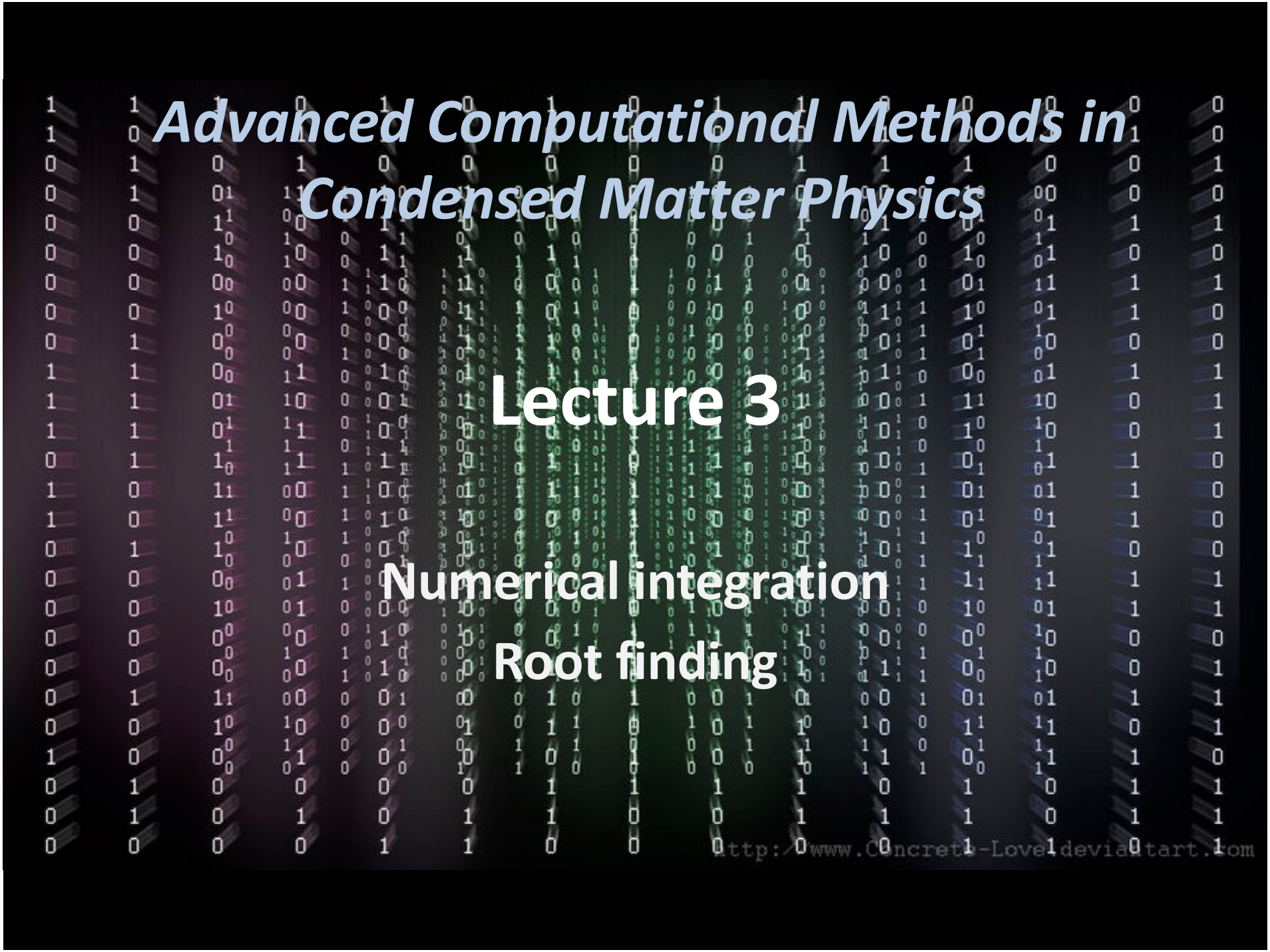

Advanced Computational Methods in Condensed Matter Physics
Lecture 3
Numerical integration
Root finding

Motivation: Definite Integral

To construct the definite integral of a function, we first define the *mesh points* $x_i^{(n)}$ according to

$$a = x_0^{(n)} < x_1^{(n)} < x_2^{(n)} \cdots < x_{n-1}^{(n)} < x_n^{(n)} = b,$$

and the *evaluation points* η_i are then each taken from within the appropriate subinterval, i.e., $\eta_i \in [x_{i-1}, x_i]$, $1 \leq i \leq n$, arbitrarily. Now, if we assume that

$$\lim_{n\to\infty} \left[\max_{1\leq i\leq n} \left(x_i^{(n)} - x_{i-1}^{(n)} \right) \right] = 0$$

(this simply means that the largest distance between adjacent points goes to zero as we take more and more points), then, under very mild conditions on f, it can be shown that the limit

$$L = \lim_{n\to\infty} \sum_{i=1}^{n} f(\eta_i)(x_i^{(n)} - x_{i-1}^{(n)})$$

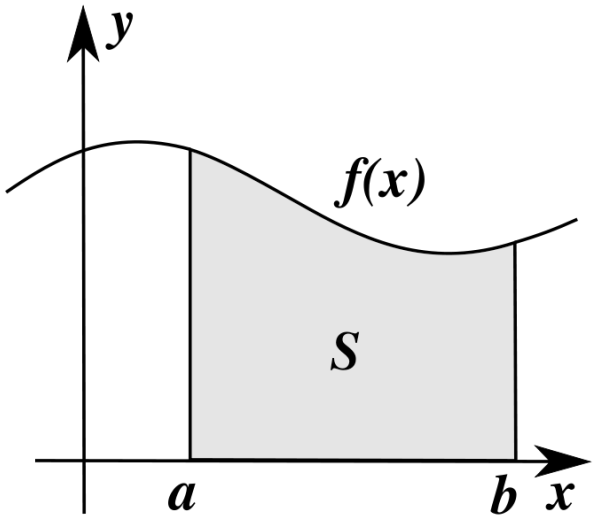
exists and its value is independent of the choices made for the grid points $\{x_i^{(n)}\}$ and evaluation points $\{\eta_i\}$. When this happens, we call this limit value the *definite integral* of f, and we write

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

(Numerical) Quadrature

- In general, a numerical integration is the approximation of a definite integration by a "weighted" sum of function values at discretized points within the interval of integration.

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{N} w_i f(x_i)$$



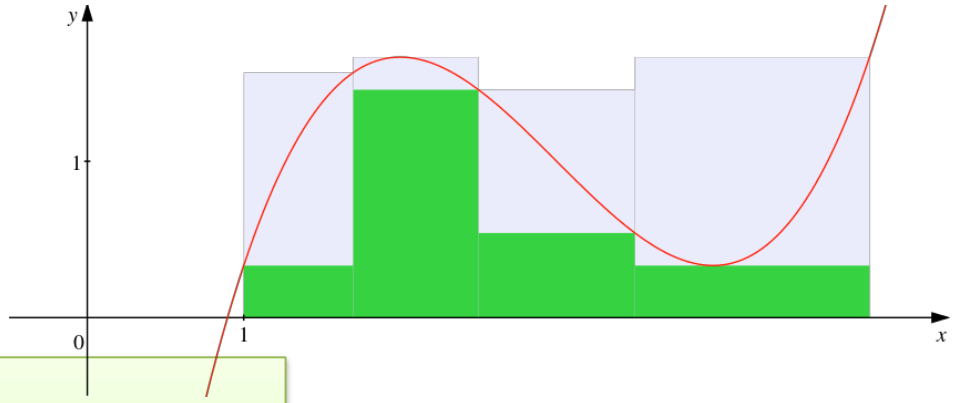
where w_i is the weighted factor depending on the integration schemes used, and $f(x_i)$ is the function value evaluated at the given point x_i

Methods

- Riemann integrals
  - Lower and Upper Sums
  - Midpoint Sums
- Newton Cotes formulas
  - Trapezoid Rules
  - Simpson's Rules
  - Adaptive Simpson's Scheme
  - Simpson's 8/3 and Bode's rule
- Gaussian Quadrature Formulas

Lower and Upper Sums

The lower and upper sums are defined as:



Lower: $$\Sigma^- \equiv \sum_{k=1}^{N} (x_k - x_{k-1}) \cdot \inf_{x_{k-1}<x<x_k} f(x)$$

Upper: $$\Sigma^+ \equiv \sum_{k=1}^{N} (x_k - x_{k-1}) \cdot \sup_{x_{k-1}<x<x_k} f(x)$$

