Common Approximation Methods:

- Linear
- Quadratic or Higher-Order Polynomial

Trapezoidal Rule

Approximate *f(x)* with linear function over each subinterval.
Area is then calculated as the sum of the areas of resulting trapezoids:



Taking equal-width intervals:

$$\Delta x = \frac{b-a}{n}$$

Area of each trapezoid is

$$A_k = \Delta x\left[\frac{f(x_{k-1}) + f(x_k)}{2}\right], \qquad k = 1, 2, \ldots, n$$

Thus,

$$\int_a^b f(x)dx \approx \frac{\Delta x}{2}[f(x_0) + f(x_1)] + [f(x_1) + f(x_2)] + \ldots + [f(x_{n-1}) + f(x_n)]\}$$

or

$$\int_a^b f(x)dx \approx \Delta x\left\{\frac{1}{2}[f(a) + f(b)] + \sum_{k=1}^{n-1} f(x_k)\right\}$$

Example: Trapezoidal Rule to evaluate

$$\int_{-1}^{1} \sqrt{1 - x^2}\, dx, \qquad n = 5$$

Width of each interval is

$$\Delta x = \frac{1 - (-1)}{5} = 0.5$$

$$x_0 = -1, \qquad x_k = x_{k-1} + \Delta x, \qquad k = 1, 2, 3, 4, 5$$

Integral is then evaluated as

$$\int_{-1}^{1} f(x)dx \approx \Delta x \left[\frac{f(-1) + f(1)}{2} + \sum_{k=1}^{4} f(x_k) \right]$$

$$\approx 0.4(0.80 + 0.98 + 0.98 + 0.80)$$

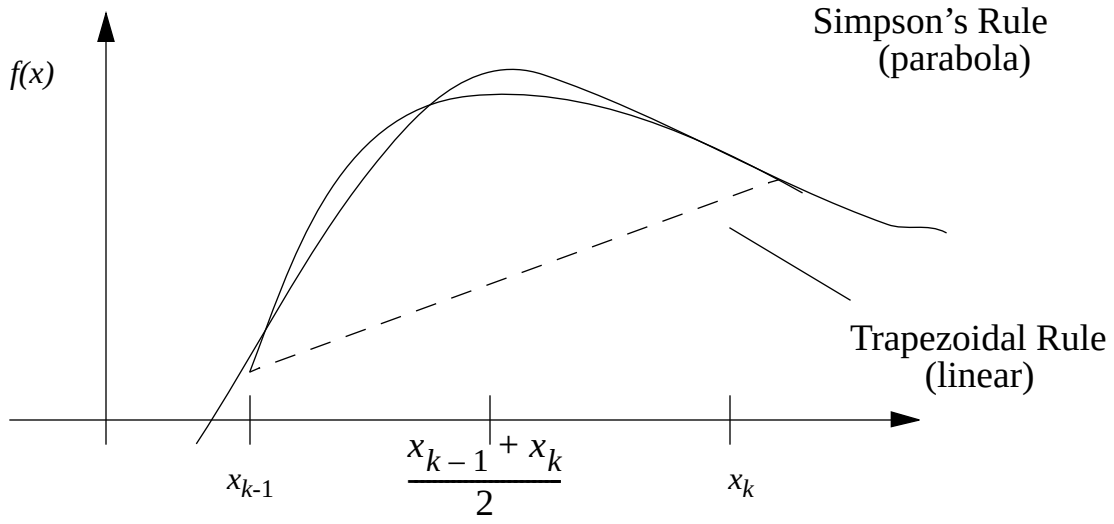
$$\approx 1.42$$

(Same result as obtained using rectangular areas.
Why?)

Simpson's Rule

Approximates *f(x)* with a quadratic function.

More accurate evaluation of integral than Trapezoidal Rule (a linear approximation).



