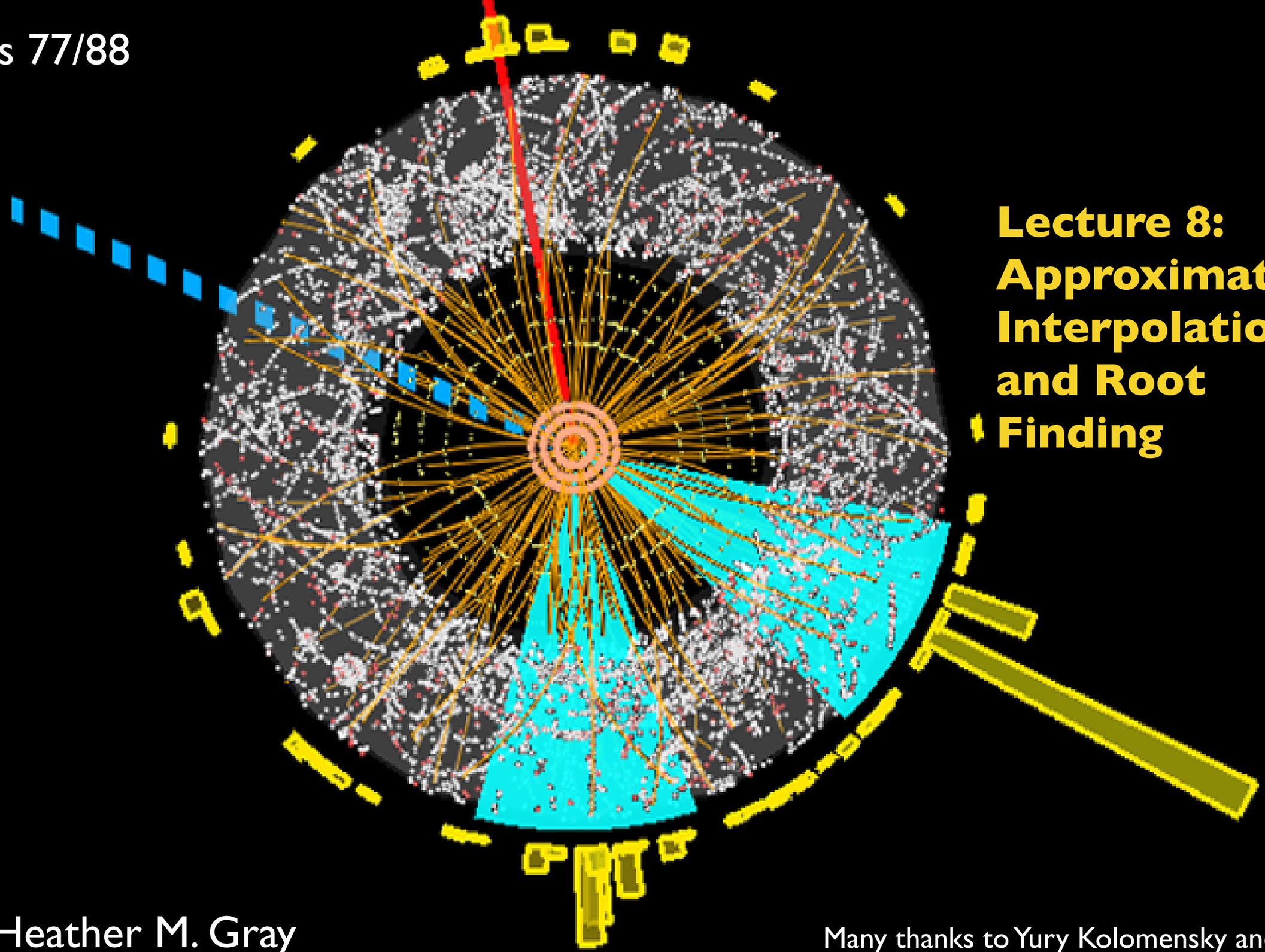
Introduction to Computational Techniques in Physics/Data Science Applications in Physics

Physics 77/88



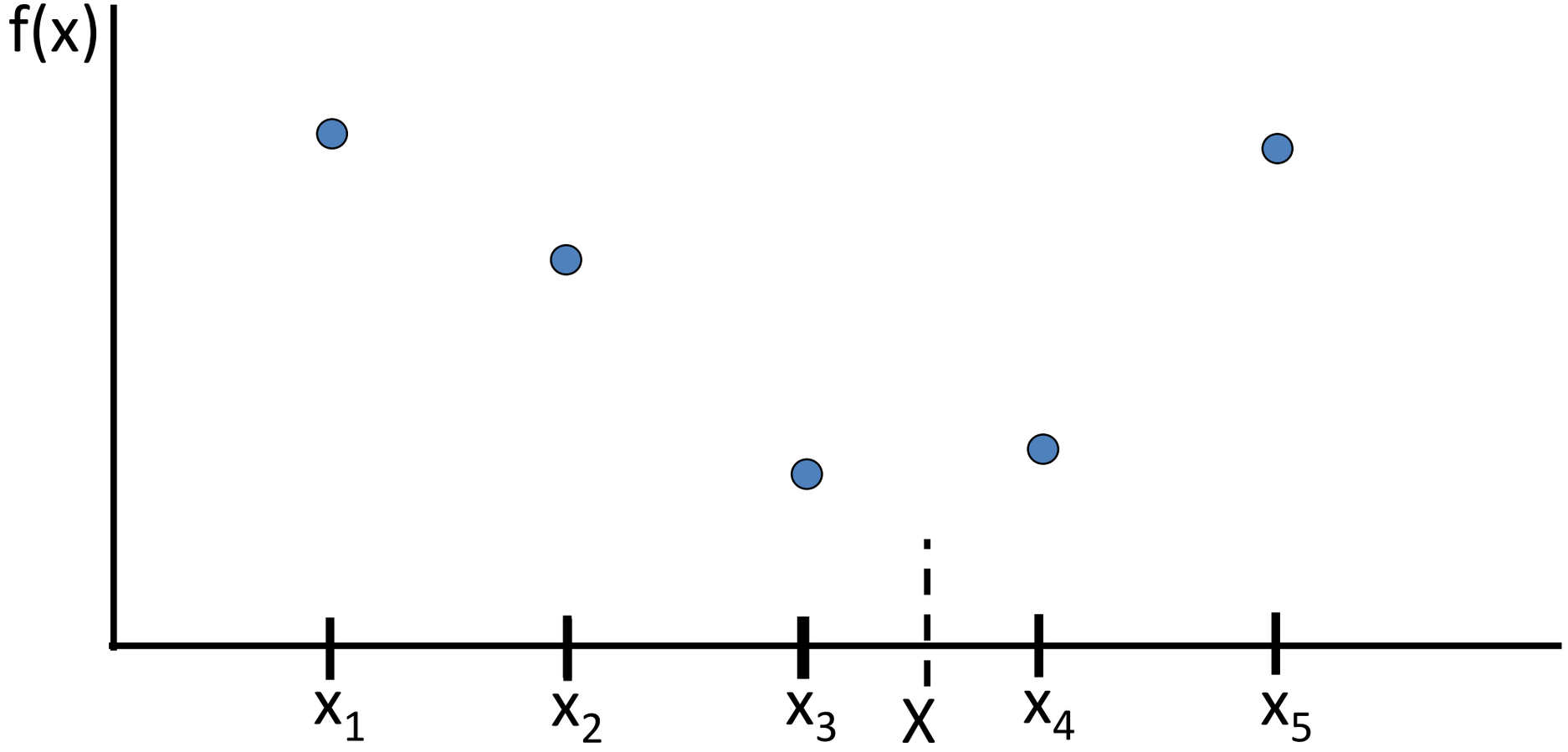
Lecture 8: Approximation: Interpolation and Root Finding

Prof. Heather M. Gray

Many thanks to Yury Kolomensky and Amin Jazaeri for material used in this course

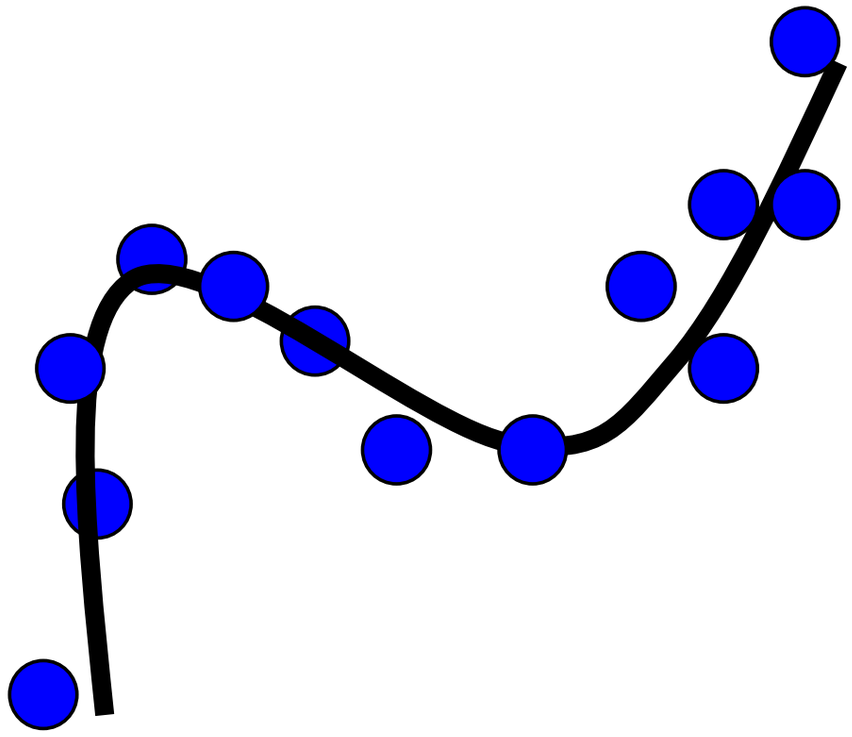
Approximations

- Suppose we have of , and we wish to values the known data points
  - This is a general problem of



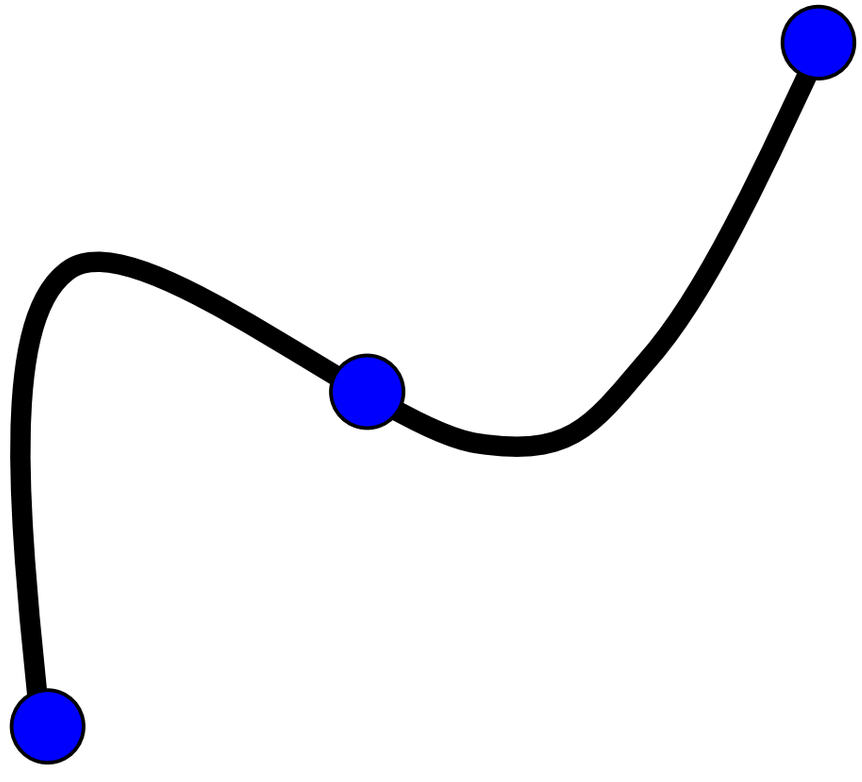
Interpolation vs Fitting

- We can to the data points, and that function the data points
  - e.g. a fit
  - Requires an (typically) form of the
  - Does that the curve would go through the points
- Appropriate when the data points have some



Interpolation vs Fitting

- Or we can require that the goes the data points: this is known as
  - Appropriate if the are known



Basis Functions

- choices of that can go through the data
  - Called
- Function families commonly used for interpolation
  -
  -
  -
  -
  -
- We will focus on interpolation using and

Existence and Uniqueness

- Existence and uniqueness of depends on number of and the number of

- If interpolant
- If : interpolant
- If : number of can be found
  - i.e. data can be
- A of basis functions is

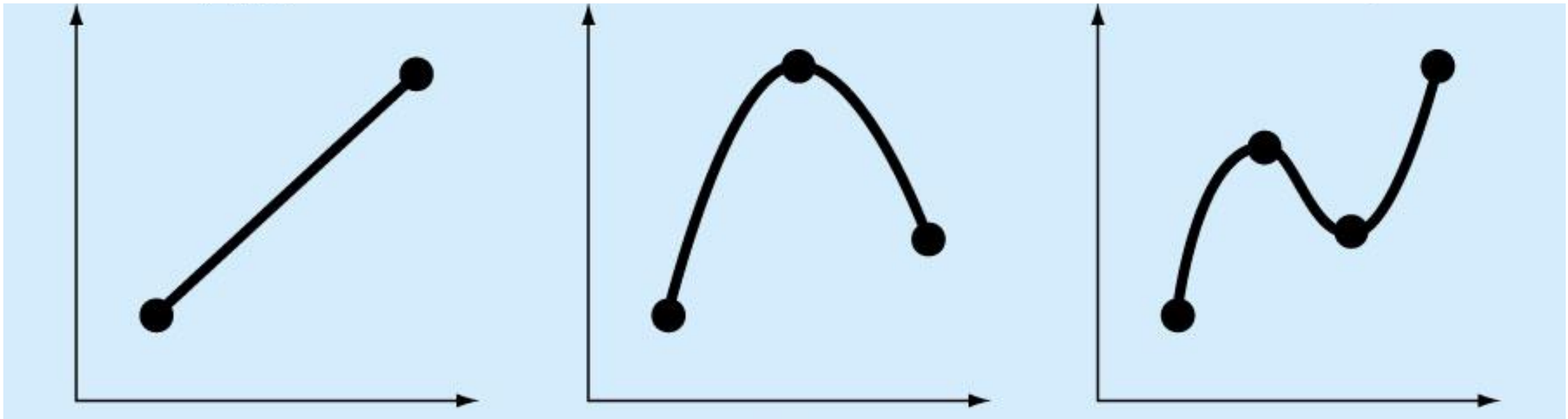
Polynomial Interpolation

- and type of interpolation uses

- Polynomial interpolation solves for an order polynomial that passes through
  - $f(x) =$