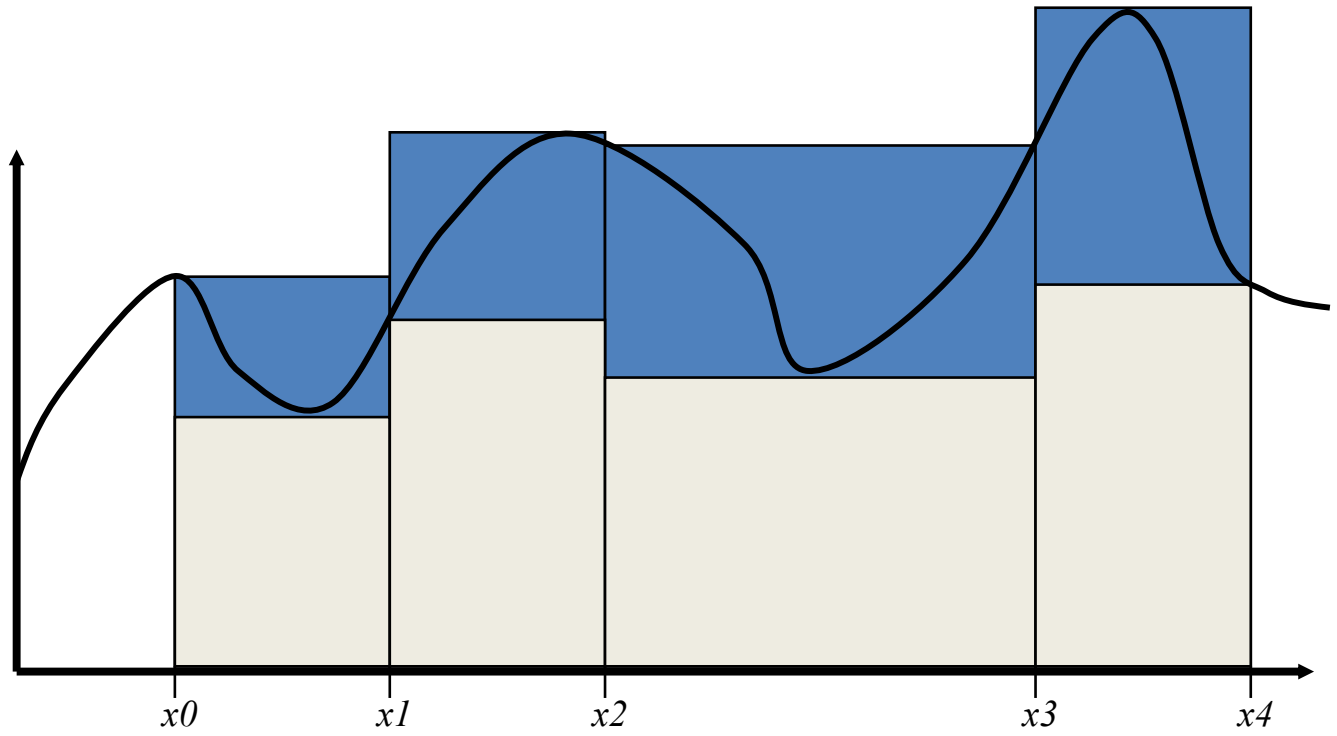
Define lower and upper bounds for the real integral, but are impractical because of “inf” and “sup”.

More useful left and right sums:

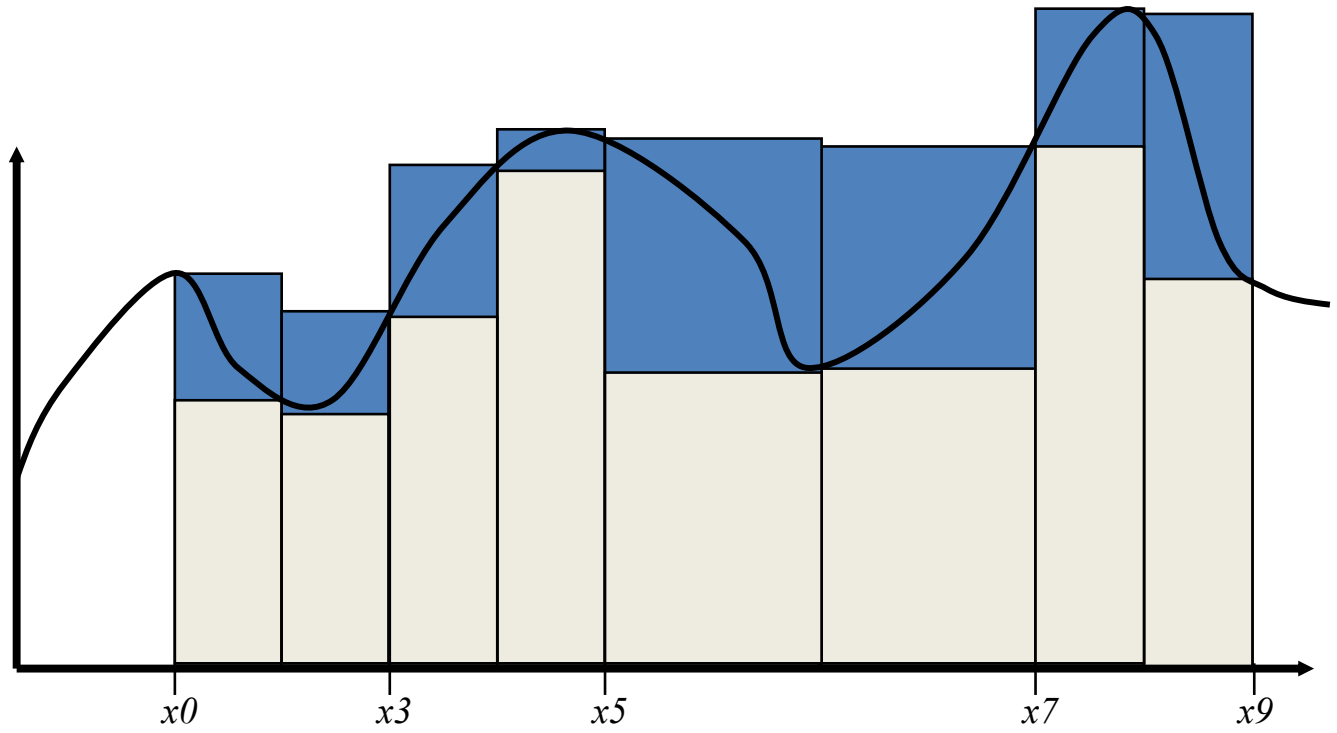
$$\Sigma^L \equiv \sum_{k=1}^{N} (x_k - x_{k-1}) \cdot f(x_{k-1}) \qquad \Sigma^R \equiv \sum_{k=1}^{N} (x_k - x_{k-1}) \cdot f(x_k)$$

Bounding approximations

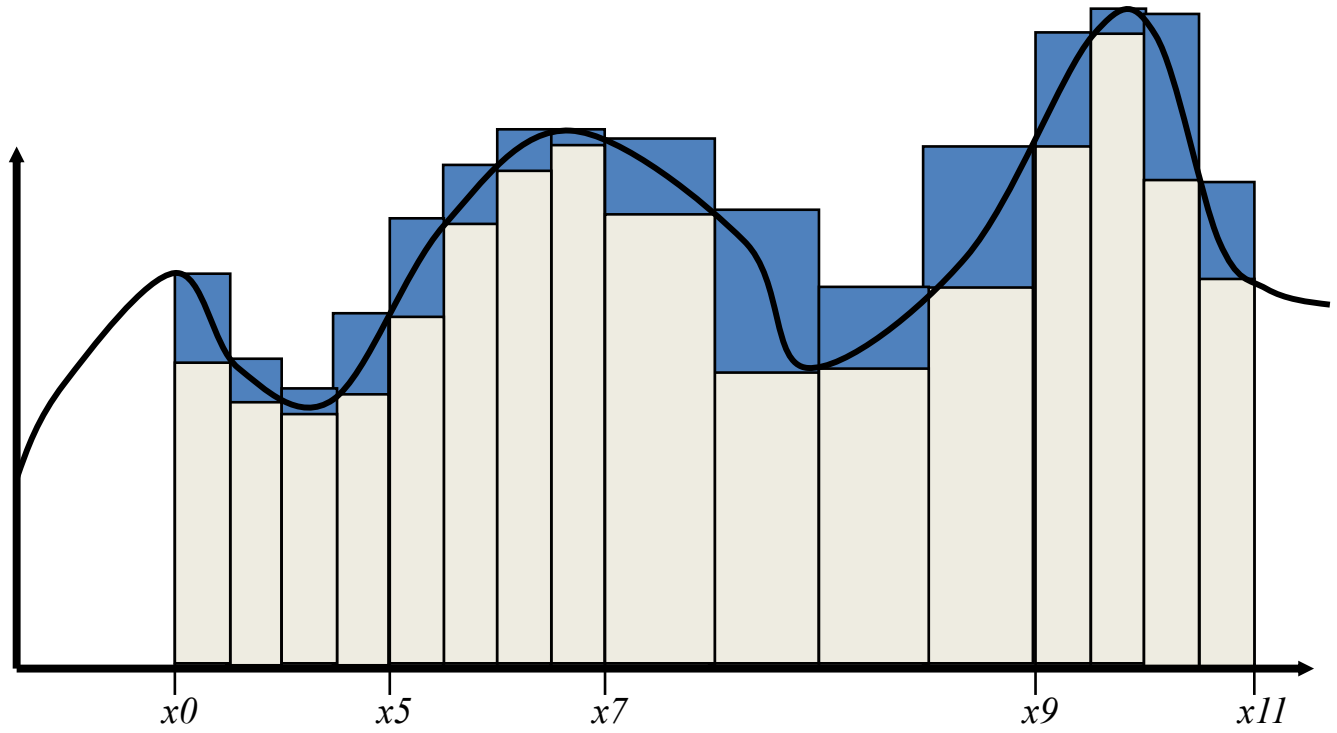
$$\Sigma^- \leq \int_a^b f(x)dx \leq \Sigma^+$$