In this case, the integral is approximated as

$$\int_a^b f(x)dx \approx \sum_{k=1}^{n} \int_{x_{k-1}}^{x_k} (\alpha_0 + \alpha_2 x + \alpha_2 x^2)dx$$

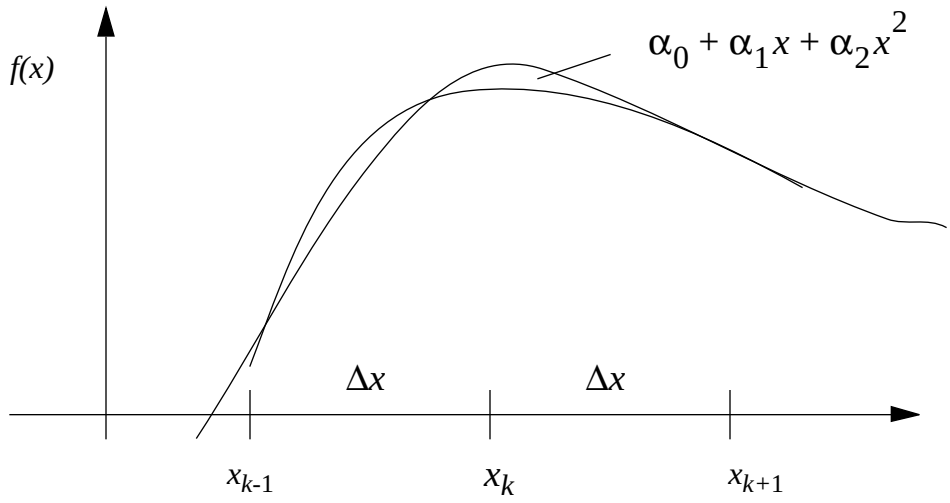
To evaluate the summation, we need to determine the parabola coefficients $\alpha_0, \alpha_1, \alpha_2$ in each of the *n* subdivisions.

We can accomplish this with three boundary conditions that require the parabola to intersect *f(x)* at the endpoints and at the midpoint of each subdidivision.

Applying Simpson's Rule:

1. Divide interval (*a, b*) into n equal subintervals, with *n* a multiple of 2.

2. Approximate *f(x)* with parabola across every two subintervals.



Obtain coefficients $\alpha_0, \alpha_1, \alpha_2$ by solving

$$\alpha_0 + \alpha_1 x_{k-1} + \alpha_2 x_{k-1}^2 = f(x_{k-1})$$
$$\alpha_0 + \alpha_1 x_k + \alpha_2 x_k^2 = f(x_k)$$
$$\alpha_0 + \alpha_1 x_{k+1} + \alpha_2 x_{k+1}^2 = f(x_{k+1})$$

over each of the intervals (x_{k-1}, x_{k+1}).

We then approximate integral as

$$\int_a^n f(x)dx \approx \sum_{\substack{k=1 \\ k\,odd}}^{n-1} \int_{x_{k-1}}^{x_{k+1}} (\alpha_0 + \alpha_1 x + \alpha_2 x^2)dx$$

$$\approx \sum_{\substack{k=1 \\ k\,odd}}^{n-1} \left[\alpha_0 x + a_1 \frac{x^2}{2} + \alpha_2 \frac{x^3}{3}\right]_{x_{k-1}}^{x_{k+1}}$$

Substituting the solution for α_0, α_1, α_2 over intervals (x_{k-1}, x_{k+1}), we obtain

$$\int_s^b f(x)dx \approx \sum_{\substack{k=1 \\ k\,odd}}^{n-1} \left\{\frac{1}{3}\Delta x[f(x_{k-1}) + 4f(x_k) + f(x_{k+1})]\right\}$$

where $\Delta x = x_{k+1} - x_k = x_k - x_{k-1}$

{Note: An easy way to derive this result is to consider a parabols specified over the interval $(-\Delta_x, \Delta_x)$.

