Numerical Methods in Engineering - LW4

Athanasios A. Markou

PhD, University Lecturer
Aalto University
School of Engineering
Department of Civil Engineering

January 31, 2024

Roots-Solving Nonlinear Equations

Introduction

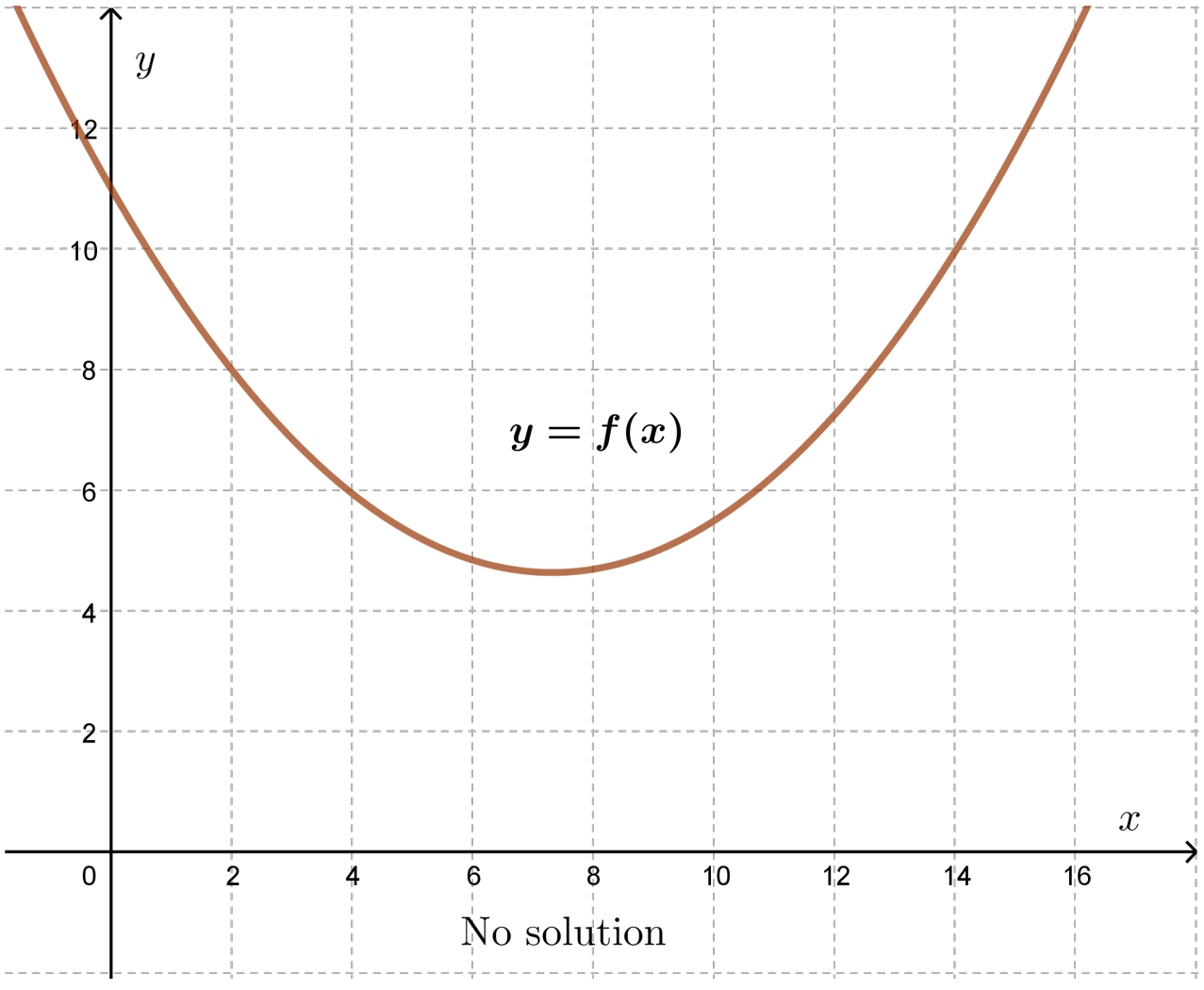
Introduction

- Solved equations are needed in all areas of science and engineering, [2].
- An equation of a single variable:

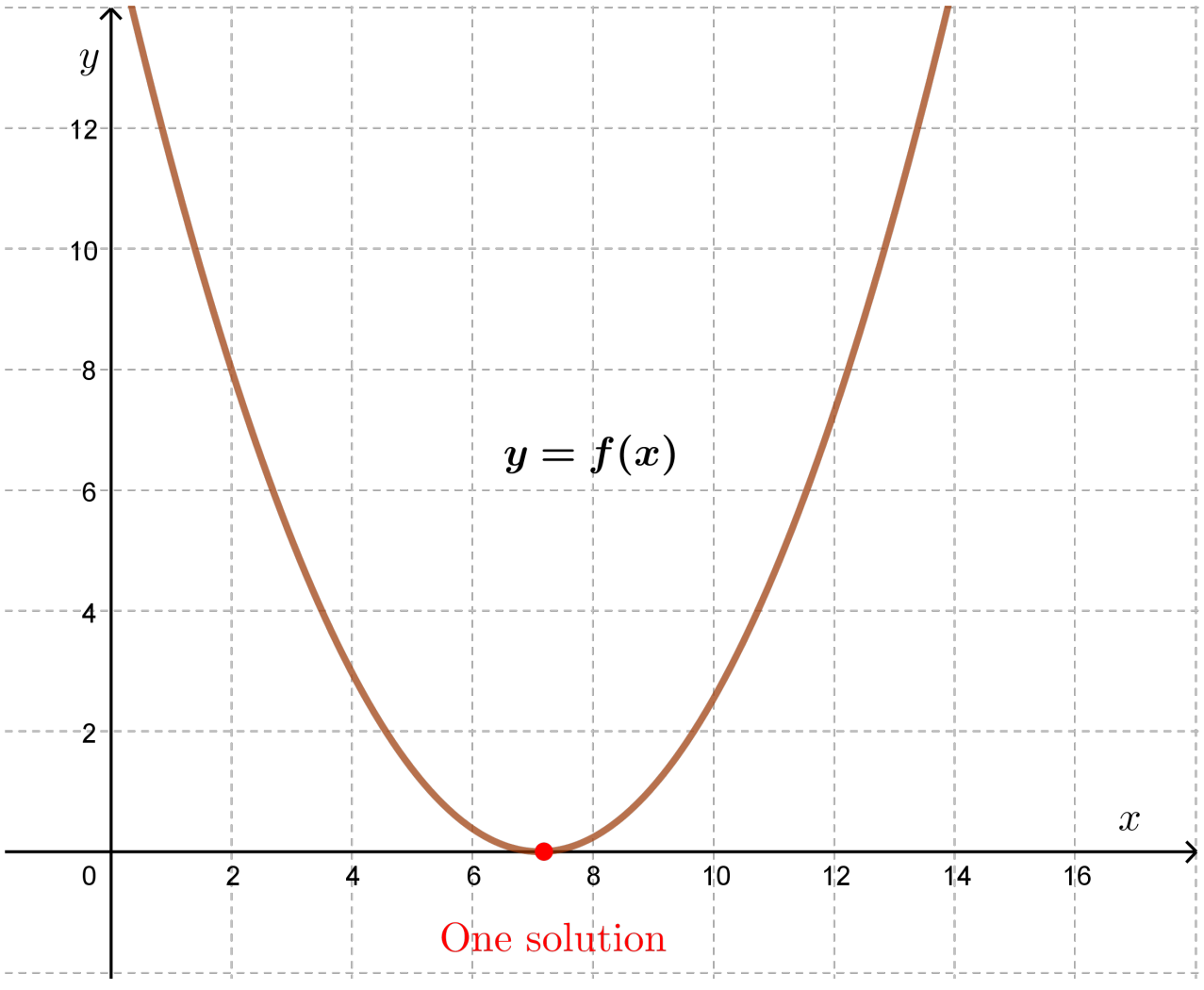
$$f(x) = 0$$

- The solution is called root.
- Roots can be real or complex
- Equations might have either no solutions or a single solution, or several solutions.
- When the equation is simple, the value of x can be determined analytically.
- In many situations the solution cannot be determined analytically.

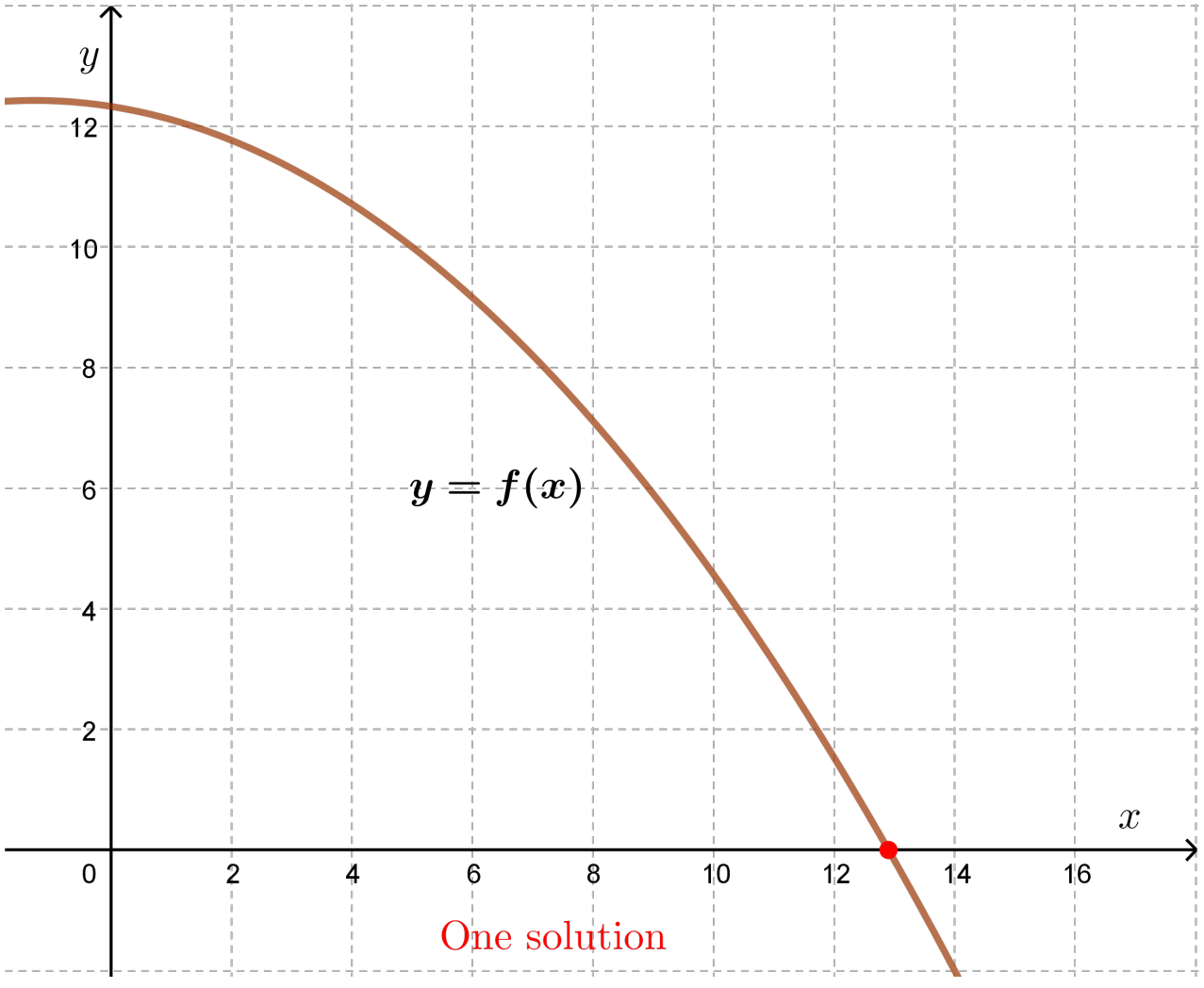
Roots-Solving Nonlinear Equations



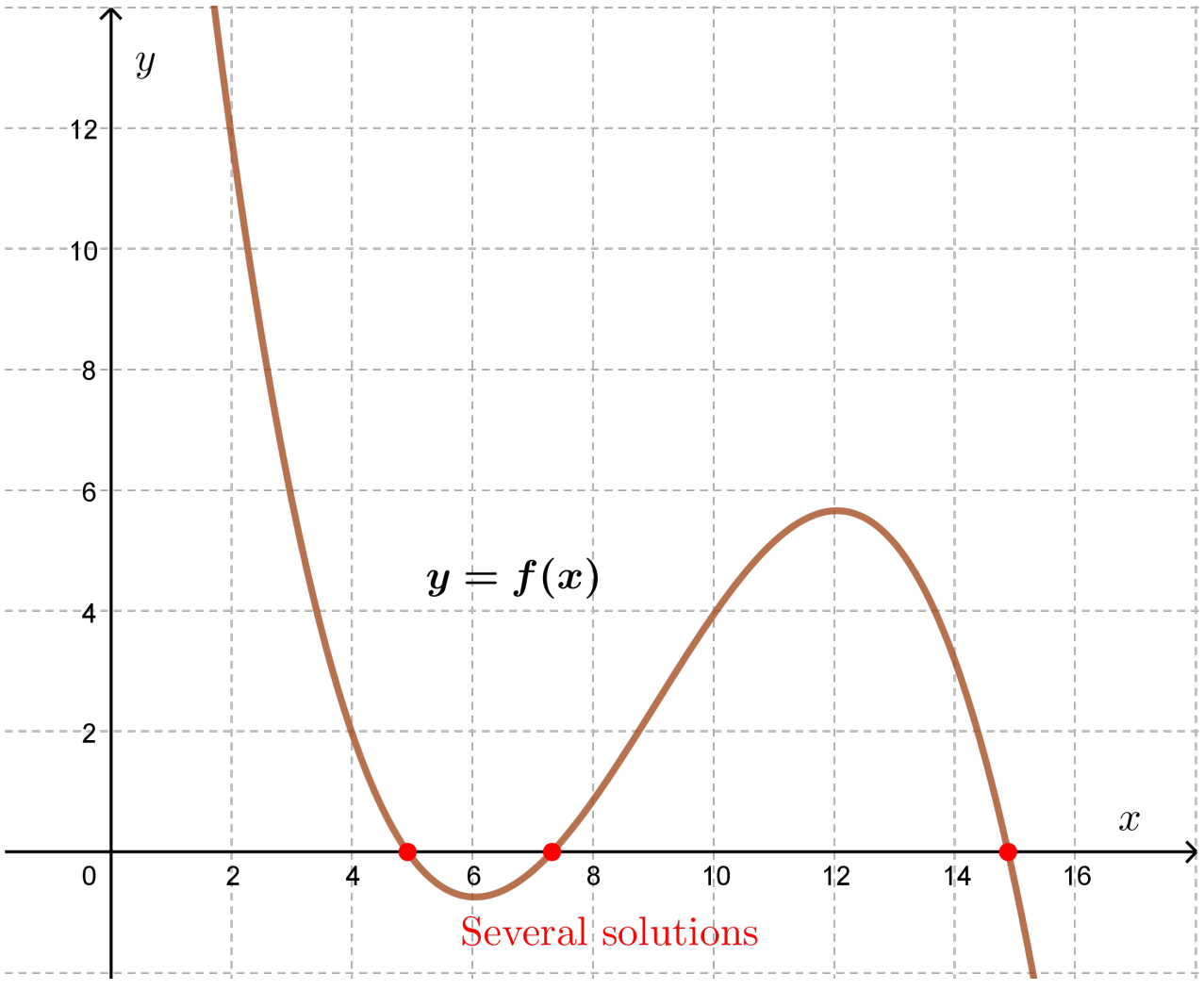
Roots-Solving Nonlinear Equations



Roots-Solving Nonlinear Equations

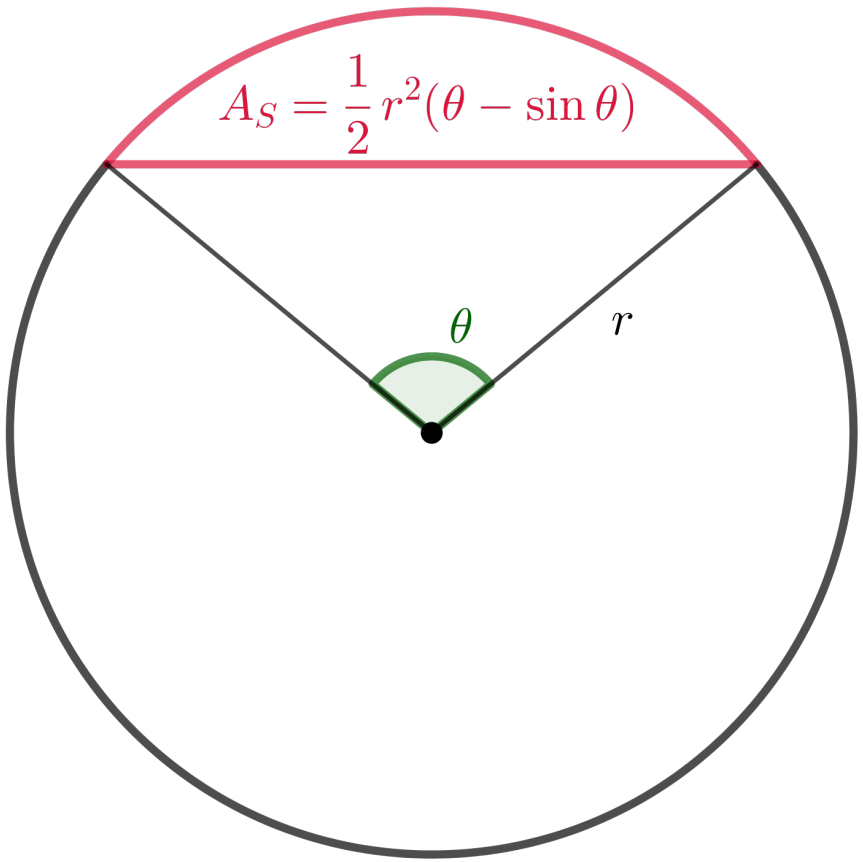


Roots-Solving Nonlinear Equations



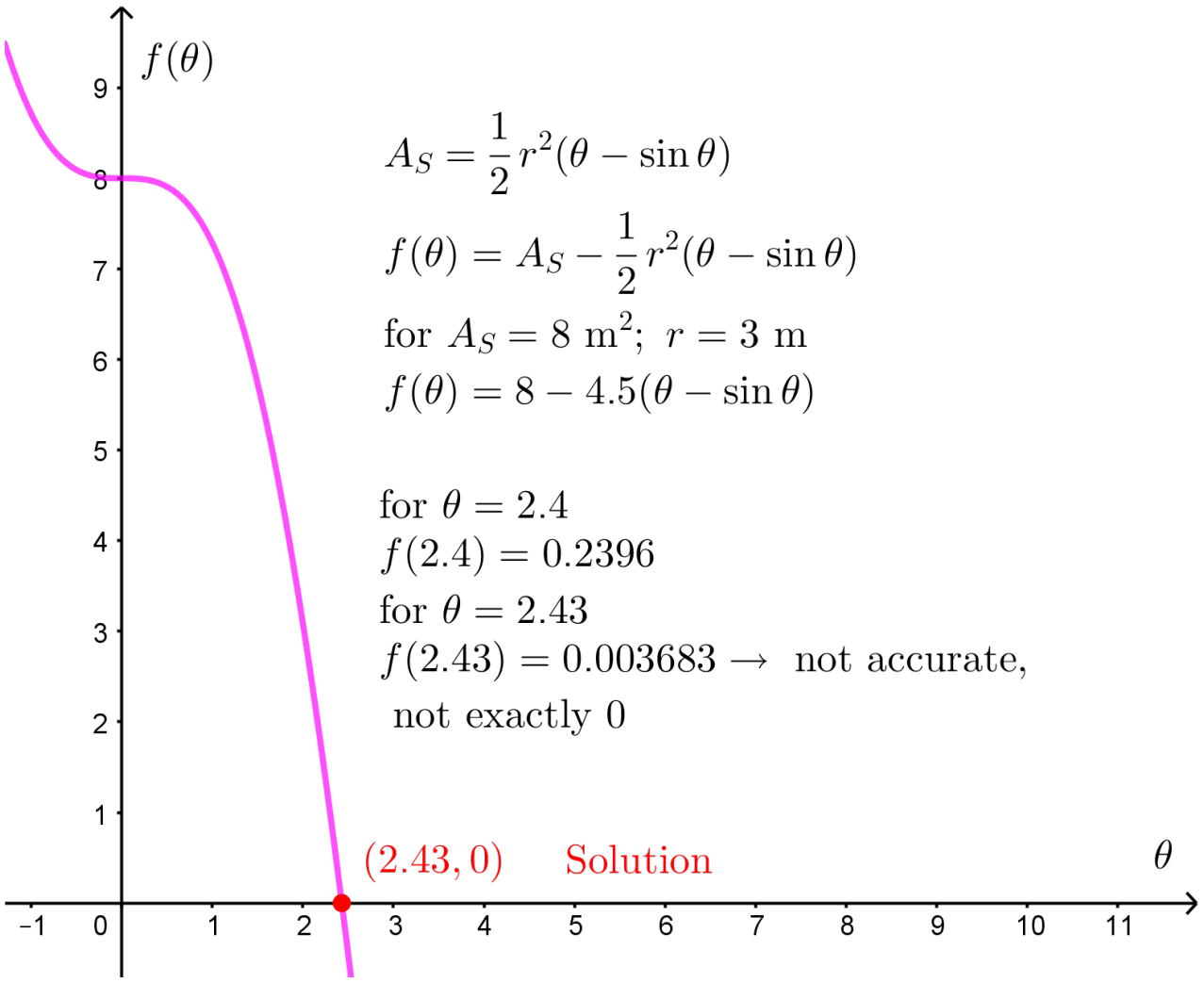
Roots-Solving Nonlinear Equations: Example

The area A_S of a segment of a circle with radius r is given $A_S = \frac{1}{2} r^2 (\theta - \sin\theta)$. The angle θ needs to be found for a fixed value of A_S, [2].



Roots-Solving Nonlinear Equations: Example

A numerical method is needed for the equation to be solved. A plot can provide with approximate information about the solution.



$$A_S = \frac{1}{2} r^2(\theta - \sin\theta)$$

$$f(\theta) = A_S - \frac{1}{2} r^2(\theta - \sin\theta)$$

for $A_S = 8$ m^2; $r = 3$ m

$$f(\theta) = 8 - 4.5(\theta - \sin\theta)$$

for $\theta = 2.4$

$f(2.4) = 0.2396$

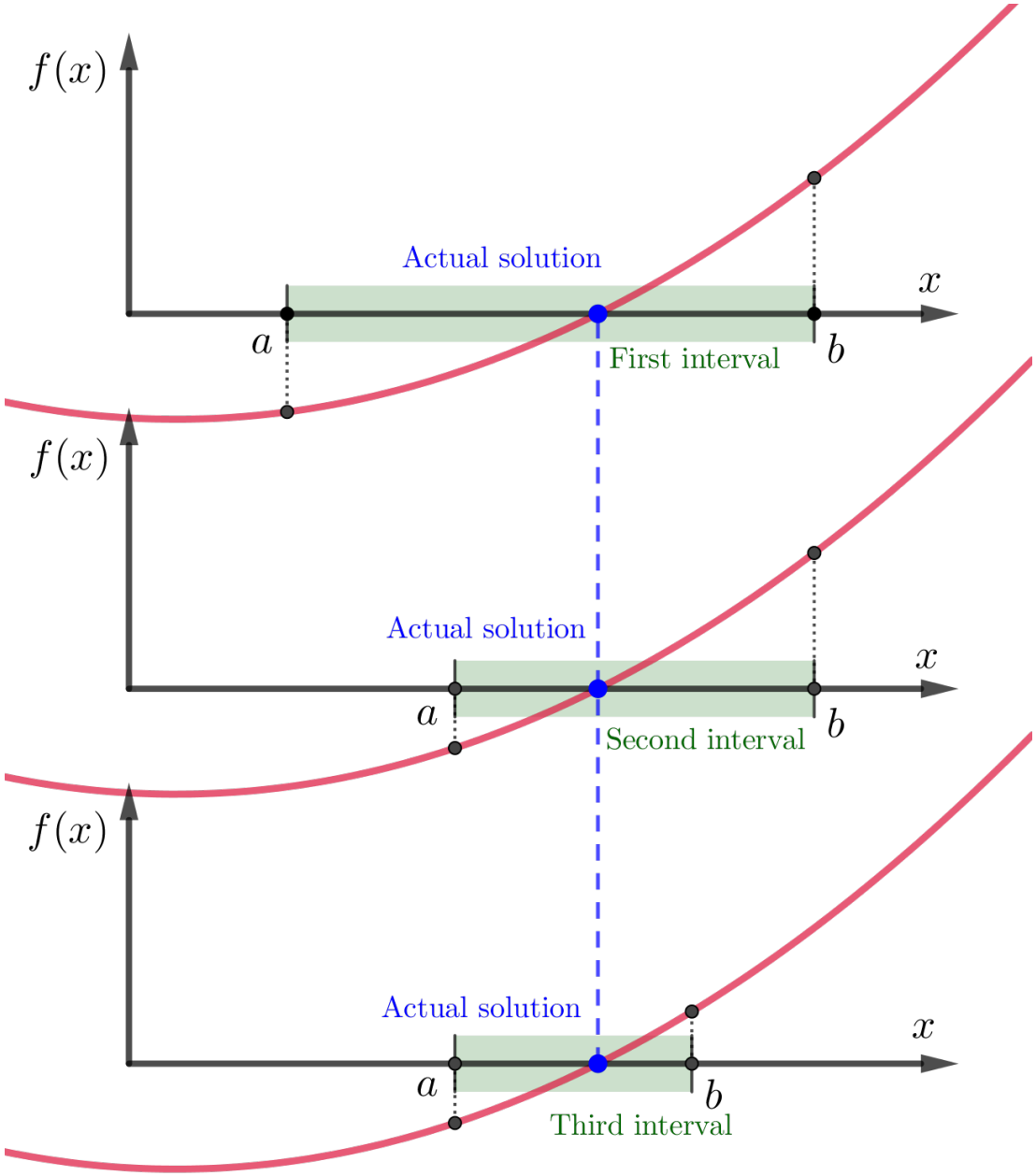
for $\theta = 2.43$

$f(2.43) = 0.003683 \rightarrow$ not accurate, not exactly 0

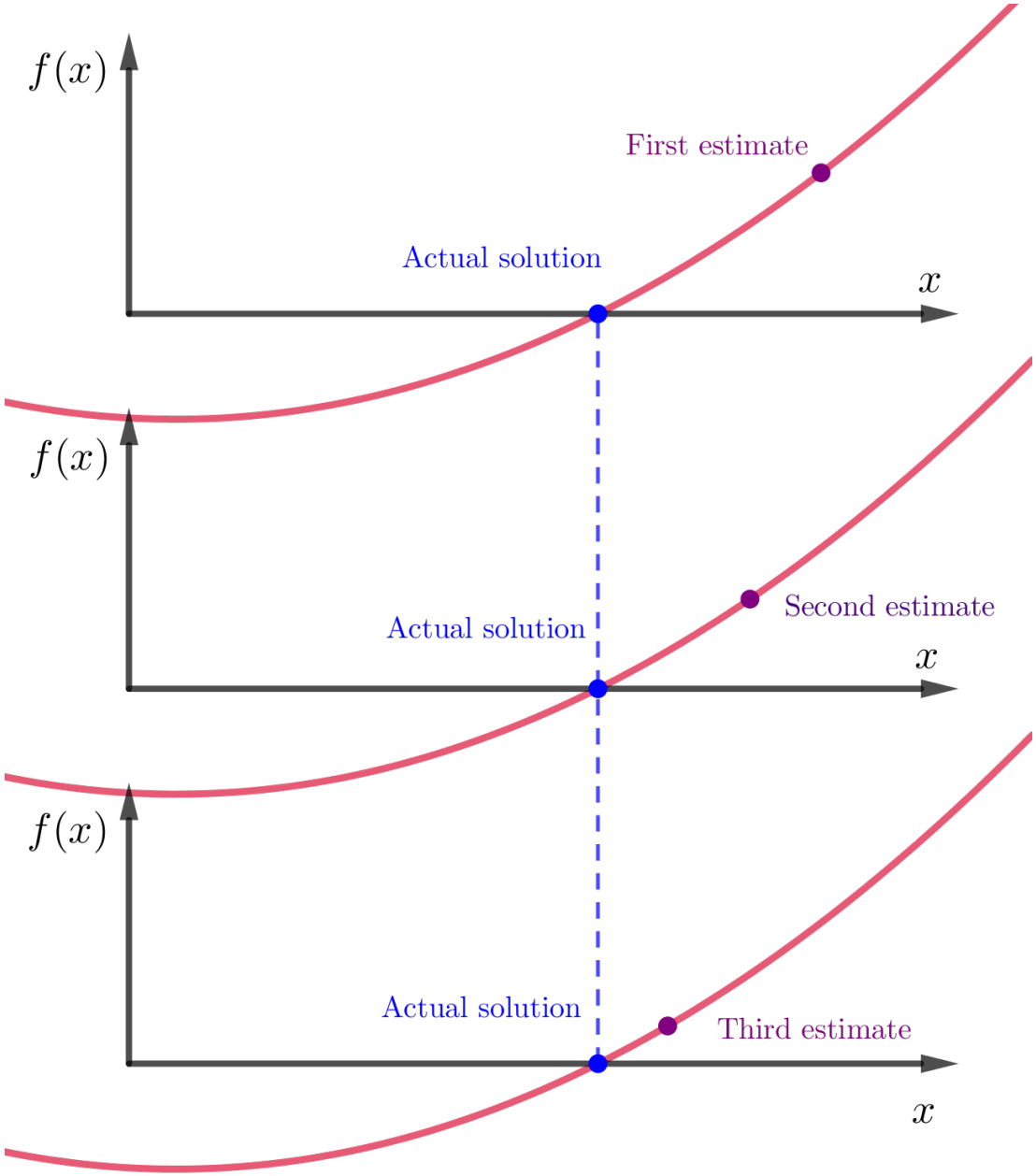
Introduction

- Numerical solution is different than analytical.
- Analytical is exact.
- Numerical solution starts from finding approximate solutions.
- Numerical solution starts with initial guess.
- A program can be written that looks for domains that contain a solution.
- Program starts at one value of x and changes the value of x to small increments.
- A change of sign in $f(x)$ indicates that the root within the last increment.