Refinement 1

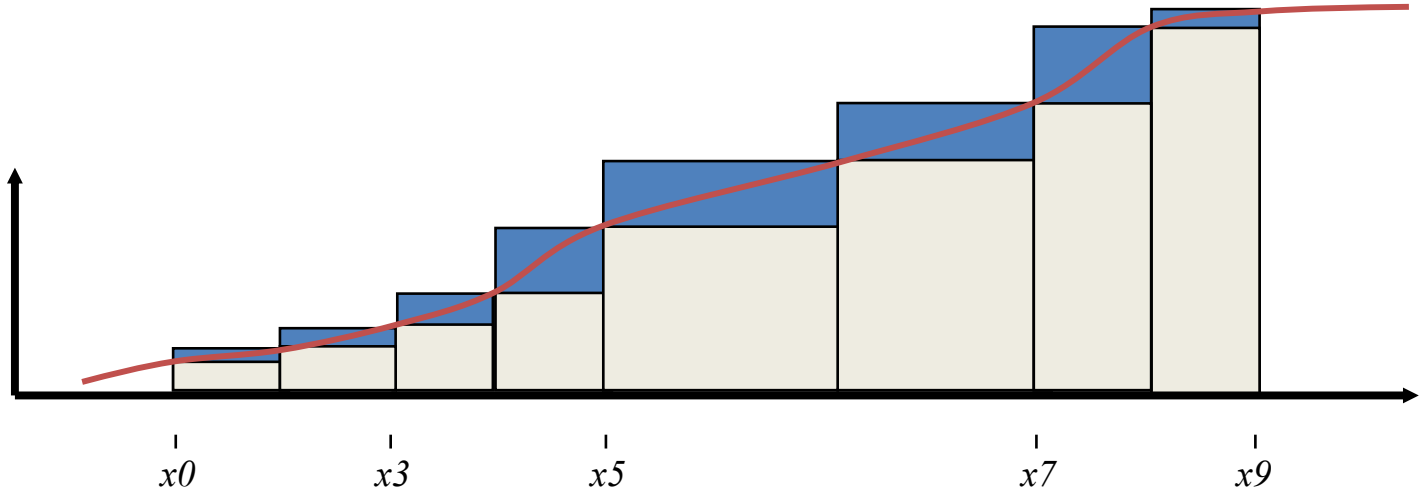


Refinement 2

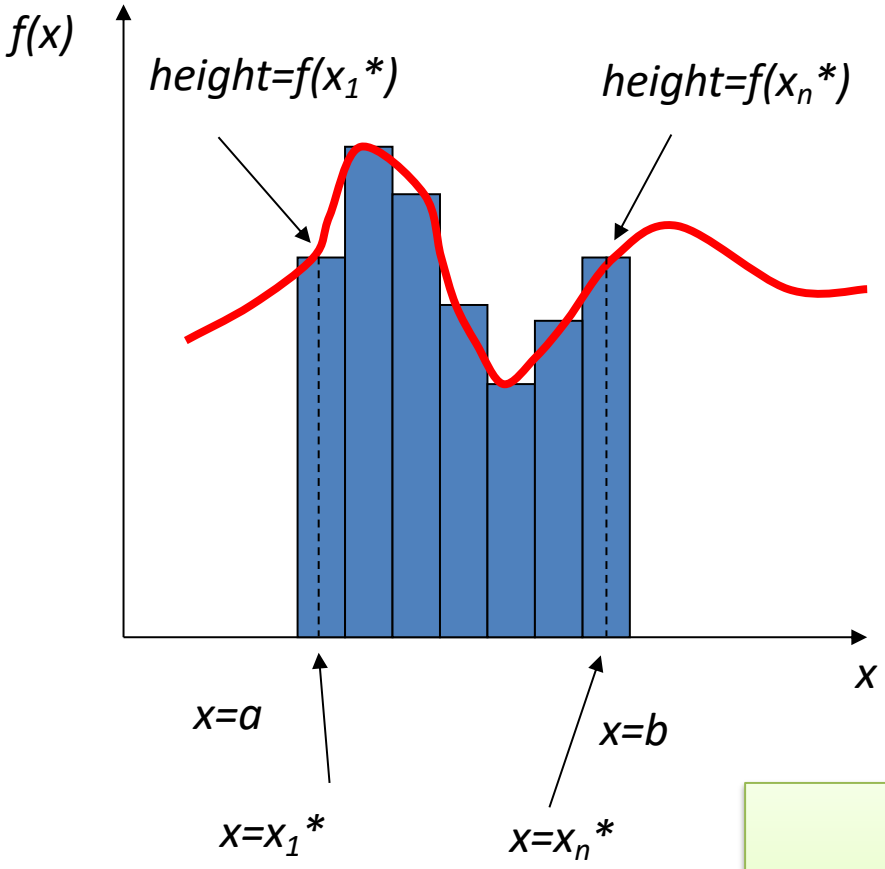


Monotonic functions

- Note that if a function is monotonically increasing (or decreasing), then the lower sum corresponds to the left partition values, and the upper sum corresponds to the right partition values.



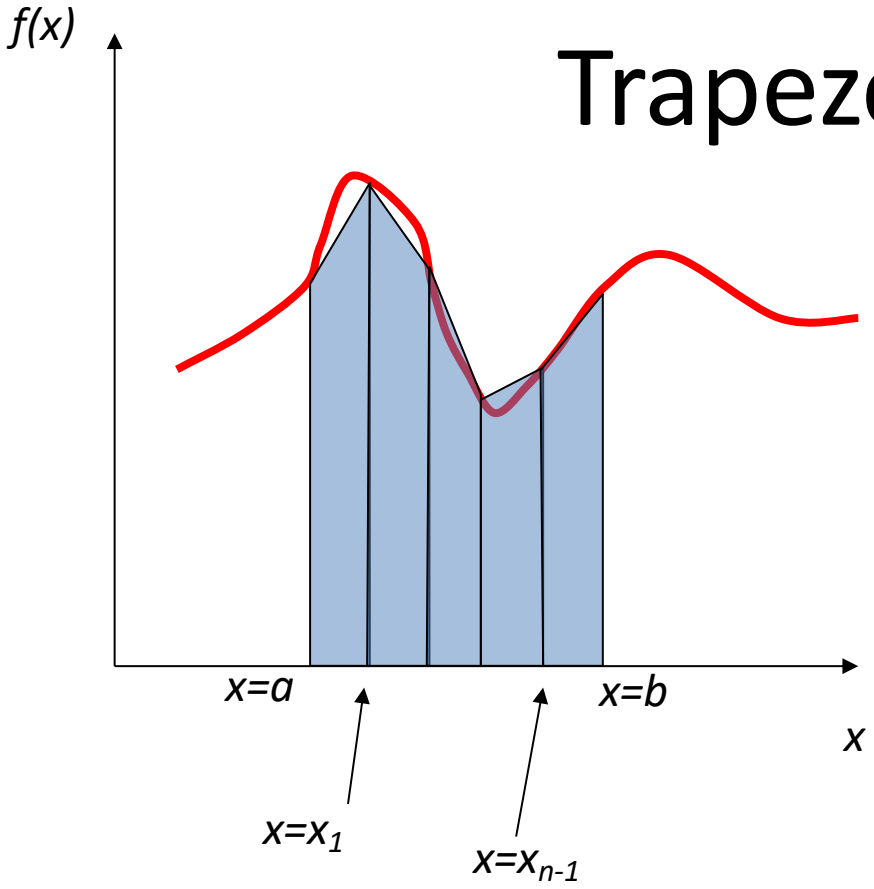
Composite Midpoint Rule



- N intervals between a and b
- Approximate function by rectangle in each interval
- Height of each rectangle: $f(x^*_k)$ with $x^*_k=(x_k+x_{k-1})/2$

$$\Sigma^M \equiv \sum_{k=1}^{N}(x_k - x_{k-1}) \cdot f[(x_k + x_{k-1})/2]$$

Trapezoidal Rule



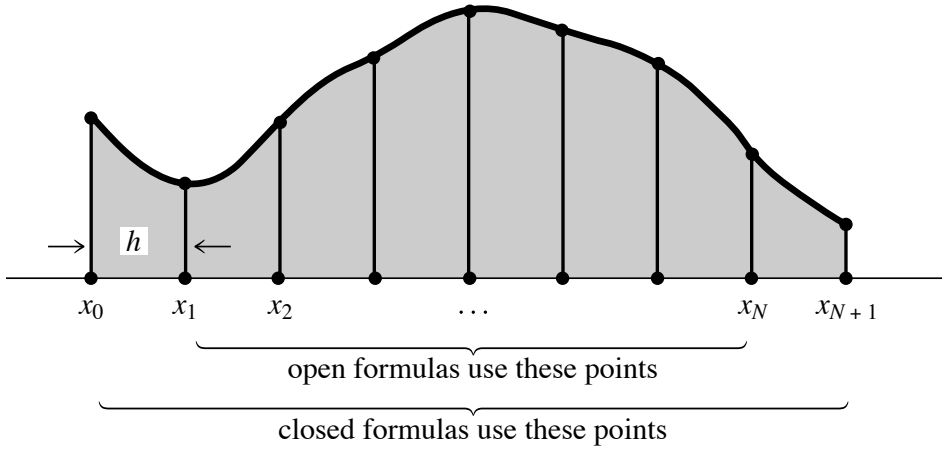
The midpoint rule can be made more accurate by using trapezoids to replace the rectangles as shown.