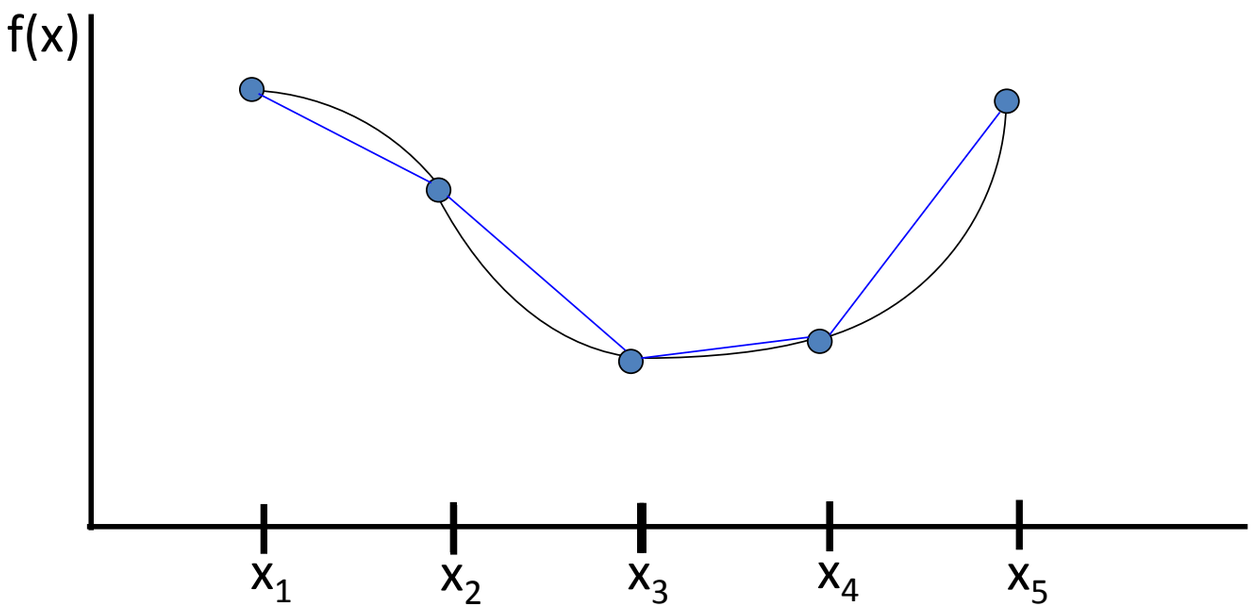
$$\bullet \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \left\{ \vdots \right\} = \left\{ \vdots \right\}$$

Choice of Polynomials



Global vs Piece-wise Interpolation

- An polynomial contains coefficients, so can go through points
- Alternatively, use a of polynomials, or , to
  - May require to be
- Splines are usually , this is an example of a interpolation



Linear Lagrange Interpolation

- Perform a Taylor's Series expansion of our function, $f(x)$ at two different points: and
  - $f(x_1) =$ $+ O(\Delta^2)$
  - $f(x_2) =$ $+ O(\Delta^2)$
- Define a first order polynomial $p(x)$ to approximate $f(x)$ by requiring $f(x_i) = p(x_i)$
  - $f(x_1) =$ $+ O(\Delta^2)$
  - $f(x_2) =$ $+ O(\Delta^2)$
- Note that $p'(x)$ is

Linear Lagrange Interpolation

- Multiply top equation by $(x - x_2)$ and bottom equation by $(x - x_1)$
  - $f(x_1)(x_2 - x) = p(x)(x_2 - x) + (x_1 - x)(x_2 - x)p'(x)$
  - $f(x_2)(x_1 - x) = p(x)(x_x - x) + (x_1 - x)(x_2 - x)p'(x)$
- Subtract bottom from top

$$
\begin{aligned}
f(x_1)(x_2 - x) - f(x_2)(x_1 - x) &= p(x)(x_2 - x) - p(x)(x_1 - x) \\
&= p(x)[(x_2 - x) - (x_1 - x)] \\
&= p(x)(x_2 - x_1)
\end{aligned}
$$

Linear Lagrange Interpolation

- Solve for $p(x)$:

$$p(x) = \frac{f(x_1)(x_2 - x)}{x_2 - x_1} - \frac{f(x_2)(x_1 - x)}{x_2 - x_1}$$

$$= \frac{f(x_1)(x_2 - x)}{x_2 - x_1} + \frac{f(x_2)(x_1 - x)}{x_1 - x_2}$$

$$= f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

Quadratic Lagrange Interpolation

- Taylor expand again, but assume the function is a second order polynomial, $p(x)$
  - $f(x_1) = p(x) + (x_1 - x)p'(x) + (x_1 - x)^2 p''(x)/2$
  - $f(x_2) = p(x) + (x_2 - x)p'(x) + (x_2 - x)^2 p''(x)/2$
  - $f(x_3) = p(x) + (x_3 - x)p'(x) + (x_3 - x)^2 p''(x)/2$

- After some more (lengthy) algebra to eliminate $p'(x)$ and $p''(x)$ to yield
  - $$p(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} f(x_1) + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} f(x_2) + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} f(x_3)$$

Lagrange Polynomials

- In general

$$P(x) = y_1 \frac{(x - x_2)(x - x_3)\cdots(x - x_n)}{(x_1 - x_2)(x_1 - x_3)\cdots(x_1 - x_n)}$$
$$+ y_2 \frac{(x - x_1)(x - x_3)\cdots(x - x_n)}{(x_2 - x_1)(x_2 - x_3)\cdots(x_2 - x_n)} + \cdots$$
$$+ y_n \frac{(x - x_1)(x - x_2)\cdots(x - x_{n-1})}{x_n - x_1)(x_n - x_2)\cdots(x_n - x_{n-1})}$$

Disadvantages

- Although the of $P_n(x)$ is the method is for
- If and the data for , the efficiency can be improved by considering in the of the for which $P_n(x)$ is sought
- However, this can degrade the approximation of

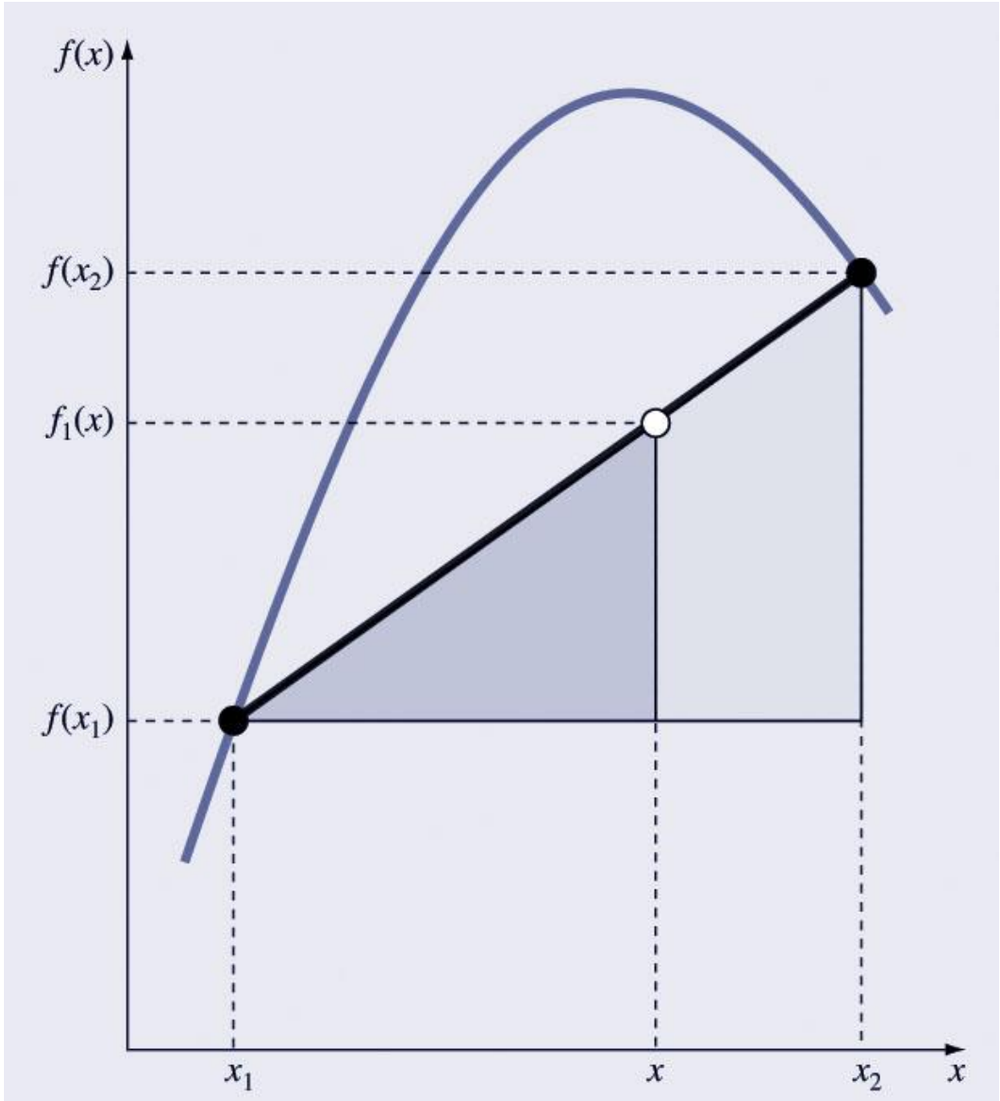
Newton's Interpolating Polynomials

- In general, points can be fitted by an polynomial of the form
- $f_n(x) =$ $+$ $+ \cdots +$
- Coefficients are evaluated as follows
  - $b_0 =$ $= f[x_0]$
  - $b_1 =$ $= f[x_1, x_0]$
  - $b_2 =$ $= f[x_2, x_1, x_0]$
  - $b_n = f[x_n, x_{n-1}, \cdots, x_1, x_0]$

First Order Newton Interpolating Polynomial

- polynomial is obtained from and

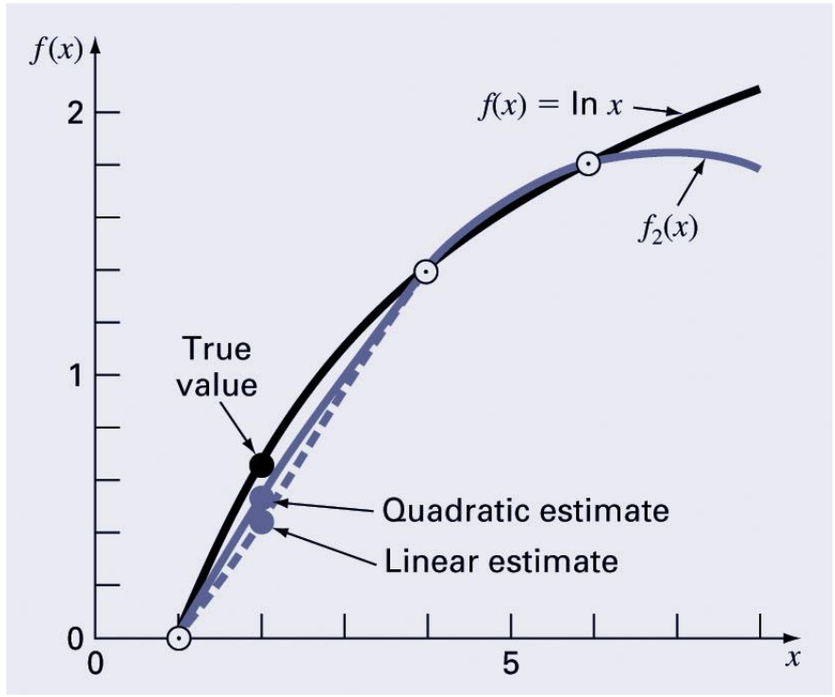
- Formula depends on known points and and and



$$f_1(x) = .$$

2nd Order Newton Interpolating Polynomial

- Newton interpolating polynomial goes the points but introduces
- Formula depends on known points and and , and



$$f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1) + \frac{\frac{f(x_3) - f(x_2)}{x_3 - x_2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_3 - x_1}(x - x_1)(x - x_2)$$