Numerical solution - Bracketing method



Numerical solution - Open method



Bracketing and Open methods

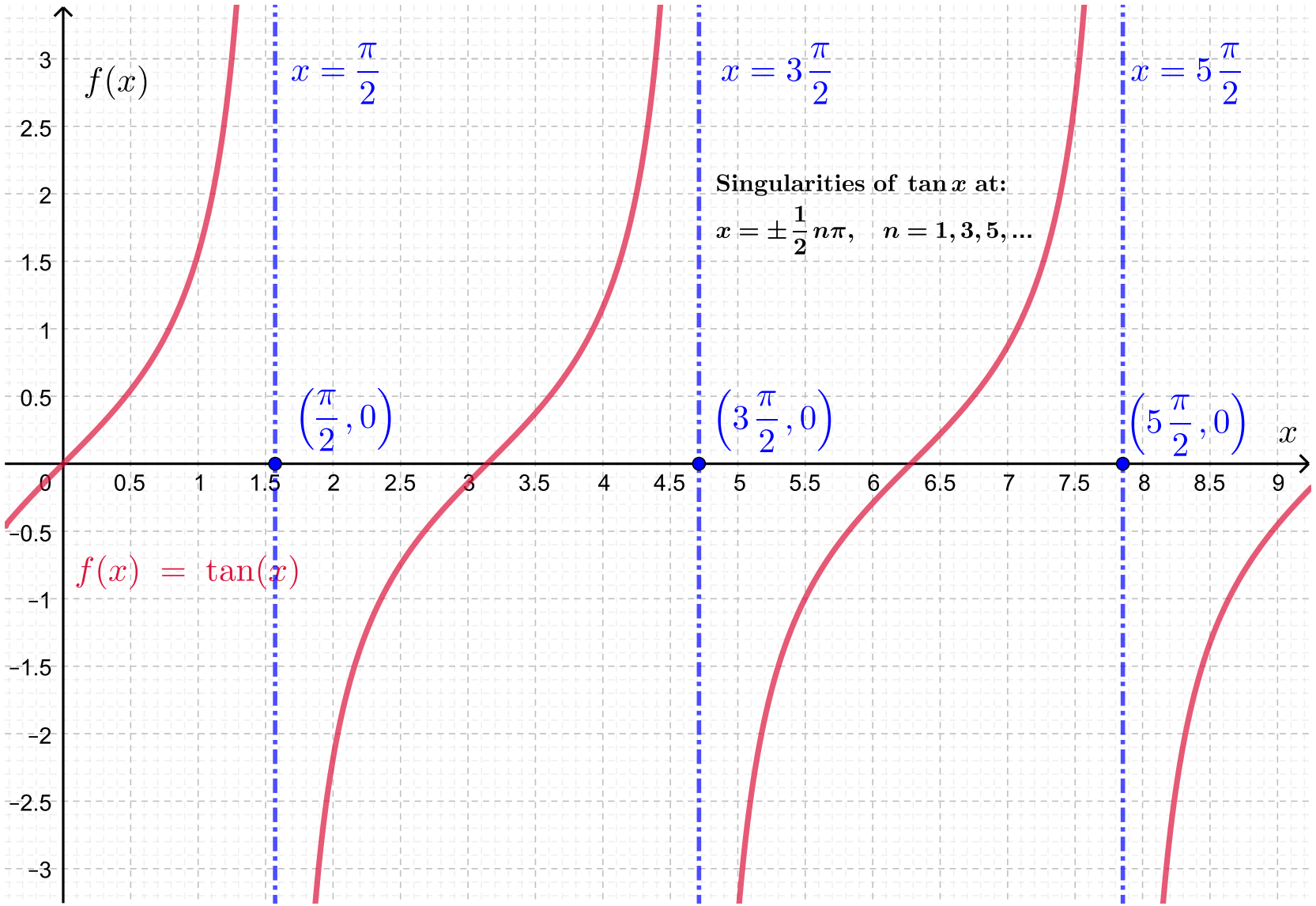
- Bracketing methods always converge
- Open methods are more efficient, but sometimes might not yield the solution
- Bracketing methods:
  - Bisection
  - Regula falsi
- Open methods:
  - Newton's
  - Secant
  - Fixed-point iteration

Incremental Search Method

Incremental Search Method

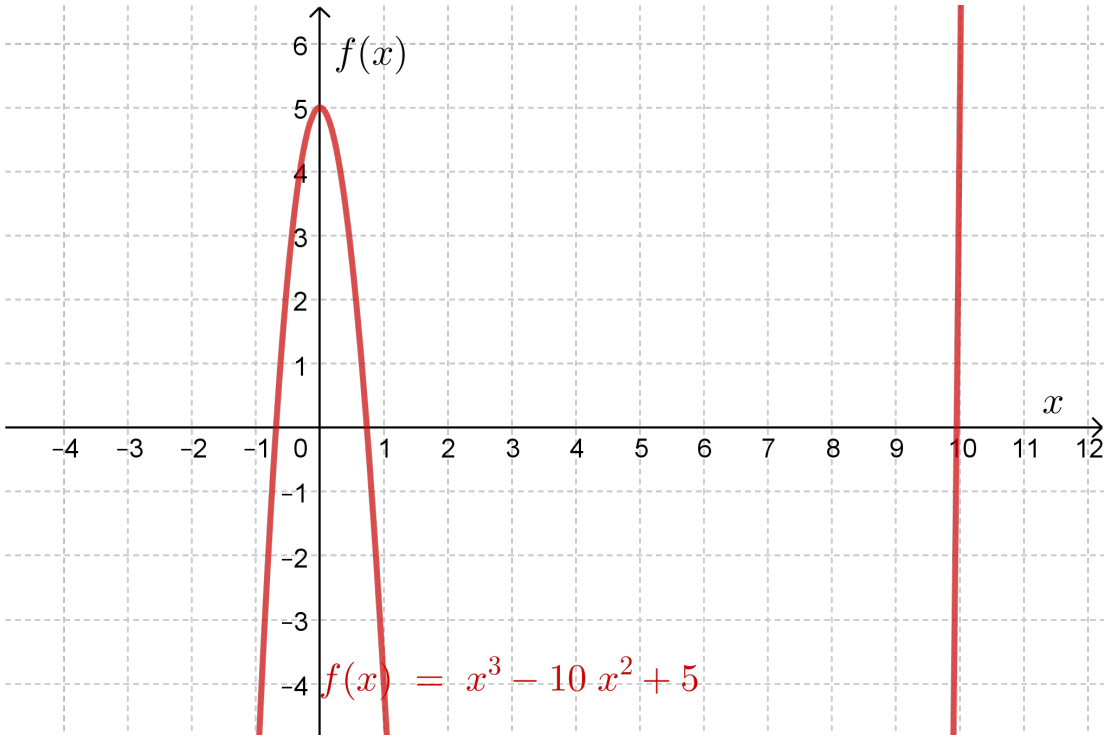
- The approximate solutions of the roots can be determined by plotting a function, which often is useful
- An alternative way is the incremental search method
- The basic idea is: if $f(x_1)$ and $f(x_2)$ have opposite signs, then there is at least one root in (x_1, x_2)
- If the interval is small, then it might contain a single root
- The function can be calculated at intervals Δx and check for change of sign.
- Problems with the method:
  1. Possible to miss two closely spaced roots, depending on the Δx
  2. Double roots cannot be detected
  3. Singularities can be mistaken for roots

Incremental Search Method: Singularities Example



Incremental Search Method: Example

Use incremental search with $\Delta x = 0.2$ to bracket the smallest positive zero of $f(x) = x^3 - 10x^2 + 5$ and $f(x) = tan(x)$



Write a user-defined MatLab function named `rootsearch` that will have as inputs a function handle, `func`, a lower bound `a`, an upper bound `b` and step `dx` and as an output will provide the values `[x1,x2]` within which the root is located.

Incremental Search Method: Code

```
function [x1,x2] = rootsearch(func,a,b,dx)
% Incremental search for a root of f(x).
% USAGE: [x1,x2] = rootsearch(func,a,d,dx)
% INPUT:
% func = handle of function that returns f(x).
% a,b = limits of search.
% dx = search increment.
% OUTPUT:
% x1,x2 = bounds on the smallest root in (a,b);
% set to NaN if no root was detected
x1 = a;       f1 = feval(func,x1);
x2 = a + dx; f2 = feval(func,x2);
while f1*f2> 0.0
    if x1 >= b
       x1  = NaN; x2 = NaN; return
    end
    x1 = x2;       f1 = f2;
    x2 = x1 + dx; f2 = feval(func,x2);
end
```

Errors

- Numerical solutions are not exact.
- The exact solution is the one that provides $f(x_{TS}) = 0$ the true solution and the approximate numerical solution is $f(x_{NS}) = \epsilon$ (where ϵ is very small). Four measures can be defined:
  1. True error: $x_{TS} - x_{NS}$
  2. Tolerance in $f(x)$: we consider the deviation of $f(x_{NS})$ from zero. The tolerance in $f(x)$ is defined as the absolute value of the difference between $f(x_{TS})$ and $f(x_{NS})$:

$$ToleranceInf = |f(x_{TS}) - f(x_{NS})| = |0 - \epsilon| = |\epsilon|$$

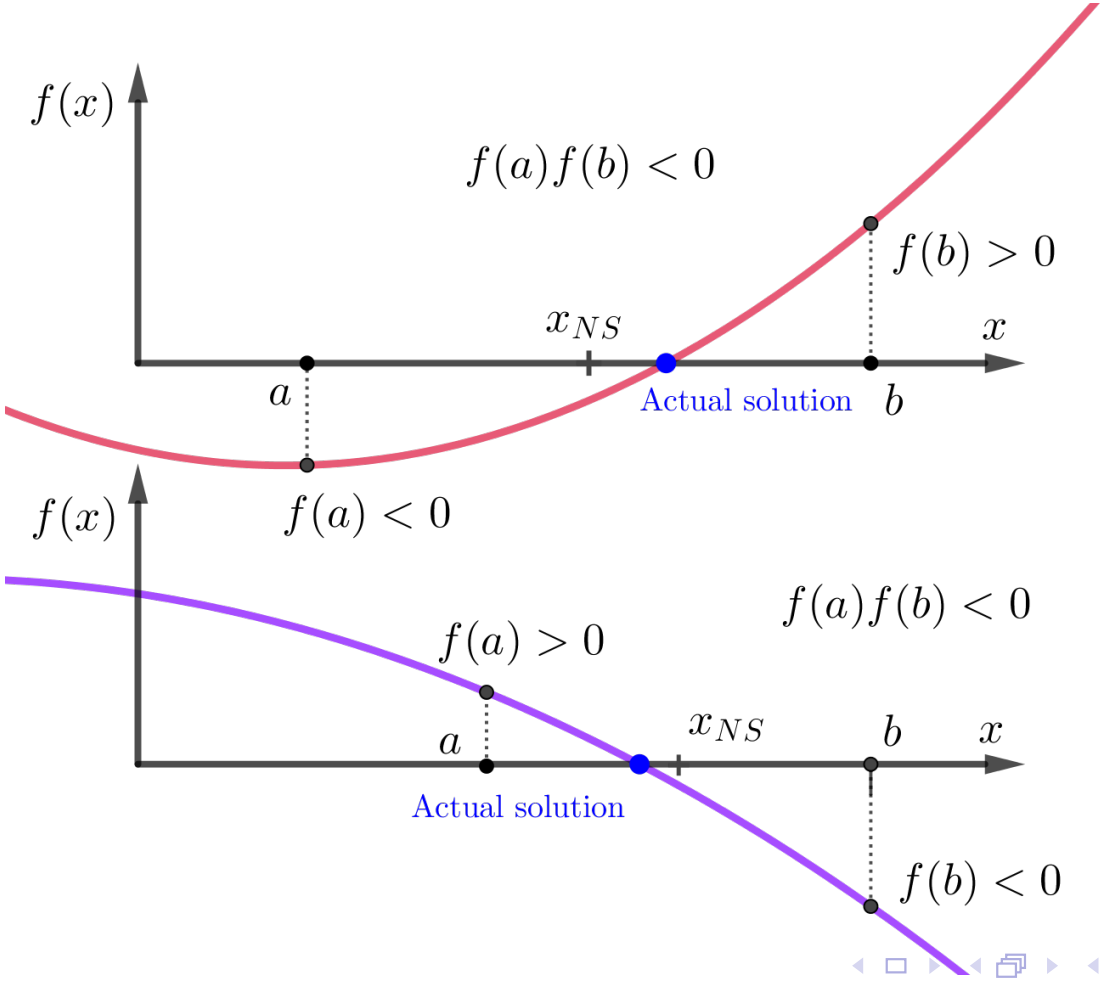
  3. Tolerance in the solution: maximum amount by which the true solution can deviate from an approximate numerical solution. It is useful for bracketing methods, where for example if the solution is in the domain $[a, b]$ then the numerical solution can be defined as the midpoint: $x_{NS} = \frac{a+b}{2}$, plus minus the $Tolerance = \left|\frac{b-a}{2}\right|$
  4. Relative error: $TrueRelativeError = \left|\frac{x_{TS} - x_{NS}}{x_{TS}}\right|$, but because the x_{TS} is not known, we estimate the $EstimatedRelativeError = \left|\frac{x_{NS}^{(n)} - x_{NS}^{(n-1)}}{x_{NS}^{(n-1)}}\right|$, where n is the last iteration

Bracketing methods

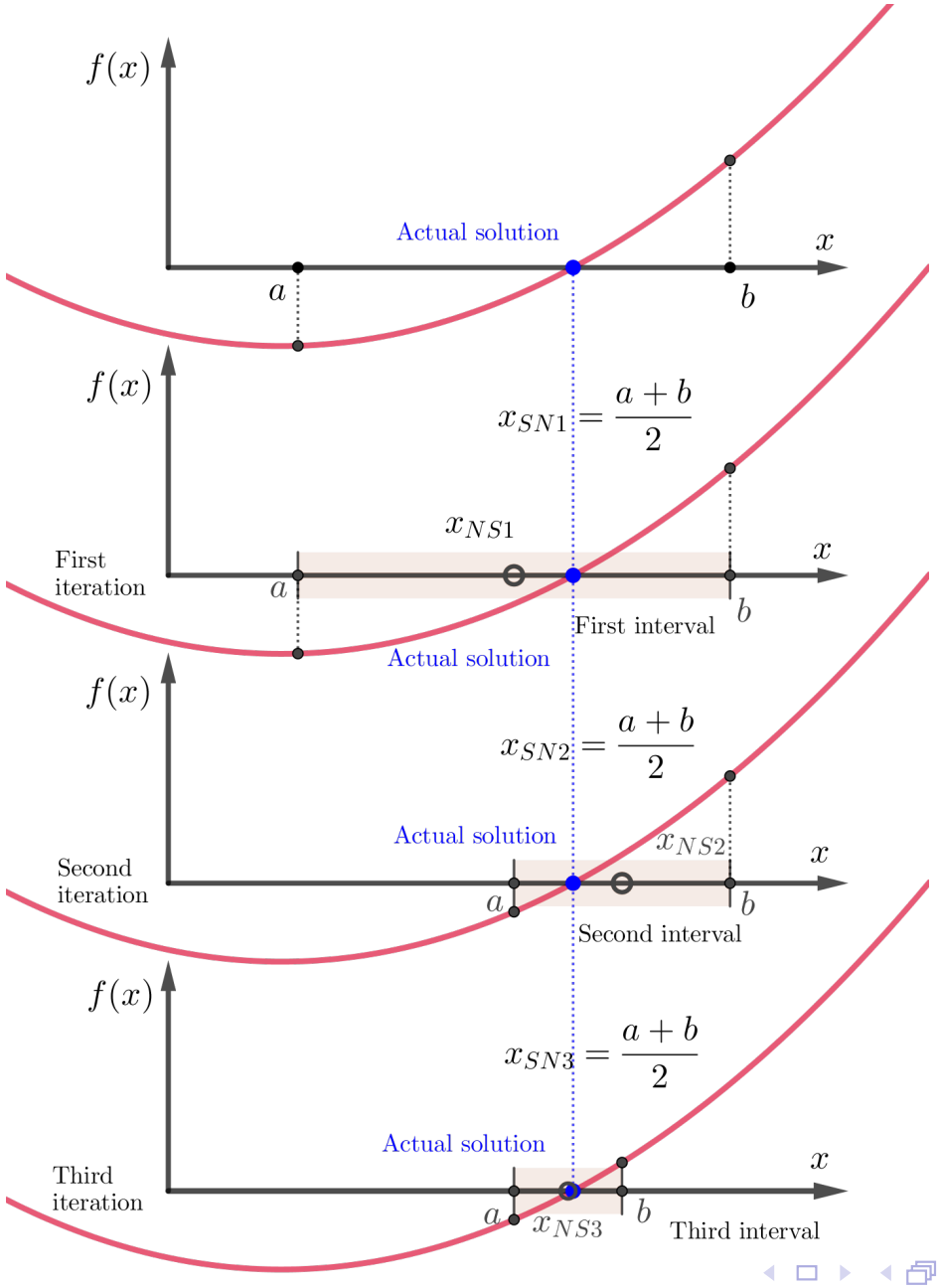
Bisection Method

Bisection Method

Bisection method is a bracketing method for finding a numerical solution of an equation $f(x) = 0$, when it is known that in the interval $[a, b]$, $f(x)$ is continuous and the equation has a solution, [2].



Bisection Method: Scheme, [2]



Bisection Method: Algorithm

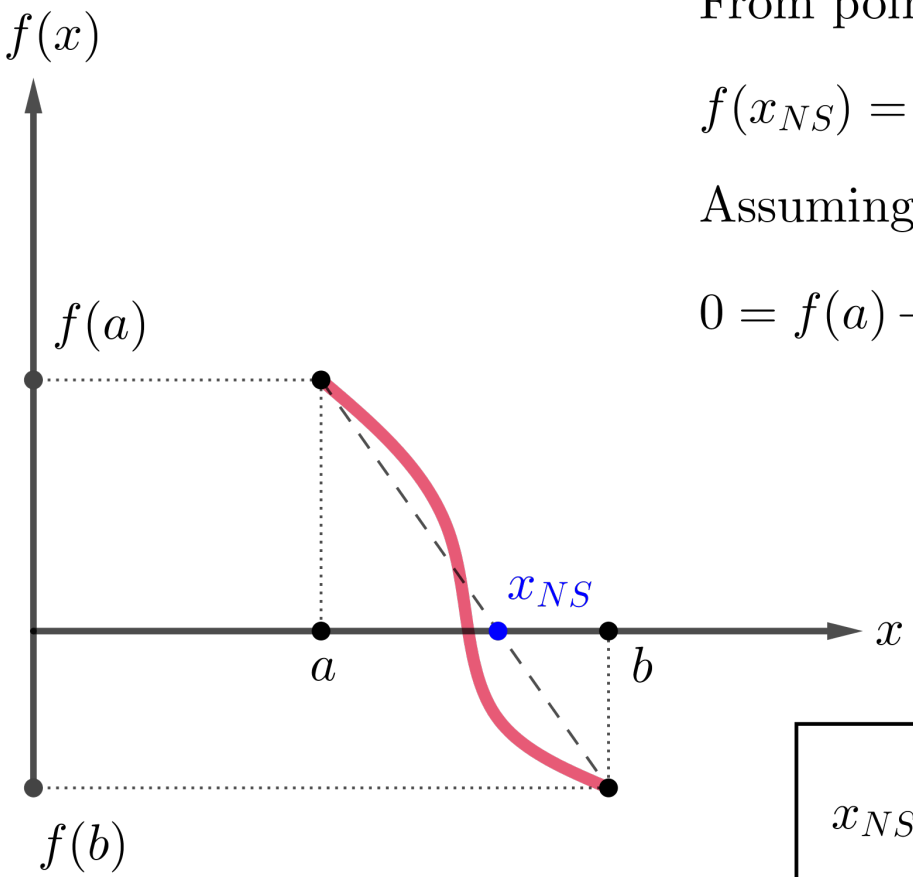
1. Chose first interval by defining a and b, so that a solution lies between them. Practically, it means that $f(a)f(b) < 0$. Points can be determined by plotting $f(x)$ versus x.
2. Calculate the first estimate of the numerical solution:

$$x_{NS1} = \frac{(a+b)}{2}$$

3. Determine if the true solution lies between a and x_{NS1} or between x_{NS1} and b. Check the product $f(a)f(x_{NS1})$:
   - if $f(a)f(x_{NS1}) < 0$, then the true solution lies between a and x_{NS1}
   - if $f(a)f(x_{NS1}) > 0$, then the true solution lies between x_{NS1} and b
4. Select the new interval that contains the true solution and (either $[a, x_{NS1}]$ or $[x_{NS1}, b]$) as the new $[a, b]$ and go to step 2. (Steps 2 to 4 are repeated until convergence is achieved).

Bisection Method: Algorithm

5. After convergence has been achieved, use linear interpolation to estimate the root from *a*:



From point a, using linear interpolation:

$$f(x_{NS}) = f(a) + \frac{f(b) - f(a)}{b - a}(x_{NS} - a)$$

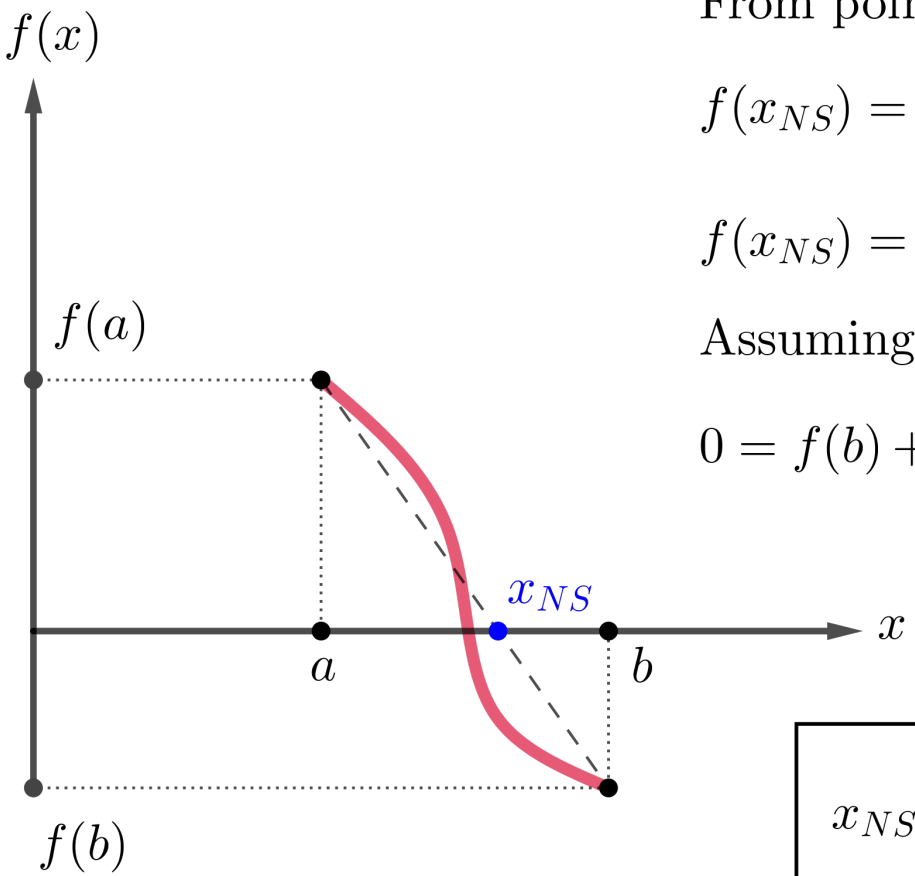
Assuming $f(x_{NS}) \approx 0$:

$$0 = f(a) + \frac{f(b) - f(a)}{b - a}(x_{NS} - a) \Leftrightarrow$$

$$x_{NS} = a - f(a)\frac{b - a}{f(b) - f(a)}$$

Bisection Method: Algorithm

5. After convergence has been achieved, use linear interpolation to estimate the root from b:



From point b, using linear interpolation:

$$f(x_{NS}) = f(b) - \frac{f(b) - f(a)}{b - a}(b - x_{NS}) \Leftrightarrow$$

$$f(x_{NS}) = f(b) + \frac{f(b) - f(a)}{b - a}(x_{NS} - b)$$

Assuming $f(x_{NS}) \approx 0$:

$$0 = f(b) + \frac{f(b) - f(a)}{b - a}(x_{NS} - b) \Leftrightarrow$$

$$\boxed{x_{NS} = b - f(b)\frac{b - a}{f(b) - f(a)}}$$

Bisection method: Convergence

- True solution generally cannot be achieved computationally
- The process is stopped when one of the error measures is smaller compared to some predefined value ϵ
- The method always converges if a root lies in the interval $[a, b]$
- The method might fail in case of a function being tangent to the axis and does not cross the x-axis
- The convergence is slow compared to other methods
- The number of bisections required to reach a tolerance ϵ is:

$$n = \frac{\ln |\Delta x| - \ln \epsilon}{\ln 2}$$

where Δx is the initial interval and after one bisection the interval is reduced to $\Delta x/2$ and after n bisections it becomes $\Delta x/2^n$. The convergence is linear.

Bisection method, Example

Use the bisection method for the function $3x^2 - 15x + 4 = 0$ for the span $[3, 6]$. Do 5 iterations.

1. Chose first interval by defining a and b, so that a solution lies between them. Practically, it means that $f(a)f(b) < 0$. Points can be determined by plotting $f(x)$ versus x.
2. Calculate the first estimate of the numerical solution:

$$x_{NS1} = \frac{(a+b)}{2}$$

3. Determine if the true solution lies between a and x_{NS1} or between x_{NS1} and b. Check the product $f(a)f(x_{NS1})$:
   - if $f(a)f(x_{NS1}) < 0$, then the true solution lies between a and x_{NS1}
   - if $f(a)f(x_{NS1}) > 0$, then the true solution lies between x_{NS1} and b
4. Select the new interval that contains the true solution and (either $[a, x_{NS1}]$ or $[x_{NS1}, b]$) as the new $[a, b]$ and go to step 2. (Steps 2 to 4 are repeated until convergence is achieved).

Bisection method, Example

Use the bisection method for the function $3x^2 - 15x + 4 = 0$ for the span $[3, 6]$. Do 5 iterations.

Solution

1st iteration

$$a = 3; \rightarrow f(a) = -14$$
$$b = 6; \rightarrow f(b) = 22$$
$$x_{NS1} = \frac{a+b}{2} = 4.5; \rightarrow f(x_{NS1}) = -2.75$$

2nd iteration

$$a = 4.5; \rightarrow f(a) = -2.75$$
$$b = 6; \rightarrow f(b) = 22$$
$$x_{NS2} = \frac{a+b}{2} = 5.25; \rightarrow f(x_{NS2}) = 7.3975....$$

Bisection method, Example

Use the bisection method for the function $3x^2 - 15x + 4 = 0$ for the span $[3, 6]$. Do 5 iterations.