A linear approximation of the function locally sometimes work much better than using the averaged value like the midpoint rule does.

$$\Sigma^T \equiv \sum_{k=1}^{N} (x_k - x_{k-1}) \cdot \frac{f(x_{k-1}) + f(x_k)}{2}$$

Higher Order Newton-Cotes formulas

From now on we assume equally spaced intervals, i.e., h=(⊿x=)(b-a)/N



Remarks:

- So far we have shown closed formulas, using the "end-"values f(a) and f(b)
- If f(a) and f(b) are difficult to compute: use open formulas
- Error of the trapezoidal rule:
  i.e. only (piecewise) linear
  functions are integrate exactly

$$\int_{x_1}^{x_2} f(x)dx = h\left[\frac{1}{2}f_1 + \frac{1}{2}f_2\right] \quad + O(h^3 f'')$$

- More accurate integration formula can be achieved by approximating the local curve by a higher order functions, e.g. polynomials.

Polynomial approximation

- *Idea:* replace *f(x)* in an interval with a known and simple function
- Here: approximate *f*(*x*) by an m^{th} order polynomial:

$$p_m(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_m x^m$$

- To determine the coefficients a_i, we need (m+1) equations
- This means we pick m+1 intermediate x-coordinates, $x_{k,i}$, within each interval $[x_{k-1},x_k]$ and solve: $f(x_{k,i})=p_m(x_{k,i})$
- Usually $x_{k,i}=x_{k-1}+i\,h/(m-1)$, i=0,…,m-1; i.e. $x_{k,m-1}=x_k$
- Examples:

m	Polynomial	Formula	Error
1	linear	Trapezoid	$O(h^2)$
2	quadratic	Simpson's 1/3	$O(h^4)$
3	cubic	Simpson's 3/8	$O(h^4)$

Example m=1

- Most simple approximation of f: **first order polynomial** (a straight line)
- Newton's form of the interpolating polynomial (N=1):

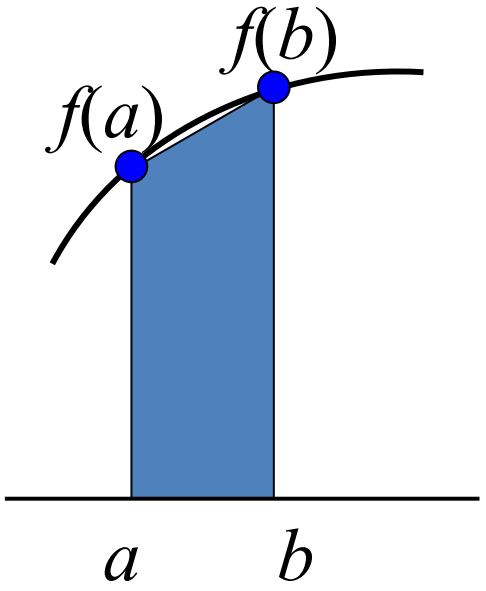
$$p_1(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a)$$

- Now, solve the integral:

$$I = \int_a^b f(x)dx \approx \int_a^b p_1(x)dx$$

→ Trapezoid role:

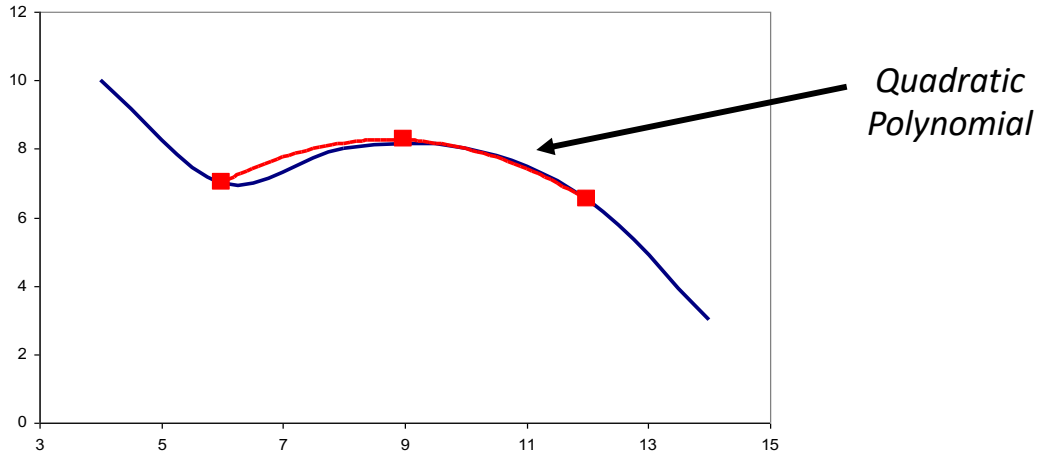
$$I \approx \frac{(b-a)}{2}[f(a) + f(b)]$$



The famous Simpson's Rule

For m=2, we obtain Simpson's 1/3 rule:

$$\Sigma^T \equiv \sum_{k=1}^{N} \frac{h}{6} \cdot (f(x_{k-1}) + 4f(x_{k-1} + h/2) + f(x_k))$$



Error: $O(h^5 f^{(4)})$ for one interval. $f^{(4)}$ is to be taken at some point within the interval

Simpson's rule for x^3

N=1:

$$\int_a^b x^3\,dx = \frac{1}{4}\left(b^4 - a^4\right) = \frac{1}{4}(b-a)\left(b^3 + b^2a + ba^2 + a^3\right)$$

$$= \frac{1}{4}(b-a)\left(\frac{2}{3}b^3 + \frac{1}{3}b^3 + b^2a + ba^2 + \frac{1}{3}a^3 + \frac{2}{3}a^3\right)$$

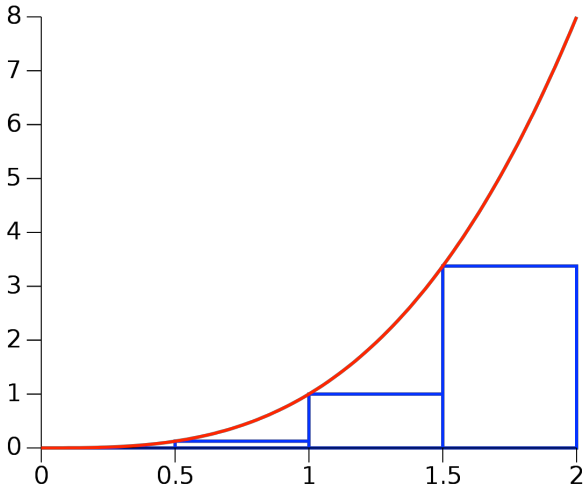
$$= \frac{1}{4}(b-a)\left(\frac{2}{3}b^3 + \frac{1}{3}(b+a)^3 + \frac{2}{3}a^3\right)$$

$$= \frac{1}{4}(b-a)\left(\frac{2}{3}b^3 + \frac{8}{3}\left(\frac{b+a}{2}\right)^3 + \frac{2}{3}a^3\right)$$

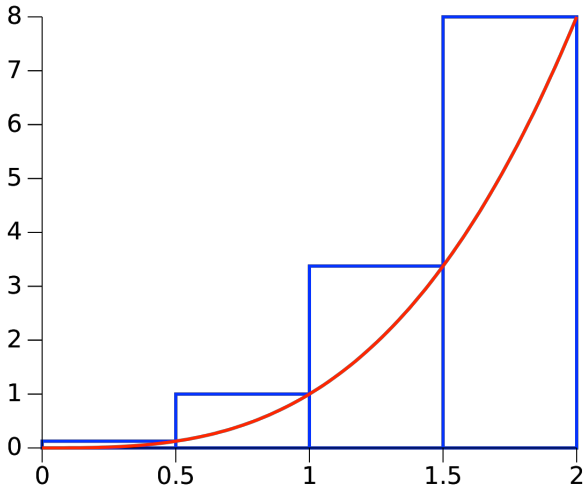
$$= \frac{1}{6}(b-a)\left(b^3 + 4\left(\frac{b+a}{2}\right)^3 + a^3\right)$$