Piecewise Polynomials

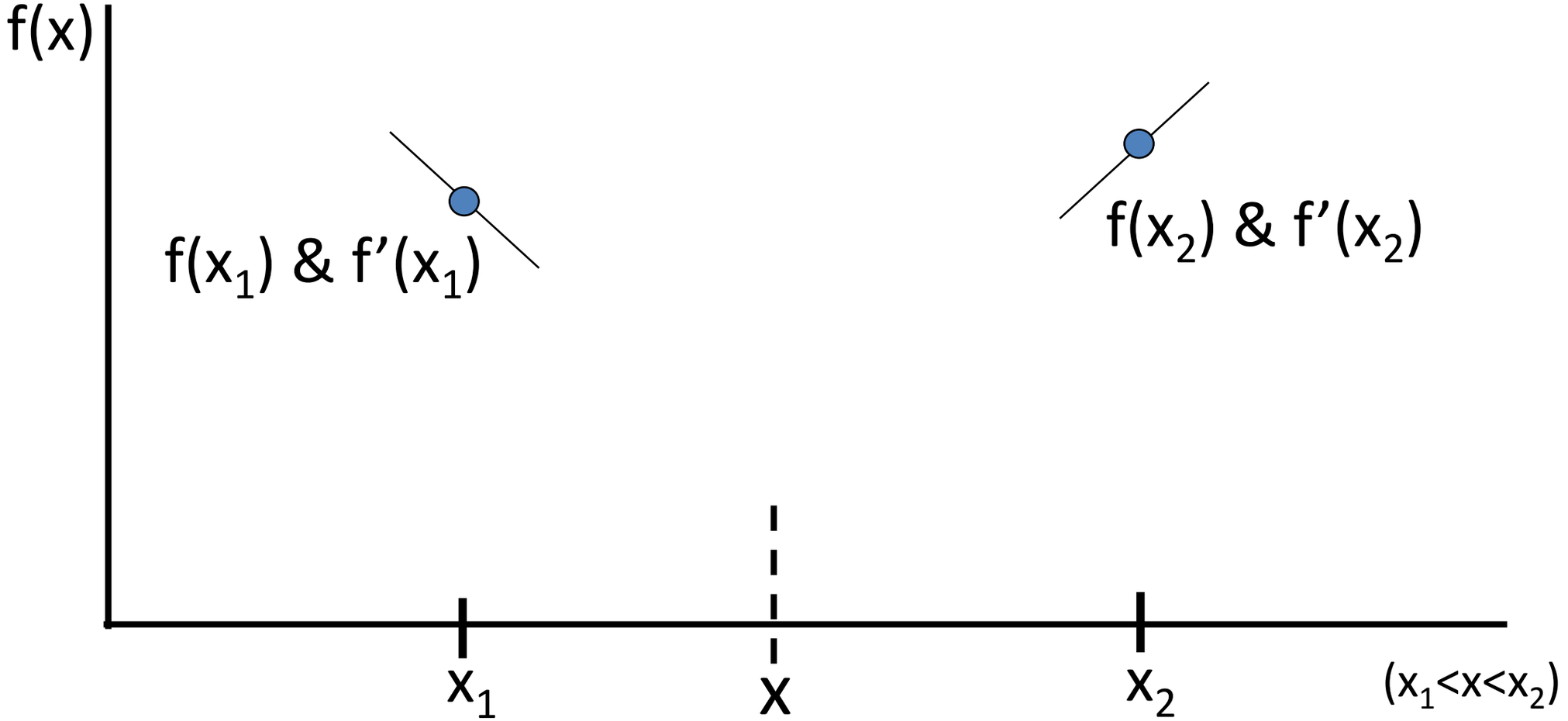
- Fitting a to many data points can yield an
- polynomials are a alternative to polynomials
  - Fit data points with polynomials
  - functions in subinterval
- Abscissas x_i are called or
  - Where interpolant changes from to

Piecewise Polynomials

- interpolation
  - Connect of data points with
- Piecewise interpolation eliminates and
  - Interpolating function is
- Many in choosing polynomial interpolant
  - Can be used to obtain interpolating function

Hermite Interpolation

- If we have the and at
  we can fully constrain a polynomial



Hermite Cubic Interpolation

- Given the general form for a cubic polynomial
  - $p(x) =$
- And its derivative
  - $p'(x) =$
- Two points
  - $f(x_1) = ax_1^3 + bx_1^2 + cx_1 + d$
  - $f(x_2) = ax_2^3 + bx_2^2 + cx_2 + d$
- Derivatives
  - $f'(x_1) = 3ax_1^2 + 2bx_1 + c$
  - $f'(x_2) = 3ax_2^2 + 2bx_2 + c$

Hermite Polynomials

- Can again solve for $p(x)$

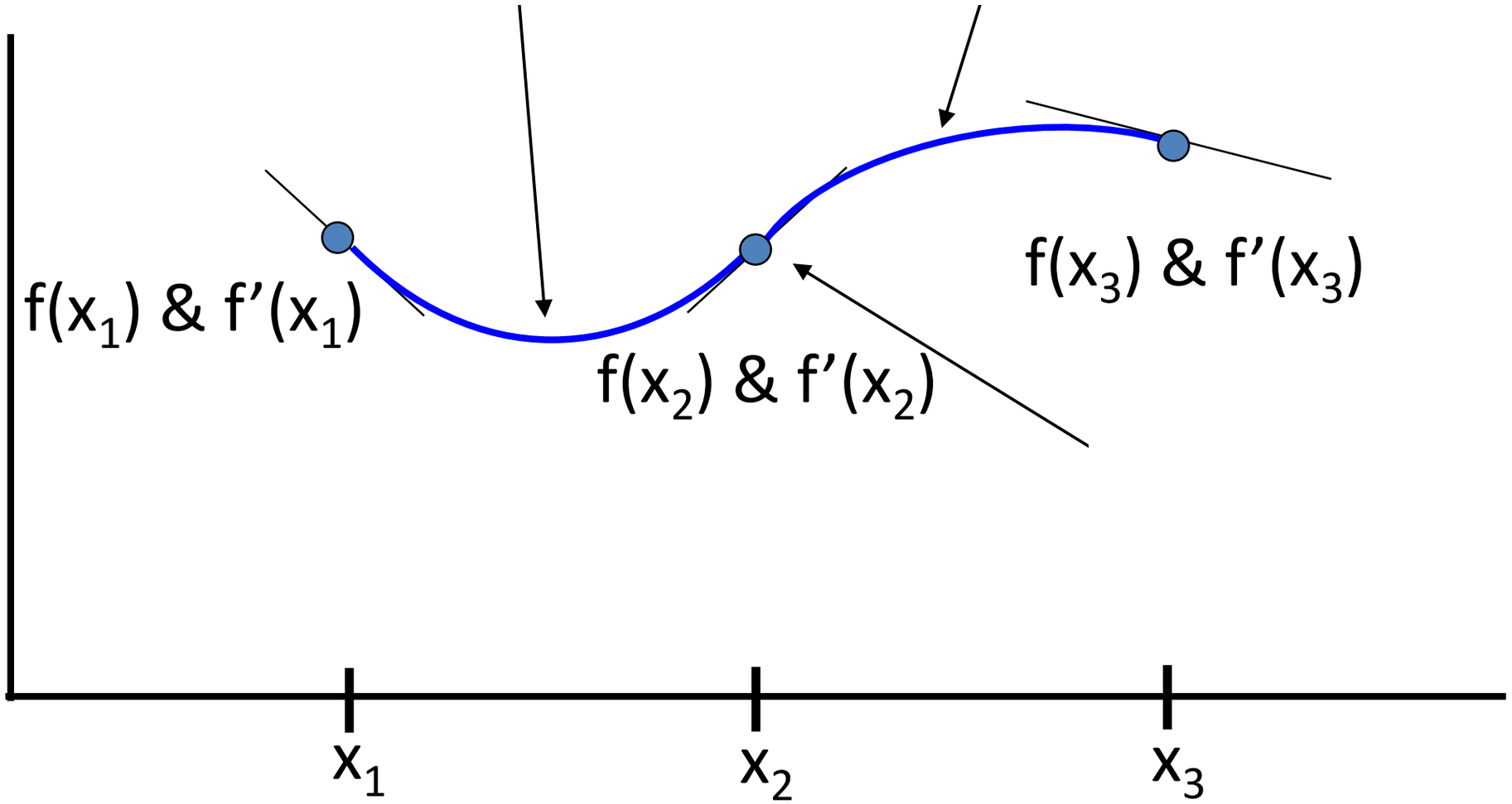
$$p_{1,2}(x) = \frac{(3x_1 - 2x - x_2)(x - x_2)^2}{(x_1 - x_2)^3} f(x_1) + \frac{(3x_2 - 2x - x_1)(x - x_1)^2}{(x_3 - x_1)^3} f(x_3)$$

- $$+ \frac{(x - x_1)(x - x_2)^2}{(x_1 - x_2)^2} f'(x_1) + \frac{(x - x_2)(x - x_1)^2}{(x_2 - x_1)^2} f'(x_2)$$

- If we fit a between and
  it will have the same derivative at x_2 as $p_{1,2}(x)$

- Interpolation between is
- are also continuous
  - interpolation is smooth

Hermite Polynomials at Boundaries



Hermite vs Lagrange Interpolation

- A series of polynomials and their are continuous at the points
  - i.e.
- A series of interpolating polynomials is at the points but their are
- Hermite interpolating polynomials can fit to : $x_1 < x < x_2$
- Lagrange interpolating polynomials require to fit a cubic: $x_1 < x_2 < x < x_3 < x_4$

Cubic Spline Interpolation

- If at each point are , can still require that the derivatives are , up to a
  - i.e. require that the derivatives of the polynomial to the and of each data point are the
  - Cubic spline: for each pair of points, a degree polynomial has unknown coefficients
    - unknowns
- Constraints:
    - function values
    - equations for 1st and 2nd derivative
    - derivatives at the

Hermite vs Cubic Spline

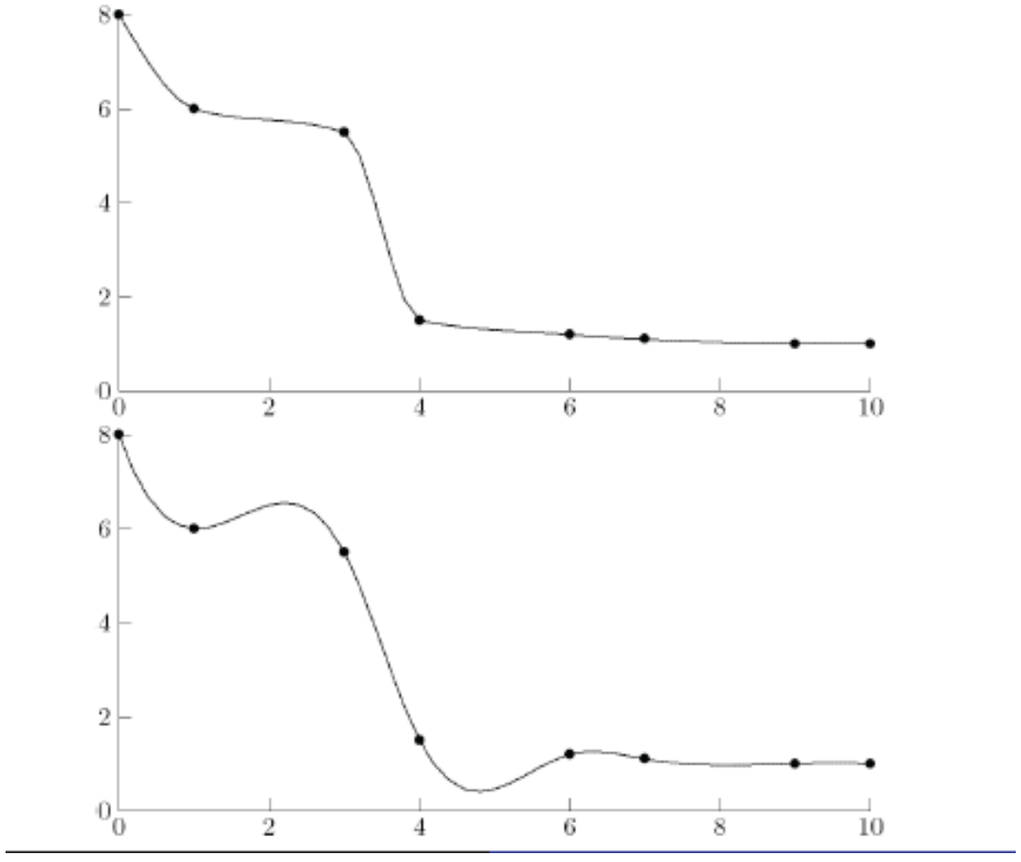
- interpolant is a piecewise cubic polynomial with a derivative
  - Piecewise cubic polynomial with has parameters to be determined
  - Interpolates given data: equations
  - One continuous derivative: additional equations
  - Total of equations and free parameters
- Hermite cubic interpolant is and remaining free parameters can be chosen to satisfy

Hermite vs Cubic Spline

- is a piecewise polynomial of degree that is times continuously differentiable
  - spline is of degree and has continuous derivatives
    - but : line
      - spline is a piecewise cubic polynomial that is differentiable
        - Like Hermite cubic, interpolating given data and requiring one continuous derivatives imposes constraints
        - Requiring continuous second derivative imposes additional constraints: free parameters

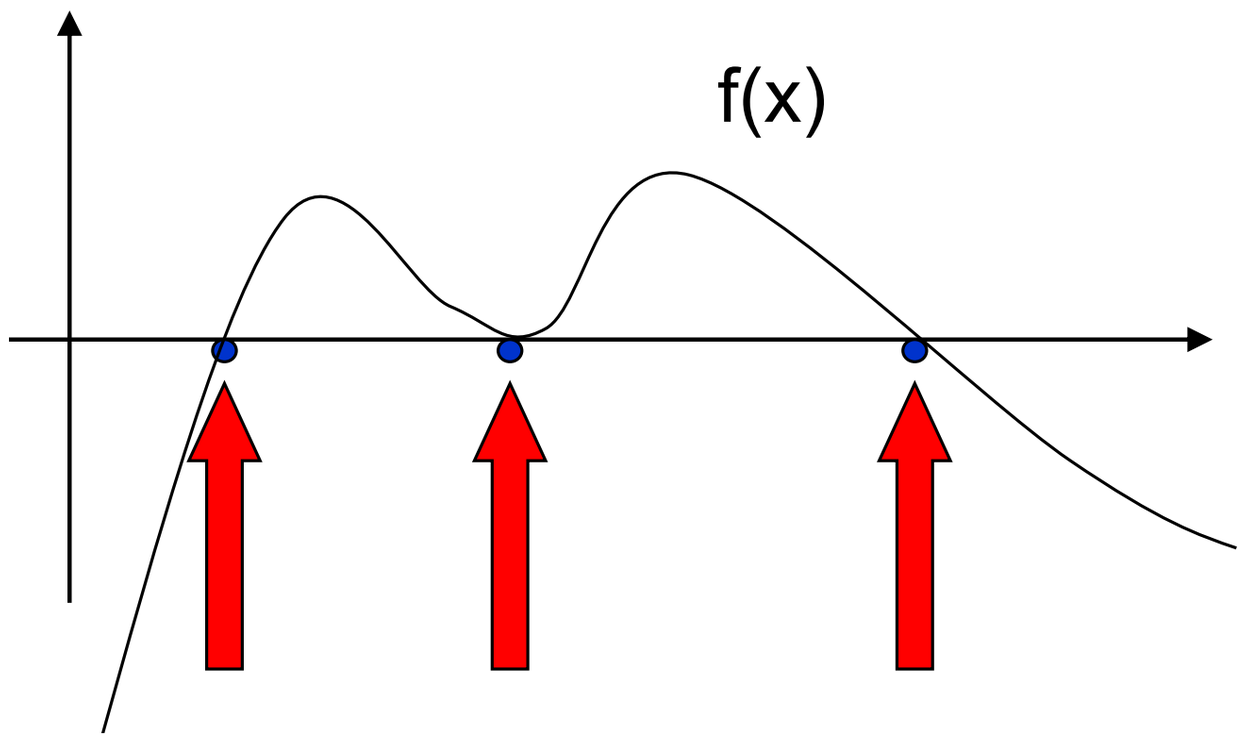
Hermite vs Spline

- Choice depends on to be
  and of the
  interpolation
- :
  interpolation
- polynomial is more
  and preserves
  of data
- Rule of thumb: plot
  and to assess of
  interpolating function