Solution

3rd iteration

$$a = 4.5; \rightarrow f(a) = -2.75$$
$$b = 5.25; \rightarrow f(b) = 7.3975....$$
$$x_{NS3} = \frac{a+b}{2} = 4.875; \rightarrow f(x_{NS3}) = 2.1718....$$

4th iteration

$$a = 4.5; \rightarrow f(a) = -2.75$$
$$b = 4.875; \rightarrow f(b) = 2.1718.....$$
$$x_{NS4} = \frac{a+b}{2} = 4.6875; \rightarrow f(x_{NS4}) = -0.3945....$$

Bisection method, Example

Use the bisection method for the function $3x^2 - 15x + 4 = 0$ for the span $[3, 6]$. Do 5 iterations.
Solution
5th iteration

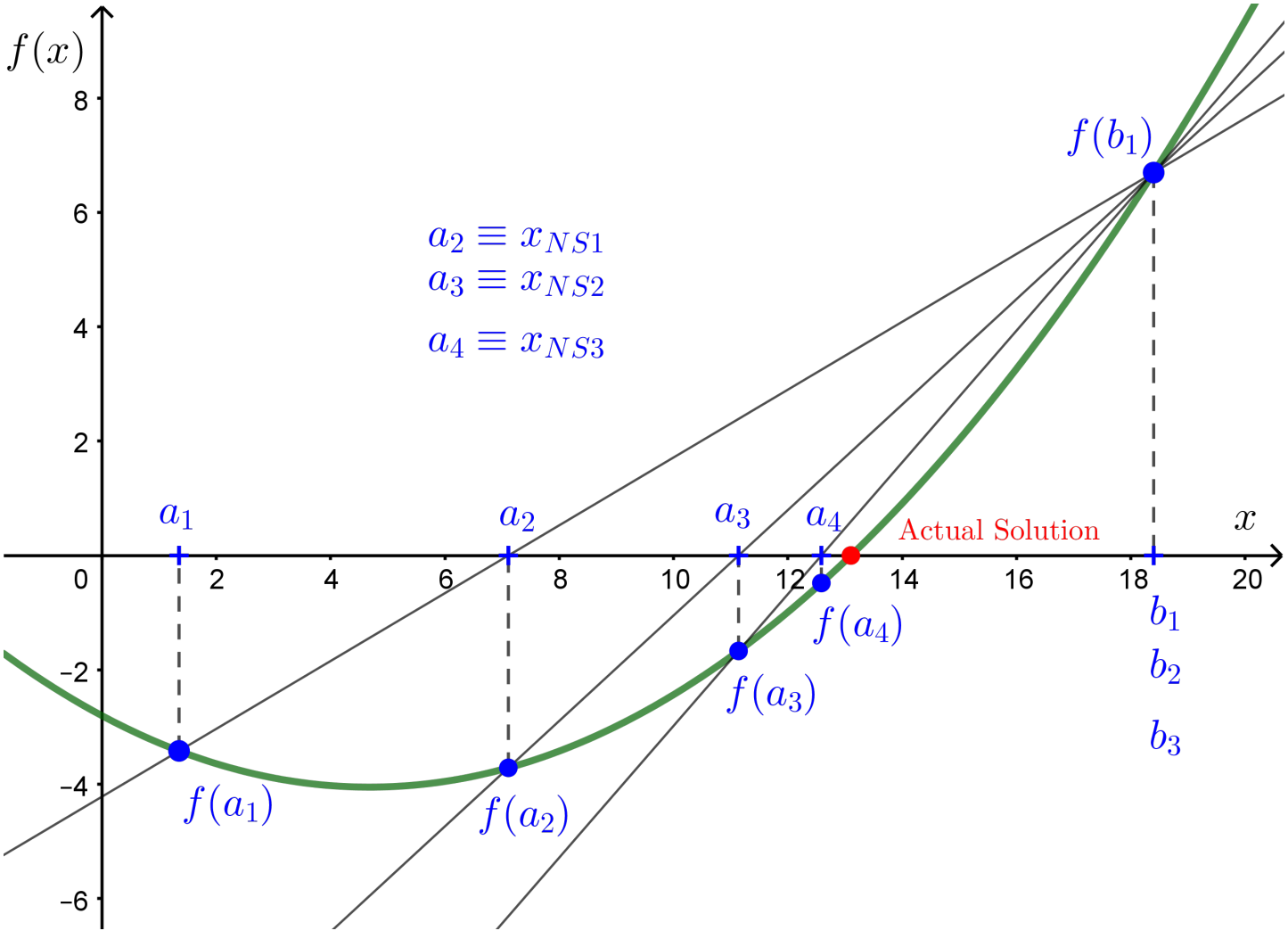
$$a = 4.6875; \rightarrow f(a) = -0.3945....$$
$$b = 4.875; \rightarrow f(b) = 2.1718.....$$
$$x_{NS5} = \frac{a+b}{2} = 4.78125; \rightarrow f(x_{NS5}) = -0.8623....$$

nth iteration

$$x_{NSn} = 4.7173....$$

Regula Falsi Method

Regula Falsi method



Regula Falsi method

- Also known as false position and linear interpolation methods.
- The values of the function at the points $f(a_1)$ and $f(b_1)$ in the interval $[a_1, b_1]$ are connected with a straight line.
- The intersection with this line and the x-axis is the first x_{NS1}.
- In contrast to bisection method where the midpoint of the interval is taken as solution.
- Next interval is defined as either $[a_1, x_{NS1}]$ (a_1 is assigned to a_2 and x_{NS1} to b_2) or $[x_{NS1}, b_1]$ (x_{NS1} is assigned to a_2 and b_1 to b_2).
- For the third iteration, a new subinterval $[a_3, b_3]$ is selected and the iterations continue in the same fashion until accurate solution has been achieved.

Regula Falsi method

- For a given interval $[a, b]$ the equation of straight line that connects the two points $(b, f(b))$ and $(a, f(a))$ is given in Newton form by:

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

- The point x_{NS} where the line intersects the x-axis is determined by substituting $y = 0$ and solving for x:

$$x_{NS} = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$$

Regula Falsi method: Algorithm

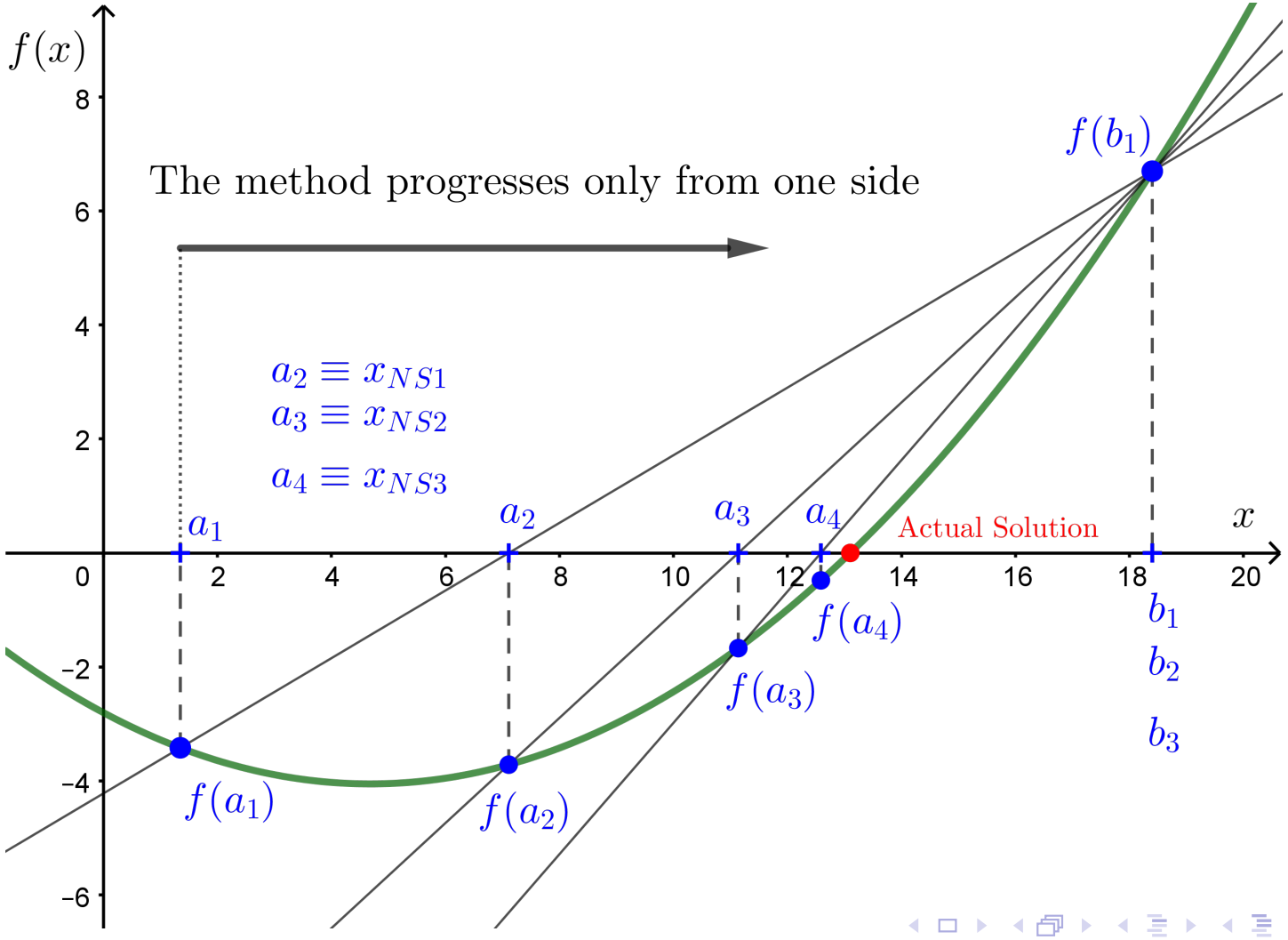
1. Select the interval so that a solution exists between points a and b. In other words, the $f(a)$ and $f(b)$ should have different signs: $f(a)f(b) < 0$. A plot can help determine the values a and b.
2. Calculate the first numerical estimation $x_{NS} = \frac{af(b)-bf(a)}{f(b)-f(a)}$
3. Check of the actual solution lies between a and x_{NS1} or x_{NS1} and b. If $f(a)f(x_{NS1}) < 0$, then the solution is between a and x_{NS1}, and if $f(a)f(x_{NS1}) > 0$, then the solution is between x_{NS1} and b.
4. The new interval (either between a and x_{NS1} or between x_{NS1} and b) is selected so that it contains the solution and is named $[a, b]$. Go back to step 2 and repeat between step 2 and step 4 until specified tolerance is satisfied.

Regula Falsi method: notes

- Method always converges as far as the solution is bracketed in the interval $[a, b]$.
- The numerical solution advances only from one side. The solution would be faster if the solution would advance from both sides.

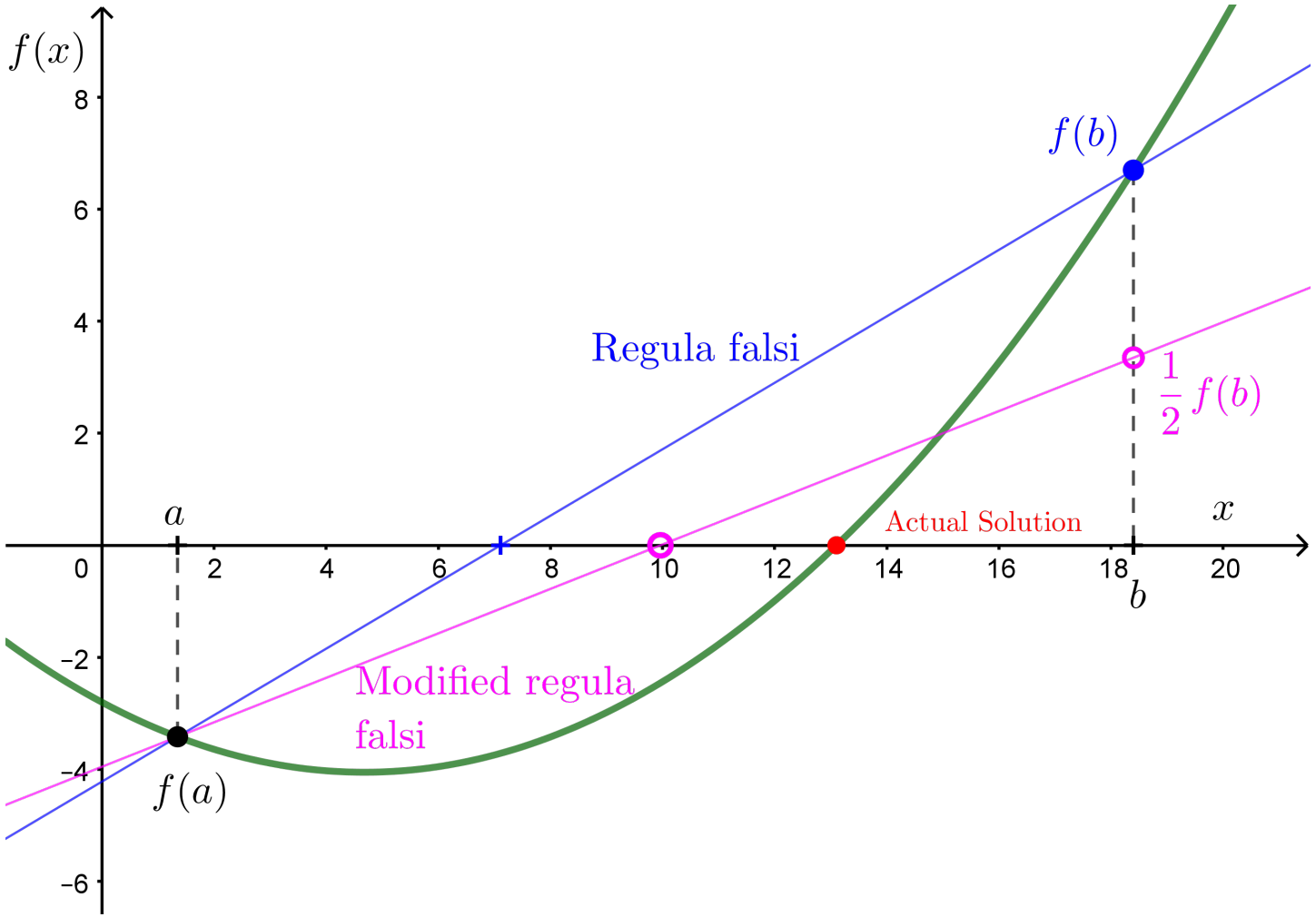
Regula Falsi method: notes

The numerical solution advances only from one side. The solution would be faster if the solution would advance from both sides (modifications have been proposed).

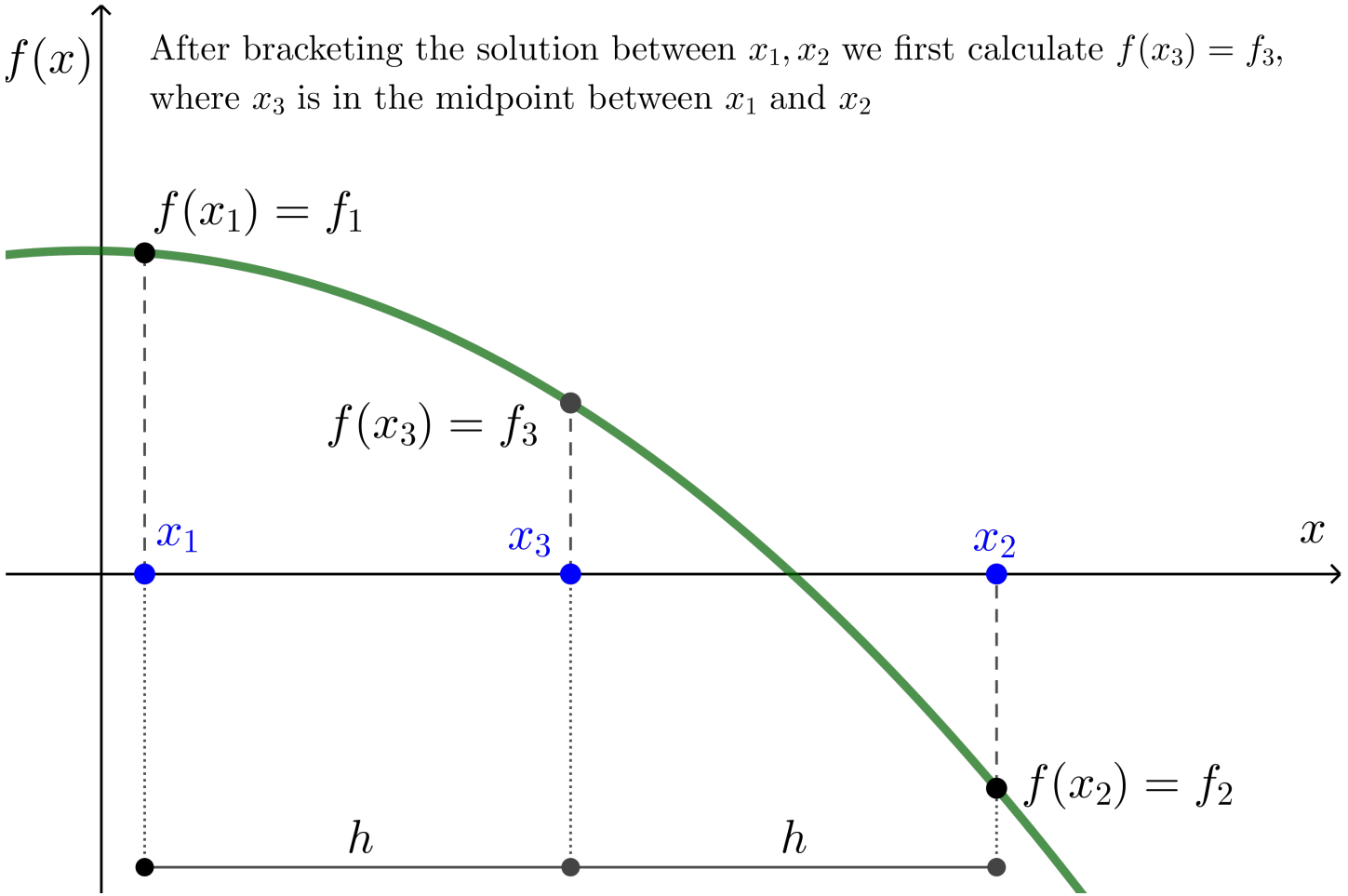


Regula Falsi method: Modification 1

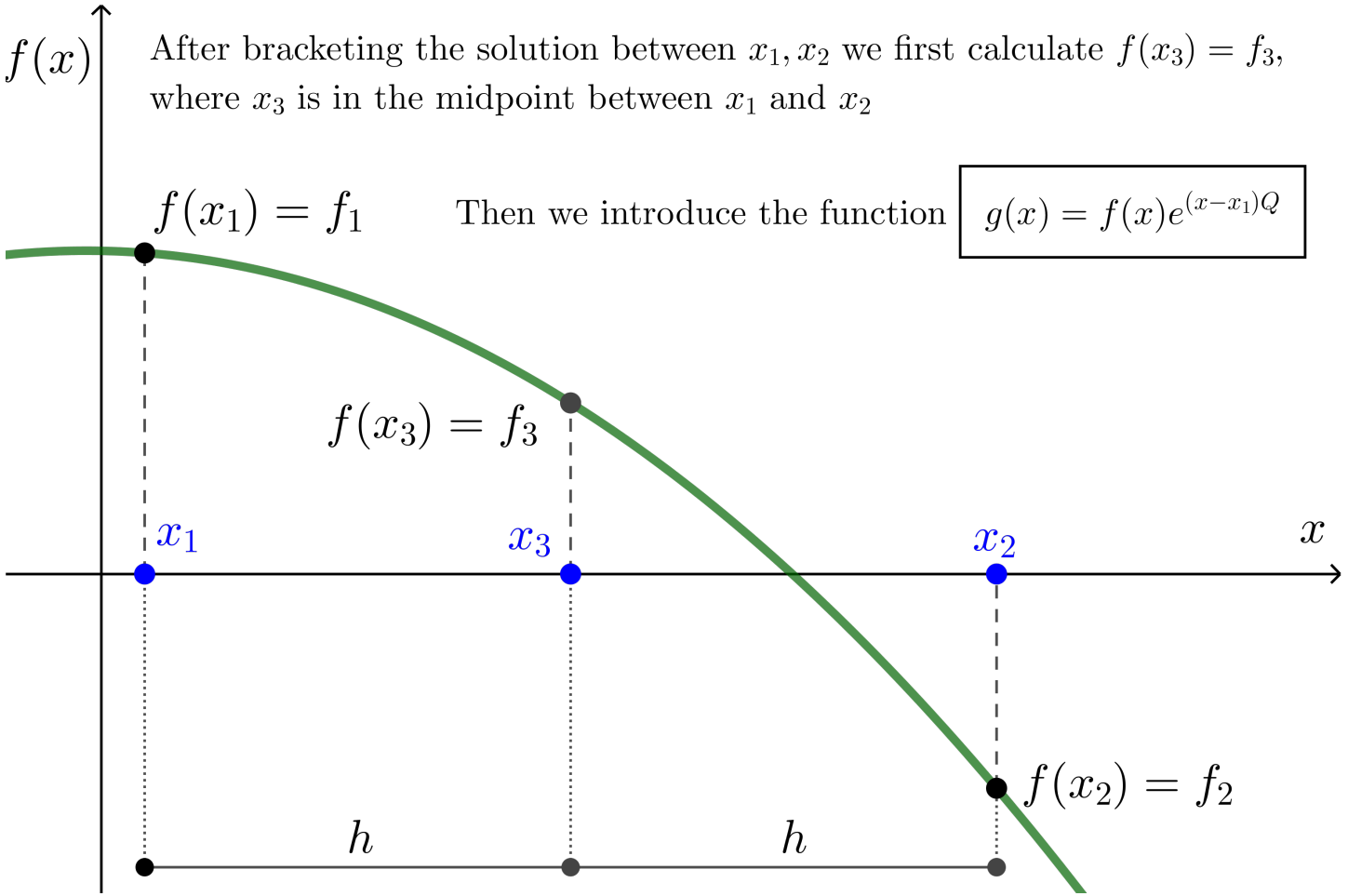
A smaller slope is implied in the straight line that connects the endpoints by dividing the value of the function at the endpoint that stays the same by 2.



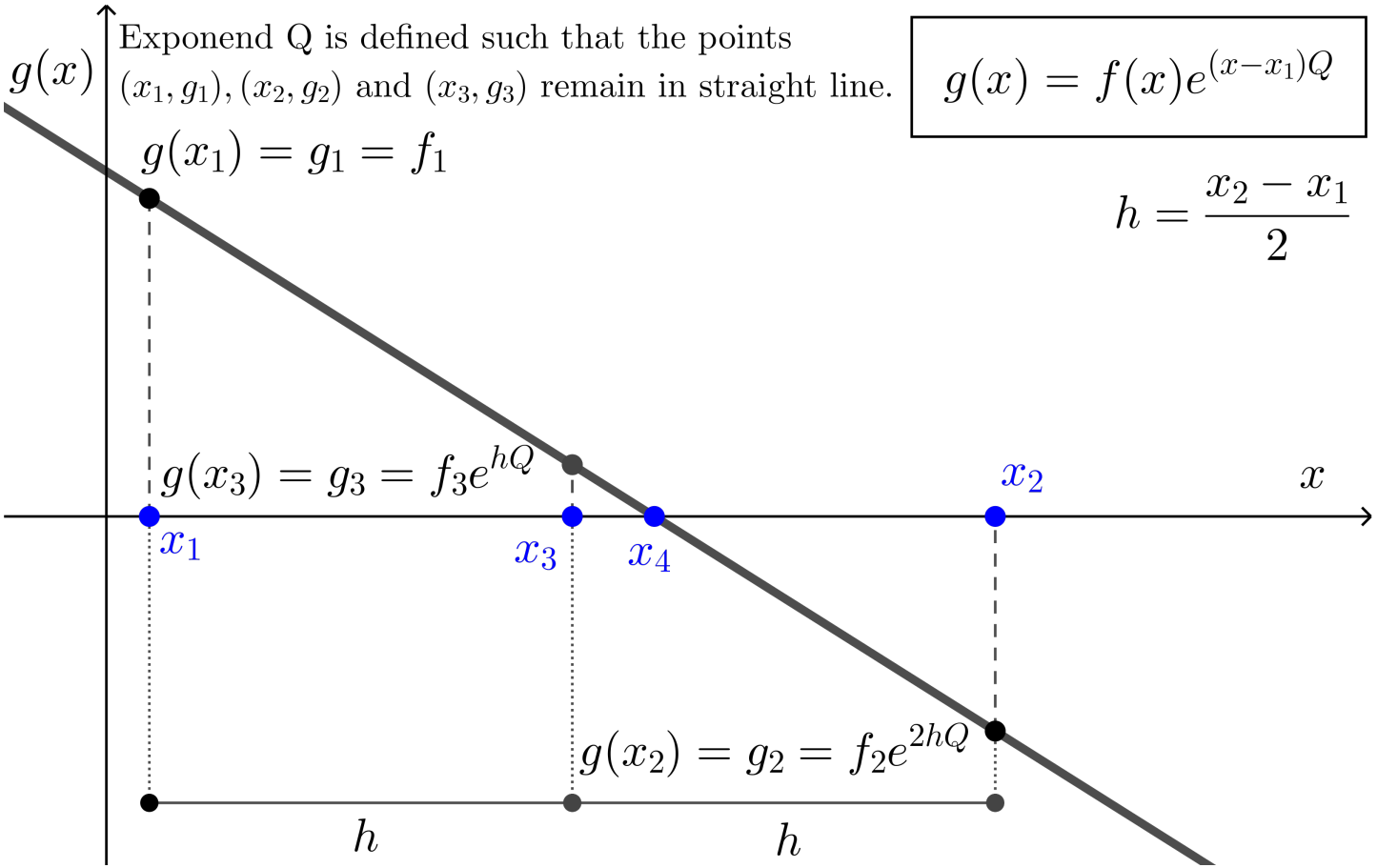
Regula Falsi method: Modification 2, Ridders method



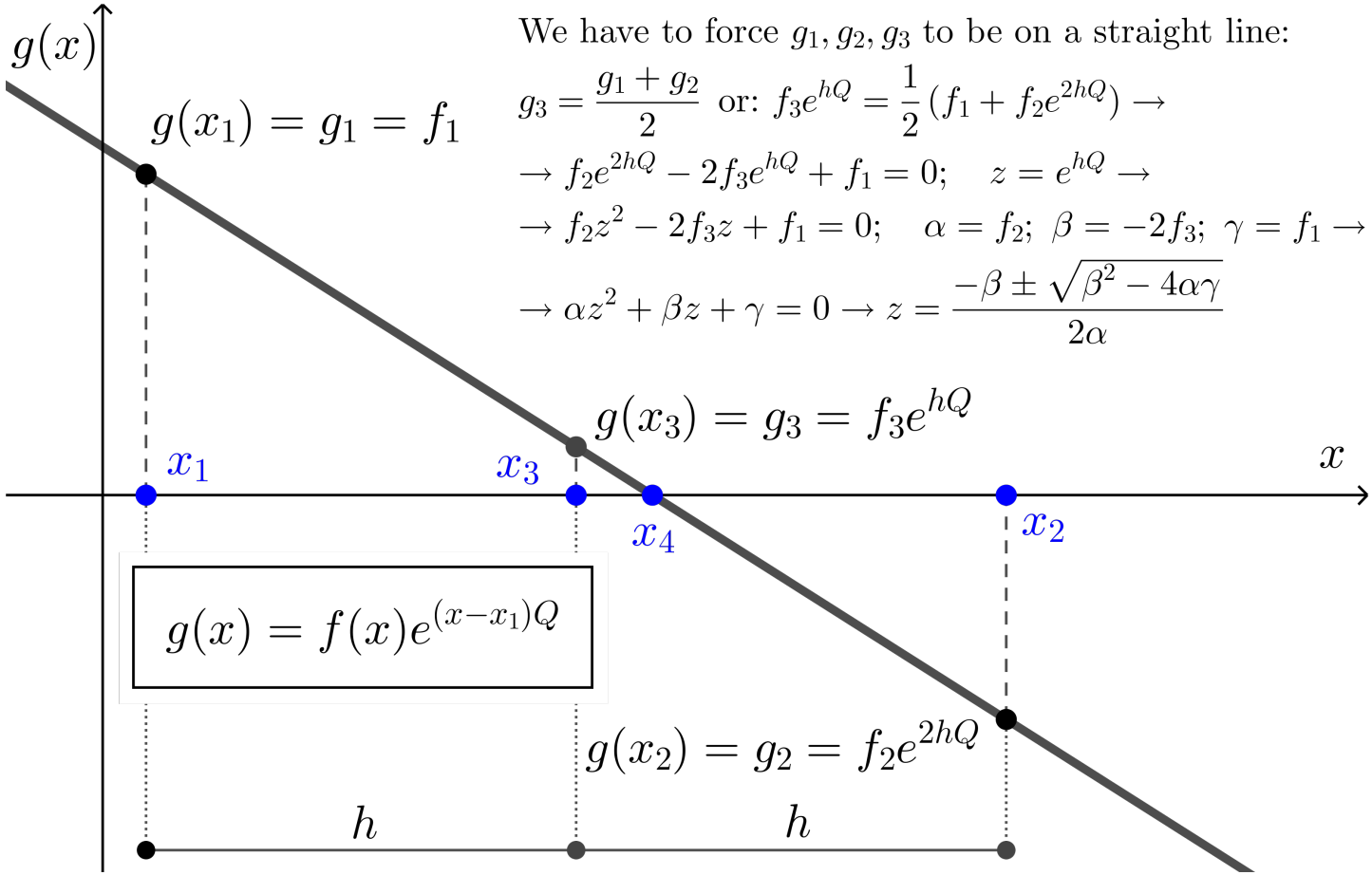
Regula Falsi method: Modification 2, Ridders method



Regula Falsi method: Modification 2, Ridders method



Regula Falsi method: Modification 2, Ridders method



We have to force g_1, g_2, g_3 to be on a straight line:

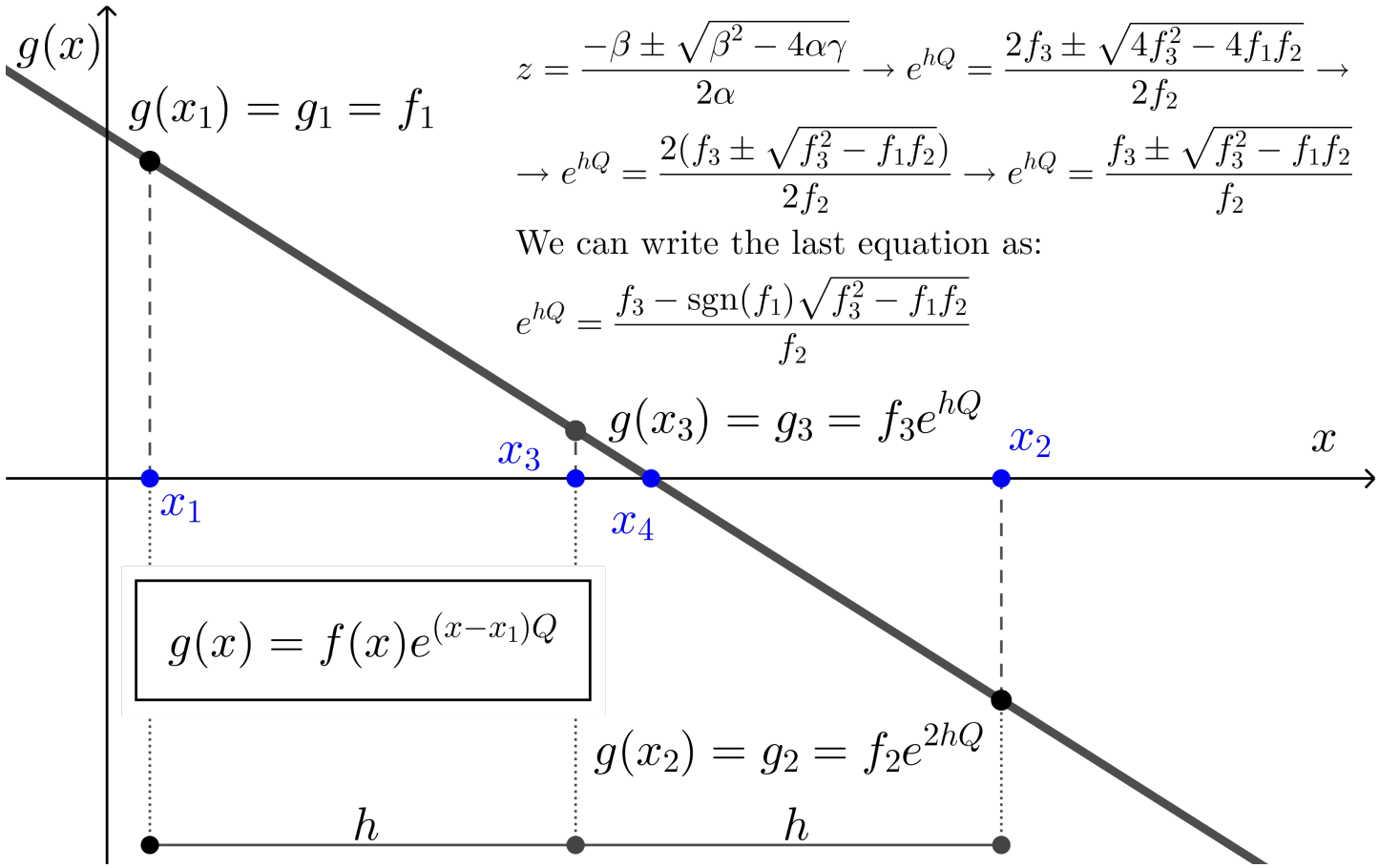
$$g_3 = \frac{g_1 + g_2}{2} \text{ or: } f_3 e^{hQ} = \frac{1}{2}(f_1 + f_2 e^{2hQ}) \rightarrow$$

$$\rightarrow f_2 e^{2hQ} - 2f_3 e^{hQ} + f_1 = 0; \quad z = e^{hQ} \rightarrow$$

$$\rightarrow f_2 z^2 - 2f_3 z + f_1 = 0; \quad \alpha = f_2;\ \beta = -2f_3;\ \gamma = f_1 \rightarrow$$

$$\rightarrow \alpha z^2 + \beta z + \gamma = 0 \rightarrow z = \frac{-\beta \pm \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha}$$

Regula Falsi method: Modification 2, Ridders method



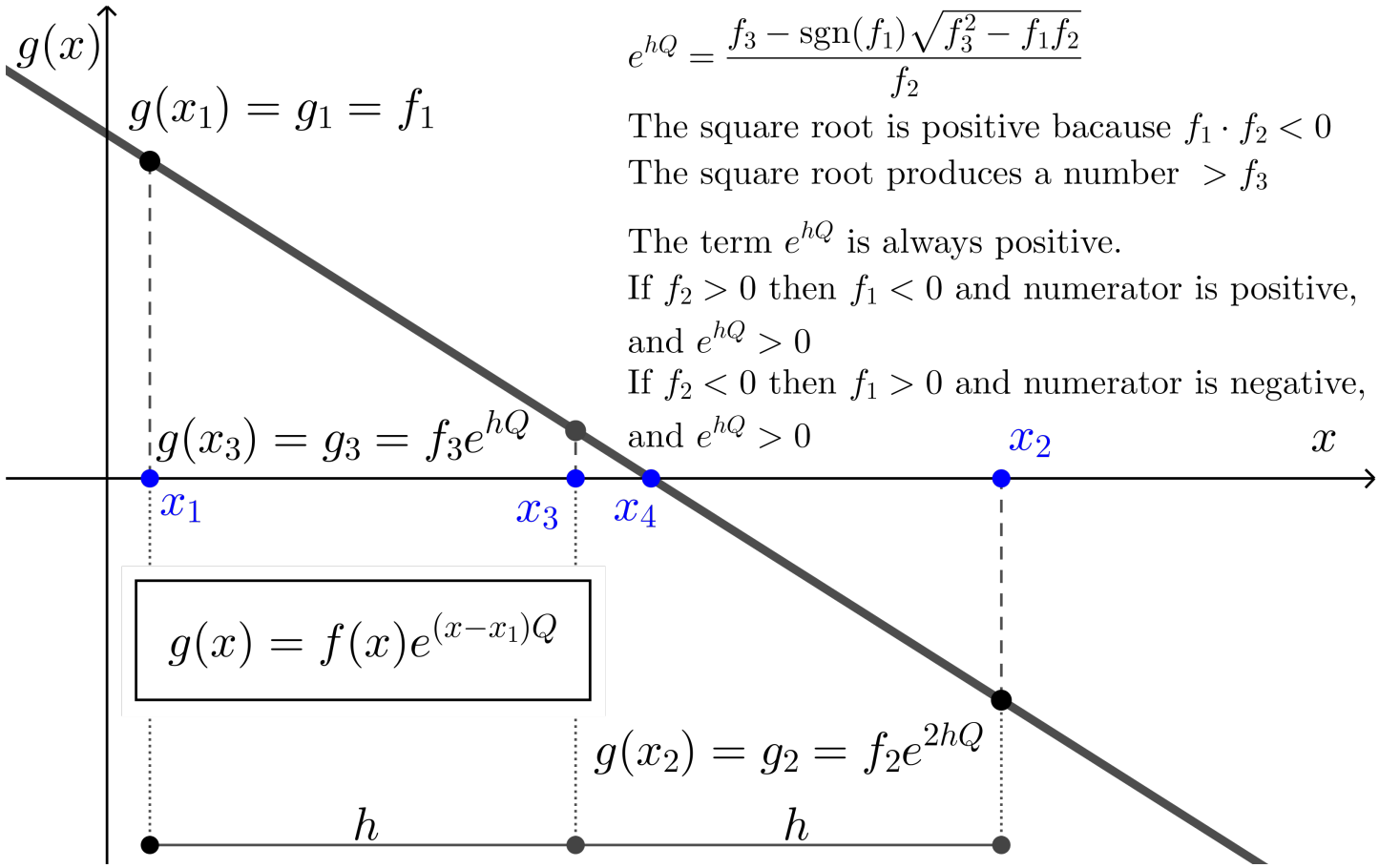
$$z = \frac{-\beta \pm \sqrt{\beta^2 - 4\alpha\gamma}}{2\alpha} \rightarrow e^{hQ} = \frac{2f_3 \pm \sqrt{4f_3^2 - 4f_1 f_2}}{2f_2} \rightarrow$$

$$\rightarrow e^{hQ} = \frac{2(f_3 \pm \sqrt{f_3^2 - f_1 f_2})}{2f_2} \rightarrow e^{hQ} = \frac{f_3 \pm \sqrt{f_3^2 - f_1 f_2}}{f_2}$$

We can write the last equation as:

$$e^{hQ} = \frac{f_3 - \mathrm{sgn}(f_1)\sqrt{f_3^2 - f_1 f_2}}{f_2}$$

Regula Falsi method: Modification 2, Ridders method



$$e^{hQ} = \frac{f_3 - \mathrm{sgn}(f_1)\sqrt{f_3^2 - f_1 f_2}}{f_2}$$

The square root is positive bacause $f_1 \cdot f_2 < 0$

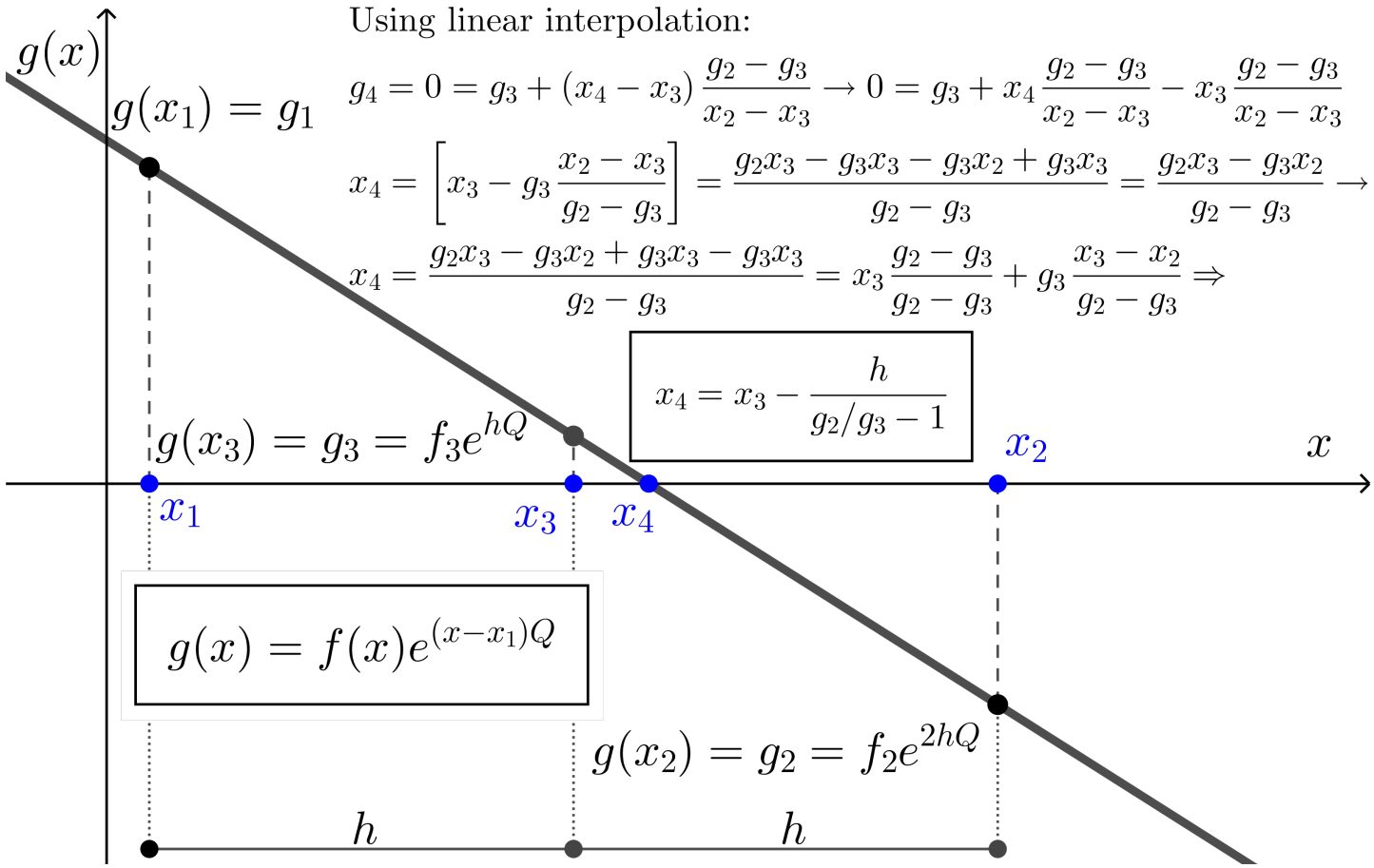
The square root produces a number $> f_3$

The term e^{hQ} is always positive.

If $f_2 > 0$ then $f_1 < 0$ and numerator is positive, and $e^{hQ} > 0$

If $f_2 < 0$ then $f_1 > 0$ and numerator is negative, and $e^{hQ} > 0$

Regula Falsi method: Modification 2, Ridders method



Using linear interpolation:

$$g_4 = 0 = g_3 + (x_4 - x_3)\frac{g_2 - g_3}{x_2 - x_3} \rightarrow 0 = g_3 + x_4\frac{g_2 - g_3}{x_2 - x_3} - x_3\frac{g_2 - g_3}{x_2 - x_3}$$

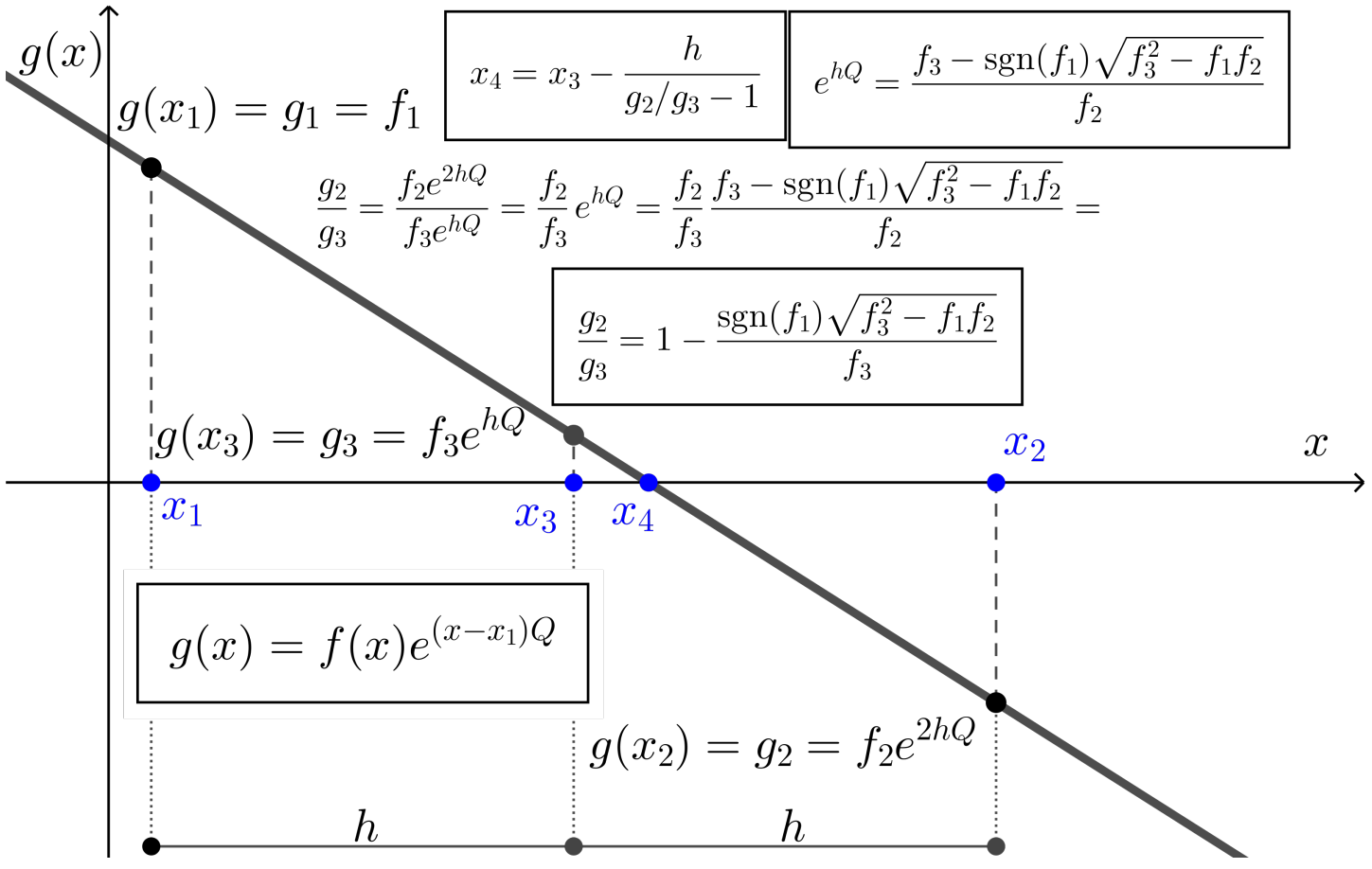
$$x_4 = \left[x_3 - g_3\frac{x_2 - x_3}{g_2 - g_3}\right] = \frac{g_2x_3 - g_3x_3 - g_3x_2 + g_3x_3}{g_2 - g_3} = \frac{g_2x_3 - g_3x_2}{g_2 - g_3} \rightarrow$$

$$x_4 = \frac{g_2x_3 - g_3x_2 + g_3x_3 - g_3x_3}{g_2 - g_3} = x_3\frac{g_2 - g_3}{g_2 - g_3} + g_3\frac{x_3 - x_2}{g_2 - g_3} \Rightarrow$$

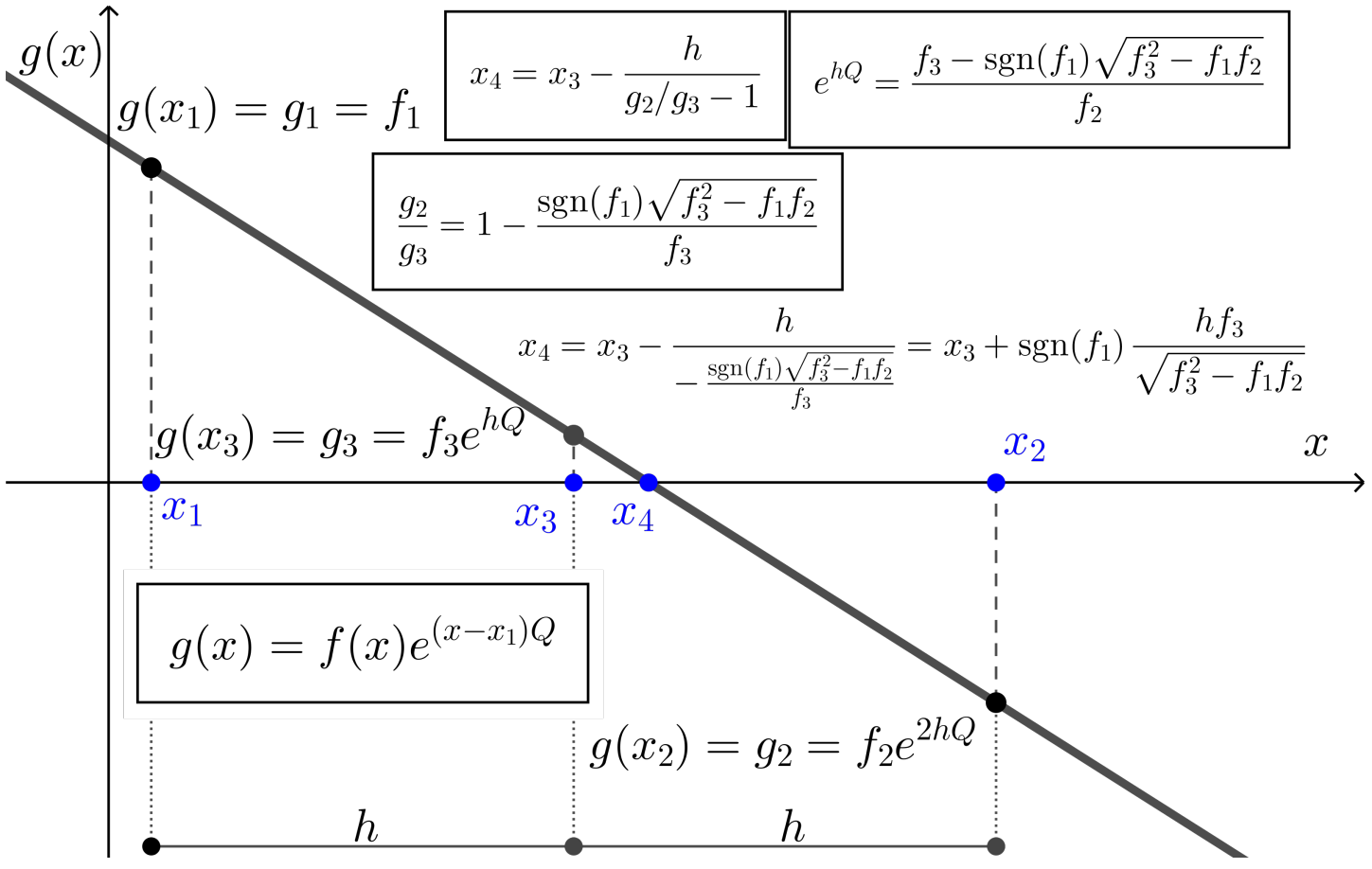
$$\boxed{x_4 = x_3 - \frac{h}{g_2/g_3 - 1}}$$

$$\boxed{g(x) = f(x)e^{(x-x_1)Q}}$$

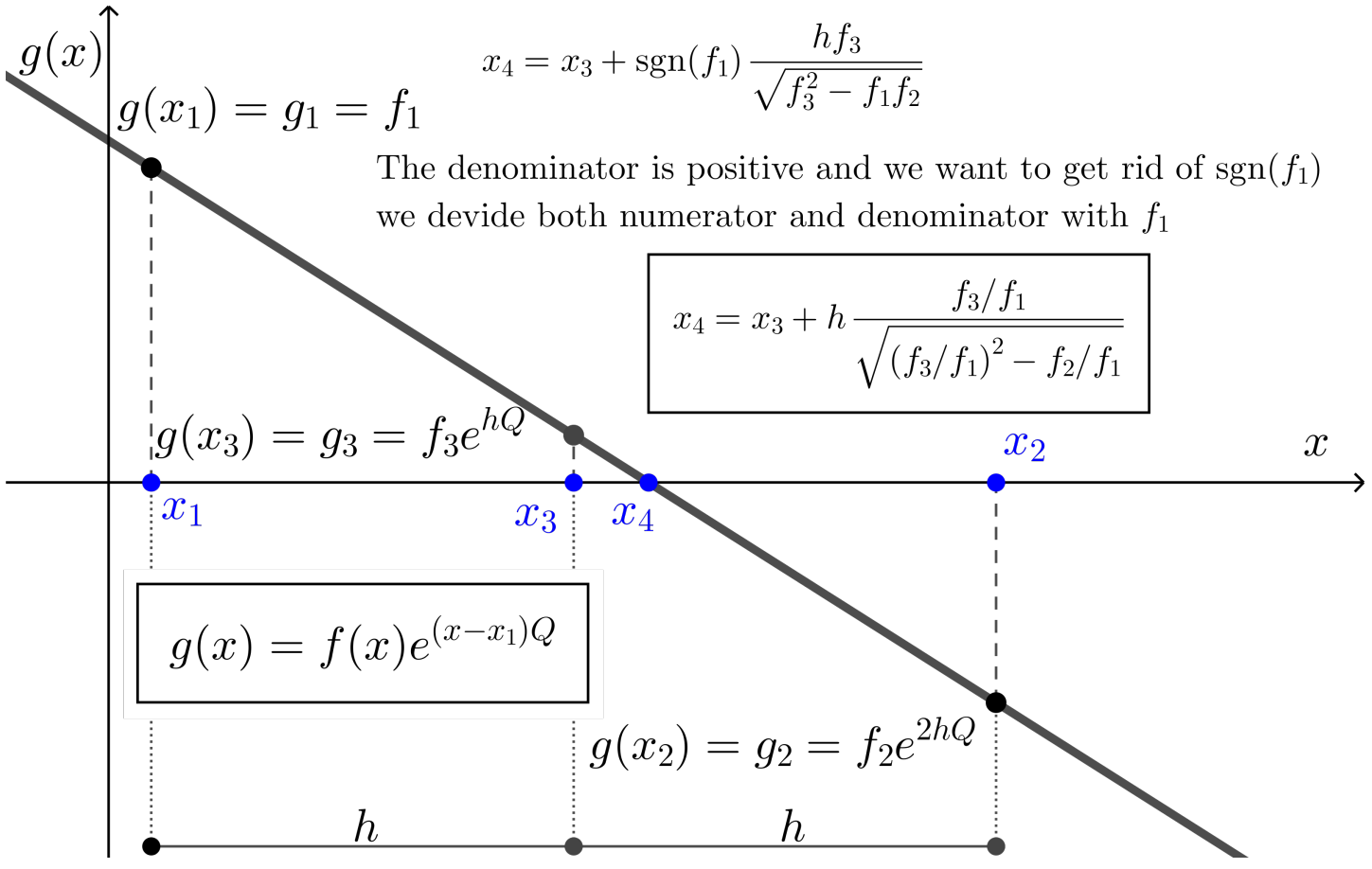
Regula Falsi method: Modification 2, Ridders method



Regula Falsi method: Modification 2, Ridders method



Regula Falsi method: Modification 2, Ridders method



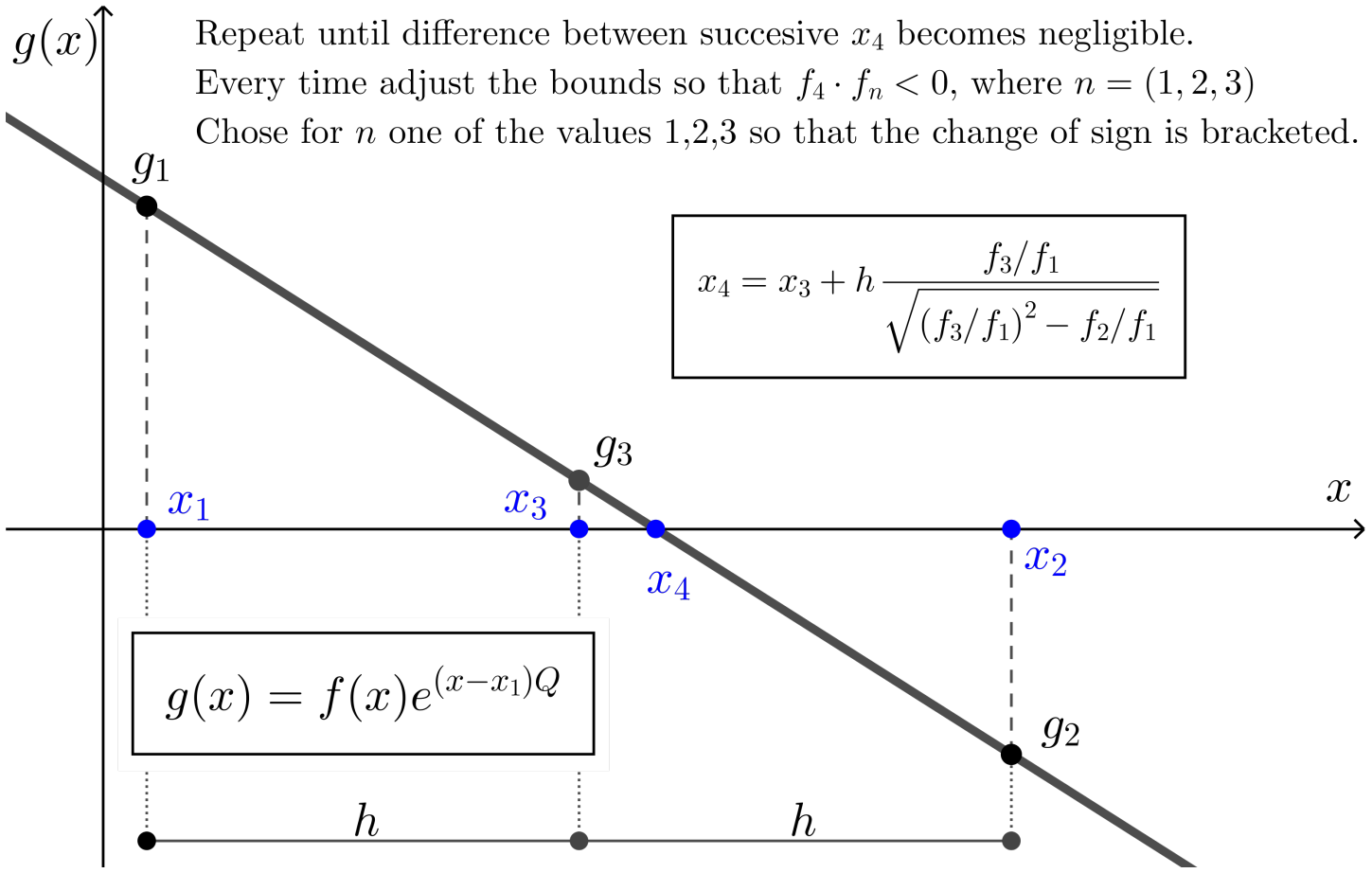
$$x_4 = x_3 + \operatorname{sgn}(f_1)\frac{hf_3}{\sqrt{f_3^2 - f_1 f_2}}$$

The denominator is positive and we want to get rid of $\operatorname{sgn}(f_1)$ we devide both numerator and denominator with f_1

$$x_4 = x_3 + h\frac{f_3/f_1}{\sqrt{(f_3/f_1)^2 - f_2/f_1}}$$

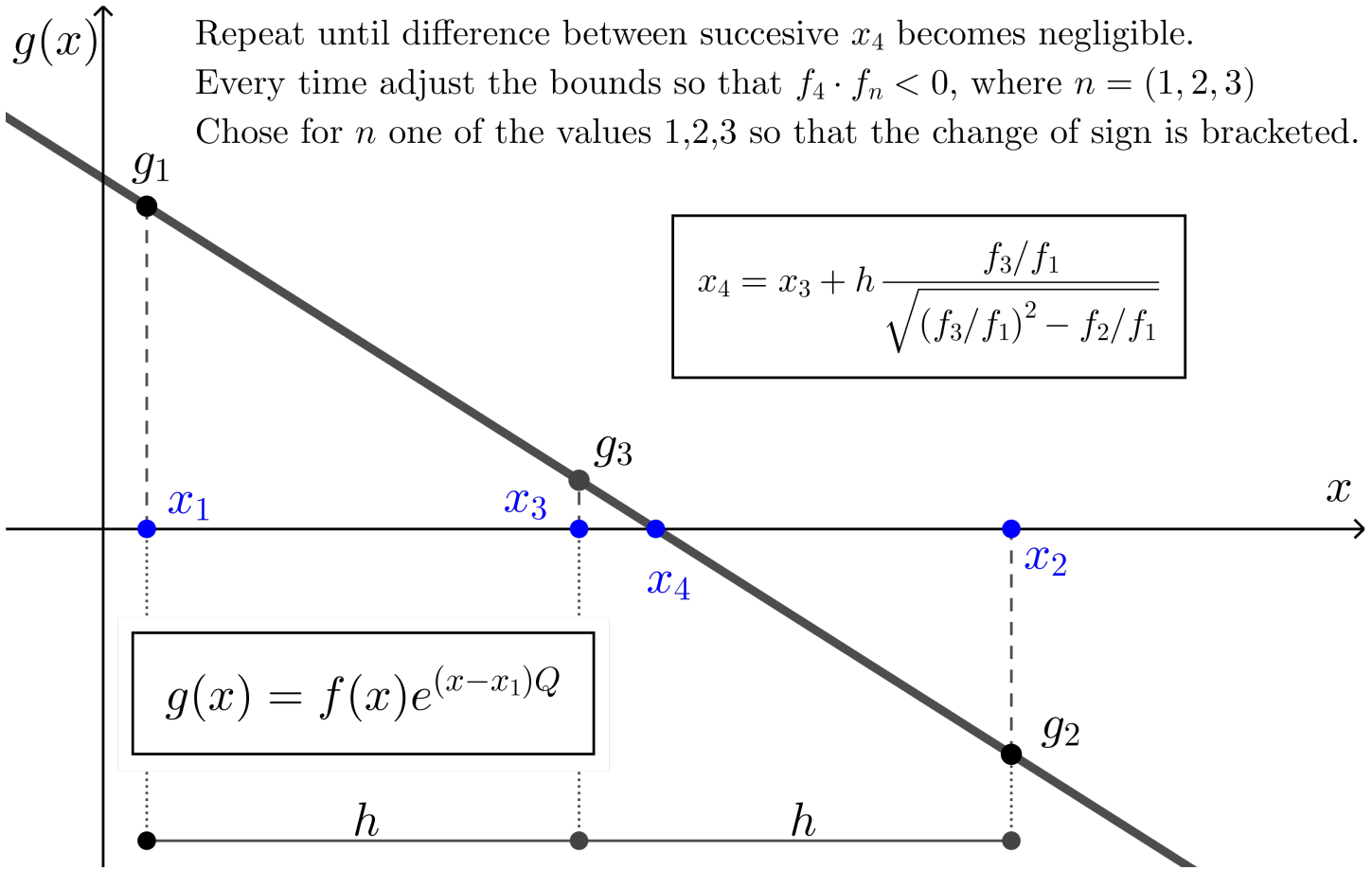
$$g(x) = f(x)e^{(x-x_1)Q}$$

Regula Falsi method: Modification 2, Ridders method



Regula Falsi method: Ridders method, Example

Find the root of the function $f(x) = x^3 - 10 \cdot x^2 + 5$ between $(0.6, 0.8)$ using the Ridders method.