$$= \frac{1}{6}(b-a)\left(f(a) + 4f\left(\frac{b+a}{2}\right) + f(b)\right)$$

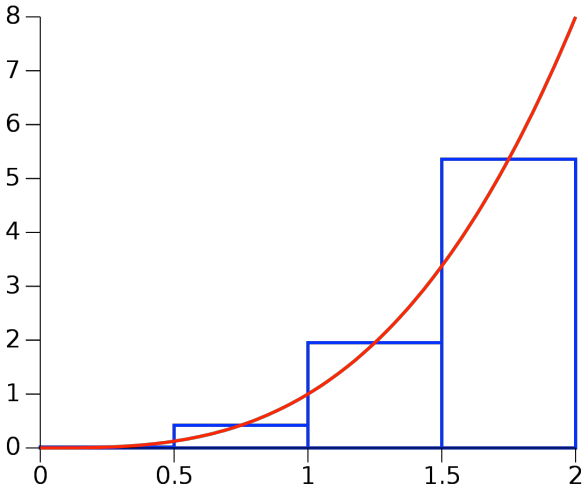
Illustrations for $f(x)=x^3$



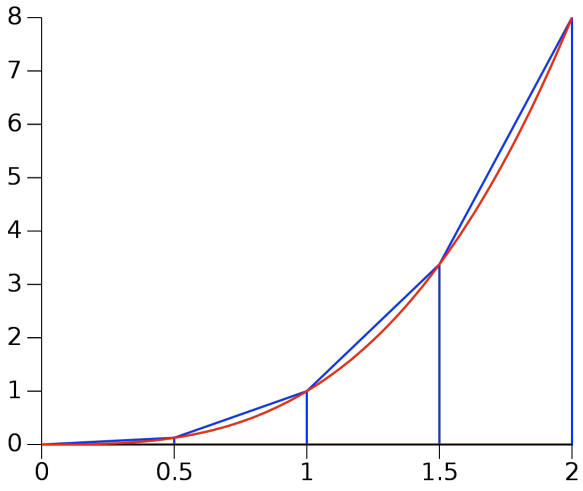
left



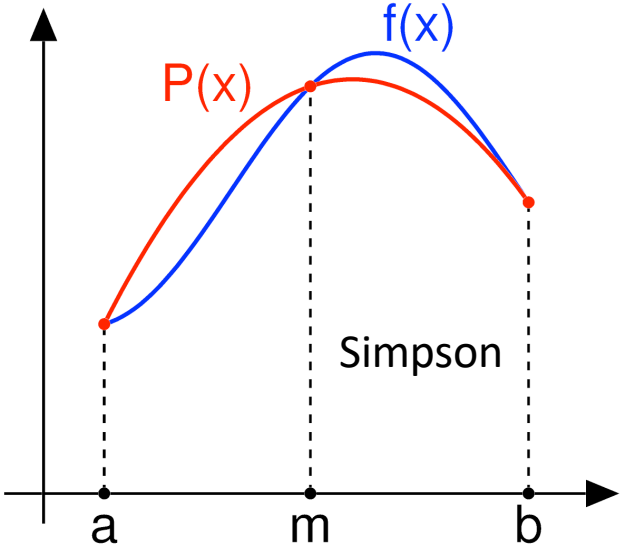
right



midpoint



trapezoidal

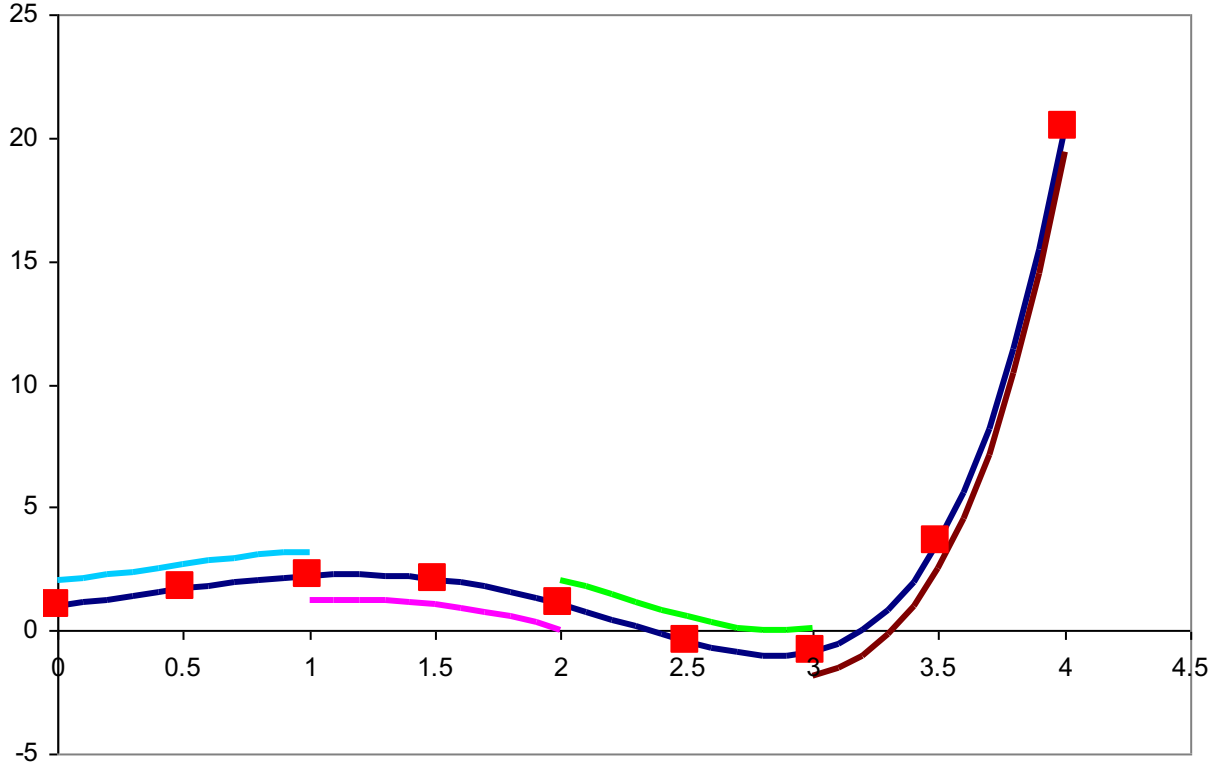


Composite Simpson 1/3

For N=2: $$\frac{h}{6}\left(f(a)+4f(a+h/2)+2f\left(a+h\right)+4f\left(a+3h/2\right)+f\left(b\right)\right)$$

i.e., the end-points have weight 1, interior sub-interval points 2, and half sub-intervals 4

- Example: 9 points, 4 intervals



m>2

m=3: Simpson's 3/8 rule: $$\int_{x_1}^{x_4} f(x)dx = h\left[\frac{3}{8}f_1 + \frac{9}{8}f_2 + \frac{9}{8}f_3 + \frac{3}{8}f_4\right] + O(h^5 f^{(4)})$$

m=4: Bode's rule: $$\int_{x_1}^{x_5} f(x)dx = h\left[\frac{14}{45}f_1 + \frac{64}{45}f_2 + \frac{24}{45}f_3 + \frac{64}{45}f_4 + \frac{14}{45}f_5\right] + O(h^7 f^{(6)})$$

Remarks:

- The above expressions use multiple intervals for the approximation, i.e., h → mh
- Simpson's 1/3 rule is exact for polynomials up to order 3!
- Simpson's 3/8 rule is exact for polynomials up to order 3! (no "lucky cancellation")
- Bode's rule is exact for polynomials up to order 5!

Demo

Numerical recipes: "with the exception of two of the most modest formulas ("extended trapezoidal rule," and "extended midpoint rule,"), the classical formulas are almost entirely useless.
They are museum pieces, but beautiful ones."

Adaptive algorithms

Much better than to predefine the number of sub-intervals, N, is to refine the integration rule until some specified degree of accuracy has been achieved.

Example:
adaptive refinement of trapeziod rule (equivalent to Simpson's 1/3 rule)

```
#include <math.h>
#define EPS 1.0e-6
#define JMAX 20

float qsimp(float (*func)(float), float a, float b)
Returns the integral of the function func from a to b. The parameters EPS can be set to the
desired fractional accuracy and JMAX so that 2 to the power JMAX-1 is the maximum allowed
number of steps. Integration is performed by Simpson's rule.
{
    float trapzd(float (*func)(float), float a, float b, int n);
    void nrerror(char error_text[]);
    int j;
    float s,st,ost=0.0,os=0.0;

    for (j=1;j<=JMAX;j++) {
        st=trapzd(func,a,b,j);
        s=(4.0*st-ost)/3.0;          Compare equation (4.2.4), above.
        if (j > 5)                   Avoid spurious early convergence.
            if (fabs(s-os) < EPS*fabs(os) ||
                (s == 0.0 && os == 0.0)) return s;
        os=s;
        ost=st;
    }
    nrerror("Too many steps in routine qsimp");
    return 0.0;                      Never get here.
}
```

Romberg integration

- Romberg's method is a natural generalization of the adaptive trapezoid rule.
- higher order than Simpson's rule.
- basic idea: use the results from k successive refinements of the extended trapezoidal rule to remove all terms in the error series up to but not including $O(1/N^{2k})$.
- goes also by the name of *Richardson's deferred approach to the limit*: Perform some numerical algorithm for various values of a parameter h, and then extrapolate the result to the continuum limit h = 0.