Root Finding

- Many problems in physics require finding of a function:
- of any function $f(x)$ are the where the graph of the function the



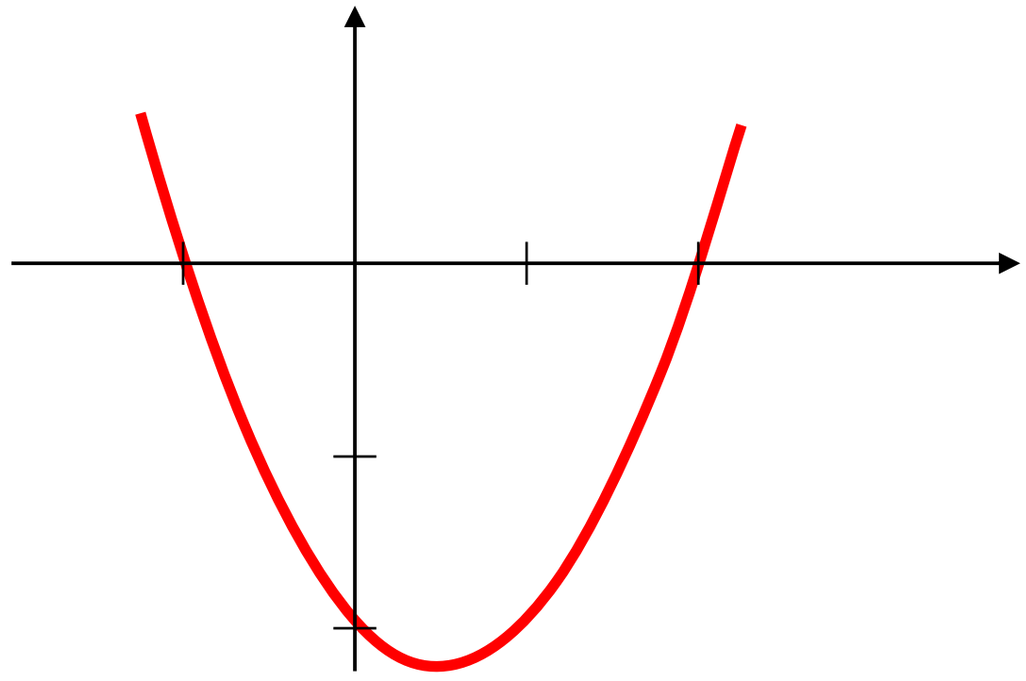
Types of Roots

- zeros

$f(x) =$

- $=$

- simple zeros
  - $x =$
  - $x =$



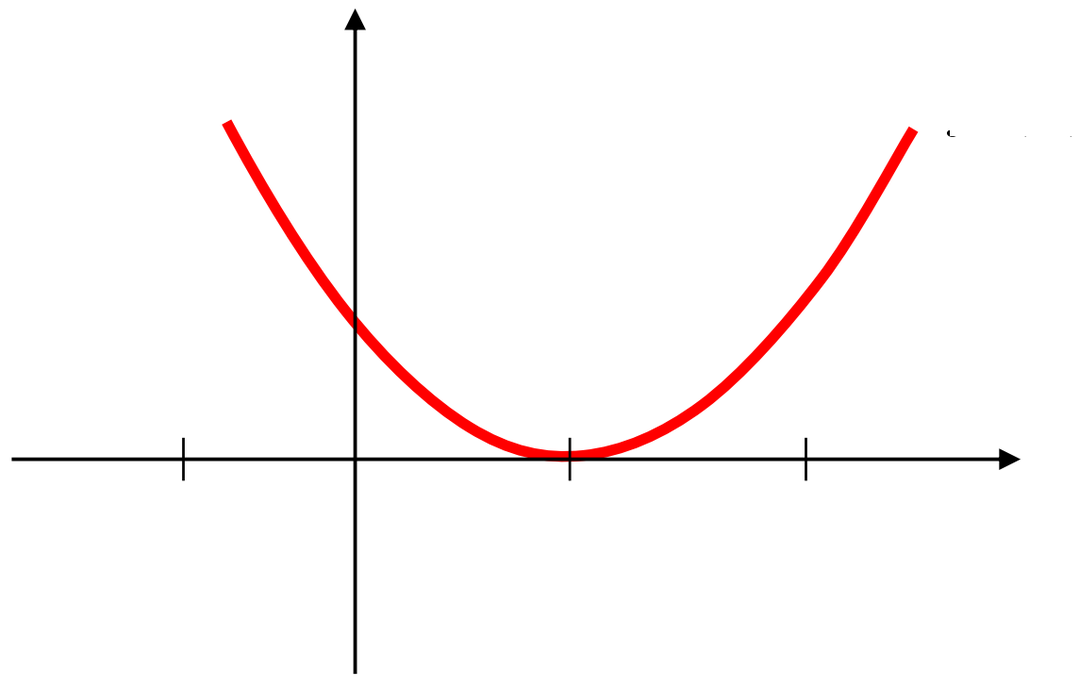
Types of Roots

- zeros

$f(x) =$

- $=$

- zeros at
  - $x =$
- Or zero with multiplicity

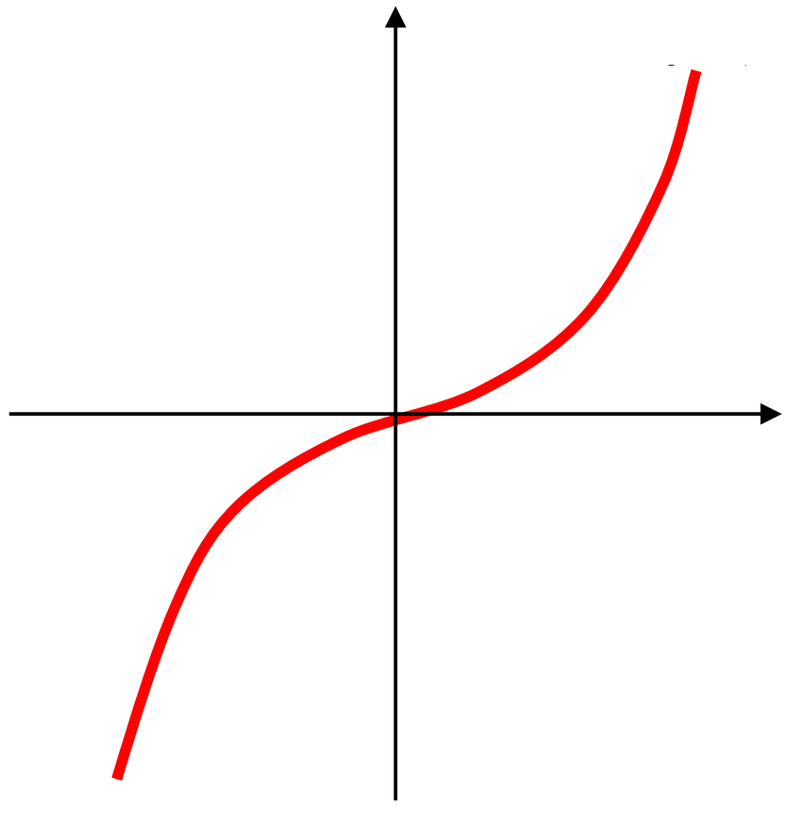


Type of Roots

- zeros

- $f(x) =$

- Zero with multiplicity at $x =$

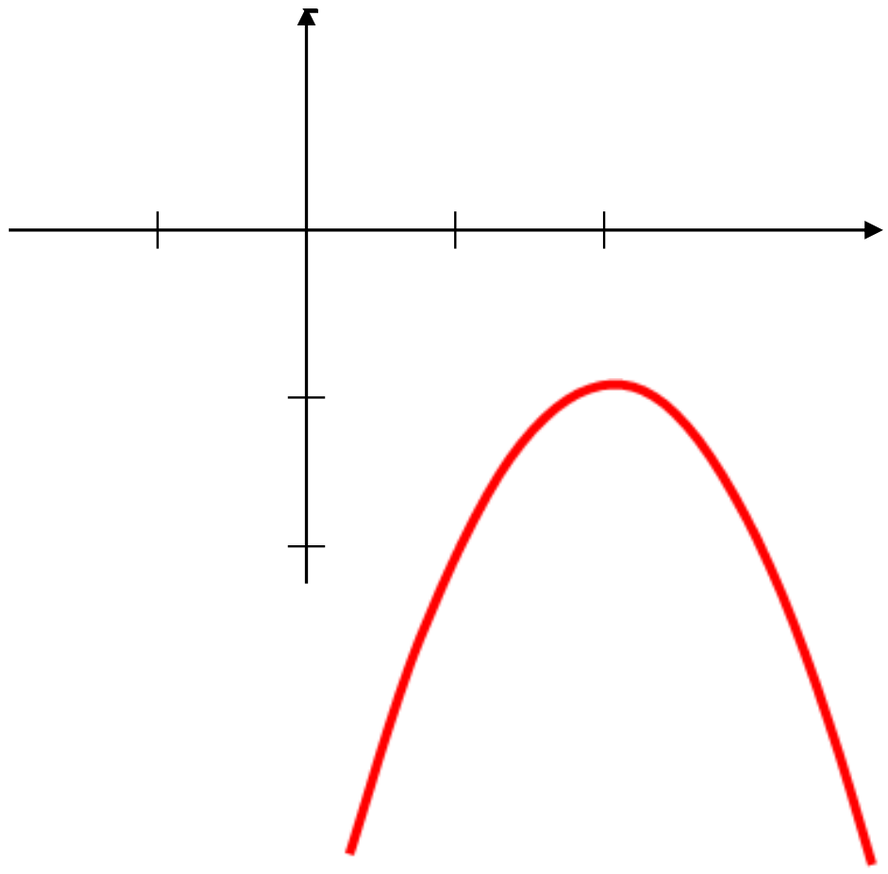


Types of Roots

- zeros

- $f(x) =$

- zeros



Root of a Polynomial

- Consider a polynomial
  - $f_n(x) = \Sigma_{i=0}^{n}$ with $a_n = 1$
- $n = 1$
  - $f_1(x) =$ is the only root
- $n = 2$
  - $f_2(x) = a_0 + a_1 x + x^2; x =$ ——————————
- $n = 3$
  - $f_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3; x =$

See http://mathworld.wolfram.com/CubicFormula.html

Polynomials

- Factored form
  - $P(x) = a_n$
- roots
- parameters
- Both and roots
- No for polynomials of or

Facts about Polynomials

- Any order polynomial has exactly
  - Includes both and zeros and their

- Any polynomial with an has at least zero

- If a function has a zero at with multiplicity
  - Function and its first derivatives are at and the derivative at r is

Root Finding

- In general, if an to the equation does not exist, can find the solution approximately using techniques
  - method
  - methods
  - method

Iterative (Relaxation) Method

- Example:
  - Take your (engineering) calculator (or app), switch angles to radians, type in any number, press "cos", and keep pressing until the number on the screen stops changing.
- What did you find ?