Regula Falsi method: Ridders method, Example

Find the root of the function $f(x) = x^3 - 10 \cdot x^2 + 5$ between $(0.6, 0.8)$ using the Ridders method.
Solution
1st iteration:

$$x_1 = 0.6; \rightarrow f_1 = f(x_1) = 1.616$$
$$x_2 = 0.8; \rightarrow f_2 = f(x_2) = -0.888$$
$$x_3 = 0.7; \rightarrow f_3 = f(x_3) = 0.443$$

Then we can apply to the formula ($h = 0.1$):

$$x_4 = x_3 + h\frac{f_3/f_1}{\sqrt{(f_3/f_1)^2 - f_2/f_1}}$$
$$x_4 = 0.7 + 0.1\frac{0.443/1.616}{\sqrt{(0.443/1.616)^2 - (-0.888)/1.616}}$$
$$x_4 = 0.7346850665; \rightarrow f_4 = -0.001066281165$$

Regula Falsi method: Ridders method, Example

Find the root of the function $f(x) = x^3 - 10 \cdot x^2 + 5$ between $(0.6, 0.8)$ using the Ridders method.

Solution

We bracket the solution between $[x_3, x_4]$ from 1st iteration.

2nd iteration:

$$x_1 = 0.7; \rightarrow f_1 = f(x_1) = 0.443$$
$$x_2 = 0.734685066; \rightarrow f_2 = f(x_2) = -0.001066281165$$
$$x_3 = 0.7173425333; \rightarrow f_3 = f(x_3) = 0.2233272428$$

Then we can apply to the formula ($h = 0.01734253327$):

$$x_4 = x_3 + h\frac{f_3/f_1}{\sqrt{(f_3/f_1)^2 - f_2/f_1}}$$

$$x_4 = 0.7173425333 + 0.01734253327\frac{0.2233272428/0.443}{\sqrt{(0.2233272428/0.443)^2 - (-0.001066281165)/0.443}}$$

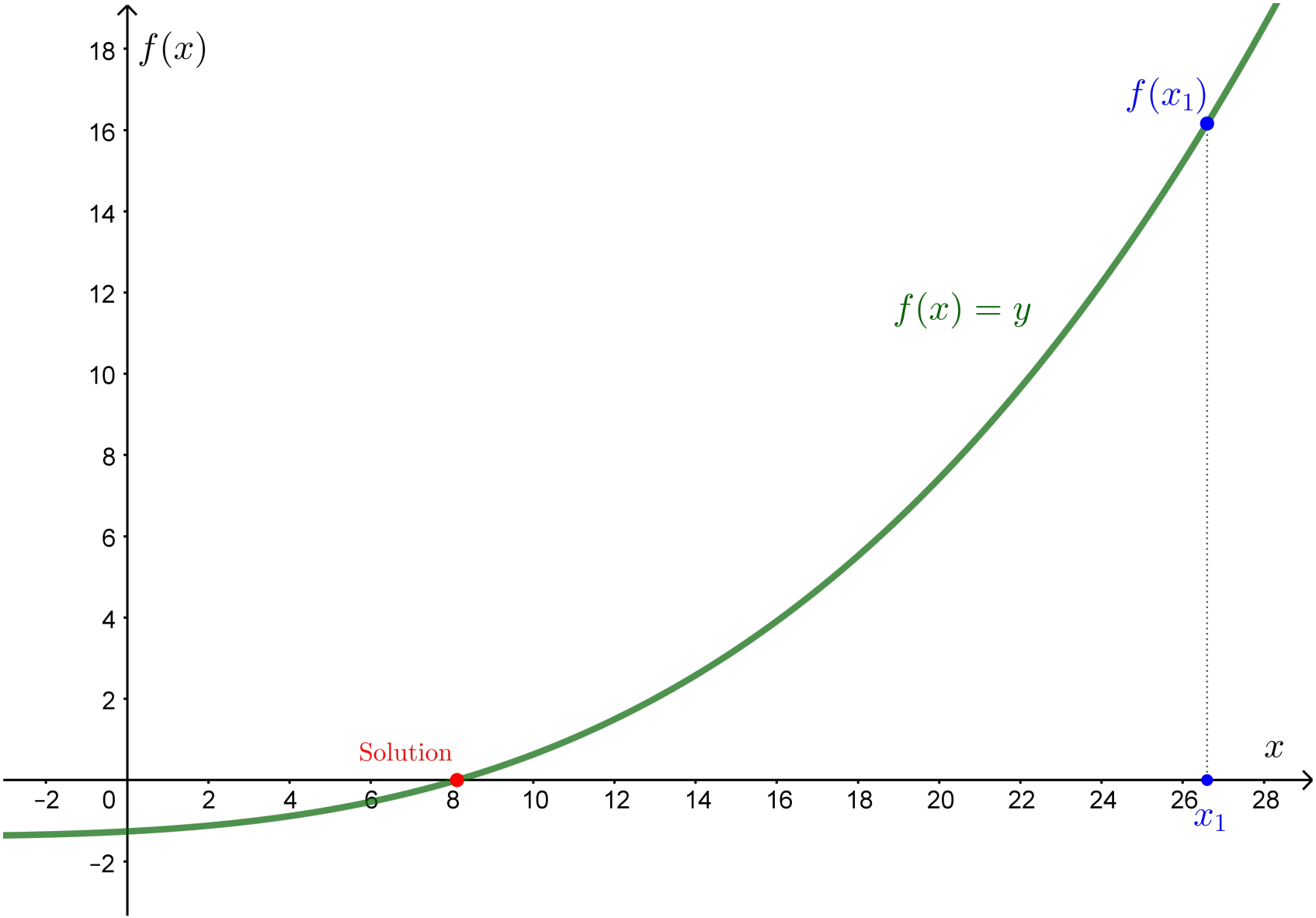
$$x_4 = 0.7346035205; \rightarrow f_4 = -0.00000016586386$$