Gaussian Quadrature

- **Gaussian quadratures are very powerful tools for approximating integrals.**
- **Quadrature rules are all based on special values of weights and Gauss points. These are pre-computed**
- They are open formulas
- Gauss points are not equidistant
- Superior accuracy over (open) Newton-Cotes formulas
- Basic form:

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

c_i : weighting factors

x_i : Gauss sampling points selected optimally

- Note that the interval is between -1 and 1
- For other intervals, a change of variables is used to transfer the problem so that it utilizes the interval [-1, 1]
- The number of unknowns (x_i & c_i) -1 determine the polynomial acuracy

Example of deriving GQ

- Here we concentrate on N=1 (composite rule are straight forward)
- For *n*=2, we have: $$I \approx c_1 f(x_1) + c_2 f(x_2)$$
- This leads to 4 unknowns: c_1, c_2, x_1, and x_2
  - two unknown weights (c_1, c_2)
  - two unknown sampling points (x_1, x_2)
- *we need four known values for the equation.*
- *If we had these, we could then attempt to solve for the four unknowns.*
- *Let's with polynomials*
- *For n=2, let's look at: 1, x,* x^2, x^3

...

- Recalling the formula: $I \approx c_1 f(x_1) + c_2 f(x_2)$
  - Constant
    - *f(x)=1*
  - Linear
    - *f(x)=x*
  - Quadratic
    - *f(x)=x²*
  - Cubic
    - *f(x)=x³*

$$\int_{-1}^{1} 1\,dx = 2 \quad = c_1 f(x_1) + c_2 f(x_2) = c_1 + c_2$$

$$\int_{-1}^{1} x\,dx = 0 \quad = c_1 f(x_1) + c_2 f(x_2) = c_1 x_1 + c_2 x_2$$

$$\int_{-1}^{1} x^2\,dx = \frac{2}{3} = c_1 f(x_1) + c_2 f(x_2) = c_1 x_1^{\,2} + c_2 x_2^{\,2}$$

$$\int_{-1}^{1} x^3\,dx = 0 \quad = c_1 f(x_1) + c_2 f(x_2) = c_1 x_1^{\,3} + c_2 x_2^{\,3}$$

2nd order GQ points/weights

Solving these non-linear equations gives:

$$c_1 = c_2 = 1$$

$$x_1 = -\frac{1}{\sqrt{3}} = -0.577$$

$$x_2 = \frac{1}{\sqrt{3}} = 0.577$$

→ Gauss-Legendre formula:

$$I \approx f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right)$$

→ This is **exact** for **all** polynomials up to and including degree 3!

3rd order GQ

$$\int_{-1}^{1} 1\,dx = 2 \quad = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) = c_1 + c_2 + c_3$$

$$\int_{-1}^{1} x\,dx = 0 \quad = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) = c_1 x_1 + c_2 x_2 + c_3 x_3$$

$$\int_{-1}^{1} x^2\,dx = \frac{2}{3} = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) = c_1 x_1^{\,2} + c_2 x_2^{\,2} + c_3 x_3^{\,2}$$

$$\int_{-1}^{1} x^3\,dx = 0 \quad = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) = c_1 x_1^{\,3} + c_2 x_2^{\,3} + c_3 x_3^{\,3}$$

$$\int_{-1}^{1} x^4\,dx = \frac{2}{5} \quad = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) = c_1 x_1^{\,4} + c_2 x_2^{\,4} + c_3 x_3^{\,4}$$

$$\int_{-1}^{1} x^5\,dx = 0 \quad = c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) = c_1 x_1^{\,5} + c_2 x_2^{\,5} + c_3 x_3^{\,5}$$

...

- Solution of these equations gives

$$c_1 = 5/9 \qquad c_2 = 8/9 \qquad c_3 = 5/9$$

$$x_1 = -\sqrt{3/5} = -0.77459669 \qquad x_2 = 0 \qquad x_2 = \sqrt{3/5} = 0.77459669$$

- Produces the three point Gauss-Legendre formula

$$I \approx c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3)$$

$$I \approx \frac{5}{9} f\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9} f(0) + \frac{5}{9} f\left(\sqrt{\frac{3}{5}}\right)$$

  - Exact for polynomials up to and including degree 5

GQ weights and Gauss points

n	2	3	4	5	6
c_i	1.0	0.5555555556	0.3478548451	0.2369268850	0.1713245
	1.0	0.8888888889	0.6521451549	0.4786286705	0.3607616
		0.5555555556	0.6521451549	0.5688888889	0.4679139
			0.3478548451	0.4786286705	0.4679139
				0.2369268850	0.3607616
					0.1713245
x_i	−0.5773502692	−0.7745966692	−0.8611363116	−0.9061798459	−0.932469514
	0.5773502692	0.0000000000	−0.3399810436	−0.5384693101	−0.661209386
		0.7745966692	0.3399810436	0.0000000000	−0.238619186
			0.8611363116	0.5384693101	0.238619186
				0.9061798459	0.661209386
					0.932469514

.. even more

TABLE 5.5 Gaussian Quadrature Nodes and Weights

n	$x_i^{(n)}$	$w_i^{(n)}$
1	0.0000000000000000E+00	0.2000000000000000E+01
2	±0.5773502691896257E+00	1.0000000000000000E+00
4	±0.3399810435848563E+00	0.6521451548625464E+00
	±0.8611363115940526E+00	0.3478548451374476E+00
8	±0.1834346424956498E+00	0.3626837833783620E+00
	±0.5255324099163290E+00	0.3137066458778874E+00
	±0.7966664774136268E+00	0.2223810344533745E+00
	±0.9602898564975362E+00	0.1012285362903697E+00
16	±0.9501250983763744E-01	0.1894506104550685E+00
	±0.2816035507792589E+00	0.1826034150449236E+00
	±0.4580167776572274E+00	0.1691565193950024E+00
	±0.6178762444026438E+00	0.1495959888165733E+00
	±0.7554044083550030E+00	0.1246289712555339E+00
	±0.8656312023878318E+00	0.9515851168249290E–01
	±0.9445750230732326E+00	0.6225352393864778E–01
	±0.9894009349916499E+00	0.2715245941175185E–01

GQ summary

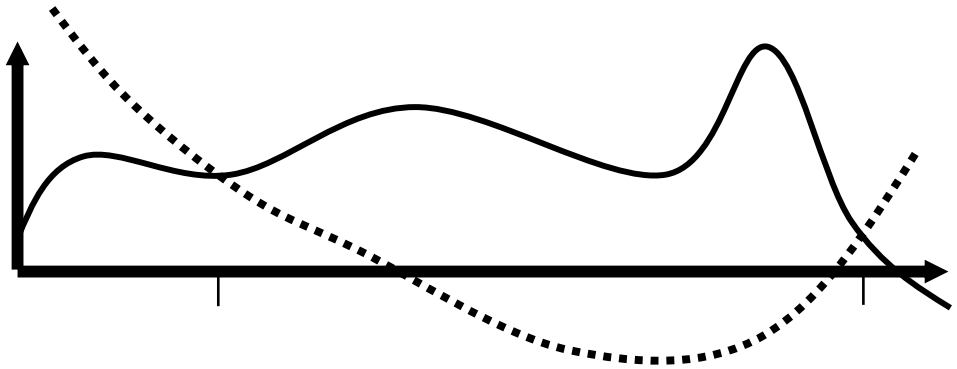
- Requires function evaluations at non-uniformly spaced points within the integration interval
  - not appropriate for cases where the function is unknown
  - not suited for dealing with tabulated data that appear in many engineering problems
- If the function is known, its efficiency can be a decided advantage

Root finding

- Bi-section Method
- Newton's method
- Secant Method
- Generalized Newton's method for systems of non-linear equations
  - The Jacobian matrix

Motivation

- Many problems can be re-written into a form such as:
  - f(x,y,z,…) = 0
  - f(x,y,z,…) = g(s,q,…)



Bisection Method

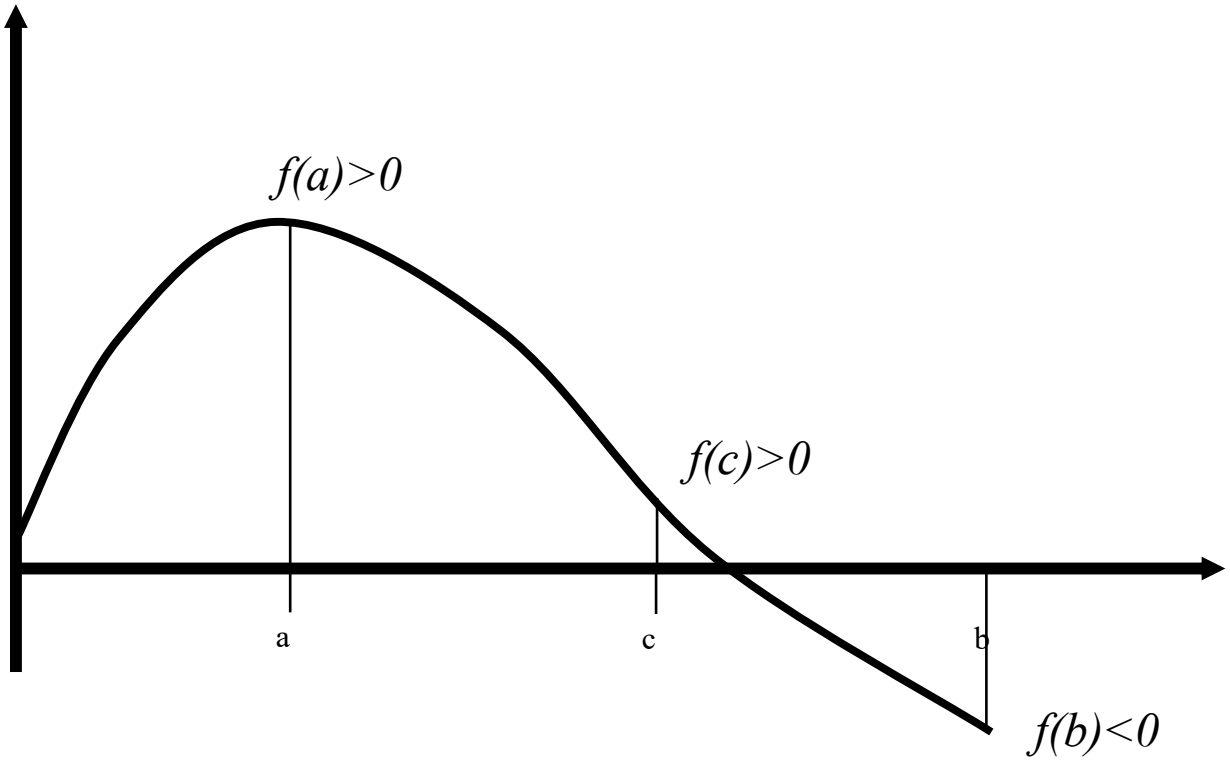
- Based on the fact that a continuous function will change signs as it passes through the root:

  f(a)*f(b) < 0

  i.e., f(x) is zero in the interval [a,b]
- Once we have a root ***bracketed***, we simply evaluate the mid-point and halve the interval.

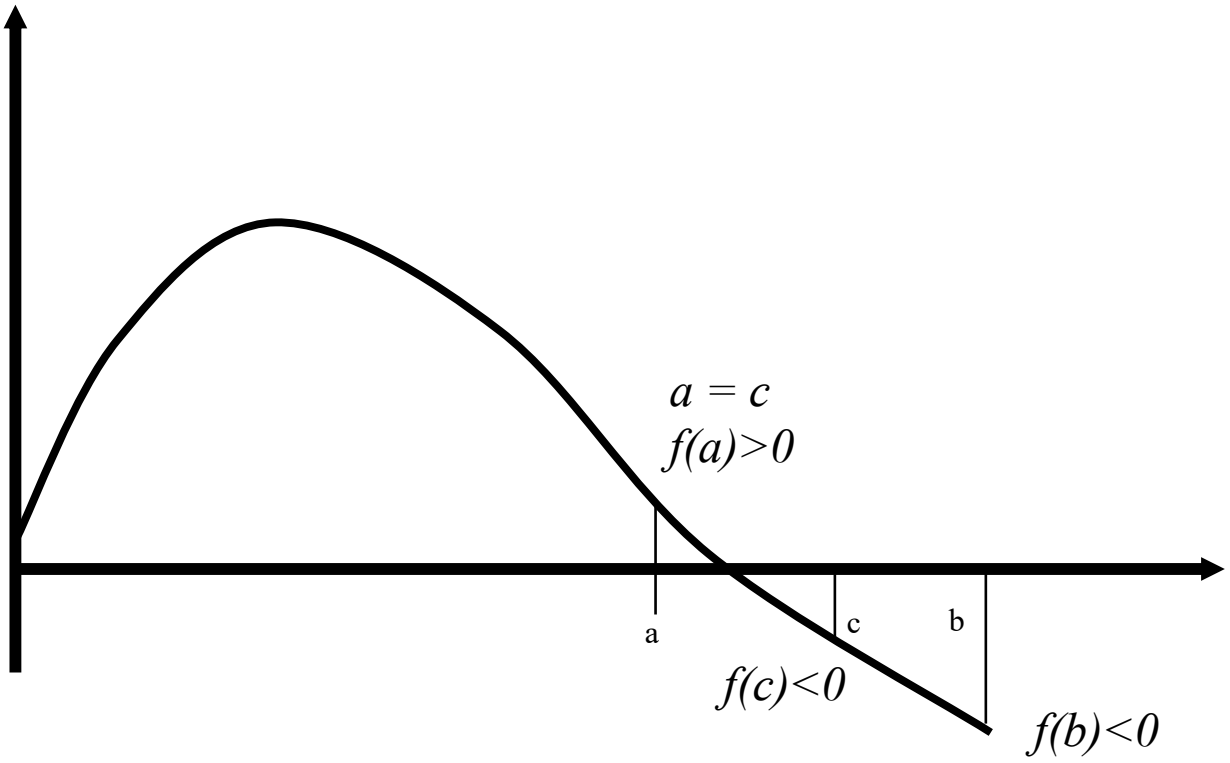
example

- c=(a+b)/2



...

- Guaranteed to converge to a root if one exists within the bracket.



...

Simple algorithm:

```
Given: a and b, such that f(a)*f(b)<0
Given: error tolerance, err

c=(a+b)/2.0; // Find the midpoint
While( |f(c)| > err ) {
    if( f(a)*f(c) < 0 )  // root in the left half
        b = c;
    else                 // root in the right half
        a = c;
    c=(a+b)/2.0; // Find the new midpoint
}
return c;
```

Bisection error

- The bisection method converges linearly or first-order to the root.
- We gain an extra bit accuracy each iteration
- If we need an accuracy of 0.0001 and our initial interval (b-a)=1, then:

  $2^{-n} < 0.0001 \Rightarrow$ 14 iterations

- Not bad, why do I need anything else?

Newton's method

- Open solution, that requires only one current guess.
- Root does not need to be bracketed.
- Consider some point x_0.
  - If we approximate *f(x)* as a line about x_0, then we can again solve for the root of the line.

$$l(x) = f'(x_0)(x - x_0) + f(x_0)$$

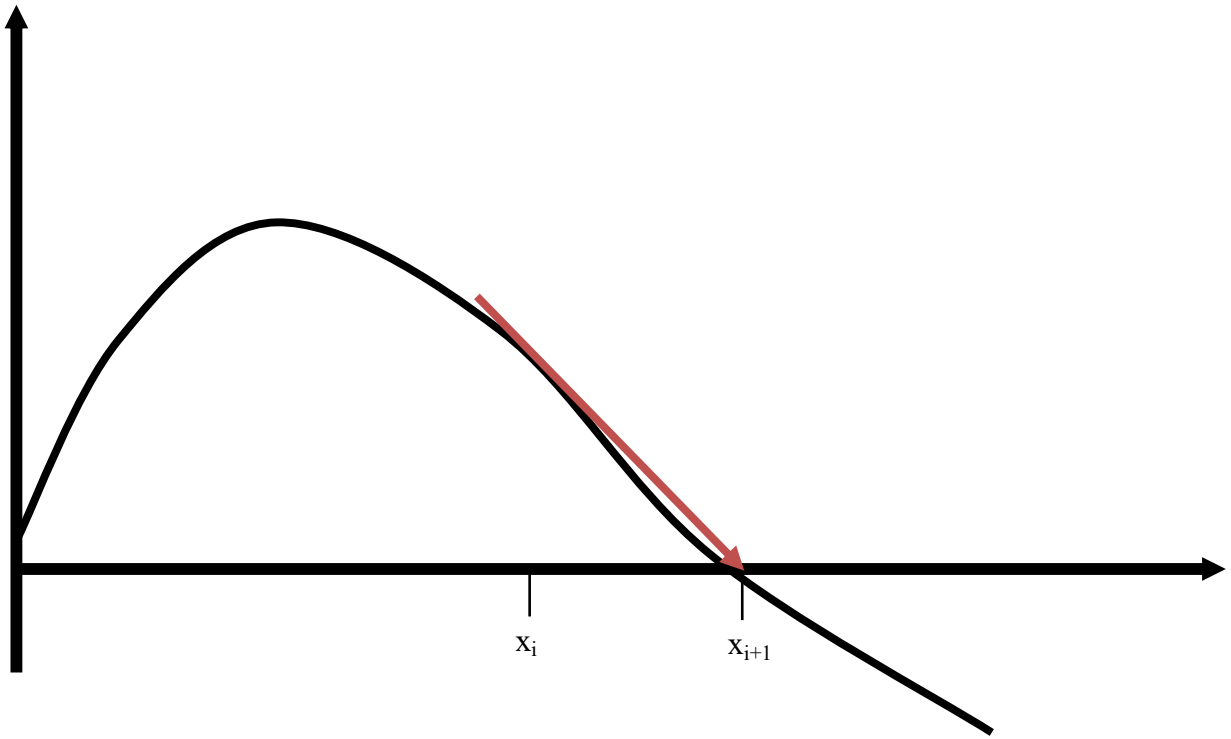
Iteration:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Example