Hence,

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}[(f_0 + 4f_1 + f_2) + (f_2 + 4f_3 + f_4) + \ldots]$$

Collecting terms, we have Simpson's Rule:

$$\int_a^b f(x)dx \approx \frac{\Delta x}{3}\left[f(a) + f(b) + 4\sum_{\substack{k=1 \\ (odd)}}^{n-1} f(x) + 2\sum_{\substack{k=2 \\ (even)}}^{n-2} f(x_k)\right]$$

with each successive x value calculated as

$$x_k = x_{k-1} + \Delta x \qquad (k = 1, 2, \ldots, n)$$

Example: Simpson's Rule Evaluation of

$$\int_{-1}^{1} \sqrt{1-x^2}\,dx, \qquad n = 4$$

Then,

$$\Delta x = \frac{1-(-1)}{4} = 0.5$$

$$x_0 = -1, \qquad x_k = x_{k=1} + \Delta x, \qquad k = 1, 2, 3, 4$$

And

$$\int_{-1}^{1} f(x)dx \approx \frac{\Delta x}{3}\left[f(-1) + f(1) + 4\sum_{\substack{k=1 \\ (odd)}}^{3} f(x_k) + 2\sum_{\substack{k=2 \\ (even)}}^{2} f(x_k) \right]$$

$$\approx \frac{\Delta x}{3}\{4[f(x_1) + f(x_3)] + 2f(x_2)\}$$

$$\approx \frac{1}{6}[4(0.87 + 0.87) + 2(1)]$$

$$\approx \frac{1}{6}(8.96)$$

$$\approx 1.49 \qquad where \int_{-1}^{1} f(x)dx \equiv 1.57$$

Thus Simplson with n=4 is bettern than Trapezoidal with n=5.

Psuedocode Algorithm - Simpson's Rule

```
Input f, a, b, n (* fcn, integer limits, no. of intervals *)
dx = (b-a)/n
dx2 = dx + dx
odds = 0.0; evens = 0.0
x = a
Do k=2, n-2, 2
        x = x + dx2
        evens = evens + f(x)
Endofdo
x = a - dx
Do k = 1, n-1, 2
        x = x + dx2
        odds = odds + f(x)
Endofdo
simpsonArea=(dx/3.0)*(f(a)+f(b)+4.0*odds+2.0*evens)
```

Summary of Integration Methods

- Approximate $f(x) = f(x_k)$ (constant) over each interval

$$A \approx \frac{b-a}{n} \sum_{k=1}^{n} f(x_k)$$

$$x_0 = a, x_n = b$$

$$x_k = x_{k-1} + (b-a)/n$$

- Trapezoidal Rule: Linear Approximation

$$A \approx \frac{b-a}{n}\left[\frac{f(a)+f(b)}{2} + \sum_{k=1}^{n-1} f(x_k)\right]$$

- Simpson's Rule: Quadratic Approximation, n even

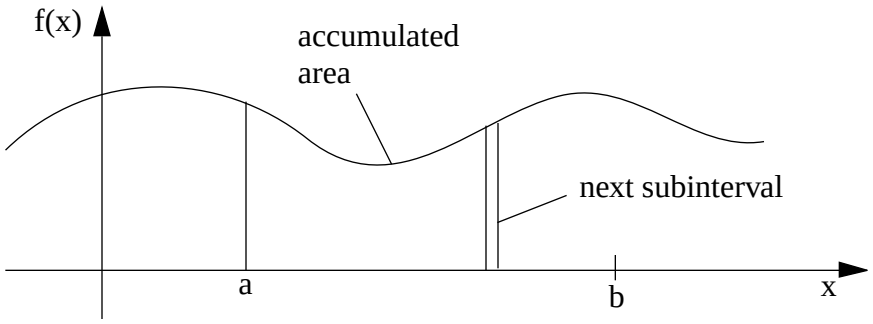
$$A \approx \frac{\Delta x}{3}\left[f(a) + f(b) + 4\sum_{\substack{k=1 \\ (odd)}}^{n-1} f(x_k) + 2\sum_{\substack{k=2 \\ (even)}}^{n-2} f(x_k)\right]$$