We could continue, but solution is approximately $x \approx 0.7346$

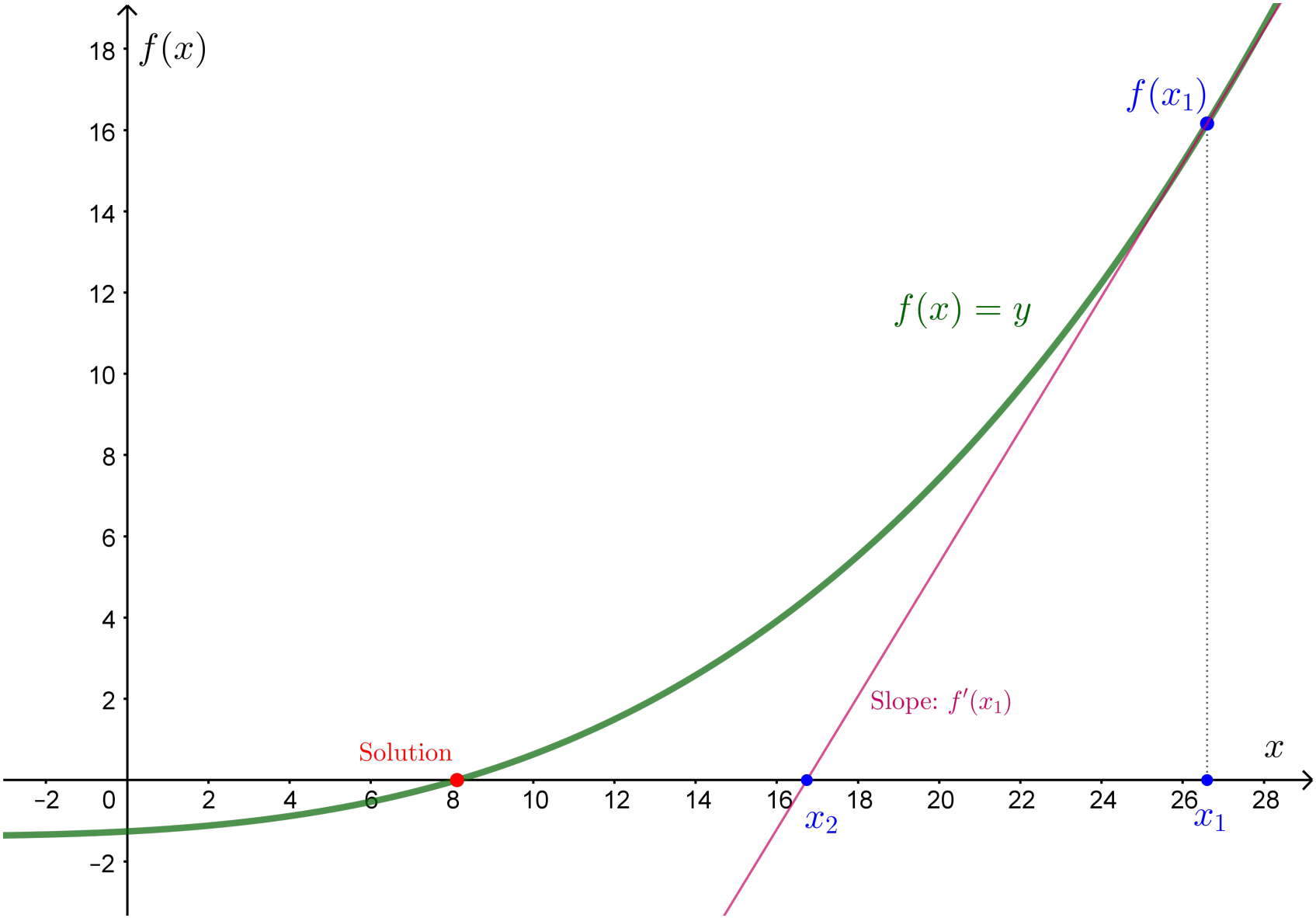
Open methods

Newton-Raphson Method

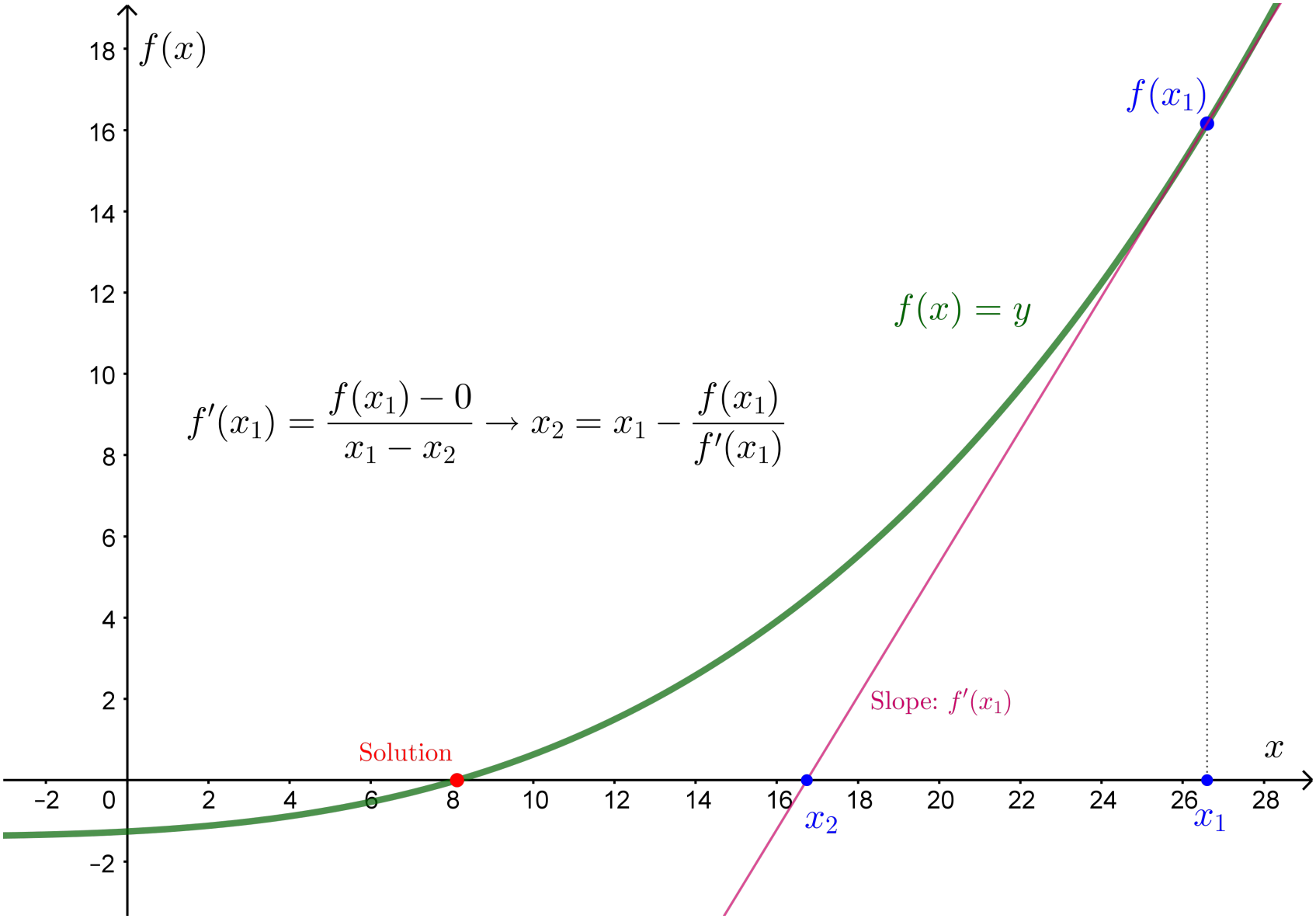
Newton-Raphson Method



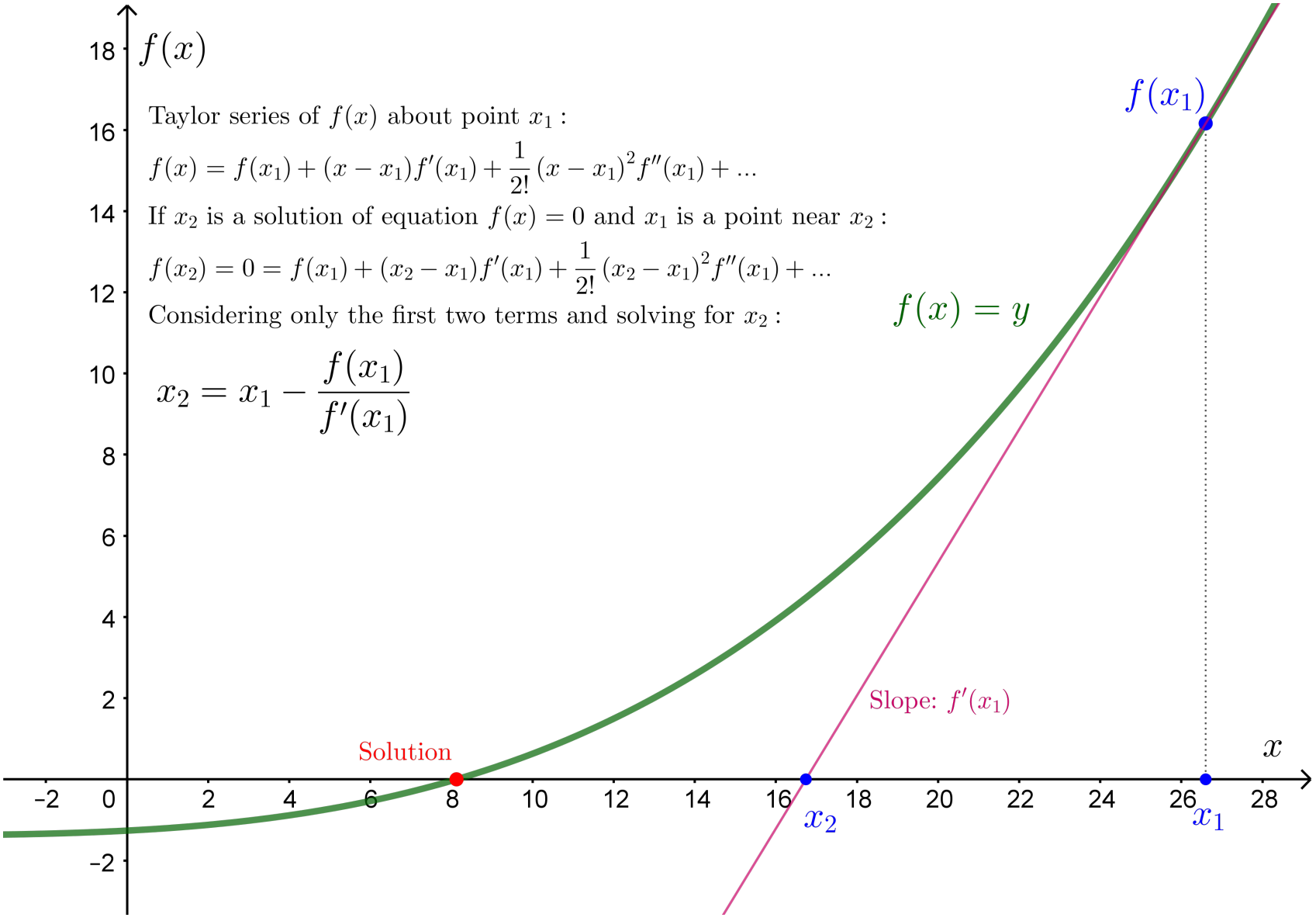
Newton-Raphson Method



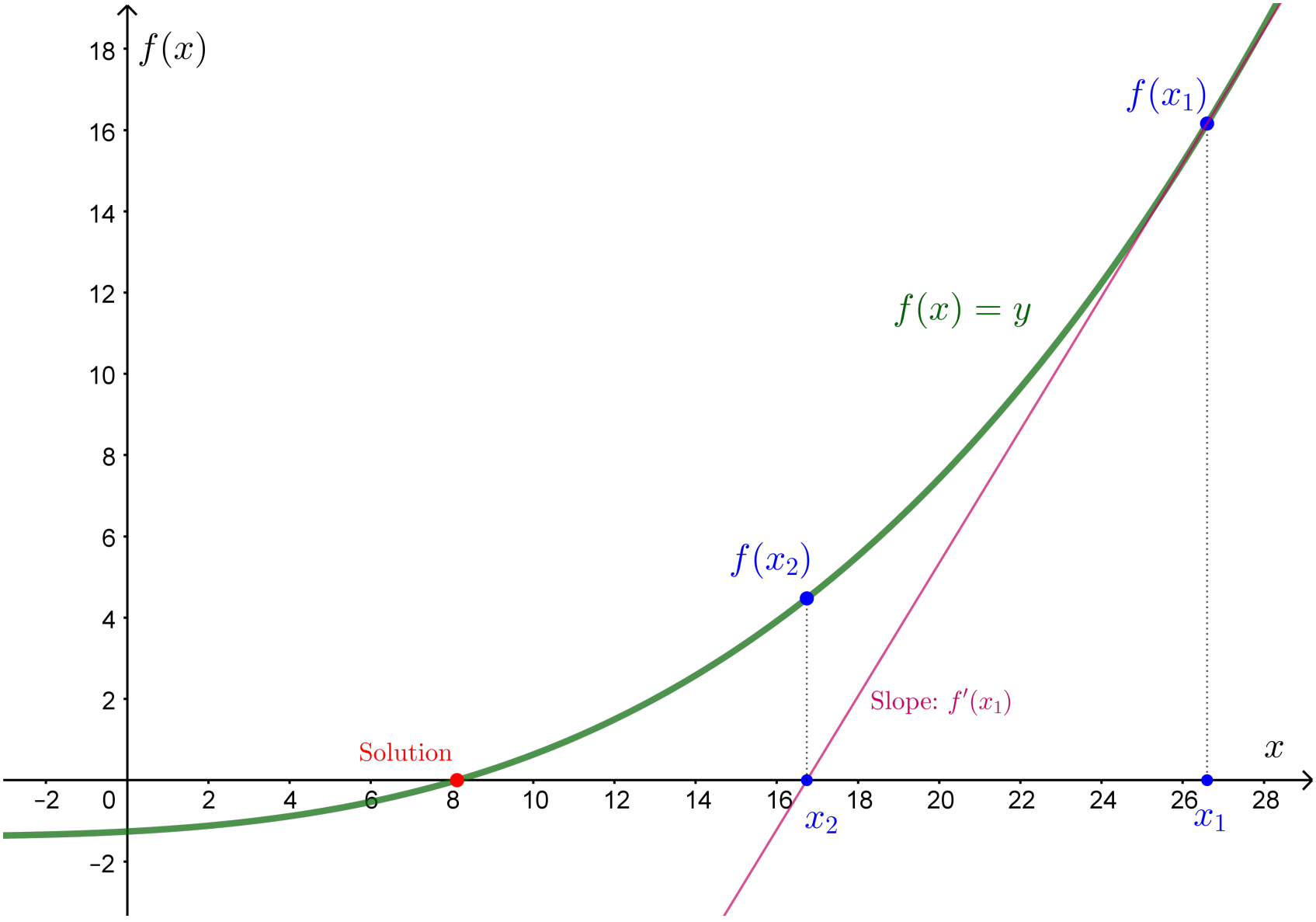
Newton-Raphson Method



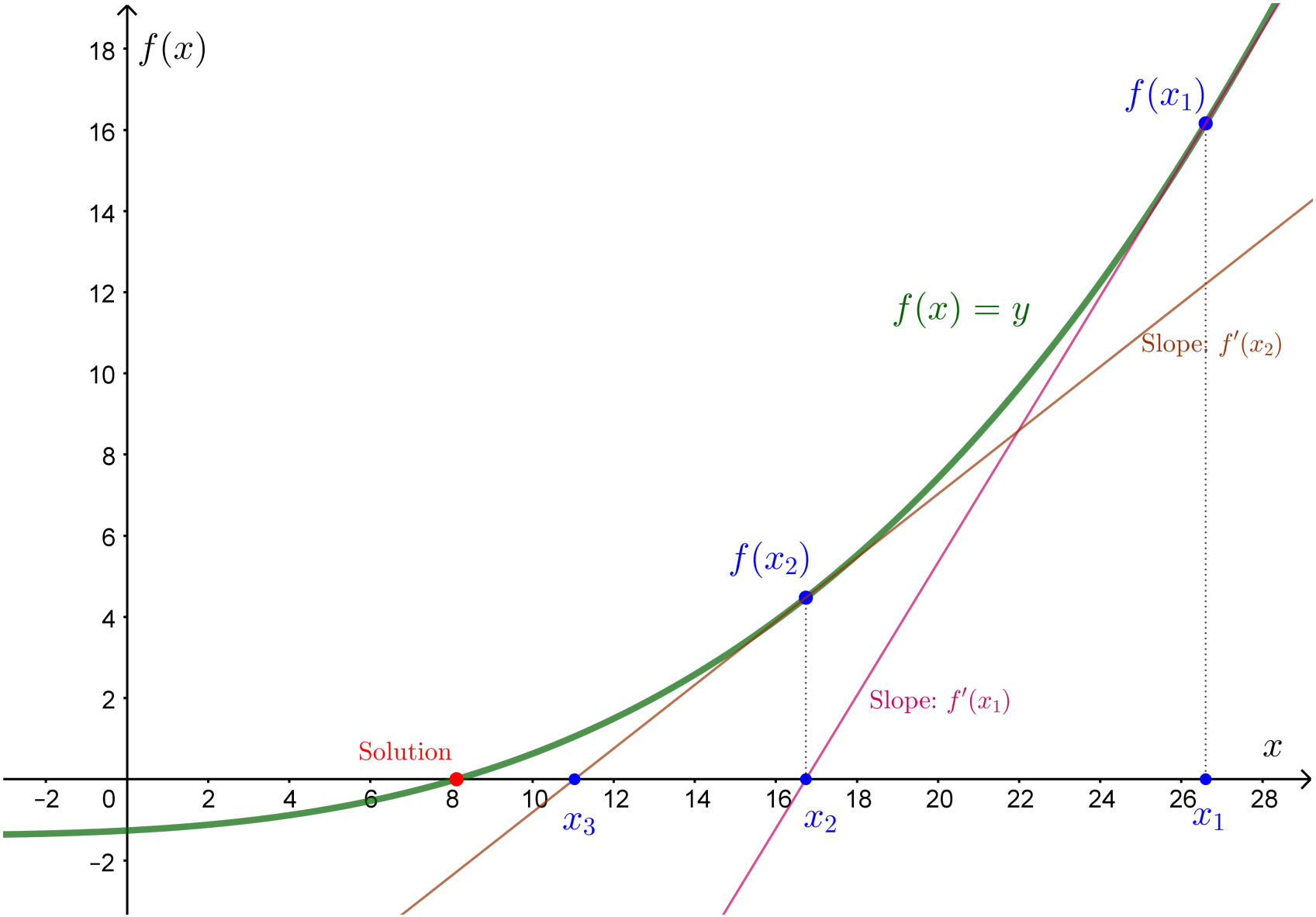
Newton-Raphson Method



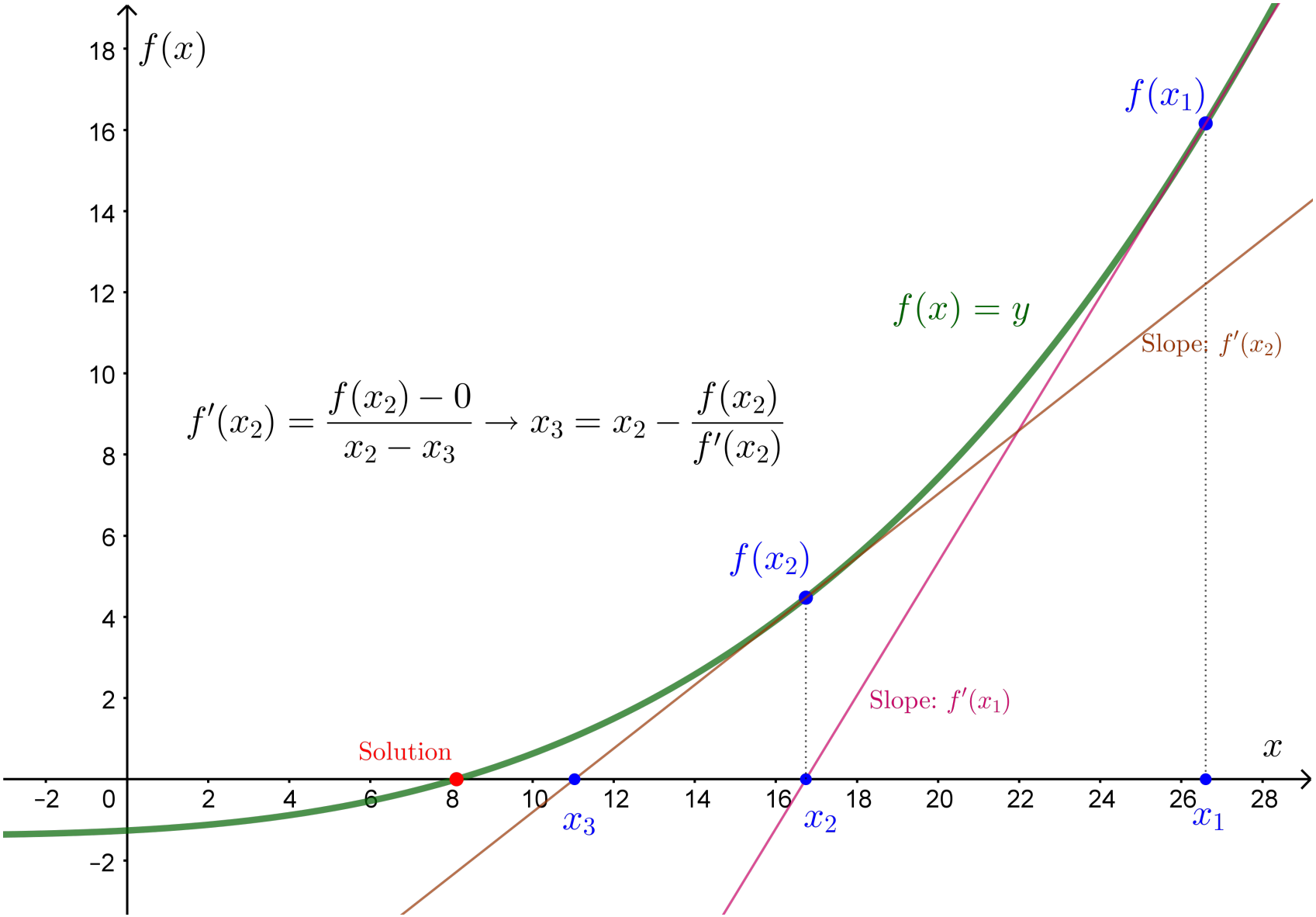
Newton-Raphson Method



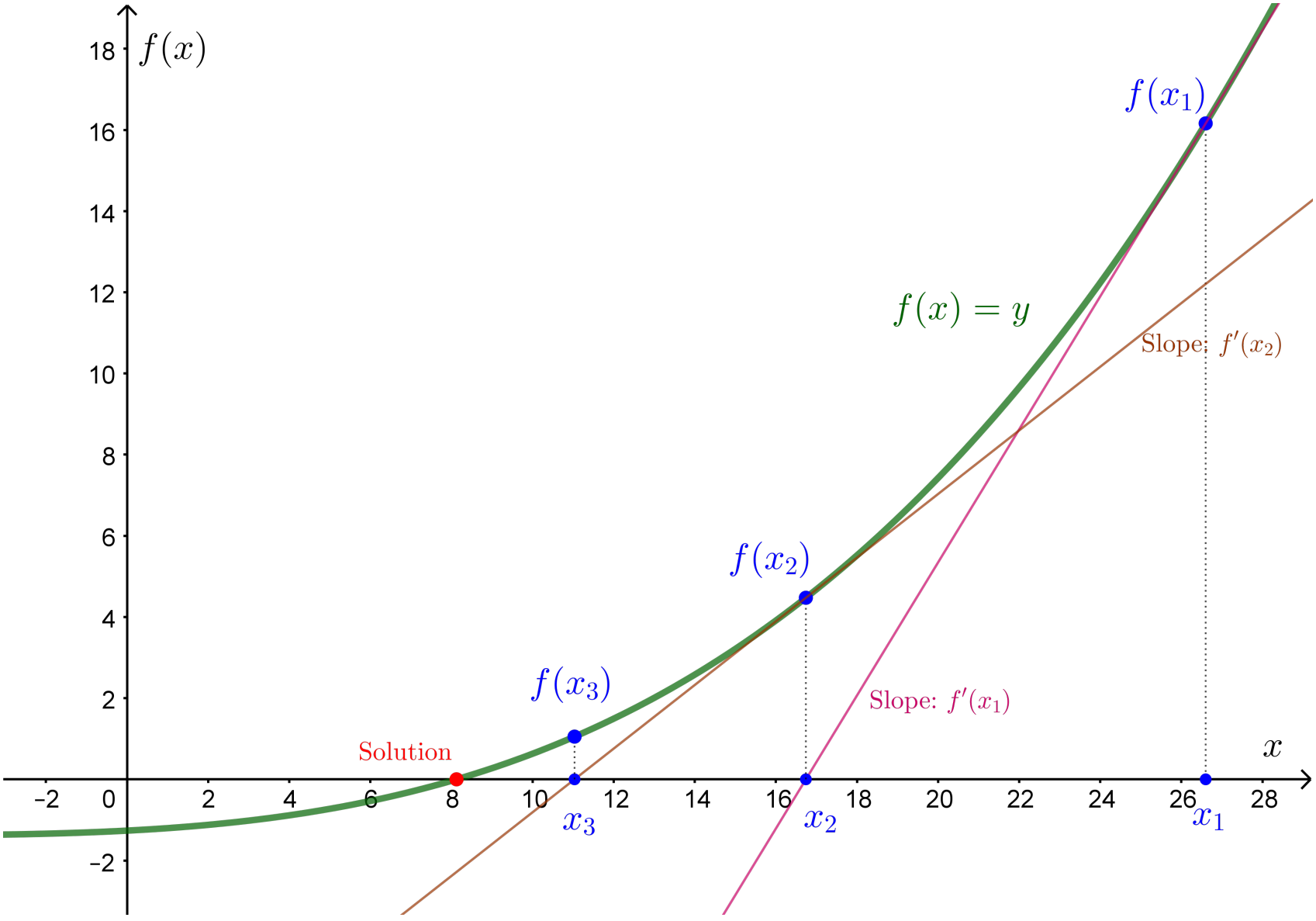
Newton-Raphson Method



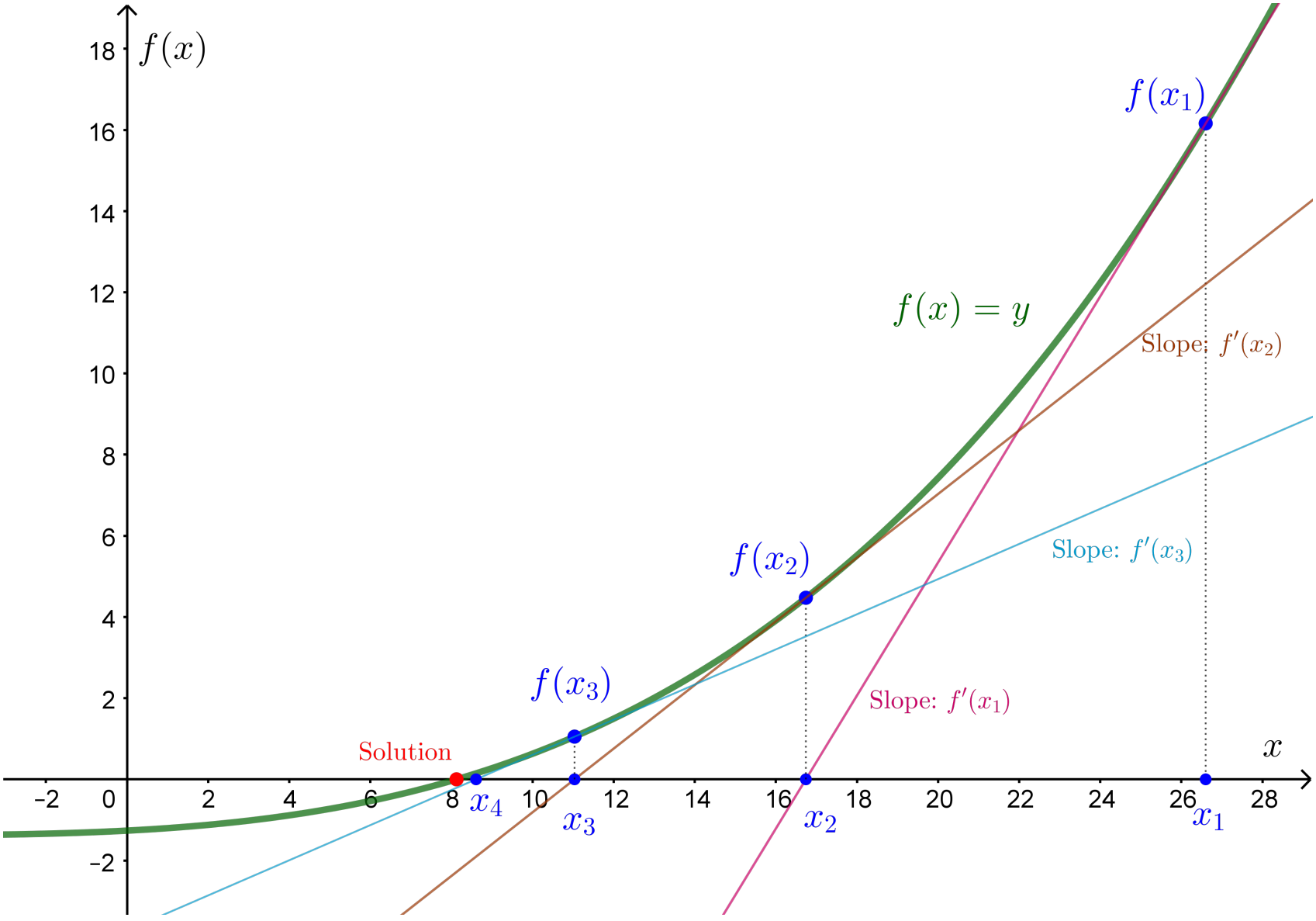
Newton-Raphson Method



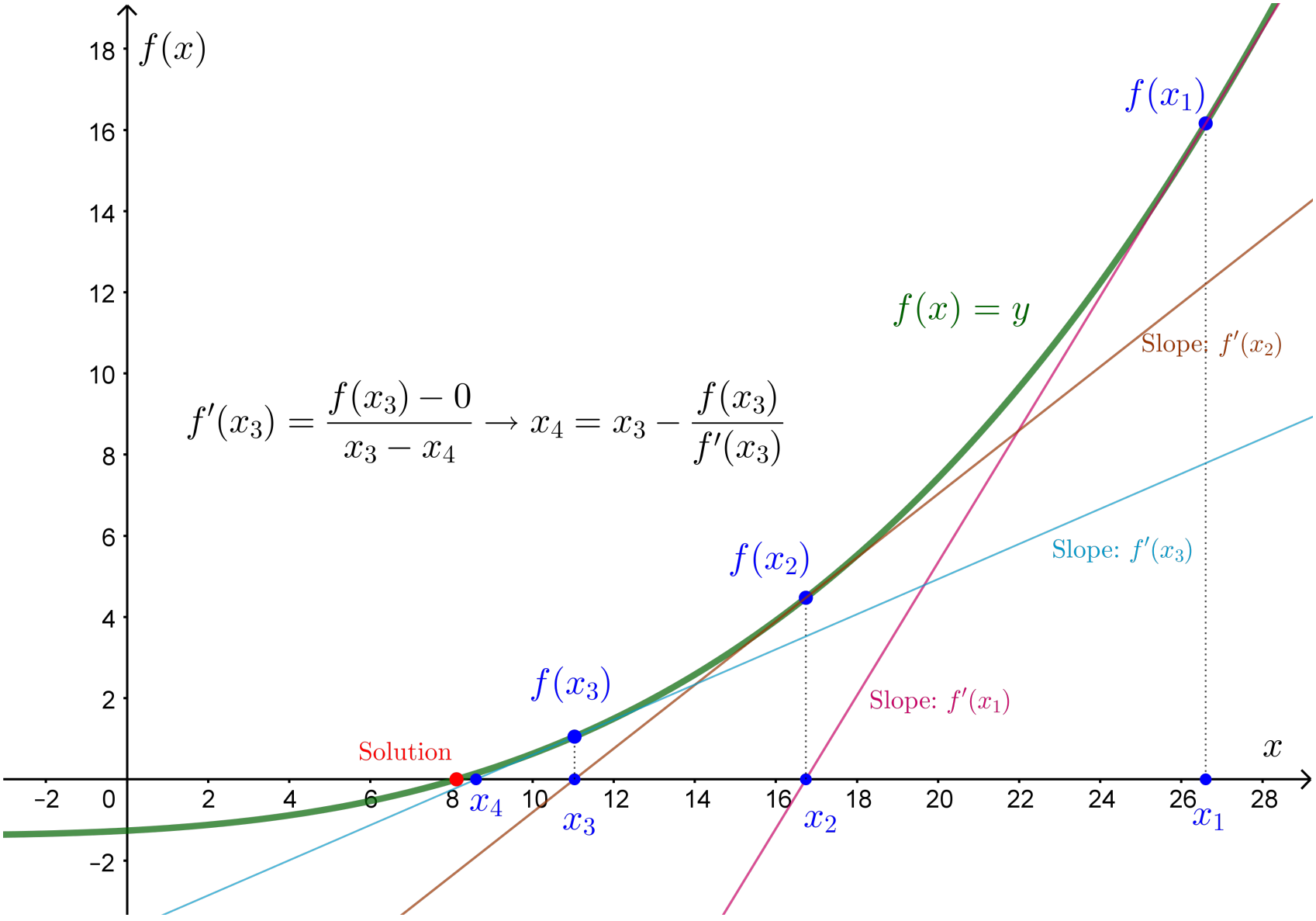
Newton-Raphson Method



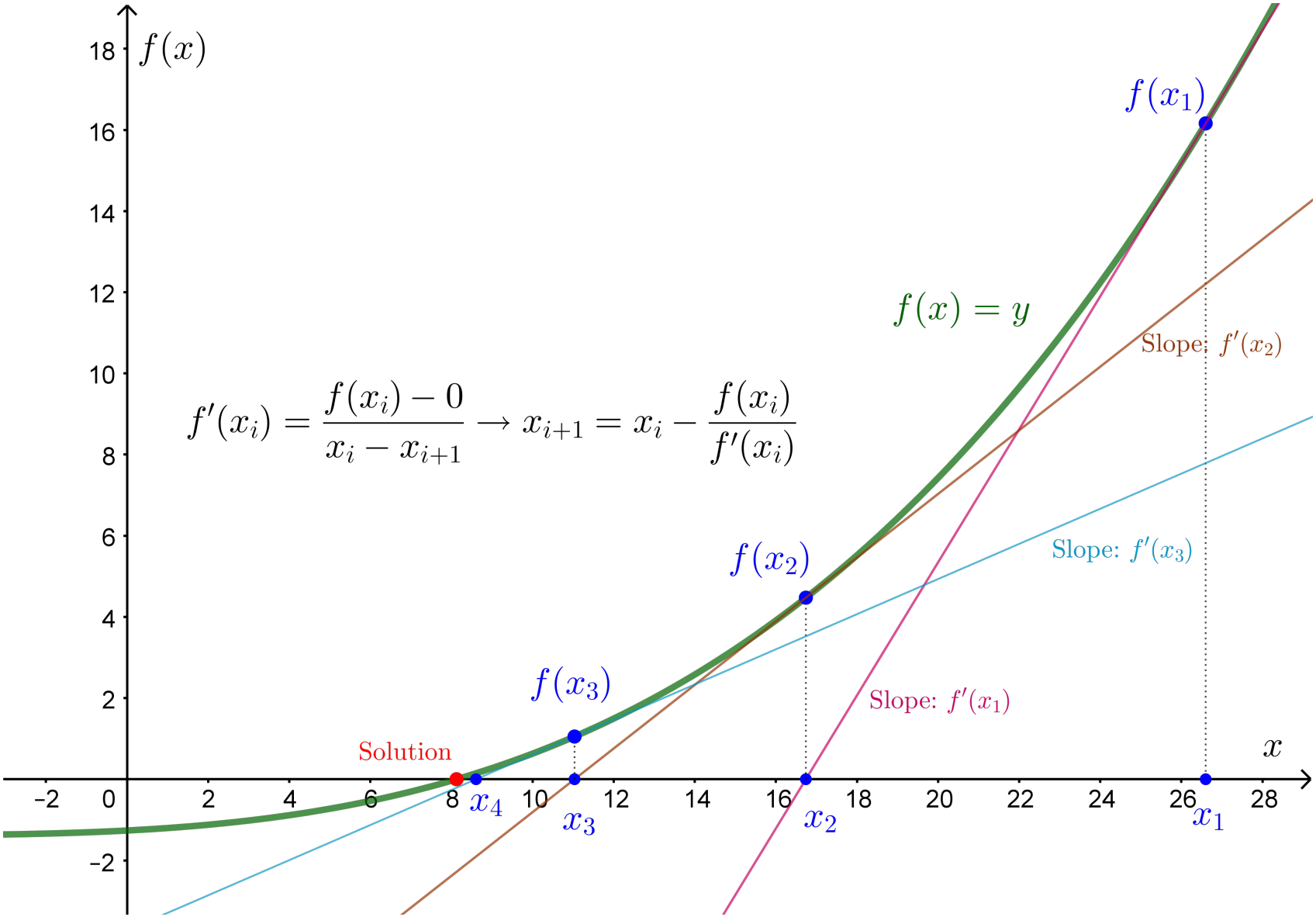
Newton-Raphson Method



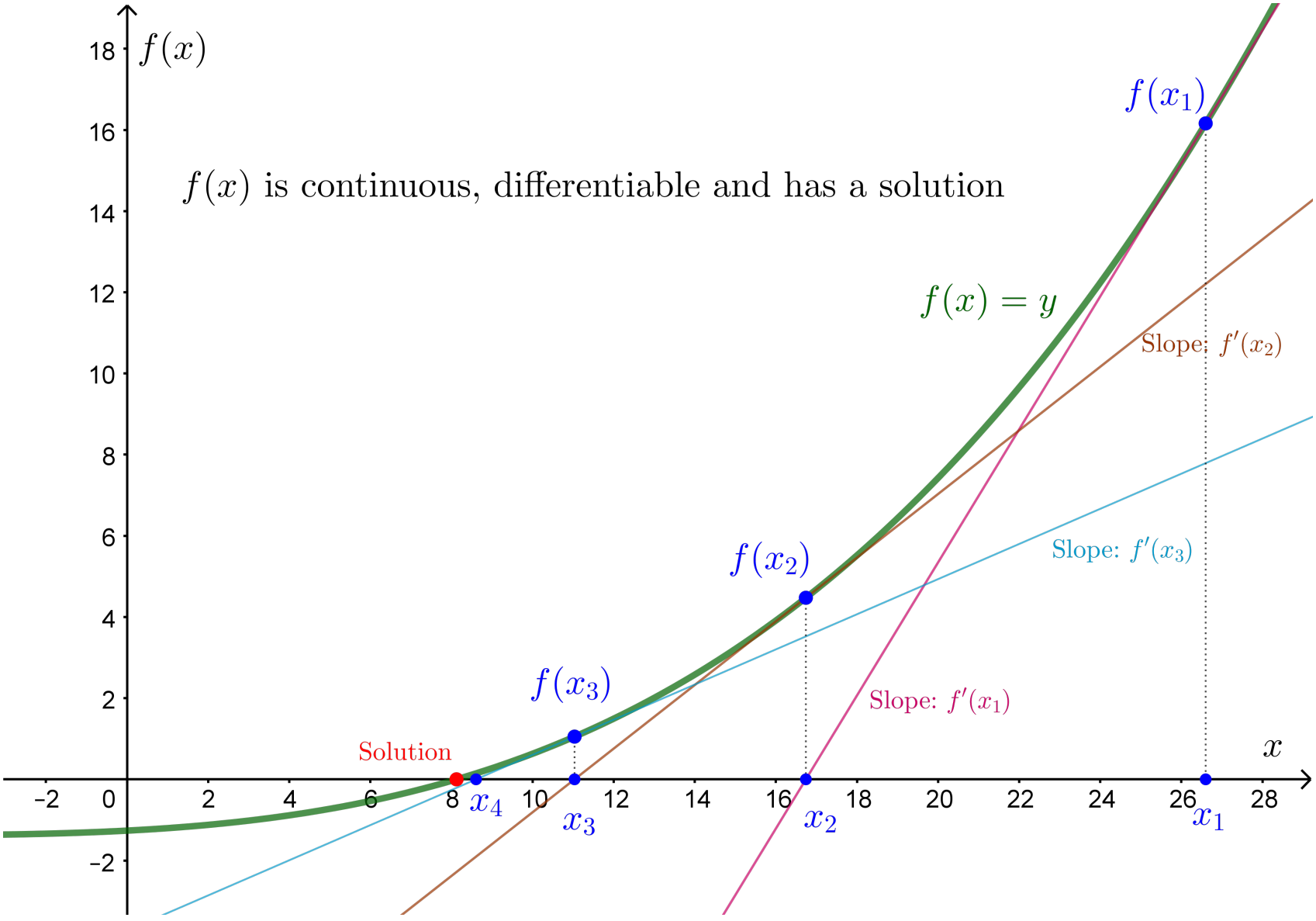
Newton-Raphson Method



Newton-Raphson Method



Newton-Raphson Method



Newton-Raphson Method: Algorithm

1. Chose point x_1 as initial guess of the solution
2. For $i = 1, 2, ..,$ until the error is smaller than a specified value, calculate $x_i + 1$ by using:

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

- Two error estimates are usually used to stop the process of the method:
  1. Estimated relative error:

$$\left| \frac{x_{i+1} - x_i}{x_i} \right| \leq \epsilon$$

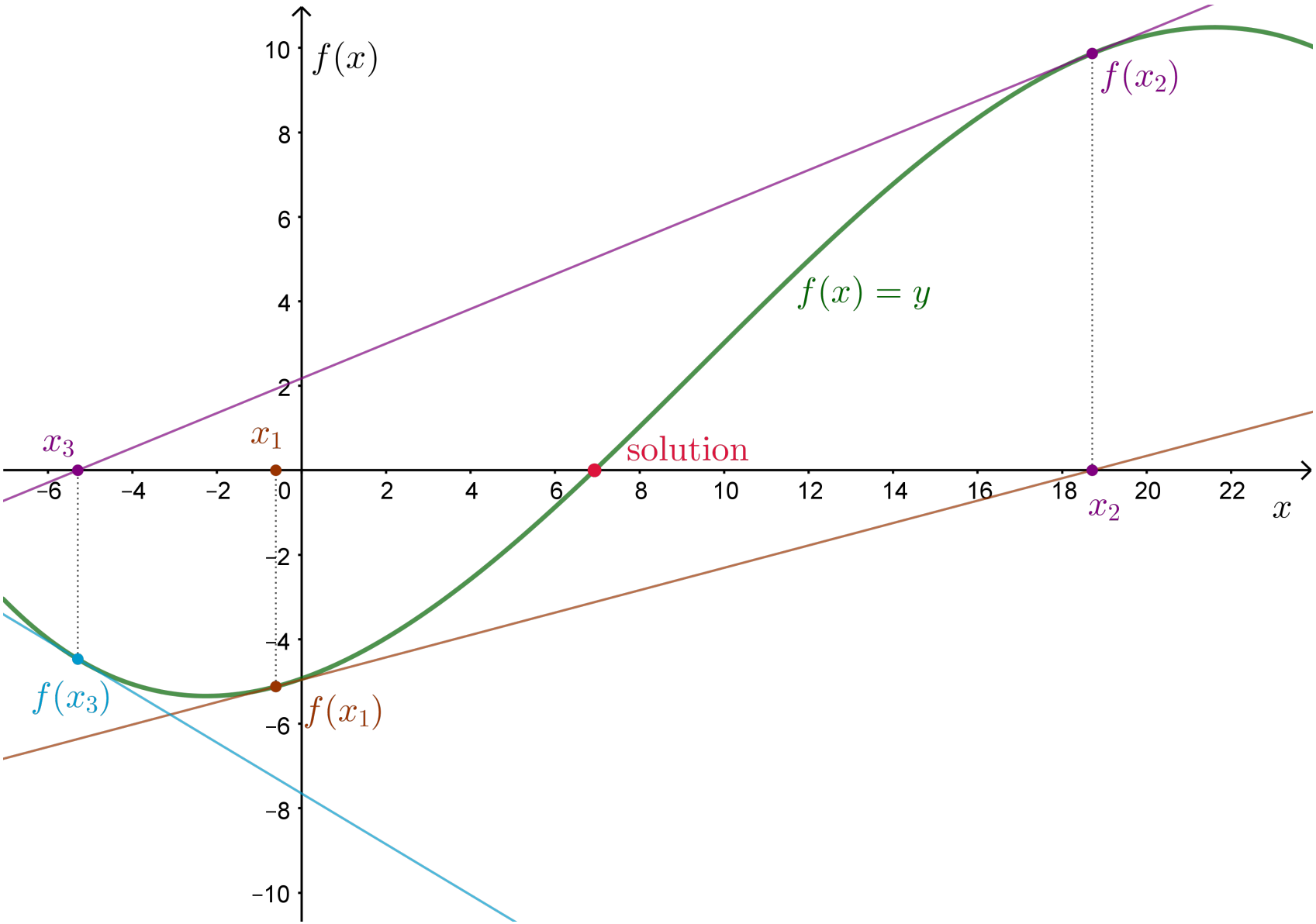
  2. Tolerance in $f(x)$:

$$|f(x_i)| \leq \delta$$

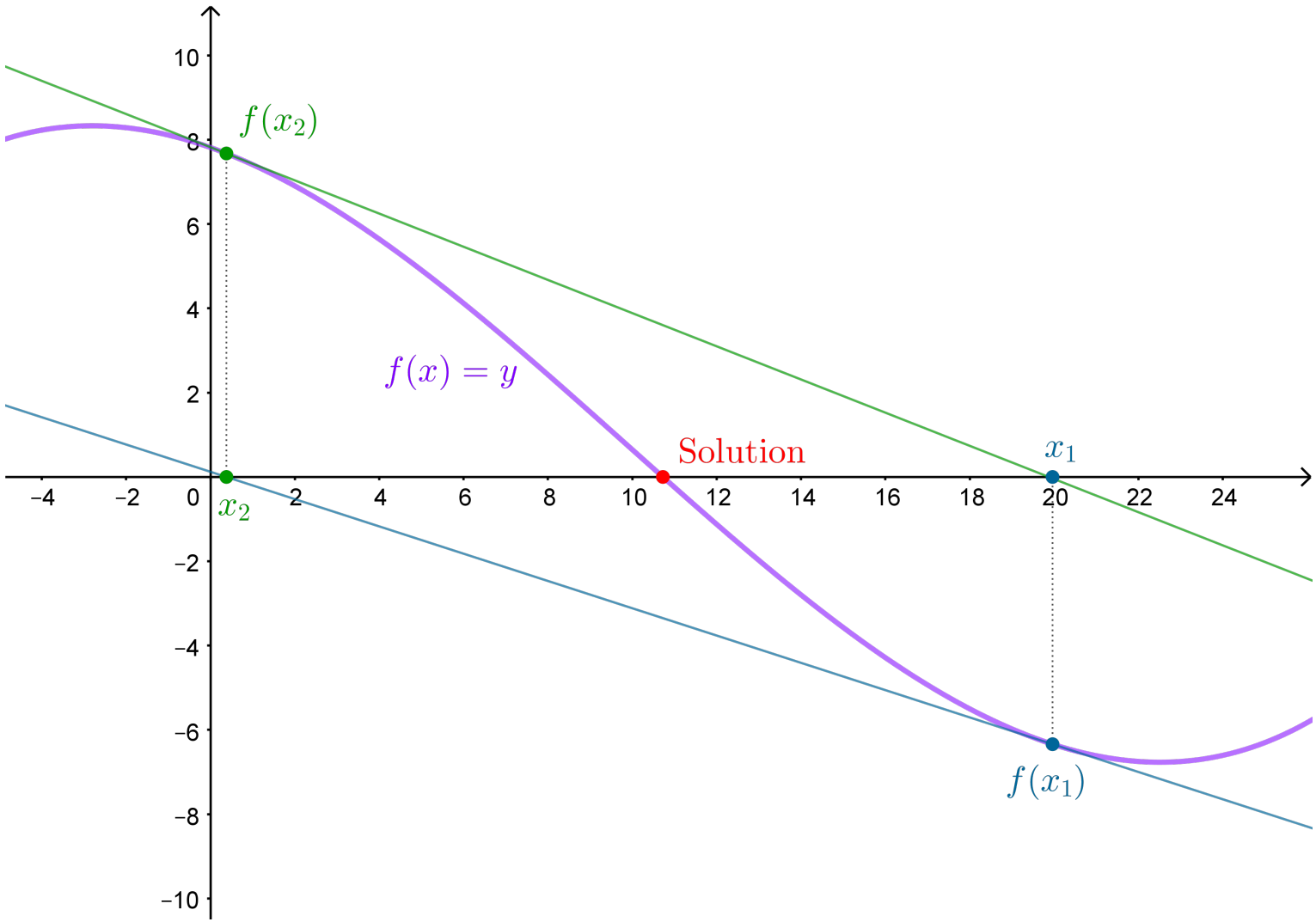
Newton-Raphson Method: Notes

- The method converges fast. A reason for not converging is due to not defining a point close enough to the solution. Convergence problems usually occur due to $f'(x)$ being close to zero where $f(x) = 0$.
- The method converges if the function $f(x)$ and its first and second derivatives are all continuous, if $f'(x)$ is not zero at the solution, assuming that the starting value x_1 is close to the actual solution.
- In many cases the derivative of the function $f(x)$, $f'(x)$ is very difficult to determine. The derivative can be determined numerically, or we can use the secant method.