Iterative (Relaxation) Method

- If the function is of the form
  - $f(x) =$
- Finding root $f(x) = 0$ can be done iteratively:
  - Pick
  - Compute $x_1 =$
  - $x_n =$
- to the solution
  - Or

Bracketing Methods

- In methods, start with an that the root
- is applied to obtain a the root
- Examples of bracketing methods
  - method
  - method

Open methods

- Start with one or more
- Obtain a of the root for
- Typically than bracketing methods
- May to the root

Iterative Root Finding Methods

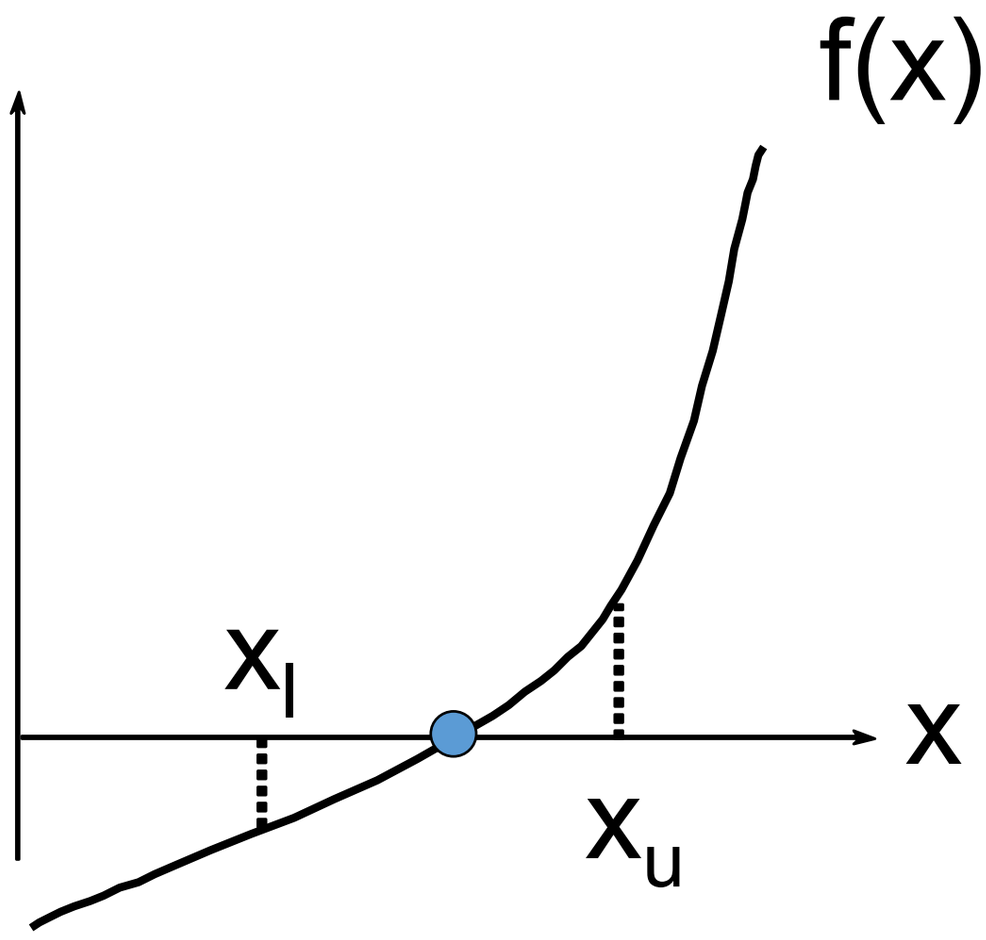
- Numerical algorithms take any $f(x)$ and search for solutions
  - No for all functions $f(x)$
  - Some for certain $f(x)$ but for others
    - Certain $f(x)$ may solutions
- Warning: don't use algorithms as
  - Know the of each algorithm

Iterative Root Finding Methods

- Bisection
- Newton-Raphson
- Secant Method
- Müller-Brent Method

Bracketing and Bisection

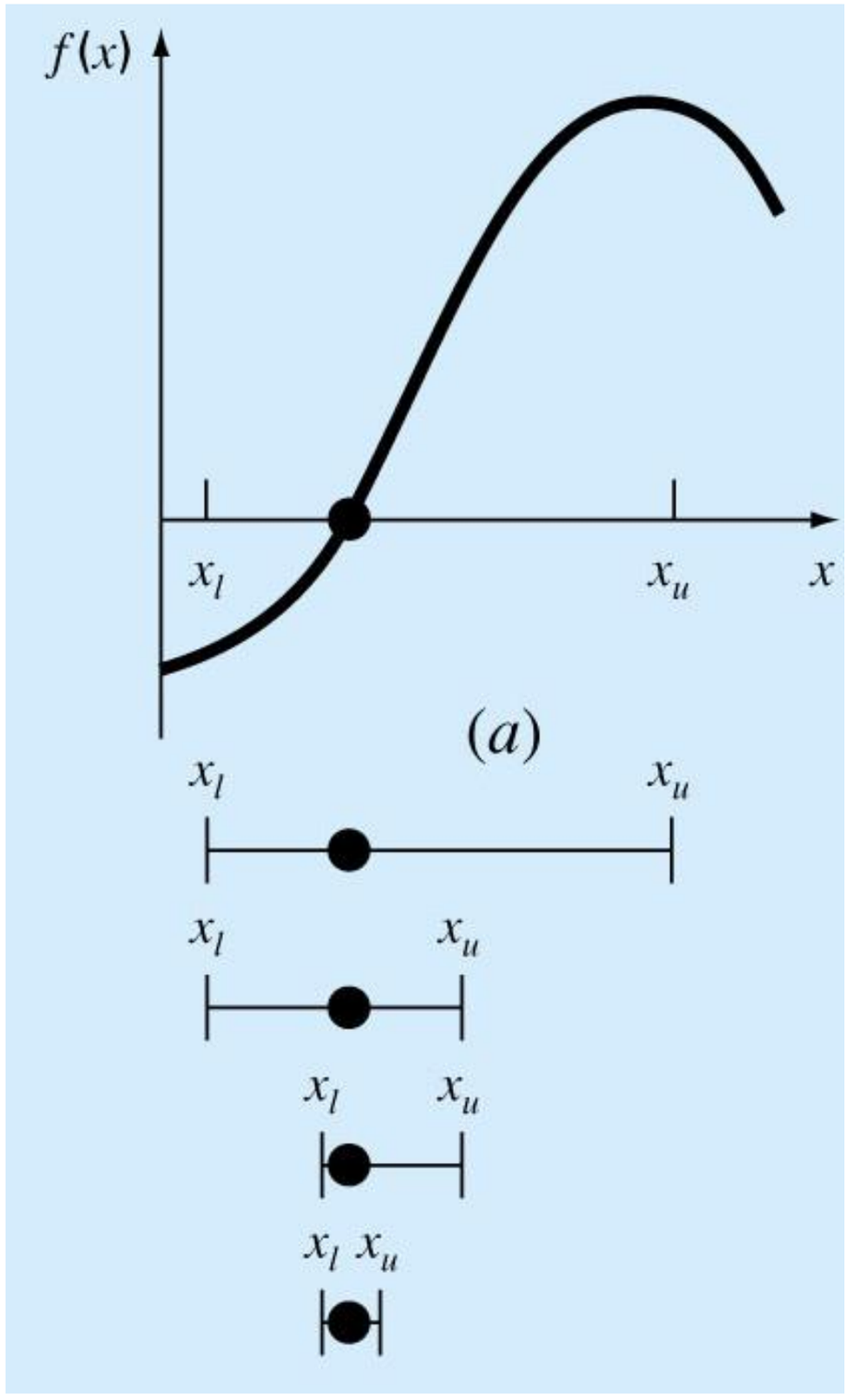
- Given a bounded by and such that the of the function $f(x)$ between $f(x_l)$ and $f(x_u)$ there must be at least