$\Delta x = (b-a)/n$

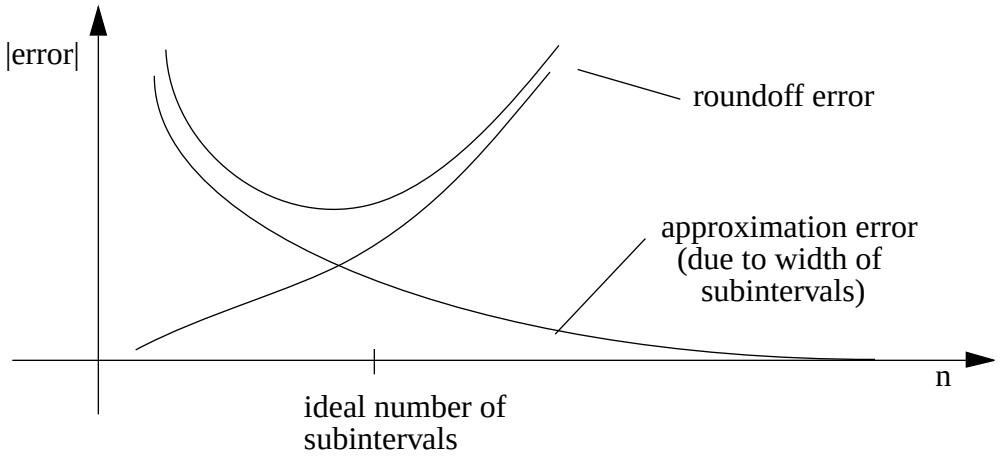
Numerical Integration Errors

- For small n, error is primarily due to approx. of f(x) (width of subintervals large).

- For large n, roundoff errors become significant; i.e., area of each subinterval small, and this value can get lost in precision of larger accumulated area:



Error Magnitude Plot:



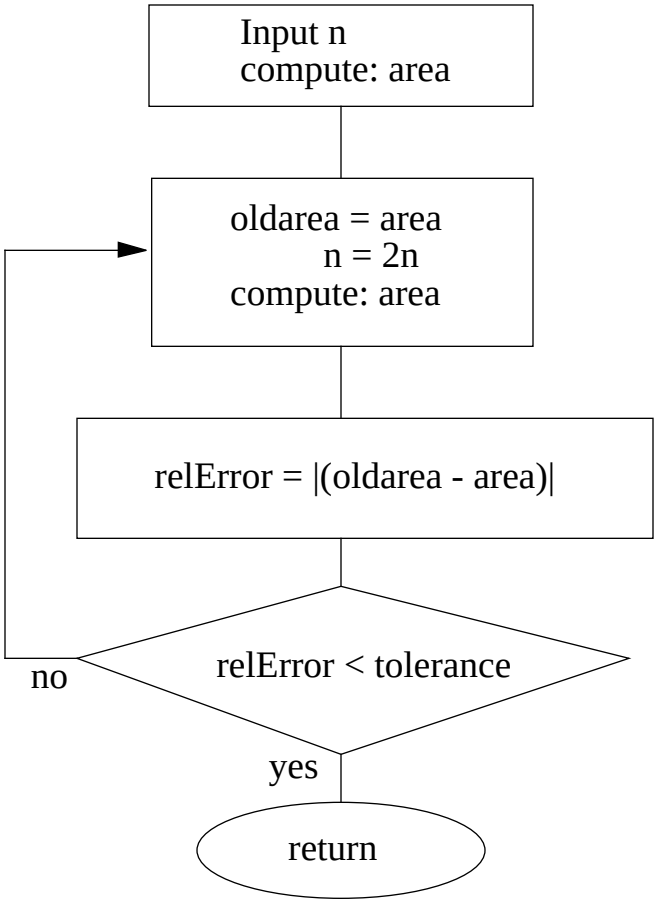
Ideal value for n depends on:

- Form of f(x)
- Integration method used
- Computer System

Numerical Integration Improvements

- Repeatedly increase number of subintervals until area estimate is within specified tolerance.

  A method for accomplishing this is to start with “large” subintervals and then double the number of intervals at each step:



- Also, could use higher-order polynomials to approximate f(x) over each of the n subintervals.