Newton-Raphson Method: Divergence 1



Newton-Raphson Method: Divergence 2



Newton-Raphson Method: Convergence

Find the solution of the equation $1/x - 2 = 0$ by using Newton's method. For the starting point (initial x estimate of the solution) use:
(a) x= 1.4, (b) x= 1, and (c) x= 0.4
Newton-Raphson expression: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$.
Solution

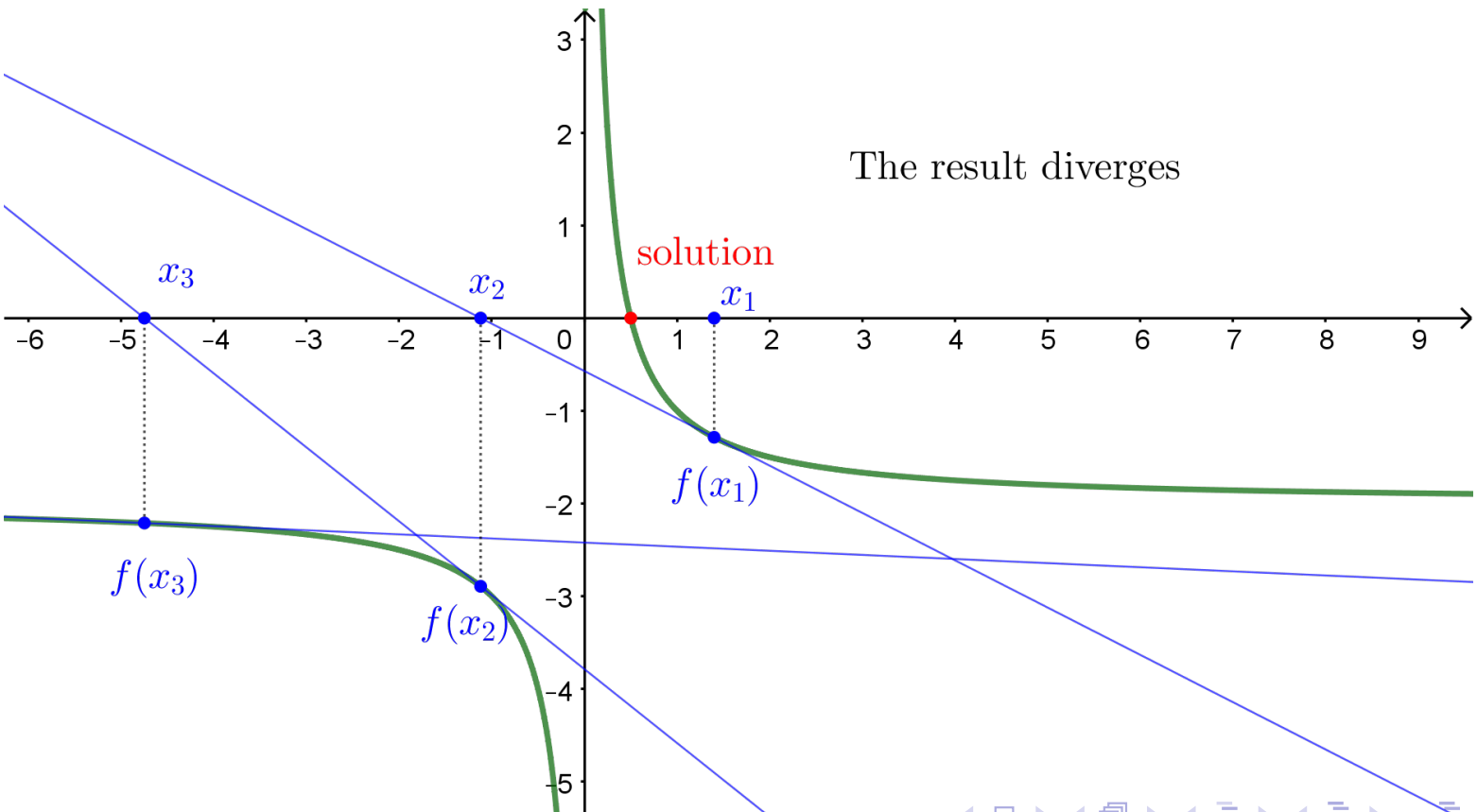
- The analytical solution of the equation is: $x = 0.5$
- The first derivative of the function is: $f'(x) = -\frac{1}{x^2}$.
- From $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$ we get

$$x_{i+1} = x_i - \frac{\frac{1}{x} - 2}{-\frac{1}{x^2}} = 2(x_i - x_i^2)$$

Newton-Raphson Method: Convergence

- (a) For $x_1 = 1.4$:

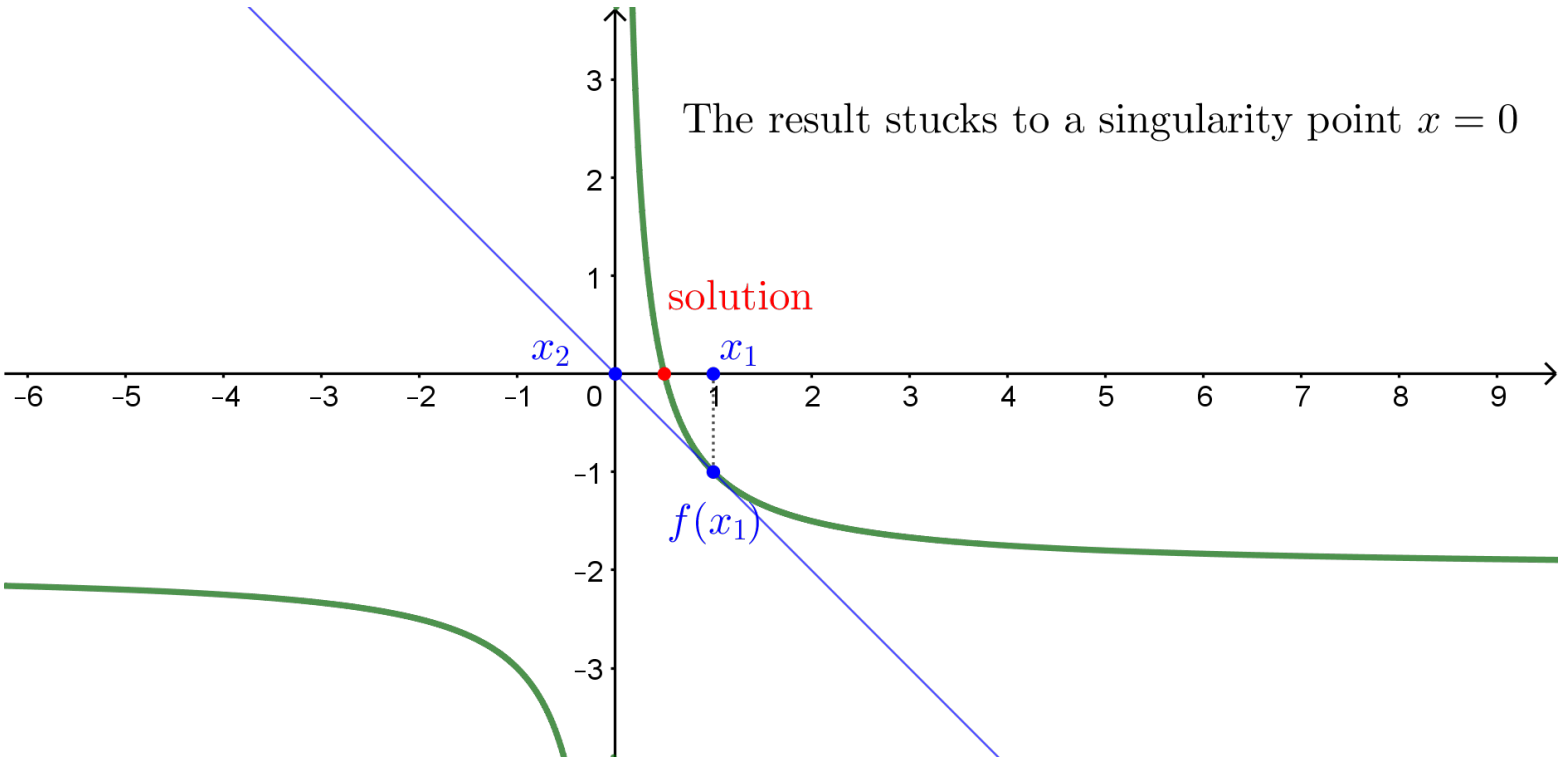
$$x_2 = 2(x_1 - x_1^2) = 2(1.4 - 1.4^2) = -1.12$$
$$x_3 = 2(x_2 - x_2^2) = 2(-1.12 - (-1.12)^2) = -4.4788$$



Newton-Raphson Method: Convergence

- (b) For $x_1 = 1$:

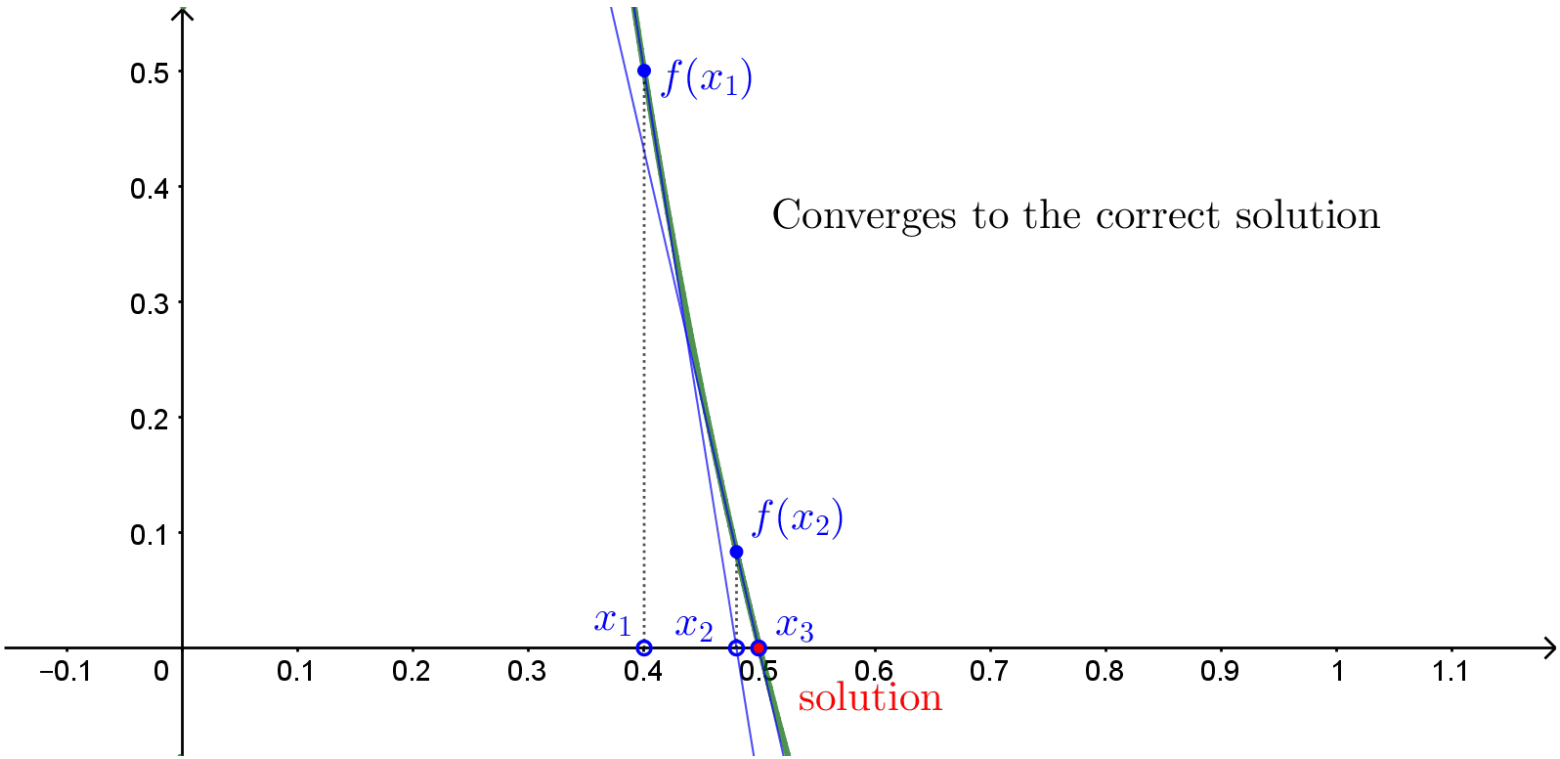
$$x_2 = 2(x_1 - x_1^2) = 2(1 - 1^2) = 0$$
$$x_3 = 2(x_2 - x_2^2) = 2(0 - 0^2) = 0$$



Newton-Raphson Method: Convergence

- (c) For $x_1 = 0.4$:

$$x_2 = 2(x_1 - x_1^2) = 2(0.4 - 0.4^2) = 0.48$$
$$x_3 = 2(x_2 - x_2^2) = 2(0.48 - 0.48^2) = 0.4992$$



Newton-Raphson Method: code

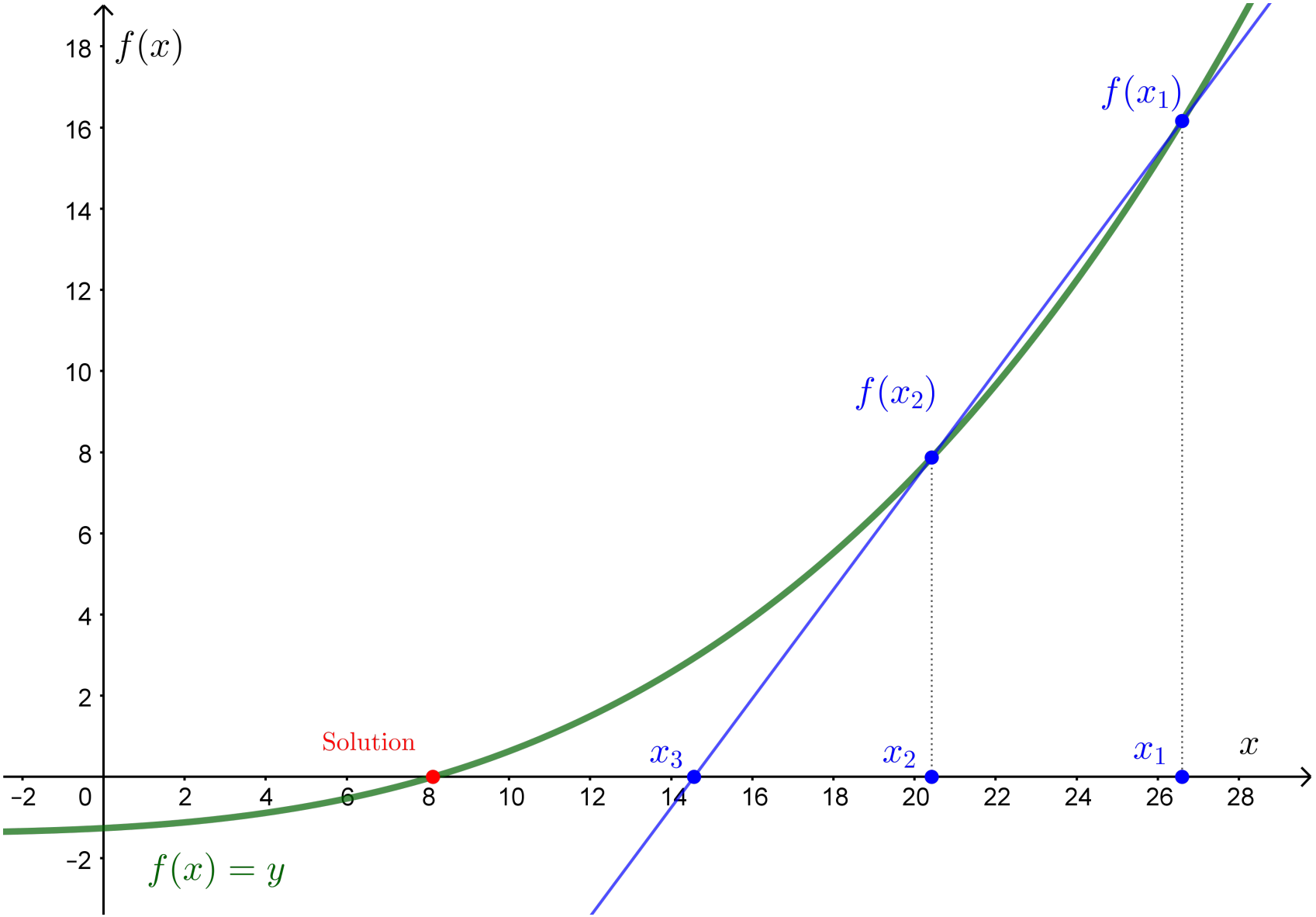
Write a user-defined function named:
`NewtonRaphsonRoot(func,funcDer,Xest,Err,imax)`
that has as an input the `func` (the function) and `funcDer` (the first derivative of the function) as function handles `Xest` as an initial estimate, the `Err` as the error and `imax` as the maximum number of iterations. The function should use the Newton-Raphson iteration method to define roots of nonlinear functions and should return the root `Xs`. If the `Err` and `imax` are not specified as inputs, the function should preselect `Err=1e-5` and `imax=100` (use function `nargin` for that). If the function has reached the `imax` print a message that there is no solution and report the number of iterations.
Use the function to derive the solution of the previous function: $1/x - 2 = 0$, start with $x_0 = 0.4$.
Newton-Raphson expression: $x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$.
The calcuation of the error: $\left|\frac{x_{i+1}-x_i}{x_i}\right|$.

Newton-Raphson Method: code

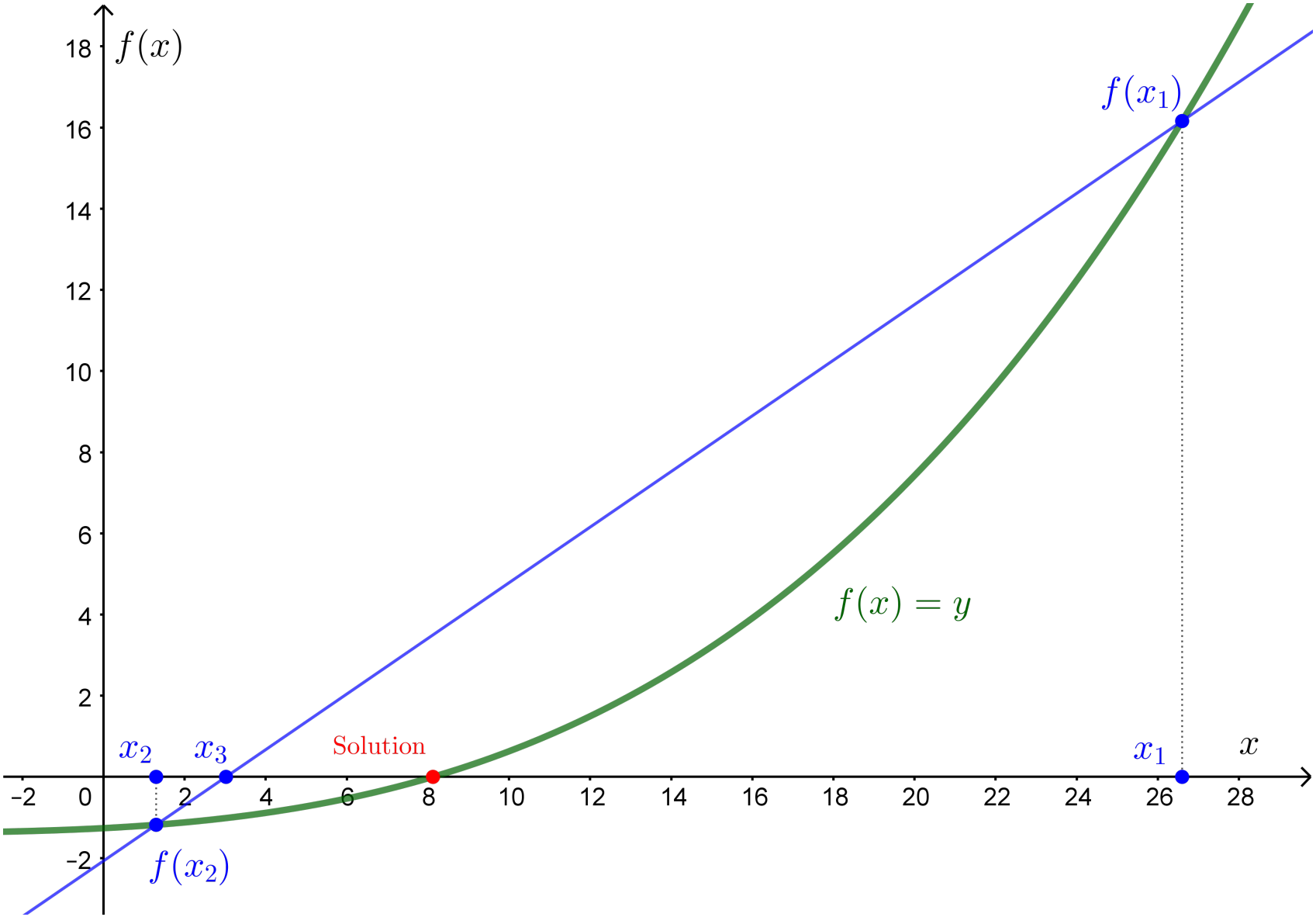
```
function Xs=NewtonRaphsonRoot(func,funcDer,Xest,Err,imax)
% Finds the root func=0 near the point Xest using Newton-Raphson method
% Input variables:
% func: handle function of the function of f(x)
% funcDer: handle function of the derivative function of f(x), namely f'(x)
% Xest: initial estimate of solution
% Err: maximum error
% imax: maximum number of iterations
% Output variable:
% Xs: root
if nargin<5; imax=100; end
if nargin<4; Err=1e-5; end
for i=1:imax
    Xi = Xest-func(Xest)/funcDer(Xest);
    if abs((Xi-Xest)/Xest)<Err
        Xs=Xi;
        break
    end
    Xest=Xi;
end
if i==imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
    Xs=('No answer');
end
```