- Assumes function is and doesn't have

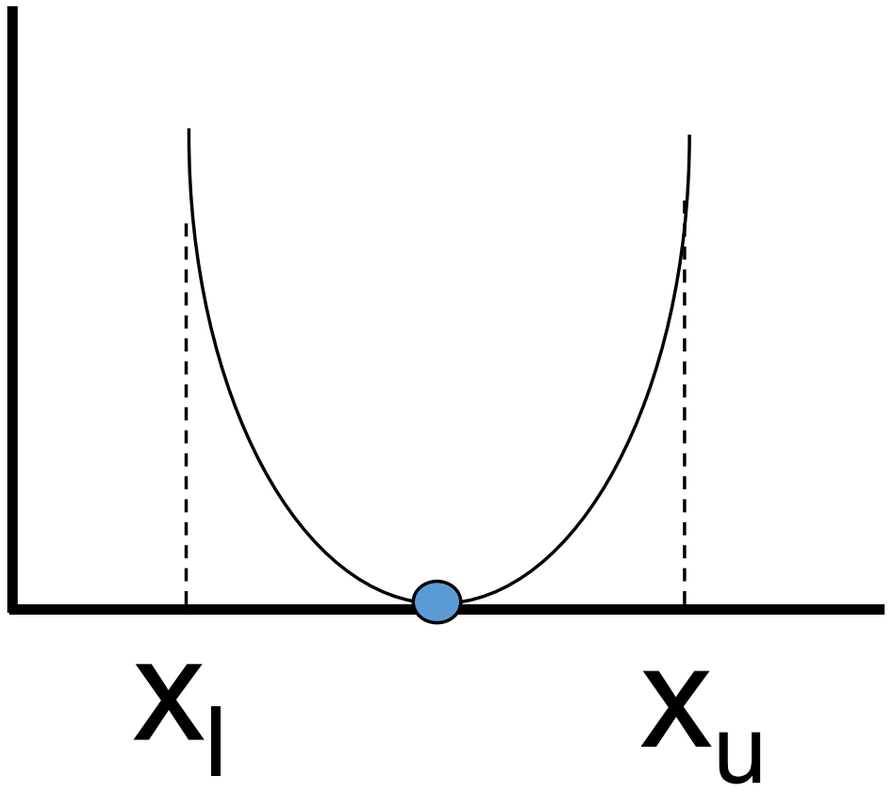
- Root is said to be by the interval

Bracketing Method

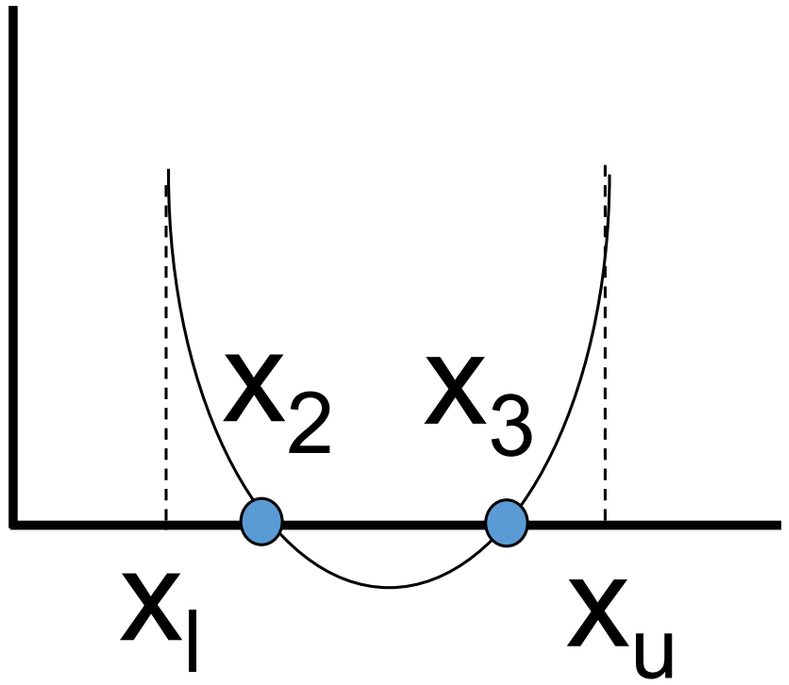


Other Possibilities

- Even if a is bracketed by two values, a root

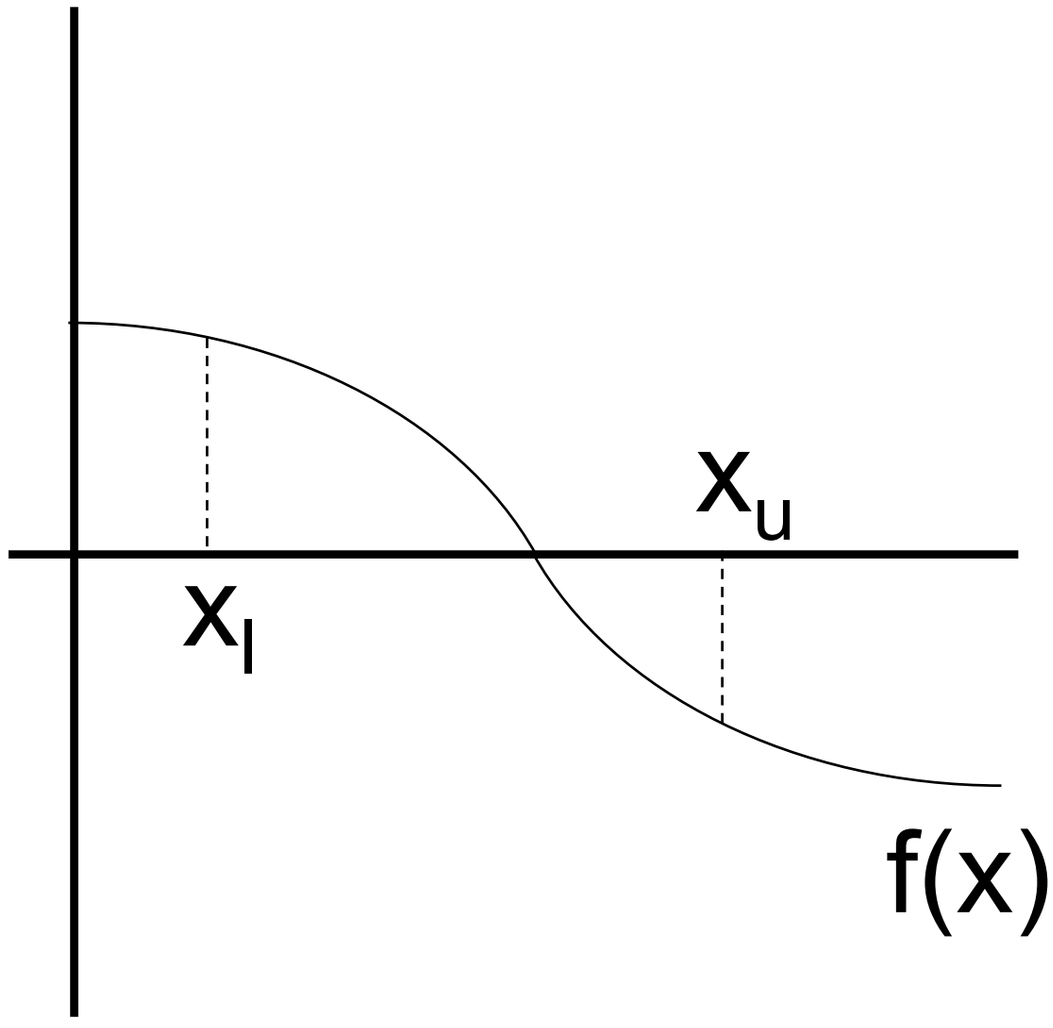


- There might even be roots in this region for functions with



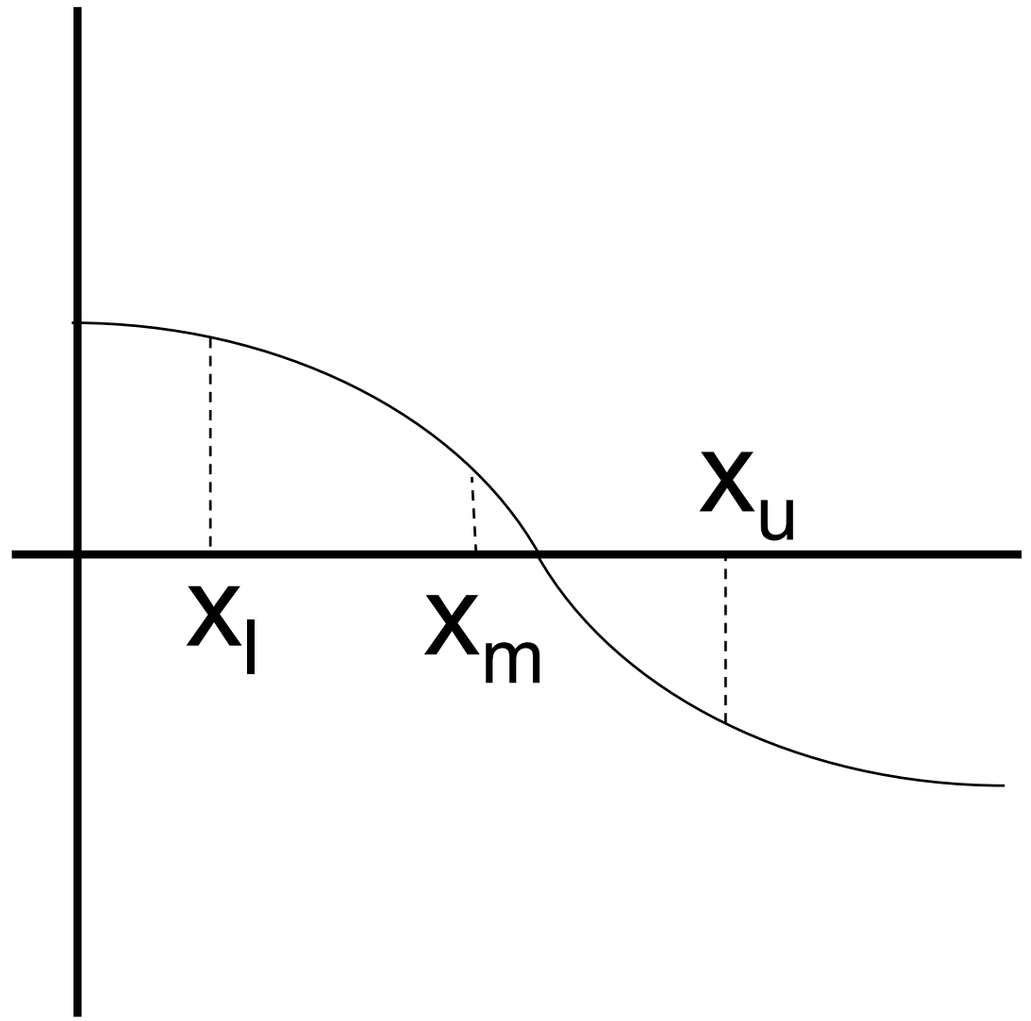
Root Finding by Bisection

- Step 1: Given that we that there an $f(x_0) = 0$ solution, choose x_l and x_u as the for the root
  - Because $f(x_l)$ and $f(x_u)$ must have opposite signs, we have: $f(x_l) \times f(x_u)$
- e.g. $f(x) =$
- $\Rightarrow \quad < x_0 <$



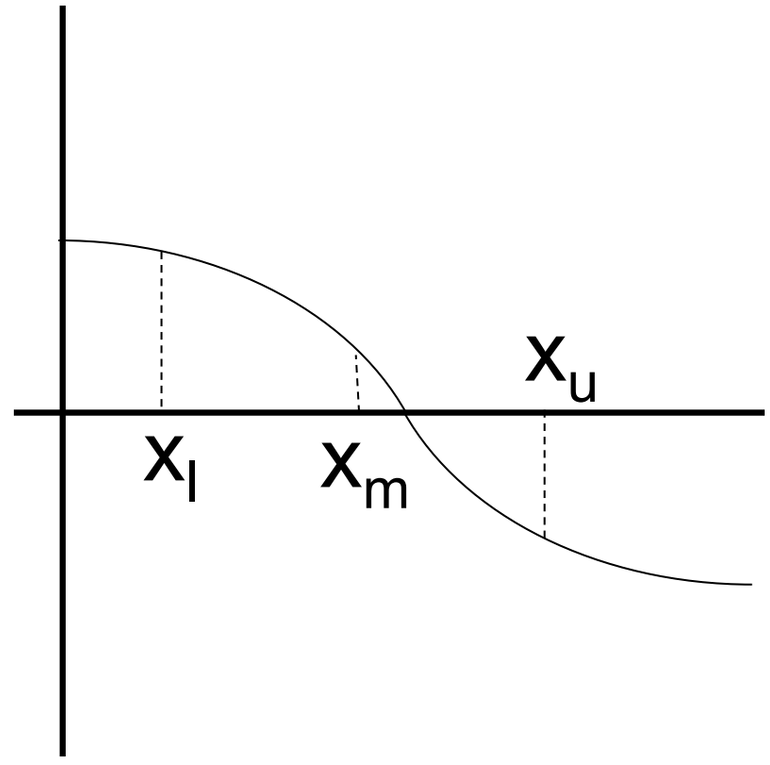
Bisection: Find the Mid-Point

- Step 2: Let
  $x_m =$
- x_m is the between the two
- It must be to the than one of x_l and x_u
- Next step is to select the from x_l, x_m and x_u
- Particular case: is the same as



Bisection: Find New Bracket

- Step 3
  - If $f(x_l)f(x_m)$
    - Root lies between and
    - $x_l = x_l; x_u = x_m$
  - If $f(x_l)f(x_m)$
    - Root lies between and
    - $x_l =$ $; x_u =$
- If $f(x_l)f(x_m) =$
  - Root is ; stop



Final Step

- Step 4: Test
  - x_m is the at this stage
  - Absolute error is but we don't know
  - Error estimate: where n corresponds to the iteration
- Check if where is the that you decided at the
  - If stop
  - If return to step 1

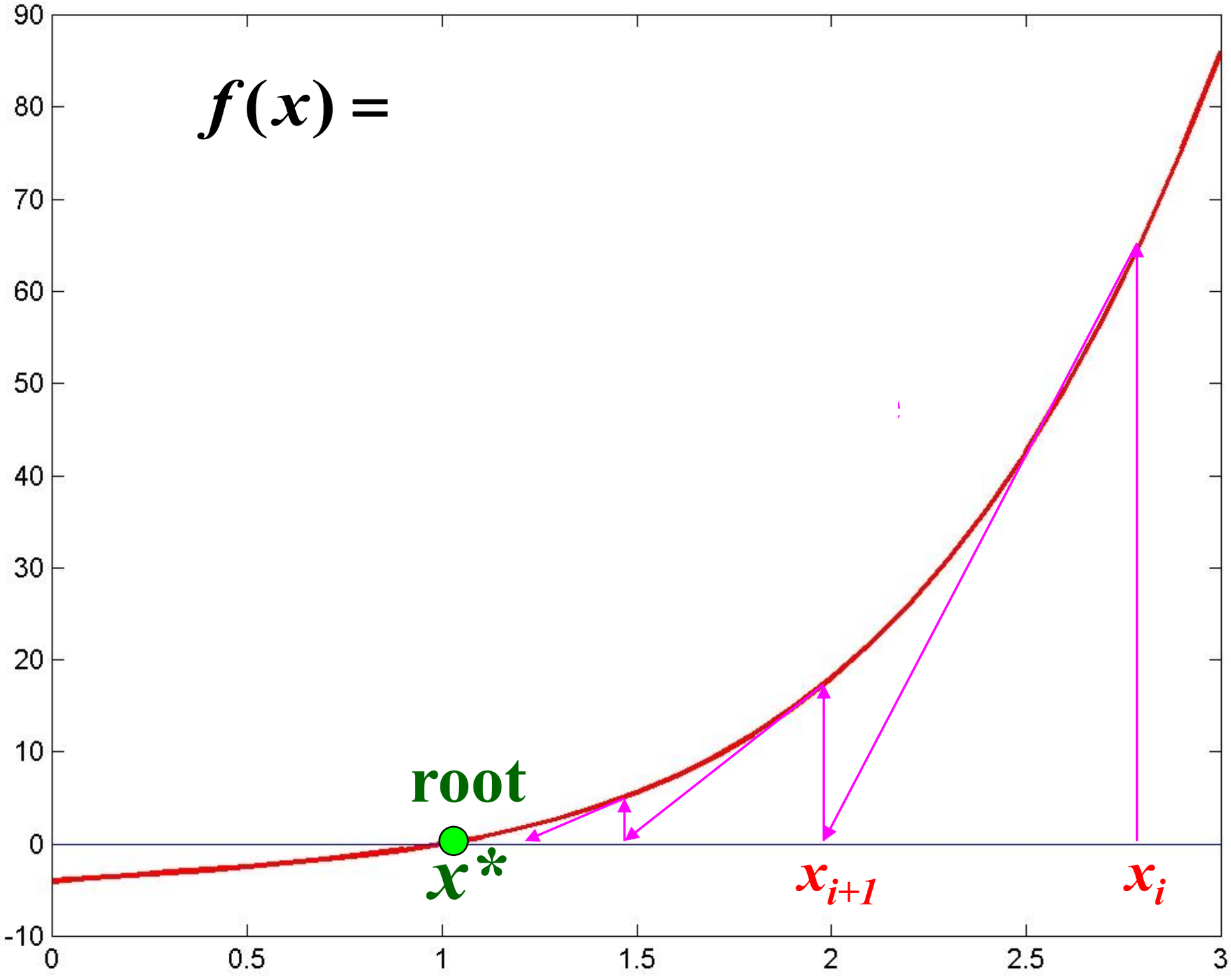
Pros and Cons of Bisection

- Pros:
  - Bisection is
    - Will find a root
  - Can be programmed
- Cons
  - Bisection converges
    - Width of bracket: $x_u^n - x_l^n = w^n = w^{n-1}/2$
      - Width at each stage
      - Know in advance that the width of the bracket after will have width
      - Will achieve tolerance when

Newton-Raphson Method

- For many , underlying algorithms begin with a
- Taylor expansion of $f(x)$ around a point x_0
  - $f(x) \approx$ …
- If x is a root
  - $0 \approx f(x_0) + (x - x_0)f'(x_0)$
  - $\Rightarrow x \approx$
- Formula applies as long as is not an

Newton-Raphson Method

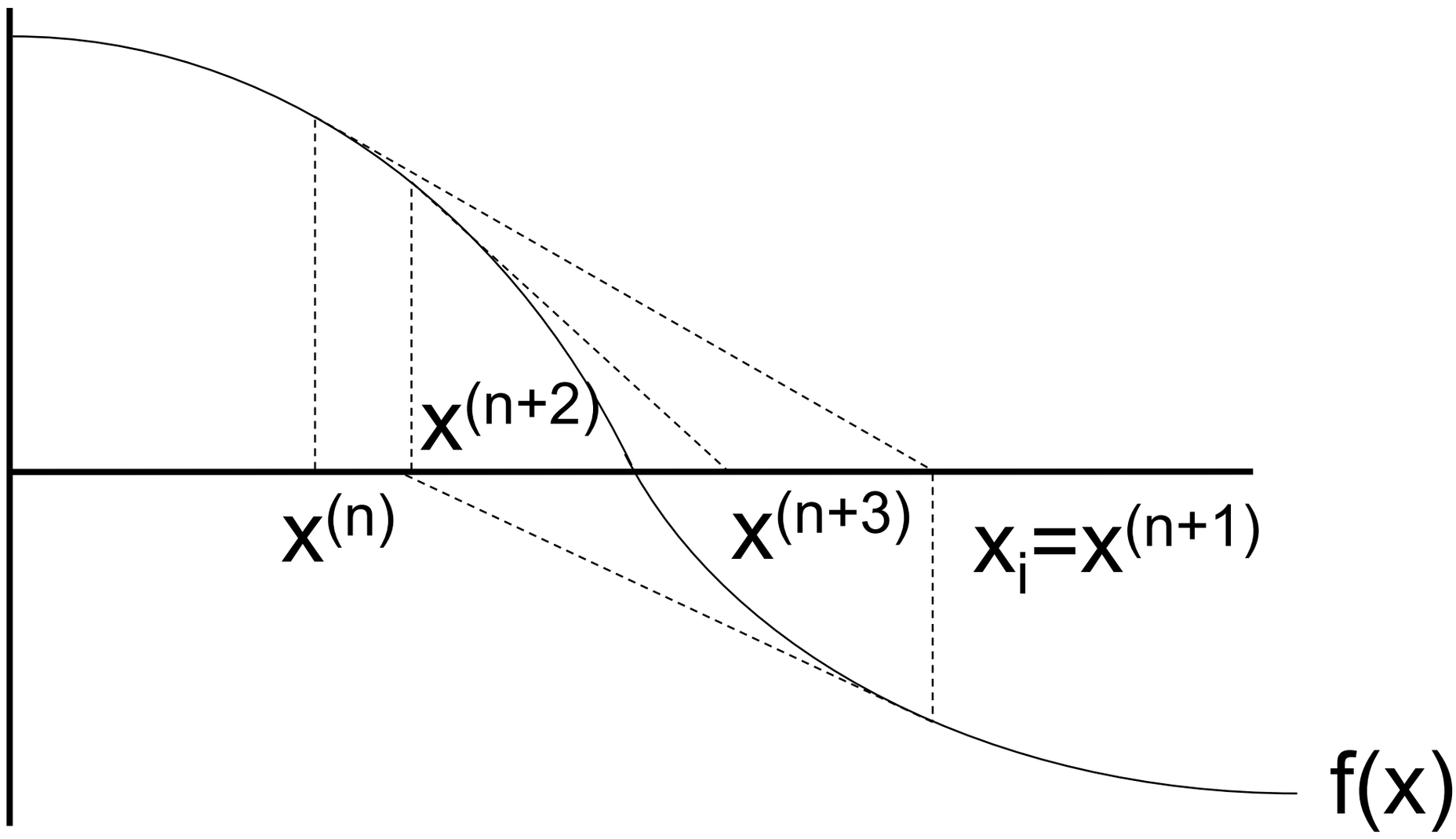


Newton Raphson Method

- Step 1: Start at the point
- Step 2: Calculate interaction of at this point and the x-axis
  - $x_2 =$
- Step 3: Check if or (tolerance)
- Step 4:
  - If , solution
  - If , repeat the