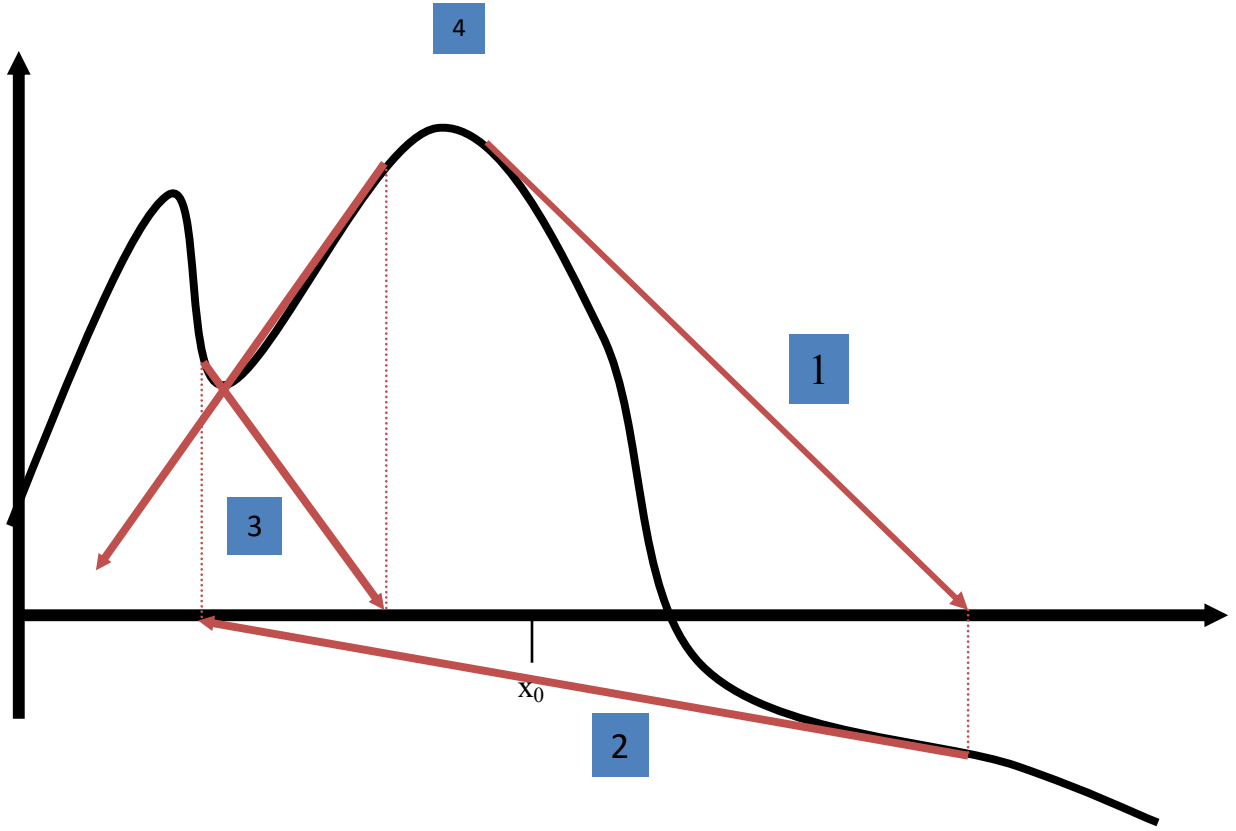
- Graphically, follow the tangent vector down to the x-axis intersection.



Problems

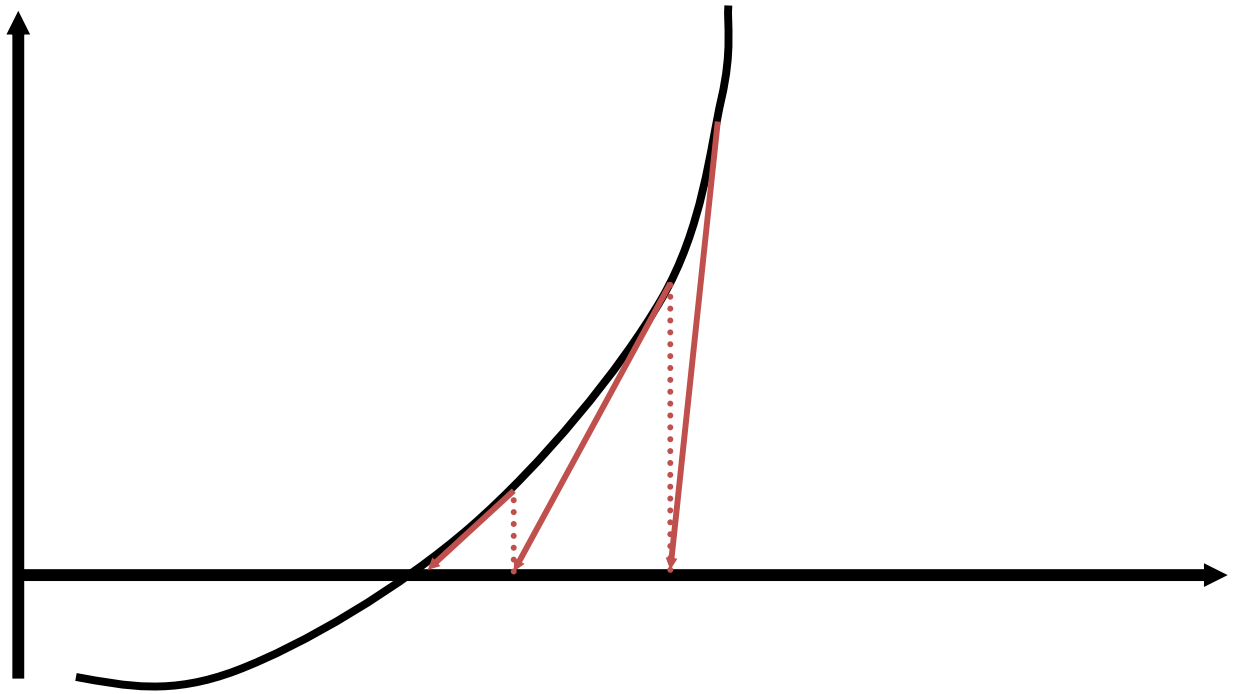
Newton's method can

- *diverge*
- *form loops*



...

- Need the initial *guess* to be close, **or**, the function to behave nearly linear within the range.

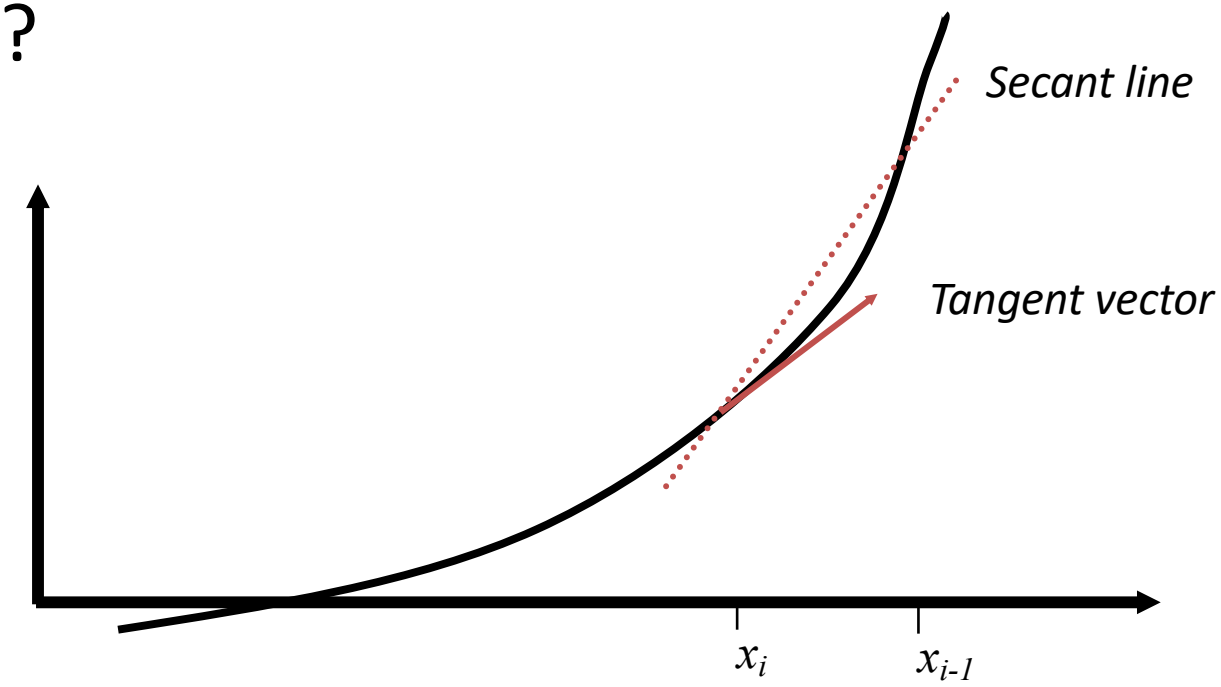


Newton's Algorithm

- Requires the derivative function to be evaluated, hence more function evaluations per iteration.
- A robust solution would check to see if the iteration is stepping too far and limit the step.
- Most uses of Newton's method assume the approximation is pretty close and apply one to three iterations blindly.

Secant Method

- What if we do not know the derivative of *f(x)*?



...

- As we converge on the root, the secant line approaches the tangent.
- Hence, we can use the secant line as an estimate and look at where it intersects the x-axis (its root).

...

- This also works by looking at the definition of the derivative:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

- Therefore, Newton's method gives:

$$x_{k+1} = x_k - \left(\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \right) f(x_k)$$

- Which is the Secant Method.

Next lecture:

- Explicit and implicit PDE discretization
- (Pseudo) random number generators
- Data analysis