Secant Method

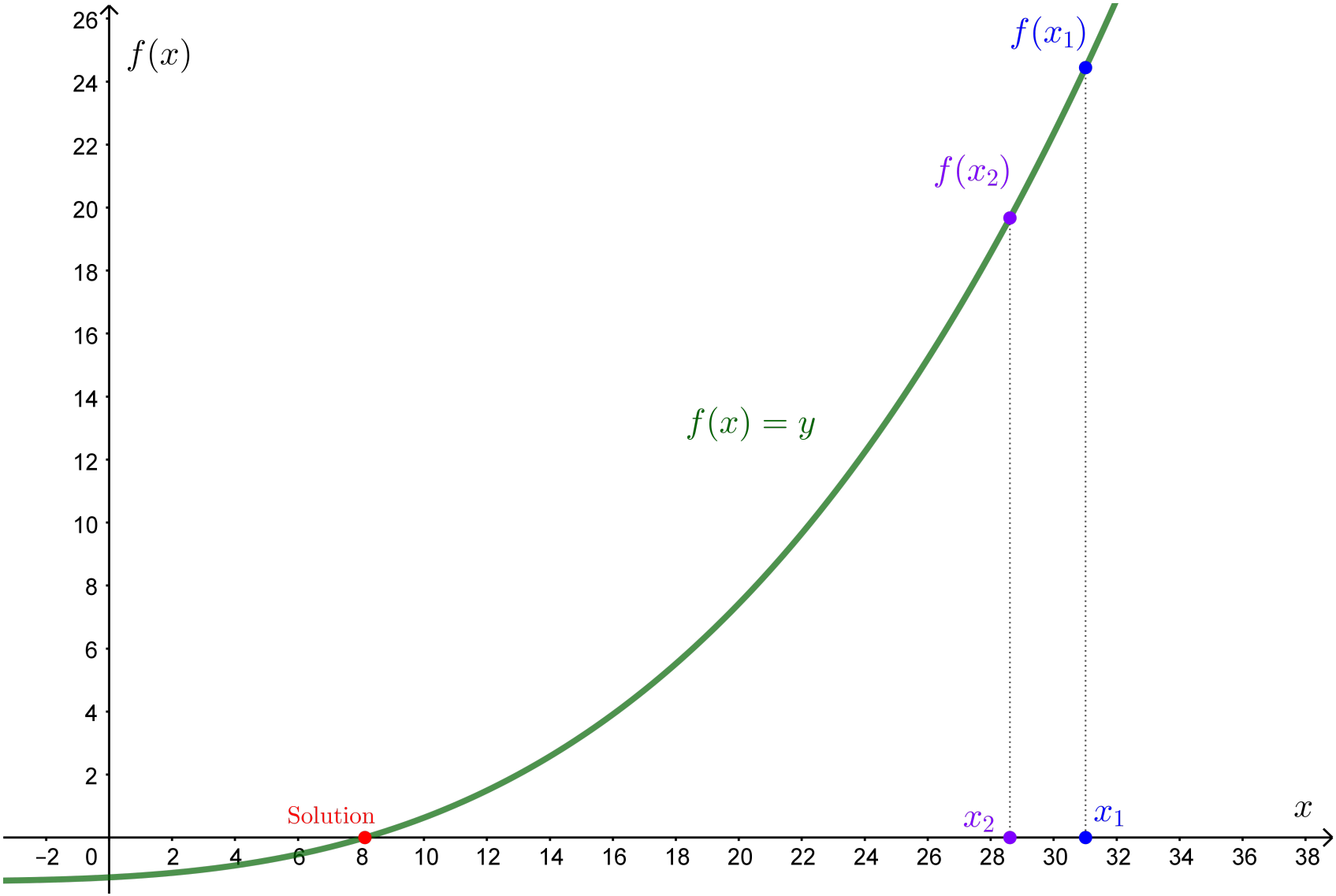
Secant Method: case 1



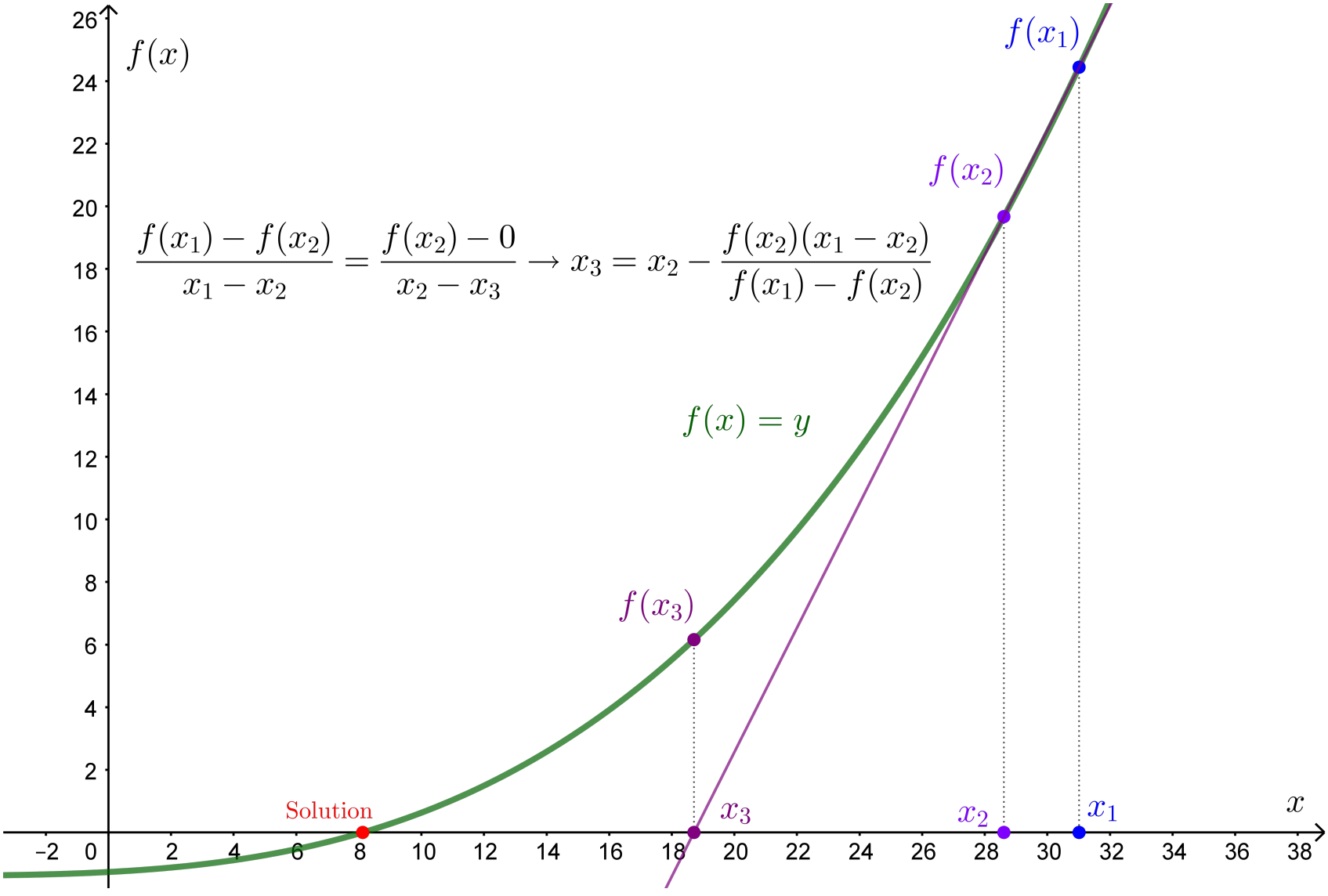
Secant Method: case 2



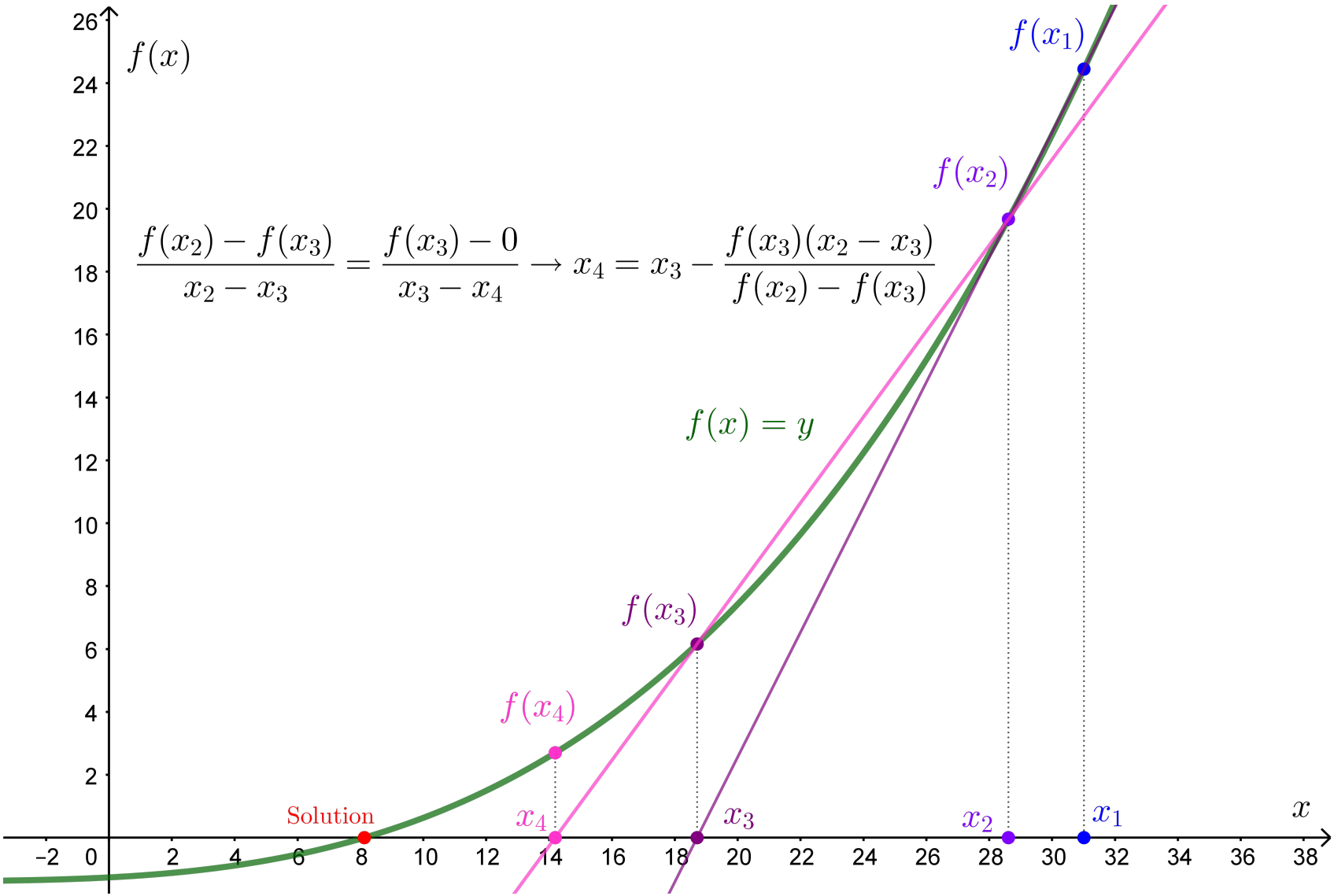
Secant Method: step 1



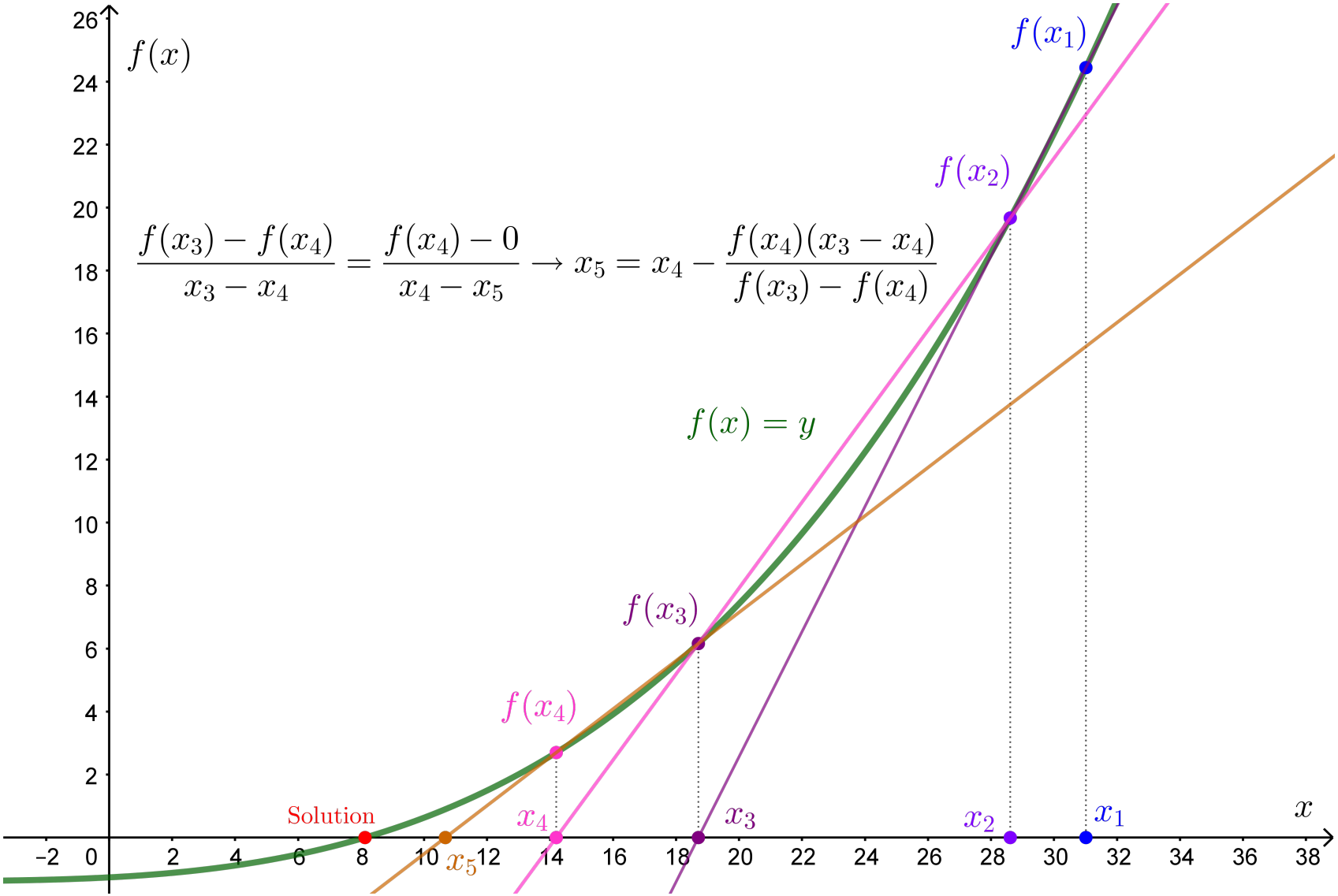
Secant Method: step 2



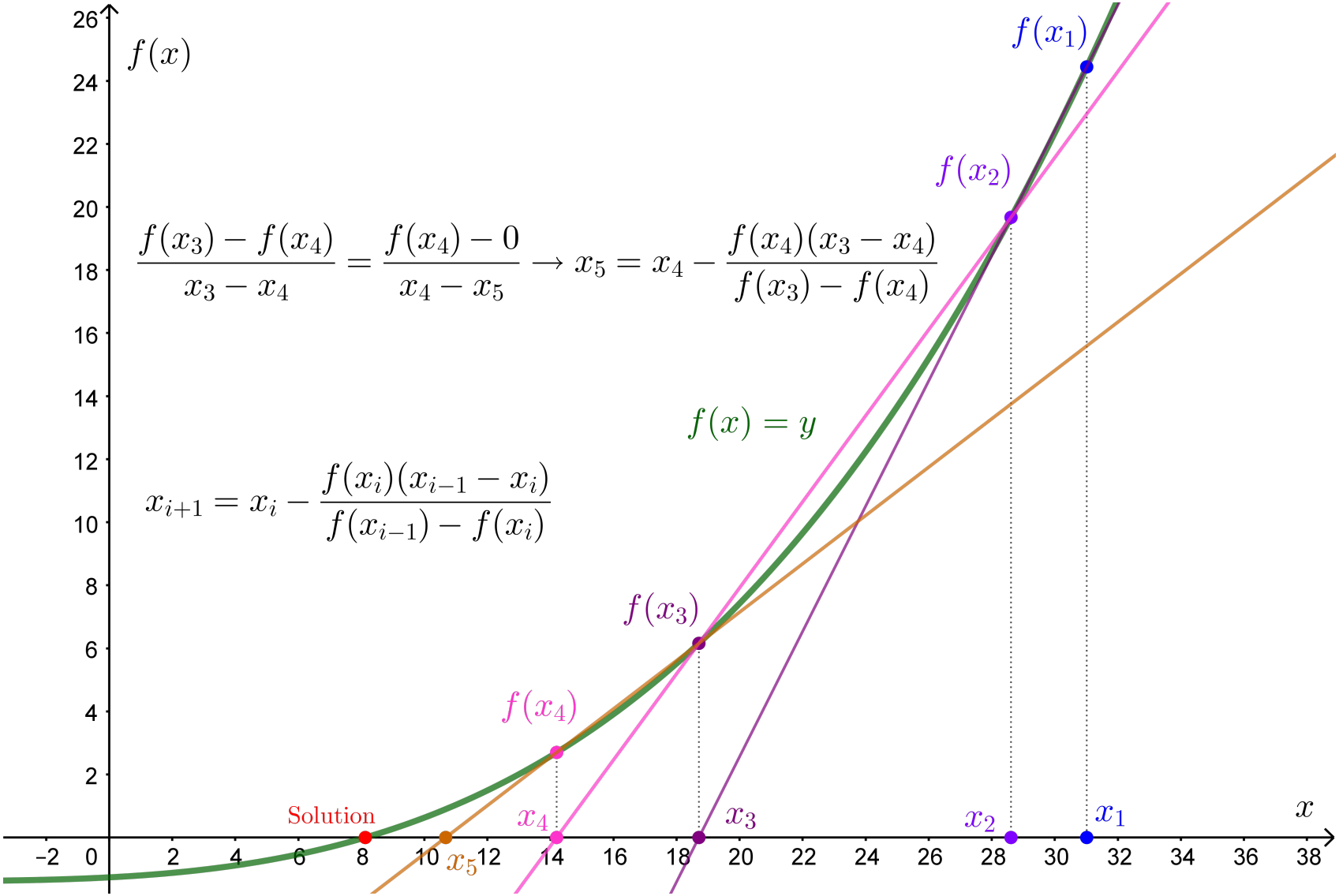
Secant Method: step 3



Secant Method: step 4



Secant Method: step 5



Secant Method - Newton Method

- When the two points x_1, x_2 are close to each other the method is an approximation of Newton's method. Rewriting the expression from the Secant method:

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

and comparing with the expression from Newton's method:

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

we see that the equations are almost identical.

- The difference is the denominator of the second term in the right hand side. In Newton's method the actual derivative needs to be calculated, while in the Secant method an approximation of the derivative is derived.

Secant Method - Regula Falsi Method

- The Secant method is similar to the Regula Falsi method

MatLab functions

Matlab build-in functions: `fzero`

- `fzero` function can be used for finding the roots of any equation.
- `fzero` is based on a modification of the method developed by Brent [1], which combines the bisection method, the secant method and inverse quadratic interpolation.
- Syntax: `x=fzero(function,x0)`, where `x` is the solution, scalar, `function` can have different ways:
  1. mathematical expression as string
  2. user-defined function
  3. anonymous function

  `x0` can be a scalar or two-element vector, if scalar a values close to the solution `x`, if vector the points need to be opposite signs of the solution ($f(x0(1))f(x0(2)) < 0$)
- For more than one solution, each solution can be derived separately
- The function needs to be written in standard form: if the function is to be solved: $xe^{-x} = 0.2$, we need to write it as: $f(x) = xe^{-x} - 0.2 = 0$

Matlab build-in functions: `fzero`

Example

Matlab build-in functions: `roots`

- `roots` function can be used for finding the roots of a polynomial.
- `roots` uses the companion matrix to determine the eigenvalues, which will be the roots of the polynomial:

$$\mathbf{C} = \begin{bmatrix} -p_1/p_0 & -p_2/p_0 & \dots & -p_{n-1}/p_0 & -p_n/p_0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix}$$

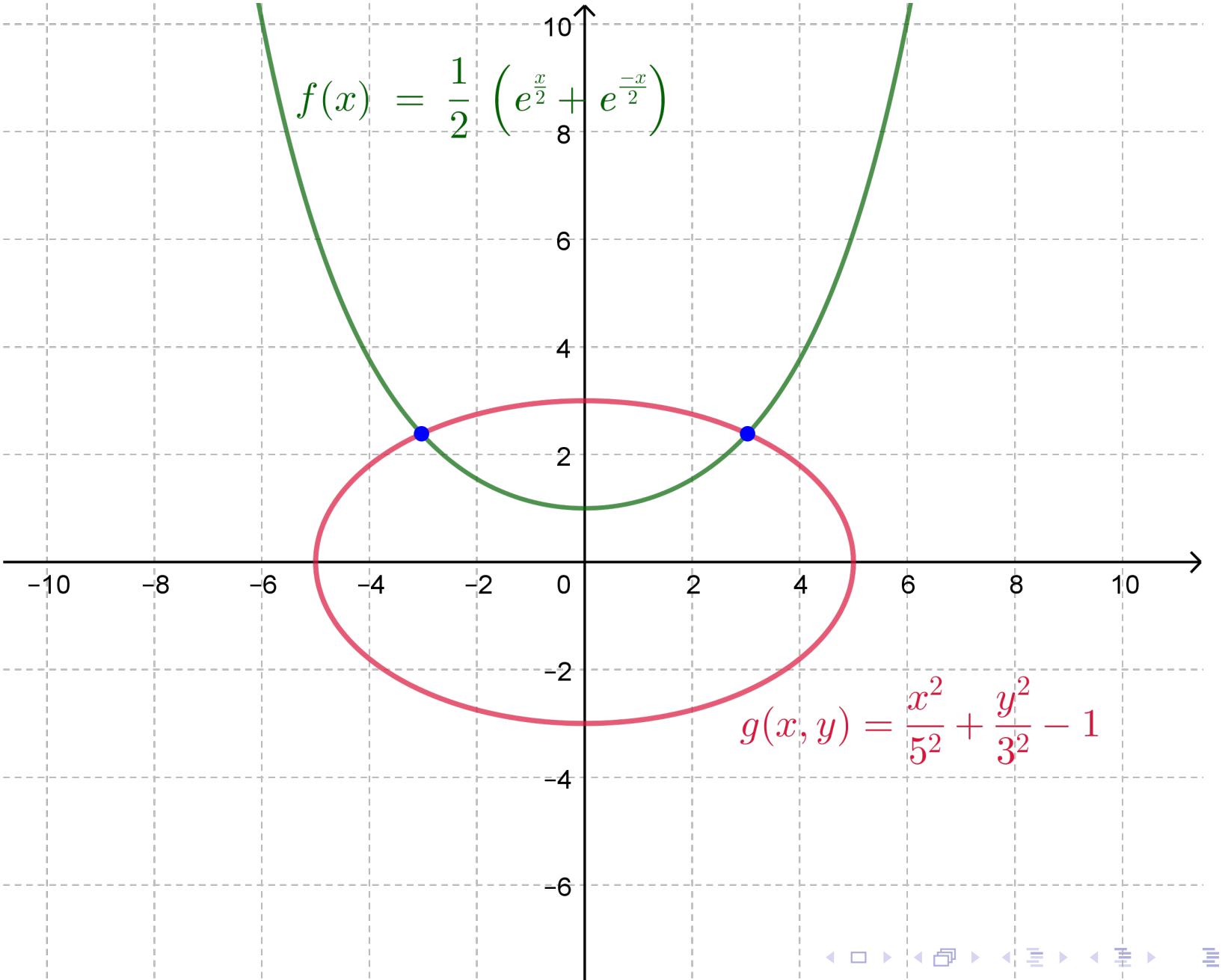
  where $p_0 x^n + p_1 x^{n-1} + .. + p_{n-1} x + p_n = 0$
- Syntax: `r=roots(p)`, where `r` is a column vector with the roots of the polynomial, `p` is a row vector with the coefficients of the polynomial

Matlab build-in functions: `roots`

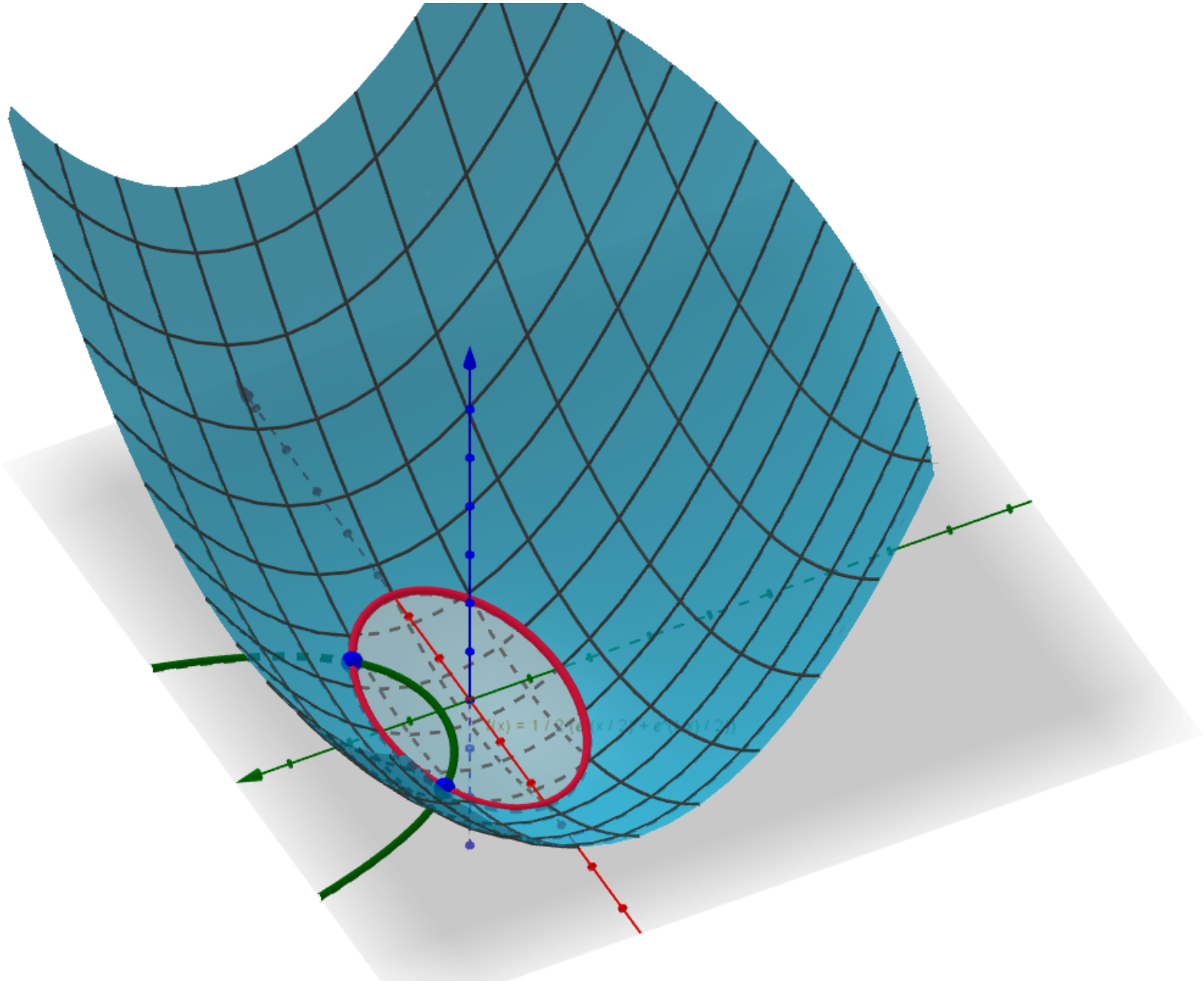
Example

System of Nonlinear Equations

System of nonlinear equations



System of nonlinear equations



System of nonlinear equations - Newton Method

- We have two equations with two unknowns, x, y:

$$f(x_1, y_1) = 0$$
$$f(x_2, y_2) = 0$$

- If x_1, y_1 are the estimated values and if x_2, y_2 is the true (unknown) solutions, we can derive the solutions of f_1 and f_2 at x_2, y_2 by using Taylor's expansion about (x_1, y_1) neglecting the higher order terms:

$$f_1(x_2, y_2) = f_1(x_1, y_1) + (x_2 - x_1)\frac{\partial f_1}{\partial x}|_{x_1,y_1} + (y_2 - y_1)\frac{\partial f_1}{\partial y}|_{x_1,y_1}$$
$$f_2(x_2, y_2) = f_2(x_1, y_1) + (x_2 - x_1)\frac{\partial f_2}{\partial x}|_{x_1,y_1} + (y_2 - y_1)\frac{\partial f_2}{\partial y}|_{x_1,y_1}$$

- Since $f_1(x_2, y_2) = f_2(x_2, y_2) = 0$:

$$\Delta x\frac{\partial f_1}{\partial x}|_{x_1,y_1} + \Delta y\frac{\partial f_1}{\partial y}|_{x_1,y_1} = -f_1(x_1, y_1)$$
$$\Delta x\frac{\partial f_2}{\partial x}|_{x_1,y_1} + \Delta y\frac{\partial f_2}{\partial y}|_{x_1,y_1} = -f_2(x_1, y_1)$$

where $\Delta x = (x_2 - x_1)$ and $\Delta y = y_2 - y_1$

System of nonlinear equations - Newton Method

- By using Cramer's rule we can solve:

$$\Delta x = \frac{-f_1(x_1, y_1)\frac{\partial f_2}{\partial y}|_{x_1,y_1} + f_2(x_1, y_1)\frac{\partial f_1}{\partial y}|_{x_1,y_1}}{J(f_1(x_1, y_1), f_2(x_1, y_1))}$$

$$\Delta y = \frac{-f_2(x_1, y_1)\frac{\partial f_1}{\partial x}|_{x_1,y_1} + f_1(x_1, y_1)\frac{\partial f_2}{\partial x}|_{x_1,y_1}}{J(f_1(x_1, y_1), f_2(x_1, y_1))}$$

where:

$$J(f_1, f_2) = \det \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix}$$

- After solving Δx and Δy:

$$x_2 = x_1 + \Delta x$$
$$y_2 = y_1 + \Delta y$$

- The process continues, by using as inputs the (x_2, y_2) until the successive answers differ by smaller amount, compared to a specific value.

System of nonlinear equations - Newton - General case

- In the general case of any number of variables and equations, we can write the system of equations as:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

where $\mathbf{f}$ and $\mathbf{x}$ are column vectors of n components. For the rth iteration (using only the first two terms):

$$\mathbf{x}^{r+1} = \mathbf{x}^r + \Delta\mathbf{x}^r; \quad r = 0, 1, 2, 3...$$

- Using the Taylor series expansion in n-dimension:

$$\mathbf{f}(\mathbf{x}^r + \Delta\mathbf{x}^r) = \mathbf{f}(\mathbf{x}^r) + \mathbf{J}_r\Delta\mathbf{x}^r$$

where $\mathbf{J}_r = [\partial f_i(\mathbf{x}^r)/\partial x_j]$ for $i = 1, 2, .., n$ and $j = 1, 2, .., n$ is the Jacobian matrix (which may be singular and the inverse cannot be calculated).

- Finally, solving for the improved approximation by assuming that $\mathbf{f}(\mathbf{x}^r + \Delta\mathbf{x}^r) = \mathbf{f}(\mathbf{x}^r) + \mathbf{J}_r\Delta\mathbf{x}^r \approx 0$:

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \mathbf{J}_r^{-1}\mathbf{f}(\mathbf{x}^r); \quad r = 0, 1, 2, 3, ...$$

- Limitations of method: (1) needs good initial approximation and (2) provide derivatives with respect each variable.

System of nonlinear equations - Newton - General case - Example

We have two equations with two variables:

$$x^2 + y^2 = 4$$
$$xy = 1$$

We need to find the solution of the system by using initial approximation as $x = 3$ and $y = -1.5$. Stop the process when the Euclidean 2-norm $\sqrt{f_1(x_r, y_r)^2 + f_2(x_r, y_r)^2} \leq 5 \cdot 10^{-4}$ (`norm` in MatLab).

- Solution
- First we can write the equations as:

$$f_1 = x^2 + y^2 - 4 = 0$$
$$f_2 = xy - 1 = 0$$

- The Jacobian and its inverse can be calculated as:

$$\mathbf{J} = \begin{bmatrix} \partial f_1/\partial x & \partial f_1/\partial y \\ \partial f_2/\partial x & \partial f_2/\partial y \end{bmatrix}; \quad \mathbf{J}^{-1} = \frac{1}{2xx - 2yy} \begin{bmatrix} x & -2y \\ -y & 2x \end{bmatrix}$$

System of nonlinear equations - Newton - General case - Example

Solution

- For $r = 0$:

$$\begin{bmatrix} x^{(1)} \\ y^{(1)} \end{bmatrix} = \begin{bmatrix} 3 \\ -1.5 \end{bmatrix} - \begin{bmatrix} 0.2222 & 0.2222 \\ 0.1111 & 0.4444 \end{bmatrix} \begin{bmatrix} 7.25 \\ -5.5 \end{bmatrix} = \begin{bmatrix} 2.6112 \\ 0.1389 \end{bmatrix}$$

The norm is equal to 2.9079.

- For $r = 1$:

$$\begin{bmatrix} x^{(2)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 2.6112 \\ 0.1389 \end{bmatrix} - \begin{bmatrix} 0.1920 & -0.0204 \\ -0.0102 & 0.3841 \end{bmatrix} \begin{bmatrix} 2.8372 \\ -0.6373 \end{bmatrix}$$

$$\begin{bmatrix} x^{(2)} \\ y^{(2)} \end{bmatrix} = \begin{bmatrix} 2.0533 \\ 0.4127 \end{bmatrix}$$

The norm is equal to 0.4153.

System of nonlinear equations - Newton - General case - Example

Solution

- For $r = 2$:

$$\begin{bmatrix} x^{(3)} \\ y^{(3)} \end{bmatrix} = \begin{bmatrix} 2.0533 \\ 0.4127 \end{bmatrix} - \begin{bmatrix} 0.2538 & -0.1020 \\ -0.0510 & 0.5075 \end{bmatrix} \begin{bmatrix} 0.3861 \\ -0.1527 \end{bmatrix}$$

$$\begin{bmatrix} x^{(3)} \\ y^{(3)} \end{bmatrix} = \begin{bmatrix} 1.9397 \\ 0.5099 \end{bmatrix}$$ The norm is equal to 0.0249.

- For $r = 3$:

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \end{bmatrix} = \begin{bmatrix} 1.9397 \\ 0.5099 \end{bmatrix} - \begin{bmatrix} 0.2769 & -0.1456 \\ -0.0728 & 0.5538 \end{bmatrix} \begin{bmatrix} 0.0223 \\ -0.0110 \end{bmatrix}$$

$$\begin{bmatrix} x^{(4)} \\ y^{(4)} \end{bmatrix} = \begin{bmatrix} 1.9319 \\ 0.5176 \end{bmatrix}$$

The norm is equal to $0.000134 \leq 0.0005$.

References

R. P. Brent.

An algorithm with guaranteed convergence for finding a zero of a function.
The Computer Journal, 14(4):422–425, 01 1971.

Amos Gilat and Vish Subramaniam.

Numerical Methods for Engineers and Scientists, An Introduction with Applications Using MATLAB.
Wiley, Danvers, Massachusetts, 2014.