Geometric Interpretation

- General step in the iteration is
  - $x^{(n+1)} \approx$



Newton-Raphson's Method

- Note than an of the (slope) is required
- You may have to do this
- method: convergence depends on the initial guess and it not guaranteed
- However, Newton's method can
  - convergence

Convergence

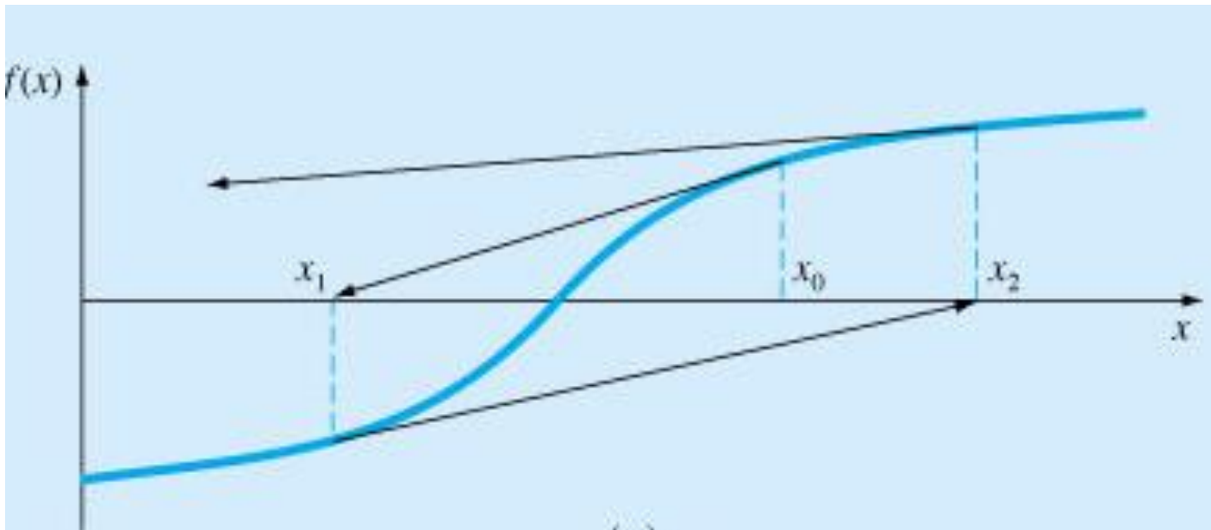
- Necessary and sufficient condition
  - $\left| \quad \right| < 1$
- Error term of:
  - $E(x) =$
- Doesn't converge if has roots

Newton-Raphson Method

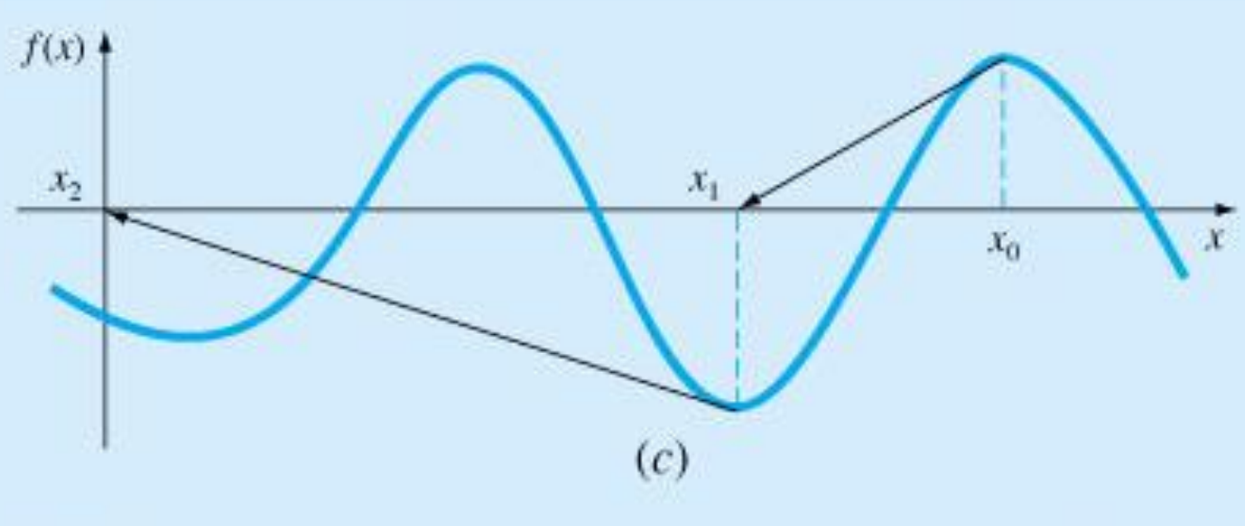
- Although Newton-Raphson converges it may
  and fail to find roots if
  - There are multiple roots
  - An , is near the root
  - There is a minimum or maximum:
  - A slope is reached

Convergence is for methods

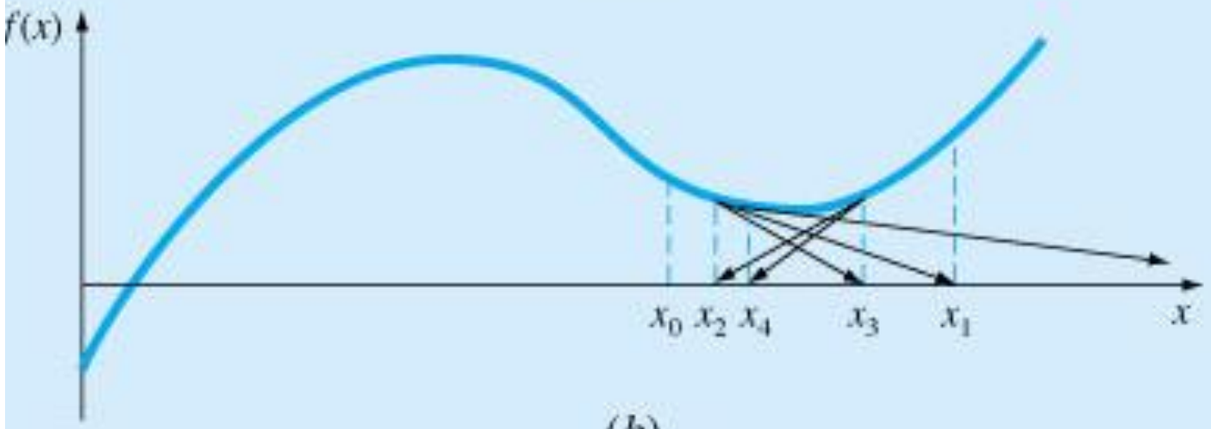
Poor convergence for Newton-Raphson



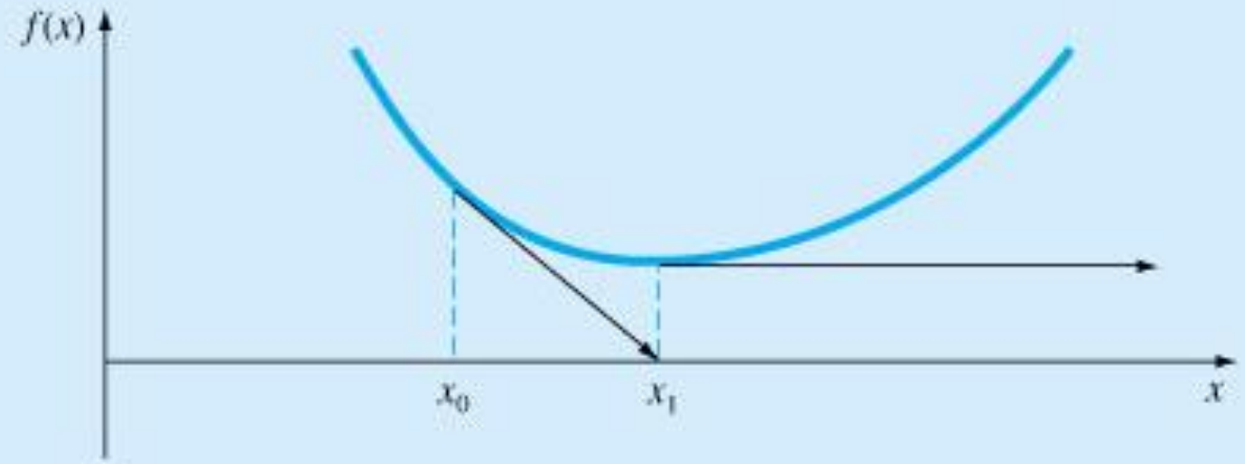
(a)



(c)



(b)



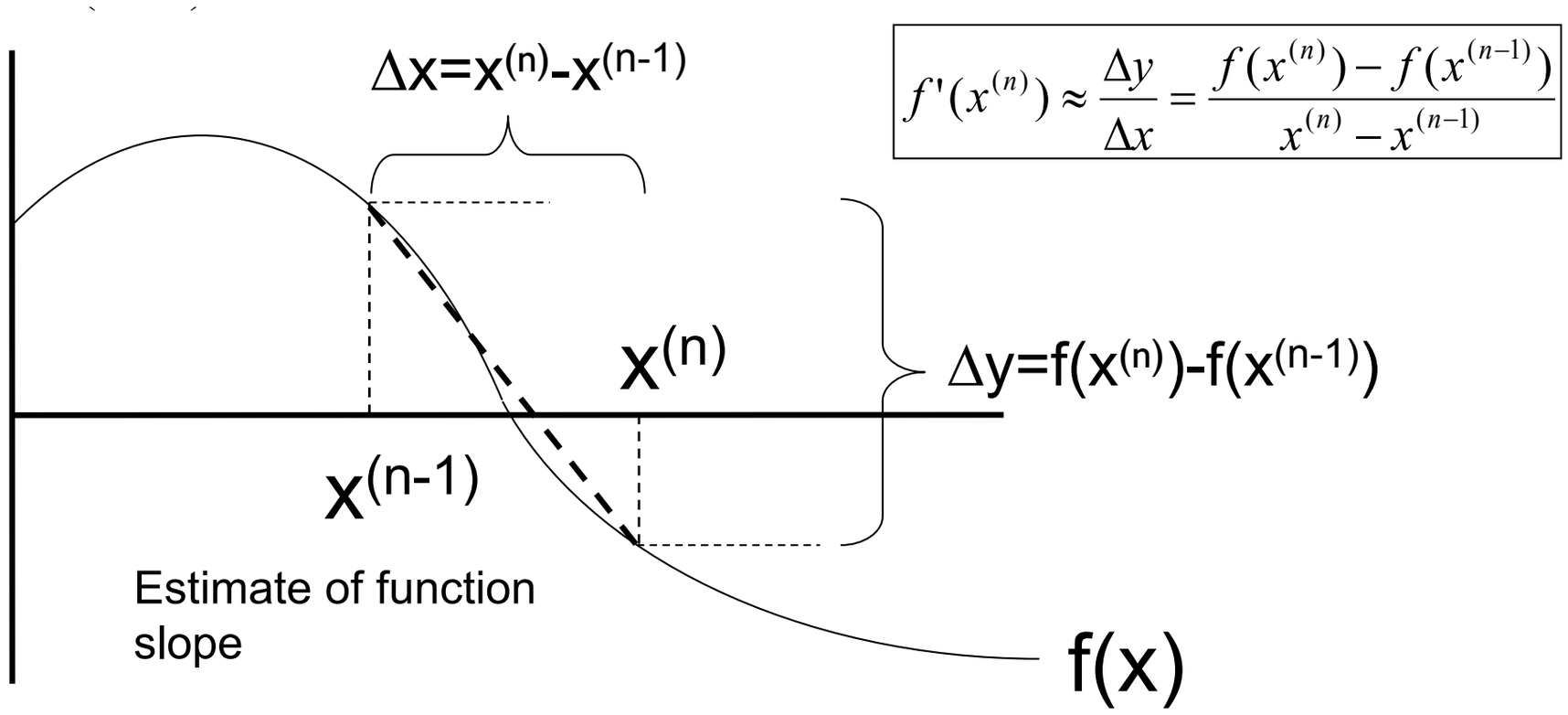
(d)

Newton-Raphson-Bisection Hybrid

- Combine the methods
  - Good from Newton-Raphson
  - convergence from bisection
- Start with $x_l < x^{(n)} < x_u$ as within the bracket
- Let $x_{NR}^{(n+1)} \approx$
  - If then take an NR step
  - Else set $x_{NR}^{(n+1)} =$
    - Check and update x_l and x_u values for
- Throw away steps: do a step instead

Secant Method

- A key issue in NR is that you need the
  - May be unknown if its
  - Could be to calculate
- Can estimate the using and