- And we could use higher precision in calculations to reduce roundoff errors.

(Tradeoff: more calculation time.)

Numerical Evaluation of Infinite Integrals

Divide integral into parts and apply a numerical integration technique to each part until contributions tend to zero.

For example:

$$\int_a^{\infty} f(x)dx = \int_{a_1}^{a_1} f(x)dx + \int_{a_1}^{a_2} f(x)dx + \int_{a_1}^{a_1} f(x)dx + \ldots$$

If integral is finite, then we must have

$$|f(x)| \to 0 \qquad as \qquad x \to \infty,$$

That is, we can compute the integral as a finite number of terms:

$$\int_a^{\infty} f(x)dx = \sum_{k=1}^{n} \int_{a_{k-1}}^{a_n} f(x)dx + \varepsilon$$

whter the error term is less than some specified tolerance.

Monte Carlo Methods

Use of random numbers to solve deterministic problems.

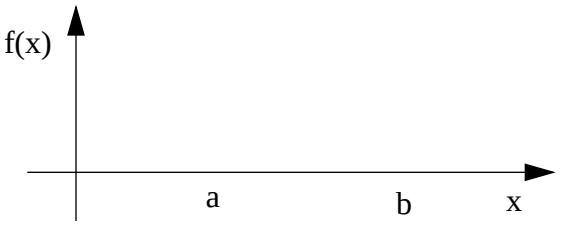
This approach useful for numerically evaluating complex integrals.

Examples:

- Multiple Integration

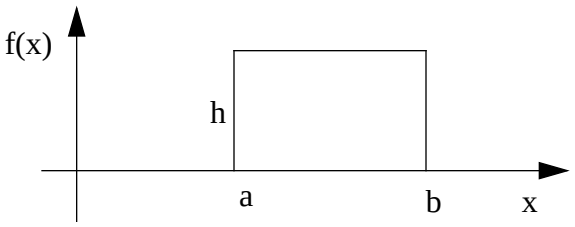
$$\int_{b_n}^{b_1}\int_{b_n}^{b_1}\cdots\int_{b_n}^{b_n} f(x_1, x_2, \ldots, x_n)dx_1 dx_2 \ldots dx_n$$

- Functions with High-Frequency Oscillations



Basic Approach:

Generate n uniformly distributed random points in an area containing f(x) and the interval (a, b) -- where area is not too large compared to integral area. Then compute:



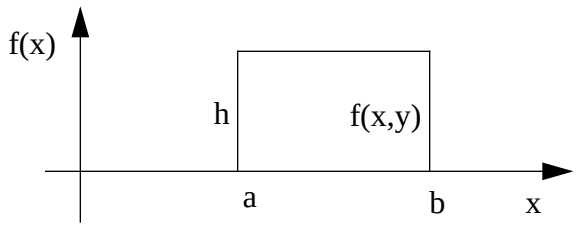
$$\frac{\int_a^b f(x)dx}{h(b-a)} = \lim_{n\to\infty}\left[\frac{\text{number of points under f(x)}}{n}\right]$$

where (b-a) may be positive or negative,
and we might have $f(x) < 0$ over part or all of the interval (a, b).

Algorithm: Monte Carlo Integration

1. Generate two uniformly distributed random numbers r_1, r_2 in the interval (0,1).
2. Assuming f*(x)* ≥ 0, calculate coordinate positions (*x*,*y*) as:

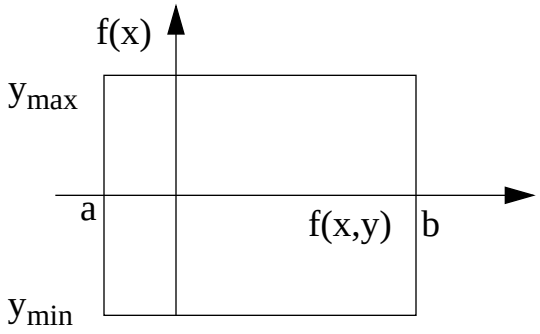
$$x = a + r_1(b - a)$$
$$y = r_2 h$$



Count point (*x*,*y*) as under *f(x)* if

$$0 \le y \le f(x) \qquad f(x) \ge 0$$

3. If *f(x)* < 0 for a value of *x* in the interval (*a*,*b*), calculate coordinate position as



$$h = y_{max} - y_{min}$$
$$x = a + r_1(b - a)$$
$$y = y_{min} + r_2 h$$

Count point (x,y) as "under" f(x) if

$$0 \ge y \ge f(x) \qquad for \qquad f(x) < 0$$

In this case, we subtract 1 from the point count; i.e.,

ptcount = *n*, -*n* where *n* is the number of points below the *x* axis.

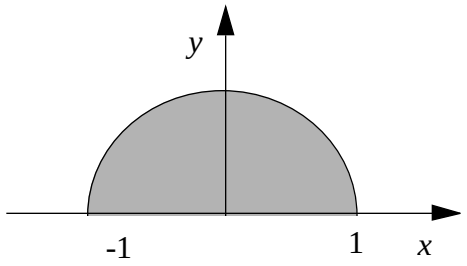
Pseudocode Algorithm - Monte Carlo Integration

```
Input f, a, b, h, n              (* assume f(x) > 0; how
width = b - a                      can this be modified
ptCount = 0                        for f(x) < 0 also? *)
Do k = 1, n
        x = a + width * Random( )
        y = h * Random( )
        If (y < f(x) ) Then
            ptCount = ptCount + 1
        Endofif
Endofdo
area = h * width * ptCount / n
```

Example: $\int_{-1}^{1} \sqrt{1 - x^2}\, dx$

$y_{min} = 0, \qquad y_{max} = 1$

$a = -1, b = 1, h = 1, n = 10$



	r_1	r_2	$x = -1 + 2r_1$	$y = r_2$	$f(x) = 1 - x_2$	ptcount1
1	0.972	0.466	0.944	0.466	0.330	0
2	0.069	0.001	-0.862	0.001	0.507	1
3	0.038	0.395	-0.924	0.395	0.382	1
4	0.740	0.267	0.480	0.267	0.877	2
5	0.152	0.948	-0.696	0.948	0.718	2
6	0.805	0.662	0.610	0.662	0.792	3

	r_1	r_2	$x = -1 + 2r_1$	$y = r_2$	$f(x) = 1 - x_2$	ptcount1
7	0.227	0.034	-0.546	0.034	0.838	4
8	0.607	0.835	0.214	0.835	0.977	5
9	0.212	0.463	0.576	0.463	0.817	6
10	0.670	0.624	0.340	0.624	0.940	7

Integration Functions in Mathematica

Integrate[f(x), x] - indefinite integral of f(x)

Integrate[f(x), {x, xmin, xmax}] - definite integral of f(x) with integration limits xmin and xmax

Integrate[f(x, y), {x, xmin, xmax}, {y, ymin, ymax}] - multiple integral of function f(x, y)

Nintegrate[. . .] - numerically evaluate integral

Examples:

```
Integrate[x^3,x]
```

$$\frac{x^4}{4}$$

```
Integrate[x^3,{x,a,b}]
```

$$\frac{-x^4}{4} + \frac{b^4}{4}$$

```
Integrate[x^3,{x,0,1}]
```
$\frac{1}{4}$

```
Integrate[Sqrt[1-x^2],{x,-1,1}]
```

$$\frac{Pi}{2}$$

```
NIntegrate[Sqrt[1-x^2],{x,-1,1}]

1.5708
```

```
f[x_, y_1] := x y^2

Integrate[f[x,y], {x,0,a}, {y,0,b}]
```

$$\frac{a^2 b^3}{6}$$

```
Integrate[f[x,y], {x,0,3}, {y,0,2}]

12

NIntegrate[f[x,y], {x,0,3}, {y,0,2}]

12.000000000000000001
```

(roundoff error)