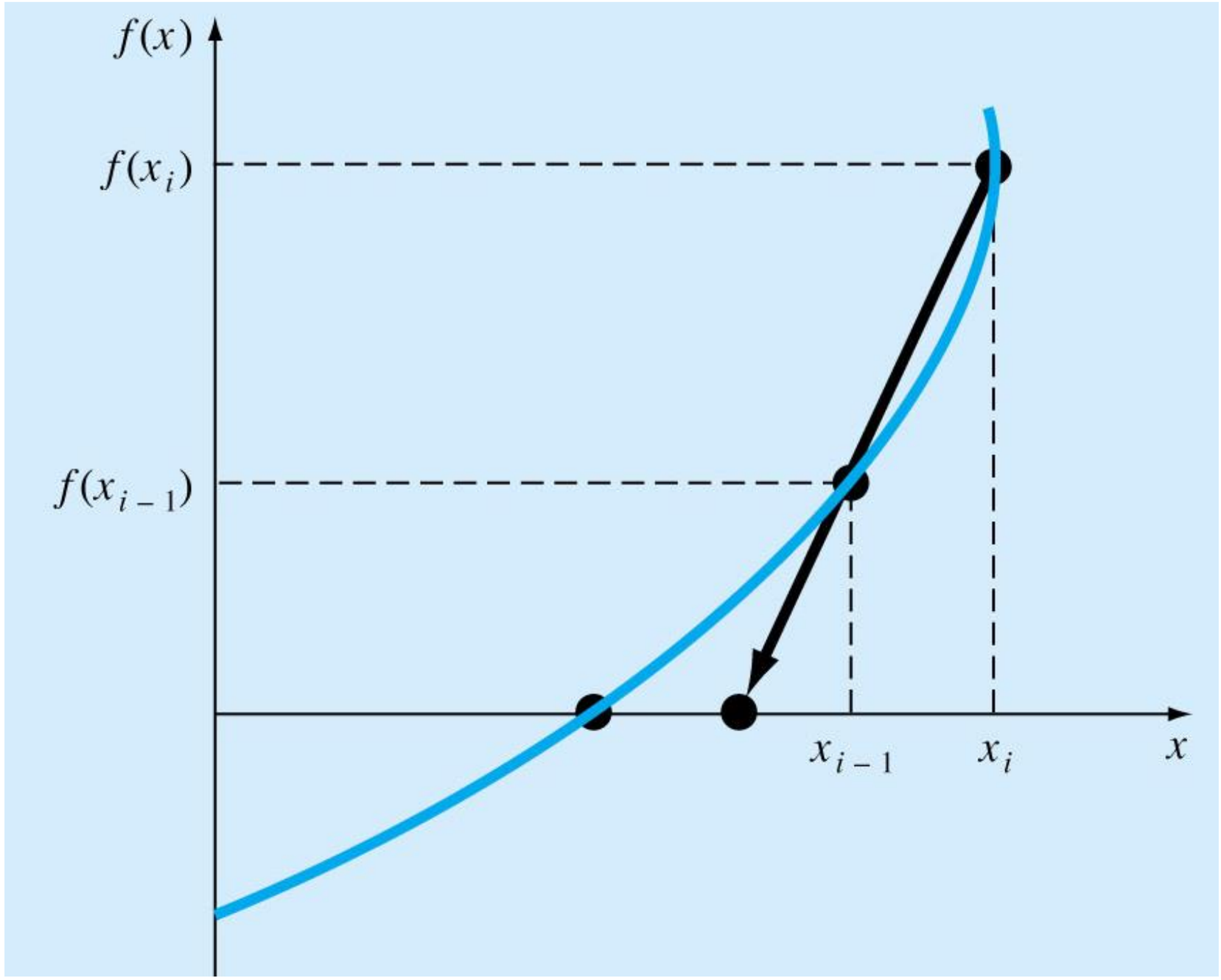
Secant Method

- After initially selecting a pair of points bracketing the root $(x^{(n-1)}, x^{(n)}), x^{(n+1)}$ is given by

  - $x^{(n+1)} =$

- Iterate to a s as in NR
- Converges almost as NR

Secant Method



Use line instead of line at $f(x_i)$

Algorithm for Secant Method

- Open Method
- Step 1: Begin with any two endpoints $[a, b] =$
- Step 2: Calculate x_2 using the secant method formula
  - $x_{i+1} =$.
- Step 3: Replace x_0 with , x_1 with and repeat from step 2 until convergence is reached
- Use the two points in iterations (not a bracket method)

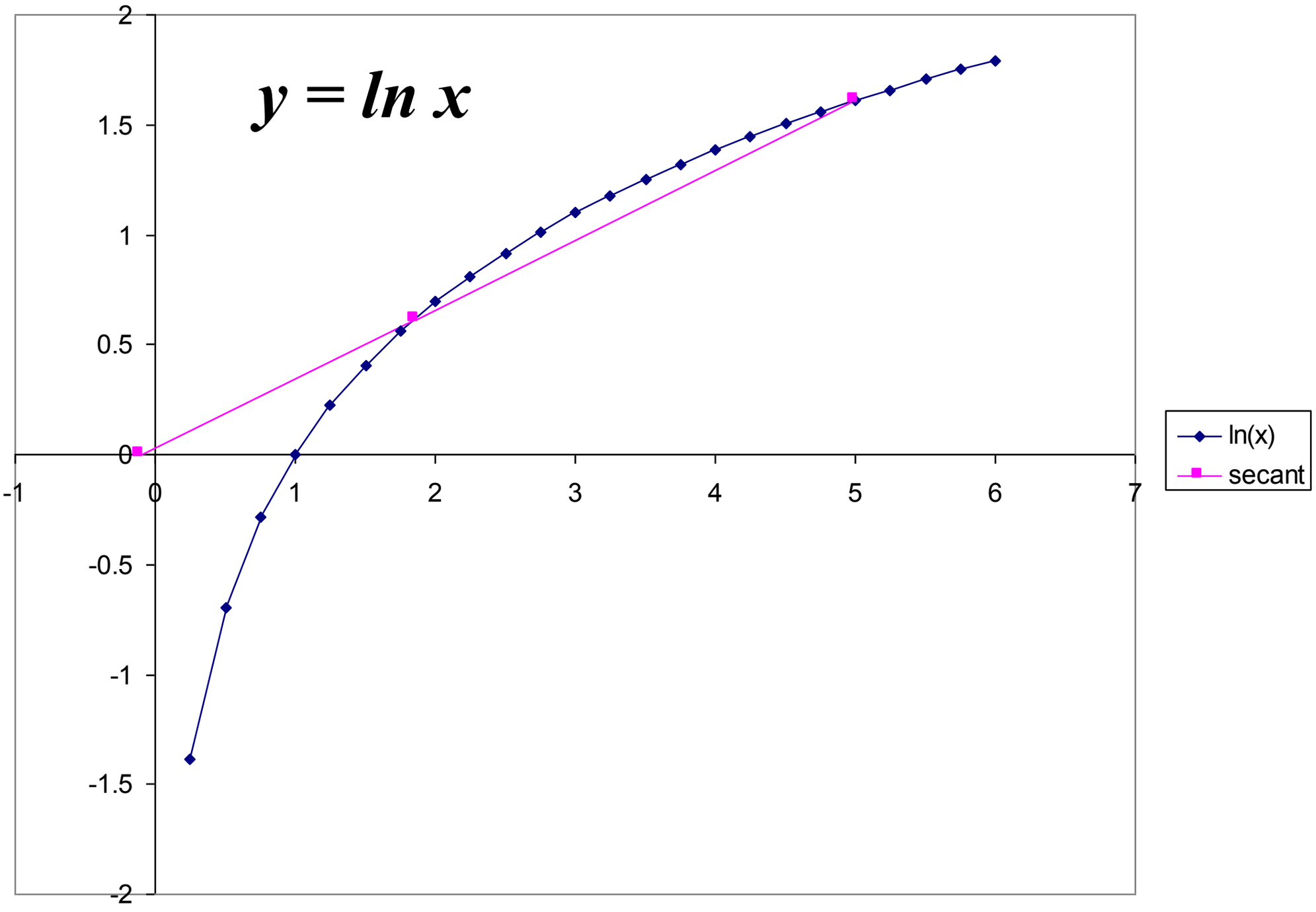
Secant Method

- Pros
  - It can and to bracket the root
- Cons
  - It is not guaranteed to c
  - It may and fail to yield an answer

Issues with the Secant Method

- Given the initial values $x_l = x^{(n-1)}$ and $x_u = x^{(n)}$ that bracket the root, the calculation of
  - Easy to , all values are easily
- Difficulty is with finding x_l and x_u since this is a problem
  - Determined by $f(x)$ over its
  - Consequently to program

Example: Lack of Convergence



Müller-Brent Method

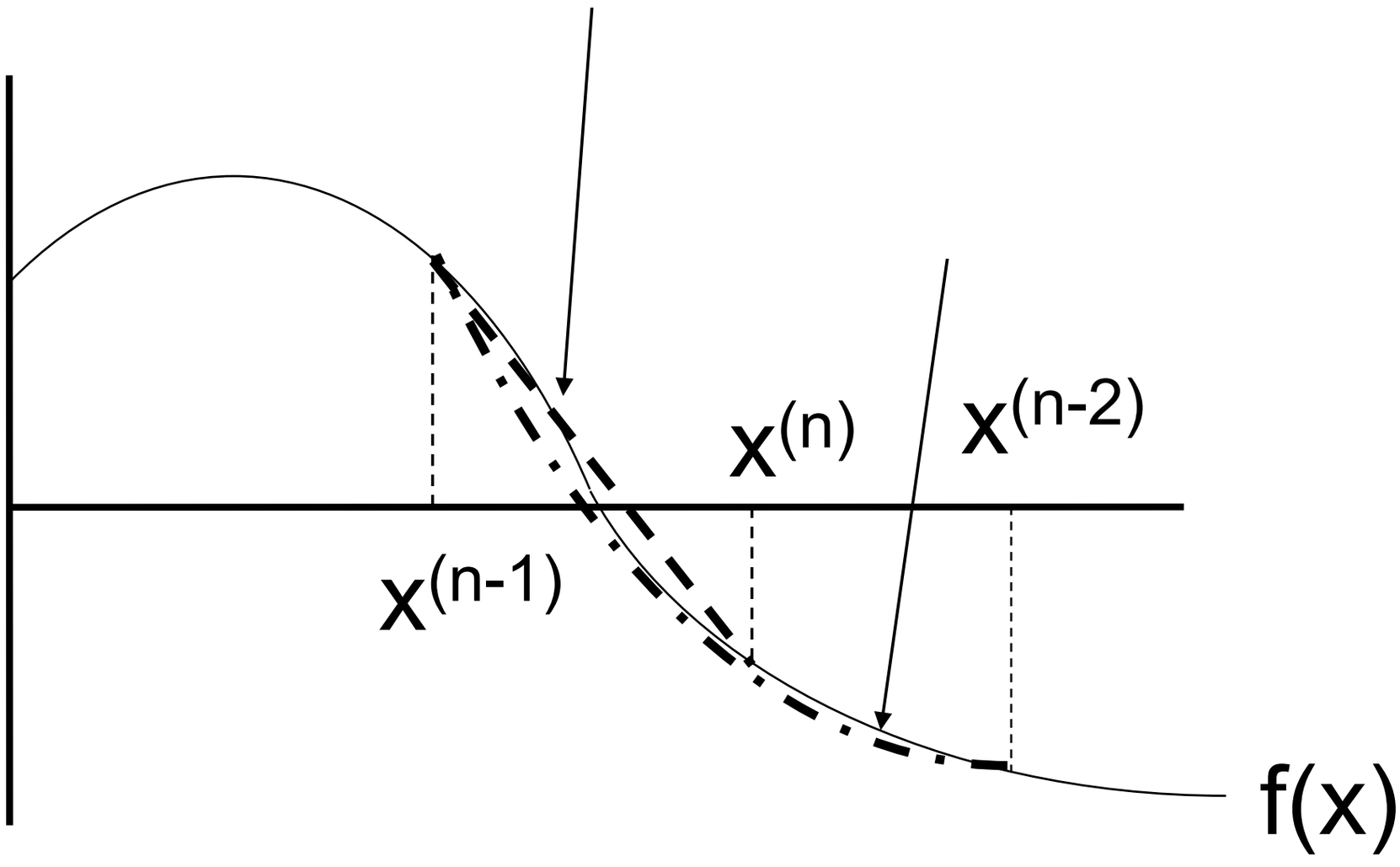
- Most expensive part of root finding algorithms is frequently the
- In Secant method the guess is based upon function evaluations from steps
  - Throw away the step
    - With only , the fitting method must be
- Could use the step to give us and use those for

Müller Method

- Method consists of of a
  that goes through the three points
- Let's write the equation in a form
  - $f_2(x) = a(x - x_2)^2 + b(x - x_2) + c$

Secant Method compared to Müller-Brent

- Secant method fits a between $(x^{(n)}, f(x^{(n)})$ and $(x^{(n-1)}, f(x^{(n-1)})$
  - Corresponds to



Fitting a Quadratic

- Three points is enough data to fit a function of the form
  - $q(x) = a(x - x^{(n)})^2 + b(x - x^{(n)}) + c$
- As we know $q(x)$ passes through all three points
  - $q(x^{(n)}) = f(x^{(n)}) \Rightarrow c = f(x^{(n)})$
  - $q(x^{(n-1)}) = f(x^{(n-1)}) = a(x^{(n-1} - x^{(n)})^2 + b(x^{(n-1} - x^{(n)}) + f(x^{(n)})$
  - $q(x^{(n-2)}) = f(x^{(n-1)}) = a(x^{(n-2} - x^{(n)})^2 + b(x^{(n-2} - x^{(n)}) + f(x^{(n)})$
- Two equations and two unknowns: a and b

Solving for Coefficients

- Given the two equations, we can eliminate a or b to get equations for the two constants

$$a = \frac{(x^{(n-1)} - x^{(n)})[f(x^{(n-2)}) - f(x^{(n)})] - (x^{(n-2)} - x^{(n)})[f(x^{(n-1)}) - f(x^{(n)})]}{(x^{(n-2)} - x^{(n-1)})(x^{(n-2)} - x^{(n)})(x^{(n-1)} - x^{(n)})}$$

$$b = \frac{(x^{(n-2)} - x^{(n)})^2[f(x^{(n-1)}) - f(x^{(n)})] - (x^{(n-1)} - x^{(n)})^2[f(x^{(n-2)}) - f(x^{(n)})]}{(x^{(n-2)} - x^{(n-1)})(x^{(n-2)} - x^{(n)})(x^{(n-1)} - x^{(n)})}$$

- Now have a, b and c, so we can use the general formula to calculate the zeros for $x^{(n+1)} - x^{(n)}$
  - $x^{(n+1)} - x^{(n)} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$